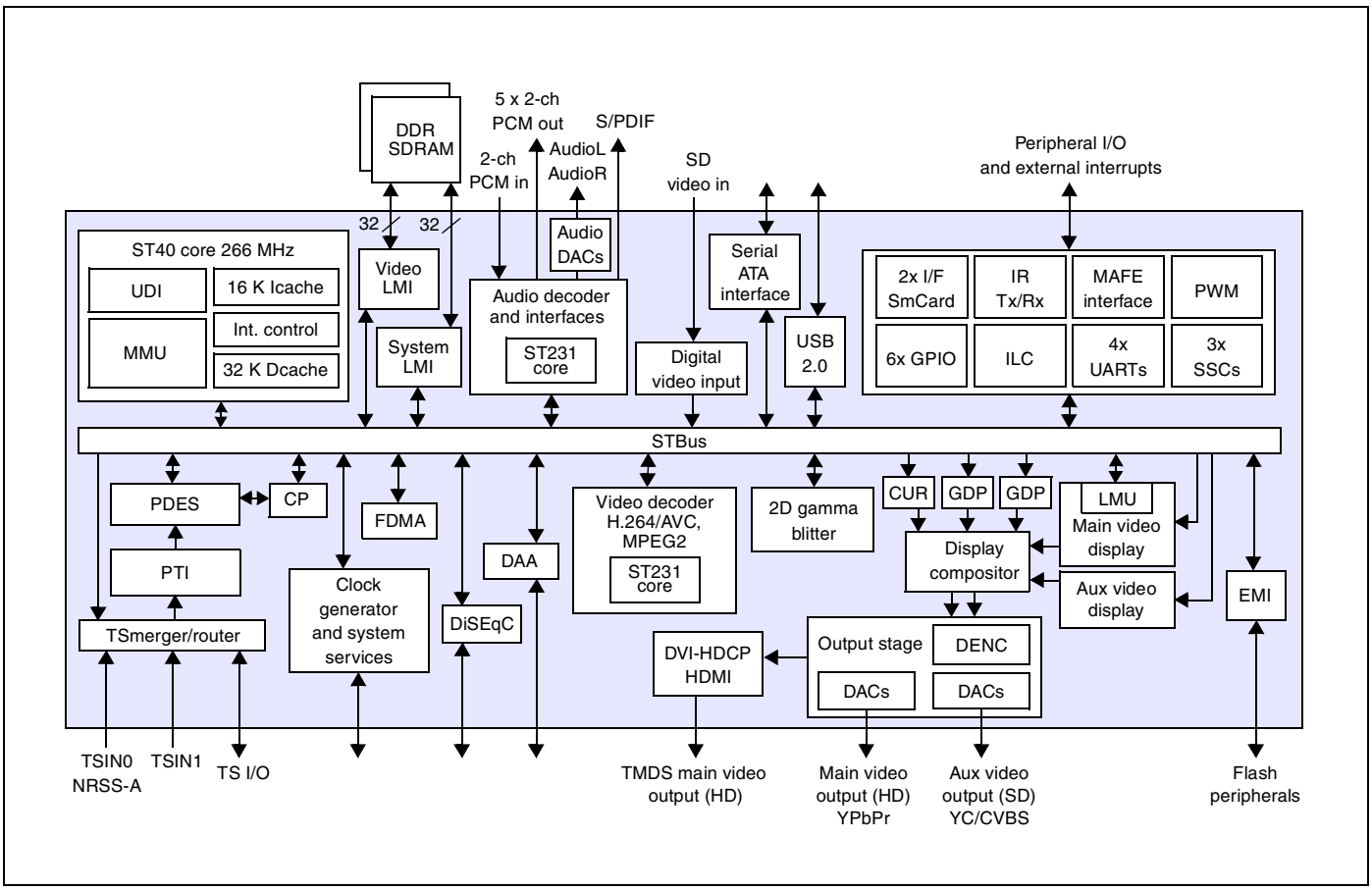
STx7100 (STb7100, STi7100, STm7100)

Low-cost HDTV set-top box decoder for H.264/AVC and MPEG-2

ADVANCE DATA

Features

- **The STx7100 is a single-chip, high-definition STB decoder including:**
  - ST40 CPU core, 266 MHz
  - dual ST231 CPU cores for audio and video decoding, both 400 MHz
  - transport filtering and descrambling
  - video decoder: H.264/AVC and MPEG-2
  - graphics engine and dual display: high-definition (HD) and standard definition (SD)
  - audio decoder
- **The STx7100 also features the following embedded interfaces:**
  - USB 2.0
  - DVI/HDMI output
  - digital audio and video auxiliary input
  - modem
  - serial ATA



Processor subsystem

ST40 32-bit superscalar RISC CPU
- 266 MHz, 2-way set associative 16-Kbyte ICache, 32-Kbyte DCache, MMU
- 5-stage pipeline, delayed branch support
- floating point unit, matrix operation support
- debug port, interrupt controller

Transport subsystem

TS merger/router
- 2 serial/parallel inputs
- 1 bidirectional interface
- merging of 3 transport streams
- transport stream from memory support
- NRSS-A module interface
- TS routing for DVB-CI and CableCARD™ modules

Programmable transport interface (PTI)
- transport stream demux: DVB, DIRECTV®, ATSC, ARIB, OpenCable, DCII
- integrated DES, AES, DVB, ICAM and Multi2 descramblers
- NDS random access scrambled stream protocol (RASP) compliant
- NDS ICAM CA
- support for VGS, Passage and DVS042 residue handling

Video/graphics subsystem

MPEG-4 AVC high profile level 4.1/ MPEG-2 MP@HL video decoder
- hardware/firmware mixed architecture
- advanced error concealment and trick mode support
- dual MPEG-2 MP@HL decode

SD (ITU-R BT 601/656) D1 digital video input

Displays
- HD display, multiformat capable (1080I, 720P, 480P/576P, 480I/576I)
  - Analog HD output RGB or YPbPr
  - HDMI encoded output
- SD display
  - Analog SD output: YPbPr or YC and CVBS

Gamma 2D/3D graphics processor
- dual source blitter engine
- alpha blending and logical operations
- color space and format conversion
- fast color fill
- arbitrary resizing with high quality filters
- acceleration of direct drawing by CPU

Gamma compositor and video processor
- 5 channels mixer for HD output
- independent 2-channel mixer for SD output
- 2 graphic display planes
- high-quality video scaler
- picture up-conversion hardware
- linear resizing and format conversions
- horizontal and vertical filtering

Copy protection
- HDMI 1.0/HDCP 1.1 copy protection hardware
- Macrovision™ copy protection for 480I, 480P, 576I, 576P outputs

Audio subsystem

Digital audio decoder
- compatible with all popular audio standards
- PCM mixing with internal or external source and sample rate conversion
- independent multichannel PCM output, S/PDIF output and analog output
- 6 to 2 channel downmixing
- PCM audio input

Stereo 24-bit audio DAC for analog output

IEC958/IEC1937 digital audio output interface (S/PDIF)

Interfaces

External memory interface (EMI)
- 16-bit interface supporting ROM, Flash, SFlash, SRAM, peripherals
- access in 5 banks
- master/slave support for interconnecting two STx7100 devices

Dual local memory interface (LMI)
- dual interface for 32-bit DDR1 200-MHz (DDR400) memories, supports 64, 128, 256 and 512 Mbit devices

USB 2.0 host interface

Serial ATA hard-disk drive support
- record and playback with trick modes
- pause and time shifting
- watch and record

On-chip peripherals
- 4 ASCs (UARTs) with Tx and Rx FIFOS, two of which can be used in smartcard interfaces
- 2 smartcard interfaces and clock generators (improved to reduce external circuitry)
- 3 SSCs for I²C/SPI master slaves interfaces
- 2 PWM outputs
- teletext serializer and DMA module
- 6 banks of general purpose I/O, 3.3 V tolerant
- SiLabs line-side (DAA) interface
- Modem analog front end (MAFE/DAA) interface
- infrared transmitter/receiver supporting RC5, RC6 and RECS80 codes
- UHF remote receiver input interface
- interrupt level controller and external interrupts, 3.3 V tolerant
- low power/RTC/watchdog controller
- integrated VCXO
- DiSEqC™ 2.0 interface

Flexible multichannel DMA

Services and package

JTAG/TAP interface

ST40 and ST231 toolset support

35 x 35 PGBA package, 580+100 balls

Contents

Part 1

Overview

1 Applications overview

The STx7100 is a new generation, high-definition set-top box decoder chip, and provides very high performance for low-cost HD systems. With enhanced performances over the STi7710, it includes an H.264/AVC decoder for new low-bitrate applications, as well as dual-decode MPEG-2 capability.

Based on the Omega2 (STBus) architecture, this system on chip is a full back-end processing solution for digital terrestrial, satellite and cable high-definition set-top boxes compliant with ATSC, DVB, DIRECTV, DCII, OpenCable and ARIB BS4 specifications.

The STx7100 demultiplexes, decrypts and decodes a single HD or SD video stream with associated multichannel audio. Video is output to two independently formatted displays: a full resolution display intended for a TV monitor, and a downsampled display intended for a VCR, or alternatively a second SD TV. Connection to a TV or display panel can be via an analog component interface or a copy protected DVI/HDMI interface. Composite outputs are provided for connection to the VCR with Macrovision protection. Audio is output with optional PCM mixing to an S/PDIF interface, PCM interface or via integrated stereo audio DACs.

Digitized analog programs can also be input to the STx7100 for reformatting and display.

The STx7100 includes a graphics rendering and display capability with a 2D graphics accelerator, two graphics planes and a cursor plane. A dual display compositor provides mixing of graphics and video with independent composition for each of the TV and VCR outputs.

The STx7100 includes a stream merger to allow three different transport streams from different sources to be merged and processed concurrently. Applications include DVR time shifted viewing of a terrestrial program while acquiring an EPG/data stream from a satellite or cable front end.

The flexible descrambling engine is compatible with required standards including DVB, DES, AES and Multi2.

The STx7100 embeds a 266 MHz ST40 CPU for applications and device control. A dual DDR1 SDRAM memory interface is used for higher performance, to allow the video decoder the required memory bandwidth for HD H.264 decoding and sufficient bandwidth for the CPU and the rest of the system. A second memory bus is also provided for flash memory storing resident software and for connection of peripherals.

A hard-disk drive (HDD) can be connected either to the serial ATA interface or as an expansion drive via the USB 2.0 port. The USB port can also be used to connect to a DOCSIS 2.0 CM gateway for interactive cable applications.

The STx7100 is supported by STMicroelectronics' STAPI software, and is compatible with several other related devices such as the STi7109.

Figure 1: Low-cost satellite HD set-top box

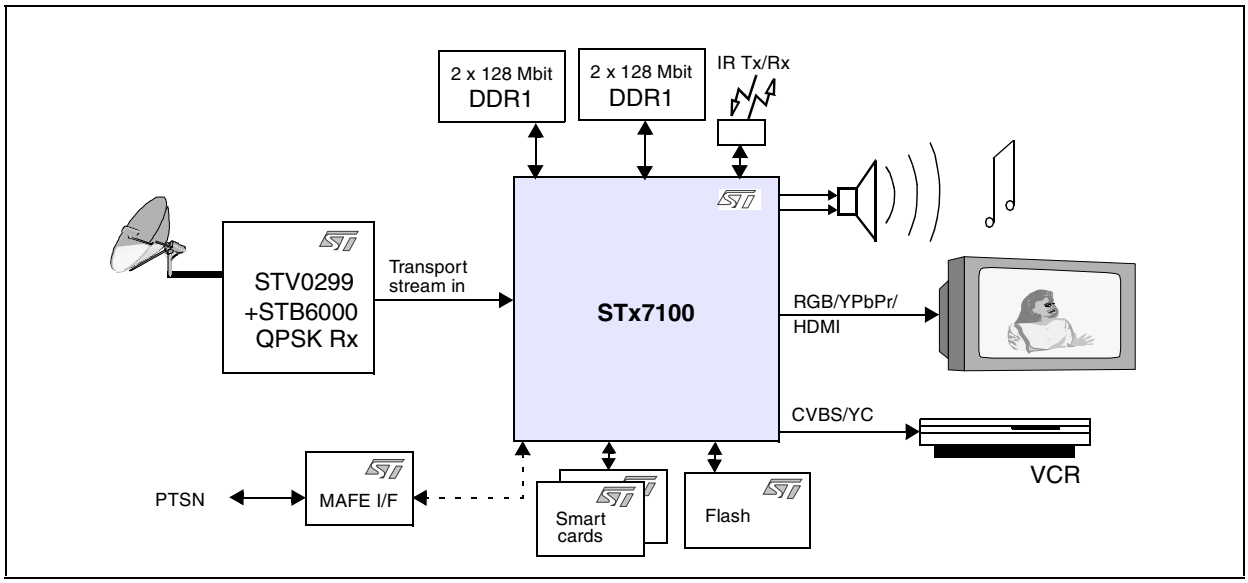


Figure 2: Low-cost cable HD set-top box with return channel

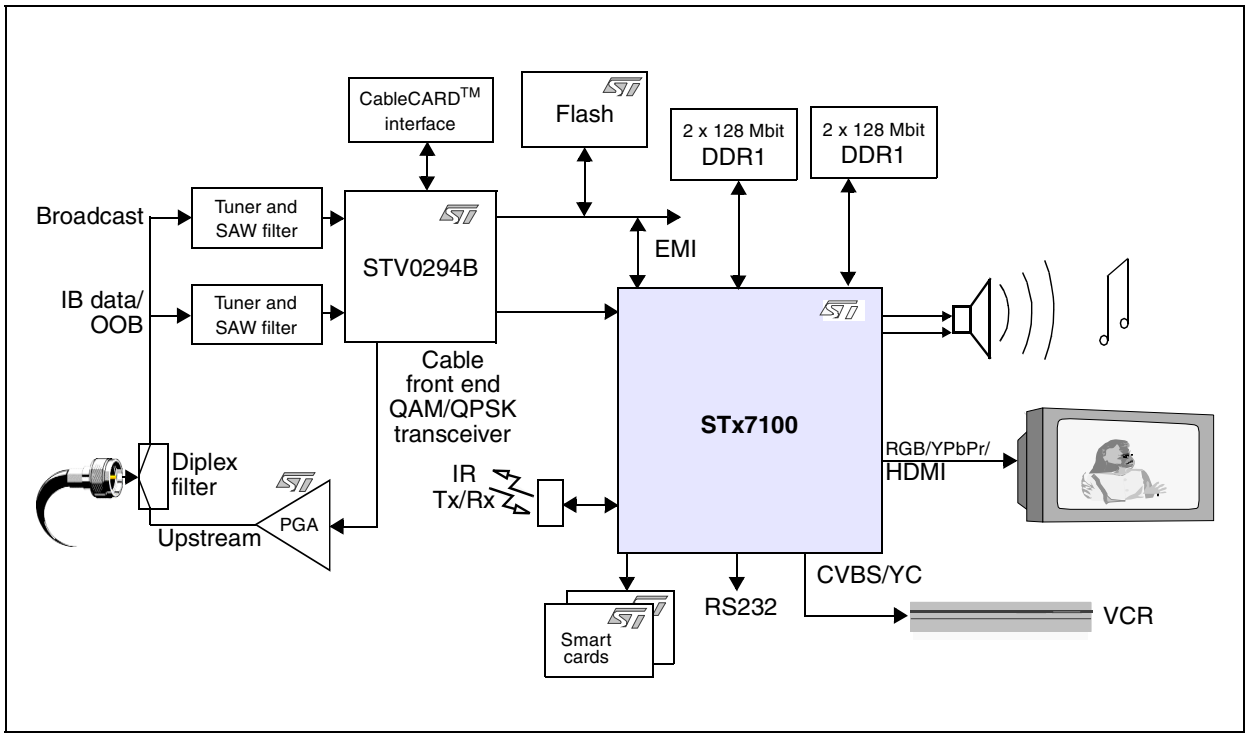


Figure 3: Low-cost dual satellite and terrestrial HD set-top box with HDD

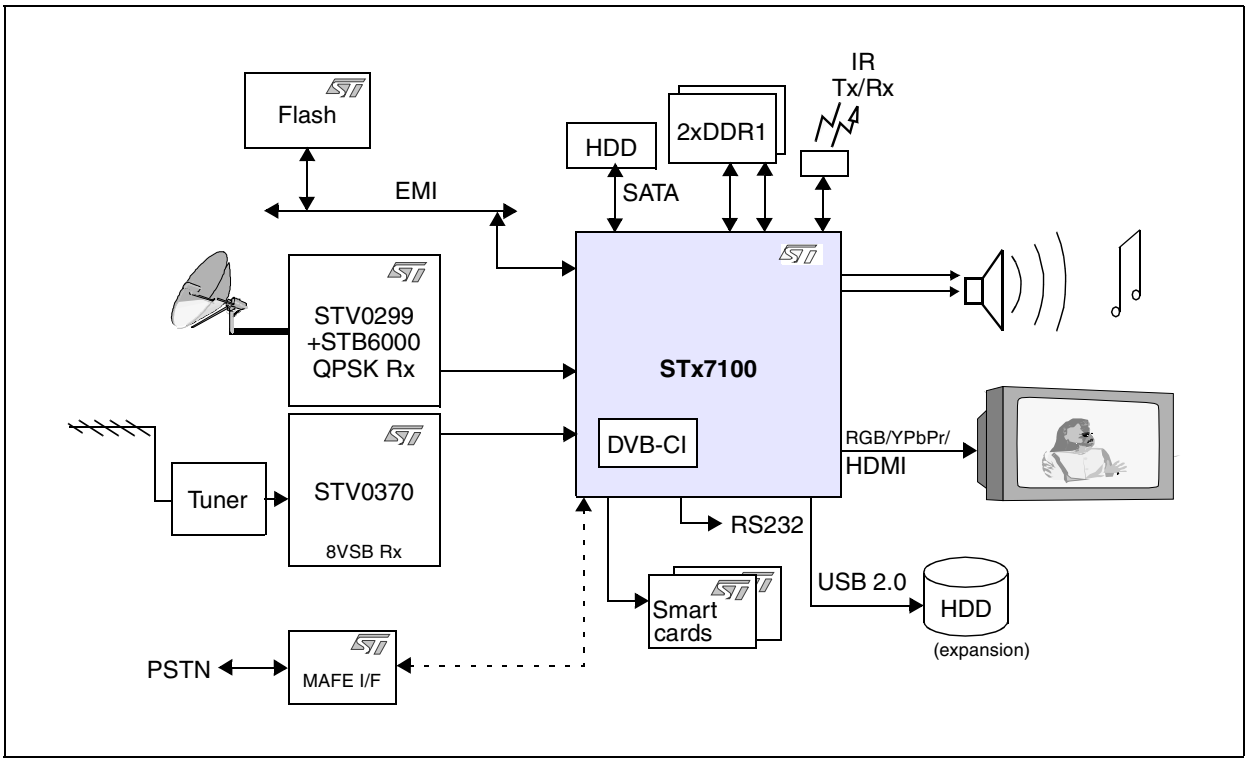
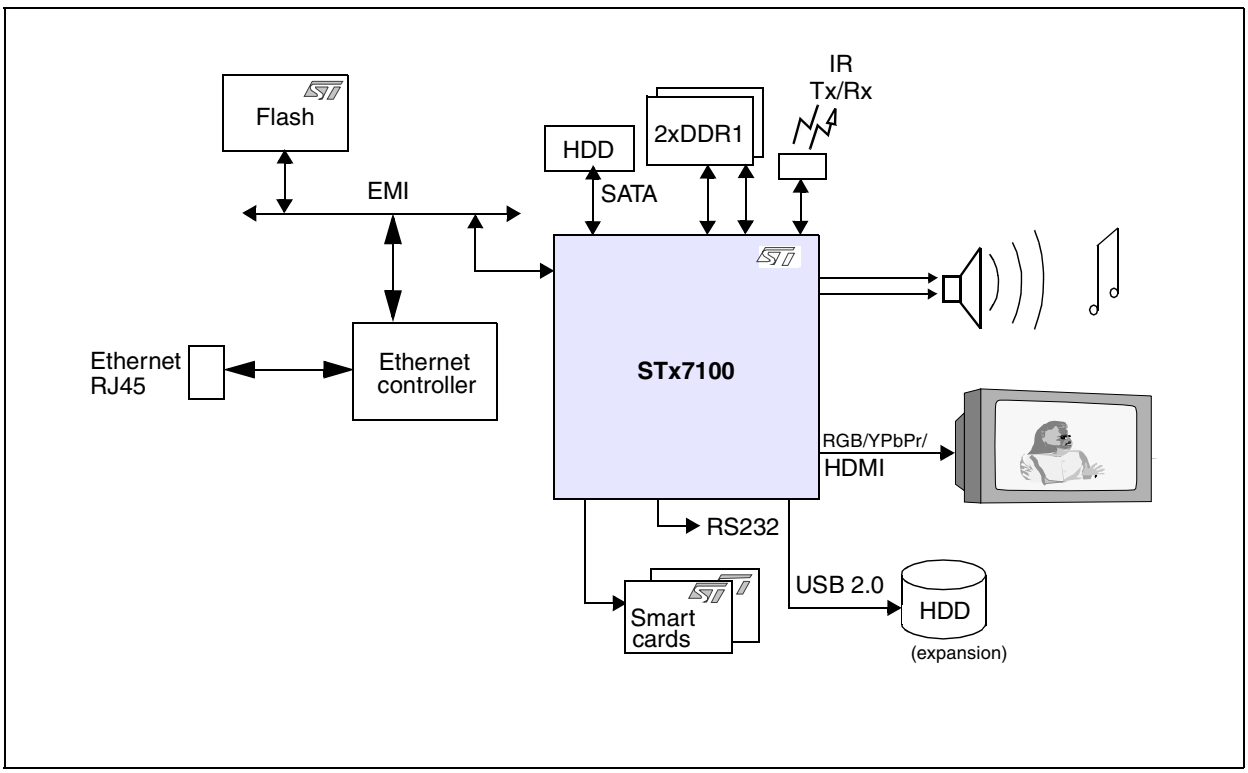


Figure 4: Low-cost HD IP-TV set-top box with HDD



2 Device variants

The STx7100 is available in several variants. These are differentiated by between two and four suffix letters in the product code, as summarised in Table 1.

Table 1: Part code suffix summary

Market type		Chip revision		Leaded or unleaded package		Delivery type	
Z	Development version, security disabled	**A**	Cut 1	**P**	Unleaded (for STi and STm 7100)	**T**	Tape-and-reel
J	Nagra C+	**B**	Cut 2	**L**	Unleaded (for STb 7100)	**-**	Standard
O	Nagra Premier	**...**	and so on	**-**	Leaded		
Q	Nagra Dedicated						
T	Viaccess France Telecom						
I	TPS						

The first letter indicates *market type*, details of which are shown in Table 2.

Table 2: Market type

Market type	Macro-vision	AC3/ Prologic	SATA	V22	DES	ICAM	USB2	Market
Z	✔	✔	✔	✔	✔	✔	✔	Development version, security disabled
J		✔						Nagra C+
O	✔	✔	✔			✔	✔	Nagra Premier
Q	✔	✔				✔	✔	Nagra Dedicated
T	✔	✔				✔	✔	Viaccess France Telecom
I	✔	✔	✔	✔		✔	✔	TPS

The second letter indicates *chip revision*.

The letter **P** or **L** indicates unleaded (Eco) packaging; see Chapter 6: *Package on page 48* for details.

Finally, for parts supplied “tape-and-reel”, the suffix **T** is added.

Examples

STi7100ZBP is a development version, cut 2, unleaded.

STb7100OCT is a Nagra Premier version, cut 3, leaded, delivered in a tape-and-reel pack.

3 Architecture

3.1 Overview

The STx7100 is designed around the well-proven Omega2 (STBus) interconnect and presents a new generation of video decoder for MPEG-4 AVC as well as MPEG-2 decoding.

Transport streams are received and processed by the TS subsystem. The resulting PES streams and section data are stored in memory buffers in DDR SDRAM attached to the local memory interfaces (system and video LMI).

A flexible DMA controller (FDMA) performs PES parsing and start code detection and routes elementary streams to audio and video bit buffers in DDR SDRAM. An MP@HL/H.264 video decoder decodes HD or SD video streams. Audio decoding and PCM mixing is performed by the audio decoder and output via S/PDIF and PCM interfaces. Audio can also be output via an integrated 24-bit stereo DACs system.

After video decoding, two independently formatted video displays (main and auxiliary) can be generated, and each mixed independently with graphics to create main and auxiliary display compositions. The main display composition (HD or SD) can be output as RGB or YPbPr analog component video and digitally via a copy-protected DVI/HDMI interface. The auxiliary display composition (SD only) can be output as Y/C analog component or composite video CVBS on a separate interface for connection to a VCR or a second TV.

A digital video input interface allows the STx7100 to receive noncompressed SD digital video, and to output this via the main and auxiliary displays in place of decoded video. A separate PCM input allows any associated audio to be received, mixed and output in place of decoded audio.

The graphics subsystem comprises a separate 2D blitter, two graphics planes, a cursor plane and dual display compositor. Graphics buffers are created, stored in and displayed from buffers in DDR SDRAM.

The STx7100 embeds an ST40 CPU core for applications and data processing and device control. The CPU boots from flash/SFlash™ on the external memory interface (EMI) and can execute in place or transfer the main executable to the DDR SDRAM and execute from there. CPU data is held in DDR SDRAM where cacheable and noncacheable regions can be programmed. The 16-bit EMI is also used for connecting to external peripherals.

System performance is enhanced with the multichannel FDMA which can be used for 2D block and stream data transfers with minimal CPU intervention.

DVR applications are supported using a hard-disk drive (HDD) connected either to the serial ATA or USB 2.0 interface.

The STx7100 also integrates a range of peripherals, system services and a clock generator module with embedded VCXO to significantly reduce external component cost.

3.2 Omega2 (STBus) interconnect

The Omega2 multipath unified interconnect provides high on-chip bandwidth and low latency accesses between modules. The interconnect operates hierarchically, with latency-critical modules placed at the top level. The multipath router allows simultaneous access paths between modules, and simultaneous read and write phases from different transactions to and from the modules. Split transactions maximize the use of the available bandwidth.

3.3 Processor cores

The STx7100 integrates one ST40 and two ST231 processor cores. The ST40 is used to control most chip functions; the ST231 processors control audio and video/H.264 decoding; see Section 3.15 and Section 3.7 respectively.

The 266 MHz ST40 processor core features a 32-bit superscalar RISC CPU and IEEE-754 compliant floating point unit (FPU). It includes 2-way associative caches, 16 Kbyte instruction cache and 32 Kbyte data cache, as well as core support peripherals such as a real-time clock and interrupt controller.

3.4 Dual local memory interface (LMI)

System LMI and video LMI are 32-bit, high-bandwidth memory interfaces that enable a unified data memory architecture through the use of DDR SDRAM. Both can operate in a 32-bit or 16-bit configuration and have a maximum operating frequency of 200 MHz.

Each LMI interface provides two chip-selects, to support up to two arrays of 1 x 32 devices or 2 x 16 devices. DDR devices of capacity 64, 128, 256, or 512 Mbits can be used, providing a maximum capacity of 4 x 64 Mbytes. When the application includes an HD-MPEG4 decode or a dual HD-MPEG2 decode that needs a high memory bandwidth, then two memory interfaces are used. With this memory configuration, most of the video decoding processes use the video LMI while the other processes of the application use the system LMI.

Accesses to DDR SDRAM go through a local look ahead buffer to enhance bandwidth utilization and to take advantage of pages that are already open, and to open pages in advance. Optimizations also include opcode merging.

Table 3 shows the memory requirements for each type of application.

Table 3: DDR memory requirements for each type of application

Decode type	Memory requirements (Mbytes)	
	System LMI	Video LMI
Single MPEG2 decode only	32	0
Dual MPEG2 decode only	32	32
H.264 decode only		
H.264 and MP2 HD concurrent	64	

3.5 External memory interface (EMI)

The EMI is a 16-bit general-purpose interface for attaching system flash or synchronous flash devices and peripherals. Up to five separate banks are available, each capable of its own strobe timing configuration, and each with its own chip select signal. One bank provides PC-card compatible strobes for implementing a DVB-CI, ATAPI or CableCARD™ module interface.

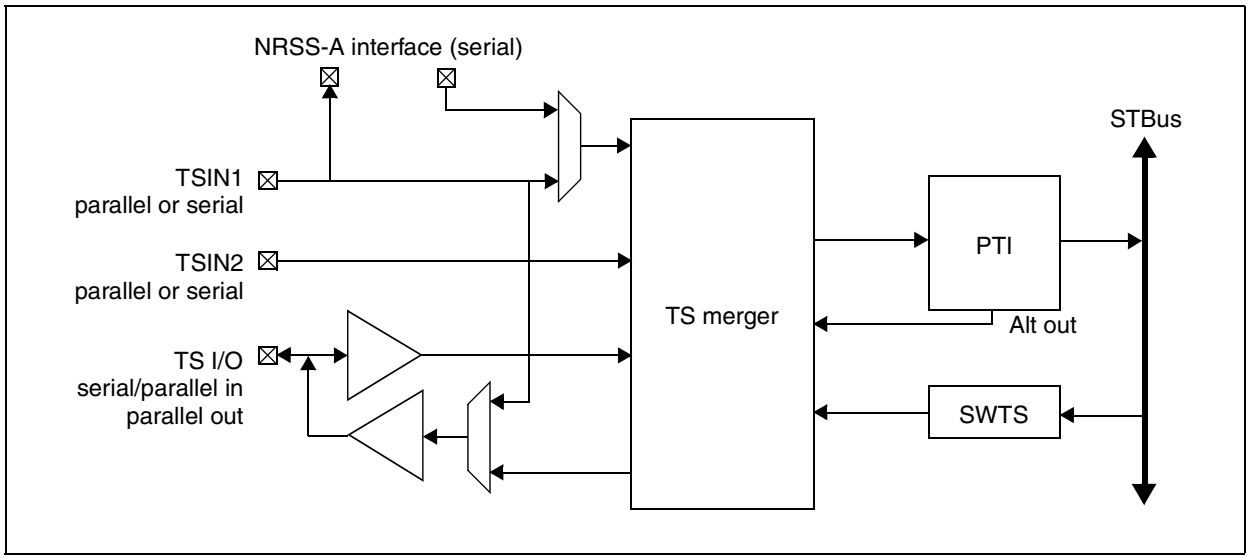
3.6 Transport subsystem

The transport stream subsystem comprises the TS merger/router and a programmable transport stream interface (PTI).

3.6.1 Transport stream input/output

Transport streams are input to the STx7100 via one of three interfaces. Two of these are parallel inputs which can also be configured for serial input if required. The third is a bidirectional parallel interface that can be configured as an input or output.

Figure 5: Transport-stream subsystem



A five-input, two-output transport stream merger/router allows any of the three external inputs to be routed to the PTI. A fourth input is provided for routing internal transport streams stored in memory to the PTI. A fifth input receives a full/partial transport stream created by the PTI and can output it from the STx7100 via the bidirectional interface when configured as an output. This allows TS streams to be sent to a D-VCR via an IEEE1394 link layer controller. Routing to the two outputs can be concurrent.

An NRSS-A interface is available for routing serial TS streams to and from an NRSS-A compatible CA module.

The stream merger/router is also capable of merging any three of the input streams into one TS stream and forwarding this to the PTI allowing it to process three independently sourced transport streams at the same time.

3.6.2 Programmable transport interface (PTI)

The PTI performs PID filtering, demultiplexing, descrambling, and data filtering on up to three transport streams at the same time. The PTI extracts PCRs with time stamps and makes them available to the CPU for clock recovery and audio/video synchronization. The PTI can only extract one PCR at a time, and only one is available for clock recovery.

PES data is transferred by DMA to memory buffers. Section data is transferred by DMA to separate buffers for further processing by the CPU. The PTI can also extract indexing information and then transfer packets, using DMA, to an intermediate buffer for writing to HDD.

Transport streams supported include DIRECTV®, DVB, ATSC, OpenCable, DCII, and ARIB BS4.

The PTI performs PID filtering to select audio, video and data packets to be processed. Up to 96 PID slots are supported.

The PTI can descramble streams using the following ciphers:

- DES-ECB including DVS-042 termination block handling,
- DES-CBC including DVS-042 termination block handling,
- DES-OFB,
- Multi2-ECB including DVS-042 termination block handling,
- Multi2-CBC including DVS-042 termination block handling,
- Multi2-OFB,
- DVB-CSA,
- AES decryption/encryption with cipher block chaining,
- NDS specific streams can also be supported by the integration of ICAM functionality.

The PTI has a 48 x 16 byte section filter core.Three filtering modes are available:

- wide match mode: 48 x 16-byte filters,
- long match mode: 96 x 8-byte filters,
- positive/negative mode: 48 x 8-byte filters with positive/negative filtering at the bit level.

Matching sections are transferred to memory buffers for processing by software.

When the PTI is required to output a transport stream, it can output the entire transport stream or selected packets filtered by PID. A latency counter is provided to ensure packet timing is preserved. Packets can also be substituted.

3.7 H.264/MPEG-2 video decoder

The STx7100's video decoder is a new generation decoder for either SD or HD streams; it is fully compliant with H.264/AVC MP@L4 and HP@L4.1 (main and high profile) video streams as well as ISO/IEC13818-2 MP@ML and MP@HL formats.

The decoder uses a mixed hardware/firmware architecture with a hardwired datapath and a 400 MHz ST231 CPU core engine. It provides flexibility for firmware upgrades, error concealment or trick modes. The core may be used for other coder/decoders at lower resolution when the H.264 decoder is not running. The H.264/MPEG2 video decoder is capable of dual MP@HL decoding.

A stream is decoded picture by picture from an elementary stream buffer. Decoding, reconstruction and prediction buffers are set up by the host CPU. CPU control of bit buffer pointers provides flexibility for trick modes and out of sequence decoding.

Semantic or syntax errors are detected by the decoder and failing macroblocks are replaced up to the next slice or picture.

3.8 Digital video input

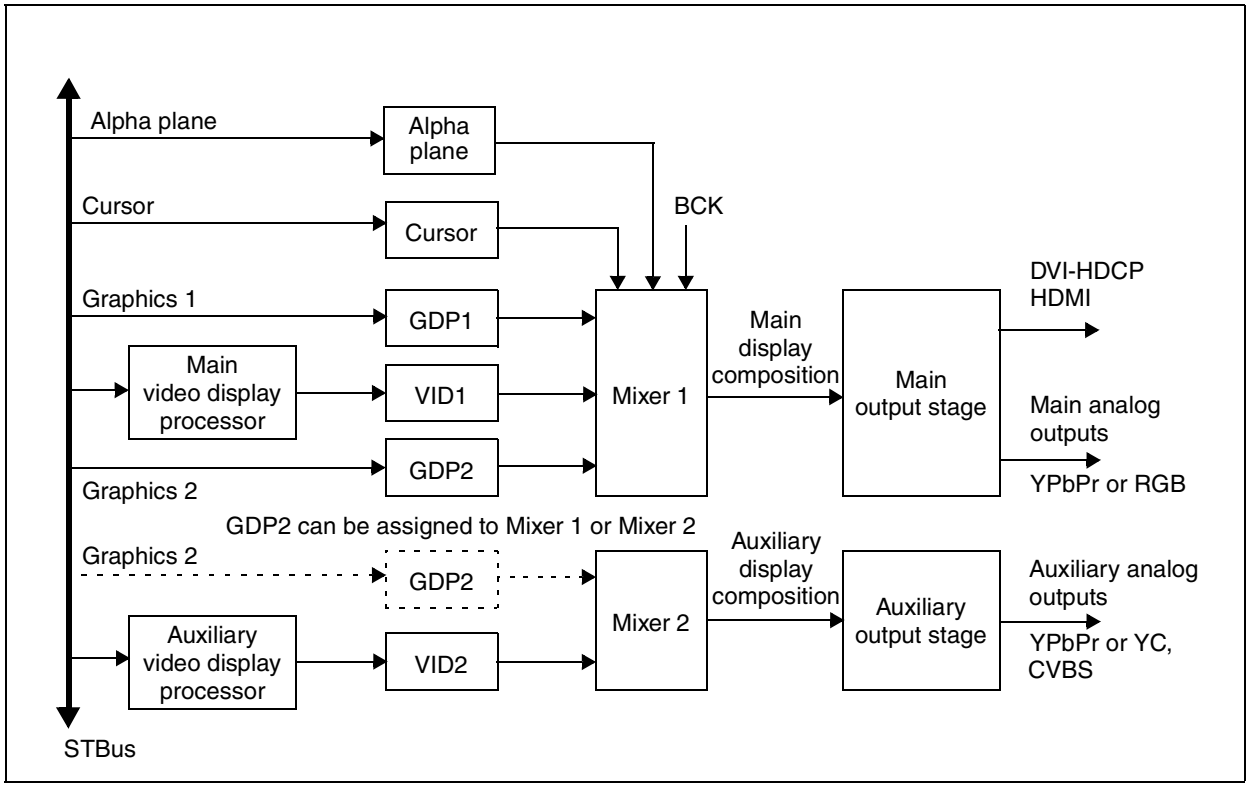
Digital SD video data can be input to the STx7100 via an 8-bit digital video input port.

The 8-bit mode is intended for inputting SD video data conforming to ITU-R BT656 with embedded syncs or ITU-R BT601 with external syncs. In ITU-R BT656 mode, ancillary data embedded in the stream can be extracted to a separate buffer.

3.9 Video display processors

The STx7100 displays video using a main and auxiliary display processors (Figure 6).

Figure 6: Display, composition and output



The video may have been decoded and reconstructed by the H.264/MPEG2 decoder or acquired via the digital video interface.

The same video or two different videos are displayed via the main and auxiliary displays, but each may be set up to format the video differently, and display with different timing. Separate video timing generators (VTGs) are provided to support this.

The display processors are used to present the video from the display buffers and adapt the decoded video format to a format suitable for display taking into account differences in scanning method, resolution, aspect ratio and scanning frequency.

The main-display processor receives decoded or acquired video from memory and performs block-to-line conversion, pan and scan, and vertical and horizontal format conversion. There is also a linear median upconverter (LMU) to perform interlace-to-progressive conversion on standard definition (SD) pictures using motion estimation.

The auxiliary-display processor receives decoded or acquired (and possibly decimated) video and performs pan and scan, vertical format conversion, horizontal format conversion. The output size is limited to an SD format on the auxiliary-display processor, and is intended to output video for VCR recording.

3.10 Graphics display

3.10.1 Graphics layers

The STx7100 has two independent and identical graphics layers known as generic display pipelines (GDPs) (see Figure 6). Each GDP receives pixel data from memory and features the following:

- link-list-based display engine, for multiple viewport capabilities,
- support for ARGBargb formats, including ARGB1555, ARGB4444, RGB565, RGB888, ARGB8565, ARGB8888,
- support for YCbCr4:2:2R, YCbCr888 formats,
- support for premultiplied or non premultiplied RGB components,
- color space conversion matrix, (YCbCr 601/709, chroma signed/unsigned to RGB),
- gain and offset adjustment,
- per-pixel alpha channel combined with per-viewport global alpha,
- 5-tap horizontal sample rate converter, for horizontal upsampling. The resolution is 1/8th pixel (polyphase filter with 8 subpositions),
- color keying capability.

3.10.2 Cursor layer

The cursor is defined as a 128 x 128 pixel area held in local memory, in ACLUT8 format. Each cursor entry is a 16-bit ARGB4444 color + alpha value. The alpha factor of 4 bits handles an antialiased cursor pattern on top of the composed output picture. The cursor-plane features are as follows:

- ACLUT8 format, with ARGB4444 CLUT entries. 256 colors can be simultaneously displayed for the cursor pattern, among 4096 colors associated with a 16-level translucency channel.
- Size is programmable up to 128 x 128.
- Hardware rectangular clipping window, out of which the cursor is never displayed (per-pixel clipping, so only part of the cursor can be out of this window, and consequently transparent).
- Current bitmap is specified using a pointer register to an external memory location, making cursor animation very easy.
- Programmable pitch, so that all cursor patterns can be stored in a single global bitmap.

3.11 Display compositor

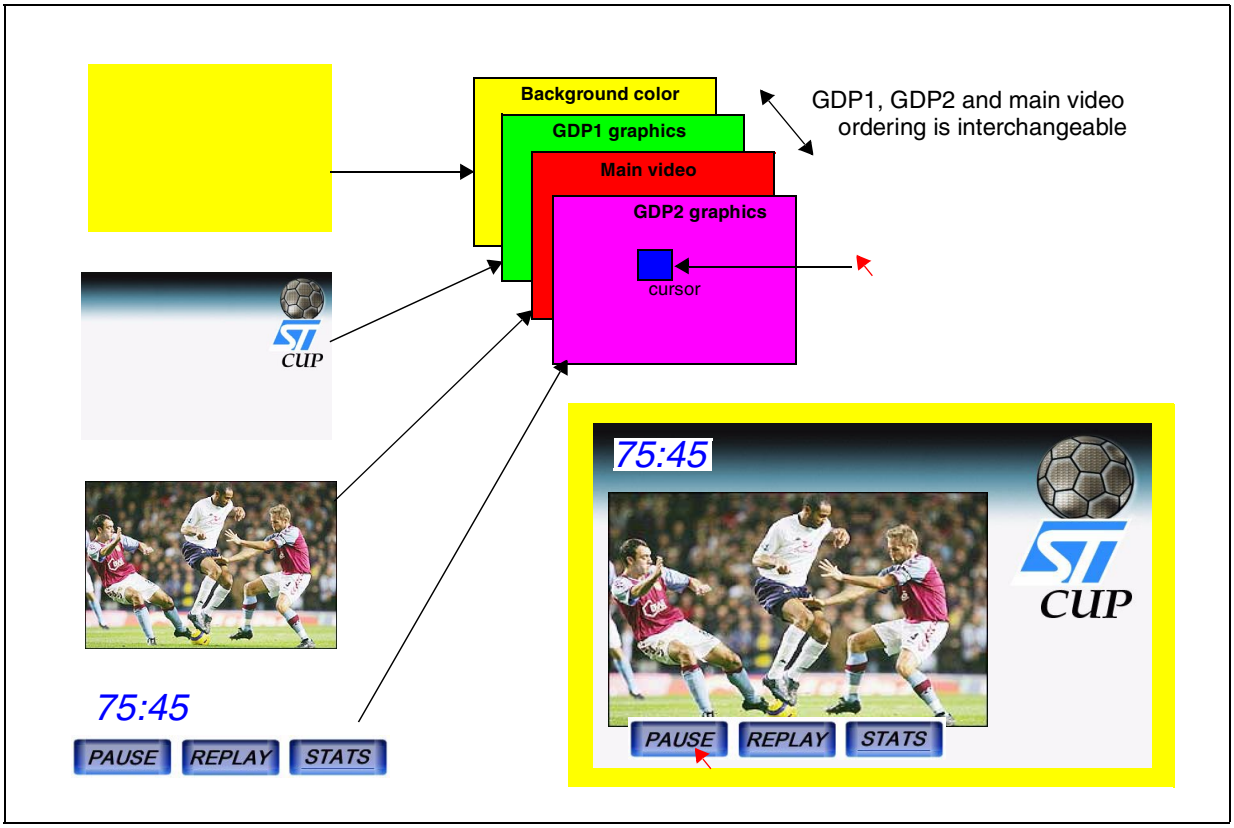
The graphic compositor consists of a 5-layer digital mixer (Mix1) intended for the main TV display (Figure 7) and a 2 layer digital mixer (Mix2) intended as an auxiliary display for applications including connection to a VCR (Figure 8).

Each mixer alpha blends graphics and video layers on a pixel by pixel basis based on alpha component values provided by each layer.

The Mix1 display planes are as follows:

- a background color (RGB888 format, programmable through the registers),
- the two graphics layers GDP1 and GDP2,
- the main video display,
- the cursor plane.

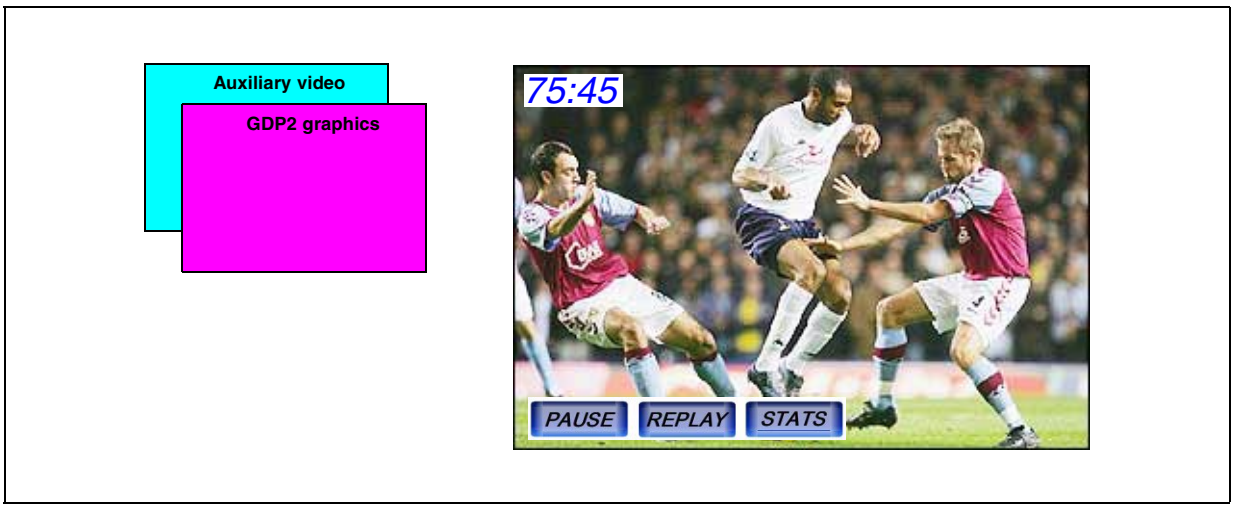
Figure 7: Gamma Mix1 planes



The Mix2 display planes are as follows:

- the auxiliary video display,
- the graphics layer GDP2.

Figure 8: Gamma Mix2 planes



3.12 Main display output stage

The display composition from Mix1 can be output on any of the main display output interfaces (Figure 6). These are:

- the main analog output,
- the DVI/HDMI output.

The main analog output interface supports YPbPr or RGB analog output with or without embedded syncs. High current HD DACs are used to minimize external component count.

Programming flexibility is provided to support different display timings and resolutions. These include support for SMPTE and ARIB SD and HD formats (480i, 480p, 720p, 1080i) and panel displays with a pixel clock up to 74.25 MHz. Timings and levels can be programmed to comply with EIA770.x (x = 1, 2, 3) requirements.

The main analog output can also have Macrovision™ encoding for 480i/480p and 576i/576p and CGMS encoding.

The HDMI output provides DVI-HDCP or HDMI compliant copy protected digital output of the main display composition.

3.13 Auxiliary display output stage

The display composition from Mix2 is output on the auxiliary display output interface (Figure 6).

This interface contains a digital encoder that encodes the output from the auxiliary mixer into standard analog baseband PAL/NTSC signals (Y, C, CVBS).

The digital encoder performs closed-caption, CGMS, WSS, teletext and VPS encoding and allows Macrovision 7.01/ 6.1 copy protection.

An integrated tri-DAC provides three analog TV outputs (S-VHS(Y/C) + CVBS).

3.14 Gamma 2D blitter

The Gamma 2D graphics processor (also called the blitter engine) is a CPU-independent engine for graphics picture processing. It functions as a dual-source 2D DMA, with a set of powerful operators.

The 2D graphics processor receives data from the local memory through two input sources, source 1 and source 2. Source 1 is used for frequent operations such as color-fill or simple source-copy; it has a 64-bit wide internal bus and performs according to the pixel format. All operators always apply to source 2. The processing pipeline bus is always a pixel bus (ARGB8888 format) whatever the format of the source inputs. Sources 1 and 2 are used simultaneously for read/modify/write operations.

The 2D graphics processor is software controlled by a link-list mechanism. Each node of the link list is an instruction that contains all the necessary information to proceed.

The blitter's features are as follows:

- solid color fill of rectangular window,
- solid color shade (fill + alpha blending),
- one source copy, with one or several operators enabled (color format conversion, 2D-scaling),
- two-source copy with alpha blending or logical operation between them,
- 4:2:2 raster/macroblock and 4:2:0 macroblock as source formats, 4:2:2 raster as a destination format,
- color space conversion RGB to/from YCbCr,

- color expansion (CLUT to true color),
- color correction (gamma, contrast, gain),
- color reduction (true color to ACLUT*n*) using an error diffusion algorithm,
- 2D resize engine with high-quality filtering,
- adaptive flicker filter from memory to memory,
- color keying capability,
- rectangular clipping,
- programmable source/target scanning direction, both horizontally and vertically, in order to cope correctly with overlapping source and destination area,
- 1-bit/8-bit clipmask bitmap for random shape clipping can be achieved in two passes,
- plane mask feature available,
- special XYLC access mode, for speeding random pixel access, or horizontal line drawing (polygon filling, run-length decoder accelerator...).

Source and destination windows can all be defined using an XY descriptor, with pixel accuracy whatever the format, from 1 to 32 bpp. Most of these operators can be combined in a single blitter pass: For instance, take a YCbCr 4:2:2 bitmap, convert it to 4:4:4 RGB, resize it and finally blend it on an RGB565 background picture.

3.15 Audio subsystem

Audio decoding

The STx7100 audio subsystem integrates a 400 MHz ST231 CPU core to decode multi-channel compressed audio streams which have been stored in a memory buffer. Once decoded, the audio decoded stream is stored in a separate memory buffer. The audio subsystem also integrates two PCM players and one S/PDIF player which read the decoded data from memory and outputs them to a stereo DAC, a multi-channel PCM output (10 channels) and an S/PDIF output.

The audio stream (encoded or decoded) can be received either from an external source via the PCM input interface or by an internal source such as the transport subsystem via the memory. The audio decoder may have to decode simultaneously two different encoded audio streams when an audio description channel is provided or when a dual MPEG decode is performed.

PCM mixing

The decoded audio stream can be mixed with a PCM file stored in memory following an optional sample rate conversion to adapt the sampling rate. The PCM mixing is fully implemented in the firmware running on the ST231 CPU core integrated in the audio subsystem.

PCM output

A multi-channel decoded PCM stream can be downmixed to generate a 2-channels PCM stream and optionally mixed with a PCM file. The downmixed stream can be then played through a stereo 24-bits DAC. A multi-channel (10 channels) digital PCM output and a digital S/PDIF output are available and can be used independently. The audio DAC, PCM and S/PDIF outputs are independent and have their own audio clock generator.

S/PDIF output

Compressed audio data can also be delivered on the S/PDIF output to be decoded by an external decoder/amplifier.

PCM input

The STx7100 audio subsystem provides a PCM reader to capture an external PCM stream. The stream is then stored in memory.

Features

The audio decoder features are as follows:

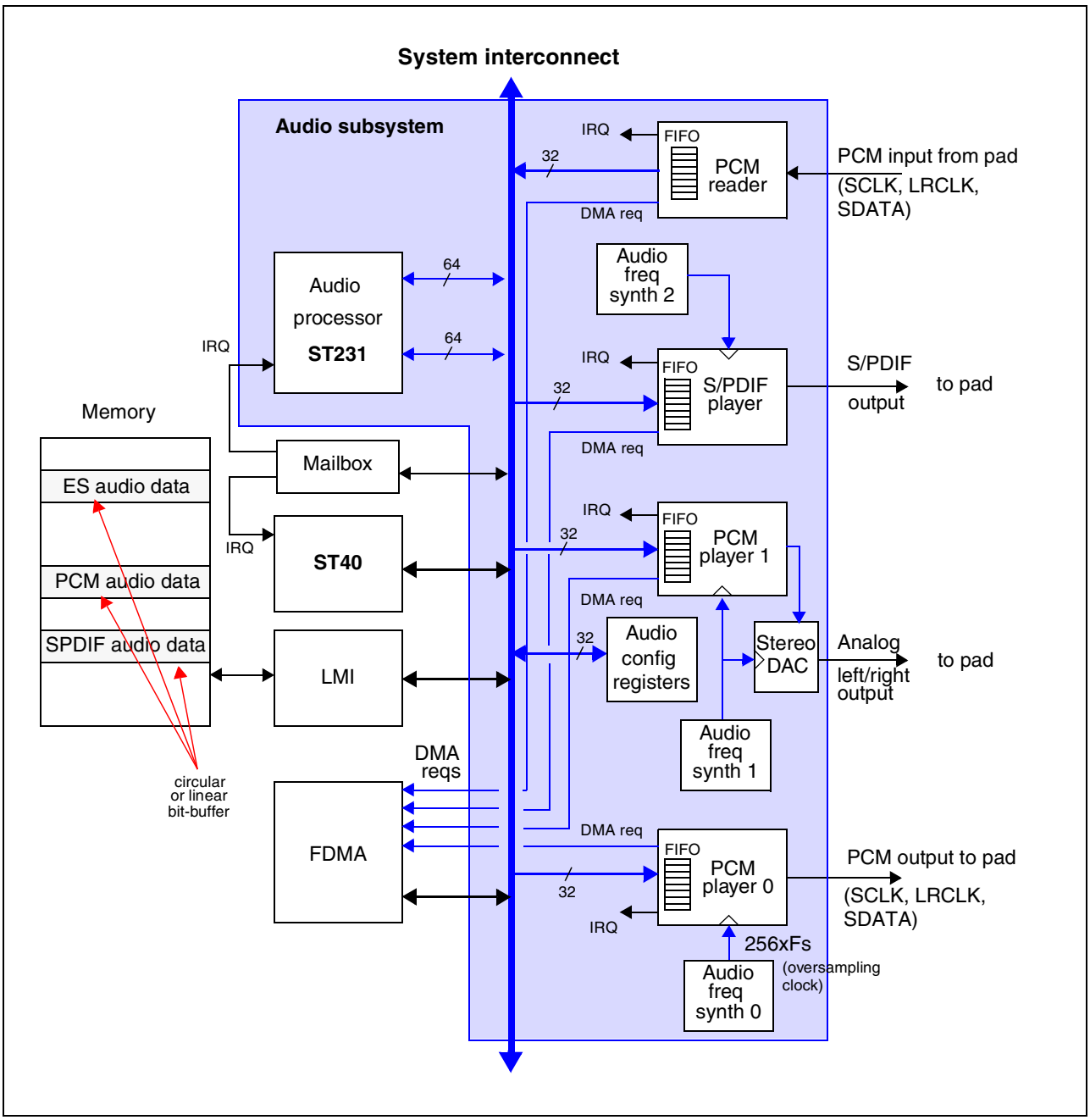
- compatible with all popular audio standards,
- PCM mixing with internal or external source with sample rate conversion,
- encoded (IEC 61937) or decoded (IEC 60958) digital audio on S/PDIF output,
- 6-to-2 channel downmixing,
- PCM audio input (I^2S format),
- multi-channel PCM digital output (10 channels),
- stereo analog output,
- 2-channel description channel decoding,
- postprocessing (channel virtualization).

During the audio decoding process, the host CPU and the FDMA are required.

As the audio decoder is a frame decoder, the host CPU (ST40) controls the ST231 audio processor core frame by frame. A mailbox is used for the communication between the two processors.

The FDMA is required to transfer the data from the memory to the DAC, PCM and S/PDIF players. It is also required to transfer the data from the PCM reader to the memory.

Figure 9: Audio subsystem diagram



3.16 FDMA controller

The STx7100 has a multichannel, burst-capable direct memory access controller that supports the following:

- fast 2D unaligned memory to memory transfers of graphics and stills,
- real-time stream transfers to and from memory with or without pacing[1],
- two external pacing signals for paced transfers to and from external peripherals.

3.17 Interfaces

3.17.1 USB

The STx7100 has an integrated USB host controller with one host port. The USB host interface is compliant with USB rev 2.0 and OHCI/EHCI rev 1.0. This allows the STx7100 to be connected to peripherals such as a hard-disk drive, printer, keyboard, or digital camera. All speeds up to 480 Mbit/s are supported.

3.17.2 Modem

Standard solutions are available for V22bis software modems and V34/V90 controllerless modems ported to the STx7100 architecture.

An interface to the SiLabs DAA is integrated, allowing a 2-wire capacitively coupled connection to the line-side device (no transformer required).

A modem analog front-end (MAFE) interface allows direct connection to an external codec to support this software modem capability. The MAFE interface has its own two-channel DMA controller for sample transfers to and from buffers.

3.17.3 Serial ATA

The STx7100 has an integrated serial ATA controller compliant with V1.0 specifications. All speeds up to 1.5 Gbit/s are supported.

3.17.4 Internal peripherals

The STx7100 has many dedicated internal peripherals for digital TV receiver applications, including:

- two smartcard controllers,
- four ASCs (UARTs), two of which are generally used by the smartcard controllers,
- teletext serializer and DMA,
- three SSCs for I²C/SPI master/slave interfaces,
- six GPIO ports (3.3 V tolerant),
- a four channel PWM module with two PWM outputs, two capture inputs and two compare inputs,
- a multi-channel, infrared blaster/decoder interface module,
- an interrupt level controller with four external interrupt inputs (3.3 V tolerant),
- a DiSEqC 2.0 compatible controller interface,
- a serial communications interface (SCI).

1. These channels are suitable for transfers with internal or external peripherals and for audio and video stream transfers within the STx7100.

3.17.5 Smartcard interfaces

Both STx7100 smartcard interfaces are ISO7816, EMV2000 and NDS compliant with the addition of a simple external power switch.

A programmable hardware power control feature allows the power control signal to be switched when the card's insertion or removal is detected.

3.18 Clock generation

The STx7100 uses two PLLs and two frequency synthesizer banks to generate all the required clocks except for audio. Low jitter PCM audio clocks are generated by three additional frequency synthesizers as described in Section 3.15. VCXO functionality is provided internally by the use of digitally controllable frequency synthesizers and an integrated clock recovery module.

3.19 System services

The STx7100 supports a number of on chip system service functions including:

- reset control,
- watchdog control and reset out,
- low-power control with wake up from internal timer or external interrupt or IR receiver,
- real-time clock,
- JTAG boundary scan,
- diagnostic control and support for the ST40 and ST231 toolsets.

4 CPUs

See also Chapter 20: *Boot modes on page 169*.

4.1 ST40

The STx7100 uses a 400 MHz ST40 as host CPU. This controls the rest of the chip, including the two ST231 CPUs.

The ST40 core and its instruction set are documented in the document *SH-4* [ST40] *Core Architecture* (ADCS 7182230).

4.2 ST231 CPU

The STx7100 uses two ST231 cores, both running at 400 MHz, to perform audio and H.264 video decoding respectively. These are referred to in this datasheet as the audio ST231 and the Delta (H.264 video) ST231.

The ST231 core is documented in the *ST231 Core and Instruction Set Architecture Manual* (ADCS 7645929).

5 Memory map

5.1 Memory map

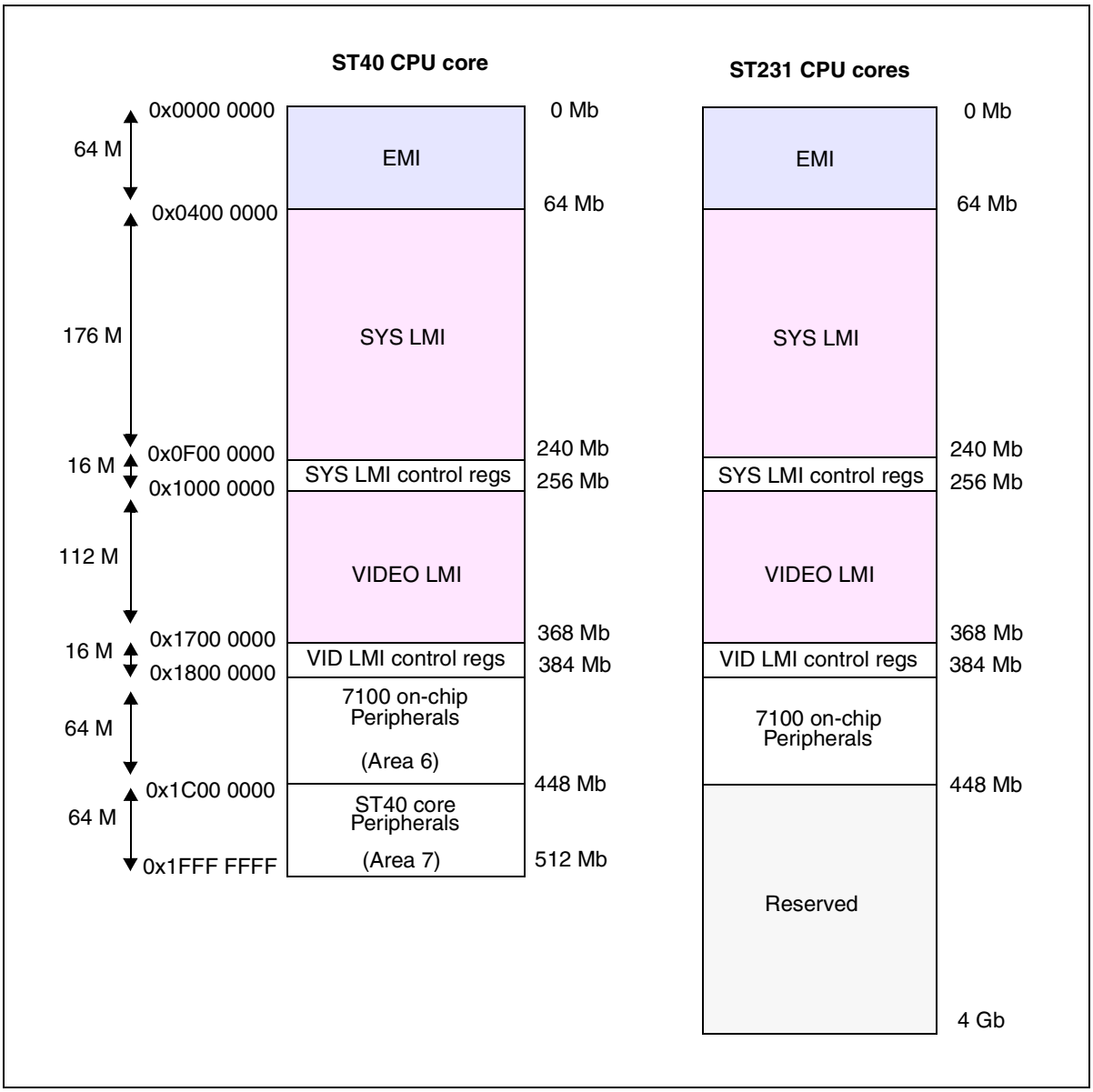
Figure 10 summarizes the memory maps of the ST40 and ST231 cores.

Note: The two ST231 cores and the internal initiators have the same memory map.

Accesses in the reserved area have no effect.

- The **EMI space** starts at base address 0x0000 and occupies **64 Mbytes**
- The **LMI_SYS memory space** starts at base address 0x0400 0000 (64 Mbytes) and occupies **176 Mbytes**. The LMI_SYS configuration registers are located at address 0x0F00 0000 and occupy 16 Mbytes.
- The **LMI_VID memory space** starts at base address 0x1000 0000 (256 Mbytes) and occupies **112 Mbytes**. The LMI_SYS configuration registers are located at address 0x1700 0000 and occupy 16 Mbytes.

Figure 10: ST40 and ST231 memory and peripherals maps



5.2 Internal peripherals map

Table 4 shows the base address of the internal peripherals.

Table 4: Internal peripherals base addresses

Address Range		Size (bytes)	Peripheral
Start	**End**		
0x1800 0000	0x180F FFFF	1 M	**COMMs** configuration registers
0x1810 0000	0x1810 0FFF	4 k	**CSS/CPxM** port and configuration registers
0x1810 1000	0x1810 1FFF	4 k	**PCM Players** port and configuration registers PCM player 0 offset: 0x000 PCM player 1 offset: 0x800
0x1810 2000	0x1810 2FFF	4 k	**PCM Reader** port and configuration registers
0x1810 3000	0x1810 3FFF	4 k	**S/PDIF Player** port and configuration registers S/PDIF player offset: 0x000 **HDMI I^2S to S/PDIF player** offset: 0x800
0x1810 4000	0x18FF FFFF	~15 M (-16 k)	Reserved
0x1900 0000	0x1900 0FFF	4 k	**Clock generator B** configuration registers
0x1900 1000	0x1900 1FFF	4 k	**System** configuration registers
0x1900 2000	0x1900 2FFF	4 k	**HD display** configuration registers
0x1900 3000	0x1900 3FFF	4 k	**SD display** configuration registers
0x1900 4000	0x1900 4FFF	4 k	**LMU** configuration registers
0x1900 5000	0x1900 5FFF	4 k	**VOS** configuration registers **DVO,VTG0, VTG1, upsampler** offset: 0x000 **HDMI** offset: 0x800 **HDCP** offset: 0x400
0x1900 6000	0x1900 FFFF	40 k	Reserved
0x1910 0000	0x191F FFFF	1 M	**USB** configuration registers
0x1920 0000	0x1920 8FFF	36 k	Reserved
0x1920 9000	0x1920 9FFF	4 k	**SATA** configuration registers
0x1920 A000	0x1920 AFFF	4 k	**Compositor** configuration registers
0x1920 B000	0x1920 BFFF	4 k	**Blitter** configuration registers
0x1920 C000	0x1920 CFFF	4 k	**DENC** configuration registers
0x1920 D000	0x1920 FFFF	12 k	Reserved
0x1921 0000	0x1921 0FFF	4 k	**Audio** configuration and **Clock generator C** configuration registers
0x1921 1000	0x1921 1FFF	4 k	**Mailbox 0 (Delta/video/H.264 decoder)** configuration registers
0x1921 2000	0x1921 2FFF	4 k	**Mailbox 1 (audio)** configuration registers
0x1921 3000	0x1921 3FFF	4 k	**Clock generator A** configuration registers
0x1921 4000	0x1921 4FFF	4 k	**H.264 video decoder** configuration registers
0x1921 5000	0x1921 FFFF	44 k	Reserved
0x1922 0000	0x1922 FFFF	64 k	**FDMA** configuration registers
0x1923 0000	0x1923 FFFF	64 k	**PTI** configuration registers
0x1924 0000	0x1924 0FFF	4 k	**MPEG-2 video decoder** configuration registers
0x1924 1000	0x19241FFF	4 k	**DVP** configuration registers
0x1924 2000	0x1924 2FFF	4 k	**TS merger** configuration registers

Table 4: Internal peripherals base addresses

Address Range		Size (bytes)	Peripheral
Start	End		
0x1924 3000	0x1924 31FF	512	Reserved
0x1924 4200	0x1924 FFFF	~52 k	
0x1925 0000	0x19FF FFFF	~14 M	Reserved
0x1A00 0000	0x1A0F FFFF	1 M	**ST231 Delta** peripherals
0x1A10 0000	0x1A1F FFFF	1 M	**EMI** configuration registers
0x1A20 0000	0x1A2F FFFF	1 M	**ST231 audio** peripherals
0x1A30 0000	0x1A30 0FFF	4 k	**TS merger SWTS** registers and FIFO
0x1a30 1000	0x1BFF FFFF	29 M	Reserved
0x1C00 0000	0x1FFF FFFF	64 M	**ST40 Core** peripherals

5.3 Memory management and organization

5.3.1 Memory access

All initiators connected to the system interconnect have access to the memory (EMI and LMI). The CPUs (ST40 and ST231) may use an address translation mechanism (MMU) while the address generated by the other initiators are never translated.

5.3.2 ST40 CPU memory management

The ST40 core supports a 29-bit external address space and a 32-bit physical address space. The ST40 also supports an address translation mechanism which translates a virtual address into a physical address using the memory-management unit (MMU), integrated in the ST40 CPU core.

When virtual addressing is used by the ST40 CPU core, the system peripherals external to the CPU access the 32-bit physical memory space using a 32-bit address in which the most significant three bits are set to zero.

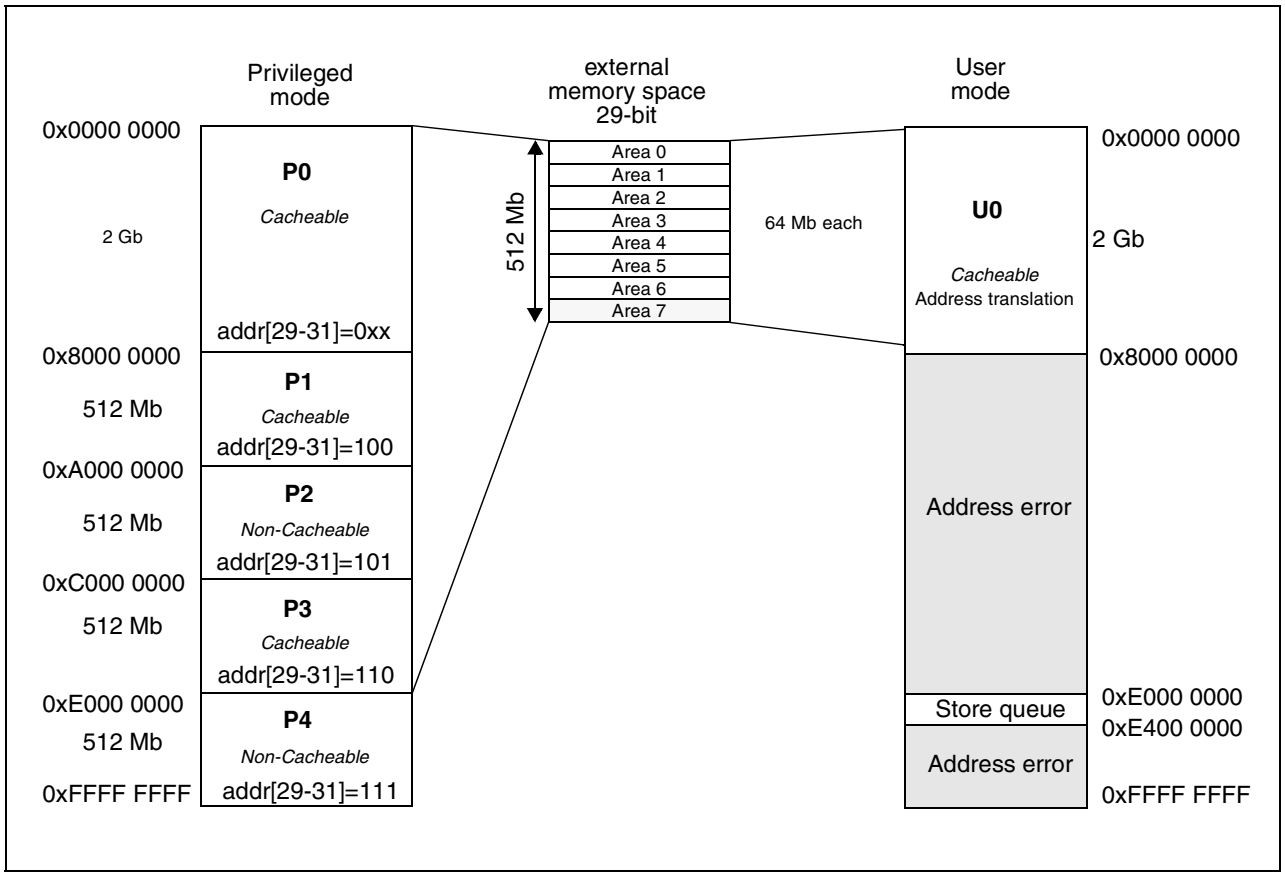
The ST40 CPU has two processor modes: user and privileged. The user mode is for normal operations. The CPU switches to privileged mode when an exception or an interrupt occurs.

5.3.3 Physical address space

When the MMU is disabled, the physical address space (address space accessed by the program) is divided into five regions (P0...P4) using the top three bits of the physical address, as described

in Figure 11. These regions principally differ in whether accesses made to them are cacheable or not. The P4 region is exclusively used to access the ST40 core internal devices.

Figure 11: ST40 CPU physical address space (MMU disabled)

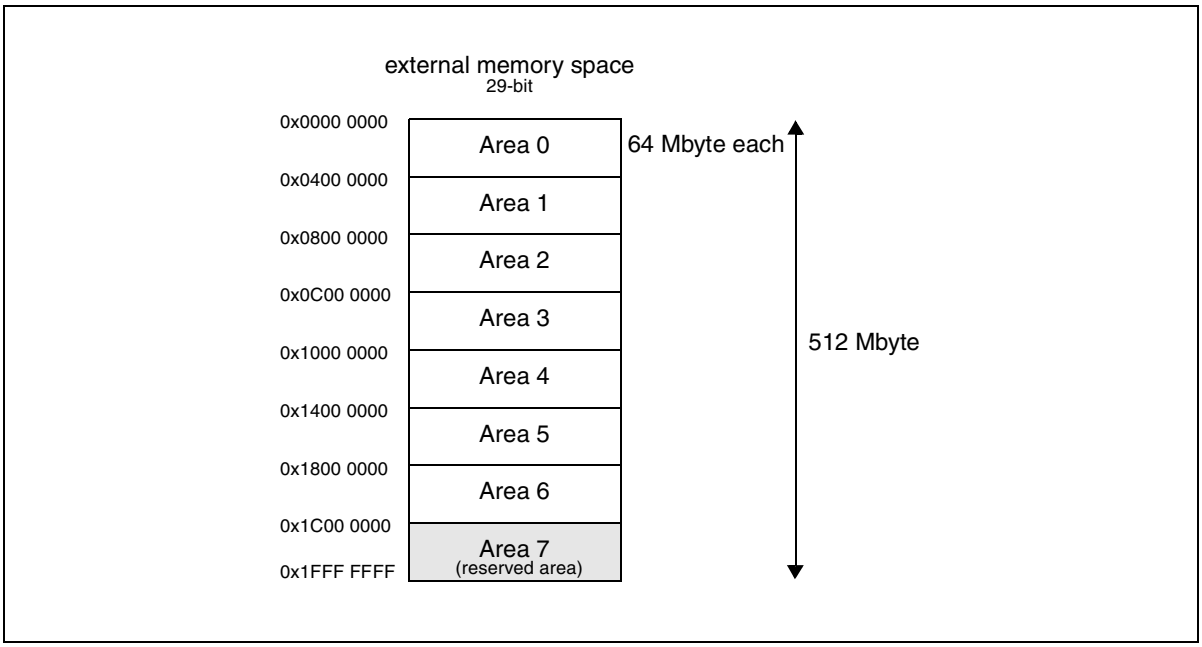


The ST40 includes two 32-byte store queues to perform high-speed writes to external memory.

5.3.3.1 External memory space

The external memory space (29-bit), is divided into 8 areas shown in Figure 12. Areas 0 to 6 relate to memory/peripherals. Area 7 is a reserved area which maps ST40 core component registers and is only accessed via the P4 region (some parts of area 7 can be accessed only in privileged mode).

Figure 12: ST40 CPU External memory space



P0, P1, P3, U0 Regions: The P0, P1, P3 and U0 regions can be accessed using the cache when it is enabled. Zeroing the top three bits of an address in these regions gives the corresponding external memory space address.

P2 Region: The P2 region cannot be accessed using the cache. In the P2 region, zeroing the upper 3 bits of an address gives the corresponding external memory space address.

P4 Region: The P4 region is mapped onto ST40 core I/O channels. This region cannot be accessed using the cache.

5.3.3.2 Virtual address space

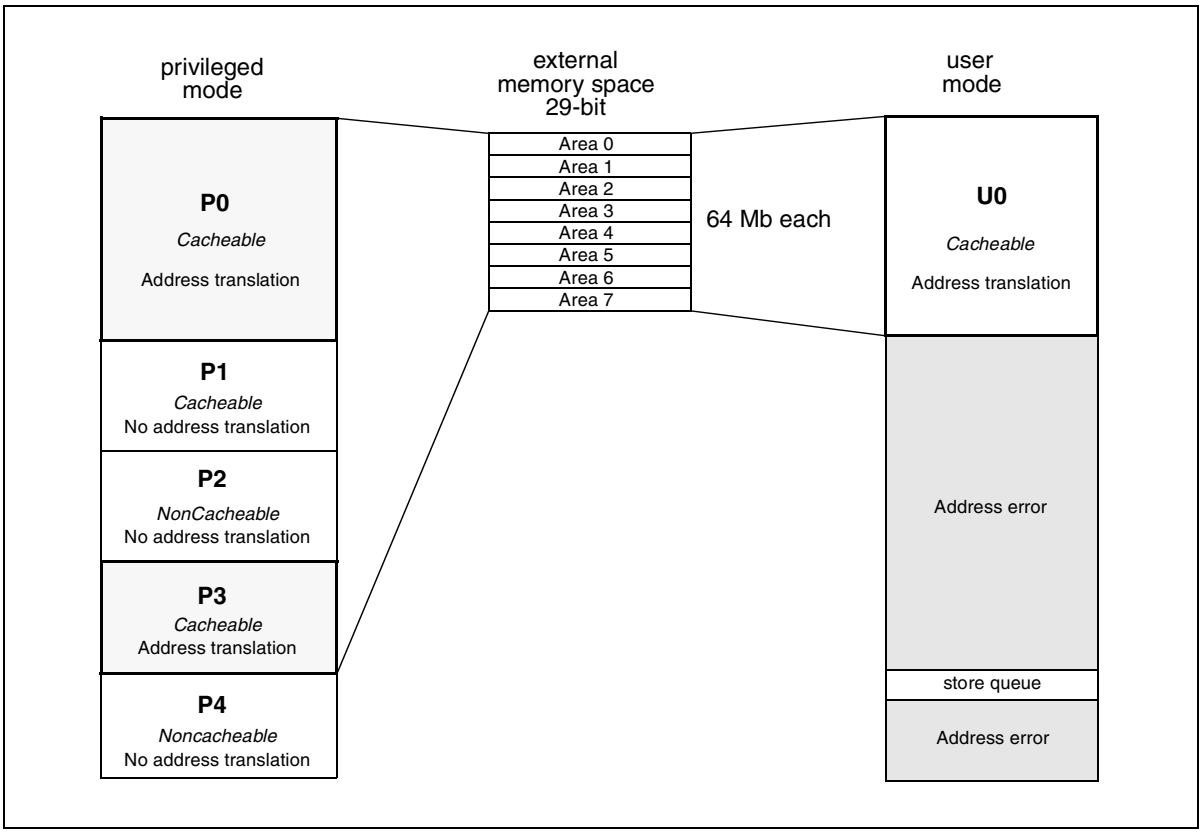
When the MMU is enabled, the regions U0, P0 and P3 can be mapped onto any external memory space in 1-, 4-, or 64-Kbyte, or 1-Mbyte, page units. Mapping from virtual address space to 29-bits external memory space is carried out using the TLB (Translation Look-aside Buffer).

P0, P3, U0 Regions: The P0, P3, and U0 regions allow access using the cache, and address translation using the TLB. These regions can be mapped onto any external memory space in 1-, 4- or 64-Kbyte, or 1-Mbyte page units.

P1, P2, P4 Region: Address translation using the TLB cannot be performed for the P1, P2, or P4 regions. Accesses to these regions are the same as for physical address space.

The ST40 virtual address space is illustrated in Figure 13.

Figure 13: ST40 CPU virtual memory space



5.3.3.3 On-chip RAM space

Inside the ST40 core, half of the (16 kBytes) operand cache can be used as an on-chip RAM. It is not possible to execute instructions out of this on-chip RAM.

5.3.3.4 Cache coherency

The ST40 include an on-chip 16-Kbyte instruction cache for instructions and a 32-Kbyte operand cache for data. Both caches are 2-way set associative with a 32-bytes cache line. The instruction cache has 256 entries and the operand cache has 512 entries.

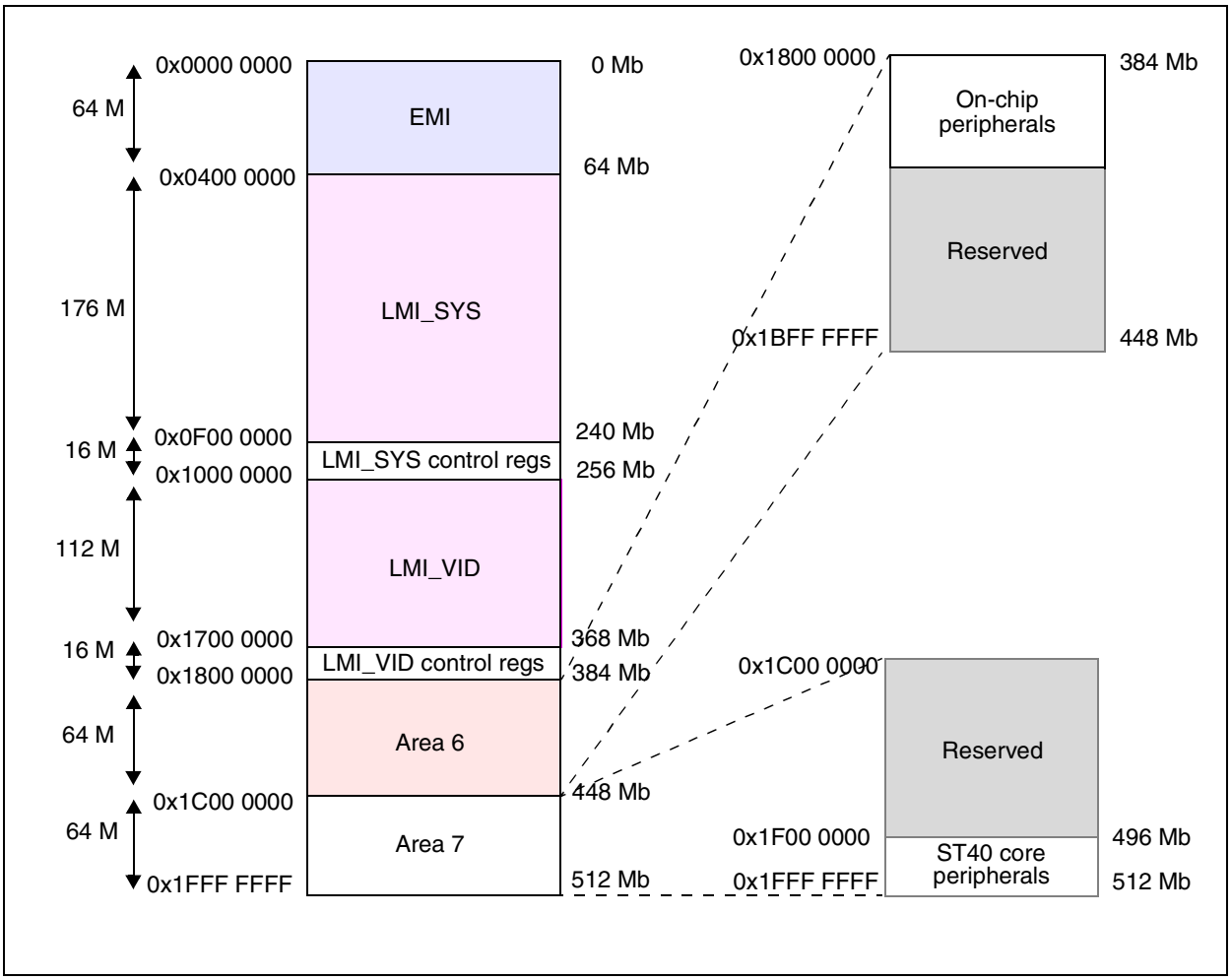
The caches can be used with copy-back or write-through mode. When in copy-back mode, a “dirty” bit set to 1 indicates a mismatch between the data in the cache-line and the external memory. The dirty bit is never set while the cache is being used in write-through mode.

5.3.3.5 ST40 peripherals and memory map

The STx7100 memory space is populated with non-volatile memories, external peripherals (EMI) at base address 0 (ST40 boots at address 0) and with DDR1-SDRAM devices (LMI) at base-address 0x0400 0000 (64 Mbytes)

The STx7100 on-chip peripherals are mapped in area 6 of the ST40.

Figure 14: STx7100 ST40 memory and peripherals map



5.3.4 ST231 CPU memory management

The ST231 core supports a 32-bit external address space and a 32-bit physical address space. The ST231 core also supports an address translation mechanism which translates a virtual address into a physical address using its memory-management unit (MMU) and its integrated TLBs. The ST231 has a small TLB (ITLB, 4 entries) for translating instruction addresses, a small TLB (DTLB, 8 entries) for translating operand addresses and a larger unified TLB (UTLB, 64 entries). The ITLB and DTLB act as a cache for instruction and address translations stored in the UTLB. When the ITLB and DTLB misses, they automatically update from the UTLB.

When the two ST231 cores are used for audio decoding and H.264/AVC decoding, the address translation mechanism is not used.

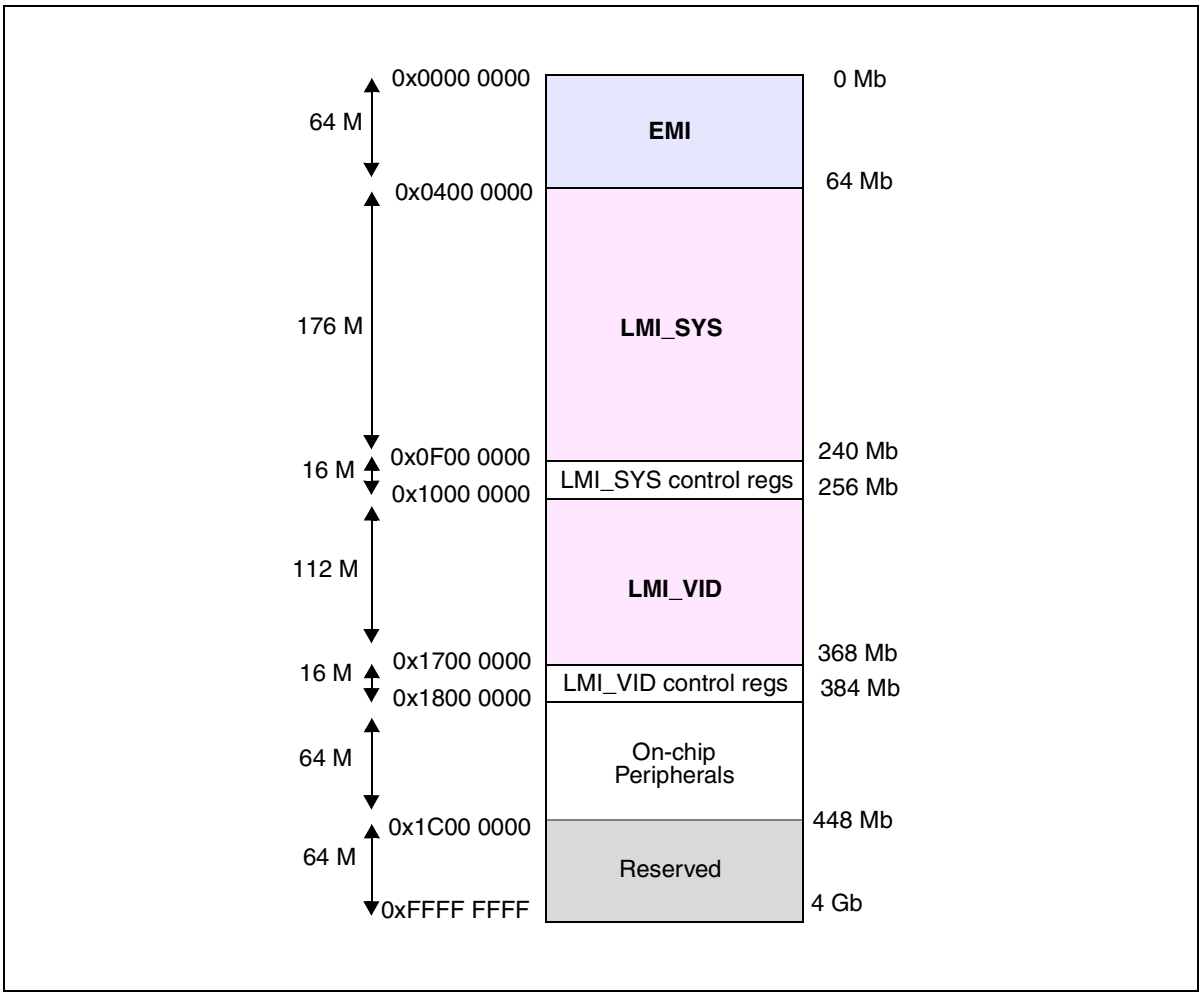
5.3.4.1 Physical address space

The ST231 TLB supports up to 32-bits (4 Gbytes) of address space. The interconnect address decoding restricts the ST231 peripherals and memory map to a subset of that 4 Gbytes space. The ST231 memory map is described in Figure 15.

5.3.4.2 ST231 peripherals and memory map

Figure 15 shows the ST231 memory and peripherals map

Figure 15: STx7100 ST231 memory and peripherals map



5.3.4.3 Virtual address space

The virtual addresses are 32-bit. The TLB performs the mapping from virtual to physical address using one of the following page sizes: 8 Kbyte, 4 Mbyte or 256 Mbyte.

Control register addresses are not translated.

5.3.4.4 Instruction and data cache

The ST231 includes an instruction cache and a data cache.

Instruction buffer

The instruction buffer attempts to fetch ahead in the instruction stream to keep its buffer full. When a branch is taken the instruction buffer is invalidated.

Instruction cache

Instructions are always cached. There is no support for uncached instruction fetching. The cache receives fetch requests from the instruction buffer and returns a group of up to four 32-bit instructions.

The instruction cache is a 32 Kbytes direct-mapped cache with 512 x 64-byte lines.

The virtual address bits [14-6] are used to index the instruction cache RAMs.

The virtual address bits [31-13] are sent to the ITLB for translation. The translated physical address bits [31-13] from the ITLB is then compared with the instruction cache tag.

The virtual address bits [5-0] are used to select the correct bytes from the cache line.

Data cache

The data cache sends write miss and dirty lines to the write buffer.

The data cache is a 32 Kbytes four-way set associative cache built around a 4 x 256 x 32-byte line.

The virtual address bits [12-5] are used to index the data cache RAMs.

The virtual address bits [31-13] are sent to the DTLB for translation. The translated physical address bits [31-13] from the DTLB is then compared with the data cache tag.

The virtual address bits [4-0] are used to select the correct bytes from the cache line.

The data cache can be partitioned into:

- a 24 Kbytes three-way cache plus an 8 Kbytes data RAM,
- a 16 Kbytes two-way cache plus a 16 Kbytes data RAM,
- an 8 Kbytes direct-mapped cache plus a 24 Kbytes data RAM,
- a 24 Kbytes three-way cache plus a separate 8 Kbytes direct-mapped cache,
- a 16 Kbytes two-way cache plus two separate 8 Kbytes direct-mapped cache
- four 8 Kbytes direct-map caches.

Write buffer

The write buffer combines write transactions and sends them out to memory.

5.4 Memory space access per initiator

All the initiators are able to access the LMI SYS memory space but not all the initiators are able to access the LMI VID memory space.

This is described in Table 5 below.

Table 5: Memory space access per Initiator

Memory port	Accessible by	Comment
LMI video high-priority port	ST40 Audio-ST231 Delta-ST231	All CPUs have access to the LMI high priority port.
LMI system high-priority port	ST40 Audio-ST231 Delta-ST231	All CPUs have access to the LMI high priority port.
LMI video low-priority port	ST40 Audio-ST231 Delta-ST231 HD Display Delta HW accelerators MPEG-2 video decoder FDMA HP port Blitter	
LMI system low-priority port	All	All initiators have accessed to this port
EMI port	All but the SD Display	All initiators have accessed to this port except the SD Display.

Part 2

Hardware

6 Package

Package type: PGBA 580 + 100 balls (plastic ball grid array).

Body: 35 x 35 mm.

Figure 16: PGBA 580 +100

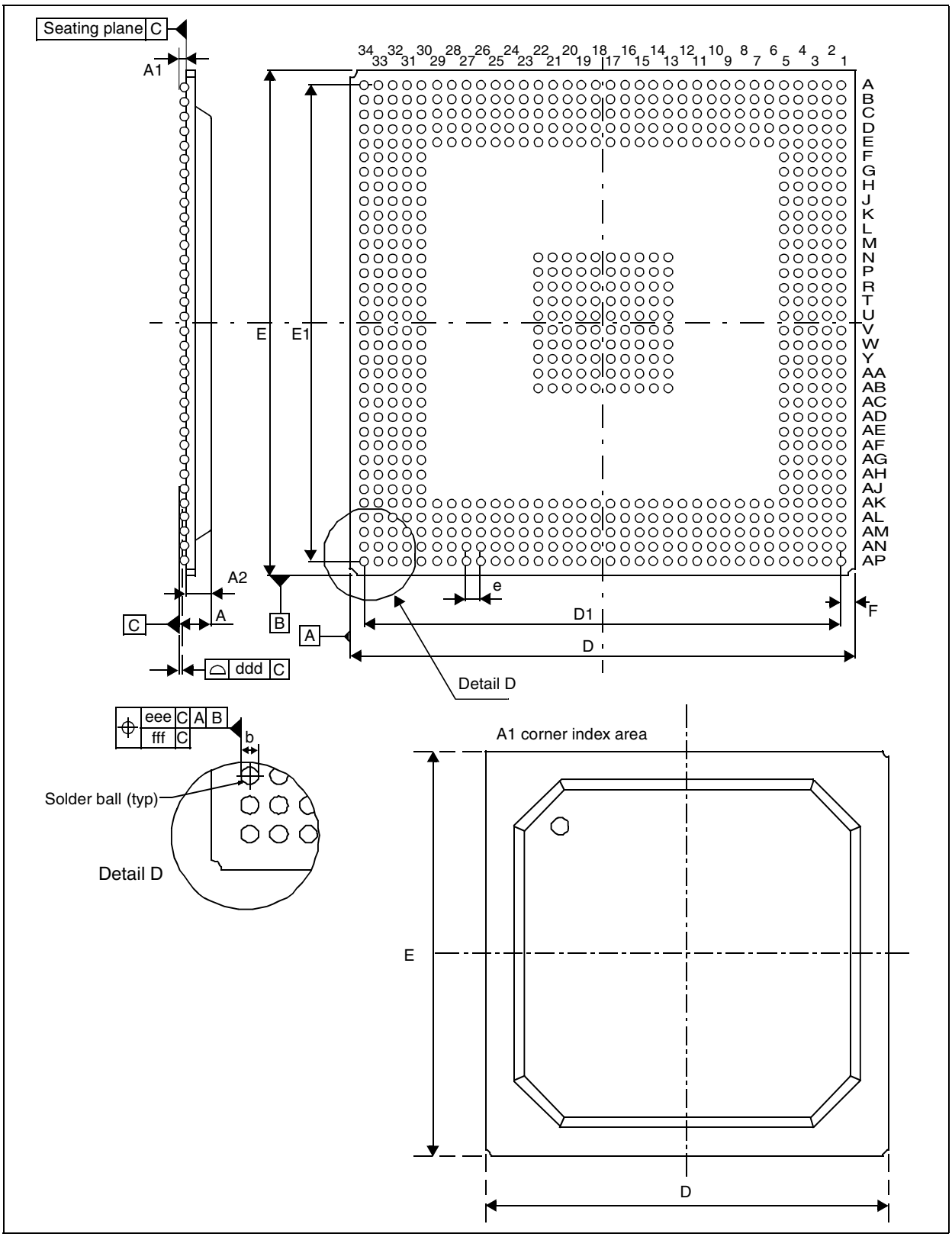


Table 6: Package dimensions

Dimension	Millimeters			Inches			Element
	Min	Typ	Max	Min	Typ	Max	
A			2.500			0.098	Overall thickness
A1	0.300			0.012			Ball height
A2			1.900			0.075	Body thickness
b	0.500	0.600	0.700	0.020	0.024	0.028	Ball diameter
D	34.800	35.000	35.200	1.370	1.378	1.386	Body size
D1		33.000			1.299		Ball footprint
E	34.800	35.000	35.200	1.370	1.378	1.386	Body size
E1		33.000			1.299		Ball footprint
e		1.000			0.039		Ball pitch
F		1.000			0.039		
ddd			0.200			0.008	Coplanarity
eee		0.250			0.010		
fff		0.100			0.004		

In order to meet environmental requirements, STMicroelectronics offers the STx7100 in normal (leaded) and ECOPACK® packages. These packages have a lead-free second level interconnect. The category of second level interconnect is marked on the package and on the inner box label, in compliance with JEDEC Standard JESD97. The maximum ratings related to soldering conditions are also marked on the inner box label. ECOPACK is an ST trademark. ECOPACK specifications are available at: www.st.com.

The Ecopack version can be identified by the letter ‘E’ beside the ST logo. Both types are compatible with ROHS.

Table 7: Package Solder composition, characteristics

	Ecopack (lead-free)	Leaded
Lead free termination surface material (with %)	Sn/Ag/Cu: Ag 3.3 - 4.3%, Cu 0.4 - 1.1%, Sn balance	Sn/Pb: Sn 63%, Pb 37% eutectic
Is the substitution component totally (=100%) without lead? (Yes/No, % of Pb)	Yes 100%	No 37%
Reflow zone: Peak temperature min/max (°C)	235/260	215/240
Reflow zone: Time above liquidus or above 220 °C (seconds)	60 to 90 s above 217°C	60 to 120 s above 183°C
Compatibility with actual SnPb 63/37 solder paste	No	Yes
Compatibility with SnAgCu solder paste with a solder peak of 250 °C	Yes	Yes
MSL impact	3	3

7 Pin list and alternative functions

7.1 Pin-out

Signal names are prefixed by NOT if they are active low; otherwise they are active high.

On the pin-out diagram, black indicates that the pin is reserved and must not be used.

The following pages give the allocation of pins to the package, shown from the top looking down using the PCB footprint.

Table 8: Key to pin-out diagram

Function	Type	Key
System	SIG	
JTAG	SIG	
Transport	SIG	
Display digital	SIG	
Display analog	SIG	
HDMI	SIG	
Audio digital	SIG	
Audio analog	SIG	
PIO/peripheral	SIG	
EMI	SIG	
LMISYS	SIG	
LMIVID	SIG	
USB	SIG	
SATA	SIG	
Analog 1.0 V powers/grounds	VDD/GND	
Analog 2.5 V powers/grounds	VDD/GND	
Analog 3.3 V powers/grounds	VDD/GND	
Digital 1.0 V powers/grounds	VDD/GND	
Digital 2.5 V powers/grounds	VDD/GND	
Digital 3.3 V powers/grounds	VDD/GND	
Not connected - do not connect	NC	

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	LMIVID DATA[8]	LMIVID DATA[10]	LMIVID DATA[12]	LMIVID DATA[14]	LMIVID DATASTR OBE[1]	LMIVID DATA MASK[3]	LMIVID DATA[24]	LMIVID DATA[26]	LMIVID DATA[28]	LMIVID DATA[30]	LMIVID DATA[31]	LMIVID DLL_VDD	LMIVID CLK	LMIVID DATA[0]	LMIVID DATA[2]	LMIVID DATA[4]	LMIVID DATA[6]
B	CKGA_PLL 1_AVDD PLL2V5	LMIVID DATA[9]	LMIVID DATA[11]	LMIVID DATA[13]	LMIVID DATA[15]	LMIVID DATA MASK[1]	LMIVID DATASTR OBE[3]	LMIVID DATA[25]	LMIVID DATA[27]	LMIVID DATA[29]	LMIVIDGN DBGCOMP	LMI VIDREF	NOTLMI VIDCLK	LMIVID DATA[1]	LMIVID DATA[3]	LMIVID DATA[5]	LMIVID DATA[7]
C	SYSA CLKIN	CKGA_PLL 2_AVDD PLL2V5	CKGA_PLL _VDDE2V5	LMIVID VDDE2V5	LMIVID VDDE2V5	LMIVID VDDE2V5	LMIVID VDDE2V5	LMIVID VDDE2V5	LMIVID VDDE2V5	LMIVID VDDE2V5	LMIVID VREF	LMIVID DLL_VSS	LMIVID VDDE2V5	GNDE	GNDE	GNDE	GNDE
D	CKGA_PLL 1_AGND PLL2V5	CKGA_PLL 2_AGND PLL2V5	CKGA_PLL 1_DGND PLL1V0	LMIVID ADD[8]	LMIVID ADD[6]	LMIVID ADD[4]	LMIVID ADD[12]	LMIVID ADD[10]	LMIVID ADD[3]	LMIVID ADD[1]	NOTLMI VIDCS[0]	LMIVID VDDE2V5	LMIVID CLKEN	LMIVID BKSEL[0]	NOTLMI VIDCAS	RTCCLKIN	TRIGGER IN
E	LMISYS DATA[9]	LMISYS DATA[8]	CKGA_PLL 2_DGND PLL1V0	CKGA_PLL 2_DVDD PLL1V0	LMIVID ADD[7]	LMIVID ADD[5]	LMIVID ADD[9]	LMIVID ADD[11]	LMIVID ADD[2]	LMIVID ADD[0]	NOTLMI VIDCS[1]	LMIVID VDDE2V5	LMIVID BKSEL[1]	NOTLMI VIDRAS	NOTLMI VIDWE	NMI	SYSCLK OUT
F	LMISYS DATA[11]	LMISYS DATA[10]	CKGA_PLL 1_DVDD PLL1V0	VDD	VDD												
G	LMISYS DATA[13]	LMISYS DATA[12]	LMISYSDL L_VSS	VDD													
H	LMISYS DATA[15]	LMISYS DATA[14]	GND	LMISYS DLL_VDD	LMISYS VREF												
J	LMISYS DATA MASK[1]	LMISYS DATASTR OBE[1]	GNDE	NOTLMI SYSCAS	NOTLMI SYSWE												
K	LMISYS DATASTR OBE[3]	LMISYS DATA MASK[3]	GNDE	LMISYSBK SEL[0]	NOTLMI SYSRAS												
L	LMISYS DATA[25]	LMISYS DATA[24]	GNDE	NOTLMI SYSCS[0]	LMISYS BKSEL[1]												
M	LMISYS DATA[27]	LMISYS DATA[26]	GNDE	LMISYS ADD[0]	NOTLMI SYSCS[1]												
N	LMISYS DATA[29]	LMISYS DATA[28]	GNDE	LMISYS ADD[2]	LMISYS ADD[1]								GND	GND	GND	VDD	VDD
P	LMISYS DATA[31]	LMISYS DATA[30]	GNDE	LMISYS ADD[10]	LMISYS ADD[3]								GND	GND	GND	GND	VDD
R	LMISYS REF	LMISYSGN DBGCOMP	GNDE	GNDE	GNDE								VDD	GND	GND	GND	VDD
T	LMISYS VDDE2V5	LMISYS VDDE2V5	GNDE	LMISYS ADD[12]	LMISYS ADD[11]								VDD	VDD	GND	GND	GND
U	LMISYS CLK	NOTLMI SYSCLK	LMISYS VDDE2V5	LMISYS ADD[4]	LMISYS ADD[9]								VDD	VDD	GND	GND	GND
V	LMISYS DATA[0]	LMISYS DATA[1]	LMISYS VDDE2V5	LMISYS ADD[6]	LMISYS ADD[5]								VDD	VDD	GND	GND	GND
W	LMISYS DATA[2]	LMISYS DATA[3]	LMISYS VDDE2V5	LMISYS ADD[8]	LMISYS ADD[7]								VDD	VDD	GND	GND	GND
Y	LMISYS DATA[4]	LMISYS DATA[5]	LMISYS VDDE2V5	LMISYS VDDE2V5	LMISYS CLKEN								VDD	GND	GND	GND	VDD
AA	LMISYS DATA[6]	LMISYS DATA[7]	LMISYS VDDE2V5	TSIN1 DATA[7]	LMISYS VDDE2V5								GND	GND	GND	GND	VDD
AB	LMISYS DATASTR OBE[0]	LMISYS DATA MASK[0]	LMISYS VDDE2V5	TSIN1 DATA[5]	TSIN1 DATA[6]								GND	GND	GND	VDD	VDD
AC	LMISYS DATA MASK[2]	LMISYS DATASTR OBE[2]	LMISYS VDDE2V5	TSIN1 DATA[3]	TSIN1 DATA[4]												
AD	LMISYS DATA[16]	LMISYS DATA[17]	GNDE	TSIN1 DATA[1]	TSIN1 DATA[2]												
AE	LMISYS DATA[18]	LMISYS DATA[19]	GNDE	TSIN1 ERROR	TSIN1 DATA[0]												
AF	LMISYS DATA[20]	LMISYS DATA[21]	GNDE	TSIN1 PACKET CLK	TSIN1 BYTECLK VALID												
AG	LMISYS DATA[22]	LMISYS DATA[23]	GNDE	TSIN0 DATA[1]	TSIN1 BYTECLK												
AH	TSIN0 DATA[6]	TSIN0 DATA[7]	GNDE	TSIN0 ERROR	TSIN0 DATA[0]												
AJ	TSIN0 DATA[4]	TSIN0 DATA[5]	GNDE	TSIN0 PACKET CLK	TSIN0 BYTECLK VALID												
AK	TSIN0 DATA[2]	TSIN0 DATA[3]	VDDE 2V5	GNDE	GNDE	TSIN0 BYTECLK	VDDE 2V5	NOT EMICSE	NOT EMICSA	EMI ADDR[2]	EMI ADDR[4]	EMI ADDR[6]	EMI ADDR[8]	EMI ADDR[10]	EMI ADDR[12]	VDD	EMI ADDR[14]
AL	TSIN2 DATA[6]	TSIN2 DATA[5]	TSIN2 DATA[7]	GNDE	GNDE		VDDE 2V5	NOT EMICSB	EMI ADDR[1]	EMI ADDR[3]	EMI ADDR[5]	EMI ADDR[7]	EMI ADDR[9]	EMI ADDR[11]	EMI ADDR[13]	VDD	EMI ADDR[15]
AM	TSIN2 DATA[4]	TSIN2 DATA[3]	TSIN2 BYTECLK	GNDE		GNDE	VDDE 2V5	NOT EMICSC	VDDE 3V3	VDDE 3V3	VDDE 3V3	VDDE 3V3	VDDE 3V3	VDDE 3V3	VDDE 3V3	VDD	VDD
AN	TSIN2 DATA[2]	TSIN2 DATA[1]	TSIN2 BYTECLK VALID	GNDE	DAA_C1A	GNDE	VDDE 2V5	VDDE 3V3		EMI RDNOT WR	NOT EMIBE[1]	EMI DATA[15]	EMI DATA[14]	EMI DATA[13]	EMI DATA[12]	VDD	EMI DATA[11]
AP	TSIN2 DATA[0]	TSIN2 ERROR	TSIN2 PACKET CLK	GNDE	DAA_C2A	GNDE	VDDE 3V3	NOT EMICSD		NOT EMIOE	NOT EMIBE[0]	EMI DATA[7]	EMI DATA[6]	EMI DATA[5]	EMI DATA[4]	VDD	EMI DATA[3]

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
A	LMIVID DATASTROBE[0]	LMIVID DATAMASK[2]	LMIVID DATA[16]	LMIVID DATA[18]	LMIVID DATA[20]	LMIVID DATA[22]	GNDE	AUDPCM OUT0	VDDE 3V3	AUDANAP LEFTOUT	AUDANAP RIGHT OUT		AUD_ GNDA	VIDANA REXT[0]	VIDANA REXT[1]	VIDAN AIDUMPC1	VIDAN AC1OUT
B	LMIVID DATA MASK[0]	LMIVID DATASTROBE[2]	LMIVID DATA[17]	LMIVID DATA[19]	LMIVID DATA[21]	LMIVID DATA[23]	GNDE	AUDPCM OUT1	VDDE 3V3	AUDANAM LEFTOUT	AUDANAM RIGHT OUT	AUD_ VCCA	AUD_ GNDAS	VIDANA GNDA REXT[0]	VIDANA GNDA REXT[1]	VIDANA IDUMPY1	VIDAN AY1OUT
C	GNDE	GNDE	GNDE	GNDE	GNDE	GNDE	GNDE	AUDPCM OUT2	VDDE 3V3	AUDANAI REF	AUDANAV BGFIL	FS0_ VCCA		FS0_ GNDA	DA_HD_0_ GNDA	VIDANAID UMPCV1	VIDAN ACV1OUT
D	TRIGGER OUT	NOT RESETIN	NOTTRST	TCK	TDI	GNDE	AUDSPDIF OUT	AUDPCM OUT3	AUDPCM CLKOUT	GND_ANA	AUDDIG STRBIN	AUDDIG DATAIN	DA_HD_0_ VCCA	VDDE2V5_ AUD_ANA	AGNDPLL 80V0	VIDANAID UMPR0	VIDAN AR0OUT
E	TMUCLK	NOT WDOG RSTOUT	NOT ASEBRK	TMS	TDO	GNDE	AUDS CLKOUT	AUDPCM OUT4	AUDLR CLKOUT	SYSBCLK INALT	AUDDIGLR CLKIN	VDDE2V5_ PLL80_ ANA	DA_SD_0_ VCCA	VDDE2V5_ FS0_ANA	GNDE_ AUD_ANA	VIDANAID UMPB0	VIDAN AB0OUT
F													DA_SD_0_ GNDA	GNDE_4F S_ANA	GNDE_ FS0_ANA	VIDANAID UMPG0	VIDANAG0 OUT
G													FS0_ VDDD	FS0_ GNDD	VDDE2V5_ 4FS_ANA	VDDE 3V3	VDDE 3V3
H													CKGB_4F S1_VDDD	CKGB_4F S1_GNDD	VDDE2V5_ VID_ANA	VIDDIG OUTYC[7]	VIDDIG OUTYC[6]
J													CKGB_4F S0_VDDD	CKGB_4F S0_GNDD	GNDE_VID _ANA	VIDDIG OUTYC[5]	VIDDIG OUTYC[4]
K													DVDD PLL80V0	DGND PLL80V0	GNDE_ PLL80_ ANA	VIDDIG OUTYC[3]	VIDDIG OUTYC[2]
L													CKGB_4F S1_VCCA		CKGB_4F S1_GNDA	VIDDIG OUTYC[1]	VIDDIG OUTYC[0]
M													CKGB_4F S0_VCCA	CKGB_4F S0_GNDA	GND_ANA	VIDDIG OUT HSYNC	VIDDIG OUT VSYNC
N	VDD	VDD	GND	GND	GND								AVDD PLL80V0	VIDDIG OUTYC[15]	TMDS VSSD	TMDS VSSC1	TMDS VSSC0
P	VDD	GND	GND	GND	GND								VIDDIGOU TYC[14]	VIDDIG OUTYC[13]	TMDS VSSP	TMDS TX2P	TMDS TX2N
R	VDD	GND	GND	GND	VDD								VIDDIG OUTYC[12]	VIDDIG OUTYC[11]	TMDS GNDE	TMDS TX1P	TMDS TX1N
T	GND	GND	GND	VDD	VDD								VIDDIG OUTYC[10]	VIDDIG OUTYC[9]	TMDS REF	TMDS TX0P	TMDS TX0N
U	GND	GND	GND	VDD	VDD								VIDDIGOU TYC[8]	GNDE 3V3	TMDS VSSSL	TMDS TXCP	TMDS TXCN
V	GND	GND	GND	VDD	VDD								GNDE 3V3	TMDS VDD	TMDS VSSCK	TMDS VSSX	TMDS VSSC2
W	GND	GND	GND	VDD	VDD								TMDS VDDE3V3	TMDS VDDC0	TMDS VDDCK	TMDS VDDX	TMDS VDDC2
Y	VDD	GND	GND	GND	VDD								PIO5[6]	PIO5[7]	TMDS VDDC1	PIO5[4]	PIO5[3]
AA	VDD	GND	GND	GND	GND								PIO4[7]	PIO5[5]	TMDS VDDSL	PIO5[2]	PIO5[1]
AB	VDD	VDD	GND	GND	GND								PIO4[5]	PIO4[6]	TMDS VDDD	PIO5[0]	PIO3[7]
AC													PIO4[3]	PIO4[4]	TMDS VDDP	PIO3[6]	PIO3[5]
AD													PIO4[1]	PIO4[2]	PIO4[0]	PIO3[4]	PIO3[3]
AE													PIO2[7]	PIO2[6]	PIO3[0]	PIO3[2]	PIO3[1]
AF													VDD	VDDE 3V3	VDDE 3V3	VDDE 3V3	VDDE 3V3
AG													PIO2[4]	PIO2[5]	GNDE 3V3	GNDE 3V3	GNDE 3V3
AH													PIO2[2]	PIO2[3]	GNDE 3V3	PIO1[7]	PIO1[6]
AJ													PIO2[0]	PIO2[1]	GNDE 3V3	PIO1[5]	PIO1[4]
AK	EMIA DDR[16]	EMI ADDR[18]	EMI ADDR[20]	EMI ADDR[22]	EMIT READYOR WAIT	VDDE3V3	VDDE3V3	SYS ITRQ[0]	SYS ITRQ[1]	SYS ITRQ[2]	SYS ITRQ[3]	GND	GND	GND	GNDE 3V3	PIO1[3]	PIO1[2]
AL	EMI ADDR[17]	EMI ADDR[19]	EMI ADDR[21]	EMI ADDR[23]	EMIDMA REQ[1]	USB VSSBS	USB VDDB3V3	USB VDDBS	SATA VDDOSC	SATA VSSOSC	SATA VDDR[1]	SATA VDDDLL	SATA VDDREF	GND	PIO0[7]	PIO1[1]	PIO1[0]
AM	VDD	VDD	EMI BUSREQ	EMI BUSGNT	GND	USB VSSP 2V5	USB VDDBC 2V5	USBREF	SATAVDD OSC2V5	SATA VSSREF	SATA VSSDLL	SATA VDDR[0]	SATA REF		PIO0[0]	PIO0[5]	PIO0[6]
AN	EMI DATA[10]	EMI DATA[9]	EMI DATA[8]	EMI FLASHCLK	EMIDMA REQ[0]	USB VSSC 2V5	USB VDDP 2V5	USBDP	GND	SYSB CLKOSC	SATA VSSR	SATA VDDT[0]	SATA TXP	SATA RXP	GND	PIO0[2]	PIO0[4]
AP	EMI DATA[2]	EMI DATA[1]	EMI DATA[0]	NOT EMIBAA	NOT EMILBA	USB VSSP	USB VDDP	USBDM	GND	SYSB CLKIN	SATA VSST	SATA VDDT[1]	SATA TXN	SATA RXN	GND	PIO0[1]	PIO0[3]

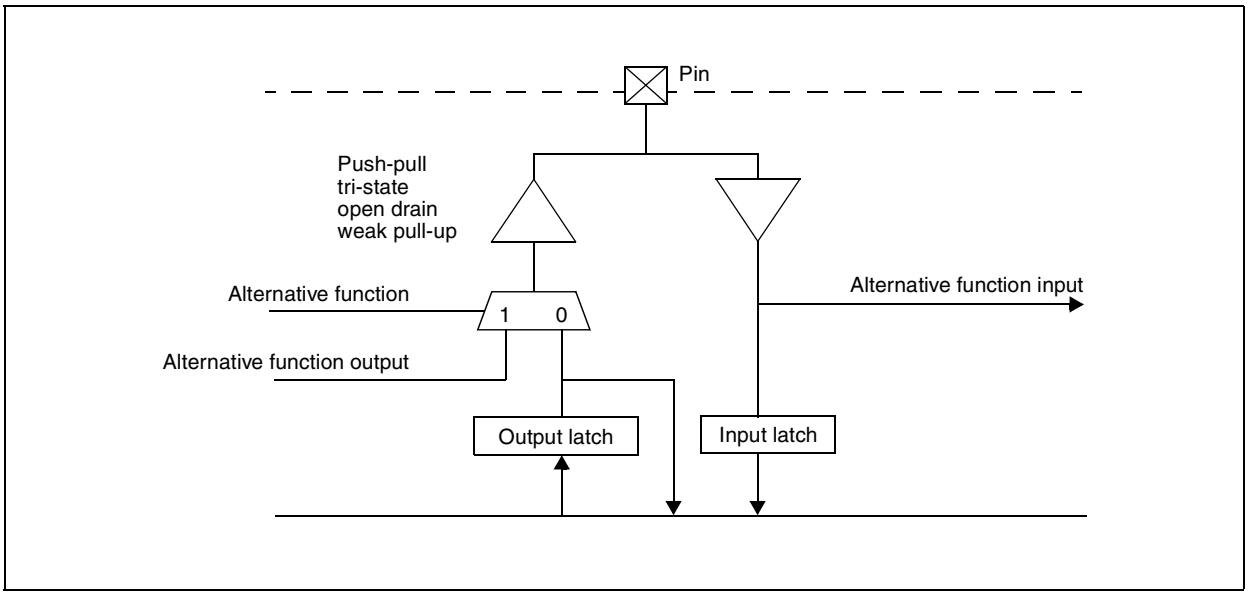
7.2 Alternative functions

To improve flexibility and to allow the STx7100 to fit into different set-top box application architectures, the input and output signals from some of the peripherals and functions are not directly connected to the pins of the device. Instead they are assigned to the alternative function inputs and outputs of a PIO port bit, or an I/O pin. This scheme allows the pins to be configured with their default function if the associated input or output is not required in that particular application.

Inputs connected to the alternative function input are permanently connected to the input pin. The output signal from a peripheral is only connected when the PIO bit is configured into either push-pull or open drain driver alternative function mode.

Alternative function signals are available on some PIO ports.

Figure 17: I/O port pins



7.2.1 Port 0 alternative functions

PIO port 0 is occupied by the UART0/generic smartcard (SC) interface (the smartcard interface consists of a UART and a smartcard clock generator). If not used for a smartcard, port 0 may be used as a standard UART.

Table 9: Port 0 alternative functions

Bit	Alternative function input	Alternative function output
0	-	SC0_DATAOUT (UART0_TXD)
1	SC0_DATAIN (UART0_RXD)	-
2	-	-
3	-	SC0CG_CLK or DSSMCD_CLK[a]
4	UART0_CTS	-
5	-	SC0_SETVCC
6	-	SC0_DIR (UART0_NOTOE)
7	SC0_DETECT	UART0_RTS

a. The smartcard clock can be derived from the smartcard clock generator or directly from clockgen B depending on the value of register SYS_CFG7[4].

7.2.2 Port 1 alternative functions

PIO port 1 is occupied by:

- the UART1/generic smartcard (SC) interface (the smartcard interface consists of a UART and a smartcard clock generator),
- the NRSS-A interface,
- an extended reset output.

Table 10: Port 1 alternative functions

Bit	Alternative function input	Alternative function output
0	-	SC1_DATAOUT (UART_1_TXD)
1	SC1_DATAIN (UART_1_RXD)	-
2	SC1_EXTCLKIN	-
3	-	SC1CG_CLK or DSSMCD_CLK[a]
4	UART1_CTS	NRSSA_CLKOUT
5	NRSSA_DATAIN	UART1_RTS
6	LONG_TIME_RST_OUT	SC1_DIR (UART1_NOTOE)
7	-	NRSSA_DATAOUT

a. the smartcard clock can be derived from the smartcard clock generator or directly from clockgen B depending on the value of SYS_CFG7[5].

7.2.3 Port 2 alternative functions

PIO port 2 is occupied by:

- the SSC0 interface with I^2C/SPI half-duplex/full-duplex modes selectable by the SYS_CFG7[1] bit value,
- the MAFE interface,
- the digital video blanking output (DVBO) selectable by the SYS_CFG7[10] bit value).

Table 11: Port 2 alternative functions

Bit	Alternative function input		Alternative function output		
	SYS_CFG7[1] = 0	SYS_CFG7[1]= 1	SYS_CFG7[10] = 0		SYS_CFG7[10] = 1
			SYS_CFG7[1] = 0	SYS_CFG7[1]= 1	
0	SSC0_SCL		SSC0_SCL		DVO_BLANK
1	SSC0_MTSR	SSC0_MRST	SSC0_MTSR	SSC0_MRST	-
2	SSC0_MRST		SSC0_MRST		-
3	-		MAFE_HC1		-
4	-		MAFE_DOUT		-
5	MAFE_DIN		-		-
6	MAFE_FS		-		-
7	MAFE_SCLK		-		-

7.2.4 Port 3 alternative functions

PIO port 3 is occupied by:

- the SSC1 interface with I^2C/SPI half-duplex/full-duplex modes selectable by the SYS_CFG7[2] bit value,
- the IR Blaster interface,
- the video horizontal and vertical synchro outputs selectable by the SYS_CFG7[10] bit value, the selection between the main and aux timing generators being done using SYS_CFG7[11].

The SSC1 and video synchro outputs are multiplexed, selectable by the SYS_CFG7[10] bit value.

Table 12: PIO port 3 alternative functions

Bit	Alternative function input		Alternative function output			
	SYS_CFG7[2] = 0	SYS_CFG7[2] = 1	SYS_CFG7[10] = 0		SYS_CFG7[10] = 1	
			SYS_CFG7[2] = 0	SYS_CFG7[2] = 1	SYS_CFG7[11] = 0	SYS_CFG7[11] = 1
0	SSC1_SCL		SSC1_SCL		VTG_MAIN_HREF	VTG_AUX_HREF
1	SSC1_MTSR	SSC1_MRST	SSC1_MTSR	SSC1_MRST	VTG_MAIN_VREF	VTG_AUX_VREF
2	SSC1_MRST		SSC1_MRST		VTG_MAIN_NTOP	VTG_AUX_NTOP
3	IRB_IR_IN		-			
4	IRB_UHF_IN		-			
5	-		IRB_IR_DATAOUT			
6	-		IRB_IR_DATAOUT_OD			
7	DENC_CFC		-		-	

7.2.5 Port 4 alternative functions

PIO port 4 is occupied by:

- the SSC2 interface with I^2C full-duplex mode, selectable by the SYS_CFG7[3] bit value,
- UART2 multiplexed with the SCIF interface from the ST40, selectable by the SYS_CFG7[0] bit value,
- the PWM0/PWM1 outputs.

Table 13: PIO port 4 alternative functions - bits 0 to 1

Bit	Alternative function input		Alternative function output	
	SYS_CFG7[3] = 0	SYS_CFG7[3] = 1	SYS_CFG7[3] = 0	SYS_CFG7[3] = 1
0	SSC2_SCL		SSC2_SCL	
1	SSC2_MTSR	SSC2_MRST	SSC2_MTSR	SSC2_MRST

Table 14: PIO port 4 alternative functions - bits 2 to 7

Bit	Alternative function input		Alternative function output	
	SYS_CFG7[0] = 0	SYS_CFG7[0] = 1	SYS_CFG7[0] = 0	SYS_CFG7[0] = 1
2	UART2_RXD	SCI_RXD	-	
3	-		UART2_TXD	SCI_TXD
4	UART2_CTS	SCI_CTS	SCI_CTS	
5	SCI_RTS		UART2_RTS	SCI_RTS
6	SCI_SCK		PWM_OUT0	SCI_SCK
7	-		PWM_OUT1	

7.2.6 Port 5 alternative functions

PIO port 5 is occupied by:

- the UART3,
- the observation of the clocks generated by clock generator B,
- the PWM capture and compare signals,
- the DiSEqC interface,
- the USB2 power-enable and over-current signals.

Table 15: PIO port 5 alternative functions

Bit	Alternative function input	Alternative function output
0	-	UART3_TXD
1	UART3_RXD	-
2	UART3_CTS	SYSCLKOUTB
3	-	UART3_RTS
4	DISEQC_RX_DATAIN	PWM_COMPAREOUT0
5	PWM_CAPTUREIN0	DISEQC_TX_DATAOUT
6	USB2_PRT_OVRCUR	PWM_COMPAREOUT1
7	PWM_CAPTUREIN1	USB2_PRT_PWR

8 Connections

This chapter contains detail of pins, alternative functions and connection diagrams, listed in the following functional groups:

- power supplies (analog and digital) on page 60,
- system on page 68,
- JTAG on page 68,
- transport input including NRSS-A interface and D1 input on page 69,
- display digital output interface on page 71,
- display analog output interface on page 72,
- HDMI interface on page 72,
- audio digital interface on page 73,
- audio analog interface on page 73,
- programmable I/O (PIO) on page 74,
- external memory interface (EMI) on page 75,
- system local memory interface (LMISYS) on page 78,
- video local memory interface (LMIVID) on page 80,
- USB2 interface on page 82,
- SATA-I interface on page 83,
- peripherals:
  - DAA interface on page 84,
  - asynchronous serial controller (ASC) on page 85,
  - infrared transmitter/receiver on page 85,
  - modem analog front-end interface on page 86,
  - PWM on page 86,
  - smartcard on page 87,
  - synchronous serial controller (SSC) on page 87,
  - DiSEqC™ 2.0 interface on page 88.

8.1 Power supplies

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
USB2					
AL24	USBVDDB3V3	P	3.3	USB1.1 mode buffer power	TBD
AM24	USBVDDBC2V5	P	2.5	USB2 2.5 V buffer power	TBD
AN23	USBVSSC2V5	G	0	USB2 2.5 V buffer ground	
AL25	USBVDDBS	P	1.0	USB2 buffer/serdes power	TBD
AL23	USBVSSBS	G	0	USB2 buffer/serdes ground	
AN24	USBVDDP2V5	P	2.5	USB2 2.5 V PLL power	TBD
AM23	USBVSSP2V5	G	0	USB2 2.5 V PLL ground	
AP24	USBVDDP	P	1.0	USB2 PLL power	TBD
AP23	USBVSSP	G	0	USB2 PLL ground	
SATA-I					
AN29	SATAVDDT[0]	P	1.0	SATA transmitter power	10 nF + 100 pF
AP29	SATAVDDT[1]	P	1.0	SATA transmitter power	10 nF + 100 pF
AP28	SATAVSST	G	0	SATA transmitter ground	-
AM29	SATAVDDR[0]	P	1.0	SATA receiver power	10 nF + 100 pF
AL28	SATAVDDR[1]	P	1.0	SATA receiver power	10 nF + 100 pF
AN28	SATAVSSR	G	0	SATA receiver ground	-
AL30	SATAVDDREF	P	1.0	SATA compensation power	10 nF + 100 pF
AM27	SATAVSSREF	G	0	SATA compensation ground	-
AM26	SATAVDDOSC2V5	P	2.5	SATA 2.5 V oscillator power	10 nF + 100 pF
AL26	SATAVDDOSC	P	1.0	SATA 1.0 V oscillator power	10 nF + 100 pF
AL27	SATAVSSOSC	G	0	SATA oscillator ground	-
AL29	SATAVDDDLL	P	1.0	SATA DLL power	10 nF + 100 pF
AM28	SATAVSSDLL	G	0	SATA DLL ground	-
TMDS					
W31	TMDSVDDC0	P	1.0	TMDS data0 channel power	TBD
N34	TMDSVSSC0	G	0	TMDS data0 channel ground	
Y32	TMDSVDDC1	P	1.0	TMDS data1 channel power	TBD
N33	TMDSVSSC1	G	0	TMDS data1 channel ground	
W34	TMDSVDDC2	P	1.0	TMDS data2 channel power	TBD
V34	TMDSVSSC2	G	0	TMDS data2 channel ground	
W32	TMDSVDDCK	P	1.0	TMDS clock channel power	TBD
V32	TMDSVSSCK	G	0	TMDS clock channel ground	
AC32	TMDSVDDP	P	1.0	TMDS serializer power	TBD
P32	TMDSVSSP	G	0	TMDS serializer ground	
W33	TMDSVDDX	P	1.0	TMDS DLL power	TBD
V33	TMDSVSSX	G	0	TMDS DLL ground	
AA32	TMDSVDDSL	P	1.0	TMDS deserializer power	TBD
U32	TMDSVSSSL	G	0		

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
AB32	TMDSVDDD	P	1.0	TMDS DLL2 power	TBD
N32	TMDSVSSD	G	0	TMDS DLL2 ground	
W30	TMDSVDDE3V3	P	3.3	TMDS 3.3 V power	TBD
R32	TMDSGNDE	G	0	TMDS 3.3 V ground	
V31	TMDSVDD	P	1.0	TMDS 1.0 V power	TBD
VDAC					
D30	DA_HD_0_VCCA	P	2.5	VDAC0 power	TBD
C32	DA_HD_0_GNDA	G	0	VDAC0 ground	
E30	DA_SD_0_VCCA	P	2.5	VDAC1 power	TBD
F30	DA_SD_0_GNDA	G	0	VDAC1 ground	
H32	VDDE2V5_VID_ANA	P	2.5	VDAC ring power	TBD
J32	GNDE_VID_ANA	G	0	VDAC ring ground	
ADAC					
B29	AUD_VCCA	P	2.5	ADAC power	TBD
A30	AUD_GNDA	G	0	ADAC ground	
B30	AUD_GNDAS	G	0	ADAC substrate ground	
D31	VDDE2V5_AUD_ANA	P	2.5	ADAC ring power	TBD
E32	GNDE_AUD_ANA	G	0	ADAC ring ground	
CLOCKGENA					
B1	CKGA_PLL1_AVDDPLL2V5	P	2.5	PLL1 analog power	TBD
D1	CKGA_PLL1_AGNDPLL2V5	G	0	PLL1 analog ground	
F3	CKGA_PLL1_DVDDPLL1V0	P	1.0	PLL1 digital power	TBD
D3	CKGA_PLL1_DGNDPLL1V0	G	0	PLL1 digital ground	
C2	CKGA_PLL2_AVDDPLL2V5	P	2.5	PLL2 analog power	TBD
D2	CKGA_PLL2_AGNDPLL2V5	G	0	PLL2 analog ground	
E4	CKGA_PLL2_DVDDPLL1V0	P	1.0	PLL2 digital power	TBD
E3	CKGA_PLL2_DGNDPLL1V0	G	0	PLL2 digital ground	
C3	CKGA_PLL_VDDE2V5	P	2.5	PLL ring power	TBD
CLOCKGENB					
M30	CKGB_4FS0_VCCA	P	2.5	CKGB QFS0 analog power	TBD
M31	CKGB_4FS0_GNDA	G	0	CKGB QFS0 analog ground	
J30	CKGB_4FS0_VDDD	P	1.0	CKGB QFS0 digital power	TBD
J31	CKGB_4FS0_GNDD	G	0	CKGB QFS0 digital ground	
L30	CKGB_4FS1_VCCA	P	2.5	CKGB QFS1 analog power	TBD
L32	CKGB_4FS1_GNDA	G	0	CKGB QFS1 analog ground	
H30	CKGB_4FS1_VDDD	P	1.0	CKGB QFS1 digital power	TBD
H31	CKGB_4FS1_GNDD	G	0	CKGB QFS1 digital ground	
G32	VDDE2V5_4FS_ANA	P	2.5	CKGB ring power	TBD
F31	GNDE_4FS_ANA	G	0	CKGB ring ground	
N30	AVDDPLL80v0	P	2.5	PLL3 analog power	TBD
D32	AGNDPLL80v0	G	0	PLL3 analog ground	

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
K30	DVDDPLL80v0	P	1.0	PLL3 digital power	TBD
K31	DGNDPLL80v0	G	0	PLL3 digital ground	
E29	VDDE2V5_PLL80_ANA	P	2.5	PLL3 ring power	TBD
K32	GNDE_PLL80_ANA	G	0	PLL3 ring ground	
CLOCKGENC					
C29	FS0_VCCA	P	2.5	CKGC FS0 analog power	TBD
C31	FS0_GNDA	G	0	CKGC FS0 analog ground	
G30	FS0_VDDD	P	1.0	CKGC FS0 digital power	TBD
G31	FS0_GNDD	G	0	CKGC FS0 digital ground	
E31	VDDE2V5_FS0_ANA	P	2.5	CKGC ring power	TBD
F32	GNDE_FS0_ANA	G	0	CKGC ring ground	
LMI					
T1	LMISYSVDDE2V5	P	2.5	LMISYS power[a]	TBD
T2					
U3					
V3					
W3					
Y3					
Y4					
AA3					
AA5					
AB3					
AC3					
H4	LMISYSDLL_VDD	P	1.0	LMISYS DLL power	TBD
G3	LMISYSDLL_VSS	G	0	LMISYS DLL ground	
C4	LMIVIDVDDE2V5	P	2.5	LMISYS power[b]	TBD
C5					
C6					
C7					
C8					
C9					
C10					
C13					
D12					
E12					
A12	LMIVIDDLL_VDD	P	1.0	LMIVID DLL power	TBD
C12	LMIVIDDLL_VSS	G	0	LMIVID DLL ground	
Analog					
D27	GND_ANA	G	0	Analog ring ground	
M32					

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
Digital 1.0 V					
F4	VDD	P	1.0	Digital 1.0 V ring power	TBD
F5					
G4					
N16					
N17					
N18					
N19					
P17					
P18					
R13					
R17					
R18					
R22					
T13					
T14					
T21					
T22					
U13					
U14					
U21					
U22					
V13					
V14					
V21					
V22					
W13					
W14					
W21					
W22					
Y13					
Y17					
Y18					
Y22					
AA17					
AA18					
AB16					
AB17					

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
AB18	VDD	P	1.0	Digital 1.0 V ring power	TBD
AB19					
AF30					
AK16					
AL16					
AM16					
AM17					
AM18					
AM19					
AN16					
AP16					
H3	GND	G	0	Digital ring ground	TBD
N13					
N14					
N15					
N20					
N21					
N22					
P13					
P14					
P15					
P16					
P19					
P20					
P21					
P22					
R14					
R15					
R16					
R19					
R20					
R21					
T15					
T16					
T17					
T18					
T19					
T20					
U15	GND	G	0	Digital ring ground	TBD
U16					
U17					

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
U18	GND	G	0	Digital ring ground	TBD
U19					
U20					
V15					
V16					
V17					
V18					
V19					
V20					
W15					
W16					
W17					
W18					
W19					
W20					
Y14					
Y15					
Y16					
Y19					
Y20					
Y21					
AA13					
AA14					
AA15					
AA16					
AA19					
AA20					
AA21					
AA22					
AB13					
AB14					
AB15					
AB20					
AB21					
AB22					
AK29					
AK30					
AK31					
AL31					
AM22					
AN26					

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
AN32	GND	G	0	Digital ring ground	TBD
AP26					
AP32					
Digital 2.5 V/3.3 V					
AK3	VDDE2V5	P	2.5	Digital 2.5 V ring power	TBD
AK7					
AL7					
AM7					
AN7					
A26	VDDE3V3	P	3.3	Digital 3.3 V ring power	TBD
B26					
C26					
G33					
G34					
AF31					
AF32					
AF33					
AF34					
AK23					
AK24					
AM9					
AM10					
AM11					
AM12					
AM13					
AM14					
AM15					
AN8					
AP7					
A24	GNDE	G	0	Digital ring ground (common to 2.5 V and 3.5 V rings)	TBD
B24					
C14					
C15					
C16					
C17					
C18					
C19					
C20					
C21					
C22					
C23					

Table 16: Power/ground pins, including decoupling requirements

Pin	Assignment	I/O	Voltage	Description	Decoupling
C24	GNDE	G	0	Digital ring ground (common to 2.5 V and 3.5 V rings)	TBD
D23					
E23					
J3					
K3					
L3					
M3					
N3					
P3					
R3					
R4					
R5					
T3					
V30					
U31					
AD3					
AE3					
AF3					
AG3					
AG32					
AG33					
AG34					
AH3					
AH32					
AJ3					
AJ32					
AK4					
AK5					
AK32					
AL4					
AL5					
AM4					
AM6					
AN4					
AN6					
AP4					
AP6					

a. LMISYS_VDDE may require a 2.6 V +/- 0.1 V supply depending on the DDR part.

b. LMISYS_VDDE may require a 2.6 V +/- 0.1 V supply depending on the DDR part.

8.2 System

Table 17: System pins

Pin	Assignment	I/O	Voltage	Description	Decoupling
C1	SYSACLKIN	I	2.5	First system clock (27 MHz)	system clocks
AP27	SYSBCLKIN	I	2.5	Second system clock (30 MHz) oscillator	
AN27	SYSBCLKOSC	O	2.5		
E27	SYSBCLKINALT	I	3.3	Second system alternate clock (30 MHz)	
E17	SYSCLKOUT	O	3.3	Programmable output clock for debug	
D16	RTCCLKIN	I	3.3	Real time clock	
E18	TMUCLK	I/O	3.3	ST40 timer clock	
D19	NOTRESETIN	I	3.3	System reset	system reset
E19	WDOGRSTOUT	O	3.3	Internal watchdog timer reset	
E20	NOTASEBRK	I/O	3.3	ST40 debugger breakpoint	CPUs debug
D17	TRIGGERIN	I	3.3	ST231 debugger controller in	
D18	TRIGGEROUT	O	3.3	ST231 debugger controller out	
AK25	SYSITRQ[0]	I/O	3.3	Interrupt line	Interrupts
AK26	SYSITRQ[1]				
AK27	SYSITRQ[2]				
AK28	SYSITRQ[3]				
E16	NMI	I	3.3	Nonmaskable interrupt	

Table 18: System pin mapping

Pin	Assignment	I/O	Voltage	Description	PIO assignment
AH34	LONG_TIME_RST_OUT	I	3.3	Long time reset out	PIO1[6]
AA33	SYSCLKOUT0	O	3.3	First system clock	PIO5[2]

See Chapter 15: *Clocks on page 120* for more information.

8.3 JTAG

Table 19: JTAG pins

Pin	Assignment	I/O	Voltage	Description	Comments
D22	TDI	I	3.3	CPUs debug port and TAP data input	no internal pull-up or pull-down resistors
E21	TMS	I	3.3	CPUs debug port and TAP mode select	
D21	TCK	I	3.3	CPUs debug port and TAP clock	
D20	NOTTRST	I	3.3	CPUs debug port and TAP logic reset	
E22	TDO	O	3.3	CPUs debug port and TAP data output	

8.4 Transport, NRSS-A and D1 input interface

Table 20: Transport pins

Pin	Assignment	I/O	Voltage	Description	Comments
AK6	TSIN0BYTECLK	I	3.3	TSIN0 control signals	TSIN 0 parallel stream
AJ5	TSIN0BYTECLKVALID				
AH4	TSIN0ERROR				
AJ4	TSIN0PACKETCLK				
AH5	TSIN0DATA[0]	I	3.3	TSIN0 parallel bit 0 or serial data in	
AG4	TSIN0DATA[1]	I	3.3	TSIN0 parallel data in	
AK1	TSIN0DATA[2]				
AK2	TSIN0DATA[3]				
AJ1	TSIN0DATA[4]				
AJ2	TSIN0DATA[5]				
AH1	TSIN0DATA[6]				
AH2	TSIN0DATA[7]				
AG5	TSIN1BYTECLK	I	3.3	TSIN1 control signals	TSIN 1 parallel stream or D1 input
AF5	TSIN1BYTECLKVALID				
AE4	TSIN1ERROR				
AF4	TSIN1PACKETCLK				
AE5	TSIN1DATA[0]	I	3.3	TSIN0 parallel bit 0 or serial data in	
AD4	TSIN1DATA[1]	I	3.3	TSIN0 parallel data in	
AD5	TSIN1DATA[2]				
AC4	TSIN1DATA[3]				
AC5	TSIN1DATA[4]				
AB4	TSIN1DATA[5]				
AB5	TSIN1DATA[6]				
AA4	TSIN1DATA[7]				
AM3	TSIN2BYTECLK	I/O	3.3	TSIN 2/1394 control signals	TSIN 2 parallel stream
AN3	TSIN2BYTECLKVALID				
AP2	TSIN2ERROR				
AP3	TSIN2PACKETCLK				
AP1	TSIN2DATA[0]	I/O	3.3	TSIN2/1394 parallel bit 0 or serial data in	
AN2	TSIN2DATA[1]	I/O	3.3	TSIN2/1394 parallel data	
AN1	TSIN2DATA[2]				
AM2	TSIN2DATA[3]				
AM1	TSIN2DATA[4]				
AL2	TSIN2DATA[5]				
AL1	TSIN2DATA[6]				
AL3	TSIN2DATA[7]				

Table 21: D1 video input mapping

Pin	Assignment	I/O	Voltage	Description	Comments
AF5	VIDINCLKIN	I	3.3	27 or 54 MHz pixel clock input	TSIN1BYTECLKVALID
AG5	VIDINHSYNCIN	I	3.3	Horizontal synchro input	TSIN1BYTECLK
AF4	VIDINVSYNCIN	I	3.3	Vertical synchro input	TSIN1PACKETCLK
AA4	VIDINDATAIN[7]	I	3.3	Single CCIR656 video or muxed YCbCr SD pixel data in	TSIN1DATA[7]
AB5	VIDINDATAIN[6]				TSIN1DATA[6]
AB4	VIDINDATAIN[5]				TSIN1DATA[5]
AC5	VIDINDATAIN[4]				TSIN1DATA[4]
AC4	VIDINDATAIN[3]				TSIN1DATA[3]
AD5	VIDINDATAIN[2]				TSIN1DATA[2]
AD4	VIDINDATAIN[1]				TSIN1DATA[1]
AE5	VIDINDATAIN[0]				TSIN1DATA[0]

Table 22: NRSS-A interface

Pin	Assignment	I/O	Voltage	Description	PIO assignment
AJ33	NRSSA_DATAIN	I	3.3	NRSS-A interface input	PIO1[5]
AH33	NRSSA_DATAOUT	O	3.3	NRSS-A interface output	PIO1[7]
AJ34	NRSSA_CLKOUT	O	3.3	NRSS-A clock	PIO1[4]

8.5 Display digital output interface

Table 23: Display digital output pins

Pin	Assignment	I/O	Voltage	Description	Comments
M33	VIDDIGOUTHSYNC	O	3.3	Horizontal display reference	
M34	VIDDIGOUTVSYNC	O	3.3	Vertical display reference	
L34	VIDDIGOUTYC[0]	O	3.3	Digital display output	
L33	VIDDIGOUTYC[1]				
K34	VIDDIGOUTYC[2]				
K33	VIDDIGOUTYC[3]				
J34	VIDDIGOUTYC[4]				
J33	VIDDIGOUTYC[5]				
H34	VIDDIGOUTYC[6]				
H33	VIDDIGOUTYC[7]				
U30	VIDDIGOUTYC[8]				
T31	VIDDIGOUTYC[9]				
T30	VIDDIGOUTYC[10]				
R31	VIDDIGOUTYC[11]				
R30	VIDDIGOUTYC[12]				
P31	VIDDIGOUTYC[13]				
P30	VIDDIGOUTYC[14]				
N31	VIDDIGOUTYC[15]				

Table 24: Display digital output pin mapping

Pin	Assignment	I/O	Voltage	Description	PIO assignment
AE33	VTG_MAIN_NTOP_BOT	O	3.3	Video timing generator output	PIO3[2]
AE34	VTG_MAIN_VREF	O	3.3		PIO3[1]
AE32	VTG_MAIN_HREF	O	3.3		PIO3[0]
AE33	VTG_AUX_NTOP_BOT	O	3.3		PIO3[2]
AE34	VTG_AUX_VREF	O	3.3		PIO3[1]
AE32	VTG_AUX_HREF	O	3.3		PIO3[0]
AJ30	DVO_BLANK	O	3.3	DVO output	PIO2[0]

8.6 Display analog output interface

Table 25: Display analog output pins

Pin	Assignment	I/O	Voltage	Description	Comments
D34	VIDANAR0OUT	O	2.5	Analog display0 red output	Connect an external 140 Ω 1% resistor to analog ground
F34	VIDANAG0OUT	O	2.5	Analog display0 green output	
E34	VIDANAB0OUT	O	2.5	Analog display0 blue output	
D33	VIDANAIDUMPR0	O	2.5	Analog display0 current return red output	Connect to analog ground
F33	VIDANAIDUMPG0	O	2.5	Analog display0 current return green output	
E33	VIDANAIDUMPB0	O	2.5	Analog display0 current return blue output	
A34	VIDANAC1OUT	O	2.5	Analog display1 chrominance output	Connect an external 140 Ω 1% resistor to analog ground
C34	VIDANACV1OUT	O	2.5	Analog display1 CVBS output	
B34	VIDANAY1OUT	O	2.5	Analog display1 luminance output	
A33	VIDANAIDUMPC1	O	2.5	Analog display0 current return chrominance output	Connect to analog ground
C33	VIDANAIDUMPCV1	O	2.5	Analog display0 current return CVBS output	
B33	VIDANAIDUMPY1	O	2.5	Analog display0 current return luminance output	
A31	VIDANAREXT[0]	A	-	DAC external resistor interface	Connect an external 7.72 kΩ 1% resistor to VIDANAGNDAREXT
A32	VIDANAREXT[1]				
B31	VIDANAGNDAREXT[0]	A	-	DAC external resistor ground interface	
B32	VIDANAGNDAREXT[1]				

8.7 HDMI interface

Table 26: HDMI pins

Pin	Assignment	I/O	Voltage	Description	Comments
U33	TMDSTXCP	O	TMDS	TMDS Control plus	
U34	TMDSTXCN	O	TMDS	TMDS Control minus	
T33	TMDSTX0P	O	TMDS	TMDS Data0 plus	
T34	TMDSTX0N	O	TMDS	TMDS Data0l minus	
R33	TMDSTX1P	O	TMDS	TMDS Data1 plus	
R34	TMDSTX1N	O	TMDS	TMDS Data1 minus	
P33	TMDSTX2P	O	TMDS	TMDS Data2 plus	
P34	TMDSTX2N	O	TMDS	TMDS Data2 minus	
T32	TMDSREF	A	-	TMDS voltage reference	

8.8 Audio digital interface

Table 27: Audio digital pins

Pin	Assignment	I/O	Voltage	Description	Comments
D29	AUDDIGDATAIN	I	3.3	Serial audio input	PCM input
D28	AUDDIGDSTRBIN	I	3.3	Strobe for serial audio input	
E28	AUDDIGDLRCLKIN	I	3.3	Left/right channel select input	
A25	AUDPCMOUT0	O	3.3	PCM data0[a]	Audio digital output
B25	AUDPCMOUT1	O	3.3	PCM data1[a]	
C25	AUDPCMOUT2	O	3.3	PCM data2[a]	
D25	AUDPCMOUT3	O	3.3	PCM data3[a]	
E25	AUDPCMOUT4	O	3.3	PCM data4[a]	
D26	AUDPCMCLKOUT	I/O	3.3	PCM clock in/out (usually 256 Fs or 384 Fs)	
E26	AUDLRCLKOUT	O	3.3	left/right channel select output	
E24	AUDSCLKOUT	O	3.3	1st audio DAC clock output (32 or 64 Fs)	
D24	AUDSPDIFOUT	O	3.3	S/PDIF digital output	

a. Data content is s/w programmable.

8.9 Audio analog interface

Table 28: Audio analog pins

Pin	Assignment	I/O	Voltage	Description	Comments
A27	AUDANAPLEFTOUT	O	2.5	DAC left channel positive differential output	
B27	AUDANAMLEFTOUT	O	2.5	DAC left channel negative differential out	
A28	AUDANAPRIGHTOUT	O	2.5	DAC right channel positive differential out	
B28	AUDANAMRIGHTOUT	O	2.5	DAC right channel negative differential out	
C27	AUDANAIREF	A	-	DAC output reference current	Connect an external 575 Ω 1% resistor to AUD_GNDA
C28	AUDANAVBGFIL	A	-	DAC filtered input reference voltage	Connect an external 10 µF capacitance to AUD_GNDA

8.10 Programmable inputs/outputs

Table 29: PIO pins

Pin	Assignment	I/O	Voltage	Description	Comments
AM32	PIO0[0]	I/O	3.3	Programmable input/output	
AP33	PIO0[1]				
AN33	PIO0[2]				
AP34	PIO0[3]				
AN34	PIO0[4]				
AM33	PIO0[5]				
AM34	PIO0[6]				
AL32	PIO0[7]				
AL34	PIO1[0]	I/O	3.3	Programmable input/output	
AL33	PIO1[1]				
AK34	PIO1[2]				
AK33	PIO1[3]				
AJ34	PIO1[4]				
AJ33	PIO1[5]				
AH34	PIO1[6]				
AH33	PIO1[7]				
AJ30	PIO2[0]	I/O	3.3	Programmable input/output	
AJ31	PIO2[1]				
AH30	PIO2[2]				
AH31	PIO2[3]				
AG30	PIO2[4]				
AG31	PIO2[5]				
AE31	PIO2[6]				
AE30	PIO2[7]				
AE32	PIO3[0]	I/O	3.3	Programmable input/output	
AE34	PIO3[1]				
AE33	PIO3[2]				
AD34	PIO3[3]				
AD33	PIO3[4]				
AC34	PIO3[5]				
AC33	PIO3[6]				
AB34	PIO3[7]				

Table 29: PIO pins

Pin	Assignment	I/O	Voltage	Description	Comments
AD32	PIO4[0]	I/O	3.3	Programmable input/output	
AD30	PIO4[1]				
AD31	PIO4[2]				
AC30	PIO4[3]				
AC31	PIO4[4]				
AB30	PIO4[5]				
AB31	PIO4[6]				
AA30	PIO4[7]				
AB33	PIO5[0]	I/O	3.3	Programmable input/output	
AA34	PIO5[1]				
AA33	PIO5[2]				
Y34	PIO5[3]				
Y33	PIO5[4]				
AA31	PIO5[5]				
Y30	PIO5[6]				
Y31	PIO5[7]				

8.11 External memory interface (EMI)

Table 30: EMI pins

Pin	Assignment	I/O	Voltage	Description	Comments
AK9	NOTEMICSA	O	3.3	Peripheral chip select A	
AL8	NOTEMICSB	O	3.3	Peripheral chip select B	
AM8	NOTEMICSC	O	3.3	Peripheral chip select C	
AP8	NOTEMICSD	O	3.3	Peripheral chip select D	
AK8	NOTEMICSE	O	3.3	Peripheral chip select E	
AP11	NOTEMIBE[0]	O	3.3	External device databus byte enable	
AN11	NOTEMIBE[1]				
AP10	NOTEMIOE	O	3.3	External device output enable	
AP22	NOTEMILBA	O	3.3	Flash device load burst address	
AP21	NOTEMIBAA	O	3.3	Flash burst address advanced	
AK22	EMITREADYORWAIT	I	3.3	External memory device target ready indicator	
AN10	EMIRDNOTWR	O	3.3	External read/write access indicator. Common to all devices	

Table 30: EMI pins

Pin	Assignment	I/O	Voltage	Description	Comments
AP20	EMIDATA[0]	I/O	3.3	External common data bus	
AP19	EMIDATA[1]				
AP18	EMIDATA[2]				
AP17	EMIDATA[3]				
AP15	EMIDATA[4]				
AP14	EMIDATA[5]				
AP13	EMIDATA[6]				
AP12	EMIDATA[7]				
AN20	EMIDATA[8]				
AN19	EMIDATA[9]				
AN18	EMIDATA[10]				
AN17	EMIDATA[11]				
AN15	EMIDATA[12]				
AN14	EMIDATA[13				
AN13	EMIDATA[14]				
AN12	EMIDATA[15]				
AL9	EMIADDR[1]	O	3.3	External common address bus	23-bit address[a]
AK10	EMIADDR[2]				
AL10	EMIADDR[3]				
AK11	EMIADDR[4]				
AL11	EMIADDR[5]				
AK12	EMIADDR[6]				
AL12	EMIADDR[7]				
AK13	EMIADDR[8]				
AL13	EMIADDR[9]				
AK14	EMIADDR[10]				
AL14	EMIADDR[11]				
AK15	EMIADDR[12]				
AL15	EMIADDR[13]				
AK17	EMIADDR[14]				
AL17	EMIADDR[15]				
AK18	EMIADDR[16]				
AL18	EMIADDR[17]				
AK19	EMIADDR[18]				
AL19	EMIADDR[19]				

Table 30: EMI pins

Pin	Assignment	I/O	Voltage	Description	Comments
AK20	EMIADDR[20]	O	3.3	External common address bus	23-bit address[a]
AL20	EMIADDR[21]				
AK21	EMIADDR[22]				
AL21	EMIADDR[23]				
AN21	EMIFLASHCLK	O	3.3	Flash clock	
AN22	EMIDMAREQ[0]	I	3.3	DMA request	
AL22	EMIDMAREQ[1]				
AM20	EMIBUSREQ	I/O	3.3	Bus access request	for master/ slave configuration
AM21	EMIBUSGNT	I/O	3.3	Bus access grant	

a. No pull-up. External resistors define mode at boot.

8.12 System local memory interface (LMISYS)

Table 31: System LMI pins

Pin	Assignment	I/O	Voltage	Description	Comments
U1	LMISYSCLK	O	2.5	Clock to DDR	
U2	NOTLMISYSCLK	O	2.5	Inverted clock to DDR	
L4	NOTLMISYSCS[0]	O	2.5	Chip select	
M5	NOTLMISYSCS[1]				
K5	NOTLMISYSRAS	O	2.5	Row address strobe	
J4	NOTLMISYSCAS	O	2.5	Column address strobe	
J5	NOTLMISYSWE	O	2.5	Write enable	
M4	LMISYSADD[0]	O	2.5	Address	
N5	LMISYSADD[1]				
N4	LMISYSADD[2]				
P5	LMISYSADD[3]				
U4	LMISYSADD[4]				
V5	LMISYSADD[5]				
V4	LMISYSADD[6]				
W5	LMISYSADD[7]				
W4	LMISYSADD[8]				
U5	LMISYSADD[9]				
P4	LMISYSADD[10]				
T5	LMISYSADD[11]				
T4	LMISYSADD[12]				
K4	LMISYSBKSEL[0]	O	2.5	Bank select	
L5	LMISYSBKSEL[1]				
V1	LMISYSDATA[0]	I/O	2.5	Bidirectional data bus	
V2	LMISYSDATA[1]				
W1	LMISYSDATA[2]				
W2	LMISYSDATA[3]				
Y1	LMISYSDATA[4]				
Y2	LMISYSDATA[5]				
AA1	LMISYSDATA[6]				
AA2	LMISYSDATA[7]				
E2	LMISYSDATA[8]				
E1	LMISYSDATA[9]				
F2	LMISYSDATA[10]				
F1	LMISYSDATA[11]				
G2	LMISYSDATA[12]				
G1	LMISYSDATA[13]				
H2	LMISYSDATA[14]				
H1	LMISYSDATA[15]				
AD1	LMISYSDATA[16]				
AD2	LMISYSDATA[17]				

Table 31: System LMI pins

Pin	Assignment	I/O	Voltage	Description	Comments
AE1	LMISYSDATA[18]	I/O	2.5	Bidirectional data bus	
AE2	LMISYSDATA[19]				
AF1	LMISYSDATA[20]				
AF2	LMISYSDATA[21]				
AG1	LMISYSDATA[22]				
AG2	LMISYSDATA[23]				
L2	LMISYSDATA[24]				
L1	LMISYSDATA[25]				
M2	LMISYSDATA[26]				
M1	LMISYSDATA[27]				
N2	LMISYSDATA[28]				
N1	LMISYSDATA[29]				
P2	LMISYSDATA[30]				
P1	LMISYSDATA[31]				
AB2	LMISYSDATAMASK[0]	O	2.5	Data write mask	
J1	LMISYSDATAMASK[1]	O	2.5	Data write mask	
AC1	LMISYSDATAMASK[2]	O	2.5	Data write mask	
K2	LMISYSDATAMASK[3]	O	2.5	Data write mask	
AB1	LMISYSDATASTROBE[0]	I/O	2.5	Write/read data strobe	
J2	LMISYSDATASTROBE[1]				
AC2	LMISYSDATASTROBE[2]				
K1	LMISYSDATASTROBE[3]				
H5	LMISYSVREF	I	2.5	SSTL reference voltage	
Y5	LMISYSCLKEN	O	2.5	Memory clock enable	
R1	LMISYSREF	A	-	LMISYS ring compensation	a
R2	LMISYSGNDBCOMP	A	-	LMISYS ring compensation ground	

a. External 120 kΩ resistor to LMISYS_GNDBCOMP

8.13 Video local memory interface (LMIVID)

Table 32: Video local memory pins

Pin	Assignment	I/O	Voltage	Description	Comments
A13	LMIVIDCLK	O	2.5	Clock to DDR	
B13	NOTLMIVIDCLK	O	2.5	Inverted clock to DDR	
D11	NOTLMIVIDCS[0]	O	2.5	Chip select	
E11	NOTLMIVIDCS[1]				
E14	NOTLMIVIDRAS	O	2.5	Row address strobe	
D15	NOTLMIVIDCAS	O	2.5	Column address strobe	
E15	NOTLMIVIDWE	O	2.5	Write enable	
E10	LMIVIDADD[0]	O	2.5	Address	
D10	LMIVIDADD[1]				
E9	LMIVIDADD[2]				
D9	LMIVIDADD[3]				
D6	LMIVIDADD[4]				
E6	LMIVIDADD[5]				
D5	LMIVIDADD[6]				
E5	LMIVIDADD[7]				
D4	LMIVIDADD[8]				
E7	LMIVIDADD[9]				
D8	LMIVIDADD[10]				
E8	LMIVIDADD[11]				
D7	LMIVIDADD[12]				
D14	LMIVIDBKSEL[0]	O	2.5	Bank select	
E13	LMIVIDBKSEL[1]				
A14	LMIVIDDATA[0]	I/O	2.5	Bidirectional data bus	
B14	LMIVIDDATA[1]				
A15	LMIVIDDATA[2]				
B15	LMIVIDDATA[3]				
A16	LMIVIDDATA[4]				
B16	LMIVIDDATA[5]				
A17	LMIVIDDATA[6]				
B17	LMIVIDDATA[7]				
A1	LMIVIDDATA[8]				
B2	LMIVIDDATA[9]				
A2	LMIVIDDATA[10]				
B3	LMIVIDDATA[11]				
A3	LMIVIDDATA[12]				
B4	LMIVIDDATA[13]				
A4	LMIVIDDATA[14]				
B5	LMIVIDDATA[15]				
A20	LMIVIDDATA[16]				
B20	LMIVIDDATA[17]				

Table 32: Video local memory pins

Pin	Assignment	I/O	Voltage	Description	Comments
A21	LMIVIDDATA[18]	I/O	2.5	Bidirectional data bus	
B21	LMIVIDDATA[19]				
A22	LMIVIDDATA[20]				
B22	LMIVIDDATA[21]				
A23	LMIVIDDATA[22]				
B23	LMIVIDDATA[23]				
A7	LMIVIDDATA[24]				
B8	LMIVIDDATA[25]				
A8	LMIVIDDATA[26]				
B9	LMIVIDDATA[27]				
A9	LMIVIDDATA[28]				
B10	LMIVIDDATA[29]				
A10	LMIVIDDATA[30]				
A11	LMIVIDDATA[31]				
B18	LMIVIDDATAMASK[0]	O	2.5	Data write mask	
B6	LMIVIDDATAMASK[1]				
A19	LMIVIDDATAMASK[2]				
A6	LMIVIDDATAMASK[3]				
A18	LMIVIDDATASTROBE[0]	I/O	2.5	Write/read data strobe	
A5	LMIVIDDATASTROBE[1]				
B19	LMIVIDDATASTROBE[2]				
B7	LMIVIDDATASTROBE[3]				
C11	LMIVIDVREF	I	2.5	SSTL reference voltage	
D13	LMIVIDCLKEN	O	2.5	Memory clock enable	
B12	LMIVIDREF	A	-	LMIVID ring compensation	a
B11	LMIVIDGNDBCOMP	A	-	LMIVID ring compensation ground	

a. External 120 kΩ resistor to LMIVID_GNDBCOMP

8.14 USB 2.0 interface

Table 33: USB pins

Pin	Assignment	I/O	Voltage	Description	Comments
AN25	USBDP	I/O	2.5	USB receive plus	
AP25	USBDM	I/O	2.5	USB receive minus	
AM25	USBREF	I	2.5	USB voltage reference	a

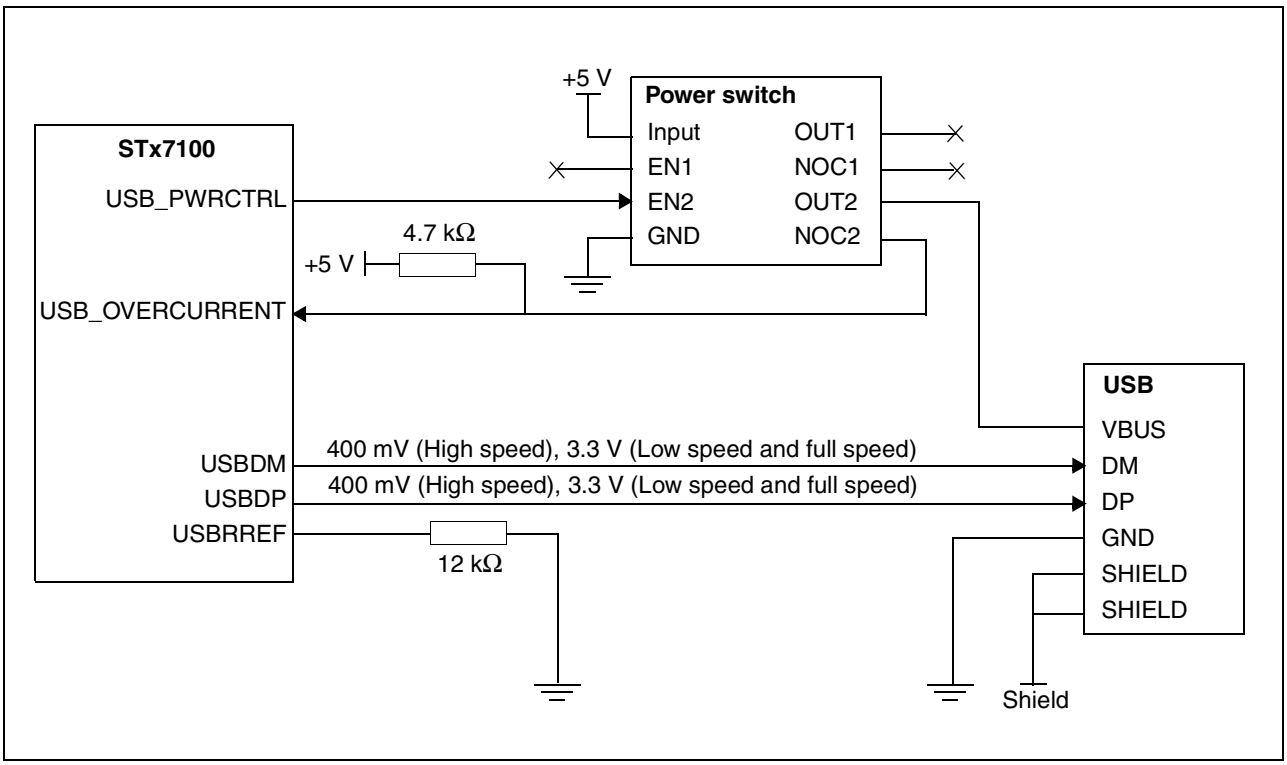
a. Connect an external 12 kΩ 1% resistor to USBVSS2V5

Table 34: USB pin mapping

Pin	Assignment	I/O	Voltage	Description	PIO assignment
Y30	USB2_PRT_OVRCUR	I	3.3	USB 2.0 interface	PIO5[6]
Y31	USB2_PRT_PWR	O	3.3		PIO5[7]

Figure 18 shows an example application circuit.

Figure 18: USB 2.0 application circuit



8.15 SATA-I interface

Table 35: SATA pins

Pin	Assignment	I/O	Voltage	Description	Comments
AN30	SATATXP	O	2.5	SATA transmit plus	
AP30	SATATXN	O	2.5	SATA transmit minus	
AN31	SATARXP	I	2.5	SATA receive plus	
AP31	SATARXN	I	2.5	SATA receive minus	
AM30	SATAREF	I	2.5	SATA voltage reference	a

a. External 475 Ω 1% resistor to SATA_VSSREF

The SATA interface has some additional design rules:

- The SATA connector must be as close as possible to the STx7100.
- There must be a specific ground plane.
- There must be a specific linear regulator for the SATA power supply (no DC/DC).
- The interface only supports internal SATA disks.
- TXP/TXN must have a specific differential impedance of 90 Ω, 45 Ω to ground.
- PCB layout must be compliant with PCI-Express layout rules.

8.16 Peripherals

8.16.1 DAA

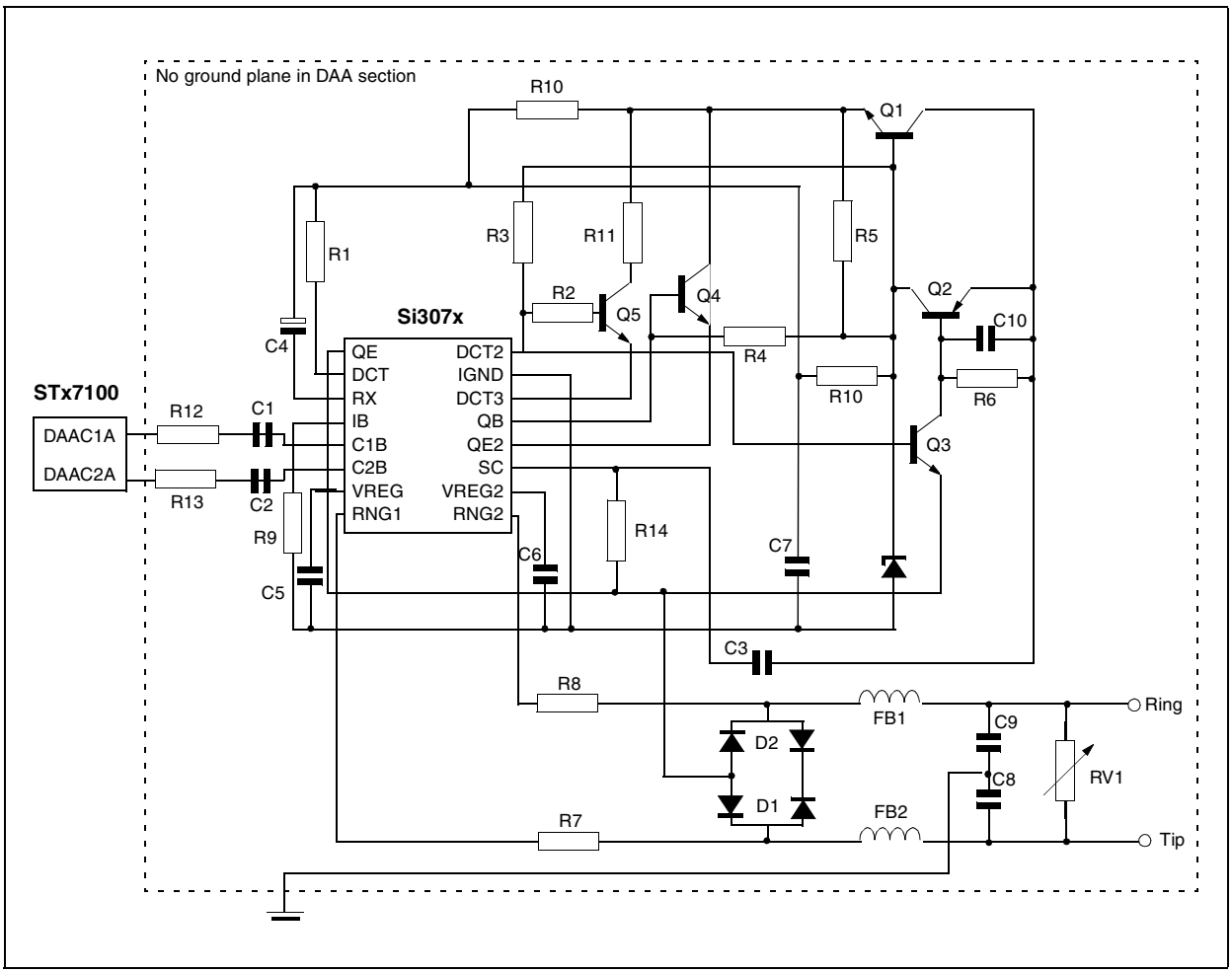
Table 36: DAA pins

Pin	Assignment	I/O	Volt age	Description	Comments
AN5	DAA_C1A	I/O	3.3	DAA differential data[a]	
AP5	DAA_C2A	I/O	3.3	DAA differential data[b]	

a. ISO-Link capacitors C1 and C2, (33 pF) should be as close to the line-side device as possible.

b. After satisfying the above, C1 and C2 should be as close to the embedded system-side DAA module as possible and no further than 6 inches away.

Figure 19: Example DAA application circuit



8.16.2 Asynchronous serial controller (ASC)

Table 37: ASC pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
ASC 0					
AH34	UART1_NOTOE	O	3.3	ASC 0	PIO1[6]
ASC 1					
AL34	UART1_TXD	O	3.3	ASC 1 transmit signal	PIO1[0]
AL33	UART1_RXD	I	3.3	ASC 1 receive signal	PIO1[1]
ASC 2					
AD31	UART2_RXD	I	3.3	ASC 2 receive signal	PIO4[2]
AD34	UART2_TXD	O	3.3	ASC 2 transmit signal	PIO4[3]
AD33	UART2_CTS	I	3.3	ASC 2 clear to send signal	PIO4[4]
AC34	UART2_RTS	O	3.3	ASC 2 request to send signal	PIO4[5]
ASC 3					
AB33	UART3_TXD	O	3.3	ASC 3 transmit signal	PIO5[0]
AA34	UART3_RXD	I	3.3	ASC 3 receive signal	PIO5[1]
AA33	UART3_CTS	I	3.3	ASC 3 clear to send signal	PIO5[2]
Y34	UART3_RTS	O	3.3	ASC 3 request to send signal	PIO5[3]
SCIF interface					
AE33	SCI_RXD	I	3.3	SCIF receive signal	PIO4[2]
AD34	SCI_TXD	O	3.3	SCIF transmit signal	PIO4[3]
AD33	SCI_CTS	I/O	3.3	SCIF clear to send signal	PIO4[4]
AC34	SCI_RTS	I/O	3.3	SCIF request to send signal	PIO4[5]
AC33	SCI_SCK	I/O	3.3	SCIF clock	PIO4[6]

8.16.3 Infrared transmitter/receiver

Table 38: Infrared transmitter/receiver pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
AD34	IRB_IR_IN	I	3.3	IR data input	PIO3[3]
AD33	IRB_UHF_IN	I	3.3	UHF data input	PIO3[4]
AC34	IRB_IR_DATAOUT	O	3.3	IR data output	PIO3[5]
AC33	IRB_IR_DATAOUT_OD	O	3.3		PIO3[6]

8.16.4 Modem analog front-end (MAFE) interface

Table 39: MAFE pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
AH31	MAFE_HC1	O	3.3	Indicates a control/status exchange	PIO2[3]
AG30	MAFE_DOUT	O	3.3	Line for serially transmitting samples	PIO2[4]
AG31	MAFE_DIN	I	3.3	Line for serially receiving samples	PIO2[5]
AE31	MAFE_FS	I	3.3	Start of a sampling period latched on falling edges of SCLK	PIO2[6]
AE30	MAFE_SCLK	I	3.3	Modem system clock	PIO2[7]

8.16.5 Pulse width modulator (PWM)

Table 40: PWM pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
PWM 0					
AC33	PWM_OUT0	O	3.3	PWM 0	PIO4[6]
AD33	PWM_COMPAREOUT0	O	3.3		PIO5[4]
AC34	PWM_CAPTUREIN0	I	3.3		PIO5[5]
PWM 1					
AB34	PWM_OUT1	O	3.3	PWM 1	PIO4[7]
Y30	PWM_COMPAREOUT1	O	3.3		PIO5[6]
Y31	PWM_CAPTUREIN1	I	3.3		PIO5[7]

8.16.6 Smartcard

Table 41: Smartcard pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
Smartcard 0					
AM33	SC0_SETVCC	O	3.3	Smartcard set voltage level	PIO0[5]
AM32	SC0_DIR	O	3.3		PIO0[6]
AL32	SC0_DETECT	I	3.3	Smartcard detection	PIO0[7]
AM32	SC0_DATAOUT	O	3.3	Serial data output	PIO0[0]
AN33	SC0_EXTCLKIN	I	3.3	External clock	PIO0[2]
AP33	SC0_DATAIN	I	3.3	Serial data input	PIO0[1]
AP34	SC0CG_CLK	O	3.3	Clock for smartcard from system clock	PIO0[3]
AP34	DSSMCD_CLK	O	3.3	Clock for smartcard from smartcard FS	PIO0[3]
Smartcard 1					
AH34	SC1_DIR	O	3.3		PIO1[6]
AK33	SC1CG_CLK	O	3.3	Clock for smartcard from system clock	PIO1[3]
AK33	DSSMCD_CLK	O	3.3	Clock for smartcard from smartcard FS	PIO1[3]
AK34	SC1_EXTCLKIN	I	3.3	External clock	PIO1[2]
AL33	SC1_DATAIN	I	3.3	Serial data input	PIO1[1]
AL34	SC1_DATAOUT	O	3.3	Serial data output	PIO1[0]

8.16.7 Synchronous serial controller (SSC)

Table 42: SSC pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
SSC 0					
AJ30	SSC0_SCL	I/O	3.3	Serial clock	PIO2[0]
AJ31	SSC0_MTSR	I/O	3.3	Serial data for SSC 0: master transmit, slave receive	PIO2[1]
AH30	SSC0_MRST	I/O	3.3	Serial data for SSC 0: master receive, slave transmit	PIO2[2]
SSC 1					
AE32	SSC1_SCL	I/O	3.3	Serial clock	PIO3[0]
AE34	SSC1_MTSR	I/O	3.3	Serial data for SSC 1: master transmit, slave receive	PIO3[1]
AE33	SSC1_MRST	I/O	3.3	Serial data for SSC 1: master receive, slave transmit	PIO3[2]
SSC 2					
AD32	SSC2_SCL	I/O	3.3	Serial clock	PIO4[0]
AD30	SSC2_MTSR	I/O	3.3	Serial data for SSC 2: master transmit, slave receive	PIO4[1]

8.16.8 DiSEqC 2.0

Table 43: DiSEqC 2.0™ pins

Pin	Assignment	I/O	Voltage	Description	PIO assignment
Y33	DISEQC_RX_DATAIN	I	3.3	DiSEqC interface input	PIO5[4]
AA31	DISEQC_TX_DATAOUT	O	3.3	DiSEqC interface output	PIO5[5]

9 Reset configuration (mode pins)

The STx7100 has static configuration pins (called mode pins). These pins are shared with EMI address pins and are only valid during the power-on-reset sequence.

External pull-up or pull-down resistors must be applied to setup the desired configuration.

The mode pins are strobe (captured) during the reset phase and are made available to the system to define operating modes (clockgen startup configuration for example).

The captured values are then recorded in the system configuration register SYS_STA4. Table 44 shows the mapping of the mode pins.

Table 44: Mode pins mapping

Mode pin	EMI pin	Purpose	Used by	Description
MODE[1:0]	EMIADDR[2:1]	PLL0 startup configuration	Clockgen A	See Section 15.2.3: *Startup configuration on page 124*
MODE[3:2]	EMIADDR[4:3]	PLL1 startup configuration		
MODE[7:4]	EMIADDR[8:5]	Reserved	-	Must be ‘00’
MODE[9:8]	EMIADDR[10:9]	EMI banks port size at boot	EMI subsystem	11: 8 bits 10: 16 bits Others: Reserved
MODE[10]	PCMOUT4	STx7100 master/slave mode	EMI subsystem	0: Master 1: Slave
MODE[12:11]	EMIADDR[13:12]	Reserved	-	Must be ‘00’
MODE[13]	EMIADDR[14]	Long resetout mode	Reset generator	0: 1.2 ms 1: 200 ms
MODE[14]	EMIADDR[15]	Reserved	-	Must be ‘0’

See also

- Section 15.2.3: *Startup configuration on page 124*, which shows default clock frequencies,
- Chapter 21: *System configuration registers on page 171*, which includes some mode pins registers.

10 Electrical specifications

Other electrical specifications TBD.

Table 45: Absolute maximum ratings

Symbol	Parameter	Min	Max	Units
Tstg	Storage temperature	- 60	150	degrees C
VCCA	Analog power supply		TBD	V
VDDD	Digital core power supply		TBD	V

10.1 Video DACs

Table 46: DAC specifications

Symbol	Parameter	Min	Typ	Max	Units
VDD	DC supply voltage		2.5		V
Tj	Operating temperature (junction)				C
PD	Power dissipation				mW
PDpd	Power dissipation during powerdown mode				µW
ILE	Integral non-linearity				LSB
DLE	Differential linearity				LSB
THD	Total harmonic distortion @100 kHz[a]				dB
SNR	Signal to noise ratio[a]				dB

a. R_{load} = 250 Ω, Rset = 649 Ω, triple DAC only

Table 47: Operating conditions

Symbol	Parameter	Min	Typ	Max	Units
BDW	Output 3 db bandwidth	30			MHz
F_CLK	Clock speed			160	MHz
VCCA	Analog supply	2.25	2.5	2.75	V
VDDD	Digital supply	0.9	1.0	1.1	V
Top	Operating junction temperature	-40	55	125	degrees C

Table 48: Static electrical performance

Symbol	Parameter	Min	Typ	Max	Units
Nb	DAC resolution		10		bits
PonAnalog	Power consumption analog/ active		108		mW
PonDigital	Power consumption digital/ active		6		mW
PHZ	Power consumption / HZ mode[a]		2.5		mW
Poff	Power consumption / Off mode[b]		2		mW
INL	Integral non linearity			+/- 1.0	LSB
DNL	Differential non linearity			+/- 0.5	LSB
DAC to DAC matching	DAC to DAC matching (3)			+/- 3	%
Compliance	Output compliance Delta I / Iout 0 V < Vout < 1.4 V			1	µA
I_{OUT} **Rref = 7.73 kΩ**	DAC output current	0 (code min)		10.0 (code max)	mA
Full scale gain error				+ / - 7	%
Rout	DAC output resistance @ dc	100			kΩ

a. Independent of clock activity.

b. Transistor off-state leakage only

Table 49: Dynamic electrical performance

Symbol	Parameter	Min	Typ	Max	Units
THD	Total harmonic distortion Fin = 1 MHz, F_clk =160 MHz	45	50		dB
SFDR	Spurious free dynamic range Fin=1 MHz, Fclk=160MHz Output full scale	47	52		dB
PSRR	**Power supply rejection ratio** @ 1 Hz (full scale)	30	60		dB
	Power supply rejection ratio @1 MHz (full scale)	22	25		dB

11 Timing specifications

11.1 Audio input AC specifications

The audio inputs are characterized with respect to the AUDDIGSTRBIN input clock (rising edge). They are clocked on the rising edge of this clock.

Figure 20: Audio inputs AC waveforms

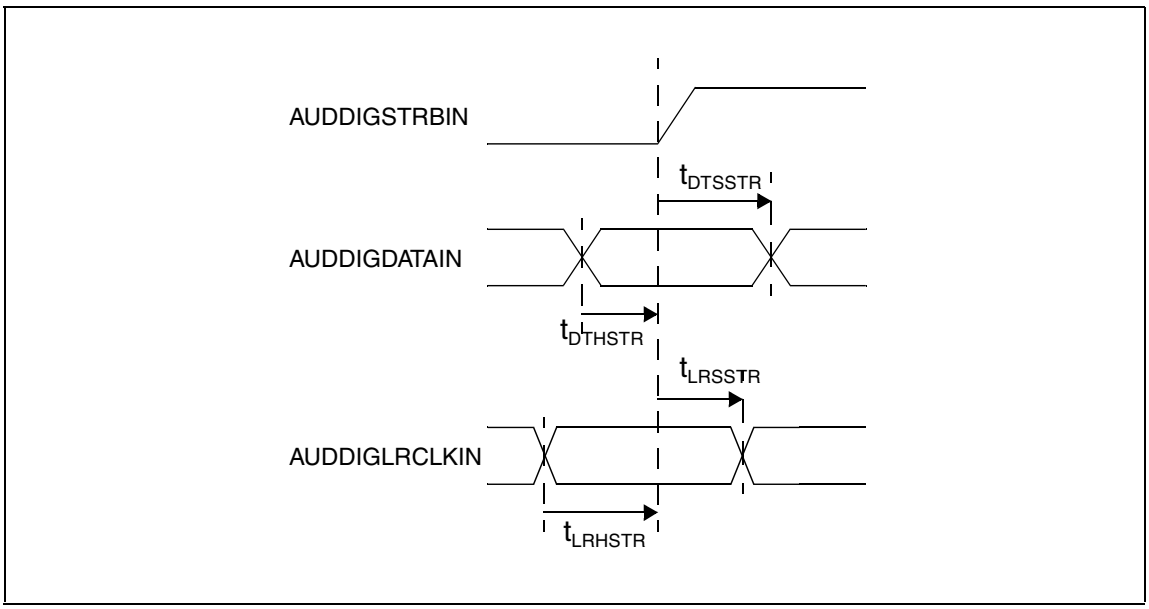


Table 50: Audio input port timings

Symbol	Parameter	Min	Max	Units
t_{DTSSTR}	AUDDIGDATAIN setup to AUDDIGSTRBIN rising edge	2.5		ns
t_{DTHSTR}	AUDDIGDATAIN hold to AUDDIGSTRBIN rising edge	2		ns
t_{LRSSTR}	AUDDIGLRCLKIN setup to AUDDIGSTRBIN rising edge	2.5		ns
t_{LRHSTR}	AUDDIGLRCLKIN hold to AUDDIGSTRBIN rising edge	2		ns

11.2 Audio output AC specifications

Output clock: AUDPCMCLKOUT

Outputs: AUDPCMOUT0, AUDPCMOUT1, AUDPCMOUT2, AUDPCMOUT3, AUDPCMOUT4, AUDLRCLKOUT.

Figure 21: Audio outputs waveforms

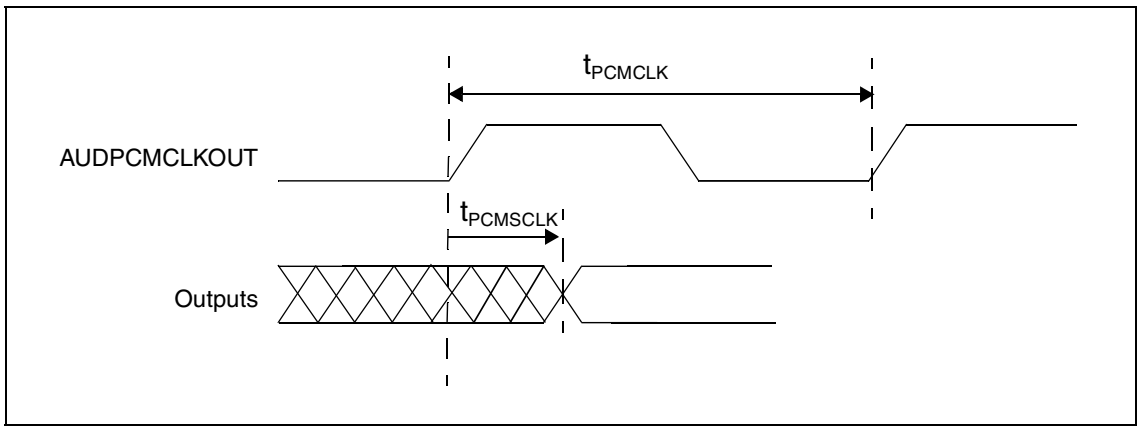


Table 51: Audio output ports timings

Symbol	Parameter	Min	Max	Units
t_{PCMCLK}	AUDPCMCLKOUT clock period	83		
$t_{PCMSCLK}$	Output delay to AUDPCMCLKOUT		4.5	ns

11.3 Video output AC specification

Output clock: SYSCLKOUT (rising edge); when video clock selected.

Figure 22: Video output waveforms

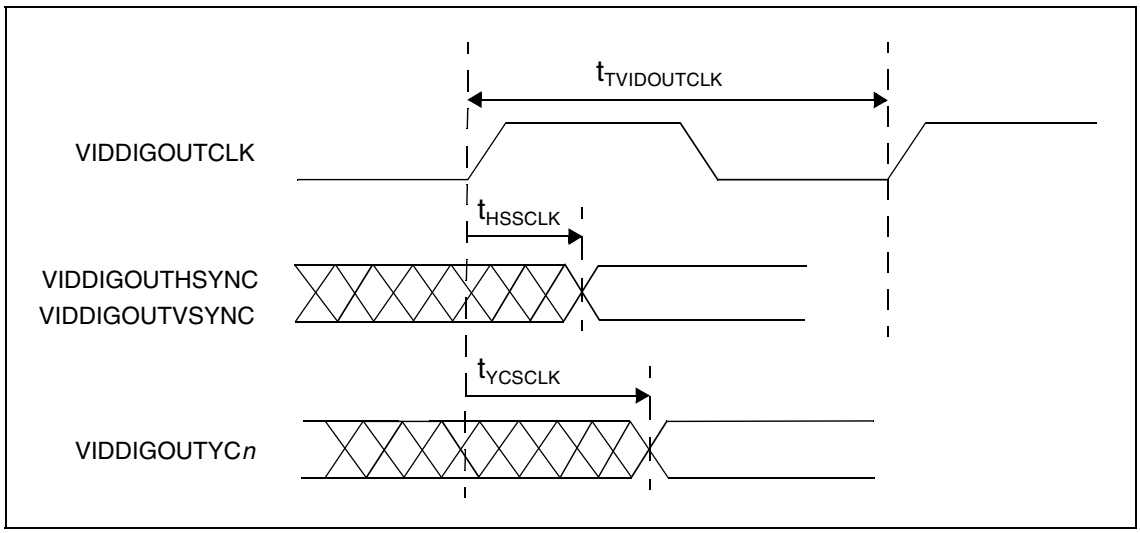


Table 52: Video output AC port timings

Symbol	Parameter	Min	Max	Units
$t_{VIDOUTCLK}$	SYSCLKOUT period	13.46		ns
t_{HSSCLK}	VIDDIGOUTHSYNC output delay to SYSCLKOUT		1	ns
t_{VCSCLK}	VIDDIGOUTYC*n* output delay to SYSCLKOUT		1	ns

11.4 Transport stream input AC specification

Input clocks: TSIN0BYTECLK, TSIN1BYTECLK, TSIN2BYTECLK (rising edge)

Inputs: TSIN0BYTECLKVALID, TSIN0ERROR, TSIN0PACKETCLK, TSIN0DATA[*], TSIN1BYTECLKVALID, TSIN1ERROR, TSIN1PACKETCLK, TSIN1DATA[*], TSIN2BYTECLKVALID, TSIN2ERROR, TSIN2PACKETCLK, TSIN2DATA[*]

Figure 23: Transport stream input waveforms

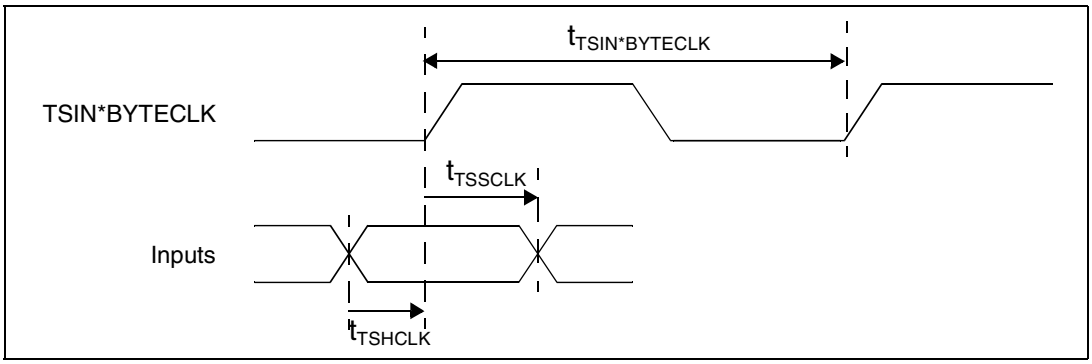


Table 53: Transport input ports AC timings

Symbol	Parameter	Min	Max	Units
Input clock	TSIN*BYTECLK period	6.66		ns
t_{DTSCLK}	inputs setup to TSIN*BYTECLK rising edge	4.5		ns
t_{DTHCLK}	inputs hold to TSIN*BYTECLK rising edge	1.5		ns

11.5 DVP inputs

DVP inputs are muxed with transport inputs and match the same timing specifications.

11.6 Transport stream output AC specification

Output clock: TSIN2BYTECLK

Outputs: TSIN2BYTECLKVALID, TSIN2ERROR, TSIN2PACKETCLK, TSIN2DATA*n*

Figure 24: Transport stream output waveforms

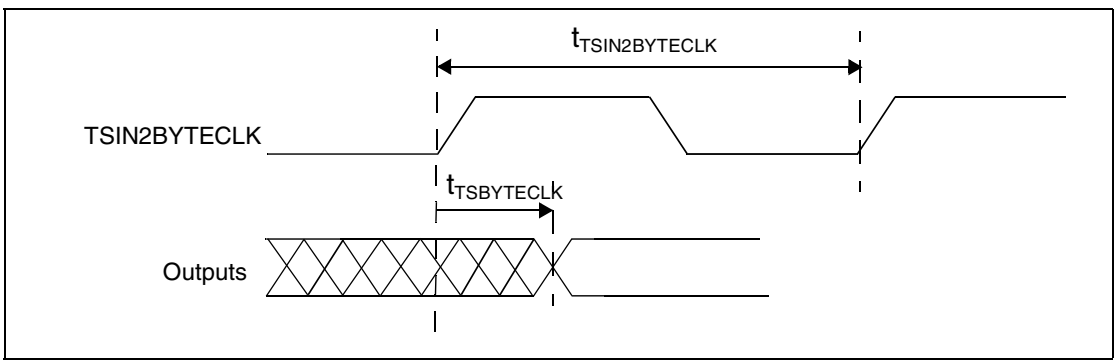


Table 54: Transport stream output timings

Symbol	Parameter	Min	Max	Units
$t_{TSIN2BYTECLK}$	TSIN2BYTECLK clock period	6.66		
$t_{TSBYTECLK}$	Output delay to AUDPCMCLKOUT		4.5	ns

11.7 JTAG interfaces AC specification

Input clocks: TCK (rising edge)

Inputs: TDI, TMS

Figure 25: JTAG interfaces waveforms

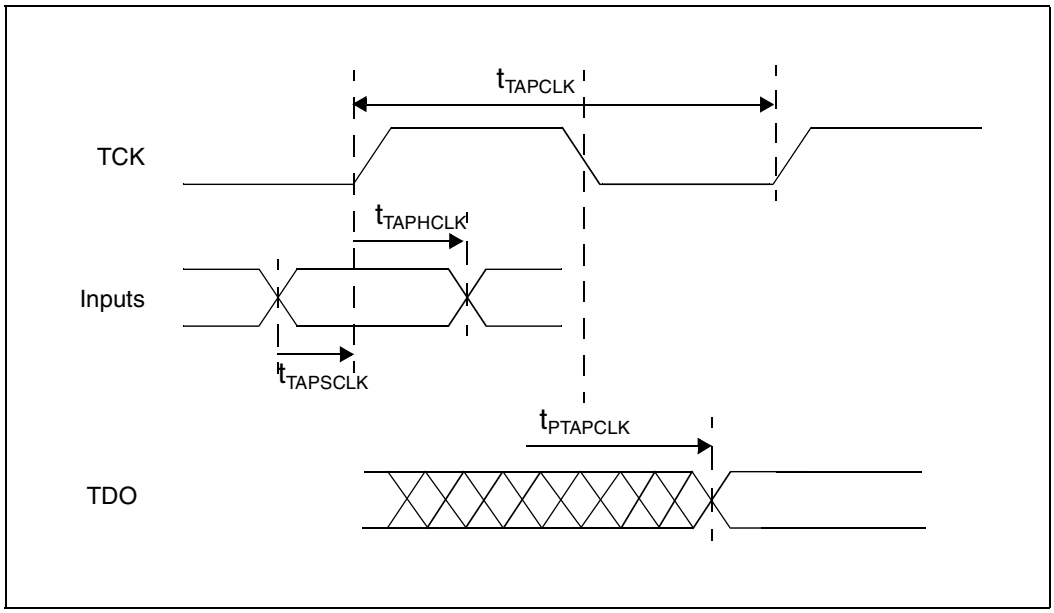


Table 55: JTAG input/output ports timings

Symbol	Parameter	Min	Max	Units
Input clock	TCK period	20		ns
$t_{TAPHCLK}$	Inputs setup to TCK rising edge	5		ns
$t_{TAPSCLK}$	Inputs hold to TCK rising edge	5		ns
$t_{PTAPCLK}$	Output delay to TCK falling edge		15	ns

11.8 EMI timings

All of the outputs come from a mux controlled by the clock. It is assumed that the EMI is programmed so that all the outputs are changed on the falling edge of the clock.

The following tables assume an external load of 25 pF on all EMI pads.

11.8.1 Synchronous devices

All synchronous transactions originate and terminate at flip flops within the padlogic. Outputs are generated with respect to the falling edge of the bus clock, and inputs are sampled with respect to the rising edge.

EMI-Clock: EMISFLASH.

EMI-outputs: EMIADDR[*], EMIDATA[*], NOTEMICS*, NOTEMIBE, NOTEMIOE, NOTEMILBA, NOTEMIBAA, EMIRDNOTWR.

EMI-inputs: EMIDATA[*], EMIREADYORWAIT

Figure 26: EMI synchronous device waveforms

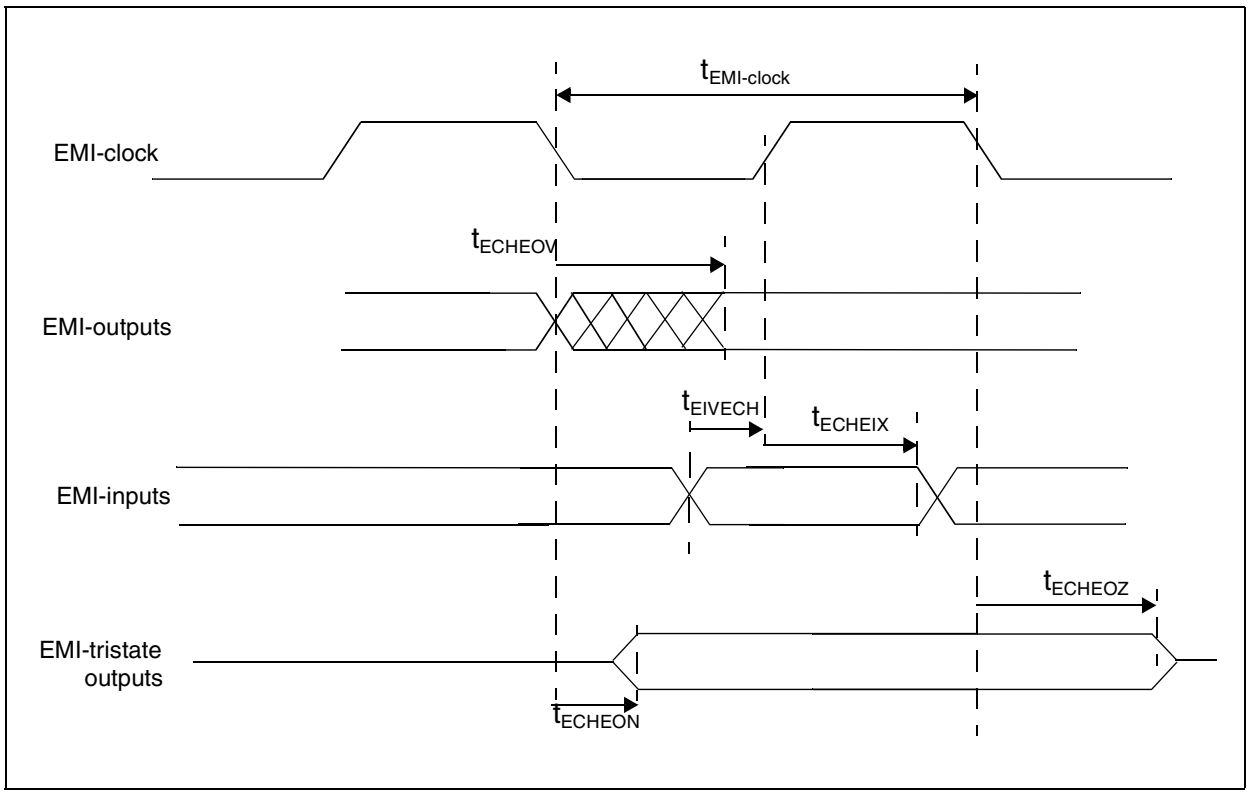


Table 56: EMI / SFLASH synchronous interface parameters

Symbol	Parameter	Min	Max	Units
Input clock	EMISFLASHCLK period	33		ns
t_{ECHEOV}	Bus clock falling edge to valid data	0	4	ns
t_{EIVECH}	Input valid to rising clock edge (input setup time)	5.5		ns
t_{ECHEIX}	Rising clock edge to input invalid (input hold time)	0		ns
t_{ECHEON}	Falling clock edge to data valid (after tristate output)	2		ns
t_{ECHEOZ}	Falling clock edge to data valid (before tri-state output)		-3	ns

These values are static offsets within a bus cycle, they should be read in conjunction with the waveforms in *External memory interface* (EMI), which are cycle accurate only.

11.8.2 Asynchronous memory/peripherals

The EMI strobes are programmed in terms of internal clock phases, that is to say with half cycle resolution. The clock to output delay for all outputs (address, data, strobes) are closely matched with a skew tolerance of -3 ns / +3 ns (assuming an external load of 25 pF on pads).

The input latch point for a read access is determined by the number of programmed EMI subsystem clock cycles for the latch point. The correction allows the latch point to be measured from the edge of an active chip select, that has been programmed to rise at the programmed read latch point.

Time between the address bus switching and a chip select or data bus output switching is n programmed phases +/-3 ns. That is worst case the chip select or data is maximum of 3 ns after the address, or worst case the chip select or data is 3 ns before the address.

For a read cycle the data is latched by the STx7100 at the programmed number of EMI subsystem clock cycles from the end of the access plus a latch point correction time, which is effectively the read setup time. The latch point correction time (read setup time) is a minimum of 5 ns + skew tolerance correction of the output signal used as a reference. This is 5 +/- 3 ns, thus the minimum read setup time relative to a strobe is 8 ns. This ensures the read hold time is always a minimum of 0 ns, guaranteed by design.

Figure 27: Asynchronous access - read waveforms

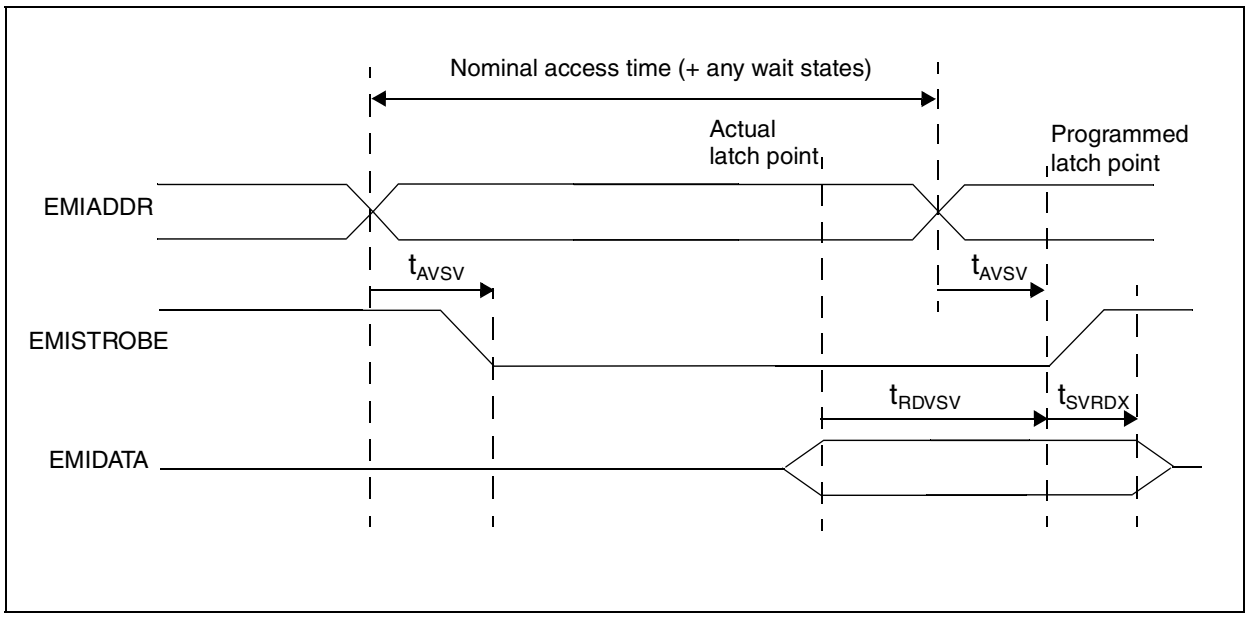


Figure 28: Asynchronous access - write waveforms

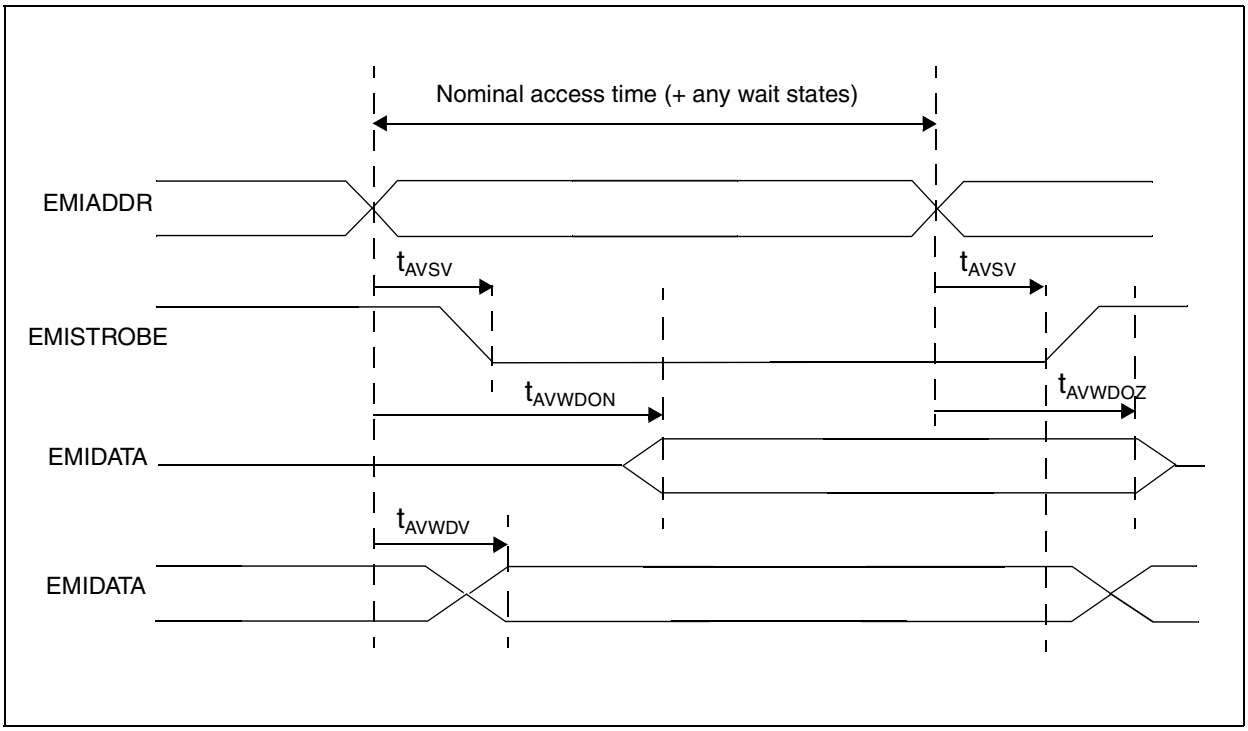


Table 57: EMI/asynchronous memory/peripherals interface parameters

Symbol	Parameter	Min	Max	Units	Note
t_{AVSV}	Address valid to output strobe valid	-1.5	3	ns	a
t_{RDVSV}	Read data valid to strobe valid (read setup time)	8		ns	b
t_{SVRDX}	Read data hold time after strobe valid (read hold time)	0		ns	c
t_{AVWDON}	Address valid to write data valid (after tristate output)	3		ns	d
t_{AVWDOZ}	Address valid to write data valid (before tristate output)		-4.5	ns	
t_{AVWDV}	Address valid to write data valid	-2	2	ns	

a. Skew plus nominal N programmed EMI subsystem clock cycles of strobe delay.
b. Skew from nominal programmed read latch point.
c. Minimum values are guaranteed by design.
d. Skew from nominal programmed phases of data drive delay.

The above table assumes an external load of 25 pF on all EMI pads.

11.9 LMI DDR-SDRAM timings (system and video)

For write sequences, the timing reference is the LMIVIDCLK or LMISYSCLK edge.

For reads, the timing reference is the LMIVIDDQS or LMISYSDQS edge.

The timings are based on the following conditions:

- input rise and fall times of 1.5 ns (10% to 90%),
- output load = 10 pF,
- output threshold = 0.5 * VDD25.

Figure 29: LMI DDR-SDRAM waveforms

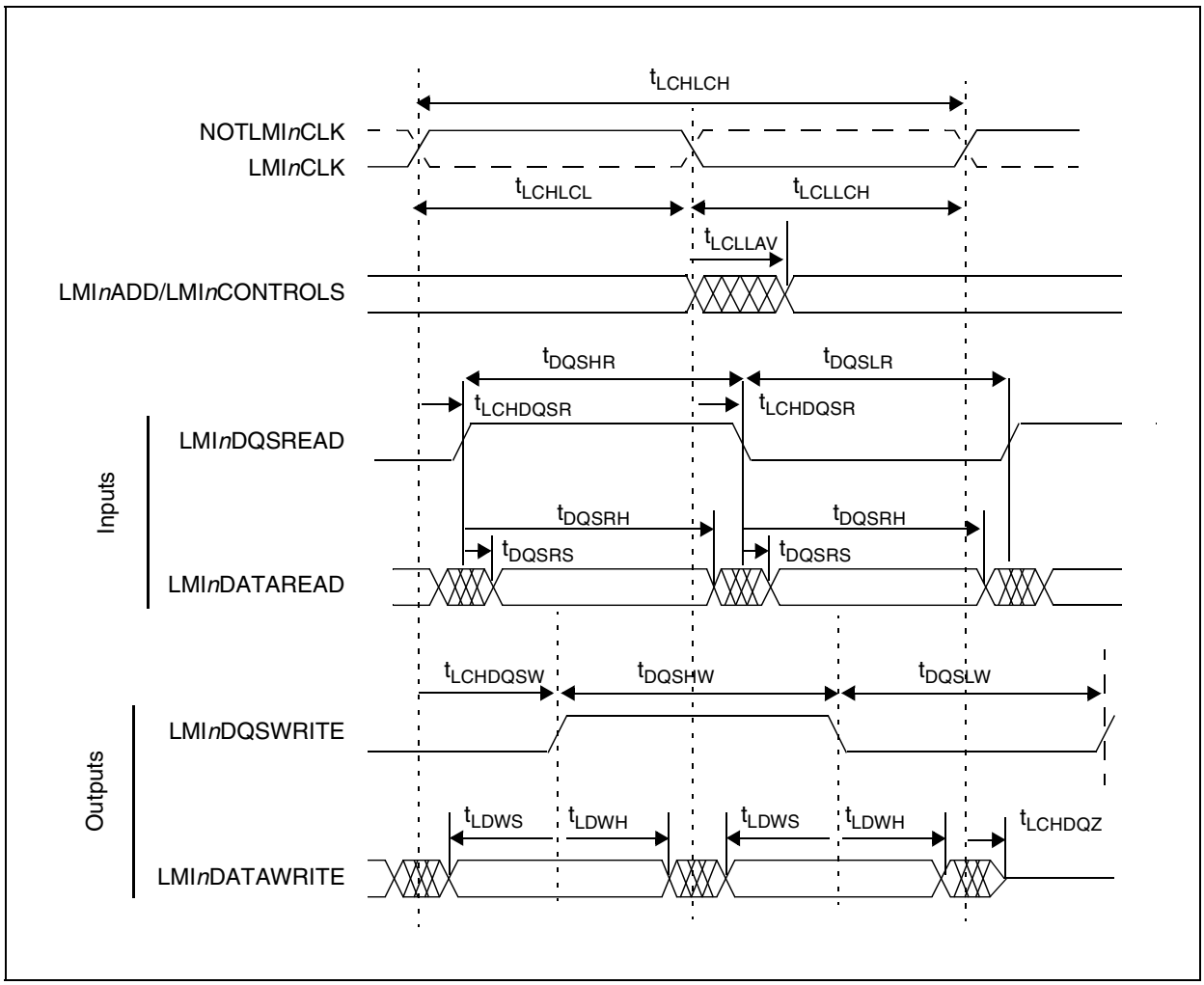


Table 58: LMI DDR-SDRAM AC timings

Symbol	Parameter	Min	Max	Units	Note
Outputs					
t_{LCHLCH}	LMI*n*CLK period	6.0		ns	
t_{LCHLCL}	LMI*n*CLK high time	0.45		t_{LCHLCH}	
t_{LCLLCH}	LMI*n*CLK low period	0.45		t_{LCHLCH}	
t_{DQSHW}	Write LMI*n*DQS high	0.45		t_{LCHLCH}	
t_{DQSLW}	Write LMI*n*DQS low	0.45		t_{LCHLCH}	
t_{LCLLAV}	LMI*n*CLK falling edge to LMI*n*ADD/LMI*n* controls valid	-1.5	+1.5	ns	a
$t_{LCHDQSW}$	LMI*n*CLK edge to write LMI*n*DQS edge	-0.8	-0.8	ns	
t_{LDWS}	Write data valid to write LMI*n*DQS edge (setup)	0.7		ns	
t_{LDWH}	LMI*n*DQS edge to write data valid (hold)	0.7		ns	
t_{LCHDQZ}	LMI*n*CLK high to write data Z		1	t_{LCHLCH}	
Inputs					
$t_{LCHDQSR}$	Allowable arrival time of LMI*n*DQS edge with respect to LMI*n*CLK edge	-0.8	+0.8	ns	
t_{DQSHR}	Read LMI*n*DQS high time	0.45		t_{LCHLCH}	
t_{DQSLW}	Read LMI*n*DQS low time	0.45		t_{LCHLCH}	
t_{DQSRS}	Latest allowable invalid read data after LMI*n*DQS edge		0.7	ns	
t_{DQSRH}	Earliest allowable invalid data before following LMI*n*DQS edge	$t_{LCHLCH}/2$ - 0.7		ns	

a. Applies to NOTLMIN[1:0], NOTLMI*n*WE, LMI*n*ADD[12:0], LMI*n*BKSEL[1:0], LMI*n*DATAMASK[3:0], LMI*n*CLKEN.

11.10 PIO output AC specification

Reference clock in this case means the last transition of any PIO signal.

Note: There are two different sets of PIO timings, one for the SSC (I2C) outputs and one for all other PIO outputs.

Figure 30: PIO output waveforms

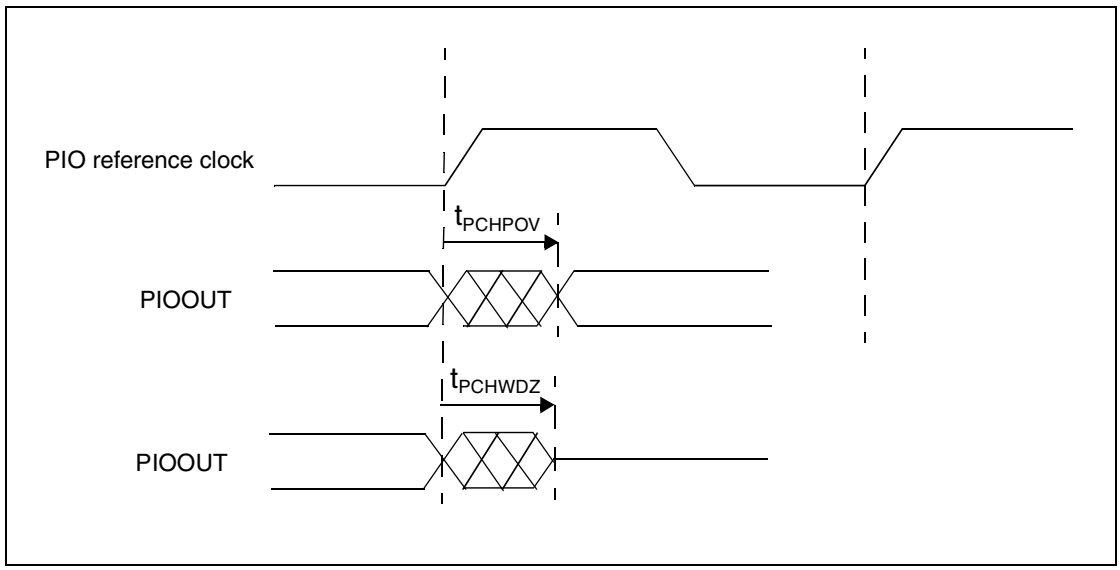


Table 59: PIO SSC output AC timings

Symbol	Parameter	Min	Max	Units
t_{PCHPOV}	PIO_REFCLOCK high to PIO output valid	-20.0	0.0	ns
t_{PCHWDZ}	PIO tristate after PIO_REFCLOCK high	-20.0	5.0	ns
t_{PIOr}	Output rise time	3.0	30.0	ns
t_{PIOf}	Output fall time	3.0	30.0	ns

Table 60: PIO other outputs AC timings

Symbol	Parameter	Min	Max	Units
t_{PCHPOV}	PIO_REFCLOCK high to PIO output valid	-6.0	0.0	ns
t_{PCHWDZ}	PIO tristate after PIO_REFCLOCK high	-6.0	5.0	ns
t_{PIOr}	Output rise time	1.0	7.0	ns
t_{PIOf}	Output fall time	1.0	7.0	ns

11.11 MAFE interface output AC specification

Input clock: MAFE_SCLK (PIO2(7) when selected

Outputs: MAFE_HC1 (PIO2[3]), MAFE_DOUT (PIO2[4])

Figure 31: MAFE interface output waveforms

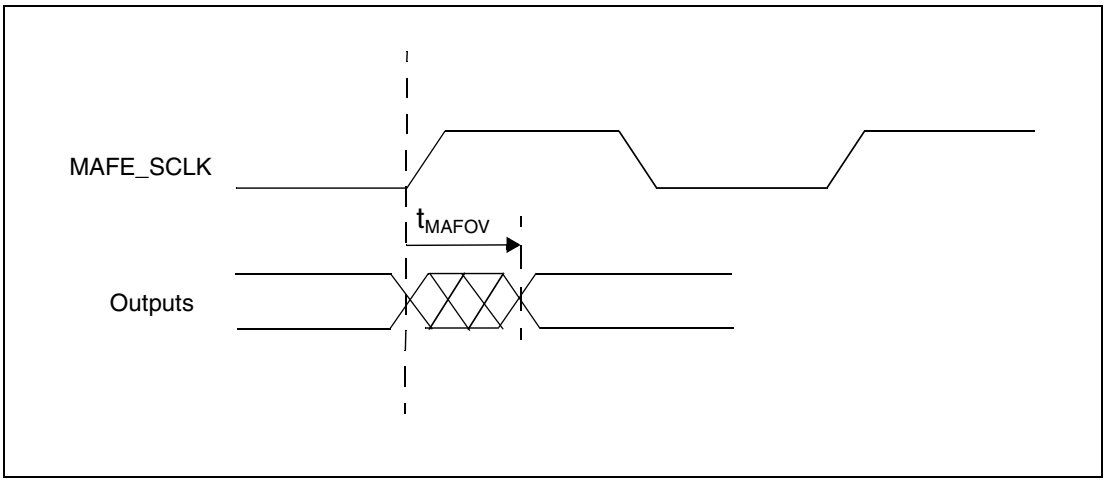


Table 61: MAFE interface output timings

Symbol	Parameter	Min	Max	Units
Input clock	MAFE_SCLK clock period		100	ns
t_{MAFOV}	Output delay to MAFE_CLK		20	ns

11.12 MAFE interface input AC specification

Input clock: MAFE_SCLK (PIO2[7]) when selected

Outputs: MAFE_DIN (PIO2[5]), MAFE_FS (PIO2[6])

Figure 32: MAFE interface input waveforms

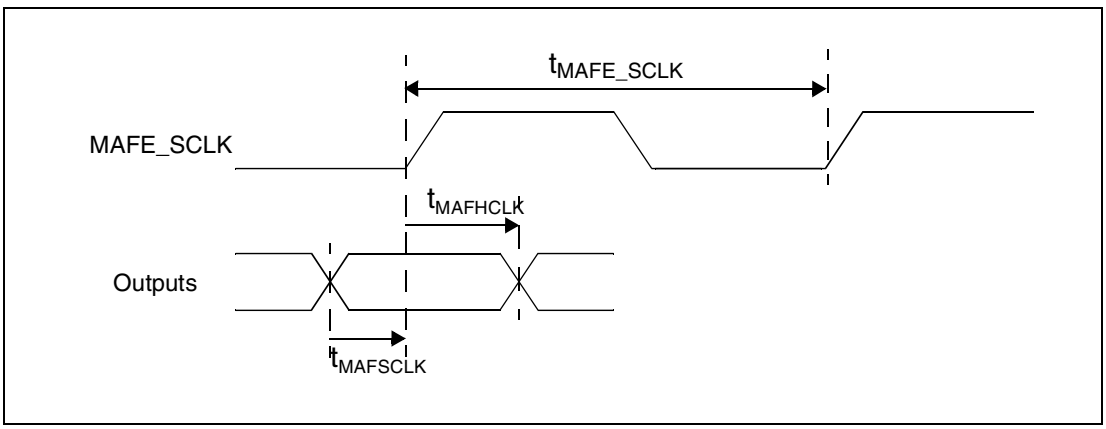


Table 62: MAFE interface input timings

Symbol	Parameter	Min	Max	Units
Input clock	MAFE_SCLK clock period		100	ns
$t_{MAFHCLK}$	Inputs setup to MAFE_CLK rising edge	20		ns
$t_{MAFSCLK}$	Inputs hold to MAFE_CLK rising edge	20		ns

12 HD and SD triple video DACs

12.1 Description

There are two identical sets of video DACs for HD and SD output. Both are triple high-performance 10-bit digital to analog converters, and consist of three 10-bit DAC modules joined together. The full-scale output for each DAC set is controlled by an external resistor.

Each DAC is able to drive 10 mA.

The blocks are powered by 2.5 V analog and 1.0 V digital supplies, with separate analog and digital grounds.

The blocks require a external precision resistor (R_{REF}) to provide a bandgap reference. The optimum R_{REF} value is 7.72 kΩ +/- 1%.

The blocks' analog current sources provide an voltage output range of 1.4 V, with an optimum linearity through an external precision resistor (R_{LOAD}). The R_{LOAD} optimum value is 140 Ω +/- 1%.

The exact calculation for the voltage output range is:

$V_{OUT} = 77.31 * (R_{LOAD}/R_{REF})$

with:

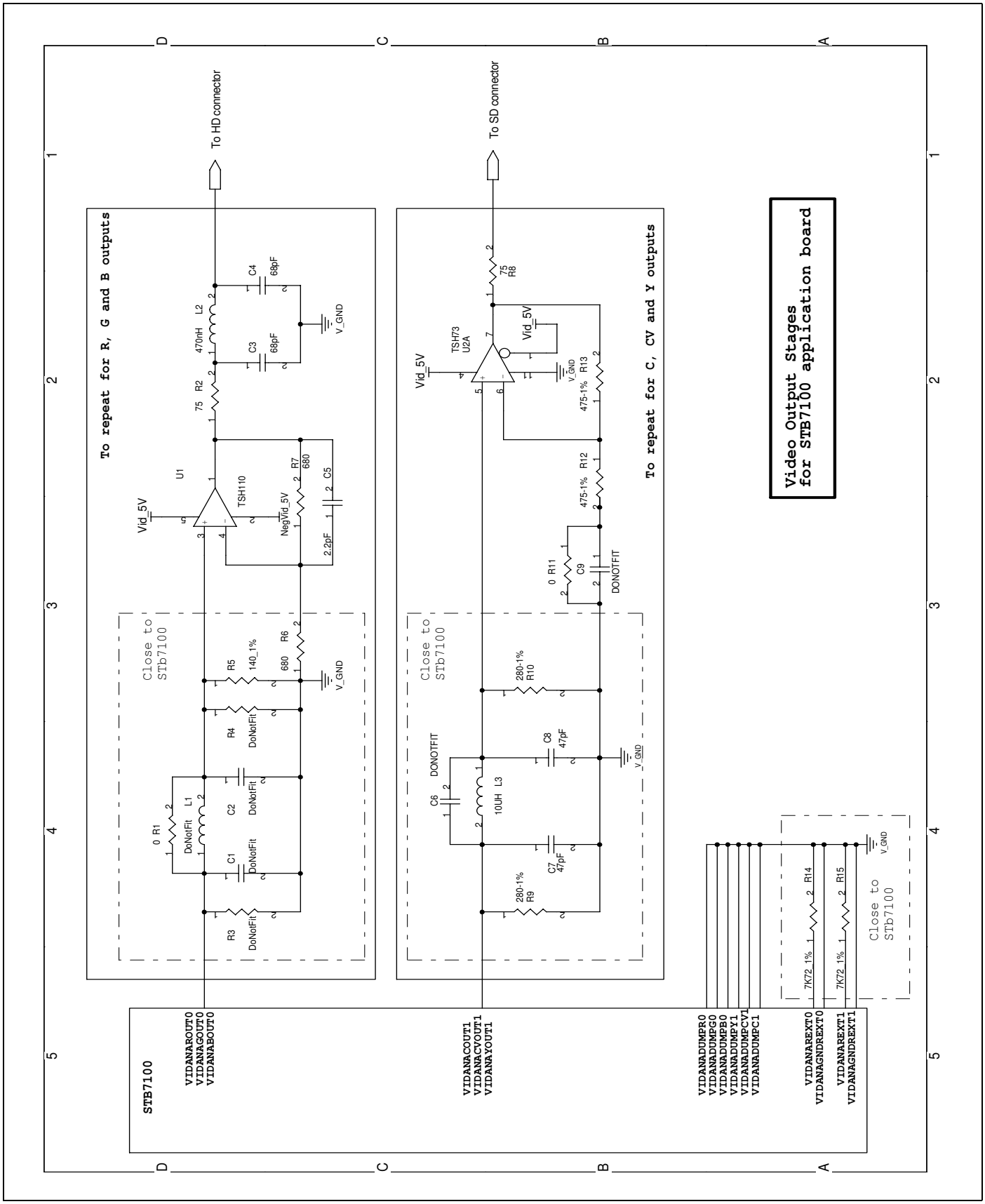
R_{REF} = reference resistor; maximum value is 7.72 kΩ

R_{LOAD} = load resistor

12.2 Output-stage adaptation and amplification

An example recommended video output stage with the external connections is shown in Figure 33.

Figure 33: Output stage schematic



13 Audio DAC

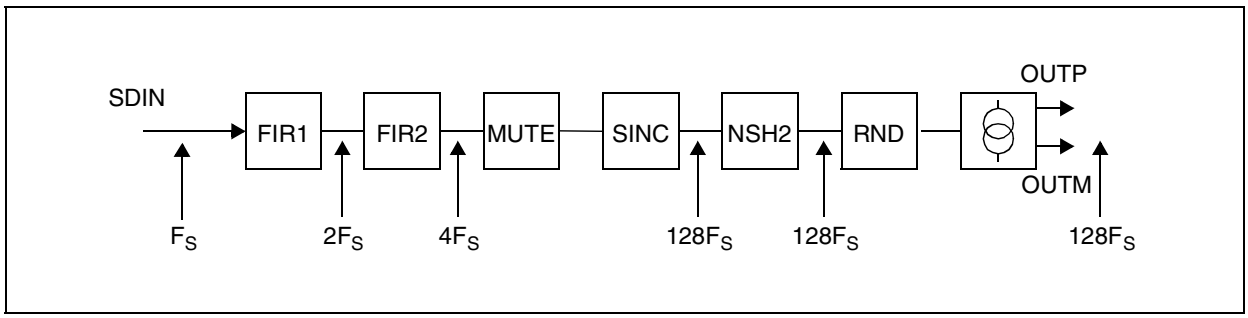
13.1 Description

The audio digital-to-analog converter (DAC) is a high performance stereo audio converter which accepts a 24-bit serial data stream from the audio decoder and converts it into a current source analog output signal. This signal is then filtered and transformed into a voltage output signal by an external analog filter.

The data converter uses a sigma-delta architecture which includes a second order noise shaper. The sigma delta modulator is followed by a 5-bit DAC to achieve at least 18-bit resolution.

This DAC can operate at sampling frequencies of 32, 44.1 and 48 kHz, or indeed any audio frequency up to 48 kHz.

Figure 34: Digital flow



The input stream SDIN derived from the audio decoder, sampled at F_S, is first interpolated by two and then filtered by a 75th order FIR filter, FIR1. This signal, at $2F_S$, is interpolated by two and filtered by a 20th order FIR filter, FIR2. The signal, at 4 F_S, can be soft muted by the MUTE block, and enters the SINC filter which interpolates by 32. The noise shaper then transforms this signal to five bits. A randomizer then expands the data to a thermometer code and permutes the sources to avoid mismatch between the 32 current sources.

The audio frequency synthesizer, within the clock generator, provides a system clock at 256 x F_S which is divided down internally to produce all other clocks.

The audio DAC group delay is 23 periods of the sampling frequency.

13.2 Operating modes

13.2.1 Reset

The audio DAC is reset by the chip global reset. The audio DAC can also be reset via the configuration register bit ADAC_CFG.NRST.

At reset, the audio DAC is disabled.

13.2.2 Power supplies

For better noise immunity, and to fulfill the specifications in terms of output range, the audio DAC has several different supply pairs.

- AUD_GNDA, AUD_GNDAS, AUD_VDDA: ground and 2.5 V analog supplies for the switches (control of the current sources) are supplied to the chip externally.
- GNDE_AUD_ANA, VDDE2V5_AUD_ANA: ground and 2.5 V analog supplies for the audio DAC pads ring are supplied to the chip externally.

13.2.3 Input/output signals

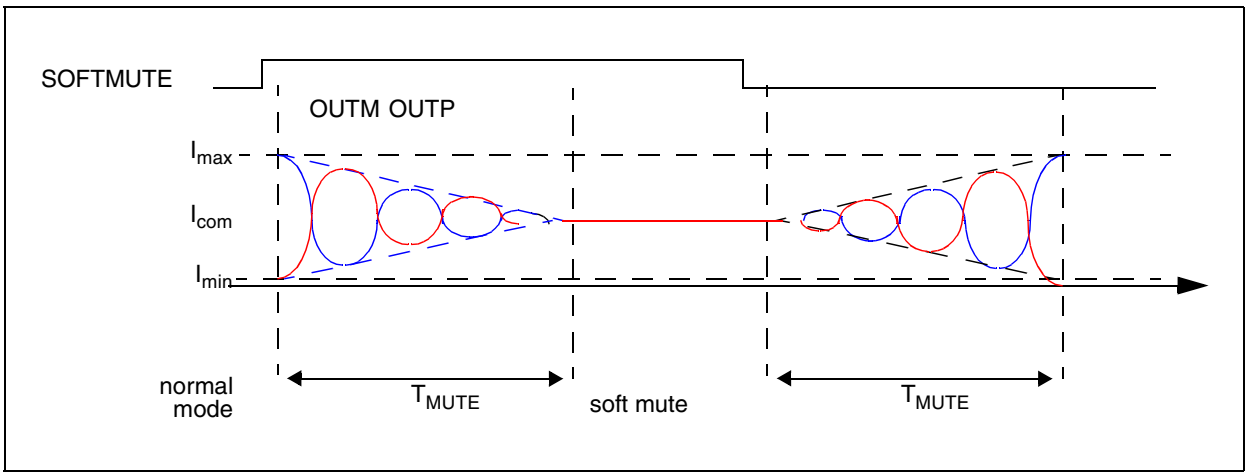
Table 63: Audio DAC output signals

Pin name	Description
AUDANAPRIGHTOUT	Right channel, differential positive analog output. The signal is then filtered.
AUDANAMRIGHTOUT	Right channel, differential negative analog output. The signal is then filtered.
AUDANAPLEFTOUT	Left channel, differential positive analog output. The signal is then filtered.
AUDANAMRIGHTOUT	Left channel, differential negative analog output. The signal is then filtered.
AUDANAIREF	DAC reference current output. This pin should be connected to an external resistor. 575 ohm +/- 1%.
AUDANAVBGFIL	DAC filtered reference voltage input. This pin should be connected to an external 10 µF capacitor (ground connection AUD_GNDA).

13.2.4 Soft mute

The mute function is controlled by bit ADAC_CFG.SMUTE. The current output signal (AUDANAPLEFTOUT/AUDANAPRIGHTOUT, AUDANAMLEFTOUT/AUDANAMRIGHTOUT) is first attenuated to 96 dB. When the output current reaches the common mode current I_{com}, the current sources are switched off one after the other in order to decrease the output current on AUDANAPLEFTOUT/AUDANAPRIGHTOUT. Once this sequence is complete, the analog part can be powered down. The total time for the mute/unmute sequence is at least 1920 sampling periods.

Figure 35: Soft mute and digital power down



13.2.5 Digital and analog power down

The digital part of the audio DAC can be disabled by setting ADAC_CFG.NSB to 0. An automatic soft mute avoids any pop noise.

The analog part of the audio DAC can be disabled by setting ADAC_CFG.PDN to 0. Depending on the external circuitry, pop noise may be unavoidable.

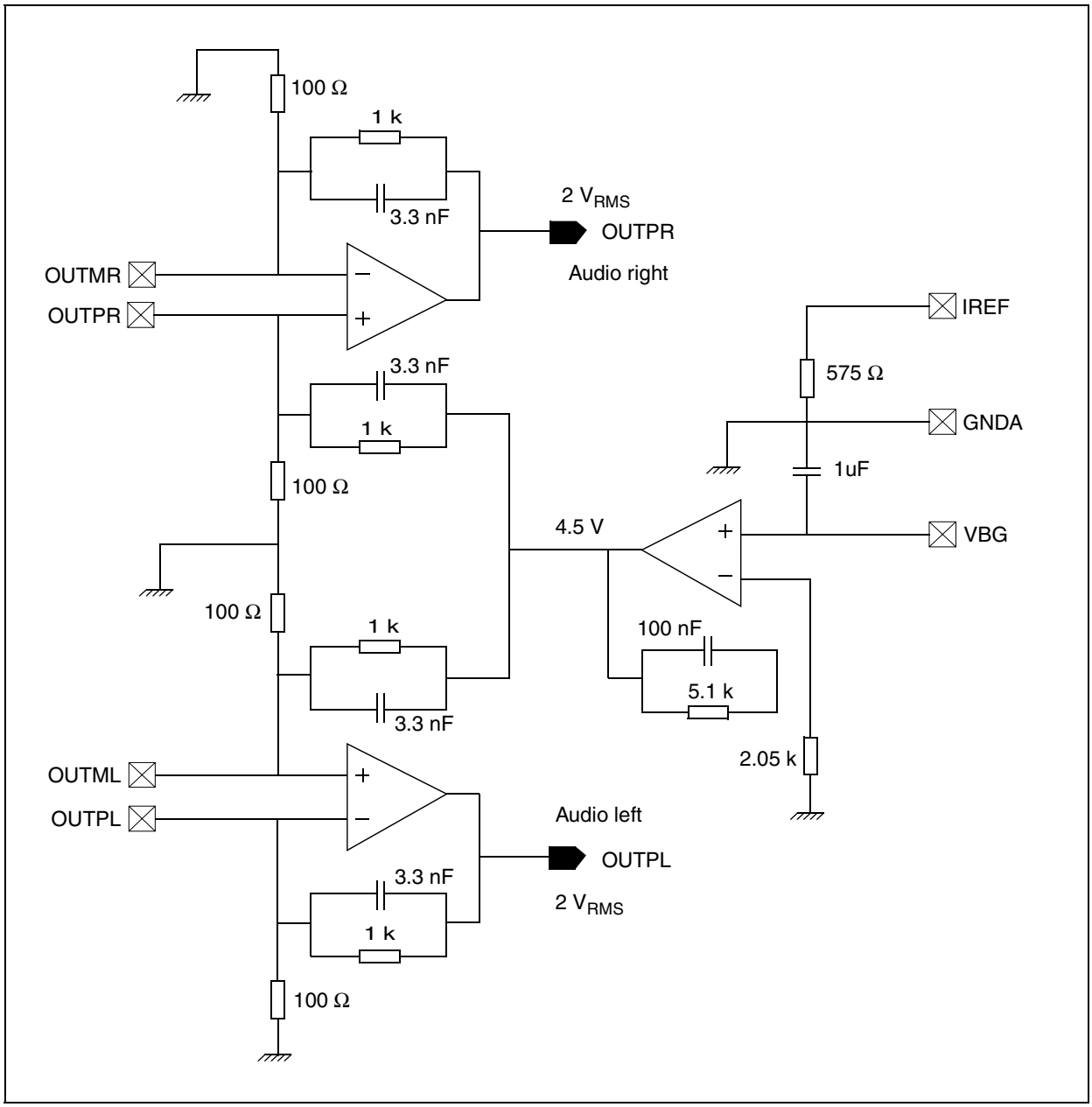
13.3 Output stage filtering

The audio DAC provides differential current source outputs for each channel. The use of a differential mode interface circuit is recommended to achieve the best signal to noise ratio performance. A single-ended mode interface circuit can be used, by grounding pins AUDANAMLEFTOUT and AUDANAMRIGHTOUT, but this is not recommended as the resulting signal to noise ratio is less than 90 dB.

An external 1% resistor R_{REF} should be connected to pin AUDANAIREF of the DAC. A typical value for R_{REF} is 575 Ω to get proper band gap functionality.

Figure 36 shows an audio output stage to deliver a 2 V_{RMS} signal.

Figure 36: 2 V_{RMS} schematic with +9 V power supply



14 PCB layout recommendations

14.1 SATA

To meet SATA specifications, some rules must be followed when designing the PCB.

Recommendations include power supply quality requirements and routing requirements for signal integrity.

14.1.1 Power supplies

To operate correctly, the SATA PHY must be isolated from the surrounding noise. This can be achieved by using a clean voltage source (linear DC regulator) and/or passive high-frequency filtering.

14.1.2 Voltage regulator

A dedicated voltage regulator for the SATA PHY is recommended.

The voltage regulator must be linear or LDO (switching or DC/DC regulators cannot be used).

A regulator with 2% spread around the nominal voltage is recommended.

The power supplies SATAVDDDLL, TMDSVDDX, LMISYSDLL_VDD, and LMIVIDDLL_VDD must be carefully designed since they are used for the high frequency clock generation.

The MDLL does not tolerate more than 50 mV of noise, specially below 10 MHz.

14.1.3 Power supplies passive filtering

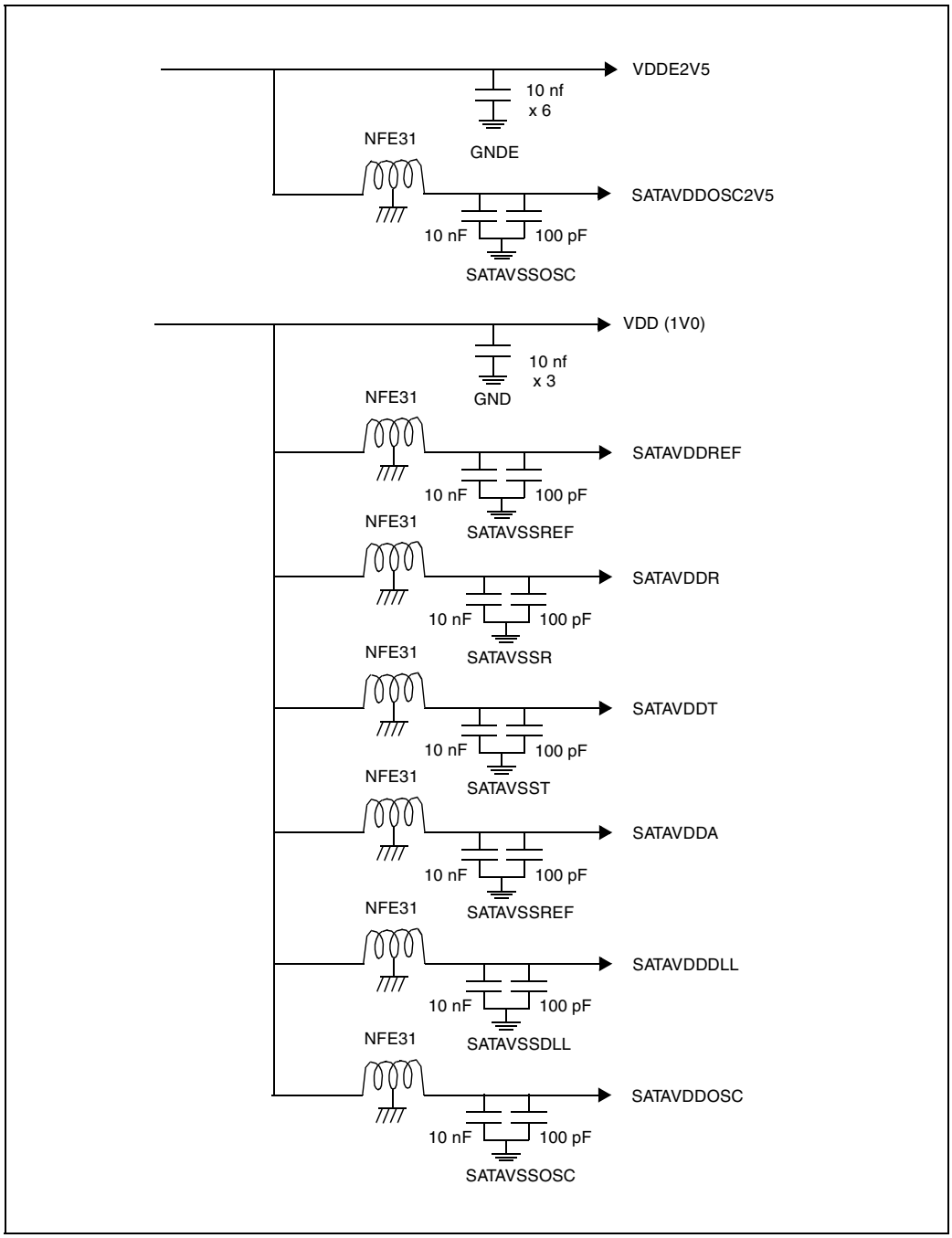
To achieve the necessary isolation from power supply noise, passive filter networks are required. This passive filtering avoids interference to the PHY from other sources, including the different power sources of the SATA. The most noisy source is VDDT.

14.1.4 Reference crystal

One possible quartz crystal to use is the EPSON FA-365, 30 MHz.

14.1.5 SATA interface filtering

Figure 37: SATA interface filtering



14.1.6 High speed serial trace routing

The PCB routing must be done with the following guidelines.

14.1.6.1 Ground plane

Microstrip with ground reference plane implementation is preferred.

No other signal than SATA should overlay the reference plane.

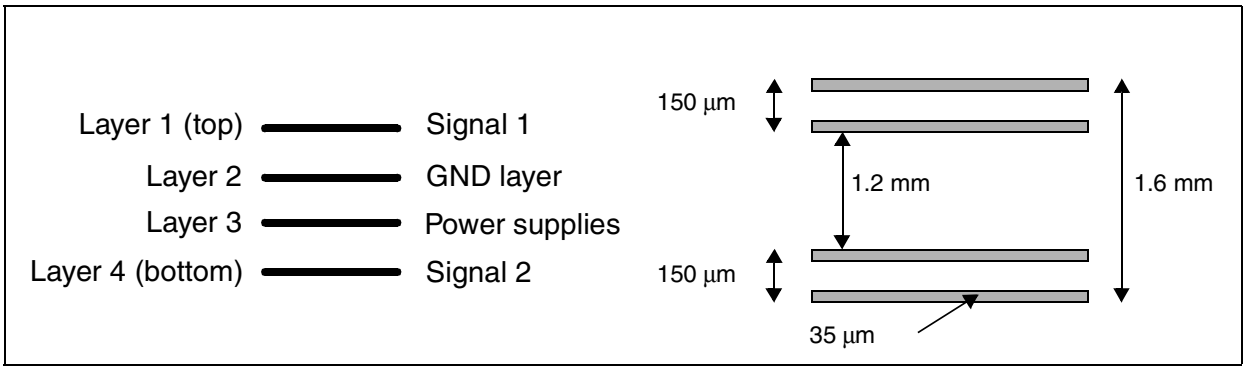
14.1.6.2 Differential signal pair routing

TXP/TXN and RXP/RXN are differential signals as defined by SATA spec.

- The distance from the chip to the SATA connector must be as short as possible.
- When routing a differential pair, the complementary signals must be matched in length. (<0.5 mm)
- No right angle allowed
- If a signal has to change layers, attention must be paid when adding a via about the return current: use a stretching ground via.
- Both traces of a differential pair must have a 50 Ω impedance to ground, without any impedance discontinuity from the chip to the SATA connector. This actually makes a differential impedance of 100 Ω and a common mode impedance of 25 Ω
- Vias act as strong impedance discontinuities: avoid them in the differential traces.
- Signal routing are preferred at top level.

Figure 38 below describes a PCB stack, with differential signals. The differential pair is preferably routed on the top layer. It is also possible but less recommended to route the differential signals on the bottom layer.

Figure 38: PCB stack with differential signals



The two traces of a differential pair must be loosely coupled, that is, their impedance to ground must be controlled and equals to 50 Ω The traces must be far enough from each other to ensure a low direct coupling between themselves.

14.1.6.3 AC coupling

The SATA PHY works in AC coupled configuration.

The AC capacitance has to be put on RX on the connector side (capacitance value = 10 nF, 12 nF max in the SATA specification).

14.2 DDR-SDRAM interface

14.2.1 Recommended PCB

To ensure better signals integrity and also to comply with the high density of the BGA package, it is recommended to use a 4-layer PCB with the following stack:

- Layer 1: Signal 1,
- Layer 2: Power plane,
- Layer 3: Ground plane,
- Layer 4: Signal 2.

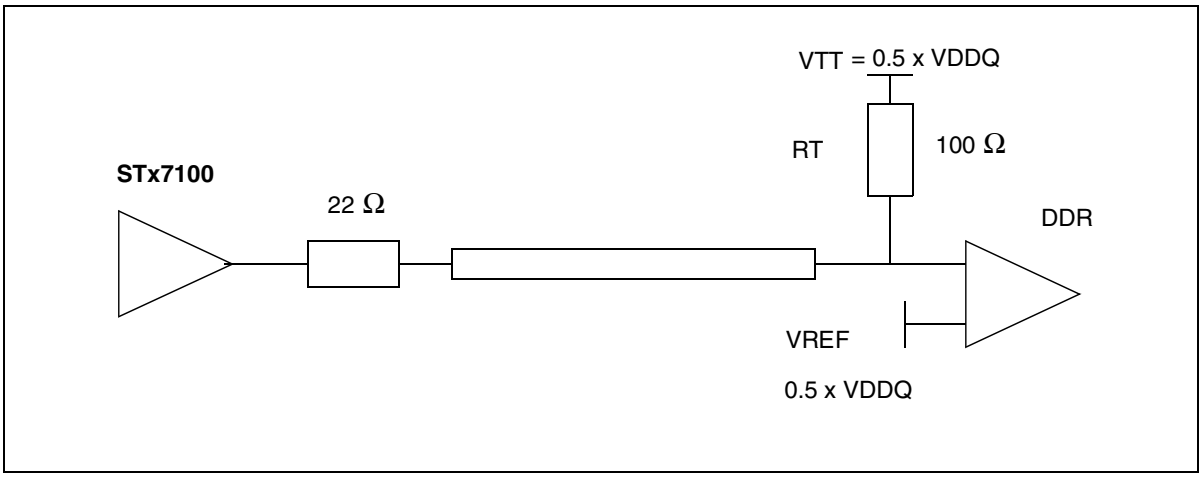
14.2.2 System considerations

To achieve a high bandwidth on the local memory interface, the STx7100 incorporates SSTL_2 buffers. At the board level it is therefore recommended to follow SSTL-2 signalling design rules.

The use of this high speed logic requires series resistors, parallel termination schemes and reference voltage circuitry.

The most suitable scheme is shown below. This reduces reflections, improves signal bandwidth and settling.

Figure 39: SSTL_2 Single terminated line



14.2.3 Power supply recommendations

An important point to take into consideration when designing such systems is to guarantee that both the STx7100 and the DDR devices have very clean digital power supplies.

To decouple the STx7100 requires more decoupling capacitors. To better achieve a good power supply decoupling, it is recommended to equally distribute 100 nF ceramic capacitors and 1 µF Tantalum capacitors over the LMI power supply pins.

14.2.4 Reference voltage

Both the STx7100 and the DDR devices require a reference voltage, so called VREF in Figure 39.

Some attention should be paid to the routing of VREF, since the SSTL_2 specification requires dynamic noise to be maintained to less than 2% of the VREF DC level.

It is recommended to decouple VREF to both VDDQ and VSSQ with balanced decoupling capacitors. The use of 100 nF ceramic capacitors is adequate to do so. A decoupling capacitor pair is recommended at each device location (DDR + STx7100).

It is necessary to keep VREF isolated from induced noise as possible. VREF should be routed over a reference plane, and preferably shielded. Layer 2 is preferable.

14.2.5 Termination voltage

The VTT generation circuit should be placed as close as possible to the parallel termination resistors to reduce the impedance and length of the signal return path.

The termination voltage should be routed to all the termination resistors through a thick copper track, with a minimum width of 1500 mils.

The termination resistors should be placed as close as possible to the DDR memory. Use of 5% resistors is adequate.

Resistor packs are acceptable, but signals within an RPACK must be from the same DDR signal group. No mixing of signals from different DDR signal groups is allowed within an RPACK. The parallel termination resistors connect directly to the VTT track with the shortest wire as possible.

VTT requires some decoupling capacitors close to RPACK. VTT should be routed over a reference plane.

On a 4-layer PCB, the most appropriate layer to route the termination voltage track is the layer 2 as it will not interfere with the signals routing that is the most critical part.

14.2.6 Memory bus layout general considerations

The memory interface bus can be split in three groups:

- the data bus and its associated strobes DQM and DQS,
- the address bus and associated control signals RAS, CAS and Write,
- the differential clock signals.

Each group requires specific attention described in details in the next paragraphs, but some general layout rules apply:

- Maintain the ground layer as a reference plane for all memory signals, that is, do not allow splits in the plane underneath both memory and STx7100 LMI.
- Series resistors should be placed as near to the close to the driver pin as possible. For the data bus, resistors should be placed at mid point because of bidirect transmission.
- As much as possible all signals should be routed without any via between the STx7100 and the memories. In all cases, minimize the usage of vias.
- All DDR traces should be as short as possible (not longer than 3 inches), and traces within a group should have close length in order to reduce the skew between different lines.
- All signal traces except clocks are routed using 5/5 rules. (5 mils traces and 5 mils minimum spacing between adjacent traces).
- Clocks are routed using 5 mils traces and 5 mils space to the 5 mils ground trace.

14.2.7 Address bus and control signals layout

For the address bus and the control signals, a branch type net topology is recommended for better signal integrity and smaller skew than daisy chain type.

14.2.8 Data bus layout

- Data lines and strobe signals within each byte group (for instance DQ[0 .. 7] and DQS0 and DQM0) should be routed so that the maximum difference in their lengths is minimum.
- Bit swapping within a byte group is permitted to facilitate routing.
- The DDR_DQS signal should have a length that is close to the longest other trace within its byte group.

14.2.9 Clock signals layout

- Minimize the usage of vias as much as possible.
- Clocks should have a 5 mill ground trace surrounding them, with vias stitched to ground at 0.5 inch intervals, or as often as routing allows.
- The CLK and N_CLK signals must be routed as differential signals in order to have a correct crossing point, that is, with identical lengths. This differential pair should be terminated by a 100 Ω resistor connected between both signals, and placed close to the memory chips.

14.3 USB2.0

14.4 Board design

Each guideline in this section (1, 2, 3…) has a reference attached to it indicating it is a board guideline.

14.4.1 Layer stacking

Guidelines provided in this document refer to a four-layer PCB stack up. In one case a two-layer PCB stackup is also shown. Board stackup can be done in two ways, as shown below. In either way, DPDN should be referenced with respect to the ground plane as shown. For an *n*-layer board (where $n > 4$), always reference the DPDN signals with respect to a ground plane.

1. DPDN on the top layer.

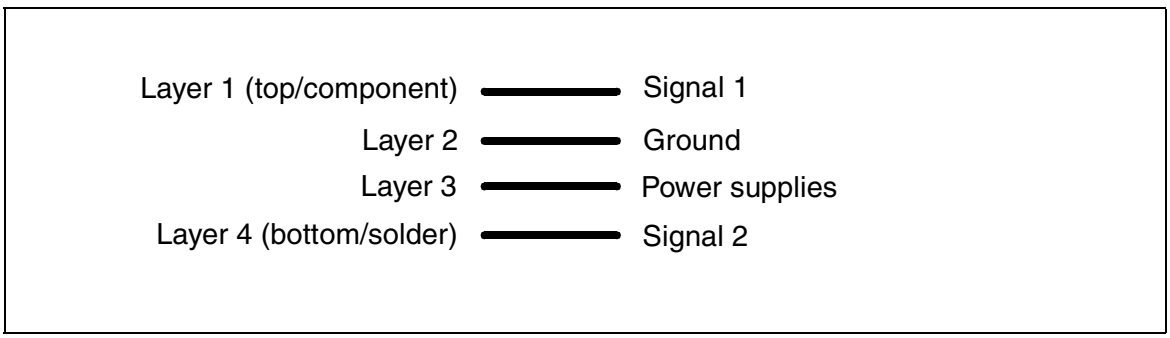


Figure 40: PCB Layer stackup - DPDN on top layer

2. DPDN on bottom layer:

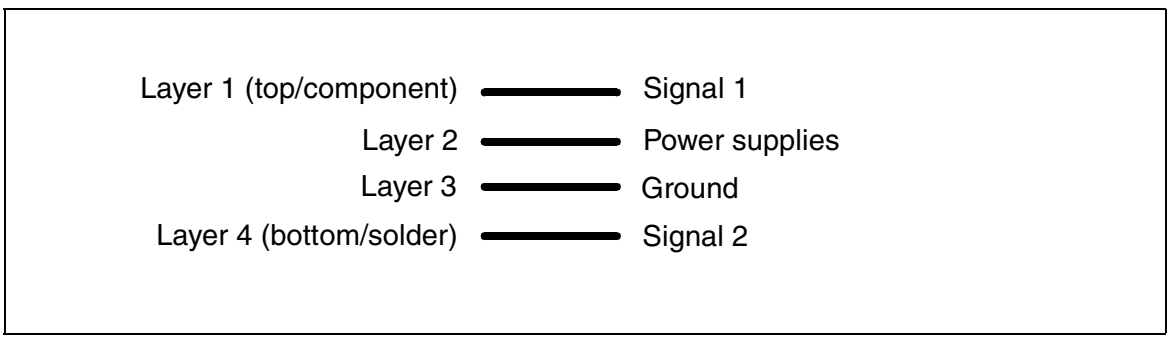


Figure 41: PCB Layer stackup - DPDN on bottom layer

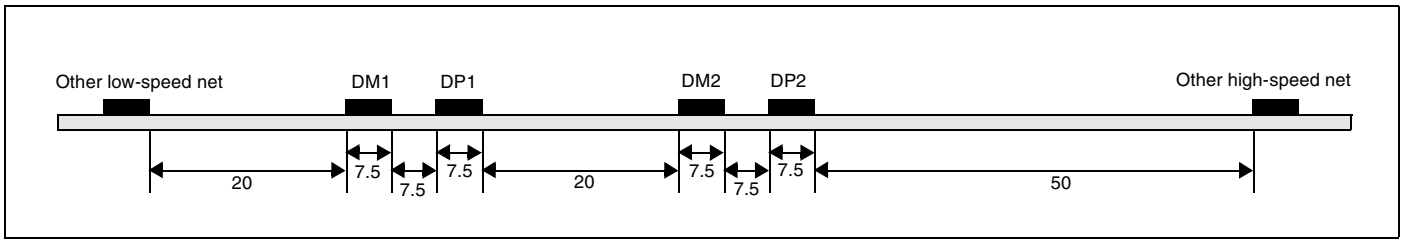


Figure 42: Recommended trace spacings (mils) for the above stackups

3. Follow Figure 42 for spacing other high- and low-speed nets; see rules 17 and 18. This diagram shows the trace geometry for a four-layer PCB. It can change with the number of layers present in the PCB.

14.4.2 Layout suggestion for the crystal

4. Avoid running traces below the crystal.
5. If surface mount crystal is used, place load capacitors on same side as surface mount crystal.
6. Do not route the traces between the crystal pins and the load capacitors through vias.
7. Load capacitors need to be located next to crystal pins.
8. Figure below shows how to do layout for the crystal. Both traces on left and right should be equidistant. A 30 MHz crystal can be chosen.
9. For the value of load cap, refer to the datasheet of crystal manufacturer. It should be a value below 33 pF.

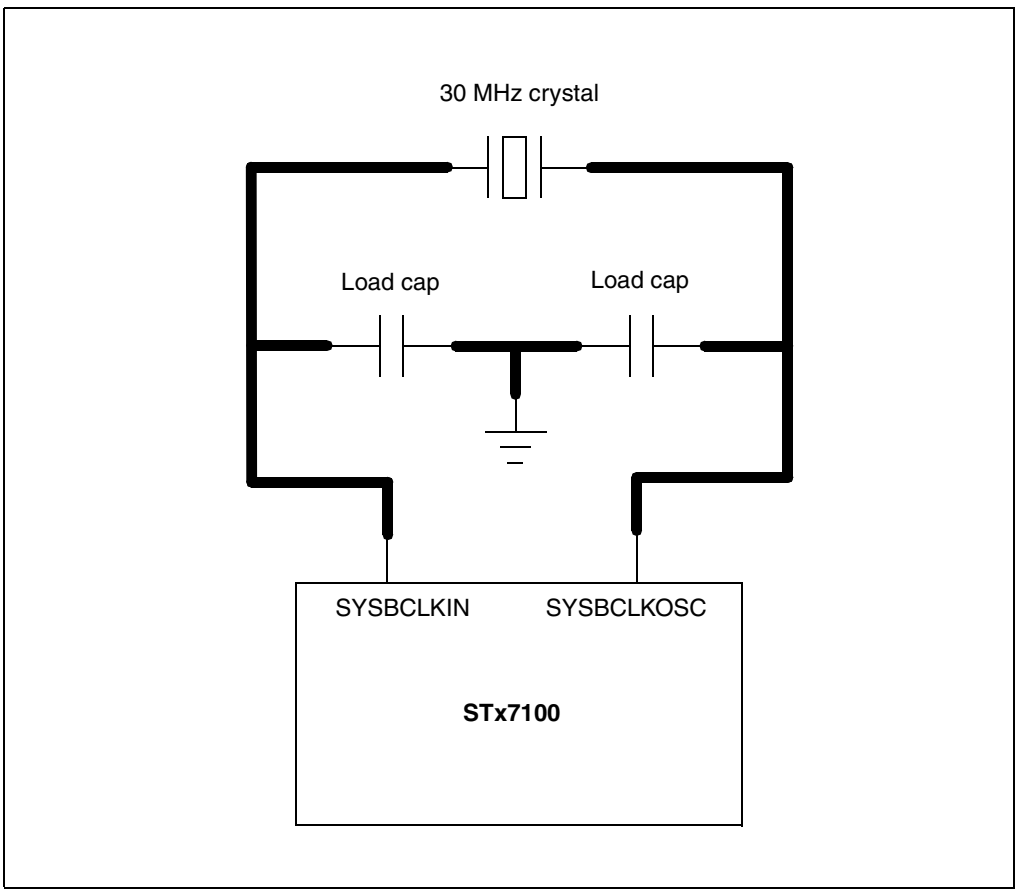


Figure 43: Recommended layout for crystal

14.4.3 USB connector

10. Pin 4 of the USB receptacle “GND” should be connected to the signal ground.
11. Chassis of the receptacle should be connected to pin 4 of USB receptacle.

14.4.4 Chassis and signal grounds

12. The chassis should be connected to the signal ground through an inductor/capacitor/ resistor.

14.4.5 DP and DN signals

13. For a 4-layer PCB, 7.5-mil traces with 7.5-mil spacing results in approximately 90 Ω differential trace impedance for DP and DN. For height above ground plane refer to rule 32. This point should be double checked by the PCB layout designer using an impedance calculator.
14. For a 2 layer PCB, Refer to rule B-31.
15. For an *n*-layer PCB where $n > 4$, rules 30 and 31 may not be applicable. Hence use an impedance calculator to get the trace geometry.
16. Use 20-mil minimum spacing between high-speed USB signal pairs (DPDN) and other low speed non periodic signal traces for optimal signal quality. This helps prevent crosstalk.
17. Use 50-mil minimum spacing between high-speed USB signal pairs (DPDN) and clock/high speed/periodic signals.
18. Differential impedance should be strictly respected: 90 Ω diff.
19. Routing should be with minimum vias (avoid if possible), no right angle, only 45 degrees (or round corners) turn with smooth edges.
20. Length of DPDN, traces should be kept at a minimum as possible.
21. High-speed USB signal pair traces should be trace-length matched. Max trace-length mismatch between High-speed USB signal pairs (such as DN and DP) should be no greater than 150 mils.
22. Do not route USB traces under crystals, oscillators, clock synthesizers, magnetic devices or ICs that use and/or duplicate clocks.
23. Do not place any zero Ω resistors on DPDN lines as this would change characteristic impedance.
24. Routing of DPDN lines should be avoided within 25 mils of any anti-etch to avoid coupling to the next split or radiating from the edge of the PCB.

14.4.6 REF connection

25. An external reference resistor R_{REF} should be connected to the ground of compensation block VSSC2V5 and its value should be 1.5 KΩ ±1%.
26. Note do not choose 10% tolerance resistors.

14.4.7 Other

27. External components quartz (24 or 30 MHz), 1.5 KΩ±1%, filters, decoupling capacitors needed.
28. No other external components required (like suppressor capacitors of 50 µF) for ESD protection of data lines. DN, DP pads have built in ESD protection.
29. Do not place any common mode chokes for controlling EMI. This would degrade the signal quality and cause eye diagrams to fail.

14.4.8 View of a PCB impedance calculator

The Following screenshots clearly define the trace parameters for 2- and 4-layer PCBs.

30. Two-layer PCB:

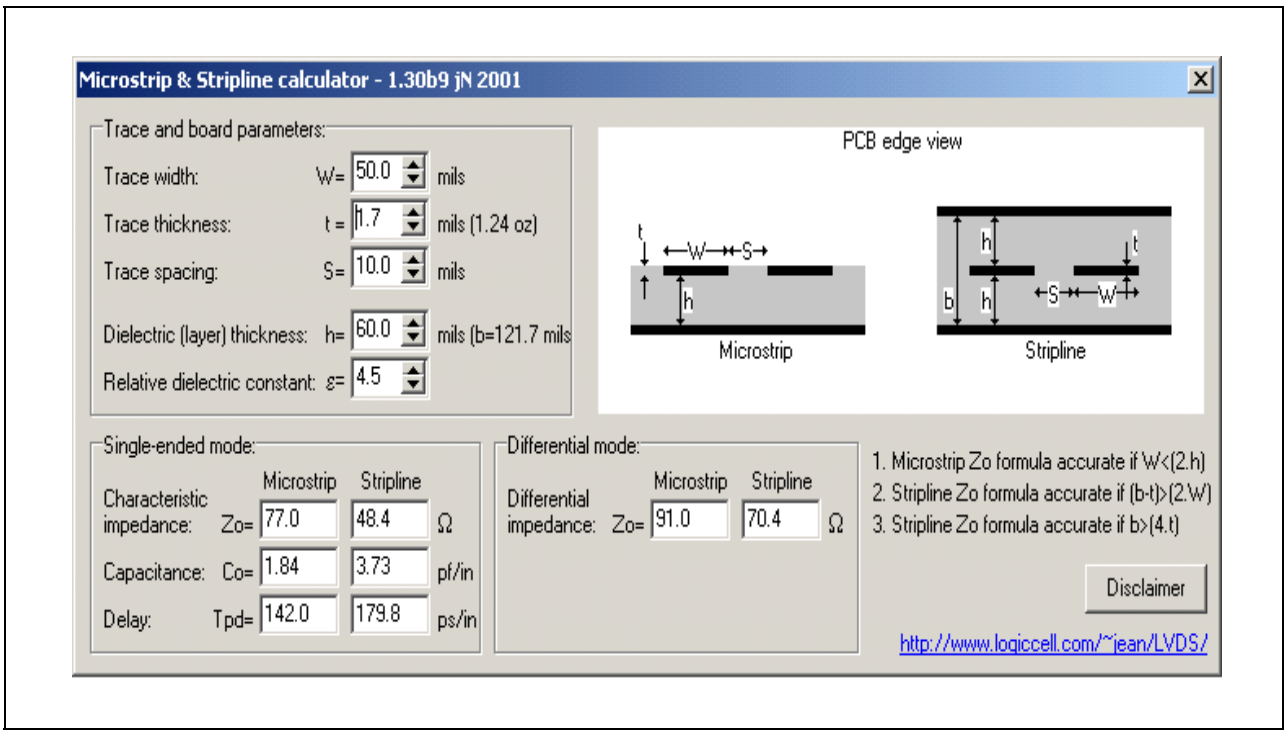


Figure 44: Trace geometry for two-layer PCB

31. Four-layer PCB:

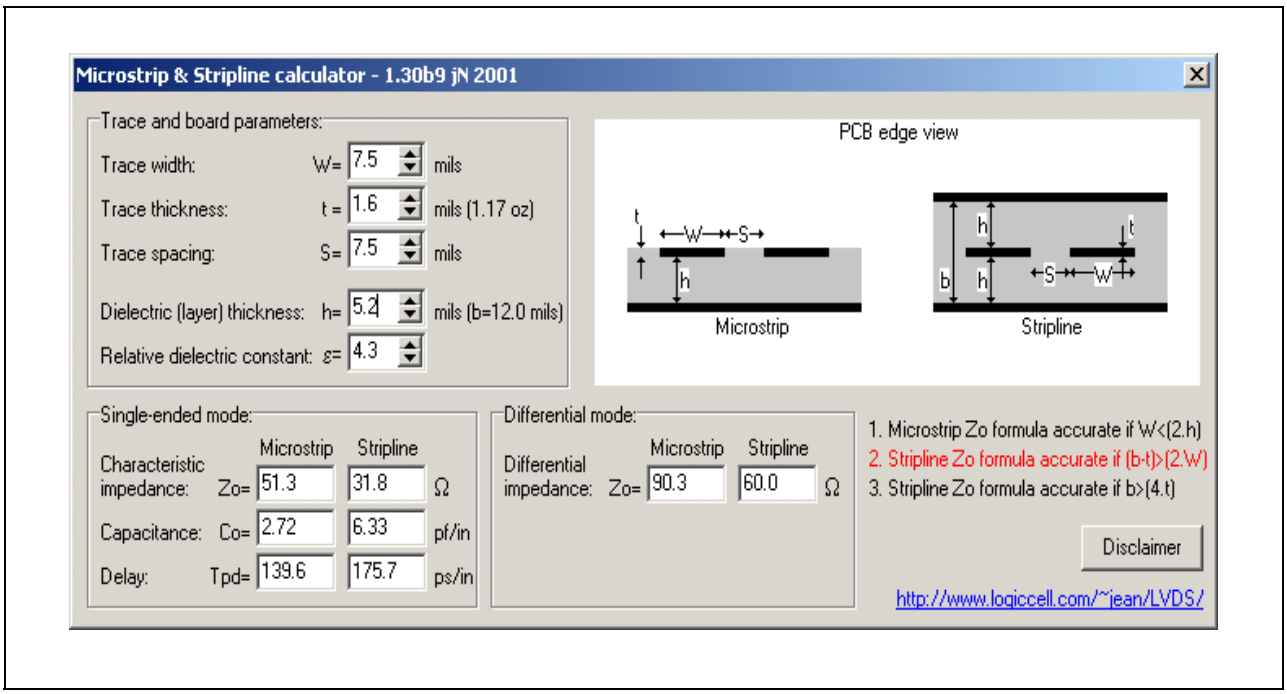


Figure 45: Trace geometry for four-layer PCB

14.5 Power supplies

14.5.1 Power supplies required by the Phy IP

Symbol	Description	HC (Volts)
USBVDDP2V5	USB2 2.5V PLL power	2.5
USBVDDP	USB2 PLL power	1.0
USBVDDB3V3	USB1.1 mode buffer power	3.3
USBVDDBC2V5	USB2 2.5V buffer power	2.5
USBVDDBS	USB2 buffer/serdes power	1.0

Table 64: Description of supply voltages

14.5.2 Recommended filters

32. The Phy IP needs > 3 dB attenuation of all frequencies above 1.2 MHz.
The attenuated noise ripple at VDDBS should be < *x* mV (pk-pk), where:
x = 20 for *f* (ripple frequency in MHz) >= 1.2
x = 20/(sin((pi**f*)/1.2)) for *f* (ripple frequency in MHz) < 1.2
Similarly, for other supply/ground, the attenuated noise ripple should be < *y* mV (pk-pk), where:
y = 50 for f (ripple frequency in MHz) >= 1
x = 50/sin(pi**f*) for *f* (ripple frequency in MHz) < 1
According to this, a suitable filter has to be chosen.
Ferrite inductor 2200 pF Type NFM60R, 25 V, 6 A component from Murata must be used.

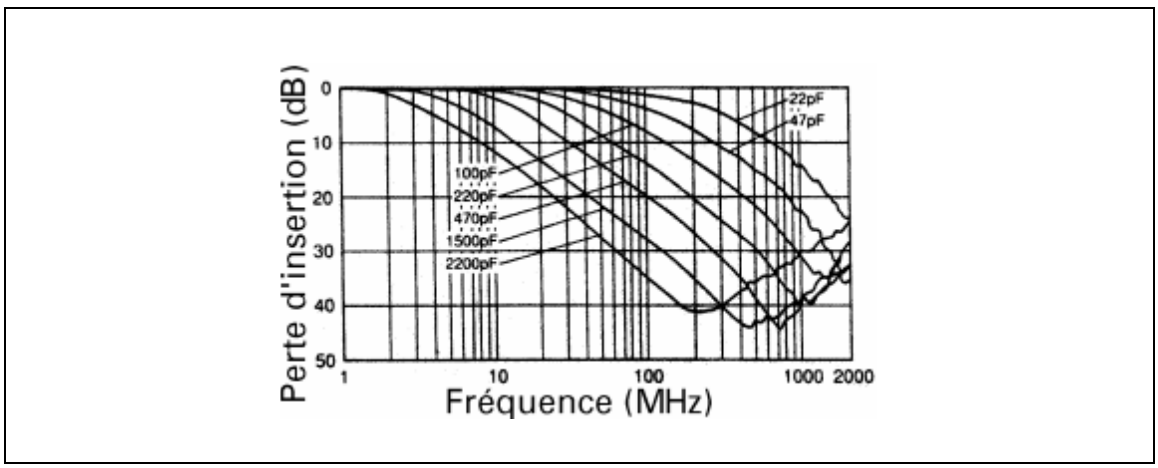


Figure 46: Recommended filter response

14.5.3 Sharing of supplies

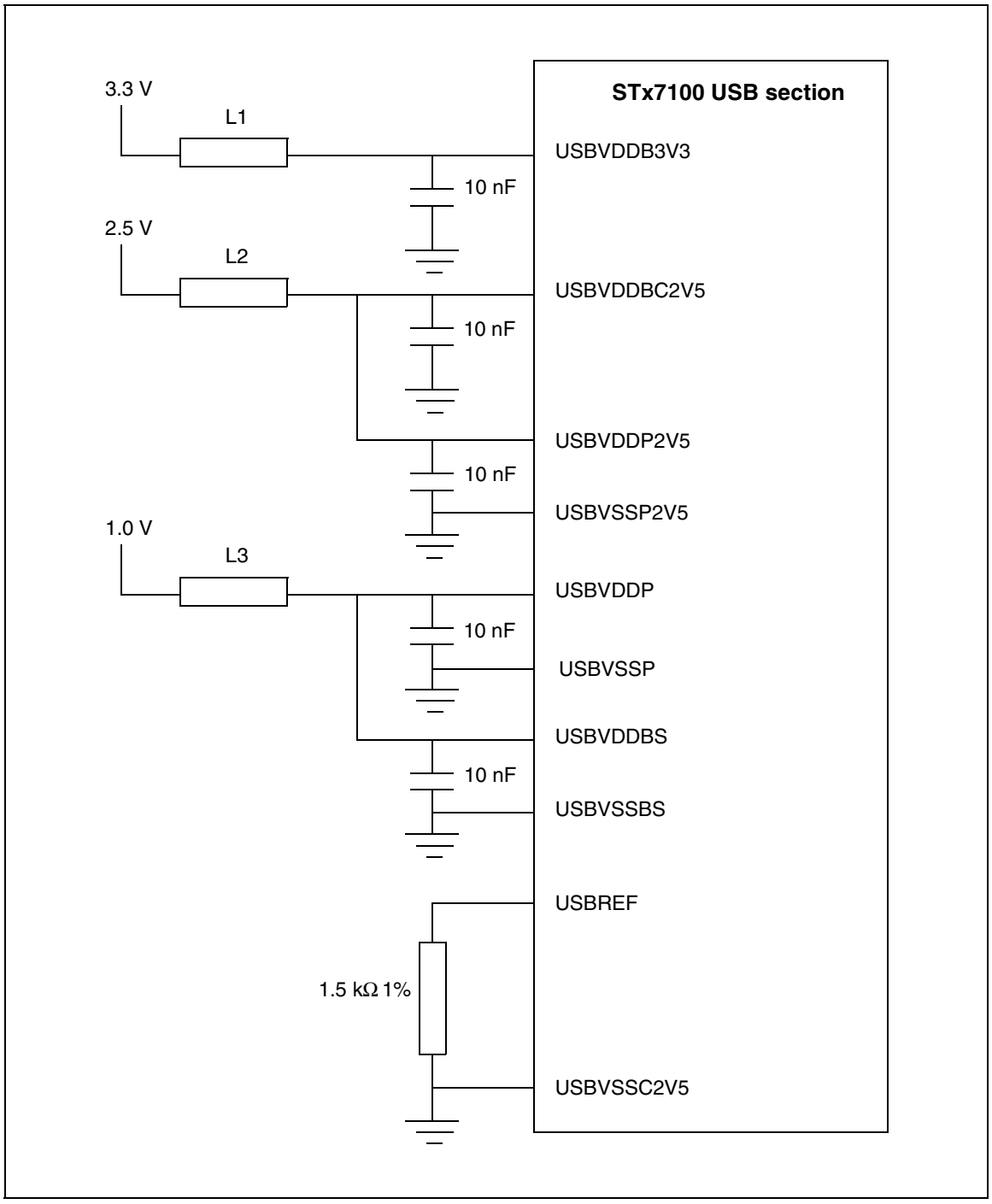
33. Connection of USB analog supplies with other digital supplies should be avoided. This creates a dependency of digital activity of the SOC and DP/DN quality.

14.5.4 Power distribution scheme and on-board filtering of individual power supplies

Filters and decoupling capacitors must be used to limit the noise environment for each and every power supply. Place all decoupling capacitors very close to the balls. R_{REF} should also be placed as close as possible to the ball of ASIC and connected to USBVSSC2V5.

34. The grounds of the decoupling capacitors are connected to their respective VSS. A value of 10 nF is recommended for decoupling capacitors.

Figure 47: Example filtering scheme



14.6 HDMI

When designing the HDMI interface, the signal traces should have the same length and must be kept as much as possible on the same layer. No additional external components are required.

Part 3

System infrastructure

15 Clocks

15.1 Overview

The STx7100 includes three clock generator (clockgen) subsystems:

- **Clock generator A** generates clocks for the CPUs, memories and STBus. It comprises two PLLs (PLL0 and PLL1) and programmable dividers to generate the group A clocks.
- **Clock generator B** generates clocks for the video, displays, transport and peripherals. It comprises two frequency synthesizers banks (FS0 and FS1), programmable dividers and a clock recovery module.
- **Clock generator C** generates the audio clocks. It comprises three independent audio frequency synthesizers (described in Chapter 59: *Audio subsystem on page 731*).

Clock input/output pins

The group A clocks take a reference input clock from the SYSACLKIN pin. This is a single input requiring an external oscillator. The default values of group A clock generation parameters are based on a 27 MHz reference clock. This is the recommended reference clock frequency; however, other clock values less than 40 MHz may be used.

The SYSBCLKIN/SYSCLKOSC pair is a crystal interface which is part of the SATA analog interface which integrates an oscillator requiring a 30 MHz crystal. In addition to driving the SATA and USB interface, this clock can be used as a reference clock to generate the group B and group C clocks.

The SYSBCLKINALT input provides an alternate reference clock for the group B and group C clocks instead of using the oscillator clock inside the SATA Phy. The default state is to use the 30 MHz SATA clock, but the alternate reference clock can be selected via registers CKGB_REF_CLK_SEL and AUD_FSYN_CFG. This input pin can be connected to the same reference clock that is input on the SYSACLKIN pin. The SYSBCLKINALT input can range from 27 MHz to 30 MHz. The programming of the frequency synthesizers must take into account the reference clock frequency.

The internal clocks can be observed or used as an auxiliary clock via the SYS_CLK_OUT pin, which gives access to the clocks of clockgen A. The clocks from clockgen B can be observed via an alternate PIO pad (PIO5 bit 2).

Encoder clock recovery

The STx7100 integrates a clock recovery module to recover the encoder clock. This module (DCXO) uses digitally controllable frequency synthesizers and an integrated digital clock recovery module. This feature can replace the external VCXO oscillator functionality and allows the use of a fixed oscillator. External VCXO functionality is still available in this mode however.

When an external VCXO is used, it must be connected to SYSBCLKINALT; SYSACLKIN can be connected to either a fixed oscillator or the VCXO.

When the DCXO is used, the SYSACLKIN pin is connected to a fixed oscillator and SYSBCLKINALT is either connected to SYSACLKIN or left unconnected (in this case, the SATA/USB 30 MHz clock is used).

Clockgen B includes a recovered 27 MHz clock. This clock is used for the programmable transport interface (PTI)'s system counter. The counter is used to compare arriving PCRs.

The alternative clocking schemes are summarized in Figure 48 and Figure 49.

Figure 48: STx7100 clocking scheme with an external VCXO

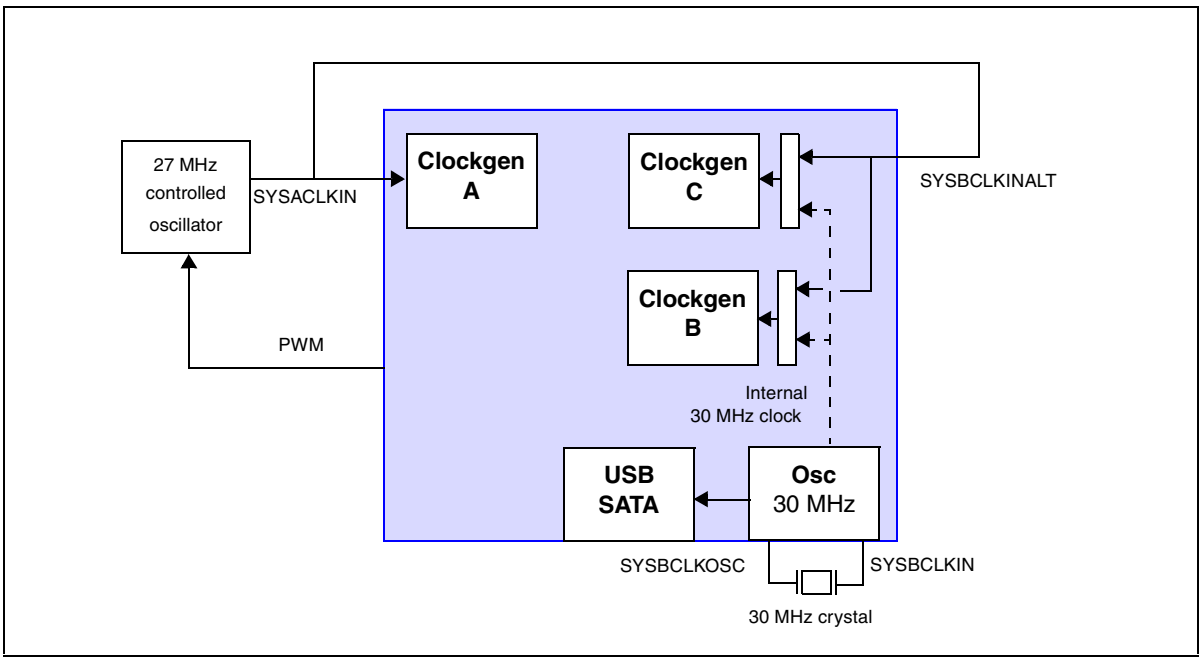
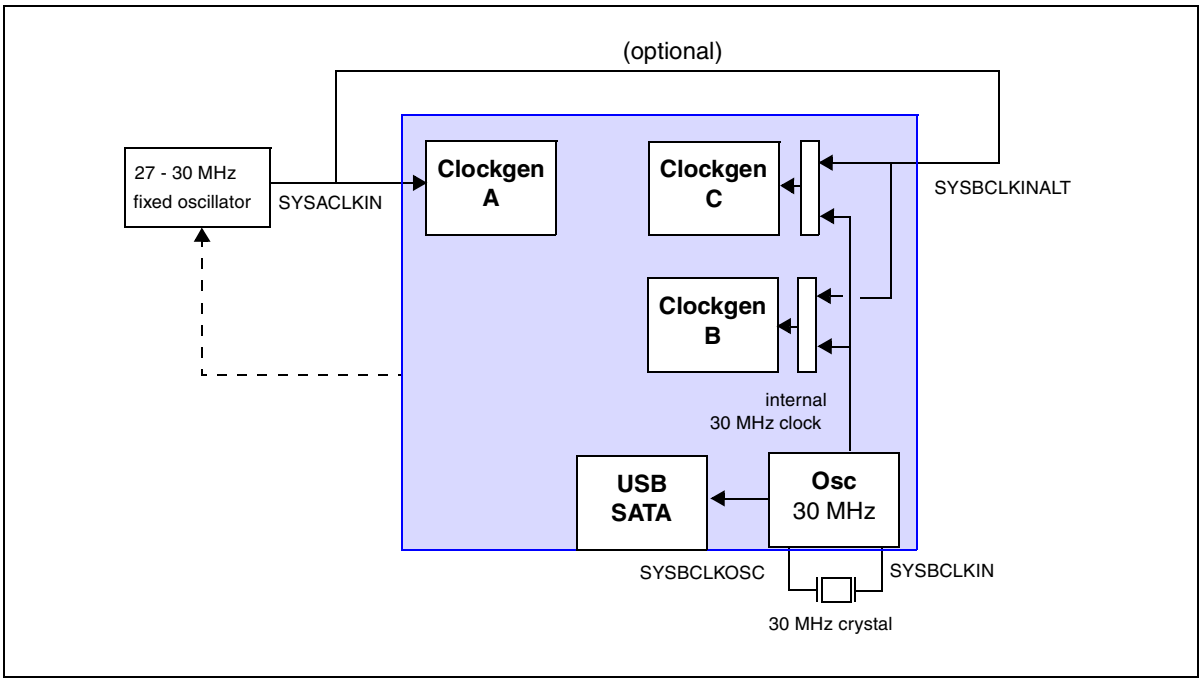


Figure 49: STx7100 clocking scheme with the internal DCXO



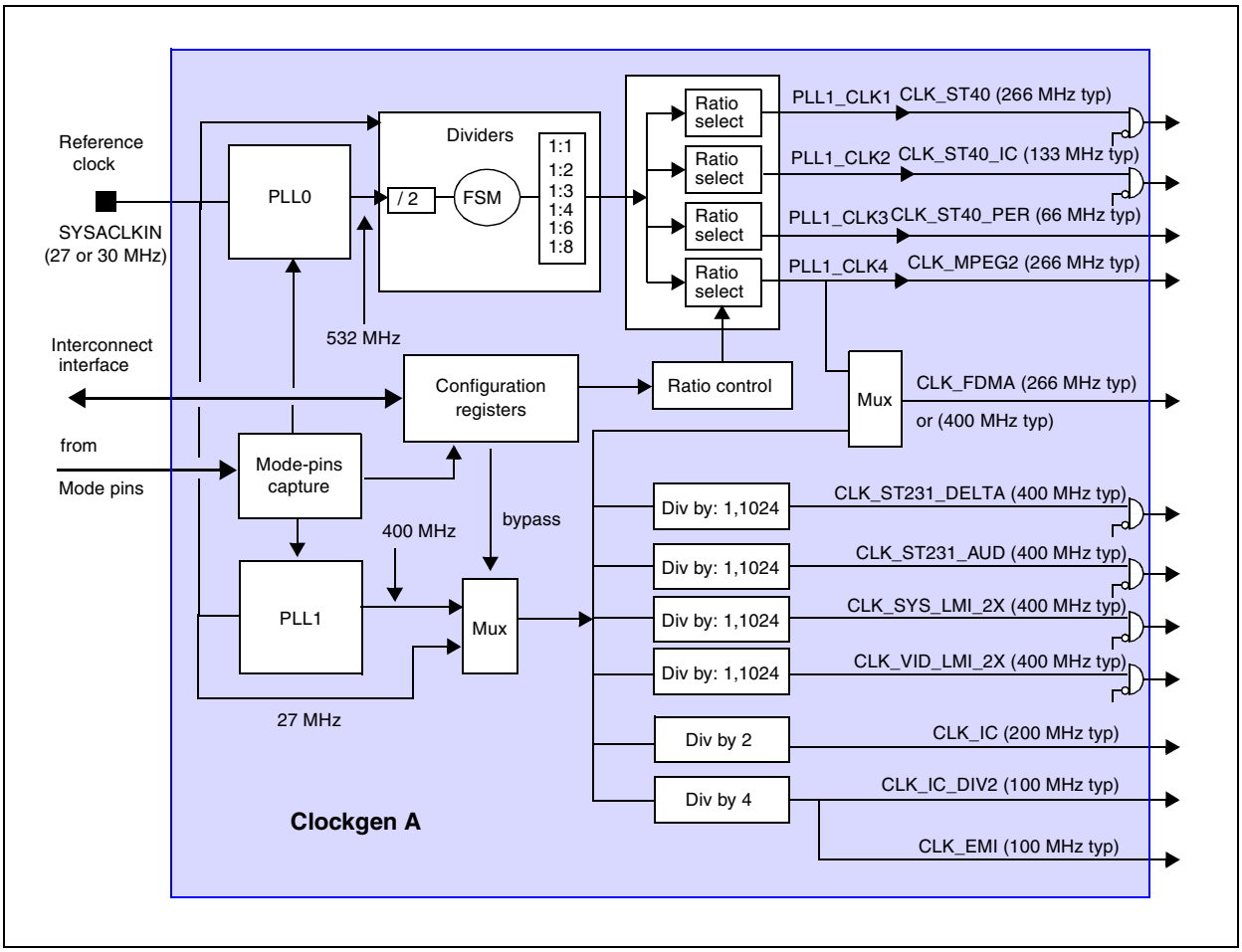
15.2 Clock generator A

Clockgen A includes two PLLs and produces the clocks for the CPUs, DDR interfaces, EMI interface, FDMA and interconnect.

The block diagram of this clockgen is shown in Figure 50.

15.2.1 Block diagram

Figure 50: Clockgen A block diagram



15.2.2 Clock signals

The group A clocks with their typical frequencies are described in Table 65.

The clock frequencies can be changed via clockgen A configuration registers.

Table 65: Clockgen A signals

Clock name	Typical frequency (MHz)	Description
CLK_ST40	266	ST40 core clock
CLK_ST40_IC	133	ST40 bus clock
CLK_ST40_PER	66	ST40 peripheral clock
CLK_FDMA	266	FDMA processing clock
CLK_MPEG2	266	MPEG2 decoder clock
CLK_SYS_LMI_2X	400	2x DDR clock for SYS LMI interface
CLK_VID_LMI_2X	400	2x DDR clock for VID LMI interface
CLK_ST231_AUD	400	Audio decoder ST231 core clock
CLK_ST231_DELTA	400	MPEG4 decoder ST231 core clock
CLK_IC	200	High-speed interconnect clock
CLK_IC_DIV2	100	Low-speed interconnect clock
CLK_EMI	100	EMI clock

15.2.3 Startup configuration

Clockgen A provides default configurations upon reset for its PLL0 and PLL1.

These configurations are defined by four external mode pins: EMIADDR[4:1] (see Chapter 9: *Reset configuration (mode pins)* *on page 89*)

The startup clock frequencies of clock generator A are shown below.

The recommended configuration is to use mode 0 for both PLL0 and PLL1.

Table 66: PLL0 startup configurations

PLL0 mode	EMI_ ADDR [2:1]	PLL0 state	MHz				
			PLL0 output	CLK_ST40	CLK_ST40_IC	CLK_ST40 _PER	CLK_FDMA
0	00	ON	532	266	133	66.5	266
1	01	ON	400	200	100	50	200
2	10	OFF	27	13.5	6.75	3.375	13.5
3	11	ON	600	300	150	75	300

Table 67: PLL1 startup configurations

PLL1 mode	EMI_ ADDR [4:3]	PLL1 state	MHz			
			PLL1 output	CLK_SYS/VID_LMI_2X CLK_ST231_AUD CLK_ST231_DELTA	CLK_IC	CLK_IC_DIV2 CLK_EMI
0	00	ON	400	400	200	100
1	01	ON	333	333	166	83
2	10	ON	266	266	133	66.5
3	11	OFF	27	27	13.5	6.75

15.2.4 Clock frequency change

The clockgen always starts with the configuration defined by the mode pins. Nevertheless, the clocks frequency can be changed with configuration registers. This initial configuration can be read in register MD_STA.

Clockgen programming lock/unlock

To prevent any unwanted clockgen reprogramming, the CGA_LCK register provides a protection mechanism. This register must be written first with the keyword “0xC0DE” to authorize any clockgen registers update. Writing any other value locks all clockgen A registers.

Clock ratio change without changing PLL0 and PLL1

The clock ratio can be changed on the fly without changing the PLL0 setup for the clocks CLK_ST40_IC, CLK_ST40, CLK_ST40_PER, CLK_FDMA and CLK_MPEG2. The clockgen design ensures a glitch-free frequency change.

The clocks CLK_SYS_LMI_2X, CLK_VID_LMI_2X, CLK_ST231_DELTA and CLK_ST231_AUD can also be divided on the fly without any change to the PLL1 setup.

PLL0 frequency change procedure

1. Write 0xC0DE in register CKGA_LCK.
2. Use the CKGA_PLL0_CFG register to move the clock-divider clock input from the PLL0 output to the SYSACLKIN input (set bit 20 to 1).
3. Disable the PLL0 (reset bit CKGA_PLL0_CFG.PLL0_EN).
4. Set new values in register CKGA_PLL0_CFG, fields PLL0_MDIV, PLL0_NDIV and PLL_PDIV.
5. Enable the PLL0 (set bit CKGA_PLL0_CFG.PLL0_EN).
6. Wait until the PLL locks by polling register CKGA_PLL0_STA.
7. Use register CKGA_PLL0_CFG to move the clock-divider clock input from the SYSACLKIN input to the PLL output (set bit 20 to 0).

PLL1 frequency change procedure

1. Use register CKGA_PLL1_BYPASS to bypass the PLL1 output and use the SYSACLKIN input (set bit 1 to 1).
2. Disable the PLL1 (reset bit PLL1_EN of register CKGA_PLL1_CFG).
3. Set new values in register CKGA_PLL1_CFG, fields PLL1_MDIV, PLL1_NDIV and PLL1_PDIV.
4. Enable the PLL1 (set bit CKGA_PLL1_CFG.PLL1_EN).
5. Wait until the PLL locks by reading register CKGA_PLL1_STA.
6. Use register CKGA_PLL1_CFG to use the PLL1 output instead of the SYSACLKIN input (set bits 20 to 0).

15.2.5 Clock slowing

Most group A clocks can be slowed to reduce power consumption without stopping the clocks.

The clocks CLK_SYS_LMI_2X, CLK_VID_LMI_2X, CLK_ST231_AUD and CLK_ST231_DELTA can be divided on the fly by 1024 using the register CKGA_CLK_DIV.

The clocks CLK_ST40_IC, CLK_ST40, CLK_FDMA and CLK_MPEG2 can be reduced using registers CKGA_PLL0_CLK1, CKGA_PLL0_CLK2, CKGA_PLL0_CLK3 and CKGA_PLL0_CLK4.

15.2.6 Clock stopping

The PLL0 clocks CLK_ST40 and CLK_ST40_IC can be stopped using the configuration register CKGA_CLK_EN.

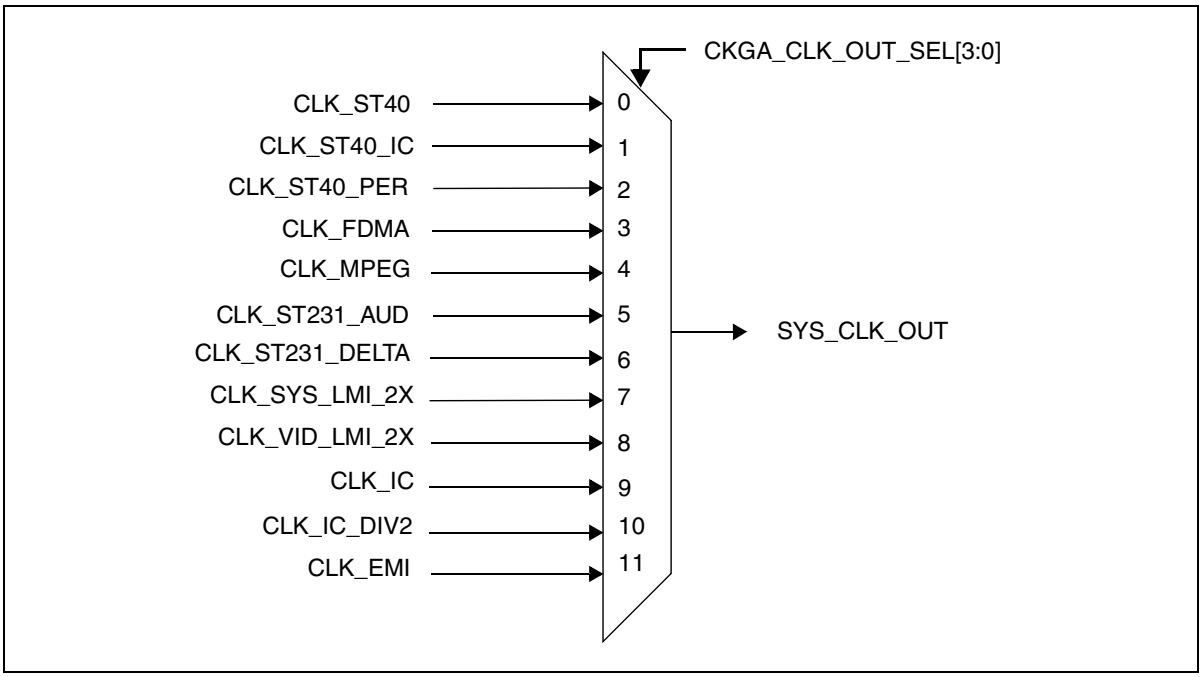
The PLL1 clocks CLK_ST231_AUD, CLK_ST231_DELTA, CLK_SYS_LMI_2X and CLK_VID_LMI_2X can be stopped using the configuration register CKGA_CLK_EN.

15.2.7 Clock observation

Any group A clock can be routed and observed on the SYS_CLK_OUT output pin.

The configuration register CKGA_CLK_OUT_SEL selects the clock to be routed to the pin.

Figure 51: PLL0 and PLL1 clocks observation



15.3 Clock generator B

This clock generator is mainly responsible for generating the clocks used by the video display pipeline. This includes the following units:

- SD and HD displays,
- compositor,
- video output stage (formatters, HDMI, DENC),
- HD and SD Video DACs.

In addition, clockgen B also generates some processing clocks for the following units:

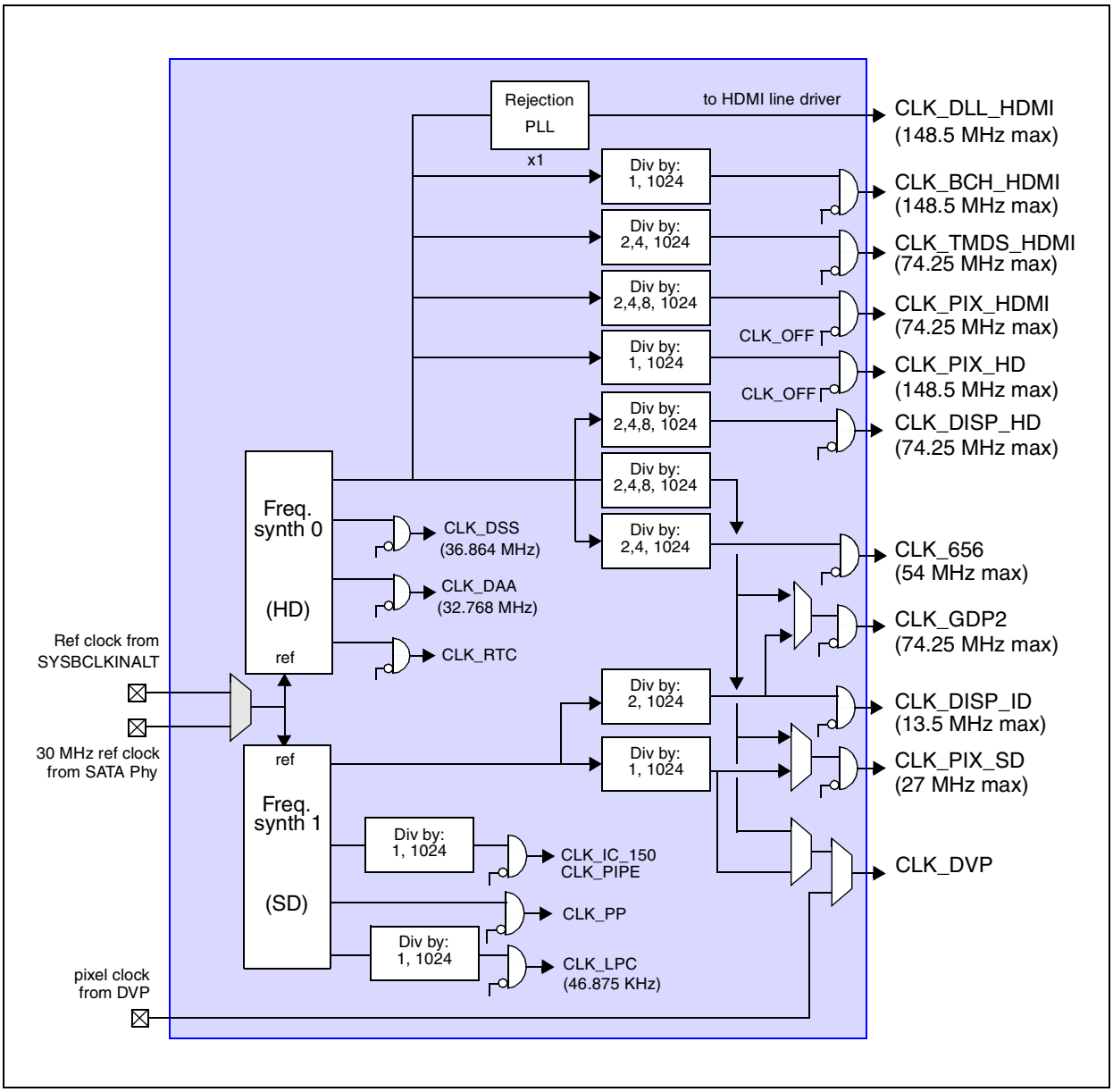
- MPEG2 video decoder,
- comms,
- Delta pre-processor,
- transport subsystem (PTI, CryptoCore, PDES and TSMerger).

Clockgen B comprises two digitally-controlled frequency synthesizers (FS0 and FS1). The reference clock can be either an internal 30 MHz clock signal or a clock connected to the SYSBCLKINALT pin. The selection is done with the configuration register CKGB_REF_CLK_SEL.

Clockgen B also includes a digital clock recovery module to recover the encoder clock.

15.3.1 Block diagram

Figure 52: Clockgen B block diagram



Note: CLK_RTC *has a hardware fixed frequency synthesizer setup. These settings are for a 30 MHz source clock.*

15.3.2 Clock signals

Table 68 lists group B clocks with their maximum frequencies.

Table 68: Clockgen B

Clock name	Maximum frequency (MHz)	Description
CLK_PIX_HD	148.5	HD pixel clock
CLK_PIX_SD	27	SD pixel clock
CLK_DISP_HD	74.25	HD display clock
CLK_DISP_ID	13.5	SD display clock
CLK_GDP2	74.25	GDP2 pixel clock (HD or SD)
CLK_656	54	DVO pixel clock
CLK_PIPE	150	Video pipeline processing clock
CLK_PP	150	Delta (H.264) pre-processor clock
CLK_IC_150	150	Interconnect clock 150 MHz
CLK_DAA	32.768	DAA clock
CLK_DSS	36.864	DSS clock
CLK_LPC		Low power controller (LPC) clock
CLK_TTXT	27	Teletext clock
CLK_SERLZR_HDMI	148.5	HDMI serializer clock
CLK_BCH_HDMI	148.5	HDMI BCH clock
CLK_TMDS_HDMI	74.25	HDMI TMDS clock
CLK_PIX_HDMI	74.25	HDMI pixel clock

The frequency of all clocks is programmable. The video clocks are particularly critical, since they must be setup with respect to the display standard in use.

Table 69 gives some programming examples with respect to the targeted application.

Table 69: Video displays clocking for various application

Clock signal	HD on main, SD on aux (GDP2 on main)	HD on main, SD on aux (GDP2 on aux)	SD progressive on main, SD interleaved on aux (GDP2 on main)	SD progressive on main, SD interleaved on aux (GDP2 on aux)	SD interleaved on main only
CLK_PIX_HD	148.5	148.5	108	108	108
CLK_DISP_HD	74.25	74.25	27	27	13.5
CLK_GDP2	74.25	13.5	27	13.5	13.5
CLK_PIX_SD	27	27	27	27	27 (from HD)
CLK_DISP_ID	13.5	13.5	13.5	13.5	Not used
CLK_656	Not used	Not used	54	54	27
CLK_DISP_HDMI	74.25	74.25	27	27	13.5
CLK_TMDS_HDMI	74.25	74.25	27	27	27
CLK_BCH_HDMI	148.5	148.5	108	108	108
CLK_DLL_HDMI	148.5	148.5	108	108	108

15.3.3 Startup configuration

After the reset phase, clockgen B is by default configured with a 13.5 MHz display clock on both the main and auxiliary video outputs.

Table 70: Clockgen B default configuration

Clock name	Frequency (MHz)	Description
CLK_PIX_HD	108	HD pixel clock
CLK_PIX_SD	27	SD pixel clock
CLK_DISP_HD	13.5	HD display clock
CLK_DISP_ID	13.5	SD display clock
CLK_GDP2	13.5	GDP2 pixel clock (HD or SD)
CLK_656	27	DVO pixel clock
CLK_PIPE	150	Video pipeline processing clock
CLK_PP	150	Delta (H.264) pre-processor clock
CLK_IC_150	150	Interconnect clock 150 MHz
CLK_DAA	32.768	DAA clock
CLK_DSS	36.864	DSS clock
CLK_LPC		Low power controller clock
CLK_TTXT	27	Teletext clock
CLK_SERLZR_HDMI	108	HDMI serializer clock
CLK_BCH_HDMI	108	HDMI BCH clock
CLK_TMDS_HDMI	27	HDMI TMDS clock
CLK_PIX_HDMI	13.5	HDMI pixel clock

15.3.4 Clock frequency change

The clock generator always starts with the default configuration defined above. Nevertheless the frequency synthesizers FS0 and FS1 can be re-configured to produce different frequencies.

Clockgen programming: lock/unlock

To prevent any unwanted clockgen reprogramming, the CKGB_LCK register provides a protection mechanism. This register must be written first with the keyword “0xC0DE” to authorize any clockgen registers update. Writing any other value locks all clockgen B registers.

Clock ratio change without changing FS0 and FS1 programming

The clocks CLK_BCH_HDMI, CLK_TMDS_HDMI, CLK_PIX_HDMI, CLK_PIX_HD, CLK_DISP_HD, CLK_656, CLK_GDP2, CLK_DISP_ID and CLK_PIX_SD are generated from a master clock (from FS0 and FS1) which is then divided by programmable dividers (set to 2, 4, 8 or 1024). These dividers can be redefined on the fly via the register CKGB_DISP_CFG without changing the FS0 and FS1 setup. The clockgen design ensures a glitch-free frequency change.

FS0 and FS1 frequency definition

The frequencies generated by FS0 and FS1 are defined by the registers CKGB_FS0/1_MDx, CKGB_FS0/1_PEx and CKGB_FS0/1_SDIVx and are given by the formula:

$$F_{OUT} = \frac{2^{15} \cdot F_{PLL}}{2^{(sdiv+1)} \times \left[\left(pe \cdot \left(1 + \frac{(md-32)}{32}\right)\right) - \left((pe - 2^{15}) \cdot \left(1 + \frac{(md-31)}{32}\right)\right)\right]}$$

with F_{PLL} = 216 MHz if the reference clock is 27 MHz or F_{PLL} = 240 MHz if the reference clock is 30 MHz.

The MD, PE and PRG_EN parameters can be changed without creating glitches at the frequency synthesizer output. Changing other parameters may cause glitches.

Procedure to change FS0 and FS1 frequency

1. Write 0xC0DE in register CKGB_LCK.
2. Set CKGB_FS0_PRG_ENx to 0.
3. Define CKGB_FS0_MDx, CKGB_FS0_PEx, CKGB_FS0_SDIVx for the targeted frequency.
4. Set CKGB_FS0_PRG_ENx to 1 to validate the new values.
5. Set CKGB_FS0_PRG_ENx to 0.

15.3.5 Clock slowing

Most group B clocks can be divided to reduce power consumption with the register CKGB_CLK_DIV without stopping the clocks.

15.3.6 Clock stopping

The PLL0 clocks CLK_ST40, CLK_ST40_IC can be stopped using the configuration register CKGA_CLK_EN.

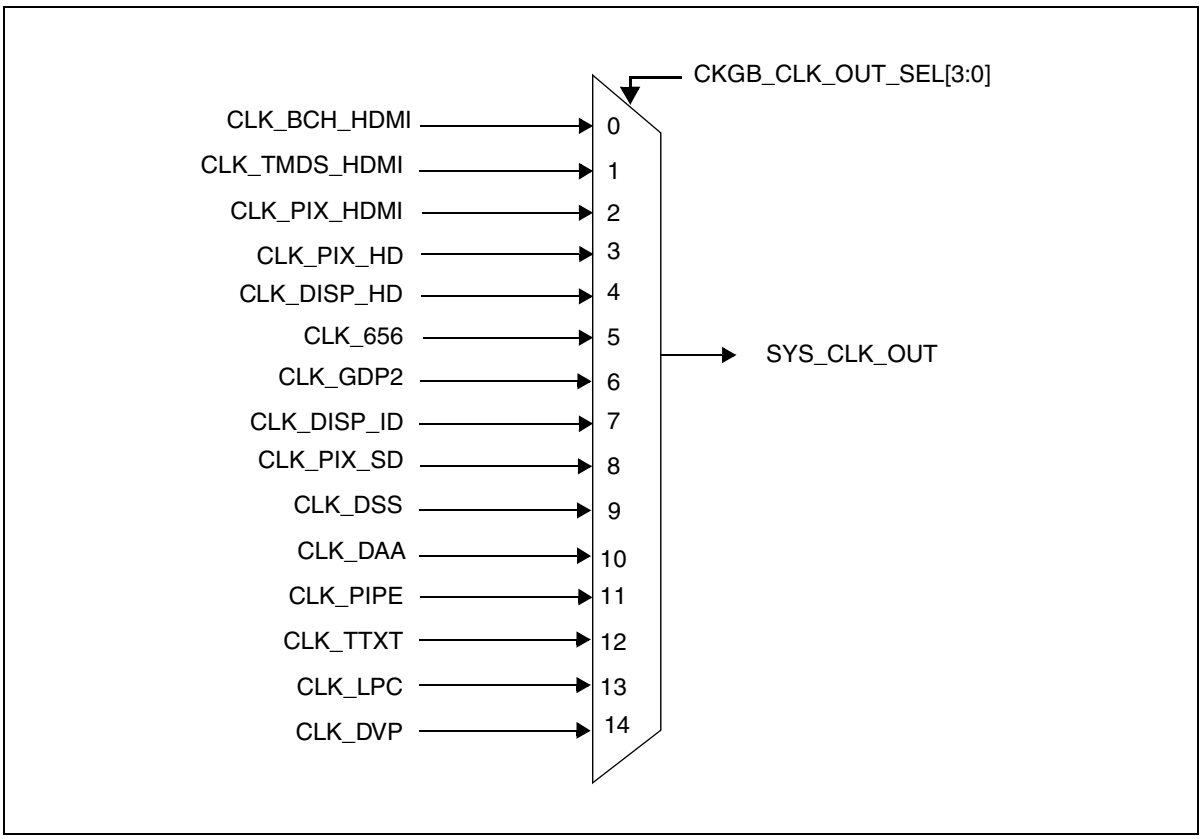
The PLL1 clocks CLK_ST231_AUD, CLK_ST231_DELTA, CLK_SYS_LMI_2X and CLK_VID_LMI_2X can be stopped using the configuration register CKGA_CLK_EN.

15.3.7 Clock observation

Any group B clock can be routed and observed on the alternate PIO5[2] output pin.

The configuration register CKGB_CLKOUT_SEL selects the clock to be routed to the pad.

Figure 53: FS0 and FS1 clocks observation



15.3.8 Clock recovery

The clock recovery is a mechanism to adjust the locally generated clocks with the encoder clock referenced in the PCR (program counter reference) located in the adaptation field of the incoming transport stream.

The STx7100 generates three local clocks from internal frequency synthesizers using a fixed and stable 27 or 30 MHz reference clock.

These three clocks are:

- CLK_PIX_SD used for SD display generated from FS0,
- CLKL_PIX_HD used for HD display generated from FS1,
- CLK_PCM used for audio output generated from FS2.

Clock recovery principle

The mechanism assumes that these three clocks are related one to the others.

The recovery is done as usual for the CLK_PIX_SD (generated from the frequency synthesizer FS0) by comparing the 42-bits PCR value located in the adaptation field of the stream with the local STC (system timer counter clocked by CLK_PIX_SD) value when a packet arrived. This generates a potential correction that is applied to the frequency synthesizer FS0 to adjust the CLK_PIX_SD clock. The frequency synthesizer is programmed with new setup values to slow or accelerate the clock.

For audio PCM clock recovery, two counters are used:

- a PCM free-running counter clocked by the audio frequency synthesizer FS2,
- a reference counter clocked by the SD video frequency synthesizer FS0. The maximum value of this counter is programmable defining the time interval between two consecutive resets. This counter is used as a time-base.

When the reference counter resets, the values of the free-running counter clocked by CLK_PCM is captured into a readable register. This event generates an interrupt to the CPU (CRU_IRQ). The CPU reads the value and compares it with the previously captured value. The difference between two adjacent values gives an indication of the correction to apply to the PCM audio frequency synthesizer FS2.

The decision to correct the frequency synthesizers setup is under the control of the software.

The same principle applies for the recovery of the CLK_PIX_HD. A free-running counter is clocked from the HD video frequency synthesizer FS1. The same reference counter is used. When this counter resets, then the output of the free-running counter clocked by CLK_PIX_HD is captured into a readable register.

Clock recovery commands

The clock recovery module accepts a single command defined by register field CKGB_CRU_CMD.ID.

When set to 1, this command loads the reference counter with the value defined in the register CKGB_REF_MAX, resets the PCM counter and HD Video counter. Resetting this command to 0 starts the counters.

Clock recovery interrupt

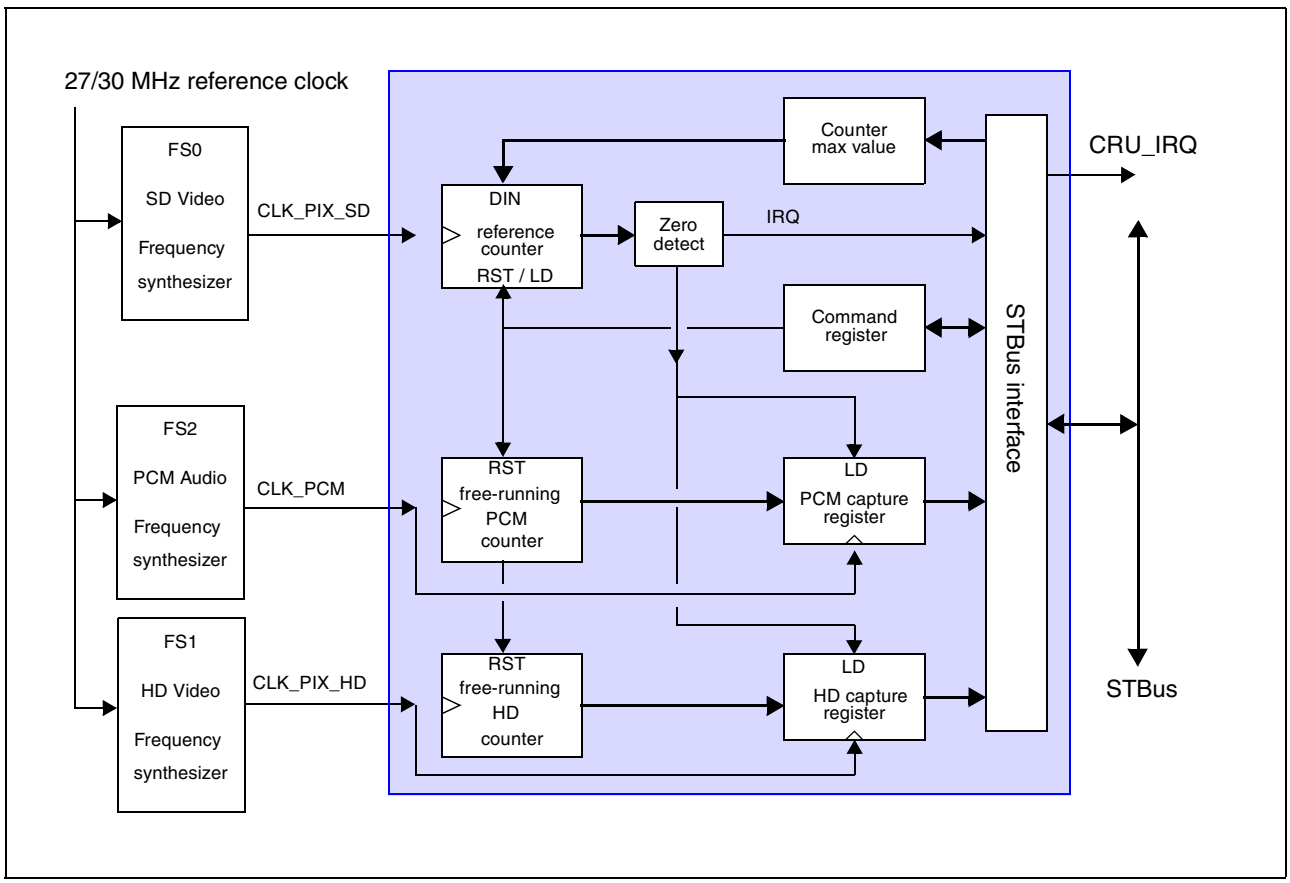
An interrupt line is available, generated by the clock recovery unit. The interrupt is asserted when the reference counter clocked at CLK_PIX_SD resets.

This interrupt status is captured in the CMD register and is cleared only when a 1 is written to bitfield CKGB_CRU_CMD.INT. Writing 0 enables new interrupts.

Block diagram

The clock recovery unit (CRU) block diagram is described in Figure 54.

Figure 54: Clock recovery unit block diagram



16 Clock registers

16.1 Summary

Register addresses are provided as:

ClkABaseAddress + offset or *ClkBBaseAddress* + offset.

The *ClkABaseAddress* is:

0x1921 3000

The *ClkBBaseAddress* is:

0x1900 0000.

Table 71: Clock generator A register summary

Name	Function	Offset	Reset value	Type
CKGA_LCK	Clockgen A lock	0x000	0x0000 0000	R/W
CKGA_MD_STA	Mode pins status	0x004	from mode pins	RO
CKGA_PLL0_CFG	PLL0 configuration	0x008	from mode pins	Mixed
Reserved	Reserved	0x00C	-	-
CKGA_PLL0_LCK_STA	PLL0 lock status	0x010	Undefined	RO
CKGA_PLL0_CLK1	CLK_ST40 ratio	0x014	0x0000 0000	R/W
CKGA_PLL0_CLK2	CLK_ST40_IC ratio	0x018	0x0000 0001	
CKGA_PLL0_CLK3	CLK_ST40_PER ratio	0x01C	0x0000 0000	
CKGA_PLL0_CLK4	CLK_FDMA ratio	0x020	0x0000 0000	
CKGA_PLL1_CFG	PLL1 configuration	0x024	From mode pins	
Reserved	Reserved	0x028	-	-
CKGA_PLL1_LCK_STA	PLL1 lock status	0x02C	Undefined	RO
CKGA_CLK_DIV	Clocks division	0x030	0x0000 0000	R/W
CKGA_CLK_EN	Clocks enabling/stopping	0x034	0x0000 003f	
CKGA_CLKOUT_SEL	SYSCLKOUT clock source selection	0x038	0x0000 0000	
CKGA_PLL1_BYPASS	PLL1 bypass and FDMA clock source	0x03C	From mode pins	

Table 72: Clock generator B register summary

Name	Function	Offset	Reset value	Type
Clock recovery				
CKGB_REF_MAX	Max value for CRU reference counter	0x000	0x0000 0000	R/W
CKGB_CMD	CRU command register	0x004	0x0000 0001	
CKGB_CPT_PCM	CRU captured audio PCM counter value	0x008	0x0000 0000	
CKGB_CPT_HD	CRU captured HD video counter value	0x00C	0x0000 0000	
Configuration				
CKGB_LCK	Clockgen B lock	0x010	0x0000 0000	R/W
CKGB_FS0_CFG	Global FS0 parameters	0x014	0x0000 0018	
CKGB_FS0_MD1	FS0 channel 1 coarse selection	0x018	0x0000 0011	
CKGB_FS0_PE1	FS0 channel 1 fine selection	0x01C	0x0000 1C72	
CKGB_FS0_PRG_EN1	FS0 channel 1 programming enable	0x020	0x0000 0001	
CKGB_FS0_SDIV1	FS0 channel 1 output clock divider	0x024	0x0000 0001	

Table 72: Clock generator B register summary

Name	Function	Offset	Reset value	Type
CKGB_FS0_MD2	FS0 channel 2 coarse selection	0x028	0x0000 001A	R/W
CKGB_FS0_PE2	FS0 channel 2 fine selection	0x02C	0x0000 7AAB	
CKGB_FS0_PRG_EN2	FS0 channel 2 programming enable	0x030	0x0000 0001	
CKGB_FS0_SDIV2	FS0 channel 2 output clock divider	0x034	0x0000 0002	
CKGB_FS0_MD3	FS0 channel 3 coarse selection	0x038	0x0000 001D	
CKGB_FS0_PE3	FS0 channel 3 fine selection	0x03C	0x0000 5A00	
CKGB_FS0_PRG_EN3	FS0 channel 3 programming enable	0x040	0x0000 0001	
CKGB_FS0_SDIV3	FS0 channel 3 output clock divider	0x044	0x0000 0002	
Reserved	-	0x048 - 0x054	-	-
CKGB_FS0_PWR_DN	FS1 digital part power-down/up	0x058	0x0000 0077	R/W
CKGB_FS1_CFG	Global FS1 parameters	0x05C	0x0000 0018	
CKGB_FS1_MD1	FS1 channel 1 coarse selection	0x060	0x0000 0011	
CKGB_FS1_PE1	FS1 channel 1 fine selection	0x064	0x0000 1C72	
CKGB_FS1_PRG_EN1	FS1 channel 1 programming enable	0x068	0x0000 0001	
CKGB_FS1_SDIV1	FS1 channel 1 output clock divider	0x06C	0x0000 0003	
CKGB_FS1_MD2	FS1 channel 2 coarse selection	0x070	0x0000 0019	
CKGB_FS1_PE2	FS1 channel 2 fine selection	0x074	0x0000 3333	
CKGB_FS1_PRG_EN2	FS1 channel 2 programming enable	0x078	0x0000 0001	
CKGB_FS1_SDIV2	FS1 channel 2 output clock divider	0x07C	0x0000 0000	
CKGB_FS1_MD3	FS1 channel 3 coarse selection	0x080	0x0000 0011	
CKGB_FS1_PE3	FS1 channel 3 fine selection	0x084	0x0000 1C72	
CKGB_FS1_PRG_EN3	FS1 channel 3 programming enable	0x088	0x0000 0001	
CKGB_FS1_SDIV3	FS1 channel 3 output clock divider	0x08C	0x0000 0003	
CKGB_FS1_MD4	FS1 channel 4 coarse selection	0x090	0x0000 0013	
CKGB_FS1_PE4	FS1 channel 4 fine selection	0x094	0x0000 0000	
CKGB_FS1_PRG_EN4	FS1 channel 4 programming enable	0x098	0x0000 0001	
CKGB_FS1_SDIV4	FS1 channel 4 output clock divider	0x09C	0x0000 0002	
CKGB_FS1_PWR_DN	FS1 digital part power-down/up	0x0A0	0x0000 00FF	
CKGB_DISP_CFG	Video display clocks configuration	0x0A4	0x0000 0469	
CKGB_CLK_SRC	Clock source selection	0x0A8	0x0000 0006	
CKGB_CLK_DIV	Individual clocks division	0x0AC	0x0000 0200	
CKGB_CLK_EN	Individual clocks enable	0x0B0	0x0000 3FFF	
CKGB_CLKOUT_SEL	Clocks observation selection	0x0B4	0x0000 0000	
CKGB_REF_CLK_SEL	Reference clock source selection	0x0B8	0	

16.2 Clock generator A

CKGA_LCK Clockgen A lock

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																LCK															

Address: *ClkABaseAddress* + 0x000
Type: R/W
Reset: 0
Description: Must be written with the value 0xC0DE to write to any clockgen A configuration register.

CKGA_MD_STA Mode pins status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												MD			

Address: *ClkABaseAddress* + 0x004
Type: RO
Reset: Captured during reset
Description: Reports the status of the mode pins captured during the power-on-reset.

[31:4] **Reserved**

[3:2] **MD[3:2]**: PLL1 startup configuration

[1:0] **MD[1:0]**: PLL0 startup configuration

CKGA_PLL0_CFG PLL0 configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											PLL0_BYPASS	PLL0_EN	PLL0_PDIV			PLL0_NDIV								PLL0_MDIV							

Address: *ClkABaseAddress* + 0x008

Type: R/W

Reset: See Table 73 and Table 74.

Description: Defines the PLL0 setup. The reset values are function of the mode pins values captured during the reset.

[31:21] **Reserved**
Bit 21 must be set to 0.

[20] **PLL0_BYPASS**
0: PLL0 output clock is used
1: PLL0 is bypass - SYSACLIN clock is used

[19] **PLL0_EN**: PLLO enable
0: PLL0 is in power-down mode
1: PLL0 is enabled

[18:16] **PLL0_PDIV**: '2^P' post-divider ratio - from 1 to 32

[15:8] **PLL0_NDIV**: N feedback divider ratio - from 3 to 255

[7:0] **PLL0_MDIV**: M pre-divider ratio - from 1 to 255

The reset values of CKGA_PLL0_CFG depends on MD[1:0], as described in Table 73 and Table 74.

Table 73: PLL0_CFG reset values

MD[1:0]	MDIV	NDIV	PDIV	Enable	PLL0 CLKOUT (MHz)
00	0x06 (6)	0x3B (59)	0x0	1	531
01	0x1B (27)	0xC8 (200)	0x0	1	400
10	0x06 (6)	0x3B (59)	0x0	0	27 (PLL bypassed)
11	0x09 (9)	0x64 (100)	0x0	1	600

Table 74: PLL0_CFG reset values

MD[1:0]	bit 21	PLL0 bypass	PLL0 enable	Description
00	0	0	1	PLL0_CLKOUT selected - PLL0 on
01	0	0	1	PLL0_CLKOUT selected - PLL0 on
10	0	1	0	SYSA_CLK_IN selected - PLL0 off
11	0	0	1	PLL0_CLKOUT selected - PLL0 on

The PLL0 frequency is given by the formula:

$$F_{PLL0} = \frac{2 \times N \times F_{sysaclkin}}{M \times 2^{P}}$$

with:

N in the interval [3, 255],
M in the interval [1, 255],
P in the interval [1, 32].

CKGA_PLL0_LCK_STA PLL0 lock status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															LCK

Address: *ClkABaseAddress* + 0x010
Type: RO
Reset: Undefined
Description: Describes the lock status of the PLL0.
0: PLL0 is unlocked
1: PLL0 is locked

CKGA_PLL0_CLK1 CLK_ST40 ratio

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													RATIO		

Address: *ClkABaseAddress* + 0x014
Type: R/W
Reset: 0
Description: Defines the clock ratio of the CLK_ST40 clock (PLL1_CLK1) with respect to the PLL0 output clock.
000: Ratio 1:1
001: Ratio 1:2
010: Ratio 1:3
011: Ratio 1:4
100: Ratio 1:6
101: Ratio 1:8
110: Ratio 1:1
111: Ratio 1:1

CKGA_PLL0_CLK2 CLK_ST40 IC ratio

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													RATIO		

Address: *ClkABaseAddress* + 0x018

Type: R/W

Reset: 0

Description: Defines the clock ratio of the CLK_ST40_IC clock (PLL1_CLK2) with respect to the PLL0 output clock (MAIN_CLOCK).

000: Ratio 1:1
001: Ratio 1:2
010: Ratio 1:3
011: Ratio 1:4
100: Ratio 1:6
101: Ratio 1:8
110: Ratio 1:2
111: Ratio 1:2

CKGA_PLL0_CLK3 CLK_ST40_PER ratio

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													RATIO		

Address: *ClkABaseAddress* + 0x01C

Type: R/W

Reset: 0

Description: Defines the clock ratio of the CLK_ST40_PER clock (PLL1_CLK3) with respect to the PLL0 output clock (MAIN_CLOCK).

000: Ratio 1:4
001: Ratio 1:2
010: Ratio 1:4
011: Ratio 1:4
100: Ratio 1:6
101: Ratio 1:8
110: Ratio 1:4
111: Ratio 1:4

CKGA_PLL0_CLK4 CLK_FDMA ratio

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													RATIO		

Address: *ClkABaseAddress* + 0x020

Type: R/W

Reset: 0

Description: Defines the clock ratio of the CLK_FDMA clock (PLL1_CLK4[1]) with respect to the PLL0 output clock (MAIN_CLOCK).

000: Ratio 1:1
001: Ratio 1:2
010: Ratio 1:3
011: Ratio 1:4
100: Ratio 1:6
101: Ratio 1:8
110: Ratio 1:3
111: Ratio 1:3

1. Also used for CLK_MPEG2

CKGA_PLL1_CFG PLL1 configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PLL1_EN	PLL1_PDIV			PLL1_NDIV								PLL1_MDIV							

Address: *ClkABaseAddress* + 0x024

Type: R/W

Reset: See Table 75

Description: Defines PLL1 setup. The reset values are functions of the mode pins values captured during the reset.

[31:20] **Reserved**

[19] **PLL1_EN**: PLL1 enable
0: PLL is in power-down mode.
1: PLL is on.

[18:16] **PLL1_PDIV**: '2^P' post-divider ratio - from 1 to 32

[15:8] **PLL1_NDIV**: N feedback divider ratio - from 1 to 255

[7:0] **PLL1_MDIV**: M pre-divider ratio - from 1 to 255

The reset values of PLL1_CFG fields depends on MD[2:0] as described in Table 75.

Table 75: PLL1_CFG startup values

MD[3:2]	MDIV	NDIV	PDIV	Enable	PLL1_CLKOUT (MHz)
00	0x1B (27)	0xC8	0x0	1	400
01	0x03 (3)	0x25	0x1	1	333
10	0x1B (27)	0x85	0x0	1	266
11	0x06 (6)	0x3B	0x0	0	27 (PLL bypassed)

The PLL1 frequency is given by the formula:

$$F_{PLL1} = \frac{2 \times N \times F_{sysaclkin}}{M \times P}$$

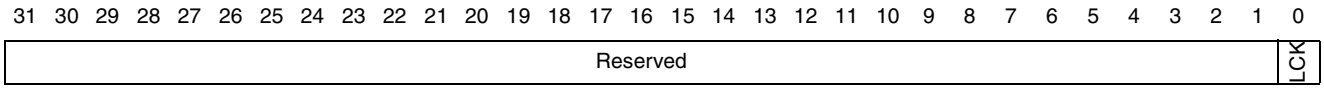
with:

N in the interval [3, 255],

M in the interval [1, 255],

P in the interval [1, 32].

CKGA_PLL1_LCK_STA PLL1 lock status



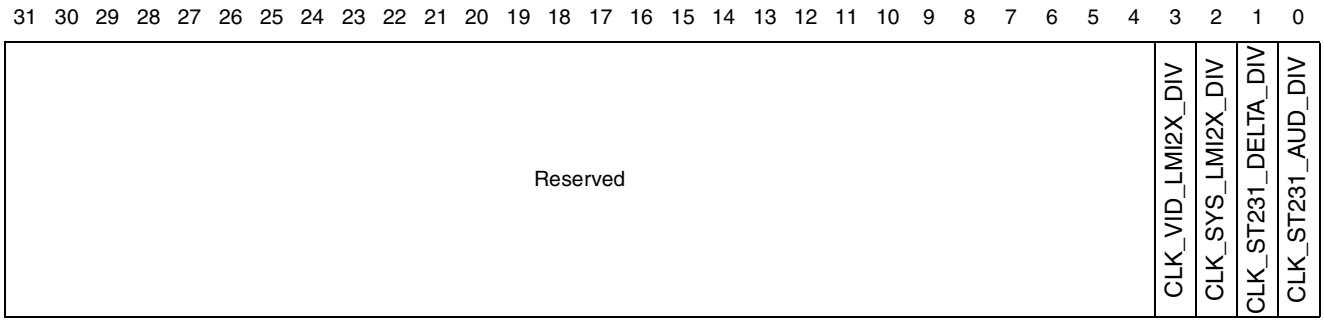
Address: *ClkABaseAddress* + 0x02C

Type: RO

Reset: Undefined

Description: Describes the lock status of PLL1.
0: PLL1 is unlocked
1: PLL1 is locked

CKGA_CLK_DIV Clocks division



Address: *ClkABaseAddress* + 0x030

Type: R/W

Reset: 0

Description: Allows some of the clocks generated from the PLL1 to be reduced in frequency. Can be used to minimize power consumption without stopping the clocks.

[31:4] **Reserved**

[3] **CLK_VID_LMI2X_DIV**
0: Nominal frequency value
1: CLK_VID_LMI_2X clock is divided by 1024

[2] **CLK_SYS_LMI2X_DIV**
0: Nominal frequency value
1: CLK_SYS_LMI_2X clock is divided by 1024

[1] **CLK_ST231_DELTA_DIV**
0: Nominal frequency value
1: CLK_ST231_DELTA clock is divided by 1024

[0] **CLK_ST231_AUD_DIV**
0: Nominal frequency value
1: CLK_ST231_AUD clock is divided by 1024

CKGA_CLK_EN Clocks enabling/stopping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CLK_LMI2X_VID_EN	CLK_LMI2X_SYS_EN	CLK_ST40_IC_EN	CLK_ST40_EN	CLK_ST231_DELTA_EN	CLK_ST231_AUD_EN

Address: *ClkABaseAddress* + 0x034

Type: R/W

Reset:

Description: Switches on or off a number of clocks generated from PLL0 and PLL1.

[31:6] **Reserved**

[5] **CLK_LMI2X_VID_EN**: CLK_LMI_2X_VID clock enable
0: Stop 1: Enable

[4] **CLK_LMI2X_SYS_EN**: CLK_LMI_2X_SYS clock
0: Stop 1: Enable

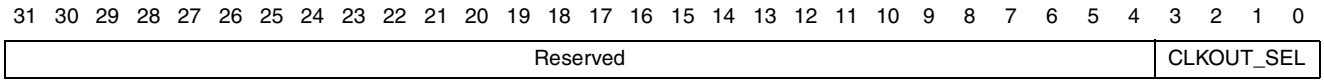
[3] **CLK_ST40_IC_EN**: CLK_ST40_IC clock
0: Stop 1: Enable

[2] **CLK_ST40_EN**: CLK_ST40 clock enable
0: Stop 1: Enable

[1] **CLK_ST231_DELTA_EN**: CLK_ST231_DELTA clock enable
0: Stop 1: Enable

[0] **CLK_ST231_AUD_EN**: CLK_ST231_AUD clock enable
0: Stop 1: Enable

CKGA_CLKOUT_SEL SYSCLKOUT clock source selection



Address: *ClkABaseAddress* + 0x038

Type: R/W

Reset: 0

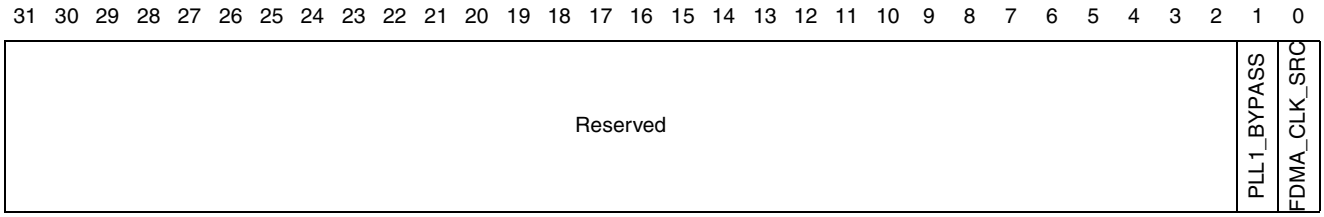
Description: Selects which clock of clockgen A to be routed to the SYSCLKOUT pin.

Table 76: SYSCLOUT pin source

CLKOUT_SEL	SYSCLKOUT
0000	CLK_ST40
0001	CLK_ST40_IC
0010	CLK_ST40_PER
0011	CLK_FDMA
0100	CLK_MPEG
0101	CLK_ST231_AUD
0110	CLK_ST231_DELTA
0111	CLK_SYS_LMI_2X

CLKOUT_SEL	SYSCLKOUT
1000	CLK_VID_LMI_2X
1001	CLK_IC
1010	CLK_IC_DIV2
1011	CLK_EMI
1100 to 1111	Reserved
1101	
1110	
1111	

CKGA_PLL1_BYPASS PLL1 bypass and FDMA clock source



Address: *ClkABaseAddress* + 0x03C

Type: R/W

Reset: See below

Description: Provides a control bit (PLL1_BYPASS) to bypass the PLL1 output clock and to use the clock driving the SYSACLKIN input pin (usually 27 MHz). This feature must be used when changing the PLL1 configuration.

The bit FDMA_CLK_SRC allows selection of either PLL0 (default) or PLL1 as a clock source.

[31:2] **Reserved**

[1] **PLL1_BYPASS**
0: PLL1_CLKOUT is used
1: PLL1_CLKOUT is bypassed and SYSA_CLK_IN clock is used instead
Reset: From mode pins

[0] **FDMA_CLK_SRC**
0: PLL0 is the PLL source for CLK_FDMA
1: PLL1 is the source PLL for CLK_FDMA
Reset: 0

16.3 Clock generator B

CKGB_REF_MAX Max value for CRU reference counter

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REF_MAX																															

Address: *ClkBBaseAddress* + 0x000

Type: R/W

Reset: 0

Description: Load value of the reference counter for the video clock recovery. The reference counter resets each time it reaches this value.

CKGB_CRU_CMD CRU command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														INT	LD

Address: *ClkBBaseAddress* + 0x004

Type: R/W

Reset: 0x01

Description: Holds the load command that can be applied to the clock recovery unit.

When the load command is set to ‘1’ the reference counter is loaded with REF_MAX, PCM and HD counters are reset to zero.

When the load command is reset to ‘0’, this starts the count.

This register also holds the interrupt status. When written to, the interrupt is cleared.

The counters and interrupt generation can be stopped by setting LD to ‘1’ after a clear interrupt has occurred.

[31:2] **Reserved**

[1] **INT**

Read: get the interrupt status

1: interrupt has occurred and the counters can be read

0: no interrupt

Write:

1: clear the current interrupt

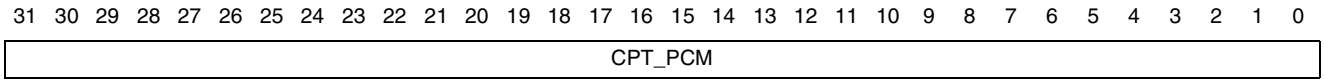
0: allows the next interrupt to occur

[0] **LD**

0: start the counters

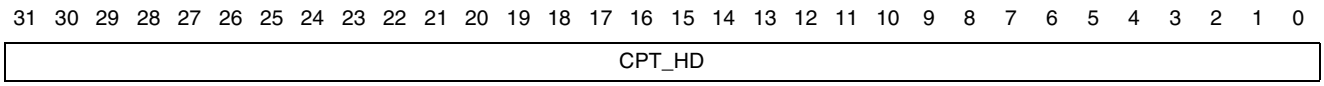
1: reset the counters and load the REF_MAX value

CKGB_CPT_PCM CRU captured audio PCM counter value



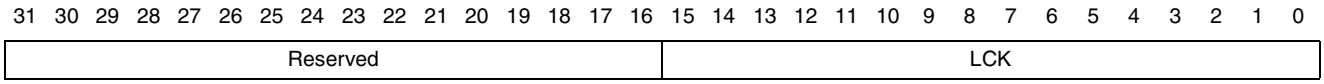
Address: *ClkBBaseAddress* + 0x008
Type: R/W
Reset: 0
Description: Value captured at the output of the PCM counter when an interrupt has occurred.

CKGB_CPT_HD CRU captured HD video counter value



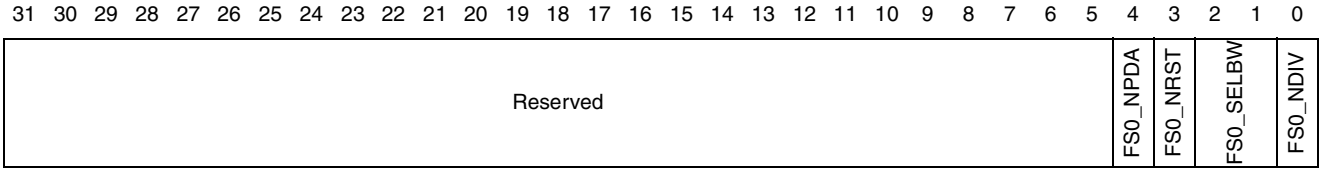
Address: *ClkBBaseAddress* + 0x00C
Type: R/W
Reset: 0
Description: Value captured at the output of the HD counter when an interrupt has occurred.

CKGB_LCK Clockgen B lock



Address: *ClkBBaseAddress* + 0x010
Type: R/W
Reset: 0
Description: Must be written with the value 0xC0DE to access any clockgen B configuration register.

CKGB_FS0_CFG Global FS0 parameters



Address: *ClkBBaseAddress* + 0x014

Type: R/W

Reset: 0x0018

Description: Defines the FS0 frequency synthesizer global parameters.

[31:5] **Reserved**

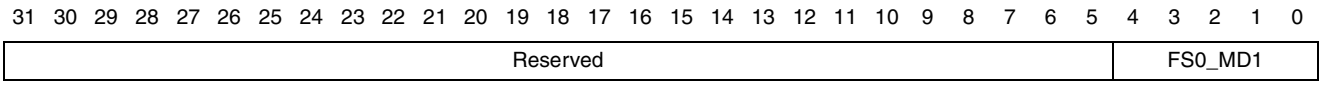
[4] **FS0_NPDA**: Analog power down mode
0: Analog power down
1: Analog power up

[3] **FS0_NRST**: Software reset
0: Reset active
1: Running

[2:1] **FS0_SELBW**: Bandwidth selection for ref clock noise reduction
00: Good reference (bandwidth = 2.4 MHz)
01: Very bad reference (bandwidth = 0.8 MHz)
10: Bad reference (bandwidth = 1.2 MHz)
11: Very good reference (bandwidth = 4 MHz)

[0] **FS0_NDIV**: Select the input frequency, if NDIV is set to
0: Ref clock frequency must be 27/30 MHz
1: Ref clock frequency must be 54/60 MHz

CKGB_FS0_MD1 FS0 channel 1 coarse selection



Address: *ClkBBaseAddress* + 0x018

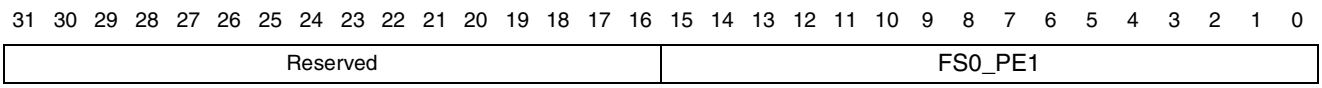
Type: R/W

Reset: 0x0011

Description: Defines the MD1 parameter (coarse selection) for the frequency synthesizer FS0 channel 1.

The MD1 reset value is defined to generate a frequency of 108 MHz on CLK_PIX_HD.

CKGB_FS0_PE1 FS0 channel 1 fine selection



Address: *ClkBBaseAddress* + 0x01C

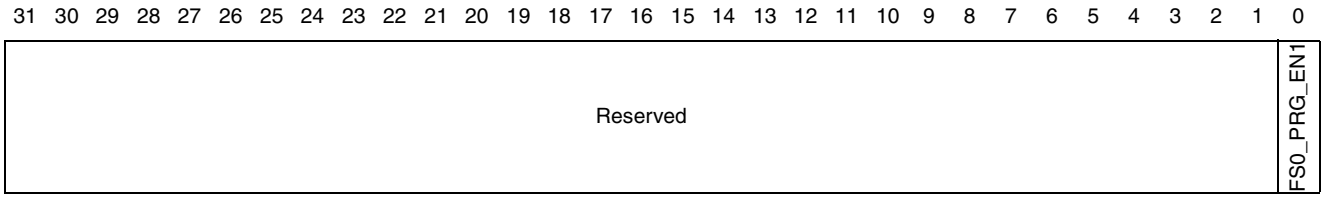
Type: R/W

Reset: 0x1C72

Description: Defines the PE1 parameter (fine selection) for the frequency synthesizer FS0 channel 1.

The PE1 reset value is defined to generate a frequency of 108 MHz on CLK_PIX_HD.

CKGB_FS0_PRG_EN1 FS0 channel 1 programming enable



Address: *ClkBBaseAddress* + 0x020

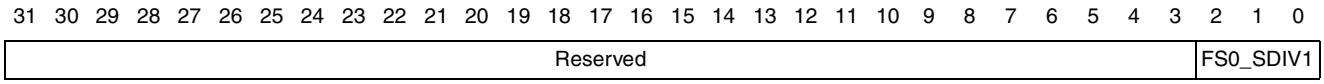
Type: R/W

Reset: 0x01

Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS0 channel 1. This parameter must be set to 1 to take into account a new configuration.

0: PE1 and MD1 parameters are ignored
1: PE1 and MD1 parameters are taken into account

CKGB_FS0_SDIV1 FS0 channel 1 output clock divider



Address: *ClkBBaseAddress* + 0x024

Type: R/W

Reset: 0x01

Description: Defines the SDIV1 parameter (output divider control) for the frequency synthesizer FS0 channel 1, from 1 to 256.

The SDIV1 reset value is defined to generate a frequency of 108 MHz on CLK_PIX_HD.

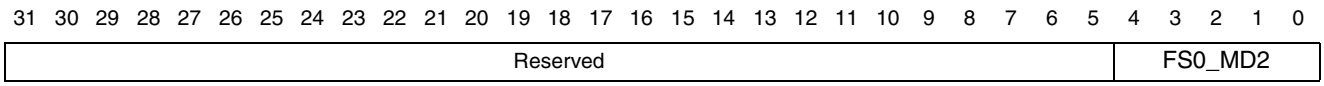
To generate a frequency of 148.5 MHz the following parameters can be used:

SDIV1 = 0x0

MD1 = 0x19

PE1 = 0x121A

CKGB_FS0_MD2 FS0 channel 2 coarse selection



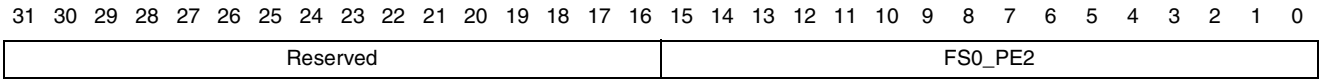
Address: *ClkBBaseAddress* + 0x028

Type: R/W

Reset: 0x001A

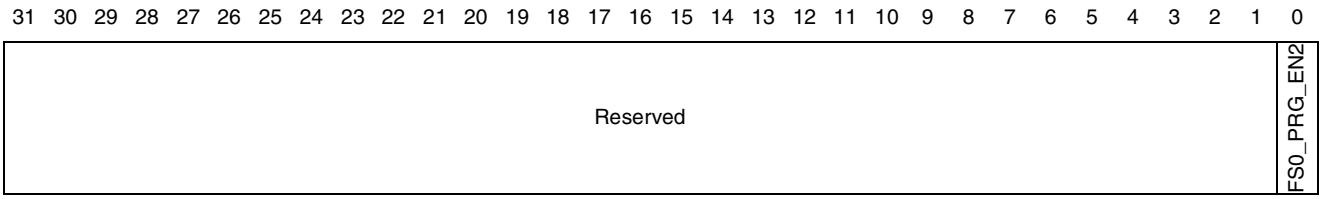
Description: Defines the MD2 parameter (coarse selection) for the frequency synthesizer FS0 channel 2.

CKGB_FS0_PE2 FS0 channel 2 fine selection



Address: *ClkBBaseAddress* + 0x02C
Type: R/W
Reset: 0x7AAB
Description: Defines the MD2 parameter (fine selection) for the frequency synthesizer FS0 channel 2.

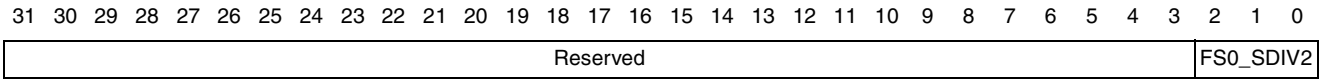
CKGB_FS0_PRG_EN2 FS0 channel 2 programming enable



Address: *ClkBBaseAddress* + 0x030
Type: R/W
Reset: 0x01
Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS0 channel 2. This parameter must be set to 1 to take into account a new configuration.

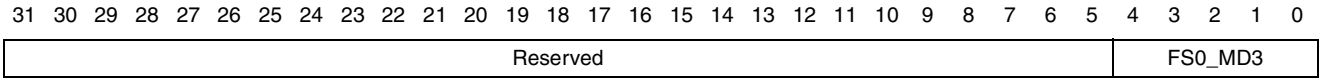
0: PE2 and MD2 parameters are ignored
1: PE2 and MD2 parameters are taken into account

CKGB_FS0_SDIV2 FS0 channel 2 output clock divider



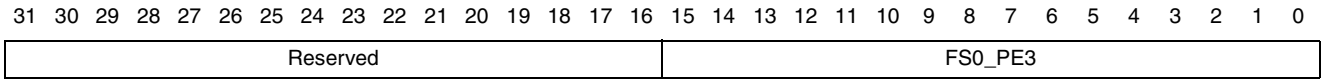
Address: *ClkBBaseAddress* + 0x034
Type: R/W
Reset: 0x02
Description: Defines the SDIV2 parameter (output divider control) for the frequency synthesizer FS0 channel 2; allowed values are from 2 to 256.

CKGB_FS0_MD3 FS0 channel 3 coarse selection



Address: *ClkBBaseAddress* + 0x038
Type: R/W
Reset: 0x1D
Description: Defines the MD3 parameter (coarse selection) for the frequency synthesizer FS0 channel 3.

CKGB_FS0_PE3 FS0 channel 3 fine selection



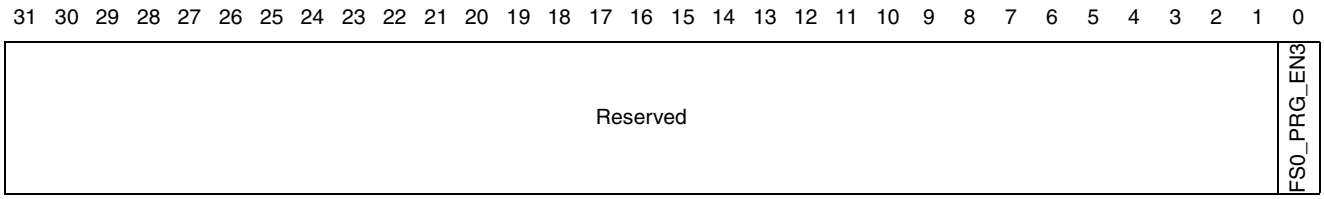
Address: *ClkBBaseAddress* + 0x03C

Type: R/W

Reset: 0x5A00

Description: Defines the MD3 parameter (fine selection) for the frequency synthesizer FS0 channel 3.

CKGB_FS0_PRG_EN3 FS0 channel 3 programming enable



Address: *ClkBBaseAddress* + 0x040

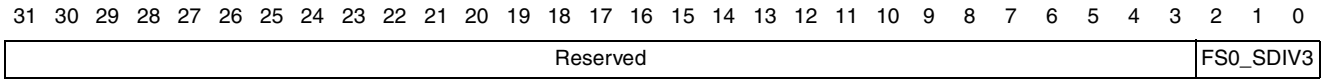
Type: R/W

Reset: 0x01

Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS0 channel 3. This parameter must be set to 1 to take into account a new configuration.

0: PE3 and MD3 parameters are ignored
1: PE3 and MD3 parameters are taken into account

CKGB_FS0_SDIV3 FS0 channel 3 output clock divider



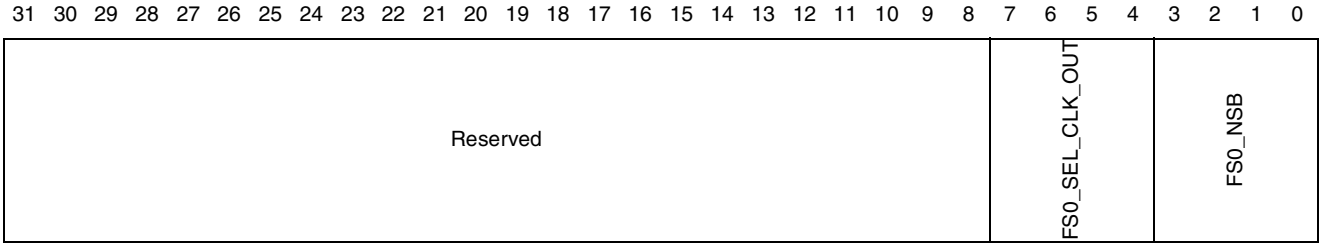
Address: *ClkBBaseAddress* + 0x044

Type: R/W

Reset: 0x02

Description: Defines the SDIV3 parameter (output divider control) for the frequency synthesizer FS0 channel 3.

CKGB_FS0_PWR_DN FS1 digital part power-down/up



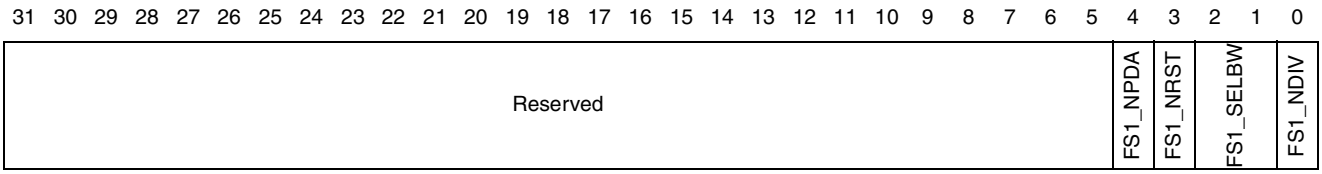
Address: *ClkBBaseAddress* + 0x58
Type: R/W
Reset: 0x77
Description: Controls the standby mode of the FS0 digital part.

[31:8] **Reserved**

[7:4] **FS0_SEL_CLK_OUT**
Must be set to 0x7

[3:0] **FS0_NSB**
Control independently the standby mode of the digital part of the 4 channels - Active low.

CKGB_FS1_CFG Global FS1 parameters



Address: *ClkBBaseAddress* + 0x5C
Type: R/W
Reset: 0x18
Description: Defines the FS1 frequency synthesizer global parameters.

[31:5] **Reserved**

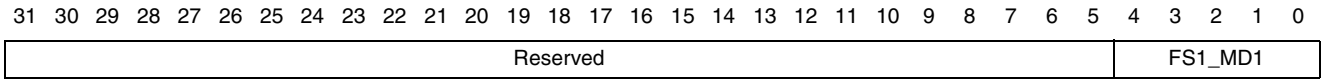
[4] **FS1_NPDA**: Analog power down mode
0: Analog power down
1: Analog power up

[3] **FS1_NRST**: Software reset
0: Reset active
1: Running

[2:1] **FS1_SELBW**: Bandwidth selection for ref clock noise reduction
00: Good reference (bandwidth = 2.4 MHz)
01: Very bad reference (bandwidth = 0.8 MHz)
10: Bad reference (bandwidth = 1.2 MHz)
11: Very good reference (bandwidth = 4 MHz)

[0] **FS1_NDIV**: Select the input frequency, if NDIV is set to
0: Ref clock frequency must be 27/30 MHz
1: Ref clock frequency must be 54/60 MHz

CKGB_FS1_MD1 FS1 channel 1 coarse selection (CLK_DISP_ID)



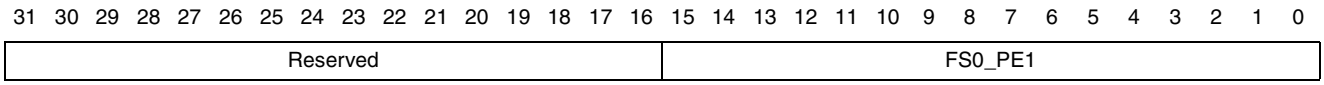
Address: *ClkBBaseAddress* + 0x060

Type: R/W

Reset: 0x09

Description: Defines the MD1 parameter (coarse selection) for the frequency synthesizer FS1 channel 1.

CKGB_FS1_PE1 FS1 channel 1 fine selection (CLK_DISP_ID)



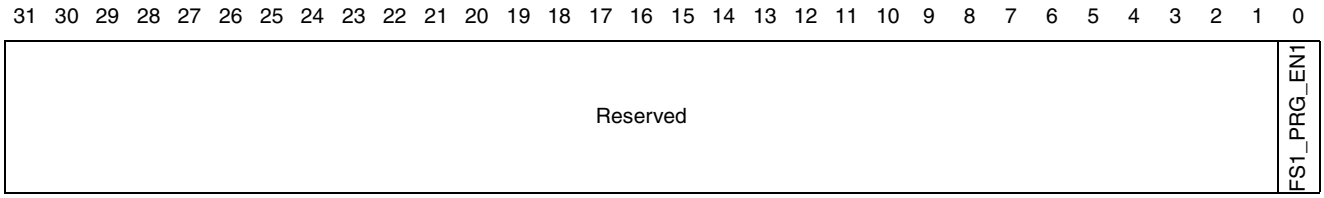
Address: *ClkBBaseAddress* + 0x064

Type: R/W

Reset: 0x1C72

Description: Defines the MD1 parameter (fine selection) for the frequency synthesizer FS1 channel 1.

CKGB_FS1_PRG_EN1 FS1 channel 1 programming enable (CLK_DISP_ID)



Address: *ClkBBaseAddress* + 0x068

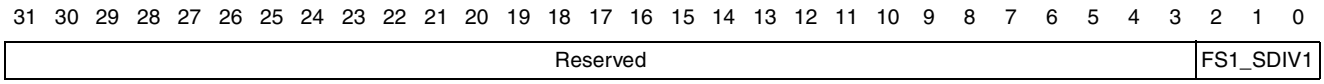
Type: R/W

Reset: 0x1

Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS1 channel 1. This parameter must be set to 1 to take into account a new configuration.

0: PE1 and MD1 parameters are ignored
1: PE1 and MD1 parameters are taken into account

CKGB_FS1_SDIV1 FS1 channel 1 output clock divider (CLK_DISP_ID)



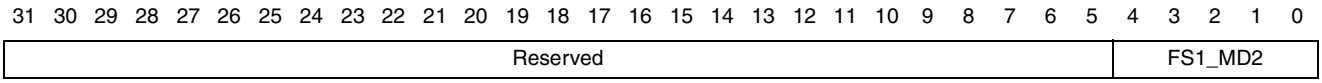
Address: *ClkBBaseAddress* + 0x06C

Type: R/W

Reset: 0x03

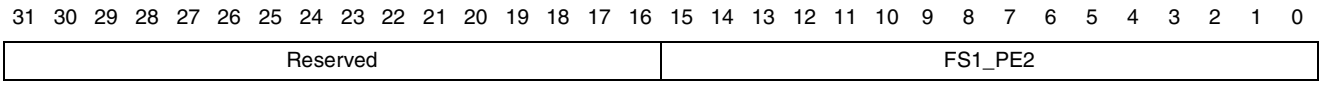
Description: Defines the SDIV1 parameter (output divider control) for the frequency synthesizer FS1 channel 1.

CKGB_FS1_MD2 FS1 channel 2 coarse selection (CLK_PIPE)



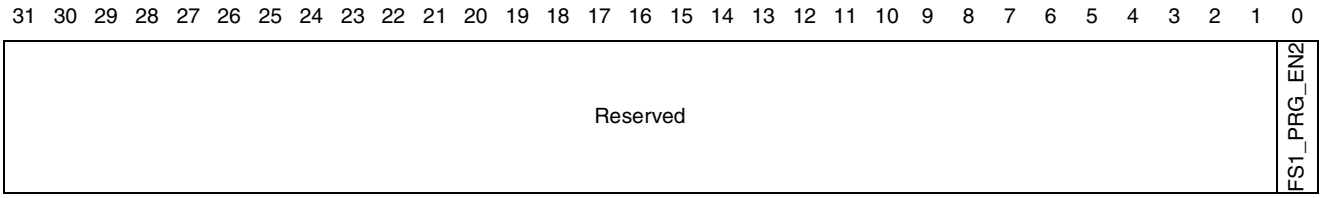
Address: *ClkBBaseAddress* + 0x070
Type: R/W
Reset: 0x19
Description: Defines the MD2 parameter (coarse selection) for the frequency synthesizer FS1 channel 2.

CKGB_FS1_PE2 FS1 channel 2 fine selection (CLK_PIPE)



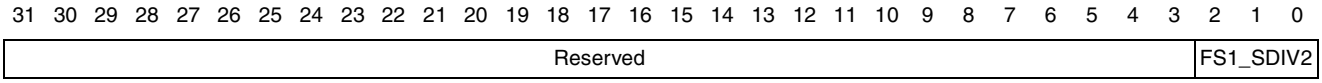
Address: *ClkBBaseAddress* + 0x074
Type: R/W
Reset: 0x3333
Description: Defines the PE2 parameter (fine selection) for the frequency synthesizer FS1 channel 2.

CKGB_FS1_PRG_EN2 FS1 channel 2 programming enable (CLK_PIPE)



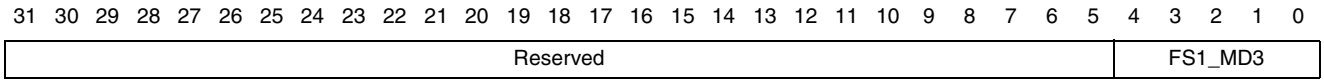
Address: *ClkBBaseAddress* + 0x078
Type: R/W
Reset: 1
Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS1 channel 2. This parameter must be set to 1 to take into account a new configuration.

CKGB_FS1_SDIV2 FS1 channel 2 output clock divider (CLK_PIPE)



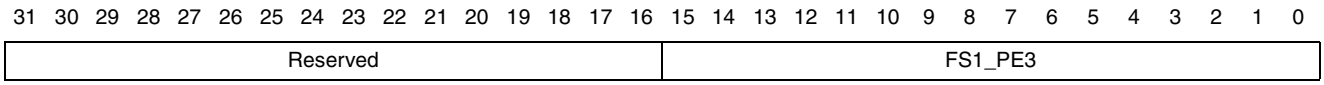
Address: *ClkBBaseAddress* + 0x07C
Type: R/W
Reset: 0
Description: Defines the SDIV2 parameter (output divider control) for the frequency synthesizer FS1 channel 2.

CKGB_FS1_MD3 FS1 channel 3 coarse selection



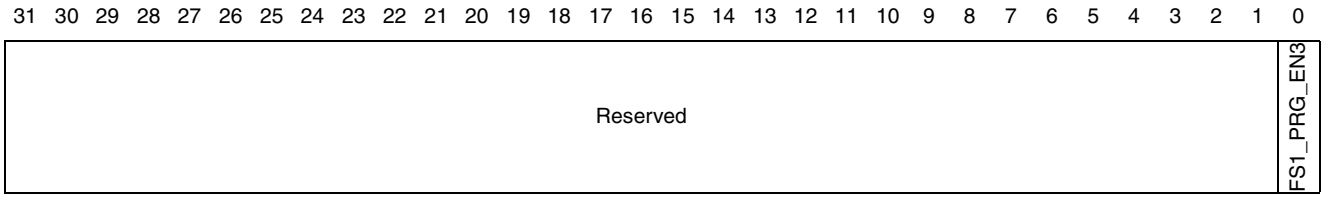
Address: *ClkBBaseAddress* + 0x080
Type: R/W
Reset: 0x11
Description: Defines the MD3 parameter (coarse selection) for the frequency synthesizer FS1 channel 3.

CKGB_FS1_PE3 FS1 channel 3 fine selection



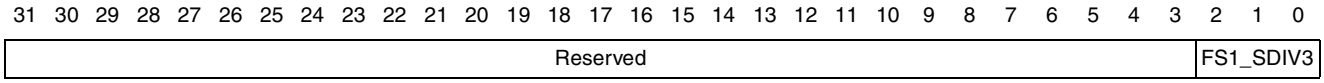
Address: *ClkBBaseAddress* + 0x084
Type: R/W
Reset: 0x1C72
Description: Defines the PE3 parameter (fine selection) for the frequency synthesizer FS1 channel 3.

CKGB_FS1_PRG_EN3 FS1 channel 3 programming enable



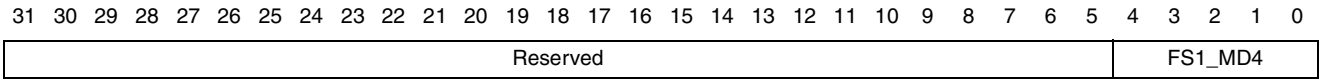
Address: *ClkBBaseAddress* + 0x088
Type: R/W
Reset: 1
Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS1 channel 3. This parameter must be set to 1 to take into account a new configuration.

CKGB_FS1_SDIV3 FS1 channel 3 output clock divider



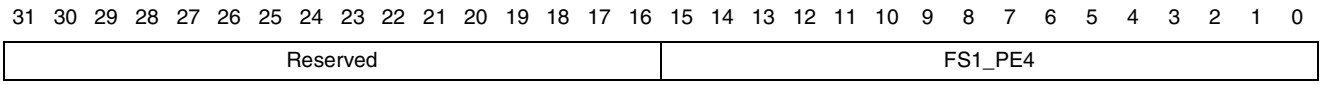
Address: *ClkBBaseAddress* + 0x08C
Type: R/W
Reset: 0x03
Description: Defines the SDIV3 parameter (output divider control) for the frequency synthesizer FS1 channel 3. Range: from 2 to 256.

CKGB_FS1_MD4 FS1 channel 4 coarse selection (CLK_LPC)



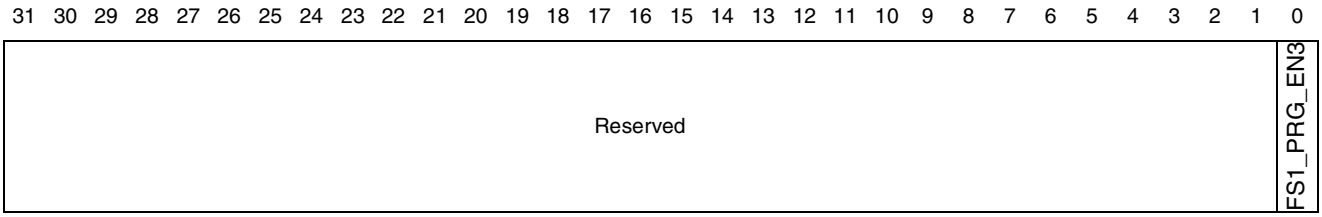
Address: *ClkBBaseAddress* + 0x090
Type: R/W
Reset: 0x13
Description: Defines the MD4 parameter (coarse selection) for the frequency synthesizer FS1 channel 4.

CKGB_FS1_PE4 FS1 channel 4 fine selection (CLK_LPC)



Address: *ClkBBaseAddress* + 0x094
Type: R/W
Reset: 0x00
Description: Defines the PE4 parameter (fine selection) for the frequency synthesizer FS1 channel 4.

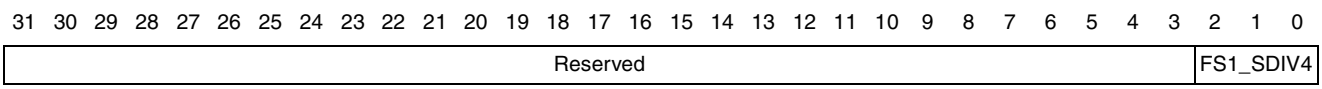
CKGB_FS1_PRG_EN4 FS1 channel 4 programming enable (CLK_LPC)



Address: *ClkBBaseAddress* + 0x098
Type: R/W
Reset: 1
Description: Defines the PRG_EN parameter (programming enable) for the frequency synthesizer FS1 channel 4. This parameter must be set to 1 to take into account a new configuration.

0: PE3 and MD3 parameters are ignored
1: PE3 and MD3 parameters are taken into account.

CKGB_FS1_SDIV4 FS1 channel 4 output clock divider (CLK_LPC)



Address: *ClkBBaseAddress* + 0x09C
Type: R/W
Reset: 2
Description: Defines the SDIV3 parameter (output divider control) for the frequency synthesizer FS1 channel 4. Range: from 2 to 256.

CKGB_FS1_PWR_DN FS1 digital part power-down/up

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												FS1_NSB			

Address: *ClkBBaseAddress* + 0x0A0
Type: R/W
Reset: 0xFF
Description: Controls stand-by mode of the FS1 digital part.

[31:4] **Reserved**
Must be set to 7.

[3:0] **FS1_NSB**
Control independently the stand-bye mode of the digital part of the four channels - Active low.

CKGB_DISP_CFG Video display clocks configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				CLK_PIX_SD_SEL		CLK_DISP_ID_SEL		CLK_656_SEL		CLK_DISP_HD_SEL		CLK_PIX_HDMI		CLK_TMDS_HDMI	

Address: *ClkBBaseAddress* + 0x0A4
Type: R/W
Reset: 0x469
Description: HD and SD video display clocks configuration.

[31:12] **Reserved**

[11:10] **CLK_PIX_SD_SEL**
00: CLK_PIX_SD = CLK_FS0_CHAN1 / 2
01: CLK_PIX_SD = CLK_FS0_CHAN1 / 4
10: CLK_PIX_SD = CLK_FS0_CHAN1 / 8
11: CLK_PIX_SD = CLK_FS0_CHAN1 / 2

[9:8] **CLK_DISP_ID_SEL**
00: CLK_DISP_ID = CLK_FS1_CHAN1 / 2
01: CLK_DISP_ID = CLK_FS1_CHAN1 / 4
10: CLK_DISP_ID = CLK_FS1_CHAN1 / 8
11: CLK_DISP_ID = CLK_FS1_CHAN1 / 2

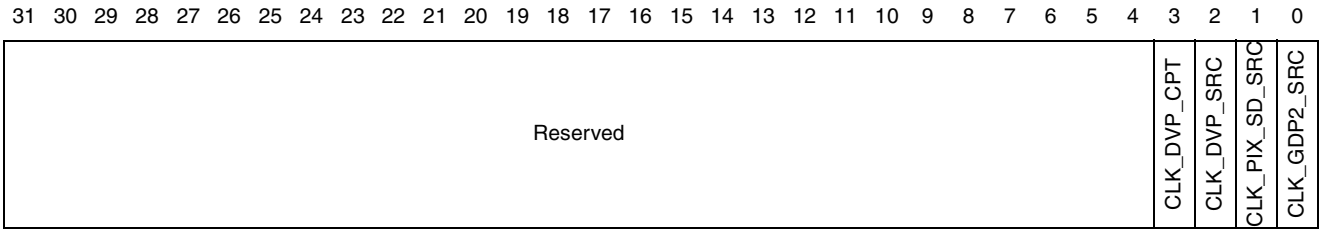
[7:6] **CLK_656_SEL**
00: CLK_656 = CLK_FS0_CHAN1 / 2
01: CLK_656 = CLK_FS0_CHAN1 / 4
10: CLK_656 = CLK_FS0_CHAN1 / 8
11: CLK_656 = CLK_FS0_CHAN1 / 2

[5:4] **CLK_DISP_HD_SEL**
00: CLK_DISP_HD = CLK_FS0_CHAN1 / 2
01: CLK_DISP_HD = CLK_FS0_CHAN1 / 4
10: CLK_DISP_HD = CLK_FS0_CHAN1 / 8
11: CLK_DISP_HD = CLK_FS0_CHAN1 / 2

[3:2] **CLK_PIX_HDMI**
00: CLK_PIX_HDMI = CLK_FS0_CHAN1 / 2
01: CLK_PIX_HDMI = CLK_FS0_CHAN1 / 4
10: CLK_PIX_HDMI = CLK_FS0_CHAN1 / 8
11: CLK_PIX_HDMI = CLK_FS0_CHAN1 / 2

[1:0] **CLK_TMDS_HDMI**
00: CLK_TMDS_HDMI = CLK_FS0_CHAN1 / 2
01: CLK_TMDS_HDMI = CLK_FS0_CHAN1 / 4
10: CLK_TMDS_HDMI = CLK_FS0_CHAN1 / 8
11: CLK_TMDS_HDMI = CLK_FS0_CHAN1 / 2

CKGB_CLK_SRC Clock source selection



Address: *ClkBBaseAddress* + 0x0A8

Type: R/W

Reset: 0x06

Description: Defines the clock source when more than one source is possible.

[31:4] **Reserved**

[3] **CLK_DVP_CPT**: CLK_DVP capture
0: CLK_DVP sourced from VIDINCLK input pin
1: CLK_DVP sourced from frequency synthesizers

[2] **CLK_DVP_SRC**
0: CLK_DVP sourced from FS0
1: CLK_DVP sourced from FS1
Note: CLK_DVP sourced from frequency synthesizers only if CLK_DVP_CPT=1

[1] **CLK_PIX_SD_SRC**
0: CLK_PIX_SD sourced from FS0
1: CLK_PIX_SD sourced from FS1

[0] **CLK_GDP2_SRC**
0: CLK_GDP2 sourced from FS0
1: CLK_GDP2 sourced from FS1

CKGB_CLK_DIV Individual clocks division

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CLK_LPC_DIV	CLK_PIPE_DIV	CLK_PIX_SD_DIV	CLK_DISP_ID_DIV	CLK_656_DIV	CLK_DISP_HD_DIV	CLK_PIX_HD_DIV	CLK_PIX_HDMI_DIV	CLK_TMDS_HDMI_DIV	CLK_BCH_HDMI_DIV

Address: *ClkBBaseAddress* + 0xAC

Type: R/W

Reset: 0

Description: Run clocks generated from clock generator B at nominal or reduced frequency, allowing reduced power consumption. The clock affected is shown by the name of the bitfield, for example CLK_LPC_DIV controls CLK_LPC.

0: Run at nominal frequency
1: Run at 1/1024 frequency

[31:10] **Reserved**

[9] **CLK_LPC_DIV**

[8] **CLK_PIPE_DIV**

[7] **CLK_PIX_SD_DIV**

[6] **CLK_DISP_ID_DIV**

[5] **CLK_656_DIV**

[4] **CLK_DISP_HD_DIV**

[3] **CLK_PIX_HD_DIV**

[2] **CLK_PIX_HDMI_DIV**

[1] **CLK_TMDS_HDMI_DIV**

[0] **CLK_BCH_HDMI_DIV**

CKGB_CLK_EN Individual clocks enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CLK_LPC_EN	CLK_TTXT_EN	CLK_PIPE_EN	CLK_PIX_HDMI_EN	CLK_PIX_SD_EN	CLK_DISP_ID_EN	CLK_GDP2_EN	CLK_656_EN	CLK_TMDS_HDMI_EN	CLK_DISP_HD_EN	CLK_PIX_HD_EN	CLK_BCH_HDMI_EN	CLK_DSS_EN	CLK_DAA_EN

Address: *ClkBBaseAddress* + 0x0B0

Type: R/W

Reset: 0x2FFF

Description: Stop or enable clocks generated from clock generator B. The clock affected is shown by the name of the bitfield, for example CLK_LPC_EN controls CLK_LPC.

0: Stop
1: Enable

[31:14] **Reserved**

[13] **CLK_LPC_EN**

[12] **CLK_TTXT_EN**

[11] **CLK_PIPE_EN**

[10] **CLK_PIX_HDMI_EN**

[9] **CLK_PIX_SD_EN**

[8] **CLK_DISP_ID_EN**

[7] **CLK_GDP2_EN**

[6] **CLK_656_EN**

[5] **CLK_TMDS_HDMI_EN**

[4] **CLK_DISP_HD_EN**

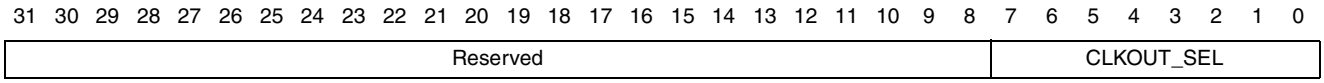
[3] **CLK_PIX_HD_EN**

[2] **CLK_BCH_HDMI_EN**

[1] **CLK_DSS_EN**

[0] **CLK_DAA_EN**

CKGB_CLKOUT_SEL Clocks observation selection



Address: *ClkBBaseAddress* + 0x0B4

Type: R/W

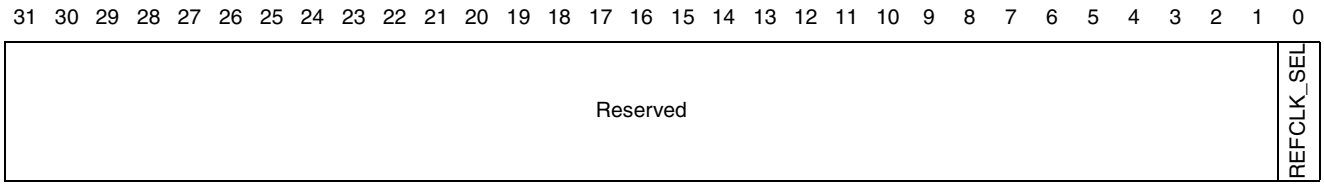
Reset: 0

Description: Selects which clock of clockgen B to be routed to the PIO5[2] (SYSCLKOUTB) pin.

Table 77: PIO5[2] pin source

CLKOUT_SEL	SYSCLKOUTB
0000 0000	CLK_BCH_HDMI
0000 0001	CLK_TMDS_HDMI
0000 0010	CLK_PIX_HDMI
0000 0011	CLK_PIX_HD
0000 0100	CLK_DISP_HD
0000 0101	CLK_656
0000 0110	CLK_GDP2
0000 0111	CLK_DISP_ID
0000 1000	CLK_PIX_SD
0000 1001	CLK_DSS
0000 1010	CLK_DAA
0000 1011	CLK_PIPE
0000 1100	CLK_TTXT
0000 1101	CLK_LPC
0000 1110	CLK_DVP
1011 1100	CLK_PP
1100 1100	CLK_IC_150
All others	Reserved

CKGB_REFCLK_SEL Reference clock source selection



Address: *ClkBBaseAddress* + 0x0B8

Type: R/W

Reset: 0

Description: Allows selection of clockgen B reference clock. this can be either the internal 30 MHz clock used by the SATA and USB interfaces or an external clock connected to the SYSBCLKINALT pin.

0: Internal 30 MHz clock
1: External clock

17 Reset

17.1 Overview

The different reset sources are:

- The power-on reset signal applied to the NOTRESETIN pad
- The watchdog reset signal generated by the ST40 internal watchdog-timer (WDT)
- The security violation reset
- The UDI reset command received via the ST40 debug port
- The software reset applied to the ST231 CPUs via the SYS_CFG27 and SYS_CFG29 configuration registers.

A reset output pin NOTWDOGRSTOUT is provided to control the reset of external devices. The length of this reset signal can be controlled with the mode pin EMIADDR[14].

Boot and reset registers are described in Chapter 21: *System configuration registers on page 171*. See also Chapter 20: *Boot modes on page 169*.

17.2 Power-on reset

The complete reset sequence is executed only during a “power-on-reset” sequence (cold reset). In this sequence, everything is reset including the clock generators and the values captured on the mode pins during the reset phase.

17.3 System reset

When the reset source is the security checker, the watchdog timer or the UDI, then a “system reset” sequence is executed (warm reset). In that sequence, everything is reset except the clock generators, the antifuses and some of the system configuration registers. The mode pins values sampled during the power-on-reset are kept unchanged.

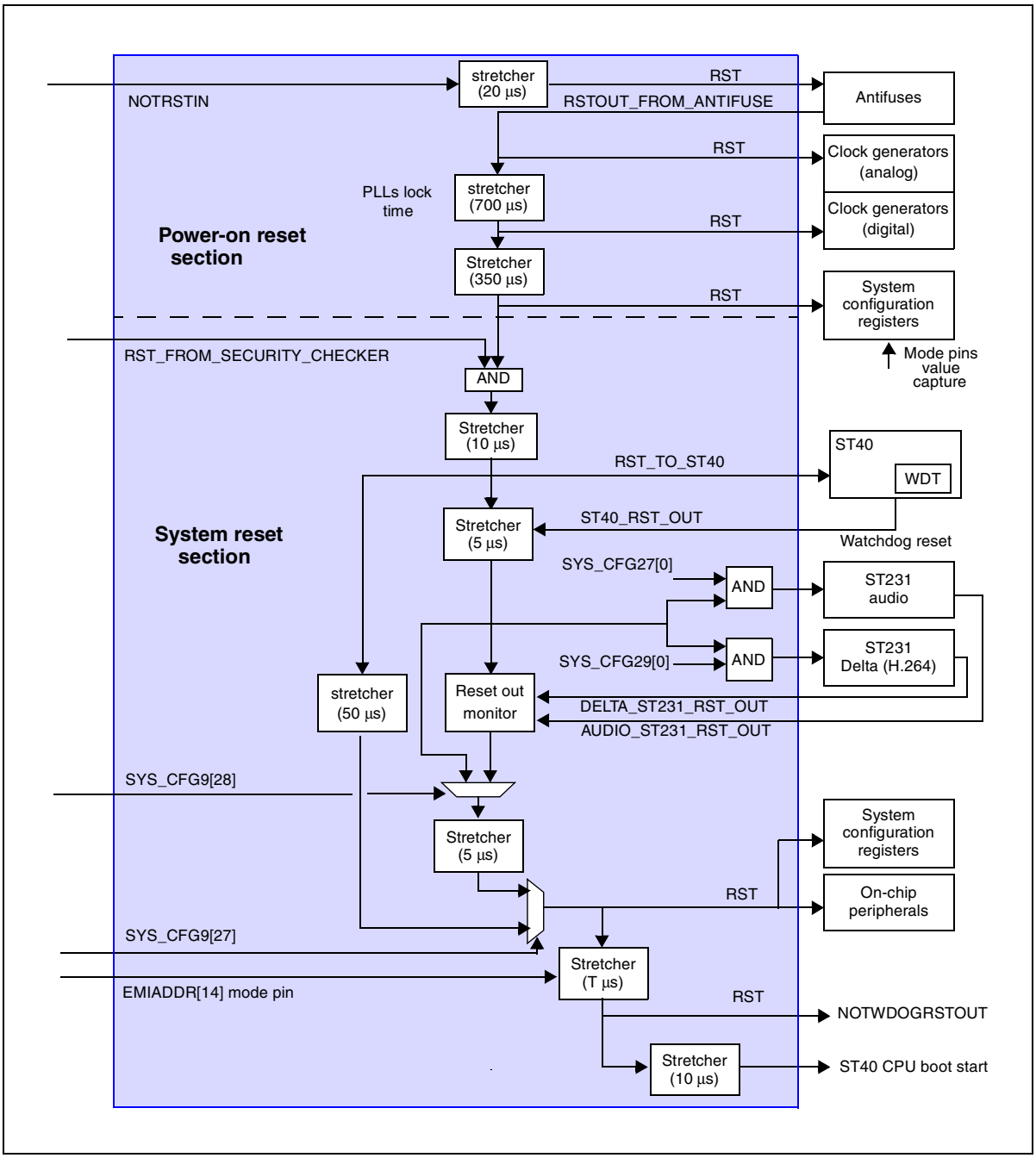
17.4 Reset sequence

The reset sequence controlled by the reset generator is as follows:

1. The clock generators A and B are reset. When the PLLs are locked and the clock generators are reset then the reset propagation continues.
2. The ST40 CPU is reset. When the ST40 internal reset sequence is ended then the reset propagation continues.
3. The audio and Delta (H.264) ST231s are reset. When the ST231s’ reset sequences are ended then the reset propagation continues. When reset, the audio and Delta ST231 boot, but their access to the memory is halted. The configuration registers SYS_CFG26[0] and SYS_CFG28[0] must be set by the ST40 to authorize memory accesses (see Chapter 20: *Boot modes on page 169*).
4. All on-chip peripherals are reset. The reset propagation continues.
5. The reset signal on the NOTWDOGRSTOUT output pin is de-asserted after a delay defined by the mode pin value EMIADDR[14].
6. The ST40 boots and starts to execute its code.

The reset sequence controlled by the reset generator takes into account the antifuse reset timing sequence, some optional reset configuration (short or long reset) and the internal reset sequences of the ST40 and ST231s CPUs. This is shown in Figure 55.

Figure 55: Reset generator block diagram



17.5 CPUs reset and reset out

17.5.1 Power-on reset

Both the ST40 and ST231s are reset when a reset signal is applied to the NOTRESETIN input pin. In that case, the reset signal is propagated and all the CPUs and internal peripherals are reset in sequence.

17.5.2 ST231 software reset

The audio and Delta ST231 CPUs can be reset by the ST40 using the configuration registers SYS_CFG27[0] and SYS_CFG29[0]. Setting the bit 0 to 1 produces a reset to the ST231. Resetting the bit 0 to 0 releases the software reset.

17.5.3 ST231 reset out bypass

When reset, the ST231 starts an internal reset sequence. It indicates that it has ended this sequence by asserting a reset out signal. By default, this reset out signal is used to propagate the reset to all the peripherals (internal and external).

Before applying a software reset to the ST231, this reset out signal must be bypassed, otherwise not only the ST231 will be reset but all the peripherals. The configuration bit SYS_CFG9.CPU_RST_OUT_BYPASS1 (bit 28) is provided to bypass both the Delta and audio ST231 reset out signals. This must be used when the default ST231 boot address is changed and a reset must be applied (see Chapter 20: *Boot modes on page 169*).

17.6 ST40 watchdog timeout reset

The ST40 includes a watchdog timer (WDT) that can generate a timeout event. In case of timeout, a system reset is generated.

Using the watchdog timer mode:

1. Set the WT/IT bit in the WTCSR (watchdog timer control/status) register to 1 to set the timer in watchdog mode, select the type of reset with the RSTS bit, and the count clock with bits CKS2-CKS0, and set the initial value in the WTCNT counter.
2. When the TME bit (timer enable) in the WTCSR register is set to 1, the count starts in watchdog timer mode.
3. During operation in watchdog timer mode, write 0x00 to the counter periodically so that it does not overflow.
4. When the counter overflows, the WDT sets the WOVF flag in the WTCSR register to 1, and generates a reset of the type specified by the RSTS bit. The counter then continues counting.

For more details on the ST40 watchdog timer, refer to the ST40 documentation cited in Chapter 4: *CPUs on page 37*.

17.7 NOTWDOGRSTOUT signal length

The NOTWDOGRSTOUT signal length can be either around 1.2 ms when the mode pin EMIADDR[14] is set to 0, or around 200 ms when the mode pin EMIADDR[14] is set to 1.

18 Low-power modes

18.1 Overview

Power saving can be achieved by halting the ST40 CPU and stopping or reducing the clock frequencies. The ST231 CPU clocks can either be stopped or kept running at a low frequency. In addition, a number of functional blocks can be disabled.

Reducing the clocks' frequency dramatically reduces the power consumption, and still enables the execution of some software.

When the ST40 is in one of the low-power modes ("sleep" or "standby"), it can be woken up either by a reset, an NMI interrupt, an ST40 peripheral interrupt or an external interrupt.

Before entering the low-power modes, the DDRs can be set in self-refresh mode via a special sequence of the memory controller.

The code to be executed by the CPU when it is awoken must reside in EMI if the DDRs are in self-refresh mode. In that case, the EMI clock must be kept running at a low frequency.

The different functional units involved in the low-power feature are:

- the ST40, which can be halted by executing the SLEEP instruction,
- the clock generators A, B and C, which can be configured by software to generate low-frequency clocks. They can also automatically enter the low-power mode when they detect an event from the low-power controller (LPC) and automatically exit from low-power mode when a wake-up interrupt is detected by the ILC. In low-power mode, the clocks are automatically divided by 1024. The unnecessary clocks can be stopped by configuration registers,
- the IRB/UHF Rx processor, which can detect an activity on the IRB/UHF inputs (PIO3[3] and PIO3[4]) and can generate a wake-up interrupt to the ST40 and exit the clock generators from their low-power mode,
- the ILC, which must be configured to map the ST40 interrupts IRL(3-0) to 4 of the following external interrupts: SYSITRQ(3-0), UHF_WAKEUP (PIO3[4]), IRDA_WAKEUP (PIO3[3]). These signals are connected to the ILC (EXT_INT[5-0] input pins in Figure 56 below) and can be used as a wake-up interrupt for the ST40 (see Table 78 below).
- the low-power controller (LPC), which includes a 40-bit low-power alarm counter (LPA). The LPC is used to put the clock generators in low-power mode. The LPA timer is used to exit from low-power after a user-defined delay. The LPA timer is clocked by the CLK_LPC clock defined in clock generator B.

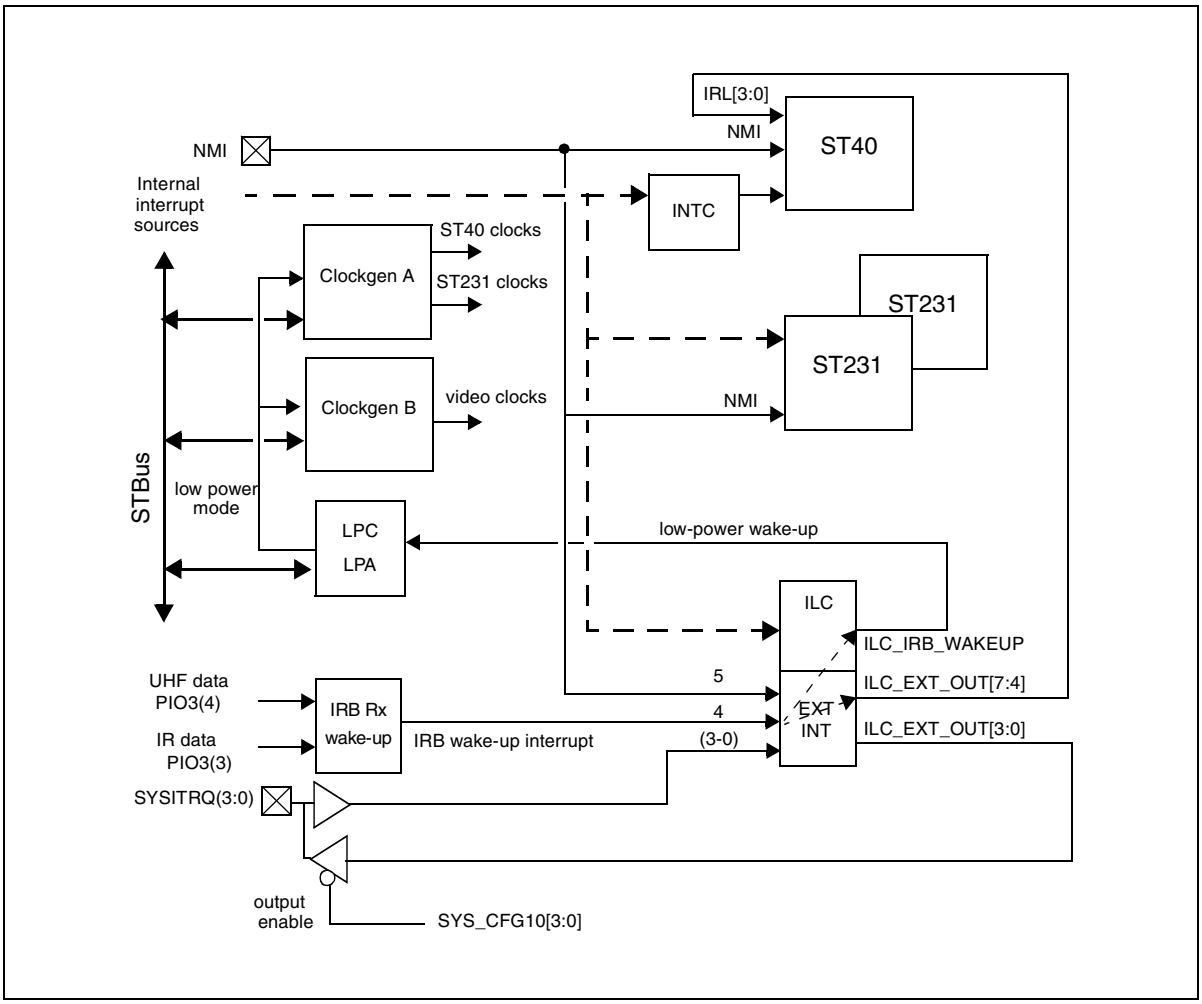
The ILC wake-up interrupt sources are described in Table 78 below.

Table 78: ILC external wake-up interrupt

ILC external interrupt input	Interrupt source	Description
EXT_INT(0)	SYSITRQ(0)	External interrupt input
EXT_INT(1)	SYSITRQ(1)	External interrupt input
EXT_INT(2)	SYSITRQ(2)	External interrupt input
EXT_INT(3)	SYSITRQ(3)	External interrupt input
EXT_INT(4)	IRB wake-up	Wake-up interrupt from the IRB Rx
EXT_INT(5)	NMI	Non-maskable interrupt

Figure 56 describes the connections between those units.

Figure 56: STx7100 low-power diagram



18.2 ST40 low-power modes

The ST40 provides two low-power modes which halt the CPU:

- sleep,
- standby.

When it is in sleep mode, the ST40 is halted, its registers are held, but the ST40 peripherals continue to operate.

When it is in standby mode, the ST40 is halted, its registers are held, but the ST40 peripherals are stopped.

Transition to sleep and standby modes

If a SLEEP instruction is executed with bit STDBY of register CPG.STBCR. set to 0, then the ST40 switches from the program execution state to sleep mode (See Chapter 4: *CPUs on page 37*). If the SLEEP instruction is executed with bit STDBY set to 1, then the ST40 enters the standby mode.

Exiting from sleep and standby modes

A reset, or a NMI interrupt or an external interrupt IRL(3-0) can be used to exit from both sleep and standby modes.

18.3 Reducing or/and stopping the clocks

Most of the clocks generated by the clockgen A and clockgen B can be put in low-power mode (divided by 1024). This can be achieved either by programming the clock generator configuration registers or automatically using the LPA (low power alarm) timer of the LPC (low power controller) module.

Using the clock generators configuration registers

See also Chapter 15: *Clocks on page 120* and Chapter 16: *Clock registers on page 134*.

All clocks can be programmed through configuration registers to be slowed by a factor of 1024 except the following:

- ST40 clocks (CLK_ST40, CLK_ST40_IC, CLK_ST40_PER),
- MPEG2 decoder clock,
- FDMA clock,
- interconnect and EMI clocks (CLK_IC, CLK_IC_DIV2, CLK_EMI),
- Delta decoder processing clock (CLK_PP),
- SmartCard, DAA and RTC clocks (CLK_DSS, CLK_DAA, CLK_RTC).

Before entering a low-power mode, these clocks can be redefined with a lower frequency or stopped.

To exit from low-power mode, the clock generator configuration registers must be reprogrammed.

Using the low power alarm

A global power-down command can be issued by the LPC controller. Bit 0 of register LPC_LPA_STRT must be set to 1 to enter the low-power mode.

First, a value must be written to the LPC_LPA registers to initialize the LPA timer. When the LPA counter reaches 0 then a global power-down event is generated that automatically exits the clock generators from the low-power mode. The clock nominal frequencies are then restored.

The duration of the LPA countdown can range from a few milliseconds to several hundred days.

Programming the minimum frequency for the ST40, EMI and interconnect clocks

To minimize power consumption, the ST40, interconnect and EMI clocks can be programmed to their smallest value.

This value is obtained when clockgen A PLL0 and PLL1 are set up for 6.25 MHz (MDIV=27, NDIV=100, PDIV=32). Hence, the ST40 clocks become 3.125 MHz, 1.5625 MHz and 781 kHz. The interconnect clocks become 3.125 MHz and 1.5625 MHz and the EMI clock becomes 781 kHz.

18.4 ILC interrupt controller: wake-up interrupt

The ILC is used to configure the wake-up interrupt. The registers ILC_WAKEUP_EN and ILC_WAKEUP_ACTIVE_LEVEL must be configured to define which interrupt will be used as a wake-up interrupt and its active polarity.

The ILC must also be configured to map the four ST40 external interrupts IRL(3-0) to the ILC inputs EXT_INT(5-0). For this purpose the registers ILC_SET_EN2, ILC_EXT_MODEn and ILC_EXT_PRIORITYn must be set accordingly. This interrupt is then used as a wake-up interrupt to exit the ST40 from its sleep or standby mode.

For more details on ILC programming, see Chapter 22: *Interrupt system on page 196*.

18.5 DDR self-refresh

To put the system LMI DDR in self-refresh mode, the CPU must set SYS_CFG11.LMI_SYS_PWRD_REQ (bit 28, DDR power-down request) to 1. For the video LMI DDRs, SYS_CFG11.LMI_VID_PWR_DN_REQ (bit 30) must be set to 1.

This write operation activates the power-down protocol of the memory controller (LMI core). The memory controller completes all the outstanding operations and then puts the DDRs in self-refresh mode. When the system DDRs are in self-refresh mode, SYS_STA12.LMI_SYS_PWRD_ACK[28] (system DDR power-down acknowledge) is set. When the video DDRs are in self-refresh mode, SYS_STA13.LMI_VID_PWRD_ACK[28] is set.

As soon as this acknowledge is received, the LMI clocks (in clockgen A) can be switched off or slowed down.

After exiting low-power mode, the DDR padlogic DLLs and the memory controller must be reset using SYS_CFG11.LMI_SYS_RST_N[6] and SYS_CFG11.LMI_VID_RST_N[15].

18.6 Low-power mode sequence

Entering the low-power modes

1. Configure the ILC to determine which interrupt will be used to exit the ST40 from sleep or standby mode. This interrupt (IRB, NMI or external) is routed to the ST40. The same interrupt must be declared as a wake-up interrupt for the LPC controller.

   Example: use EXT_INT(4) as a wake-up interrupt (corresponds to IRB wake-up)

   set bit 4 of ILC_WAKEUP_EN to define EXT_INT(4) as a wake-up interrupt

   set bit 4 of ILC_WAKEUP_ACTIVE_LEVEL to define the active level of the interrupt

   set bit 4 of ILC_SET_EN2 to enable EXT_INT(4)

   write 0x01 in register ILC_EXT_MODE4 to define a high-level interrupt

   write 0x8004 in register ILC_EXT_PRIORITY4 to map the EXT_INT(4) interrupt to the ST40 IRL(0)
2. Put the DDRs in self-refresh mode
3. Configure the clockgen A and B to stop the unnecessary clocks and to slow-down the ST40, EMI and interconnect clocks.
4. Configure the LPA counter time-out by writing in LPC_LPA_LS and LPC_LPA_MS
5. Enter the clock generators low-power mode by setting LPC_LPA_START.STRT.
6. Execute the SLEEP instruction (the ST40 will be either in sleep or standby mode depending on the value of CPG.STBCR).

Exiting the low-power modes

The STx7100 exits from low-power mode on either of the following events:

- the LPA counter reaches 0,
- a wake-up interrupt has occurred (reset, NMI, external, IRB wake-up).

After exiting a low-power mode, the nominal clock frequencies are restored. The clocks that were stopped can be restarted.

If the DDRs are in self-refresh mode then a software reset to the DDR padlogic and memory controller must be applied.

LPC clock frequency

In standby mode, the LPC clock frequency must be less or equal to the RTC clock frequency.

19 Low power registers

Register addresses are provided as *LPC_LPABaseAddress* + offset.

The *LPC_LPABaseAddress* is:

0x1800 8000.

Table 79: Low power alarm register summary

Register	Description	Offset	Type
LPC_LPA_LS	LPA - low power alarm timer (LSB)	0x0410	R/W
LPC_LPA_LS	LPA - low power alarm timer (MSB)	0x0414	R/W
LPC_LPA_START	LPA - low power alarm timer start	0x0418	R/W

LPC_LPA_LS Low power alarm timer low value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPA_TMR[31:0]																															

Address: *LPC_LPABaseAddress* + 0x0410

Type: R/W

Reset: 0

Description: Defines the four least significant bytes of the low power alarm counter.

LPC_LPA_MS Low power alarm timer high value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								LPA_TMR[39:32]							

Address: *LPC_LPABaseAddress* + 0x0414

Type: R/W

Reset: 0

Description: Defines the most significant byte of the low power alarm counter.

LPC_LPA_START Low power alarm timer start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															START

Address: *LPC_LPABaseAddress* + 0x0418

Type: R/W

Reset: 0

Description: Setting the bit 0 to 1 starts the low power alarm counter. The counter starts to count down from the value defined in LPC_LPA_MS and LPC_LPA_LS.

20 Boot modes

The STx7100 has three boot modes, controllable by two external pins: BOOTMODE[1:0]. These pins are strobed at the end of the reset period (power-on or watch-dog resets). The boot code must be stored in flash memory attached to the EMI bus.

The boot system is closely connected with reset; see Chapter 17: *Reset on page 161*. Boot and reset registers are described in Chapter 21: *System configuration registers on page 171*.

- **ST40 mode** (Default): This is the standard mode, in which the ST40 boots first from EMI. The Delta and audio ST231 boot requests are both halted by the interconnect. It is up to the ST40 software to control the boot sequence of the two ST231. The boot address of the ST231s must be defined in the system configuration registers before the boot request is released by the ST40.
- **Delta (H.264) ST231 mode**: The ST40 boot request is halted by the interconnect, but the boot request from the Delta ST231 is released, allowing the Delta ST231 to boot from EMI. In this mode, the Delta ST231 uses the default boot address (0x0 0000). This mode may be useful for low-level software debugging; in this mode, the ST40 is not expected to boot.
- **Audio ST231 mode**: The ST40 boot request is halted by the interconnect, but the boot request from the audio ST231 is released, allowing the audio ST231 to boot from EMI. In this mode, the audio ST231 uses the default boot address (0x0 0000). This mode may be useful for low-level software debugging; in this mode the ST40 is not expected to boot.

For security purpose, an antifuse can be set to disable the Delta and audio ST231 modes.

Table 80: Boot mode selection

BOOTMODE pins	Boot mode
00	**Default**: ST40 boots first. Delta and audio ST231 boot sequences are controlled by the ST40.
01	**Delta ST231**: ST40 boot is halted. Delta ST231 boots first and controls the ST40 and audio ST231 boot sequences.
10	**Audio ST231**: ST40 boot is halted. Audio ST231 boots first and controls the ST40 and Delta ST231 boot sequences.

20.1 Default mode

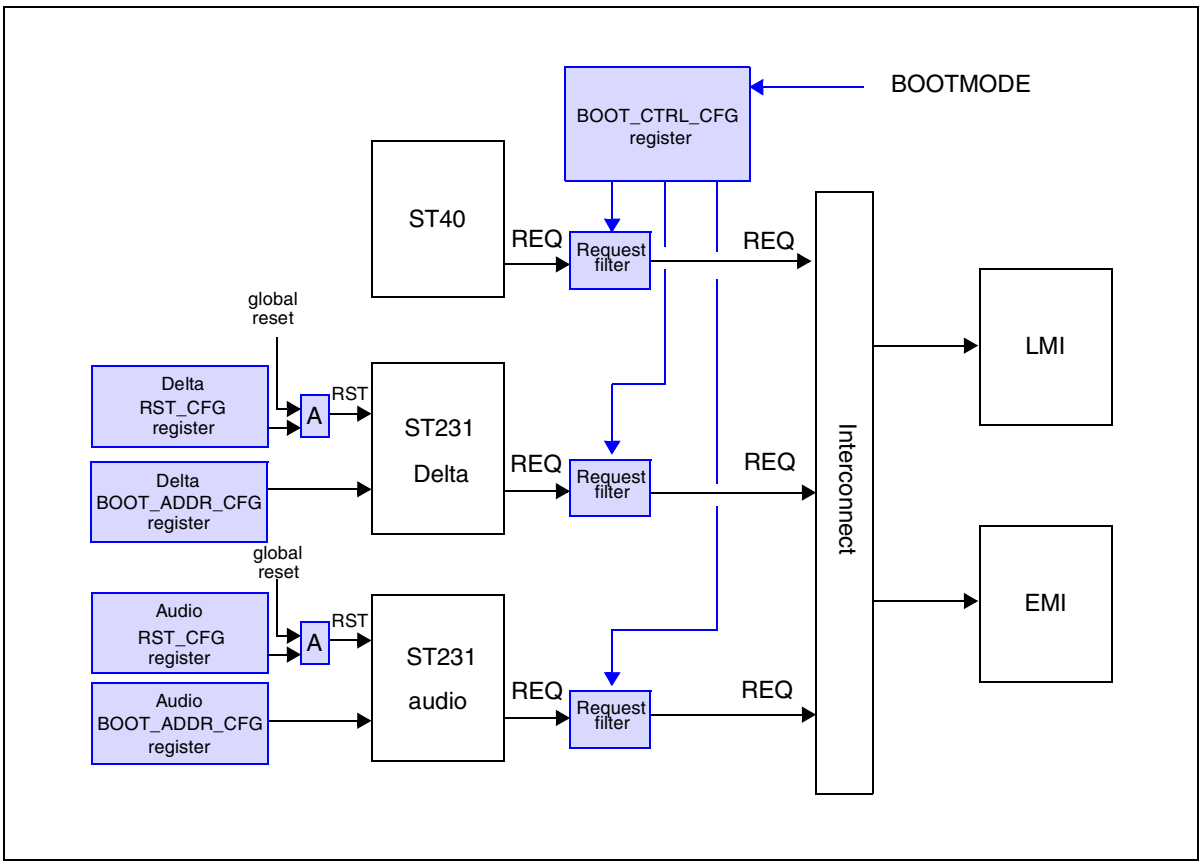
In the default boot mode, the requests of the two ST231s are filtered and halted. The ST40 boots from EMI and typically goes through the following sequence for each ST231:

1. Writes the boot code of the ST231 in EMI or LMI.
2. Updates the boot address configuration register of the ST231.
3. Resets the ST231 with the RST_CFG register.
4. Releases the request filter of the ST231 with the BOOT_CTRL_CFG register.
5. Releases the reset of the ST231.

The ST231 then executes its boot sequence.

The operation is identical for both Delta and audio ST231s.

Figure 57: Boot control block diagram



20.2 Delta ST231 mode

In this mode, the requests of the ST40 and audio ST231 are filtered and halted. The Delta ST231 boots using the default value of its boot control configuration register, which points to the EMI bottom address.

The Delta ST231 controls the boot sequence of the audio ST231 as described above.

The ST40 does not boot.

20.3 Audio ST231 mode

In this mode, the requests of the ST40 and Delta ST231 are filtered and halted. The audio ST231 boots using the default value of its boot control configuration register, which points to the EMI base address.

The audio ST231 controls the boot sequence of the Delta ST231 as described above.

The ST40 does not boot.

21 System configuration registers

21.1 Overview

The system configuration unit includes a number of configuration and status registers.

The **status registers** report the state of the following blocks:

- device identifier,
- USB and SATA,
- mode pins captured values,
- system and video LMI Padlogic.

The **configuration registers** allow the setup of the following blocks:

- transport subsystem,
- NRSSA,
- USB,
- EMI bridge,
- SATA,
- comms,
- ST40 boot,
- reset generator,
- interrupt pads direction,
- video and system LMI padlogic,
- audio ST231 boot and reset,
- Delta (H.264) ST231 boot and reset,
- SATA and USB software JTAG.

21.2 Summary

Register addresses are provided as *SysConfigBaseAddress* + offset.

The *SysConfigBaseAddress* is:

0x1900 1000

Table 81: System configuration registers summary

Name	Function	Offset	Type
Device identity			
DEVICE_ID	Device identifier	0x000	RO
EXTRA_DEVICE_ID	Extra device identifier	0x004	
Status			
SYS_STA0	USB2/SATA status	0x008	RO
SYS_STA1	Mode pin status captured during power-on-reset	0x00C	
SYS_STA2 - 11	Reserved	0x010 - 0x034	
SYS_STA12	System LMI padlogic status	0x038	
SYS_STA13	Video LMI padlogic status	0x03C	
SYS_STA14	Reserved	0x040	
Configuration			
SYS_CFG0	Transport configuration	0x100	R/W
SYS_CFG1	NRSSA configuration	0x104	
SYS_CFG2	USB configuration	0x108	
SYS_CFG3	Video DACs and HDMI configuration	0x10C	
SYS_CFG4	EMI bridge configuration	0x110	Mixed
SYS_CFG5	Reserved	0x114	-
SYS_CFG6	SATA configuration	0x118	R/W
SYS_CFG7	COMMs configuration	0x11C	
SYS_CFG8	ST40 boot control	0x120	
SYS_CFG9	Reset generator, EMI configuration	0x124	
SYS_CFG10	SYSITRQ pads configuration	0x128	
SYS_CFG11	LMI general configuration	0x12C	
SYS_CFG12	System LMI padlogic configuration	0x130	
SYS_CFG13	Video LMI padlogic configuration	0x134	
SYS_CFG14	System LMI padlogic DLL1 control	0x138	
SYS_CFG15	System LMI padlogic DLL2 control	0x13C	
SYS_CFG16 - 19	Reserved	0x140 - 0x14C	-
SYS_CFG20	Video LMI padlogic DLL1 control	0x150	R/W
SYS_CFG21	Video LMI padlogic DLL2 control	0x154	
SYS_CFG22 - 25	Reserved	0x158 - 0x164	-
SYS_CFG26	Audio ST231 boot control	0x168	R/W
SYS_CFG27	Audio ST231 reset control	0x16C	
SYS_CFG28	Delta ST231 boot control	0x170	
SYS_CFG29	Delta ST231 reset control	0x174	
SYS_CFG30 - 32	Reserved	0x178 - 0x180	-
SYS_CFG33	USB and SATA software JTAG configuration	0x184	R/W

21.3 Device identity

DEVICE_ID JTAG device ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION				GROUP_ID						DEVICE_ID										MANUFACTURER_ID											JB

Address: *SysConfigBaseAddress* + 0x000
Type: RO
Reset: 0x0D42 4041
Description:

[31:28] **VERSION**: 0000

[27:22] **GROUP_ID**: 11 0101

[21:12] **DEVICE_ID**: 00 0010 0100

[11:1] **MANUFACTURER_ID**: 000 0010 0000

[0] **JB**: JTAG bit: 1

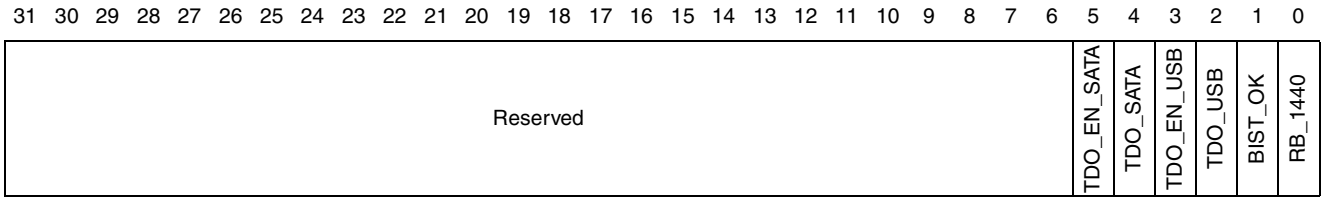
EXTRA_DEVICE_ID Extra device ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHIP_ID																															

Address: *SysConfigBaseAddress* + 0x004
Type: RO
Reset:
Description:

21.4 System status

SYS_STA0 System status 0 (USB & SATA JTAG, USB PLL)



Address: *SysConfigBaseAddress* + 0x008

Type: RO

Reset: 0

Description:

[31:6] **Reserved**

[5] **TDO_EN_SATA**
1: The SATA PHY TAP controller is shifting out

[4] **TDO_SATA**: SATA PHY TDO signal

[3] **TDO_EN_USB**
1: The USB PHY TAP controller is shifting out

[2] **TDO_USB**: USB2 PHY TDO signal

[1] **BIST_OK**: USB PHY bist status
1: No errors detected

[0] **RB_1440**: USB2 PHY PLL state
1: The PLL is locked

SYS_STA1 System status 1 (mode pins values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	MODE														

Address: *SysConfigBaseAddress* + 0x00C

Type: RO

Reset: Not applicable

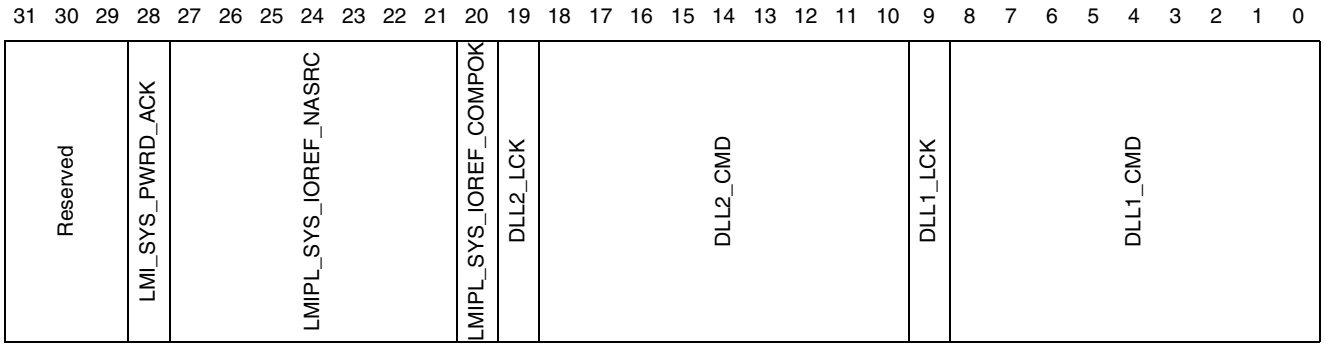
Description: Indicates the mode pins values captured during the reset phase.

[31:15] **Reserved**

[14:0] **MODE**: Values of the mode pins captured during the power-on-reset phase

		Output
[14]	Reserved	-
[13]	Short or long reset signal	Reset generator
[12:11]	Boot mode selection: 0: ST40 boot others: Reserved	System configuration
[10]	Master/slave mode	EMI subsystem
[9:8]	EMI Banks port size at boot	EMI subsystem
[7:6]	TAPmux bypass select	Reserved
[5:4]	Reset bypasses (CPU_RST_OUT_BYPASS[1:0])	Reset generator
[3:2]	PLL1 startup configuration	Clock generator A
[1:0]	PLL0 startup configuration	Clock generator A

SYS_STA12 System status 12: LMI padlogic



Address: *SysConfigBaseAddress* + 0x038

Type: RO

Reset: From padlogic

Description: Returns the status of the system LMI padlogic DLL.

[31:29] **Reserved**

[28] **LMI_SYS_PWRD_ACK**: System LMI power-down acknowledge
1 LMI is in power-down mode.
Reset:0

[27:21] **LMIPL_SYS_IOREF_NASRC**: Compensation code

[20] **LMIPL_SYS_IOREF_COMPOK**
High only in normal mode when a new measured code is available on the ASRC lines.

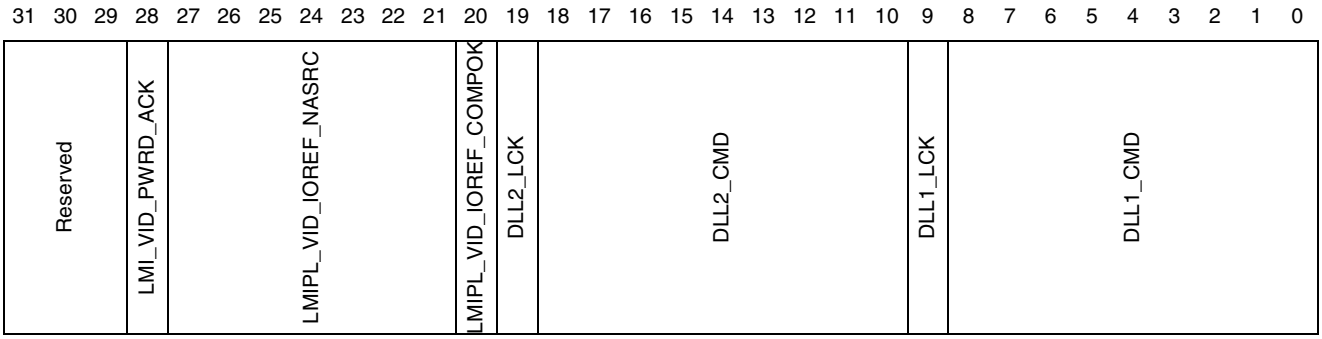
[19] **DLL2_LCK**: DLL2 lock value

[18:10] **DLL2_CMD**: DLL2 command
Reports the command currently being generated by DLL2.

[9] **DLL1_LCK**: DLL1 lock value

[8:0] **DLL1_CMD**: DLL1 command
Reports the command currently being generated by DLL1.

SYS_STA13 System status 13: video LMI padlogic



Address: *SysConfigBaseAddress* + 0x3C

Type: RO

Reset: From padlogic

Description: Returns the state of the video LMI padlogic DLL.

[31:29] **Reserved**

[28] **LMI_VID_PWRD_ACK**: Video LMI power-down acknowledge
1: LMI is in power-down mode.
Reset:0

[27:21] **LMIPL_VID_IOREF_NASRC**: Compensation code

[20] **LMIPL_VID_IOREF_COMPOK**
High only in normal mode when a new measured code is available on the ASRC lines.

[19] **DLL2_LCK**: DLL2 lock value

[18:10] **DLL2_CMD**: DLL2 command
Reports the command currently being generated by DLL2.

[9] **DLL1_LCK**: DLL1 lock value

[8:0] **DLL1_CMD**: DLL1 command
Reports the command currently being generated by DLL1.

21.5 Configuration

SYS_CFG0 System configuration 0: TS merger

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													CFG_TS0_TS1_SEL	INPUT_SEL_MUX	BYPASS_TS

Address: *SysConfigBaseAddress* + 0x100

Type: R/W

Reset: 0

Description: Defines the transport stream merger (TSMerger) inputs configuration.

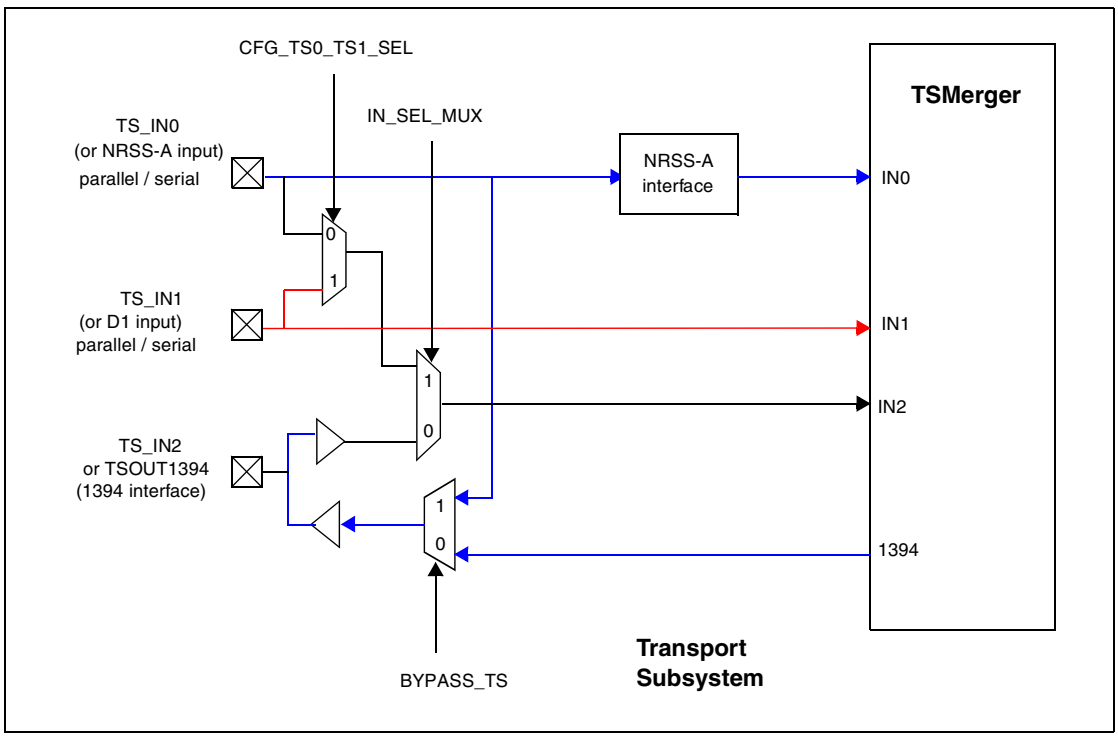
[31:] **Reserved**

[2] **CFG_TS0_TS1_SEL**: Configure TS0 TS1 select
0: TSIN0 routed to TSIN2
1: TSIN1 routed to TSIN2

[1] **IN_SEL_MUX**: Input select multiplex
0: TSMerger TSIN2 receives TSIN2
1: TSMerger TSIN2 receives TSIN0 or TSIN1 depending on CFG_TS0_TS1_SEL

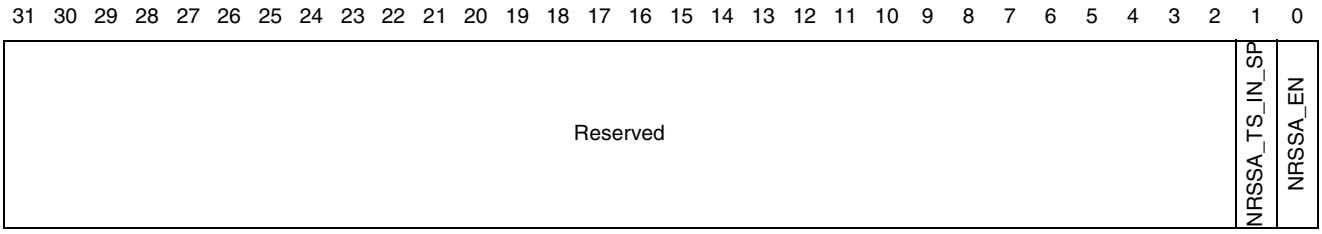
[0] **BYPASS_TS**
0: TS interface is as indicated by TSMerger configuration bits
1: TSIN0 is routed directly to TSOUT1394 output

Figure 58: TSMerger inputs configuration



Note: *See Chapter 38: Transport stream merger and router on page 347.*

SYS_CFG1 System configuration 1: NRSS-A



Address: *SysConfigBaseAddress* + 0x104

Type: R/W

Reset: 0

Description: Configure the NRSS-A interface.

[31:2] **Reserved**

[1] **NRSSA_TS_IN_SP**
0: Bypass mode: TS_IN0 goes directly to the TSmerger
1: NRSS-A interface is enabled: TS_IN0 is formatted and passed through an external NRSS-A module before going to the TSmerger

[0] **NRSSA_EN**
Select TS_IN0 format when NRSS-A is enabled.
0: TS_IN1 in parallel mode
1: TS_IN0 in serial mode

SYS_CFG2 System configuration 2: USB2.0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						SOFT_JTAG_EN	BYPASS_LFSONLY	MSEL_24_30	SEND_IDLE	RX_EN	TX_EN	LOOP_EN	START_BIST	USB_AT	Reserved

Address: *SysConfigBaseAddress* + 0x108

Type: R/W

Reset: 0x81

Description:

[31:10] **Reserved**

[9] **SOFT_JTAG_EN**
1: USB2 and SATA JTAG are controlled by the SOFT_JTAG register.

[8] **BYPASS_LFSONLY**
0: disables REF_CLK_BYPASS_LFSONLY (PLL outputs are used).
1: REF_CLK_BYPASS_LFSONLY will be used for LFS Serdes operation (in FS and LS only modes). REF_CLK_BYPASS_LFSONLY should be connected to a 48 MHz clock in that case.

[7] **MSEL_24_30**
0: selects a 24MHz input clock.
1: selects a 30 MHz input clock.

[6] **SEND_IDLE**
1: the BIST sends IDLE word on TXN (TXP) signals.

[5] **RX_EN**
1: set the input buffer in normal mode.

[4] **TX_EN**
1: set the output buffer in normal mode.

[3] **LOOP_EN**
1: creates an internal loop between the serializer and the deserializer. This loop is implemented before the buffering stage.

[2] **START_BIST**
1: start the PHY BIST.

[1] **USB_AT**
1: set the USB PHY in self-test mode.

[0] **Reserved**

SYS_CFG3 System configuration 3: video DACs and HDMI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL_S_HDMI_MDIV								PLL_S_HDMI_NDIV								PLL_S_HDMI_PDIV			PLL_S_HDMI_EN	S_HDMI_RST_N	Reserved	TST_DAC_HD_CMDR	TST_DAC_HD_CMDS	TST_DAC_SD_CMDR	TST_DAC_SD_CMDS	DAC_HD_HZU	DAC_HD_HZV	DAC_HD_HZW	DAC_SD_HZU	DAC_SD_HZV	DAC_SD_HZW

Address: *SysConfigBaseAddress* + 0x10C

Type: R/W

Reset: 0x3264 4000

Description: Configure the video DACs and HDMI serializer.

[31:24] **PLL_S_HDMI_MDIV**: Set the dividing factor of the 8-bit programmable input divider

[23:16] **PLL_S_HDMI_NDIV**: Set the dividing factor of the 8-bit programmable loop divider

[15:13] **PLL_S_HDMI_PDIV:** Set the dividing factor of the 3-bit programmable output divider

[12] **PLL_S_HDMI_EN**: Determine the mode of operation of the PLL
0: the PLL is powered down.
1: the PLL is running.

[11] **S_HDMI_RST_N**
0: HDMI serializer reset - active low.

[10] **Reserved**

[9] **TST_DAC_HD_CMDR**
Functions with CMDS signals. Can be used to force DAC HD output.

[8] **TST_DAC_HD_CMDS**
Functions with CMDR signal. Can be used to force DAC HD output.

[7] **TST_DAC_SD_CMDR**
Functions with CMDS signals. Can be used to force the DAC SD output.

[6] **TST_DAC_SD_CMDS**
Functions with CMDR signal. Can be used to force the DAC SD output.

[5] **DAC_HD_HZU**
1: Disable the DAC HD output current and put the output in high impedance mode, but leaves the reference circuitry powered for fast recovery to active mode.

[4] **DAC_HD_HZV**
1: Disable the DAC HD output current and put the output in high impedance mode, but leaves the reference circuitry powered for fast recovery to active mode.

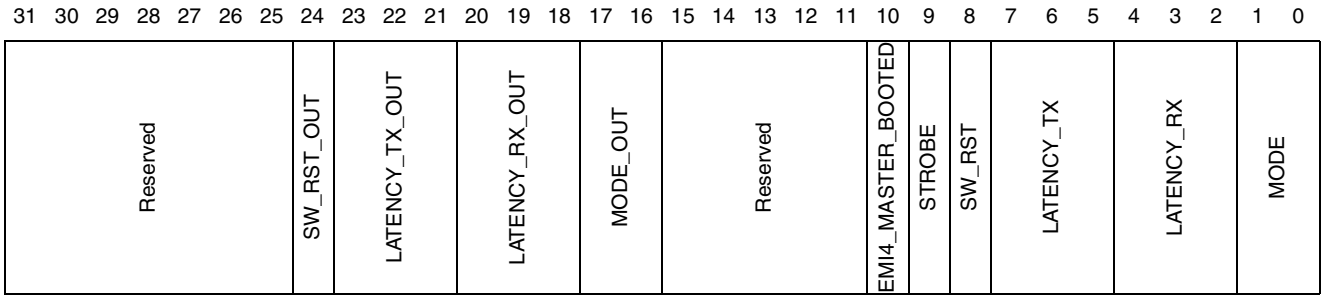
[3] **DAC_HD_HZW**
1: Disable the DAC HD output current and put the output in high impedance mode, but leaves the reference circuitry powered for fast recovery to active mode.

[2] **DAC_SD_HZU**
1: Disable the DAC SD output current and put the output in high impedance mode, but leaves the reference circuitry powered for fast recovery to active mode.

[1] **DAC_SD_HZV**
1: Disable the DAC SD output current and put the output in high impedance mode, but leaves the reference circuitry powered for fast recovery to active mode.

[0] **DAC_SD_HZW**
1: Disable the DAC SD output current and put the output in high impedance mode, but leaves the reference circuitry powered for fast recovery to active mode.

SYS_CFG4 System configuration 4: EMI asynchronous bridge



Address: *SysConfigBaseAddress* + 0x110

Type: Mixed

Reset: 0x0127 0027

Description: Configures the EMI asynchronous bridge.

A rising edge on the strobe signal validates a new bridge configuration. To write a new configuration, the software must write the new configuration (MODE/LATENCY/ SW_RST) while keeping the strobe bit at 0. Setting the strobe bit to 1 creates a rising edge which validates the new setting.

The status of the control bits returned by the bridge can also be read back from this register.

[31:25] - **Reserved**

[24] RO **SW_RST_OUT**: Software reset - active low

[23:21] RO **LATENCY_TX_OUT**: Transmit latency

[20:18] RO **LATENCY_RX_OUT**: Receive latency

[17:16] RO **MODE_OUT**: Mode

[15:11] - **Reserved**

[10] R/W **EMI4_MASTER_BOOTED**
Bit used to gate the request from external slave until the master has completed its boot.

[9] R/W **STROBE**: Strobe to validate a new configuration

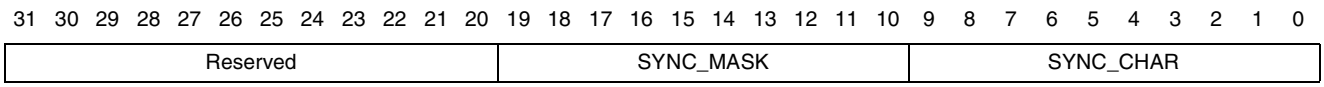
[8] R/W **SW_RST**: Software reset - active high

[7:5] R/W **LATENCY_TX**: Transmit latency value

[4:2] R/W **LATENCY_RX**: Receive latency value

[1:0] R/W **MODE**: Mode pins of memory bridge

SYS_CFG6 System configuration 6: SATA



Address: *SysConfigBaseAddress* + 0x118

Type: R/W

Reset: 0x000F FD7C

Description:

[31:20] **Reserved**

[19:10] **SYNC_MASK**: Defines the 10-bit comma word mask

[9:0] **SYNC_CHAR**: Defines the 10-bit comma word

SYS_CFG7 System configuration 7: PIOs alternate

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			DAA_SER_EN	AUX_NOT_MAIN	DVO_OUT_ON	IRB_DATA_OUT_POL_OD	SC_DETECT_POL	SC_COND_VCC_EN	Reserved	PIO1_SCCLK_NOT_CLKDSS	PIO0_SCCLK_NOT_CLK_DSS	SSC2_MUX_SEL	SSC1_MUX_SEL	SSC0_MUX_SEL	SCIF_PIO_OUT_EN

Address: *SysConfigBaseAddress* + 0x11C
Type: R/W
Reset: 0
Description: COMMs configuration

[31:13] **Reserved**

[12] **DAA_SER_EN**: DAA serializer enable
0: DAA serializer disable 1: DAA serializer enable

[11] **AUX_NOT_MAIN**: VTG signals selection; see Section 7.2: *Alternative functions on page 54*.

[10] **DVO_OUT_ON**: DVO signals selection; see Section 7.2: *Alternative functions on page 54*.

[9] **IRB_DATA_OUT_POL_OD**: IRB polarity selection
0: Polarity of IRB_DATA_OUT_OD is inverted.
1: IRB_DATA_OUT_OD has same polarity as IRB_DATA_OUT.

[8] **SC_DETECT_POL**: SC_DETECT input signal polarity
When configuration bit SC_COND_VCC_EN is set:
0: output SC_NOT_SETVCC = SC_DETECT.
1: output SC_NOT_SETVCC = NOT(SC_DETECT)
If SC_COND_VCC_EN is 0, then this bit has no effect.

[7] **SC_COND_VCC_EN:** Smartcard VCC control upon detection of smartcard removal or insertion
0: Alternate PIO output pin SC_NOT_SETVCC is controlled according to input SC_DETECT.
1: Alternate PIO output pin SC_NOT_SETVCC is driven permanently low.
This bit is overridden by PDES_SC_MUXOUT which is driven by a configuration bit in the PDES IP

[6] **Reserved**

[5] **PIO1_SCCLK_NOT_CLKDSS**: Smartcard clock muxing selection
0: Smartcard clock sourced from smartcard clock generator (COMMs): SC_CLKGEN1_CLK_OUT.
1: Smartcard clock sourced from CLK_DSS (clockgen B).

[4] **PIO0_SCCLK_NOT_CLK_DSS**: Smartcard clock muxing selection
0: Smartcard clock sourced from CLK_DSS (clockgen B).
1: Smartcard clock sourced from smartcard clock generator (COMMs): SC_CLKGEN0_CLK_OUT.

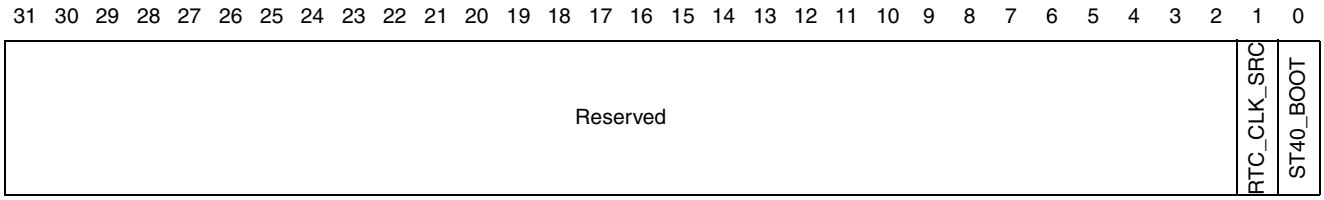
[3] **SSC2_MUX_SEL**: SSC2 muxing selection; see Section 7.2: *Alternative functions on page 54*.
0: On separate PIOs.
1: SSC0_MRST replaces SSC0_MTSR on the corresponding PIO alternate function.

[2] **SSC1_MUX_SEL**: SSC1 muxing selection; see Section 7.2: *Alternative functions on page 54*.
0: Default assignment
1: SSC0_MRST replaces SSC0_MTSR on the corresponding PIO alternate function.

[1] **SSC0_MUX_SEL**: SSC0 muxing selection; see Section 7.2: *Alternative functions on page 54*.
0: Default assignment
1: SSC0_MRST replaces SSC0_MTSR on the corresponding PIO alternate function.

[0] **SCIF_PIO_OUT_EN**
0: Regular PIO 1: Select the SCIF output

SYS_CFG8 System configuration 8: ST40 boot control



Address: *SysConfigBaseAddress* + 0x120

Type: R/W

Reset: 1 when mode[12:11]=00, otherwise 0

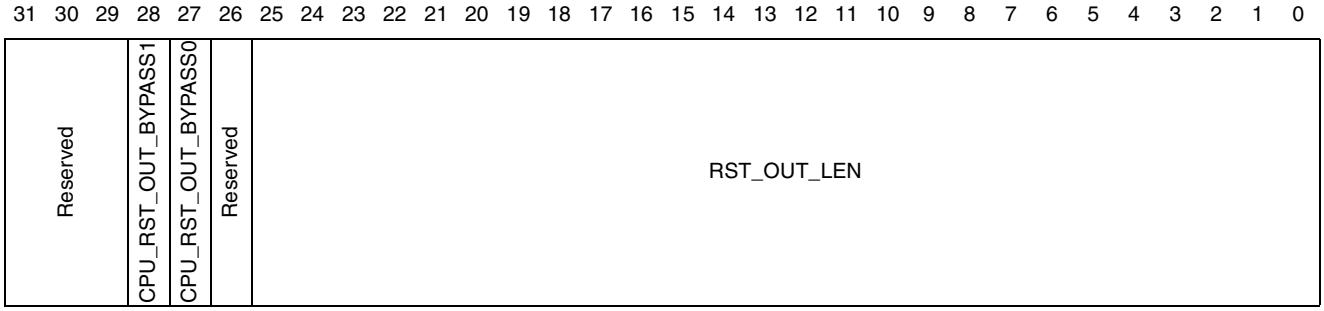
Description:

[31:2] **Reserved**

[1] **RTC_CLK_SRC**: RTC clock source
0: The ST40 RTC sources its clock from the input pin RTCCLKIN
1: The ST40 RTC sources its clock from clock generator B (fixed frequency 32.768 kHz)

[0] **ST40_BOOT**: Controls whether the ST40 is allowed to boot or not after a reset. Its value depends on the mode pin [12] captured during the reset phase.
0: Boot halted 1: Boot enabled

SYS_CFG9 System configuration 9: reset generator



Address: *SysConfigBaseAddress* + 0x124

Type: R/W

Reset: See below

Description: Configures the reset generator.

[31:29] **Reserved**

[28] **CPU_RST_OUT_BYPASS1**: CPU reset out bypass 1
0: The reset signal is propagated through the ST231.
1: Bypass the audio ST231 and Delta ST231 reset out signals.
Reset: from mode pin [5].

[27] **CPU_RST_OUT_BYPASS0**: CPU reset out bypass 0
0: The reset signal is propagated through the ST231s and the ST40
1: Bypass the ST40, Audio ST231 and Delta ST231 reset out signals.
Reset: from mode pin [4].

[26] **Reserved**

[25:0] **RST_OUT_LEN**: Length of WDGRTS_OUT in 27 MHz cycles
In long reset out mode, the reset value guarantees a 200 ms WDGRTS_OUT signal.
In short reset out mode, the WDGRTS_OUT signal lasts 100 µs.
This register allows for a max reset out of 2.48 sec.
Reset: 0x52 65C0 in long reset out mode[a], 0x00 0A8C in short reset out mode

a. Long reset out mode is selected when MODE_PIN[13] is set to '1'.

SYS_CFG10 System configuration 10: SYS_IRQ pins

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												SYS_IRQ3_DIR	SYS_IRQ2_DIR	SYS_IRQ1_DIR	SYS_IRQ0_DIR

Address: *SysConfigBaseAddress* + 0x128

Type: R/W

Reset: 0xF

Description: Configure the SYSITRQ pins as either input or output.

[31:4] **Reserved**

[3] **SYS_IRQ3_DIR**: SYSITRQ3 pin direction
0: SYSITRQ3 configured as output
1: SYSITRQ3 configured as input.

[2] **SYS_IRQ2_DIR**: SYSITRQ2 pin direction
0: SYSITRQ2 configured as output.
1: SYSITRQ2 configured as input.

[1] **SYS_IRQ1_DIR**: SYSITRQ1 pin direction
0: SYSITRQ1 configured as output.
1: SYSITRQ1 configured as input.

[0] **SYS_IRQ0_DIR**: SYSITRQ0 pin direction
0: SYSITRQ0 configured as output.
1: SYSITRQ0 configured as input.

SYS_CFG11 System configuration 11: system and video padlogic

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LMI_VID_PWR_DN_REQ	Reserved	LMI_SYS_PWRD_REQ	BYPASS_DQS_EN	LMI_VID_SWAP_ADD_13_11	LMI_VID_SWAP_ADD_13_10	LMI_SYS_SWAP_ADD_13_11	LMI_SYS_SWAP_ADD_13_10	LMI_VID_HP_AP_EN	LMI_VID_LP_AP_EN	LMI_SYS_HP_AP_EN	LMI_SYS_LP_AP_EN	LMI_VID_PL_RETIME	LMI_SYS_PL_RETIME	LMI_VID_CKG_RSTN	LMI_VID_RST_N	LMI_VID_PBS_IN							Reserved	LMI_SYS_CKG_RSTN	LMI_SYS_RST_N	LMI_SYS_PBS_IN				

Address: *SysConfigBaseAddress* + 0x012C

Type: R/W

Reset: 0x0800 0000

Description: Controls the configuration of the system and video LMI padlogics, reset of the system, video memory controllers and their padlogics. Can also put the memory controllers in power-down mode.

[31] **Reserved**

[30] **LMI_VID_PWR_DN_REQ**: Video LMI - power down request

[29] **Reserved**

[28] **LMI_SYS_PWRD_REQ**: System LMI - power down request

[27] **BYPASS_DQS_EN**
Bypass gating logic for DQS strobe during postamble. Must be set to 1.

[26] **LMI_VID_SWAP_ADD_13_11**
Video LMI - Enable swapping between bit 13 and bit 11 of STBus address.

[25] **LMI_VID_SWAP_ADD_13_10**
Video LMI - Enable swapping between bit 13 and bit 10 of STBus address.

[24] **LMI_SYS_SWAP_ADD_13_11**
System LMI - Enable swapping between bit 13 and bit 11 of STBus address.

[23] **LMI_SYS_SWAP_ADD_13_10**
System LMI - Enable swapping between bit 13 and bit 10 of STBus address.

[22] **LMI_VID_HP_AP_EN**
Video LMI - Enable read with autoprecharge on high priority port.

[21] **LMI_VID_LP_AP_EN**
Video LMI - Enable read with autoprecharge on low priority port.

[20] **LMI_SYS_HP_AP_EN**
System LMI - Enable read with autoprecharge on high priority port.

[19] **LMI_SYS_LP_AP_EN**
System LMI - Enable read with autoprecharge on low priority port.

[18] **LMI_VID_PL_RETIME**
Video LMI - retiming stage enable. Active high

[17] **LMI_SYS_PL_RETIME**: System LMI - retiming stage enable. Active high

[16] **LMI_VID_CKG_RSTN**: Video LMI - padlogic clockgen reset. Active low.

[15] **LMI_VID_RST_N**: Video LMI - subsystem reset and padlogic reset. Active low.

[14:9] **LMI_VID_PBS_IN**: Video LMI - padlogic pin buffer strength.
Set the output buffer drive strength for the SSTL2 pads. Each 2-bits control field has the following encoding:
00: 5 pf 01: 15 pf
10: 25 pf 11: 35 pf
[10:9]: Clock buffer strength. This field selects the drive strength for the CK and CK# pins.
[12:11]: Command buffer strength. This field selects the drive strength for the CKE, CS, RAS, CAS, WE, A and BA DDR pins.
[14:13]: Data buffer strength. This field selects the drive strength for the DQ, DM and DQS DDR pins.

[8] **Reserved**

[7] **LMI_SYS_CKG_RSTN**: System LMI - padlogic clockgen reset. Active low

[6] **LMI_SYS_RST_N**: System LMI - subsystem reset and padlogic reset. Active low.

[5:0] **LMI_SYS_PBS_IN**: System LMI - padlogic pin buffer strength
Set the output buffer drive strength for the SSTL2 pads. Each 2-bit control field has the following encoding:
00: 5 pf 01: 15 pf
10: 25 pf 11: 35 pf
[1:0]: Clock buffer strength. This field select the drive strength for the CK and CK# pins.
[3:2]: Command buffer strength. This field selects the drive strength for the CKE, CS, RAS, CAS, WE, A and BA DDR pins.
[5:4]: Data buffer strength. This field selects the drive strength for the DQ, DM and DQS DDR pins.

SYS_CFG12 System configuration 12: LMI padlogic

31	30	29	28	27	26 25 24	23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4	3 2 1 0
LMI_SYS_IOREF_TQ	LMI_SYS_IOREF_ACC	LMI_SYS_IOREF_FRZ	LMI_SYS_IOREF_COMPTQ	LMI_SYS_IOREF_COMP	LMI_SYS_DLL2_LCK_CTRL	LMI_SYS_IOREF_RASRC	LMI_SYS_DLL1_LCK_CTRL	Reserved	LMI_SYS_DDR_CTRL

Address: *SysConfigBaseAddress* + 0x130

Type: R/W

Reset: 0x4000 0XXD

Description: Control the system LMI DDR padlogic. The drive of the SSTL2 pads, the padlogic DLLs (DLL1 and DLL2) and the compensation cell parameters can be controlled.

[31] **LMI_SYS_IOREF_TQ**: Compensation cell IDDQ mode select
Must be 0 in normal mode
1: puts all IOs and reference generators in IDDQ mode

[30] **LMI_SYS_IOREF_ACC**
Compensation cell accurate mode select (use an external resistor to provide an accurate compensation code). Must be 1 in normal mode.

[29] **LMI_SYS_IOREF_FRZ**: Compensation cell code freeze
Must be 0 in normal mode.
1: Freezes the compensation code at its running value.

[28] **LMI_SYS_IOREF_COMPTQ**: Compensation cell operating mode
Must be 0 in normal mode.

[27] **LMI_SYS_IOREF_COMP**: Compensation cell operating mode
Must be 0 in normal mode.

[26:23] **LMI_SYS_DLL2_LCK_CTRL**: DLL2 lock control
Defines the number of consecutive up/down pulses to trigger the lock status (LCK goes high).

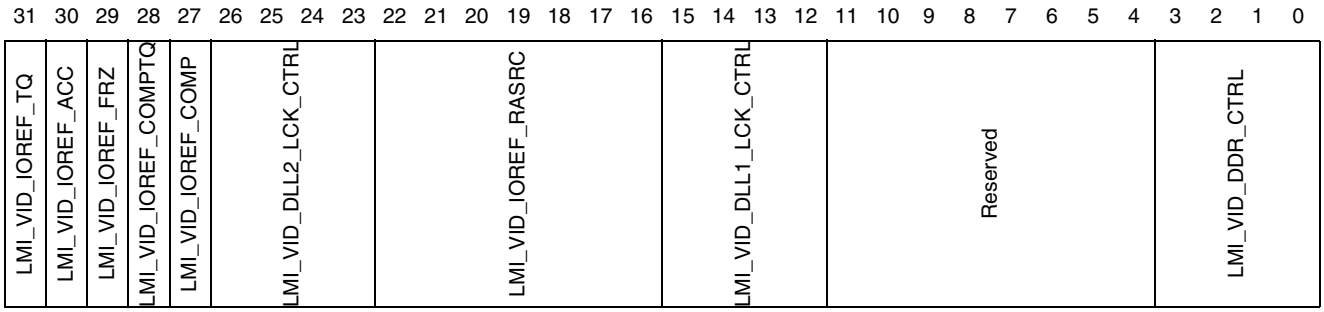
[22:16] **LMI_SYS_IOREF_RASRC**: System LMI compensation cell input code

[15:12] **LMI_SYS_DLL1_LCK_CTRL**: DLL1 lock control
Defines the number of consecutive up/down pulses trigger the lock status (LCK goes high).

[11:4] **Reserved**

[3:0] **LMI_SYS_DDR_CTRL**: System LMI DDR Pad control:
[0] Output buffer impedance (I/O ZOUTPROGA)
0: 25 Ω (Strong SSTL2) 1: 40 Ω (Weak SSTL2)
[1] Buffer mode select (I/O MODE)
0: 2.5V SSTL2 with 25/40 Ω impedance. 1: 2.5V Matched impedance with 40 Ω
[2] VREF mode select (I/O ENVREF)
0: VREF is generated internally for each pad. 1: VREF is supplied externally.
[3]: Receiver mode select (I/O MODEZI)
0: Differential 2.5V receiver. 1: 2.5V Digital CMOS receiver.

SYS_CFG13 System configuration 13: video LMI padlogic



Address: *SysConfigBaseAddress* + 0x134

Type: R/W

Reset: 0x4000 0XXD

Description: Control the video LMI DDR padlogic. The drive of the SSTL2 pads, the padlogic DLLs (DLL1 and DLL2) and the compensation cell parameters can be controlled.

[31] **LMI_VID_IOREF_TQ**: Compensation cell IDDQ mode select
Must be 0 in normal mode.

[30] **LMI_VID_IOREF_ACC**: Compensation cell accurate mode select
Must be 1 in normal mode.

[29] **LMI_VID_IOREF_FRZ**: Compensation cell code freeze
Must be 0 in normal mode.

[28] **LMI_VID_IOREF_COMPTQ**: Compensation cell operating mode
Must be 0 in normal mode.

[27] **LMI_VID_IOREF_COMP**: Compensation cell operating mode
Must be 0 in normal mode.

[26:23] **LMI_VID_DLL2_LCK_CTRL**: DLL2 lock control
Defines the number of consecutive up/down pulses trigger the lock status (LCK goes high).

[22:16] **LMI_VID_IOREF_RASRC**: Compensation cell input code

[15:12] **LMI_VID_DLL1_LCK_CTRL**: DLL1 lock control
Defines the number of consecutive up/down pulses trigger the lock status (LCK goes high).

[11:4] **Reserved**

[3:0] **LMI_VID_DDR_CTRL**: Video LMI DDR pad control

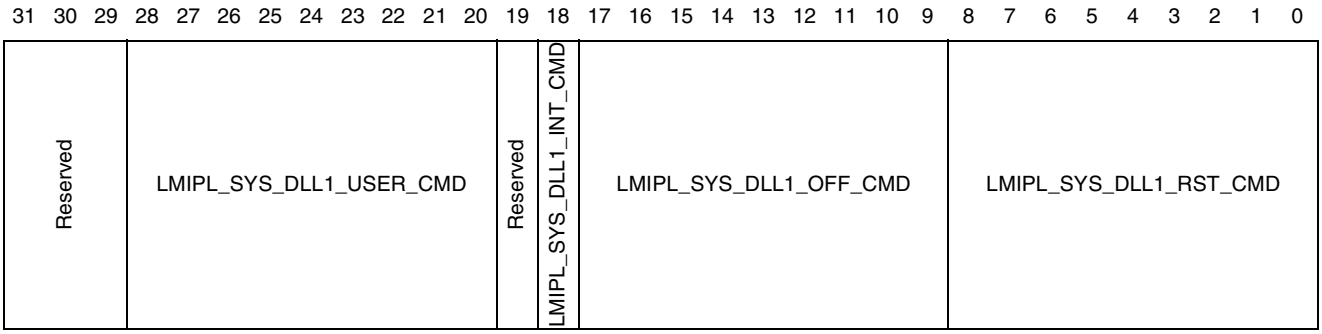
[3] **Receiver mode select** (I/O MODEZI)
0: Differential 2.5V receiver. 1: 2.5V Digital CMOS receiver

[2] **VREF mode select** (I/O ENVREF)
0: VREF is generated internally for each pad. 1: VREF is supplied externally

[1] **Buffer mode select** (I/O MODE)
0: 2.5V SSTL2 with 25/40 Ω impedance. 1: 2.5V Matched impedance with 40 Ω

[0] **Output buffer impedance** (I/O ZOUTPROGA)
0: 25 Ω (Strong SSTL2) 1: 40 Ω (Weak SSTL2)

SYS_CFG14 System configuration 14: LMI padlogic DLL1



Address: *SysConfigBaseAddress* + 0x138

Type: R/W

Reset: 0

Description: Controls the delay generated by the DLL1 of the system LMI padlogic. The DLL1 generates only internal delay commands. It controls the timings of the write-path to the DDR memory.

[31:29] **Reserved**

[28:20] **LMIPL_SYS_DLL1_USER_CMD**
User delay command used by DLL1 when bit 18 is 1.

[19] **Reserved**

[18] **LMIPL_SYS_DLL1_INT_CMD**: Controls which internal delay command is used by DLL1
0: FSM-generated command
1: user command

[17:9] **LMIPL_SYS_DLL1_OFF_CMD**
DLL1 offset value added to the FSM generated delay command. Can be negative and must be coded in twos complement.

[8:0] **LMIPL_SYS_DLL1_RST_CMD**
DLL1 command from which the FSM start to count-up after reset.

SYS_CFG15 System configuration 15: LMI padlogic DLL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LMIPL_SYS_DLL2_USER_CMD									LMIPL_SYS_DLL2_EXT_CMD	LMIPL_SYS_DLL2_INT_CMD	LMIPL_SYS_DLL2_OFF_CMD									LMIPL_SYS_DLL2_RST_CMD								

Address: *SysConfigBaseAddress* + 0x13C

Type: R/W

Reset: 0

Description: Controls the delay generated by the DLL2 of the system LMI padlogic. The DLL2 delay can be generated by either internal and external commands. This delay from the DLL2 is used to control the read-path timings the from the DDR memory.

[31:29] **Reserved**

[28:20] **LMIPL_SYS_DLL2_USER_CMD**: User delay command used by DLL2 when bit 18 and/or bit 19 is 1

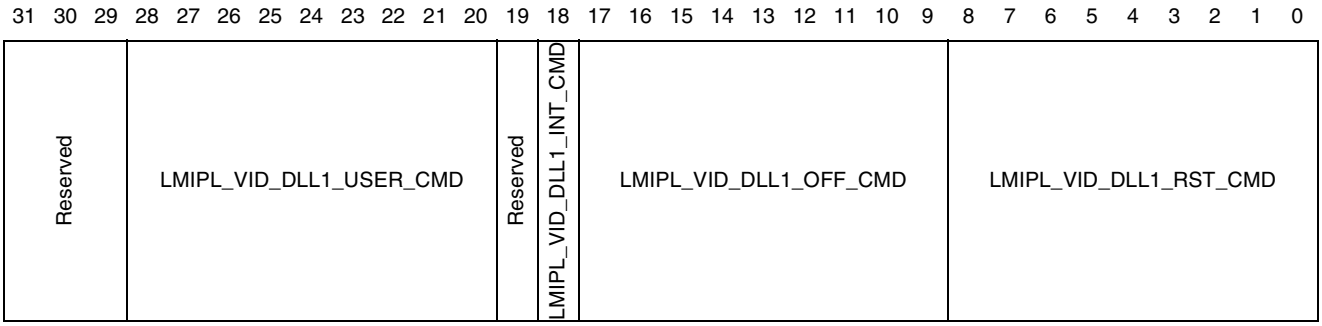
[19] **LMIPL_SYS_DLL2_EXT_CMD**: Controls which external delay command is used for DLL2
0: FSM-generated delay command
1: user external delay command

[18] **LMIPL_SYS_DLL2_INT_CMD**: Controls which internal delay command is used for DLL2
0: FSM-generated delay command
1: user internal delay command

[17:9] **LMIPL_SYS_DLL2_OFF_CMD**: DLL2 offset value added to the FSM generated delay command
Can be negative and must be coded in 2's complement.

[8:0] **LMIPL_SYS_DLL2_RST_CMD**: DLL2 command from which the FSM start to count-up after reset

SYS_CFG20 System configuration 20: video LMI padlogic DLL1



Address: *SysConfigBaseAddress* + 0x150

Type: R/W

Reset: 0

Description: Controls the delay generated by the DLL1 of the video LMI padlogic and the timings of the write-path to the DDR memory. The DLL1 generates only internal delay commands. controls.

[31:29] **Reserved**

[28:20] **LMIPL_VID_DLL1_USER_CMD**: User delay command used by DLL1 when the bit 18 is 1.

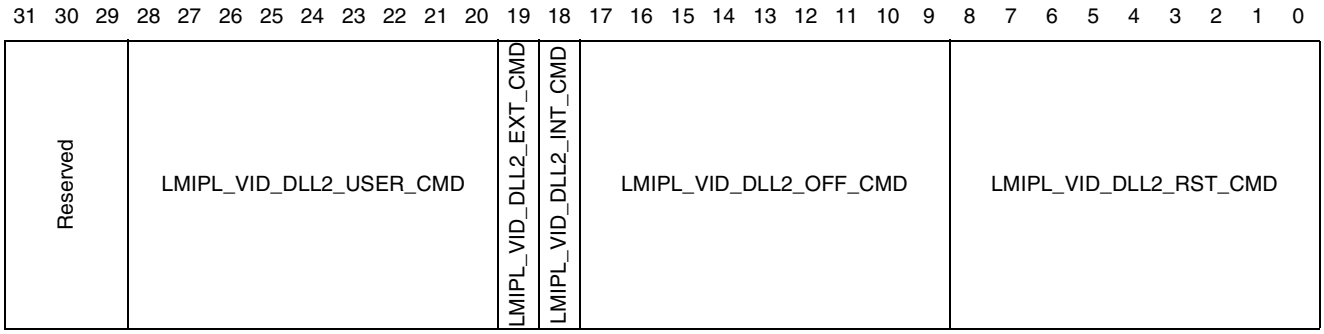
[19] **Reserved**

[18] **LMIPL_VID_DLL1_INT_CMD**: Controls which internal delay command is used by DLL1
0: FSM-generated command
1: user command

[17:9] **LMIPL_VID_DLL1_OFF_CMD**
DLL1 offset value added to the FSM generated delay command. Can be negative and must be coded in 2's complement.

[8:0] **LMIPL_VID_DLL1_RST_CMD**: DLL1 command from which the FSM start to count-up after reset.

SYS_CFG21 System configuration 21: video LMI padlogic DLL2



Address: *SysConfigBaseAddress* + 0x154

Type: R/W

Reset: 0

Description: Controls the delay generated by the DLL2 of the system LMI padlogic and timings of the read-path from the DDR memory. The DLL2 generates internal and external delay commands.

[31:29] **Reserved**

[28:20] **LMIPL_VID_DLL2_USER_CMD**
User delay command used by DLL2 when bit 18 and/or bit 19 is 1

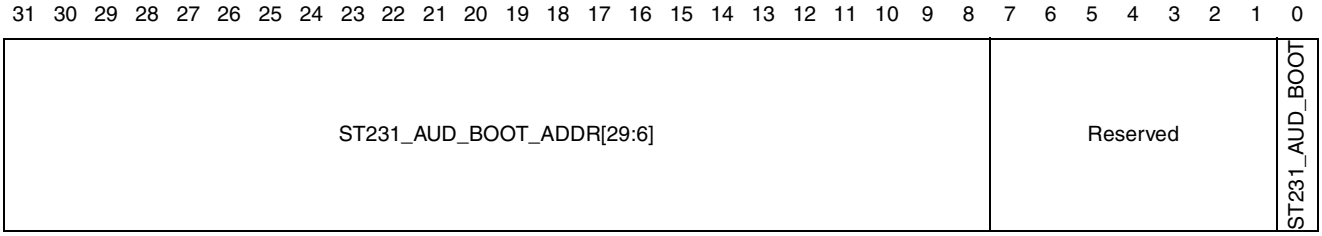
[19] **LMIPL_VID_DLL2_EXT_CMD**
Controls which external delay command is used for DLL2.
0: FSM-generated delay command
1: user external delay command

[18] **LMIPL_VID_DLL2_INT_CMD**
Controls which internal delay command is used for DLL2.
0: FSM-generated delay command
1: user internal delay command

[17:9] **LMIPL_VID_DLL2_OFF_CMD**
DLL2 offset value added to the FSM generated delay command. Can be negative and must be coded in 2's complement.

[8:0] **LMIPL_VID_DLL2_RST_CMD**
DLL2 command from which the FSM start to count-up after reset.

SYS_CFG26 System configuration 26: audio ST231 boot control



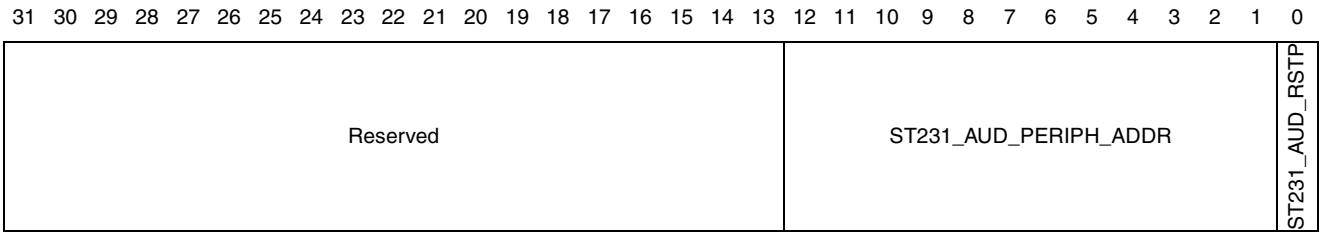
Address: *SysConfigBaseAddress* + 0x
Type: R/W
Reset: 0
Description: Allows the audio ST231 to boot and to define its boot address.

[31:8] **ST231_AUD_BOOT_ADDR**[29:6]: Audio ST231 boot address

[7:1] **Reserved**

[0] **ST231_AUD_BOOT**: Audio ST231 boot enable
0: boot halted
1: boot enabled

SYS_CFG27 System configuration 27: audio ST231 reset



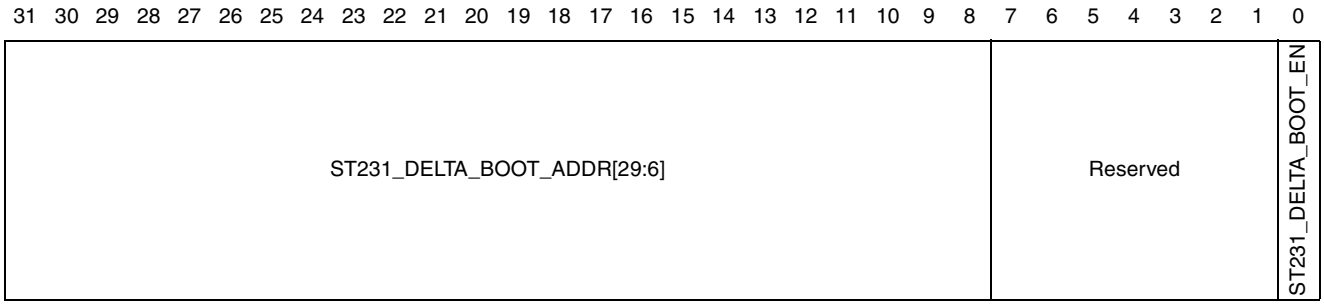
Address: *SysConfigBaseAddress* + 0x1C
Type: R/W
Reset: 0x344
Description: do a software reset of the audio ST231. When the boot-address is changed then the ST231 must be reset to take into account its new boot-address.

[31:13] **Reserved**

[12:1] **ST231_AUD_PERIPH_ADDR**: Audio ST231 peripherals address (must not be changed).

[0] **ST231_AUD_RSTP**: Audio ST231 reset control:
0: ST231 reset input is driven by the hardware reset
1: ST231 reset is asserted

SYS_CFG28 System configuration 28: Delta ST231 boot



Address: *SysConfigBaseAddress* + 0x170

Type: R/W

Reset: 0

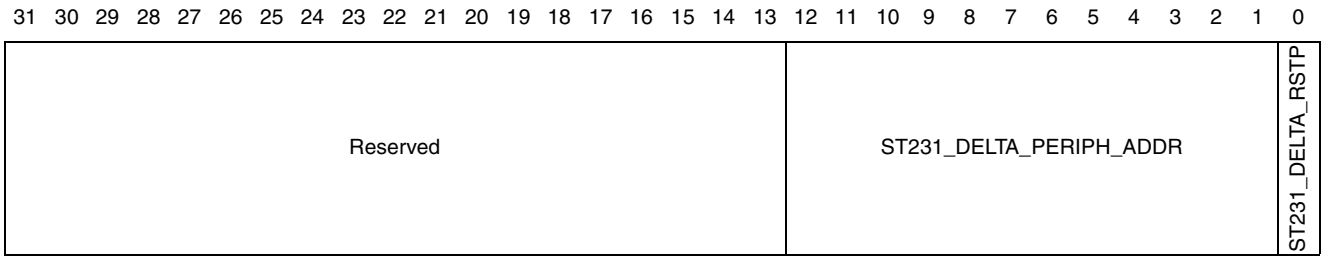
Description: Enables the Delta ST231 to boot and defines its boot address.

[31:8] **ST231_DELTA_BOOT_ADDR**[29:6]

[7:1] **Reserved**

[0] **ST231_DELTA_BOOT_EN**: Delta ST231 boot enable
0: Boot halted
1: Boot enabled

SYS_CFG29 System configuration 29: Delta ST231 reset



Address: *SysConfigBaseAddress* + 0x174

Type: R/W

Reset: 0x340

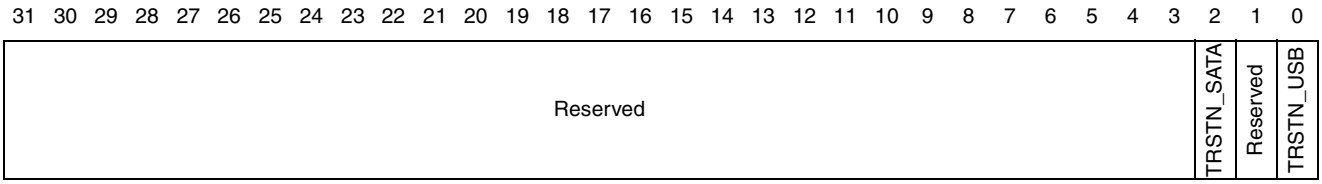
Description: Perform a software reset of the Delta ST231. When the boot-address is changed then the ST231 must be reset to take into account its new boot address.

[31:13] **Reserved**

[12:1] **ST231_DELTA_PERIPH_ADDR**: Delta ST231 peripherals address (must not be changed).

[0] **ST231_DELTA_RSTP**: Delta ST231 reset control:
0: ST231 reset input is driven by the hardware reset
1: ST231 reset is asserted

SYS_CFG33 System configuration 33: USB and SATA software JTAG



Address: *SysConfigBaseAddress* + 0x184

Type: R/W

Reset: 0

Description: Controls the JTAG port of the SATA and USB PHY. A software JTAG controller can be implemented in software.

[31:3] **Reserved**

[2] **TRSTN_SATA**: Asynchronous reset SATA TAP

[1] **Reserved**

[0] **TRSTN_USB**: Asynchronous reset USB2 TAP

22 Interrupt system

The STx7100 has two interrupt networks. One is associated with the ST40 CPU and the other is associated with the Delta (H.264) ST231 CPU when it used as an application processor. The interrupt lines are routed to both the ST40 and the Delta ST231. For both processors, it is up to software to handle the interrupts.

22.1 ST40 Interrupt network

22.1.1 Internal and external interrupts

The ST40 CPU has two types of interrupts:

External Interrupts

- NMI (Non-Maskable Interrupt): External interrupt source.
- SYSITRQ(3-0): 4 external interrupt sources SYSITRQ(3-0) which can be configured as 4 independent interrupts or encoded to provide 15 external interrupt levels.

These interrupts are managed by the INTC2 interrupt controller integrated into the ST40 CPU core.

Internal Peripherals Interrupts

- ST40-P130 peripherals interrupts:
  - UDI (user debug interface)
  - TMU0, 1 (timer unit)
  - RTC (real time clock)
  - SCIF (serial controller interface)

  These interrupts are controlled by the interrupt controller INTC2, and expansion of the INTC. All interrupts are assigned a priority level between 0 and 15: level 15 is the highest and level 1 the lowest, level 0 means that the interrupt is masked. The NMI is defined to have a fixed priority level of 16.
- On-chip peripherals interrupts

  These interrupts are managed by the INTC2 (interrupt controller).

  The INTC2 accepts 16 groups of 4 interrupts (64 total); each group can be assigned a priority by software (INTPRI*xx* registers). Within each group (of 4 interrupts), there is a fixed priority, with interrupt 4 having the highest priority. All interrupts are resynchronized in INTC2.

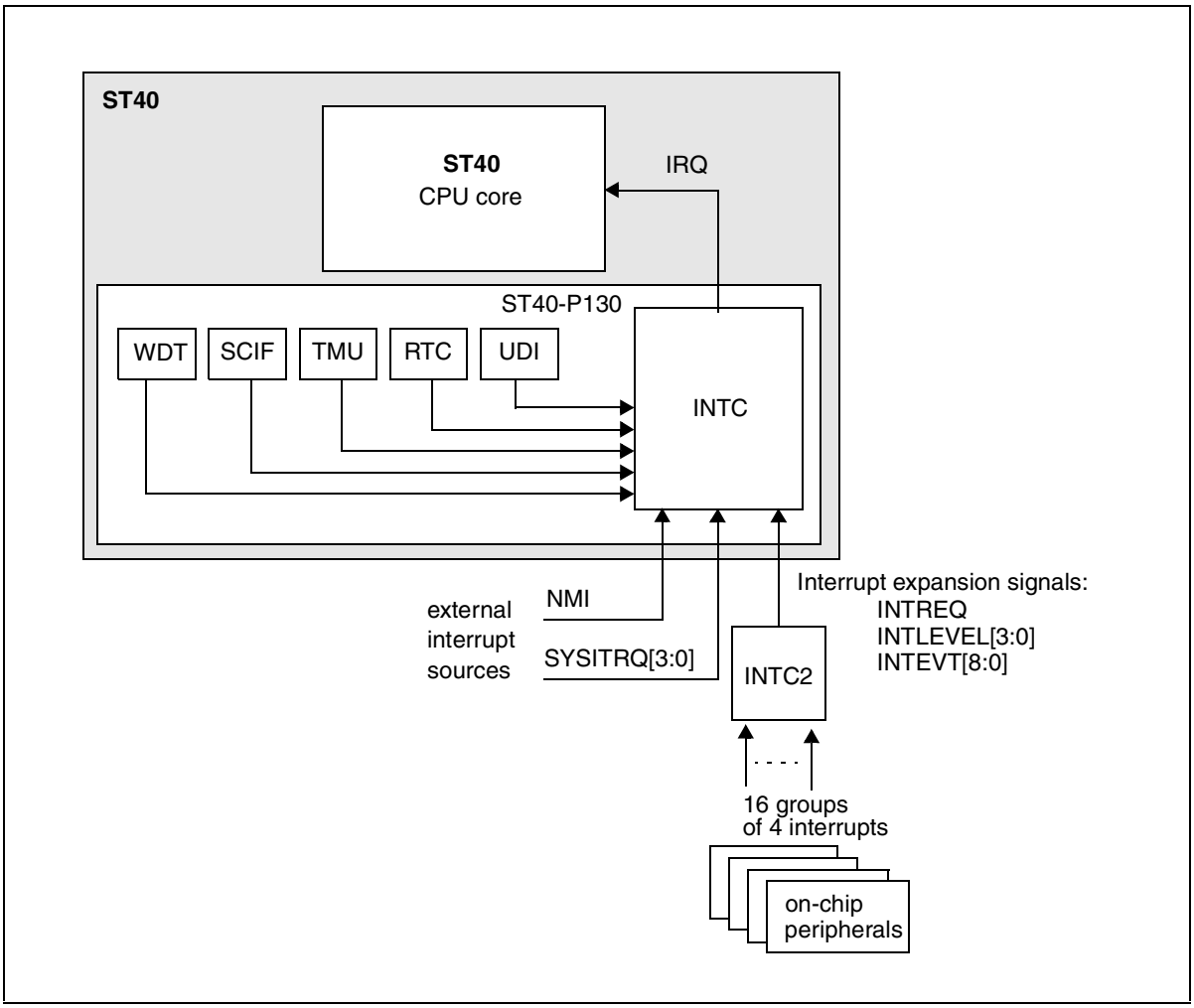
22.1.2 Interrupt service routine address

Whenever an interrupt occurs, the ST40 CPU branches to the interrupt handling vector address determined by adding the fixed offset 0x600 to the vector base address (VBR) register. Each interrupt type is assigned a code which is stored in the INTEVT (interrupt event) register when the interrupt occurs. This enables the interrupt service routine to identify the interrupt source type.

The interrupt controller is responsible for mapping each interrupt to its code (INTEVT).

Figure 59 shows the ST40 interrupt network, and Table 83: *ST40 on-chip peripheral interrupts on page 201* lists the internal interrupts with their INTEVT code.

Figure 59: ST40 interrupt network



22.1.3 INTC2 secondary interrupt controller

The INTC2 selects the highest priority, unmasked interrupt from all of its interrupt inputs, and passes the INTEVT code of this interrupt to the INTC2 for further processing.

The INTC2 allows interrupts to be grouped together. These groups contain one or more interrupts. For each interrupt within a group, and for each group, the INTC2 provides the following:

Each interrupt group is allocated a four-bit section of the INTC2_PRI0*n* register. This gives the group its overall interrupt priority. Within each group, each individual interrupt has its own priority. (By setting a group's priority to 0x0, the whole group can be masked).

Each interrupt is allocated a single bit of the INTC2_REQ0*n* and INTC2_MSK0*n* registers. Register INTC2_REQ0*n* shows the status of an individual interrupt request signal irrespective of the state of register INTC2_MSK0*n*. The INTC2_MSK0*n* register allows each individual interrupt to be separately masked.

Each priority group of interrupts asserts its highest priority interrupt. A tree structure is used to compare all asserted interrupts to determine the highest priority interrupt, which is then passed to the INTC. The INTEVT code which is sent to the INTC.

22.2 Delta (H.264) ST231 interrupt network

The ST231 accepts 60 external interrupts (from 63 to 3). Interrupts 0 to 2 are reserved for the ST231 internal timers. All the interrupts are maskable but with a single level of priority. Multiple level priority must be implemented in software.

Table 85: *Delta ST231 Interrupts* *on page 204* describes the mapping of the interrupts on the ST231 interrupt controller.

When used as an application processor, the Delta ST231 processor receives the same internal interrupts than the ST40 processor except the interrupts generated by the Delta coprocessors (CABAC preprocessor 1 and 2).

The Delta ST231 receives also the external interrupts via the ILC interrupt controller.

22.3 STx7100 interrupt network

ILC interrupt level controller

The STx7100 interrupt network includes an ILC interrupt level controller. The ILC accepts 64 synchronous interrupt inputs (the on-chip peripherals interrupts) and six asynchronous interrupt inputs (the 4 external interrupts SYSITRQ(3-0), NMI and the IRB wake-up interrupt). The external asynchronous interrupts can have up to 5 programmable triggering conditions (active high, active low, falling edge, rising edge or any edge).

The ILC can map any of the synchronous internal interrupts and asynchronous interrupts onto a group of 8 internal interrupts ILC_EXT_OUT(7-0).

The interrupts ILC_EXT_OUT(3-0) can be used as external interrupt outputs through the SYSITRQ(3-0) pins.

The interrupts ILC_EXT_OUT(7-4) are routed to the ST40 interrupts IRL(3-0).

The ILC mapping is described in Table 86: *ILC interrupt mapping* *on page 206*.

Wake up by Interrupt

The ILC has also an interrupt output dedicated to the wake-up process that is used by the low power controller. A pulse stretcher receives a transition from the UHF and IR input pins and generates an interrupt that can be routed through the ILC to one of the ILC_EXT_OUT(7-4) interrupt lines and used as a wake-up interrupt to the ST40 (for details, see Chapter 18: *Low-power modes* *on page 164*).

Internal peripheral interrupts

Both the ST40 and the Delta ST231 receive all the internal interrupt requests. It is up to the software to define the CPU that will serve each interrupt request.

All the internal interrupts are routed to the INTC2, ST231 Interrupt controller and ILC. All the internal interrupts are level sensitive and active high.

External interrupts inputs

The four external asynchronous interrupts are routed to the ILC interrupt controller before reaching the ST40 and ST231 in order to synchronize and change the polarity if needed. The ST40 expects the interrupts to be active high. These interrupts are then output from the ILC (ILC_EXT_OUT(7-4)) to be routed toward the ST40 and Delta ST231.

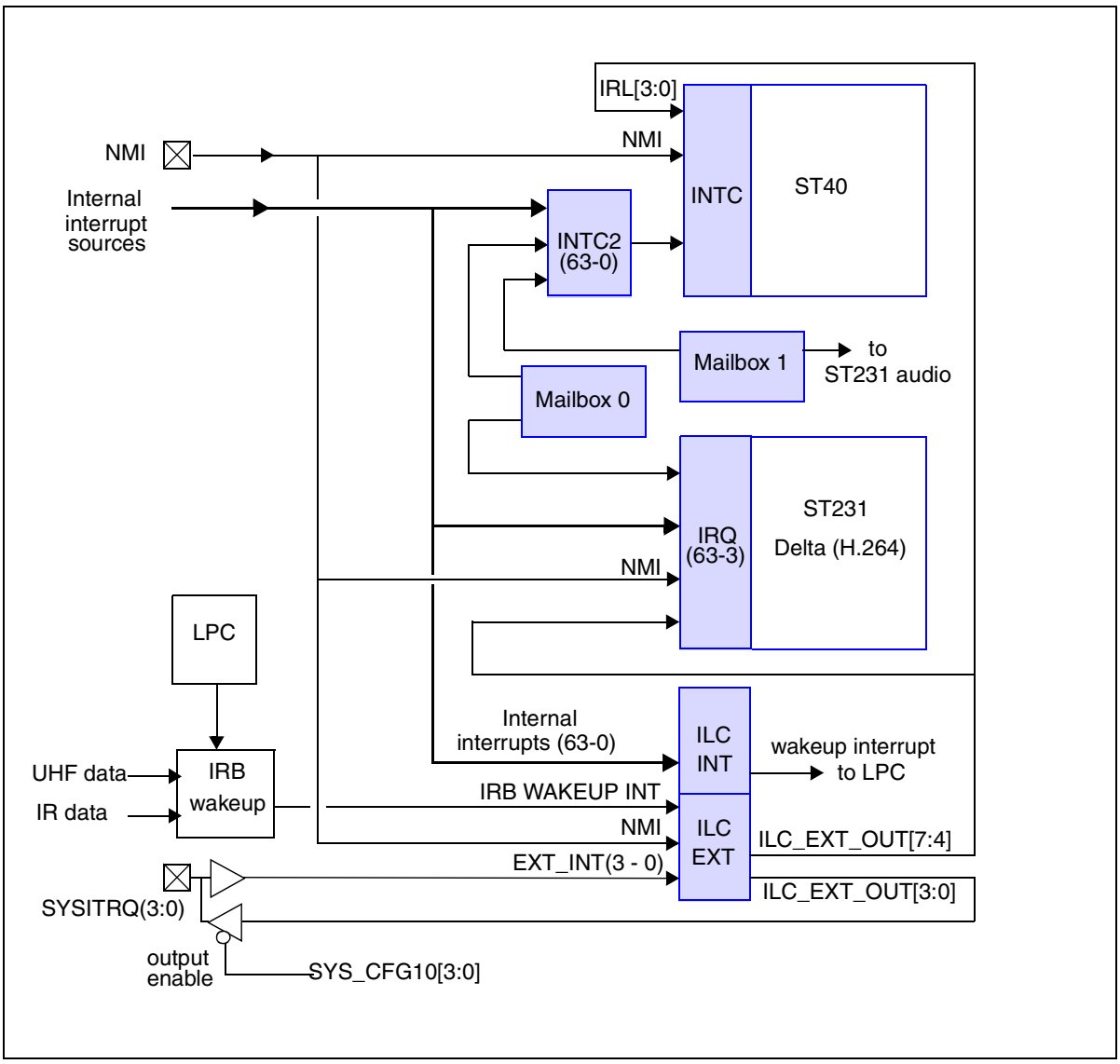
External interrupts outputs

The ILC has the capability to output a subset of the interrupts that are connected to it. Four of these interrupts (ILC_EXT_OUT[3 - 0]) are software selectable to be output externally for remote devices.

Control of the external interrupts direction

The direction of external interrupts is controlled by the configuration register SYS_CFG10[3:0]. By default, pins SYSITRQ(3-0) are configured as inputs (SYS_CFG10 bits set to logic 1). Setting these bits to 0 configures the pins as outputs.

Figure 60: STx7100 interrupt network



23 Interrupt maps

This chapter contains Interrupt mapping tables for the:

- ST40,
- INTC2
- Delta (H.264) ST231,
- STx7100 ILC.

23.1 ST40 interrupts

Table 82: ST40 P130 interrupts

Interrupt source		INTEVT code	IPR (bit numbers)	Interrupt priority (initial value)
NMI		0x1C0	-	16
IRL independent encoding	IRL0	0x240	IPRD[15:12]	15-0 (13)
	IRL1	0x2A0	IPRD[11:8]	15-0 (10)
	IRL2	0x300	IPRD[7:4]	15-0 (7)
	IRL3	0x360	IPRD[3:0]	15-0 (4)
IRL	level encoding	0x200 - 0x3C0	-	1-15
TMU0	TUNI0	0x400	IPRA[15:12]	15-0 (0)
TMU1	TUNI1	0x420	IPRA[11:8]	15-0 (0)
TMU2	TUNI2	0x440	IPRA[7:4]	15-0 (0)
	TICPI2	0x460		
RTC	ATI	0x480	IPRA[3:0]	15-0 (0)
	PRI	0x4A0		
	CUI	0x4C0		
SCIF	ERI	0x4E0	IPRB[7:4]	15-0 (0)
	RXI	0x500		
	BRI	0x520		
	TXI	0x540		
WDT	ITI	0x560	IPRB[15:12]	15-0 (0)
UDI	H-UDI	0x600	IPRC[3:0]	15-0 (0)

Table 83: ST40 on-chip peripheral interrupts

Group	Interrupt source		INTEVT code	IPR (bit numbers)	INTREQ/ INTMSK (bit number)
-	Reserved		0xA00	INTPRI00[3:0] INTPRI00[7:4]	INTREQ00[0]
			0xA20		INTREQ00[1]
			0xA40		INTREQ00[2]
			0xA60		INTREQ00[3]
			0xA80		INTREQ00[4]
			0xB00	INTPRI00[11:8]	INTREQ00[5]
			0xB20		INTREQ00[6]
			0xB40		INTREQ00[7]
			0xB60		INTREQ00[8]
			0xB80		INTREQ00[9]
			0xBA0		INTREQ00[10]
			0xBC0		INTREQ00[11]
-	COMMs/PIO	PIO0_INT	0xC00	INTPRI00[15:12]	INTREQ00[12]
		PIO1_INT	0xC80	INTPRI00[19:16]	INTREQ00[13]
		PIO2_INT	0xD00	INTPRI00[23:20]	INTREQ00[14]
0	COMMs/PIO	PIO3_INT	0x1060	INTPRI04[3:0]	INTREQ04[3]
		PIO4_INT	0x1040		INTREQ04[2]
		PIO5_INT	0x1020		INTREQ04[1]
		Reserved	0x1000		
1	SSC	SSC0_INT	0x10E0	INTPRI04[7:4]	INTREQ04[7]
		SSC1_INT	0x10C0		INTREQ04[6]
		SSC2_INT	0x10A0		INTREQ04[5]
		Reserved	0x1080		
2	COMMs/UART	UART0_INT	0x1160	INTPRI04[11:8]	INTREQ04[11]
		UART1_INT	0x1140		INTREQ04[10]
		UART2_INT	0x1120		INTREQ04[9]
		UART3_INT	0x1100		INTREQ04[8]
3	COMMs/MAFE	MAFE_INT	0x11E0	INTPRI04[15:12]	INTREQ04[15]
	COMMs/PWM	PWM_INT	0x11C0		INTREQ04[14]
	COMMs/IRB	IRB_INT	0x11A0		INTREQ04[13]
		IRB_WAKEUP_INT	0x1180		INTREQ04[12]
4	COMMs/TTXT-DAA-DiSEqC	TTXT_INT	0x1260	INTPRI04[19:16]	INTREQ04[19]
		DAA_INT	0x1240		INTREQ04[18]
		DISQEC_INT	0x1220		INTREQ04[17]
		Reserved	0x1200		-
5	SBAG	SBATM_INT	0x12E0	INTPRI04[23:20]	INTREQ04[23]
		TSMRGR_RAMOVF_INT (cut 2)	0x12C0		-
		Reserved	0x12A0		-
			0x1280		

Table 83: ST40 on-chip peripheral interrupts

Group	Interrupt source		INTEVT code	IPR (bit numbers)	INTREQ/ INTMSK (bit number)
6	CLOCKGEN	Reserved	0x1360	INTPRI04[27:24]	-
		DCXO_INT	0x1340		INTREQ04[26]
	MAILBOXes	ST231_AUD_INT	0x1320		INTREQ04[25]
		ST231_DELTA_INT	0x1300		INTREQ04[24]
7	AUDIO	CPXM_INT	0x13E0	INTPRI04[31:28]	INTREQ04[31]
		I2S2SPDIF_INT	0x13C0		INTREQ04[30]
	FDMA	FDMA_GP0_INT	0x13A0		INTREQ04[29]
		FDMA_MBOX_INT	0x1380		INTREQ04[28]
8	AUDIO	SPDIFPLYR_INT	0x1460	INTPRI08[3:0]	INTREQ08[3]
		PCMRDR_INT	0x1440		INTREQ08[2]
		PCMPLYR1_INT	0x1420		INTREQ08[1]
		PCMPLYR0_INT	0x1400		INTREQ08[0]
9	VIDEO	DELTA_MBE_INT	0x14E0	INTPRI08[7:4]	INTREQ08[7]
		DELTA_PRE1_INT	0x14C0		INTREQ08[6]
		DELTA_PRE0_INT	0x14A0		INTREQ08[5]
		GLH_INT	0x1480		INTREQ08[4]
10	VOS/VTGs	VTG2_INT	0x1560	INTPRI08[11:8]	INTREQ08[11]
		VTG1_INT	0x1540		INTREQ08[10]
	DISPLAY/LMU	LMU_INT	0x1520		INTREQ08[9]
		Reserved	0x1500	-	-
11	HDMI/HDCP	HDCP_INT	0x15E0	INTPRI08[15:12]	INTREQ08[15]
		HDMI_INT	0x15C0		INTREQ08[14]
	DVP	DVP_INT	0x15A0		INTREQ08[13]
	GAMMA	BLT_INT	0x1580		INTREQ08[12]
12	PDES	Reserved	0x1660	INTPRI08[19:16]	-
		PDES_INT	0x1640		INTREQ08[18]
	PTI	Reserved	0x1620		-
		PTI_INT	0x1600		INTREQ08[16]
13	CryptoCore	Reserved	0x16E0	INTPRI08[23:20]	-
		SEC_CP_INT	0x16C0		INTREQ08[22]
		DMA_FIN_INT	0x16A0		INTREQ08[21]
		SIG_CHK_INT	0x1680		INTREQ08[20]
14	SATA	Reserved	0x1760	INTPRI08[27:24]	
		SATA_INT	0x1740		INTREQ08[26]
	USBH	EHCI_INT	0x1720		INTREQ08[25]
		OHCI_INT	0x1700		INTREQ08[24]
15	Reserved		0x17E0	INTPRI08[31:28]	-
			0x17C0		
			0x17A0		
			0x1780		

23.2 INTC2 secondary interrupt controller

23.2.1 Priority group and bit allocation of interrupts

Each individual interrupt is allocated a bit position within the INTC2_REQ, INTC2_MSK and INTC2_MSK_CLR registers, and is also allocated to a priority group within the INT_PRI register.

23.2.2 Priority groups and bit allocation of extended interrupts

Table 84: INTC2 extended interrupts register allocation

Interrupt group	Register allocation
1	INTxxx04[3:0]
2	INTxxx04[7:4]
3	INTxxx04[11:8]
4	INTxxx04[15:12]
5	INTxxx04[19:16]
6	INTxxx04[23:20]
7	INTxxx04[27:24]
8	INTxxx04[31:28]
9	INTxxx08[3:0]
10	INTxxx08[7:4]
11	INTxxx08[11:8]
12	INTxxx08[15:12]
13	INTxxx08[19:16]
14	INTxxx08[23:20]
15	INTxxx08[27:24]
16	INTxxx08[31:28]

23.3 Delta (H.264) ST231 interrupts

Table 85: Delta ST231 Interrupts

Interrupt source		INT number
ST231 Timers	TIMER0_INT	0
	TIMER1_INT	1
	TIMER2_INT	2
	Reserved	3
COMMs/PIO	PIO0_INT	4
	PIO1_INT	5
	PIO2_INT	6
	PIO3_INT	7
	PIO4_INT	8
	PIO5_INT	9
SSC	SSC0_INT	10
	SSC1_INT	11
	SSC2_INT	12
	Reserved	13
COMMs/UART	UART0_INT	14
	UART1_INT	15
	UART2_INT	16
	UART3_INT	17
COMMs/MAFE	MAFE_INT	18
COMMs/PWM	PWM_INT	19
COMMs/IRB	IRB_INT	20
	IRB_WAKEUP_INT	21
COMMs/TTXT	TTXT_INT	22
COMMS/DAA	DAA_INT	23
COMMS/DISEQc	DISEQC_INT	24
TSMERGER	SBATM_INT	25
	Reserved	26
	TSMRGR_RAMOVF_INT (version 2)	27
CLOCKGEN	DCXO_INT	28
MAILBOX	ST40_DELTA_INT	29
	Reserved	30
AUDIO	CPXM_INT	31
	I2S2SPDIF_INT	32
FDMA	FDMA_GP0_INT	33
	FDMA_MBOX_INT	34
AUDIO	SPDIFPLYR_INT	35
	PCMRDR_INT	36
	PCMPLYR1_INT	37
	PCMPLYR0_INT	38
VIDEO	GLH_INT	39

Table 85: Delta ST231 Interrupts

Interrupt source		INT number
VTGs	VTG2_INT	40
	VTG1_INT	41
DISPLAY/LMU	LMU_IRQ	42
	Reserved	43
HDMI/HDCP	HDCP_INT	44
	HDMI_INT	45
DVP	DVP_INT	46
GAMMA	BLT_INT	47
PDES	Reserved	48
	PDES_INT	49
PTI	Reserved	50
	PTI_INT	51
CryptoCore	SIG_CHK_INT	52
	DMA_FIN_INT	53
	SEC_CP_INT	54
SATA	SATA_INT	55
USBH	EHCI_INT	56
	OHCI_INT	57
PADS	NMI_INT	58
	Reserved	59
External interrupts	EXT_INT[0]	60
	EXT_INT[1]	61
	EXT_INT[2]	62
	EXT_INT[3]	63

23.4 STx7100 interrupts

Table 86: ILC interrupt mapping

Interrupt source		ILC mapping
COMMs/PIO	PIO0_INT	0
	PIO1_INT	1
	PIO2_INT	2
	PIO3_INT	3
	PIO4_INT	4
	PIO5_INT	5
SSC	SSC0_INT	6
	SSC1_INT	7
	SSC2_INT	8
	Reserved	9
COMMs/UART	UART0_INT	10
	UART1_INT	11
	UART2_INT	12
	UART3_INT	13
COMMs/MAFE	MAFE_INT	14
COMMs/PWM	PWM_INT	15
COMMs/IRB	IRB_INT	16
COMMs/TTXT	TTXT_INT	17
COMMS/DAA	DAA_INT	18
COMMS/DISEQc	DISEQC_INT	19
INTERCO/Sbag	SBATM_INT	20
Clockgen	DCXO_INT	21
	Reserved	22
Mailbox	ST40_DELPHI_INT	23
	DELPHI_ST40_INT	24
	ST40_AUDIO_INT	25
	AUDIO_ST40_INT	26
Audio	CPXM_INT	27
	I2S2SPDIF_INT	28
FDMA	FDMA_GP0_INT	29
	FDMA_MBOX_INT	30
Audio	SPDIFPLYR_INT	31
	PCMRDR_INT	32
	PCMPLYR1_INT	33
	PCMPLYR0_INT	34
Video	GLH_INT	35
VTGs	VTG2_INT	36
	VTG1_INT	37
Display/LMU	LMU_INT	38
	Reserved	39

Table 86: ILC interrupt mapping

Interrupt source		ILC mapping
HDMI/HDCP	HDCP_INT	40
	HDMI_INT	41
DVP	DVP_INT	42
Gamma	BLT_INT	43
PDES	Reserved	44
	PDES_INT	45
PTI	Reserved	46
	PTI_INT	47
	Reserved	48
	DMA_FIN_INT	49
	SEC_CP_INT	50
SATA	SATA_INT	51
USBH	EHCI_INT	52
	OHCI_INT	53
-	Reserved	54
		55
		56
		57
		58
		59
	TSMGR_RAMOVF (cut 2)	60
-	Reserved	61
		62
		63
External Interrupts from external devices	EXT_INT[0]	0
	EXT_INT[1]	1
	EXT_INT[2]	2
	EXT_INT[3]	3
From pins via pulse stretcher	IRB_WAKEUP_INT	4
NMI pin	NMI_INT	5
-	Reserved	6
Low power alarm	LPA_INT (cut 2)	7

24 Interrupt registers

Register addresses are provided as either

IntControllerBaseAddress or *ILCBaseAddress or INTC2BaseAddress* + offset.

The *IntControllerBaseAddress* is:

0xFFD0 0000.

The *ILCBaseAddress* is:

0x1800 0000.

The *INTC2BaseAddress* is:

0x1900 1000.

Note: The areas not allocated are reserved and must not be accessed.

Table 87: Interrupt level controller system registers

Register name	Offset	Description	Type
Interrupt level controller (ILC)			
ILC_INPUT_INT	0x080, 0x084, 0x088	Input interrupt	RO
ILC_STA	0x200, 0x204, 0x208	Status	
ILC_CLR_STA	0x280, 0x284, 0x288	Clear status locations	WO
ILC_EN	0x400, 0x404, 0x408	Enable	R/W
ILC_CLR_EN	0x480, 0x484, 0x488	Clear enable locations	WO
ILC_SET_EN	0x500, 0x504, 0x508	Set enable locations	
ILC_WAKEUP_EN	0x608	Wake up enable	R/W
ILC_WAKEUP_ACTIVE_LEVEL	0x688	Wake up level	
ILC_PRIORITY*n* (*n*=0 to 63)	0x800 + 8*n*	Priority *n*	
ILC_EXT_PRIORITY*n* (*n*=0 to 7)	0xA00 + 8*n*	External priority *n*	
ILC_MODE*n* (*n*=0 to 7)	0x804 + 8*n*	Mode *n*	
INTC2 interrupt controller			
INTC2_PRI00	0x300	Interrupt priority levels	R/W
INTC2_PRI04	0x304		
INTC2_PRI08	0x308		
INTC2_REQ00	0x320	Interrupt requests	RO
INTC2_REQ04	0x324		
INTC2_REQ08	0x328		
INTC2_MSK00	0x340	Interrupt mask	Read, Set
INTC2_MSK04	0x344		
INTC2_MSK08	0x348		
INTC2_MSK_CLR00	0x360	Interrupt mask clear	WO
INTC2_MSK_CLR04	0x364		
INTC2_MSK_CLR08	0x368		
INTC2_MODE	0x380	INTC2 mode	R/W

24.1 Interrupt level controler (ILC)

ILC_INPUT_INT Input interrupt

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08C	Reserved																															
0x088	Reserved																								EXT7	RSVD	EXT5	EXT4	EXT3	EXT2	EXT1	EXT0
0x084	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT359	INT38	INT37	INT36	INT35	INT34	INT33	INT32
0x080	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: *ILCBaseAddress* + 0x080, 0x084, 0x088 and 0x08C

Type: Read

Reset: 0

Description: The synchronized version of all the interrupt numbers can be read from this register. There are two input interrupt registers from 0x080 to 0x084. The contents of this register are set to logic 0 on reset.

ILC_STA Status

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20C	Reserved																															
0x208	Reserved																								EXT7	RSVD	EXT5	EXT4	EXT3	EXT2	EXT1	EXT0
0x204	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT359	INT38	INT37	INT36	INT35	INT34	INT33	INT32
0x200	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: *ILCBaseAddress* + 0x200, 0x204, 0x208 and 0x20C

Type: Read

Reset: 0

Description: The status of the interrupt numbers is inferred by reading this register. The bit in the status register is at logic 1 on the following conditions.

- If the corresponding interrupt number is internal (synchronous), then the interrupt number should be at logic 1.
- If the corresponding interrupt number is external (asynchronous), then the interrupt number should match the programmed trigger condition.

There are two status registers from 0x200 to 0x27C. The contents of this register are at logic 0 on reset.

ILC_CLR_STA Clear status locations

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28C	Reserved																															
0x288	Reserved																								EXT7	RSVD	EXT5	EXT4	EXT3	EXT2	EXT1	EXT0
0x284	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT359	INT38	INT37	INT36	INT35	INT34	INT33	INT32
0x280	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: *ILCBaseAddress* + 0x280, 0x284, 0x288 and 0x28C

Type: Write only

Reset: Undefined

Description: This location is used to clear bits in the status register. The locations that correspond to external interrupts are valid. The status is cleared only if the interrupt trigger mode is edge sensitive that is the rising edge, falling edge or any edge. To clear the status bit, a clear bit operation is to be performed on this location. Clear bit operation is performing a write operation with logic 1 on data bus corresponding to the locations which have to be cleared.

ILC_EN Enable

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40C	Reserved																															
0x408	Reserved																								EXT7	RSVD	EXT5	EXT4	EXT3	EXT2	EXT1	EXT0
0x404	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT359	INT38	INT37	INT36	INT35	INT34	INT33	INT32
0x400	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: *ILCBaseAddress* + 0x400, 0x404, 0x408 and 0x40C

Type: Read/write

Reset: 0

Description: Interrupt generation from an interrupt number is enabled only if the corresponding bit in the enable register is set to logic 1. The contents of this register are at logic 0 on reset.

ILC_CLR_EN Clear enable locations

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x48C	Reserved																															
0x488	Reserved																								EXT7	RSVD	EXT5	EXT4	EXT3	EXT2	EXT1	EXT0
0x484	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT359	INT38	INT37	INT36	INT35	INT34	INT33	INT32
0x480	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: *ILCBaseAddress* + 0x480, 0x484, 0x488 and 0x48C

Type: Write only

Reset: Undefined

Description: Any bit in the enable register can be cleared by performing a clear bit operation on the appropriate location.

ILC_SET_EN Set enable locations

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50C	Reserved																															
0x508	Reserved																								EXT7	RSVD	EXT5	EXT4	EXT3	EXT2	EXT1	EXT0
0x504	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT359	INT38	INT37	INT36	INT35	INT34	INT33	INT32
0x500	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: *ILCBaseAddress* + 0x500, 0x504, 0x508 and 0x50C

Type: Write only

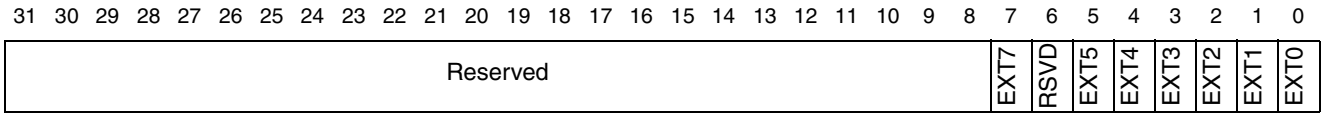
Reset: Undefined

Description: Any bit in the enable register can be set by performing a set bit operation on the appropriate location (set bit operation is performing a write operation with logic 1 on the data bus corresponding to the location which has to be set).

There are two locations at 0x500 and 0x504 corresponding to respective enable registers.

Note: Interrupts EXT4 to EXT7 are not valid outputs.

ILC_WAKEUP_EN Wake up enable



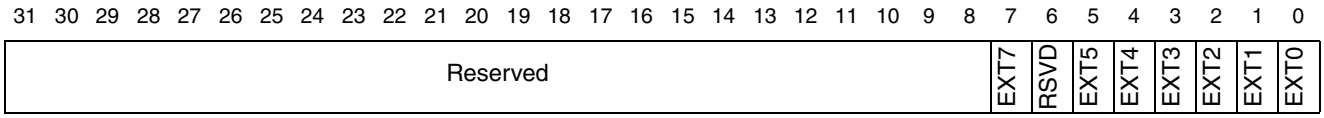
Address: *ILCBaseAddress* + 0x608

Type: Read/write

Reset: 0

Description: Enables the external interrupt (asynchronous) for wake up by interrupt generation. Only the locations corresponding to external interrupts are valid. The external interrupt enables the wake up by interrupt generation only if the corresponding bit in this register is set to logic 1. The contents of this register are set to logic 0 on reset.

ILC_WAKEUP_ACTIVE_LEVEL Wake up level



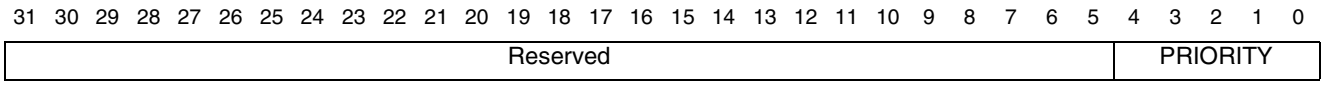
Address: *ILCBaseAddress* + 0x688

Type: Read/write

Reset: 1

Description: Used to program the polarity of the external interrupt line on which a wake up by interrupt is to be generated. If a bit in this register is set to 1 then a wake up by interrupt is generated, only if the corresponding external interrupt is at logic 1. Writing logic 0 generates the wake up by interrupt when corresponding external interrupt pin is at logic 0. The contents of this register are set to logic 1 on reset.

ILC_PRIORITYn Priority



Address: *ILCBaseAddress* + 0x800 + (*n* x 0x008)

Type: Read/write

Reset: 0

Description: The PRIORITY*n* register does the mapping of the ILC internal interrupt input *n* to one of the output interrupts ILC_EXT_OUT[7-0] connected to the ST40 or SYSITRQ[3-0]. The assignment is done by writing appropriate word into the priority register.

In the STx7100, the valid values are between 0x8000 and 0X8007.

Values between 0x8000 and 0x8003 correspond to ILC_EXT_OUT[3-0], connected to pins SYSITRQ[3-0].

Values between 0x8004 and 0x8007 correspond to ILC_EXT_OUT[7-4], connected to ST40 interrupts IRL[3-0].

Example:

To map internal interrupt 10 onto SYSITRQ0, 0x8000 must be written into ILC_PRIORITY10.

ILC_EXT_PRIORITYn Priority

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PRIORITY			

Address: *ILCBaseAddress* + 0xA00 + (*n* x 0x008)

Type: Read/write

Reset: 0

Description: The EXT_PRIORITY*n* register maps the ILC external interrupt input EXT_INT*n* to one of the ILC output interrupts ILC_EXT_OUT[7-0] connected to the ST40 or SYSITRQ[3-0]. The assignment is done by writing appropriate word into the priority register.

In the STx7100, the valid values are between 0x8000 and 0X8007.

Values between 0x8000 and 0x8003 correspond to ILC_EXT_OUT[3-0], connected to pins SYSITRQ[3-0].

Values between 0x8004 and 0x8007 correspond to ILC_EXT_OUT[7-4], connected to ST40 interrupts IRL[3-0].

Example:

To map ILC_EXT_INT[4] onto IRL0, 0x8004 must be written into ILC_EXT_PRIORITY4.

ILC_MODEn Mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													MODE2	MODE1	MODE0

Address: *ILCBaseAddress* + 0x800 + (n x 0x008) + 0x004

Type: Read/write

Reset: 0

Description: The mode register is used to program the trigger mode for a given interrupt number. This is a 3-bit register. The mode register is present only for external interrupts. No mode register is present for internal interrupts. All the internal interrupt numbers are assumed active high.

- The trigger modes can be programmed to appropriate trigger levels by writing the values as shown in the table below. Use the syntax 0x*nnnn nnnn* for the address.
- Include the values outside the address range.

[31:3] **Reserved**

[2:0] **MODE**

0x00: No trigger mode
0x01: High level trigger mode
0x02: Low level trigger mode
0x03: Rising edge trigger mode
0x04: Falling edge trigger mode
0x05: Any edge trigger mode
0x06: No trigger mode
0x07: No trigger mod

The mode register is set to 0x00 on reset. The address of mode register corresponding to interrupt number *n* is at 0x800 + (*n* x 0x008) + 0x004.

24.2 INTC2 interrupt controller

INTC2_PRI0n Interrupt priority level 0/4/8

	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
00	Reserved		PIO2	PIO1	PIO0	DMA_INT0 DMA_INT1 DMA_INT2 DMA_INT3 DMA_INT4 DMA_ERR4	PCI_ERR PCI_AD PCI_PWR_DWN	PCI_SERR
04	INT_GRP8	INT_GRP7	INT_GRP6	INT_GRP5	INT_GRP4	INT_GRP3	INT_GRP2	INT_GRP1
08	INT_GRP16	INT_GRP15	INT_GRP14	INT_GRP13	INT_GRP12	INT_GRP11	INT_GRP10	INT_GRP9

Address: *INTC2BaseAddress* + 0x300, 0x304, 0x308

Type: R/W

Reset:

Description: Priority levels for interrupts and interrupt groups (INT_GRP*m*), 0 - 15. Within each group, each individual interrupt has its own priority. (By setting a group's priority to 0x0, the whole group can be masked.)

INTC2_REQ0n Interrupt requests 0/4/8

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Reserved	PIO2	PIO1	PIO0	DMA_ERR4	Reserved	DMA_INT4	DMA_INT3	DMA_INT2	DMA_INT1	DMA_INT0	Reserved	PCI_PWR_DWN	PCI_AD	PCI_ERR	PCI_SERR
04	INT_GRP8	INT_GRP7	INT_GRP6	INT_GRP5	INT_GRP4		INT_GRP3		INT_GRP2				INT_GRP1			
08	INT_GRP16	INT_GRP15	INT_GRP14	INT_GRP13	INT_GRP12		INT_GRP11		INT_GRP10				INT_GRP9			

Address: *INTC2BaseAddress* + 0x320, 0x324, 0x328

Type: RO

Description: Status of interrupt and interrupt group (INT_GRP*m*) signals, irrespective of the state of register INTC2_MSK0*n*

INTC2_MSK0n Interrupt mask 0/4/8

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Reserved																	PIO2	PIO1	PIO0	DMA_ERR4	Reserved	DMA_INT4	DMA_INT3	DMA_INT2	DMA_INT1	DMA_INT0	Reserved	PCI_PWR_DWN	PCI_AD	PCI_ERR	PCI_SERR
04	INT_GRP8				INT_GRP7				INT_GRP6				INT_GRP5				INT_GRP4				INT_GRP3				INT_GRP2				INT_GRP1			
08	INT_GRP16				INT_GRP15				INT_GRP14				INT_GRP13				INT_GRP12				INT_GRP11				INT_GRP10				INT_GRP9			

Address: *INTC2BaseAddress* + 0x340, 0x344, 0x348
Type: Read, Set
Reset:
Description: Masks for interrupt and interrupt groups (INT_GRP*m*).

INTC2_MSK_CLR0n Interrupt mask clear 0/4/8

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Reserved																	PIO2	PIO1	PIO0	DMA_ERR4	Reserved	DMA_INT4	DMA_INT3	DMA_INT2	DMA_INT1	DMA_INT0	Reserved	PCI_PWR_DWN	PCI_AD	PCI_ERR	PCI_SERR
04	INT_GRP8				INT_GRP7				INT_GRP6				INT_GRP5				INT_GRP4				INT_GRP3				INT_GRP2				INT_GRP1			
08	INT_GRP16				INT_GRP15				INT_GRP14				INT_GRP13				INT_GRP12				INT_GRP11				INT_GRP10				INT_GRP9			

Address: *INTC2BaseAddress* + 0x360, 0x364, 0x368
Type: WO
Reset: -
Description: Clear masks for interrupt and interrupt groups (INT_GRP*m*).

INTC2_MODE INTC2 mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_GRP16	INT_GRP15	INT_GRP14	INT_GRP13	INT_GRP12	INT_GRP11	INT_GRP10	INT_GRP09	INT_GRP08	INT_GRP07	INT_GRP06	INT_GRP05	INT_GRP04	INT_GRP03	INT_GRP02	INT_GRP01	Reserved	PIO2	PIO1	PIO0	DMA_ERR4	Reserved	DMA_INT4	DMA_INT3	DMA_INT2	DMA_INT1	DMA_INT0	Reserved	PCI_PWR_DWN	PCI_AD	PCI_ERR	PCI_SERR

Address: *INTC2BaseAddress* + 0x380
Type: R/W
Reset:
Description: Modifies INTEVT codes and default interrupt priorities. If the corresponding INTC2_MODE bit is set, then the interrupts have values between 0x200 and 0x3E0 which map onto the INTEVT Codes allocated to the IRLs in the INTC (see Chapter 23: *Interrupt maps* *on page 200*).

25 Programmable I/O (PIO) ports

There are 48 bits of programmable I/O configured in six ports. Each bit is programmable as output or input. The output can be configured as a totem-pole or open-drain driver. The input compare logic can generate an interrupt on any change of any input bit. Many programmable I/O have alternate functions and can be connected to an internal peripheral signal such as a UART or SSC.

The PIO ports can be controlled by registers, mapped into the device address space. The registers for each port are grouped in a 4 Kbyte block, with the base of the block for port *n* at the address *PIO*n*BaseAddress*. At reset, all of the registers are reset to zero and all PIO pads put in input mode with internal pull-up.

Each 8-bit PIO port has a set of eight-bit registers. Each of the eight bits of each register refers to the corresponding pin in the corresponding port. These registers hold:

- the output data for the port (PIO_PnOUT),
- the input data read from the pin (PIO_PnIN),
- PIO bit configuration registers (PIO_PnC[2:0]),
- the two input compare function registers (PIO_PnCOMP and PIO_PnMASK).

Each of the registers, except PIO_PnIN, is mapped on to two additional addresses so that bits can be set or cleared individually.

- The PIO_SET_x registers set bits individually. Writing 1 in these registers sets a corresponding bit in the associated register x; 0 leaves the bit unchanged.
- The PIO_CLR_x registers clear bits individually. Writing 1 in these registers resets a corresponding bit in the associated register x; 0 leaves the bit unchanged.

26 Programmable I/O ports registers

Each 8-bit PIO port has a set of eight-bit registers. Each of the eight bits of each register refers to the corresponding pin in the corresponding port.

Register addresses are provided as *PIOnBaseAddress* + offset.

The *PIO0BaseAddresses* are:

PIO0: 0x1802 0000,
PIO1: 0x1802 1000,
PIO2: 0x1802 2000,
PIO3: 0x1802 3000,
PIO4: 0x1802 4000,
PIO5: 0x1802 5000.

Table 88: Programmable I/O ports register summary

Register	Description	Offset	Type
PIO_CLR_PnCm	Clear bits of PnCm	0x28, 38, 48	WO
PIO_CLR_PnCOMP	Clear bits of PnCOMP	0x58	WO
PIO_CLR_PnMASK	Clear bits of PnMASK	0x68	WO
PIO_CLR_PnOUT	Clear bits of PnOUT	0x08	WO
PIO_PnCm	PIO configuration	0x20, 30, 40	R/W
PIO_PnCOMP	PIO input comparison	0x50	R/W
PIO_PnIN	PIO input	0x10	RO
PIO_PnMASK	PIO input comparison mask	0x60	R/W
PIO_PnOUT	PIO output	0x00	R/W
PIO_SET_PnC[2:0]	Set bits of PnC[2:0]	0x24, 34, 44	RO
PIO_SET_PnCOMP	Set bits of PnCOMP	0x54	WO
PIO_SET_PnMASK	Set bits of PnMASK	0x64	WO
PIO_SET_PnOUT	Set bits of PnOUT	0x04	WO

There are is an additional comms register described in Chapter 21: *System configuration registers on page 171*.

PIO_CLR_PnCm Clear bits of PnCm

	7	6	5	4	3	2	1	0
0x28	CLR_PC0[7:0]							
0x38	CLR_PC1[7:0]							
0x48	CLR_PC2[7:0]							

Address: *PIOnBaseAddress* + 0x28 (PIO_CLR_PnC0), 0x38 (PIO_CLR_PnC1), 0x48 (PIO_CLR_PnC2)

Type: WO

Description: PIO_CLR_PnCm allows the bits of registers PIO_PnCm to be cleared individually. Writing 1 in one of these register clears the corresponding bit in the corresponding PIO_PnCm register, while 0 leaves the bit unchanged.

PIO_CLR_PnCOMP Clear bits of PnCOMP

7	6	5	4	3	2	1	0
CLR_PCOMP[7:0]							

Address: *PIOnBaseAddress* + 0x58

Type: WO

Description: PIO_CLR_PnCOMP allows bits of PIO_PnCOMP to be cleared individually. Writing 1 in this register clears the corresponding bit in the PIO_PnCOMP register, while 0 leaves the bit unchanged.

PIO_CLR_PnMASK Clear bits of PnMASK

7	6	5	4	3	2	1	0
CLR_PMASK[7:0]							

Address: *PIOnBaseAddress* + 0x68

Type: WO

Description: PIO_CLR_PnMASK allows bits of PIO_PnMASK to be cleared individually. Writing 1 in this register clears the corresponding bit in the PIO_PnMASK register, while 0 leaves the bit unchanged.

PIO_CLR_PnOUT Clear bits of PnOUT

7	6	5	4	3	2	1	0
CLR_POUT[7:0]							

Address: *PIOnBaseAddress* + 0x08

Type: WO

Description: PIO_CLR_PnOUT allows bits of PIO_PnOUT to be cleared individually. Writing 1 in this register clears the corresponding bit in the PIO_PnOUT register, while 0 leaves the bit unchanged.

PIO_PnCm PIO configuration

	7	6	5	4	3	2	1	0
0x20	CONFIGDATA0[7:0]							
0x30	CONFIGDATA1[7:0]							
0x40	CONFIGDATA2[7:0]							

Address: *PIOnBaseAddress* + 0x20 (PIO_PnC0), 0x30 (PIO_PnC1), 0x40 (PIO_PnC2)

Type: R/W

Reset: 0

Description: There are three configuration registers (PIO_PnC0, PIO_PnC1 and PIO_PnC2) for each port. These are used to configure the PIO port pins. Each pin can be configured as an input, output, bidirectional, or alternative function pin (if any), with options for the output driver configuration.

Three bits, one bit from each of the three registers, configure the corresponding bit of the port. The configuration of the corresponding I/O pin for each valid bit setting is given in Table 89.

Table 89: PIO bit configuration encoding

PnC2[y]	PnC1[y]	PnC0[y]	Bit y configuration	Bit y output
0	0	0	Input	Weak pull up (default)
0	0 or 1	1	Bidirectional	Open drain
0	1	0	Output	Push-pull
1	0	0 or 1	Input	High impedance
1	1	0	Alternative function output	Push-pull
1	1	1	Alternative function bidirectional	Open drain

The PIO_PnC[2:0] registers are each mapped on to two additional addresses, PIO_SET_PnC and PIO_CLR_PnC, so that bits can be set or cleared individually.

PIO_PnCOMP PIO input comparison

7	6	5	4	3	2	1	0
PCOMP[7:0]							

Address: *PIOnBaseAddress* + 0x50

Type: R/W

Reset: 0

Description: The input compare register PIO_PnCOMP can be used to cause an interrupt if the input value differs from a fixed value.

The input data from the PIO ports pins are compared with the value held in PIO_PnCOMP. If any of the input bits is different from the corresponding bit in the PIO_PnCOMP register and the corresponding bit position in PIO_PnMASK is set to 1, then the internal interrupt signal for the port is set to 1.

The compare function is sensitive to changes in levels on the pins. For the comparison to be seen as a valid interrupt by an interrupt handler, the change in state on the input pin must be longer in duration than the interrupt response time.

The compare function is operational in all configurations for each PIO bit, including the alternative function modes.

The PIO_PnCOMP register is mapped on to two additional addresses, PIO_SET_PnCOMP and PIO_CLR_PnCOMP, so that bits can be set or cleared individually.

PIO_PnIN PIO input

7	6	5	4	3	2	1	0
PIN[7:0]							

Address: *PIOnBaseAddress* + 0x10

Type: RO

Reset: 0

Description: The data read from this register gives the logic level present on the input pins of the port at the start of the read cycle to this register. Each bit reflects the input value of the corresponding bit of the port. The read data is the last value written to the register regardless of the pin configuration selected.

PIO_PnMASK PIO input comparison mask

7	6	5	4	3	2	1	0
PMASK[7:0]							

Address: *PIOnBaseAddress* + 0x60

Type: R/W

Reset: 0

Description: When a bit is set to 1, the compare function for the internal interrupt for the port is enabled for that bit. If the respective bit ([7:0]) of the input is different from the corresponding bit in the PIO_PnCOMP register, then an interrupt is generated.

The PIO_PnMASK register is mapped on to two additional addresses, PIO_SET_PnMASK and PIO_CLR_PnMASK, so that bits can be set or cleared individually.

PIO_PnOUT PIO output

7	6	5	4	3	2	1	0
POUT[7:0]							

Address: *PIOnBaseAddress* + 0x00

Type: R/W

Reset: 0

Description: Holds output data for the port. Each bit defines the output value of the corresponding bit of the port.

The PIO_PnOUT register is mapped on to two additional addresses, PIO_SET_PnOUT and PIO_CLR_PnOUT, so that bits can be set or cleared individually.

PIO_SET_PnC[2:0] Set bits of PnC[2:0]

	7	6	5	4	3	2	1	0
0x24	SET_PC0[7:0]							
0x34	SET_PC1[7:0]							
0x44	SET_PC2[7:0]							

Address: *PIOnBaseAddress* + 0x24 (PIO_SET_PnC0), 0x34 (PIO_SET_PnC1), 0x44 (PIO_SET_PnC2)

Type: WO

Description: PIO_SET_PnC[2:0] allow the bits of registers PIO_PnC[2:0] to be set individually. Writing 1 in one of these registers sets the corresponding bit in the corresponding PIO_PnC[2:0] register, while 0 leaves the bit unchanged.

PIO_SET_PnCOMP Set bits of PnCOMP

7	6	5	4	3	2	1	0
SET_PCOMP[7:0]							

Address: *PIOnBaseAddress* + 0x54

Type: WO

Description: PIO_SET_PnCOMP allows bits of PIO_PnCOMP to be set individually. Writing 1 in this register sets the corresponding bit in the PIO_PnCOMP register, while 0 leaves the bit unchanged.

PIO_SET_PnMASK Set bits of PnMASK

7	6	5	4	3	2	1	0
SET_PMASK[7:0]							

Address: *PIOnBaseAddress* + 0x64

Type: WO

Description: PIO_SET_PnMASK allows bits of PIO_PnMASK to be set individually. Writing 1 in this register sets the corresponding bit in the PIO_PnMASK register, while 0 leaves the bit unchanged.

PIO_SET_PnOUT Set bits of PnOUT

7	6	5	4	3	2	1	0
SET_POUT[7:0]							

Address: *PIOnBaseAddress* + 0x04

Type: WO

Description: PIO_SET_PnOUT allows bits of PIO_PnOUT to be set individually. Writing 1 in this register sets the corresponding bit in the PIO_PnOUT register, while 0 leaves the bit unchanged.

27 Mailboxes

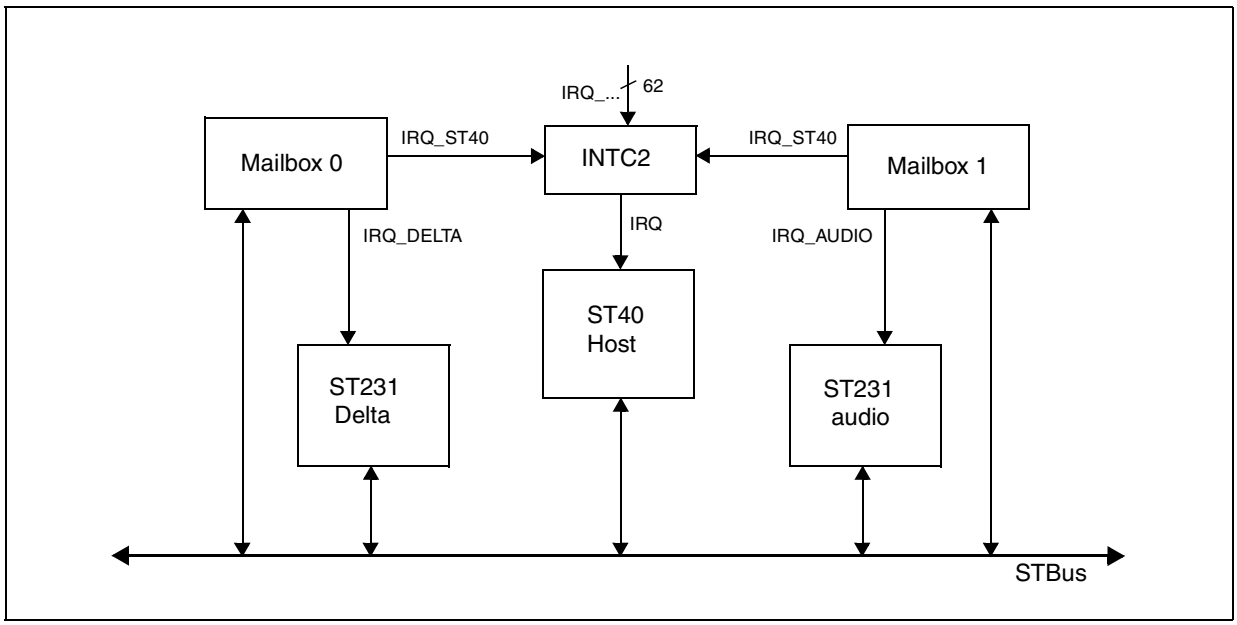
The main data transfer between the STx7100 memory and processors is through the STBus. However, software controls the processing of these data structures, fills and empties buffers via various DMAs. These transfers and processing require signaling after completion to indicate events or that a structure is available.

There are two mailboxes, which coordinate communication between the ST40 and two ST231 processors using the STBus. They allow interrupts to be raised by any processor in software to provide the necessary signalling.

- Mailbox 0 links the ST40 and the Delta (H.264) ST231.
- Mailbox 1 links the ST40 and the audio ST231.

The CPUs can read and write to the mailbox configuration registers with standard memory accesses.

Figure 61: Mailboxes block diagram



Each processor runs on its own clock, and may be running more than one software process. Information is exchanged by writing into a common area (accessible to both linked processors) and generating a hardware interrupt. For example: if the ST40 writes information into the mailbox 0 common area, an interrupt is generated for the Delta ST231, and vice-versa.

On receiving an interrupt, the interrupt service routine (ISR) infers the destination process for which the information is to be routed by reading the mailbox interrupt status registers. The ISR then routes the information to the destination process. The interrupt is then cleared by executing a clear-bit operation.

The interrupt sources can be enabled or disabled by setting or resetting the contents of the corresponding interrupt enable register.

27.1 Mailbox register operation

27.1.1 Interrupt status registers

An interrupt can be generated by setting any bit in this 32-bit register. Four interrupt generation registers are present to generate interrupts from 128 sources.

The interrupt generation registers MBXn_ST231_INT_STAm generate interrupts for the ST231. Registers MBXn_ST40_INT_STAm generate the interrupts for the ST40.

Additional write-only registers MBXn_(ST40/ST231)_INT_STAn_SET and MBXn_(ST40/ST231)_INT_STAn_CLR set and clear individual bits in the interrupt status registers.

27.1.2 Interrupt enable registers

The interrupt enable registers MBXn_(ST40/ST231)_INT_ENn enable or disable the interrupts generated by the corresponding bits in the interrupt status registers. There are four interrupt enable registers corresponding to four interrupt status registers.

The write-only registers MBXn_(ST40/ST231)_INT_ENn_SET and MBXn_(ST40/ST231)_INT_ENn_CLR can be used to set and clear individual bits in the interrupt enable registers.

Set-bit operation

Any bit in the status registers can be set by performing the set-bit operation at location MBXn_(ST40/ST231)_INT_STAn_SET. This operation is as follows.

To set one or more bits of the register, a value with the corresponding bits set to 1 is written. All other bits should be set to 0.

For example: to set bits 0 and 4, the register is written with the value 0x0000 0011. The bit locations corresponding to bit 0 and bit 4 of the register are set, leaving the other bit locations unaltered.

If any of the other bits are already set to 1, the same state is maintained. For example, if bit 6 is already set to 1, the bit location 6 continues to hold logic 1 even after the above write operation.

Clear-bit operation

Any bit in the registers can be cleared by performing the clear-bit operation at location MBXn_(ST40/ST231)_INT_STAn_CLR. This operation is as follows.

To clear one or more bits of the register, a value with the corresponding bits set to 1 is written. All other bits should be set to 0.

For example: to clear bits 0 and 4, the value 0x0000 0011 is written to the register. This clears only bits 0 and 4. The status of other bits is not affected.

27.1.3 Interrupt generation for the ST40

The interrupts for the ST40 are generated by the ST231 processor. The interrupts are generated by setting the bits in register MBXn_ST40_INT_STAm by performing the set-bit operation on MBXn_ST40_INT_STAm_SET. The interrupts are cleared by a clear-bit operation on MBXn_ST40_INT_STAm_CLR.

The interrupt generated by any bit in MBXn_ST40_INT_STAm is enabled by setting the corresponding bit in MBXn_ST40_INT_ENm (using the set-bit operation on MBXn_ST40_INT_ENm_SET). The interrupt generation is disabled by clearing the corresponding bit in the interrupt enable register (using the clear-bit operation on MBXn_ST40_INT_ENm_CLR).

27.1.4 Interrupt generation for the ST231

The interrupts for ST231 are generated by the ST40 processor. The interrupts are generated by setting the bits in register MBXn_ST231_INT_STAm by performing the set-bit operation on MBXn_ST231_INT_STAm_SET. The interrupts are cleared by a clear-bit operation on MBXn_ST231_INT_STAm_CLR.

The interrupt generated by any bit in MBXn_ST231_INT_STAm is enabled by setting the corresponding bit in MBXn_ST231_INT_ENm (using the set-bit operation on MBXn_ST231_INT_ENm_SET). The interrupt generation is disabled by clearing the corresponding bit in the interrupt enable register (using the clear-bit operation on MBXn_ST231_INT_ENm_CLR).

28 Mailbox registers

Register addresses are provided as *MBXxBaseAddressx* + offset.

MBX0BaseAddress (ST40 - ST231 Delta) is:

0x1921 1000.

MBX1BaseAddress (ST40 - ST231 audio) is:

0x1921 2000.

Table 90: Mailbox register summary

Register	Description	Offset	Type
MBXn_ID_VER	Mailbox version	0x000	RO
MBXn_ST231_INT_STAm	Mailbox ST231 status 1 to 4	0x004 - 0x010	R/W
MBXn_ST231_INT_STAm_SET	Set mailbox ST231 interrupt 1 to 4	0x024 - 0x030	WO
MBXn_ST231_INT_STAm_CLR	Clear mailbox ST231 interrupt 1 to 4	0x044 - 0x050	WO
MBXn_ST231_INT_ENm	Enable mailbox ST231 interrupt 1 to 4	0x064 - 0x070	R/W
MBXn_ST231_INT_ENm_SET	Set mailbox ST231 interrupt enable 1 to 4	0x084 - 0x090	WO
MBXn_ST231_INT_ENm_CLR	Clear mailbox ST231 interrupt enable 1 to 4	0x0A4 - 0x0B0	WO
MBXn_ST40_INT_STAm	Mailbox ST40 status 1 to 4	0x104 - 0x110	R/W
MBXn_ST40_INT_STAm_SET	Set mailbox ST40 status 1 to 4	0x124 - 0x130	WO
MBXn_ST40_INT_STAm_CLR	Clear mailbox ST40 status 1 to 4	0x144 - 0x150	WO
MBXn_ST40_INT_ENm	Enable mailbox ST40 interrupt 1 to 4	0x164 - 0x170	R/W
MBXn_ST40_INT_ENm_SET	Set mailbox ST40 interrupt enable 1 to 4	0x184 - 0x190	WO
MBXn_ST40_INT_ENm_CLR	Clear mailbox ST40 interrupt enable 1 to 4	0x1A4 - 0x1B0	WO
MBXn_LCKm	Mailbox lock 00 to 31	0x200 - 0x27C	WTS

MBXn_ID_VER Mailbox version

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_VER																															

Address: *MBXxBaseAddress* + 0x0000

Type: RO

Reset: Undefined

Description: Version number and identity register

MBXn_ST231_INT_STAm Mailbox ST231 status 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_STA																															

Address: *MBXxBaseAddress* + 0x004 (REG1), 0x008, 0x00C, 0x010 (REG4)
Type: R/W
Reset: 0x00
Description: Generates interrupts by bit access for the ST231 processor.

MBXn_ST231_INT_STAm_SET Set mailbox ST231 interrupt 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_STA_SET																															

Address: *MBXxBaseAddress* + 0x024 (REG1), 0x028, 0x02C, 0x030 (REG4)
Type: WO
Reset: 0x00
Description: Used to set bits in MBXn_ST231_INT_STAm.

MBXn_ST231_INT_STAm_CLR Clear mailbox ST231 interrupt 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_STA_CLR																															

Address: *MBXxBaseAddress* + 0x044 (REG1), 0x048, 0x04C, 0x050 (REG4)
Type: WO
Reset: 0x00
Description: Used to clear bits in MBXn_ST231_INT_STAm.

MBXn_ST231_INT_ENm Enable mailbox ST231 interrupt 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN																															

Address: *MBXxBaseAddress* + 0x064 (REG1), 0x068, 0x06C, 0x070 (REG4)
Type: R/W
Reset: 0x00
Description: Enables interrupts in MBXn_ST231_INT_STAm.

MBXn_ST231_INT_ENm_SET Set mailbox ST231 interrupt enable 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN_SET																															

Address: *MBXxBaseAddress* + 0x084 (REG1), 0x088, 0x08C, 0x090 (REG4)
Type: WO
Reset: 0x00
Description: Sets enables in MBXn_ST231_INT_ENm.

MBXn_ST231_INT_ENm_CLR Clear mailbox ST231 interrupt enable 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CLR																															

Address: *MBXxBaseAddress* + 0x0A4 (REG1), 0x0A8, 0x0AC, 0x0B0 (REG4)
Type: WO
Reset: 0x00
Description: Clears enables in MBXn_ST231_INT_ENm.

MBXn_ST40_INT_STAm Mailbox ST40 status 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_STA																															

Address: *MBXxBaseAddress* + 0x104 (REG1), 0x108, 0x10C, 0x110 (REG4)
Type: R/W
Reset: 0x00
Description: Generates interrupts by bit access for the ST40 processor.

MBXn_ST40_INT_STAm_SET Set mailbox ST40 status 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_STA_SET																															

Address: *MBXxBaseAddress* + 0x124 (REG1), 0x128, 0x12C, 0x130 (REG4)
Type: WO
Reset: 0x00
Description: Used to set bits in MBXn_ST40_INT_STAm.

MBXn_ST40_INT_STAm_CLR Clear mailbox ST40 status 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_STA_CLR																															

Address: *MBXxBaseAddress* + 0x144 (REG1), 0x148, 0x14C, 0x150 (REG4)
Type: WO
Reset: 0x00
Description: Used to clear bits in MBXn_ST40_INT_STAm.

MBXn_ST40_INT_ENm Enable mailbox ST40 interrupt 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN																															

Address: *MBXxBaseAddress* + 0x164 (REG1), 0x168, 0x16C, 0x170 (REG4)
Type: R/W
Reset: 0x00
Description: Enables interrupts in MBXn_ST40_INT_STAm.

MBXn_ST40_INT_ENm_SET Set mailbox ST40 interrupt enable 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN_SET																															

Address: *MBXxBaseAddress* + 0x184 (REG1), 0x188, 0x18C, 0x190 (REG4)
Type: WO
Reset: 0x00
Description: Sets enables in MBXn_ST40_INT_ENm.

MBXn_ST40_INT_ENm_CLR Clear mailbox ST40 interrupt enable 1 to 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN_CLR																															

Address: *MBXxBaseAddress* + 0x1A4 (REG1), 0x1A8, 0x1AC, 0x1B0 (REG4)
Type: WO
Reset: 0x00
Description: Clears enables in MBXn_ST40_INT_ENm.

MBXn_LCKm Mailbox lock 00 to 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBX_LCK																															

Address: *MBXnBaseAddress* + 0x200 + 4*m* (*m* = 0 to 31)
Type: Write, test and set
Buffer: 0x00
Description: Each mailbox has 32 registers to support locking of indivisible operations. Each register has one usable bit (bit 0) which can be toggled to 1 by reading it, or set to 0 by writing into it. If this register is read when the bit 0 is 0, 0 is returned and the value is then updated to 1. If the register is read when the value of bit 0 is 1, 1 is retuned. The write operation just updates the bit 0 to 0. This implementation allows software to efficiently manage atomic operations in shared memory.

Part 4

Memory

29 Local memory interface (LMI)

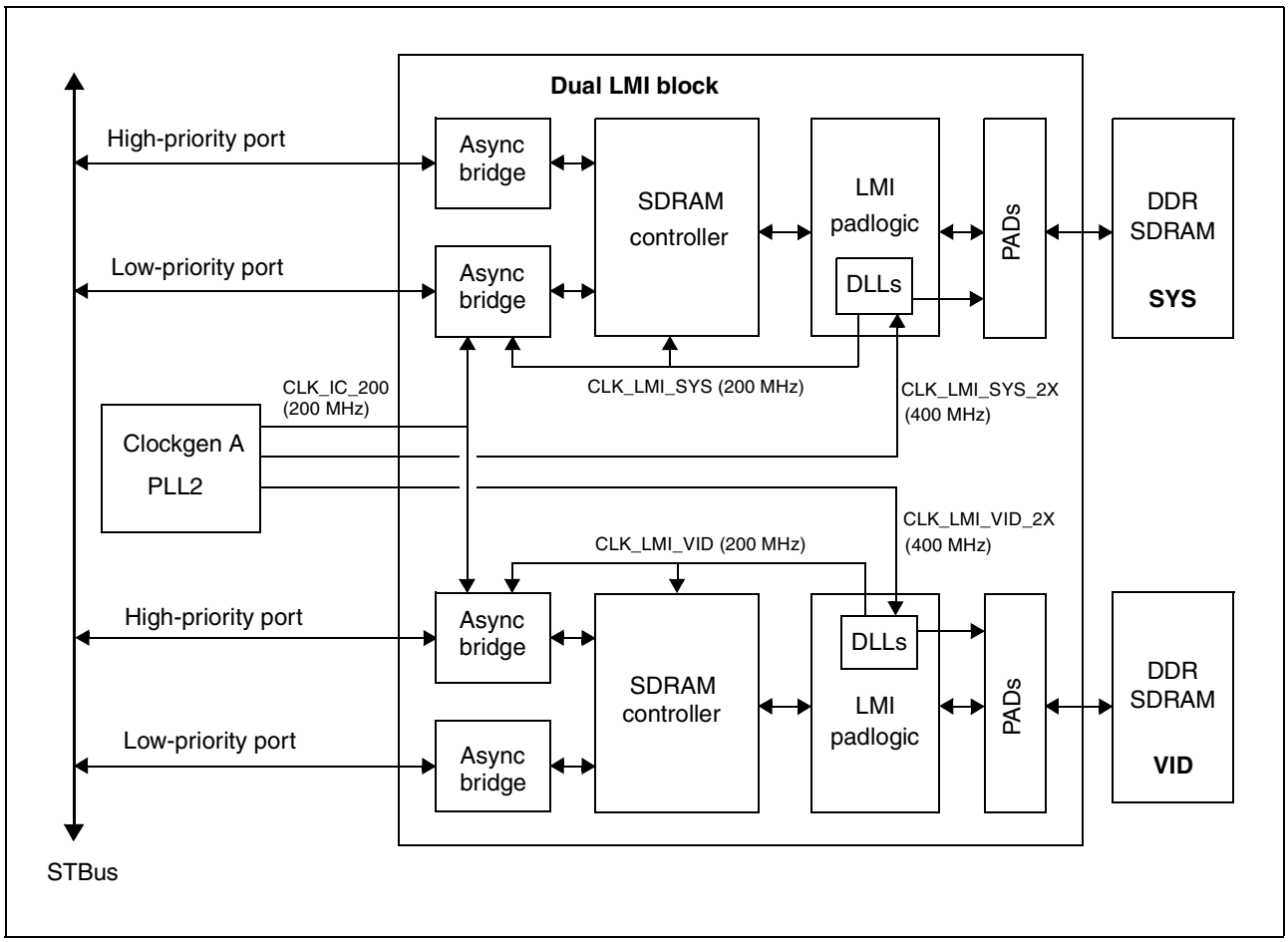
29.1 Introduction

The local memory interface (LMI) consists of dual blocks providing an interface for both system (SYS) and video (VID) main-memory subsystems.

Chapter 5: *Memory map* *on page 38* contains tables showing LMI arrangement within the STx7100 as a whole. The main LMI registers are described in Chapter 30 on page 242; however, there are several additional LMI registers described in Chapter 21: *System configuration registers* *on page 171*.

The LMI's two blocks are shown in Figure 62. They are clocked by the 400 MHz CLK_LMI_SYS/VID_2X signals, which are divided by internal DLLs to provide 200 MHz clocks (CLK_LMI_SYS/VID) for the SDRAM controllers.

Figure 62: LMI subsystem block diagram



The rest of this chapter describes a single LMI system.

PCB Layout recommendations for DDR-SDRAM signals are given in Chapter 14: *PCB layout recommendations* *on page 108*.

Main-memory organization

- The array is organized as rows.
- Each row consists of one or more discrete devices or DIMM (single or double sided) modules arranged in sockets on a PCB.

SDRAM controller features

- programmable external bus width: 16- and 32-bit,
- dual or quad bank double data rate (DDR) SDRAM,
- main memory size: up to 192 Mbytes,
- memory modules supported: 2 rows of discrete SDRAM, single and double density DIMMs,
- SDRAM technology: 16, 64, 128, 256 and 512 Mbit,
- SDRAM speed: up to 200 MHz (DDR400).

29.2 SDRAM interface

29.2.1 Main memory configuration

The main memory is organized in rows. The data bus width can be programmed to 16 or 32 bits using bit LMI_MIM.BW. The population on each row ranges from 2 Mbytes to 2 Gbytes. The different rows may have different sizes or technology of SDRAM population, but must have the same data bus width, burst length and share the same timing parameters defined in register LMI_STR.

Note: If the bank remapping feature is enabled (see register LMI_COC) the different rows must also have the same split of row and column address bits and the same bank number. The bank remapping feature has to be disabled or enabled on both rows, so no mixed configuration (enabled on one row and disabled on the other) is allowed.

SDRAM devices on the same row must be the same kind (for example, 4 M x 16, 2-bank).

Note: The term row is used in two ways: an SDRAM device's internal row address and main memory subsystem's row array. In this chapter, row indicates subsystem's row, while (internal) row means SDRAM's row address.

The upper boundary address of each row is defined in LMI_SDRA[0:1].UBA. The request address [31:21] is compared to UBA[31:21] to determine which NOTLMICS (chip selection) signal is to be asserted. A NOTLMICS signal is applied to all SDRAM devices on the same row.

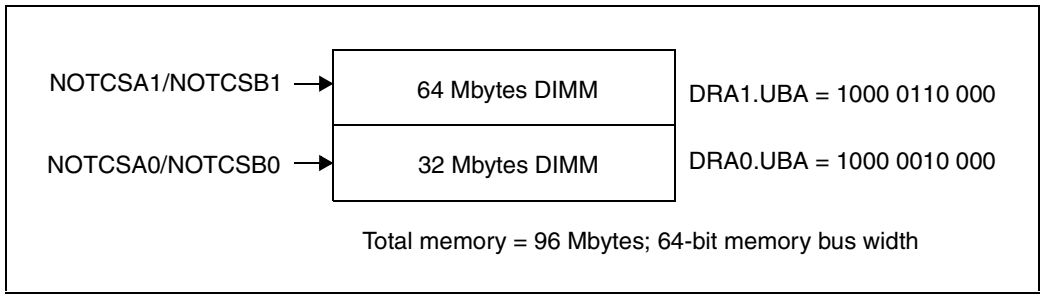
Memory locations in these two rows must be contiguous in physical address space. LMI_SDRA1.UBA must be larger or equal to LMI_SDRA0.UBA. If the system consists of only one row (or DIMM), then it needs to be placed in the area corresponding to NOTLMICS0 and LMI_SDRA0.UBA must be programmed = LMI_SDRA1.UBA. NOTLMICS0 is asserted if the STBus request access to the LMI data block and the request address [31:21] is less than LMI_SDRA0.UBA (exclusive).

LMI_SDRA0.UBA has priority over LMI_SDRA1.UBA if they are equal. If the physical address is less than LMI_SDRA0.UBA, notLMICS0 is asserted. If it is not less than LMI_SDRA0.UBA but is less than LMI_SDRA1.UBA, notLMICS1 is asserted. Other cases are errors and are recorded in LMI_VCR's error flag.

Memory configuration is fixed as little-endian.

Figure 63 depicts a 64-bit wide, 96 Mbyte main memory subsystem. It is assumed that **VCR.mem_base_addr** = 0x80.

Figure 63: Main memory configuration example



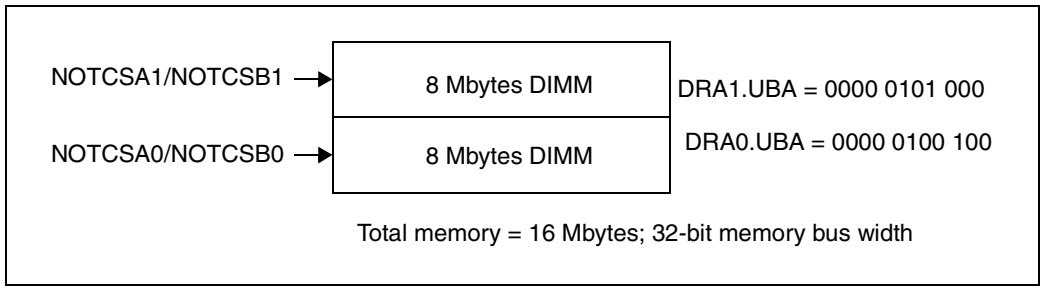
Due to the values of VCR.mem_base_addr, DRA0.UBA and DRA1.UBA we have:

0x82000000 - 0x80000000 = 32 Mbytes

0x86000000 - 0x82000000 = 64 Mbytes

Figure 64 depicts a 32 -bit wide, 16 Mbyte main memory subsystem. It is assumed that **VCR.mem_base_addr** = 0x04.

Figure 64: Main memory configuration example



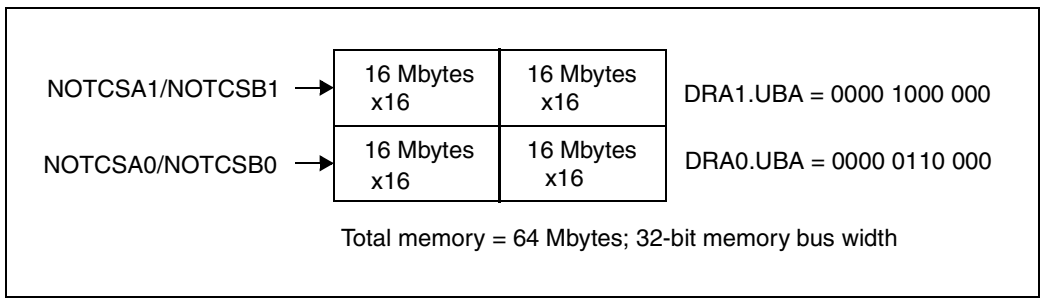
Due to the values of VCR.mem_base_addr, DRA0.UBA and DRA1.UBA we have:

0x04800000 - 0x04000000 = 8 Mbytes

0x05000000 - 0x04800000 = 8 Mbytes

Figure 65 depicts a 32 -bit wide, 64 Mbyte main memory subsystem. It is assumed **VCR.mem_base_addr** = 0x04. Each array is built using two x16 bit devices.

Figure 65: Main memory configuration example



Due to the values of VCR.mem_base_addr, DRA0.UBA and DRA1.UBA we'll have :

0x06000000 - 0x04000000 = 32 Mbytes

0x08000000 - 0x06000000 = 32 Mbytes

29.2.2 SDRAM interface pins

The external pins are described in Table 91.

To accommodate various loading conditions, the buffer strength of the pins is programmable. This feature can minimize unnecessary power consumption while still meeting the SDRAM device's timing requirements. See Section 29.2.4 for details.

Table 91: SDRAM interface pins

Name	I/O	Size (pins)	Description
LMI_CLK	O	1	SDRAM clock output: 66, 100, 133, 166 or 200 MHz
NOT_LMI_CLK	O	1	LMICLK and NOTLMICLK are differential clock outputs to DDR SDRAM
LMI_CLK_EN	O	1	Clock enable. Activates the clock signal when high and deactivates when low. By deactivating the clock, LMICLKEN low initiates the power-down mode, self-refresh mode or suspend mode.
NOT_LMI_CS[1:0]	O	2	Chip select. Selects particular SDRAM components during the active state.
NOT_LMI_WE	O	1	Write enable signal. NOTLMIWE is asserted during writes to SDRAM.
LMI_ADDR[12:0]	O	13	Row and column address
LMI_BK_SEL[1:0]	O	2	Bank address
LMI_DATA[31:0]	I/O	32	Memory data
LMI_DATA_STROBE[3:0]	I/O	4	Input/output data strobe These pins provide the read and write data strobe signal to/from the receiver circuit of the DRAM controller. The LMI drives DQS pins in write (store) cycles, while the DDR SDRAM drives them in read (load) cycles.
LMI_DATA_MASK[3:0]	O	4	Input/output data mask. For regular SDRAM, these pins act as synchronized output enables during read cycles and as byte enables during write cycles. For DDR SDRAM, these pins act as byte enables during write cycles.
NOT_LMI_RAS	O	1	Row address strobe
NOT_LMI_CAS	O	1	Column address strobe
LMI_VREF	I	1	Input reference voltage

29.2.3 SDRAM devices

The LMI splits the physical memory address into banks, (internal) row and column addresses. The LMI contains 15 external address pins. LMIBKSEL[1:0] specifies which bank, while LMIADD[12:0] indicates row and column addresses in each bank. The (internal) row address selects a page in an SDRAM. The column address selects data in a row. The LMI supports memories where row addresses are up to 13 bits.

The following three tables (Table 92, Table 93 and Table 94) summarize various SDRAM devices which are used to construct the memory subsystem in 2 different data bus widths. They also illustrate LMIADD pins muxing versus SDRAM address split. Pins LMIADD[12:0] are directly connected to SDRAM's A[14:0]. The address split column in the table specifies the row and column address split within a given bank.

Note: The mapping of LMIBKSEL[1:0] (bank select address bits) described in Table 92 and Table 93 can be bypassed enabling the bank remapping feature (see fields LMI_COC.BA_EN[1:0]).

Using the second entry as an example, 2 of 16 Mbit (2 M x 8 type, 2 banks) SDRAMs are used to construct a row of main memory. The SDRAMs' internal row and column address bits are 11 and 9, respectively. The page size is 1 Kbyte. Total memory on this row is 4 Mbytes. The CPU's physical address PA[11] is output to pin LMIBKSEL[0] in both RAS and CAS phases. LMIADD[10] is driven with PA[12] in RAS phase. The AP (auto precharge) option is output to LMIADD[10] in CAS phase.

16-bit data bus interface

Table 92: Row and column addressing for memory size and number of banks (16-bit interface)

SDRAM type	Address split	Page size	Row size	RAS CAS	LMIBKSEL		LMIADD					
					1	0	12	11	10/ AP	9	8	[7:0]
a Mbit x *b*			Mbyte									
16 Mbit 2 bank												
1 x 16	11 x 8	512 byte	2	RAS CAS		11 11			12 AP	10	9	[20:13] [8:1]
2 x 8	11 x 9	1 Kbyte	4	RAS CAS		11 11			12 AP	10	21 9	[20:13] [8:1]
4 x 4	11 x 10	2 Kbyte	8	RAS CAS		11 11			12 AP	22 10	21 9	[20:13] [8:1]
64 Mbit 2 bank												
4 x 16	13 x 8	512 byte	8	RAS CAS		11 11	12	10	9 AP	22	21	[20:13] [8:1]
8 x 8	13 x 9	1 Kbyte	16	RAS CAS		11 11	12	10	23 AP	22	21 9	[20:13] [8:1]
16 x 4	13 x 10	2 Kbyte	32	RAS CAS		11 11	12	24	23 AP	22 10	21 9	[20:13] [8:1]
64 Mbit 4 bank												
4 x 16	12 x 8	512 byte	8	RAS CAS	12 12	11 11		10	9 AP	22	21	[20:13] [8:1]
8 x 8	12 x 9	1 Kbyte	16	RAS CAS	12 12	11 11		10	23 AP	22	21 9	[20:13] [8:1]
16 x 4	12 x 10	2 Kbyte	32	RAS CAS	12 12	11 11		24	23 AP	22 10	21 9	[20:13] [8:1]
128 Mbit 4 bank												
8 x 16	12 x 9	1 Kbyte	16	RAS CAS	12 12	11 11		10	23 AP	22	21 9	[20:13] [8:1]
16 x 8	12 x 10	2 Kbyte	32	RAS CAS	12 12	11 11		24	23 AP	22 10	21 9	[20:13] [8:1]
32 x 4	12 x 11	4 Kbyte	64	RAS CAS	12 12	25 25		24 11	23 AP	22 10	21 9	[20:13] [8:1]
256 Mbit 4 bank												
16 x 16	13 x 9	1 Kbyte	32	RAS CAS	12 12	11 11	10	24	23 AP	22	21 9	[20:13] [8:1]
32 x 8	13 x 10	2 Kbyte	64	RAS CAS	12 12	11 11	25	24	23 AP	22 10	21 9	[20:13] [8:1]
64 x 4	13 x 11	4 Kbyte	128	RAS CAS	12 12	26 26	25	24 11	23 AP	22 10	21 9	[20:13] [8:1]
512 Mbit 4 bank												
32 x 16	13 x 10	2 Kbyte	64	RAS CAS	12 12	11 11	10	24	23 AP	22	21 9	[20:13] [8:1]

32-bit data bus interface

Table 93: Row and column addressing for memory size and number of banks (32-bit interface)

SDRAM type	Address split	Page size	Row size	RAS CAS	LMIBKSEL		LMIADD					
					1	0	12	11	10/ AP	9	8	[7:0]
a Mbit x b		Kbyte	Mbyte									
16 Mbit 2 bank												
1 x 16	11 x 8	1	4	RAS CAS		13 13			12 AP	11	10	[21:14] [9:2]
2 x 8	11 x 9	2	8	RAS CAS		13 13			12 AP	11	22 10	[21:14] [9:2]
4 x 4	11 x 10	4	16	RAS CAS		13 13			12 AP	23 11	22 10	[21:14] [9:2]
64 Mbit 2 bank												
4 x 16	13 x 8	1	16	RAS CAS		13 13	12	11	10 AP	23	22	[21:14] [9:2]
8 x 8	13 x 9	2	32	RAS CAS		13 13	12	11	24 AP	23	22 10	[21:14] [9:2]
16 x 4	13 x 10	4	64	RAS CAS		13 13	12	25	24 AP	23 11	22 10	[21:14] [9:2]
64 Mbit 4 bank												
2 x 32	11 x 8	1	8	RAS CAS	12 12	13 13			10 AP	11	22 AP*	[21:14] [9:2]
4 x 16	12 x 8	1	16	RAS CAS	12 12	13 13		11	10 AP	23	22	[21:14] [9:2]
8 x 8	12 x 9	2	32	RAS CAS	12 12	13 13		11	24 AP	23	22 10	[21:14] [9:2]
16 x 4	12 x 10	4	64	RAS CAS	12 12	13 13		25	24 AP	23 11	22 10	[21:14] [9:2]
128 Mbit 4 bank												
4 x 32	12 x 8	1	16	RAS CAS	12 12	13 13		11	10 AP	23	22 AP*	[21:14] [9:2]
8 x 16	12 x 9	2	32	RAS CAS	12 12	13 13		11	24 AP	23	22 10	[21:14] [9:2]
16 x 8	12 x 10	4	64	RAS CAS	12 12	13 13		25	24 AP	23 11	22 10	[21:14] [9:2]
32 x 4	12 x 11	8	128	RAS CAS	26 26	13 13		25 12	24 AP	23 11	22 10	[21:14] [9:2]
256 Mbit 4 bank												
16 x 16	13 x 9	2	64	RAS CAS	12 12	13 13	11	25	24 AP	23	22 10	[21:14] [9:2]
32 x 8	13 x 10	4	128	RAS CAS	12 12	13 13	26	25	24 AP	23 11	22 10	[21:14] [9:2]
64 x 4	13 x 11	8	256	RAS CAS	27 27	13 13	26	25 12	24 AP	23 11	22 10	[21:14] [9:2]
512 Mbit 4 bank												
32 x 16	13 x 10	4	128	RAS CAS	12 12	13 13	26	25	24 AP	23 11	22 10	[21:14] [9:2]

Note: AP pin: the LMI uses bit LMI_MIM.BY32AP to determine if bit LMIADD8 is used to indicate the PRE and PALL commands.

29.2.4 Initializing SDRAM devices

The boot/driver software must initialize SDRAM devices after power-on reset. No software should access SDRAM before initialization is complete.

There are also several configuration fields to set up in the system configuration register SYS_CFG11, see Chapter 21: *System configuration registers on page 171*.

Double data rate SDRAM

1. The system provides four supplies in a certain sequence: VDD first, VDDQ next, then VREF and VTT. VTT is not provided to the LMI, it is externally connected through a series of termination registers. This is required to prevent latch-up in SDRAM devices. During and after power-on reset, LMICLKEN must be kept low.
2. After all power supply and reference voltages are stable, and the clock is stable, a 200 µs pause is necessary.
3. LMICLKEN should be brought high with the **deselect** command. After this point, unless the LMI sends another command, the LMI has to send the **deselect** command.
4. A precharge all (PALL) is issued to the SDRAM.
5. A mode register set (MRS) command is issued to program the extended mode register to enable the DLL. The MRS command is issued to program the mode register, reset the DLL in SDRAM and program the operating parameters, for example burst length and CAS latency.
6. A precharge all (PALL) is issued to the SDRAM.
7. Wait ten cycles after the DLL reset and send two CBR commands to the SDRAM.
8. A MRS command is issued to de-assert the DLL initialization bit in the mode register. Other programing parameters should be the same as in previous programing. For some memory vendors, this step can be skipped because they support auto clearing of the DLL initialization bit.
9. After 200 cycles from DLL reset, external memory becomes accessible.

The LMI SDRAM controller provides two mechanisms to accomplish the initialization sequence.

1. NOP, PALL, CKEH and CBR:

   Field LMI_SCR.SMS (SDRAM mode select) is written with appropriate values to prompt the SDRAM controller to start issuing one of these commands. For instance, SMS = 100 results in a single CBR cycle on the SDRAM interface. When SMS = 011, the LMICLKEN signal goes high. See LMI_SCR for details.
2. Setting the SDRAM device's mode register:

   The SDRAM's mode register must be initialized before actual operation. The software (boot code) initiates a write cycle to LMI_MIM, and then a write, to register LMI_SDMR[0:1] in the control block. The SDRAM controller then issues an MRS command to all SDRAM devices on row *n*.

Example: issuing MRS command to array 0

Software does a dummy write to SDMR0; the physical address and data must be arranged in the following way:

- STBus Addr = 0x(VCR.reg_base_addr {SYS:0x0F00 0000 or VID:0x1700 0000}) + 0x0048,
- STBus data bus [15:0] contains the value to be written to the SDRAM's mode register,
- data[9:0] is copied to MA[9:0], data[15:12] to MA[13:10] and data [11:10] to BA[1:0] when an MRS command is issued to the SDRAM devices.

Software needs to ensure that the SDRAM timing specification (between the MRS command and the first operational command) is met. One way to ensure this is to perform several SDRAM control register reads.

Subsequently, LMI_SCR.SMS is written with 0b000; the normal SDRAM operation can then be started.

Note: Software must program register LMI_MIM before writing to LMI_SDMR[0:1].

29.2.5 Operations

The SDRAM controller supports most DDR SDRAM commands. The following truth table lists all commands supported.

Table 94: SDRAM command truth table

Function	Symbol	LMICLKEN		NOTLMI...				LMIADD			LMI BKSEL [1: 0]
		[*n* - 1]	[*n*]	CS	RAS	CAS	WE	11	AP[a], (10,8)	[9, 0]	
Device deselect	DSEL	H	X	H	X	X	X	X	X	X	X
No operation	NOP	H	X	L	H	H	H	X	X	X	X
Burst stop in read	BST	H	X	L	H	H	L	X	X	X	X
Read	READ	H	X	L	H	L	H	V	L	V	V
Write	WRITE	H	X	L	H	L	L	V	L	V	V
Bank activate	ACT	H	X	L	L	H	H	V	V	V	V
Precharge select bank	PRE	H	X	L	L	H	L	V	L	X	V
Precharge all banks	PALL	H	X	L	L	H	L	X	H	X	X
Auto refresh	CBR	H	H	L	L	L	H	X	X	X	X
Self refresh entry from idle	SLFRSH	H	L	L	L	L	H	X	X	X	X
Self refresh exit	SLFRSHX	L	H	H	X	X	X	X	X	X	X
Power-down entry from idle	PWRDN	H	L	X	X	X	X	X	X	X	X
Power-down exit	PWRDNX	L	H	H	X	X	X	X	X	X	X
Mode register set	MRS	H	X	L	L	L	L	V	V	V	V

a. AP pin: the LMI uses LMI_MIM.BY32AP to determine if LMIADD8 pin is used to indicate PRE and PALL commands.

Note: The LMI does not support full-page burst operation. The LMI issues a BST command to terminate the burst read-only in DDR SDRAM mode.

The timing for issuing these commands is governed by the SDRAM timing register (see LMI_STR for details). The LMI SDRAM controller can open up to four pages for each SDRAM row and fully exploit the multi-bank architecture of modern SDRAM devices by tightly pipelining SDRAM commands. The LMI is capable of detecting multiple consecutive requests to the same SDRAM page. The SDRAM controller may combine same-page requests into a single same-page access, providing that the timing of the requests is suitable.

29.2.6 Refresh

When DRAM refresh enable is 1 (LMI_MIM.DRE = 1). The LMI can automatically generate refresh cycles. A 12-bit quantity (LMI_MIM.DRI, DRAM refresh interval) specifies the number of memory clock cycles between refreshes. Software should program DRI in the inclusive range [128:4095]. The behavior of the DRAM controller is undefined if the LMI is enabled and if DRI is less than 128.

At the start of a refresh interval, the SDRAM controller loads the DRI 12-bit value into an internal counter. This counter is decremented by 1 in each memory clock cycle. When the counter reaches 0, the DRI value is reloaded into the counter and the next refresh interval is started.

All banks must be closed before a refresh operation can be performed. The SDRAM controller issues a precharge all (PALL) command if there are any open pages. It then issues an auto refresh command (CBR) after the Trp parameter is satisfied. The next row ACT command can be issued one Trfc clock period (LMI_STR.SRFC) later.

The SDRAM controller performs exactly one refresh operation for each refresh interval. It attempts to perform CBR as soon as possible within the refresh interval. When the counter ≤128 and CBR is not issued in the current refresh interval, the SDRAM controller causes any current SDRAM access to complete in a timely manner by ensuring that the detection of same-page SDRAM access is prevented. Subsequently it performs PALL and CBR commands.

The maximum refresh rate that the LMI can support is one row every 128 clock cycles. At this rate, however, the detection of same-page SDRAM accesses is permanently disabled.

As an example, the hard reset value of DRI is 1562. For 100 MHz LMICLK, this allows 1024 refreshes in less than 16 ms.

Note: On average, the interval between two refreshes is determined by the DRI setting. However the interval between any two successive refreshes could be larger or smaller than DRI by (a page miss 32-byte transfer) clocks.

29.2.7 Power management

The LMI provides a single power management mechanism, contolled through configuration register SYS_CFG11, see Chapter 21: *System configuration registers on page 171*.

When the LMI receives a standby request, the LMI prepares the SDRAM rows to enter a low-power state. The sequence of events for both entering and leaving standby mode is described below. To make the correct sequence, cooperation with the software driver is important.

Entering standby

1. First, no initiators should be issuing transaction requests to the LMI.
2. The standby management program issues CBR command as the last command to the LMI.
3. The standby management program asserts STBY_REQ to the LMI.
4. All outstanding transaction requests are serviced.
5. The SDRAM controller issues a self-refresh command and lowers LMICLKEN to both SDRAM rows. The SDRAM autonomously refreshes itself for the duration of the power-down mode.
6. The memory clock (LMICLK) can now be stopped. The clock controlling the padlogic flip-flops can be switched off.

The LMI core switches off its clock under these conditions (STBY_REQ and STBY_ACK both high).

Leaving standby by causes other than power-on reset

1. STBY_REQ is de-asserted. As a result of this, the LMI STBus clock and SDRAM clock inside the LMI core resume. The same happens in the padlogic if the same clock gating logic (STBY_REQ/STBY_ACK combined to act as enable of gated clock cell) are implemented outside the LMI core.
2. The LMI de-asserts STBY_ACK, and starts to count down from (256 x LMI_SCR.CST) to zero every LMICLK cycle.
3. When the countdown reaches zero, the SDRAM controller asserts LMICLKEN and sends DESELECT commands continuously. All SDRAM rows exit from self-refresh mode.
4. The first valid command can be issued ten cycles after LMICLKEN's rising edge.
5. For a DDR SDRAM, the LMI issues numbers of CBR commands set by LMI_SCR.BRFSH.

29.2.8 Caution when programming SDRAM's mode register

The LMI SDRAM controller uses DT (SDRAM type) and BW (external data bus width) to determine the burst length (register LMI_MIM).

For a 16-bit external data bus width, the LMI splits an STBus load32 or store32 packet into multiple SDRAM transactions, with a burst of 8 for each transaction. The BL field of the SDRAM device's mode register must thus be programmed to match the LMI burst length behavior in the third column of Table 95 below.

Table 95: Determining burst length for load32 and store32 packets

MIM.DT device type	MIM.BW bus width	burst length
0: PC SDRAM	00: 16-bit	8
	01: 32-bit	
	10: 64-bit	4
1: DDR SDRAM	00: 16-bit	8
	01: 32-bit	8
	10: 64-bit	4

29.2.9 Using registered DIMM

When using registered DIMM, bit LMI_MIM.DIMM needs to be set to 1, so that the LMI can:

- delay data output by one cycle to synchronize with the buffered (on DIMM card) command signals before they reach the SDRAM devices during a write operation,
- add one LMICLK cycle to the setting of bit LMI_STR.SCL (CAS latency). SCL should be programmed with the same CL latency as the CL setting in the SDRAM device's mode register.

29.2.10 Others

Memory access to the address range of the base address + (0x0400 0000 to 0xEFF FFFF), is routed to the LMI. This address range may not be fully populated with SDRAMs. Data access outside the populated addresses, as defined in LMI_SDRA1.UBA, does not result in an external memory transaction. Software that dynamically sizes the amount of external memory must use an algorithm that is aware of this property. In the case of DIMMs, software can use I/O pins to implement a serial presence detect (SPD) mechanism for dynamic sizing of main memory.

30 LMI registers

There are two sets of LMI registers, for system and video LMI respectively.

Register addresses are provided as
LMIBaseAddress + offset or *MicroSysGlueConfigBaseAddress* + offset.

LMIBaseAddress is set to either *LMISysBaseAddress or LMIVidBaseAddress* depending on which LMI is being addressed.

The *LMISysBaseAddress* is:

0x0F00 0000.

The *LMIVidBaseAddress* is:

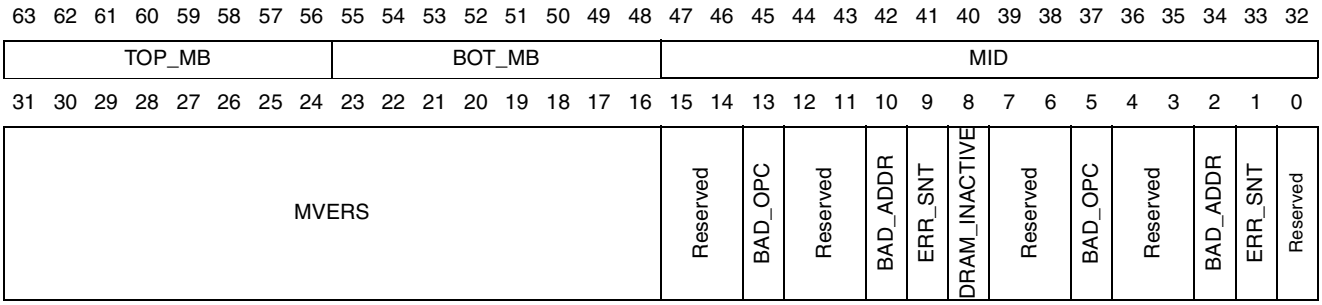
0x1700 0000.

Table 96: LMI register summary

Register	Description	Size (bits)	Offset	Type
LMI_VCR	Version control	64	0x00	R/W
LMI_MIM	Memory interface mode	32	0x08	
LMI_SCR	SDRAM control	32	0x10	
LMI_STR	SDRAM timing	32	0x18	
Reserved	-	-	0x20	-
LMI_COC	Latency and strobe parameters for ASB	32	0x28	R/W
LMI_SDRA[0:1]	SDRAM row attribute	32	0x30, 0x38	
LMI_CIC	Latency out parameters for ASB	32	0x40	RO
LMI_SDMR[0:1]	SDRAM mode	32	0x48, 0x50	WO

There are several additional LMI registers described in Chapter 21: *System configuration registers on page 171*.

LMI_VCR Version control



Address: *LMIBaseAddress* + 0x0000, 0x0004

Type: R/W

Reset: 0x0F08 1000 0002 0000

Description:

[63:56] **TOP_MB**: Top data memory block; Identifies top of data memory block. Read only. Reset: 0x0F

[55:48] **BOT_MB**: Bottom data memory block. Identifies bottom of data memory block. Read only. Reset: 0x08

[47:32] **MID**: Module identity. Identifies module. Read only. Reset: 0x1000

[31:16] **MVERS**: Module version. Indicates module version number. Read only. Reset: 0x0002

[15:8] **MERR**: Memory error flags. Indicates errors in the LMI. The error status due to access to the LMI data block is recorded in MERR field. The set of supported MERR flags is given below.

[15:14] **Reserved**: Reset: 0

[13] **BAD_OPC**: A request with an unsupported opcode has been received (read/write). This bit is set by the LMI hardware if a request with an unsupported opcode is received by LMI from STBus. Reset: 0

[12:11] **Reserved**: Reset: 0

[10] **BAD_ADDR**: Request to an out-of-range or unpopulated address received (read/write). This bit is set by the LMI hardware if a request directed to an out-of-range address or an unpopulated address in data block is received. Reset: 0

[9] **ERR_SNT**: Error response sent (read/write). This bit is set by the LMI hardware if an error response is sent by the LMI to STBus. It indicates that an earlier request to the LMI data block was invalid. Reset: 0

[8] **DRAM_INACTIVE**: Access to LMI data block (that is, external memory) when DRAM controller is disabled (read/write). This bit is set by the LMI hardware if a request is made to access external memory while DRAM controller is disabled. Reset: 0

[7:0] **PERR**: Port error flags. Indicates errors in the interface between LMI and the packet-router. The error status due to access to the LMI control block is recorded in the PERR field. The set of supported PERR flags is given below.

[7:6] **Reserved**: Reset: 0

[5] **BAD_OPC**: Unsupported opcode (read/write). This bit is set by the LMI hardware if a request with an unsupported opcode is received by LMI from STBus. Reset: 0

[4:3] **Reserved**: Reset: 0

[2] **BAD_ADDR**: Undefined control register (read/write). This bit is set by the LMI hardware if the LMI hardware receives a request for an undefined control register. Reset: 0

[1] **ERR_SNT**: Error response sent (read/write). This bit is set by the LMI hardware if an error response is sent by the LMI to STBus. It indicates that an earlier request to the LMI was invalid. Reset: 0

[0] **Reserved**. Reset: 0

LMI_MIM Memory interface mode

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
STO_TOUT								LD_TOUT								Reserved							LIMIT					FRAME			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRAME								DRI								Reserved				BY32AP	DIMM	DRE	Reserved	BW		Reserved				DT	DCE

Address: *LMIBaseAddress* + 0x0008

Type: R/W

Reset: 0x061A 0080

Description: This register specifies the configuration of the DRAM interface.

[63:56] **STO_TOUT**: Store timeout
store_timeout for LMI arbiter
Specifies the store_timeout
0: Inactive 1: 8 cycles 2: 16 cycles ...
n: 8 * load_timeout
n can be in the range 0 - 255 (so *load_timeout* is in the range 0 - 2040 clock cycless).

[55:48] **LD_TOUT**: Load timeout
load_timeout for arbiter
0: Inactive 1: 8 cycles 2: 16 cycles ...
n: 8 * load_timeout
n can be in the range 0 - 255 (so *load_timeout* is in the range 0 - 2040 clock cycless).

[47:41] **Reserved**

[40:38] **LIMIT**: Limit parameter for bandwidth limiter
Specifies the maximum number of 8-byte words that the initiator can transfer, according to the formula:
0: 2 words 1: 4 words ... *n*: 2 * (*words_limit* + 1)
n can be in the range 0 - 31 (so *limit* is in the range 2 - 64 words or equivalent 16 - 512 bytes).

[35:28] **FRAME**: Frame size parameter for bandwidth limiter
Specifies the frame size window
0: Inactive 1: 8 cycles 2: 16 cycles ...
n: 8 * *frame_size* cycles
n can be in the range 0 - 255 (so *frame_size* is in the range 0 - 2040 clock cycles)

[27:16] **DRI**: DRAM refresh interval
When enabled, determines the maximum number of memory clock cycles between row refreshes.

[15:12] **Reserved**

[11] **BY32AP**: Interfacing x32 SDRAM devices
Pre-charges all bank command's indicator for x32 SDRAM devices.
0: BY32AP (LMIADD8 pin is not used when LMI issues PRE or PALL commands).
1: BY32AP (LMIADD8 pin is used when LMI issues PRE or PALL commands).

[10] **DIMM**: Registered DIMM module
Constructs the external array.
1: Data output delayed by one LMICLK cycle and one LMICLK cycle added to CAS latency

[9] **DRE**: DRAM refresh enable. Enable refresh mechanism.

[8] **Reserved**

[7:6] **BW**: Bus width
Indicates the data bus width of the LMI SDRAM interface.
00: 16 01: 32 10: 64 11: Reserved

[5:2] **Reserved**

[1] **DT**: SDRAM type
0: PC-SDRAM 1: DDR SDRAM

[0] **DCE**: DRAM controller enable
Indicates whether the SDRAM controller is enabled or disabled.
When the SDRAM controller is disabled, the LMI generates an error responses to STBus requests (for data block access) directed to the LMI.
0: SDRAM controller is disabled 1: SDRAM controller is enabled

LMI_SCR SDRAM control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CST												Reserved									BRFSH			PDSE	SMS			

Address: *LMIBaseAddress* + 0x0010
Type: R/W
Reset: 0
Description:

[63:28] **Reserved**

[27:16] **CST**: Clock stabilization time
These bits specify clock stabilization time on return from module standby mode.
0: No stabilization time before LMI start operation with SDRAM.
<>0: Count down this number with 1/256 LMICLK. When it reached to 0, LMI starts operation.

[15:7] **Reserved**

[6:4] **BRFSH**: Burst refresh. This bit enables burst refresh after wake up from standby.
000: No 001: 32
010: 512 011: 1024
100: 2048 101: 4096
110: Reserved 111: Reserved

[3] **PDSE**: Power-down SDRAM enable
Allows the SDRAM controller to issue a power-down command to an idle SDRAM row. The SDRAM controller issues a power down exit command when there is a request to access this idle row.
0: Disable 1: Enable

[2:0] **SMS**: SDRAM mode select
Enables the SDRAM controller to perform normal SDRAM operation and to issue NOP, PALL and CBR which are required in the SDRAM device initialization sequence for power up or reset.
000: Normal SDRAM operation when LMI_MIM.DCE = 1.
001: NOP command enable. When SMS is written with this value and DCE = 1, the LMI issues 1 NOP command to the SDRAM interface. To have *n* NOP commands, SMS must be written with 001, *n* times. This command applies to all external SDRAM rows.
010: Precharge all banks. When SMS is written with this value and DCE = 1, the LMI issues 1 precharge all command to the SDRAM interface. To have *n* PALL commands, SMS must be written with 010, *n* times. This command applies to all external SDRAM rows.
011: Clock enable signals LCLKEN0 and LCKLKEN1 active. At reset the clocks are disabled.
100: CBR enable. When SMS is written with this value and DCE = 1, the LMI issues 1 CBR command to the SDRAM interface. To have *n* CBR commands, SMS must be written with 011, *n* times. This command applies to all external SDRAM rows.
101, 110, 111: Reserved

LMI_STR SDRAM timing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			SXSR				SRCDW		SWTR	TWTR_DIS	SPDL[1]		SRFC			SCL[3]	SCL[2:0]			SPDL[0]	SRRD	SRAS			SRC				SRCDR		SRP	

Address: *LMIBaseAddress* + 0x0018

Type: R/W

Reset: 0

Description:

[31:30] **Reserved**

[29:26] **SXSR**: Exit self-refresh to next command
Specify the number of cycles to wait after the self-refresh exit command before restarting sending commands to the DDR
0: No time to wait.
1+: Count down this number with 1/16 memory clock. When it reaches 0, LMI starts operation.

[25:24] **SRCDW**: T_{RCDW}, RAS to write CAS delay
Controls the number of memory clock cycles from a ROW ACTIVATE command to a write column command (for the same bank) for WRITE operations.
00: 2 clocks of RAS to CAS delay 01: 3 clocks of RAS to CAS delay
10: 4 clocks of RAS to CAS delay 11: 5 clocks of RAS to CAS delay

[23] **SWTR**: T_{WTR}, write to read interruption. T_{WTR} parameter
Last “data-in (during write) to read command” parameter
0: 1 clock 1: 2 clocks

[22] **TWTR_DIS**: Disable the write to read interruption
0: Enable write to read interruption. 1: Disable write to read interruption.

[21] **SDPL[1]**: see SDPL[0]

[20:17] **SRFC**: T_{RFC}, auto refresh RAS cycle time. Minimum delay between auto refresh and ACT (to the same bank), auto refresh and auto refresh (to the same bank).
0000: 6 clocks 0001: 7 clocks 0010: 8 clocks 0011: 9 clocks
0100: 10 clocks 0101: 11 clocks 0110: 12 clocks 0111: 13 clocks
1000: 14 clocks 1001: 15 clocks 1010: 16 clocks 1011: 17 clocks
All others: Reserved

[16:13] **SCL**: SDRAM CAS latency (CL). Controls the number of LMICLKs between when a read command is sampled by the SDRAMs and when the processor samples read data from SDRAMs.
0010: 2 clocks 0011: 3 clocks 1001: 1.5 clocks 1010: 2.5 clocks
0100: 4 clocks All others: Reserved

[12] **SDPL[1]**: SDRAM T_{DPL}, as well as DDR SDRAM’s T_{WR}. SDRAM: last write-data to PRE or PALL command period DDR SDRAM: from the end of postamble to PRE or PALL command.
STR[12] STR[21]
00: 1 clock 01: 2 clocks 10: 3 clocks 11: 4 clocks

[11] **SRRD**: T_{RRD}, RAS to RAS active delay. Specifies delay from ACT bank *n* to ACT bank *i* command (different bank).
0: 2 clocks 1: 3 clocks

[10:8] **SRAS**: T_{RAS}, RAS active time. ACT to PRE command (for the same bank).
001: 5 clocks 010: 6 clocks 011: 7 clocks 100: 8 clocks
101: 9 clocks 110: 10 clocks All others: Reserved

[7:4] **SRC**: T_{RC}, RAS cycle time. Minimum delay between ACT and auto refresh (to the same bank), ACT and ACT (to the same bank).
0000: 6 clocks 0001: 7 clocks 0010: 8 clocks 0011: 9 clocks
0100: 10 clocks 0101: 11 clocks 0110: 12 clocks 0111: 13 clocks
1000: 14 clocks 1001: 15 clocks All others: Reserved

[3:2] **SRCDR**: T_{RCD}, RAS to CAS delay. Controls the number of LMICLKs from a ROW ACTIVATE command to a column command (for the same bank) for READ operations.
00: 2 clocks of RAS to CAS delay 01: 3 clocks of RAS to CAS delay
10: 4 clocks of RAS to CAS delay 11: 5 clocks of RAS to CAS delay

[1:0] **SRP**: T_{RP}, RAS precharge to ACT command. Controls the number of LMICLKs for RAS precharge to ACT command (for the same bank).
00: 2 clocks of RAS precharge 01: 3 clocks of RAS precharge
10: 4 clocks of RAS precharge 11: 5 clocks of RAS precharge

LMI_COC Latency and strobe parameters for ASB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		RX_LATENCY_1			TX_LATENCY_1			STROBE_1	RX_LATENCY_0			TX_LATENCY_0			STROBE_0

Address: *LMIBaseAddress* + 0x0028
Type: R/W
Reset: 0
Description: Latency and strobe parameters for asynchronous bridge (ASB)

[31:14] **Reserved**

[13:11] **RX_LATENCY_1**: Receive latency for ASB attached to high-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

[10:8] **TX_LATENCY_1**: Transmit latency for ASB attached to high-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

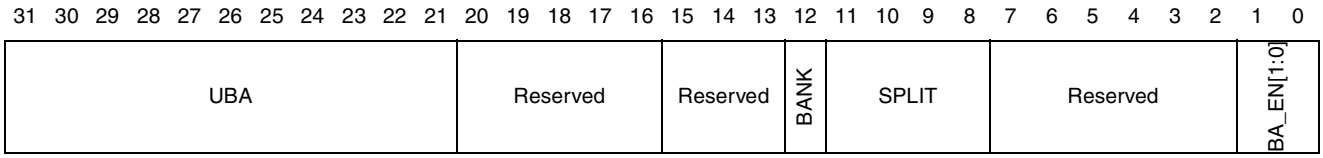
[7] **STROBE_1**: Strobe signal to validate configuration data for ASB attached to high-priority port

[6:4] **RX_LATENCY_0**: Receive latency for ASB attached to low-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

[3:1] **TX_LATENCY_0**: Transmit latency for ASB attached to low-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

[0] **STROBE_0**: Strobe signal to validate configuration data for ASB attached to low-priority port

LMI_SDRA[0:1] SDRAM row attribute



Address: *LMIBaseAddress* + 0x0030, 0x0038

Type: R/W

Reset: 0x00

Description:

[63:32] **Reserved**

[31:21] **UBA**: Row upper boundary address.
Defines the upper boundary address of the external SDRAM row in 2 Mbyte granularity. UBA specifies the external row's upper boundary address [21, 31].

[20:13] **Reserved**

[12] **BANK**: SDRAM device bank number. Defines the SDRAM device bank number of the associated physical memory row.
0: Dual-bank 1: Quad-bank

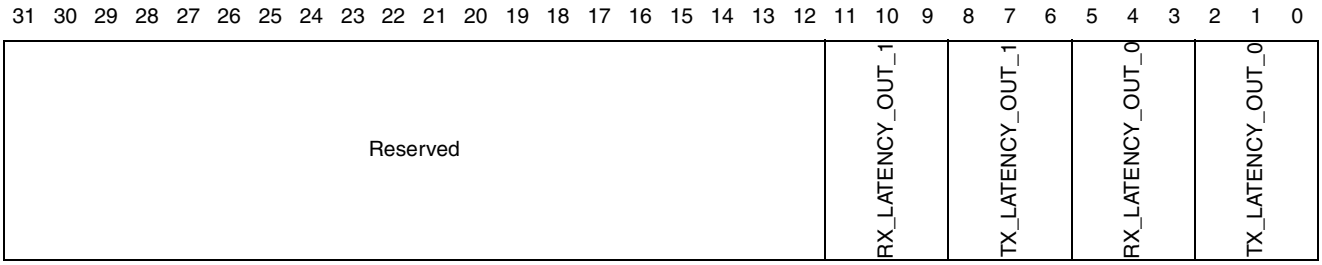
[11:8] **SPLIT**: SDRAM device address split for each bank. Defines the split of row and column address bits for a given bank within an SDRAM device.
0000: 11x8 0001: 11x9
0010: 11x10 0011: Reserved
0100: 12x8 0101: 12x9
0110: 12x10 0111: Reserved
1000: 13x8 1001: 13x9
1010: 13x10 1011: 13x11
11*nn*: Reserved

[7:2] **Reserved**

[1] **BA1_EN**: Enable/disable bank remapping for LMIBKSEL1.
If this feature is enabled, bit 9 of the incoming STBus address is swapped with the STBus address bit that the LMI considers to use as LMIBKSEL1 according to its configuration as in tables 6, 7, 8. This swap is done before the STBus address is processed by the LMI and only in case of external memory accesses (that is, the swap is not done during a modeset operation and internal register accesses).
0: Disable bank remapping feature for LMIBKSEL1 1: Enable bank remapping feature for LMIBKSEL1

[0] **BA0_EN**: Enable/disable bank remapping for LMIBKSEL0.
If this feature is enabled, bit 8 of the incoming STBus address is swapped with the STBus address bit that the LMI considers to use as LMIBKSEL0 according to its configuration as in tables 6, 7, 8. This swap is done before the STBus address is processed by the LMI and only in case of external memory accesses (that is, the swap is not done during a modeset operation and internal register accesses).
0: Disable bank remapping feature for LMIBKSEL0 1: Enable bank remapping feature for LMIBKSEL0

LMI_CIC Latency out parameters for ASB



Address: *LMIBaseAddress* + 0x0040

Type: RO

Reset: 0

Description: Latency out parameters for asynchronous bridge (ASB)

[31:12] **Reserved**

[11:9] **RX_LATENCY_OUT_1**: Receive latency out for ASB attached to high-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

[8:6] **TX_LATENCY_OUT_1**: Transmit latency out for ASB attached to high-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

[5:3] **RX_LATENCY_OUT_0**: Receive latency out for ASB attached to low-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

[2:0] **TX_LATENCY_OUT_0**: Transmit latency out for ASB attached to low-priority port
000: Latency of 8 cycles 001: Latency of 7 cycles ...

LMI_SDMR[0:1] SDRAM mode

Address: *LMIBaseAddress* + 0x0048, 0x0050

Type: WO

Description: These are write-only virtual registers, since physically they are not contained in the processor chip. A write to these virtual registers triggers an SDRAM mode register set command to be issued to a array of SDRAM devices.

The value on physical address A[9:0] is copied to LMIADD[9:0] pins, A[11:10] is output to LMIBKSEL[1:0] and A[15:12] is driven to LMIADD[13:10].

In response to the mode register set command, an SDRAM or DDR SDRAM device then latches LMIADD[11:0] and LMIBKSEL[1:0] into its mode register. A read to these registers returns undefined value.

Refer to the SDRAM and DDR SDRAM manufacture's datasheets for the definition of each bit in its mode register and extended mode register.

Note: The definition of a mode register's bit field varies with different SDRAM density.

31 External memory interface (EMI)

31.1 Overview

The EMI is a general purpose external memory interface which allows the system to support a number of memory types, external process interfaces and devices. This includes glueless support for up to five independent memories or devices.

31.2 Features

The main features include:

- 16-bit interface,
- up to 100 MHz operating frequency,
- support for up to five external memory banks,
- asynchronous or burst Flash support (AMD AM29BL162C, ST M58LW064, INTEL 28F160F3 compatible) (connectable banks 0 to 4),
- asynchronous peripheral support (SRAM and ROM) (connectable banks 0 to 4),
- optional PC Card support for PCMCIA and CableCARD™ (POD) modules in banks 3 and 4.

The EMI memory map is divided into five regions (EMI banks) which may be independently configured to accommodate one of SRAM, ROM, asynchronous or burst Flash.

Each bank can only accommodate one type of device, but different device types can be placed in different banks to provide glueless support for mixed memory systems.

EMI endianness is fixed at system reset and cannot be changed dynamically. Bit positions are numbered left to right from the most significant to the least significant. Thus in a 32-bit word, the leftmost bit, 31, is the most significant bit and the rightmost bit, 0, is the least significant.

The external data bus can be configured to be 8 or 16 bits wide on a per-bank basis.

Note: The STx7100 does not support SDRAM on its EMI interface.

31.3 EMI interface pins

The external pins are described in Table 97.

Table 97: EMI interface pins

Name	I/O	Size	Description
NOTEMICS0	Output	1	All devices. Device chip select for bank 0.
NOTEMICS1	Output	1	All devices. Device chip select for bank 1.
NOTEMICS2	Output	1	All devices. Device chip select for bank 2.
NOTEMICS3	Output	1	All devices. Device chip select for bank 3.
NOTEMICS4	Output	1	All devices. Device chip select for bank 4.
NOTEMIBE[1:0]	Output	2	All devices. Device databus byte enable. Alternatively, bit[1] low order address bit for 8 bit devices. NOTEMIBE[0] becomes PCCARD_IOWR# during PC Card accesses.
NOTEMIOE	Output	1	All devices. Device output enable. NOTEMIOE becomes PCCARD_OE# during PC Card accesses.
NOTEMILBA	Output	1	SFlash devices only. Load burst address. NOTEMILBA becomes PCCARD_WE# during PC Card accesses.
NOTEMIBAA	Output	1	Some SFlash devices only. Burst address advanced. NOTEMIBAA becomes PCCARD_IORD# during PC Card accesses.
EMITREADYORWAIT	Input	1	Device target ready indicator.
EMIRDNOTWR	Output	1	All devices. Common read/write access indicator.
EMIDATA[15:0]	I/O	16	All devices. Common data bus.
EMIADDR[23:1]	Output	23	All devices. Common address bus.
EMIFLASHCLK	Output	1	SFlash devices only. SFlash clock.
EMIPRTSIZE	Output	1	Boot device port size (0 = 16 bits; 1 = 8 bits).
EMIBUSREQ	Input	1	EMI Bus ownership request from external agent.
EMIBUSGNT	Output	1	EMI Bus ownership grant to external agent.

31.4 EMI address map

The EMI is allocated a 16 Mbyte memory region mapped onto five user configurable memory banks, plus a configuration space used to control the behavior of the EMI.

The configuration address space is organized as shown in Table 102: *EMI configuration register summary on page 264*.

Each EMI_BANK*n* (*n* = 0 to 4) contains a set of four 32-bit registers that are used to configure each bank depending on the type of device that is connected.

The type and organization of each set of bank registers depends on the value in DEVICETYPE (EMI_CFG_DATA0) which defines the type of memory or device attached to that bank.

The EMI supports nonmultiplexed address and data bus.

Each memory type and the associated control registers are described later in this chapter.

31.5 EMI operation

The EMI is a highly flexible memory device which is able to support a large range of memory components gluelessly. It accepts memory operations from the system and, depending on the address of the operation, either accesses its internal configuration space or one of the possible five external memory banks.

The position, size, clock frequency and memory type supported is dependent on how the associated control registers, EMI_BANKS[0:4], are programmed.

Following reset, all banks start with the same configuration which allows the system to boot from a large range of nonvolatile memory devices.

As part of the boot process, the user should program the EMI configuration registers to match the memory supported in that system, defining the memory size, the location in the address and the device type connected.

31.5.1 Bank programming

Refer to Section 31.8: *Chip select allocation/bank configuration on page 260* for full bank programming details.

31.5.2 Clock reconfiguration for synchronous interfaces

Following reset, the clocks for synchronous interfaces are disabled. This is due to the default reset assuming a memory which may be accessed asynchronously.

To access the synchronous memory, the user sets up the configuration state associated with that bank. The user then programs the required clock ratio in the register EMI_*xxx*CLKSEL associated with that memory type.

The external clocks and associated clock dividers are then enabled by a write of 1 to register EMI_CLK_EN. Once enabled, any attempt to reprogram the clock ratios may have undefined effects.

31.6 Default/reset configuration

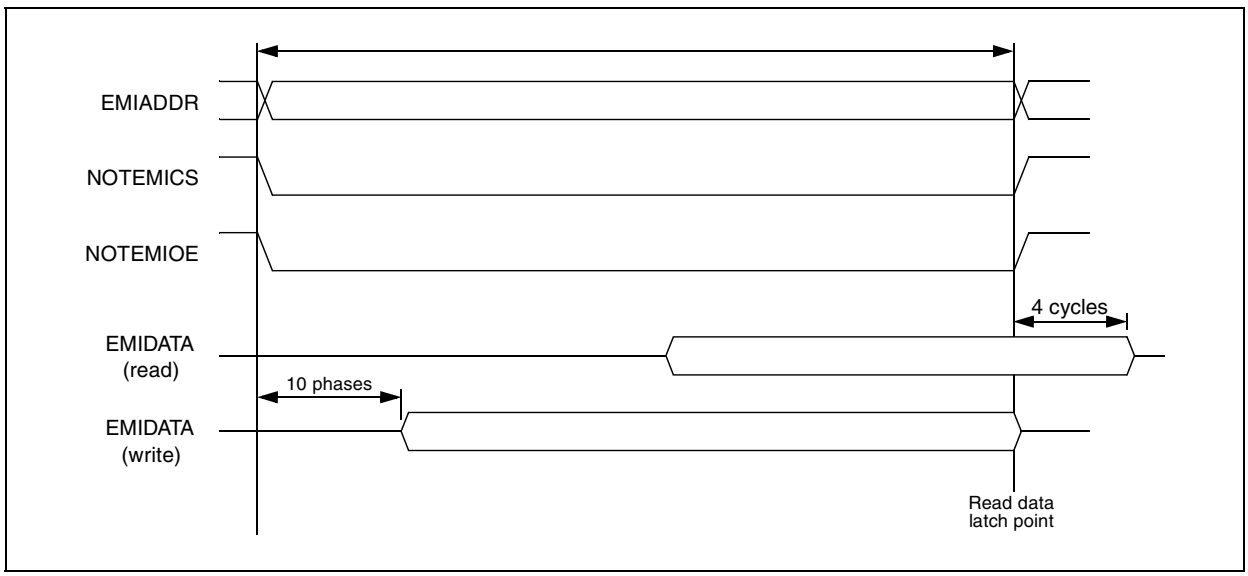
Following reset, a default configuration setting is loaded into all five banks. This allows the EMI to access data from a slow ROM or FLASH memory. The default settings are detailed in Table 98.

Table 98: Default configuration for asynchronous boot

Parameter	Default value
DATADRIVEDELAY	10 phases
BUSRELEASETIME	4 cycles
CSACTIVE	Active during read only
OEACTIVE	Active during read only
BEACTIVE	Inactive
PORTSIZE	Value of the signal EMI4_PRTSZ_INIT
DEVICETYPE	Peripheral
ACCESSTIMEREAD	(18 + 2 = 20 cycles)
CSE1TIMEREAD	0 phases
CSE2TIMEREAD	0 phases
OEE1TIMEREAD	0 phases
OEE2TIMEREAD	0 phases
LATCHPOINT	End of access cycle
WAITPOLARITY	Active high
CYCLEnotPHASE	Phase
BE1TIMEREAD	3 phases
BE2TIMEREAD	3 phases

The remaining configuration parameters are not relevant for an asynchronous boot; that is the aim of the default configuration.

Figure 66: Default asynchronous configuration



31.7 Peripheral interface (with synchronous Flash memory support)

31.7.1 Overview

A generic peripheral (for example SRAM, EPROM, SFlash) access is provided which is suitable for direct interfacing to a wide variety of SRAM, ROM, Flash, SFlash and other peripheral devices.

Note: Refer to Section 31.8 on page 260 *for specific STx7100 settings.*

Figure 67 shows a generic access cycle and the allowable values for each timing field.

Figure 67: Generic access cycle

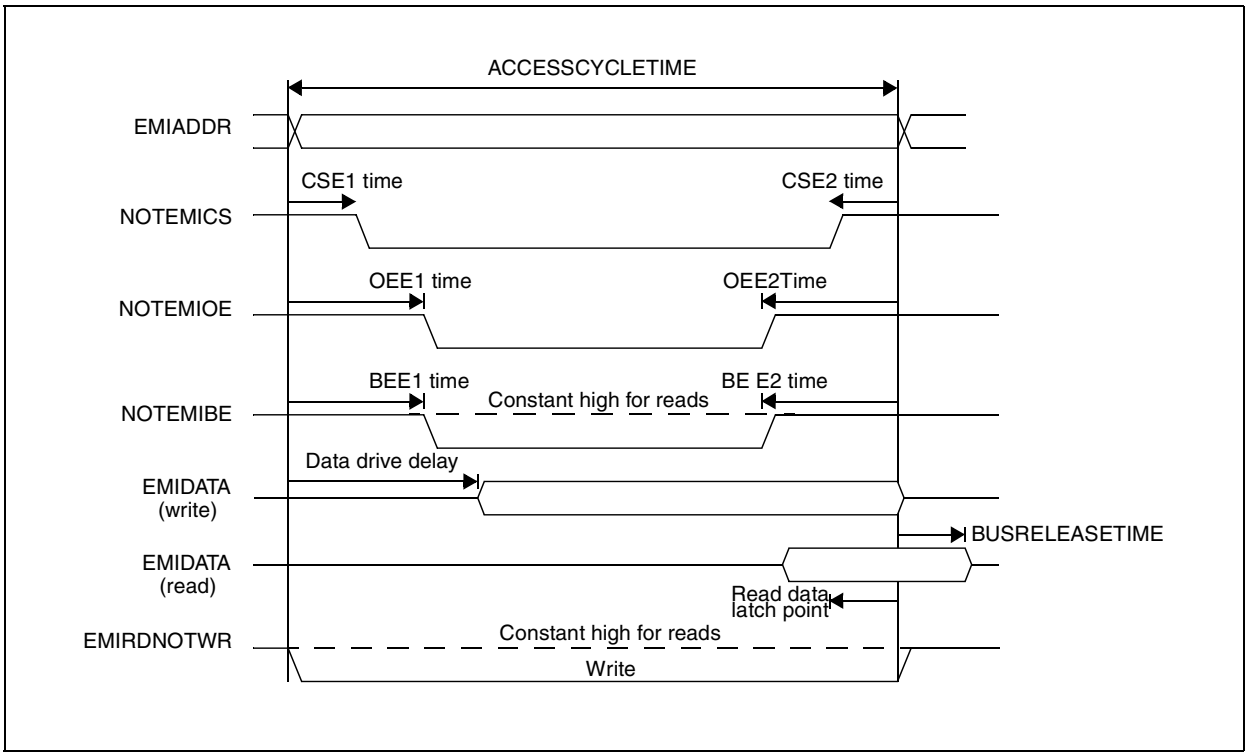


Table 99: Strobe timing parameters for peripheral

Name	Programmable value
ACCESSTIME	2 cycles + 0 to125 cycles
BUSRELEASETIME	0 to15 cycles
DATADRIVEDELAY	0 to 31 phases after start of access cycle
CSE1TIME	Falling edge of CS. 0 to15 phases or cycles after start of access cycle
CSE2TIME	Rising edge of CS. 0 to15 phases or cycles before end of access cycle
OEE1TIME	Falling edge of OE. 0 to15 phases or cycles after start of access cycle
OEE2TIME	Rising edge of OE. 0 to15 phases or cycles before end of access cycle.
BEE1TIME	Falling edge of BE. 0 to15 phases or cycles after start of access cycle
BEE2TIME	Rising edge of BE. 0 to15 phases or cycles before end of access cycle
LATCHPOINT	0: End of access cycle. 1 to 16: 1 to 16 cycles before end of access cycle.

Separate configuration parameters are available for reads and writes. In addition, each strobe can be configured to be active on read, writes, neither or both.

Table 100: Active code settings

CS/OE/BE active code	Strobe activity
00	Inactive
01	Active during read only
10	Active during write only
11	Active during read and write

31.7.2 Synchronous burst Flash support

Burst mode Flash accesses consist of multiple read accesses which must be made in a sequential order. The EMI maps system memory operations onto one or more burst Flash accesses depending on the burst size configuration, operation size and the starting address of the memory access.

The EMI supports the following memory devices:

- AMD AM29BL162C,
- ST M58LW064A/B,
- Intel 28F800F3/ 28F160F3,
- and any new part in these families with identical access protocol.

Table 101 provides a brief description and comparison of EMI-supported Flash memories.

Note: Not all memory features are supported. When a feature is not supported, this is highlighted.

The EMI implements a superset of operational modes so that it is compatible with most of the main functions listed for the three Flash families. The following sections contain a brief description of the EMI Flash interface functionality.

Table 101: ST/AMD/Intel Flash features comparison

	AM29BL162C	STM58LW064A/B	Intel 28F800F3/28F160F3
Size	16 Mbits	64 Mbits	8/16 Mbits
Max[a] operating frequency	40 MHz	60 MHz	60 MHz
Data bus	16 bits fixed	16/32 bits	16 bits fixed
Main operations	Async single access write Sync burst read Async single access read	Async single access write Sync burst read Async single access read Async page read Not supported by EMI	Async single access write Sync burst read Async single access read Async page read Not supported by EMI Sync single access read Not supported by EMI
Burst size	32 word Not supported by EMI: max burst is 8 words	1-2-4-8 words[b] or continuous. Set by burst configuration register. Continuous not supported by EMI	4 to 8 words or continuous Set by read configuration register Continuous is not supported by EMI
Burst style[c]	Linear burst -32 words	Sequential burst Interleaved burst (Not supported)	Linear burst Intel burst (Not supported)
X-latency[d]	70-90-120 ns	7-8-9-10-12[e] cycles	2-3-4-5-6 cycles
Y-latency[f]	1 cycle	1-2 cycles	1 cycle
Burst suspend/ resume[g]	Yes via burst address advance (NOTEMIBAA) input	Yes via burst address advance (NOTEMIBAA) input	No automatic advance
Ready/busy pin[h]	Yes (RD/BY)	Yes (RD/BY)	No
Ready for burst[i]	No	Yes (R)	Yes (W)

a. The Flash operating frequency, clock divide ratios and system frequency should be consistent with the maximum operating frequency.

b. A burst length of eight words is not available in the x32 data bus configuration.

c. Modulo burst is equivalent to linear burst and sequential burst. Interleaved burst is equivalent to Intel burst. On AMD the burst is enabled by four async write operations. On ST and Intel the burst is enabled synchronously via the burst configuration register.

d. X latency is the time elapsed from the beginning of the accesses (address put on the bus) to the first valid data that is output during a burst. For ST, it is the time elapsed from the sample valid of starting address to the data being output from memory for Intel and AMD.

e. 10 to 12 only for F = 50 MHz.

f. Y-latency is the time elapsed from the current valid data that is output to the next data valid in output during a burst.

g. In AMD and ST devices, BAA (or B) can be tied active. This means that the address advance during a burst is noninterruptable (Intel likewise). EMI assumes these pins are tied active and does not generate a BAA signal.

h. When the pin is low, the device is busy with a program/erase operation. When high, the device is ready for any read, write operation.

i. These signals are used to introduce wait states. For example, in the continuous burst mode the memory may incur an output delay when the starting address is not aligned to a four word boundary. In this case a wait is asserted to cope with the delay.

31.7.3 Operating modes

Two different programmable read modes are supported:

- asynchronous single read,
- synchronous burst mode (default four words length: configurable to 1, 2, 4 and 8 words) using a specific lower frequency clock selected using register EMI_FLASH_CLK_SEL.

Note: 1 Continuous burst is not supported by the EMI.

2 32 words burst size is partially supported by the EMI; the burst is interrupted when the required data has been read.

3 Asynchronous page mode read is not supported by the EMI.

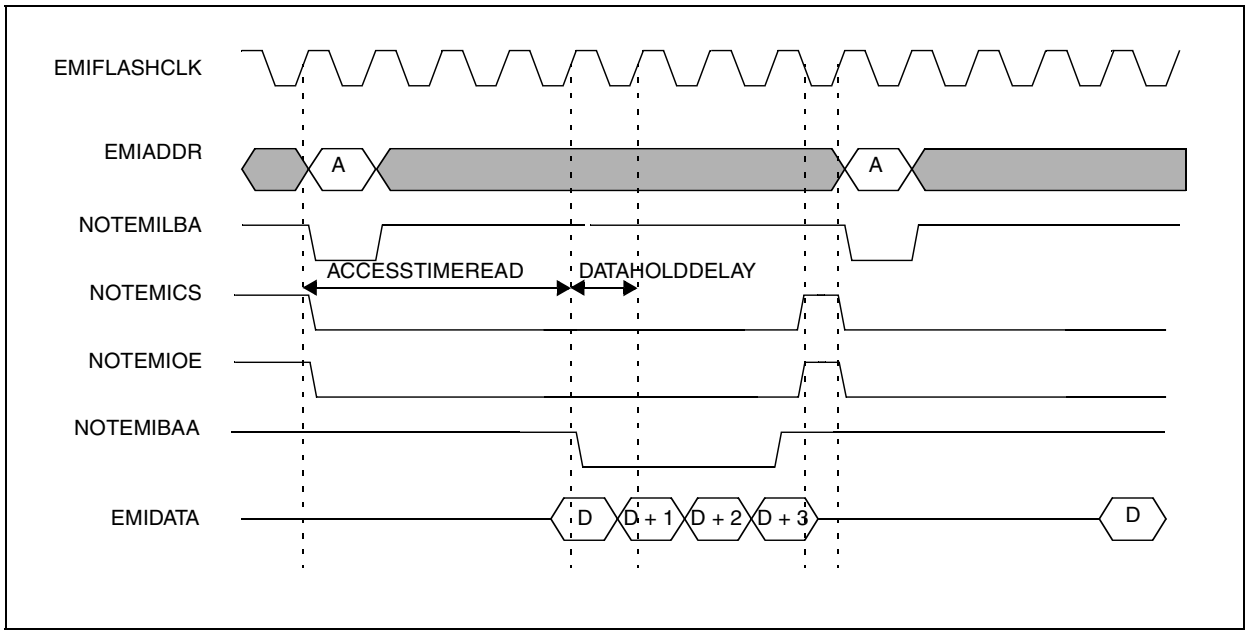
4 Interleaved burst mode is not supported by the EMI due to the implementation of multiple reads only using synchronous burst mode (feature provided by all three families of Flash chips adopted).

The EMI supports a asynchronous single write.

The asynchronous single read/write uses the same protocol as that of the normal peripheral interface.

Figure 68 shows a typical burst access with burst length of four words.

Figure 68: Synchronous burst mode Flash read (burst length = 4)



The ACCESSTIMEREAD parameter is used to specify the time taken by the device to process the burst request. The rate at which subsequent accesses can be made is then specified by the DATAHOLDDELAY parameter, e1 and e2 delays can also be specified.

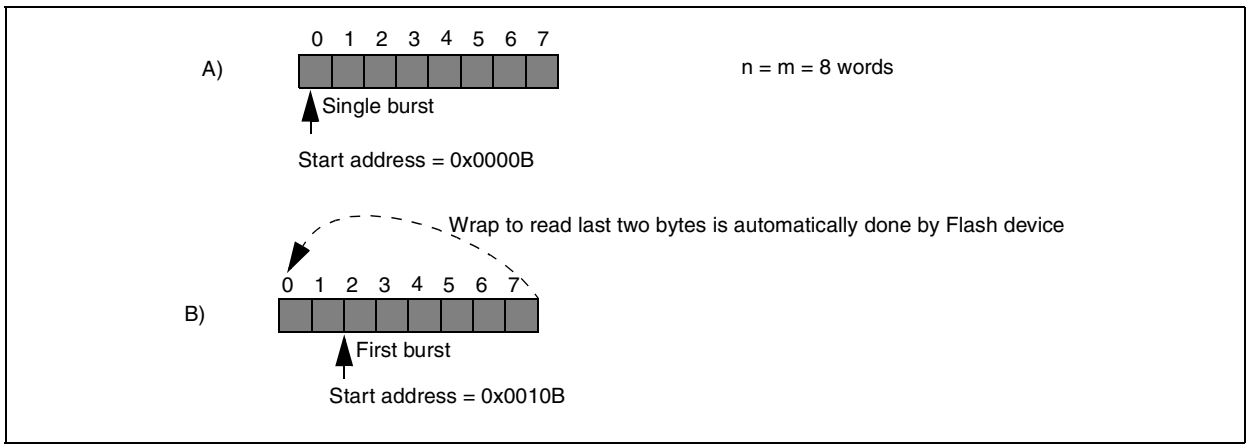
31.7.4 Burst interrupt and burst reiteration

The EMI interrupts the burst after the required amount of data has been read, thus making the chip select of the burst device inactive. This operation is allowed by all three families of Flash devices (burst read interrupt for an ST device, standby for Intel, terminate current burst read for AMD). Due to this operation, the Flash device puts its outputs in tri-state. If a new burst operation is then required, a new chip select and load burst address is provided (NOTEMILBA) to the memory chip.

If the Flash interface is configured to a burst sequence of n bytes, and a burst read request of m bytes is presented to the EMI on the STBus interface, there are three possible outcomes.

- ***n = m***
  The EMI performs one burst access during which it gets the exact number of words as requested (see example A on Figure 69 with n = m = 8). Depending on the starting address, there is possibly a wrap that is automatically completed by the Flash device. The wrap occurs when the starting address is not aligned on an n-byte word boundary.

Figure 69: Burst on a Flash with a single access



- ***n > m***
  If the starting address is aligned on an m-byte word boundary, the EMI gets m bytes from a single burst sequence as explained in the previous paragraph. Then the transfer on Flash is interrupted making the chip select inactive. This terminates the burst transfer and puts the memory device in standby mode, waiting for a new request and starting address for a new burst.
  If the starting address is not aligned on an m-byte word boundary, a first burst on the Flash executes until the m-byte word boundary is crossed. The burst on the Flash is interrupted and there follows another burst with a starting address that wraps to an m-byte boundary (directly given by STBus interface) to read the remaining data. After all the required bytes have been read, the burst access on Flash can be interrupted.
- ***n < m***
  The EMI needs to perform more burst accesses until it gets the required m words.
  If the starting address is aligned on an n-byte word boundary, there are a series of Flash burst accesses until the exact number of bytes is met.
  If the starting address is not aligned on an n-byte word boundary, there is a first access on Flash to read data until the n-byte word boundary is met. This access is then interrupted and new series of accesses are started on a new address provided by STBus (that eventually wraps at the m-bytes boundary). This is repeated until the exact number of bytes is reached. This happens in the middle of the last Flash burst that is interrupted in the usual manner.

31.7.5 Synchronous burst enable

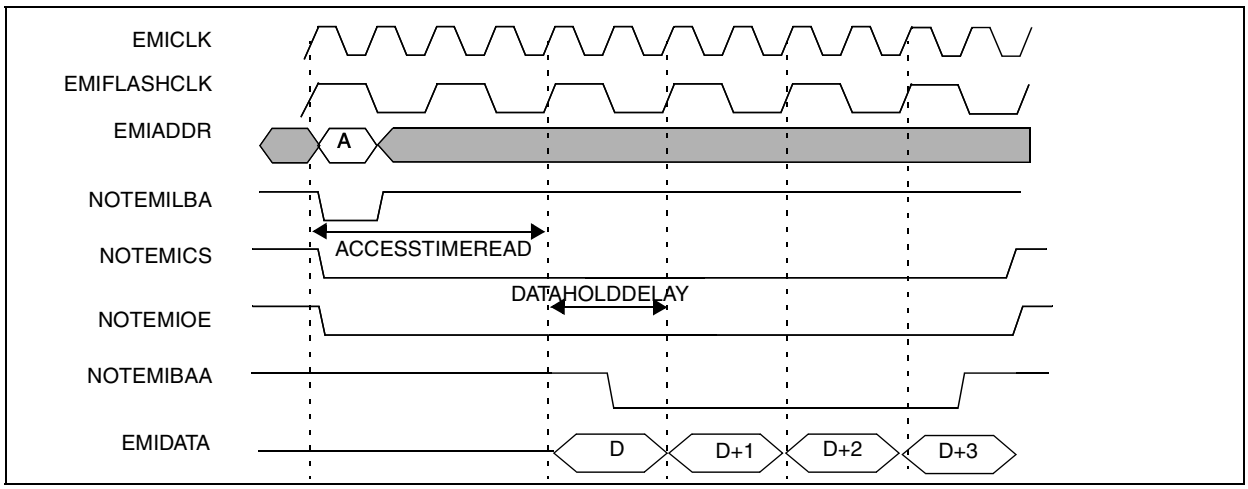
This operation is controlled by software and must only be performed when all other configuration registers in the EMI have been programmed.

Table 101 shows that for ST and Intel devices to operate in synchronous burst mode, the configuration parameters must be set in a special configuration register inside the memory device. The configuration software routine starts two asynchronous write operations for each bank of burst memory, where address and data, respect precise configuration rules. However, for AMD the burst enable is performed by a sequence of four normal asynchronous writes.

31.7.6 Support for lower clock rates

Many SFlash devices operate in the 30 to 50 MHz clock range (Table 101) whereas the EMI operates up to a clock frequency of 100 MHz. To deal with this difference, the EMI can run in a lower speed mode. The hardware in the EMI needed for this mode forces accesses to always start on the rising edge of the slower clock. It is up to the user to configure the other EMI timings, to setup and latch, on the appropriate edge of this slower clock.

Figure 70: Half speed EMI SFlash clock



31.7.7 Initialization sequence

Peripheral interfaces are used immediately after reset to boot the device. Therefore, the default state must be correct for either synchronous or normal ROM. An SFlash device can be interfaced to normal ROM strobes with the addition of only the address valid signal and the clock. When the CPU has run the initial bootstrap, it can configure both the SFlash device and the EMI to make use of the burst features.

Note: The Flash devices are in asynchronous read mode after reset.

Caution

The process of changing from default configuration to synchronous mode is not interruptible. Therefore the CPU must not be reading from the device at the same time as changing the configuration as there is a small window where the EMI's configuration is inconsistent with the memory device's.

31.7.8 Use of Flash memories in different banks with contiguous memory spaces

As shown in Table 101 on page 256, the maximum size of memory chip for SFlash is 64 Mbits. This may not be enough for some of the variants that use the EMI.

It is however possible to place two Flash devices in different banks and have their memory space located contiguously in the overall address space. With this arrangement, the two Flash devices are seen the same way as a single, larger memory device from the software point of view.

This is done through the use of the bank reconfiguration capability, controlled through the EMI Buffer Registers.

31.8 Chip select allocation/bank configuration

Each of the five EMI banks can be configured separately to support different types of devices. There are restrictions on certain banks. The STx7100 provides five chip selects at its outputs, one per bank.

31.9 Address bus extension to low order bits

The STx7100 EMI is able to use either 8- or 16-bit wide memory devices. Selection is done through field PORTSIZE of registers EMI_CFG_DATA0, see Section 32.2.1: *Configuration register formats for peripherals on page 269*. The width of the boot bank is selected in hardware by the logic level to which pin PORTSIZE is tied.

When using a 16-bit device, the low order address bit is on pin EMIADDR[1]. Pins NOTEMIBE[1:0] are byte selectors: bit [0] enables the low order byte and bit [1] enables the high order byte.

When using an 8-bit device, the low order address bit is on pin NOTEMIBE[1] (that is, NOTEMIBE[1] is a virtual EMIADDR[0]). Pins NOTEMIBE[0] acts as byte enable.

31.10 PC card interface

The STx7100 offers a PC Card interface able to support PCMCIA and CableCARD modules (to the ANSI/SCTE-28/2001 standard). This is done through the EMI interface pins, with some of them modified as described further on.

The PC Card signals not available directly from the EMI or anywhere else in the STx7100 application are:

PCCARD_OE#, PCCARD_WE#, PCCARD_IORD#, PCCARD_IOWR#

(asserted at logic low level).

Banks 3 and 4 support PC Card accesses.

Using PCMCIA terminology:

- For common (or attribute memory) read access, PCCARD_OE# gets asserted.
- For I/O read access, PCCARD_IORD# gets asserted.
- For common (or attribute memory) write access, PCCARD_WE# gets asserted.
- For I/O write access, PCCARD_IOWR# gets asserted.

Distinction between memory accesses and IO accesses to the PC Card are based on subdecoding:

- If EMI address line A[15] is 1 then it is a memory access,
- if 0 then it is an I/O access.

Glue logic internal to the STx7100 generates the specific PC Card signals and multiplex them with some regular EMI signals as follows:

NOTEMIOE = PCCARD_OE#
NOTEMILBA = PCCARD_WE#
NOTEMIBAA = PCCARD_IORD#
NOTEMIBE[0] = PCCARD_IOWR#

Enabling of banks 3 and 4 as PC Card banks is based on bits [3] and [4] of EMI General Configuration bits EMI_GENCFG (refer to EMI configuration register list).

When these enables are set, during access to one of these banks (as indicated by the state of EMI_nCS[3] or EMI_nCS[4]) the regular EMI signal is replaced by the PC Card specific signals.

Note: To avoid potential glitch problems, timings of the EMI when accessing a PC Card should be set such that the NOTEMIOE low pulse is completely contained within the NOTEMICSx pulse, with a little margin.

31.11 External bus mastering

It is possible under some conditions to share the EMI bus with an external agent (for example an external host CPU) using pins EMIREQ and EMIGNT. This enables sharing common memory devices (for example a Flash memory) between the STx7100 and an external host.

Arbitration is always performed by the STx7100, which can release the EMI bus by turning off all its EMI I/O drivers upon request for access from an external host. The procedure is as follows:

1. The external host requests access to the EMI bus by asserting STx7100 input EMIREQ.
2. The STx7100 completes any pending access, tri-states its EMI I/Os, and grants access to the bus by asserting output EMIGNT.
3. The external host performs its accesses. When done, it de-asserts signal EMIREQ, the STx7100 *n* turn de-asserts signal EMIGNT and takes back ownership of the EMI bus.

Whilst pin EMIREQ is a pure input to the STx7100, pin EMIGNT is shared with PIO3[7]: the alternate output function of this PIO must be enabled by appropriate programming of the PIO registers to have access to EMIGNT.

Note: It may be possible to operate without the EMIGNT *signal if the external host guarantees a latency from assertion of* EMIBUSREQ *to driving the bus. This latency should be more than the maximum duration of an* STx7100 *EMI access (application dependent).*

During reset, the STx7100 does not drive the EMI bus, so it is also possible to hold off the STx7100 and boot an external host from the EMI bus by holding off the hardware reset supplied to the STx7100.

31.12 EMI buffer

The EMI buffer block defines the size of the five banks mapped in the internal memory map. All banks are contiguous and the mapping is done by programming the top address of each bank, with the base address of Bank 0 fixed at 0x4000 000. The address granularity is 4 Mbytes per bank. The default configuration is 16 Mbytes per bank and all five banks enabled.256MBytes for Banks 0 to 3 and 8 Mbyte for Bank 4.

Note: There is actually a fifth "virtual" bank, Bank 5, whose associated ChipSelect is not brought out of the chip so which is not usable. However, knowledge of its existence helps program correctly the EMI Buffer registers.

32 EMI registers

32.1 Overview

Register addresses are provided as

EMIConfigBaseAddress + offset,
EMIBufferBaseAddress + offset, or
EMIBankBaseAddress + offset.

The *EMIConfigBaseAddress* is:

0x1A10 0000.

The *EMIBufferBaseAddress* is:

0x1A10 0800.

The *EMIBankBaseAddress* is:

EMIConfigBaseAddress + *EMIBank(n)* with *n* = {0,1,2,3,4}

where

EMIBank0 = 0x100,
EMIBank1 = 0x140,
EMIBank2 = 0x180,
EMIBank3 = 0x1C0,
EMIBank4 = 0x200,
EMIBank5 = 0x240 (“virtual” bank: accessible, but no associated external Chip Select signal)

There are a few additional EMI registers described in Chapter 21: *System configuration registers on page 171*.

Table 102: EMI configuration register summary

Register	Description	Offset from *EMIConfigBaseAddress*	Type
EMI_VCR	Reserved for version control EMI register	0x000	RO
EMI_STA_CFG	Status register (configuration flags update)	0x010	RO
EMI_STA_LCK	Lock register (configuration flags lock)	0x018	WO
EMI_LCK	Lock register	0x020	WO
EMI_GEN_CFG	General purpose configuration register set	0x028	R/W
Reserved	-	0x030 to 0x048	-
EMI_FLASH_CLK_SEL	Select clock speed for Flash devices	0x050	WO
Reserved	-	0x058	-
EMI_CLK_EN	Enable clock generation for different devices	0x068	WO
Reserved	-	0x070 to 0x0F8	-
EMI_BANK0	Bank0 configuration register set (EMI_CFG_DATA[0:3] + reserved space)	0x100 to 0x138	R/W
EMI_BANK1	Bank1 configuration register set (EMI_CFG_DATA[0:3] + reserved space)	0x140 to 0x178	R/W
EMI_BANK2	Bank2 configuration register set (EMI_CFG_DATA[0:3] + reserved space)	0x180 to 0x1B8	R/W
EMI_BANK3	Bank3 configuration register set (EMI_CFG_DATA[0:3] + reserved space)	0x1C0 to 0x1F8	R/W
EMI_BANK4	Bank4 configuration register set (EMI_CFG_DATA[0:3] + reserved space)	0x200 to 0x238	R/W
Reserved	-	0x240 to 0xFFF8	-

The configuration region of each bank is divided as shown in Table 103.

Table 103: EMI_BANK*n* register organization

Register	Offset from *EMIBankBaseAddress*
EMI_CFG_DATA0	0x00
EMI_CFG_DATA1	0x08
EMI_CFG_DATA2	0x10
EMI_CFG_DATA3	0x18

The EMI buffer is characterized by six internal registers:

- five related to the accessible external memory banks (each composed of six bits),
- one related to the value of the total number of banks registers enabled at the same time (composed of three bits).

Table 104: EMI buffer register summary

Register	Description	Offset from *EMIBufferBaseAddress*	Type
EMIB_BANK0_TOP_ADDR	External memory bank 0 address bits [27:22]	0x0000	R/W
EMIB_BANK1_TOP_ADDR	External memory bank 1 address bits [27:22]	0x0010	R/W
EMIB_BANK2_TOP_ADDR	External memory bank 2 address bits [27:22]	0x0020	R/W
EMIB_BANK3_TOP_ADDR	External memory bank 3 address bits [27:22]	0x0030	R/W
EMIB_BANK4_TOP_ADDR	External memory bank 4 address bits [27:22]	0x0040	R/W
EMIB_BANK5_TOP_ADDR	“Virtual” memory bank 5 address bits [27:22]	0x0050	R/W
EMIB_BANK_EN	Total number of enabled banks	0x0060	R/W

32.2 EMI register descriptions

EMI_STA_CFG EMI status configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											CFG_UPDATED			

Address: *EMIConfigBaseAddress* + 0x0010

Type: Read

Reset: Undefined

Description: If bit *n* is set, then all configuration registers associated with bank *n* have been written to at least once.

EMI_STA_LCK EMI status configuration lock

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											CFG_LCK			

Address: *EMIConfigBaseAddress* + 0x0018

Type: Read

Reset: Undefined

Description: If bit *n* is set, then all configuration registers associated with bank *n* are locked and further write accesses are ignored.

EMI_LCK EMI lock

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PROTECT			

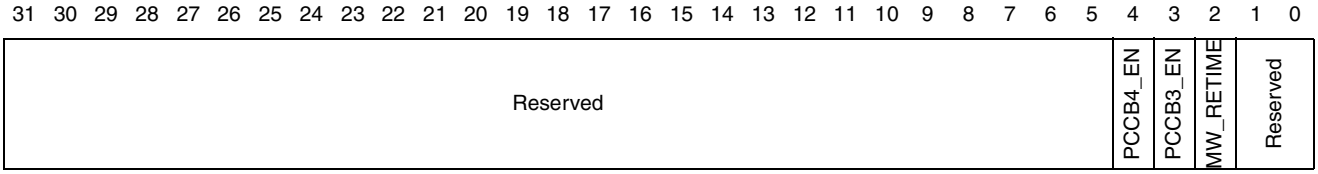
Address: *EMIConfigBaseAddress* + 0x0020

Type: R/W

Reset: Undefined

Description: If bit *n* is set, then the registers EMI_BANK*n*.EMI_CFG_DATA[0:3] may only be read. Subsequent writes to these registers are ignored.

EMI_GEN_CFG EMI general purpose



Address: *EMIConfigBaseAddress* + 0x0028
Type: R/W
Reset: 0x00
Description: Used to propagate general purpose outputs.

[31:5] **Reserved**

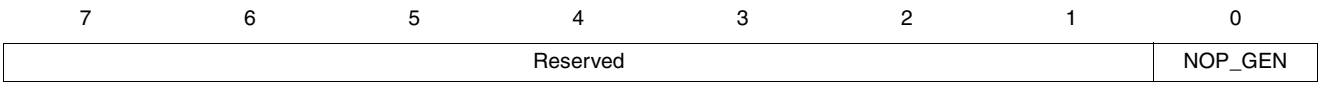
[4] **PCCB4_EN**: Enable PC card bank 4
When set, during access to Bank4, some regular EMI signals (NOTEMIOE, NOTEMILBA, NOTEMIBAA, NOTEMIBE[0]) are replaced by PC Card-specific signals (PCMCIA_OE#, PCMCIA_WE#, PCMCIA_IORD#, PCMCIA_IOWR#)

[3] **PCCB3_EN**: Enable PC card bank 3
When set, during access to Bank3, some regular EMI signals (NOTEMIOE, NOTEMILBA, NOTEMIBAA, NOTEMIBE[0]) are replaced by PC Card-specific signals (PCMCIA_OE#, PCMCIA_WE#, PCMCIA_IORD#, PCMCIA_IOWR#)

[2] **EWAIT_RETIME**: Determine the number of retime stages for signal EMITREADYORWAIT
0: 2
1: 1
The signal can be dynamically changed between a single retime input stage or a double retime input stage. Both retime stages on the retimed input are synchronized to the current clock.
If EMITREADYORWAIT is set at the beginning of the access:
ACCESSTIMEREAD > LATCHPOINT + (2 + EWAIT_RETIME).

[1:0] **Reserved**

EMI_SDRAM_NOP_GEN EMI NOP generate



Address: *EMIConfigBaseAddress* + 0x0030
Type: WO
Reset: Undefined
Description: Reserved

EMI_FLASH_CLK_SEL EMI Flash burst clock select

7	6	5	4	3	2	1	0
Reserved						FLASH_CLK_SEL	

Address: *EMIConfigBaseAddress* + 0x0050
Type: WO
Reset: Undefined
Description:

[7:2] **Reserved**

[1:0] **FLASH_CLK_SEL**: Set clock ratio for burst Flash clock.
00: 1:1 Flash operates at EMICLK
01: 1:2 Flash operates at 1/2 of EMICLK
10: 1:3 Flash operates at 1/3 of EMICLK
11: Reserved

EMI_CLK_EN EMI clock enable

7	6	5	4	3	2	1	0
Reserved							CLK_EN

Address: *EMIConfigBaseAddress* + 0x0068
Type: WO
Reset: 0x00
Description: A write of 1 to this bit causes the Flash clocks to be updated.
This operation can only occur once, further writes to this register may lead to undefined behavior.

32.2.1 Configuration register formats for peripherals

The following is a summary of the configuration register formats for peripherals.

EMI_CFG_DATA0 EMI configuration data 0

31 30 29 28 27	26	25	24 23 22 21 20	19 18 17 16 15	14 13 12 11	10 9	8 7	6 5	4 3	2 1 0
Reserved	WE_USE_OE_CFG	WAITPOLARITY	LATCHPOINT	DATADRIVEDELAY	BUSRELEASETIME	CSACTIVE	OEACTIVE	BEACTIVE	PORTSIZE	DEVICETYPE

Address: *EMIBankBaseAddress* + 0x00
Type: R/W
Reset: 0x00
Description:

[31:27] **Reserved**

[26] **WE_USE_OE_CFG**: This bit must be set to one in case the SFlash bank (such as STM58LW064A/B) requires a configurable EMIRDNOTWR signal for async write operation.
When this bit is set to one the WE becomes low following the same timing defined for OEE1TIMEWRITE and OEE2TIMEWRITE
Otherwise (bit set to 0) the EMIRDNOTWR becomes low at the start of the access and is deactivated at the end of the access

[25] **WAITPOLARITY**: Set the wait signal polarity:
0: Wait active high 1: Wait active low

[24:20] **LATCHPOINT**: Number of EMI subsystem clock cycles before end of access cycle.
0 0000: End of access cycle 0 0001: 1 cycle
0 0010: 2 cycles 0 0011: 3 cycles
0 0100: 4 cycles 0 0101: 5 cycles
0 0110: 6 cycles 0 0111: 7 cycles
0 1000: 8 cycles 0 1001: 9 cycles
0 1010: 10 cycles 0 1011: 11 cycles
0 1100: 12 cycles 0 1101: 13 cycles
0 1110: 14 cycles 0 1111: 15 cycles
1 0000: 16 cycles Other: Reserved

[19:15] **DATADRIVEDELAY**: 0 to 31 phases

[14:11] **BUSRELEASETIME**: 0 to15 cycles

[10:9] **CSACTIVE**: See Table 100 on page 255.

[8:7] **OEACTIVE**: See Table 100 on page 255.

[6:5] **BEACTIVE**: See Table 100 on page 255.

[4:3] **PORTSIZE**
00: Reserved 01: Reserved
10: 16-bit 11: 8-bit

[2:0] **DEVICETYPE**
001: Normal peripheral or 100: Burst Flash

EMI_CFG_DATA1 EMI configuration data 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCLE NOT PHRD	ACCESSTIMEREAD							CSE1TIMEREAD				CSE2TIMEREAD			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEE1TIMEREAD				OEE2TIMEREAD				BEE1TIMEREAD				BEE2TIMEREAD			

Address: *EMIBankBaseAddress* + 0x08

Type: RO

Reset: 0x00

Description:

[31] **CYCLENOTPHRD**: Change measure unit for e1/e2 time accesses from phases to cycles:
0: The e1(e2) write times for CS, BE and OE are expressed in system clock phases.
1: Write times are expressed in cycles.

[30:24] **ACCESSTIMEREAD**: 2 to 127 EMI subsystem clock cycles: value 0 and 1 are reserved.

[23:20] **CSE1TIMEREAD**: Falling edge of CS. 0 to 15 phases/cycles after start of access cycle.

[19:16] **CSE2TIMEREAD**: Rising edge of CS. 0 to 15 phases/cycles before end of access cycle.

[15:12] **OEE1TIMEREAD**: Falling edge of OE. 0 to 15 phases/cycles after start of access cycle.

[11:8] **OEE2TIMEREAD**: Rising edge of OE. 0 to 15 phases/cycles before end of access cycle.

[7:4] **BEE1TIMEREAD**: Falling edge of BE. 0 to 15 phases/cycles after start of access cycle.

[3:0] **BEE2TIMEREAD**: Rising edge of BE. 0 to 15 phases/cycles before end of access cycle.

EMI_CFG_DATA2 EMI configuration data 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCLE_NOT_PHWR	ACCESS_TIME_WRITE							CSE1_TIME_WRITE				CSE2_TIME_WRITE			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEE1_TIME_WRITE				OEE2_TIME_WRITE				BEE1_TIME_WRITE				BEE2_TIME_WRITE			

Address: *EMIBankBaseAddress* + 0x10

Type: R/W

Reset: 0x00

Description:

[31] **CYCLE_NOT_PHWR**: Change measure unit for e1/e2 time accesses from phases to cycles:
0: The e1(e2) write times for CS, BE, OE are expressed in system clock phases.
1: Write times are expressed in cycles.

[30:24] **ACCESS_TIME_WRITE**: 2 to 127 cycles: value 0 and 1 are reserved

[23:20] **CSE1_TIME_WRITE**: Falling edge of CS. 0 to 15 phases/cycles after start of access cycle

[19:16] **CSE2_TIME_WRITE**: Rising edge of CS. 0 to 15 phases/cycles before end of access cycle

[15:12] **OEE1_TIME_WRITE** (WEE1TIMEWRITE): Falling edge of OE. 0 to 15 phases/cycles after start of access cycle. Also used for falling edge of WE if bit WE_USE_OE_CFG (reg0, bit 26) is set to one.

[11:8] **OEE2_TIME_WRITE** (WEE2TIMEWRITE): Rising edge of OE. 0 to 15 phases/cycles before end of access cycle. Also used for rising edge of WE if bit WE_USE_OE_CFG (reg0, bit 26) is set to 1.

[7:4] **BEE1_TIME_WRITE**: Rising edge of BE. 0 to 15 phases/cycles after start of access cycle

[3:0] **BEE2_TIME_WRITE**: Falling edge of BE. 0 to 15 phases/cycles before end of access cycle

EMI_CFG_DATA3 EMI configuration data 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					STROBEONFALLING	Reserved																BURST_SIZE			DATALATENCY					DATAHOLDDELAY	BURSTMODE

Address: *EMIBankBaseAddress* + 0x18

Type: R/W

Reset: 0x00

Description: Configuration of EMI_CFG_DATA0, EMI_CFG_DATA1, EMI_CFG_DATA2 relates only to the asynchronous behavior (normal peripheral and normal asynchronous behavior of Flash). These registers must be programmed in terms of EMICLK clock cycle.

EMI_CFG_DATA3 must be configured only if there is burst Flash and refers to the synchronous behavior of Flash. It does not need to be configured for a normal asynchronous peripheral. The parameters in this register must be programmed in terms of Flash clock cycles.

[31:27] **Reserved**: must be set to 0.

[26] **STROBEONFALLING**: Flash clock edge for burst strobe generation.
0: rising edge 1: falling edge
The strobe on falling feature of EMI means only that strobes, data and address are generated on the falling edge of the SFlash clock. This does not imply that the same signals are sampled on the falling edge by the memories. The EMI assumes memory always samples on the rising edge. The strobes on falling feature has been implemented only to extend the hold time of half a cycle to help padlogic implementation.

[25:10] **Reserved**: must be set to 0.

[9:7] **BURST_SIZE**: The number of bytes which map onto the device's burst mode (only valid in burst mode).
000: 2 001: 4
010: 8 011: 16
100: 32 101: 64
110: 128 111: Reserved
The 64/128 byte burst mode is due to the possible usage of the AMD device that has a fixed 32-word burst length. STBus interface max transfer is 32 bytes on EMI, so in these cases the burst on Flash is always interrupted.

[6:2] **DATALATENCY**: The number of SFlash clock cycles between the address valid and the first data valid.
00010: 2 cycles 00011: 3 cycles
00100: 4 cycles ...
01001: 17 cycles Others: Reserved

[1] **DATAHOLDDELAY**: Extra delay when accessing same bank consecutively when in cycles between words in burst mode.
0: one Flash clock cycle 1: two Flash clock cycles

[0] **BURSTMODE**: Select synchronous Flash burst mode. If this bit is set, only ACCESSTIMEREAD and DATAHOLDDELAY are relevant for strobe generation timing during read operations

32.3 EMI buffer register descriptions

All the registers described in this section are nonvolatile

EMIB_BANK0_TOP_ADDR External memory bank 0 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANK0_TOP_ADD							

Address: *EMIBufferBaseAddress* + 0x000

Type: R/W

Reset: 0x3F

Description: Contains bits [29:22] of the top address of external memory bank x (the complete 32 bit top address being obtained by setting LSBs to 1 and MSBs to 0). The BASE address of memory bank x is the location that follows the top address of bank x-1, which means banks are all contiguous (for Bank 0 the base address is always 0x4000 0000).

EMIB_BANK1_TOP_ADDR External memory bank 1 top address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANK1_TOP_ADD							

Address: *EMIBufferBaseAddress* + 0x010

Type: R/W

Reset: 0x7F

Description: Contains the top address bits [27:22] of external memory bank 1. Accesses to this address space cause transfer on EMI bank 1.

EMIB_BANK2_TOP_ADDR External memory bank 2 top address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANK2_TOP_ADD							

Address: *EMIBufferBaseAddress* + 0x020

Type: R/W

Reset: 0xBF

Description: Contains the top address bits [27:22] of external memory bank 2. Accesses to this address space cause transfer on EMI bank 2.

EMIB_BANK3_TOP_ADDR External memory bank 3 top address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANK3_TOP_ADD							

Address: *EMIBufferBaseAddress* + 0x030

Type: R/W

Reset: 0xFB

Description: Contains the top address bits [27:22] of external memory bank 3. Accesses to this address space cause transfer on EMI bank 3.

EMIB_BANK4_TOP_ADDR External memory bank 4 top address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANK4_TOP_ADD							

Address: *EMIBufferBaseAddress* + 0x040

Type: R/W

Reset: 0xFD

Description: Contains the top address bits [27:22] of external memory bank 4. Accesses to this address space cause transfer on EMI bank 4.

EMIB_BANK5_TOP_ADDR Virtual memory bank 5 top address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANK5_TOP_ADD							

Address: *EMIBufferBaseAddress* + 0x050

Type: R/W

Reset: 0xFF

Description: This is a “virtual bank”; its associated ChipSelect signal is not brought out of the chip so cannot be used. Be aware of this when programming register EMIB_BANK_EN.

EMIB_BANK_EN Enabled bank registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BANKS_EN							

Address: *EMIBufferBaseAddress* + 0x060

Type: R/W

Reset: 0x110

Description: Contains the total number of bank registers enabled. When the number of banks is reduced by register EMIB_BANK_EN, the last bank (that is, the top bank) takes its own area plus the remaining area of the banks disabled. For example, if only five banks are enabled, BANK5 is disabled, then BANK4 region contains its own area plus the BANK5 area. At reset all the banks are enabled.

[31:8] **Reserved**

[7:0] **BANKS_EN**: Banks enabled

001: Bank 0 only enabled
010: Banks 0 to 1 enabled
011: Banks 0 to 2 enabled
100: Banks 0 to 3 enabled
101: All banks with associated ChipSelect enabled (0 to 4)
110: All banks enabled - including “shadow” bank 5 which has no associated ChipSelect.

33 DMA network

33.1 DMA requests and channels

The STx7100 integrates a multiple channel general purpose (flexible) DMA engine. Its purpose is to move data efficiently from memory-to-memory, memory-to-peripheral and peripheral-to-peripheral.

The FDMA includes stream parsing functionality which facilitates video PES header and video start-code detection (PES/SCD) for MPEG2 and H.264.

The FDMA supports up to 16 active DMA channels, accepts up to 30 DMA requests or “events” and supports real-time traffic between main memory and internal real-time sources/targets (audio peripherals, SWTS, Comms) and external real-time sources/targets.

Requests 0 and 31 are reserved for the counter request. They are internally connected to the FDMA counter to provide timed DMA channels. Request 0 has lowest priority and request 31 has highest priority

This chapter describes how the FDMA is integrated with the rest of the STx7100. For detailed information on the FDMA itself, see next Chapter 34: *Flexible DMA (FDMA)*.

Table 105: FDMA request summary

Channel ID	Request source	Request signal	Request rate	Description
0	FDMA counter	O_COUNTER_REQ	-	FDMA free-running internal counter output, used for timed DMA-channel - memory-memory transfers.
1 - 2	Reserved	GND	-	-
3	Video - HDMI	HDMI_AVI_BUFF_EMPTY	-	HDMI request
4	Comms/DiSEqC	DISECQ_TX_HALF_EMPTY	-	DiSEqC requests
5		DISECQ_RX_HALF_FULL	-	
6	ST40/SCIF	SCIF_RXBUF_READY	-	ST40 SCIF
7		SCIF_TXBUF_READY	-	
8 - 10	Comms/SSCs (x3)	SSC*n*_RXBUF_FULL	-	SSCs requests (*n* = 0, 1, 2)
11 - 13		SSC*n*_TXBUF_EMPTY	-	
14 - 17	Comms/UARTs (x4)	UART*n*_RX_BUF_HALF_FULL	-	SSCs requests (*n* = 0, 1, 2, 3)
18 - 21		UART*n*_TX_BUF_HALF_EMPTY	-	
22	External DMA 0	DREQ0 (PIO2 bit 5)	-	External FDMA requests through PIOs alternate function.
23	External DMA 1	DREQ1 (PIO2 bit 6)	-	
24	CSS/CPxM decryption	CPXMOUT_FDMA_REQ	-	CPxM decrypted data request
25		CPXMIN_FDMA_REQ	-	CPxM encrypted data request
26	Audio PCM player 0	PCMPLYR0_FDMA_REQ	48 kHz (2 x 32 bits)	Mixed PCM player request
27	Audio PCM player 1	PCMPLYR1_FDMA_REQ	48 kHz (2 x 32bits)	Unmixed PCM player request
28	Audio PCM reader	PCMRDR_FDMA_REQ	48 kHz (2 x 32bits)	PCM reader request
29	Audio SPDIF player	SPDIFPLYR_FDMA_REQ	48 kHz (2 x 32bits)	SPDIF player request
30	TS merger	SWTS_REQ	-	Software transport stream play through PTI
31	FDMA counter	O_COUNTER_REQ	-	FDMA internal counter output, used for timed DMA-channel (PES mode)

34 Flexible DMA (FDMA)

The flexible DMA (FDMA) is a general-purpose direct memory access controller capable of supporting 16 independent DMA channels. It is used to perform block moves thus reducing the load on the CPU. Moves may be from memory-to-memory or between memory and paced latency-critical real-time targets.

This chapter describes how the FDMA works, and how to program it. Chapter 33: *DMA network on page 275* explains how the FDMA is integrated within the STx7100.

The FDMA supports the following features:

- 16 independent DMA channels,
- transfer of information to or from aligned or unaligned data structures of up to 4 Gbytes in the following organizations:
  - single location (0D),
  - incrementing linear arrays (1D),
  - incrementing rectangular arrays (2D),
- transfer units of 1 to 32 bytes,
- programmable opcodes for paced transfer,
- paced or free running timing models,
- support for up to 30 request generating peripherals,
- a single interrupt which signals completion of:
  - a list of transfers,
  - each transfer of a node in a list,
- little endian data organization,
- linked list control which allows,
  - a set of DMA transfers to be sequenced,
  - complex operations such as scatter-gather without CPU intervention,
- special channel configurations for:
  - PES parsing/SCD
    - H.264 and MPEG2 start code detect
    - Dual stream on a single FDMA channel supported
  - Memory-to-memory moves or free running transfers,
  - Paced[1] transfers,
  - S/PDIF output.

1. A channel is paced if an external request causes a single data unit to be transferred per request. The FDMA may require multiple requests to complete the operation.

34.1 Channel structures

The STx7100 FDMA has three types of channels:

- standard 0D, 1D, 2D and paced memory channels,
- an SCD/PES parsing channel,
- an S/PDIF channel.

All channels use linked lists of FDMA operations stored in memory. SCD/PES parsing channels require additional control data which is written to FDMA data memory before the channel is started.

Each linked list is composed of a series of nodes in main memory. Each node is a data structure containing parameters that describe an FDMA transfer.

Typically a linked list of nodes is set up in main memory, the pointer to the first node is written to the FDMA and the channel is started. This bootstraps the list, loading the first entry node and continuing until completion of all the DMAs in the list.

On completion of a node the channel may generate an interrupt indicating completion and/or trigger a channel update.

A channel update causes the next node in the list to be loaded from memory. The location of the node structure in memory is given by a pointer.

This may be used to extend the channel's operation to support features such as scatter gather sequences, or building DMA sequences with only the final completion requiring CPU intervention via an interrupt.

Note: To emulate a ping-pong buffer using a two-node (looped) linked list it is possible to generate an interrupt on completion of one node while moving directly to the next (interrupt but no pause). In other words, the interrupt does not stall the channel.

34.1.1 DMA transfer units

The FDMA uses the most efficient opcode for the requested transaction. The supported unit size for the paced transfer is programmable at 1 x 4, 2 x 2, 4, 8, 16 and 32 bytes. Byte enables are used for non–word-aligned cases.

Paced transfers must be aligned to their opcode size.

34.1.2 Alignment

The node structures must be aligned to a 32 byte boundary.

The most efficient data transfers are aligned to a 128 byte boundary and the number of bytes transferred should be a multiple of 32.

For paced channels, the paced-side data must be aligned to the paced opcode (OP32 must be 32-byte aligned, OP16 must be 16-byte aligned and so on). The memory side of a paced transfer must be 32-byte aligned. There are no alignment restrictions for the memory side of S/PDIF transfers.

For SCD/PES parsing, the PES buffer is treated as linear. A circular PES buffer can be described using two linked nodes. The PES buffer is 128-byte aligned. The ES buffer and start codes list buffers are 32-byte aligned.

34.2 FDMA timing model

34.2.1 Free-running

The FDMA is free-running. Once a channel is started, operations occur without requiring a request to begin or control the timing of the transfer. It continues to operate without further intervention until disabled, or the transfer is complete.

This is the model generally used for memory-to-memory moves.

A sleep mechanism can be used to slow down these accesses.

34.2.2 Paced

A channel is paced if a single data unit is transferred to or from a peripheral upon request. The request may be associated with either the source or destination memory location.

The FDMA supports up to 30 physical request signals.

Only the DREQ request protocol is supported. The peripheral uses the STBus request to the data FIFO to determine that a requested transfer has been acknowledged by the FDMA and it clears the DREQ signal (when applicable).

The FDMA implements a hold off mechanism to ensure that it ignores the DREQ for a certain period of time after having serviced a data request.

Figure 71: FDMA request routing

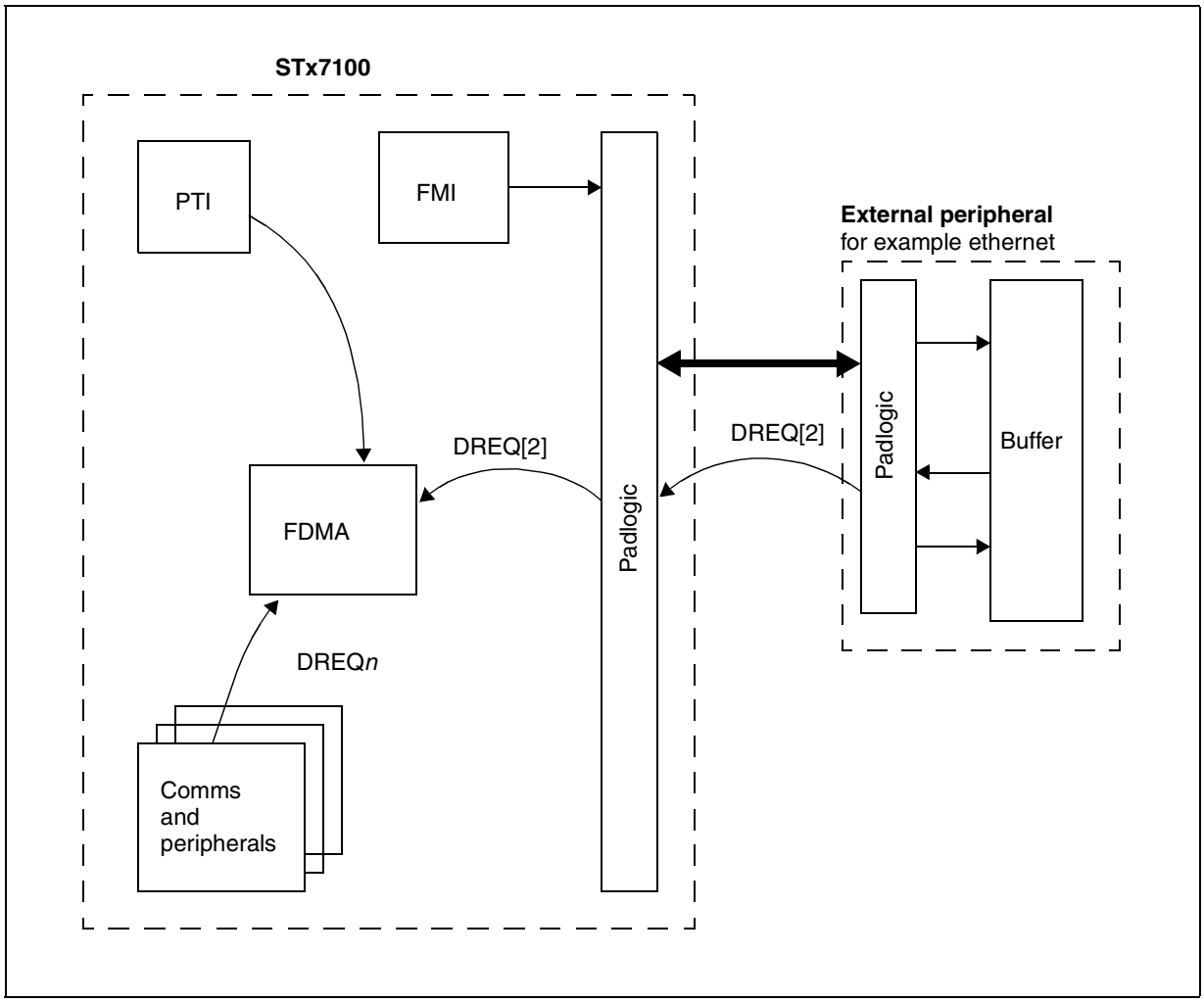


Table 106: FDMA pacing inputs

Block	Block port/signal name	FDMA requests	Description
Reserved		0 - 2	
Video_ HDMI	HDMI_AVIBUFF_EMPTY	3	
COMMS/DiSEqC	DISEQC_RX_HALF_FULL	4	
	DISEQC_RX_HALF_EMPTY	5	
ST40/SCIF	SCIF_RXBUF_READY	6	
	SCIF_TXBUF_READY	7	
Comms/SSC (x3)	SSC*n*_RXBUFF_FULL	8 - 10	
	SSC*n*_TXBUFF_EMPTY	11 - 13	
Comms/UART (x4)	UART*n*_RX_HALF_FULL	14 - 17	
	UART*n*_TX_HALF_EMPTY	18 - 21	
External	DMAREQ[0:1]	22, 23	
Audio	AUD_PCM0	26	Mixed PCM player request
	AUD_PCM1	27	Unmixed PCM player request
	AUD_PCM_READER	28	PCM player reader
	AUD_SPDIF	29	S/PDIF player
TSMerger SWTS	SWTS_REQ	30	TS merger
Reserved		31	

34.2.3 SCD/PES parsing channel

An SCD/PES parsing channel runs until the list is complete or a pause at the end of a node is encountered.

34.2.4 S/PDIF channel

The S/PDIF channel is treated as a standard paced channel.

34.3 Operating the FDMA

Before operation, the FDMA is loaded with low-level binary firmware supplied with the STx7100.

Communication between the CPU and the FDMA is achieved through two 32-bit mailbox registers, plus one command and status register (FDMA_CMD_STA) per channel.

- The command mailbox register sends commands from the CPU to the FDMA.
- The interrupt mailbox sends interrupts from the FDMA to the CPU.

Two bits of each mailbox register are associated with each channel.

- The CPU sets bits in the command mailbox and clears bits in the interrupt mailbox.
- The FDMA clears bits in the command mailbox and sets bits in the interrupt mailbox.

Interrupts and flags are generated for each channel by setting mask bits in the mailbox register. Any masked nonzero bits in the mailbox either raise an interrupt (interrupt mailbox) or generate a flag in the FDMA (command mailbox).

There are three commands for each channel, set in the command mailbox:

- START: start and initialize channel *n* (no initialize if the channel is restarting after a PAUSE),
- PAUSE: pause channel *n*,
- FLUSH: flush and pause channel *n*.

34.3.1 Channel arbitration

Types of transfer are not associated with channel numbers, but are specified in the node's control word (see field REQ_MAP). Each of the 16 independent channels competes for the service of the FDMA.

- The paced channels have the highest priority.
- SCD/PES parsing have medium priority.
- Memory-to-memory moves use the remaining bandwidth and are arbitrated using a round-robin scheme.

When more than one paced channel requests servicing, the channel with the highest external request line has the highest priority.

34.3.2 Starting a channel

In idle mode, the FDMA waits for any of the channels to be started. All channels require at least one node in main memory which contain all or part of the necessary information required to execute a DMA transfer. For SCD/PES parsing, additional information is required and should be written to the appropriate registers in the FDMA data memory before the channel is started. See Section 34.4.3: *SCD/PES parsing on page 286*.

The channel initialization procedure is:

1. Create a linked list of nodes describing the transfer in main memory.
2. For SCD/PES parsing write the additional parameters:
   - 2.1 initial write pointer: SC_WRITE,
   - 2.2 size of the start code list: SC_SIZE,
   - 2.3 elementary stream buffer parameters: ESBUF_TOP, ESBUF_READ, ESBUF_WRITE, ESBUF_BOT).
3. Write the pointer to the first node and command data in the FDMA_CMD_STA register for the channel.
4. Write the **start** command to the command mailbox.

When a command is flagged in the command mailbox, the FDMA processes the command on the appropriate channel and then clears the bits in the command mailbox. Register FDMA_CMD_STA provides the current status of the channel and a pointer to the current node in the linked list.

Note: The FDMA_CMD_STA register must only be written to if the channel is idle or paused.

Before the **start** command is issued, the channel status and the address of the last node to have been loaded is provided from the FDMA_CMD_STA register for the channel.

A channel can be restarted (no channel initialization) if it was previously paused. When a channel is restarted, a pointer to a node in memory should not be written to FDMA_CMD_STA.

Note: Only one command for a channel can be sent in each direction until it is acknowledged. This means the CPU or FDMA must check the mailbox before writing a new command. Only if the bits corresponding to the channel are 0 (the previous message was acknowledged) is it safe to send another message.

34.3.3 Pausing a transfer

A transfer may be paused either by specifying this in the node structure or by sending a **pause** command to the FDMA. A pause is interpreted as a requirement to pause the channel at the earliest possible safe moment. The FDMA stops sending new requests over the STBus but continues to process outstanding return data (data is processed and placed into the FDMA's internal buffers but not sent to destination). A paced channel's DREQ is not processed once the **pause** command has been received.

The FDMA signals when the channel is paused by writing the status in the channels FDMA_CMD_STA register and setting the channel's bit in the interrupt mailbox (only if the **pause** command is issued by the host). The number of bytes remaining to be transferred for the current node and the pointer to the current node are available from the channel's FDMA_CNT and FDMA_CMD_STA registers respectively.

34.3.4 Flushing a channel

When a **pause** command is issued, the FDMA may still contain valid data inside its internal buffers. If this data is required, a **flush** command is issued instead of **pause**. When the **flush** command is received, the FDMA pauses the channel but also flushes any data it may have in the internal buffers. Once the internal buffers have been flushed, the channel is returned to the idle state and the host is interrupted. Flush only applies to paced channels where the source of data is paced.

Note: It is possible to flush a channel even if it is in the paused state.

34.3.5 Restarting a paused transfer

Once a channel has been paused, the host may restart the channel by issuing the **start** command. The FDMA continues processing the node from where it was paused.

Note: When a channel is restarted a pointer to a node does not need to be written to FDMA_CMD_STA.

34.3.6 Abort

Once a channel is paused, the host can start a new transfer simply by starting the channel with a new linked list of nodes. This effectively aborts the previous transfer.

34.3.7 Error handling

Errors are signalled by the FDMA setting the channel's error bit in the interrupt mailbox register. This generates an interrupt request if the mailbox error bit is masked. The interrupt mailbox shows which channel is signalling the error.

The type of error is specified in the FDMA_CMD_STA register for the channel (bits 2, 3 and 4). The error bits in the FDMA_CMD_STA register are only valid if the error bit in the interrupt mailbox is set, otherwise they should be ignored.

If the FDMA cannot continue processing, it pauses after signalling the error. The status is read from FDMA_CMD_STA to determine whether the channel was paused or whether it is still running.

The interrupt is acknowledged by clearing both the error bit and the interrupt bit in the interrupt mailbox.

34.4 Setting up FDMA transfers

An FDMA channel transfers units of data from a source data structure to a destination data structure using the de-coupling internal DRAM.

After channel initialization, the FDMA acknowledges the **start** command by clearing the channel's command mailbox register bits and begins the transfer by loading nodes one at a time from main memory. The transfer specified by each node is executed before the next node is loaded. The transfer continues until:

- the transfer is complete, or
- the FDMA is required to pause at the end of a node, or
- there is an error condition and it is not possible to continue.

A node is complete when all NBYTES of the node have been transferred. At the end of each node (specified in the FDMA_*NODE_NBYTES register) the FDMA may do one of the following:

- continue on to the next node without interrupting the host, or
- interrupt the host and continue on to the next node, or
- interrupt the host and pause the channel until the host tells the FDMA to continue, or
- interrupt the host and return the channel to the idle state if there are no more nodes to process

Note: It is mandatory for the FDMA to interrupt the host on completion of the last node.

34.4.1 Memory-to-memory moves and free running transfers

This channel type is described entirely by the generic node structure. Data is organized in a number of different ways, including:

- single location (0D),
- incrementing linear arrays (1D),
- incrementing rectangular arrays (2D).

All possible combinations of source and destination data organization is possible for a transfer. For example from 0D source to 2D destination or 1D source to 2D destination.

All data structures are specified with respect to an origin or initial byte address and the address of subsequent transfers is calculated from the origin using information provided in the node.

Single location (0D)

If the data structure is fixed or single location, the same address is used throughout the transfer and no further information is needed. This is set in each node by
FDMA_NODE_CTRL.SRC_INC = constant source for an 0D source, or
NODE.CONTROL.DST_INC = constant source for an 0D destination.

Incrementing (1D)

A structure is 1D if NODE_NBYTES = NODE_LEN or if NODE_SSTRIDE = NODE_LEN.

For any 1D transfer the address is incremented by one byte until *NODE_NBYTES bytes are transferred.

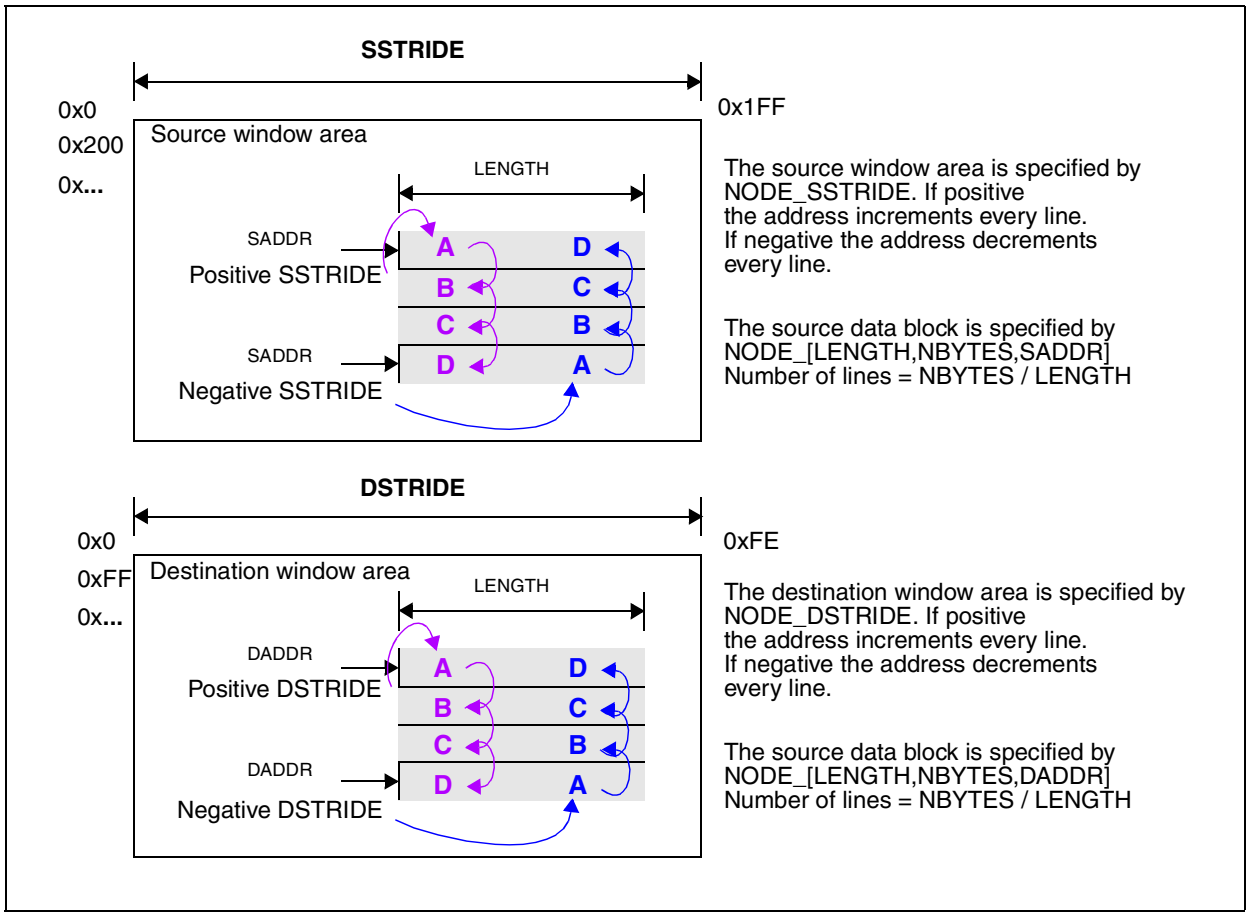
In the case of a incrementing transfer from address NODE_SADDR of NODE_NBYTES bytes, the following bytes at the following addresses are transferred:

NODE_SADDR to NODE_SADDR + (NODE_NBYTES - 1)

Rectangular array (2D)

If NODE_NBYTES ≠ NODE_LEN and abs(NODE_[SID]STRIDE) ≠ NODE_LEN then the transfer is 2D

Figure 72: 2D transfers



The bytes transferred during a 2D transfer are specified by:

```
for (y=0;y<(NODE.NBYTES / NODE_LEN);y++)
     for (x=0;x<NODE_LEN;x++)
*(char*)(NODE.DADDR+x+NODE.DSTRIDE*y) = *(char*)(NODE_SADDR+x+NODE.SSTRIDE*y)
```

Note: To do a 1D to 2D (or 2D to 1D) transfer, the source or destination stride must be specified (depending on whether the source or destination should be 1D) to be the same as the length.

Table 107: Summary of node parameter settings for different combinations of data organization

Source buffer	Destination buffer		
	0D	1D	2D
0D	L = source location size SS = 0 DS = 0	L = source location size SS = 0 DS = L	L = source location size SS = 0 DS = destination stride
1D	L = source location size SS = L DS = 0	L = NBytes SS = 0 DS = 0	L = destination line length SS = L DS = destination stride
2D	L = source location size SS = source stride DS = 0	L = source line length SS = source stride DS = L	L = destination line length SS = source stride DS = destination stride

Table 108: Definitions

Acronym	Description	Variable name
L	Length	*Length*
SS	Source stride	*SourceStride*
DS	Destination stride	*DestinationStride*
NBytes	Number of bytes to transfer	*NumberBytes*

34.4.2 Paced transfers

The FDMA transfers data to or from paced peripherals, using the data memory as a temporary store. Paced channels are triggered by a DREQ signal from a peripheral device. The memory side of the transfer is described by the generic node structure see (Section 34.1: *Channel structures on page 278*). The paced side is described by the FDMA_REQ_CTRL*n* parameters. The node's control word (NODE_CTRL) specifies a REQ_MAP number describing which DREQ line and FDMA_REQ_CTRL word to use for the transfer.

Once the channel is started, the FDMA writes and reads data from the paced channel when the DREQ for the channel is asserted. A holdoff mechanism ensures that a DREQ is not sampled immediately after it has been serviced (the holdoff period is specified in the FDMA_REQ_CTRL registers). Latency in the interconnect may cause the DREQ to remain high for some time after it has been serviced. The channel's circular FIFO in FDMA data memory is filled or emptied of data depending on whether the level of data is below or above the threshold (typically half the FIFO size).

Data is stored temporarily in a circular buffer in data memory before being transferred to the destination.

Supported paced peripherals and corresponding channel IDs are shown in Table 105: *FDMA request summary on page 276*.

Paced peripheral is data source

When the paced peripheral is the data source, data is fetched from the peripheral when its DREQ is asserted. When the read request is sent to the peripheral, the DREQ is masked using the holdoff mechanism (the holdoff period is specified in the FDMA_REQ_CTRL word). This prevents the DREQ from being sampled high again immediately after it has been serviced. The data is then stored in the channel's circular buffer. If the buffer level exceeds the threshold, data is read from the buffer and sent to the destination.

Paced peripheral is destination

Data is read from the source and placed into the circular buffer in data memory when the buffer level falls below the threshold. When the DREQ for a paced peripheral is asserted, data is fetched from the circular buffer and sent to the peripheral. The holdoff mechanism is used to mask the DREQ to prevent it being sampled high again immediately after it has been serviced (see FDMA timing model, Section 34.2.2: *Paced on page 279*).

34.4.3 SCD/PES parsing

Introduction

A single FDMA channel can be used to perform PES parsing and start code detect (SCD) on an incoming stream of data. The intended use is as part of the video decode flow. Data is fetched from a source buffer. The source data may be either PES data or ES data. If the source data is PES, FDMA can parse the PES stream, extracting the PTS (program transport stream) from the PES header and the ES data from the PES payload. During the transfer of the ES data, SCD can be performed on the ES data. A list of PTS and start codes is sent to a start codes list buffer. A PES parsing and SCD channel is described by a linked list of nodes and an entry in the additional data regions, as shown in Figure 73.

Each node in the list describes a transfer from the source buffer. The source buffer can contain either PES (PES parsing enabled) or ES (PES parsing disabled). The node also specifies which additional data region is associated with it. The additional data region contains the output ES buffer descriptors, the start codes list descriptors and the PES and ES start code ranges and mode selection flags.

Figure 73: PES parsing/start code detect data structures

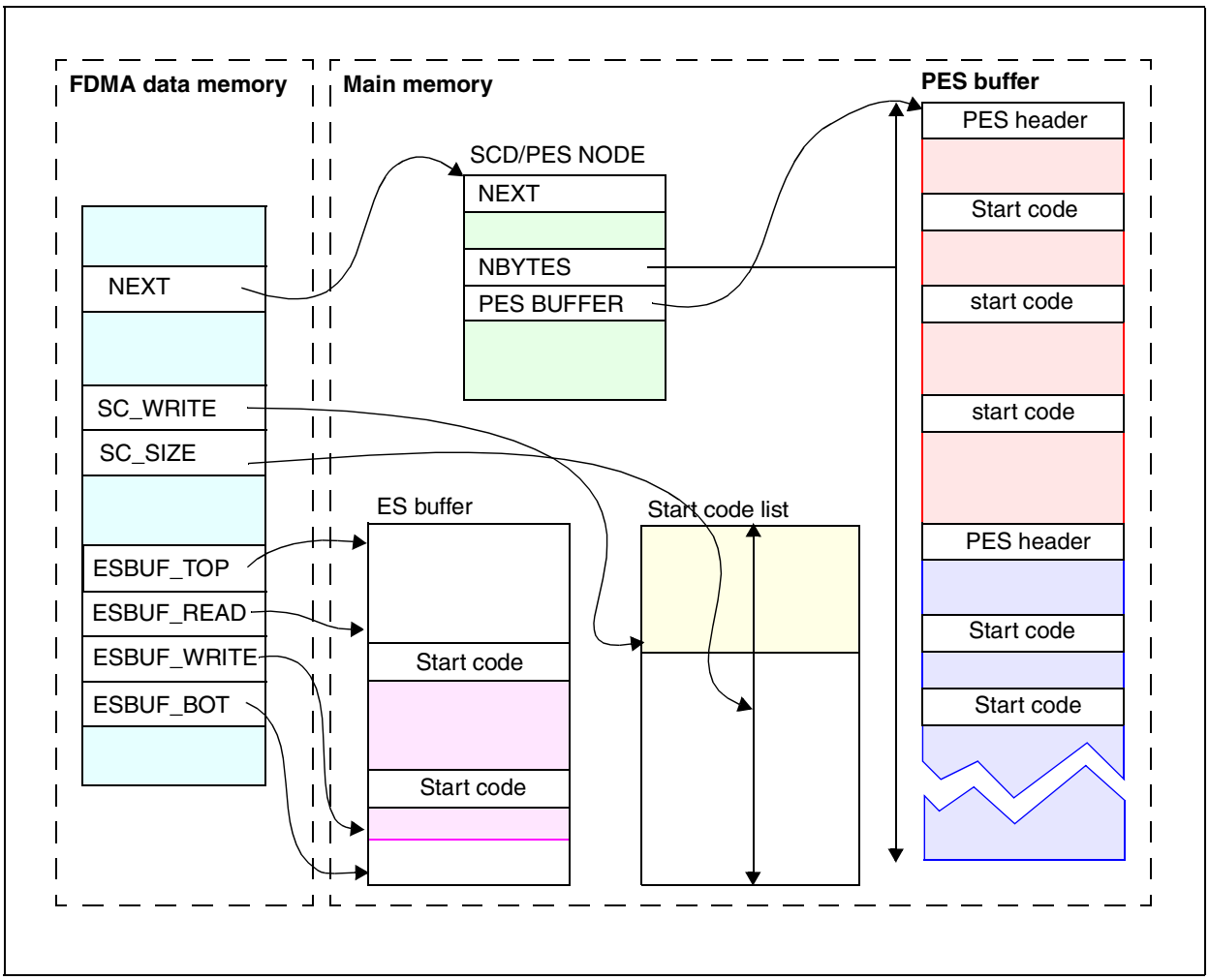
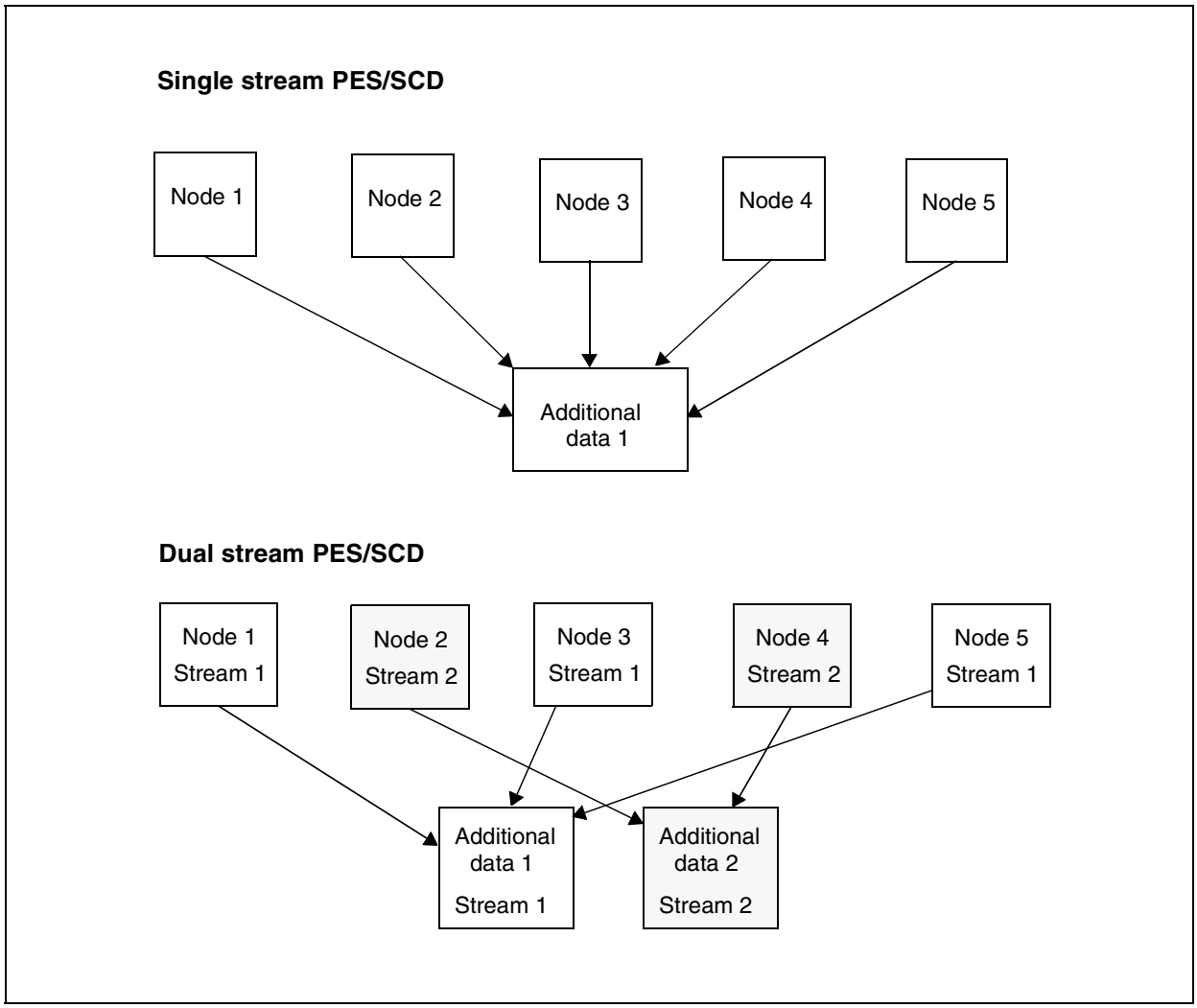


Figure 74: Single and dual stream PES parsing/start code detect



Single stream PES parsing/start code detect (SCD) is characterized by a linked list of nodes and one additional data region. For dual stream PES/SCD, two additional data regions are used, one for each stream, but a single FDMA channel is used. The transfers should be multiplexed on this single channel (alternating nodes in the linked list). This is shown in Figure 74.

FDMA is able to support PES/SCD in two modes - MPEG2 and H.264.

PES Parsing

A channel configured as a PES Parsing/SCD channel fetches data from a **128 byte aligned linear buffer** in main memory (a circular buffer should be handled with a pair of nodes). If PES parsing is enabled, the ES payload is extracted from the stream and dumped to the ES Buffer. The PES startcode range is specified in the PES Control word in the additional data region. It is also possible to enable PTS detection and extraction whereby the presence of a PTS in the PES header is detected and the PTS is dumped to the start codes list. Transfers from the source buffer must be 128 byte aligned and multiples of 128 bytes. If PES parsing is disabled, the incoming stream is treated as ES and simply copied to the ES buffer.

Start code detect

During the transfer of the ES data to the ES Buffer, start code detection can be performed on the data. FDMA supports both MPEG2 mode and H.264 mode start code detect. The SCD mode is specified in the SC1_CTRL word in the additional data region.

MPEG2 mode start code detect

When MPEG2 mode SCD is enabled, the FDMA detects any start codes falling inside two ranges specified in the additional data region. For each of the two start code ranges, **normal detection** and **one-shot detection** can be selected independently. Normal detection for a range causes all start codes falling within the range to be output to the start codes list. If one-shot detection is enabled for a range, the first start code detected in the range is output to the start codes list. Detection in the range is subsequently be disabled until a start code in the other range is detected, detection in the range is then enabled again.

H.264 mode start code detect

Only one start code range can be specified when H.264 SCD is enabled. When a start code within the range is found, the FDMA decodes the SC_VAL.NAL_UNIT_TYPE and perform different operations depending on its value:

Table 109: NAL_UNIT_TYPE actions

NAL_UNIT_TYPE	Action performed

1, 5	**Exp golomb decode first_mb_in_slice field
	if(egd(first_mb_in_slice) < egd(previous
	first_mb_in_slice)) {
	// start of picture
	output start code entry with slice count
	set slice counter to 1
	}else{
	// not start of picture
	increment slice counter
	}**

6, 9, 10, 11	Output start code entry with slice count set slice counter to 1
7, 8	Output start code entry without slice count
Others	Do nothing

Specifying start code ranges

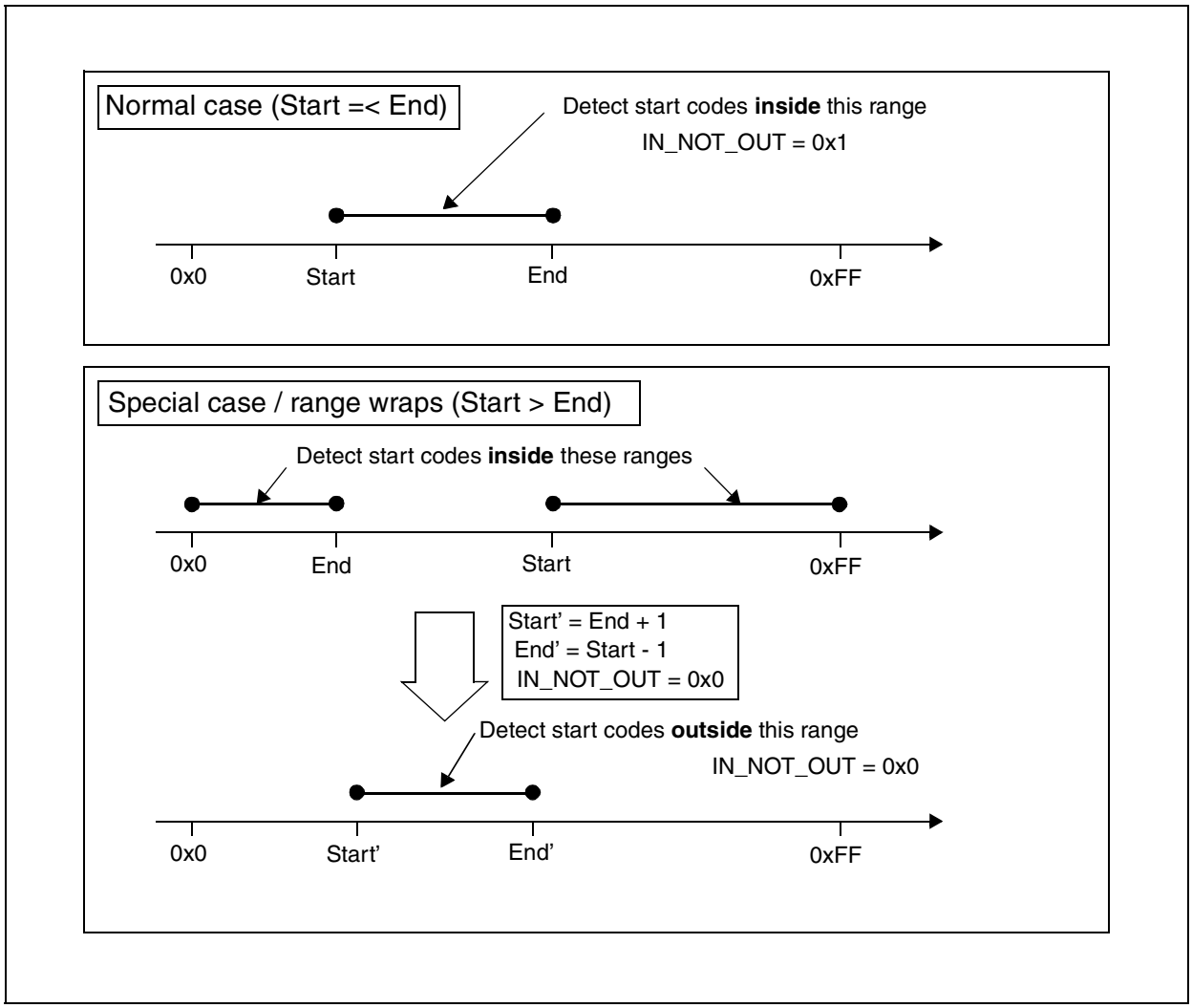
When specifying the start code ranges the host must ensure that

RANGE_END >= RANGE_START

(see FDMA_PES_CTRLn and FDMA_PES_CTRLn). In order to respect this rule when the range wraps, a start code range [Start, End] must be transformed into a nonwrapping range using the following procedure:

```
if(Start <= End)
{
   IN_not_OUT = 1
   RANGE_START = Start
   RANGE_END = End
}
else
{
   IN_not_OUT = 0
   RANGE_START = End + 1
   RANGE_END = Start -1
}
```

Figure 75: Start code range setup



Overflow handling

There may be occasions when either the start codes list or the ES buffer overflows.

When the start codes list overflows the FDMA sets bit 0 in the SC_WRITE register for the channel and continues parsing the PES data and outputting data to the ES buffer. However, no start code is added to the list. If the incoming data is ES, it is simply copied to the ES buffer.

When the ES buffer overflows, the FDMA signals this by setting bit 0 in the ESBUF_WRITE register for the channel. The FDMA ignores the overflow condition and continues processing data thereby overwriting data already in the ES buffer.

Start codes list

The start codes list is output as a linear array of data structures. Each structure is four words in size (16 bytes) and may contain either start code data or PTS data. A label specifies whether the structure contains a start code or PTS. To know how many entries have been placed in the start codes list, the driver must read SC_WRITE and compare it with the value it had written at the start of the transfer. The difference, divided by 4 gives the number of entries in the start codes list.

34.4.4 Audio data output through S/PDIF player

The S/PDIF channel is a paced channel in which the FDMA formats data before outputting the data to the S/PDIF player. S/PDIF output involves the PTI, ST40, FDMA and S/PDIF player. The ST40 parses the PES stream coming from PTI and sets up a transfer on the S/PDIF channel.

Each node in the linked list contains the data to enable the FDMA to fetch data from a buffer, format it and output it when the S/PDIF player raises a DREQ. The node's source address points to the start of the audio frame. The node also contains the information required by the FDMA to format the incoming data. The formatted data is output to the S/PDIF player when it raises the DREQ. In order to correctly describe the data transfers for a single burst, it may be necessary to use a linked list of two or more nodes. The BURST_END bit indicates to the FDMA when the last set of data for the burst is being transferred. This allows the FDMA to automatically generate the correct stuffing to complete a burst. The node following a node in which BURST_END = 1 is considered the first node of the next burst.

Valid nodes

The S/PDIF channel nodes' control word contains a node valid bit. This bit indicates whether the node is valid. If a node is loaded by FDMA and the valid bit is not set, the FDMA completes the current burst by sending stuffing data and then return the channel to idle. The CPU is interrupted.

This mechanism only works correctly if the words are read in order (from lowest address to highest) when fetching the node from LMI. This is because the audio driver is expected to fill the node structure and validate it once the node structure has been completed.

End of burst

The BURST_END bit, part of the S/PDIF node's CONTROL parameter is used to indicate that the node describes the last node of an S/PDIF burst. The FDMA uses this to know when it should start to output stuffing. The node following a node with BURST_END = 0x1 is considered the start of a burst.

Output of stuffing

The FDMA needs to output stuffing to the S/PDIF player when the data for a burst is finished but the end of the burst has not been reached. The FDMA keeps a count of the outstanding frames to output for a burst. When an end of burst node is received and all the data associated with that node is transferred or when an invalid node is received, the FDMA outputs stuffing until the outstanding frames counter goes to zero.

35 Flexible DMA (FDMA) registers

Register addresses are provided as *FDMABaseAddress* + offset or *MemoryOffset* + offset.

The *FDMABaseAddress* is:

0x1922 0000.

All node registers are nonvolatile. Other registers are volatile.

SCD and PES parsing channels require additional data. These transfer types are described by nodes which include an additional data region number. Each additional data region is composed of 16 words and the format is transfer-type dependent.

Table 110: FDMA register summary (FDMABaseAddress)

Register	Description	Offset	Type
FDMA interface			
FDMA_ID	Hardware ID	0x0000	R/W[a]
FDMA_VER	Version number	0x0004	R/W[a]
FDMA_EN	Enable controller	0x0008	R/W[a]
Channel interface			
FDMA_REV_ID	Revision number	0x8000	R/W[a]
FDMA_CH_CMD_STAn	Command and status for channel n	0x8040 + *n* x 4	Special
FDMA_PTRn	Pointer to next node for channel n	0x9180 + *n* x 64	RO
FDMA_CNTn	Byte count	0x9188 + *n* x 64	RO
FDMA_SADDRn	Source address for channel n node	0x918C + *n* x 64	RO
FDMA_DADDRn	Destination address for channel n node	0x9190 + *n* x 64	RO
FDMA_REQ_CTRLn	Request control	0x9780 + *n* x 4	WO
Command mailbox			
FDMA_CMD_STA	Command status	0xBFC0	RO
FDMA_CMD_SET	Set command	0xBFC4	WO
FDMA_CMD_CLR	Clear command	0xBFC8	WO[a]
FDMA_CMD_MASK	Mask command	0xBFCC	WO
Interrupt mailbox			
FDMA_INT_STA	Interrupt status	0xBFD0	RO
FDMA_INT_SET	Set interrupt	0xBFD4	WO[a]
FDMA_INT_CLR	Clear interrupt	0xBFD8	WO
FDMA_INT_MASK	Interrupt mask	0xBFDC	WO

a. Writable only during initialization

Table 111: FDMA register summary (MemoryOffset)

Register	Description	Offset	Type
Memory-to-memory moves and paced transfer			
FDMA_NODE_NEXT	Next register set pointer	0x0000	R/W
FDMA_NODE_CTRL	Channel control	0x0004	R/W
FDMA_NODE_NBYTES	Transfer count	0x0008	R/W
FDMA_NODE_SADDR	Channel source address	0x000C	R/W
FDMA_NODE_DADDR	Channel destination address	0x0010	R/W
FDMA_NODE_LEN	2D line length	0x0014	R/W
FDMA_NODE_SSTRIDE	2D source stride	0x0018	R/W
FDMA_NODE_DSTRIDE	2D destination stride	0x001C	R/W
S/PDIF			
FDMA_SPD_NODE_NEXT	Next node pointer	0x0000	R/W
FDMA_SPD_NODE_CTRL	S/PDIF node control	0x0004	R/W
FDMA_SPD_NODE_NBYTES	Number of bytes to read	0x0008	R/W
FDMA_SPD_NODE_SADDR	Source address	0x000C	R/W
FDMA_SPD_NODE_DADDR	Destination address	0x0010	R/W
FDMA_SPD_NODE_PA_PB	PA and PB preamble words	0x0014	R/W
FDMA_SPD_NODE_PC_PD	PC and PD preamble words	0x0018	R/W
FDMA_SPD_NODE_BST_PER	Burst period	0x001C	R/W
FDMA_SPD_NODE_CHn_STA_LO	Channel status LSB	0x0020, 0x0028	R/W
FDMA_SPD_NODE_CHn_STA_HI	Channel status MSB	0x0024, 0x002C	R/W
SCD/PES parsing			
FDMA_PES_NODE_NEXT	Next node pointer	0x0000	R/W
FDMA_PES_CDP_NODE_CTRL	Node control	0x0004	R/W
FDMA_PES_NODE_NBYTES	Number of bytes to be read from PES buffer	0x0008	R/W
FDMA_PES_BUFF	Read pointer to PES buffer	0x000C	R/W
Additional data regions			
FDMA_SC_WRITEn	MPEG2/H.264 start code list write pointer (region n)	0x9580, 0x9600, 0x9680, 0x9700	R/W
FDMA_SC_SIZEn	MPEG2/H.264 start code list size (region n)	0x9584, 0x9604, 0x9684, 0x9704	R/W
FDMA_ESBUF_TOPn	MPEG2/H.264 top address of elementary stream buffer (region n)	0x9588, 0x9608, 0x9688, 0x9708	R/W
FDMA_ESBUF_READn	MPEG2/H.264 elementary stream buffer read pointer (region n)	0x958C, 0x960C, 0x968C, 0x970C	R/W
FDMA_ESBUF_WRITEn	MPEG2/H.264 elementary stream buffer write pointer (region n)	0x9590, 0x9610, 0x9690, 0x9710	R/W
FDMA_ESBUF_BOTn	MPEG2/H.264 elementary stream buffer bottom address (region n)	0x9594, 0x9614, 0x9694, 0x9714	R/W
FDMA_PES_CTRLn	MPEG2/H.264 PES header start code range control (region n)	0x9598, 0x9618, 0x9698,	R/W
FDMA_SC1_CTRLn	MPEG2/H.264 start code range control (region n)	0x959C, 0x95A0, 0x961C, 0x9620 0x969C, 0x96A0	R/W

Table 111: FDMA register summary (MemoryOffset)

Register	Description	Offset	Type
FDMA_SC2_CTRLn	MPEG2 start code range control (region n)	0x959C, 0x95A0, 0x961C, 0x9620 0x969C, 0x96A0	R/W
FDMA_SCD_STAn	Start code detector state (region n)	0x95A4 - 0x95BF, 0x9624 - 0x963F, 0x96A4 - 0x95BF	R/W
Start code entries			
FDMA_SC_TYPE	Type of data in this entry	0x0000	R/W
FDMA_SC_ADDR	Memory address of start code	0x0004	R/W
FDMA_SC_VAL	Start code value	0x0008	R/W
PTS entries			
FDMA_PTS_TYPE	Type of data in this entry	0x0000	R/W
FDMA_PTS_ADDR	Memory address of PTS	0x0004	R/W
FDMA_PTS_HI	MSB of PTS value	0x0008	R/W
FDMA_PTS_LO	LSB of PTS value	0x000C	R/W

35.1 FDMA interface

FDMA_ID Hardware ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_NUM																															

Address: *FDMABaseAddress* + 0x0000
Type: R/W (writable only during initialization)
Reset: 0
Description: Holds hardware ID number.

FDMA_VER Version number

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VER_NUM																															

Address: *FDMABaseAddress* + 0x0004
Type: R/W (writable only during initialization)
Reset: 0
Description: Holds hardware version number.

FDMA_EN Enable controller

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															EN

Address: *FDMABaseAddress* + 0x0008
Type: R/W (writable only during initialization)
Reset: 0
Description: 1: Block enabled or running
0: Block stopped, CPU can access embedded memory.

35.2 Channel interface

FDMA_REV_ID Revision number

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV_NUM																															

Address: *FDMABaseAddress* + 0x8000
Type: R/W (writable only during initialization)
Reset: Undefined
Description: Holds low level revision number.

FDMA_CH_CMD_STAn Command and status for channel n

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Channel running	DATA																												ERR		STA	
Channel idle or paused	DATA																												CMD			

Address: *FDMABaseAddress* + 0x8040 + *n* x 4 (where *n* = 0 to 15)
Type: RO (when channel is running)

R/W (when channel is idle or paused)

Reset:

Description: The FDMA_CH_CMD_STA registers are used both as command and status registers. When the channel is running or enters the paused state, the register is read only and provides the current channel status and the address of the node being processed. When the channel is idle or paused, the register provides the node pointer and additional information prior to issuing a START command.

Channel running

[31:5] **DATA**
When channel is idle, address of last node to have been loaded, or 0x0 if this channel has never been used. Otherwise, current node address.

[4:2] **ERR**: Error type (only valid if interrupt mailbox error bit is set)
000: Interrupt missed
001: Start code list overflow
010: ES buffer overflow
Others: Reserved

[1:0] **STA:** Channel status
00: Channel is idle
10: Channel is running
11: Channel is paused
01: Reserved

Channel idle or paused

[31:5] **DATA**
Command pointer to node if command is START
Reserved otherwise

[4:0] **CMD**: Command
0 0001: START and initialize channel *n*
0 0000: RESTART channel *n* (no initialization) from where it was paused
Others: Reserved

FDMA_PTRn Pointer to next node for channel n

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NODE_PTR*n*																											Reserved					

Address: *FDMABaseAddress* + 0x9180 + *n* x 64 (where *n* = 0 to 15)

Type: Read only

Reset: Undefined

Description: Address of next node (32-byte aligned)

FDMA_CNTn Byte count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT																															

Address: *FDMABaseAddress* + 0x9188 + *n* x 64 (where *n* = 0 to 15)

Type: Read only

Reset: Undefined

Description: Number of bytes remaining to be transferred for current node.

FDMA_SADDRn Source address for channel n node

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_ADDR																															

Address: *FDMABaseAddress* + 0x918C + *n* x 64 (where *n* = 0 to 15)
Type: Read only
Reset: Undefined
Description: Current source address for channel *n* node.

FDMA_DADDRn Destination address for channel n node

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEST_ADDR																															

Address: *FDMABaseAddress* + 0x9190 + *n* x 64 (where *n* = 0 to 15)
Type: Read only
Reset: Undefined
Description: Current destination address for channel *n* node.

FDMA_REQ_CTRLn Request control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		INC_ADDR		NUM_OPS				SRC_DEST_NUM		Reserved							WNR	Reserved						OPCODE				HOLDOFF			

Address: *FDMABaseAddress* + 0x9780 + *n* x 4 (where *n* = 0 to 30)
Type: Write only
Reset: Undefined
Description:

[31:30] **Reserved**

[29] **INC_ADDR**: Increment address
0: No address increment between transfers
1: Increment address between transfers

[28:24] **NUM_OPS**: Number of operations per request serviced
0x0: 1 transfer
0x1: 2 transfers
0x2: 3 transfers
0x3: 4 transfers
Others: Reserved

[23:22] **SRC_DEST_NUM**: Source destination number
0x0: Software transport stream
0x1: All other paced channels (See Table 105: *FDMA request summary* *on page 276*)
Others: reserved

[21:15] **Reserved**

[14] **WNR**: Write not read
0: Read from paced peripheral — 1: Write to paced peripheral

[13:8] **Reserved**

[7:4] **OPCODE**: STBus opcode
0x0: LD/ST1 [1] — 0x1: LD/ST2 [2]
0x2: LD/ST4 — 0x3: LD/ST8
0x4: LD/ST16 — 0x5: LD/ST 32

[1] intended as four LD/ST1 (4 bytes transferred) — [2] intended as two LD/ST 2 (4 bytes transferred)

[3:0] **HOLDOFF**: Holdoff value
DREQ is masked for:
0x0: 0: 0.5 µs — 0x1: 0.5 - 1 µs
0x2: 1: 1.5 µs — Others: Reserved

35.3 Command mailbox

FDMA_CMD_STA Command status

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Address: *FDMABaseAddress* + 0xBFC0
Type: Read only
Reset: Undefined
Description: Indicates a pending command for channel *n*.

[2 x *n* + 1:2 x *n*] **CH*n***
00: Reserved — 01: Start channel *n*
10: Pause channel *n* — 11: Flush and pause channel *n*

FDMA_CMD_SET Set command

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Address: *FDMABaseAddress* + 0xBFC4
Type: Write only
Reset: Undefined
Description: Notify FDMA there is a command for channel *n*.

[2 x *n* + 1:2 x *n*] **CH*n***
00: Reserved — 01: Start channel *n*
10: Pause channel *n* — 11: Flush and pause channel *n*

FDMA_CMD_CLR Clear command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	

Address: *FDMABaseAddress* + 0xBFC8

Type: Write only at initialization

Reset: Undefined

Description: When set these bits acknowledge the command for channel *n* by clearing bits in the FDMA_CMD_STA register.

FDMA_CMD_MASK Mask command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	

Address: *FDMABaseAddress* + 0xBFCC

Type: Write only

Reset: Undefined

Description: Enable interrupt generation for channel *n*.

[2 x *n* + 1:2 x *n*] **CH*n***
00: Disable channel *n* messaging
11: Enable channel *n* messaging
Others: Reserved

35.4 Interrupt mailbox

FDMA_INT_STA Interrupt status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR15	CH15	ERR14	CH14	ERR13	CH13	ERR12	CH12	ERR11	CH11	ERR10	CH10	ERR9	CH9	ERR8	CH8	ERR7	CH7	ERR6	CH6	ERR5	CH5	ERR4	CH4	ERR3	CH3	ERR2	CH2	ERR1	CH1	ERR0	CH0

Address: *FDMABaseAddress* + 0xBFD0

Type: Read only

Reset: Undefined

Description: When a channel is running, its FDMA_CMD_STA register is written by the FDMA and always contains the current state of the channel. The interrupt mailbox register is used by the FDMA to interrupt the CPU. When the CPU receives an interrupt the mailbox should be read to determine which channel generated the interrupt and the interrupt acknowledged by clearing the channel's bit in the interrupt mailbox. The channel's status can be read from the FDMA_CMD_STA register.

The FDMA may raise an interrupt either if this is specified in the node or if a pause command is issued by the host. If the host issues a pause command but does not want to be interrupted by the FDMA when the channel enters the paused state, it should unmask the interrupt by clearing the channels mask bit in the interrupt mailbox.

Each channel has two bits associated with it in the interrupt mailbox: an interrupt bit and an ERR bit.

[*n* x 2] **CH*n***: If bit *n* = 1, message pending for channel *n*

[*n* x 2 + 1] **ERR*n***: If bit *n* = 1, there is an error for channel *n*

FDMA_INT_SET Set interrupt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR15	CH15	ERR14	CH14	ERR13	CH13	ERR12	CH12	ERR11	CH11	ERR10	CH10	ERR9	CH9	ERR8	CH8	ERR7	CH7	ERR6	CH6	ERR5	CH5	ERR4	CH4	ERR3	CH3	ERR2	CH2	ERR1	CH1	ERR0	CH0

Address: *FDMABaseAddress* + 0xBFD4

Type: R/W (writable only on initialization)

Reset: Undefined

Description: These bits generate an interrupt for channel *n*.

[*n* x 2] **CH*n***: Generate interrupt for channel *n*

[*n* x 2 + 1] **ERR*n***: Generate interrupt for error on channel *n*

FDMA_INT_CLR Clear interrupt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR15	CH15	ERR14	CH14	ERR13	CH13	ERR12	CH12	ERR11	CH11	ERR10	CH10	ERR9	CH9	ERR8	CH8	ERR7	CH7	ERR6	CH6	ERR5	CH5	ERR4	CH4	ERR3	CH3	ERR2	CH2	ERR1	CH1	ERR0	CH0

Address: *FDMABaseAddress* + 0xBFD8

Type: Write only

Reset: Undefined

Description: When set these bits acknowledge an interrupt for channel *n* by clearing the relevant status bits.

[*n* x 2] **CH*n***: Acknowledge message for channel *n*

[*n* x 2 + 1] **ERR*n***: Acknowledge error for channel *n*

FDMA_INT_MASK Interrupt mask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR15	CH15	ERR14	CH14	ERR13	CH13	ERR12	CH12	ERR11	CH11	ERR10	CH10	ERR9	CH9	ERR8	CH8	ERR7	CH7	ERR6	CH6	ERR5	CH5	ERR4	CH4	ERR3	CH3	ERR2	CH2	ERR1	CH1	ERR0	CH0

Address: *FDMABaseAddress* + 0xBFDC

Type: Write only

Reset: Undefined

Description: When set these bits enable interrupt generation for channel *n*.

[*n* x 2] **CH*n***: If bit *n* = 1, message pending for channel *n*

[*n* x 2 + 1] **ERR*n***: If bit *n* = 1, there is an error for channel *n*

35.5 Memory-to-memory moves and paced transfer

FDMA_NODE_NEXT Next register set pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXT_PTR																											0	0	0	0	0

Address: *MemoryOffset* + 0x0000
Type: R/W
Reset: Undefined
Description: Pointer to the next node in the linked list. Aligned on a 32-byte boundary.
Note: 0x0 terminates the list and halts the channel.

FDMA_NODE_CTRL Channel control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN	PAU_EN	Reserved																					DST_INC		SRC_INC		REQ_MAP				

Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description:

[31]: **INT_EN**: End-of-node interrupt
0: Do not generate an interrupt at the completion of a node
1: Generate an interrupt at end of node

[30] **PAU_EN**: Pause at end-of-node
0: No pause at end of node
1: Pause at end of node

[29:9] **Reserved**

[8:7] **DST_INC**: Destination address increment
00: Reserved
01: Constant destination address
10: Incrementing destination address
11: Reserved

[6:5] **SRC_INC**: Source address increment
00: Reserved
01: Constant source address
10: Incrementing source address
11: Reserved

[4:0] **REQ_MAP**: DREQ mapping
0: Free running
[1 .. 30]: Paced, select DREQ signal and REQ_CTRL as listed in Table 105: *FDMA request summary on page 276*):
31: Extended node type (see Section 35.7: *SCD/PES parsing on page 305*

FDMA_NODE_NBYTES Transfer count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBYTES																															

Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: The number of bytes to be transferred for this node.

FDMA_NODE_SADDR Channel source address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDR																															

Address: *MemoryOffset* + 0x000C
Type: R/W
Reset: Undefined
Description: The source address from which the transfer begins. For the memory-side of a paced transfer, bits 0 to 4 must be 0, that is, aligned to 32 bytes.

FDMA_NODE_DADDR Channel destination address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DADDR																															

Address: *MemoryOffset* + 0x0010
Type: R/W
Reset: Undefined
Description: The target address for the transfer. For the memory-side of a paced transfer, bits 0 to 4 must be 0, that is, aligned to 32 bytes.

FDMA_NODE_LEN 2D line length

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENGTH																															

Address: *MemoryOffset* + 0x0014
Type: R/W
Reset: Undefined
Description: The length of a line in a 2D data move measured in bytes.

FDMA_NODE_SSTRIDE 2D source stride

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSTRIDE																															

Address: *MemoryOffset* + 0x0018
Type: R/W
Reset: Undefined
Description: The stride between lines in source 2D data structures measured in bytes.

FDMA_NODE_DSTRIDE 2D destination stride

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSTRIDE																															

Address: *MemoryOffset* + 0x001C
Type: R/W
Reset: Undefined
Description: The stride between lines in destination 2D data structures measured in bytes.

35.6 S/PDIF

FDMA_SPD_NODE_NEXT Next node pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXT_PTR																											0	0	0	0	0

Address: *MemoryOffset* + 0x0000
Type: R/W
Reset: Undefined
Description: Pointer to the next node in the linked list. Aligned on a 32-byte boundary.
Note: 0x0 terminates the list and halts the channel.

FDMA_SPD_NODE_CTRL S/PDIF node control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN	PAU_EN	NODE_VALID	BURST_END	Reserved															DREQ					TYPE			REQ_MAP				

Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description:

[31] **INT_EN**: Interrupt at end of node
0: No interrupt at end of node 1: Interrupt host at end of node

[30] **PAU_EN**: Pause at end of node
0: Continue processing next node 1: Pause at end of this node

[29] **NODE_VALID**: Node valid bit
0: Node is not valid 0: Node is valid

[28] **BURST_END**: End of burst
0: node describes burst start or continuation
1: node describes the last data to transfer for the burst. Once this data has been transferred the FDMA will start sending stuffing until the end of the burst.

[27:13] **Reserved**

[12:8] **DREQ**: DREQ mapping
0x0 and 0x1F: Reserved 0x1 - 0x1E: Use register FDMA_REQ_CTRL[1 .. 30]

[7:5] **TYPE**: Node type
0x0: SCD/PES parsing node 0x1: S/PDIF node
0x2: 0x7 Reserved

[4:0] **REQ_MAP**: REQ_MAP/extended node type
0x1F: Extended node type 0x0 - 0x1E: Reserved
(DREQ mapping for standard node)

FDMA_SPD_NODE_NBYTES Number of bytes to read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBYTES																															

Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: Number of bytes to read from buffer.

FDMA_SPD_NODE_SADDR Source address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDR																															

Address: *MemoryOffset* + 0x000C
Type: R/W
Reset: Undefined
Description: The source address from which the transfer begins. For the memory-side of a paced transfer, bits 0 to 4 must be 0, that is, aligned to 32 bytes.

FDMA_SPD_NODE_DADDR Destination address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DADDR																															

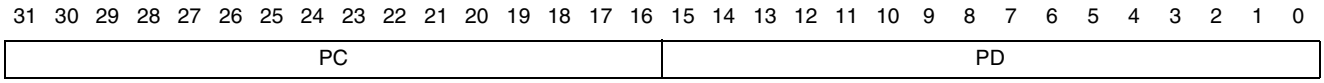
Address: *MemoryOffset* + 0x0010
Type: R/W
Reset: Undefined
Description: The target address for the transfer. For the memory-side of a paced transfer, bits 0 to 4 must be 0, that is, aligned to 32 bytes.

FDMA_SPD_NODE_PA_PB PA and PB preamble words

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA																PB															

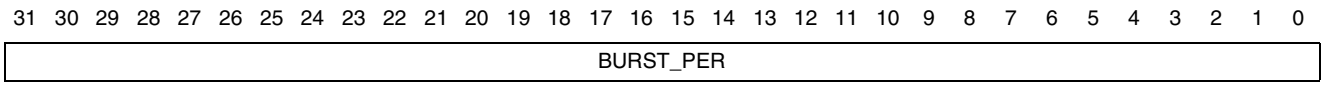
Address: *MemoryOffset* + 0x0014
Type: R/W
Reset: Undefined
Description: Preamble data for S/PDIF.

FDMA_SPD_NODE_PC_PD PC and PD preamble words



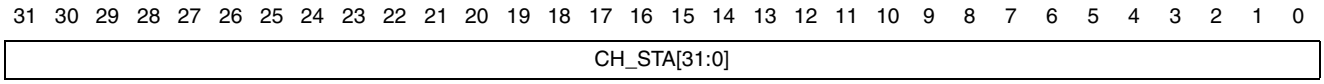
Address: *MemoryOffset* + 0x0018
Type: R/W
Reset: Undefined
Description: Preamble data for S/PDIF.

FDMA_SPD_NODE_BST_PER Burst period



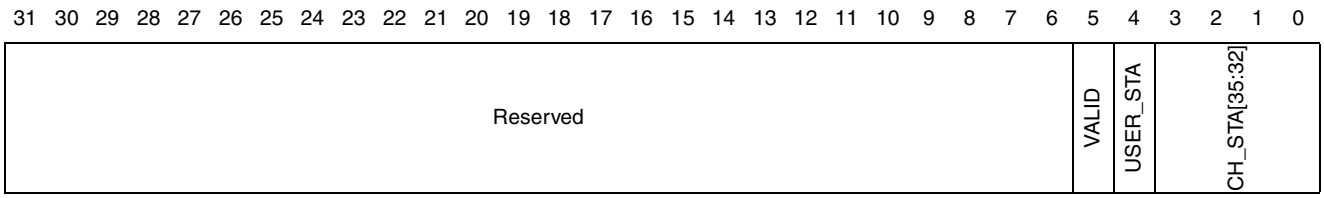
Address: *MemoryOffset* + 0x001C
Type: R/W
Reset: Undefined
Description: Number of S/PDIF frames in the burst.

FDMA_SPD_NODE_CHn_STA_LO Channel status LSB



Address: *MemoryOffset* + 0x0020 (Channel 0), 0x0028 (Channel 1)
Type: R/W
Reset: Undefined
Description: Channel status bits 31 to 0.

FDMA_SPD_NODE_CHn_STA_HI Channel status MSB



Address: *MemoryOffset* + 0x0024 (Channel 0), 0x002C (Channel 1)
Type: R/W
Reset: Undefined
Description:

[31:6] **Reserved**

[5] **VALID**: Validity bit

[4] **USER_STA**: User status bit

[3:0] **CH_STA[35:32]**: Channel status bits 35 to 32

35.7 SCD/PES parsing

FDMA_PES_NODE_NEXT Next node pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXT_PTR																											0	0	0	0	0

Address: *MemoryOffset* + 0x0000
Type: R/W
Reset: Undefined
Description: Pointer to the next node in the linked list. Aligned on a 32-byte boundary.
Note: 0x0 terminates the list and halts the channel.

FDMA_PES_CDP_NODE_CTRL Node control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN	PAU_EN	Reserved						TAG								Reserved				ADD_DATA				TYPE			REQ_MAP				

Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description:

[31] **INT_EN**: Interrupt at end of node
0: No interrupt at end of node
1: Interrupt host at end of node

[30] **PAU_EN**: Pause at end of node
0: Continue processing next node
1: Pause at end of this node

[29:24] **Reserved**

[23:16] **TAG**: Node tag. This tag is copied to the data structures put into the start codes list.

[15:12] **Reserved**

[11:8] **ADD_DATA**: Additional data associated with node
0x0: Additional data region 0
0x1: Additional data region 1
0x2: Additional data region 2
Others: Reserved

[7:5] **TYPE**: Node type
0x0 SCD/PES Parsing
0x1 S/PDIF
0x2 - 0x7: Reserved

[4:0] **REQ_MAP**: REQ_MAP/extended node type
0x1F: Extended node type
0x0 - 0x1E: Reserved (DREQ mapping for standard node)

FDMA_PES_NODE_NBYTES Number of bytes to be read from PES buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBYTES																															

Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: Number of bytes in PES buffer on which SCD/PES parsing should be performed.

FDMA_PES_BUFF Read pointer to PES buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PES_BUFF_READ																															

Address: *MemoryOffset* + 0x000C
Type: R/W
Reset: Undefined
Description: Read pointer to PES buffer.

35.7.1 Additional data regions

FDMA_SC_WRITEn MPEG2/H.264 start code list write pointer (region n)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SC_WRITE																												Reserved		SC_OVF	

Address: *FDMABaseAddress* +
0x9580 (region 0), 0x9600 (region 1), 0x9680 (region 2), 0x9700 (region 3)
Type: R/W
Reset: 0
Description:

[31:4] **SC_WRITE**: Write pointer for start codes list (must be 4-word aligned)

[3:1] **Reserved**

[0] **SC_OVF**: SC list overflow flag
0: No overflow 1: Overflow

FDMA_SC_SIZEn MPEG2/H.264 start code list size (region n)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SC_SIZE																														Res.	

Address: *FDMABaseAddress* +
0x9584 (region 0), 0x9604 (region 1), 0x9684 (region 2), 0x9704 (region 3)
Type: R/W
Reset: 0
Description: Size of start codes list (in number of words).

FDMA_ESBUF_TOPn MPEG2/H.264 top address of elementary stream buffer (region n)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESBUF_TOP																											Reserved					

Address: *FDMABaseAddress* +
0x9588 (region 0), 0x9608 (region 1), 0x9688 (region 2), 0x9708 (region 3)

Type: R/W

Reset: 0

Description: Address of top of elementary stream buffer (32-byte aligned)

FDMA_ESBUF_READn MPEG2/H.264 elementary stream buffer read pointer (region n)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESBUF_READ																															

Address: *FDMABaseAddress* +
0x958C (region 0), 0x960C (region 1), 0x968C (region 2), 0x970C (region 3)

Type: R/W

Reset: 0

Description:

FDMA_ESBUF_WRITEn MPEG2/H.264 elementary stream buffer write pointer (region n)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESBUF_WRITE																															

Address: *FDMABaseAddress* +
0x9590 (region 0), 0x9610 (region 1), 0x9690 (region 2), 0x9710 (region 3)

Type: R/W

Reset: 0

Description:

FDMA_ESBUF_BOTn MPEG2/H.264 elementary stream buffer bottom address (region n)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESBUF_BOT																											Reserved					

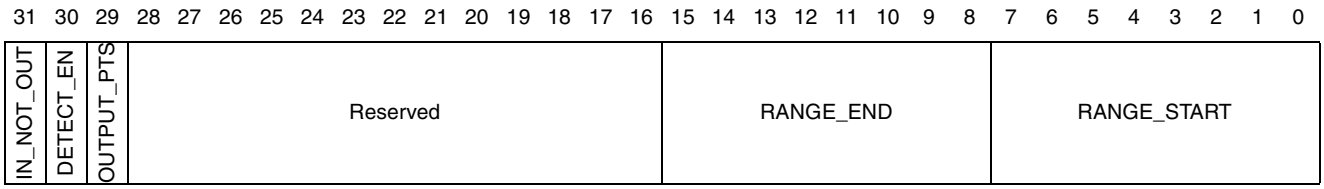
Address: *FDMABaseAddress* +
0x9594 (region 0), 0x9614 (region 1), 0x9694 (region 2), 0x9714 (region 3)

Type: R/W

Reset: 0

Description: ES buffer bottom pointer (32-byte aligned).

FDMA_PES_CTRLn MPEG2/H.264 PES header start code range control (region n)



Address: *FDMABaseAddress* + 0x9598 (region 0), 0x9618 (region 1), 0x9698 (region 2)

Type: R/W

Reset: 0

Description:

[31] **IN_NOT_OUT**: Range mode
0: Detect start codes outside this range 1: Detect start codes within this range

[30] **DETECT_EN**: Enable start code detection for this range
0: No detection 1: Detect

[29] **OUTPUT_PTS**: Output PTS to start code list
0: Disable 1: Enable PTS output

[28:16] **Reserved**

[15:8] **RANGE_END**: End of PES start code range[a]

[7:0] **RANGE_START**: Start of PES start code range[a]

a. RANGE_END >= RANGE_START

FDMA_SC1_CTRLn MPEG2/H.264 start code range control (region n)

31	30	29	28	27 … 16	15 … 8	7 … 0
IN_NOT_OUT	DETECT_EN	ONESHOT_MODE	H264_SCD	Reserved	RANGE_END	RANGE_START

Address: 0x959C (region 0), 0x961C (region 1), 0x969C (region 2)

Type: R/W

Reset: 0

Description:

[31] **IN_NOT_OUT**: Range mode
0: Detect start codes outside this range 1: Detect start codes within this range

[30] **DETECT_EN**: Enable start code detection for this range
0: No detection 1: Detect

[29] **ONESHOT_MODE**: Enable one shot mode
0: One shot mode disabled 1: One shot mode enabled

[28] **H264_SCD**: H.264 Mode SCD
0: MPEG2 start code detect 1: H.264 start code detect

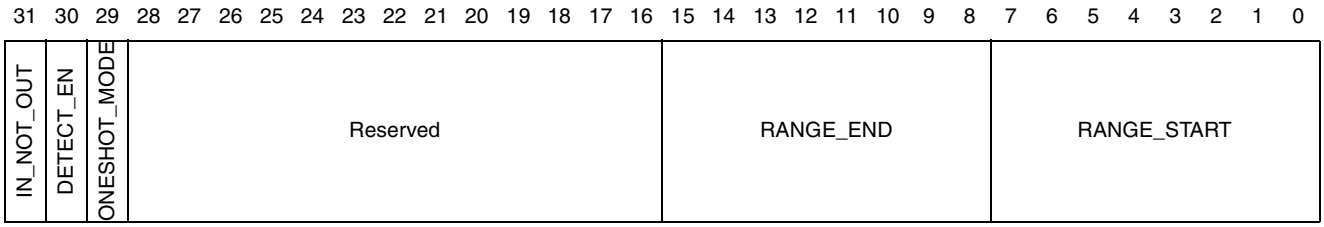
[27:16] **Reserved**

[15:8] **RANGE_END**: End of start code range (inclusive)[a]

[7:0] **RANGE_START**: Start of PES start code rang

a. RANGE_END >= RANGE_START

FDMA_SC2_CTRLn MPEG2 start code range control (region n)



Address: 0x95A0 (region 0), 0x9620 (region 1), 0x96A0 (region 2)

Type: R/W

Reset: 0

Description:

[31] **IN_NOT_OUT**: Range mode
0: Detect start codes outside this range 1: Detect start codes within this range

[30] **DETECT_EN**: Enable start code detection for this range
0: No detection 1: Detect

[29] **ONESHOT_MODE**: Enable one shot mode
0: One shot mode disabled 1: One shot mode enabled

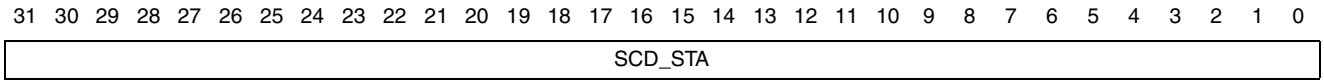
[28:16] **Reserved**

[15:8] **RANGE_END**: End of start code range (inclusive)[a]

[7:0] **RANGE_START**: Start of start code range (inclusive)[a]

a. RANGE_END >= RANGE_START

FDMA_SCD_STAn Start code detector state (region n)



Address: 0x95A4 to 0x95BF (region 0), 0x9624 to 0x963F (region 1), 0x96A4 to 0x95BF (region 2)

Type: R/W

Reset: 0

Description: Must be set to 0 when a new stream is started on this additional data area.
Start code entries

35.7.2 Start code entries

FDMA_SC_TYPE Type of data in this entry

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TAG								Reserved						TYPE	

Address: *MemoryOffset* + 0x0000
Type: R/W
Reset: Undefined
Description:

[31:16] **Reserved**

[15:8] **TAG**: Copy of TAG value from node

[7:2] **Reserved**

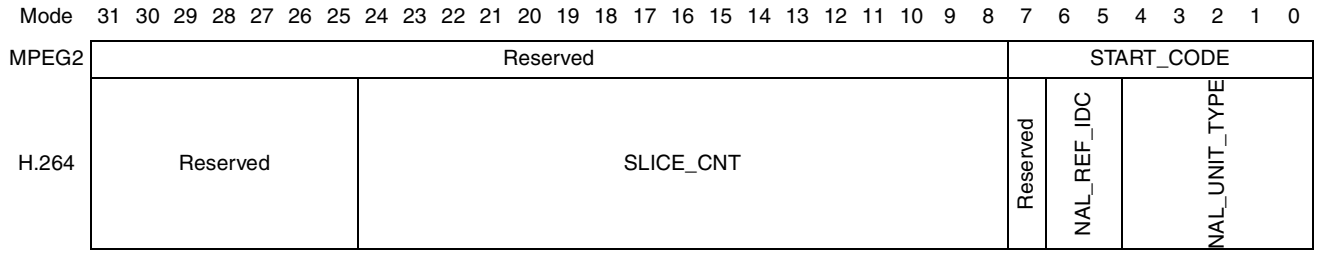
[1:0] **TYPE**: Structure type
00: Start code
01: PTS
10: Reserved
11: Reserved

FDMA_SC_ADDR Memory address of start code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

Address: *MemoryOffset*+ 0x0004
Type: R/W
Reset: Undefined
Description: Memory address of start code in ES.

FDMA_SC_VAL Start code value



Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description:

MPEG-2 mode:

[31:8] **Reserved**

[7:0] **START_CODE**

H.264 mode:

[31:25] **Reserved**

[24:8] **SLICE_CNT**: Number of slices in previous picture

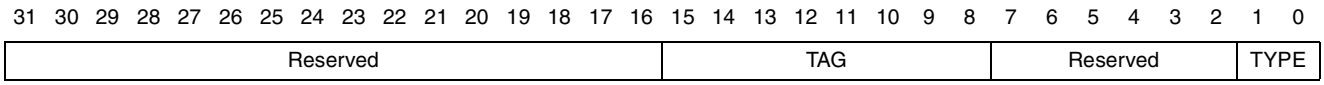
[7] **Reserved**

[6:5] **NAL_REF_IDC**: Field from start code

[4:0] **NAL_UNIT_TYPE**: Field from start code
See Section *H.264 mode start code detect on page 288*

35.7.3 PTS entries

FDMA_PTS_TYPE Type of data in this entry



Address: *MemoryOffset* + 0x0000
Type: R/W
Reset: Undefined
Description:

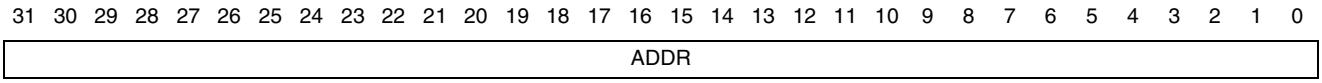
[31:16] **Reserved**

[15:8] **TAG**: Copy of TAG value from node

[7:2] **Reserved**

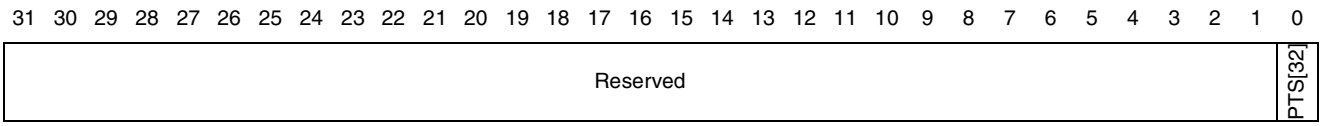
[1:0] **TYPE**: Structure type
00: Start code
01: PTS
10: Reserved
11: Reserved

FDMA_PTS_ADDR Memory address of PTS



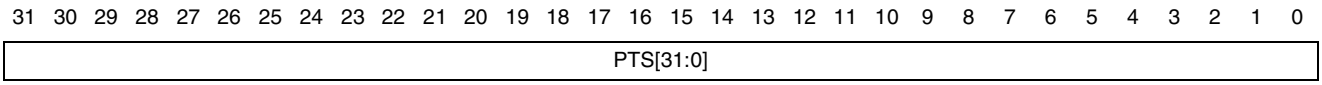
Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description: Address where PTS would appear in ES.

FDMA_PTS_HI MSB of PTS value



Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: Bit 32 of program transport stream (PTS).

FDMA_PTS_LO LSB of PTS value



Address: *MemoryOffset* + 0x000C
Type: R/W
Reset: Undefined
Description: Bits 0 to 31 of PTS.

Part 5

Transport interface

36 Programmable transport interface (PTI)

36.1 Overview

The PTI is a dedicated transport engine. It contains its own CPU and handles the transport deMUX functionality of the set-top box.

The PTI maintains an internal system time clock timer (STC) which keeps track of the encoder clock. The STC is clocked off PIX_CLK.PIX_CLK is controlled by clock recovery software running on the ST40 processor such that it is locked on to the encoder's clock.

The ST40 sets up the PTI and loads its program through the STBus interface. On reception of a new transport packet, the PTI writes its content to the ST40 memory space through the interface.

The PTI module parses and demultiplexes the transport stream, using a mixture of hardware and software running on an application-specific processor called the transport controller (TC). The TC gives the PTI the level of flexibility normally associated with software based demultiplexing of transport streams without the overhead of this processing being placed on the ST40 CPU.

The PTI is configured by registers and programmed by two blocks of static shared memory contained within the PTI, one block containing instructions and the other data. The data block contains structures shared with the ST40 CPU plus structures private to the TC. Code for the TC is downloaded into the PTI instruction memory by a PTI software driver running on the ST40.

The functionality of the PTI is therefore defined by a combination of the PTI hardware, the software running on the TC, and the software driver running on the ST40. This arrangement allows great flexibility by changing the code to be run. Many parameters of the code are modified to change the behavior and features of the PTI. The TC code and PTI driver software are provided by STMicroelectronics. Different versions of these software components are available, with support for generic MPEG-2/DVB transport stream parsing, descrambling and demultiplexing.

Specific details of the data structures and mechanisms used to communicate between the TC and the PTI driver running on the ST40 are contained in the documentation for these software components.

PTI operation is controlled by a software API supplied by STMicroelectronics.

The remainder of this chapter is a description of the hardware components of the PTI and the features and operation implemented by ST software.

36.2 PTI functions

The PTI provides great flexibility, since many features are implemented in either hardware, software or a combination of the two. What follows is a description of the features of the PTI when running the first version of the generic DVB code.

The PTI hardware performs the following functions:

- serial and parallel interface for transport stream input,
- framing of transport packets (sync byte detection),
- section filtering up to 96 x 8 bytes or 48 x 16 bytes using three programmable filtering modes,
- CRC checking of sections (CRC32) and PES data (CRC16),
- DMA and buffering of streams in circular buffers in memory.

This behavior is changed with TC software:

- DMA of three data streams to audio and video MPEG decoders via buffers in memory,
- DMA support for block moves,
- fast search for packet ID (PID) using dedicated hardware engine,
- time stamp checking in two formats.

Software extends the hardware's capability:

- PID filtering of more than 48 PIDs,
- adaptation field parsing - PCR detection and time stamping,
- special purpose section filters allowing total flexibility in processing transport streams,
- demultiplexing of transport stream by PID, supported by hardware,
- communication to ST40 CPU of buffer state.

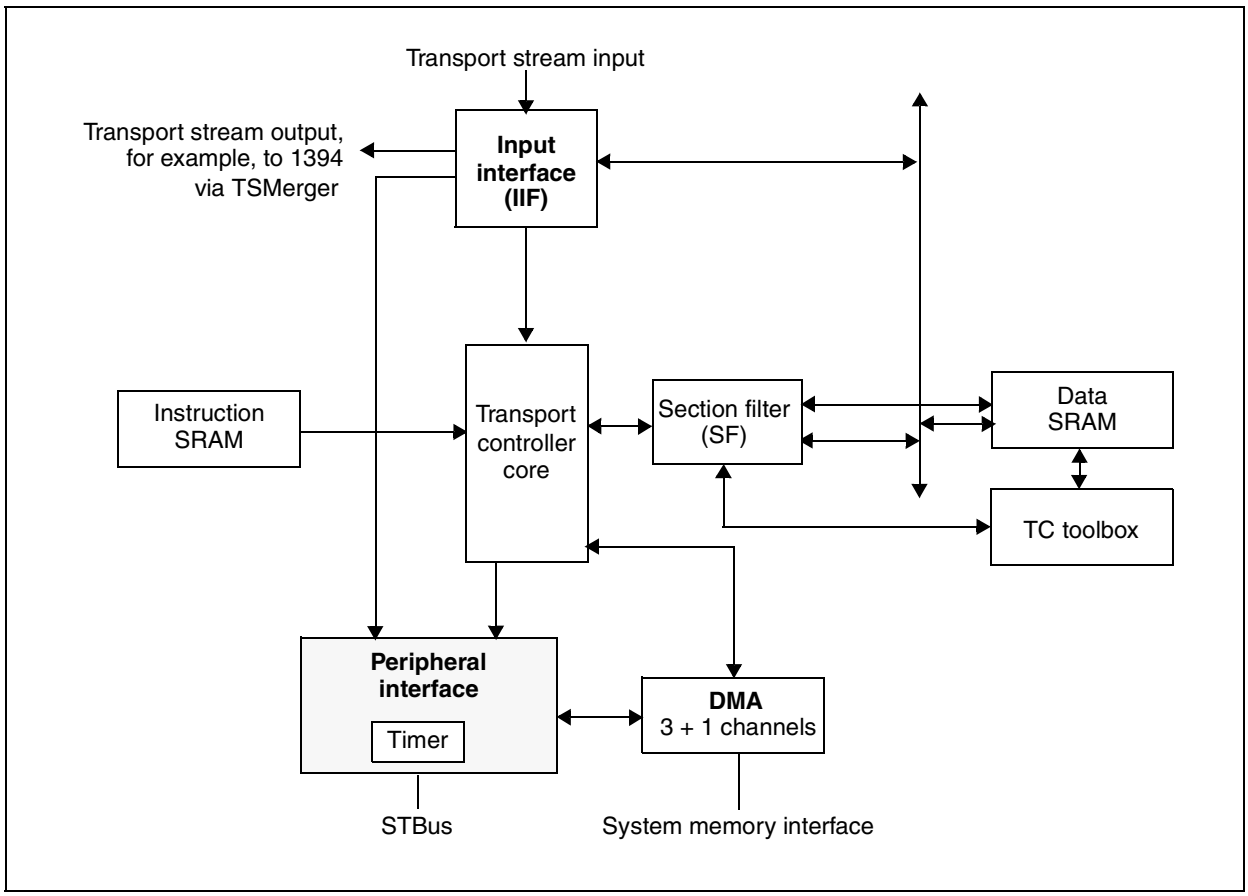
In addition to these transport device functions, the interface copies the entire transport stream or transport packets with selected PIDs from the transport stream through to the PTI output stream interface.

Details of how to control these features are contained in the PTI application programming interface (API) and the PTI software documentation for the particular version of the PTI code.

36.3 PTI architecture

The PTI consists of the TC containing the TC core, input interface (IIF), DMA, and peripheral interface blocks. This is illustrated in Figure 76.

Figure 76: Programmable transport interface architecture



36.3.1 Transport controller (TC)

The TC incorporates the TC core which implements the transport deMUX software. The TC core is a simple 16-bit RISC CPU which uses hardware accelerators to implement PID searching (SE) and section filtering (SF). There is also hardware support for CRC checking.

The TC is responsible for parsing transport packets delivered via the IIF, and controls the entire PTI operation via memory mapped registers. Processed transport data is passed out from the TC to the dedicated DMAs where it is written to ST40 memory via the STBus.

Search engine (SE)

The SE performs a table look-up on the PID to determine whether the current packet is required. It supports searching on up to 48 PIDs and also implements range-checking.

Section filter (SF)

The SF is a hardware accelerator block dedicated to the section filtering task. It supports a number of standard section filtering schemes (short matching mode, long matching mode, positive negative mode) and any software based filtering scheme programmed using the PTI_SFFILTERMASK, PTI_SFFILTERDATA and PTI_SFNOTMATCH registers. Standard filtering is implemented using a double RAM-based CAM.

The SF operates in two modes: automatic or manual.

- **Automatic mode**
  The SF takes over the entire filtering process, reading data directly from the TC core input register and writing directly to the output register. The TC merely reads the packet header, sets up the SF with the appropriate configuration, and sets it going. Breakpoints are allowed in the process to allow the TC to intervene and customize the filtering algorithm at specific points, allowing maximum flexibility at top performance (sometimes called semiautomatic mode).
- **Manual mode**
  The TC core is in overall control of the process but uses the dedicated SF to perform specific filtering tasks.

36.3.2 Shared memory

The PTI contains 6.5 Kbytes of data memory, which is accessed as 32-bit words by the ST40 CPU and as 16-bit words by the TC. This memory is used to hold the private data structures of the TC and data shared between the two processors.

The 9 Kbyte instruction memory holds the instructions that the TC executes. It is accessed as 32-bits wide by both the TC and the ST40 CPU, and is loaded with code by the ST40 before enabling the TC. The ST40 cannot access the instruction memory while the TC is executing.

36.3.3 Input interface (IIF)

The IIF provides the TC with a stream of transport packets for parsing. It also allows data to be copied to an output port via TSMerger, for example, one supporting the IEEE 1394 protocol.

The IIF is responsible for inputting the synchronous data stream to the PTI and passing data to the TC for processing. The start of a packet is detected either from the incoming packet clock or by sync byte detection. Under the control of TC software, the IIF routes data to the TC input register via the IIF's H- or header FIFO.

The IIF, like other DMA modules, is controlled by registers. Parameters programmed in these registers include:

- operating conditions for sync byte detection,
- packet length up to 256 bytes long (default 188),
- data transfer parameters, for example, number of bytes to be passed to the TC or whether descrambling is required,
- alternate output parameters.

36.3.4 DMA

The PTI's DMA block contains four DMA channels. Channel 0 is the PTI's main DMA write channel used for transferring demultilpexed payloads to memory buffers under the control of the TC.

Channels 1 to 3 are general purpose paced DMA channels used for memory to memory and memory to device transfers. These DMA channels can be used in a circular buffer configuration if required combined with DMA channel 0. (see Section 36.6: *DMA operation*). A programmable delay (the holdoff time) is built into the data transfer to ensure all data has reached its target before the request signal is read for the next transfer.Also, a programmable write length ensures that all data has storage room in the target upon arrival.

Channel 0 outputs single words or 4-, 8- or 16-word bursts. It may also be configured to output data directly to the decoders. Channels 1 to 3 are normally set to write whole words but are configurable to write to byte-wide devices.

The DMA block is controlled by a number of registers which are programmed by the TC software. Functions controlled by these registers include:

- setting up and manipulating the circular data buffers,
- configuring write channel (channel 0) operation,
- configuring channels 1 to 3 to write to the decoders (or memory),
- channel 0 status,
- memory map selection (programming mode).

36.3.5 PTI DMA channel usage

In the STx7100, the normal A/V data flow for stream decoding uses the PTI DMA Channel 0 to transfer demultiplexed transport stream payloads to memory buffers and then the FDMA is used for performing start code detection, PES parsing and transfers to A/V bit buffers which are accessed by the decoders (see Section 34.4.3: *SCD/PES parsing on page 286*).

PTI DMA channels 1-3 are still available for generic application use if required.

36.3.6 Peripheral interface

The peripheral interface allows the PTI to interact with the ST40 and communicate data between them. It handles all address decoding (for example, DMA writes and interrupts) and allows joint access to the instruction and data SRAMs and configuration registers. It also implements the timer and time stamp functions and soft reset (see *Section 36.4.2: Soft reset on page 320*).

36.3.7 Timer module

The timer module in the PER contains the system time clock (STC) and three time stamp registers:

- packet start time register,
- audio PTS (presentation time stamp) latch register,
- video PTS latch register.

These registers are loaded with the STC value according to the events shown in Table 112.

Table 112: Timer module register update events

Event	Action
Rising edge of packet clock (packet start)	STC is loaded into the packet start time register
Beginning of audio frame output	STC is loaded into the audio PTS register
VSYNC	STC is loaded into the video PTS register

The packet start time register is read by the TC and used to determine the arrival time of a program clock reference (PCR). The CPU cannot directly access this register. TC software stores this arrival time together with the PCR in shared data SRAM so it can be read by the CPU.

The audio and video PTS registers are read directly by the CPU and used by driver software to synchronize the audio and video. Each register consists of two words to accommodate 33-bit time stamp values, one word holds the lower 32 bits and the other holds the most significant bit.

36.4 PTI operation

The programmable transport interface (PTI) performs the transport parsing and processing functions without intervention of the ST40 CPU. The block is controlled by the ST40 and communicates with it via a shared data SRAM block local to the PTI, registers in the PTI, data structures in the ST40 memory written by the PTI and an interrupt from the PTI.

The shared data SRAM is used to hold the main data structures for the PTI including:

- PID values,
- descrambler keys for each PID,
- control bits for each PID to set up DMA parameters, to mark the PCR PID, to control section CRC checking, and to mark PIDs which need copying to the selective transport output interface,
- PID state information such as a transport or PES level descrambling flag, partial sections for filtering, partial section CRC values, and current continuity count values,
- descriptors and pointers to the circular buffers where the streams from each PID are sent,
- the last adaptation field and its time stamp from the local system clock.

Registers are provided to allow the ST40 CPU to initialize and control the block and to provide interrupt status and control.

36.4.1 Initialization

After device reset, the TC in the PTI is halted and the PTI block remains idle. It stays in this state until:

- the TC code is loaded into the instruction SRAM by the ST40,
- initialization is performed as described below,
- the TC is enabled by setting the TC_EN bit of the PTI_TCMODE register high.

There are a number of initialization steps that must be performed before the TC is enabled.

1. The data SRAM must be initialized with any data structures required by the TC software.
2. The interrupt status registers must be cleared.
3. The IIFFIFO_EN register bit must be set high to enable the input FIFO.

36.4.2 Soft reset

A soft reset feature is available on the PTI by setting a bit in the TC configuration register. The DMA should be flushed using register PTI_DMAFLUSH before the reset is initiated to ensure all outstanding signals on the bus are cleared.

36.4.3 Typical operation

When the TC is running, the software waits for a transport packet to arrive. Having detected the start of a packet, the TC code then sets up the PTI hardware search engine to perform PID filtering. The search engine searches a contiguous block of PID values within data SRAM for a match with the PID in the incoming transport packet header. If the packet is to be rejected, the software discards the packet and waits for the next packet to arrive. If the packet is one to be processed, the TC examines the data structure for the PID in the data SRAM to determine what other processing is required. The type of transport packet is recorded in this data structure by the ST40. The PTI driver sets variables in this data structure to configure the TC software to perform various tasks.

Typical tasks might be:

- descrambling at the transport or the PES level with a key pair,
- section filtering, with a set of filters and section CRC checking for streams containing sections,
- directing the output stream either to a circular buffer in memory or to a compressed data FIFO of an audio or video decoder,
- enabling or disabling a stream,
- appending or indexing extra information for further data processing by the ST40.

Having examined the PID data structures, the TC sets up the rest of the hardware in the PTI to perform the required descrambling and DMA operations before starting to parse the rest of the packet. Processing varies depending on the contents of the transport packet, which includes:

- PES data,
- section data,
- adaptation fields,
- continuity count fields,
- time stamp.

Typical processing for different packet types and fields is described in the rest of this section.

PES data

Transport packets which contain PES data and are not rejected by PID filtering, are CRC checked and descrambled if required. The PES data is DMA transferred into a circular buffer. The DMA features of the PTI buffer a PES stream in memory and then transfer the data to a decoder without the CPU being involved. Optionally an interrupt is generated to the ST40 when the buffer for a PES stream has data added to it and the state of the buffer changes from empty to nonempty. An interrupt is raised and an error flag set in the data SRAM if the buffer overflows. In such cases, the most recent data is lost.

Section data

Transport packets which contain section data and are not rejected by PID filtering, are subjected to section filtering on each section or partial section in the packet.

The PTI contains a hardware section filter which implements two standard filter modes:

- short match mode (SMM): 96 filters of 8 bytes each,
- long match mode (LMM): 48 filters of 16 bytes each.

Positive/negative matching mode is also supported.

Section filtering is highly flexible. Any subset of the filters, including all or none, are applied to any PID, and filtering modes are mixed within an application.

When a section passes the filtering, the complete section is written to the ST40 memory space, either to a circular buffer or to defined locations for a set of sections.

The PTI hardware also automatically detects (using the section syntax indicator bit) if CRC has been applied to the section, and performs CRC checking if required. If the CRC check fails, the TC software removes the incorrect section from the section buffer, discards the current PID, and waits until a new packet arrives (detected by the unit start indicator). CRC checking is enabled or disabled, and the TC can also be programmed to keep a corrupt section.

The TC software uses a bit in the interrupt status registers to raise an interrupt for the ST40 signalling a buffer having a section placed in it.

There are no restrictions on any of:

- the alignment of the sections in transport packets,
- the lengths of sections other than those in the MPEG-2 standard,
- the numbers of sections in a packet when filtering standard sections.

Section filtering is implemented by a mixture of TC code with the hardware section filter. Alternatively, it may be performed purely in TC code to implement a small number of longer or special purpose filters. In this case there may be some restrictions on the minimum length of a section or the number per transport packet, to ensure that the processing is performed within the period of one transport packet interval.

Adaptation fields

Typically only the program counter reference (PCR) would be extracted from this field although the TC software could extract other data.

If a PID is flagged as the source for PCR values then any adaptation field in a transport packet with this PID containing a PCR has the PCR value extracted and stored in the data SRAM. The value is stored with a time stamp, which is the time when the transport packet arrived, as given by the system time clock (STC) counter value. An interrupt is raised to the ST40 and the interrupt bit itself is used as a handshake for the processing of the PCR by the ST40. Until the bit is cleared, no more PCRs are captured.

The STC counter is clocked by the 27 MHz input clock to the device and is initialized by the ST40 CPU.

Continuity count field

The TC software uses this field to check for missing transport packets. If a continuity count error is detected, the software discards any partial units of data, such as a partially complete section, and search for a new data unit starting point.

36.4.4 Direct output of transport data

Concurrently with the parsing, processing, and DMA transfer of the transport packets, some or all of the transport stream can be copied to an external interface, for example, an external IEEE 1394 controller. The ST40 software directs specific packets to be output, with or without descrambling. Modifications of the transport stream, including modification of tables and substitution of packets, are possible by suitable programming of the TC.

36.5 Interrupt handling

The PTI may interrupt the ST40 under a number of conditions, for example when a DMA operation changes a buffer from being empty to nonempty, or in the case of an error condition. The interrupt is generated by the TC writing into one of the 64 interrupt status register bits. These bits are ORed together to produce one interrupt and fed to the interrupt controller via the interrupt level controller. The ST40 CPU then uses this interrupt to schedule a process to deal with this condition.

The TC sets a given interrupt by writing to the appropriate bit position in the interrupt status register and the ST40 resets interrupts by writing to the appropriate bit position in the appropriate interrupt acknowledge register. The ST40 determines which interrupts are currently set by reading the appropriate interrupt status register.

The ST40 enables or disables interrupts by writing 1 or 0 into the appropriate bit position in the appropriate interrupt enable register. Using this mechanism, the ST40 processes a buffer until a read to the write pointer (held in the data SRAM) shows the buffer to be empty. The process then clears the status bit which corresponds to that buffer by writing 1 to the corresponding interrupt acknowledge register bit.

The detection of the empty condition on a buffer and the acknowledgement of an interrupt does not lock out the TC from writing the write pointer after the ST40 has checked, and setting the interrupt status bit before the ST40 has acknowledged. The buffer state must be reconfirmed before waiting on a semaphore for that buffer. Rechecking the write pointer avoids data being left in the buffer until the next data arrives and the TC sets the interrupt again. If the buffer is still empty then the ST40 process enables the interrupt by setting the correct interrupt enable bit before making the process wait on a semaphore.

Figure 77 shows the TC and the ST40 processes and mechanism described above.

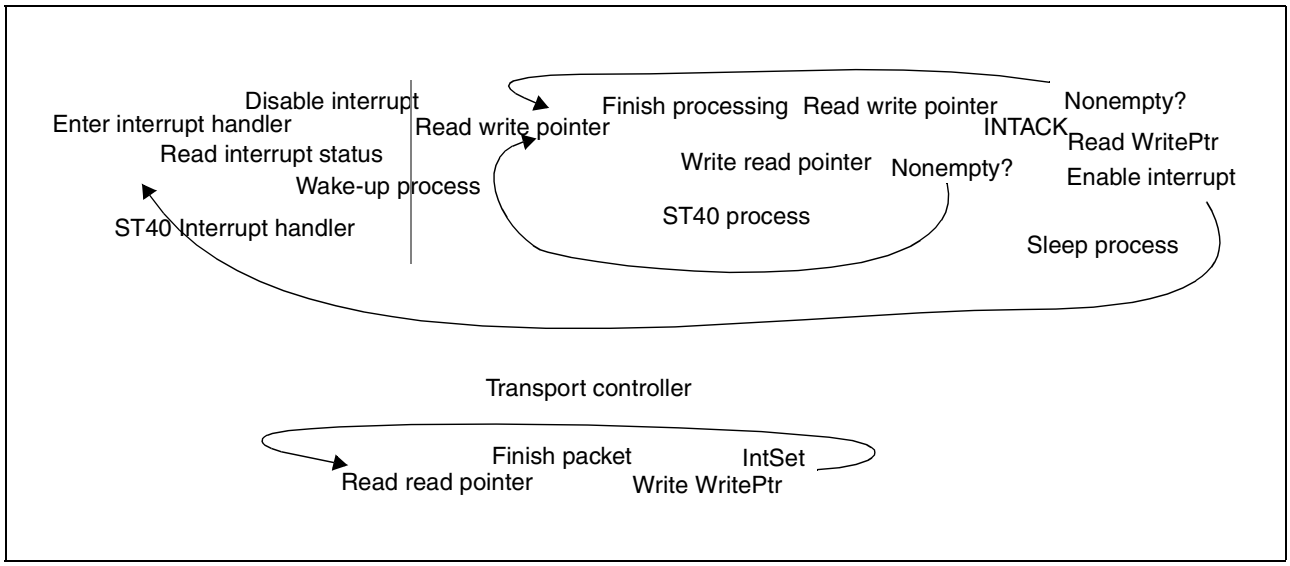
Note: At any given time each process is at any point during the critical regions of code. There is no implied timing for each step of a process only an ordering of steps.

After the buffer has been refilled the TC sets the interrupt status bit causing the PTI interrupt handler to be run. When the interrupt handler finds the buffer process semaphore status bit is set then the interrupt handler signals to the semaphore to restart the process and disable that interrupt bit. Therefore, the process itself disables the interrupt at the PTI level, and only enables it when it is about to sleep.

An error condition would be handled in a similar manner.

The association of interrupt bits with particular conditions and events is determined by the TC code and the corresponding PTI driver running on the ST40 CPU.

Figure 77: PTI buffer interrupt handling



36.6 DMA operation

The PTI contains a four channel DMA controller which is programmed by the TC and the ST40 CPU. The channels are used to write or read data to or from circular buffers defined by a base and a top pointer. For each channel there is also a read and a write byte pointer.

A separate unit called the PDES (programmable descrambler) is responsible for descrambling transport streams. This is described in Chapter 75 on page 899. The PTI has a direct interface with the PDES and is under the control of the TC. Packets to be descrambled are passed directly to the PDES and received back after descrambling for further processing if required.

The TC uses DMA channel 0, a write only DMA, to send the data from a transport packet to a circular buffer in the ST40 memory space, or when required, directly to a decoder mapped on to a fixed memory location. Since the TC reloads the registers of channel 0 at the start of processing each transport packet, this allows payloads from transport packets with different PIDs to be output to different buffers. It also enables the TC code to support output buffers with data structures other than circular buffers.

Channel 0 is configured to write data directly to the decoders by specifying the address to be written to in the appropriate buffer registers. When using this feature channel 0 should be set to holdoff mode.

It is also possible to configure channel 0 to ignore the decoder ready signal. This feature should be used with caution as it allows data to be written without flow control.

Channels 1 to 3 can be used to read from circular buffers and write the data to a device such as an audio or video decoder mapped to a fixed address in ST40 memory, in response to a request signal. Each time the request signal is active (low) and there is data to be read from the buffer (that is, the read pointer is not equal to the write pointer), a programmable number of bytes is transferred. This is followed by a hold-off time during which the request is not sampled, allowing the request signal time to become valid again. In the STx7100, the FDMA is typically instead used for transferring A/V data to the decoder bit buffers performing PES parsing and start code detection along the way (see section Section 34.4.3: *SCD/PES parsing on page 286*)

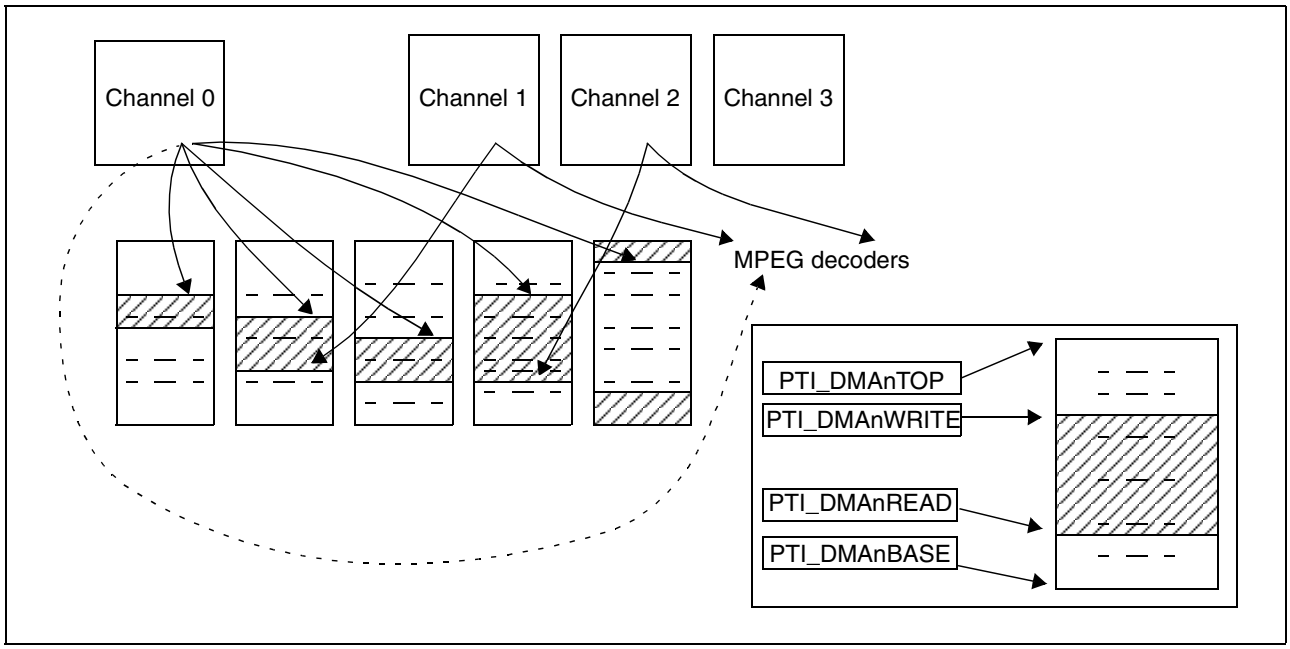
Block moves are supported on channels 1 to 3 using a special DMA mechanism which increments the input address after each write.

During operation the read and write pointers are examined by the hardware to determine if there is data in the buffers to be transferred. If there is data in the buffer and the request line for that channel is active then the data is transferred and the read pointer updated. If channel 0 DMA is writing into the buffer then the write pointer is updated by the TC; otherwise the ST40 CPU updates the write pointer after adding data to a buffer.

Once each transport packet has been output on channel 0, the DMA write pointer of the corresponding buffer data structure in the PTI data SRAM is updated. If the transport packet did not contain the end of a complete data unit such as a section, a temporary write pointer variable is used. This is done so that the ST40 process only sees a complete unit of data to be processed. The temporary write pointer is available for reading by the TC software in a special register. When the data unit is complete, the write pointer used by the ST40 process is updated and an interrupt is set to signal to the ST40 process that data is in that buffer. This mechanism of updating the write pointers and interrupting, is not used in the special case that the buffers are being transferred by DMA to an audio, video or other decoder.

Channels 1 to 3 have the write pointers updated either by the TC software after data has been placed in the corresponding buffer by DMA channel 0, or, by the ST40 CPU if this is writing data into the buffer that DMA channels 1 to 3 are reading from.

Figure 78: DMA channels



36.6.1 Circular buffers

The inset in Figure 78 shows how the buffer pointer registers are used to implement circular buffers for the four DMA channels. The base register points to the base word of the buffer, and must be 16-byte aligned, so bits 0 to 3 must be zero. The top register points to the top byte of the buffer. If a circular buffer is being used then this address must be one byte below a 16-byte aligned address, so bits 0 to 3 must be 1. The buffer for channel 0 only is reduced to a single address by setting the top register equal to the base register. In this case, the data is written to a fixed address defined by the write pointer and the write pointer is not updated.

The read and write buffers point to the next word to be read or written respectively.

At initialization the read and write pointers are set to the same value, so that the buffers are empty. The base and top pointers are initialized to point to the beginning and end of the buffers.

36.6.2 Channel arbitration

Only one of the four DMA channels may have access to the memory bus at any time. To ensure smooth flow of data and to avoid mutual lockout, there is a fair and efficient arbitration scheme between the four DMA channels.

The scheme used is the least recently used (LRU) method, that is, the channel that has not requested an access for the longest time is guaranteed the next access. This ensures that none of the channels locks out the others, and has the advantage that the LRU arbiter does not waste any clock cycles on an inactive channel.

Although channel 0 appears to have no priority over channels 1 to 3 to write incoming transport data, in fact the performance of channel 0 is enhanced because:

- it transfers data four times faster than channels 1 to 3 using four-word bursts,
- it never performs read accesses, which are inherently slower because they cannot be posted ahead.

36.6.3 Flushing the DMA

Before doing a PTI soft reset (see *Section 36.4.2: Soft reset on page 320*), the DMA must be flushed to ensure it restarts cleanly without any outstanding request, grants or valid signals. This is performed using register PTI_DMAFLUSH.

36.6.4 Performance

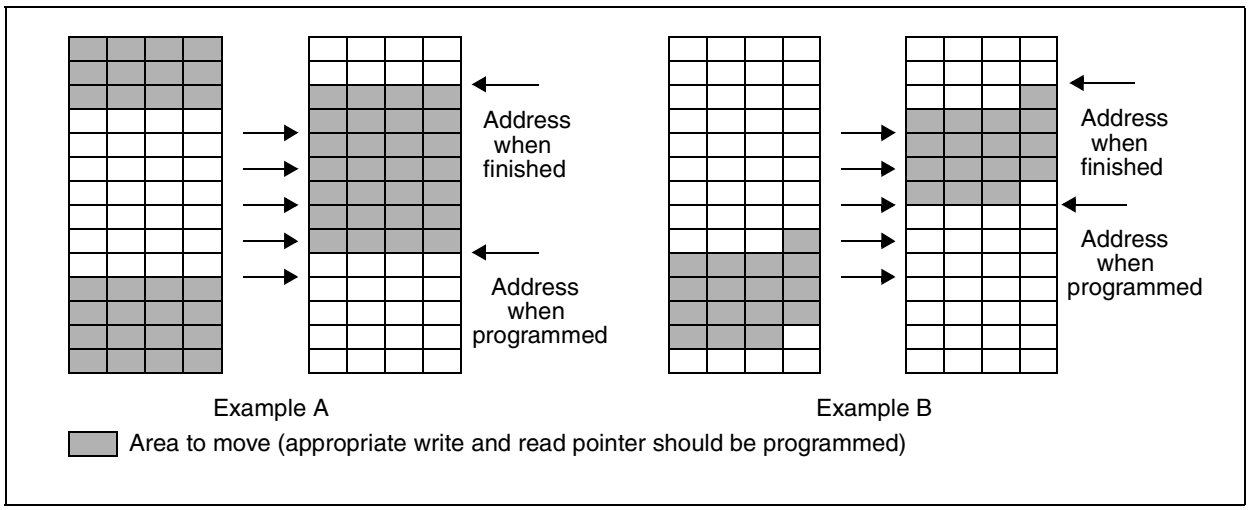
Each DMA channel performs one read, write or burst access when it has been granted by the memory arbiter. When more than one channel is active, one performs an access while another is waiting for valid pulses to come back from the interconnect. So the accesses are interleaved, guaranteeing a high performance.

Performance is enhanced by being able to post writes to the STBus that is, a read or write is initiated without waiting for a valid signal on the previous access. Reads cannot be posted since the next operation may be a write that depends on the result of the previous read.

36.6.5 Block move

In normal operation DMA channels 1 to 3 read from a circular buffer and write to a fixed address (the memory-mapped address of the decoder). Channels 1 to 3 are also configured to perform a block move by setting a bit in the channels status register. This causes the decoder address to be incremented after each write and effectively performs a circular- to- linear blockmove function. This is illustrated in *Figure 79*.

Figure 79: DMA blockmove



36.7 Section filter

The section filter in the PTI hardware and TC software parses the section information in an MPEG-2 transport stream packet. Sections that pass the filter are transferred via the channel 0 DMA to ST40 memory. These sections have a fixed format and are defined by the MPEG-2 systems specification[1].

The data sections arrive at a faster rate than the system processes them, so a filter selects only those sections that are required and thus reduces the required processing rate. In addition, the sections that are used to construct tables are repeated regularly, so it is possible to build up an information table by capturing a proportion of them using one set of values in the filters, and then capturing the remainder of the table by setting the filters up to select the missing sections.

The hardware filter system looks for a match to the programmed filters, for example, using short match mode the match would be to a total of 96 filters of 8 bytes each. Each bit of each of the filters may be individually masked, so that no comparison is performed on that bit of the filter. In addition to the filtering operation the PTI performs CRC checking on the sections which match a filter.

1. *Generic Coding Of Moving Pictures And Associated Audio: Systems, Recommendation H.222.0*, ISO/IEC 13818-1

The CRC result is programmed to be acted upon for:

- all sections,
- no sections,
- only those sections that have the section syntax indicator bit set via bits in the PID data structure read by the TC code.

A 1-byte result report is output when sections are accepted.

A filter mask word in each PID data structure specifies which set of filters is applied to sections in a transport packet with that PID.

Filter matching is programmed by writing a set of masks and filters in the CAM memory. The CAM has two memory cores named CAMA and CAMB.

Each even four-byte word memory address of the CAMA or CAMB memory array is a set of filter bytes, and each odd word address is either a set of filter bytes or mask bytes. A configuration bit is used to enable the masks. With masks enabled the total number of filters and masks available is 48 8-byte or 96 16-byte filter plus mask sets. If masks are not required each CAM has 48 8-byte or 96 16-byte filters.

The filters and masks are arranged in columns, interleaving one layer of filter data and, if masks are enabled, its relevant mask. The first entry to CAMA is a 32-bit word formed by the first byte of each of the first four filters. The first entry to CAMB is a 32-bit word formed by the first byte of each of the fifth to eighth filters. The second entry to the CAMA, when masks are enabled, will be a 32-bit word formed by the first byte of each of the masks of the first four filters and so on.

The position of the first filter in the memory is also programmable. Figure 80 illustrates different examples of CAM configuration.

Figure 80: Memory organization when CAM_CFG NOT_MASK bit = 1 (masks disabled).

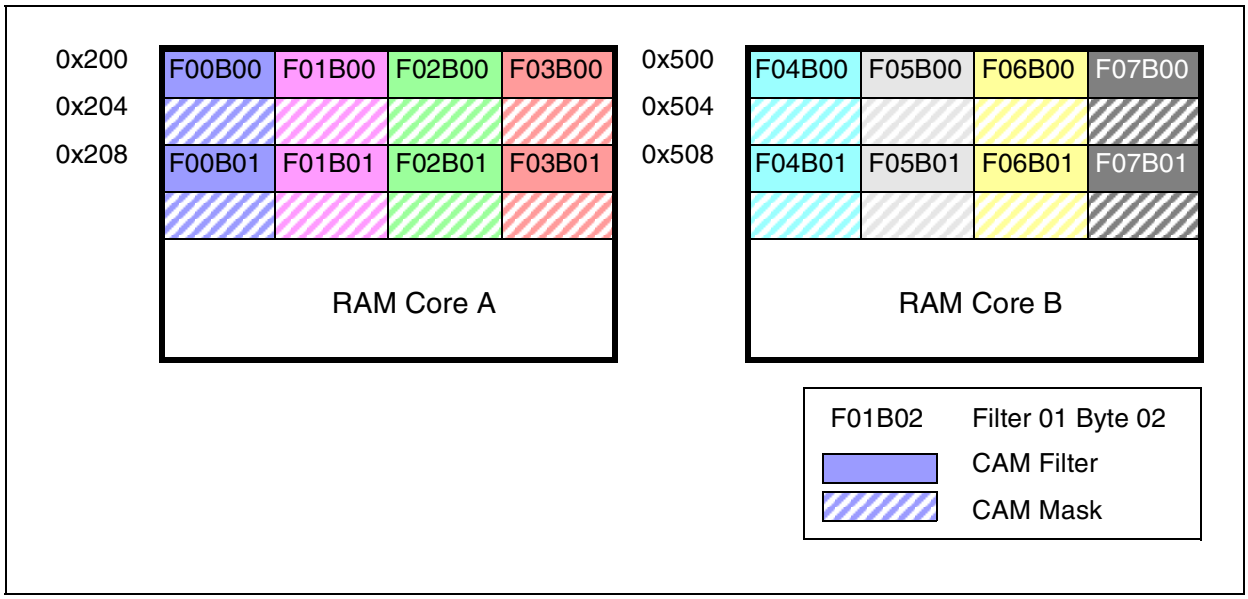


Figure 81: Memory organization with CAM_CFG NOT_MASK bit = 0 (masks enabled)

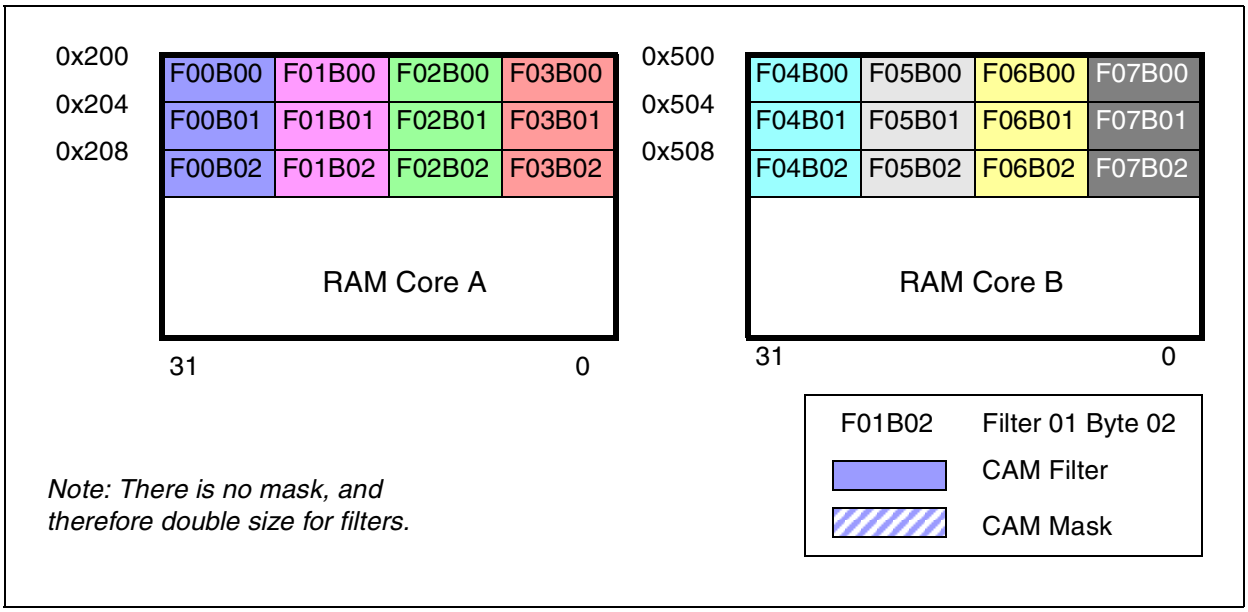


Figure 82: Memory organization for 16 filters

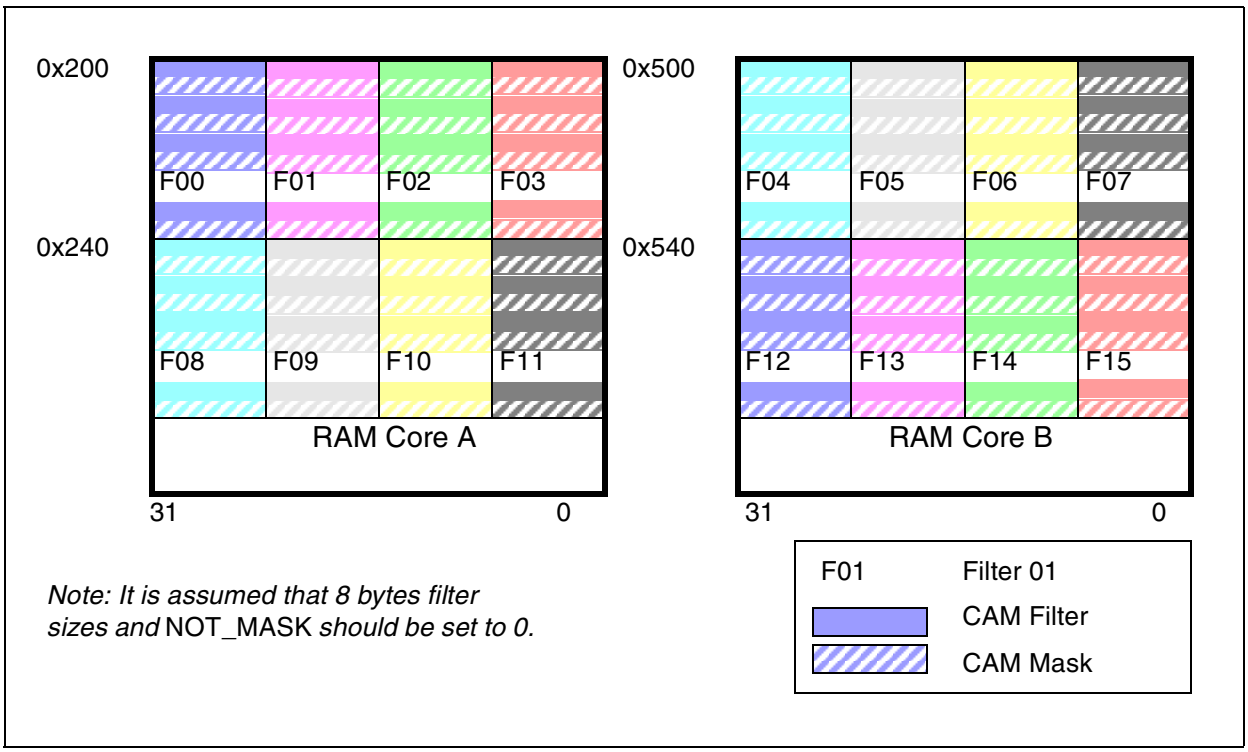
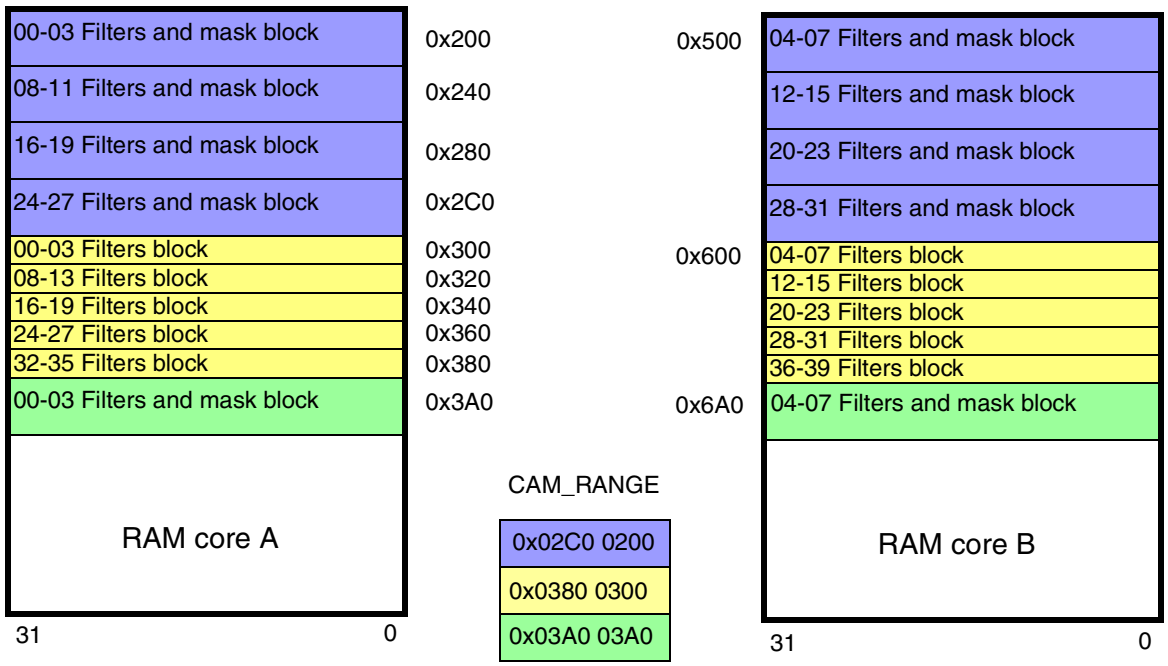


Figure 83: Memory organization when different and mixed cases.



Note: This example has filters for three different types of streams. The second (40 filters in yellow) without mask. It has been assumed (see the memory address) that the filter for all the cases are 8 bytes filters. To select between the different types, the CAM_RANGE *register should be as shown over these lines. It shows the expected programed values to use each of the sets in different contexts.*

36.7.1 Automatic and manual modes

The SF operates in automatic or manual mode.

Automatic mode

In automatic mode the entire filtering process is performed in the SF with no intervention by the ST40. The ST40 processes the packet header, sets up the SF with the required configuration, and starts the SF engine. The SF then reads data directly from the TC input register and parses the entire payload, outputting section data to the TC output register.

To allow the TC to intervene and customize automatic filtering, breakpoints are set at the following events:

- after filtering any section (wanted or unwanted),
- after a wanted section has been output.

This enables the TC to add extra filtering, recover unwanted sections, or intervene between sections.

Breakpoints are enabled by setting the appropriate bits in the SF configuration register.

Manual mode

In manual mode the TC core controls the filtering process, writing section data in order to the SF header registers and reading the results from the SF, effectively using the section filtering and CRC modules as special purpose hardware engines.

36.7.2 Filtering modes

Standard filtering modes supported are:

- short match mode (SMM): matches on CAM A or CAM B,
- long match mode (LMM): matches on CAM A line n and CAM B line n,
- positive/negative matching mode: matches on CAM A line n and not CAM B line n,

Standard filtering is performed by means of two CAM 32-bit wide memory cores (A and B), the outputs of which are ANDed to give the final result. CAM A has an additional register (PTI_SFNOTMATCH) containing five bits of data and one bit to enable not matching. CAM lines are enabled or disabled using mask registers. Almost any CAM-based filtering match over 18 bytes can be programmed. The mapping of data to CAM A and CAM B for a single filter on each of these modes is shown in Table 113 to Table 115.

PTI_SFDATAn has to be arranged inside CAM memory space as in Section 36.7: *Section filter on page 326*

Table 113: SMM mapping

Section bytes	CAM A	CAM B
5[5:1][a]	PTI_SFNOTMATCHn[4:0]	Not used
0	PTI_SFDATAn[63:56]	PTI_SFDATAn[63:56]
1	Not used	
2		
3	PTI_SFDATAn[55:48]	PTI_SFDATAn[55:48]
4	PTI_SFDATAn[47:40]	PTI_SFDATAn[47:40]
5	PTI_SFDATAn[39:32]	PTI_SFDATAn[39:32]
6	PTI_SFDATAn[31:24]	PTI_SFDATAn[31:24]
7	PTI_SFDATAn[23:16]	PTI_SFDATAn[23:16]
8	PTI_SFDATAn[15:8]	PTI_SFDATAn[15:8]
9	PTI_SFDATAn[7:0]	PTI_SFDATAn[7:0]
10	Not used	
11		
12		
13		
14		
15		
16		
17		

a. Used when bit ENn of register PTI_SFNOTMATCHn = 1

Table 114: LMM mapping

Section bytes	CAM A	CAM B
5[5:1][a]	PTI_SFNOTMATCHn[4:0]	Not used
0	PTI_SFDATAn[63:56]	Not used
1	Not used	
2		

Table 114: LMM mapping

Section bytes	CAM A	CAM B
3	PTI_SFDATAn[55:48]	Not used
4	PTI_SFDATAn[47:40]	
5	PTI_SFDATAn[39:32]	
6	PTI_SFDATAn[31:24]	
7	PTI_SFDATAn[23:16]	
8	PTI_SFDATAn[15:8]	
9	PTI_SFDATAn[7:0]	
10	Not used	PTI_SFDATAn[63:56]
11		PTI_SFDATAn[55:48]
12		PTI_SFDATAn[47:40]
13		PTI_SFDATAn[39:32]
14		PTI_SFDATAn[31:24]
15		PTI_SFDATAn[23:16]
16		PTI_SFDATAn[15:8]
17		PTI_SFDATAn[7:0]

a. Used when bit ENn of register PTI_SFNOTMATCHn = 1

Table 115: Positive/negative matching mode mapping

Section bytes	CAM A	CAM B
5[5:1][a]	PTI_SFNOTMATCHn[4:0]	Not used
0	PTI_SFDATAn[63:56]	PTI_SFDATAn[63:56]
1	Not used	
2		
3	PTI_SFDATAn[55:48]	PTI_SFDATAn[55:48]
4	PTI_SFDATAn[47:40]	PTI_SFDATAn[47:40]
5	PTI_SFDATAn[39:32]	PTI_SFDATAn[39:32]
6	PTI_SFDATAn[31:24]	PTI_SFDATAn[31:24]
7	PTI_SFDATAn[23:16]	PTI_SFDATAn[23:16]
8	PTI_SFDATAn[15:8]	PTI_SFDATAn[15:8]
9	PTI_SFDATAn[7:0]	PTI_SFDATAn[7:0]
10	Not used	
11		
12		
13		
14		
15		
16		
17		

a. Used when bit ENn of register PTI_SFNOTMATCHn = 1

36.7.3 CRC checking

A CRC check is performed after the section has been processed by the section filter. If the check fails the software ignores the corrupt section in the circular buffer and waits for it to be transmitted again. The address of the start of the section is stored in a temporary register, and when the packet is encountered again in the data stream this address is loaded and the section data is directed to the correct circular buffer, where it overwrites the corrupted data.

36.7.4 Section filter registers

The section filter is configured and controlled by the TC software via a set of registers. These perform the following functions:

- section filter set up and configuration: automatic/manual mode; filter mode; CRC type; mask enable; breakpoint enable,
- section filter run,
- section filter header data,
- section filter process data: section filter state, CRC state, DMA write address,
- CAM engine configuration: number of filters in memory, filter length, first filter in memory, prefetch ability,
- CAM matching results,
- CAM mask data,
- CAM not matching results,
- selection of memory map (programming mode).

37 Programmable transport interface (PTI) registers

Register addresses are provided as *PTIBaseAddress* + offset.

The *PTIBaseAddress* is:

0x1923 0000.

There are a few additional transport subsystem registers described in Chapter 21: *System configuration registers* *on page 171*.

Table 116: PTI register summary

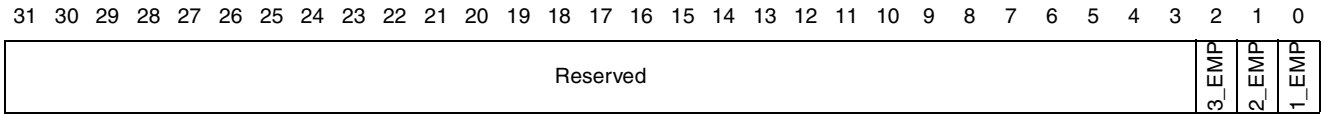
Register	Description	Offset	Type
DMA			
PTI_DMA_EMPTY_STA	DMA empty status	0x0034	RO
PTI_DMA_EMPTY_EN	DMA empty enable	0x0038	R/W
PTI_DMA0_STA	DMA channel 0 status	0x1018	R/W
PTI_DMAn_BASE	DMA buffer base address	0x1000, 0x1020, 0x1040, 0x1060	WO
Reserved	-	0x1014 - 0x1074	-
PTI_DMAn_READ	DMA buffer read address	0x100C - 0x106C	WO, R/W
PTI_DMA0_SETUP	DMA channel 0 byte not word mode and block move	0x1010	R/W
PTI_DMAn_SETUP	DMA channels 1 to 3 byte not word mode and block move	0x1030, 0x1050, 0x1070	R/W
PTI_DMAn_TOP	DMA buffer top address	0x1004 - 0x1064	WO
PTI_DMAn_WRITE	DMA buffer write address	0x1008, 0x1028, 0x1068	R/W, WO
PTI_DMAn_CD_ADDR	Configuration of CD FIFO address for channels 1 to 3	0x1038, 0x1058, 0x1078	WO
PTI_DMA_EN	DMA enable	0x101C	WO
PTI_DMA_FLUSH	Flush ready for a soft reset	0x105C	R/W
Input interface			
PTI_IIF_ALT_FIFO_CNT	IIF alternative FIFO count	0x6004	RO
PTI_IIF_ALT_LATENCY	IIF alternative output latency	0x6010	WO
PTI_IIF_FIFO_CNT	IIF count	0x6000	RO
PTI_IIF_FIFO_EN	IIF FIFO enable	0x6008	R/W
PTI_IIF_SYNC_DROP	IIF sync drop	0x6018	WO
PTI_IIF_SYNC_CLK	IIF sync lock	0x6014	WO
PTI_IIF_SYNC_CFG	IIF sync configuration	0x601C	WO
PTI_IIF_SYNC_PER	IIF sync period	0x6020	WO
PTI_IIF_HFIFO_CNT	IIF header FIFO count	0x6024	RO

Table 116: PTI register summary

Register	Description	Offset	Type
PTI configuration			
PTI_CFG	PTI configuration	0x0058	R/W
PTI_AUD_PTS	Audio presentation time stamp	0x0040, 0x0044	RO
PTI_INT_ACKn	PTI interrupt acknowledgment	0x0020 - 0x002C	WO
PTI_INT_ENn	PTI interrupt enable	0x0010 - 0x001C	R/W
PTI_INT_STAn	PTI interrupt status	0x0000 - 0x000C	RO
PTI_VID_PTS	Video presentation time stamp	0x0048, 0x004C	RO
PTI_STC_TIMER	Set STC timer	0x0050 - 0x0054	WO
Transport controller mode			
PTI_TC_MODE	Transport controller mode	0x0030	R/W

37.1 DMA

PTI_DMA_EMPTY_STA DMA empty status



Address: *PTIBaseAddress* + 0x0034
Type: RO
Reset: 0
Description:

[31:3] **Reserved**

[2] **3_EMP**: DMA channel 3 buffer empty

[1] **2_EMP**: DMA channel 2 buffer empty

[0] **1_EMP**: DMA channel 1 buffer empty

PTI_DMA_EMPTY_EN DMA empty enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													3_IEN	2_IEN	1_IEN

Address: *PTIBaseAddress* + 0x0038
Type: R/W
Reset: 0
Description:

[31:3] **Reserved**

[2] **3_IEN**: Enable interrupt on channel 3 empty

[1] **2_IEN**: Enable interrupt on channel 2 empty

[0] **1_IEN**: Enable interrupt on channel 1 empty

PTI_DMA0_STA DMA channel 0 status

7	6	5	4	3	2	1	0
Reserved					FIFO_FULL	DMA0_OVF	DMA0_DONE

Address: *PTIBaseAddress* + 0x1018
Type: R/W
Reset: 0
Description: The PTI_DMA0STATUS register shows whether DMA channel 0 has overflowed. This is only used when debugging TC code. The TC code is normally designed to read the DMA0OVERFLOW bit and signal this condition to the ST40 software via one of the interrupt status bits. The interrupt bit is also used as a handshake that the ST40 software has acknowledged the condition. Data is discarded by DMA channel 0 if the buffer it is writing overflows.

[7:3] **Reserved**

[2] **FIFO_FULL**
1: The input FIFO has reached a full condition and stalled the interface to the TC, preventing any data loss.
Write 1 to reset.

[1] **DMA0_OVF**: DMA 0 overflow
1: The channel 0 circular buffer has overflowed. Reset by writing 1 to this bit.

[0] **DMA0_DONE**
Set to 1 after inserting the last byte of the stream. This tells the DMA to flush data from the FIFO into memory.
Reset to 0 by the DMA on completion. This occurs even if no data has been put in the FIFO.

PTI_DMAn_BASE DMA buffer base address

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x1000/ 0x1000	DMA0BASE
0x1004/ 0x1020	DMA1BASE
0x1008/ 0x1040	DMA2BASE
0x100C/ 0x1060	DMA3BASE

Address: *PTIBaseAddress* + 0x1000, 0x1020, 0x1040 and 0x1060

Type: WO

Description: Each of these registers holds the base address of the DMA buffer for the corresponding DMA channel. This address must be aligned to a 16-byte boundary, so bits 3 to 0 must be written as 0. Bits 31 to 30 of this address must be the same as bits 31 to 30 of the corresponding PTI_DMAnTOP address register. The DMA channel 0 register would be set up for each PID in the PID data structure in the PTI data memory.

PTI_DMAn_READ DMA buffer read address

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x1030 0x100C	DMA0_READ
0x1034 0x102C	DMA1_READ
0x1038 0x104C	DMA2_READ
0x103C 0x106C	DMA3_READ

Address: *PTIBaseAddress* + 0x100C to 0x106C

Type: R/W except DMA0READ which is write only

Reset: Undefined

Description: Each of these registers holds the read address within the DMA buffer for the corresponding DMA channel. The address in the register is always a pointer to the next byte to be read except when the buffer is empty. The pointer must remain between the addresses defined by the base and top register. The channel 0 register would be initialized for each PID in the PID data structure in the PTI data shared memory, and an updated read pointer would be written into the same data structure.

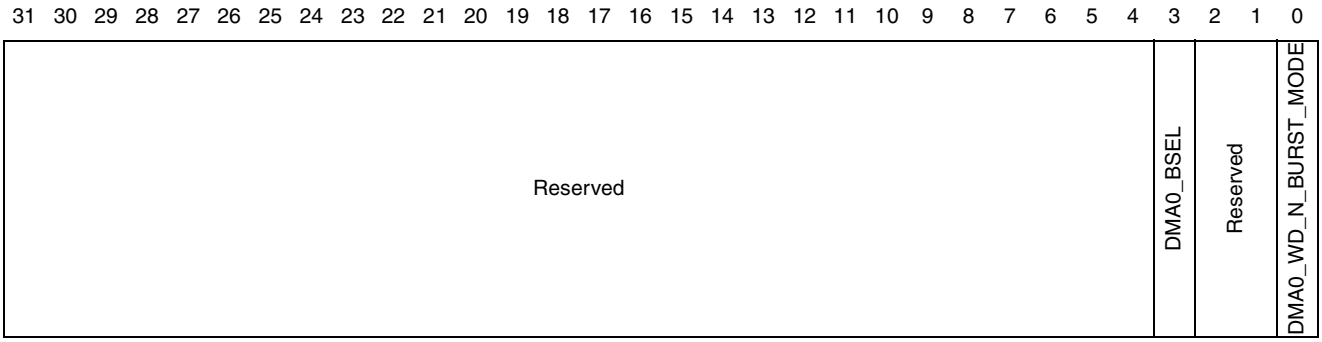
The read pointer is initialized to be equal to the write pointer. As data is read from the buffer, the hardware updates the read pointer.

For channels 1 to 3, the read pointer is automatically wrapped round, and cannot overtake the write pointer. On reaching the top pointer address the read pointer wraps around to continue reading from the base address. If the read pointer is equal to the write pointer then no data is read.

For channel 0, if the buffer is not being read by any of the DMA channels 1 to 3, then the ST40 software must perform the checks below.

- If the read address is not less than the top address, then the read address must be wrapped round to the base address.
- If the read address is equal to the write address then the buffer is empty and data should not be read.

PTI_DMA0_SETUP DMA channel 0 byte not word mode and block move



Address: *PTIBaseAddress* + 0x1010

Type: R/W

Reset: 0

Description:

[31:4] **Reserved**

[3] **DMA0_B_SEL**: DMA0 burst select
Sets word burst mode in conjunction with DMA0_WD_N_BURST_MODE; see Table 117 below.

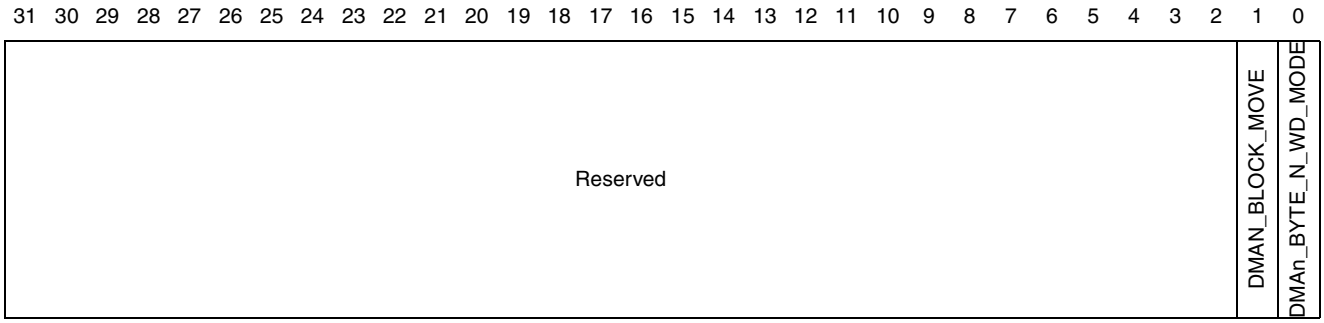
[2:1] **Reserved**

[0] **DMA0_WD_N_BURST_MODE**: DMA0 word not burst mode
Sets word burst mode in conjunction with DMA0_BSEL; see Table 117 below.

Table 117: Word burst mode select

[3]	[0]	Word burst mode
DMA0_BSEL	DMA0_WD_N_BURST_MODE	
0	0	4-word-bursts
0	1	Word-at-a-time
1	0	8-word-bursts
1	1	16-word-bursts

PTI_DMAn_SETUP DMA channels 1 to 3 byte not word mode and block move



Address: *PTIBaseAddress* + 0x1030, + 0x1050 and + 0x1070

Type: R/W

Reset: 0

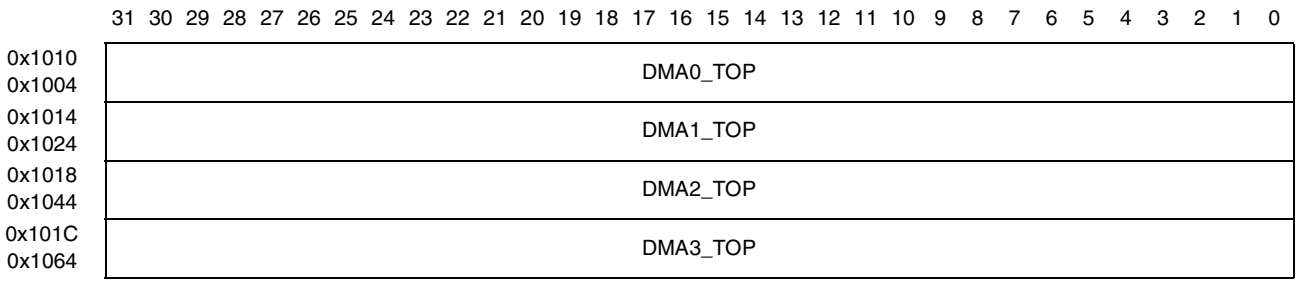
Description:

[31:2] **Reserved**

[1] **DMAn_BLOCKMOVE**
0: CDADDR remains fixed.
1: CDADDR is incremented after each write. In this mode the CD FIFO mode is off, and so *Holdoff* and *Writelength* are not considered.

[0] **DMAn_BYTE_N_WD_MODE:** DMAn byte not word mode
0: Write word-at-a-time 1: Write byte-at-a-time

PTI_DMAn_TOP DMA buffer top address



Address: *PTIBaseAddress* + 0x1004 to 0x1064

Type: WO

Description: Each of these registers holds the top address of the DMA buffer for the corresponding DMA channel. If the buffer size is not zero then this address must be one less than a 16-byte boundary, so bits 3 to 0 must be written as 1. Bits 31 to 30 of this address must be the same as bits 31 to 30 of the corresponding DMAn_BASE address register.

For channel 0 only, if the buffer size is zero then this address must be equal to the corresponding DMAn_BASE address. The DMA channel 0 register would be set up for each PID in the PID data structure in the PTI data SRAM.

PTI_DMAn_WRITE DMA buffer write address

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x1020 0x1008	DMA0_WRITE
0x1024 0x1028	DMA1_WRITE
0x1028 0x1048	DMA2_WRITE
0x102C 0x1068	DMA3_WRITE

Address: *PTIBaseAddress* + 0x1008, + 0x1028, + 0x1048, + 0x1068

Type: WO except DMA0_WRITE which is R/W

Reset: Undefined

Description: Each of these registers holds the write address within the DMA buffer for the corresponding DMA channel. The address in the register is always a pointer to the next location for a byte to be written. The pointer must remain between the addresses defined by the base and top registers. The DMA channel 0 register would be set for each PID in the PID data structure in the PTI data SRAM and an updated write pointer would be read from the same data structure.

The write pointer is initialized to be equal to the same address as the read pointer. As data is written into the buffer, the write pointer is updated, and, after reaching the top pointer address, wraps around to write the next byte at the base pointer address. In the case of channel 0 the DMA hardware updates the write pointer, the TC copies this back to the data SRAM at the end of a packet, and the CPU should read the pointer from the SRAM. In the case of channels 1 to 3, the write pointer is updated by the TC if channel 0 is writing to the buffer for that channel or by the CPU in the case that the CPU is writing the data into the buffer.

PTI_DMAn_CD_ADDR Configuration of CD FIFO address for channels 1 to 3

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
0x1064 0x1038	DMA1_CD_ADDR	Res.
0x1068 0x1058	DMA2_CD_ADDR	Res.
0x106C 0x1078	DMA3_CD_ADDR	Res.

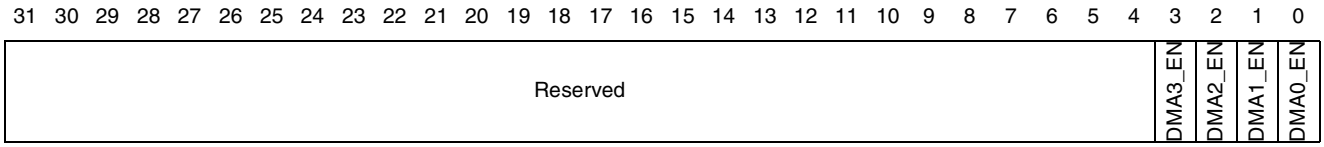
Address: *PTIBaseAddress* + 0x1038, + 0x1058, + 0x1078

Type: WO

Reset: Undefined

Description: PTI_DMAn_CD_ADDR gives the address of the corresponding CD FIFO for channels 1 to 3. The addresses must be aligned to a word boundary: the two LS bits are assumed to be 0. DMAn_CD_ADDR is only defined for MPEG channels (MDC channels).

PTI_DMA_EN DMA enable



Address: *PTIBaseAddress* + 0x101C

Type: WO

Reset: Undefined

Description: This register controls the enabling of the four DMA channels. Disabling channels 1 to 3 does not result in any data being lost inside the DMA controller but data may be lost by buffer overflow. Disabling channel 0 may result in lost data depending on the input data rate to the PTI and the length of time that the channel is disabled.

[31:4] **Reserved**

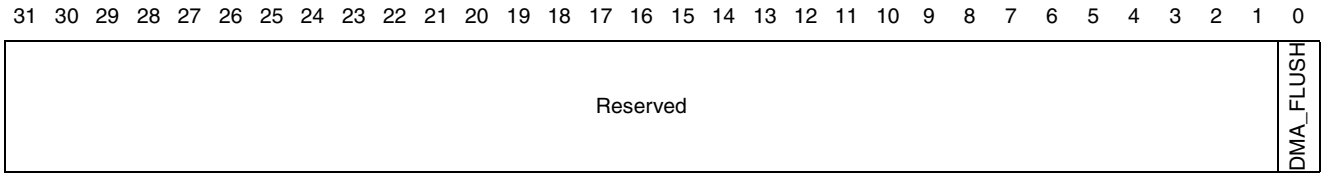
[3] **DMA3_EN**: DMA3 enable
0: Disable 1: Enable

[2] **DMA2_EN**: DMA2 enable
0: Disable 1: Enable

[1] **DMA1_EN**: DMA1 enable
0: Disable 1: Enable

[0] **DMA0_EN**: DMA0 enable
0: Disable 1: Enable

PTI_DMA_FLUSH Flush ready for a soft reset



Type: *PTIBaseAddress* + 0x105C

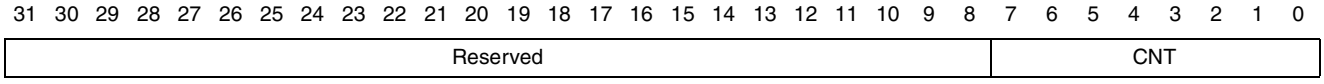
Type: R/W

Reset: 0

Description: 1 is written to DMA_FLUSH when a flush is required. It is reset by writing 0.

37.2 Input interface

PTI_IIF_ALT_FIFO_CNT IIF alternative FIFO count



Address: *PTIBaseAddress* + 0x6004

Type: RO

Reset: Undefined

Description: The number of bytes in the alternative FIFO. This FIFO does not overflow since there is flow control on its input.

PTI_IIF_ALT_LATENCY IIF alternative output latency

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								LATENCY							

Address: *PTIBaseAddress* + 0x6010
Type: WO
Reset: Undefined
Description: The number of byte clock cycles from a transport packet header being latched at input to data being available at the alternative output.

PTI_IIF_FIFO_CNT IIF count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								FULL	CNT						

Address: *PTIBaseAddress* + 0x6000
Type: RO
Reset: Undefined
Description:

[31:8] **Reserved**

[7] **FULL**
A full flag which is set when the FIFO becomes full. It is reset when the ST40 reads this register.

[6:0] **CNT**: Number of bytes in the input FIFO

PTI_IIF_FIFO_EN IIF FIFO enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CNT							

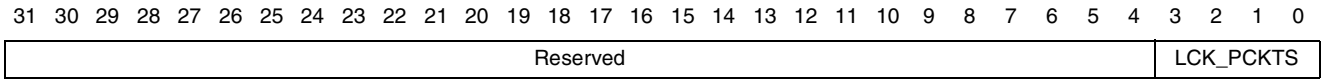
Address: *PTIBaseAddress* + 0x6008
Type: R/W
Reset: Undefined
Description: Allows data into the input FIFO. When this field is zero, the input FIFO is reset. After the ST40 completes the initialization sequence, it should set this field to 1.

PTI_IIF_SYNC_DROP IIF sync drop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														DROP_PKT	

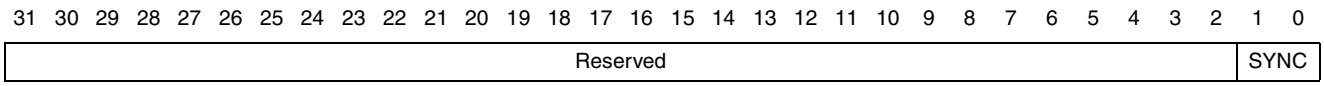
Address: *PTIBaseAddress* + 0x6018
Type: WO
Reset: Undefined
Description: Number of successive erroneous sync bytes found before the lock is treated as lost.

PTI_IIF_SYNC_CLK IIF sync lock



Address: *PTIBaseAddress* + 0x6014
Type: WO
Reset: Undefined
Description: Number of successive correct sync bytes found before the sync detection is locked.

PTI_IIF_SYNC_CFG IIF sync configuration



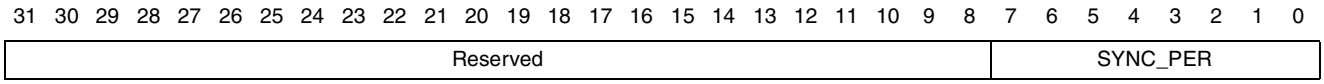
Address: *PTIBaseAddress* + 0x601C
Type: WO
Reset: Undefined
Description:

[31:2] **Reserved**

[1:0] **SYNC**

00: Default, if SYNC is activated use the internal clock
01: Use SOP
10: Use incoming TS_PKT_CLK

PTI_IIF_SYNC_PER IIF sync period

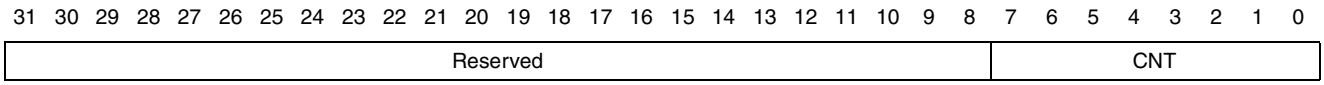


Address: *PTIBaseAddress* + 0x6020
Type: WO
Reset: 0xBC
Description:

[31:8] **Reserved**

[7:0] **SYNC_PER**: Synchronization period
Specifies the expected number of TS_IN_BYTECLK_PULSE cycles between SYNC bytes.
On reset set to 188.

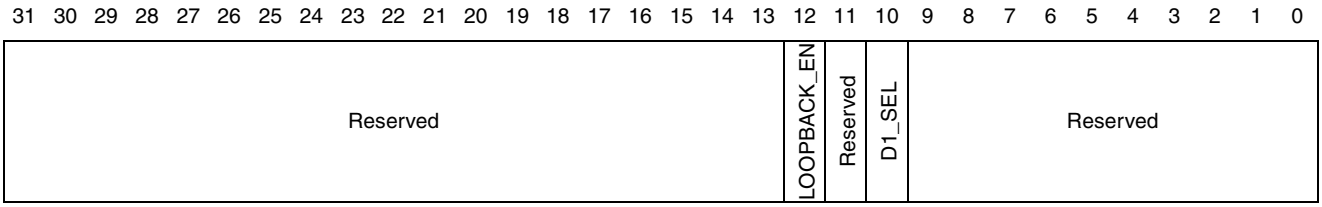
PTI_IIF_HFIFO_CNT IIF header FIFO count



Address: *PTIBaseAddress* + 0x6024
Type: RO
Reset: Undefined
Description: Number of bytes in the PTI Header FIFO.

37.3 PTI configuration

PTI_CFG PTI configuration



Address: *PTIBaseAddress* + 0x0058
Type: R/W
Reset: 0
Description:

[31:13] **Reserved**

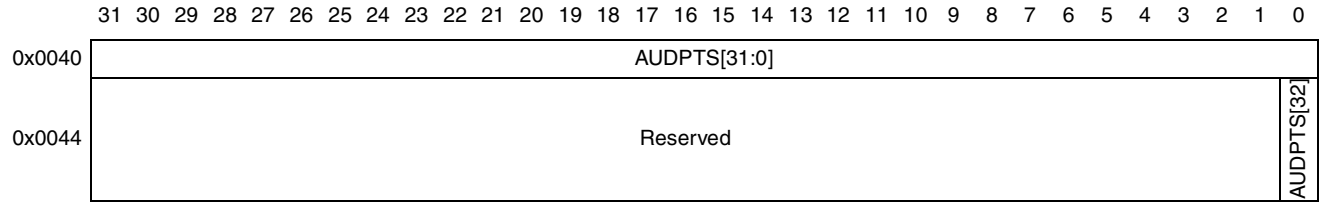
[12] **LOOPBACK_EN:** DVB-CI loopback mode
Reserved, do not modify.

[11] **Reserved**

[10] **D1_SEL**: D1 on TSIN1
Reserved, do not modify.

[9:0] **Reserved**

PTI_AUD_PTS Audio presentation time stamp



Address: *PTIBaseAddress* + 0x0040 and 0x0044
Type: RO
Reset: Undefined
Description: The system time clock (STC) value is latched into this register at the beginning of audio frame output.

The audio and video PTS registers are used by driver software to synchronize the audio and video. The time stamps are 33-bit values, so the most significant bit is held at a separate address.

Note: Because the PTI divides the 27 MHz clock by 300, the PTI_AUDPTS register timebase is 90 kHz.

PTI_INT_ACKn PTI interrupt acknowledgment

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x0020	Reserved	INT_ACK0
0x0024	Reserved	INT_ACK1
0x0028	Reserved	INT_ACK2
0x002C	Reserved	INT_ACK3

Address: *PTIBaseAddress* + 0x0020 to 0x002C
Type: WO
Reset: Undefined
Description: Acknowledge the corresponding interrupt bit when a bit is written as 1.

PTI_INT_ENn PTI interrupt enable

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x0010	Reserved	INT_EN0
0x0014	Reserved	INT_EN1
0x0018	Reserved	INT_EN2
0x001C	Reserved	INT_EN3

Address: *PTIBaseAddress* + 0x0010 to 0x001C
Type: R/W
Reset: Undefined
Description: The corresponding interrupt is enabled when a bit is 1.

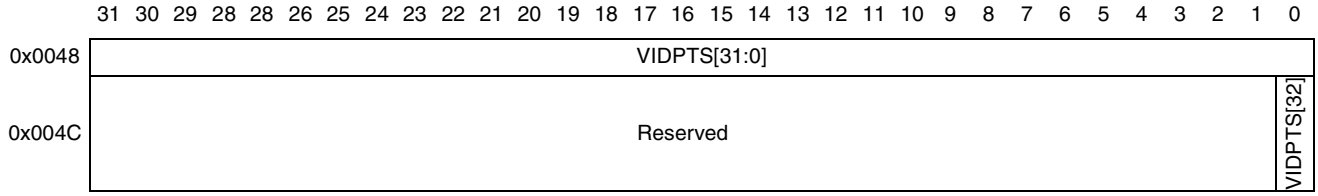
PTI_INT_STAn PTI interrupt status

	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x0000	Reserved	INT_STA0
0x0004	Reserved	INT_STA1
0x0008	Reserved	INT_STA2
0x000C	Reserved	INT_STA3

Address: *PTIBaseAddress* + 0x0000 to 0x000C
Type: RO
Reset: Undefined
Description: An interrupt is set when any bit of one of these registers is 1. The PTI_INT_STA[3:0] registers are used by the TC to raise an interrupt to the ST40 by writing 1 to a bit in one of the registers. Each of the four registers has 16 bits. All 64 bits are ORed together to produce a single interrupt for the PTI.

Once the TC has set the INT_STA bits to 1, each bit stays set until the interrupt is acknowledged by the ST40 software by writing 1 to the corresponding bit of the corresponding PTI_INT_ACK register. An interrupt is masked by writing 0 to the corresponding enable bit of the corresponding PTI_INT_EN register.

PTI_VID_PTS Video presentation time stamp



Address: *PTIBaseAddress* + 0x0048 and 0x004C

Type: RO

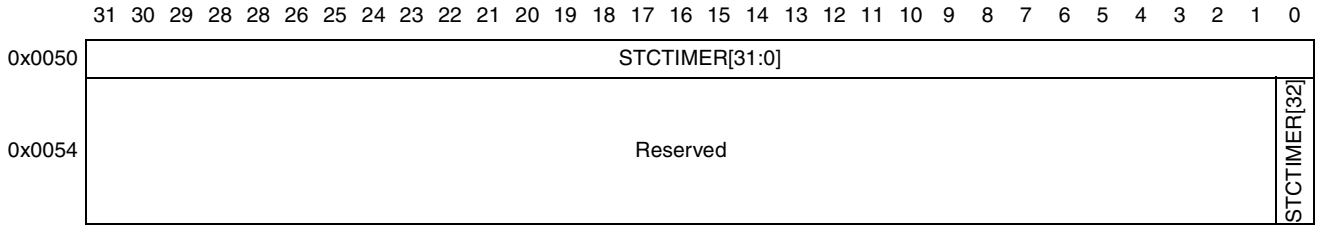
Reset: Undefined

Description: The STC value is latched into this register at VSYNC.

The audio and video PTS registers are used by driver software to synchronize the audio and video. The time stamps are 33-bit values, so the most significant bit is held at a separate address.

Note: Because the PTI divides the 27 MHz clock by 300, the PTI_VIDPTS register timebase is 90 kHz.

PTI_STC_TIMER Set STC timer



Address: *PTIBaseAddress* + 0x0050 to 0x0054

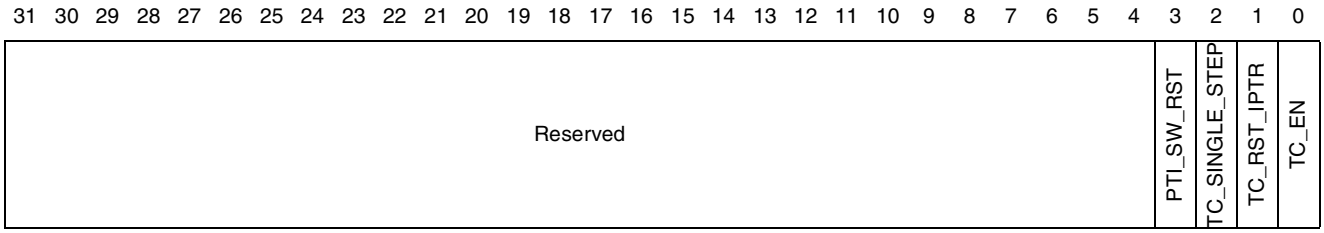
Type: WO

Description: These registers, when written, load a new value into the STC timer counter. The load is performed on the next clock edge of the 27 MHz clock after the write.

The most significant bit of the value to be loaded must be written to the register containing STC_TIMER[32] before writing to the STC_TIMER[31:0] register, as the write to the STC_TIMER[31:0] causes the update of the 33-bit counter value.

37.4 Transport controller mode

PTI_TC_MODE Transport controller mode



Address: *PTIBaseAddress* + 0x0030

Type: R/W

Reset: Undefined

Description: The PTI_TCMODE register allows the ST40 software to control the transport controller. Under normal operation the software should only set the TC_EN bit to start the TC executing code.

Resetting the TC instruction pointer

1 Clear the TC_EN bit.
2 Set the TC_RST bit.
3 Clear the TC_RST bit.
4 Set the TC_EN bit.

[31:4] **Reserved**

[3] **PTI_SW_RST**
1: The PTI is completely reset.
Default reset value 0.

[2] **TC_SINGLE_STEP**
1: Reset TC_EN after each instruction via falling edge on incoming EXECUTING signal

[1] **TC_RST_IPTR:** Reset the instruction pointer of the TC (level sensitive).

[0] **TC_EN**
0: Disables the TC disabling instruction fetch: Reading a zero after a single step means the TC has finished executing the previous instruction.
1: Enables the TC

38 Transport stream merger and router

38.1 Overview

The STx7100 supports concurrent transport stream processing of up to three independent transport streams, using an SRAM-based packet manager and a single PTI. The incoming transport packets are tagged with source ID and 42-bit time stamp.

The transport stream merger (TSMerger) has the following functionality:

- a bidirectional half-duplex interface that supports 1394 out and transport stream in,
- routes any stream to and from a PTI,
- DVB and DSS packet size support,
- output stream dejittering mechanism,
- 100 Mbits/s 1394 data output (maximum),
- 3:1 stream merger capability at the PTI target using interleaved transport stream packets with added tagging bytes.

The merger can produce an output stream from the PTI alternate output or the SWTS interface. The outgoing packets are buffered in the merger and are presented to the output pins using a dejittering mechanism that compares each packet time stamp with a fixed programmable offset.

Multiplexing in front of the merger block provides a bypass route to allow for CableCARD and DVB-CI support.

The PTI alternate output allows the entire transport stream or selected packets to be output using the merger to an external device such as a digital VCR or IEEE1394 link layer controller. The output pins can be tri-stated under software control to support low cost DVB-CI implementations and similar module interfaces.

Also under software control, the transport stream input on TS0IN can be output directly using the TS2INOUT pins. This is again to support low cost DVB-CI implementations. The returned stream from the CI module can be input on the TS2 pins (TS2IN).

With this arrangement it is possible to support CableCARD and DVB-CI configurations.

The SWTS interface allows the CPU or FDMA to send transport streams from memory to the merger for routing to the PTI or to be output via TS2. The SWTS FIFO generates an active high pacing (request) signal that is routed to the DMA source for flow control. SWTS supports a throughput of 100 Mbits/s.

The merger block manages two asynchronous input clock domains (TSBYTECLOCK0 AND TSBYTECLOCK1 from TS0 and TS1 respectively), and generates a local clock derived from the system clock for the transport stream output TS2. All other interfaces are synchronous with the system clock.

Figure 84: System overview

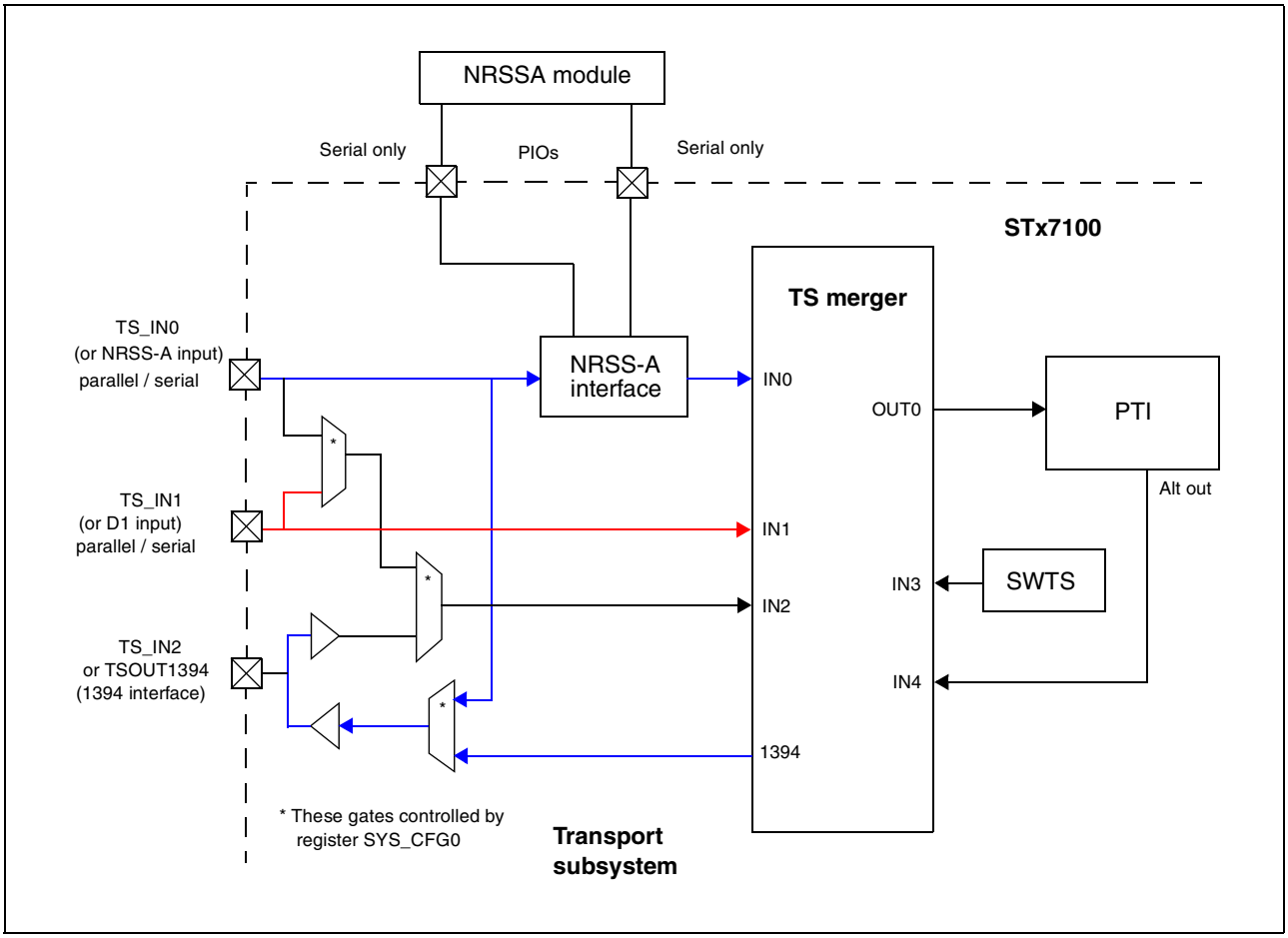


Figure 84 shows the three live input streams delivered using three sets of transport stream interfaces from the pads. Another stream is delivered from the return channel on the PTI (PTI alternative output). The TSMerger supports parallel or serial streams on the three live stream interfaces.

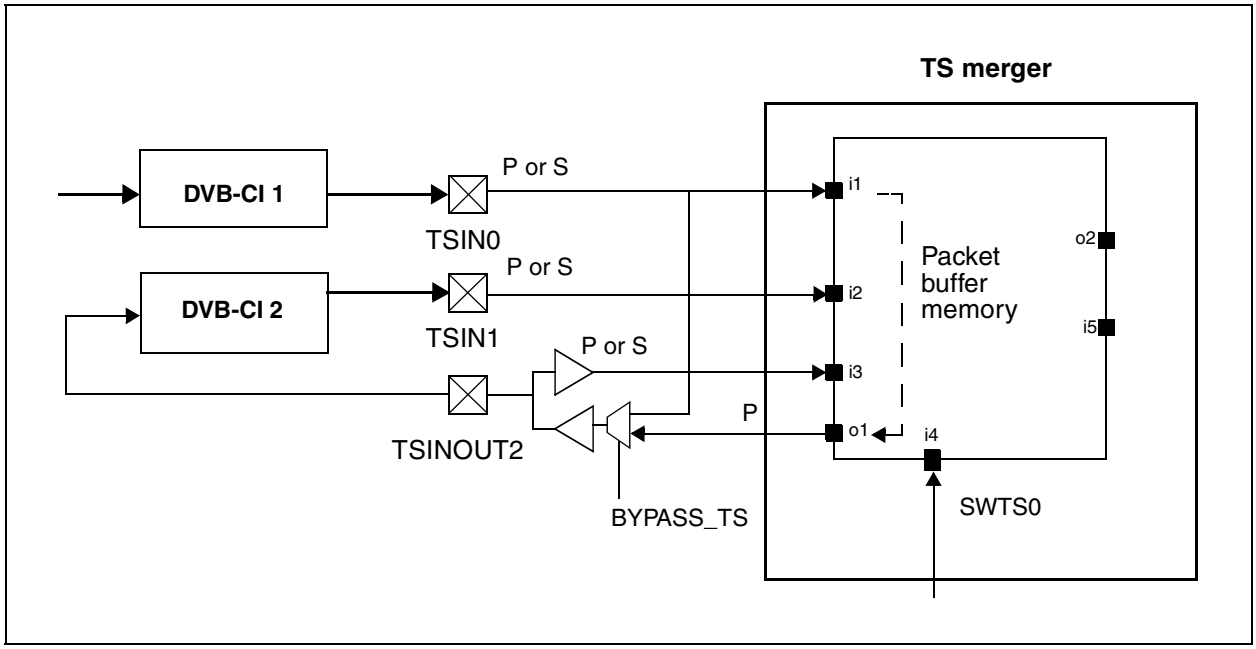
On the STx7100, the input pins TS1 are shared pins. This interface can be used for inputting transport streams for processing by the PTI or for inputting digitized video (D1) conforming to ITU-R BT 656 for processing by the digital video port (DVP).

Multiplexing and formatting performed around the TSMerger itself are controlled by software via the system configuration registers. For details, see Chapter 21: *System configuration registers on page 171*.

38.2 DVB common interface

In the DVB common interface, each module has an MPEG input port (clock, packet start, valid data, data bus) and an MPEG output port composed of the same signal. This allows successive modules to be daisy-chained. The STx7100 supports a chain of two DVB-CI modules.

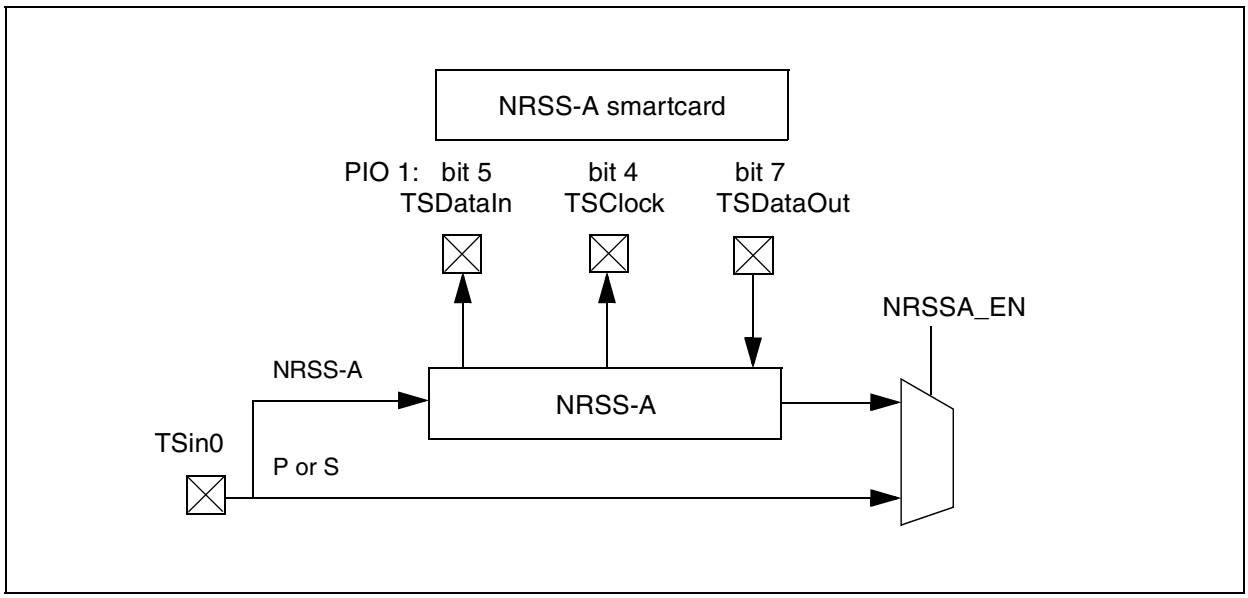
Figure 85: DVB-CI module chain



38.3 NRSS-A interface

When receiving an ATSC stream on TS0, the STx7100 can format it and pass it to an external NRSS-A CA module; this module in turn supplies a descrambled signal TS which goes back to the STx7100 and is input to the TS Merger in place of the original stream from the TS0 input. The NRSSA interface is located as an alternative function on the PIOs.

Figure 86: NRSS-A interface



38.4 Input examples

Figure 87: Basic ATSC STB with NTSC input

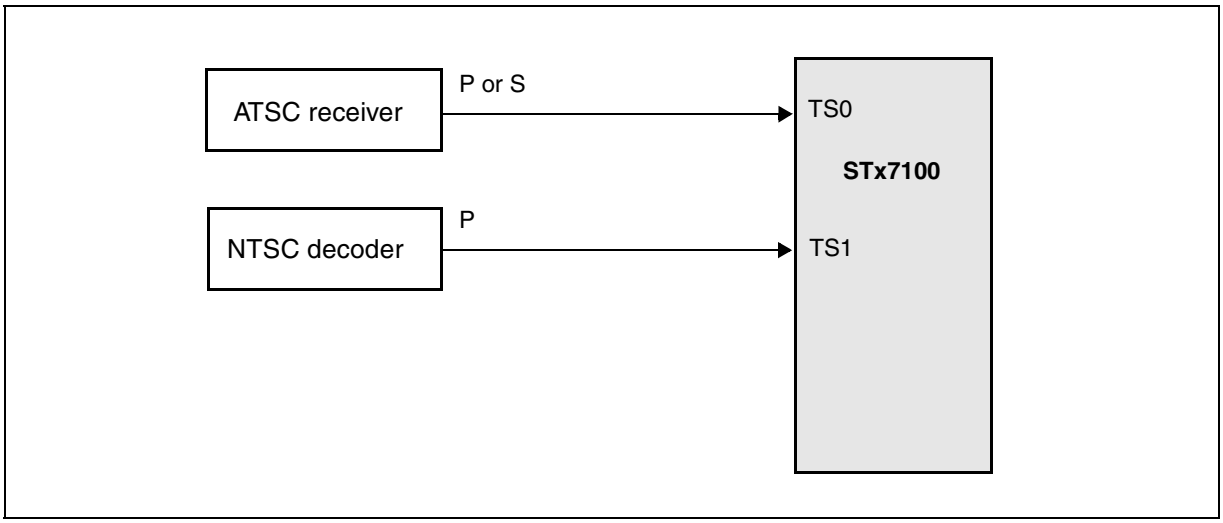


Figure 88: Satellite/cable system

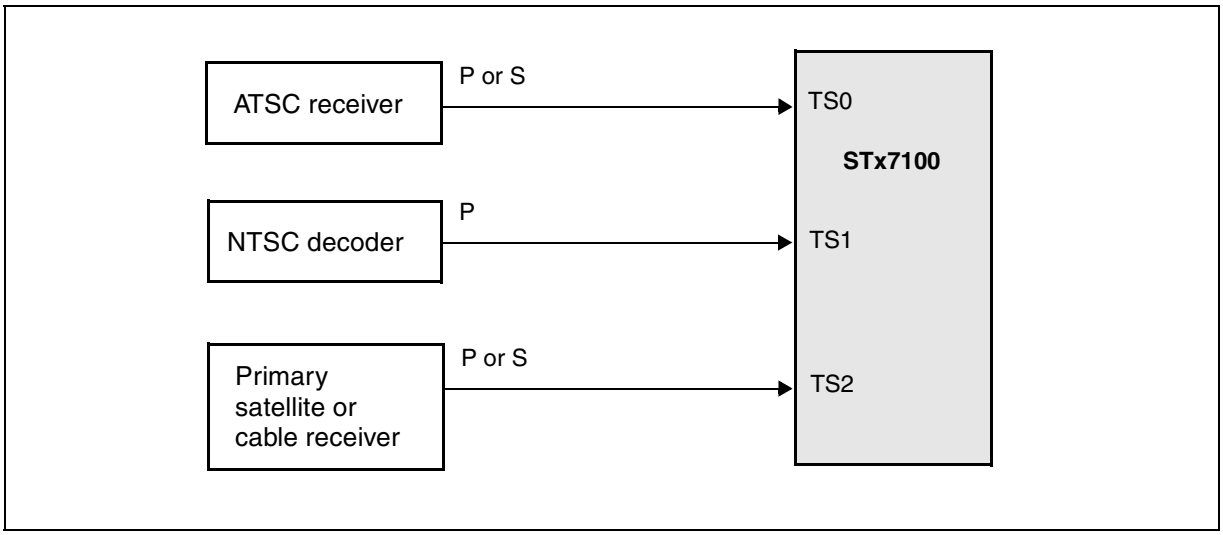


Figure 89: Multi-tuner DVR satellite/cable system with NTSC and ATSC receivers

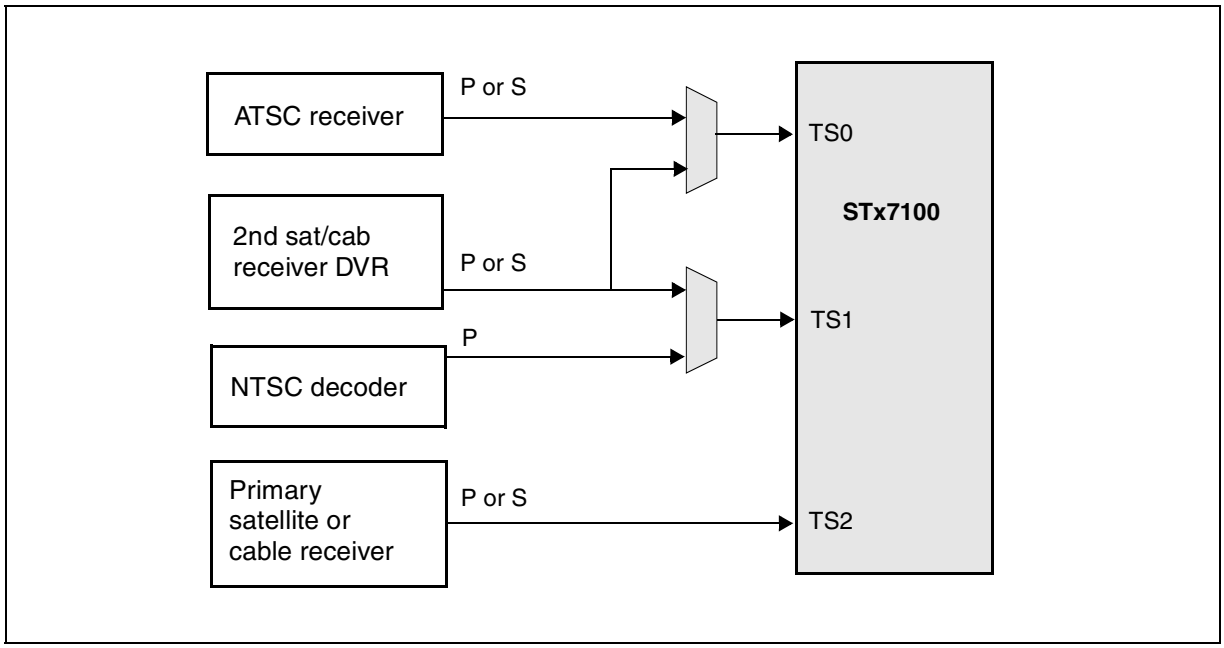


Figure 90: Japan (with 1394 and optional DVR)

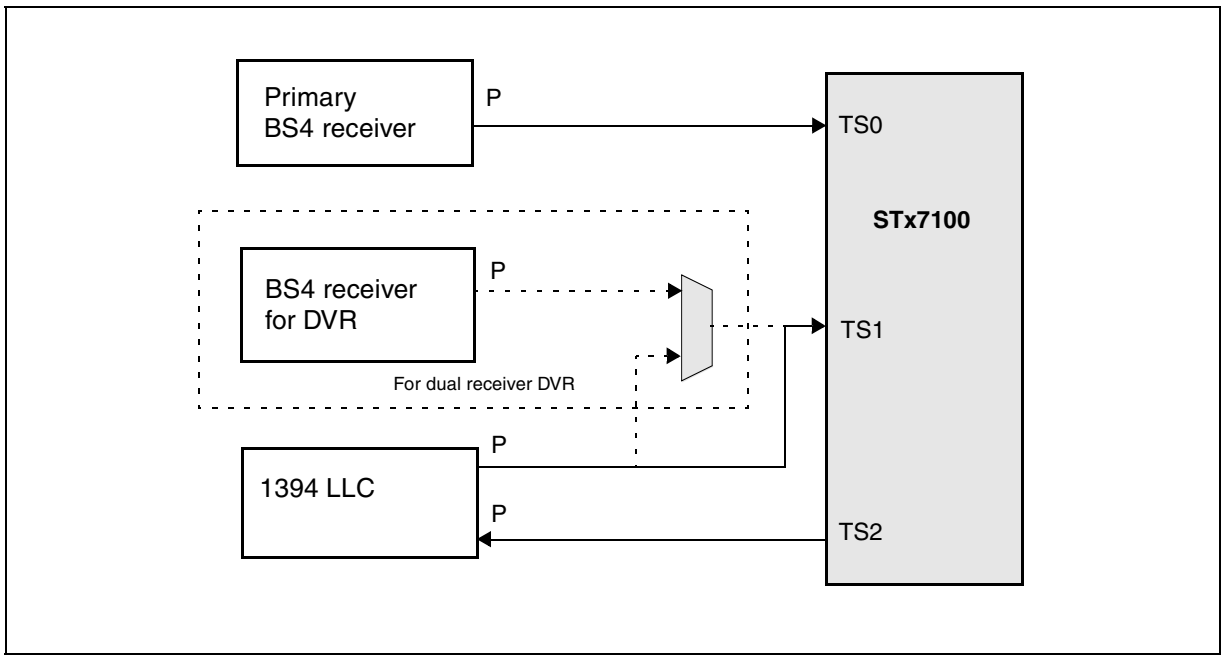


Figure 91: HD Open Cable with D1

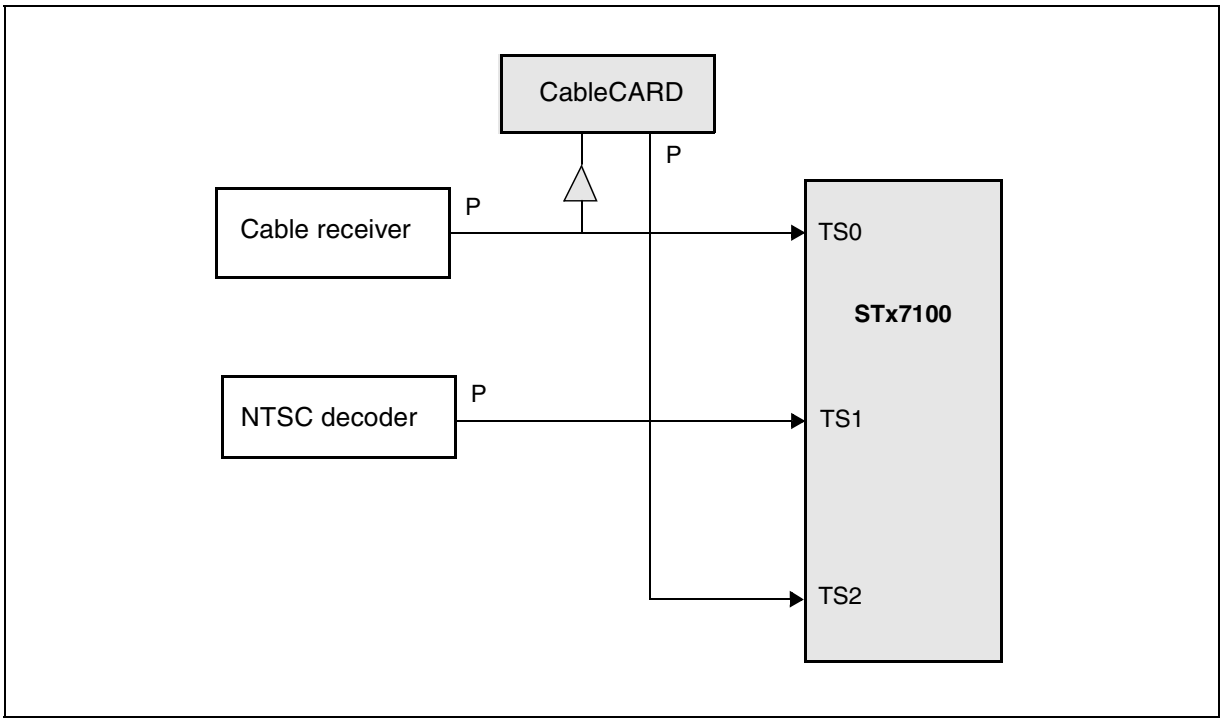
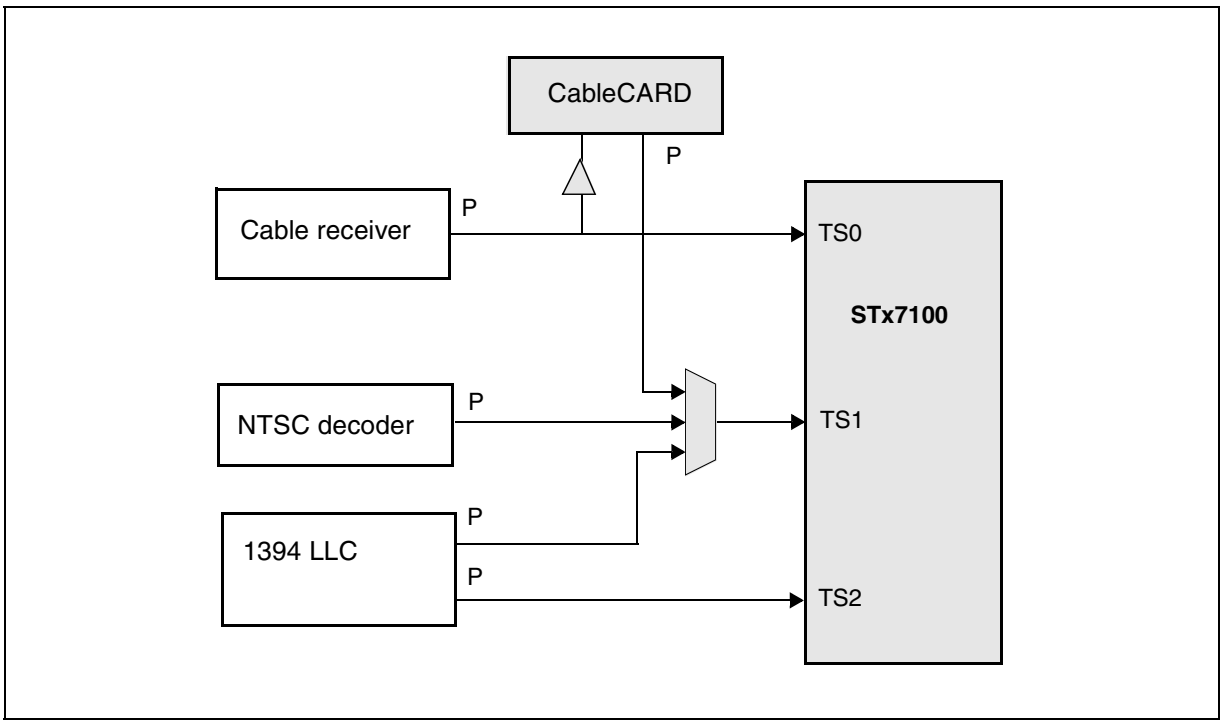
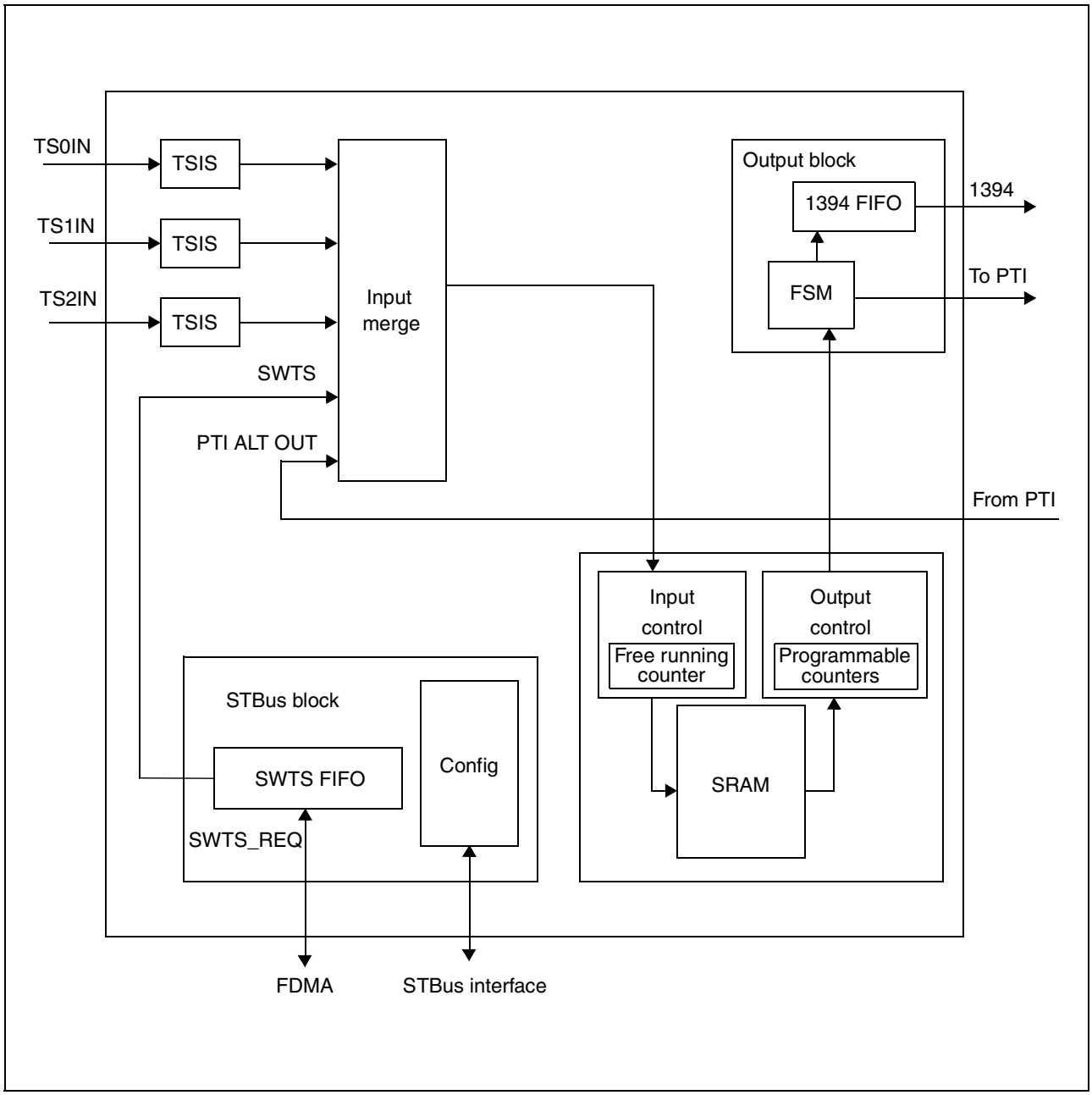


Figure 92: HD Open Cable with D1 and full duplex 1394



38.5 Architecture

Figure 93: Block diagram



38.6 Transport stream formats

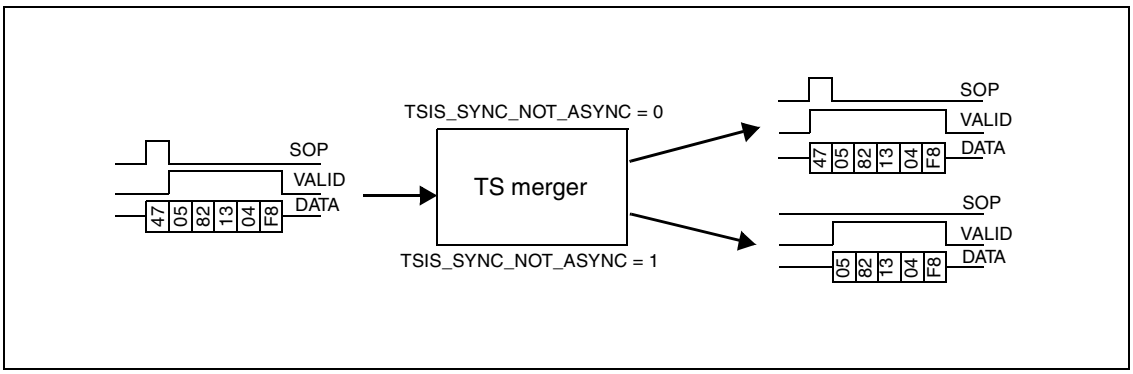
Transport streams arrive in many forms which the TS merger must deal with correctly. In order to do this each stream has a number of configuration bits which must be set correctly in order to attain the desired outcome. In the following sections a number of scenarios describe how the TS merger stream configuration bits affect TS merger functionality.

38.6.1 Off-chip live streams from FECs

Four bits in the TSM_STREAMn_CFG register control the configuration.

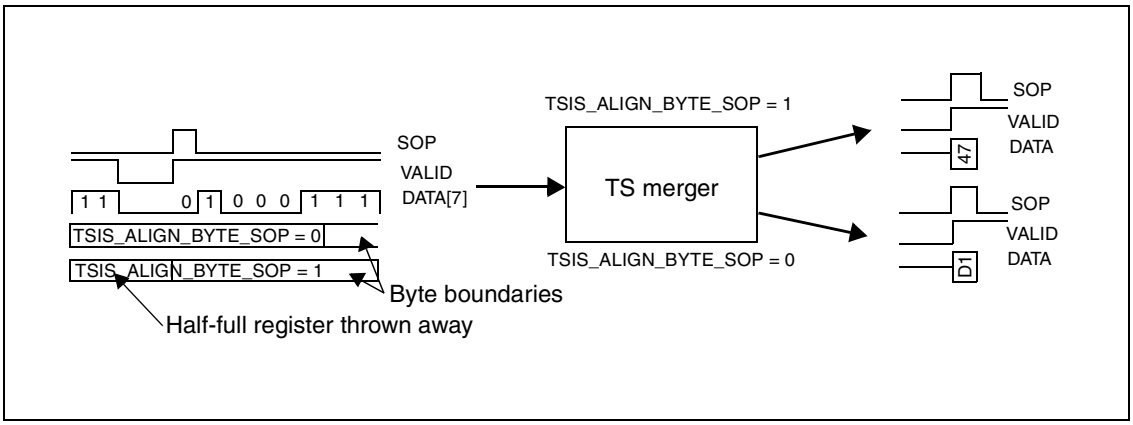
- **Serial or parallel**: The default is serial.
- **Synchronous or asynchronous data**: TSIS_SYNC_NOT_ASYNC is 1 if the data is synchronous, that is valid high for all bytes in the packet (including the start of packet byte). If the transport stream has the first byte of the packet with valid low and start of packet high, use asynchronous mode to allow the start of packet signal to be registered.

Figure 94: TSIS_SYNC_NOT_ASYNC functionality



- **Byte-aligned**: TSIS_ALIGN_BYTE_SOP is used with serial transport streams. When this bit is set to 0 the TSIS continues to take in bytes irrespective of where the PACKETCLK goes high. When this bit is set high then the PACKETCLK is examined and the first bit of the serial packet is put in to the MSB position of the outgoing parallel transport stream. The advantage of having this bit high is that the start-of-packet is detected and the stream leaving the TSIS is byte-aligned, negating the use of sync lock and drop further down stream.

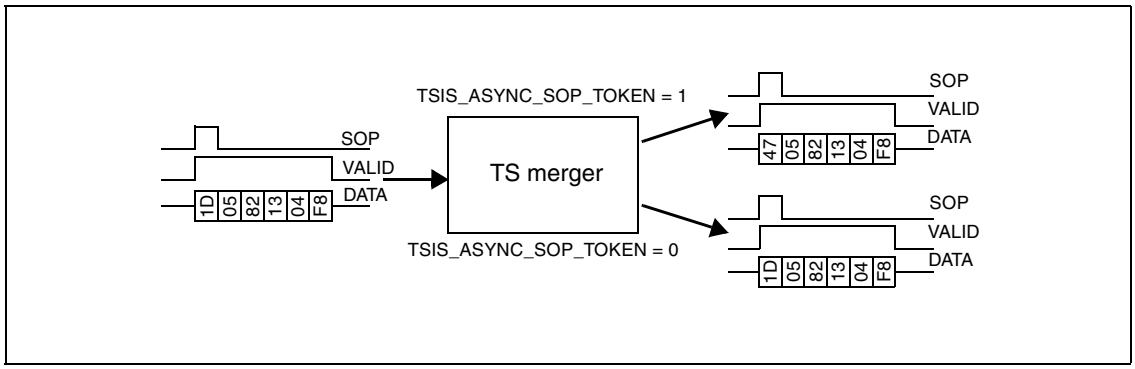
Figure 95: TSIS_ALIGN_BYTE_SOP functionality



- **Start-of-packet detection**: Enables the sync lock and drop mechanism on this stream when the stream is played back from HD.

Note: The SOP_TOKEN *in TSM_STREAMn_SYNC is reset to 0x47, but can be changed. The value of the start-of-packet token which is inserted into the stream is the value held within this register field.*

Figure 96: TSIS_ASYNC_SOP_TOKEN functionality



- **BYTECLK inversion**: When INV_BYTECLOCK is high, the BYTECLK goes through an inverter cell.

38.6.2 SWTS

SWTS playback data is sent across the STBus and into the TS merger. The data contains no other information about the stream. It is assumed that every byte arriving into the SWTS register is valid data and that there are no erroneous packets (there is no error signal to specify this). A repeating start of packet token (for example 0x47) is expected in order to reproduce a start-of-packet signal. The remaining role of the TS merger is to reproduce the transport packet at the correct frequency so there is little jitter and the playback of the video is correct. There are two mechanisms for doing this. One way is to use the counter value placed within the packet. The TS merger holds a packet in the SRAM until a programmable counter reaches the value of the three lower counter bytes held in the packet header. Once the value is reached by the programmable counter the packet can be sent on to the destination. While the current packet is held, the SWTS data continues to send data from the next packet and store it in the SRAM. When the SRAM circular buffer is full data propagation stops and no more DMA requests are made. This allows the SWTS pacing to be controlled by hardware instead of software.

Data flow control

To set up the hardware SWTS data flow control mechanism, set PACE_MODE_SEL in TS_SWTS_CFG to 1. If however the SWTS does not contain the four bytes of ID and counter stamp then the SWTS can be configured by setting up a pace counter value within the TS_SWTS_CFG register. This number refers to the number of system clock cycles between data bytes on the SWTS. In order to control data flow using the software pace register, the PACE_MODE_SEL bit must be set to 0. If the SWTS does not have the counter bytes in its header then it cannot self-pace and as soon as a packet is completed within the SRAM the packet is sent to the target (PTI or 1394 device).

A similar mechanism exists for PTI alternate output data. A channel exists through the TS merger for PTI return data. The user must select which PTI to connect this to and this is handled by the PACE_MODE_SEL bit in TSM_PTI_ALT_OUT_CFG.

38.7 Packet buffering

A large SRAM collects transport data and builds up transport stream packets before merging to their destination. Each stream needs a portion of the RAM in order to build up its packets and this needs to be configured before data is received at the memory.

The method of setting this is to use the RAM_ALLOC_START field in TSM_STREAMn_CFG which defines the start address of the RAM for each stream. The amount of memory assigned to a stream depends on its start address and the next streams start address.

The TS merger uses an SRAM to store packets before merging them to the destination (for example the PTI). The SRAM is split into a number of circular buffers (FIFOs) of limited size. The recommended size of these buffers is:

- live: 640 bytes,
- SWTS and PTIALT: 384 bytes.

Packets only leave the buffer once the destination has completed previously-scheduled packets in a queue. This means that the buffer must be large enough to hold one packet plus any extra transport data arriving while the packet is stored. The space required therefore increases as the number of streams to a single destination increases, that is 3:1 rather than 2:1.

If the FIFO circular buffer is about to overflow the hardware takes action to delete the uncompleted packet and begins to store again from the next packet start. In this instance, some packets may be lost. This scenario should not occur in a well-configured system and the described mechanism is the cleanest way of getting out of the overflow condition.

38.8 Transport stream inputs

To use the TS merger, the transport streams must be modified so the PTI can distinguish between sources. An example is where a live stream is received from TS0IN and sent to the PTI, which stores the stream on to the hard disk. The stream is then read from the hard disk using SWTS0 (stream 3) and the software stream is also sent to the PTI. The PTI distinguishes between the live stream and the software stream. Section 38.8.1 describes how this happens.

38.8.1 Packet tags

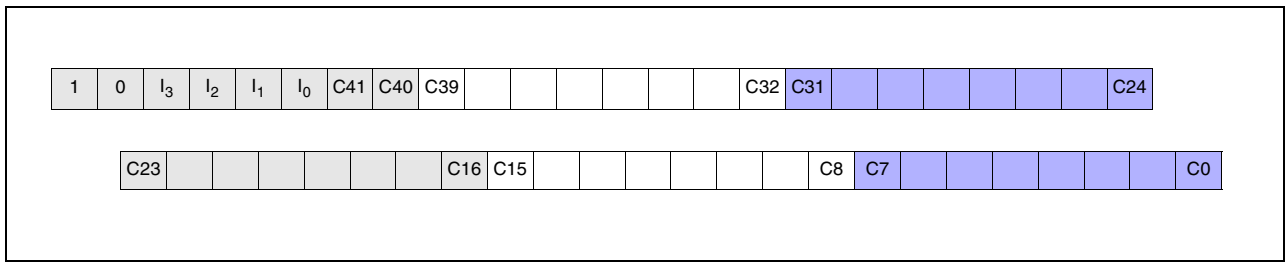
When a packet is injected into the TS merger, the start of the packet is identified and extra bytes can be placed after the start of packet byte when ADD_TAG_BYTES is set to 1 in TSM_STREAMn_CFG, see Figure 98. The structure of the tagging bytes is shown in Figure 97. Six tagging bytes are inserted after the first byte of a packet. The first of the tag bytes contains the four-bit source ID, encoded as shown in Table 118. When sending live data through the TS merger, tagging should always be set, and distinguishing between tagged and nontagged data is simple (known set-up). However, reading transport streams off disk is more difficult, because it is unknown whether the packets contain the tagging bytes. The MS bit of the ID tag byte is set to 1 as this indicates that the stream has tag bytes added. If there are no tag bytes added then this byte is the second byte of the transport packet and the MS bit of this byte is a transport error indicator. No packets with this bit set are recorded to disk.

Table 118: ID encoding

Stream number	Stream name	I_3	I_2	I_1	I_0
0	TS0IN	0	0	0	0
1	TS1IN	0	0	0	1
2	TS2IN	0	0	1	0
3	SWTS0	0	0	1	1
4	PTI_ALT_OUT	0	1	0	0

The five following bytes contain a counter stamp reference which is used during autopaced playback, explained in Section 38.9.2: *Delaying outgoing packets on page 358*.

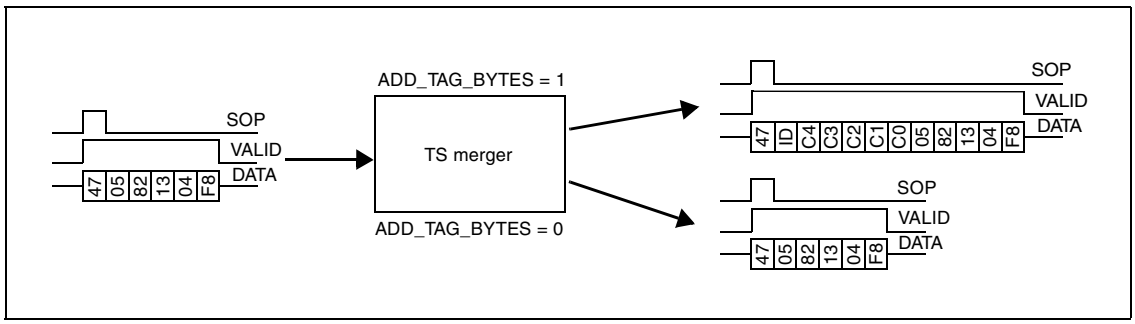
Figure 97: Tag bytes



Enable packet tagging

This register bit enables the packet tagging function. When high, an additional 6 bytes are placed within the stream, an ID byte which indicates which stream the packet is from (important for the PTI for context switching), and 5 bytes of a free running counter stamp (which is used to dejitter streams off disk or from the PTI).

Figure 98: ADD_TAG_BYTES functionality



Not all streams require the ADD_TAG_BYTES field to be set because streams coming back from hard disk that have already been through the TS merger when they were live and already have the ID and counter bytes inserted. Having ADD_TAG_BYTES set for this type of stream causes an extra six bytes to be inserted.

Streams that have already been sent through the TS merger once, for example SWTSs, do not require additional bytes to be placed in their headers, however the stream must be modified to update the ID field.

Setting stream ID

REPLACE_ID_TAG in TSM_STREAMn_CFG replaces the second byte in the packet with the stream ID. If the incoming stream had not had the six extra bytes inserted originally then the substitution still goes ahead, corrupting the first or second byte of data.

38.8.2 Sync lock and drop

The sync lock and drop mechanism is used to detect start of packets when the start of packet signal is not present. Each stream can have different configuration and is defined by the TSM_STREAMn_SYNC register. The mechanism searches for a byte value within the stream defined by the SOP_TOKEN field. Once found, the mechanism waits for PKT_LEN (bits 31 to 16) bytes and tries to identify the SOP_TOKEN in the expected placement. If SOP_TOKEN appears again then the process continues for SYNC (bits 3 to 0) packets. Once SYNC packets are located, the stream is locked and, together with a start of packet signal, is sent to SRAM for storage. If a SOP_TOKEN is missing from the stream and the stream is not locked then the mechanism searches for the next occurrence of SOP_TOKEN and starts the process again. If a SOP_TOKEN byte is missing for DROP times (bits 7 to 4) in succession and the stream is locked, then the stream becomes unlocked and a new sync lock process begins.

To use the PACKETCLK signal instead of the sync lock and drop mechanism, set SYNC to 0000.

38.9 Transport stream output

The output block of the TS merger uses the TSM_1394_DEST and TSM_PTI_DEST registers to decide where source streams are sent.

38.9.1 1394 output configuration

The 1394 interface allows transport stream data to be sent to and received from an external device. An example use of the 1394 ports would be to connect to a chip which interfaces to a digital video camera or to a chip which interfaces to an external hard disk.

The 1394 output stream can be configured in a number of ways. The data leaving the TS merger can be setup on the falling edge of the 1394 input BYTECLK or an internal clock can be generated using the 1394_PACE field in TS_1394_CFG. Outgoing 1394 packets can be stripped of any tagging bytes by setting 1394_REM_TAG_BYTES to 1.

38.9.2 Delaying outgoing packets

The free running counter and programmable counters are each split into two fields. The lower 9 bits are incremented using a 27 MHz clock and count from 0 to 299. Instead of moving from 299 to 300, the lower 9 bits roll over to 0, and the upper 15 bits (programmable counters) or 33 bits (free running counter) are incremented. For the free-running counter, this results in a 33-bit counter being effectively clocked at 90 kHz, which the PTI requires.

The role of the programmable counters is to delay the outgoing transport stream packets so that they resemble the original transport stream as closely as possible (that is, jittering is reduced). When an original stream goes through the transport stream merger an ID tag and 42 bits of free-running counter can be inserted into the header of the stream. When this stream is played back, the counter value is examined and the packet is allowed to be read from SRAM only if the lower three bytes of the programmable counter associated with the stream reaches the lower three bytes of the header counter value. The effect is that the reproduced stream should be dejittered. The counters can be initialized in two ways:

- Software can write the value of the counter into the counter register.
- Hardware can automatically start the counter at the correct point (for the first packet in a stream, it sets the programmable counter to the header counter value and sends the packet to the destination).

When using the software method TSM_PROG_CNTn.AUTO_CNT should be set to 0. TSM_STREAMn_STA.CNT_VAL holds the currently waiting packet header counter value so that the programmable counter can be set to this figure. When AUTO_CNT = 1 and CNT_INIT = 0

then the programmable counter is set to the value of the waiting packet header counter value. Once set, CNT_INIT is set to 1. To reinitialize the counter at any point the CNT_INIT field can be set to 0 to force hardware reinitialization. Software can change the programmable counter value at will; this does not affect the behavior of CNT_INIT or AUTO_CNT.

The playback speed is set by the CNT_INC field, for example 1 for normal playback, when the counter is incremented by 1 every 27 MHz, 2 for x 2 playback (twice every 27 MHz). If the counters are not required (for example there is no counter value to match in the header) then the CNT_INC value should be set to 0.

38.9.3 Dejittering mechanism

The dejittering mechanism can be broken if the latency within the transport stream merger PTI route is too variable. The following example demonstrates this process.

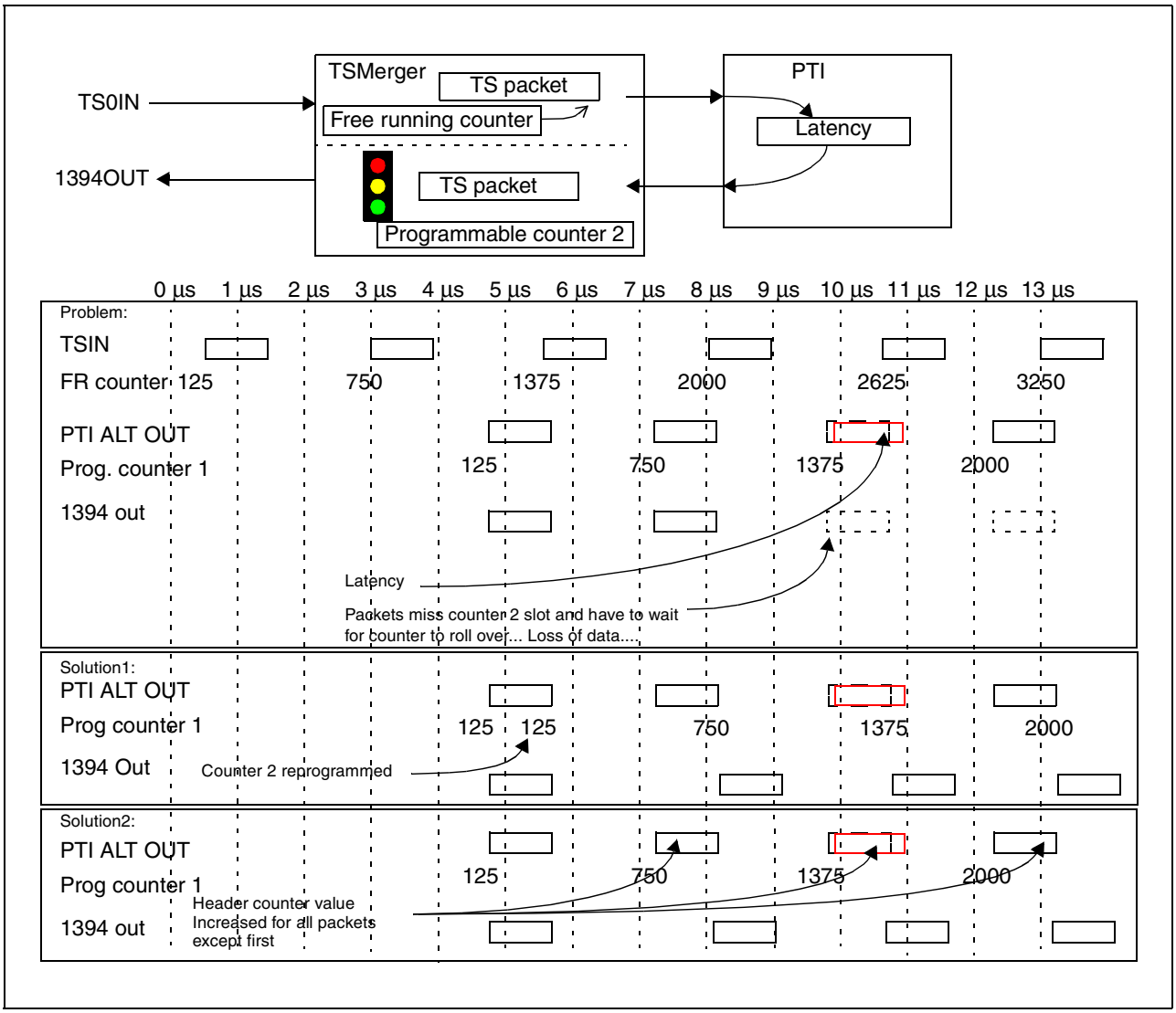
Missing programmable counter 2 slot

1. A live stream is injected using TS0IN and tagging bytes are added to each packet header. These packets are sent to the PTI where they are processed and returned back to the transport stream merger to be sent dejittered to the 1394 ports.
2. Counter 2 is autoinitialized to the first packet header counter value and sends the packet to 1394.
3. The third packet is delayed longer than average in transport stream merger PTI and is returned later than expected, by which time the programmable counter 2 has reached a value of 1500 (missing the 1375 slot).
4. The third packet has to wait for counter 2 to roll over and reach 1375 before it can be sent on to the 1394. The result of this is one (or more depending upon the size of the circular buffer assigned to stream 8 in SRAM) very delayed packet and a large number of missing packets.

This problem has two solutions.

- **Reprogram programmable counter 2**
  After receiving the first packet from the PTI, wait an amount of time and reprogram counter 1 to a smaller value. This allows larger amounts of latency to be absorbed by transport stream merger as it appears to transport stream merger that the subsequent packets are arriving earlier from the PTI.
- **Edit the packet header counter values**
  The TC can reprogram the header counter values so that it appears to transport stream merger that the packets are able to be sent on later.

Figure 99: Breaking the dejittering mechanism



38.9.4 Stream status

Each stream has a status register (TSM_STREAMn_STA) with the following fields:

CNT_VAL (counter value)

This field is the counter value held within the header of the packet that is next in the queue to be delivered to its destination. The purpose of having this visibility is that the software can check the header value and adjust the programmable counter accordingly if any misalignment occurred and the packet is holding up the stream unnecessarily.

ERR_PKTS (erroneous packets)

For every packet that is received with the PACKETERROR signal high on the first byte of the packet ERRONEOUS_PACKETS is incremented. The register counts the number of packets which are erroneous due to the PACKETERROR signal being high.

RAM_OVF (RAM overflow)

RAM_OVERFLOW indicates that the RAM circular buffer has overflowed. This should not occur with SWTS or PTI return streams in AUTO_PACE mode. If an overflow is detected then the packet that could not be completed is removed from within RAM and the hardware waits for the next start of packet and starts storing that. The worst case RAM_OVF scenario is that a packet may be lost. No half packets arrive at the PTI.

IN_FIFO_OVF (input FIFO overflow)

IN_FIFO_OVF indicates that bytes of data could not be loaded into the input FIFO because the FIFO was already full. This results in shorter packets arriving at the destination or that sync lock is lost. In order to combat this occurrence, the stream arbitrator which decides which stream to process each cycle, looks at each FIFO level and the most full FIFO gets priority. If two FIFOs are equally full then the PRIORITY field decides between the tied FIFO levels. For high data rate streams, priority should be set to a high value (for example 1111) and low data rate streams should be programmed with a low priority (for example 0000), as the lower data rate stream is less likely to overflow than a high data rate stream.

STREAM_LCK (stream lock)

Indicates that the stream is allowed to continue to RAM because there is either successful sync lock or PACKETCLK signal is used to identify packets.

39 Transport stream merger and router registers

Register addresses are provided as:

TSMergerBaseAddress + offset.

The *TSMergerBaseAddress* is:

0x1924 2000.

Table 119: Register summary

Name	Function	Offset	Type
TSM_STREAM0_CFG	Input stream TSIN0 configuration	0x0000	R/W
TSM_STREAM0_SYNC	Input stream TSIN0 synchronization	0x0008	
TSM_STREAM0_STA	Input stream TSIN0 status	0x0010	RO
Reserved	-	0x0018	-
TSM_STREAM1_CFG	Input stream TSIN1 configuration	0x0020	R/W
TSM_STREAM1_SYNC	Input stream TSIN1 synchronization	0x0028	
TSM_STREAM1_STA	Input stream TSIN1 status	0x0030	RO
Reserved	-	0x0038	-
TSM_STREAM2_CFG	Input stream TSIN2 configuration	0x0040	R/W
TSM_STREAM2_SYNC	Input stream TSIN2 synchronization	0x0048	
TSM_STREAM2_STA	Input stream TSIN2 status	0x0050	RO
Reserved	-	0x0058	-
TSM_STREAM3_CFG	Software transport stream 0 configuration	0x0060	R/W
TSM_STREAM3_SYNC	Software transport stream 0 synchronization	0x0068	
TSM_STREAM3_STA	Software transport stream 0 status	0x0070	RO
Reserved	-	0x0078	-
TSM_STREAM4_CFG	PTI alternate output stream configuration	0x0080	R/W
TSM_STREAM4_SYNC	PTI alternate output stream synchronization	0x0088	
TSM_STREAM4_STA	PTI alternate output stream status	0x0090	RO
Reserved	-	0x0098 - 0x01F8	-
TSM_PTI_SEL	Select stream sources for PTI	0x0200	R/W
TSM_1394_SEL	Select stream sources for 1394 output	0x0210	
Reserved	-	0x0218 - 0x03F8	-
TSM_PROG_CNT0	Programmable counter 0 for SWTS0	0x0400	R/W
TSM_PROG_CNT1	Programmable counter 1 for PTI alternate output	0x0410	
Reserved	-	0x0418 - 0x05F8	-
TSM_SWTS_CFG	SWTS configuration	0x0600	Mixed
Reserved	-	0x0610 - 0x07F8	R/W
TSM_PTI_ALT_OUT_CFG	PTI alternate output configuration	0x0800	R/W

Table 119: Register summary

Name	Function	Offset	Type
TSM_1394_CFG	1394 port configuration	0x0810	R/W
TSM_SYS_CFG	System configuration	0x0820	R/W
TSM_SW_RST	Software reset	0x0830	R/W
TSM_SWTS	Data for SWTS	0x10B E000	

See also Chapter 21: *System configuration registers* *on page 171*, registers SYS_CFG0 and SYS_CFG1.

TSM_STREAMn_CFG Stream n configuration

31–13	12–8	7	6	5	4	3	2	1	0
Reserved	RAM_ALLOC_START	STREAM_ON	REPLACE_ID_TAG	ADD_TAG_BYTES	INV_BYTECLK	TSIS_ASYNC_SOP_TOK	TSIS_ALIGN_BYTE_SOP	TSIS_SYNC_NOT_ASYNC	TSIS_SER_NOT_PAR
	19–16: PRI; 15–13: Reserved								

Address: *TSMergerBaseAddress* + 0x0000, 0x0020, 0x0040, 0x0060, 0x0080

Type: R/W

Reset: 0 (TSIS_SERIAL_NOT_PARALLEL: 1)

Description:

[31:20] **Reserved**

[19:16] **PRI**: Stream priority
0000: Lowest priority 1111: highest priority
The priority should be set according to data rate of the stream. Higher priority should be given to streams with higher data rates.

[15:13] **Reserved**

[12:8] **RAM_ALLOC_START**
The RAM is separated into 64-byte words; this field designates the lowest word that is allocated to stream *x*.

[7] **STREAM_ON**: Gate the transport stream
This register bit stops transport data propagating to the SRAM before the circular buffers are setup.
If the stream is from off-chip, the byte clocks are disabled when this bit is low. If the stream is from the PTI alternate outputs or from SWTS, the FIFO enables are disabled when this bit is low.

[6] **REPLACE_ID_TAG**
Controls whether the packet second byte is replaced with a new ID field, for use with SWTS or PTI alternative output streams where the original ID is pointing to the source live stream ID.

[5] **ADD_TAG_BYTES**: Controls whether the stream header includes 6 bytes of ID and counter value

[4] **INV_BYTECLK**: Sense of the incoming BYTECLKs
0: The live stream BYTECLK is not inverted 1: The live stream BYTECLK is inverted.

[3] **TSIS_ASYNC_SOP_TOKEN**: TSIS async SOP token
1: The TSIS replaces the first byte of each packet with the value held in the SOP_TOKEN register. This function is included in order to make DSS and DVB streams appear similar for the PTI.

[2] **TSIS_ALIGN_BYTE_SOP**
This bit is only valid for live input streams in serial mode and is ignored for live parallel stream, SWTS and PTI alternate outputs.
0: TSIS builds up bytes bit by bit, until 8 bits are collated and sends out a byte.
1: TSIS looks at start of packet signal and starts a new byte with the data associated with this flag. The previous byte is aborted in this instance.

[1] **TSIS_SYNC_NOT_ASYNC**
This bit is only valid for live input streams and is ignored for SWTS and PTI alternate outputs.
0: TSIS is in asynchronous mode. Asynchronous mode assumes all bytes with either valid high or PACKETCLK high are valid.
1: TSIS is in synchronous mode. Synchronous mode assumes all bytes with valid high are valid.

[0] **TSIS_SER_NOT_PAR**: TSIS serial not parallel
This bit is only valid for live input streams and is ignored for SWTS and PTI alternate outputs.
0: TSIS is in parallel mode. 1: TSIS is in serial mode.

TSM_STREAMn_SYNC Stream n synchronization

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PKT_LEN																SOP_TOKEN								DROP				SYNC			

Address: *TSMergerBaseAddress* + 0x0008, 0x0028, 0x0048, 0x0068, 0x0088

Type: R/W

Reset: See below

Description:

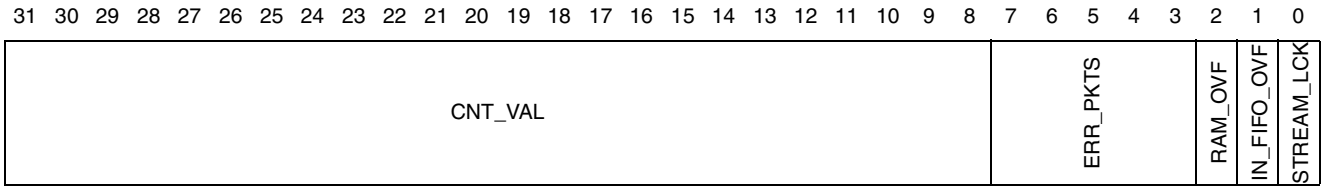
[31:16] **PKT_LEN**
Expected packet size in bytes, that is, a start of packet token should be found every PKT_LEN bytes. Reset: 0000 0000 1011 1100 (0xBC)

[15:8] **SOP_TOKEN**
Start of packet token to be sought when not using the start-of-packet signal (that is using sync lock and drop). Reset: 0100 0111 (0x47)

[7:4] **DROP**
Number of start of packet tokens to be missing from the expected placement in a locked stream before the stream is unlocked and data transfer to the PTI is aborted.
If the start of packet signal is to be used as the start of packet detection mechanism then this bit field should be set to 0000. Reset: 0000

[3:0] **SYNC**
Number of start of packet tokens to be identified in the correct place before the stream is locked and passed on to the PTI.
If the start of packet signal is to be used as the start of packet detection mechanism then this bitfield should be set to 0000. Reset: 0000

TSM_STREAMn_STA Stream status



Address: *TSMergerBaseAddress* + 0x0010, 0x0030, 0x0050, 0x0070, 0x0090

Type: RO

Reset: 0

Description:

[31:8] **CNT_VAL**
Indicates the lower 24 bits of counter value in the header of the next packet waiting in RAM for nonlive stream. This is only valid for PTI alternate output streams and software transport streams. This allows software to know what value to program the programmable playback counters.

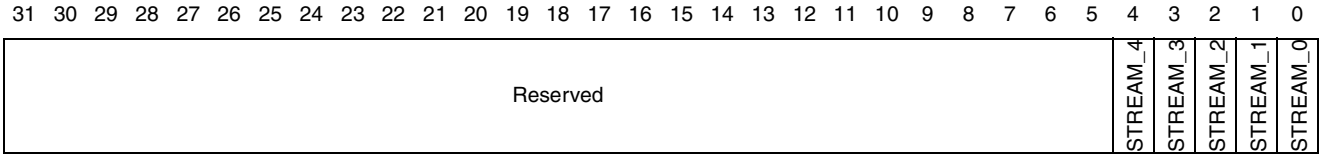
[7:3] **ERR_PKTS**
Counts the number of packets thrown away due to errors. This does not include packets thrown away due to bad check sum and it only includes packets where the error signal is high for the first byte of the packet.

[2] **RAM_OVF**: RAM overflow
0: Indicates that the RAM block has not overflowed.
1: Indicates that the RAM block has overflowed at some point and is sticky.

[1] **IN_FIFO_OVF**: Input FIFO overflow
0: Indicates that the input FIFO has not overflowed.
1: Indicates that the input FIFO has overflowed at some point and is sticky.

[0] **STREAM_LCK**
0: Indicates that a sync lock and drop stream is not locked.
1: Indicates that a stream is locked when using sync lock and drop or that the stream is using packet clocks for start of packet detection.

TSM_PTI_DEST PTI stream destination



Address: *TSMergerBaseAddress* + 0x0200
Type: R/W
Reset: 0
Description:

[31:5] **Reserved**

[4] **STREAM_4**
0: Stream 4(PTI_ALT_OUT) does not go to the PTI 1: Stream 4(PTI_ALT_OUT) goes to the PTI

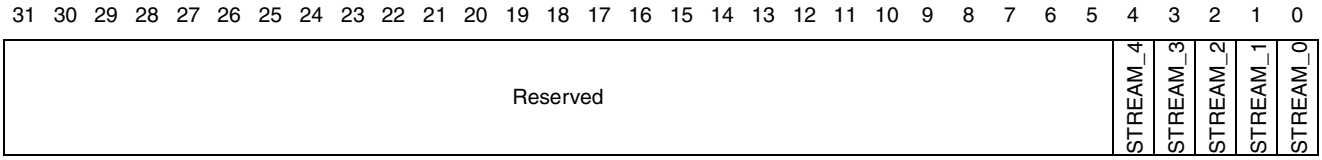
[3] **STREAM_3**
0: Stream 3 (SWTS0) does not go to the PTI 1: Stream 3 (SWTS0) goes to the PTI

[2] **STREAM_2**
0: Stream 2(TS2INOUT) does not go to the PTI 1: Stream 2 (TS2INOUT) goes to the PTI

[1] **STREAM_1**
0: Stream 1 (TS1IN) does not go to the PTI 1: Stream 1 (TS1IN) goes to the PTI.

[0] **STREAM_0**
0: Stream 0 (TS0IN) does not go to the PTI 1: Stream 0 (TS0IN) goes to the PTI.

TSM_1394_DEST 1394 interface stream destination



Address: *TSMergerBaseAddress* + 0x0210
Type: R/W
Reset: 0
Description:

[31:5] **Reserved**

[4] **STREAM_4**
0: Stream 4(PTI_ALT_OUT) does not go to the 1394 interface
1: Stream 4(PTI_ALT_OUT) goes to the 1394 interface

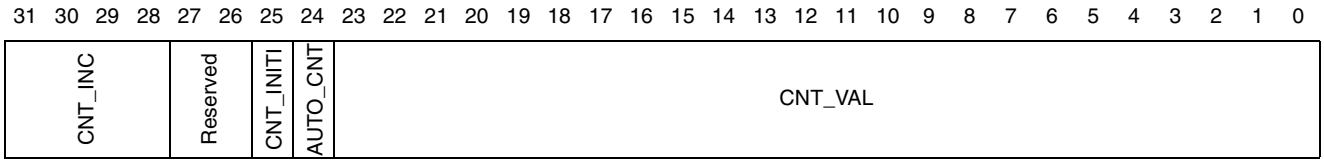
[3] **STREAM_3**
0: Stream 3 (SWTS0) does not go to the 1394 interface.
1: Stream 3 (SWTS0) goes to the 1394 interface

[2] **STREAM_2**
0: Stream 2(TS2INOUT) does not go to the 1394 interface
1: Stream 2 (TS2INOUT) goes to the 1394 interface

[1] **STREAM_1**
0: Stream 1 (TS1IN) does not go to the 1394 interface
1: Stream 1 (TS1IN) goes to the 1394 interface

[0] **STREAM_0**
0: Stream 0 (TS0IN) does not go to the 1394 interface
1: Stream 0 (TS0IN) goes to the 1394 interface

TSM_PROG_CNTn Program counter



Address: *TSMergerBaseAddress* + 0x0400, 0x0410

Type: R/W

Reset: 0

Description: Control and read programmable counter 0; used to dejitter SWTS0 (*n*=0) and PTI alternate output (*n*=1).

[31:28] **CNT_INC**: Counter increment step
Used to define increment step of the counter for each 27 MHz rising edge detected. This allows trick modes to be used with the counter dejittering mechanism. Setting this value to 0 means that the counter value in the header is not checked and that the counter is not incremented.

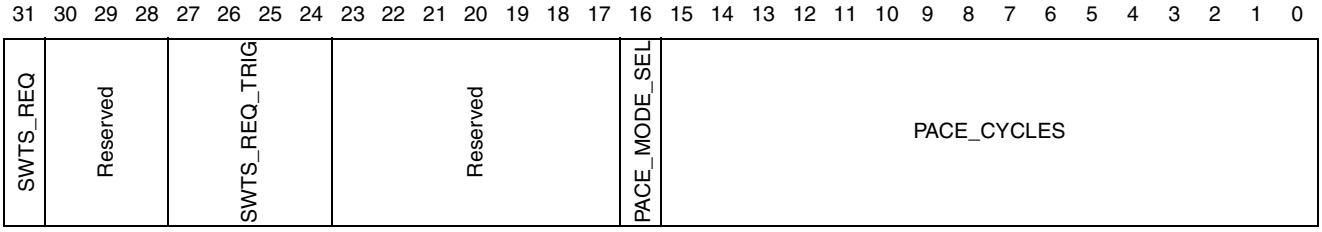
[27:26] **Reserved**

[25] **CNT_INIT**: Counter initialized
Only used when AUTO_CNT = 1
Transport stream merger sets this bit to 1 when the counter is initialized to a value of the counter in the first packet of a stream within RAM. Setting this bit to 0 reinitializes the counter to that of the counter in the first waiting packet in RAM,

[24] **AUTO_CNT**: Automatic counter
This bit defines the method of counter initialization.
0: The counter is used whatever is in the field CNT_VAL.
1: the counter is automatically set to the value in the header of the first packet in a stream.

[23:0] **CNT_VAL**: Counter value
Set the programmable counter value when store operation and read the current counter value with a load operation.

TS_SWTS_CFG SWTS configuration



Address: *TSMergerBaseAddress* + 0x0600

Type: Mixed

Reset: 0

Description:

[31] **SWTS_REQ**: Software request
This bit is asserted when at least half of the FIFO is empty. The software request stays asserted until the FIFO is over half-full. Read-only bit.

[30:28] **Reserved**

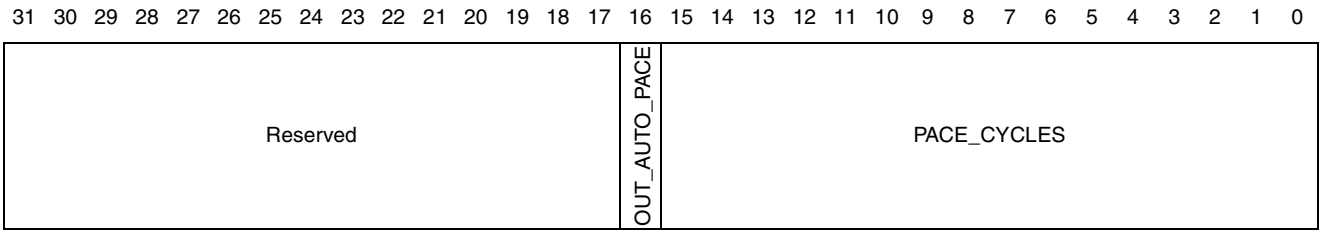
[27:24] **SWTS_REQ_TRIG**
Valid range 0000 to 1111. The SWTS request signal (O_SWTS_REQ) is high when the fill level falls below (SWTS_REQ_TRIG * 16) bytes. Trigger levels are 0, 16, 32, 64, 96 up to 224, 240, 256 bytes.

[23:17] **Reserved**

[16] **PACE_MODE_SEL**: Select pace mode of the SWTS data
0: Controlled by the 16 bits of the SWTS_PACE field.
1: Controlled by room in FIFOs and RAM circular buffers.

[15:0] **PACE_CYCLES**: Number of cycles between enable pulses for the SWTS data FIFO. If the FIFO is empty then the edge still occurs but the valid signal is not active.
Note: The pace field refers to byte frequency not word frequency.

TSM_PTI_ALT_OUT_CFG PTI alternate output configuration



Address: *TSMergerBaseAddress* + 0x0800

Type: R/W

Reset: 0

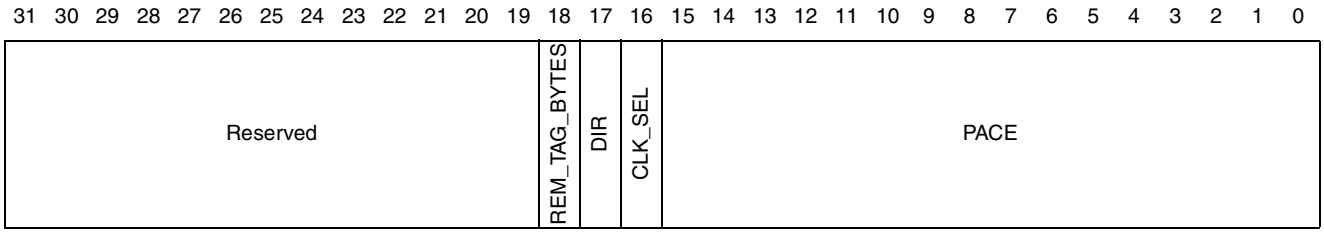
Description:

[31:17] **Reserved**

[16] **PACE_MODE_SEL**: Select pace mode of the PTI alternative output data requests
0: Controlled by the 16 bits of the OUT_PACE field.
1: Controlled by room in FIFOs and RAM circular buffers.

[15:0] **PACE_CYCLES**: Number of cycles between enable pulses on the PTI alternative output interface

TS_1394_CFG 1394 port configuration



Address: *TSMergerBaseAddress* + 0x0810

Type: R/W

Reset: 0

Description:

[31:19] **Reserved**

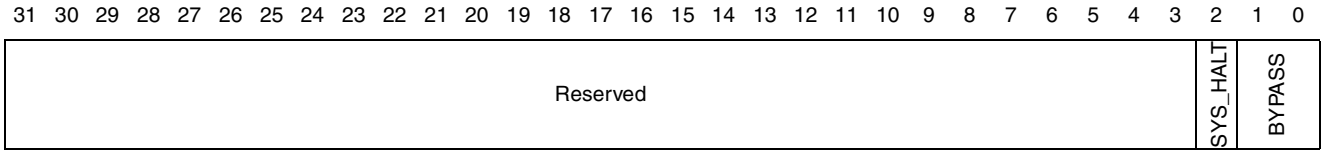
[18] **REM_TAG_BYTES**: 1394 remove tagging bytes
0: Outgoing stream is unmodified.
1: Outgoing stream has four bytes removed after start of packet byte.

[17] **DIR**: 1394 direction out not in
0: 1394 data ports are set to input.
1: 1394 data ports are set to output

[16] **CLK_SEL**: 1394 Clock selection
0: Selects the incoming I_1394_BYTECLK port as the clock with which to synchronize data output.
1: Selects an internally generated clock as a clock source for 1394 clock pad.

[15:0] **PACE**: 1394 Pace
Defines the divide ratio of CLK_SYS that the 1394 port produces to clock data out.
Only used if 1394_CLKSRC = 1

TS_SYS_CFG System configuration



Address: *TSMergerBaseAddress* + 0x0820

Type: R/W

Reset: 0

Description:

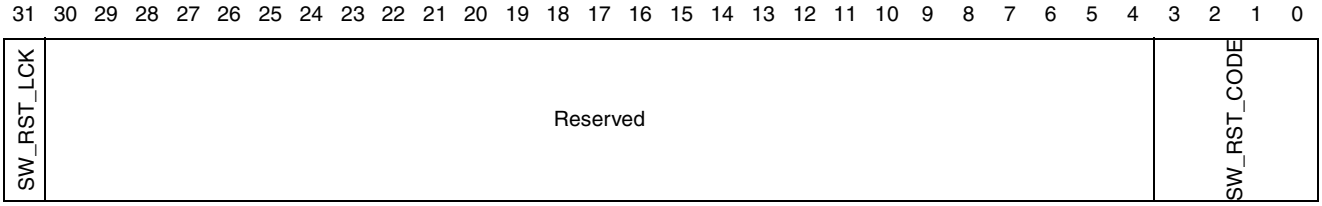
Note: Read and write accesses to the configuration registers are always allowed.

[31:3] **Reserved**

[2] **SYS_HALT**: Pause system
No read or write accesses to the central SRAM are allowed causing the system to halt.
0: The system is active.
1: The system is halted.

[1:0] **BYPASS**: Control bypass mode
See encoding below (the bit order is “<bit1><bit0>”):
00: The system route data through the SRAM and exhibit the full functionality of the TS merger
01: Behavior undefined.
10: The system is in bypass mode and routes the output of TSIS0 into the PTI.
11: The system is in bypass mode and routes the output of SWTS0 FIFO to the PTI directly.

TS_SW_RST Software reset



Address: *TSMergerBaseAddress* + 0x0830

Type: R/W

Reset: SW_RST_CODE: 0, SW_RST_LCK: 1

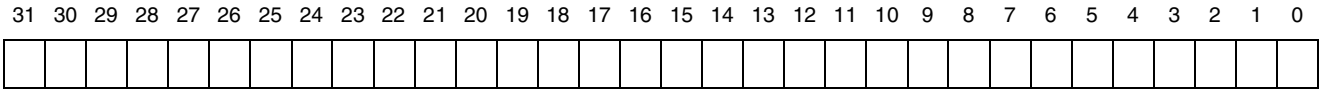
Description:

[31] **SW_RST_LCK**: Software reset lock
This controls access to the SW_RST_CODE to minimize an accidental software reset occurring. The lock bit must be reset in a different access to the access that sets SW_RST_CODE.

[30:4] **Reserved**

[3:0] **SW_RST_CODE**: Software reset code
If this 4-bit field is set to 0110 (0x6) when SW_RST_LCK = 0 then a software reset signal is pulsed for one CLK_SYS cycle, after which the SW_RST_CODE field returns to 0000 and SW_RST_LCK returns high.

TSM_SWTS Data for SWTS



Address: *TSMergerBaseAddress* + 0x010B E000

Type:

Reset:

Description:

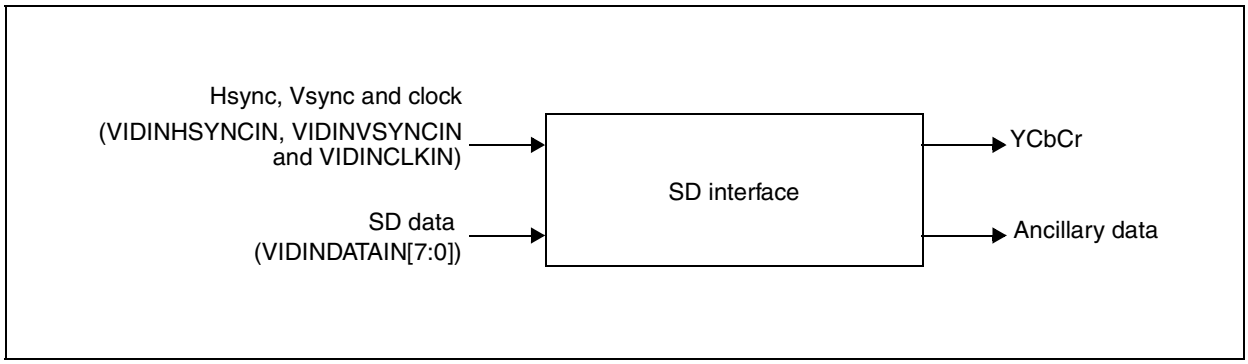
Part 6

Video

40 Digital video port (DVP) input

Video data enters the device through a standard-definition interface, the digital video port (DVP), which is described in this chapter.

Figure 100: Video input interface



40.1 Features

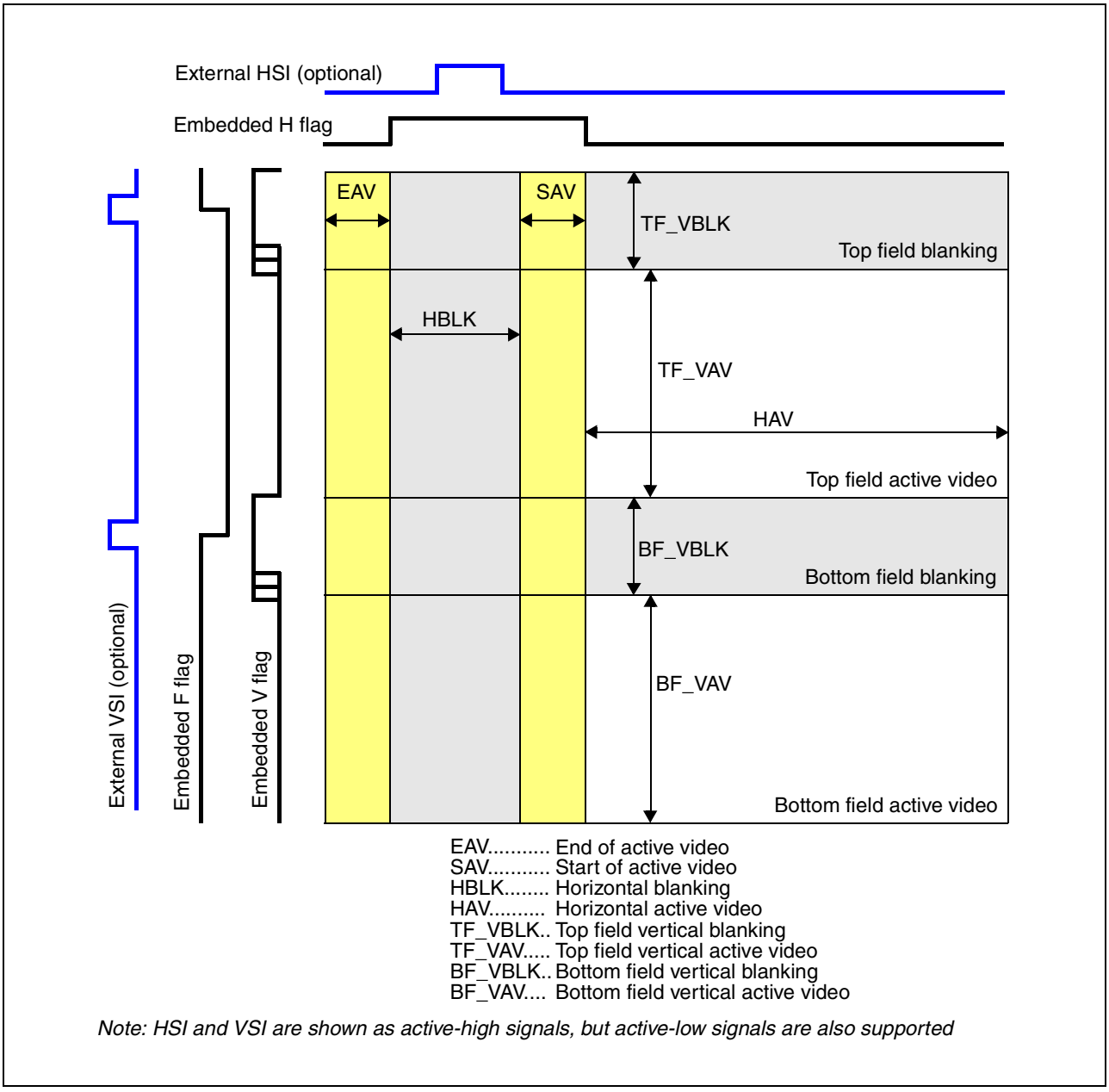
The main DVP features are:

- ITU-R BT.601 / 656 compliance, 525/60i 720x480, 625/50i 720x576 nominal formats.
- Support for SQ pixel video, 525/60i 640x480, 625/50i 768x576.
- External sync support for video streams that do not provide embedded code words (SAV/EAV protocol).
- Video data is captured into the system local memory, in YCbCr4:2:2 raster format (compatible with 2D-blitter as well as with the GDP display).
- User-defined capture window to select a subregion within the active area of the incoming video.
- Generic ancillary data capture processor (SMPTE291M, ITU-RBT-1364), in a paged circular buffer, for post-processing by a host CPU.

40.2 Video decoder

The frame architecture for an ITU-R BT.656 or ANSI/SMPTE 293M-1996 compliant video is shown in Figure 101. Y, Cb, Cr are multiplexed at twice the pixel rate on an 8-bit bus, according to a 4:2:2 sampling pattern

Figure 101: Interlaced video format derived from ITU-R BT.656



Note: HSI and VSI are shown as active-high signals, but active-low signals are also supported

An SAV and EAV header is made of four consecutive codewords, FF-00-00-XY: a codeword of all ones, two codewords of all zeros and a codeword including F (top field/bottom field), V (blanking/active), H (horizontal) and P3, P2, P1, P0 which are parity bits. The fourth codeword (XY) depends on the value of F, V and H; see Table 120

Table 120: XY codeword

bit 7 = 1	bit 6 = F	bit 5 = V	bit 4 = H	bit 3 = P3	bit 2 = P2	bit 1 = P1	bit 0 = P0	XY	SAV	EAV
1	0	0	0	0	0	0	0	0x80	x	
1	0	0	1	1	1	0	1	0x9D		x
1	0	1	0	1	0	1	1	0xAB	x	
1	0	1	1	0	1	1	0	0xB6		x
1	1	0	0	0	1	1	1	0xC7	x	
1	1	0	1	1	0	1	0	0xDA		x
1	1	1	0	1	1	0	0	0xEC	x	
1	1	1	1	0	0	0	1	0xF1		x

- SAV sequence is identified with H = 0 and EAV with H = 1.
- F and V can only change during EAV.
- Bit 7 is always set to 1.

Table 121 shows typical frame/field parameters for popular SD formats.

Table 121: ITU-R BT656 parameters for 525/60i and 625/50i formats

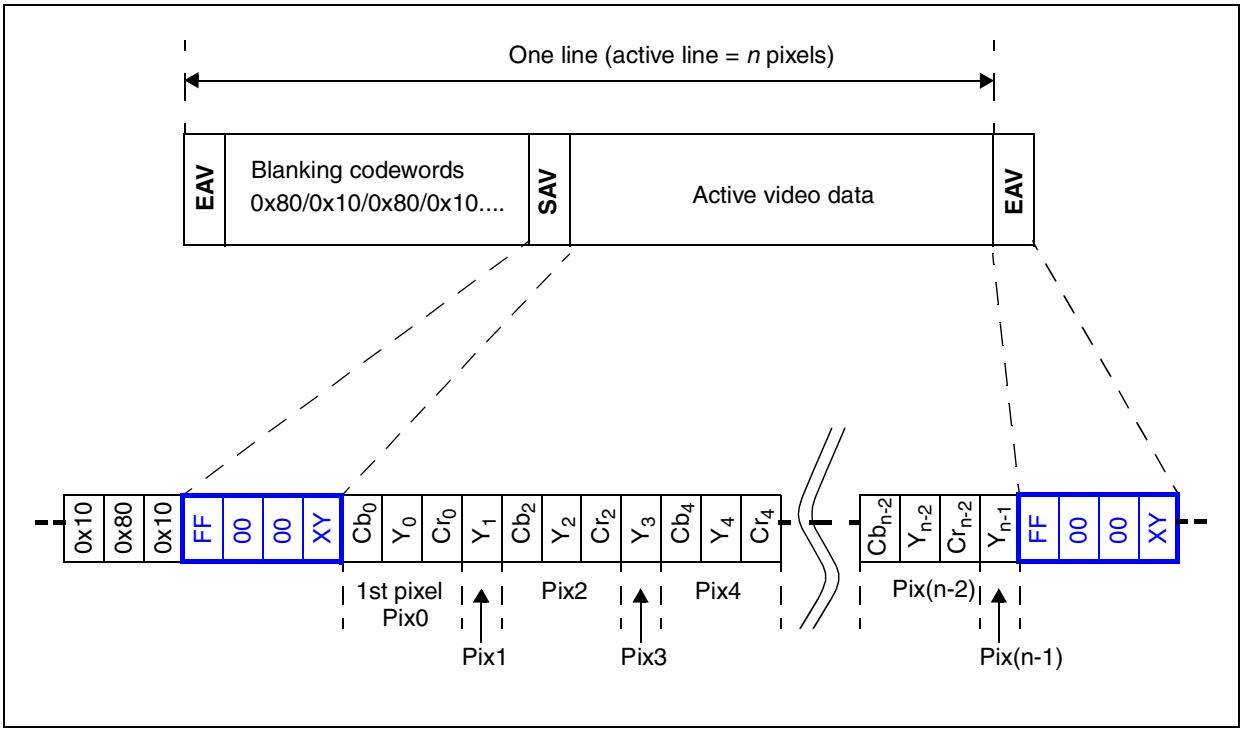
Standard	Parameter	Size	Units
ITU-R BT.656 525-line interlaced	PixClk frequency	13.5	MHz
	EAV	4	PixClk2X cycles
	SAV	4	
	HBLK	268	
	HAV	1440	
	TF_VBLK	19(23)	lines
	BF_BLK	19(22)	
	TF_VAV	244(240)	
	BF_VAV	243(240)	
ITU-R BT.656 625-line interlaced	PixClk frequency	13.5	MHz
	EAV	4	PixClk2X cycles
	SAV	4	
	HBLK	268	
	HAV	1440	
	TF_VBLK	24	lines
	BF_BLK	25	
	TF_VAV	288	
	BF_VAV	288	

There are no restriction on the values of the different parameters. The interface must support customized 656-like format: for example, 525 lines with 511 active lines (256 top + 255 bottom), or square pixel formats (640 x 480 active area or 768 x 576 active area), and so on.

The only limitation is the PixClock2X highest frequency: 29.5 MHz

The 8-bit multiplexed luma/chroma bus scheme is shown in Figure 102.

Figure 102: 4:2:2 multiplex scheme: byte ordering



- The first active pixel (pix0) is always a complete YCbCr pixel. The last active pixel (pix*n*) is nominally a Y-only pixel, but DVP must support external chroma decoders that generate a variable number of pixels per line (with a very poor quality analog source for instance).
- More generally, even-indexed pixel are YCbCr pixel (24 bits), odd-indexed pixel are Y-only pixel (8 bits). Average pixel size is 16-bit.
- Y nominal range is 16 .. 235 (black to white), 1 .. 254 is the valid range that can be supported by the DVP module.
- The Cb and Cr nominal range is 16 .. 240. 1 .. 254 is the valid range that can be supported by the DVP module.
- 0 and 255 are reserved words, forbidden as valid video data.
- XY parity errors are detected and corrected when possible.

40.3 Ancillary data decoder[1]

The ancillary data can be captured in a circular paged-buffer in external memory. The size of this buffer is defined by two registers: DVP_ABA (ancillary data base address), and DVP_AEA (ancillary data end address).

When the ancillary data capture process is enabled, any incoming packet is captured; the code identification is not taken into account and no filtering is done, as this parameter varies from one chroma decoder to another for the same kind of VBI data.

Note: The capture state machine is reset on the internal DVP VRef event. This means that:

- no ancillary data capture can be done during the first four lines of the vertical blanking interval.

- the CPU will have to identify the data once it is stored in the memory buffer. The CPU will either discard or decode a packet.

Two possible schemes can be used for managing the circular paged-buffer:

- **All pages have a fixed size**, as specified in register DVP_APS (ancillary data page size).
  The software handling is quite simple, but it cannot manage consecutive adjacent packets (some data may be lost). It can address basic (more common) applications, with only one or no packets per line. The page must be greater than the biggest expected packet (generally, of teletext data).
  Initially, the first byte location of each page in the buffer must be set to 0 by the CPU, and sets a track pointer on the first page. Then, on each DVP Vsync interrupt, the CPU checks whether a new data packet has been received, by checking the first location of the page currently pointed by the track pointer. If the byte value is 0, no new packet has been captured. If the value is 0xFF (the first value captured is always the second 0xFF byte of the header), then a new packet is there. The CPU must process this packet (it discards the packet or posts a message to another driver depending on the identification code), and reset the first byte location of the page. Then it jumps to the next page, repeats the operation, and so on until a 0 value is found as a first byte. Once that occurred, the track pointer is finally updated for the next Vsync interrupt.
- **The current page size always reflects the exact size on the current packet**.
  The length of the packet is extracted from the header (in 32-bit word units, from the sixth byte of the header {6 LSBs}), and the exact number of bytes is captured. Consecutive packets can be supported. In that case, the first byte to be captured is always the second 0xFF of the header, and the last byte to be captured is the checksum value (this hardware version does not control the checksum value).

1. See specifications (SMPTE291M, ITU-RBT-1364).

41 Digital video port (DVP) registers

Register addresses are provided as *DVPBaseAddress* + offset.

The *DVPBaseAddress* is:

0x1924 1000.

Table 122: DVP register summary

Register	Description	Offset	Type
DVP_CTRL	DVP control	0x00	R/W
DVP_TFO	X-Y top field offset	0x04	
DVP_TFS	X-Y top field stop	0x08	
DVP_BFO	X-Y bottom field offset	0x0C	
DVP_BFS	X-Y bottom field stop	0x10	
DVP_VTP	Video top field memory pointer	0x14	
DVP_VBP	Video bottom field memory pointer	0x18	
DVP_VMP	Video memory pitch	0x1C	
DVP_CVS	Captured video window size	0x20	
DVP_VSD	Vertical synchronization delay	0x24	
DVP_HSD	Horizontal synchronization delay	0x28	
DVP_HLL	Half line length	0x2C	
Reserved	-	0x30 - 0x94	-
DVP_ITM	Interrupt mask	0x98	R/W
DVP_ITS	Interrupt status	0x9C	RO
DVP_STA	DVP Status	0xA0	
DVP_LN_STA	Line number status	0xA4	
DVP_HLFLN	Half lines per vertical field	0xA8	R/W
DVP_ABA	Ancillary data base address	0xAC	
DVP_AEA	Ancillary data end address	0xB0	
DVP_APS	Ancillary data page size	0xB4	
Reserved	-	0xB8 - 0xF8	-
DVP_PKZ	Maximum packet size	0xFC	R/W

DVP_CTRL DVP control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DVP_RST	Reserved							BIG_NOT_LITTLE	Reserved			MIX_CAPT_PRE_REQ	MIX_CAPT_EN	MIX_CAPT_PHASE	EXTENDED-1_254	SYNCHRO_PHASE_NOTOK	Reserved	VALID_ANC_PAGE_SIZE_EXT	Reserved			ODDEVEN_NOT_VSYNC	PHASE		V_REF_POL	H_REF_POL	EXT_SYNC	VSYNC_BOT_HALF_LINE_EN	EXT_SYNC_POL	ANCILLARY_DATA_EN	VID_EN

Address: *DVPBaseAddress* + 0x00

Type: R/W

Buffer: Double-buffered: update on DVP V_INIT event, except if indicated by ** (immediate)

Reset: 0x8008 0000

Description: Provides the operating mode of the DVP pipe, for the current capture

If this register contents change, it must be loaded after all others registers have been loaded. After a hard reset, all registers must be loaded before DVP_CTRL, then DVP_RST must be cleared to be able to take into account synchronization inputs (external and embedded).

[31] **DVP_RST**
0: Synchro inputs enabled
1: Synchro input disabled (either external or embedded)

[30:24] **Reserved**

[23] **BIG_NOT_LITTLE**
0: Memory data in little endian format
1: Memory data in big endian format

[22:20] **Reserved**

[19] **MIX_CAPT_PRE_REQ**[a]: Mix capture pre-request
0: No compositor capture
1: Compositor capture enabled
Note: This unbuffered signal allows VTG_HREF*,* VTG_VREF *and* VTG_PIX2_CK *to be taken into account as input signals.*

[18] **MIX_CAPT_EN**
0: No compositor capture
1: Compositor capture enabled
Note: this double buffered signal allows starting compositor capture at the beginning of field.

[17] **MIX_CAPT_PHASE**
0: Compositor data resynchronized with positive edge of pixel clock
1: Compositor data resynchronized with negative edge of pixel clock

[16] **EXTENDED-1_254**[a]
0: Input clipping range: 16/235 for luma and 16/240 for chroma
1: Input clipping range: 1/254 for both luma and chroma

[15] **SYNCHRO_PHASE_NOTOK**[a]
0: External vertical and horizontal synchronization signals are presumed to be in phase to initialize a top field
1: External vertical and horizontal synchronization signals may be out of phase, in this case the vertical synchronization signal must be earlier or later than horizontal synchronization signal with a maximum of 1/4 of line length

[14] **Reserved**

[13] **VALID_ANC_PAGE_SIZE_EXT**
0: ANC_PAGE_SIZE extracted from ancillary data bit[5:0] from header sixth byte
1: ANC_PAGE_SIZE given by DVP_APS register

[12:10] **Reserved**

[9] **ODDEVEN_NOT_VSYNC**[a]
0: In external sync mode, vertical reference is a Vsync signal
1: In external sync mode, vertical reference is an odd/even signal

[8:7] **PHASE**: First pixel signification of capture window during 8-bit video data capture
phase[7] = 0 first pixel is complete (CB0_Y0_CR0)
phase[7] = 1 first pixel is not complete (Y1)
phase[8] = 0 number of pixel to capture is even
phase[8] = 1 number of pixel to capture is odd

[6] **V_REF_POL**[a]: Vref polarity
0: The negative edge of V_{REF} (if bit[4]=1) is taken as reference for the vertical counter reset.
1: The positive edge of V_{REF} (if bit[4]=1) is taken as reference for the vertical counter reset
Note: In EAV/SAV mode, (bit[4] = 0). the positive edge is always active but the normal rising edge of V is phased with the next active edge of H to respect the top field configuration

[5] **H_REF_POL**[a]: Href polarity
0: The negative edge of H_{REF} (defined by bit[4]) is taken as reference for the horizontal counter reset.
1: The positive edge of H_{REF} (defined by bit[4]) is taken as reference for the horizontal counter reset

[4] **EXT_SYNC**[a]: External sync
0: Extract the H/V/F reference sync flags from embedded EAV/SAV codes
(H becomes H_{REF}, V becomes V_{REF})
1: Use external HSI/VSI signals as sync reference (HSI becomes H_{REF}, VSI becomes V_{REF})

[3] **VSYNC_BOT_HALF_LINE_EN**
0: VSOUT starts at the beginning of the last top field line
1: VSOUT starts at the middle of the last top field line

[2] **EXT_SYNC_POL**[a]: External synchronization polarity
0: negative for both horizontal and vertical (no meaning if EXT_SYNC= 0)
1: positive for both horizontal and vertical

[1] **ANCILLARY_DATA_EN**
0: No ancillary data is captured
1: Ancillary data is captured into the device local memory

[0] **VID_EN**
0: No video data is captured
1: Video data is captured into the device local memory

a. Bits not double buffered (taken into account when loaded by CPU). Others bits are validated by vertical synchronization signal.

DVP_TFO X-Y top field offset

31 30 29 28	28 26 25 24 23 22 21 20 19 18 17 16	15 14 13	12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved	YDO	Reserved	XDO

Address: *DVP BaseAddress* + 0x04

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0x000 0000

Description: Provides the x, y location of the viewport top-left pixel for the top field, with respect to the (0,0) origin of the incoming video, as defined in the control register.

[31:28] **Reserved**

[26:16] **YDO**: Y location for the first line of the viewport (top), with respect to field numbering

[15:13] **Reserved**

[12:0] **XDO**: X location for the first pixel of the viewport (left), in sample unit

DVP_TFS X-Y top field stop

31 30 29 28	28 26 25 24 23 22 21 20 19 18 17 16	15 14 13	12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved	YDS	Reserved	XDS

Address: *DVP BaseAddress* + 0x08

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Provides the x, y location of the viewport bottom-right pixel for the top field, with respect to the (0,0) origin of the incoming video, as defined in the control register.

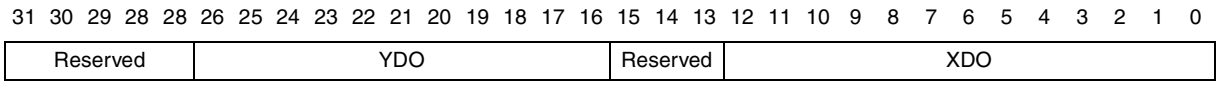
[31:28] **Reserved**

[26:16] **YDS**: Y location for the last line of the viewport (bottom), with respect to field numbering

[15:13] **Reserved**

[12:0] **XDS**: X location for the last pixel of the viewport (right), in sample unit

DVP_BFO X-Y bottom field offset



Address: *DVP BaseAddress* + 0x0C
Type: R/W
Buffer: Double-buffered, update on DVP V_INIT event
Reset: 0
Description: Provides the x, y location of the viewport top-left pixel for the bottom field, with respect to the (0,0) origin of the incoming video, as defined in the control register.

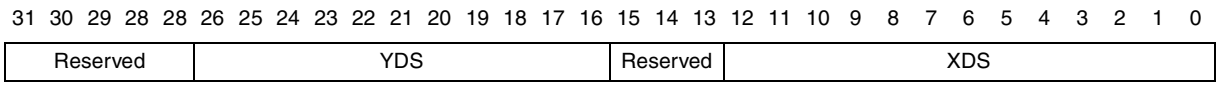
[31:28] **Reserved**

[26:16] **YDO**: Y location for the first line of the viewport (top), with respect to field numbering

[15:13] **Reserved**

[12:0] **XDO**: X location for the first pixel of the viewport (left), in sample unit

DVP_BFS X-Y bottom field stop



Address: *DVP BaseAddress* + 0x10
Type: R/W
Buffer: Double-buffered, update on DVP V_INIT event
Reset: 0
Description: Provides the x, y location of the viewport bottom-right pixel for the bottom field, with respect to the (0,0) origin of the incoming video, as defined in the control register.

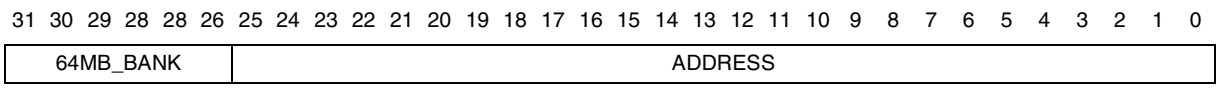
[31:28] **Reserved**

[26:16] **YDS**: Y location for the last line of the viewport (bottom), **with respect to field numbering**

[15:13] **Reserved**

[12:0] **XDS**: X location for the last pixel of the viewport (right), **in sample unit**

DVP_VTP Video top field memory pointer



Address: *DVP BaseAddress* + 0x14
Type: R/W
Buffer: Double-buffered, update on DVP V_INIT event
Reset: 0
Description: Contains the memory location for the top field first pixel to be stored (top-left corner).

[31:26] **64_MBYTE_BANK**: 64 MByte bank number

[25:0] **ADDRESS**: Top field first pixel byte address, in the selected 64 MByte bank
Note: the whole captured video frame must be totally included into the same bank.

DVP_VBP Video bottom field memory pointer

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64MB_BANK						ADDRESS																									

Address: *DVP BaseAddress* + 0x18
Type: R/W
Buffer: Double-buffered, update on DVP V_INIT event
Reset: 0
Description: Contains the memory location for the bottom field first pixel to be stored (top-left corner).

[31:26] **64_MBYTE_BANK**: 64 MByte bank number

[25:0] **ADDRESS**: Bottom field first pixel byte address, in the selected 64 MByte bank
Note: the whole captured video frame must be totally included into the same bank.

DVP_VMP Video memory pitch

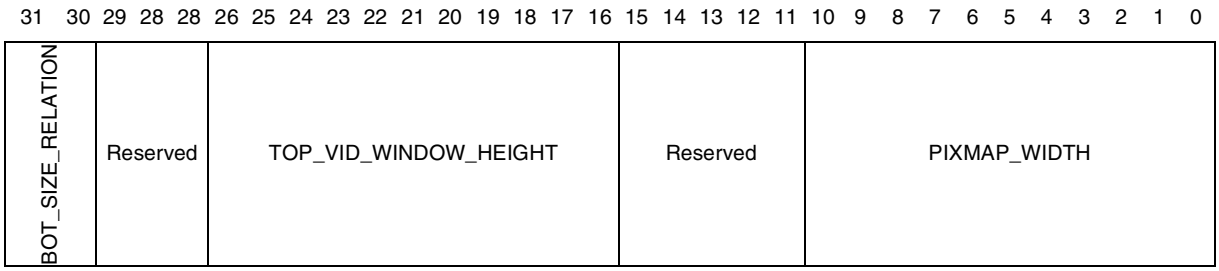
31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			PITCH_VAL												

Address: *DVP BaseAddress* + 0x1C
Type: R/W
Buffer: Double-buffered, update on DVP V_INIT event
Reset: 0
Description: Contains the memory pitch for the captured video, as stored in the memory.

[31:13] **Reserved**

[12:0] **PITCH_VAL**: Memory pitch for the captured video.
Note: the pitch is the distance inside the memory, in bytes, between two vertically adjacent pixels within a field

DVP_CVS Captured video window size



Address: *DVP BaseAddress* + 0x20

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Provides the size of the displayed pixel map attached to the viewport.

[31:30] **BOT_SIZE_RELATION**
00: bottom size equals top size
01: bottom size equals top size plus 1
10: bottom size equals top size minus 1

[31:28] **Reserved**

[26:16] **TOP_VID_WINDOW_HEIGHT**
Pixmap height, in lines, being defined as the number of lines that must be read from memory for the current field in an interlaced display

[31:28] **Reserved**

[10:0] **PIXMAP_WIDTH**: Pixmap width in pixels

DVP_VSD Vertical synchronization delay

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VSYNC_V_BOT_DLY												Reserved												VSYNC_V_TOP_DLY										

Address: *DVP BaseAddress* + 0x24

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Provides the delay (in number of lines) between the input original vertical reference (VSI or embedded synchronization V signal), and the vertical reference provided by the DVP to the VTG.

Details (VSYNC_H_DLY value is given in DVP_HSD register; see next register, DVP_HSD):

TOP FIELD:

VSYNC_H_DLY is the delay of vsout from sample counter clear,

VSYNC_V_TOP_DLY is the delay from line counter clear;

The total delay from sample counter clear is:

VSYNC_H_DLY + VSYNC_V_TOP_DLY * NUM_SAMPLES_PER_LINE

BOTTOM FIELD:

External synchronization or
(internal synchronization with VSYNC_BOT_HALF_LINE_EN = 0):

VSYNC_H_DLY is the delay of VSOUT in samples units from external vertical synchronisation input.

VSYNC_V_BOT_DLY is the delay from line counter clear;

the total number of samples delay from vertical synchronization input is:

VSYNC_H_DLY + VSYNC_V_BOT_DLY * NUM_SAMPLES_PER_LINE

internal synchronization with VSYNC_BOT_HALF_LINE_EN = 1 this delay becomes:

VSYNC_H_DLY + VSYNC_V_BOT_DLY * NUM_SAMPLES_PER_LINE + HALF_LINE_LEN

to provide VSOUT on bottom field at the middle of last line of top field

Note: 1 VSYNC_V_TOP_DLY *and* VSYNC_V_BOT_DLY *values* ***must*** *be lower than the number of lines per field*

*2 This register is not double buffered (*VSYNC_V_TOP_DLY *and* VSYNC_V_BOT_DLY *are taken into account when loaded by the CPU)*

[31:28] **Reserved**

[26:16] **V_BOT_DLY**: Vertical delay for bottom field, in lines

[15:11] **Reserved**

[10:0] **V_TOP_DLY**: Vertical delay for top field, in samples

DVP_HSD Horizontal synchronization delay

31 30 29	28 28 26 25 24 23 22 21 20 19 18 17 16	15 14 13	12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved	VSYNC_H_DLY	Reserved	HSYNC_H_DLY

Address: *DVP BaseAddress* + 0x28

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Provides the size of the horizontal delay of HSOUT:

HSYNC_H_DLY is the delay of hsout in samples units from the sample counter clear signal

VSYNC_H_DLY is the delay of VSOUT in samples units from the sample counter clear signal (see DVP_VSD).

Note: HSYNC_H_DLY *and* VSYNC_H_DLY *values* ***must*** *be lower than the number of samples per line.*

Note: *This register is not double buffered (*VSYNC_V_TOP_DLY *and* VSYNC_V_BOT_DLY *are taken into account when loaded by the CPU).*

[31:29] **Reserved**

[28:16] **VSYNC_H_DLY**: Horizontal delay of HSOUT, expressed as number of 27 MHz samples

[15:13] **Reserved**

[12:0] **HSYNC_H_DLY**: Horizontal delay of hsout, expressed as number of 27 MHz samples (see DVP_VSD)

DVP_HLL Half line length

31 30 29	28 28 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
Reserved		HALF_LINE_LEN

Address: *DVP BaseAddress* + 0x2C

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

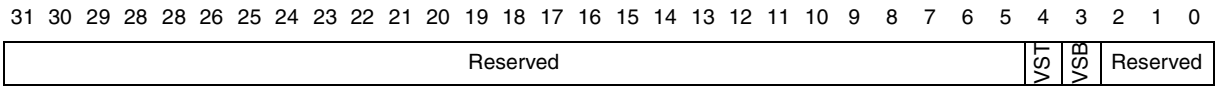
Description: Provides the number of samples for in a half line. (This is used internally to have VSOUT shifted one half line after HSOUT, in embedded synchronization mode when VSYNC_BOT_HALF_LINE_EN = 1 in the control register).

Note: *This is equivalent to the number of pixels per line.*

[31:28] **Reserved**

[11:0] **HALF_LINE_LEN**: Half line length, in samples

DVP_ITM Interrupt mask



Address: *DVPn BaseAddress* + 0x98

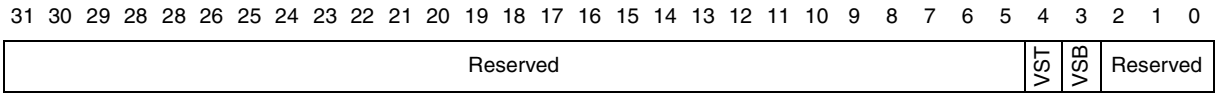
Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Any bit set in this register enables the corresponding interrupt in the DVP_IRQ line. An interrupt is generated whenever a bit in the DVP_STA register changes from 0 to 1 and the corresponding mask bit is set.

DVP_ITS Interrupt status



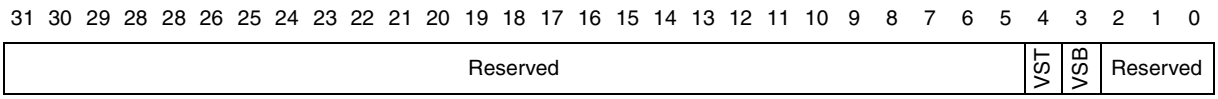
Address: *DVPn BaseAddress* + 0x9C

Type: RO

Reset: 0

Description: When a bit in the DVP_STA register changes from 0 to 1, the corresponding bit in this register is set, independent of the state of DVP_ITM. If any set bit is unmasked, the line DVP_IRQ is asserted. The reading of DVP_ITS clears all bits in that register. When DVP_ITS is zero, the DVP_IRQ line is deasserted.

DVP_STA DVP Status



Address: *DVPn BaseAddress* + 0xA0

Type: RO

Reset: 0

Description: Any change from 0 to 1 of these bits sets the corresponding bit of the DVP_ITS register, and can thus potentially cause an interrupt on DVP_IRQ line. VST and VSB are pulses and are unlikely ever to be read as a 1.

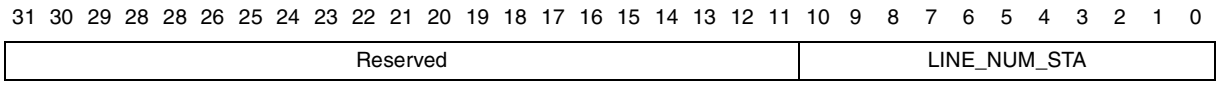
[31:5] **Reserved**

[4] **VST**: Vsync top
Set for a short time at the beginning of the top field, corresponding to the falling edge of the internal BNOTT signal.

[3] **VSB**: Vsync bottom
Set for a short time at the beginning of the bottom field, corresponding to the rising edge of the internal BNOTT signal.

[2:0] **Reserved**

DVP_LN_STA Line number status



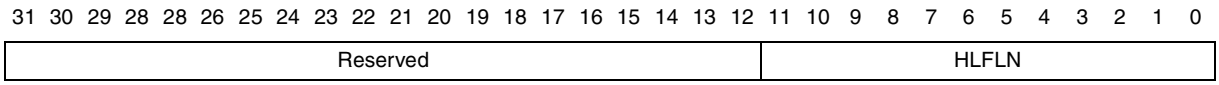
Address: *VTG BaseAddress* + 0xA4

Type: RO

Reset: 0

Description: This register is loaded on VTG V_{REF} output signal (vertical synchronization) with the value of DVP line counter

DVP_HLFLN Half lines per vertical field



Address: *VTG BaseAddress* + 0xA8

Type: R/W

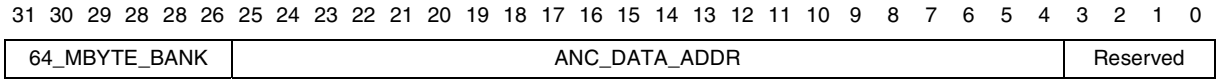
Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: This register specifies the number of half lines per vertical period.

This value is used when the bit SYNCHRO_PHASE_NOT_OK is set, then the vertical synchronization is forced to be in phase with the horizontal synchronization signal to generate a top field, the external vertical signal is ignored.

DVP_ABA Ancillary data base address



Address: *DVP BaseAddress* + 0xAC

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP ancillary data base address register contains the memory location for the first ancillary data page. The buffer must be aligned on a 128-bit word address boundary.

[31:26] **64_MBYTE_BANK**: 64 MByte bank number

[25:4] **ANC_DATA_ADDR**: Base address for the ancillary data buffer, in the selected 64 MByte bank

[3:0] **Reserved**: Write 0

DVP_AEA Ancillary data end address

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64_MBYTE_BANK						ANC_DATA_ADDR																						Reserved			

Address: *DVP BaseAddress* + 0xB0

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Contains the last memory location for the ancillary data buffer (128-bit word address, then the address counter wraps to the base address location).

[31:26] **64_MBYTE_BANK**: 64 MByte bank number (programmed value must match the DVP_ABA bank)

[25:4] **ANC_DATA_ADDR**: Base address for the ancillary data buffer, in the selected 64 MByte bank

[3:0] **Reserved**: write 0

DVP_APS Ancillary data page size

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				PAGE_SIZE								Reserved			

Address: *DVP BaseAddress* + 0xB4

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Provides the size in bytes, of the memory page used for a single ancillary data packet. The size granularity is 16 bytes, the maximum size is 4096 bytes.

[31:12] **Reserved**

[11:4] **PAGE_SIZE**: Memory size allocated to an ancillary data packet

[3:0] **Reserved**: write 0

DVP_PKZ Maximum packet size

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													PKT_SIZE		

Address: *DVP BaseAddress* + 0xFC

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: Controls the maximum size of a data packet during an STBus transaction. Must be set to 0.

[31:3] **Reserved**

[2:0] **PKT_SIZE**: Maximum packet size during an STBus transaction

000: Message size
001: 16 STBus words
010: 8 STBus words
011: 4 STBus words
100: 2 STBus words
101: 1 STBus words

42 MPEG2 video decoder

42.1 Overview

The STx7100 contains a real-time MPEG1 and MPEG2 video decoder. The decoder is compliant with the main profile at high level specified in ISO/IEC 11172_2 and ISO/IEC 13818-2.

As a picture (frame or field) decoder, the MPEG2 video decoder needs CPU support for each picture.

When a decoding task is launched by the CPU, the MPEG2 video decoder starts to read the video elementary stream stored in an external memory through the interconnect. The area in memory where the stream is stored is called the bit buffer. The decoded picture is reconstructed (written in external memory) in macroblocks (MBs). Each area in memory where a decoded picture is stored is called a frame buffer.

An interrupt informs the CPU when decode is completed or when a problem occurs. The decoder stops automatically when:

- all macroblocks have been decoded and it has found a start code different from a slice start code, or
- when it has reached a programmed read limit.

The MPEG2 video decoder supports three extra features which are not part of the MPEG2 standard:

- overwrite mode, see Section 42.5.3: *Picture decoding configuration* *on page 400*,
- possible simplified B decoding, see Section 42.5.7: *Simplified B decoding* *on page 407*,
- error statistics, see Section 42.5.6: *Error statistics* *on page 406*.

42.2 Main functions

Figure 103 shows the video decoder in a typical environment.

Figure 103: MPEG2 video decoder in the STx7100

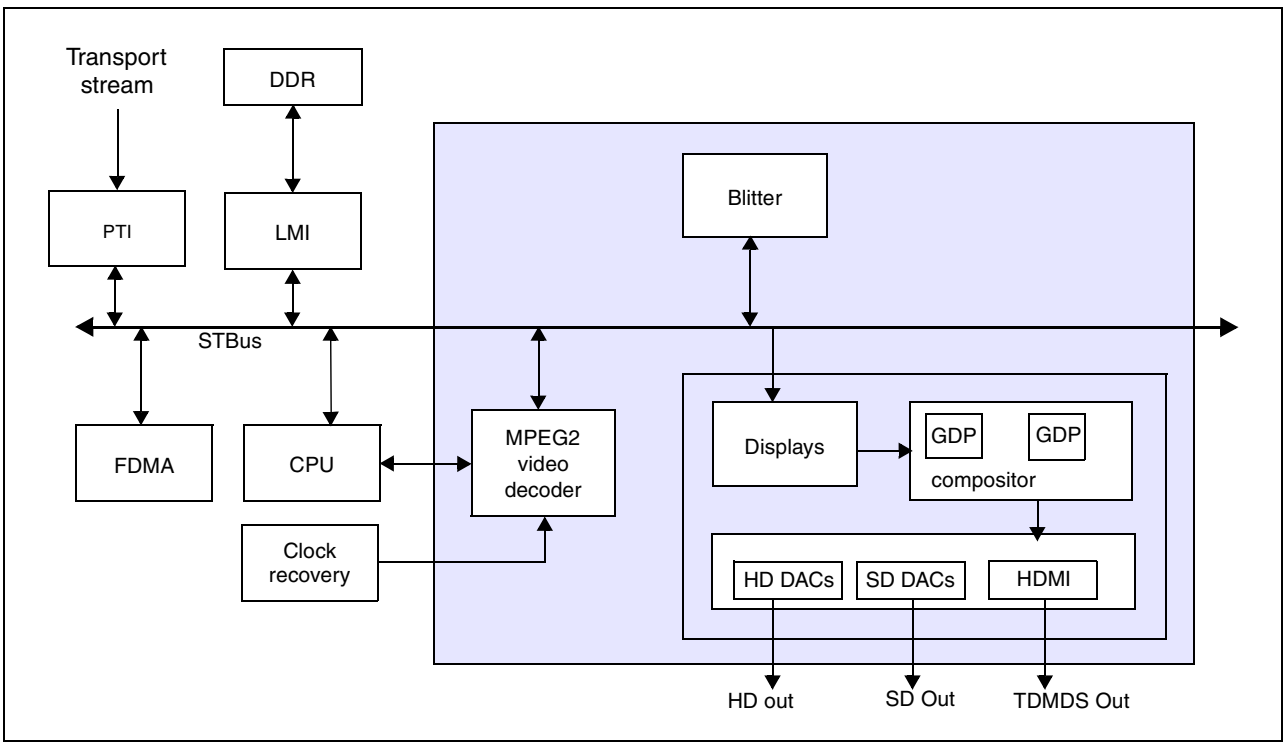
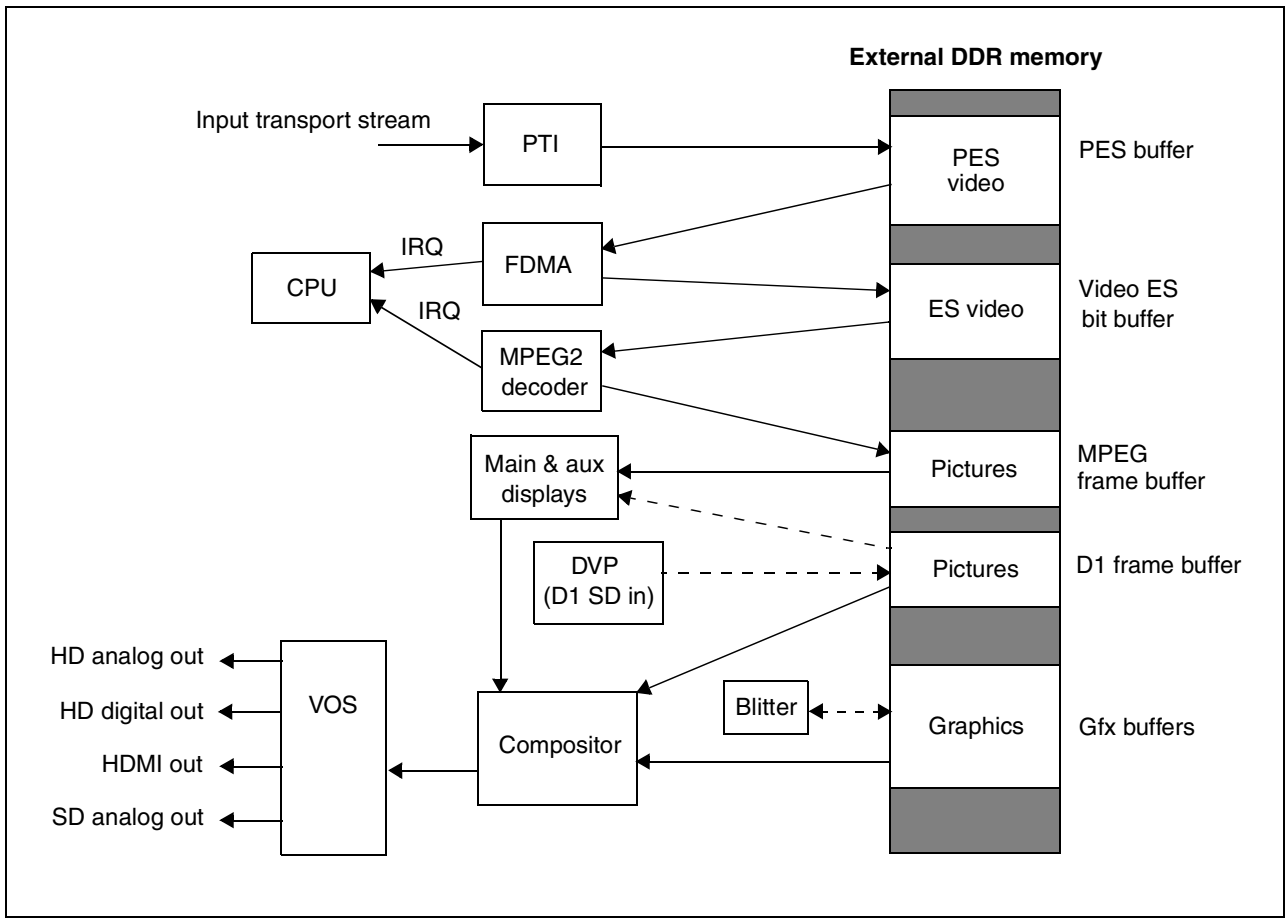


Figure 104 shows an example of dataflow around the MPEG2 video decoder at the system level.

Figure 104: System video data flow



- The PTI extracts the packet of elementary streams (PES) video from the input stream and sends them into the external memory through the interconnect.
- The PES are read back by the FDMA to extract the video elementary stream (ES). Each part of the stream corresponding to a picture is written into memory in video linear picture bit buffers.
- The ES is read by the MPEG2 video decoder, decoded and reconstructed into frame buffers.
- Frame buffers are accessed by display(s).

42.3 Buffer organization

There are two types of buffer in the MPEG2 decoder:

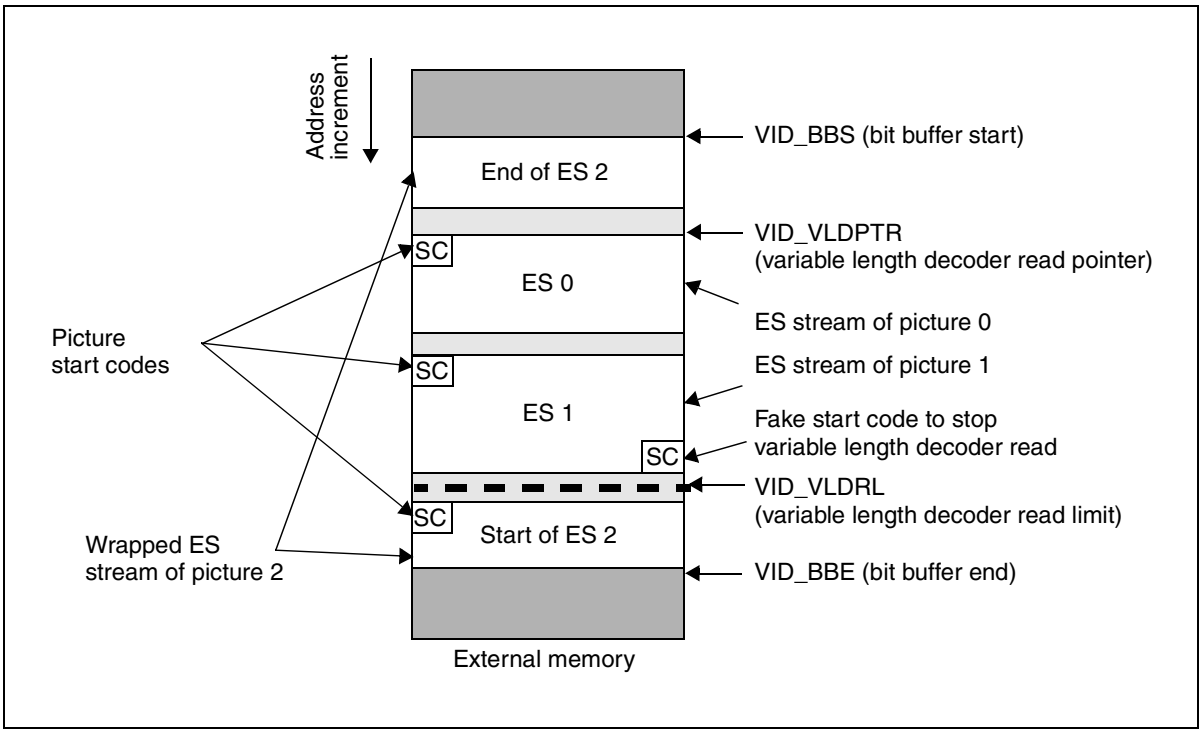
- **bit buffer**: area in external memory where the bitstream is stored,
- **frame buffer**: area in external memory in which decoded pictures are stored.

42.3.1 Bit buffer organization

The FDMA writes the ES of each picture in external memory. Each picture ES may be stored linearly anywhere in the memory.

The use of VID_BBS and VID_BBE is optional. It allows a memory area to be defined in which ES can be wrapped (see Figure 105).

Figure 105: Bit buffer organization



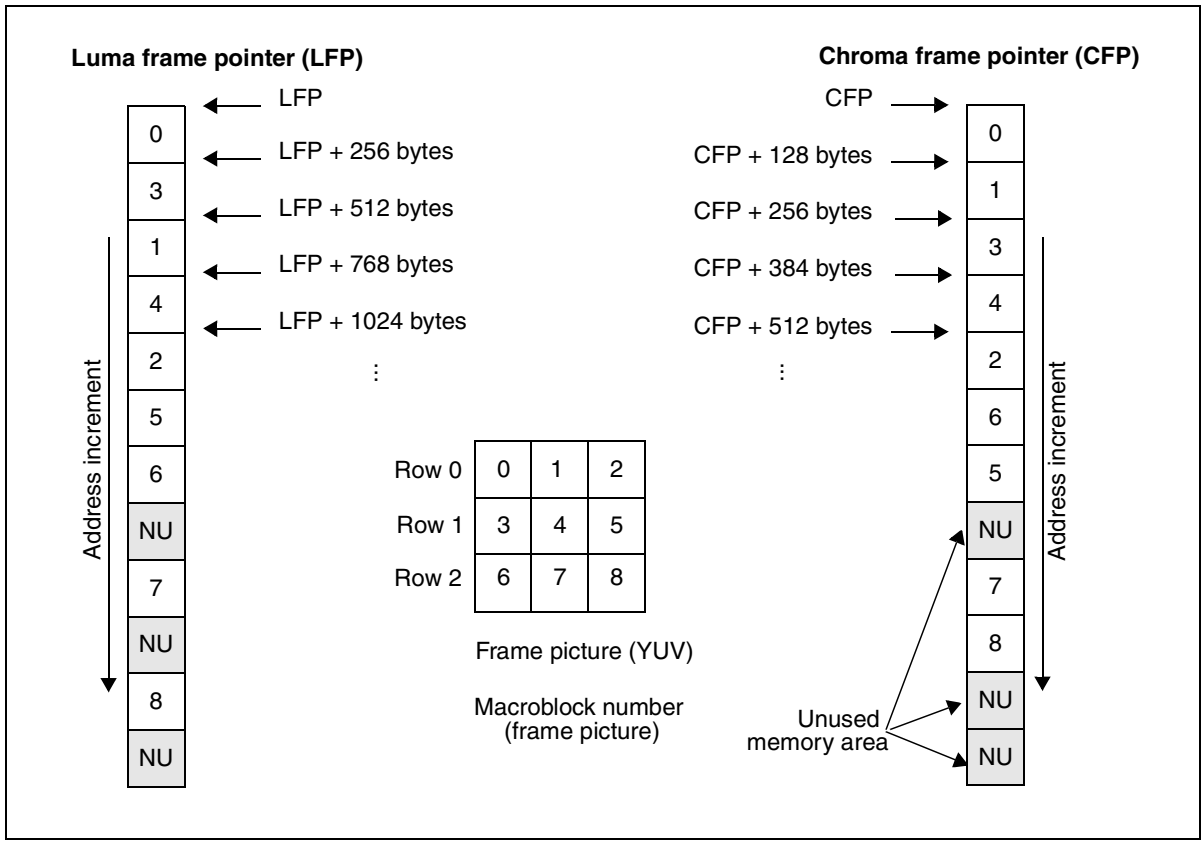
When a decode is launched, the variable length decoder reads data until it reaches a start code different from a slice start code. It can be the picture start code of the subsequent picture, or a fake start code added at the end of the picture to stop the variable length decoder. The variable length decoder read process can also be stopped with VID_VLDRL (see *Stream pointers for variable length decoder read access* *on page 401*).

42.3.2 Frame buffer organization

The video decoder stores both frame and field pictures in frame buffers, the size of which are dependent on picture size/resolution and decimation factors (see the *LumaBufferSize* and *ChromaBufferSize* equations below). However a frame is reconstructed (whether as one frame picture or two field pictures), the final buffer has the same frame organization. When a frame picture is reconstructed, the whole buffer is filled. When a field picture is reconstructed, half the buffer is filled. The other half is filled after the second field is reconstructed.

Figure 106 shows an example of a frame buffer with an odd width and height (3 x 3 macroblocks).

Figure 106: Frame buffer organization, mapping of 3 x 3 macroblocks



Where there is an odd number of vertical or horizontal macroblocks, memory is wasted, and this has to be taken into account when allocating memory space for frame buffers.

The formulas below give the size of frames buffers depending on macroblock width and height:

- *LumaBufferSize* = *MBwidth* x ((*MBheight* + 1) DIV 2) x 512
- *ChromaBufferSize* = ((*MBwidth* + 1) DIV 2) x ((*MBheight* + 1) DIV 2) x 512

where DIV corresponds to the integer part of a division, so *x* DIV *y* = integer part of (*x*/*y*)

A luminance macroblock (Y) contains 16 words of 128 bits. Inside this macroblock, data is grouped by field and stored in the order:

- left part of top field,
- right part of top field,
- left part of bottom field,
- right part of bottom field.

A description is given in Figure 107.

Chroma macroblock (Cb + Cr) data contains 8 words of 128 bits. The chroma macroblocks are stored in the same order as luma data (as listed above) but a word of 128 bits contains 8 bytes of Cr data and 8 bytes of Cb data. A description is given in Figure 108.

Figure 107: Y macroblock storage

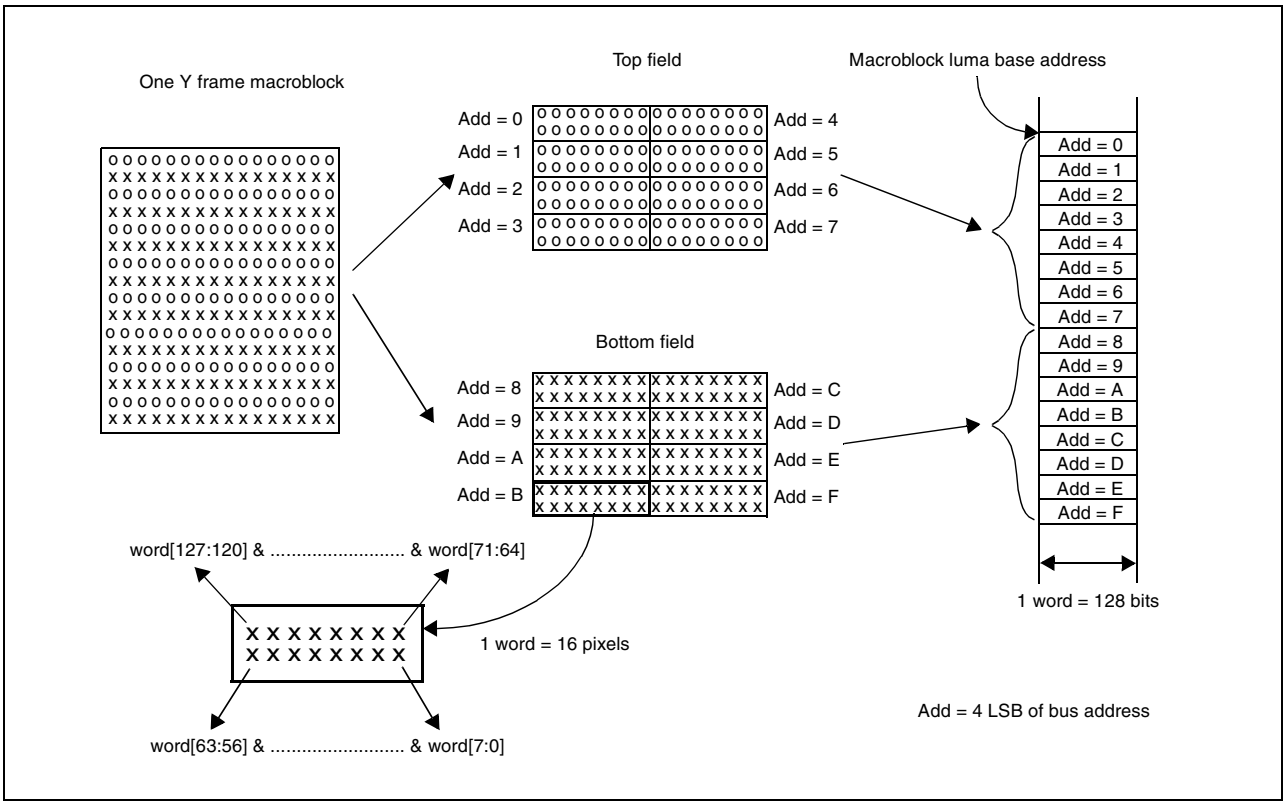
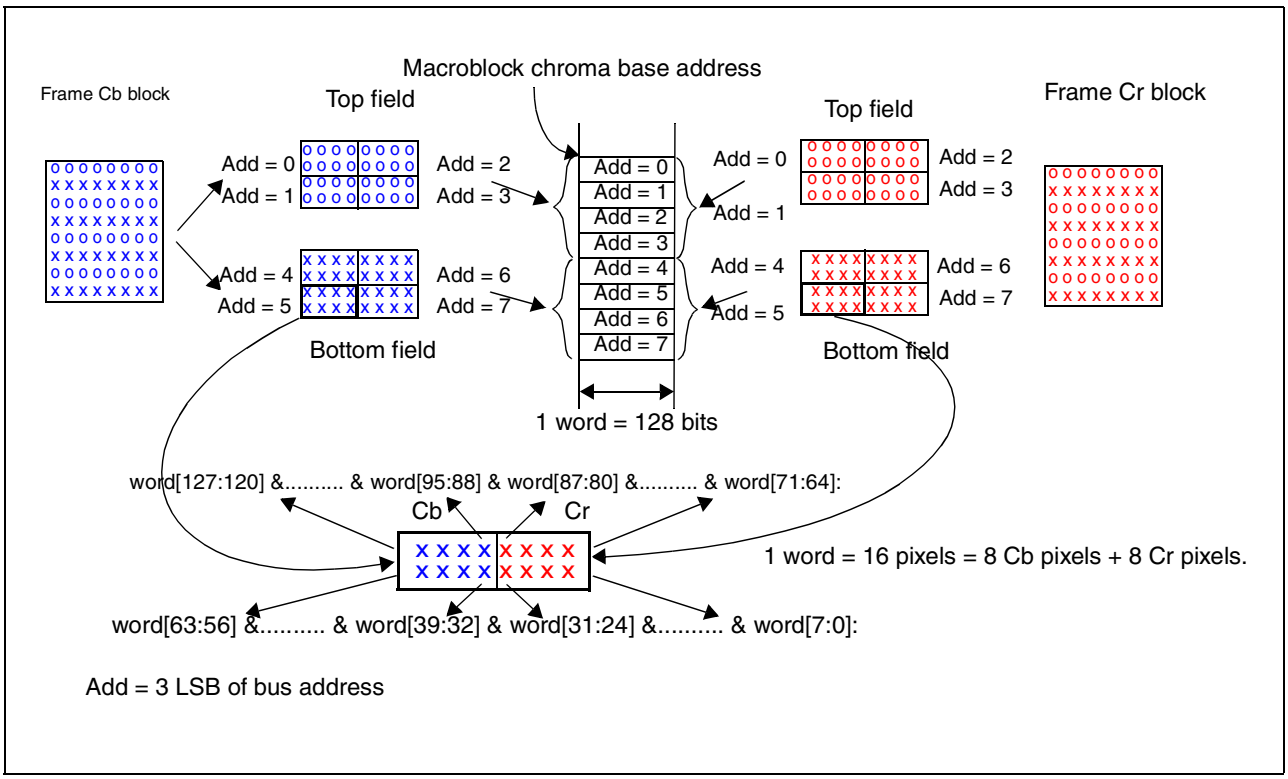


Figure 108: Cr/Cb macroblock storage



Address calculation in frame buffer

Macroblock position to address formula

MBnb = MB number (from 0 to *frame_size* - 1)

MBrow = MB row (from 0 to *frame_height* - 1)
= MBnb DIV *frame_width*

MBcol = MB column (from 0 to *frame_width* - 1)
= MBnb MOD *frame_width*

Luma base address of the macroblock (MBrow, MBcol)

YMBba = Luma macroblock base address
= ((MBrow DIV 2) x *frame_width* + MBcol) x 512 +
(MBrow MOD 2) x 256

Chroma base address of the macroblock (MBrow, MBcol)

CMBba = Chroma macroblock base address
= (((MBrow DIV 2) x *frame_width* + MBcol) DIV 2) x 512 +
(MBrow MOD 2) x 256 +
(((MBrow DIV 2) x *frame_width* + MBcol) MOD 2) x 128

Address to macroblock position formula

For Luma:

MBrow = ((address DIV 512) DIV *frame_width*) * 2 + ((address DIV 256) MOD 2)

MBcol = (address DIV 512) MOD *frame_width*

For chroma

If ((*frame_width* MOD 2) = 1) AND (((address DIV 512) MOD *frame_width*) = (*frame_width* DIV 2)) then

MB_row= (((address DIV 512) * 2) DIV *frame_width*) * 2 +
((address DIV 256) MOD 2) +
((address DIV 128) MOD 2) * 2

else

MB_row= (((address DIV 512) * 2) DIV *frame_width*) * 2 +
((address DIV 256) MOD 2)

end if

MBcol = (((address DIV 512) * 2) + ((address DIV 128) MOD 2)) MOD *frame_width*

42.3.3 Memory requirement

Video decoding requires a large amount of external memory for frame buffers and bit buffers. The exact need depends on the application.

Frame buffers

The maximum size of a frame buffer, for HD PAL pictures (1920 x 1152 pixels), is 26.55 Mbits (17.7 Mbits for luma frame buffer and 8.85 Mbits for chroma frame buffer, see Section 42.3.2 for details).

A minimum of three frame buffers is required to decode field pictures. A minimum of four is needed for frame pictures. The overwrite feature allows only three frame buffers to be used even with frame-encoded pictures.

For trick modes (slow, normal or fast, backward or forward play), the number of frame buffers is much higher, depending on the speed of the decoder and the required quality of fluidity.

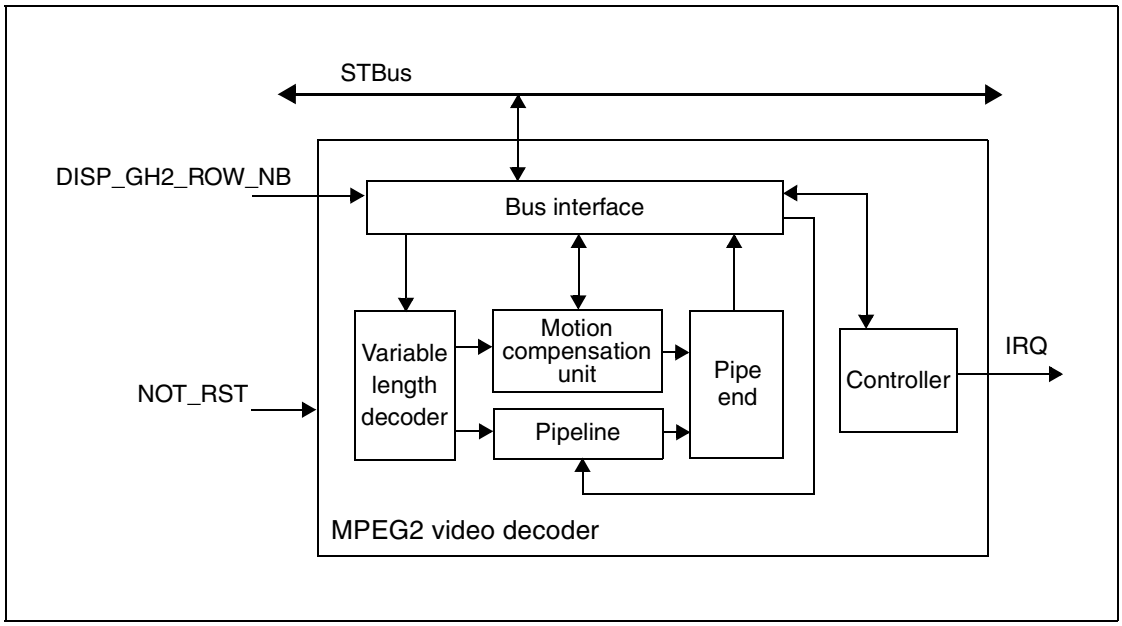
Bit buffer

The memory need in terms of bit buffer is application dependant. The constraint imposed by the standard (ISO 13818-2 annex C), called video buffering verifier (VBV) can be applied on the ES or on the PES buffer. If it is applied on the PES buffer, the application must take into account the PES headers' size.

42.4 Video decoding tasks

The decoding of a single picture is known as a task. It is specified by the task instruction set up before the decoding of each picture in register VID*n*_TIS.

Figure 109: MPEG2 video decoder block diagram



42.4.1 Controller

The control of decoding tasks is managed by the controller block. The controller block:

- holds the VID_EXE register to execute a decoding task and manages its synchronization on both clock domains,
- holds the VID_SRS register to do a soft reset,
- provides a set of interrupts which gives the status of the decoding process.

42.4.2 Variable length decoder

The variable length decoder is the first stage of the decoding pipeline. The bitstream is read from the bit buffer into the variable length decoder FIFO.

The variable length decoder performs the following functions:

- variable length decodes the slice layer, macroblock layer and block layer,
- decodes the macroblock address increment and process skipped macroblock,
- decodes the motion vectors and delivers them to the motion compensation unit,
- detects bitstream errors and conceals corrupted and missing data (see Section 42.5.5: *Error recovery on page 403*),
- provides the number and the location of corrupted macroblocks in the picture (see Section 42.5.6: *Error statistics on page 406*).

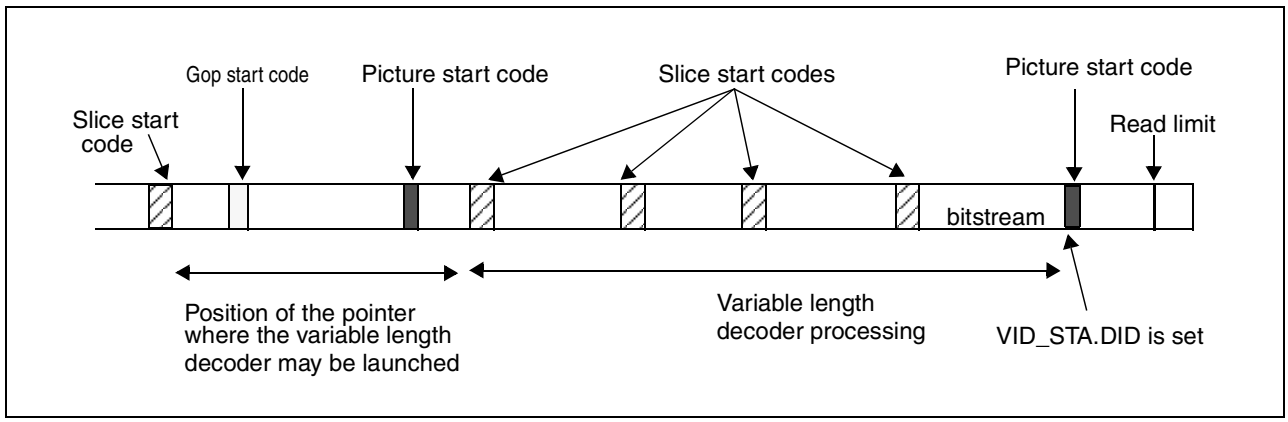
The variable length decoder is launched by the VID_EXE register for each picture to be decoded:

When the variable length decoder is launched with a pointer defined in bytes, processing starts at the exact byte location, but the data sent to the variable length decoder in its FIFO is aligned on a 128 bytes burst, that is the eight LSBs of the pointer are ignored for memory access.

The variable length decoder may be launched with a pointer not aligned with a picture start code. The internal state machine of the variable length decoder, when a decoding task is launched, is to search only for slice start codes. When the first slice start code is detected, processing starts (Figure 110).

During the decoding task the variable length decoder FIFO requests data when there is space for one burst, that is when the FIFO is up to half full.

Figure 110: Variable length decoder processing possible pointer position for good behavior



The variable length decoder enters in an idle state at the end of the decoding process when it detects a start code different from a slice start code. If there is no start code at the end of the picture in memory, the start code may be inserted by the application software at the very end of the picture. If no start code has been found, then the variable length decoder automatically stops on the read limit pointer (VID_VLDRL). Analyzing the bitstream after the end of a picture (when no start code is present) is not recommended because the variable length decoder may find other slice start codes and generate an overflow error (DOE interrupt).

42.4.3 MPEG2 pipeline

The variable length decoder sends run length coefficients to the MPEG2 pipeline. The pipeline processes the six blocks of every macroblock in the following order: Y0, Y1, Y2, Y3, Cb and Cr (refer to MPEG2 2 standard, ISO 13818-2, Macroblock structure). The pipeline performs run length decoding, inverse zig-zag, inverse quantization and the inverse DCT. It holds a set of two quantization tables (intra and nonintra).

42.4.4 Motion compensation unit

In parallel, the motion compensation unit computes the predictors. Predictors are fetched from the appropriate reference frame buffer and processed according to the macroblock type and the motion vectors sent by the variable length decoder. When the macroblock is an intra macroblock, no predictor is fetched. For nonintra macroblocks, four bursts of data are always read per direction (forward or backward), two for luminance data and two for chrominance data. A luma burst is always 15 words of 128 bits, and all words belong to the same field. A chroma burst is always nine words of 128 bits and all words belong to the same field. The motion compensation unit does not compute the predictor addresses in the reference picture. This task is performed by the bus access controller.

42.4.5 Pipe end

Finally, macroblocks are reconstructed by the pipe end, by adding coefficients from the pipe and the predictors from the motion compensation unit. Then macroblocks are sent to the reconstructed frame buffer through the bus access controller and the interconnect.

42.4.6 Bus access controller

The bus access controller performs read and write accesses to the frame buffers and read accesses to the bit buffers. It receives requests from the blocks, computes the addresses corresponding to the requests, arbitrates the requests and finally performs the memory access on the STBus. For prediction accesses, the bus access controller receives motion vectors and the macroblock type from motion compensation unit.

The STBus interface is used to read the bit buffer for the variable length decoder, for the prediction and the reconstruction processes. The packet size is programmable through register CFG_VIDIC. If R_OPC flags an error during a transaction, the CPU is interrupted with bit VID_STA.ROPC.

42.5 Video decoding

42.5.1 Control of decoding

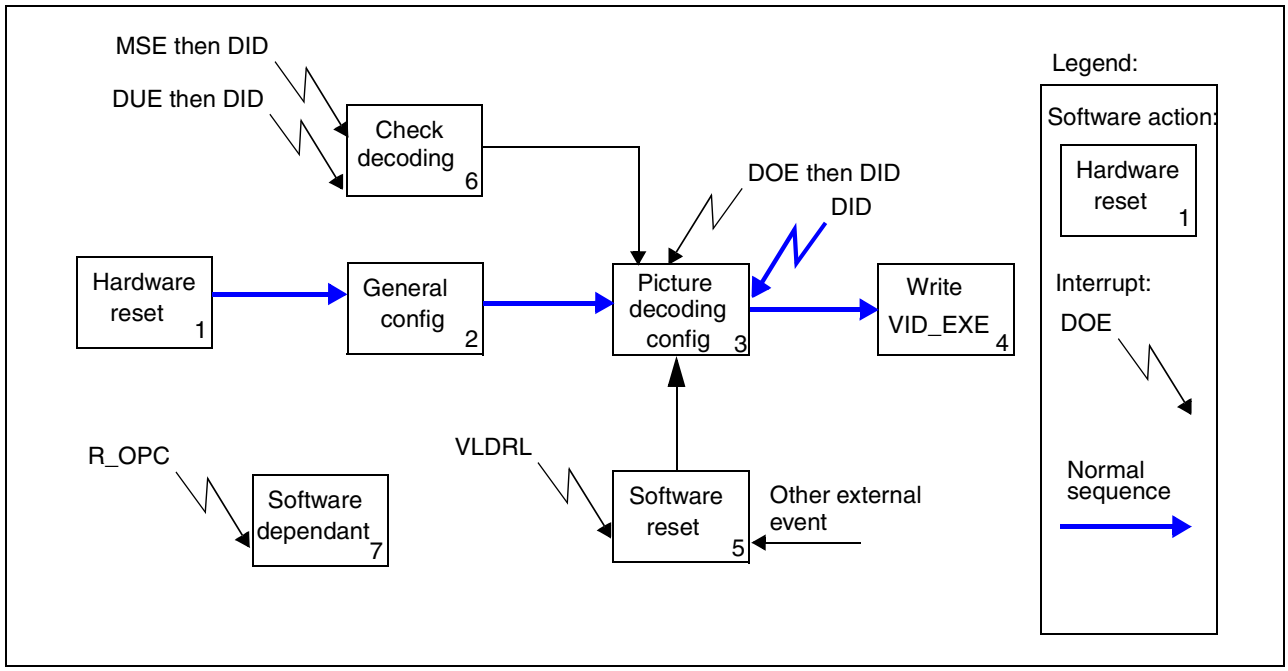
The control is performed by the CPU. The CPU launches the decoder one picture at a time.

Decoding performed without error

The normal sequence is described below.

1. hardware reset,
2. general configuration of the MPEG2 video decoder (see *General configuration* *on page 400*),
3. picture decoding configuration (see *Picture decoding configuration* *on page 400*),
4. launch decoding by writing in VID_EXE LSBYTE.
5. loop to action 3 on a DID (decoder idle) interrupt.
   See description of VID_STA in Chapter 43: *MPEG2 video decoder registers* *on page 408* for more details on interrupts).

Figure 111: Software and hardware interactions



Abnormal cases

This covers decoding tasks that end by an interrupt different from DID or that is interrupted by a software reset. See below all the different cases and the possible software action.

- **DOE interrupt**
  For overflow errors, the decoder can be programmed for the next decoding task, no reset or specific action is needed. The software may decide whether to display the decoded picture.
- **DUE interrupt**
  For underflow errors, the decoder can be programmed for the next decoding task, no reset or specific action is needed. The software may read debug registers (6) to decide to display or not the decoded picture.
- **MSE interrupt**
  For syntax or semantic errors the decoder can be programmed for the next decoding task, no reset or specific action is needed. The software may read the error statistic registers (VID_MBE*x*) registers (6) to decide whether to display the decoded picture.
- **VLDRL interrupt**
  The decoder has reached the read limit without having completed the decoding task and is in an unknown state. A soft reset (5) must be performed before executing the next decoding task.
- **R_OPC interrupt**
  Software decides what to do (7).
- **Other external event**
  Software can decide to interrupt the decoding task at any time for several reasons: for example a channel switch or the decoding task is too long (overtime or overrun). The current decoding task can be interrupted at any time by a software reset (5).

Note: It is not recommended to write in VID_EXE during a decoding task, that is, when VID_STA.DID = 0, because the current and the next decoding tasks may fail. To execute a decoding task correctly while another one is still active, STMicroelectronics recommends stopping the current one by writing to VID_SRS.

42.5.2 General configuration

The general configuration consists of CFG_VIDIC (video decoder interconnect configuration) and VID_ITM (interrupt mask).

42.5.3 Picture decoding configuration

Quantization table loading

The two quantization matrices (intra and nonintra) used by the inverse quantizer must be initialized. Since there are no built-in quantization matrices, they must be loaded either with default matrices or with those extracted from the bitstream by the CPU.

Quantization tables do not need to be loaded at every decoding task. If they are only present in the bitstream at the beginning of a sequence, they may be loaded only for the first picture.

The quantization tables are written in registers VID_QMWIP and VID_QMWNIP.

Decoded picture size

VID_DFW gives the picture width in number of macroblocks and VID_DFH gives the number of rows of macroblocks in the frame picture. VID_DFS holds the whole number of macroblocks in the frame picture.

This data is extracted from the sequence header and is relative to the frame picture format.

Picture pointers for decoding

Before the decoding of each picture, the following frame buffer pointers must be set up:

- VID_RFP, VID_RCHP: reconstructed frame pointers for luminance and chrominance for main reconstruction,
- VID_FFP, VID_FCHP: forward prediction frame pointers for luminance and chrominance,
- VID_BFP, VID_BCHP: backward prediction frame pointers for luminance and chrominance.

VID_FFP, VID_FCHP, VID_BFP and VID_BCHP define the areas in memory for the predictors. How these four pointers are used depends on the prediction mode. The rules are given below.

Note: Pictures are always stored as frames, and to access a field (top or bottom), the starting address of the frame must be defined.

- **P-frame picture (frame, field or dual-prime prediction)**
  VID_FFP and VID_FCHP are set to the address of the predictor frame. VID_BFP and VID_BCHP are not used.
- **B-frame picture (frame or field prediction)**
  VID_FFP and VID_FCHP are set to the address of the forward predictor frame. VID_BFP and VID_BCHP are set to the address of the backward predictor frame.
- **P-field picture (field, 16 x 8 or dual-prime prediction)**
  When decoding either field, VID_FFP and VID_FCHP are set to the address of the previous decoded I or P frame. VID_BFP and VID_BCHP are not used.
- **B-field-picture (field or 16 x 8 prediction)**
  VID_FFP and VID_FCHP are set to the address of the frame in which the two forward predictor fields lie. VID_BFP and VID_BCHP are set to the address of the frame in which the two backward predictor fields lie.

For I-picture decoding, no predictors are necessary, but VID_FFP and VID_FCHP must be set to the address of the last decoded I or P-picture for use by the automatic error concealment function.

For the P-field picture, an on-chip mechanism selects the pointers which must be used for the prediction between the forward frame and the reconstructed frame pointers. If the decoded field is the first field, the prediction is done in the forward reference frame. If it is the second field, the

prediction is done using both forward and reconstructed frame pointers. The field number (first or second) is computed by the MPEG2 video decoder. It may be overwritten by the application with FFN (in VID_PPR). This may be useful for trick mode applications.

Picture parameters

These parameters are extracted from the bitstream. They have to be programmed in VID_PPR.

Decoding task instruction

Decoding task instructions are programmed in VID_TIS. This register gives options for error recovery and decoding (simplified B or overwrite mode).

Stream pointers for variable length decoder read access

The starting position of the encoded picture in the external memory must be programmed in the variable length decoder read pointer register (VID_VLDPTR).

A variable length decoder read limit may be set in register VID_VLDRL. When the read pointer reaches VID_VLDRL (variable length decoder read limit), reading in the buffer is stopped and the VLDRL is set in VID_STA.

A bit buffer area can be defined with VID_BBS (bit buffer start) and VID_BBE (bit buffer end).

- If these registers are left in their reset state (0x0000 0000) or if the same value is written in both of them, they have no effect.
- If different values are programmed in both registers when the internal variable length decoder read pointer reaches VID_BBE it jumps to VID_BBS. This allows a video ES to be wrapped in a predefined memory area.

Note: 1 The variable length decoder read pointer (VID_VLDPTR) has to be programmed before each decoding task, even if the variable length decoder has stopped on the picture start code of the next picture to be decoded. This is because the picture start code could have been stored in the variable length decoder input FIFO at the end of the previous picture decode and this FIFO is flushed before starting the next decoding task.

2 The read limit to be programmed must be enlarged by the size of one burst, that is 128 bytes. This is due to internal pipeline processing in the variable length decoder. When the variable length decoder FIFO is empty, there is still about 64 bytes in the variable length decoder pipeline that are not processed.

42.5.4 Main and secondary reconstruction

The decoder has two ways of reconstructing the picture in external memory. The normal way is through the main reconstruction. The reference frames used for MPEG2 prediction are always stored through this path. When required to reduce picture size before memory storage, in preparation for a reduced display size, a secondary reconstruction path can be used.

The use of hardware secondary reconstruction depends on the application.

It is useful when large zoom out factors are needed; part of the zoom-out may be done using secondary reconstruction, the rest being done in the display.

In any case, reference pictures (I and P) must be stored in full size (main reconstruction). If secondary reconstruction is needed for I and P pictures then they have to be stored in both formats (full size and decimated).

Main reconstruction is enabled by register VID_RCM.ENM, secondary reconstruction by VID_RCM.ENS. B pictures may be only reconstructed with the secondary reconstruction. I and P frames, since they are used as a reference, must be reconstructed using both paths.

The secondary reconstruction is processed according to the value of VID_RCM, which gives the horizontal and vertical ratio (1,1/2 and 1/4 for both direction) and the progressive sequence flag. Horizontal decimation is performed as described in Figure 113 and vertical decimation is

performed as described in Figure 112. When decimation is needed in both direction, the vertical decimation is performed after the horizontal decimation.

If, due to decimation, the resulting picture is not a multiple of 16 pixels horizontally or vertically (size of luma stored macroblock), the actual stored picture is rounded up to the closest multiple of 16 pixels in the appropriate direction(s). Right side and/or bottom side of the stored picture is the garbage and has to be removed within the display processor prior to display.

When the decoded picture belongs to a progressive sequence, vertical decimation is done in the frame. Else, vertical decimation is done in the field.

Reconstructed frame pointers for luminance and chrominance for secondary reconstruction are VID_SRFP and VID_SRCHP.

At any time, the CPU may read the position of the next macroblock to be decoded in the reconstructed frame buffer in register VID_MBNM for the main reconstruction and VID_MBNS for the secondary reconstruction.

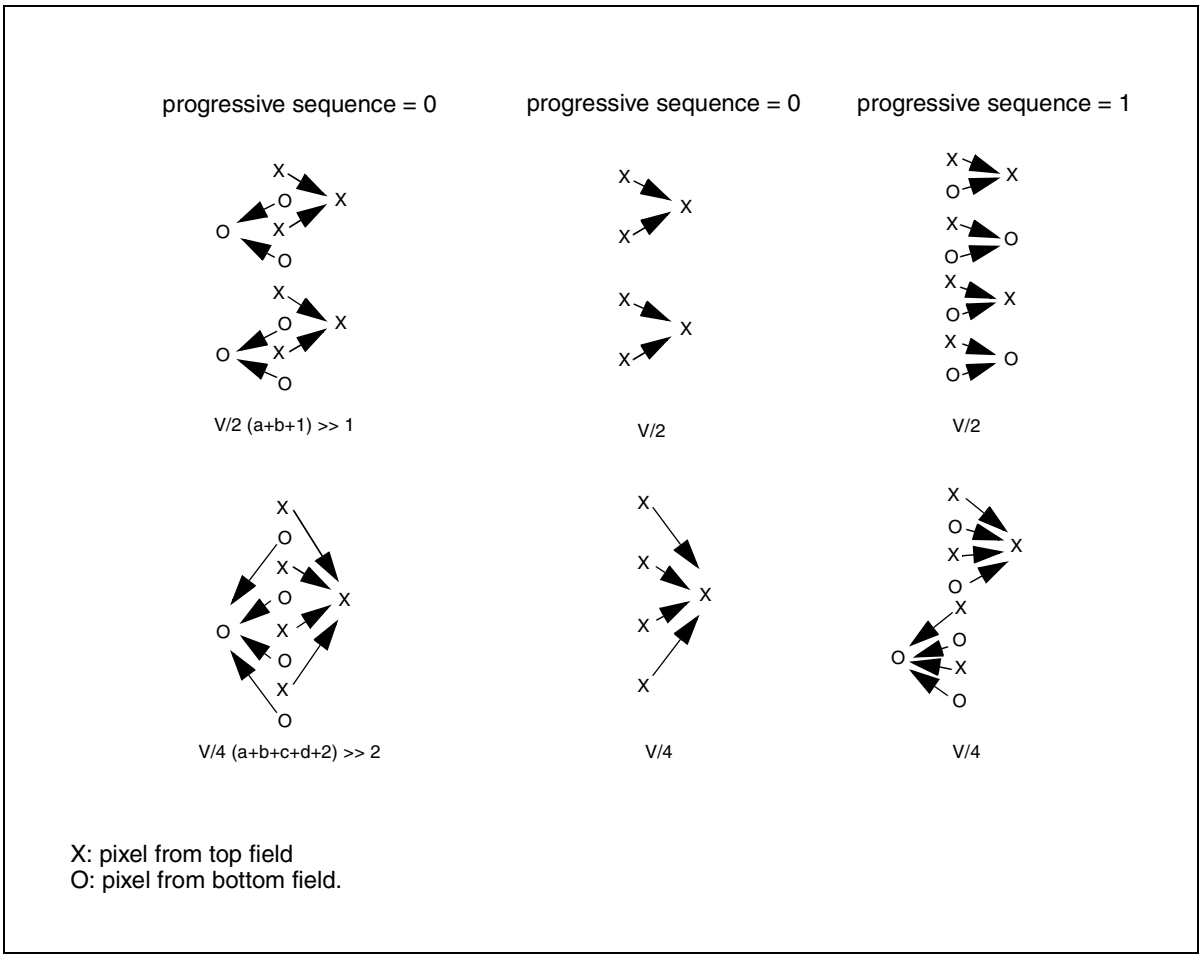


Figure 112: Vertical decimation in field picture secondary reconstruction

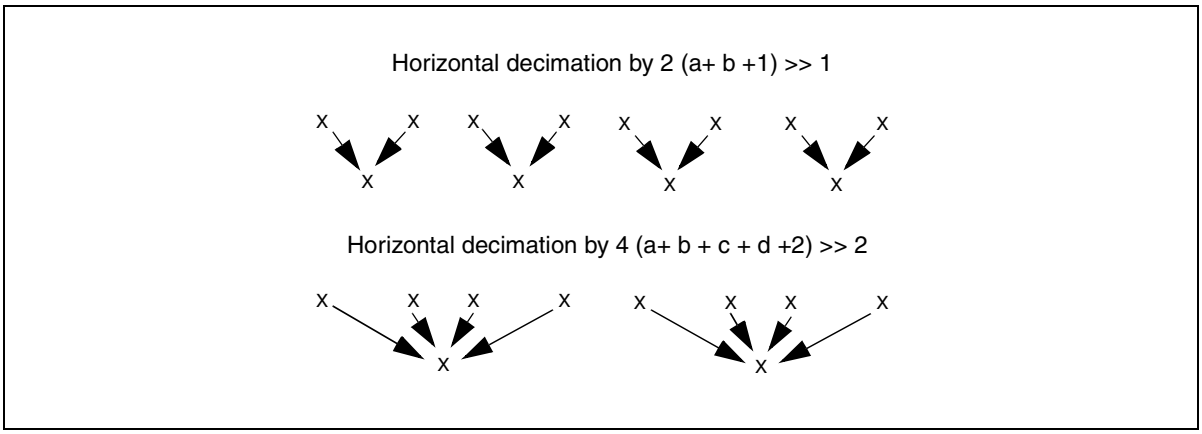


Figure 113: Horizontal decimation in secondary reconstruction

42.5.5 Error recovery

There are three levels of error detection in the video decoder:

- bitstream syntax/semantic error detection with automatic missing macroblocks concealment,
- pipeline underflow error detection,
- pipeline overflow error detection.

Syntax and semantic error detections

The variable length decoder detects the syntax and semantic errors listed below. The way to conceal errors is different depending on the detected errors. Table 123 describes errors that are detected by the MPEG2 video decoder. Error concealment is described below the table.

Table 123: Error detection

Layer	Description	Concealment	MBE[a]
Syntax error: A variable length code which does not exist in the tables or a fixed length code with a forbidden value			
Slice	**quantizer_scale_code** = 0	Previous **quantizer_scale_code** is used (macroblock or slice). After a reset (HW, SRS), value is set to 1.	*N*
Macroblock	**macroblock_address_increment**	See *Procedure one* on page 405	Yes
	macroblock_type		
	frame_motion_type and **field_motion_type**: the value 00 is reserved		
	quantiser_scale_code	Previous **quantizer_scale_code** (slice or macroblock) is used. After a reset (HW, SRS), value is set to 1.	No
	marker_bit	See *Procedure one* on page 405	Yes
	coded_block_pattern_420		
	end_of_macroblock for D pictures: it shall be equal to 1.		
	motion_code[r][s][t]		
Block	**first_dct_coef**		
	next_dct_coef		
Semantic error: The way of encoding slices and macroblocks does not follow MPEG2 rules.			
Slice	Slice start code combined with macroblock address increment gives a macroblock number already decoded.	See *Procedure one* on page 405	Yes
	Slice start code combined with macroblock address increment gives a macroblock number higher than the macroblock number to be decoded.		
Macroblock	mb_addr_increment may result in a macroblock address greater than the picture size. The macroblock is outside the picture.	See *Procedure two* on page 406	Yes
	The processing of the different parameters in **motion_vectors**(s) may result in a motion vector outside the reference picture.	If bit MVC in VID_TIS is one, the vector is clipped to point in the picture area. Else, a prediction out of the reference picture is done.	No
	QFS[0] may not lie in the range 0 to ((2^(8+intra_dc_precision))-1).	If negative value, the coefficient is set to 0 but the predictor remains negative	No
Block	Too many coefficients may be found in a block.	See *Procedure one* on page 405	Yes

a. MBE: when yes, the error is taken into account in VID_MBE*n* registers. When no, the error is not taken into account.

Procedure one

If the variable length decoder detects a syntax or semantic error in the bitstream, the pipeline copies macroblocks from the previous picture. It uses the motion vectors reconstructed for the previous row of macroblocks in the current picture, and scans the bitstream until a slice start code is detected. At this point, normal decoding resumes.

If the error occurred in the last slice in the picture, concealment continues until the end of the picture. The pipeline then stops normally, assuming that the following picture start code is intact. Macroblocks are concealed using the vectors of the macroblock immediately above the lost macroblock.

Concealment macroblocks are accessed using the forward and backward reference frames. Lost macroblocks in the first row are copied directly from the previous pictures, that is as P-macroblocks with zero motion vectors. If an intra macroblock is coded with concealment motion vectors, the concealment motion vectors are used. If not, concealment is a simple copy from the previous picture using zero vectors.

Table 124 shows the rules used to fetch concealment macroblocks. Skipped macroblocks are not mentioned since they always refer to one of the prediction types listed below.

Table 124: Macroblock type recovery in case of error concealment.

Picture structure/ picture type	Prediction type of the macroblock aligned vertically in the row above	Prediction type of the concealed macroblock 1 vector = (H,V)
I,P,B frame picture	Intra no concealment	Forward frame (vector = 0)
	Intra with concealment	Forward frame (with sent vector)
I, P, B field picture	Intra no concealment	Forward field (vector = 0)
	Intra with concealment	Forward field (with sent vectors)
P frame picture	Forward frame	Forward frame (with same vectors)
	Forward field	Forward field (with same vectors)
	Dual	Forward field (with vectors of same field parity, that is vectors sent)
P field picture	Forward field	Forward field (with same vectors)
	16x8	Forward field (with same top vectors)
	Dual	Bidirectional field (with same vectors)
B frame picture	Forward frame	Forward frame (with same vectors)
	Forward field	Forward field (with same vectors)
	Back frame	Back frame (with same vectors)
	Back field	Back field (with same vectors)
	Bidirectional frame	Bidirectional frame (with same vector)
	Bidirectional field	Forward field (with same forward vectors)
B field picture	Forward field	Forward field (with same vectors)
	Forward 16 x 8	Forward field (with same top vectors)
	Back field	Back field (with same vectors)
	Back 16 x 8	Back field (with same top vectors)
	Bidirectional field	Bidirectional field (with same vectors)
	Bidirectional 16 x 8	Forward field (with same top vectors)

Procedure two

When the decoder detects that the input stream is not compliant with the restricted slice structure, the decoder reconstructs missing macroblocks by copying macroblocks from the forward reference picture with a null motion vector using frame prediction in a frame picture and field prediction in a field picture.

The decoder detects a nonrestricted slice structure when the slice start code combined with the first macroblock address increment gives an absolute address which is higher (+2 at least) than the previous decoded macroblock number. The decision (restricted or nonrestricted slice structure) is taken on the first macroblock following the slice start code.

An underflow occurs when the last slices are missing and a different start code to a slice start code or an error start code is detected. Missing macroblocks are reconstructed if RMM in VID_TIS = 1, and are not reconstructed if RMM = 0. For underflow errors, the underflow flag in the status register is always set.

Overflow and underflow errors

An overflow error occurs when the variable length decoder detects more macroblocks in the bitstream than the number corresponding to the picture size, VID_DFS. The variable length decoder stops processing macroblocks and resumes a start code search on the next picture start code, then, it returns in idle state. The overflow condition is flagged by the bit VID_STA.DOE.

An underflow error occurs when the variable length decoder has found in the bitstream fewer macroblocks than the number defined by the decoded picture size, VID_DFS, and when it stops on a picture start code. Then, it returns in idle state. The underflow condition is flagged by the bit DUE in VID_STA.

The number of macroblocks reconstructed in memory is equal to VID_DFS if RMM in VID_TIS = 1 (missing macroblocks are concealed).

42.5.6 Error statistics

The number and the location of the errors in the pictures determine whether the picture is to be displayed. The decoder gives statistics in the VID_MBE*n* (macroblock errors) registers. The VID_MBE*n* registers give the locations of both semantic and syntax errors in the picture. These registers are stable at the end of the decoding task until the start of the next picture decoding task whatever the video channel. At the end of picture decoding, if at least one error occurred during picture reconstruction, bit VID_STA.MSE is set. If no error occurred, this bit is 0.

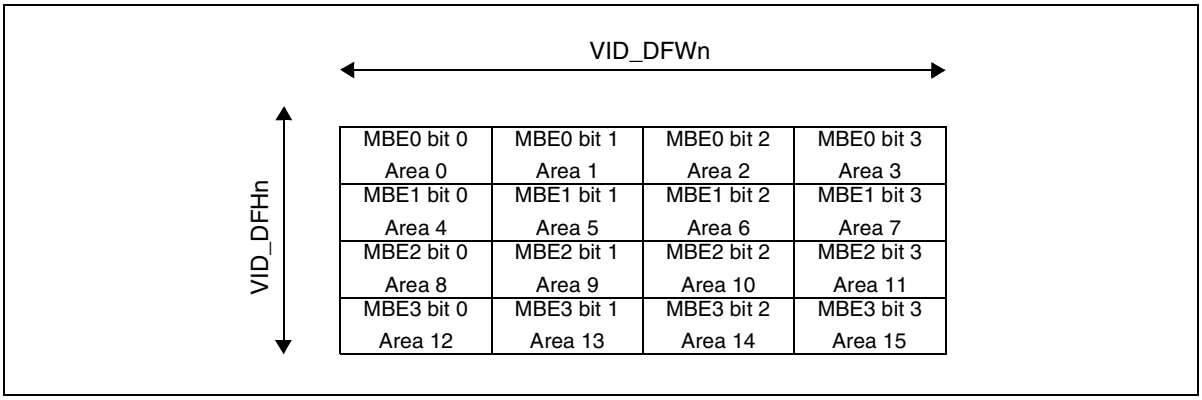
The decoded picture is divided into 16 areas, defined by four horizontal and four vertical slides. The areas are numbered from 0 (top left) to 15 (bottom right). This number is incremented from left to right, top to bottom (see Figure 114). For each area, VID_MBE*n* gives the number of macroblocks reconstructed with error concealment.

If the picture size in macroblock units is a multiple of four in both directions, each of the 16 areas contains the same number of macroblocks DFS >> 4. If it is not a multiple of four macroblocks, the number of macroblocks in each area is:

```
For i and j in [0..2]:
  area(i*4+j)= (DFH >> 2) * (DFW >> 2) MBs
For i=3 and j in [0..2]:
  area(12+j) = (DFH >> 2 + DFH mod 4) * (DFW >> 2) MBs
For i in [0..2] and j=3:
  area(i*4+3) = (DFH >> 2) * (DFW >> 2 + DFW mod 4) MBs
For i = 3 and j=3:
  area(15) = (DFH >> 2 + DFH mod 4) * (DFW >> 2 + DFW mod 4) MBs
```

Note: In the field picture structure, DFH must be divided by two.

Figure 114: Macroblock error statistics



42.5.7 Simplified B decoding

Simplified B decoding allows bandwidth consumption to be reduced up to 40% while decoding a B-picture. This is done by converting all bidirectional predictors of B-pictures in forward predictors.

The maximum bandwidth required to decode a B-picture is usually 4216 Mbit/s, but for simplified B, this is equivalent to a P-picture at 2712 Mbits/s.

This feature has an impact on picture quality, but has no effect on the P dual prime picture.

Secondary decoding requires an additional 50% of main picture bandwidth - 2108 Mbit/s (B) or 1356 Mbit/s (SB/P).

42.6 Resets

42.6.1 Hardware reset

After a hardware reset, the MPEG2 video decoder is in an idle state. No processing is done and no request is sent until video decoding is launched.

42.6.2 Software reset

Two software resets are implemented.

- VID_SRS: the reset is active when the CPU writes the least significant byte of the VID_SRS register. Software reset is a synchronous active high reset. It resets:
  - variable length decoder flags: DID, MSE, DOE and DUE in VID_STA,
  - variable length decoder input FIFO controller,
  - error statistic registers: VID_MBE0, VID_MBE1, VID_MBE2 and VID_MBE3.

  After the reset no processing is performed until video decoding is started. The MPEG2 video decoder is in an idle state.

  Software reset has no impact on register contents.
- VID_EXE: has the same effect as software reset (VID_SRS), but it also starts the variable length decoder.

43 MPEG2 video decoder registers

Register addresses are provided as *MPEGBaseAddress* + offset.

The *MPEGBaseAddress* is:

0x1924 0000.

All registers are reset by hardware. Read only registers can also be affected by SRS and EXE; see Section 42.6.2: *Software reset on page 407*. When registers are read, the value of the reserved bits are 0 unless specified. No registers are buffered: write action effect is immediate.

Table 125: MPEG2 video decoder register summary

Register	Description	Offset	Type
VID_EXE	Execute decoding task	0x0008	WO
Reserved	-	0x0010	-
VID_MBEn	Macroblock error statistic (n = 0 to 3)	0x0070 - 0x007C	RO
VID_QMWIp	Quantization matrix data, intra-table	0x0100 - 0x013F	R/W
VID_QMWNIp	Quantization matrix data, non–intra-table	0x0180 - 0x01BF	
VID_TIS	Task instruction	0x0300	
VID_PPR	Picture parameters	0x0304	
VID_SRS	Decoding soft reset	0x030C	WO
VID_ITM	Interrupt mask	0x03F0	R/W
VID_ITS	Interrupt status	0x03F4	RO
VID_STA	Status	0x03F8	
VID_DFH	Decoded frame height	0x0400	R/W
VID_DFS	Decoded frame size	0x0404	
VID_DFW	Decode frame width	0x0408	
VID_BBS	Video elementary stream bit buffer start	0x040C	
VID_BBE	Video elementary stream bit buffer end	0x0414	
VID_VLDRL	Variable length decoder read limit	0x0448	
VID_VLD_PTR	Variable length decoder read pointer	0x045C	
Reserved	-	0x0480 - 0x0484	-

Table 125: MPEG2 video decoder register summary

Register	Description	Offset	Type
VID_BCHP	Backward chroma frame buffer	0x0488	R/W
VID_BFP	Backward luma frame pointer	0x048C	
VID_FCHP	Forward chroma frame buffer	0x0490	
VID_FFP	Forward luma frame pointer	0x0494	
VID_RCHP	Main reconstructed chroma frame pointer	0x0498	
VID_RFP	Main reconstructed luma frame pointer	0x049C	
VID_SRCHP	Secondary reconstructed chroma frame buffer	0x04A0	
VID_SRFP	Secondary reconstructed luma frame buffer	0x04A4	
VID_RCM	Reconstruction mode	0x04AC	

VID_EXE Execute decoding task

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VID_EXE																															

Address: *MPEGBaseAddress* + 0x0008
Type: WO
Reset: Undefined
Description: Writing to the least significant byte of this register starts a decoding task.

VID_MBEn Macroblock error statistic (n = 0 to 3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBEx3[7:0]								MBEx2[7:0]								MBEx1[7:0]								MBEx0[7:0]							

Address: *MPEGBaseAddress* + 0x0070 (VID_MBE0), 0x0074 (VID_MBE1), 0x0078 (VID_MBE2), 0x007C (VID_MBE3)
Type: RO
Reset: 0
Description: Decoded pictures are divided into 16 areas. For each area, MBE gives the number of macroblock errors (semantic and syntax) detected by the variable length decoder. These data are stable at the end of the picture decoding task until the start of the next picture decoding task (whether the video channel is 1 or 2).

VID_QMWIp Quantization matrix data, intra-table

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
IQCOEFF(4p+3)[7:0]	IQCOEFF(4p+2)[7:0]	IQCOEFF(4p+1)[7:0]	IQCOEFF(4p)[7:0]

Address: *MPEGBaseAddress* + 0x0100 to 0x013F

Type: R/W

Reset: 0

Description: The quantization coefficients for an intra table must be written to these addresses in increasing order in the order in which they appear in the bitstream (in zig-zag order). Thus, the first coefficient which appears in the bitstream is located at the lower register address and the last coefficient in the bitstream is located at the higher register address. The order must be strictly respected. Each 32-bit word contains four 8-bit coefficients.

VID_QMWNIp Quantization matrix data, non–intra-table

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
NIQCOEFF(4p+3)[7:0]	NIQCOEFF(4p+2)[7:0]	NIQCOEFF(4p+1)[7:0]	NIQCOEFF(4p)[7:0]

Address: *MPEGBaseAddress* +0x0180 to 0x01BF

Type: R/W

Reset: 0

Description: The quantization coefficients for a non-intra table must be written to these addresses in increasing order in the order in which they appear in the bitstream (in zig-zag order). Thus, the first coefficient which appears in the bitstream is located at the lower register address and the last coefficient in the bitstream is located at the higher register address. The order must be strictly respected. Each 32-bit word contains four 8-bit coefficients.

VID_TIS Task instruction

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4	3	2	1	0
Reserved	RMM	MVC	SBD	OVW

Address: *MPEGBaseAddress* +0x0300

Type: R/W

Reset: 0

This register contains the decoding task instruction.

[31:4] **Reserved**

[3] **RMM**: reconstruct missing macroblock.
1: In case of underflow, the missing macroblocks are reconstructed.

[2] **MVC:** motion vector check
1: Ensures that motion vectors used for prediction remain inside the picture (see Table 123: *Error detection* *on page 404*).

[1] **SBD:** simplified B picture decoding
1: B picture decoding is simplified to save bandwidth: all bidirectional macro blocks are processed as forward.

[0] **OVW**
1: Enable overwrite mechanism during decoding

VID_PPR Picture parameters

31	30	29 28	27	26	25	24	23	22	21 20	19 18	17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Reserved	MP2	FFN	TFF	FRM	CMV	QST	IVF	AZZ	PCT	DCP	PST	BFV	FFV	BFH	FFH

Address: *MPEGBaseAddress* + 0x0304

Type: R/W

Reset: 0

Description: This register contains parameters of the picture to be decoded. These parameters are extracted from the bitstream. In MPEG-1 mode (when MP2 in VID_PPR is reset), only PCT, BFH and FFH have to be set. Other bits must be reset.

[31] **Reserved**

[30] **MP2**: MPEG-2 mode
0: An MPEG-1 bitstream is expected
1: The decoder expects an MPEG-2 video bitstream

[29:28] **FFN**: Force field number
This is used for prediction to select the reference pictures.
01 or 10: The field is forced to first field picture 11: The field is forced to second field picture
00: Internal field number is used
The field number use for prediction is computed internally by the MPEG2 video decoder. Meanwhile, if an error occurs in the stream and a field is lost, this bit must be set at the proper value by the application. It may be used also in still picture decoding.

[27] **TFF**: Top field first
Set equal to the TOP_FIELD_FIRST bit of the MPEG-2 picture coding extension.

[26] **FRM**
Set equal to the FRAME_PRED_FRAME_DCT bit of the picture coding extension.

[25] **CMV**: Set equal to the CONCEALMENT_MOTION_VECTORS bit of the MPEG-2 picture coding extension. Indicates that motion vectors are coded for intra macroblocks.

[24] **QST**: Set equal to the Q_SCALE_TYPE bit of the picture coding extension.

[23] **IVF**: Set equal to the INTRA_VLC_FORMAT bit of the picture coding extension.

[22] **AZZ**
Set equal to the ALTERNATE_SCAN bit of the picture coding extension.

[21:20] **PCT[1:0]**: Set to equal to the two least significant bits of PICTURE_CODING_TYPE in the picture header.

[19:18] **DCP[1:0]**: Set equal to INTRA_DC_PRECISION of the picture coding extension. The value 11, defining a precision of 11 bits, is not allowed.

[17:16] **PST[1:0]**: Set equal to the PICTURE_STRUCTURE bits of the MPEG-2 picture coding extension.
Note: code 00 also indicates frame structure, even though this value is illegal in the MPEG-2 variable.
00: Frame picture 01: Top field
10: Bottom field 11: Frame picture

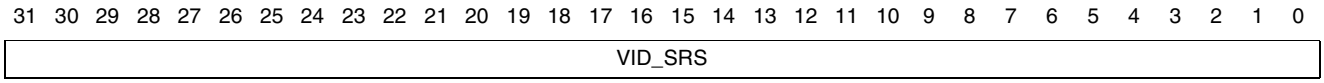
[15:12] **BFV[3:0]**: Set equal to BACKWARD_VERTICAL_F_CODE of the picture coding extension.

[11:8] **FFV[3:0]**: Set equal to FOWARD_VERTICAL_F_CODE of the picture coding extension.

[7:4] **BFH[3:0]**
In MPEG-1 mode BFH[3] is set equal to FULL_PEL_BACKWARD_VECTOR of the picture header, and BFH[2:0] is set equal to BACKWARD_F_CODE of the picture header. In MPEG-2 mode BFH[3:0] is set equal to BACKWARD_HORIZONTAL_F_CODE of the picture coding extension.

[3:0] **FFH[3:0]**
In MPEG-1 mode FFH[3] is set equal to FULL_PEL_FORWARD_VECTOR of the picture header, and FFH[2:0] is set equal to FORWARD_F_CODE of the picture header. In MPEG-2 mode FFH[3:0] is set equal to FORWARD_HORIZONTAL_F_CODE of the picture coding extension.

VID_SRS Decoding soft reset



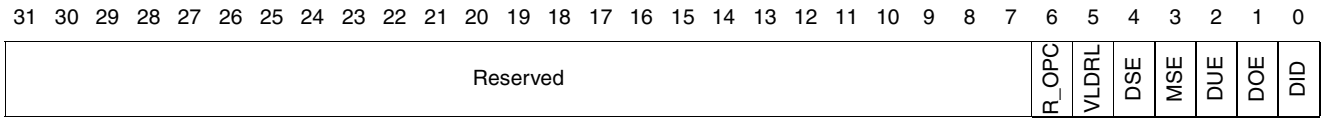
Address: *MPEGBaseAddress* + 0x030C

Type: WO

Reset: Undefined

Description: Writing to the least significant byte of this register starts the software reset sequence. The reset is just activated once on writing.

VID_ITM Interrupt mask



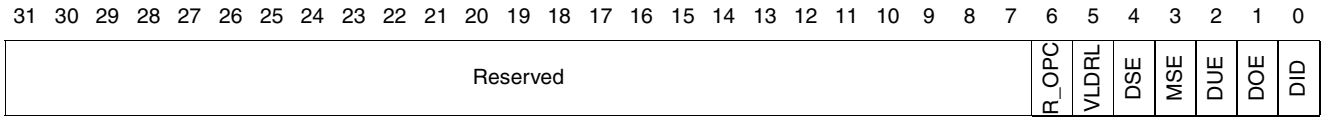
Address: *MPEGBaseAddress* + 0x03F0

Type: R/W

Reset: 0

Description: Any bit set in this register enables the corresponding interrupt on the VID_IRQ line. An interrupt is generated when a bit in the VID_STA register changes from 0 to 1 and the corresponding mask bit is set.

VID_ITS Interrupt status



Address: *MPEGBaseAddress* + 0x03F4

Type: RO

Reset: 0

Description: When a bit in the VID_STA register changes from 0 to 1, the corresponding bit in VID_ITS is set (independently of ITM). When there is at least one bit set to one in VID_ITS which is not masked, the interrupt line is set to 1. The reading of VID_ITS clears all bits in that register and de-asserts the interrupt line.

VID_STA Status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									R_OPC	VLDRL	DSE	MSE	DUE	DOE	DID

Address: *MPEGBaseAddress* +0x03F8

Type: RO

Reset: 0x0000 0021

Description: This register contains a set of bits which represent the status of the decoder at any instant. Any change from 0 to 1 of any of these bits sets the corresponding bit of the VID_ITS register, and can thus potentially cause an interrupt on the GL1_IRQ line. Some bits are pulses and are unlikely ever to be read as 0.

[31:7] **Reserved**

[6] **R_OPC:** set when an R_OPC is 1 for one of the two interconnect plugs. This signal is a one cycle pulse.

[5] **VLDRL:** variable length decoder read limit. This flag is set to 1 when the variable length decoder read pointer reaches VID_VLDRL pointer. The bit is reset when the condition is not true.

[4] **DSE:** a one pulse bit, set to 1 when decoding semantic or syntax error is detected by the variable length decoder.

[3] **MSE:** set to 1 at the end of a decoding task if a semantic or a syntax error has been detected by the variable length decoder during the decoding process of the picture. It is reset on a write action on VID_EXE or VID_SRS LS byte.

[2] **DUE:** set to 1 at the end of a decoding task if a underflow error has been detected by the variable length decoder during the decoding process of the picture. It is reset on a write action on VID_EXE or VID_SRS LS byte.

[1] **DOE:** set to 1 at the end of a decoding task if an overflow error has been detected by the variable length decoder during the decoding process of the picture. It is reset on a write action on VID_EXE or VID_SRS LS byte.

[0] **DID:** decoder idle. Set to 1 when pipeline is idle and the variable length decoder is in idle state or on a write action on VID_SRS LS byte. The condition is reset on a write action on VID_EXE.

VID_DFH Decoded frame height

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									DFH[6:0]						

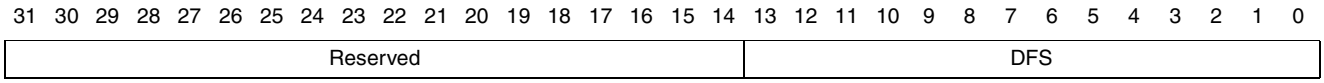
Address: *MPEGBaseAddress* + 0x0400

Type: R/W

Reset: 0

Description: Decoded frame height (in rows of macroblocks). This register is used for error concealment and macroblocks error statistics. This is derived from the vertical size value transmitted in the sequence header. It is divided internally by two for field picture decoding.

VID_DFS Decoded frame size



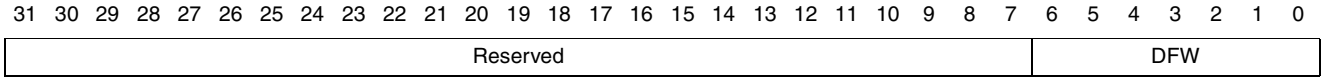
Address: *MPEGBaseAddress* + 0x0404

Type: R/W

Reset: 0

Description: This register is set up with a value equal to the number of macroblocks in the decoded picture. This is derived from the horizontal and vertical size values transmitted in the sequence header. It is divided internally by two for field picture decoding.

VID_DFW Decode frame width



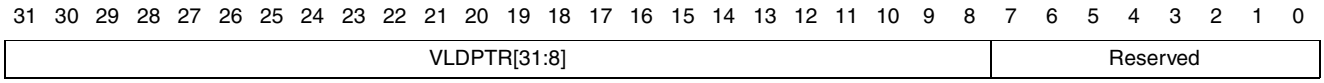
Address: *MPEGBaseAddress* + 0x0408

Type: R/W

Reset: 0

Description: This register is set up with a value equal to the width in macroblocks of the decoded picture. This is derived from the horizontal size value transmitted in the sequence header.

VID_BBS Video elementary stream bit buffer start



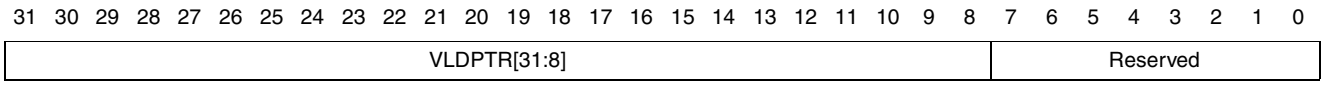
Address: *MPEGBaseAddress* + 0x040C

Type: R/W

Reset: 0

Description: Memory address of the video elementary stream bit buffer start, defined in 256-byte units.

VID_BBE Video elementary stream bit buffer end



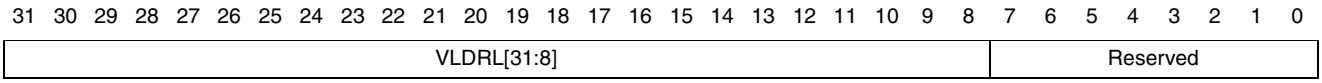
Address: *MPEGBaseAddress* + 0x0414

Type: R/W

Reset: 0

Memory address of the video elementary stream bit buffer end, defined in 256-byte units.

VID_VLDRL Variable length decoder read limit



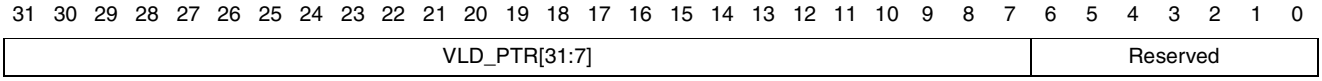
Address: *MPEGBaseAddress* + 0x0448

Type: R/W

Reset: 0

Description: This register stores the variable length decoder read limit. When the variable length decoder read pointer reaches VID_VLDRL, reading in the buffer is stopped and the bit VLDRL is set to 1 in the status register. This pointer is defined in 256-byte units.

VID_VLD_PTR Variable length decoder read pointer



Address: *MPEGBaseAddress* + 0x045C

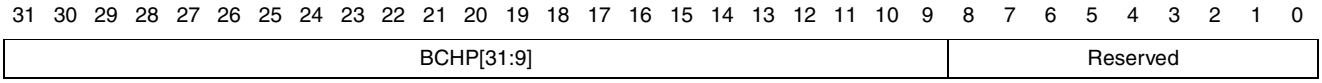
Type: R/W

Reset: 0

Description: Memory address of the variable length decoder read pointer, in bytes. The address where the variable length decoder should start decoding the task must be written here before each decoding task.

This value is taken into account as soon as VID_EXE is accessed. It then holds the current variable length decoder read pointer address given in 128-byte units.

VID_BCHP Backward chroma frame buffer



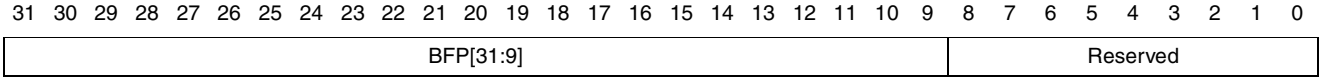
Address: *MPEGBaseAddress* + 0x0488

Type: R/W

Reset: 0

Description: Start address of the backward frame chroma prediction buffer, defined in 512-byte units.

VID_BFP Backward luma frame pointer



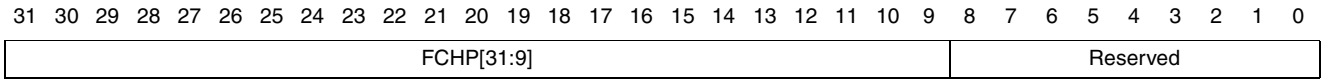
Address: *MPEGBaseAddress* + 0x048C

Type: R/W

Reset: 0

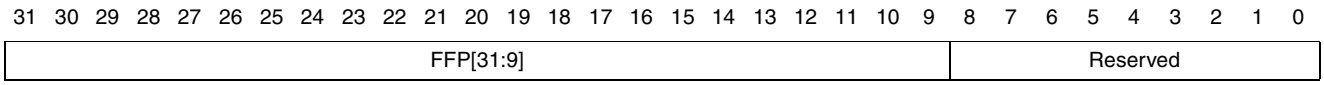
Description: Start address of the luma backward prediction frame picture buffer, defined in 512-byte units.

VID_FCHP Forward chroma frame buffer



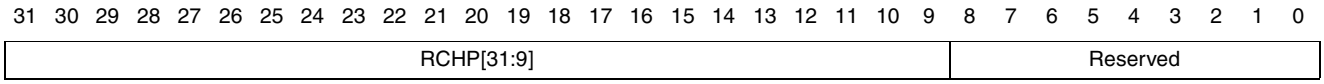
Address: *MPEGBaseAddress* + 0x0490
Type: R/W
Reset: 0
Description: Start address of the chroma forward prediction frame picture buffer, defined in 512-byte units.

VID_FFP Forward luma frame pointer



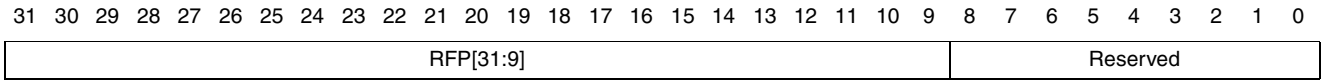
Address: *MPEGBaseAddress* + 0x0494
Type: R/W
Reset: 0
Description: Start address of the luma forward prediction frame picture buffer, defined in 512-byte units.

VID_RCHP Main reconstructed chroma frame pointer



Address: *MPEGBaseAddress* + 0x0498
Type: R/W
Reset: 0
Description: Start address of the reconstructed chroma frame picture buffer, defined in 512-byte units.

VID_RFP Main reconstructed luma frame pointer



Address: *MPEGBaseAddress* + 0x049C
Type: R/W
Reset: 0
Description: Start address of the reconstructed (decoded) luma picture buffer, defined in 512-byte units.

VID_SRCHP Secondary reconstructed chroma frame buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRCHP[31:9]																							Reserved								

Address: *MPEGBaseAddress* + 0x04AC
Type: R/W
Reset: 4
Description: Start address of the secondary reconstructed chroma frame buffer, defined in 512-byte units.

VID_SRFP Secondary reconstructed luma frame buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRFP[31:9]																							Reserved								

Address: *MPEGBaseAddress* + 0x04A4
Type: R/W
Reset: 0
Description: Start address of the secondary reconstructed luma frame buffer, defined in 512-byte units.

VID_RCM Reconstruction mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									VDEC		HDEC		ENM	ENS	PS

Address: *MPEGBaseAddress* + 0x04AC
Type: R/W
Reset: 0x04
Description: Parameters of reconstruction (main and secondary). After hardware reset, main reconstruction is enabled and secondary is disabled.

[31:7] **Reserved**

[6:5] **VDEC**: Vertical decimation mode
00: no decimation
01: V/2 decimation
10: V/4 decimation
Others: Illegal.

[4:3] **HDEC**: Horizontal decimation mode
00: no decimation
01: H/2 decimation
10: H/4 decimation
Others: Illegal.

[2] **ENM**: Enable reconstruction in main buffer

[1] **ENS**: Enable reconstruction in secondary buffer

[0] **PS**: Progressive sequence.
This bit should be set when a progressive sequence is being decoded.
1: Decimation is performed by averaging pixel lines in the frame.

44 H.264 video decoder

44.1 Introduction

This chapter describes the Delta decoder (decode engine for low-bit-rate TV applications) which is the STx7100 H.264 high profile video decoder (also known as MPEG-4 part 10, MPEG-4/AVC and H26L).

44.2 Decoder main features

44.2.1 Supported formats

The Delta decoder is able to decode in real time:

- one H.264 high profile - level 4.1 video stream, or
- up to three H.264 high profile - level 3 video streams.

44.2.2 Input streams

The Delta decoder is a picture level decoder. Additional syntax elements like H.264 sequence parameter set and picture parameter set are handled by the ST40 host processor.

The Delta decoder accepts only **H.264 byte stream NALs**. It only processes slices; all other information (like PTS-DTS, sequence, picture, SEI) are managed externally. The host processor launches the picture decoding process interrupting the embedded processor (ST231).

44.2.3 Miscellaneous features

- Decimated second output (by two or four horizontally and/or by two vertically) in order to reduce memory bandwidth requirements for zoom out. If the picture horizontal size is not a multiple of the decimation factor, the Delta can insert dummy macroblocks. Vertically, the Delta can handle half-blocks.
- Error concealment

Note: The Delta decoder does not support frame width larger than 1920 pels and frame height bigger than 2032 lines.

The minimum picture width is 2 MBs.

The minimum picture height is 2 lines of MBs.

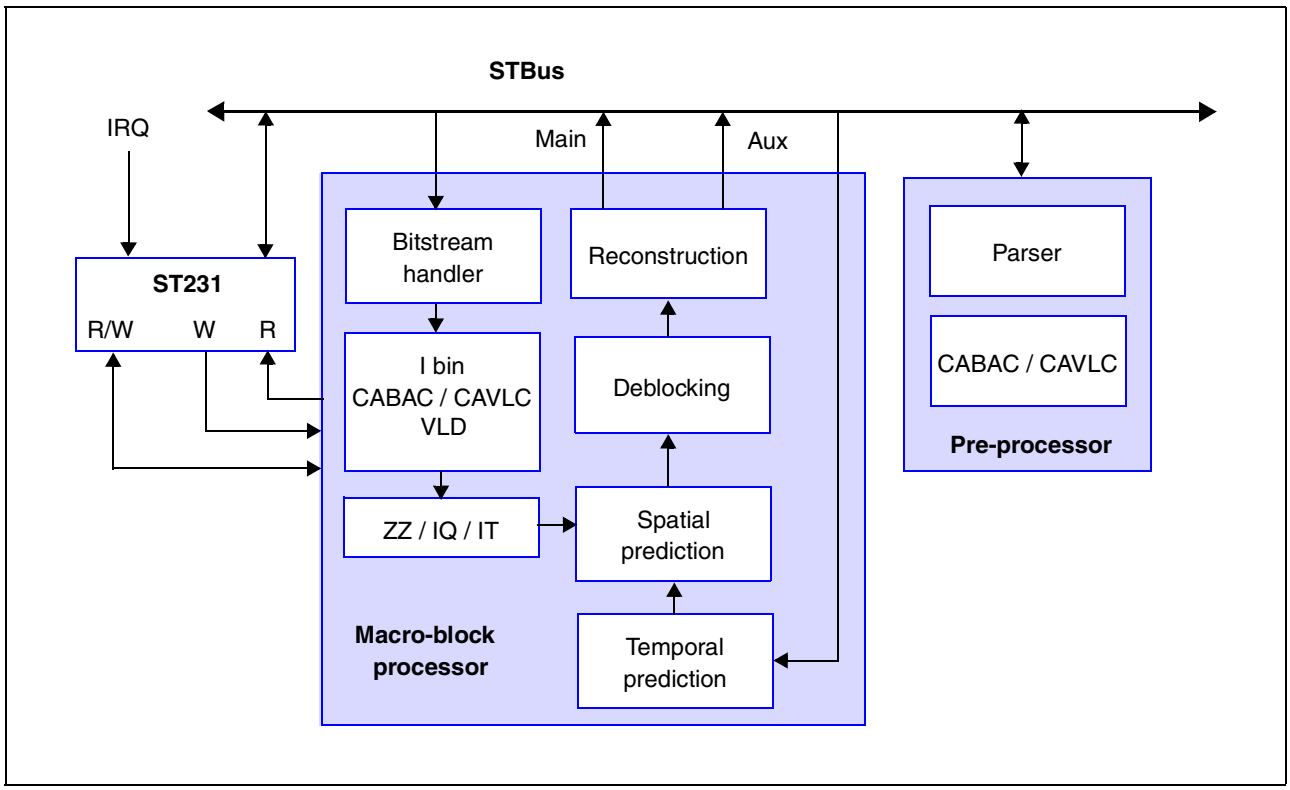
44.2.4 Decoder overview

The Delta decoder is made of three units whose main functions are described below:

- Pre-processor unit (PP),
  - Slice header/data syntax analysis,
  - Macroblock header preprocessing,
  - CABAC and CAVLC entropy decoder (partial decoding),
- Macro-block engine (MBE),
  - DMA management,
  - UE/SE/iBin decoding for H.264 syntax elements (MB headers decoding),
  - CABAC and CAVLC Residual entropy decoding,
  - Inverse zig-zag, IQ, IT,
  - Inter MB motion vectors computation,
  - Temporal and spatial prediction,
  - Deblocking filtering,
  - Picture reconstruction in external memory,
- Embedded processor,
  - ST231 with firmware, used for syntax parsing, error concealment, intraprediction, motion vector processing,
  - Hardwired functions control.

44.2.5 Decoder block diagram

Figure 115: Delta decoder block diagram



44.2.6 Decoding data flow

The Delta decoder is a frame decoder which decodes ES data under the control of the ST40. It is the FDMA which reads the PES previously stored in memory by the PTI and does the PES-to-ES parsing.

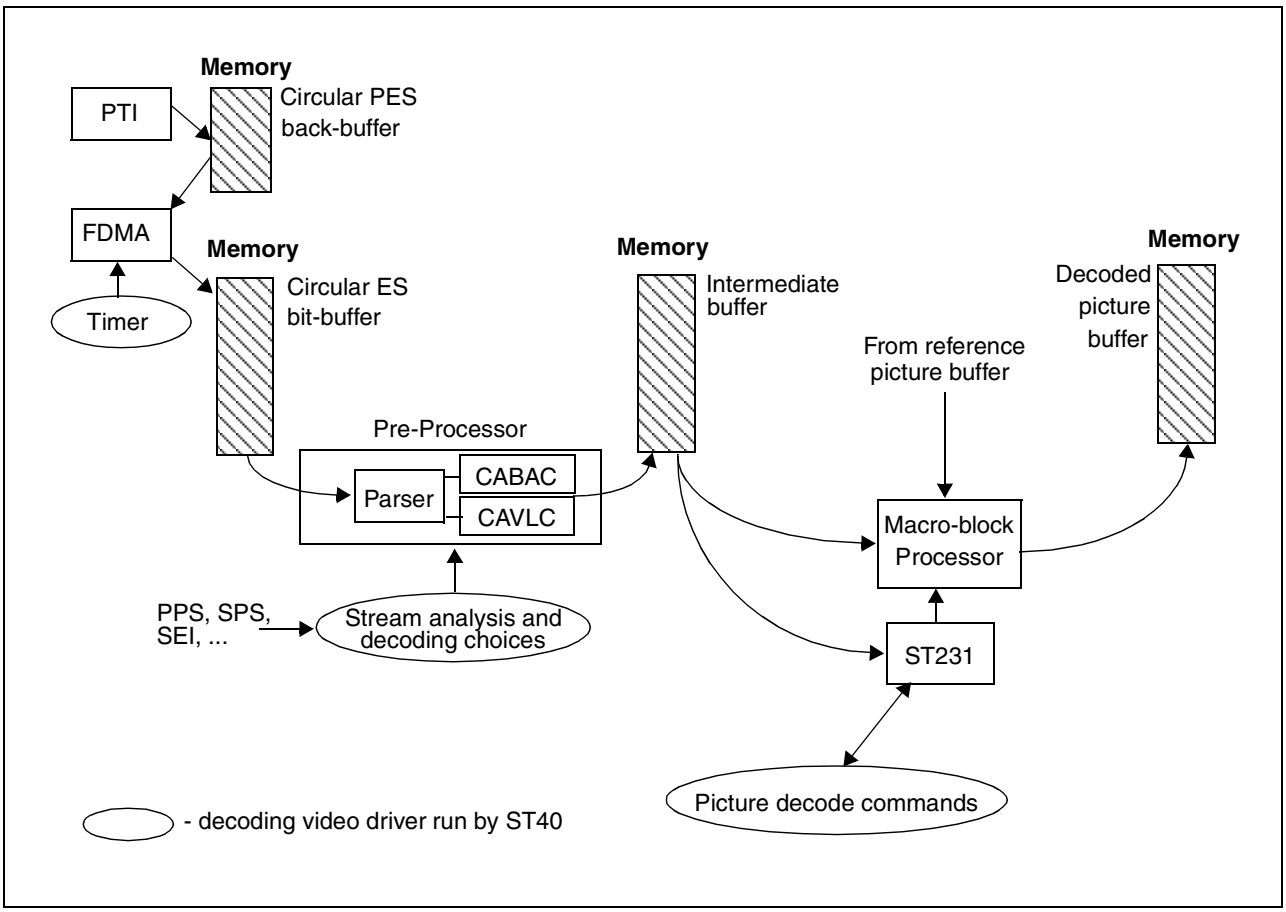
The Delta pre-processor reads the ES data from a circular bit-buffer in memory and stores the resulting data (partially decoded data) into an “intermediate buffer” in memory.

The intermediate buffer is then read by the Delta macro-block processor to complete the decoding. The final decoded frames are stored in the “decoded picture buffer” in memory.

A timer is used to interrupt regularly the ST40 which reads the PTI write-pointer and launches the FDMA to do the PES parsing.

The data flow is shown in Figure 116.

Figure 116: H264 decode dataflow



44.2.7 Buffering requirements

The Delta decoder has the following buffering requirements:

- **Bit buffer (BB)**: circular buffer that contains an elementary stream - ES (NAL byte stream in H.264) - compressed data coming from the MPEG-TS demultiplexer or from a mass storage device (this buffer is compliant with the HRD (hypothetical reference decoder) CPB buffer).
- **Intermediate picture buffer (IB)**: this buffer contains pre-processed picture data (MB header and CABAC/CAVLC already partially decoded).
- **Decoded picture buffer (DPB)**: All decoded pictures (including decimated ones) are stored in memory using the same macroblock arrangement as the MPEG2 decoder, for display and reference purpose.
- **Decimated decoded picture buffer**: to reduce the display bandwidth consumption, it can be useful to decimate decoded picture before storing them in memory. The decoder can also write non-decimated picture in this buffer.
- **Picture parameter buffer (PPB)**: If a picture is marked as used for reference, additional data must be stored in a separate picture parameter buffer. This additional buffer is used for motion vector and reference picture prediction in H.264 temporal direct mode.

Each MB structure is stored in memory with the size defined by MBDescrSize (firmware) - 40 bytes are stored for level 4 and above, 160 for levels below level 4.

44.3 H.264 decoder interface

44.3.1 ST40 host CPU tasks overview

The ST40 host CPU runs the video driver and controls the Delta decoder.

The host is responsible for:

- FDMA control for PES processing / SCD (start code) extraction,
- SPS / PPS / SEI / VUI (video usability information) analysis:
  - sequence parameter set (SPS): the host CPU has to manage up to 32 sets,
  - picture parameter set (PPS): the host CPU has to manage up to 256 sets,
  - supplemental enhancement information messages (SEI): these messages assist in the processes related to decoding or display of video but are not required for reconstructing the luma or chroma samples,
- control of the decoder (pre-processor and core) on a picture basis,
- decoded picture buffer (DPB) management (reconstruction, prediction, display),
- intermediate buffer (IB) management,
- picture boundaries detection in the bit buffer using the slice start-code and the first field of the slice header. The read stop-address passed to pre-processor is the address of the first byte of next picture start code minus one.

44.3.2 Host ST40 - decoder ST231 communication

The Delta decoder is controlled by the ST40 on a picture-by-picture basis using the following methods:

- The ST40 writes picture level decoding commands in external memory.
- The ST40 and the Delta ST231 can each receive and generate interrupts with the dedicated mail box (see Chapter 27: *Mailboxes on page 223*).

The commands or the status placed in memory by the ST40 or by the embedded processor form the multimedia engine (MME) API.

44.4 Firmware

The H.264 firmware is the code executed by the Delta ST231 and is delivered with the software driver.

44.4.1 Picture processing

The decoder firmware gets a picture decode request from the ST40 through a mechanism based on interruption and parameters placed in external memory. The firmware analyzes this request, sends a corresponding picture command to configure the hardware, and can start the processing of the first slice.

Before starting a new picture decode, the firmware checks that all the data of the previous picture has been written to memory.

44.4.2 Slice processing

The firmware gets the slice syntax from the intermediate buffer. It analyzes these syntax elements and sends the corresponding slice command to configure the hardware.

44.4.3 Macroblock processing

The firmware gets the macroblock syntax elements up to residuals, then, the firmware sends a pipeline command that requests the residual entropy decoder (hardware block) to read all the residuals of the current macroblock from the bitstream.

While the pipeline is emptying the input FIFO, the firmware computes all the parameters needed to control the prediction and the deblocking. If direct mode is used for this slice, the firmware read the collocated MB context from PPB buffer (picture parameter buffer). It then sends another MB command to configure the prediction and deblocking engines.

After this processing, the firmware reads the hardware status in order to determine if:

- all the residual have already been read,
- an error occurred on the residual entropy decoding.

44.5 H.264 decoder reset

The preprocessor can be reset by software independently from the core decoder.

Pre-processor reset

The preprocessor is reset by writing a value in the PRE_SW_RST register. An interrupt is generated when the soft reset is completed.

Delta ST231 and macroblock processor reset

The core decoder (ST231 and MB processor) is reset by the global STx7100 reset signal. It is maintained in the reset state until the ST40 releases the reset signal using the configuration register SYS_CFG29. For more details, see Chapter 17: *Reset* *on page 161* and Chapter 20: *Boot modes* *on page 169*.

45 Teletext DMA

45.1 Teletext DMA overview

The STx7100 features a teletext block integrated with the DENC to support one video stream. Teletext data is retrieved from memory, serialized and transferred to the DENC by a dedicated teletext DMA. The digital encoder encodes teletext data according to the *CCIR/ITU-R Broadcast Teletext System B* specification (also known as world system teletext).

45.2 Teletext packet format

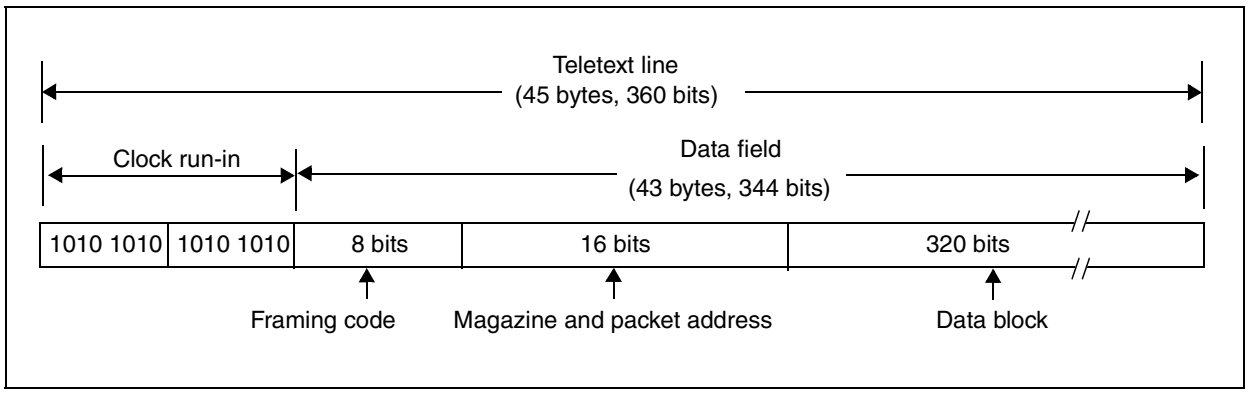
One teletext packet (also called a teletext line) is a stream of 360 bits, transferred at an average frequency of 6.9375 MHz. The data format is the same as the contents of the PES data packet as defined in the ETSI specification. The DMA reads in multiples of 46 bytes and transfers lines of 45 bytes to the digital encoder.

Each teletext packet is composed of the clock run-in and the data-field, as shown in Figure 117.

- The clock run-in is composed of two bytes, each with the value 0xAA (binary 1010 1010).
- The data-field consists of three fields: framing code, magazine and packet address, and data block fields. These three fields provide the block of teletext data.

The framing code is a single byte of hexadecimal value 0xE4[1]. The data is transmitted in order, from the LSB to the MSB of each byte in memory.

Figure 117: Teletext packet format



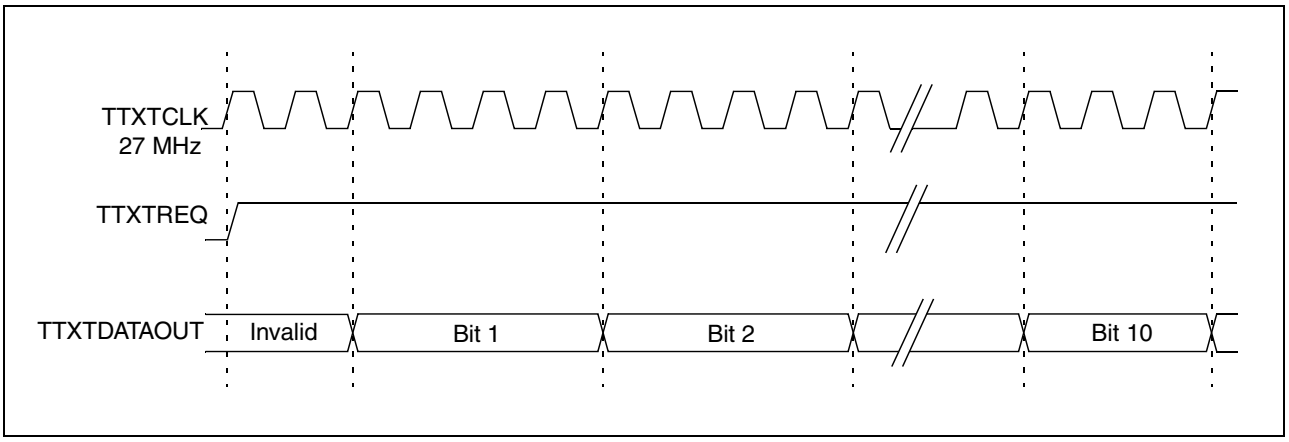
45.3 Data transfer sequence

The digital encoder issues a teletext request signal to the teletext DMA, this is shown by the rising edge of signal TTXTREQ (PIO6[6]) in Figure 118. After a delay, programmable from 2 to 9 master-clock periods, the teletext DMA transmits the first valid teletext data bit of the teletext packet.

The 360 bits of output data are defined as nine 37-bit sequences, ending with one 27-bit sequence. Within each sequence, each bit is transmitted in four 27 MHz cycles, except bits 10, 19, 28 and 37, which are transmitted in three 27 MHz cycles. This is illustrated in Figure 118 for bits 0 to 10.

1. Specification for conveying ITU-R Systems B teletext in digital video broadcasting (DVB) bitstreams.

Figure 118: Teletext data transfer sequence



The duration of the TTXS window is 1402 reference clock periods (51.926 µs), which corresponds to the duration of 360 teletext bits.

The delay between signal TTXTREQ becoming high and the transfer of the first bit of the teletext packet is between two and nine 27 MHz clock cycles. This delay is programmed by register bits DEN_TTX1.TTXDEL[2:0]. The value written to this register is increased by two 27 MHz clock cycles, so the value 0 corresponds to an overall delay of 2 x 27 MHz clock cycles, and the value 7 corresponds to a delay of 9 x 27 MHz clock cycles.

45.4 Interrupt control

Teletext interrupts can be programmed by register TTXT_INT_EN to interrupt the CPU whenever one of the following occurs.

- A teletext data transfer is complete.
- The current video frame toggles odd-to-even or even-to-odd.

The interrupt status is given by register TTXT_INT_STA and masked by TTXT_INT_EN. The interrupt bits are reset when the CPU writes to the acknowledge register, or when a DMA operation is completed.

45.5 DENC teletext registers

Five dedicated digital encoder registers program the teletext encoding in various areas of the vertical blanking interval (VBI) of each field. Four of these areas (that is, blocks of contiguous teletext lines) can **independently** be defined within the two VBIs of one frame (for example, two blocks in each VBI, or three blocks in field 1 VBI and one in field 2 VBI). In certain circumstances, up to four areas may be defined in each VBI.

The five dedicated digital encoder registers are DEN_TTX1 to DEN_TTX5.

45.6 Teletext external interface

Table 126: Teletext interface pins

Pin	Type	Function name	Function description
PIO6[6]	Input	TTXTREQ	Teletext request
PIO6[7]	Output	TTXTDATAOUT	Teletext output data

46 Teletext DMA registers

Register addresses are provided as *TTXTBaseAddress* + offset.

The *TTXTBaseAddress* is:

0x0200 6000.

Table 127: Teletext register summary

Register	Function	Offset	Type
TTXT_DMA_ADDR	Teletext DMA address	0x00	R/W
TTXT_DMA_CNT	Teletext DMA count	0x04	R/W
TTXT_OUT_DLY	Teletext output delay	0x08	R/W
TTXT_INT_STA	Teletext interrupt status	0x18	RO
TTXT_INT_EN	Teletext interrupt enable	0x1C	R/W
TTXT_ACK_ODDEVEN	Teletext acknowledge odd or even	0x20	WO
TTXT_ABORT	Teletext abort	0x24	WO

TTXT_DMA_ADDR Teletext DMA address

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA_ADDR																															

Address: *TTXTBaseAddress* + 0x00

Type: R/W

Reset: Undefined

Description: This 32-bit register specifies the base address in memory for the DMA transfer from memory.

TTXT_DMA_CNT Teletext DMA count

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DMA_CNT										

Address: *TTXTBaseAddress* + 0x04

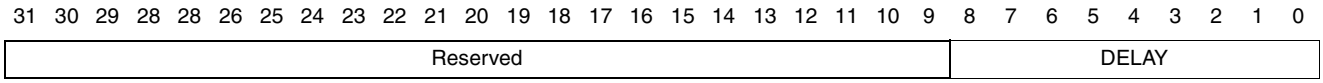
Type: R/W

Reset: Undefined

Description: This register specifies the number of bytes to be transferred from memory during the DMA operation. A write to this register also starts the teletext output operation.

- For teletext output operation, this value must be:
  46 x *number_of_teletext_lines_to_send* bytes.
- For teletext input operation, the value must be:
  42 x *number_of_teletext_lines_to_receive* bytes.

TTXT_OUT_DLY Teletext output delay



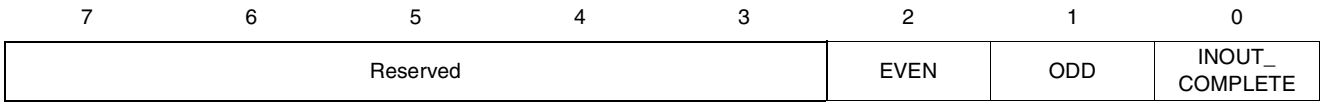
Address: *TTXTBaseAddress* + 0x08

Type: R/W

Reset: Undefined

Description: This register programs the delay, in 27 MHz clock periods, from the rising edge of TTXTREQ to the first valid teletext data bit, that is, TTXTDATAOUT starting to transmit.

TTXT_INT_STA Teletext interrupt status



Address: *TTXTBaseAddress* + 0x18

Type: Read only

Reset: Undefined

Description: This register gives the current state of the teletext operations. If the appropriate bits in TTXT_INT_EN are set, interrupts can be driven by the state of this register.

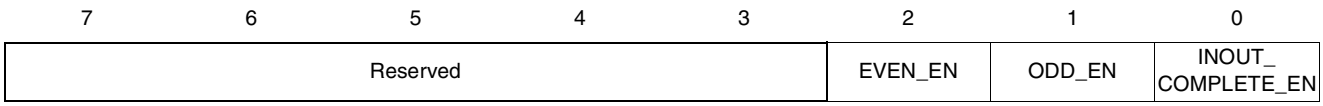
[31:3] **Reserved**

[2] **EVEN**: Current (video encoder) field is even.

[1] **ODD**: Current (video encoder) field is odd.

[0] **INOUT_COMPLETE**: Teletext input or output operation completed.

TTXT_INT_EN Teletext interrupt enable



Address: *TTXTBaseAddress* + 0x1C

Type: R/W

Reset: Undefined

Description: This register allows masking of register TTXT_INT_STA.

[31:3] **Reserved**

[2] **EVEN_EN**: Enable even field interrupt.

[1] **ODD_EN**: Enable odd field interrupt.

[0] **INOUT_COMPLETE_EN**: Enable teletext input or output operation completed interrupt.

TTXT_ACK_ODDEVEN Teletext acknowledge odd or even

Address: *TTXTBaseAddress* + 0x20

Type: WO

Description: This register acknowledges the odd/even toggle interrupt. Any write to this register clears bits ODD and EVEN of register TTXT_INT_STA.

TTXT_ABORT Teletext abort

Address: *TTXTBaseAddress* + 0x24

Type: WO

Description: Any write to this address causes the teletext interface to abort the current operation. The state of the teletext operation is reset, and the teletext data transfer is interrupted. The DMA engine is reset only after the current word read or write is complete.

47 2D graphics processor (blitter)

47.1 Overview

The 2D-graphics processor is a CPU accelerator for graphics picture handling. It is a dual-source DMA, with a set of powerful operators. It receives data from the local memory through two input sources, source 1 and source 2.

- Source 1 is used for frequent operations such as color-fill or simple source-copy; it has a 64-bit wide internal bus and performs according to the pixel format.
- All operators always apply to source 2. The processing pipeline bus is always a pixel bus (ARGB8888 format), whatever the format of the source inputs.
- Sources 1 and 2 are used simultaneously for read/modify/write operations.

The 2D-graphics processor is software-controlled by a link-list mechanism. Each node of the link list is an instruction that contains all the necessary information to proceed.

The 2D-graphics processor operates at a clock rate of up to 153 MHz. For each operation involving source 2, the maximum output rate is 153 Mpixel/s, whatever the pixel depth. This rate is constant except for 2D resizing for downsampling, where it is 100 x HSF x VSF (horizontal and vertical scaling factors). When source 1 is used in direct-copy mode, the internal bus width is 64 bits and the maximum output rate is 800 Mbyte/s. These values are maximum performances that the blitter can reach; they assume that no read/write memory word request has been postponed by the memory arbiter mechanism.

Table 128: Performance for typical operations over a range of target pixel sizes (Mpix/s, at output), for a 100 MHz clock

Operation	Performance (in megapixels)						
	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp	32 bpp
Fill (source 1)	4800	2400	1200	600	300	200	150
One-source blit (source 1)	4800	2400	1200	600	300	200	150
One-source blit (source 2) shade	100 x GSF[a]	100 x GSF	100 x GSF	100 x GSF	100 x GSF	100 x GSF	100 x GSF
Two-source blit	100 x GSF	100 x GSF	100 x GSF	100 x GSF	100 x GSF	100 x GSF	100 x GSF

a. GSF (global scaling factor) = HSF x VSF.

The 4:2:0 plane can be used as a source for the 2D-graphics engine. Because this plane uses both the source 1 and source 2 buffers, special modes are required. These modes are described in *Section 47.6 on page 452*.

47.2 Functions

- Solid color fill of rectangular window.
- Solid color shade (fill plus alpha blending).
- One source copy, with one or several operators enabled (color format conversion, 2D scaling).
- Two-source copy with alpha blending or logical operations between them.
- 4:2:2 raster as source or destination format.
- 4:2:2 and 4:2:0 macroblock as a source format.
- Color space conversion RGB to or from YCbCr.
- Color expansion (CLUT to true color).
- Color correction (gamma, contrast, gain).
- Color reduction (true color to CLUT1, CLUT8 and CLUT4 or CLUT2) using an error diffusion algorithm.
- 2D resize engine with high quality filtering.
- Adaptive flicker filter from memory to memory.
- Color keying capability.
- Rectangular clipping.
- Programmable source/target scanning direction, both horizontally and vertically, in order to handle overlapping source and destination areas correctly.
- 1-bit/8-bit clipmask bitmap so random shape clipping can be achieved in two passes.
- Plane mask feature available.
- Special XYL access mode, to speed random pixel access, or horizontal line drawing (such as polygon filling or run-length decoder accelerator). See Section 47.7 on page 453.

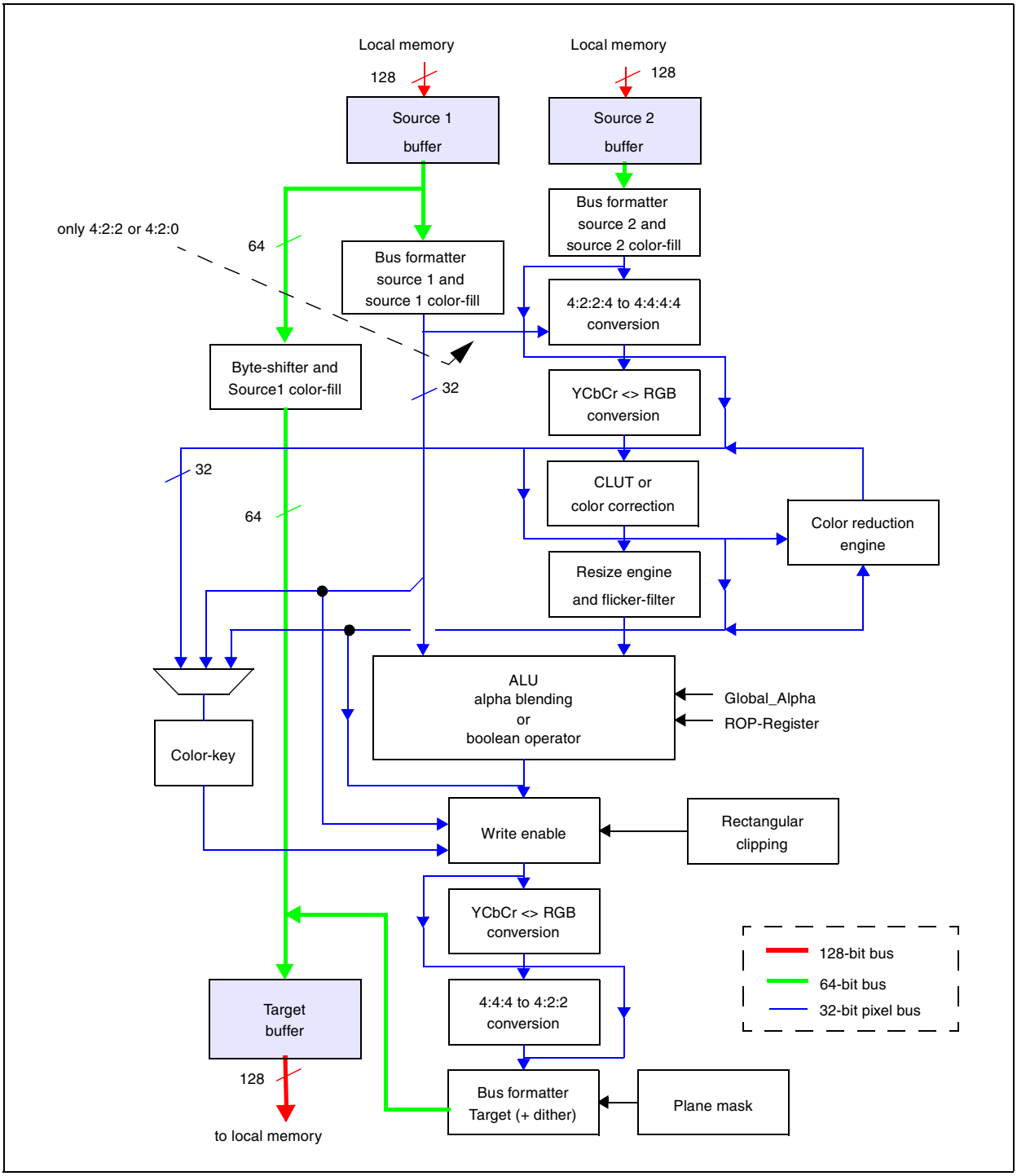
Source and destination windows are all defined using an XY descriptor, with pixel accuracy from 1 to 32 bpp, whatever the format.

Most of these operators can be combined in a single blitter pass. For instance, a YCbCr 4:2:2 bitmap can be converted to 4:4:4 RGB, resized and finally blended on an RGB565 background picture.

47.3 Functional block diagram

The following figure illustrates the blitter functions that can be performed. Each of the blitter functions is briefly described in Section 47.5 on page 433.

Figure 119: 2D graphics processor data flow



The blitter works from memory to memory with a dual or single source and one target. The target can be one of the sources. A set of overlaps between sources and target is supported, provided that the pixmap horizontal and vertical scan ordering are correctly programmed for each source and the target. Each source and target may be programmed independently.

Each source and the target is associated with a specific set of registers. Another register (BLT_INS) is used to control the data flow with operator enables. Each time an operator is

enabled, the user has to specify its behavior with operator specific registers. If the blitter is not in direct mode (64-bit internal bus), the ALU operator is always enabled and must be programmed.

47.3.1 The blitter process

The 2D graphics processor has control and status registers allowing global blitter operation management:

- BLT_CTRL is the control register where the user can launch a blitter link-list sequence, suspend the blitter sequence and reset the hardware block.
- BLT_STA1, BLT_STA2, BLT_STA3 are status registers used to monitor the blitter operation states, such as the hardware block state (ready or busy), the current blitter node address being processed, the current target line number, and some interrupt management.

The basic concept of a blitter is to program one or several nodes in memory, where each node operates on all blitter registers except the control and status registers. For several nodes, a link-list mechanism must be set. The hardware can then automatically read the next node located inside the local memory and continue a sequence of blitter operations. This mechanism allows multipass operations for complex sequences in order to limit software load. This link-list mechanism is provided by register BLT_NIP that contains the next blitter address. If this pointer is set to zero, the hardware understands that the link-list mechanism has ended.

The procedure to start the blitter is the following:

1. Prepare the link list of nodes in the local memory.
2. Program the first blitter node address inside hardware BLT_NIP.
3. Set the trigger start condition if needed (BLT_INS.TRIG_COND_CTRL) and start the blitter by setting bit START (BLT_CTRL), and then setting it back to 0.
4. If trigger on a raster scan line has been selected, the user must program register BLT_RST to set the line number.

Note: The blitter sees a 32-bit address space, but always writes inside a 64 Mbyte bank for a given object (for example, source-target node).

The blitter can set three kinds of interrupt that may be masked by IRQ_MASK (BLT_INS):

- once a blitter pass is completed and before the next has been loaded,
- after the blitter instructions have been loaded,
- once the blitter has reached the idle state following the suspend command.

For each interrupt, the blitter stops and the user should restart it to continue.

47.3.2 Data flow control

SOURCE1 (register BLT_INS) sets the source 1 mode: direct copy, direct fill, fetch pixel flow from memory, or color fill.

If the direct mode is selected for source 1 (high pixel throughput using the 64-bit internal bus), the bus formatter is removed from the path. The target format should thus match the source 1 format, and no graphics operation may be used except for color fill or pixel copy.

Color fill mode replaces the pixel flow from the local memory by an internal flow of constant color pixels that match the source specification (format, number of pixels: window size, expansion mode).

SOURCE2 (BLT_INS) sets the source 2 mode: either fetch pixel flow from memory, or color fill.

Bit ICSC enables the input color space converter for source 2, and bit OCSC enables the output color space converter for the target (BLT_INS).

Note: The output color space converter (from RGB to YCbCr or from YCbCr to RGB) and the input color space converter, if enabled simultaneously, can only perform opposite conversions.

Bit CLUTOP enables the CLUT operator on source 2, 2DRESCALE enables the 2D resize operator on source 2, and FLICK_FILT enables the flicker filter operator on source 2 (BLT_INS).

Bit RECT_CLIP enables the rectangle clipping operator on target, bit CKEY enables the color key operator from source 1 or source 2 to target, and bit PLANE_MASK enables the plane mask bit protection operation on target with respect to source 1 (BLT_INS).

User information

Register BLT_USR is used as user information for the node for software purposes.

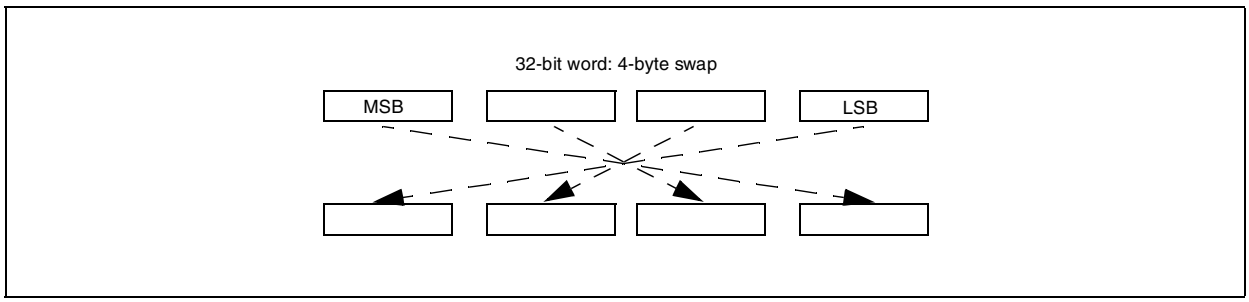
Note: As a result of the number of input formats, output formats, available operators and different configurations, many register combinations have no meaning in terms of the graphics blitter. Registers must be programmed with this in mind.

47.4 Endianness support

Bit (BLT_PKZ.BIGNOTLITTLE) indicates whether the nodes, filters coefficients, CLUT palette, and XYL memory buffer content are stored in big or little endian format. This bit must be compliant with the host CPU endianness (little endian). The EMPI endianness setting should also be the same.

When the nodes, filter coefficients and CLUT palette are fetched, a four-byte swap is performed if BLT_PKZ.BIGNOTLITTLE is set.

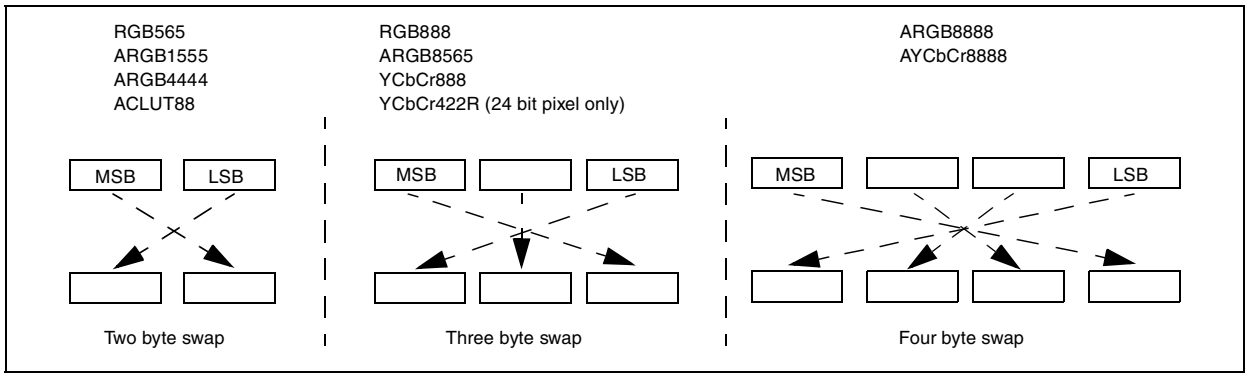
Figure 120: Endianness conversion for nodes, filter coefficients, CLUT, XYL parameters



For a given node, the endianness may be defined for the source 1 bitmap, source 2 bitmap, and target bitmap (BIGNOTLITTLE in BLT_S1TY, BLT_S2TY and BLT_TTY). Usually bitmap endianness is in line with the host CPU endianness, and these bits are programmed in the same way as BIGNOTLITTLE in BLT_PKZ. However, when importing a bitmap, for example, its endianness can be converted. This corrects all potential endianness issues. The endianness

format conversion affects only the 16 bpp, 24 bpp and 32 bpp formats. For YCbCr422R format, the pixels that only have 8 bit luminance remain unaffected, only the 24 bit pixels are swapped.

Figure 121: Endianness conversion for big endian pixels



In direct copy mode, the endianness of the bitmap is not taken into account. This is a pure memory to memory operation without any modification.

47.5 Blitter functions

47.5.1 Source 2 address generator and bus formatter

The source 2 address generator scans a rectangular window, according to an XY addressing scheme. The XY pixel address is converted into a physical memory address, using the source 2 description registers.

Reference formula for address conversion:

MemAddress (*x*, *y*) = *BaseAddress* + *y* * *Pitch* + *x* * (*number of bytes per pixel*)

Register BLT_S2BA is the memory base address of the bitmap selected for source 2 (absolute (0,0) location, top-left pixel).

Register BLT_S2TY provides the specific properties of source 2: format, pitch, horizontal and vertical scan order, sub-byte ordering (for sub-byte formats), bit accuracy expansion mode, chroma sign (for YCbCr formats), chroma features (YCbCr macro-block formats).

Register BLT_S2XY is the coordinate that specifies the start of the input source 2 window with respect to the base address.

Register BLT_S2SZ specifies the size of the source 2 window.

The source 2 bus formatter converts a 128-bit input bus into a pixel bus, depending on the current format selected for source 2. The internal pixel bus is 32 bits wide, but a subset of the data is used by many color format modes.

All formats are available.

The pixel bus format is as follows:

Table 129A: Source 2 formatter output / true color 4:4:4

True color	Alpha		Color data					
	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
RGB565	128		R5/3MSB or 000		G6/2MSB or 00		B5/3MSB or 000	
RGB888	128		R8		G8		B8	
ARGB1555	A1/0000000		R5/3MSB or 000		G5/3MSB or 000		B5/3MSB or 000	
ARGB8565	A8		R5/3MSB or 000		G6/2MSB or 00		B5/3MSB or 000	
ARGB8888	A8		R8		G8		B8	
ARGB4444	0/A4/100 (but keep 0 and 128)		R4/4MSB or 0000		G4/4MSB or 0000		B4/4MSB or 0000	
YCbCr888	128		Cr8		Y8		Cb8	
AYCbCr8888	A8		Cr8		Y8		Cb8	

Table 129B: Source 2 formatter output / CLUT-based formats

CLUT-based formats	Alpha		Index value					
	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
ACLUT1	0 (don't care)						0000 000I_0	
ACLUT2	0 (don't care)						0000 00I_1I_0	
ACLUT4	0 (don't care)						0000 $I_3I_2I_1I_0$	
ACLUT8	0 (don't care)						$I_7I_6I_5I_4$ $I_3I_2I_1I_0$	
ACLUT44	0/A4/100 (but keep 0 and 128)		0 (don't care)				0000 $I_3I_2I_1I_0$	
ACLUT88	A8		0 (don't care)				$I_7I_6I_5I_4$ $I_3I_2I_1I_0$	

Table 130: Source 2 formatter output / alpha only formats

Alpha-only formats	Alpha value		Color data					
	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
A1	A1/0000000		0					
A8	A8		0					

Table 131: Source 2 formatter output / YCbCr 4:2:2 raster format

YCbCr 4:2:2	Color data							
	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
YCbCr 4:2:2 (1st pixel YCbCr)	Cb 1st pixel		Y 1st pixel		Cr 1st pixel		Y 2nd pixel	
YCbCr 4:2:2 (1st pixel Y-only)	Y 1st pixel		Cb 2nd pixel		Y 2nd pixel		Cr 2nd pixel	

Note: In this last mode, no alpha value is produced by the block, and the bus carries two pixels simultaneously. The alpha channel (128 = opaque) is inserted by the 4:2:2 to 4:4:4 converter (next block in the pipeline), that is automatically enabled in this configuration.

When the input picture has an 8-bit per pixel alpha component, this component can have a 0 to 128 or 0 to 255 range. The internal 8-bit alpha format being 0 to 128, a 0 to 255 alpha component is converted using the following formula: $A_{0\ to\ 128} = (A_{0\ to\ 255} + 1) \times 2^{-1}$. The alpha range formatter is present for source 1, source 2 and target.

47.5.2 Source 1 address generator and bus formatter

The source 1 address generator scans a rectangular window, according to an XY addressing scheme. The XY pixel address is converted into a physical memory address, using the source 1 registers.

Each time the 2D graphics engine performs a read/modify/write cycle to combine background information with new data, the source 1 bus formatter converts a 128-bit input bus into a pixel bus that feeds the ALU operator.

Source 1 is used for frequent operations such as color-fill or simple source-copy.

In fast direct-copy mode, the source 1 bus formatter is unused, and the transfer uses a 64-bit wide internal bus, whatever the color format and its bit-depth. Rectangular graphics areas are transferred faster, but sub-byte color formats are not supported. Neither the plane mask nor rectangular clipping features can be used.

In normal mode, source 1 data can be combined with source 2 data.

Register BLT_S1BA is the memory base address of the bitmap selected for source 2 (absolute (0,0) location, top-left pixel).

BLT_S1TY provides the specific properties of source 1: format, pitch, horizontal and vertical scan order, sub-byte ordering (for sub-byte formats), color depth expansion mode.

BLT_S1XY is the coordinate that specifies the start of the source 1 input window with respect to the base address.

BLT_S1SZ specifies the size of the source 1 window (equals the target size).

47.5.3 Color fill

Registers BLT_S1CF and BLT_S2CF can provide the solid-color value for filling during a blitter operation. BLT_S1CF allows the direct-fill mode to be used, for faster performance, but not in sub-byte modes.

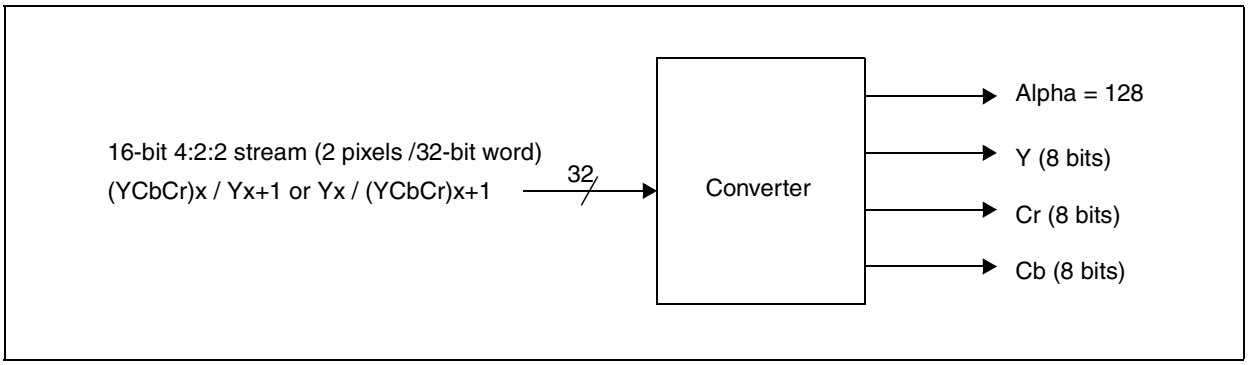
The number of significant bits varies from 1 bpp to 32 bpp according to the bit-depth of the color format.

All color formats are supported, except YCbCr4:2:0MB and YCbCr4:2:2MB.

47.5.4 4:2:2 to 4:4:4 conversion (horizontal chroma upsampler)

This conversion applies to source 2 only if the source format set in register BLT_S2TY is YCbCr422raster. The 4:2:2 input is a color-only raster signal. An opaque alpha channel is added at the block output.

Figure 122: 4:2:2 to 4:4:4 converter



Missing chroma samples are recovered with a 2-tap interpolator. Possible side effects on the edges are handled using sample duplication, if required, using the following schemes.

The input source line starts at address x_0, and contains *n* pixels.

Convention: when x_0 is even, the first pixel is a complete CbYCr pixel, when x_0 is odd, the first pixel is a Y-only pixel. Four cases must be taken into account:

Table 132: x_0 even / *n* even

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Input						
$Cb_1Y_1Cr_1$	Y_2	$Cb_3Y_3Cr_3$	Y_4	.....	$Cb_{n-1}Y_{n-1}Cr_{n-1}$	Y_n
Outputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_1	$(Cb_1+Cb_3) / 2$	Cb_3	$(Cb_3+Cb_5) / 2$	.....	Cb_{n-1}	Cb_{n-1}
Cr_1	$(Cr_1+Cr_3) / 2$	Cr_3	$(Cr_3+Cr_5) / 2$	.....	Cr_{n-1}	Cr_{n-1}

Table 133: x_0 even / *n* odd

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Input						
$Cb_1Y_1Cr_1$	Y_2	$Cb_3Y_3Cr_3$	Y_4	.....	Y_{n-1}	$Cb_nY_nCr_n$
Outputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_1	$(Cb_1+Cb_3) / 2$	Cb_3	$(Cb_3+Cb_5) / 2$	.....	$(Cb_{n-2}+Cb_n) / 2$	Cb_n
Cr_1	$(Cr_1+Cr_3) / 2$	Cr_3	$(Cr_3+Cr_5) / 2$	.....	$(Cr_{n-2}+Cr_n) / 2$	Cr_n

Table 134: x_0 odd / *n* odd

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Input						
Y_1	$Cb_2Y_2Cr_2$	Y_3	$Cb_4Y_4Cr_4$	.....	$Cb_{n-1}Y_{n-1}Cr_{n-1}$	Y_n
Outputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_2	Cb_2	$(Cb_2+Cb_4) / 2$	Cb_4	.....	Cb_{n-1}	Cb_{n-1}
Cr_2	Cr_2	$(Cr_2+Cr_4) / 2$	Cr_4	.....	Cr_{n-1}	Cr_{n-1}

Table 135: x_0 odd / *n* even

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Input						
Y_1	$Cb_2Y_2Cr_2$	Y_3	$Cb_4Y_4Cr_4$	.....	Y_{n-1}	$Cb_nY_nCr_n$
Outputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_2	Cb_2	$(Cb_2+Cb_4) / 2$	Cb_4	.....	$(Cb_{n-2}+Cb_n) / 2$	Cb_n
Cr_2	Cr_2	$(Cr_2+Cr_4) / 2$	Cr_4	.....	$(Cr_{n-2}+Cr_n) / 2$	Cr_n

47.5.5 4:4:4 to 4:2:2 conversion (horizontal chroma downsampler on target)

This block is automatically enabled by the hardware if the target output format is YCbCr 4:2:2 raster. In this case, neither vertical resize nor flicker filter can be used at the same time as horizontal resize.

The sampling rate conversion for the chroma component uses a three-tap 1:2:2 digital filter, as shown in the next tables.

The output target line starts at address x_0, and contains *n* pixels.

Convention: when x_0 is even, the first pixel is a complete CbYCr pixel, when x_0 is odd, the first pixel is a Y-only pixel. Four cases must be taken into account:

Table 136: x_0 even / *n* even

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Inputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_1	Cb_2	Cb_3	Cb_4	.....	Cb_{n-1}	Cb_n
Cr_1	Cr_2	Cr_3	Cr_4	.....	Cr_{n-1}	Cr_n
Outputs						
Y_1 $(\mathbf{Cb_1}+2Cb_1+Cb_2) / 4$ $(\mathbf{Cr_1}+2Cr_1+Cr_2) / 4$	Y_2	Y_3 $(Cb_2+2Cb_3+Cb_4) / 4$ $(Cr_2+2Cr_3+Cr_4) / 4$	Y_4	.....	Y_{n-1} $(Cb_{n-2}+2Cb_{n-1}+Cb_n) / 4$ $(Cr_{n-2}+2Cr_{n-1}+Cr_n) / 4$	Y_n

Table 137: x_0 even / *n* odd

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Inputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_1	Cb_2	Cb_3	Cb_4	.....	Cb_{n-1}	Cb_n
Cr_1	Cr_2	Cr_3	Cr_4	.....	Cr_{n-1}	Cr_n
Outputs						
Y_1 ($\mathbf{Cb_1}$+2Cb_1+Cb_2) / 4 ($\mathbf{Cr_1}$+2Cr_1+Cr_2) / 4	Y_2	Y_3 (Cb_2+2Cb_3+Cb_4) / 4 (Cr_2+2Cr_3+Cr_4) / 4	Y_4	.....	Y_{n-1}	Y_n (Cb_{n-1}+2Cb_n+$\mathbf{Cb_n}$) / 4 (Cr_{n-1}+2Cr_n+$\mathbf{Cr_n}$) / 4

Table 138: x_0 odd / *n* odd

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Inputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_1	Cb_2	Cb_3	Cb_4	.....	Cb_{n-1}	Cb_n
Cr_1	Cr_2	Cr_3	Cr_4	.....	Cr_{n-1}	Cr_n
Outputs						
Y_1	Y_2 (Cb_1+2Cb_2+Cb_3) / 4 (Cr_1+2Cr_2+Cr_3) / 4	Y_3	Y_4 (Cb_3+2Cb_4+Cb_5) / 4 (Cr_3+2Cr_4+Cr_5) / 4	.....	Y_{n-1} (Cb_{n-2}+2Cb_{n-1}+Cb_n) / 4 (Cr_{n-2}+2Cr_{n-1}+Cr_n) / 4	Y_n

Table 139: x_0 odd / *n* even

Pixel 1 (x0)	Pixel 2	Pixel 3	Pixel 4	.....	Pixel *n*-1	Pixel *n*
Inputs						
Y_1	Y_2	Y_3	Y_4	.....	Y_{n-1}	Y_n
Cb_1	Cb_2	Cb_3	Cb_4	.....	Cb_{n-1}	Cb_n
Cr_1	Cr_2	Cr_3	Cr_4	.....	Cr_{n-1}	Cr_n
Outputs						
Y_1	Y_2 (Cb_1+2Cb_2+Cb_3) / 4 (Cr_1+2Cr_2+Cr_3) / 4	Y_3	Y_4 (Cb_3+2Cb_4+Cb_5) / 4 (Cr_3+2Cr_4+Cr_5) / 4	.....	Y_{n-1}	Y_n (Cb_{n-1}+2Cb_n+$\mathbf{Cb_n}$) / 4 (Cr_{n-1}+2Cr_n+$\mathbf{Cr_n}$) / 4

47.5.6 YCbCr-to-RGB conversion

The 2D graphics engine color converter complies with ITU-R BT.601 and ITU-R BT.709 color systems, and the RGB components are gamma-corrected.

Graphics matrix

The following ranges are assumed:

0 ≤R, G, B ≤255
16 ≤Y ≤235
16 ≤Cb, Cr ≤240 in offset binary representation, -112 ≤Cb, Cr ≤+112 in two's complement signed representation

Converting from YCbCr to RGB:

- The Y component is clipped between 16 and 235, before applying the matrix.
- The Cb and Cr components can be signed or unsigned.
- Once signed, the chroma range is -112 to +112.
- RGB output components are saturated between 0 and 255.

The next tables show the hardwired matrices used for the conversion (assuming offset binary chroma):

Table 140: 601 colorimetry / floating-point matrix / digital range

YCbCr to RGB reference floating-point matrix									
R	=	1.1641	x (Y - 16)	+	0.0	x (Cb - 128)	+	1.5958	x (Cr - 128)
G	=	1.1641	x (Y - 16)	-	0.3914	x (Cb - 128)	-	0.8135	x (Cr - 128)
B	=	1.1641	x (Y - 16)	+	2.0178	x (Cb - 128)	+	0.0	x (Cr - 128)

Table 141: 601 colorimetry / integer matrix / digital range

YCbCr to RGB integer matrix as implemented (1.0 <> 256)									
R	=	298	x (Y - 16)	+	0	x (Cb - 128)	+	409	x (Cr - 128)
G	=	298	x (Y - 16)	-	100	x (Cb - 128)	-	208	x (Cr - 128)
B	=	298	x (Y - 16)	+	517	x (Cb - 128)	+	0	x (Cr - 128)

Table 142: 709 colorimetry / floating-point matrix / digital range

YCbCr to RGB reference floating-point matrix									
R	=	1.1644	x (Y - 16)	+	0.0	x (Cb - 128)	+	1.7930	x (Cr - 128)
G	=	1.1644	x (Y - 16)	-	0.2129	x (Cb - 128)	-	0.5326	x (Cr - 128)
B	=	1.1644	x (Y - 16)	+	2.1128	x (Cb - 128)	+	0.0	x (Cr - 128)

Table 143: 709 colorimetry / floating-point matrix / digital range

YCbCr to RGB integer matrix as implemented (1.0 <> 256)									
R	=	298	x (Y - 16)	+	0	x (Cb - 128)	+	459	x (Cr - 128)
G	=	298	x (Y - 16)	-	54	x (Cb - 128)	-	136	x (Cr - 128)
B	=	298	x (Y - 16)	+	541	x (Cb - 128)	+	0	x (Cr - 128)

Video matrix

In the case of video matrix, Y, Cb and Cr are not clipped as in the graphics matrix. The following ranges are assumed:

0 ≤R, G, B ≤255
0 ≤Y ≤255
0 ≤Cb, Cr ≤255 in offset binary representation, -128 ≤Cb, Cr ≤+127 in two's complement signed representation.

Converting from YCbCr to RGB:

- The Y component is not clipped before applying the matrix.
- The Cb and Cr components can be signed or unsigned.
- Once signed, the chroma range is -128 to +127.
- RGB output components are saturated between 0 and 255.

The next tables show the hardwired matrices used for the conversion (assuming offset binary chroma):

Table 144: 601 colorimetry / floating-point matrix / digital range

YCbCr to RGB reference floating-point matrix									
R	=	1	x (Y)	+	0.0	x (Cb - 128)	+	1.3828	x (Cr - 128)
G	=	1	x (Y)	-	0.3359	x (Cb - 128)	-	0.6992	x (Cr - 128)
B	=	1	x (Y)	+	1.7344	x (Cb - 128)	+	0.0	x (Cr - 128)

Table 145: 601 colorimetry / integer matrix / digital range

YCbCr to RGB integer matrix as implemented (1.0 <> 256)									
R	=	256	x (Y)	+	0	x (Cb - 128)	+	351	x (Cr - 128)
G	=	256	x (Y)	-	86	x (Cb - 128)	-	179	x (Cr - 128)
B	=	256	x (Y)	+	444	x (Cb - 128)	+	0	x (Cr - 128)

Table 146: 709 colorimetry / floating-point matrix / digital range

YCbCr to RGB reference floating-point matrix									
R	=	1	x (Y)	+	0	x (Cb - 128)	+	1.5391	x (Cr - 128)
G	=	1	x (Y)	-	0.1836	x (Cb - 128)	-	0.4570	x (Cr - 128)
B	=	1	x (Y)	+	1.8125	x (Cb - 128)	+	0	x (Cr - 128)

Table 147: 709 colorimetry / integer matrix / digital range

YCbCr to RGB integer matrix as implemented (1.0 <> 256)									
R	=	256	x (Y))	+	0	x (Cb - 128)	+	394	x (Cr - 128)
G	=	256	x (Y)	-	47	x (Cb - 128)	-	117	x (Cr - 128)
B	=	256	x (Y)	+	464	x (Cb - 128)	+	0	x (Cr - 128)

47.5.7 RGB-to-YCbCr conversion

Graphics matrix

Converting from RGB to YCbCr:

- The Y component is clipped between 16 and 235, after applying the matrix.
- The chroma range is -112 to +112, and can be optionally encoded in offset-binary format (+128, 16/240 range).

In register BLT_CCO, CCO_INCOL sets the input color converter colorimetry (601 or 709), and CCO_INSIGN sets the input color converter chroma color sign. CCO_INDIR sets the input color converter direction (RGB to YCbCr or YCbCr to RGB), and the output color converter is automatically programmed with the other direction.

CCO_OUTCOL sets the output color converter colorimetry (601 or 709), and CCO_OUTSIGN sets the output color converter chroma color format: signed or unsigned.

CCO_INGFX*n*VID sets the input color converter to graphics or video matrix, and CCO_OUTGFX*n*VID sets the output color converter to graphics or video matrix.

The next tables show the hardwired matrices used for the conversion (assuming offset binary chroma):

Table 148: 601 colorimetry / floating-point matrix / digital range

RGB to YCbCr reference floating-point matrix											
Y	=		0.257	xR	+	0.504	xG	+	0.098	xB	+	16
Cb	=	-	0.148	xR	-	0.291	xG	+	0.439	xB	+	128
Cr	=		0.439	xR	-	0.368	xG	-	0.071	xB	+	128

Table 149: 601 colorimetry / integer matrix / digital range

RGB to YCbCr integer matrix as implemented (1.0 <> 1024)											
Y	=		263	xR	+	516	xG	+	100	xB	+	16
Cb	=	-	152	xR	-	298	xG	+	450	xB	+	128
Cr	=		450	xR	-	377	xG	-	73	xB	+	128

Table 150: 709 colorimetry / floating-point matrix / digital range

RGB to YCbCr reference floating-point matrix											
Y	=		0.1825	xR	+	0.6144	xG	+	0.0619	xB	+	16
Cb	=	-	0.1006	xR	-	0.3386	xG	+	0.4392	xB	+	128
Cr	=		0.4392	xR	-	0.3990	xG	-	0.0402	xB	+	128

Table 151: 709 colorimetry / integer matrix / digital range

RGB to YCbCr integer matrix as implemented (1.0 <> 1024)											
Y	=		187	xR	+	629	xG	+	63	xB	+	16
Cb	=	-	103	xR	-	347	xG	+	450	xB	+	128
Cr	=		450	xR	-	409	xG	-	41	xB	+	128

Video matrix

Converting from RGB to YCbCr:

- The Y component is clipped between 1 and 254, after applying the matrix.
- The chroma range is -127 to +126, and can be optionally encoded in offset-binary format (+128, 1/254 range).

The next tables show the hardwired matrices used for the conversion (assuming offset binary chroma):

Table 152: 601 colorimetry / floating-point matrix / digital range

RGB to YCbCr reference floating-point matrix											
Y	=		0.2988	xR	+	0.5869	xG	+	0.1143	xB	+	
Cb	=	-	0.1729	xR	-	0.3389	xG	+	0.5107	xB	+	128
Cr	=		0.5107	xR	-	0.4277	xG	-	0.0830	xB	+	128

Table 153: 601 colorimetry / integer matrix / digital range

RGB to YCbCr integer matrix as implemented (1.0 <> 1024)											
Y	=		306	xR	+	601	xG	+	117	xB	+	
Cb	=	-	177	xR	-	347	xG	+	523	xB	+	128
Cr	=		523	xR	-	438	xG	-	85	xB	+	128

Table 154: 709 colorimetry / floating-point matrix / digital range

RGB to YCbCr reference floating-point matrix												
Y	=		0.2129	xR	+	0.7148	xG	+	0.0723	xB	+	
Cb	=	-	0.1172	xR	-	0.3945	xG	+	0.5117	xB	+	128
Cr	=		0.5117	xR	-	0.4648	xG	-	0.0469	xB	+	128

Table 155: 709 colorimetry / integer matrix / digital range

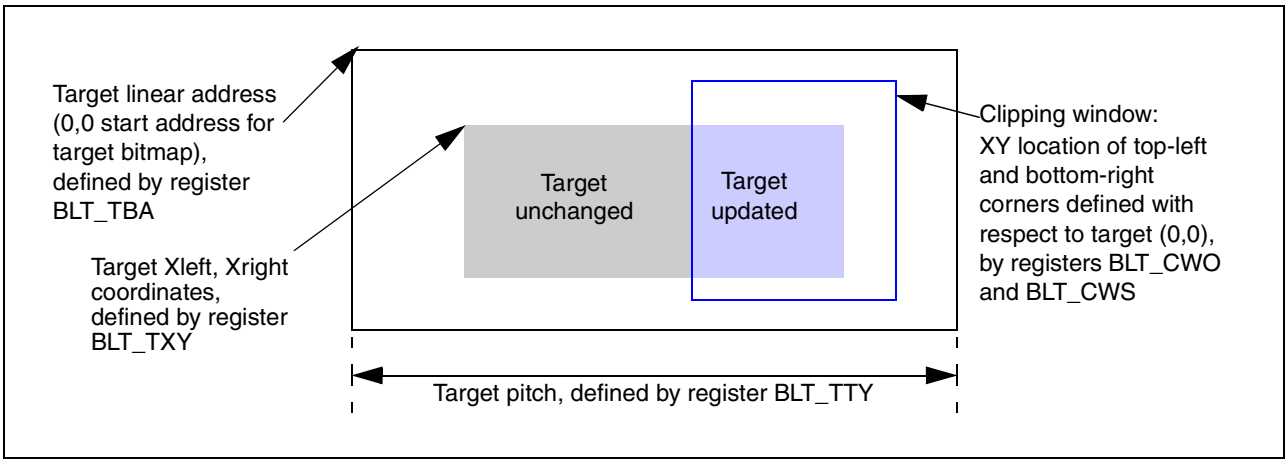
RGB to YCbCr integer matrix as implemented (1.0 <> 1024)												
Y	=		218	xR	+	732	xG	+	74	xB	+	
Cb	=	-	120	xR	-	404	xG	+	524	xB	+	128
Cr	=		524	xR	-	476	xG	-	48	xB	+	128

47.5.8 Rectangular clipping

Each write access to the target plane can be enabled on a pixel-per-pixel basis, with respect to a rectangular window.

The base address format of the 2D-graphics engine is in x-y coordinates. Hardware clipping uses four registers to define the rectangular clipping area. The figure below shows a block transfer (BLT) operation inside a buffer, using a hardware clipping window. The buffer is defined using a linear start address. The BLT area and the clipping window are defined by x-y coordinates. The block transfer function is only valid inside the clipping window.

Figure 123: Hardware clipping window



Configuration bit BLT_CWO.INTNL inverses the clipping window; the target area is updated outside the clipping window and is protected inside the clipping window.

XDO and YDO (register BLT_CWO) define the window start with respect to the target base address.

XDS and YDS (BLT_CWS) define the window stop with respect to the target base address.

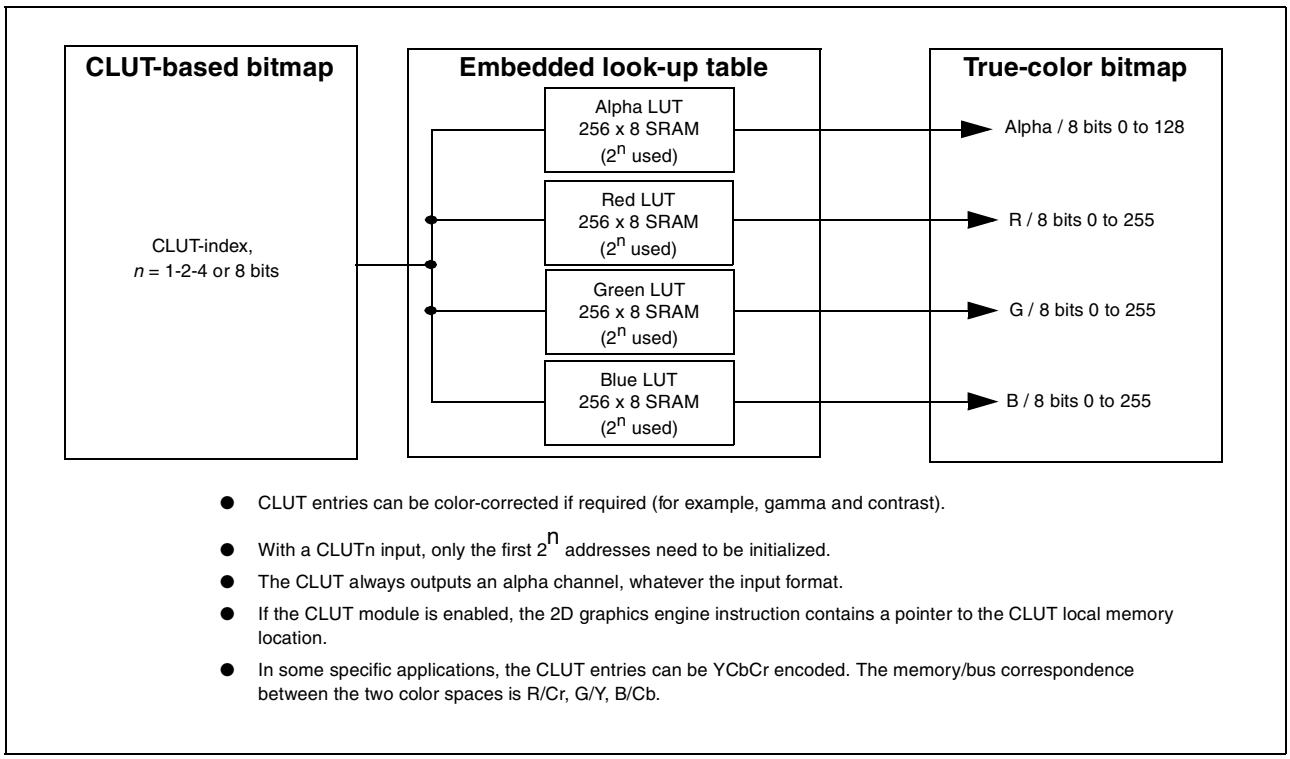
47.5.9 CLUT and color correction on source 2

The internal 256 x 32 LUT can perform color expansion, color reduction and color correction.

Color expansion

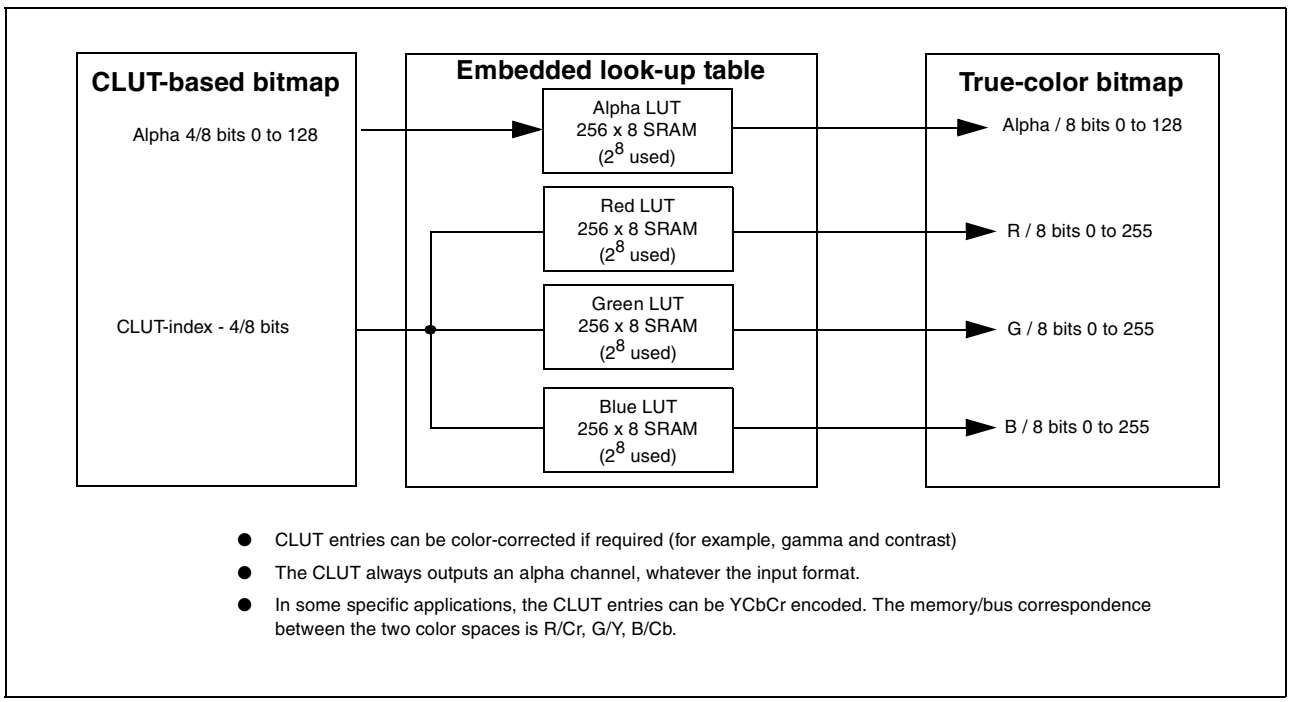
During color expansion a CLUT-based bitmap is transformed into a true-color bitmap, using the embedded look-up table (256x32 SRAM). The following two figures illustrate a CLUT-to-RGB conversion and a CLUT88/44-to-RGB conversion.

Figure 124: ACLUTn to (A)RGB conversion



- CLUT entries can be color-corrected if required (for example, gamma and contrast).
- With a CLUTn input, only the first 2^n addresses need to be initialized.
- The CLUT always outputs an alpha channel, whatever the input format.
- If the CLUT module is enabled, the 2D graphics engine instruction contains a pointer to the CLUT local memory location.
- In some specific applications, the CLUT entries can be YCbCr encoded. The memory/bus correspondence between the two color spaces is R/Cr, G/Y, B/Cb.

Figure 125: ACLUT88/44 to ARGB conversion



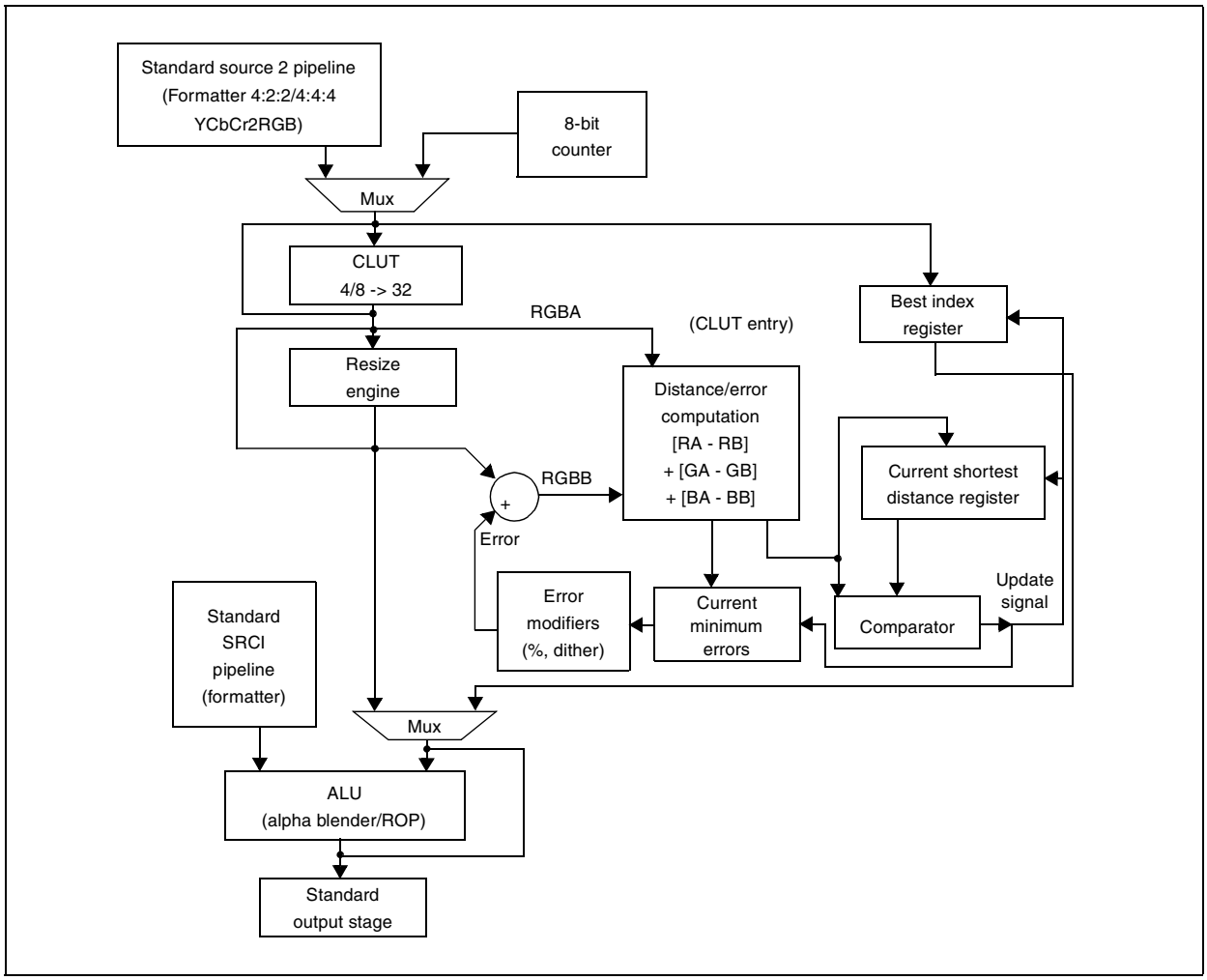
- CLUT entries can be color-corrected if required (for example, gamma and contrast)
- The CLUT always outputs an alpha channel, whatever the input format.
- In some specific applications, the CLUT entries can be YCbCr encoded. The memory/bus correspondence between the two color spaces is R/Cr, G/Y, B/Cb.

Color reduction

The color reduction engine performs the following tasks:

- Converts true-color RGB bitmaps into CLUT*n* bitmaps.
- Resizes a CLUT*n* bitmap.
- Converts ACLUTn_1 bitmaps into ACLUTn_2 bitmaps in two stages, as follows:
  - The ACLUTn_1 bitmap is expanded to an intermediate RGB bitmap (optionally resized), using its CLUT,
  - The intermediate RGB bitmap is mapped to the CLUT of the ACLUTn_2 bitmap.

Figure 126: Color reduction engine architecture



The color reduction mechanism can be summarized as follows (see also Figure 126):

- Best match strategy: the whole CLUT is scanned for each new RGB pixel from source 2. A distance is computed for each CLUT entry and the current source 2 pixel. The entry with the shortest distance is considered as the best possible match, and its index is used to represent the RGB pixel in the target CLUT-based format.
- Adaptive mode: a mono-dimensional error diffusion algorithm weighting can be added to the best match. All or part of the error made can be diffused on the next pixel (X + 1, same Y), depending on the LSBs of the address in the target bitmap (*x*LSB, *y*LSB). The distance is evaluated using the sum of the absolute values of the differences on each component:

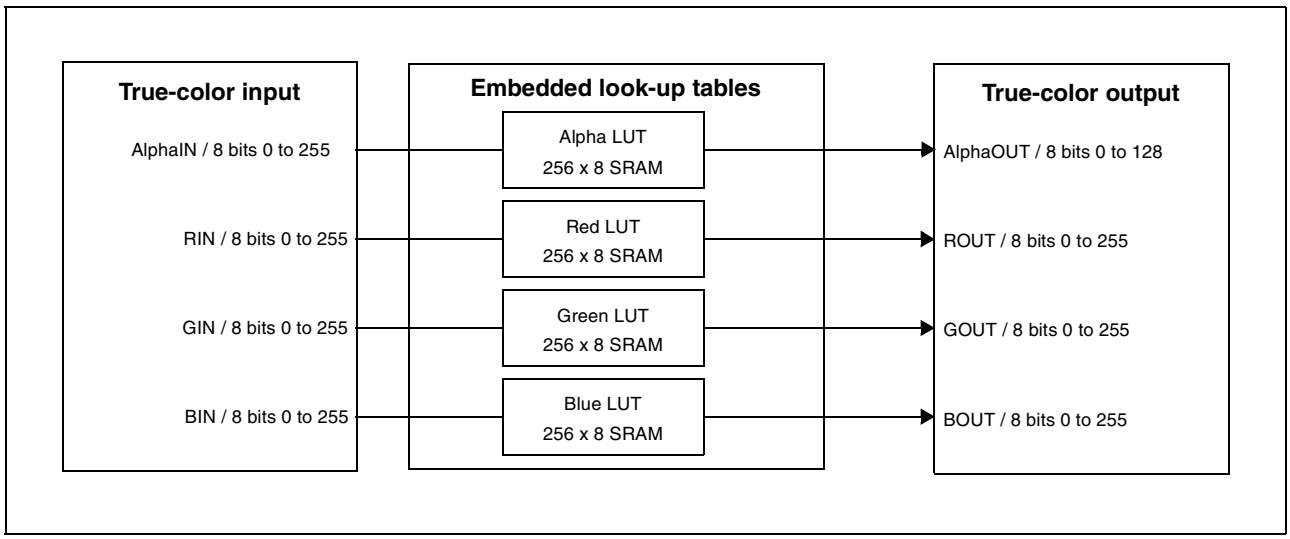
  distance = abs[R(scr2) - R(CLUT)] + abs[G(scr2) - G(CLUT)] + abs[B(scr2) - B(CLUT)]

Note: The best match search can be made in the YCbCr domain: the input color space converter must be used on source 2, and the target CLUT converted from RGB to YCbCr entries.

Color correction

The CLUT can be used as four independent 256 x 8 LUTs, applying any transformation on the input components. This can be used for gamma correction, contrast adjustment, gain, offset. Used in conjunction with the 2D-graphics engine color space converters (at the input and the output), the correction can also be made in the YCbCr space (for example, color effect, conversion to a gray-scale bitmap).

Figure 127: Color correction for true-color inputs



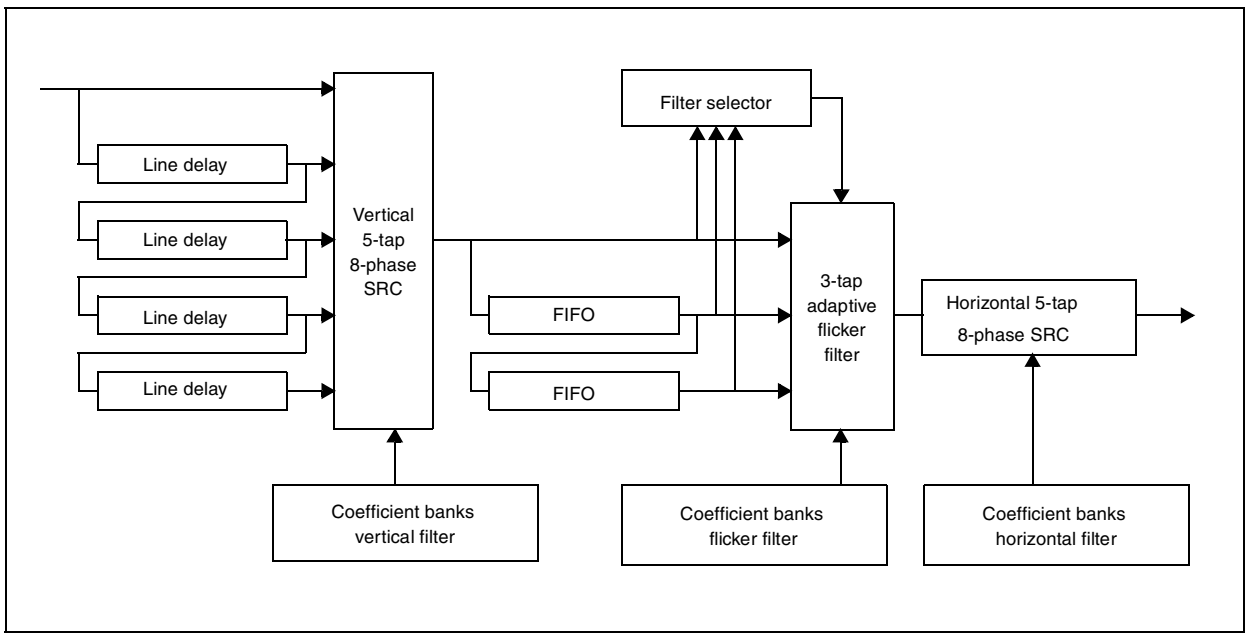
Register BLT_CML is a pointer to the local memory CLUT address.

CLUT_UPDATE enables CLUT refresh from memory, CLUT_MODE sets the CLUT mode (color expansion, color correction, color reduction), and CLUT_ERRDIFF sets the working mode of the diffusion weight for CLUT color reduction mode (register BLT_CCO).

47.5.10 2D resize engine

The memory-to-memory 2D resize engine includes a vertical SRC, a horizontal SRC and a context-sensitive flicker filter. It is illustrated in the figure below.

Figure 128: 2D resize engine architecture



In a single pass, the 2D graphics engine can output a flicker-free downscaled/upscaled picture. The operator is neither limited by source size nor destination size. It contains sufficient memory to apply a 5 x 5 filter. The source is automatically split into vertical stripes, the maximum width of which is 128 pixels.

Note: This is a parallel four-component operator: the alpha channel is processed in the same way as the RGB components.

Typically, the 2D resize engine application is used to generate high-quality anti-aliased fonts: a 1 bpp source 2 bitmap of any oversampled font character is first expanded with the CLUT operator, then downsized with high-quality filtering.

There are two ways to resize CLUT-based bitmaps:

- Use a two-pass 2D graphics engine operation, as described in *Color reduction* *on page 444*.
- Use a single-pass resize engine and disable the filters. This uses a skip/repeat technique, and can be used for data that represent indexes, not true-color pixels. This method usually gives poor quality graphics.

OFFSET_HSRC (register BLT_RZC) sets the initial phase of the filter for the 1st pixel of the line.

HSRC_INC (BLT_RSF) sets the horizontal scaling factor. 2DHF_MODE (BLT_RZC) enables the horizontal filter and sets the part of the ARGB bus where it is available. Register BLT_HFP is a pointer to the filter coefficients in the local memory.

OFFSET_VSRC (BLT_RZC) sets the initial phase of the filter for the 1st pixel of the column.

VSRC_INC (BLT_RSF) sets the vertical scaling factor. 2DVF_MODE (BLT_RZC) enables the vertical filter and sets the part of ARGB bus where it is available. Register BLT_VFP is a pointer to the filter coefficients in the local memory

The CLUT pixmap may only be resized (work on index) if the horizontal and vertical filters are inactive, unless they are expanded to true color using the CLUT operator.

- The vertical and horizontal sample-rate-converter are based on a 5-tap polyphase filter (8 phases) followed by a decimator.
  - For correct filtering of the first two lines, the first source line is used three times
  - For correct filtering of the last two lines, the last source line is used three times.
  - For correct filtering of the first two pixels in a line, the first source pixel is used three times.
  - For correct filtering of the last two pixels in a line, the last source pixel is used three times.

  The initial phase can be programmed to achieve subpixel positioning (1/8 pixel).
- Coefficients:
  - Coefficient format: 8 bits S1.6 (1.0 is encoded as 64)
  - Coefficient sum = 1.0 = 2^6 = 64
  - SRC accumulator, increment has a 6.10 format.
- Total memory: 3 Kbyte (FIFO depth can be tuned to allow maximum efficiency on burst accesses to external memory, while keeping a reasonable amount of embedded SRAM).
- Total number of U8 x S8 multipliers: approximately 52 x (5 x 4 + 5 x 4 + 3 x 4)
- All the filter coefficients can be downloaded from the instruction block (stored in external memory). Generally, they are computed depending on the rescaling factor, but the cell can be used as a general-purpose 2D filter as well (for example to produce a blurring effect).

47.5.11 Flicker filter

Three vertically adjacent pixels are input to the flicker filter. The luminance steps between line *n* - 1 and line *n*, and line *n* and line *n* + 1 are estimated for each pixel.They are encoded by four bits (0 to 15). The absolute value of the maximum step is compared to three programmable threshold values, and the vertical 3 x 1 filter with the most appropriate response is selected:

0	≤step (line *n*)	≤threshold 1	filter 0 is selected - equivalent to no filter
threshold 1	< step (line *n*)	≤threshold 2	filter 1 is selected - soft filter
threshold 2	< step (line *n*)	≤threshold 3	filter 2 is selected - medium filter
threshold 3	< step (line *n*)	≤15	filter 3 is selected - strong filter

In test mode, the pixel color information can be replaced by a value to identify the filter (for example, filter 3 = white, filter 2 = green, filter 1 = red, filter 0 = black).

In a read/modify/write operation involving the alpha blender, the pixel alpha value can be divided by two before being used for blending on the source 2 window borders. This removes flicker on edge lines, as the three-tap vertical flicker-filter cannot operate correctly on edge lines. Horizontally, a similar mechanism is provided for the first and last pixels of each line to smooth vertical edges of the target window.

FF_MODE (BLT_RZC) selects:

- the filter 0 only mode,
- the adaptive mode, that follows an estimation of the luma difference,
- the test mode to evaluate the threshold selected in the adaptive mode.

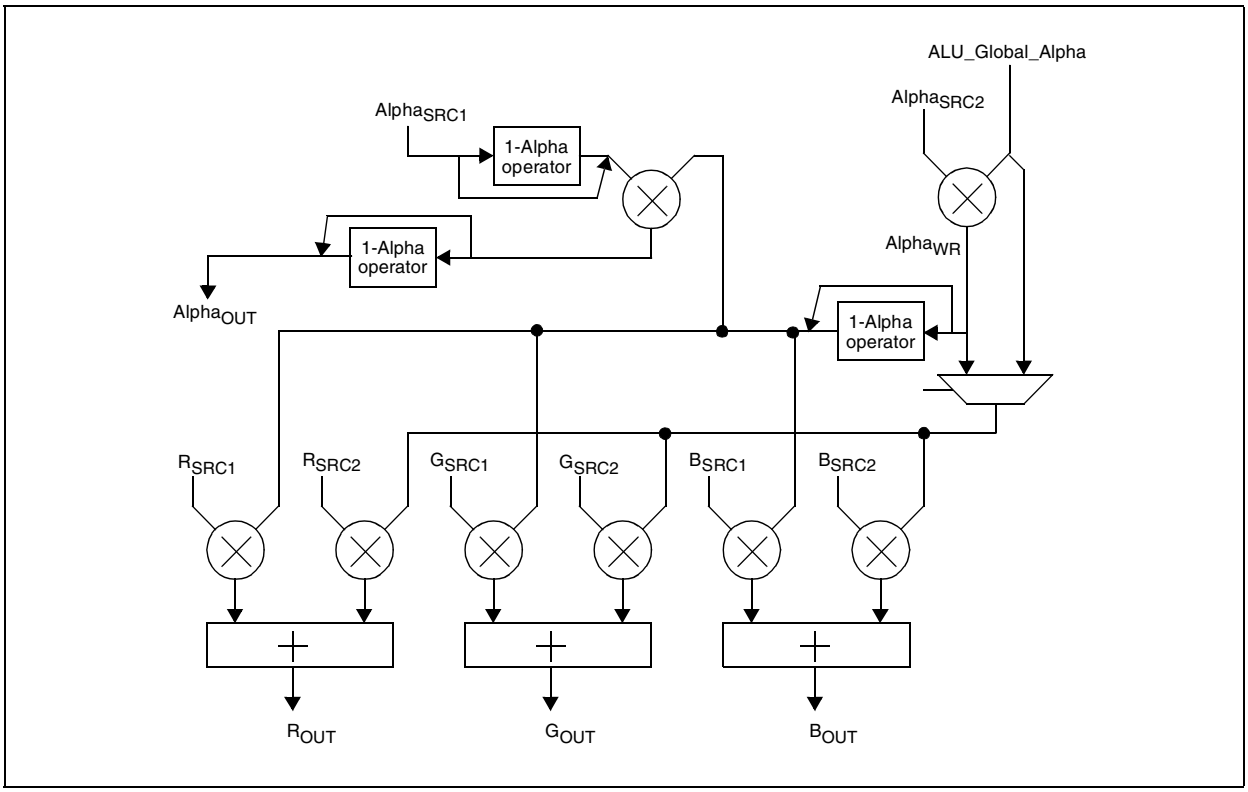
Registers BLT_FF0, BLT_FF1, BLT_FF2, BLT_FF3 define the four filters in terms of their (*n* - 1), *n*, and (*n* + 1) coefficients, and the thresholds of the first three filters.

47.5.12 ALU (alpha blending and boolean operator)

Alpha blending

The alpha blender interpolates colors between source 1 and source 2 (see Figure 129).

Figure 129: Alpha blender architecture



Each source may have its own per-pixel alpha component. Source 2 is always blended on top of source 1; source 1 is usually the source for the background plane. Sources 1 and 2 are blended using their own alpha, combined using GALPHA_ROPID (register BLT_ACK).

Source 2 supports alpha-premultiplied and non–alpha-premultiplied color formats (premultiplied ARGB formats are in fact AR, AG and AB). The multiplexer is thus required to select between $Alpha_{WR}$ and ALU_Global_Alpha.

The blending equations are given below:

If source 2 is not pre-multiplied:

$$RGB_{OUT} = A_{SRC2} \times ALU\_Global\_Alpha \times RGB_{SRC2} + (1 - A_{SRC2} \times ALU\_Global\_Alpha) \times RGB_{SRC1}$$

If source 2 is pre-multiplied:

$$RGB_{OUT} = ALU\_Global\_Alpha \times RGB_{SRC2} + (1 - A_{SRC2} \times ALU\_Global\_Alpha) \times RGB_{SRC1}$$

In any case, the resulting translucency is:

$$\mathbf{(1 - Alpha_{OUT}) = (1 - A_{SRC2} \times ALU\_Global\_Alpha) \times (1 - A_{SRC1})}$$

This is equivalent to:

$$Alpha_{OUT} = A_{SRC2} \times ALU\_Global\_Alpha + A_{SRC1} \times (1 - A_{SRC2} \times ALU\_Global\_Alpha)$$

or $$Alpha_{OUT} = A_{SRC1} + A_{SRC2} \times ALU\_Global\_Alpha \times (1 - A_{SRC1})$$

The $Alpha_{OUT}$ component is written into the target bitmap; only the target format includes a per-pixel alpha component.

Note: When a target format has a per-pixel alpha component, the color components RGB_{out} are computed as pre-multiplied by this alpha value. The display pipeline (such as a GDP) should be aware of this when blending such a 2D graphics layer with the video layer.

A third source can be blended with sources 1 and 2, to create a single output. The third source must be a 1 bpp or 8 bpp bitmap mask. This three-source blending is implemented in two stages:

- The texture (or foreground picture) uses the source 1 pipeline, and the third source (bitmap mask) uses the source 2 pipeline. They are combined into an intermediate bitmap that must have a per-pixel alpha component, such as ARGB8888. In this case:

  $RGB_{OUT} = RGB_{SRC1}$

  $Alpha_{OUT} = A_{SRC1} - A_{SRC2} - ALU\_Global\_Alpha$

- The intermediate bitmap uses the source 2 pipeline and the background picture uses the source 1 pipeline. They are blended together.

Boolean operators

This is a 2 operand logical unit. The operator always applies to the whole pixel width, including the alpha component if there is one.The boolean operator performs the following 16 operations.

Table 156: Boolean operations

Mode name	Result	Memory cycle
CLEAR	*result* = all 0	Write
AND	*new* AND *old*	Read/modify/write
ANDrev	*new* AND (NOT *old*)	Read/modify/write
COPY	*new*	Write
ANDinvert	(NOT *new*) AND *old*	Read/modify/write
NOOP	*old*	none
XOR	*new* XOR *old*	Read/modify/write
OR	*new* OR *old*	Read/modify/write
NOR	(NOT *new*) AND (NOT *old*)	Read/modify/write
EQUIV	(NOT *new*) XOR *old*	Read/modify/write
INVERT	(NOT *old*)	Read/modify/write
ORreverse	*new* OR (NOT *old*)	Read/modify/write
COPYinv	(NOT *new*)	Write
ORinvert	(NOT *new*) OR *old*	Read/modify/write
NAND	(NOT *new*) OR (NOT *old*)	Read/modify/write
SET	all 1	Write

In boolean operations, a third source can be blended with sources 1 and 2, to create a single output. The third source must be a 1 bpp bitmap mask. Combining is performed on a pixel-by-pixel basis. The raster operation is executed, depending on this 1-bit value.

Three-source boolean combining is implemented in two stages:

- The bitmap mask is combined with the texture, without destroying any existing texture per-pixel alpha. The result is stored in an intermediate bitmap, with an 8-bit per pixel fourth component (alpha or flag value).
  The texture uses source 1 pipeline, and the bitmap mask uses source 2 pipeline. If the bit value is 0, then the 8-bit fourth component is set to 255. If the bit value is 1, then the 8-bit fourth component is set to the alpha value of source 1 (128 if source 1 has no alpha channel). The color components remain unchanged.
- The intermediate bitmap uses the source 2 pipeline and the background picture uses the source 1 pipeline. These are combined. If the fourth component is 255, then the logical operation is ignored and the output directly corresponds to source 1. If the fourth component is not 255, then a standard logical operator is applied.

Bypass and concatenation modes are available. In bypass mode, either source 1 or source 2 may be bypassed. Concatenation mode allows 4:2:0 source 1/source 2 YCbCr concatenation.

MODE (BLT_ACK) specifies 5 operation modes: single source bypass, logical operation, blending operation, clipmask mode for three-source management in two passes (two sources and one mask), and concatenation mode for macro-block formats.

- Single source bypass: bypass source 1 or source 2. For example, if source 2 is bypassed, and source 1 is set, only source 2 affects the target and takes data from the local memory.
- Logical operation: this operates on a single source (1 or 2) and on dual sources. GALPHA_ROPID (BLT_ACK) is used to program the logical operation code.
- Blending operation: this operates only on dual sources. GALPHA_ROPID (BLT_ACK) is used to program the global alpha blending value. It is also possible to activate a horizontal and/or vertical alpha border on 1 pixel width with bits AB_H and AB_V (BLT_RZC).
- Three-source clipmask management in blend or logical mode. This specific mode is used in a two-pass blitter operation to blend two sources without a uniform global alpha value or logical operator. The pixel accuracy is programmed inside a third source that plays the role of the mask. In logical mode, only the A1 format is available for the mask plane; in blending mode, A1 and A8 formats are available.
- The concatenation mode operates for YCbCr 4:2:0 macroblock formats for sources 1 and 2, when the luma from source 1 is not merged with the chroma inside source 2 (so it is usable only if LINE_RPT = 0 and COLOR_FORM = YCbCr420Mb in register BLT_S2TY).

47.5.13 Color key

The 2D graphics engine has two methods of color keying: source color key and destination color key. These methods are exclusive and are both used for sprite-based animation.

- In source color keying, the source color or range of source colors is not written to the destination area. It is provided by the source 2 pipeline, before or after the CLUT.
- In destination color keying, no color can be written onto the destination color or range of destination colors. The destination color or range of colors is provided by the source 1 pipeline.

Color keying can operate on true-color bitmaps or CLUT-based bitmaps, as follows:

- For RGB true-color bitmaps:
  - A range of colors can be specified for each component, with a minimum and a maximum value.
  - The current input color is compared to the range, on a component-by-component basis.
  - Each color comparator can be enabled or disabled, and the color match result set inside or outside the range.
  - A YCbCr input can be used with the following color correspondence: G/Y, Cb/B, Cr/R.
  - A CLUT-based bitmap can be used after it is expanded to true color (after the CLUT).
- For CLUT-based bitmaps:
  - The input is taken before the CLUT, if provided by source 2.
  - When operating on an index, the color key reacts when the input matches a unique or range of index values for the blue component, specified by registers BLT_KEY1/2.

CKEY_SEL (BLT_ACK) selects the color key mode: source color key on source 2 before CLUT, source color key on source 2 after CLUT (different from the previous case only if CLUTOP in BLT_INS is enabled), destination color key on source 1 (this supposes that the target matches with source 1: read/modify/write).

ACK_CKEY (BLT_ACK) defines for each component whether the color key is inside or outside the range, or always matches.

Register BLT_KEY1 defines the minimum range for each component on 8 bits. BLT_KEY2 defines the maximum range for each component on 8 bits.

The color key match C-equation is as follows:

```
COLOR_KEY_MATCH =
   [(Rin < Rmin) OR (Rin > Rmax)] AND (Routside = True) AND (EnableR = True)
OR (Rmin <= Rin <= Rmax) AND (Routside = False) AND (EnableR = True)
OR (EnableR = FALSE)
AND
   [(Gin < Gmin) OR (Gin > Gmax)] AND (Goutside = True) AND (EnableG = True)
OR (Gmin <= Gin <= Gmax) AND (Goutside = False) AND (EnableG = True)
OR (EnableG = FALSE)
AND
   [(Bin < Bmin) OR (Bin > Bmax)] AND (Boutside = True) AND (EnableB = True)
OR (Bmin <= Bin <= Bmax) AND (Boutside = False) AND (EnableB = True)
OR (EnableB = FALSE)
```

47.5.14 Plane mask and output formatter on target

The output formatter converts the pixel bus into a 128-bit STBus format, according to the destination color format.

In true-color mode, a 2 x 2 dither mechanism rounds-off (for example, to target an RGB565 format with an internal 32-bit ARGB8888 pixel bus).

Note: The sub-byte format requires the blitter to operate in read/modify/write mode.

The plane mask is a register, whose width equals the number of bits-per-pixel in the destination format. Each bit in the target pixel word can be individually protected.

- If bit *n* in the plane mask register is 1, the corresponding *n*-weight bit in source 1 is updated according to the current 2D graphics engine operation.
- If bit *n* in the plane mask register is 0, the corresponding *n*-weight bit in source 1 remains unchanged whatever the 2D-graphics engine operator applied.

Register BLT_PMK provides the write-protect mask. BLT_TBA is the memory base address of the target bitmap (absolute (0,0) location, top-left pixel).

BLT_TTY is the specific properties of source 2: format, pitch, horizontal and vertical scan order, sub-byte ordering (for sub-byte formats), bit accuracy reduction mode. BLT_TXY is the coordinate that specifies the start of the input source 2 window with respect to the base address.

Register BLT_S1SZ is the size of the target window (equals source 1 size).

All formats are available except YCbCR420MB and YCbCR422MB.

47.6 Using the 4:2:x macroblock-based plane as a source

The 4:2:x plane can be used as a 2D graphics engine source. Because the 4:2:x plane is built from two buffers (one for luma and one for chroma), source 1 and source 2 input buffers must both be used to retrieve one 4:2:x plane. Consequently, 2D graphics engine operations with a 4:2:x input must be a one-source blitter. Blending or boolean combinations, involving a 4:2:x macroblock source, require a two-pass blitter operation.

The following two methods use the 4:2:x plane as a source:

- 4:2:0 chroma vertical repeat mode (BLT_S2TY.LINE_RPT) or 4:2:2 macroblock formats:
  For 4:2:0, the 2D graphics engine duplicates the chroma lines to generate a 4:2:2 stream. Source 1 is used for the luma component, source 2 for chroma components. All the operators except ALU can be used.
  For 4:2:2/4:2:0 macroblock conversion, if using the resize operator on the entire 4:2:x picture:
  LINE_RPT (source 2 chroma line repeat) = enable for 4:2:0 macroblock format,
  BLT_S2SZ.SRC2_WIN_HEIGHT = luma line height,
  BLT_S2SZ.SRC2_WIN_WIDTH = luma pixel width,
  BLT_S1SZ.WIN_HEIGHT = new line height,
  BLT_S1SZ.WIN_WIDTH = new pixel width,
  BLT_ACK.MODE = bypass source 2,
  BLT_INS.2DRESCALE = enabled, if desired,
  BLT_RZC = required parameters,
  BLT_RSF = required parameters.
- Chroma vertical interpolation mode:
  Source 1 is used for luma, source 2 for chroma. Chroma information is 2D resized to upscale these components by two and add a half-line vertical translation correction. The ALU is configured to concatenate the unchanged luma, with the upscaled chroma components, to build 4:4:4 YCbCr pixels. These pixels can be optionally converted to RGB or YCbCr 4:2:2 raster, in the last 2D graphics engine processing stages.
  For 4:2:0 macroblock conversion using the resize operator for chroma upsampling:
  BLT_S2SZ.SRC2_WIN_HEIGHT = chroma line height = luma line height / 2,
  BLT_S2SZ.SRC2_WIN_WIDTH = luma pixel width,
  BLT_S1SZ.WIN_HEIGHT = luma line height,
  BLT_S1SZ.WIN_WIDTH = luma pixel width,
  BLT_ACK.MODE = logical operation OR or 4:2:0 source1/ source 2 YCbCr concatenation,
  BLT_INS.2DRESCALE = enable,
  BLT_RZC.2DHF_MODE = disable,
  BLT_RZC.2DVF_MODE = enable on color channel,
  BLT_RZC.VSRC_OFFSET = 0,
  BLT_RSF.VSRC_INC = 512.

A macro-block frame buffer may be accessed in frame mode, or in field mode, depending on bits BLT_S1TY.MB_FIELD and BLT_S2TY.MB_FIELD. In frame mode, all source lines are read, in field mode, the blitter reads every other line. This is very useful for handling interlaced video, in a blitter-based PIP application for instance.

47.7 Blitter XYL mode

47.7.1 XYL standard mode

Instead of scanning a rectangular area to generate XY addresses, the blitter can directly access random XY addresses, pre-computed by the CPU, and stored linearly in a memory buffer. Typical use: shapes (circles, ellipses), fonts, polygon lines and fill, run-length decoder.

The address of this buffer is given in BLT_XYP. The buffer contains a list of subinstructions. A subinstruction consists either of a single pixel processing, or of a horizontal line processing. For a given blitter XYL operation (that is, corresponding to a single blitter node), all the subinstructions must be of the same type.

Four subinstruction formats are supported in XYL standard mode:

- XY: single pixel access. Only XY coordinates are provided, the drawing color is given by the drawing context (color fill register BLT_S2CF).
- XYC: single pixel access. XY coordinates are provided, as well as the drawing color C that may change for each pixel (including the alpha component).
- XYL: horizontal line access. XY coordinates correspond the left pixel of the line, the L parameter provides the line length, in pixel unit.The drawing color is given by the drawing context (color fill register BLT_S2CF).
- XYLC: all the parameters are specified for any subinstruction, XY coordinates, line length and drawing color.

Whatever the subinstruction format, a subinstruction is always stored as 4-consecutive 32-bit words in memory:

Table 157: XYL subinstruction formats in memory

	XY	XYC	XYL	XYLC
1st 32-bit word	X	X	X	X
2nd 32-bit word	Y	Y	Y	Y
3rd 32-bit word	unused	unused	L	L
4th 32-bit word	unused	C	unused	C

The number of subinstructions involved in a blitter XYL instruction is provided in register BLT_XYL. An XYL operation can be considered as a normal blitter operation, except that instead of self-addressing rectangular areas, the blitter reads the pixel addresses in memory, in what is called the subinstructions buffer.

In standard XYL mode, the blitter performs a write-only access at the specified XY location in the target bitmap if the ALU is in bypass source 2 operating mode. Source 1 must be disabled.

The blitter performs read/modify/write, if source 1 is enabled and a combining operation (blending or logical boolean) has been programmed in the ALU. In this case, source 1 and the target must have the same bitmap context. The blitter first reads the current pixel at the XY location, using the source 1 resources for memory access. It then combines this pixel with the source 2 color fill register (BLT_S2CF), or the color provided in the subinstruction if XYC/XYLC. The result is written back to the same memory location.

When XYL or XYLC subinstructions are selected, this sequence reiterates for each pixel of the horizontal line (pipeline implementation).

The XYL blitter feature supports any color format with a 4:4:4 sampling structure; 4:2:2 and 4:2:0 modes are not supported.

The color (register BLT_S2CF or C field in the XYC or XYLC subinstruction) must be formatted (that is, programmed as internally represented at the source 2 formatter output, see Section 47.5.1 on page 433).

For example, in RGB565 mode, to write 0xABCD (R = 21, G = 30, B = 11), BLT_S2CF must be programmed with 0x80A8 7858 (an opaque alpha channel added plus expansion to 8 bits per component).

Table 158 shows the features available when working in XYL mode.

Table 158: Blitter XYL capabilities

Blitter feature	XYL mode availability	Comment
Color fill	Yes	Source 2 color fill only, with XY and XYL subinstruction format
4:2:X to 4:4:4	No	No meaning
4:4:4 to 4:2:X	No	No meaning
Programmable H/V scan	No	No meaning
Color space conversion	Yes	Output color space converter only
CLUT operator	No	
2D-resize	No	
Flicker reduction	No	
Color keying	Yes	
Rectangular clipping	Yes	
Plane-mask	Yes	
Clipmask	Yes	See description below

Clipmask feature when working in XYL mode:

The clipmask feature, when working in XYL mode, requires source 2 memory access to access the clipmask data. That means the whole source 2 address generator must be programmed to access this 1-bpp or 8-bpp data, and each register takes the following meaning:

- BLT_S2TY: clipmask bitmap parameters (such as pitch, format).
- BLT_S2BA: memory address for the (0,0) clipmask value.
- BLT_S2XY: clipmask positioning, that is, XY location of the (0,0) clipmask origin, with respect to the target (0,0) origin.
- BLT_S2SZ: unused.

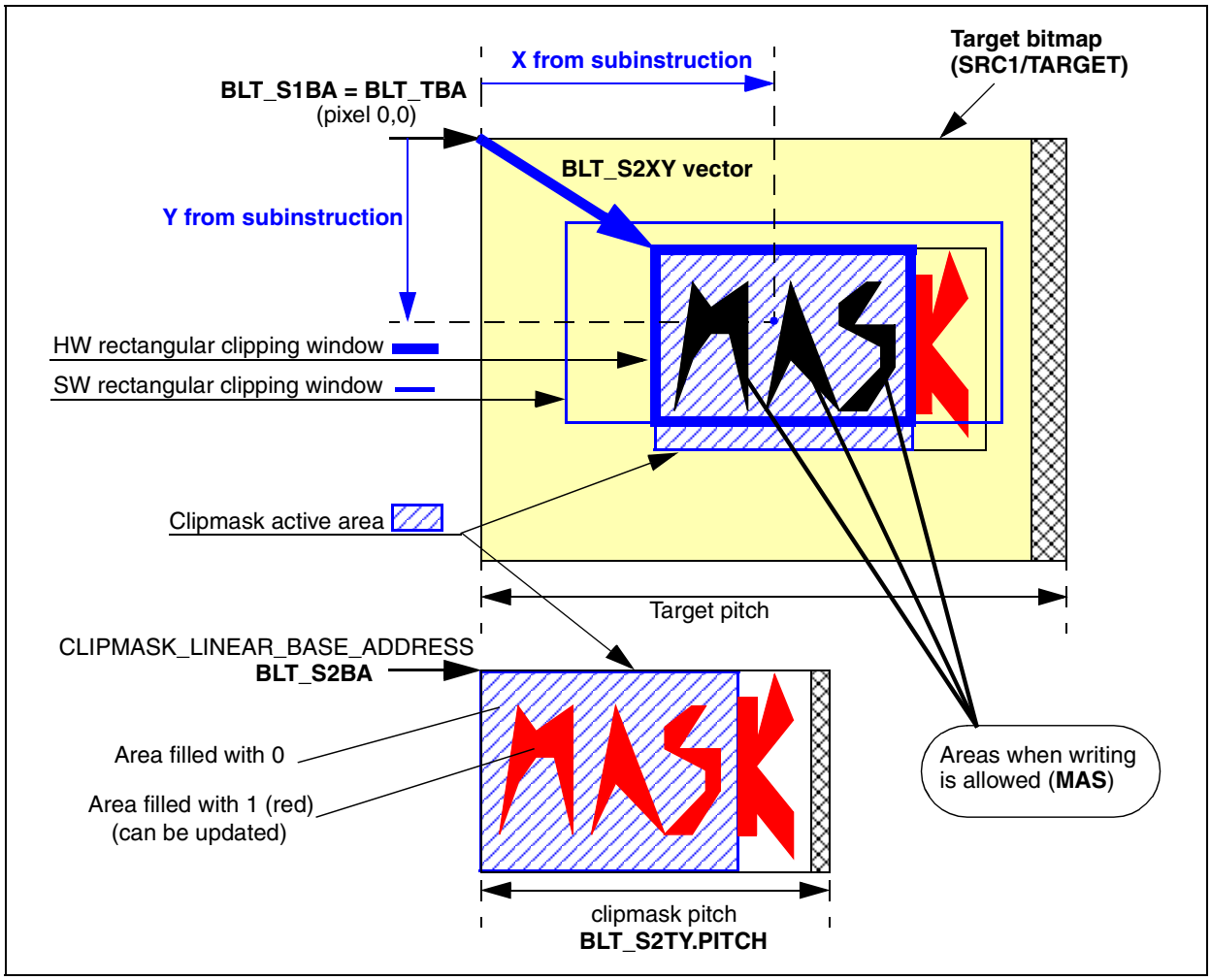
The ALU must be programmed for clipmask in XYL mode with either a logical operation, or blending with source 2 not premultiplied, or blending with source 2 premultiplied (MODE in register BLT_ACK).

In a MediaHighway drawing context, a clipmask validity window is defined, as well as a standard rectangular clipping window. For the blitter XYL to be compliant, the hardware rectangular clipping window must be programmed (BLT_CWO and BLT_CWS), with the intersection of the two software drawing context windows. Outside this resulting window, no write is performed.

The following hardware sequence occurs:

- Read the current target pixel at the (X,Y) location, in the target color bitmap, using source 1 address generator.
- Read the corresponding clipmask data XCOORD and YCOORD (BLT_S2XY) in the clipmask bitmap, using source 2 address generator.
- Combine the source 1 pixel with the BLT_S2CF color in the ALU, according to the ALU operation code, and depending on the current clipmask data. See Section 47.5.12 on page 448 for details about ALU behavior with respect to clipmask data (clipmask is equivalent to bitmap mask).
- If this x, y location is valid (inside the clipping window), then write the resulting pixel at the (X, Y) location, in the target color bitmap, using the target address generator.

Figure 130: Example of an XY access with clipmask (logical ROP)



47.7.2 How to use XYL mode

Set bit XYL (register **BLT_INS**) to program the blitter in XY mode.

The byte memory location for the XYL subinstruction buffer is set in register BLT_XYP. The XYL subinstruction size is always 128 bits, whatever the format of these subinstructions: XY, XYL, XYC, or XYLC. The subinstruction format is set using SUBINS (BLT_XYL), and their number is given by NUM_SUBINS.

The different ALU operations set by MODE (BLT_ACK) can then be performed. Not all ALU operations are allowed in XYL mode. The following ones are allowed and are described below:

0001: logical operation

0010: Blending mode, source 2 unweighted

0011: Blending mode, source 2 weighted

0111: bypass source 2

1001: Clipmask in XY mode, logical op

1010: Clipmask in XY mode, source 2 unweighted

1011: Clipmask in XY mode, source 2 weighted

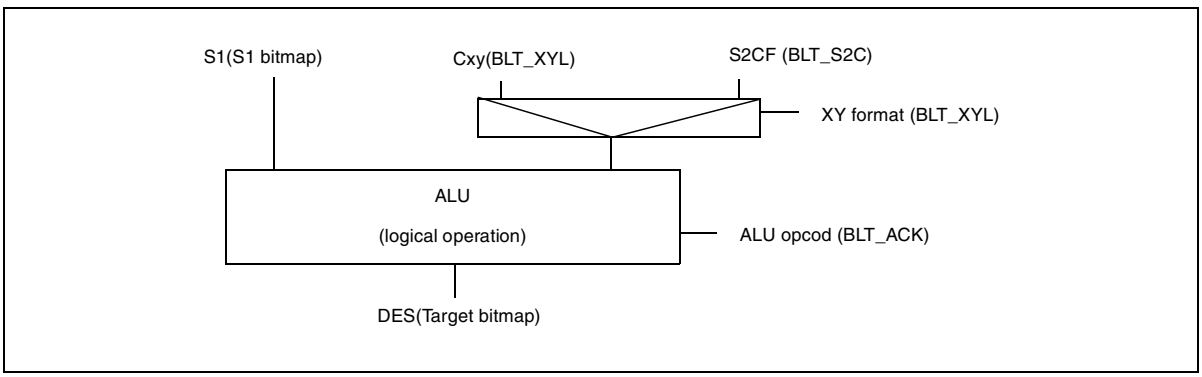
0001, 0010, 0011: logical operation, and blend

These operations are performed between two sources. Source 1 accesses the destination area using BLT_S1BA and BLT_S1TY.

If a color is defined inside the XY subinstruction (XYuC or XYLC), this color is used. Otherwise the color set in BLT_S2CF is used (source 2 color fill register). The ALU performs the operation between source 1, unmodified, and the static color. Source 2 weight is handled using the normal ALU datapath. The resulting composition is sent back to memory at its initial position using BLT_TBA and BLT_TTY, which are equal to BLT_S1BA and BLT_S1TY.

If these two registers (BLT_TBA and BLT_TTY) are not set as equal to their S1 equivalent (BLT_S1BA and BLT_S1TY), the XYL result can be stored in another memory destination.

Figure 131: Datapath in ROP or blend (weighted/unweighted)



0111: bypass source 2

In this mode, source 1 is not accessed, as no operation is performed with the background. It only writes the new color at the destination location. This is the only mode where read/modify/write are not performed for data in the memory. Only a write is performed.

1001: clipmask pass in XY mode (ROP)

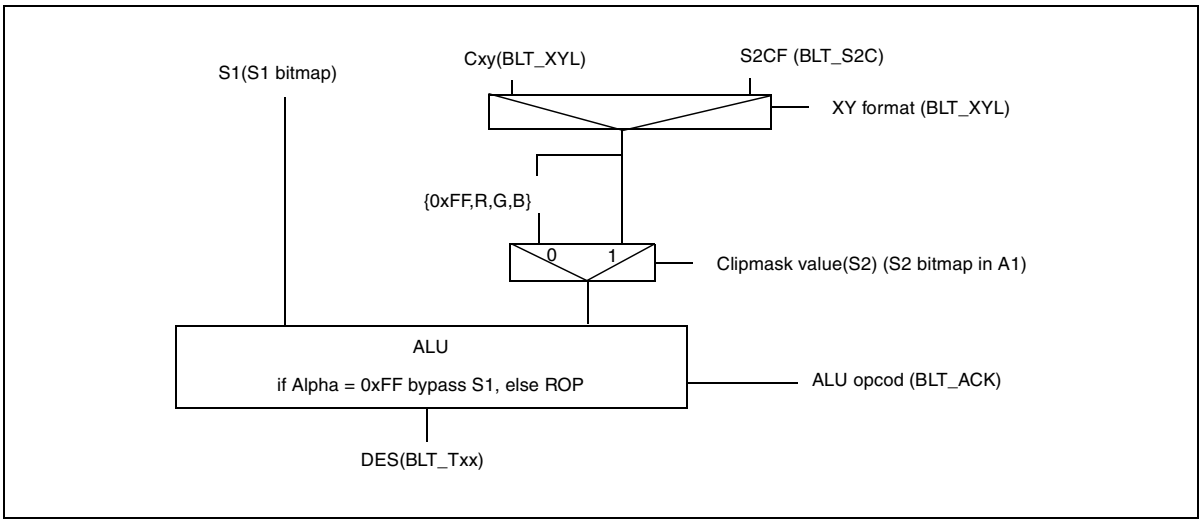
The clipmask is accessed through the source 2 interface. It means source 2 has to be fully set to access the clipmask bitmap (BLT_S2BA, BLT_S2TY and BLT_S2SZ).

BLT_S2CF remains the second operand for ROP if no color is defined in the XYLC subinstruction.

The source 2 bitmap is used as clipmask. In ROP (logical operation) mode, the clipmask (source 2 bitmap) can be used only in A1 format. When A1 = 0, the background is passed. This means that the source 2 alpha is replaced by 0xFF, so that the existing datapath (Aa in clipmask pass 2) is not modified.

When A1 = 1 the operation is executed normally.

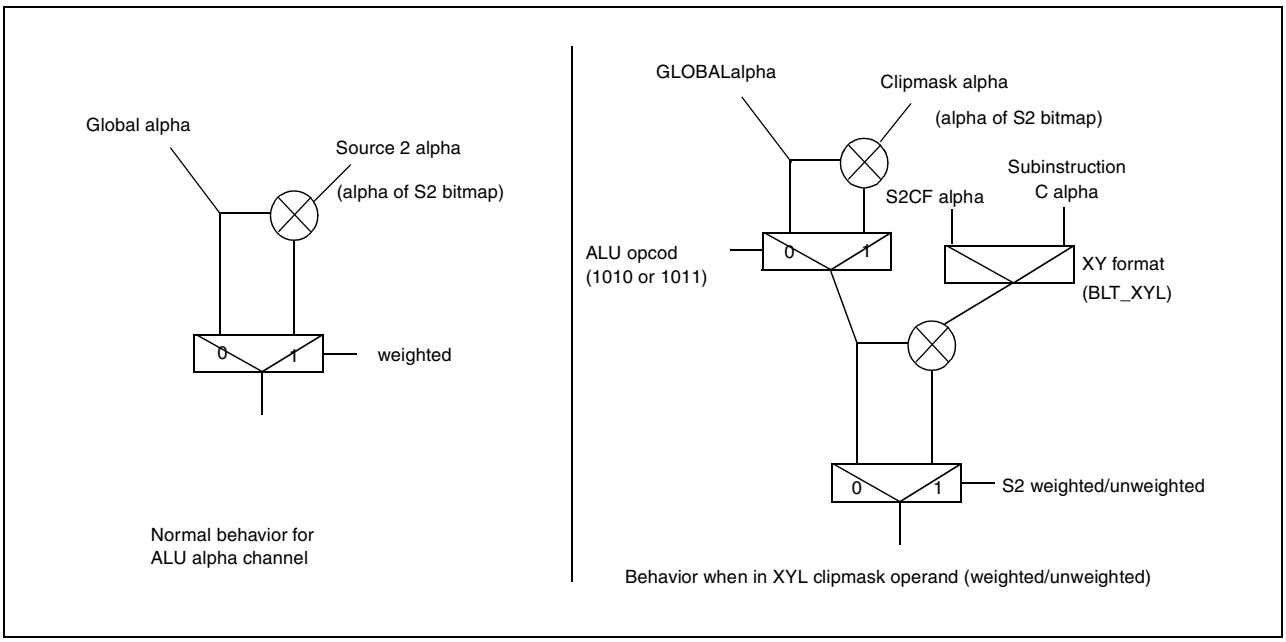
Figure 132: Clipmask ROP in XYL format



1010/1011: Clipmask pass in XY mode (blend, source 2 unweighted/weighted)

The clipmask is again accessed through the source 2 interface, as in the previous mode, but this time it can be either A1 or A8 format. Only the alpha channel is modified, and the computing of alpha changes as shown in Figure 133.

Figure 133: Destination alpha computation

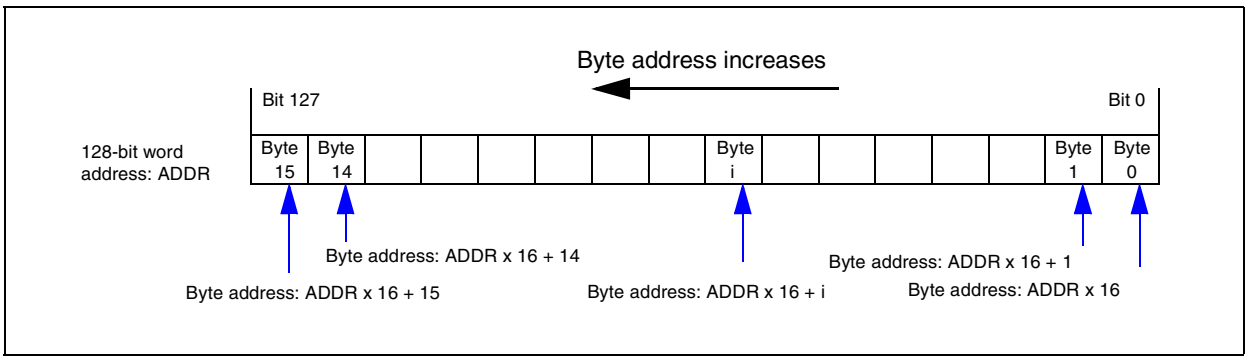


47.8 Local memory storage of supported graphics formats

47.8.1 General aspects

Memory storage (128-bit word) for all graphics data is little endian.

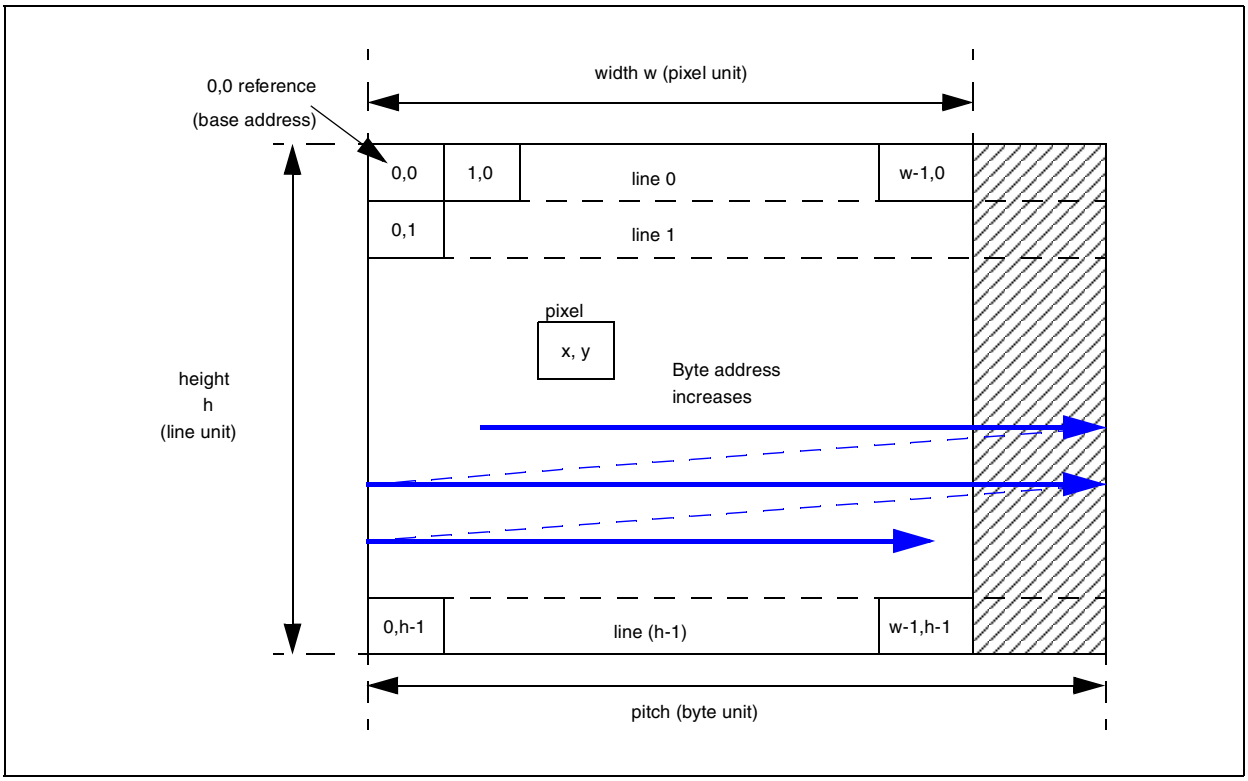
Figure 134: Little endian convention for a 128-bit word



For any graphics format (except 4:2:0 planes), a bitmap can always be considered as a rectangular area, with a width in pixel units, and a height in line units.

The (0, 0) origin is always the upper-left corner of the bitmap. Any pixel can be internally addressed using an (x, y) model, with the (0, 0) reference pixel being the top-left location of the rectangular area.

Figure 135: 2D-graphics frame-based raster storage



When stored in the local memory, pixel (0,0) is stored at the lowest byte address. The scan order is from left to right horizontally, and from top to bottom vertically. For each line (constant y coordinate), the pixel address increases when the x coordinate increments.

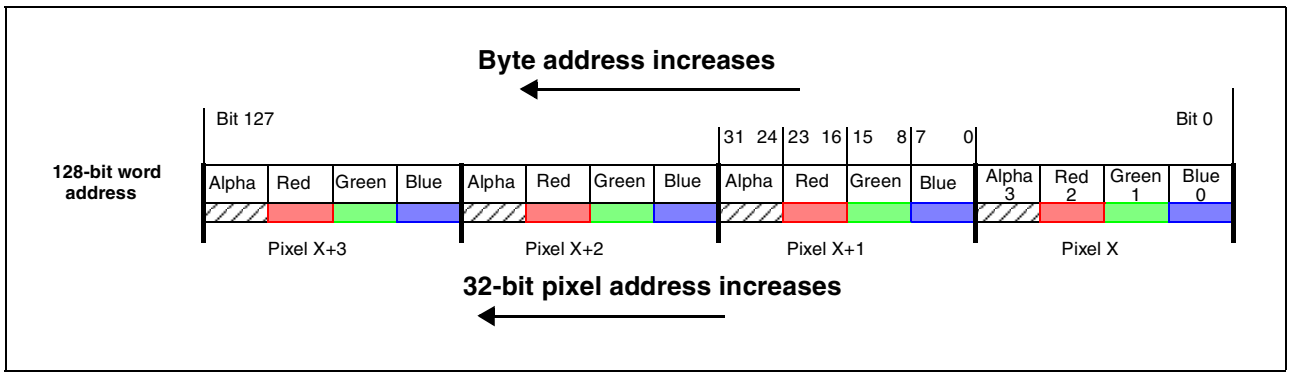
The pitch is the distance in bytes between any pair of vertically adjacent pixels (same x coordinate, y/y+1). The pitch is at least the width multiplied by the number of bytes per pixel, but

can be greater (for instance, byte stuffing for memory alignment considerations, or global bitmaps containing elementary bitmaps, side-by-side).

The specific organization within a 128-bit word is described below for each format.

47.8.2 32-bit per pixel format - true color

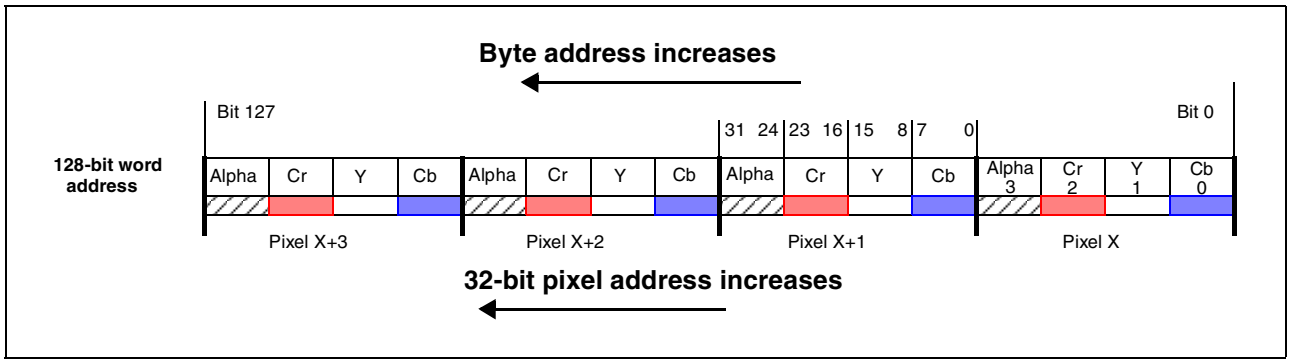
Figure 136: ARGB8888 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

Figure 137: AYCbCr8888 alignment in a 128-bit word

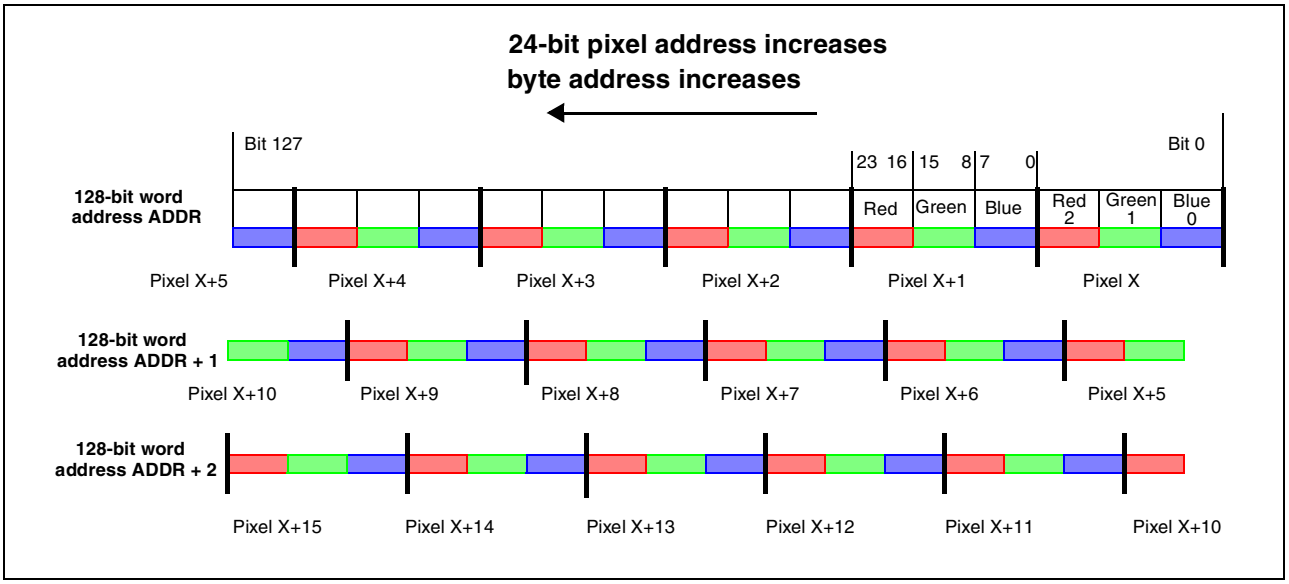


Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

47.8.3 24-bit per pixel format - true color

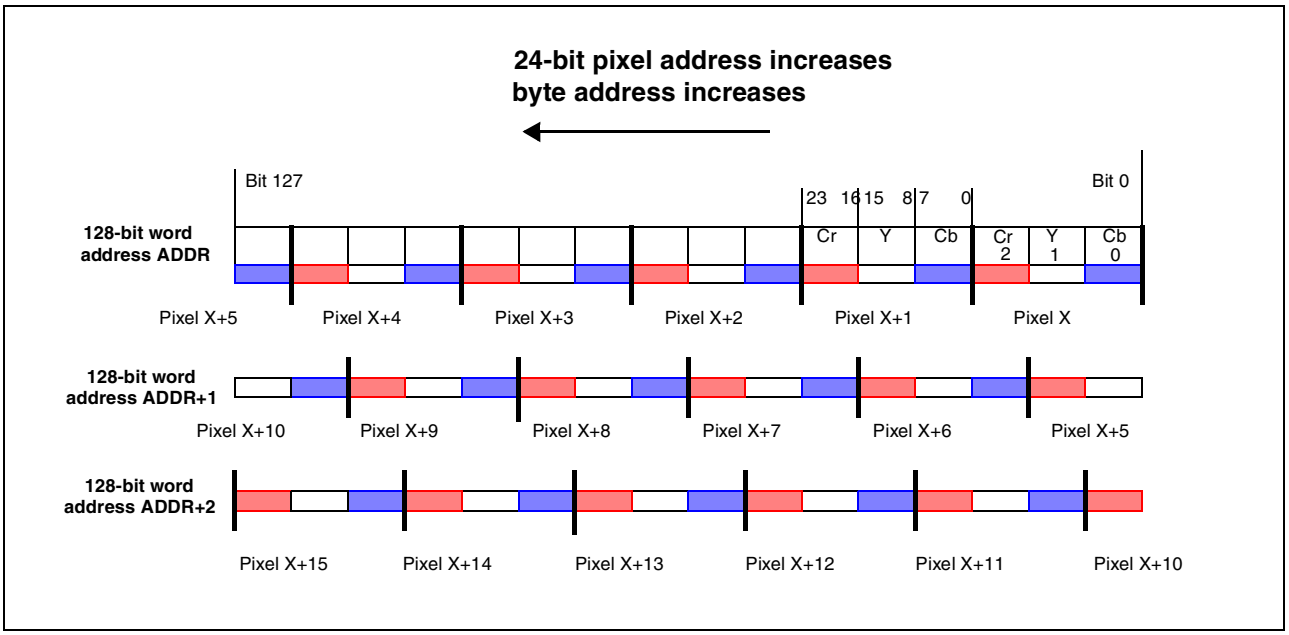
Figure 138: RGB888 alignment in a 128-bit word (3-word period/16-pixel period)



Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

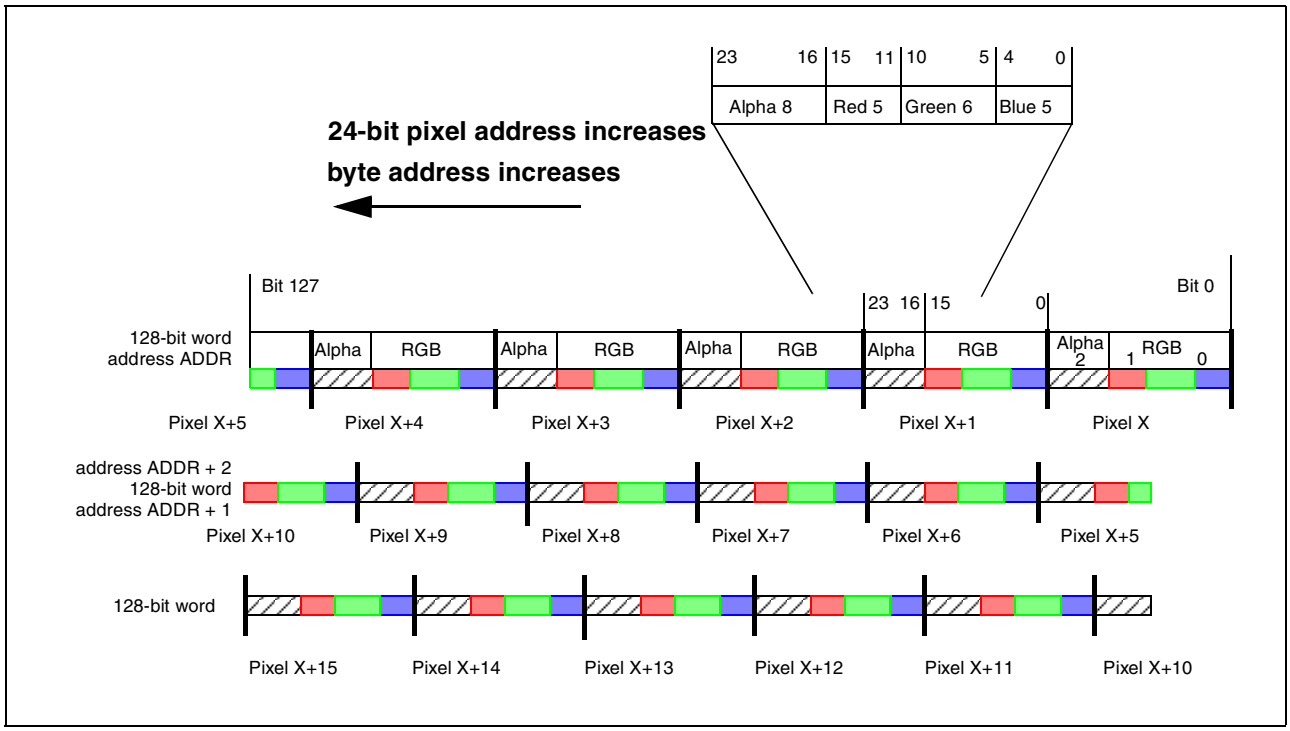
Figure 139: YCbCr888 alignment in a 128-bit word (3-word period/16-pixel period)



Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

Figure 140: ARGB8565 alignment in a 128-bit word (3-word period/16-pixel period)

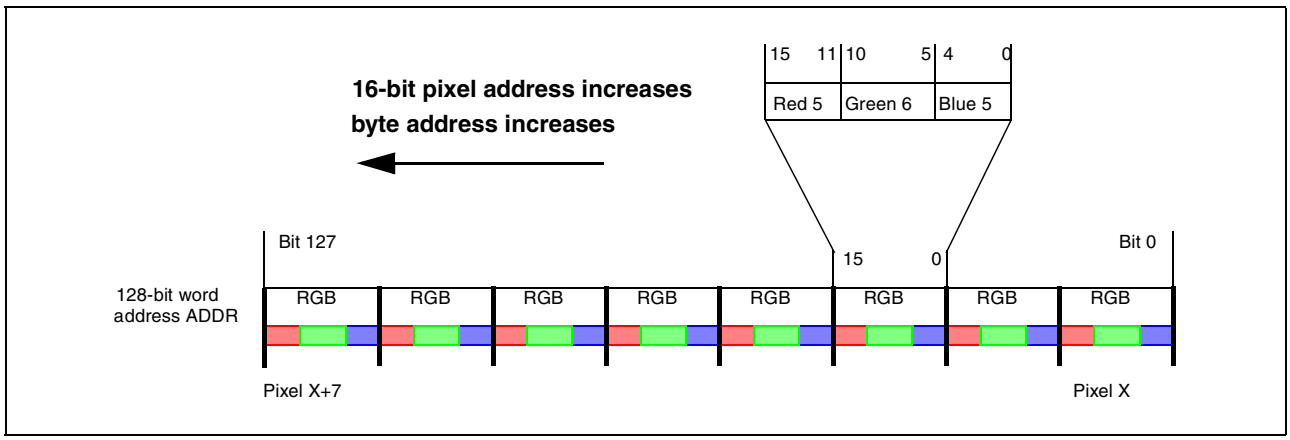


Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

47.8.4 16-bit per pixel format - true color

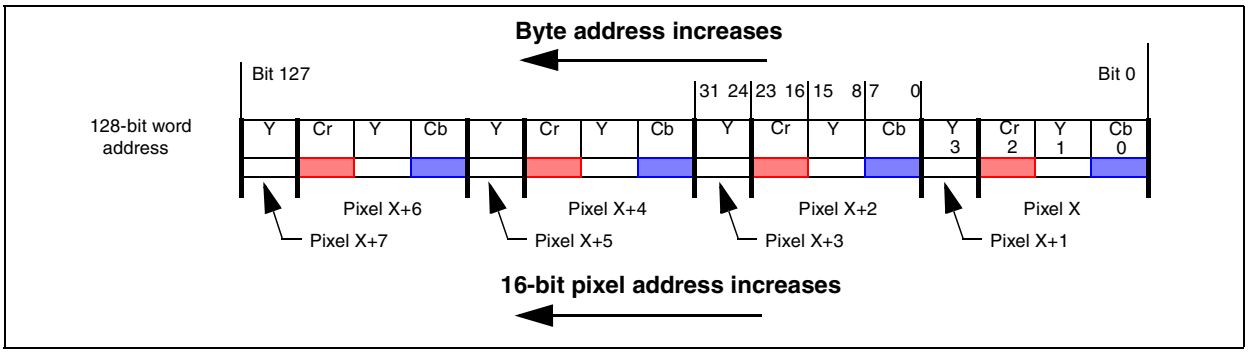
Figure 141: RGB565 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be aligned on a 16-bit word address boundary.

2 The pitch value must be a multiple of two bytes.

Figure 142: YCbCr 4:2:2 (16 bpp)

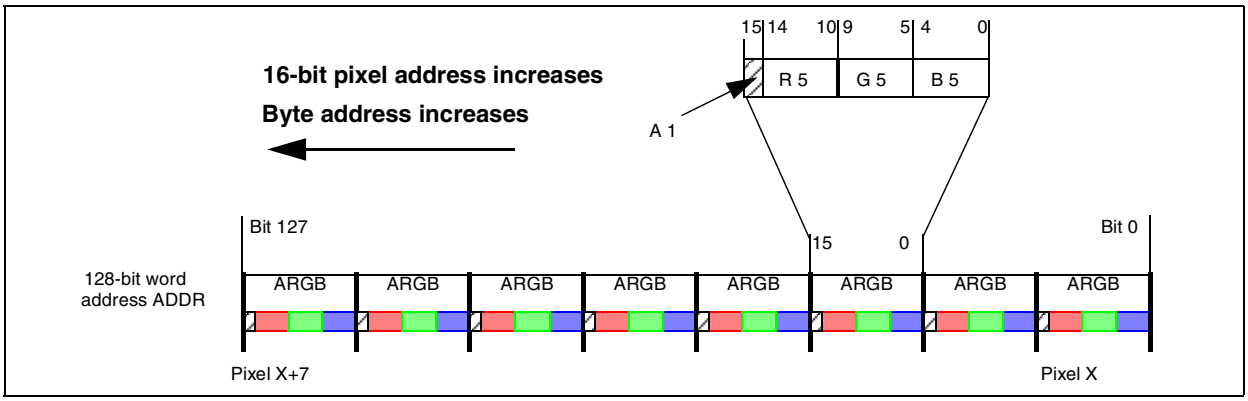


Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

3 X is even on this diagram.

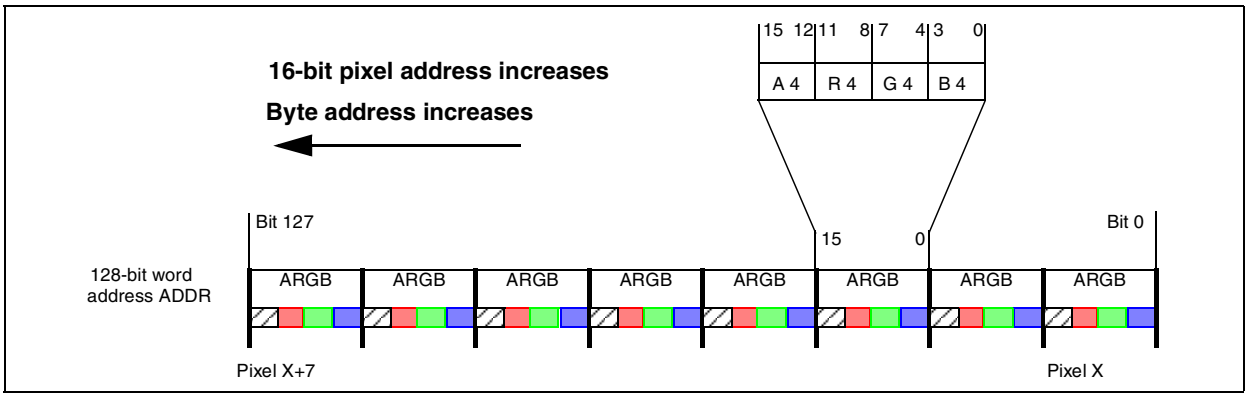
Figure 143: ARGB1555 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be aligned on a 16-bit word address boundary.

2 The pitch value must be a multiple of two bytes.

Figure 144: ARGB4444 alignment in a 128-bit word

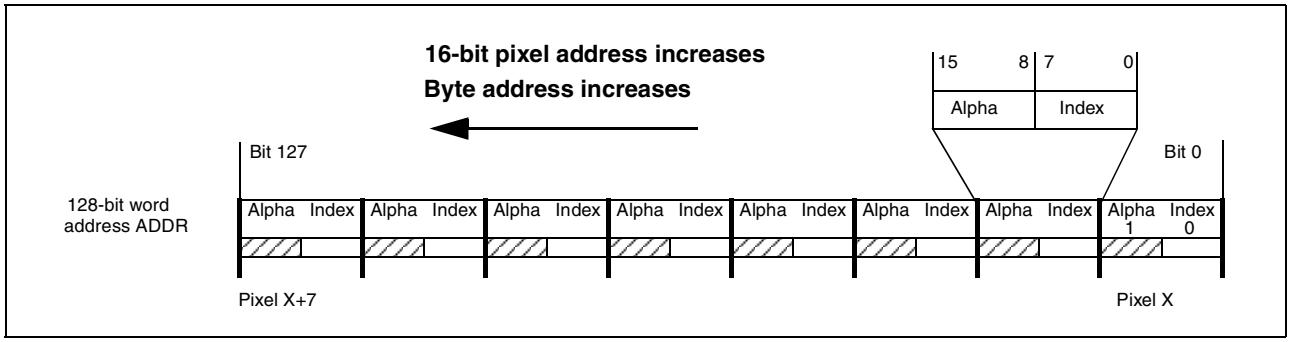


Note: 1 The memory address for pixel (0,0) must be aligned on a 16-bit word address boundary.

2 The pitch value must be a multiple of two bytes.

47.8.5 16-bit per pixel format - CLUT based

Figure 145: ACLUT88 alignment in a 128-bit word

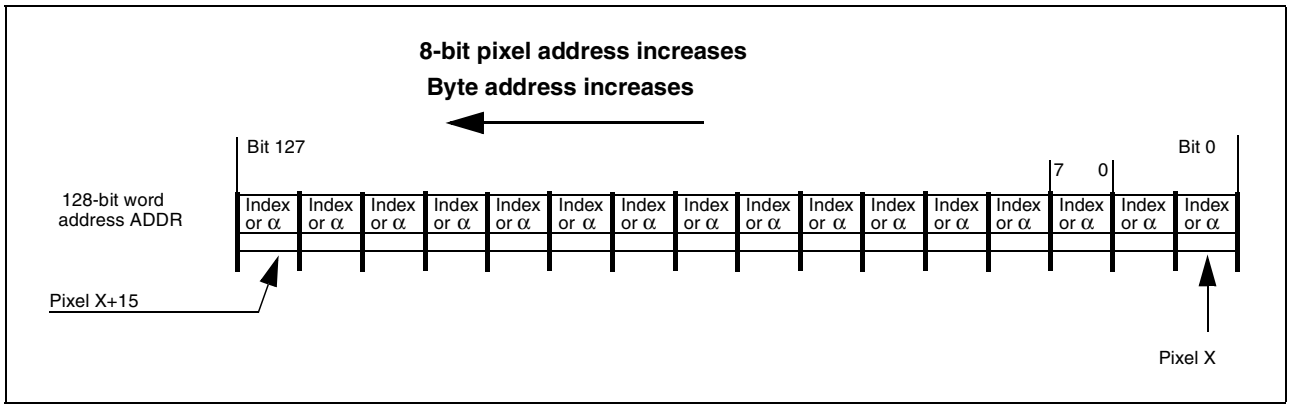


Note: 1 The memory address for pixel (0,0) must be aligned on a 16-bit word address boundary.

2 The pitch value must be a multiple of two bytes.

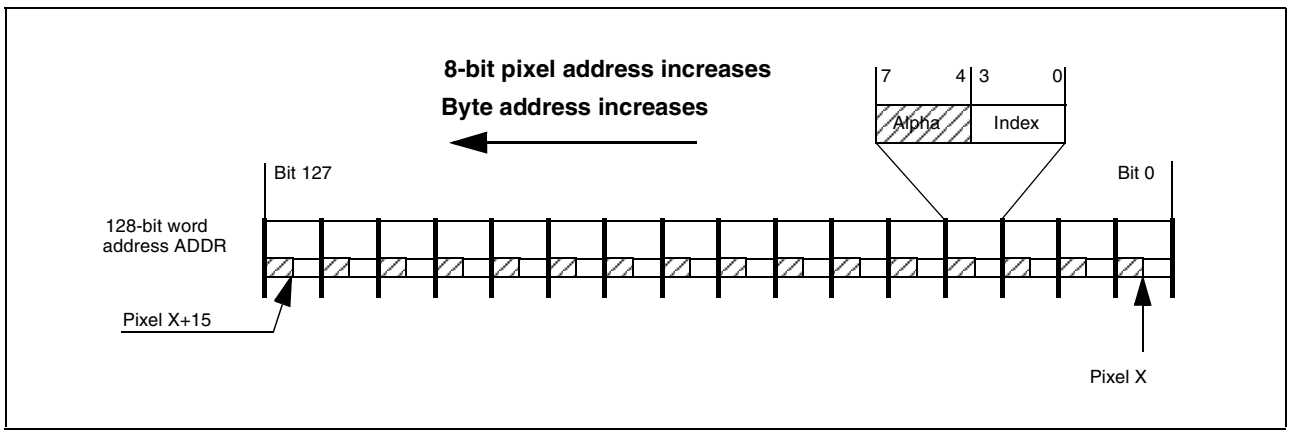
47.8.6 8-bit per pixel format

Figure 146: CLUT8 / A8 alignment in a 128-bit word



Note: The memory address for pixel (0,0) must be byte-aligned.

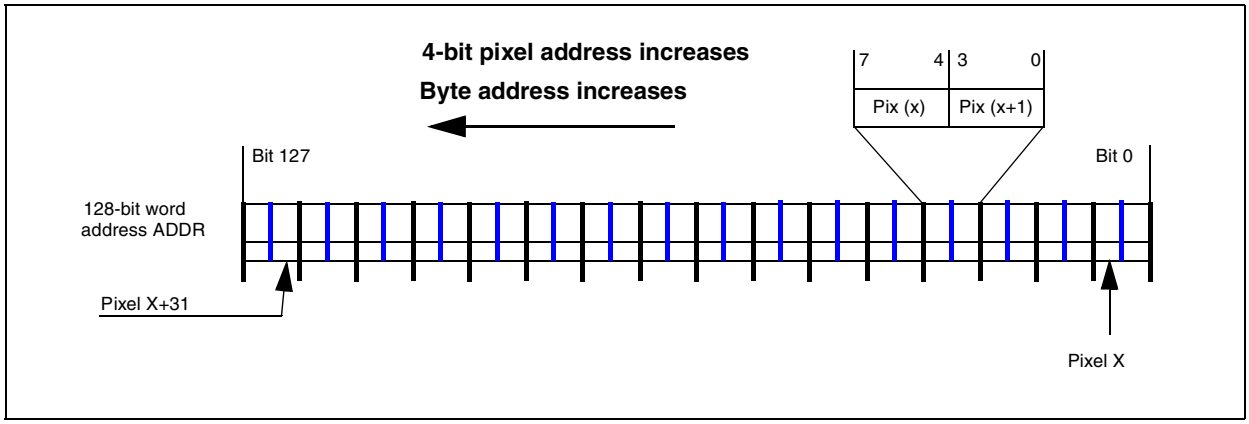
Figure 147: ACLUT44 alignment in a 128-bit word



Note: The memory address for pixel (0,0) must be byte-aligned.

47.8.7 1/2/4 bpp format

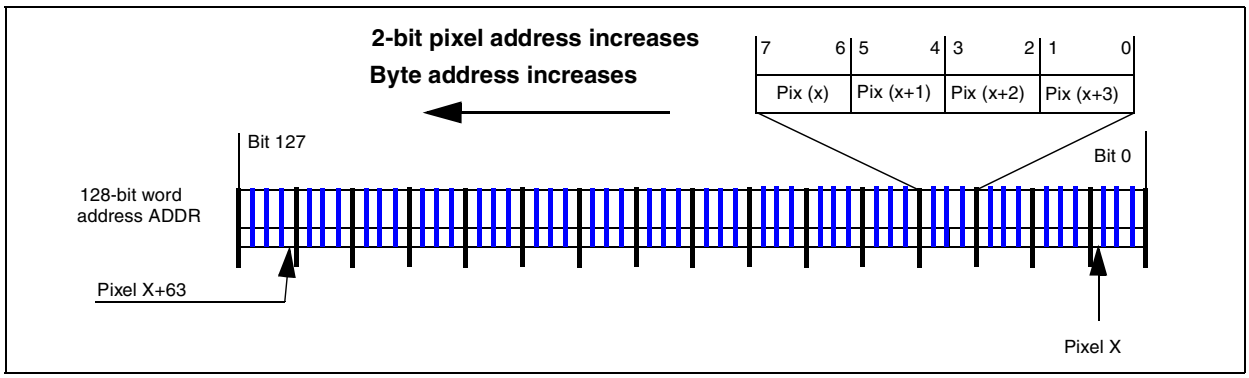
Figure 148: ACLUT4 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be byte-aligned.

2 The pitch is specified in bytes.

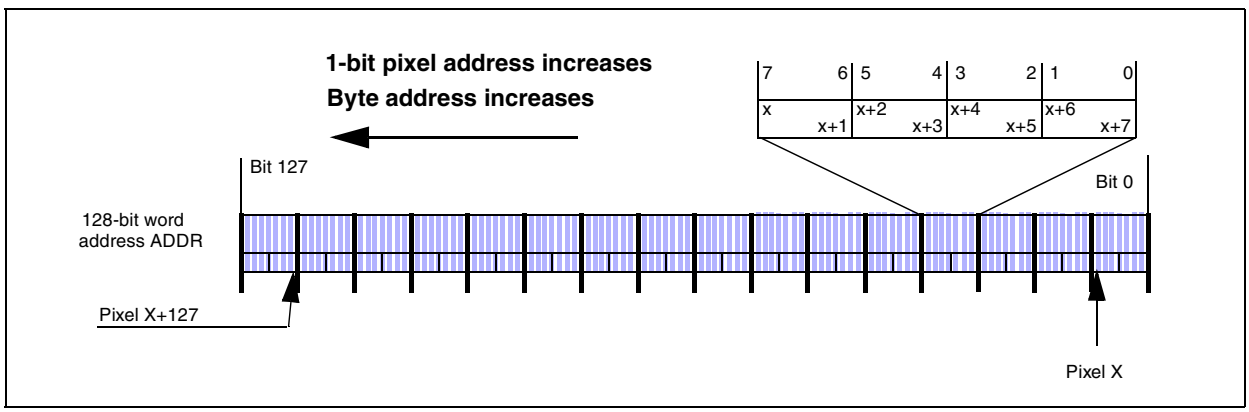
Figure 149: CLUT2 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be byte-aligned.

2 The pitch is specified in bytes.

Figure 150: ACLUT1 / A1 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be byte-aligned.

2 The pitch is specified in bytes.

3 The pixel ordering within a byte, as shown in the previous three figures, is the one supported by the display pipelines. The blitter can support either the pixel ordering as shown, or the reverse order, so that the cell can handle any organization.

48 2D graphics processor registers

Blitter register addresses are provided as
BlitterBaseAddress + *BlitterRegsBaseAddress* + offset.

Blitter 2D coefficient addresses are provided as
BlitterBaseAddress + *Blitter2DFilterOffset* + offset.

The *BlitterBaseAddress* is:
0x1920 B000.

The *BlitterRegsBaseAddress* is:
0x000.

The *Blitter2DFilterOffset* is:
0x800.

48.1 Register map

Table 159: Blitter register map overview

Register function	Register	Offset	Type	128-bit word alignment	
CPU Control / status registers	BLT_CTRL	0x00	R/W		
	BLT_STA1	0x04	RO		
	BLT_STA2	0x08	RO		
	BLT_STA3	0x0C	Mixed		
General configuration	BLT_NIP	0x10	R/W/LLU	Memory node	Node 128-bit word 1
	BLT_USR	0x14	RO/LLU		
	BLT_INS	0x18	RO/LLU		
Source 1 configuration	BLT_S1BA	0x1C	RO/LLU		
	BLT_S1TY	0x20	RO/LLU		Node 128-bit word 2
	BLT_S1XY	0x24	RO/LLU		
	BLT_S1SZ	0x28	RO/LLU		
	BLT_S1CF	0x2C	RO/LLU		
Source 2 configuration	BLT_S2BA	0x30	RO/LLU		Node 128-bit word 3
	BLT_S2TY	0x34	RO/LLU		
	BLT_S2XY	0x38	RO/LLU		
	BLT_S2SZ	0x3C	RO/LLU		
	BLT_S2CF	0x40	RO/LLU		Node 128-bit word 4
Target configuration	BLT_TBA	0x44	RO/LLU		
	BLT_TTY	0x48	RO/LLU		
	BLT_TXY	0x4C	RO/LLU		

Table 159: Blitter register map overview

Register function	Register	Offset	Type	128-bit word alignment	
Rectangular clipping window	BLT_CWO	0x50	RO/LLU	Memory node	Node 128-bit word 5
	BLT_CWS	0x54	RO/LLU		
CLUT and color space conversion	BLT_CCO	0x58	RO/LLU		
	BLT_CML	0x5C	RO/LLU		
2D-rescaler	BLT_RZC	0x60	RO/LLU		Node 128-bit word 6
	BLT_HFP	0x64	RO/LLU		
	BLT_VFP	0x68	RO/LLU		
	BLT_RSF	0x6C	RO/LLU		
Flicker filter	BLT_FF0	0x70	RO/LLU		Node 128-bit word 7
	BLT_FF1	0x74	RO/LLU		
	BLT_FF2	0x78	RO/LLU		
	BLT_FF3	0x7C	RO/LLU		
ALU / Color key	BLT_ACK	0x80	RO/LLU		Node 128-bit word 8
	BLT_KEY1	0x84	RO/LLU		
	BLT_KEY2	0x88	RO/LLU		
Plane mask	BLT_PMK	0x8C	RO/LLU		
Raster scan trigger	BLT_RST	0x90	RO/LLU		Node 128-bit word 9
XYL mode	BLT_XYL	0x94	RO/LLU		
	BLT_XYP	0x98	RO/LLU		
Reserved	-	0x9C	-		
STBus bandwidth limiter	BLT_STB	0xF0	R/W		
STBus protocol / packet maximum size	BLT_PKZ	0xFC	R/W		

48.2 Register descriptions

BLT_CTRL Blitter control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												SUSP2	SUSP1	START	RST

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x00
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: This register controls the blitter activity.

[31:4] **Reserved**

[3] **SUSP2**
1: Suspend the blitter as soon as possible:
- after the current instruction is completed if the 2D-resize engine or the flicker-filter is enabled,
- at the end of the current target line in other cases.

[2] **SUSP1**
1: Suspend the blitter after the current instruction is completed

[1] **START**
Write 1 then 0 to start the blitter: The node at the address stored in BLT_NIP is fetched and executed

[0] **RST**: Soft reset: the blitter goes back to an idle state.

BLT_STA1 Blitter status 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTR_PTR_MEM_ADDR																															

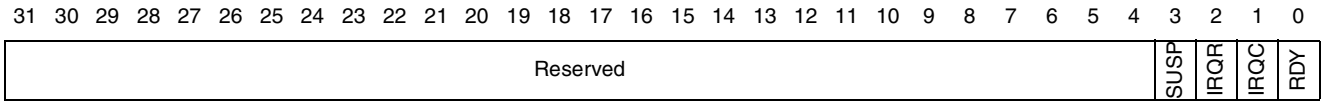
Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x04
Type: RO
Buffer: Immediate
Reset: Undefined
Description: Provides the memory address of the instruction pointer for the current blitter operation.

BLT_STA2 Blitter status 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				TARGET_LINE_NUM											

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x08
Type: RO
Buffer: Immediate
Reset: Undefined
Description: This register provides the current line number to be processed by the blitter, with reference to the target height. The first line transferred is line 0 (and can be the top or the bottom line, depending on the programmed vertical scan order in TTY).
Note: *This value is useful when a blitter instruction has been suspended to enable a higher priority blitter operation.*

BLT_STA3 Blitter status 3



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x0C

Type: Mixed: R/W except for bit 0, RO

Buffer: Immediate

Reset: Undefined

Description: This register shows the activity of the blitter (to identify the interrupt source, if any).

[31:4] **Reserved**

[3] **SUSP**: CURRENT_BLIT_SUSPENDED

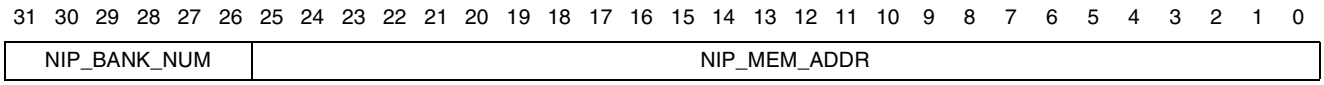
[2] **IRQR**: CURRENT_BLIT_READY2START IRQ
If the blitter has stopped in a Ready2Start state, a start command, register bit BLT_CTRL.START, is required to resume; no new node is fetched prior to the execution.

[1] **IRQC**: Current_Blit_Completed IRQ

[0] **RDY**: Ready (read only)
0: Busy
1: Ready
Bit RDY is reset on a start command, and is set again once the whole link-list has been executed (next instruction pointer = 0), or on a reset command.

Note: Setting the appropriate bit in this register resets the interrupt request (for example, writing 0x2 in this register clears the CURRENT_BLIT_COMPLETED *interrupt status bit, once processed).*
Status bits 1, 2 and 3 have no meaning if the corresponding interrupt is disabled (see BLT_INS[20:18])

BLT_NIP Next instruction pointer



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x10

Type: R/W/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the memory location of the next instruction to execute. The blitter stops when this register is 0 (last node of the link-list).

[31:26] **NIP_BANK_NUM**: 64 MB bank number

[25:0] **NIP_MEM_ADDR**: Memory address for the next instruction pointer for current BLIT

BLT_USR User-specific modes

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GENERAL_PURPOSE																															

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x14
Type: RO/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register is available for the software to implement special synchronization schemes, or any other required tricks.

BLT_INS Blitter instruction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRIG_COND_CTRL							XYL	Reserved		IRQ_MASK			Reserved			PLANE_MASK	Reserved	OCSC	CKEY	RECT_CLIP	FILCK_FILT	2DRESCALE	CLUTOP	ICSC	Reserved	SRC2		SRC1		

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x18
Type: RO/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register controls the data flow configuration of the blitter engine, the interrupt model and the trigger condition.

[31] **Reserved**

[30:24] **TRIG_COND_CTRL**: Set trigger condition control
[30]: wait for Vsync event, start of bottom field
[29]: wait for Vsync event, start of top field
[28]: wait for end of capture event, bottom field
[27]: wait for end of capture event, top field
[26]: wait for raster compare event, bottom field
[25]: wait for raster compare event, top field
[24]: VTG selection: 0:VTG1, 1:VTG2
Note: A raster compare event is generated when the current video line number, provided by the selected VTG, equals the value programmed in BLT_RST (with a progressive display, use top-field control bits only).

[23] **XYL**: When set to 1, this bit indicates that the current instruction is a XYL blitter operation

[22:21] **Reserved**

[20:18] **IRQ_MASK**: IRQ enable mask
[20]: Enable Current_Blit_Suspended interrupt. If set, an interrupt is generated once the blitter has reached the idle state, following a suspend command.
[19]: Enable CURRENT_BLIT_READY2START interrupt. If set, an interrupt is generated after the new instruction node has been captured, before the execution starts. The blitter is then in an idle state, waiting for the CPU to activate the start command in BLT_CTRL.
[18]: Enable Current_Blit_Completed interrupt. If set, an interrupt is generated once the blitter operation is completed. Then the blitter stops, and a start command is mandatory to continue.

[17:15] **Reserved**

[14] **PLANE_MASK**: Enable plane mask

[13] **Reserved**

[12] **OCSC**: Enable output color space converter

[11] **CKEY**: Enable color key

[10] **RECT_CLIP**: Enable rectangular clipping

[9] **FLICK_FILT**: Enable flicker filter on source 2

[8] **2DRESCALE**: Enable 2D-rescaling engine on source 2

[7] **CLUTOP**: Enable CLUT-based operator on source 2

[6] **ICSC**: Enable input color space converter

[5] **Reserved**

[4:3] **SRC2**: Source 2 mode
00: Source 2 disabled
01: Source 2 fetched from memory
10: Reserved
11: Source 2 color fill register

[2:0] **SRC1**: Source 1 mode
000: Source 1 disabled
001: Source 1 fetched from memory
010: Reserved
011: Source 1 color fill register
100: Source 1 direct-copy mode
101: Reserved
110: Reserved
111: Source 1 direct-fill mode

BLT_RST Raster scan trigger

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															VTG	Reserved					TRIG_LINE_NUM										

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x90
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register contains the video line number for conditionally starting a blitter operation.

[31:17] **Reserved**

[16] **VTG**: 0: VTG1 1: VTG2

[15:11] **Reserved**

[10:0] **TRIG_LINE_NUM**: expected video line number to trigger

BLT_S1BA Source 1 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S1BA_BANK_NUM						S1BA_MEM_ADDR																									

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x1C
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register provides the byte memory location of the pixmap (pixel (0,0)), for retrieving graphics data for blitter source 1. Pixel (0,0) is always the upper-left pixel.

[31:26] **S1BA_BANK_NUM**: 64 MB bank number

[25:0] **S1BA_MEM_ADDR**: byte address of the first pixel

Note: In XYL standard mode, this register must be programmed with the base address of the target bitmap.

BLT_S1XY Source 1 XY location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
YCOORD																XCOORD															

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x24

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the XY location of the first pixel to be transferred, with reference to the top-left corner of the blitter source 1 bitmap [0,0]. This XY location depends on the programmed horizontal and vertical scan order (register BLT_S1TY).

[31:16] **YCOORD**: Y coordinate of the first pixel to be transferred (signed)

[15:0] **XCOORD**: X coordinate of the first pixel to be transferred (signed)

BLT_S1TY Source 1 type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	BIGNOTLITTLE	RGB_EXP	SUBBYTE	Reserved		VSO	HSO	MB_FIELD	Reserved	ALPHA_RANGE	COLOR_FORM					PIXMAP_PITCH															

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x20

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register configures the color format, pitch and raster scan order for the blitter source 1 pixmap.

[31] **Reserved**

[30] **BIGNOTLITTLE**: Bitmap endianness
0: Little endian 1: Big endian

[29] **RGB_EXP**: RGB expansion mode
0: Missing LSBs are filled with 0 1: MSBs are duplicated on missing LSBs

[28] **SUBBYTE**: Sub-byte formats, pixels ordering
0: Screen most right pixel in most significant bits 1: Screen most right pixel in least significant bits

[27:26] **Reserved**

[25] **VSO**: Vertical scan order
0: Top to bottom) 1: Bottom to top

[24] **HSO**: Horizontal scan order
0: Left to right 1: Right to left

[23] **MB_FIELD**: access mode in macro-block organized frame buffers (YCbCr420MB and YCbCr422MB)
0: Access in frame mode 1: Access in field mode (every other line)

[22] **Reserved**

[21] **ALPHA_RANGE**: 8-bit alpha range (for ARGB8565, ARGB8888, ACLUT88 and A8 formats only)
0: 0 to 128 (128 means opaque) 1: 0 to 255 (255 means opaque)

[20:16] **COLOR_FORM:** pixmap color format

RGB types	CLUT types
0 0000: RGB565	0 1000: CLUT1
0 0001: RGB888	0 1001: CLUT2
0 0100: ARGB8565	0 1010: CLUT4
0 0101: ARGB8888	0 1011: CLUT8
0 0110: ARGB1555	0 1100: ACLUT44
0 0111: ARGB4444	0 1101: ACLUT88
YCbCr types	Miscellaneous types
1 0000: YCbCr888	1 1000: A1
1 0010: Reserved	1 1001: A8
1 0011: YCbCr422MB	1 1111: BYTE
1 0100: YCbCr420MB	
1 0101: AYCbCr8888	

For modes YCbCr4;2;2MB and YCbCr4;2;0MB (macroblock based, dual buffer), source 1 and source 2 are both used, and must be programmed with the same color format

[15:0] **PIXMAP_PITCH**: Pixmap pitch in bytes

Note: In XYL standard mode, this register must be filled with the bitmap characteristics of the target bitmap. Bits 24 and 25 have no meaning and should be set to 0.

BLT_S1SZ Source 1/target window size

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
Reserved	WIN_HEIGHT	Reserved	WIN_WIDTH

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x28

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the width and height of the rectangular window to be transferred. The size of the target always matches source 1 size (no scaling is provided on source 1 path). Hence, there is only one physical register.

[31:28] **Reserved**

[27:16] **WIN_HEIGHT**: Height (in lines) of window to be transferred

[15:12] **Reserved**

[11:0] **WIN_WIDTH**: Width (in pixels) of window to be transferred

Note: In XYL standard mode, this register is not used.

BLT_S1CF Source 1 color fill

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
SRC1_COLOR_FILL

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x2C

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the solid color value to be used for filling or shading from source 1 during a blitter operation. The number of significant bits varies (from 1 bpp to 32 bpp) according to the bit depth of the selected color format. The value is always right-justified in the 32-bit physical register.

BLT_S2BA Source 2 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S2BA_BANK_NUM						S2BA_MEM_ADDR																									

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x30

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

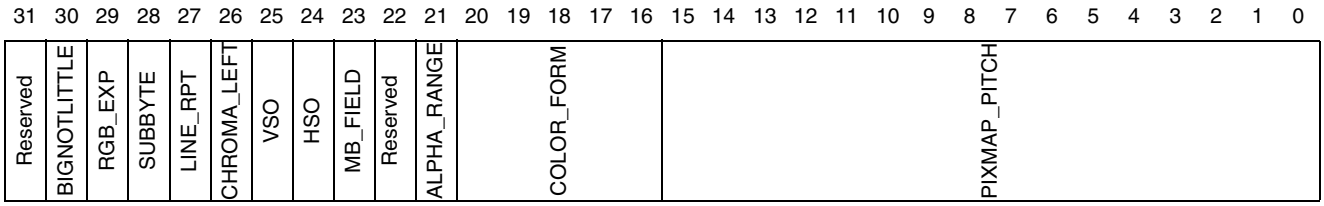
Description: This register provides the byte memory location address of the pixmap (pixel (0,0)), for retrieving graphics data for blitter source 2. Pixel (0,0) is always the upper-left pixel.

[31:26] **S2BA_BANK_NUM**: 64 MB bank number

[25:0] **S2BA_MEM_ADDR**: Byte address of the first pixel

Note: In XYL standard mode, this register must be programmed with the base address of the clipmask bitmap, if any.

BLT_S2TY Source 2 type



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x34

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register configures the color format, pitch and raster scan order for the blitter source 2 pixmap.

[31] **Reserved**

[30] **BIGNOTLITTLE**: Bitmap endianness
0: Little endian 1: Big endian

[29] **RGB_EXP**: RGB expansion mode
0: Missing LSBs are filled with 0 1: MSBs are duplicated on missing LSBs

[28] **SUBBYTE**: Sub-byte formats, pixels ordering
0: Screen most right pixel in most significant bits 1: Screen most right pixel in least significant bits

[27] **LINE_RPT**: Chroma line repeat mode if source is YCbCr420MB
0: No line repeat: the vertical chroma upsampling must be done using the 2D-resize
1: The vertical chroma upsampling uses a line repeat scheme

[26] **CHROMA_LEFT**: Chroma left extended
0: If the first chroma sample in a line is Y-only, its chroma is estimated from the following pixel only
1: If the first chroma sample in a line is Y-only, its chroma is estimated by averaging the following pixel and the previous sample (out of the defined S2 window)

[25] **VSO**: Vertical scan order
0: Top to bottom) 1: Bottom to top

[24] **HSO**: Horizontal scan order
0: Left to right 1: Right to left

[23] **MB_FIELD**: Access mode in macro-block organized frame buffers (YCbCr420MB and YCbCr422MB)
0: Access in frame mode 1: Access in field mode (every other line)

[22] **Reserved**

[21] **ALPHA_RANGE**: 8-bit alpha range (for ARGB8565, ARGB8888, ACLUT88 and A8 formats only)
0: 0 to 128 (128 means opaque) 1: 0 to 255 (255 means opaque)

[20:16] **COLOR_FORM**: Pixmap color format

RGB types	CLUT types
0 0000: RGB565	0 1000: CLUT1
0 0001: RGB888	0 1001: CLUT2
0 0100: ARGB8565	0 1010: CLUT4
0 0101: ARGB8888	0 1011: CLUT8
0 0110: ARGB1555	0 1100: ACLUT44
0 0111: ARGB4444	0 1101: ACLUT88
YCbCr types	MISC types
1 0000: YCbCr888	1 1000: A1
1 0010: YCbCr422R	1 1001: A8
1 0011: YCbCr422MB	1 1111: BYTE
1 0100: YCbCr420MB	
1 0101: AYCbCr8888	

For modes YCbCr4;2;2MB and YCbCr4;2;0MB (macroblock based, dual buffer), sourceS 1 and 2 are both used, and must be programmed with the same color format.

[15:0] **PIXMAP_PITCH**: Pixmap pitch in bytes

Note: In XYL standard mode, this register must be filled with the characteristics of the clipmask bitmap, if any. Bits [24, 25, 26, 27, 29] have no meaning and should be set to 0.

BLT_S2XY Source 2 XY location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
YCOORD																XCOORD															

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x38

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the XY location of the first pixel to be transferred, with reference to the top-left corner of the blitter source 2 bitmap (0,0). This XY location depends on the programmed horizontal and vertical scan order (register BLT_S2TY).

[31:16] **YCOORD**: Y coordinate of the first pixel to be transferred (signed).

[15:0] **XCOORD**: X coordinate of the first pixel to be transferred (signed).

Note: In XYL standard mode, this register contains the 2D-vector that gives the location of the (0,0) clipmask value, with respect to the (0,0) pixel location of the target bitmap. XCOORD provides the horizontal offset, YCOORD provides the vertical offset (both values are 16-bit signed integers).

BLT_S2SZ Source 2 window size

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SRC2_WIN_HEIGHT												Reserved				SRC2_WIN_WIDTH											

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x3C

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the width and height of the rectangular window to be transferred.

[31:28] **Reserved**

[27:16] **SRC2_WIN_HEIGHT**: Height (in lines) of source 2 window to be transferred

[15:12] **Reserved**

[11:0] **SRC2_WIN_WIDTH**: Width (in pixels) of source 2 window to be transferred

BLT_S2CF Source 2 color fill

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC2_COLOR_FILL																															

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x40

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the solid color value to be used for filling or shading from source 2 during a blitter operation. The number of significant bits varies from 1 bpp to 32 bpp according to the bit depth of the selected color format. The value is always right-justified in the 32-bit physical register.

Note: In XYL standard mode, configured for either XY or XYL sub-instruction format, this register contains the drawing color data. When XYC or XYLC format is used, the register content is "don't care".

BLT_TBA Target base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBA_BANK_NUM						TBA_MEM_ADDR																									

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x44

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the byte memory location of the target pixmap (pixel (0,0)), for storing the result of the blitter operation (target data). Pixel (0,0) is always the top-left pixel.

[31:26] **TBA_BANK_NUM**: 64 MB bank number

[25:0] **TBA_MEM_ADDR**: Byte address of the first pixel

Note: In XYL standard mode, this register must be programmed with the base address of the target bitmap.

BLT_TXY Target XY location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
YCOORD																XCOORD															

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x4C

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the XY location of the first pixel to be transferred, with reference to the top-left corner of the target bitmap (0,0). This XY location depends on the programmed horizontal and vertical scan order (register BLT_TTY).

[31:16] **YCOORD:** Y coordinate of the first pixel to be transferred (16-bit signed)

[15:0] **XCOORD:** X coordinate of the first pixel to be transferred (16-bit signed)

Note: In XYL standard mode, this register is unused.

BLT_TTY Target type

31	30	29	28	27	26	25	24	23 22	21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved	BIGNOTLITTLE	Reserved	SUBBYTE	Reserved	RGB_ROUND	VSO	HSO	Reserved	ALPHA_RANGE	COLOR_FORM	PIXMAP_PITCH

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x48

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register configures the color format, pitch and raster scan order for the blitter target pixmap.

[31] **Reserved**

[30] **BIGNOTLITTLE**: Bitmap endianness
0: Little endian 1: Big endian

[29] **Reserved**

[28] **SUBBYTE**: Sub-byte formats, pixels ordering
0: Screen most right pixel in most significant bits
1: Screen most right pixel in least significant bits

[27] **Reserved**

[26] **RGB_ROUND**: Rounding mode
0: Normal rounding (+0.5 then truncation) 1: Enable 2x2 dither when rounding

[25] **VSO**: Vertical scan order
0: Top to bottom) 1: Bottom to top

[24] **HSO**: Horizontal scan order
0: Left to right 1: Right to left

[23:22] **Reserved**

[21] **ALPHA_RANGE:** 8-bit alpha range (for ARGB8565, ARGB8888, ACLUT88 and A8 formats only)
0: 0 to 128 (128 means opaque) 1: 0 to 255 (255 means opaque)

[20:16] **COLOR_FORM**: Pixmap color format

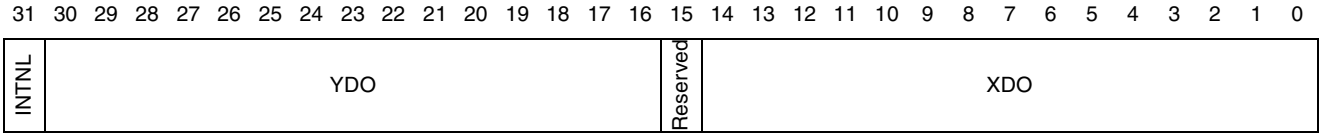
RGB types	CLUT types
0 0000: RGB565	0 1000: CLUT1
0 0001: RGB888	0 1001: CLUT2
0 0100: ARGB8565	0 1010: CLUT4
0 0101: ARGB8888	0 1011: CLUT8
0 0110: ARGB1555	0 1100: ACLUT44
0 0111: ARGB4444	0 1101: ACLUT88
YCbCr types	Misc types
1 0000: YCbCr888	1 1000: A1
1 0010: YCbCr422R	1 1001: A8
1 0011: YCbCr422MB	1 1111: BYTE
1 0100: YCbCr420MB	
1 0101: AYCbCr8888	

For modes YCbCr4;2;2MB and YCbCr4;2;0MB (macroblock based, dual buffer), source 1 and source 2 are both used, and must be programmed with the same color format

[15:0] **PIXMAP_PITCH**: Pixmap pitch in bytes

Note: In XYL standard mode, this register must be filled with the characteristics of the target bitmap. Bits[24, 25, 27] have no meaning and should be set to 0.

BLT_CWO Clipping window offset



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x50

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the XY location of the top-left corner of the clipping window, with reference to the target (0,0) origin (The clipping window boundaries are always specified with positive values, as there is a default clipping mechanism for any X or Y negative address).

[31] **INTNL**: Internal or external clipping
0: Writing is only allowed inside the window (including the borders)
1: Writing is only allowed outside the window (excluding the borders)

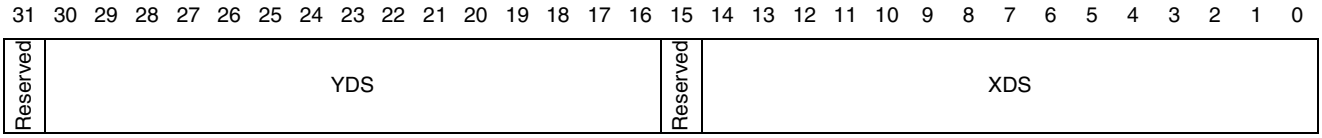
[30:16] **YDO**: Y coordinate (unsigned)

[15] **Reserved**

[14:0] **XDO**: X coordinate (unsigned)

Note: In XYL standard mode, the rectangular clipping window can be used. In the MediaHighway clipmask implementation, the clipping window must be programmed to the intersection of the regular rectangular clipping and the rectangular clipmask window.

BLT_CWS Clipping window stop



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x54

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register gives the XY location of the bottom-right corner of the clipping window, with reference to the target (0,0) origin (The clipping window boundaries are always specified with positive values, as there is a default clipping mechanism for any X or Y negative address).

[31] **Reserved**

[30:16] **YDS**: Y coordinate (unsigned)

[15] **Reserved**

[14:0] **XDS**: X coordinate (unsigned)

Note: In XYL standard mode, the rectangular clipping window can be used. In the MediaHighway clipmask implementation, the clipping window must be programmed to the intersection of the regular rectangular clipping, and the rectangular clipmask window.

BLT_CCO CLUT and color conversion operators

31–23	22–19	18	17–16	15–11	10	9	8	7–4	3	2	1	0
Reserved	CLUT_ERRDIFF	CLUT_UPDATE	CLUT_MODE	Reserved	CCO_OUTVID*n*GFX	CCO_OUTSIGN	CCO_OUTCOL	Reserved	CCO_INVID*n*GFX	CCO_INDIR	CCO_INSIGN	CCO_INCOL

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x58

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register controls the two color space converters and the CLUT operator.

CLUT operators

[31:23] **Reserved**

[22:19] **CLUT_ERRDIFF**: Error diffusion weight (if color reduction)
0000: 0% (disabled) 0001: 100%
0010: 75% 0011: 50%
0100: 25% 1000: Adaptive

[18] **CLUT_UPDATE**: Enable CLUT update

[17:16] **CLUT_MODE**: CLUT operation modes
00: Color expansion mode ((A)CLUT*n* >> true color)
01: Color correction (true color >> true color) 10: Color reduction mode (true color >> CLUT*n*)

[15:11] **Reserved**

Color-space converters

[10] **CCO_OUTVID*n*GFX**: Output color space converter uses
0: Graphics matrix 1: Video matrix

[9] **CCO_OUTSIGN:** Output color converter chroma format
0: Offset binary 1: Two's complement, signed

[8] **CCO_OUTCOL:** Output color converter colorimetry
0: ITU-R BT.601 1: ITU-R BT.709

[7:4] **Reserved**

[3] **CCO_INVID*n*GFX**: Input color space converter uses
0: Graphics matrix 1: Video matrix

[2] **CCO_INDIR**: Input color converter direction
0: YCbCr2RGB 1: RGB2YCbCr
The direction for the output color converter is automatically the opposite

[1] **CCO_INSIGN**: Input color converter chroma format
0: Offset binary 1: Two's complement, signed

[0] **CCO_INCOL**: Input color converter colorimetry
0: ITU-R BT.601 1: ITU-R BT.709

Note: The CLUT and the input color space converter cannot be used in XYL mode. But the output color space converter is available.

BLT_CML CLUT memory location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CML_BANK_NUM						CML_MEM_ADDR																						Reserved			

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x5C

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

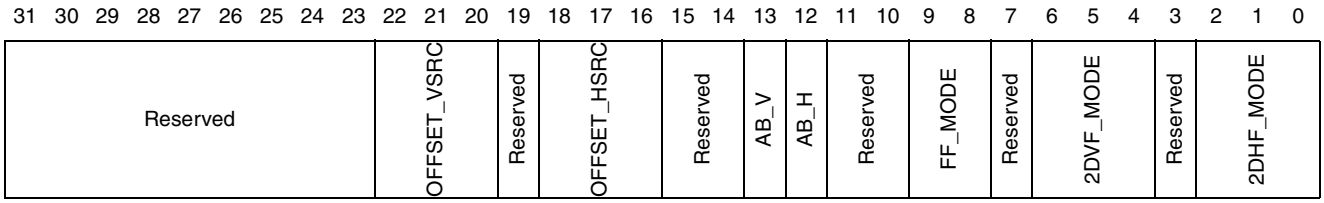
Description: Provides the byte-memory location to retrieve the CLUT data in the local memory. As the CLUTs are always aligned on 128-bit word boundary, the four LSBs are ignored.

[31:26] **CML_BANK_NUM**: 64 MB bank number

[25:4] **CML_MEM_ADDR**: Byte address

[3:0] **Reserved**

BLT_RZC 2D resize control



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x60

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register configures the blitter 2D resize engine.

[31:23] **Reserved**

[22:20] **OFFSET_VSRC**: Initial subposition for the VSRC

[19] **Reserved**

[18:16] **OFFSET_HSRC**: Initial subposition for the HSRC

[15:14] **Reserved**

[13] **AB_V**: Enable AlphaVBorder

[12] **AB_H**: Enable AlphaHBorder

[10:11] **Reserved**

[9:8] **FF_MODE**: Flicker filter mode

00: Force filter 0
01: Adaptive flicker filtering (RGB format only)
10: Test mode
11: Reserved

[7] **Reserved**

[6:4] **2DVF_MODE**: 2D-filter mode (vertical)

0xx: Vertical resizer disabled

100: V.resizer enabled	Vfilter inactive
101: V.resizer enabled	Vfilter active on color channels, inactive on alpha channel
110: V.resizer enabled	Vfilter inactive on color channels, active on alpha channel
111: V.resizer enabled	Vfilter active on both color and alpha channels

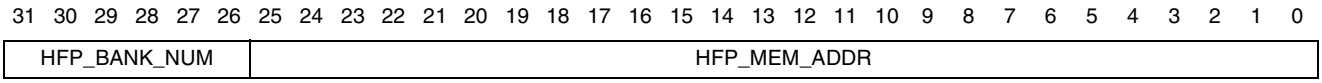
[3] **Reserved**

[2:0] **2DHF_MODE**: 2D-filter mode (horizontal)

0xx: Horizontal resizer disabled

100: H.resizer enabled	Hfilter inactive
101: H.resizer enabled	Hfilter active on color channels, inactive on alpha channel
110: H.resizer enabled	Hfilter inactive on color channels, active on alpha channel
111: H.resizer enabled	VHilter active on both color and alpha channels

BLT_HFP Horizontal filter coefficients pointer



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x64

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

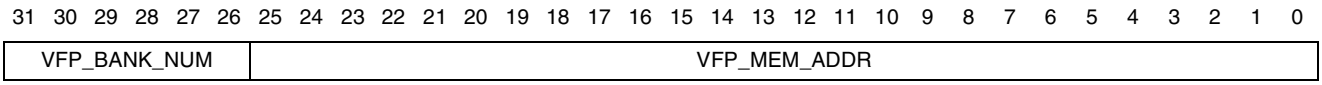
Reset: 0x00

Description: This register provides the byte memory location for retrieving the horizontal filter coefficients in the local memory (40 bytes).

[31:26] **HFP_BANK_NUM**: 64 MB bank number

[25:0] **HFP_MEM_ADDR**: Horizontal filter first coefficient byte address

BLT_VFP Vertical filter coefficients pointer



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x68

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

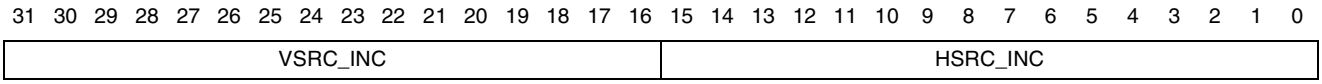
Reset: 0x00

Description: This register provides the byte memory location for retrieving the vertical filter coefficients in the local memory (40 bytes).

[31:26] **VFP_BANK_NUM**: 64 MB bank number

[25:0] **VFP_MEM_ADDR**: Vertical filter first coefficient byte address

BLT_RSF Resize scaling factor



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x6C

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

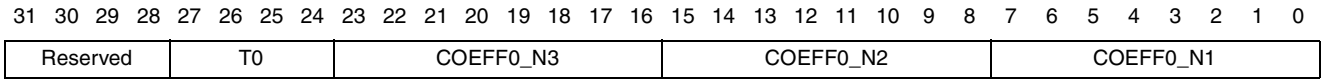
Reset: 0x00

Description: This register provides the scaling factors for both the horizontal and the vertical sample rate converters.

[31:16] **VSRC_INC**: Increment value for VSRC (6.10 format)

[15:0] **HSRC_INC**: Increment value for HSRC (6.10 format)

BLT_FF0 Flicker filter 0



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x70

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the flicker filter 0 coefficients, and the threshold value used to select F0. F0 is used between 0 and T0-1.

[31:28] **Reserved**

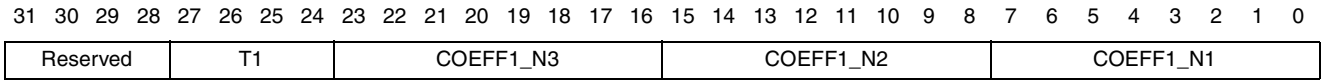
[27:24] **T0**: Threshold value

[23:16] **COEFF0_N3**: (*n* + 1) coefficient (1.7 format)

[15:8] **COEFF0_N2**: (*n*) coefficient (1.7 format)

[7:0] **COEFF0_N1**: (n - 1) coefficient (1.7 format)

BLT_FF1 Flicker filter 1



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x74

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the flicker filter 1 coefficients, and the threshold value used to select F1. F1 is used between T0 and T1-1.

[31:28] **Reserved**

[27:24] **T1**: Threshold value

[23:16] **COEFF1_N3**: (*n* + 1) coefficient (1.7 format)

[15:8] **COEFF1_N2**: (*n*) coefficient (1.7 format)

[7:0] **COEFF1_N1**: (*n* - 1) coefficient (1.7 format)

BLT_FF2 Flicker filter 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				T2				COEFF2_N3								COEFF2_N2								COEFF2_N1							

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x78

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the flicker filter 2 coefficients, and the threshold value used to select F2. F2 is used between T1 and T2-1.

[31:28] **Reserved**

[27:24] **T2**: Threshold value

[23:16] **COEFF2_N3**: (*n* + 1) coefficient (1.7 format)

[15:8] **COEFF2_N2**: (*n*) coefficient (1.7 format)

[7:0] **COEFF2_N1**: (*n* - 1) coefficient (1.7 format)

BLT_FF3 Flicker filter 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								COEFF3_N3								COEFF3_N2								COEFF3_N1							

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x7C

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register provides the flicker filter 3 coefficients, which operate in the range T2 to 15.

[31:24] **Reserved**

[23:16] **COEFF3_N3:** (*n*+1) coefficient (1.7 format)

[15:8] **COEFF3_N2:** (*n*) coefficient (1.7 format)

[7:0] **COEFF3_N1**: (*n*-1) coefficient (1.7 format)

BLT_ACK ALU and color key control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CKEY_SEL		ACK_CKEY						GALPHA_ROPID								Reserved				MODE			

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x80

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register configures the ALU and the color key module of the blitter.

[31:24] **Reserved**

[23:22] **CKEY_SEL**: Color key input selection
00: Source 1 pixel bus (destination color key)
01: Source 2 pixel bus before the CLUT (source color key)
10: Source 2 pixel bus after the CLUT (source color key)
11: Reserved

[21:16] **ACK_CKEY**: Color key configuration

[5:4] x0: Red / index component ignored (disabled = always match)
01: Red / index enabled: match if ($R_{min} \leq R \leq R_{max}$)
11: Red / index enabled: match if (($R < R_{min}$) or ($R > R_{max}$))
When using the color key feature on a CLUT bitmap, only the blue channel parameters are relevant, as internally, the blue component and the index information are sharing the same data path.

[3:2] **MODE[3:2]**
x0: Green component ignored (disabled = always match)
01: Green enabled: match if ($G_{min} \leq G \leq G_{max}$)
11: Green enabled: match if (($G < G_{min}$) or ($G > G_{max}$))

[1:0] **MODE[1:0]**
x0: Blue component ignored (disabled = always match)
01: Blue enabled: match if ($B_{min} \leq B \leq B_{max}$)
11: Blue enabled: match if (($B < B_{min}$) or ($B > B_{max}$))

[15:8] **GALPHA_ROPID**
ALU global alpha (8 bits / 0 to 128), in blending mode, or ROP identifier (4 LSBs), in logical mode.
Raster operation table (GALPHA_ROPID[3:0]):

0000: CLEAR	result = all 0	0001: AND	result = S2 AND S1
0010: ANDrev	result = S2 AND (NOT S1)	0011: COPY	result = S2
0100: ANDinvert	result = (NOT S2) AND S1	0101: NOOP	result = S1
0110: XOR	result = S2 XOR S1	0111: OR	result = S2 OR S1
1000: NOR	result = (NOT S2) AND (NOT S1)	1001: EQUIV	result = (NOT S2) XOR S1
1010: INVERT	result = (NOT S1)	1011: ORreverse	result = S2 OR (NOT S1)
1100: COPYinv	result = (NOT S2)	1101: ORinvert	result = (NOT S2) OR S1
1110: NAND	result = (NOT S2) OR (NOT S1)	1111: SET	result = all 1

[7:4] **Reserved**

[3:0] **MODE**: ALU operation modes

0000: Bypass source 1	0001: Logical operation
0010: Blending mode, source 2 not premultiplied	0011: Blending mode, source 2 premultiplied
0100: Clipmask pass in logical mode / First pass	0101: Clipmask pass in blending mode
0110: 4:2:0 source 1 / 2 YCbCr concatenation	0111: Bypass source 2
1000: Clipmask pass in logical mode / 2nd pass	1001: Clipmask in XYL mode, logical operation

1010: Clipmask in XYL mode, blending operation, source 2 not premultiplied
1011: Clipmask in XYL mode, blending operation, source 2 premultiplied

Note: In XYL standard mode, only ALU operation modes 1, 2, 3, 7, 9, 10 and 11 are available.

BLT_KEY1 Color key 1

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	RED_MIN	GREEN_MIN	BLUE_MIN

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x84
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register contains the color key minimum values for the RGB components.

[31:24] **Reserved**

[23:16] **RED_MIN**: Red component minimum value

[15:8] **GREEN_MIN**: Green component minimum value

[7:0] **BLUE_MIN**: Blue component minimum value / index

BLT_KEY2 Color key 2

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	RED_MAX	GREEN_MAX	BLUE_MAX

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x88
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register contains the color key maximum values for the RGB components.

[31:24] **Reserved**

[23:16] **RED_MAX**: Red component maximum value

[15:8] **GREEN_MAX**: Green component maximum value

[7:0] **BLUE_MAX**: Blue component maximum value / index

Note: When the color key module is intended to work on CLUT type data, only the blue path is used (green and red should be disabled). Thus, it is possible to track a single index, as well as a range of indexes.

BLT_PMK Plane mask

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
PLANE_MASK

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x8C
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register provides the plane mask when the feature is enabled in the blitter register bit BLT_INS[14]. The number of valid bits (1 to 32) in a given configuration must match the selected target pixel width. The value is always right-justified if bpp < 32.

Note: A bit in the target pixel can be updated by the new color bit, only if the corresponding bit in the mask has been set to 1. The feature is available in XYL standard mode.

BLT_XYL XYL mode control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM_SUBINS																Reserved						SUBINS		Reserved							

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x94
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register allows to control a blitter execution when operating in XYL mode.

[31:16] **NUM_SUBINS**: Number of subinstructions in the XYL blitter instruction

[15:10] **Reserved**

[9:8] **SUBINS**: Subinstructions format
00: XY subinstruction
01: XYL subinstruction
10: XYC subinstruction
11: XYLC subinstruction

[7:1] **Reserved**

[0] **Reserved**: Must be set to 0.

BLT_XYP XY subinstructions pointer

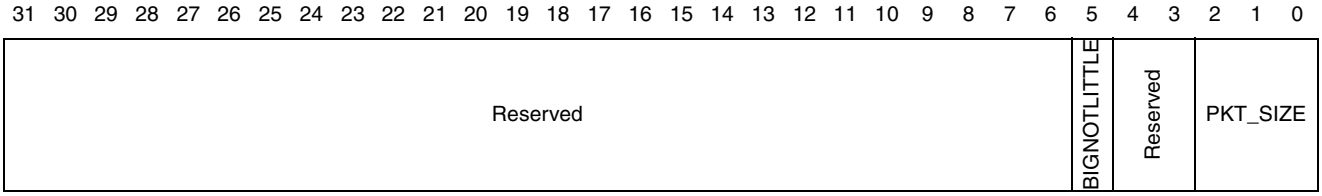
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBI_BANK_NUM						SUBI_MEM_ADDR																									

Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0x98
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register provides the byte memory location for the subinstruction buffer

[31:26] **SUBI_BANK_NUM**: 64 MB bank number

[25:0] **SUBI_MEM_ADDR**: Subinstructions buffer address

BLT_PKZ Packet size control



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0xFC
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: This register controls the packet size on an STBus transaction.

[31:6] **Reserved**

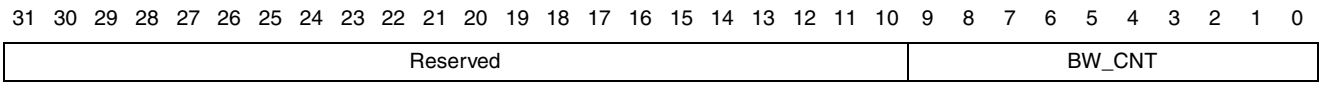
[5] **BIGNOTLITTLE**: Bitmap endianness
0: Little endian 1: Big endian

[4:3] **Reserved**

[2:0] **PKT_SIZE**: Should be set to 000.
000: Packet size = message size 001: Packet size = 16 x 128-bit words
010: Packet size = 8 x 128-bit words 011: Packet size = 4 x 128-bit words
100: Packet size = 2 x 128-bit words 101: Packet size = 1 x 128-bit words

Note: The reason for splitting a message transaction into a smaller entity called packet, is that a CPU access can be inserted between two consecutive packets.

BLT_STB STBus bandwidth limiter



Address: *BlitterBaseAddress* + *BlitterRegsBaseAddress* + 0xF0
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: This register controls STBus bandwidth limiter.

[31:10] **Reserved**

[9:0] **BW_CNT**: Bandwidth count
Sets the number of cycle between two requests on the STBus

48.3 Blitter 2D-filter coefficients

48.3.1 Register map

Table 160: Blitter 2D-filter register map overview

Register function	Offset	Name	128-bit word alignment	
HFilter coefficients	0x00	BLT_HFC1	HFilter	HFilter 128-bit word 1
	0x04	BLT_HFC2		
	0x08	BLT_HFC3		
	0x0C	BLT_HFC4		
	0x10	BLT_HFC5		HFilter 128-bit word 2
	0x14	BLT_HFC6		
	0x18	BLT_HFC7		
	0x1C	BLT_HFC8		
	0x20	BLT_HFC9		HFilter 128-bit word 3
	0x24	BLT_HFC10		
Reserved	0x28			
	0x2C			
VFilter coefficients	0x30	BLT_HFC1	VFilter	VFilter 128-bit word 3
	0x34	BLT_HFC2		
	0x38	BLT_HFC3		
	0x3C	BLT_HFC4		
	0x40	BLT_HFC5		VFilter 128-bit word 4
	0x44	BLT_HFC6		
	0x48	BLT_HFC7		
	0x4C	BLT_HFC8		
	0x50	BLT_HFC9		VFilter 128-bit word 5
	0x54	BLT_HFC10		
Reserved	0x58			
	0x5C			

The filter coefficients are loaded from memory each time a 2D-resize operation with filtering is programmed. The pointers, BLT_HFP and BLT_VFP must be aligned on a 128-bit word boundary (but the coefficients for the horizontal and vertical filters are totally independent, they can be stored anywhere in the blitter memory space, not necessarily at adjacent locations).

48.3.2 Blitter 2D-filter registers description

BLT_HFCn **Horizontal filter coefficients 1 to 10**

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
HFC1	K0.3	K0.2	K0.1	K0.0
HFC2	K1.2	K1.1	K1.0	K0.4
HFC3	K2.1	K2.0	K1.4	K1.3
HFC4	K3.0	K2.4	K2.3	K2.2
HFC5	K3.4	K3.3	K3.2	K3.1
HFC6	K4.3	K4.2	K4.1	K4.0
HFC7	K5.2	K5.1	K5.0	K4.4
HFC8	K6.1	K6.0	K5.4	K5.3
HFC9	K7.0	K6.4	K6.3	K6.2
HFC10	K7.4	K7.3	K7.2	K7.1

Address: *BlitterBaseAddress* + *Blitter2DFilterOffset* + 0x00 (HFC1), 0x04 (HFC2), 0x08 (HFC3), 0x0C (HFC4), 0x10 (HFC5), 0x14 (HFC6), 0x18 (HFC7), 0x1C (HFC8), 0x20 (HFC9), 0x24 (HFC10)

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: These registers store the horizontal sample rate converter (HSRC) coefficients. There are eight subfilters for the HSRC; each subfilter has five 8-bit coefficients. Coefficient *j* of subfilter *i* (subposition *i*) is designated K*i.j*[7:0] and the format is S1.6 (1.0 is encoded as 0x40). The corresponding filter coefficient structure (40 bytes) must be aligned on a 128-bit word boundary when stored in the local memory.

BLT_VFCn Vertical filter coefficients 1 to 10

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
VFC1	K0.3	K0.2	K0.1	K0.0
VFC2	K1.2	K1.1	K1.0	K0.4
VFC3	K2.1	K2.0	K1.4	K1.3
VFC4	K3.0	K2.4	K2.3	K2.2
VFC5	K3.4	K3.3	K3.2	K3.1
VFC6	K4.3	K4.2	K4.1	K4.0
VFC7	K5.2	K5.1	K5.0	K4.4
VFC8	K6.1	K6.0	K5.4	K5.3
VFC9	K7.0	K6.4	K6.3	K6.2
VFC10	K7.4	K7.3	K7.2	K7.1

Address: *BlitterBaseAddress* + *Blitter2DFilterOffset* + 0x30 (VFC1), 0x34 (VFC2), 0x38 (VFC3), 0x3C (VFC4), 0x40 (VFC5), 0x44 (VFC6), 0x48 (VFC7), 0x4C (VFC8), 0x50 (VFC9), 0x54 (VFC10)

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: These registers store the vertical sample rate converter (VSRC) coefficients. There are eight subfilters for the VSRC; each subfilter has five 8-bit coefficients. Coefficient *j* of subfilter *i* (subposition *i*) is designated K*i.j*[7:0] and the format is S1.6 (1.0 is encoded as 0x40). The corresponding filter coefficients structure (40 bytes) must be aligned on a 128-bit word boundary when stored in the local memory.

49 Main and auxiliary displays

49.1 Overview

The display processors form a high quality scaling engine to provide video display processing. They take video lines in 4:2:2 raster (R) format (one buffer) or 4:2:0 macroblock (MB) format (two buffers) from external memory. 4:2:2 MB input format is also supported. Images are read from memory in either frame format or field format independently programmable for chroma and luma. This allows frame or field based vertical filtering.

The display processors can perform zoom out up to a factor of four (horizontally and vertically) and unlimited zoom in. They provide YCbCr digital output in 4:4:4 or 4:2:2 10-bit format.

Sub-pixel and sub-line resolution pan/scan components can be specified within the source image to determine the start of the image to be displayed from the display buffer.

49.2 Fully programmable horizontal sample rate converters

The horizontal sample rate converters include the following features:

- 8 taps interpolation luma/chroma filters
- time varying coefficients using a polyphase filtering technique,
- 32 phases and 10 bit-coefficients (value -512 to 511) stored in RAM, that is, 16 x 8 x 8 + 3 x 32 bits for luma (the same for chroma). See Chapter 50: *Main and auxiliary display registers on page 506* for details,
- horizontal linear resizing of video,
- linear resizing: continuous resizing of video horizontally, by a programmable factor from 1/4 up to x 8192 with 2^(-13) step,
- equidistant fractional pixel position down to 1/32 of pixel,
- horizontal pan down to 1/32 pixel,
- 150 Mpixel/s maximum pixel generation rate.

49.3 Fully programmable vertical sample rate converters

Applications: zoom in/out, letter box resizing chroma upsampling (4:2:0 to 4:2:2).

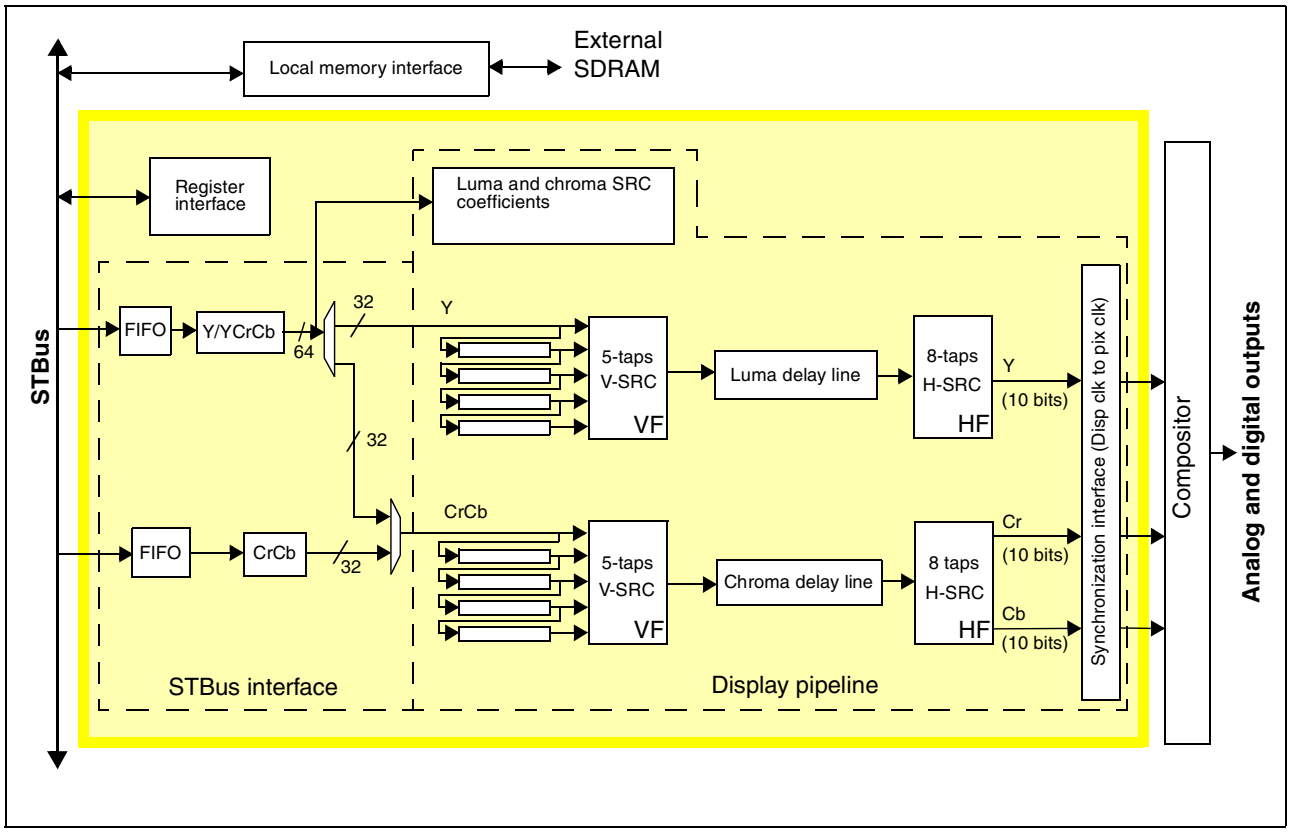
The vertical sample rate converters include the following features:

- five taps interpolation luma/chroma filters,
- time varying coefficients using a polyphase filtering technique,
- 32 phases and 10 bit-coefficients stored in RAM, that is, 16 x 5 x 8 + 2*32 bits for luma, same for chroma.
- equidistant fractional line position down to 1/32 of line,
- field or frame based interpolation modes,
- vertical scan down to 1/32 line,
- vertical linear resizing of video,
- linear resizing: continuous resizing of video vertically, by a programmable factor from 1/4 and up to x8192 with 2^(-13) step,
- 150 Mpixel/s maximum pixel generation rate.

49.4 Block diagram

Figure 151 provides a functional system overview of the HD and ID display block.

Figure 151: Graphics and video generic compositor overview



There are four parallel vertical filters and two 8-tap chroma filters after the chroma delay lines.

49.5 Linear motion upconverter (LMU)

The LMU de-interlaces standard-definition pictures. When enabled, it generates twice as many lines as it receives, calculating every other line from the incoming field and the last two fields. The original lines pass through unchanged, and the last two fields are stored with motion estimation values in the LMU chroma and luma buffers. It interfaces to an external BIST controller for testing of the internal line memories.

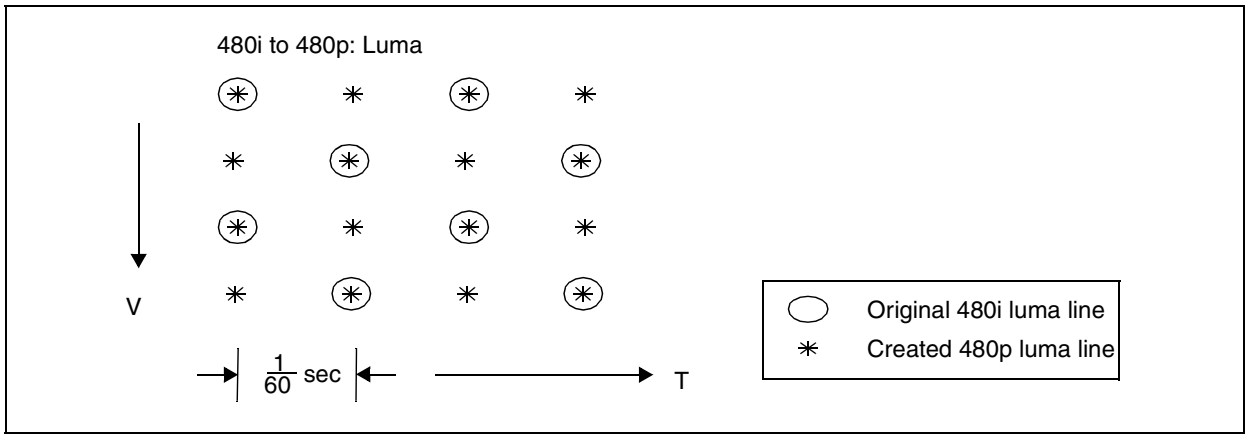
The LMU uses a motion-adaptive, spatial/temporal de-interlacing algorithm controlled by external registers. The luma and chroma blocks are independent but share LMU_BPPL and LMU_VINL register values.

The input line size must not exceed 720 pixels due to the limited size of the internal line memories. The de-interlacing algorithm is only required for image sizes up to 720 x 480 interlaced, that is CCIR601 resolution. These images can originate from the MPEG decoding process or as input from the standard-definition pixel port.

Luminance and chrominance signals are converted by the LMU de-interlacing algorithm, using information from three adjacent source fields to generate each progressive output frame.

Figure 152 illustrates the vertical/temporal relationship of lines before and after the luma de-interlacing process.

Figure 152: Vertical/temporal input and output line relationships



In the above example, the LMU block calculates extra luma lines to generate a progressive 480 line 60 frame/s display from an interlaced 480 line 30 frame/s input. For each incoming field, lines corresponding to that field are passed right through the block. Lines corresponding to the previous field must be estimated by the LMU algorithm. Four different algorithms can be selected:

- Motion adaptive de-interlacing, used for video sources. Missing lines are interpolated based on the amount of motion estimated from the previous field and frame.
- Missing lines are obtained from the following field, used for film sources.
- Missing lines are obtained from the previous field, used for film sources.
- Missing lines are blanked, simulating an interlaced display.

The associated film-mode detector accumulates frame differences over an entire field and passes to the micro two 8-bit values representing the frame difference of both the luma and chroma components. For optimum motion sensitivity in small regions of the picture and noise rejection, the data is collected as follows:

1. Pixels are partitioned in groups of 16.
2. The absolute value of the frame difference is accumulated for each 16 pixel group.
3. The largest accumulated value over all groups in the field is stored in a register and is available to the host processor.

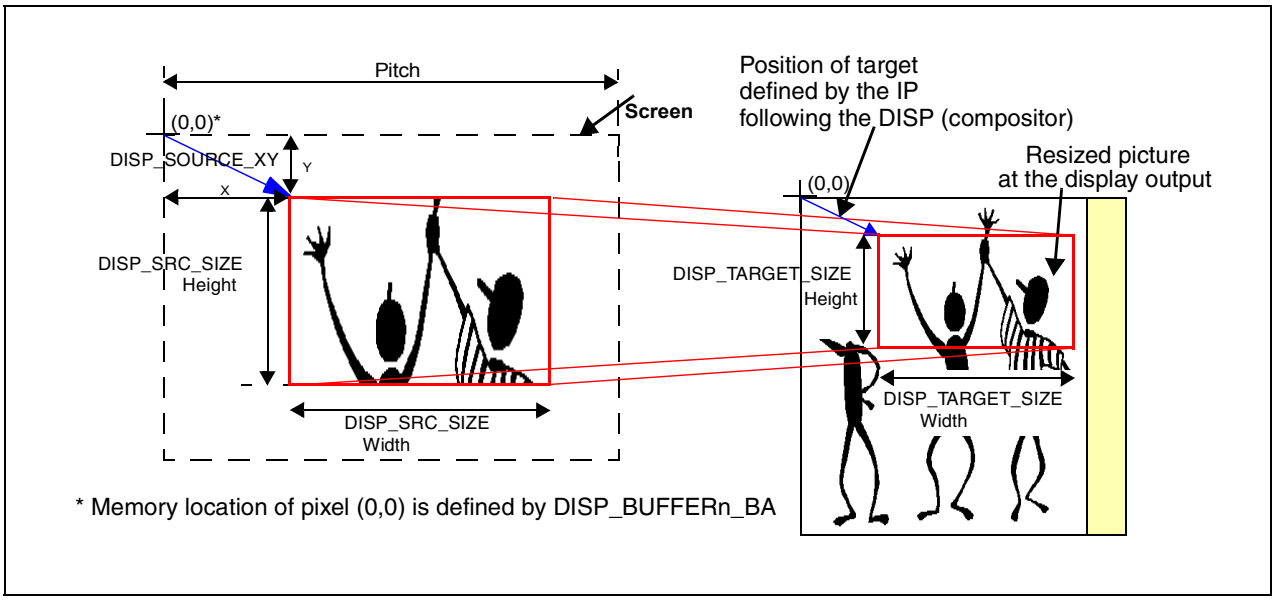
The LMU can be activated independently for luma and chroma.

49.6 Functional description

49.6.1 Memory addressing of the source picture

The source picture to be resized and sent to video output is stored in external memory. It is defined by the base address of the first pixel of picture stored in memory (DISP_LUMA_BA and DISP_CHR_BA). Two base addresses are needed when the picture is stored in 4:2:0 MB (or 4:2:2 MB) format (for luma and chroma) and only one base address for 4:2:2 R format (luma and chroma are stored together in the memory). The DISP_PMP register defines the pitch. This is the distance in memory between two vertically adjacent samples used for vertical filtering when 4:2:2 R input format is used, and line length in pixel unit when 4:2:0 MB or 4:2:2 MB input format is used. By changing pitch value, field or frame filtering is done in vertical SRC when 4:2:2 R input format is used. Inside this complete picture upper left corner of rectangular part to be resized is defined by DISP_LUMA(CHR)_XY and its size by DISP_LUMA(CHR)_SIZE (see Figure 153). This is the only part of picture read from external memory and is considered as 'source' in the following sections.

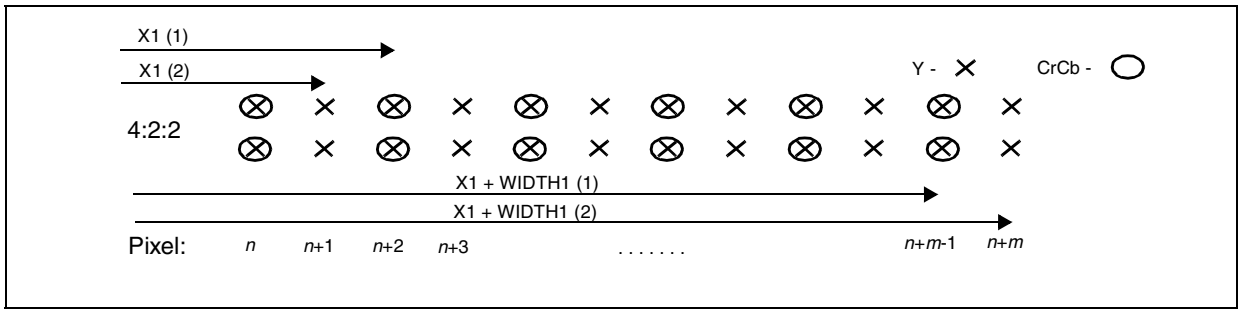
Figure 153: Source and target definition



When the 4:2:2 R input format is used, the source picture position is defined by DISP_LUMA_XY.X1 and .Y1, and size by DISP_LUMA_SIZE.WIDTH1 and .HEIGHT1. Both position and size are given with pixel precision. This means that for odd values of X1 and/or (X1+WIDTH1), pixels with only luma information are pointed (see Figure 154). In that case (an odd number of pixels from (0,0) to top left corner of source) first existing chroma sample **right** to border defined by X1 is taken as first horizontal chroma sample. In the example of Figure 154, for both values of X1(1) and X1(2), chroma corresponding to pixel number *n*+2 is taken as first left chroma sample of the source picture. In a similar manner, for last horizontal chroma sample when X1 + WIDTH1 is odd, chroma from pixel left to one defined by X1+WIDTH1 is taken as the last horizontal chroma sample in the source picture. In the example of Figure 154, for both values

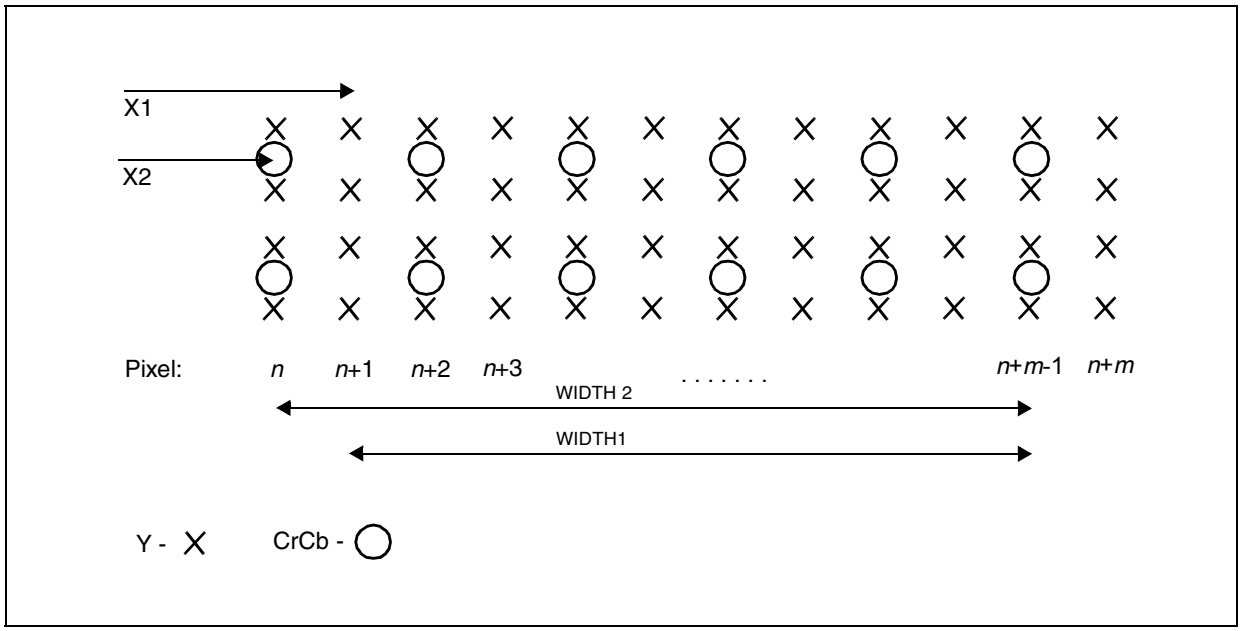
X1+WIDTH1 (1) and X1+WIDTH1 (2), chroma from pixel number *n*+*m*-1 is taken as the last horizontal chroma sample within the source.

Figure 154: Chroma sampling for odd-number-sized sources (example)



When the 4:2:0 MB or 4:2:2 MB input format is used, first and last luma and chroma samples are defined independently. Luma samples of source are defined by X1, Y1, WIDTH1 and HEIGHT1 (registers DISP_LUMA_XY and DISP_LUMA_SIZE) comparing to top-left luma sample of complete picture defined by DISP_LUMA_BA. In the same manner, chroma samples of the source picture are defined by X2, Y2, WIDTH2 and HEIGHT2 (registers DISP_CHR_XY and DISP_CHR_SIZE) comparing to top left luma sample of complete picture defined by DISP_CHR_BA. Figure 155 shows an example of this.

Figure 155: 4:2:0 input format (example)

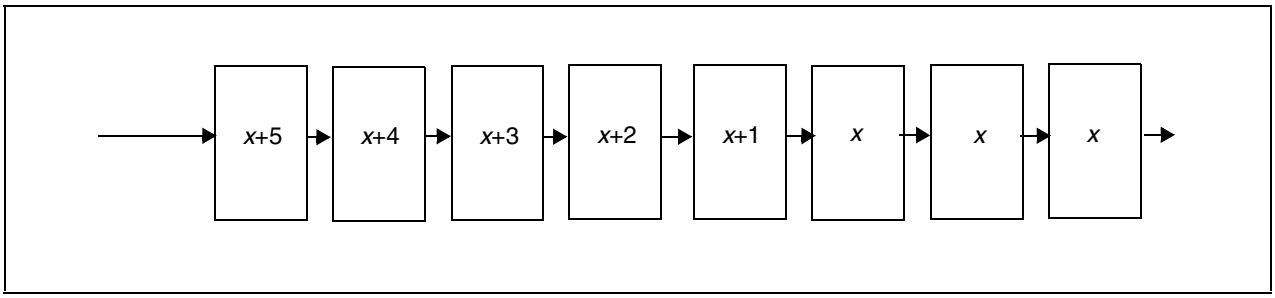


49.6.2 Processing of first and last pixels

In order to have a clean picture on vertical and horizontal borders, some special configurable processing is performed. If the first (or last) pixel of source picture to be resized is not the first (or last) pixel of the whole picture stored in memory, it is possible to take into account pixels that are before (or after) that first (or last) pixel of the source. In this case, DISP_LUMA(CHR)_XY and DISP_LUMA(CHR)_SIZE should be programmed larger by a few pixels then the picture we want to resize. In other words, DISP_LUMA(CHR)_XY and DISP_LUMA(CHR)_SIZE define part of a picture that are loaded in filters (accessed from memory). DISP_LUMA(CHR)_V(H)SRC bits FYPR, FYLR, FCPR and FCLR define how many times the first/last pixel/line of that picture is

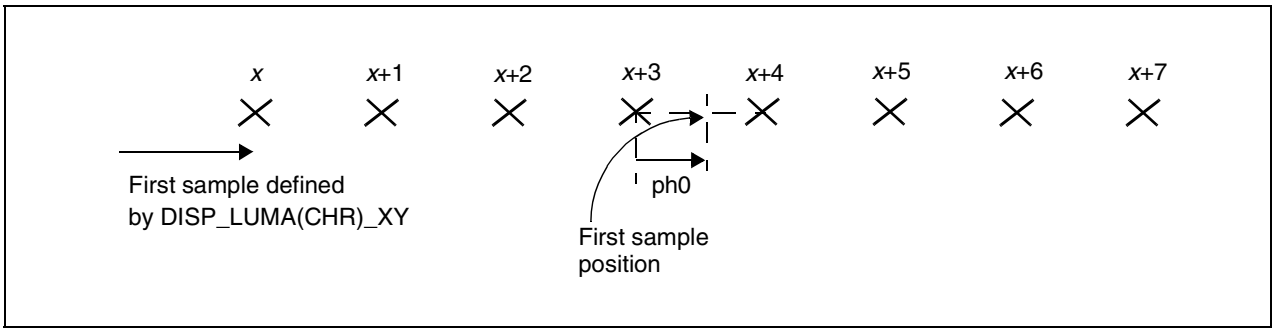
repeated in the horizontal/vertical source filter. Figure 156 shows an example of horizontal filter initialization where the first sample is repeated two times.

Figure 156: Initiations of horizontal SRC pipeline example (first sample repeated twice)



The horizontal and vertical position of the source picture can be defined with an accuracy of 1/32 of pixel/sample. It is defined by initial phase of polyphase DTO, which is a 13-bit register (DISP_LUMA/C_HSRC) giving 1/2^13 accuracy, but internally rounded to 1/32. Figure 157 shows an example of horizontal pan and scan, where FI(c)pr bits defines 0 repeat of the first sample and where initial phase (ph0) is different from 0. In Figure 157, the initial sample is between samples (*x*+1) and (*x*+2).

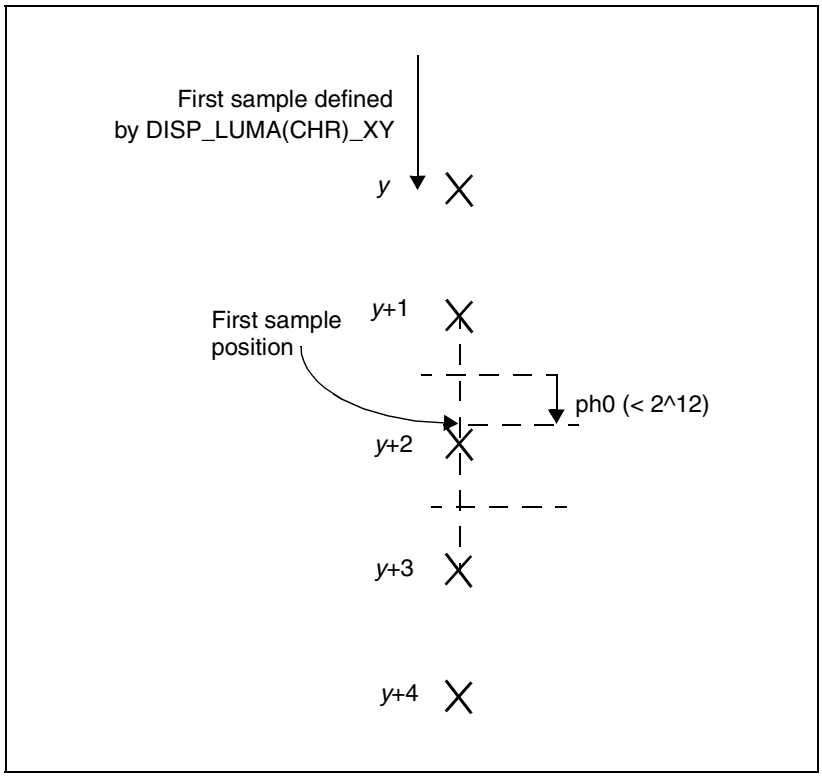
Figure 157: Horizontal pan and scan example



If initial phase is 0 then the first sample is (*x*+3) in previous example and (*x*+1) in the Figure 157 example.

Vertical pan and scan is defined in a similar manner, taking into account that the number of taps is five (an odd number). Where FI(c)lr bits defines repeat of the first sample and where initial phase is 2^12 (1/2 of sample), first sample is (*y*+2). For smaller values of initial phase it is between half distance between *y*+1 and *y*+2 on one side and *y*+2 at the other side. For values greater then 2^12 it is between sample y+2 on one side and the half distance between samples y+2 and y+3 at the other side (see Figure 158):

Figure 158: Vertical pan and scan example



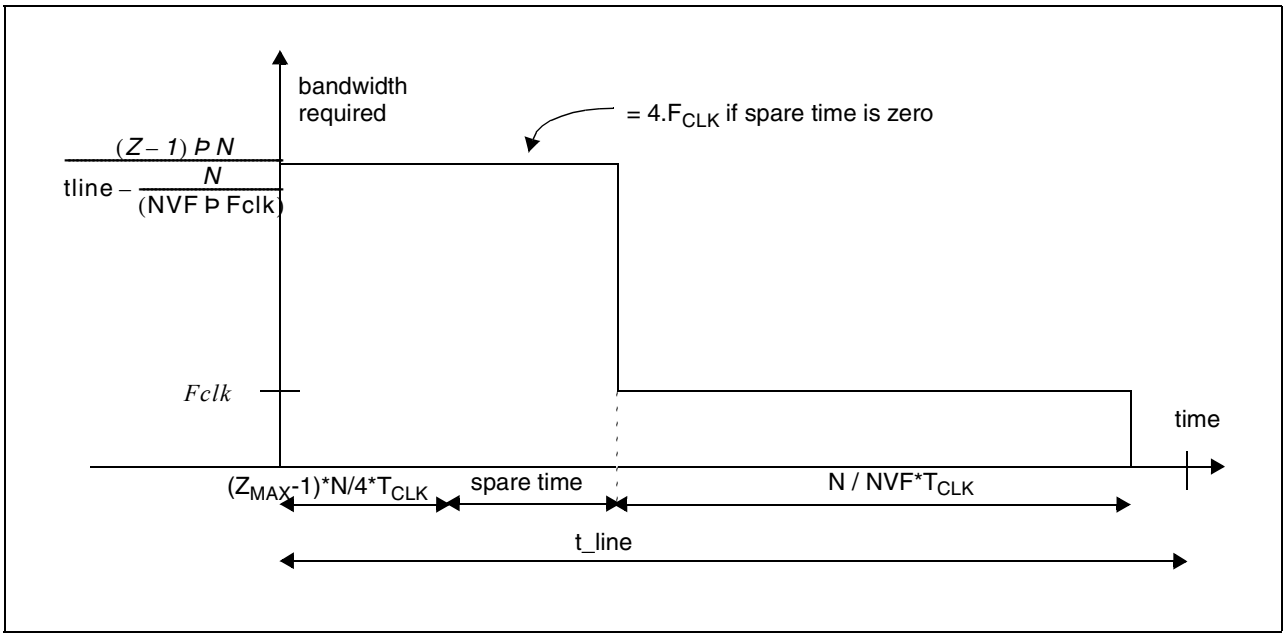
49.6.3 Bandwidth considerations

From the input buffers, luma and chroma samples are separated and sent to delay lines for vertical filtering (up and downsizing); see Figure 151 for block diagram. In HD, possible sources are 4:2:2 R, 4:2:0 MB and 4:2:2 MB formats, 1920 pixels per line maximum (960 for ID version). Each delay line is made of two 960 byte RAMs (two 480 byte RAMS for ID version). Pixels are grouped by four on each address as shown on Figure 160, that is, each delay line is made of two 240*32 bits RAMs (120*32 bits RAMS for ID version). Two RAMs are used to ping-pong write and read operations on a cycle basis, allowing maximum throughput.

49.6.3.1 Operation of the vertical filter and associated delay lines

The vertical filter uses four delay lines and a fifth input directly provided by the STBus interface FIFOs to perform its five tap filter operation. Where zoom requires less than one line fetch per line generated (zoom in), the display can read the same line from memory several times. If zoom requires more than one line fetch per line generated (zoom out), the display fetches some lines while the vertical filter is not operating; filtering occurs while the last needed line is coming in from the STBus.

Figure 159: Vertical filter input requirements



N: number of pixels in the source video line (1920 max in HD, 960 max in ID)

NVF: number of vertical filters: 1 vertical filter

Zmax: maximum zoom out factor rounded to greater integer value (maximum number of lines to be fetched while generating one output line

Fclk, **Tclk**: clock frequency and clock period for pipeline (typical Fclk = 150 MHz)

t_line output (target) line duration: VTGW / Fpel (depends on the screen definition)

VTGW: VTG line width in pixels (full line width, typical 2200 for HD, 858 for NTSC)

Fpel: pixel frequency (typical 74.25 MHz in HD, 13.5 MHz in PAL or NTSC)

We need (Zmax-1)*N/4 cycles to load the delay lines without filtering operation. During this time, (Zmax-1) video lines are loaded in display.

If we have NVF vertical filters, we need N/NVF cycles to filter and generate a video line at the output. During this time one video line is read by the display (last line read is processed in real time).

This means that to be able to work in real time we need to respect:

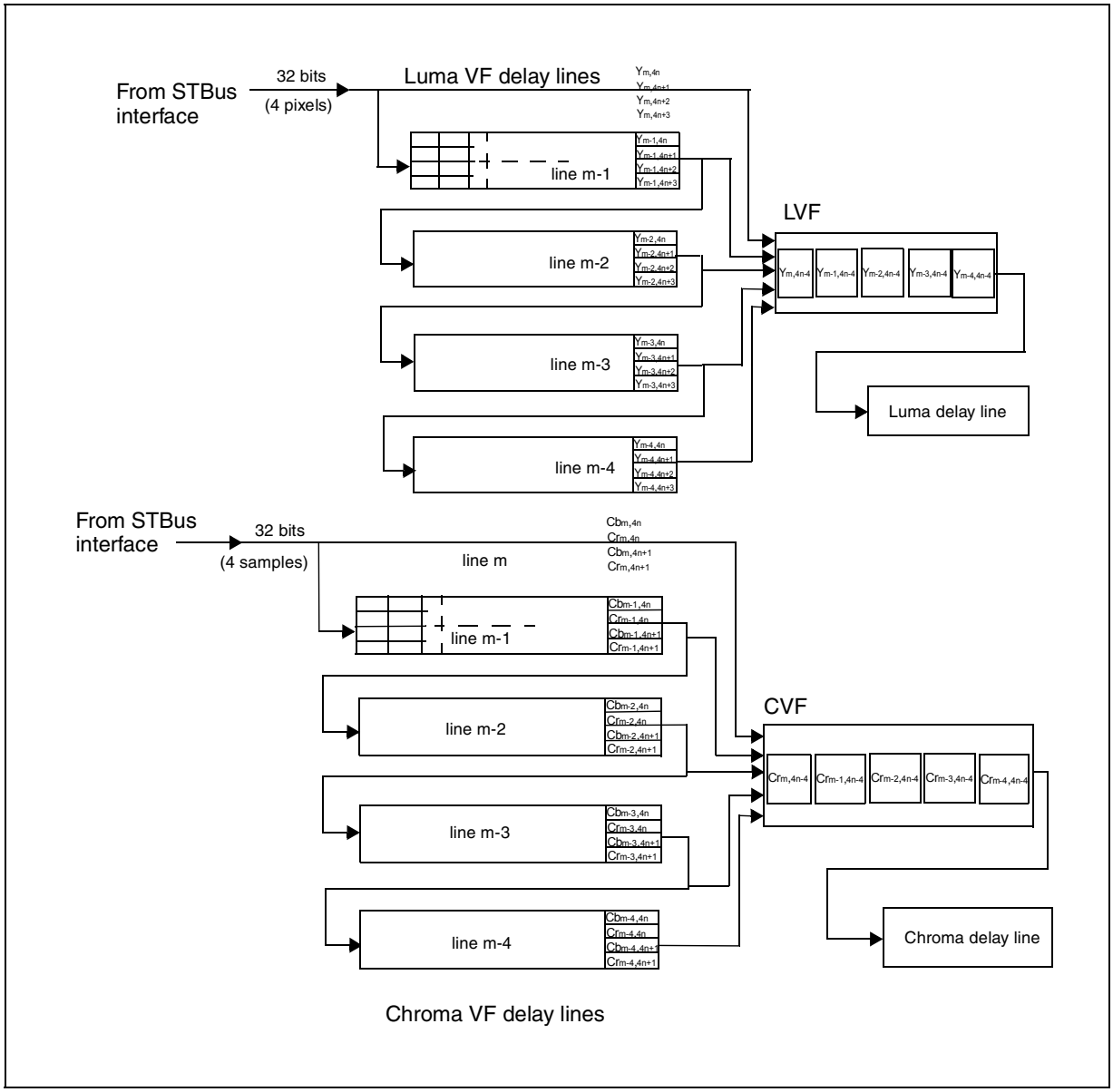
$$\frac{N}{NVF} \cdot T_{ck} + \frac{Zmax - 1}{4} \cdot N \cdot T_{ck} \leq t_{line}$$

So with single VF (NVF =1), we have the maximum zoom out factor:

$$Z_{max} \leq \left(\frac{4.VTGW}{N} \cdot \frac{F_{clk}}{F_{pel}} \right) - 3$$

49.6.4 Vertical filters (VF)

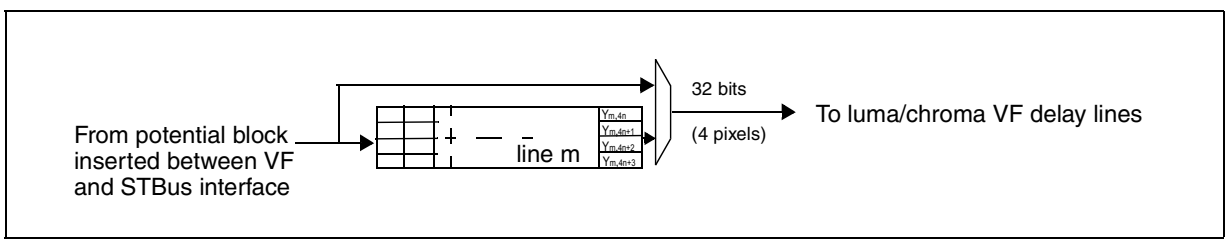
Figure 160: Vertical filters



Chroma samples are stored as they come: 1 sample of Cb, 1 sample of Cr,... (see Figure 160).

Between the delay lines for vertical filtering and VF itself, a small serializer reads the samples four by four, and feeds the VF one sample at a time. This means that the RAMs are read one cycle out of four for vertical filtering.

Figure 161: Luma vertical SRC



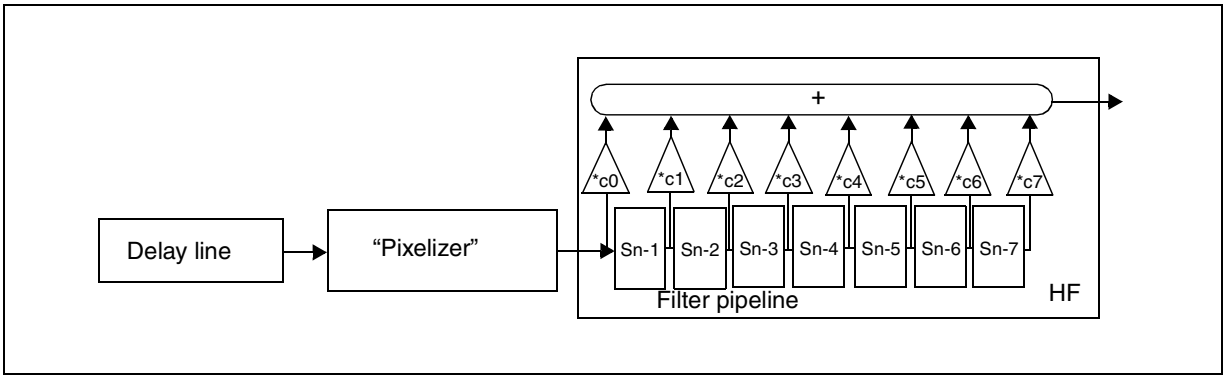
49.6.5 Horizontal SRC

Luma and chroma delay lines after vertical filters are built with one single port RAM each. One video line is written in the delay line from the VF, and read by the horizontal filter (HF) at the same time. The arbiter gives the priority to the HF for read.

Each RAM is 480 x 40 bit SPS - 1920 pixels (10 bits per pixel).

Horizontal filtering is represented in Figure 162:

Figure 162: Horizontal SRC



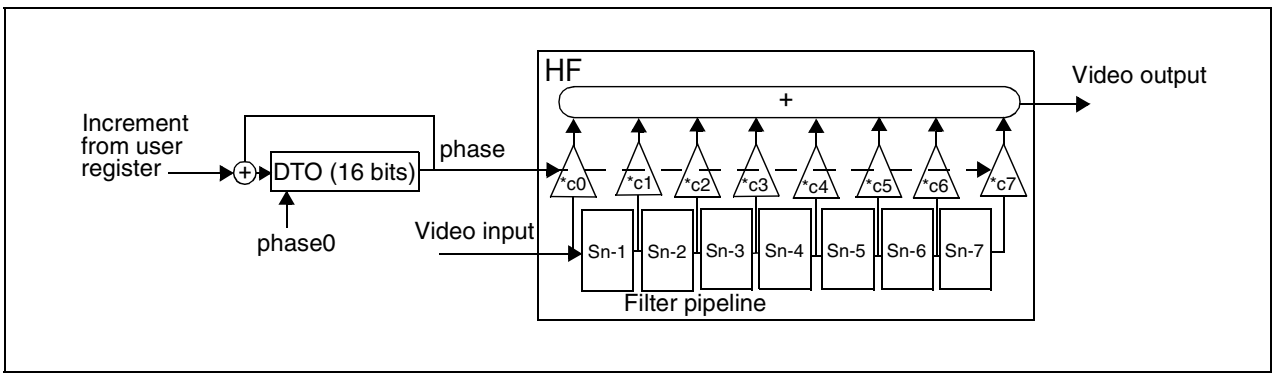
In the worst case (zoom out by four), in luma horizontal filter we should be able to shift the samples in the filter pipeline by up to four and feed up to four new samples from the "pixelizer". Every clock cycle, the pixelizer must be able to read 0 or 4 samples and output 0, 1, 2, 3 or 4 samples to the filter.

There are two horizontal chroma filters. If the output is 4:4:4, the maximum zoom out is two (programmed zoom out by four, plus upsampling by two). In this case, at maximum, two Cr or Cb samples are fed in the filters each PIX1X clock cycle. If 4:2:2 output format is programmed, zoom out by 4 of chroma is possible. That means 4 new chroma samples have to be loaded in filter pipeline in the worst case. In the same time the output of chroma HF has to be generated one cycle out of two of PIX1X clock.

49.6.6 Filter description

All filters are classical polyphase filters. At each clock cycle, an increment value is added to the 13 LSBs of the previous value of DTO, and put as a new value in the DTO register. The value of DTO is used to control the filter pipeline. It is taken from DTO register and 2^7 value is added to that value for rounding. The 3 MSBs of that value give the information on how many new video samples are added into the filter pipeline (0, 1, 2, 3 or 4). The next five bits give the set of coefficients to be used in the filter (from 0 to 31; see Figure 163).

Figure 163: Polyphase filter block diagram

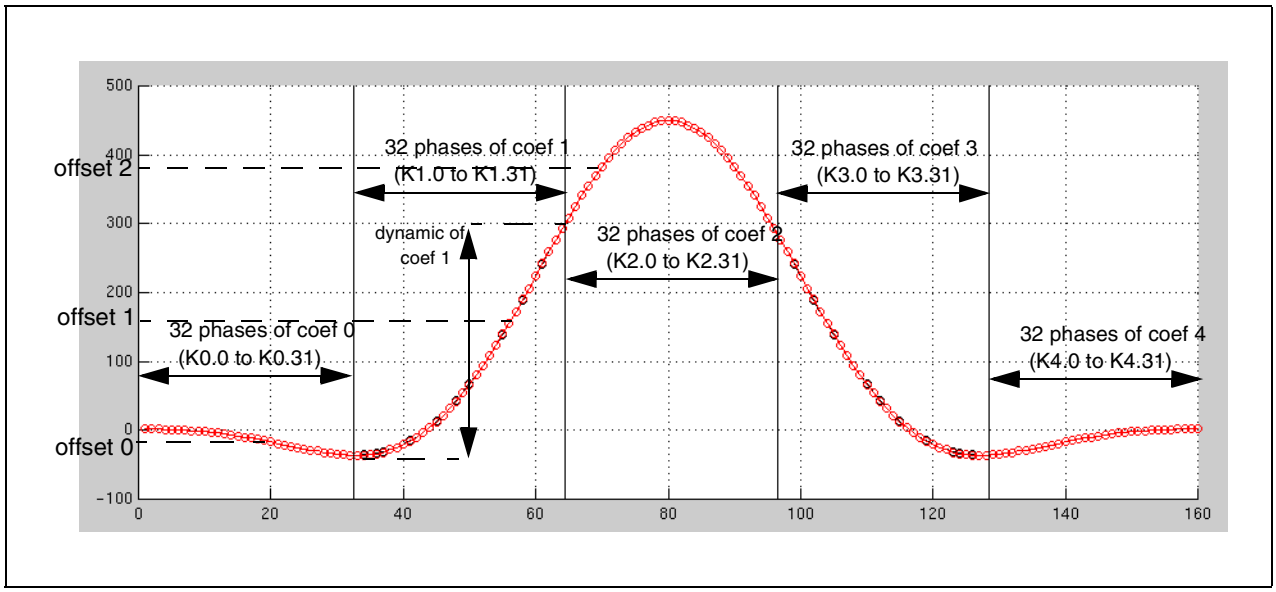


49.6.7 Programming of filter coefficients

An example of five tap polyphase filter (VSRC) set of coefficients is shown in Figure 155. There are 5*32 coefficients on sine curve which is shown. It is easy to see that it is symmetrical curve, so only half of coefficients need to be stored in memory. All 5*32 coefficients are 10-bits values. Nevertheless, 8 or 9 bits are enough for the dynamic of each group of 32 coefficients (32 phases of a coefficient in 5 tap filter). That is why each coefficient is defined by 3 values:

- 8-bit signed value,
- 10-bit offset value, common to all 32 phases (in fact 64 because of symmetry), and
- 1-bit shift value, common to all 32 phases (64 because of symmetry) which means that an 8-bit value is multiplied by 2 or not to cover the dynamic of 32 phases (coef1 set of 32 phases in the above example).

Figure 164: Example: five tap polyphase filter (VSRC) set of coefficients



49.6.8 Programming of source/target size and increment

The increment of vertical luma filter should be $Height_{SOURCE}/Height_{TARGET}$ * 2^13. If the input format is YCbCr 4:2:2, then the increment (and initial phase) for vertical chroma filter must be the same as luma one. If input format is YCbCr 4:2:0 macro block then the increment of vertical chroma filter is the half of vertical luma filter increment because chroma upsampling is done in the same time as filtering (4:2:0 -> 4:2:2).

Increment of horizontal luma filter should be $Width_{SOURCE}/Width_{TARGET}$ * 2^13. For the horizontal chroma filter, the increment should be half of the luma increment because the horizontal upsampling is done in the same time (4:2:2 -> 4:4:4).

The minimum value for source height is 5, for width, 34 (17 samples of chroma).

The increment value and initial phase are used to generate pixels at the output. The DISP_TARGET_SIZE register determines the number of pixels/lines generated. This means, if the increment does not respect the ratio between target and source size, the generation of pixels will be stopped after target size is reached, or the last pixel is repeated to reach the target size. In the case where it should be stopped earlier then all pixels of source picture are consumed, VF and HF continue to send the requests to previous block until all pixels of the source picture are sent (until the last pixel/line is received). This is useful where the de-interlacer operates before the vertical filters because it needs to clean all the pixels of previously processed line from its FIFOs.

49.6.9 Nonlinear zoom

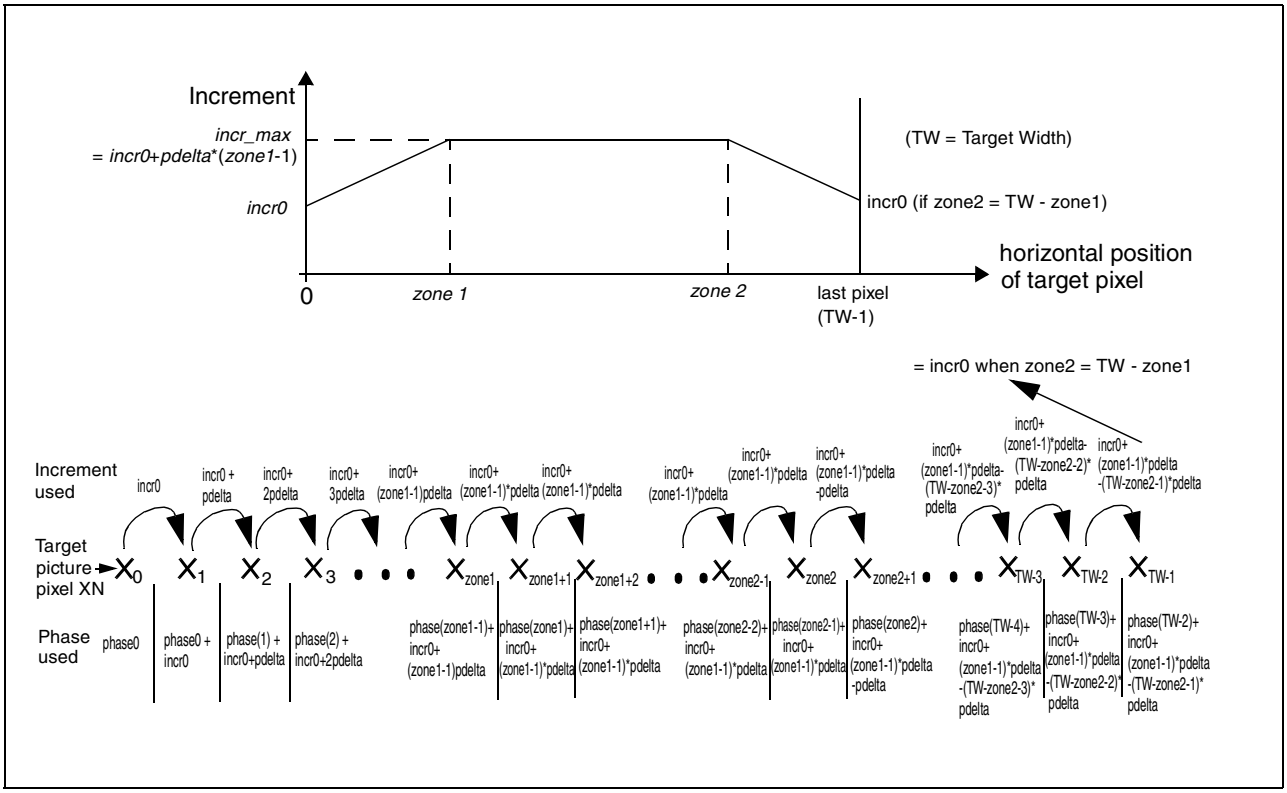
A typical application of nonlinear zoom is to display a 4:3 picture on a 16:9 screen, with minimum deformation of the central part of the picture and greater deformation of the left and right borders. This feature is implemented by linearly changing the increment (zoom factor) **only** in horizontal SRC at the beginning and at the end of the screen as shown in Figure 165.

The first generated pixel uses *phase0* to find the corresponding coefficients; the second uses *phase0* + *incr0*; the third uses *phase1* + *incr0* + *pdelta* and so on. *phase0* and *incr0* are the initial phase and increment as defined by registers DISP_CHR_HSRC and DISP_LUMA_HSRC (defined for nonlinear zoom). *pdelta* is the 'increment of the increment' defined by register DISP_PDELTA. Zones where nonlinear zoom is applied are defined by pixel/samples zone 1 and zone 2 as shown. This means that the increment increases from pixel/sample 0 to pixel/sample zone 1, and decreases from pixel/sample zone 2 to the last pixel/sample of the target picture.

Note: The maximum value of increment is 2^15 (0x8000), which corresponds to zoom out by 4. If this value is reached before zone1, then incr_max is 2^15 and the increment stops growing. When zone2 is reached, the increment goes down, and at the end of picture it reaches a value smaller then incr0 (when zone2 = TW - zone1). This should be avoided, by programming values of incr0, pdelta and zone1 that do not give incr_max = 2^15 before zone 1.

Zones 1 and 2 are defined by registers DISP_NLZZD_Y and DISP_NLZZD_C respectively for luma and chroma HSRC.

Figure 165: Increment function for nonlinear zoom



49.6.10 Output format

The display can respond to a request from the compositor by sending pixels in either 4:4:4 or 4:2:2 format.

4:4:4 output format means there will be luma and chroma sent back to the compositor request every PIX1X clock cycle.

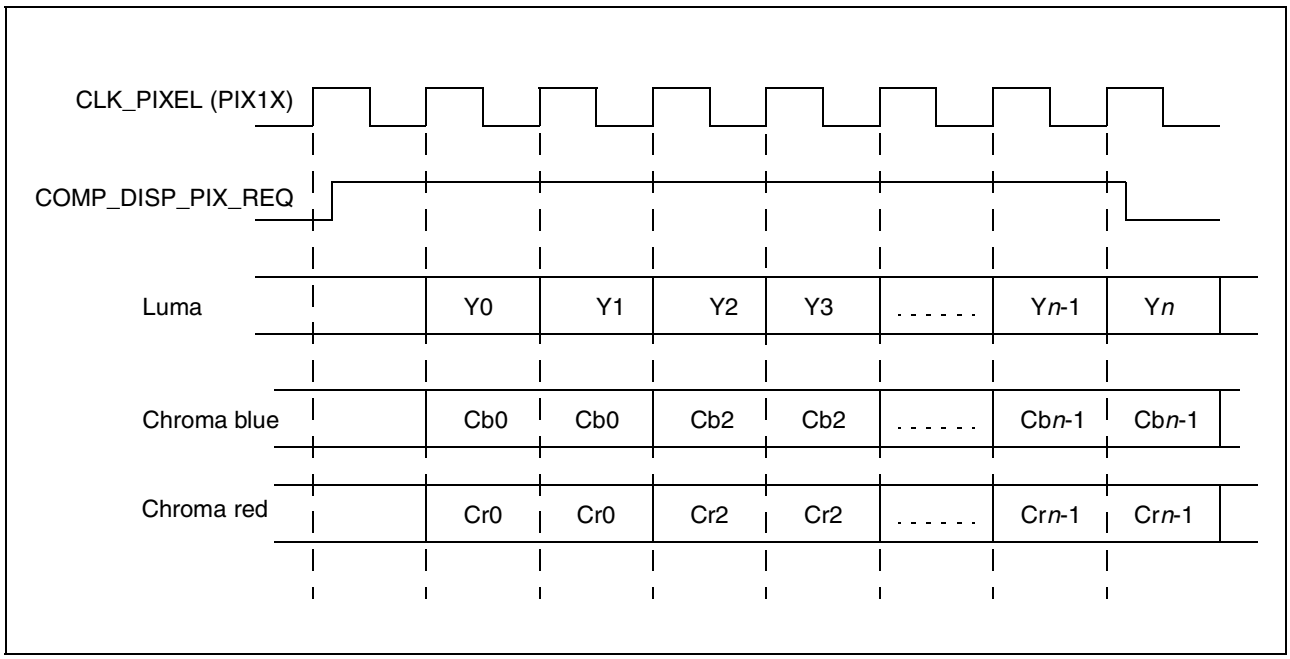
4:2:2 output format means there will be luma sent back every PIX1X clock cycle but chroma is sent back one cycle out of two PIX1X clock cycles.

The number of pixels per line provided to or by the VFC & HFC block is always even.

The first pixel going in the VFC and HFC is always an existing pixel (for example Y0, Cb0, Cr0). The block never starts on a pixel without chroma Cb Cr data.

In Figure 166, variable *n* is odd (for example if we have to display four pixels, *n* = 3). The chroma cb(*n*-1) and cr(*n*-1) pixels are repeated for luma Y*n* pixel.

Figure 166: 4:2:2 output format



49.7 Soft reset

Soft reset clears the pipeline and prepares the block for a new field/frame properly: it reset all FSMs, delay lines, pipeline flags (first/last pixel/line flags...), sample/line counters and stops all STBus requests. No user register is reset by soft reset. There is no specific user register bit for soft reset but it is done on each vertical synchronization active edge (‘automatic’ soft reset).

50 Main and auxiliary display registers

All control registers are double buffered and are accessible through the STBus interface. Registers containing the filter coefficients (DISP_L(C)H(V)F_COEF) are loaded directly from memory and are not double buffered.

Register addresses are provided as either

MainDisplayBaseAddress + offset,

AuxDisplayBaseAddress + offset or

LMUBaseAddress + offset.

The *MainDisplayBaseAddress* is:

0x1900 2000.

The *AuxDisplayBaseAddress* is:

0x1900 3000.

The *LMUBaseAddress* is:

0x1900 4000.

Table 161: Main and auxiliary display processor register summary

Register	Description	Offset	Type
DISP_CTRL	DISP control	0x000	R/W
DISP_LUMA_HSRC	DISP luma horizontal source parameters	0x004	R/W
DISP_LUMA_VSRC	DISP luma vertical source parameters	0x008	R/W
DISP_CHR_HSRC	DISP chroma horizontal source FSM parameters	0x00C	R/W
DISP_CHR_VSRC	DISP chroma vertical source parameters	0x010	R/W
DISP_TARGET_SIZE	DISP target pixmap size	0x014	R/W
DISP_NLZZD_Y	DISP nonlinear zoom - zone definition for luma	0x018	R/W
DISP_NLZZD_C	DISP nonlinear zoom - zone definition for chroma	0x01C	R/W
DISP_PDELTA	DISP nonlinear zoom - increment step definition	0x020	R/W
Reserved	-	0x024 - 0x07C	-
DISP_MA_CTRL	DISP memory access control	0x080	R/W
DISP_LUMA_BA	DISP luma source pixmap memory location	0x084	R/W
DISP_CHR_BA	DISP chroma source pixmap memory location	0x088	R/W
DISP_PMP	DISP pixmap memory pitch	0x08C	R/W
DISP_LUMA_XY	DISP luma first pixel source position	0x090	R/W
DISP_CHR_XY	DISP chroma first pixel source position	0x094	R/W
DISP_LUMA_SIZE	DISP memory source pixmap size (luma)	0x098	R/W
DISP_CHR_SIZE	DISP memory source pixmap size (chroma)	0x09C	R/W
DISP_HFP	DISP horizontal filters pointer	0x0A0	R/W
DISP_VFP	DISP vertical filters pointer	0x0A4	R/W
Reserved	-	0x0A8 - 0x0F8	-
DISP_PKZ	DISP maximum packet size	0x0FC	R/W
DISP_LHF_COEF	DISP luma horizontal filter coefficients	0x100 - 0x188	RO/LLU
DISP_LVF_COEF	DISP luma vertical filter coefficients	0x18C - 0x1E0	RO/LLU
DISP_CHF_COEF	DISP chroma horizontal filter coefficients	0x200 - 0x288	RO/LLU
DISP_CVF_COEF	DISP chroma vertical filter coefficients	0x28C - 0x2E0	RO/LLU

Filter coefficients size: 476 bytes, others 84 bytes - **total: 560 bytes**.

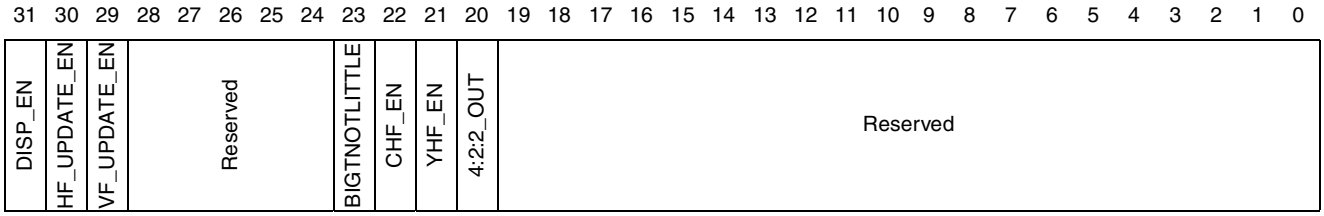
Table 162: LMU register map

Register	Description	Offset	Type
LMU_CTRL	LMU control	0x00	R/W
LMU_LMP	LMU luma buffer start pointer	0x04	R/W
LMU_CMP	LMU chroma buffer start pointer	0x08	R/W
LMU_BPPL	LMU number of block-pairs per line	0x0C	R/W
LMU_CFG	LMU configuration	0x10	R/W
LMU_VINL	LMU number of input lines	0x14	R/W
LMU_MRS	LMU minimum space between STBus requests	0x18	R/W
LMU_CHK	LMU maximum chunk size	0x1C	R/W
LMU_STA	LMU status	0x20	RO
LMU_ITM	LMU interrupt mask	0x24	R/W
LMU_INT_STA	LMU interrupt status	0x28	RO
LMU_AFD	LMU accumulated field difference	0x2C	R/W

50.1 Main and auxiliary display registers

The following registers are duplicated for both main and auxiliary display processors. Their addresses are described as *DisplayBaseAddress* + offset, where *DisplayBaseAddress* is either *MainDisplayBaseAddress* or *AuxDisplayBaseAddress.*

DISP_CTRL DISP control



Address: *DisplayBaseAddress* + 0x000

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Provides the operating mode of the DISP pipe, for the current viewport display.

[31] **DISP_EN**: DISP bloc enable[a]
0: DISP is disabled (no output is generated) 1: DISP is enabled

[30] **HF_UPDATE_EN**: Horizontal filter update enable
0: The coefficients for the H filters are not loaded
1: The coefficients for the H filters are updated from memory after next vsync
This bit is autocleared after it is taken into account (on next Vsync)

[29] **VF_UPDATE_EN**: Vertical filter update enable
0: The coefficients for the V filters are not loaded
1: The coefficients for the V filters are updated from memory after next vsync
This bit is autocleared after it is taken into account (on next VSYNC)

[28:24] **Reserved**

[23] **BIGTNOTLITTLE**: Bitmap
0: Little endian bitmap 1: Big endian bitmap[b]

[22] **CHF_EN**
0: Chroma HF is disabled (coefficients are 0001 0000 for all DTO phases)
1: Chroma HF is enabled (coefficients taken from DISP_CHF_COEF registers)[c]

[21] **YHF_EN**
0: Luma HF is disabled (coefficients are 0001 0000 for all DTO phases)
1: Luma HF is enabled (coefficients taken from DISP_LHF_COEF registers)

[20] **4:2:2_OUT**
0: 4:4:4 output format (luma and chroma output each cycle of PIX1X clock)
1: 4:2:2 output format (chroma output only one cycle out of two of PIX1X clock)

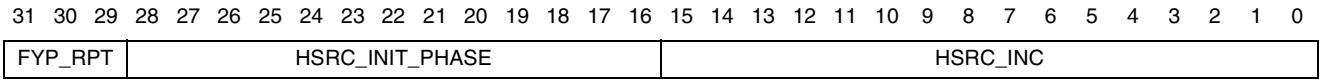
[19:0] **Reserved**

a. This bit is taken into account after next vsync active edge. All STBus requests are stopped properly and video pipeline is purged. If power down is to be done (clock off) then it should be done after vsync active edge following this one.

b. This concerns the video picture pixmap loaded from memory when 4:2:2 R input forma is selected.

c. This feature is needed only for horizontal SRC and not for vertical, because in the vertical SRC this can be achieved by software (programming coefficients 0 0100 for all 32 phases). In HSRC because of symmetry and **even** number of taps what can be programmed is 0001 1000 and not 0001 0000!

DISP_LUMA_HSRC DISP luma horizontal source parameters



Address: *DisplayBaseAddress* + 0x004

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

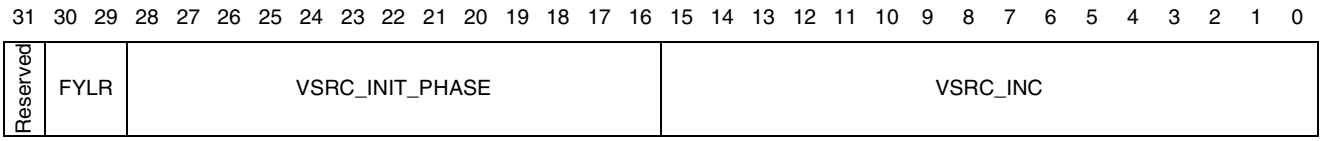
Description: Provides the configuration for the horizontal sample rate conversion of luma samples.

[31:29] **FYP_RPT**: First luma pixel repeat
First pixel is repeated FYP_RPT times when loaded into luma HF. This is a 3-bit nonsigned value with max value of four (1xx values are taken as four - 100). For example:
001: First pixel is repeated once (this means for the first HF output sample generation there are two occurrences of first pixel in HF pipe)

[28:16] **HSRC_INIT_PHASE**
The horizontal sample rate converter state-machine initial phase, 0.13 format.

[15:0] **HSRC_INC**
The horizontal sample rate converter state-machine increment, in 3.13 format. Maximum value is 100.0000 0000 0000 0. Any value 1xx.xxxx xxxx xxxx x is taken as 100.0000 0000 0000 0

DISP_LUMA_VSRC DISP luma vertical source parameters



Address: *DisplayBaseAddress* + 0x008

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Provides the configuration for the vertical sample rate converter.

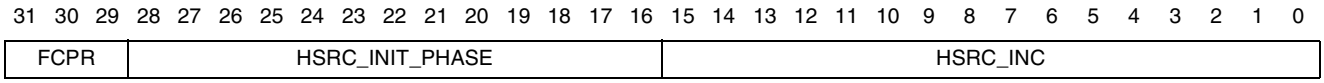
[31] **Reserved**

[30:29] **FYL_RPT**: First luma line repeat
First line is repeated FYL_RPT times when loaded into luma VF. This is 2-bit nonsigned value with a maximum value of 2 (11 value is taken as two - 10). For example:
11: First line is repeated twice (this means for the first VF output line generation there are three occurrences of the first line in the VF pipe)

[28:16] **VSRC_INIT_PHASE**
The vertical sample rate converter state-machine initial phase, 0.13 format.

[15:0] **VSRC_INC**
The vertical sample rate converter state-machine increment, in 3.13 format. Maximum value is 100.0000 0000 0000 0. Any value 1xx.xxxx xxxx xxxx x is taken as 100.0000 0000 0000 0

DISP_CHR_HSRC DISP chroma horizontal source FSM parameters



Address: *DisplayBaseAddress* + 0x00C

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

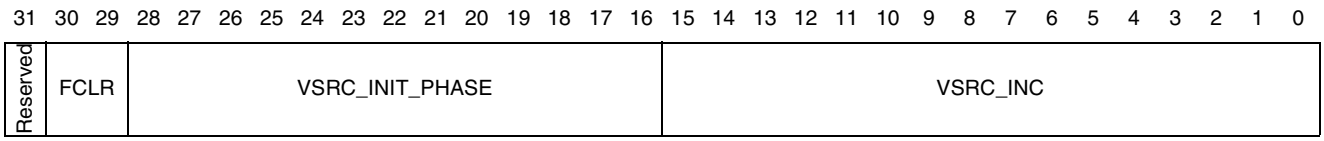
Description: The DISP horizontal chroma sample rate converter's final state machine parameters register provides the configuration for the horizontal sample rate converting of chroma components.

[31:29] **FCP_RPTR**
First chroma pixel is repeated FCP_RPT (First chroma pixel repeat) times when loaded into chroma HF. This is 3 bit nonsigned value with max value of four (1xx values are taken as four - 100). For example:
011: First pixel is repeated three times (this means for the first HF output sample generation there are four occurrences of first chroma pixel in HF pipe)

[28:16] **HSRC_INIT_PHASE**
The horizontal sample rate converter state-machine initial phase, 0.13 format

[15:0] **HSRC_INC**
The horizontal sample rate converter state-machine increment, in 3.13 format. Maximum value is 010.0000 0000 0000 0. Any greater value is taken as 010.0000 0000 0000 0

DISP_CHR_VSRC DISP chroma vertical source parameters



Address: *DisplayBaseAddress* + 0x010

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP chroma vertical sample rate converter's parameters register provides the configuration for vertical sample rate converting of chroma components[1].

[31] **Reserved**

[30:29] **FCLR**
First chroma line is repeated FCLR (First chroma line repeat) times when loaded into chroma VF. This is 2 bit nonsigned value with max value of two (11 value is taken as two - 10). For example:
00: First line is not repeated (this means for the first VF output line generation there is only one occurrence of first chroma line in VF pipe)

[28:16] **VSRC_INIT_PHASE**
The vertical sample rate converter state-machine initial phase, 0.13

[15:0] **VSRC_INC**
The vertical sample rate converter state-machine increment, in 3.13 format. Maximum value is 100.0 0000 0000 0000. Any value 1xx.x xxxx xxxx xxxx is taken as 100.0 0000 0000 0000

1. If 4:2:2 R input format is set, VSRC Increment, VSRC Initial Phase and F_{CPR} must be programmed as luma (DISP_CHR_VSRC and DISP_LUMA_VSRC must have the same value)

DISP_TARGET_SIZE DISP target pixmap size

31 30 29 28 27	26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
Reserved	HEIGHT	Reserved	WIDTH

Address: *DisplayBaseAddress* + 0x014
Type: R/W
Buffer: Double-bank, automatic hardware toggle
Reset: 0
Description: Provides the size of resized picture to be displayed.

[31:27] **Reserved**

[26:16] **HEIGHT**: Pixmap height in lines, defined as the resize resulting number of lines that is to be displayed.

[15:11] **Reserved**

[10:0] **WIDTH**: Pixmap width in pixels

DISP_NLZZD_Y DISP nonlinear zoom - zone definition for luma

31 30 29 28 27	26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
Reserved	LUMA_ZONE1	Reserved	LUMA_ZONE2

Address: *DisplayBaseAddress* + 0x018
Type: R/W
Buffer: Double-bank, automatic hardware toggle
Reset: 0
Description: Provides the borders of nonlinear zoom in the target picture for luma in pixel units; see Section 49.6.9: *Nonlinear zoom* *on page 504*.

[31:27] **Reserved**

[26:16] **LUMA_ZONE1**: Limit between first nonlinear zone and linear zone of the target picture in pixels

[15:11] **Reserved**

[10:0] **LUMA_ZONE2**: Limit between linear zone and second nonlinear zone of the target picture in pixels

DISP_NLZZD_C DISP nonlinear zoom - zone definition for chroma

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CHROMA_ZONE1											Reserved					CHROMA_ZONE2											

Address: *DisplayBaseAddress* + 0x01C

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Provides the borders of nonlinear zoom in the target picture for chroma in output chroma sample units; see Section 49.6.9: *Nonlinear zoom on page 504*.

[31:27] **Reserved**

[26:16] **CHROMA_ZONE1**
Limit between first nonlinear zone and linear zone of the target picture in output chroma sample units.

[15:11] **Reserved**

[10:0] **CHROMA_ZONE2**
Limit between linear zone and second nonlinear zone of the target picture in output chroma sample units.

DISP_PDELTA DISP nonlinear zoom - increment step definition

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDELTA_LUMA																PDELTA_CHROMA															

Address: *DisplayBaseAddress* + 0x020

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Provides the steps used to increment luma and chroma increments in HSRCs for nonlinear zooms; see Section 49.6.9: *Nonlinear zoom on page 504*. If this feature is not used, PDELTA_LUMA and PDELTA_CHROMA should be 0 (linear zoom).

[31:16] **PDELTA_LUMA**
Step with which the luma increment of luma HSRC is incremented within the range [pixel0, pixel luma_zone1] and decremented within the range [pixel luma_zone2, last pixel]. pixel0, and last pixel correspond to first and last chroma sample of target picture.

[15:0] **PDELTA_CHROMA**
Step with which the chroma increment of chroma HSRC is incremented within the range [sample0, sample chroma_zone1] and decremented within the range [sample chroma_zone2, last sample]. Sample0 and last sample correspond to first and last chroma sample of target picture.

DISP_MA_CTRL DISP memory access control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MB_FIELDC	MIN_SPC_BETWEEN_REQS										PIX_LOAD_LINE					COEF_LOAD_LINE					MB_FIELDY	IN_FORMAT				

Address: *DisplayBaseAddress* + 0x080

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Provides memory access control of the DISP.

[31:27] **Reserved**

[26] **MB_FIELDC**
Chroma access mode in macro-block organized frame buffers (YCbCr420 MB and YCbCr422 MB format)
0: Access in frame mode
1: Access in field mode (every other line)

[25:16] **MIN_SPC_BETWEEN_REQS**
Defines the minimum number of STBus cycles between two memory messages (bursts).[a]
The number of cycles between two messages is max('Min Space Between Reqs' + 1, message duration)

[15:11] **PIX_LOAD_LINE**
Defines how many lines STBus plug waits after vsync before sending the first request in the field to load the display pipe. When changing the coefficient set in VSRC and HSRC the pixel request should be sent after new coefficient set is loaded, that is, Pix load line > Coef load line (see above). If this is not the case, then the display pipe will be loaded just after new coefficients load.

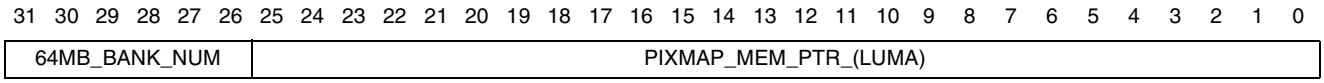
[10:6] **COEF_LOAD_LINE**
Video line number after VSYNC, during which the filter coefficients are loaded via the STBus interface (buffer 1) when V/HFilter update enable is 1. This is a nonsigned value from 0 to 31. Loading of new coefficients is done after Coef Load Line's hsync active edge.

[5] **MB_FIELDY**
Luma access mode in macro-block organized frame buffers for (YCbCr420 MB and YCbCr422 MB format)
0: Access in frame mode
1: Access in field mode (every other line)

[4:0] **IN_FORMAT**: Input format
10010: YCbCr4:2:2 R 0x12
10011: YCbCr4:2:2 MB (or 4:2:2 MB) 0x13
10100: YCbCr4:2:0 MB (or 4:2:0 MB) 0x14

a. For example, in the case of vertical zoom out, four video lines need to be loaded from time to time during the display of one video line (64 µs) which means 4 x 45 requests of 128 bit words of luma (12 messages) should be done and there is 64 µs/133 MHz = 8512 STBus cycles. So, these requests can be spaced by 8512/12 = 709 STBus clock cycles (by putting 709 value in register, the 'peaks' on the STBus can be avoided). The counter used for this purpose is reset by the STBus request

DISP_LUMA_BA DISP luma source pixmap memory location



Address: *DisplayBaseAddress* + 0x084

Type: R/W

Buffer: Double-bank, automatic hardware toggle

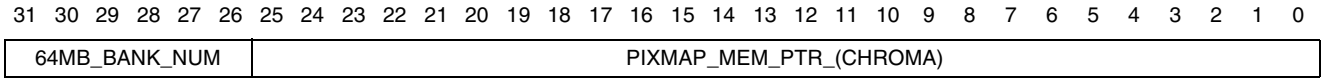
Reset: 0

Description: Gives the location of first Y of the FRAME in 4:2:0 MB and 4:2:2 MB mode or first Cb or Cr of the field in 4:2:2 R mode (of the frame in progressive video mode)**.** It contains the memory location for the first pixel (top-left corner) of the source: of first luma in the frame when 4:2:0 MB and 4:2:2 MB input format is used or first Cb or Cr in the field when 4:2:2 R input format is used. In 4:2:2 R mode, the memory address for pixel [0,0] must be aligned on a 32-bit word address boundary (2 LSBs at 0). In 4:2:X MB mode, the memory address for pixel [0,0] must be aligned on a 2048-bit word address boundary (8 LSBs at 0).

[31:26] **64MB_BANK_NUM**: 64 MByte bank number

[25:0] **PIXMAP_MEM_PTR_(LUMA)**: Pixel map memory pointer (luma)
First Y/YCbCr pixel byte address, in the selected 64 MByte bank (note that the whole bitmap to be displayed must be totally included into the same bank)

DISP_CHR_BA DISP chroma source pixmap memory location



Address: *DisplayBaseAddress* + 0x088

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Gives the location of first chroma sample of the FRAME in 4:2:0 MB and 4:2:2 MB mode. The memory address for pixel [0,0] must be aligned on a 2048-bit word address boundary (8 LSBs at 0).

[31:26] **64MB_BANK_NUM**: 64 MByte bank number

[25:0] **PIXMAP_MEM_PTR_(CHROMA)**: Pixel map memory pointer (chroma)
First chroma byte address, in the selected 64 MByte bank (note that the whole bitmap to be displayed must be totally included into the same bank)

DISP_PMP DISP pixmap memory pitch

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PITCH_VAL															

Address: *DisplayBaseAddress* + 0x08C

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP pixmap memory pitch for the displayed pixmap, as stored in the memory, in 4:2:0 MB and 4:2:2 MB format or of YCrCb in 4:2:2 R format, in bytes.

Note: *The pitch is the distance inside the memory, in bytes, between two vertically adjacent samples.*

In 4:2:0 MB and 4:2:2 MB format, the picture is always stored in FRAME format and the pitch gives the number of luma pixels per line rounded to higher integer number of macroblocks. If the line length in luma pixel unit is 16**n* + *m* (where 0 < *m* < 16), the pitch value should be programmed to 16*(*n*+1) (the four LSBs are always at 0).

In 4:2:2 R format, the pitch value is always used as it is (in field access it defines the distance in bytes between two vertically adjacent samples of the same field and in frame access it defines the distance between two vertically adjacent samples in the frame). The pitch value must be a multiple of four bytes (the two LSBs are always 0).

DISP_LUMA_XY DISP luma first pixel source position

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					Y1											Reserved					X1										

Address: *DisplayBaseAddress* + 0x090

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register gives the position of first source pixel when 4:2:2 R input format is used or first luma sample when 4:2:0 MB or 4:2:2 MB input format is used. Register provides XY location in pixel unit of the first source pixel to be accessed in memory comparing to the top left corner of complete source image (0, 0) defined by DISP_LUMA_BA (see Figure 153: *Source and target definition on page 496*).

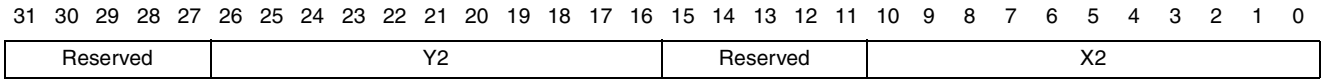
[31:27] **Reserved**

[26:16] **Y1**: First source image line number in complete original picture that should be accessed from memory.

[15:11] **Reserved**

[10:0] **X1**: First source image pixel number in complete original line that should be accessed from memory.

DISP_CHR_XY DISP chroma first pixel source position



Address: *DisplayBaseAddress* + 0x094

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP position of first chroma sample of the source picture that will be loaded and resized in DISP when 4:2:0 MB or 4:2:2 MB is the input format. Register provides XY location in chroma sample unit of the first source chroma sample to be accessed in memory comparing to the top left corner of complete source image (0, 0) defined by DISP_CHR_BA. Example: X2 = 5 means 6th Cr and 6th Cb are the first chroma samples in the line.

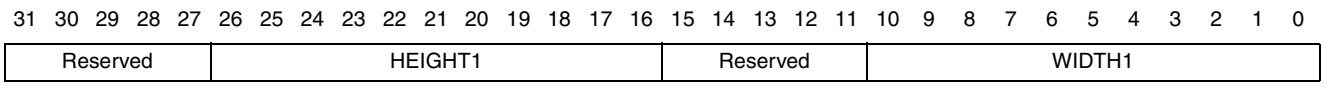
[31:27] **Reserved**

[26:16] **Y2**
First source image chroma line number in complete original picture that should be accessed from memory

[15:11] **Reserved**

[10:0] **X2**
First source image chroma pixel number in complete original line that should be accessed from memory

DISP_LUMA_SIZE DISP memory source pixmap size (luma)



Address: *DisplayBaseAddress* + 0x98

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP memory source pixmap size register provides the size of the source pixmap in pixel unit (part of complete source picture to be loaded from memory) when 4:2:2 R input format is used or pixmap size in luma pixel unit when 4:2:0 MB or 4:2:2 MB format is used. Minimum allowed value of WIDTH1 is 34. Minimum allowed value of HEIGHT1 is 5.

[31:27] **Reserved**

[26:16] **HEIGHT1**
Pixmap height, in lines, being defined as the number of video lines that must be read from memory

[15:11] **Reserved**

[10:0] **WIDTH1**: Pixmap width in pixels

DISP_CHR_SIZE DISP memory source pixmap size (chroma)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					HEIGHT2											Reserved					WIDTH2										

Address: *DisplayBaseAddress* + 0x09C

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP memory source pixmap size register provides the size of the source pixmap in chroma sample unit (part of complete source picture to be loaded from memory) when 4:2:0 MB or 4:2:2 MB input format is used. Minimum allowed value of HEIGHT2 is 5. Minimum allowed value of WIDTH2 is 16. Example: WIDTH2 = *n* means there are *n* Cb and *n* Cr to be read!

[31:27] **Reserved**

[26:16] **HEIGHT2**
Pixmap height in chroma lines, being defined as the number of chroma lines that must be read from memory.

[15:11] **Reserved**

[10:0] **WIDTH2**: Pixmap width in chroma samples

DISP_HFP DISP horizontal filters pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64MB_BANK_NUM						H_FILTER_PTR																						Reserved			

Address: *DisplayBaseAddress* + 0x0A0

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Contains a memory pointer to the set of filter coefficients that must be used for the horizontal sample rate converters (Luma and Chroma).

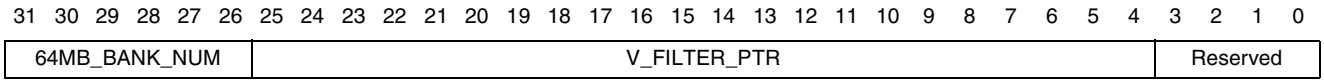
The coefficients are loaded only if DISP_CTRL[30]= 1 (HFilter update enable) after the next vertical synchronization pulse.

[31:26] **64MB_BANK_NUM**: 64 MByte bank number

[25:4] **H_FILTER_PTR**
The four LSBs address bits are “don’t care”, because **the filter coefficients structure must be aligned on a 128-bit word boundary**
Memory location where to retrieve the filter coefficients (35*2 32-bit words that must be fully contained in the specified bank). First 35 32-bit words are luma coefficients, then next thirty-five 32-bit words are chroma coefficients for HSRC.

[3:0] **Reserved**

DISP_VFP DISP vertical filters pointer



Address: *DisplayBaseAddress* + 0x0A4

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Contains a memory pointer to the set of filter coefficients that must be used for the vertical sample rate converters (Luma and chroma).

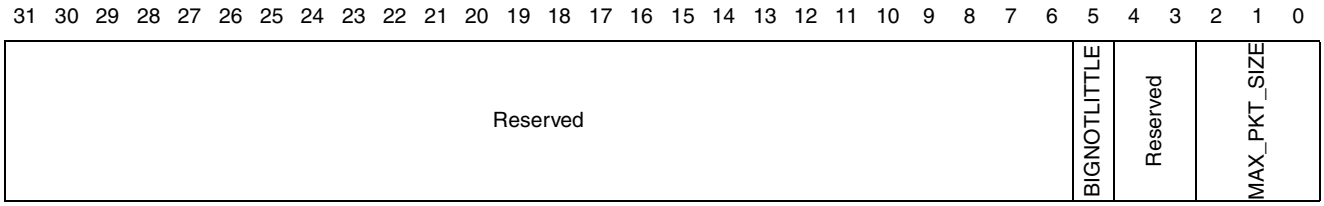
The coefficients are loaded only if DISP_CTRL[29] = 1 (VFilter Update enable) after next vertical synchronization pulse.

[31:26] **64MB_BANK_NUM**: 64 MByte bank number

[25:4] **V_FILTER_PTR**
4 LSBs address bits are “don’t care”, because **the filter coefficients structure must be aligned on a 128-bit word boundary**
Memory location where to retrieve the filter coefficients (22*2 32-bit words that must be fully contained in the specified bank). First 22 32-bit words are luma coefficients, then next twenty-two 32-bit words are chroma coefficients for VSRC.

[3:0] **Reserved**

DISP_PKZ DISP maximum packet size



Address: *DisplayBaseAddress* + 0x0FC

Type: R/W

Buffer: Immediate

Reset: 0

Description: The DISP PKZ register is a 5-bit register for controlling the static parameters of the DISP pipelines.

[31:6] **Reserved**

[5] **BIGNOTLITTLE**: CPU endianess
0: Little endian CPU
1: Big endian CPU[a]

[4:3] **Reserved**

[2:0] **MAX_PKT_SIZE**: Maximum packet size during an STBus transaction
000: Message size (16 packets)
001: 16 STBus words
010: 8 STBus words
011: 4 STBus words
100: 2 STBus words
101: 1 STBus words
others: Reserved

a. This concerns filter coefficients load from memory

DISP_LHF_COEF DISP luma horizontal filter coefficients

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x00	K 7.3 (K 0.29)	K 7.2 (K 0.30)	K 7.1 (K 0.31)	K 7.0
0x04	K 7.7 (K 0.25)	K 7.6 (K 0.26)	K 7.5 (K 0.27)	K 7.4 (K 0.28)
0x08	K 7.11 (K 0.21)	K 7.10 (K 0.22)	K 7.9 (K 0.23)	K 7.8 (K 0.24)
0x0C	K 7.15 (K 0.17)	K 7.14 (K 0.18)	K 7.13 (K 0.19)	K 7.12 (K 0.20)
0x10	K 7.19 (K 0.13)	K 7.18 (K 0.14)	K 7.17 (K 0.15)	K 7.16 (K 0.16)
0x14	K 7.23 (K 0.9)	K 7.22 (K 0.10)	K 7.21 (K 0.11)	K 7.20 (K 0.12)
0x18	K 7.27 (K 0.5)	K 7.26 (K 0.6)	K 7.25 (K 0.7)	K 7.24 (K 0.8)
0x1C	K 7.31 (K 0.1)	K 7.30 (K 0.2)	K 7.29 (K 0.3)	K 7.28 (K 0.4)
0x20	K 6.3 K(1.29)	K 6.2 K(1.30)	K 6.1 K(1.31)	K 6.0
0x24	K 6.7 K(1.25)	K 6.6 K(1.26)	K 6.5 K(1.27)	K 6.4 K(1.28)
0x28	K 6.11 K(1.21)	K 6.10 K(1.22)	K 6.9 K(1.23)	K 6.8 K(1.24)
0x2C	K 6.15 K(1.17)	K 6.14 K(1.18)	K 6.13 K(1.19)	K 6.12 K(1.20)
0x30	K 6.19 K(1.13)	K 6.18 K(1.14)	K 6.17 K(1.15)	K 6.16 K(1.16)
0x34	K 6.23 K(1.9)	K 6.22 K(1.10)	K 6.21 K(1.11)	K 6.20 K(1.12)
0x38	K 6.27 K(1.5)	K 6.26 K(1.6)	K 6.25 K(1.7)	K 6.24 K(1.8)
0x3C	K 6.31 K(1.1)	K 6.30 K(1.2)	K 6.29 K(1.3)	K 6.28 K(1.4)
0x40	K 5.3 K(2.29)	K 5.2 K(2.30)	K 5.1 K(2.31)	K 5.0
0x44	K 5.7 K(2.25)	K 5.6 K(2.26)	K 5.5 K(2.27)	K 5.4 K(2.28)
0x48	K 5.11 K(2.21)	K 5.10 K(2.22)	K 5.9 K(2.23)	K 5.8 K(2.24)
0x4C	K 5.15 K(2.17)	K 5.14 K(2.18)	K 5.13 K(2.19)	K 5.12 K(2.20)
0x50	K 5.19 K(2.13)	K 5.18 K(2.14)	K 5.17 K(2.15)	K 5.16 K(2.16)
0x54	K 5.23 K(2.9)	K 5.22 K(2.10)	K 5.21 K(2.11)	K 5.20 K(2.12)
0x58	K 5.27 K(2.5)	K 5.26 K(2.6)	K 5.25 K(2.7)	K 5.24 K(2.8)
0x5C	K 5.31 K(2.1)	K 5.30 K(2.2)	K 5.29 K(2.3)	K 5.28 K(2.4)
0x60	K 4.3 K(3.29)	K 4.2 K(3.30)	K 4.1 K(3.31)	K 4.0
0x64	K 4.7 K(3.25)	K 4.6 K(3.26)	K 4.5 K(3.27)	K 4.4 K(3.28)
0x68	K 4.11 K(3.21)	K 4.10 K(3.22)	K 4.9 K(3.23)	K 4.8 K(3.24)
0x6C	K 4.15 K(3.17)	K 4.14 K(3.18)	K 4.13 K(3.19)	K 4.12 K(3.20)
0x70	K 4.19 K(3.13)	K 4.18 K(3.14)	K 4.17 K(3.15)	K 4.16 K(3.16)
0x74	K 4.23 K(3.9)	K 4.22 K(3.10)	K 4.21 K(3.11)	K 4.20 K(3.12)
0x78	K 4.27 K(3.5)	K 4.26 K(3.6)	K 4.25 K(3.7)	K 4.24 K(3.8)
0x7C	K 4.31 K(3.1)	K 4.30 K(3.2)	K 4.29 K(3.3)	K 4.28 K(3.4)
0x80	K 0.0	K 1.0	K 2.0	K 3.0
0x84	Reserved / Shift 1 & 6 / Offset K1 & K6		Reserved / Div factor / Shift 0 & 7	Offset K0 & K7
0x88	Reserved / Shift 3 & 4 / Offset K3 & K4		Reserved / Shift 2 & 5	Offset K2 & K5

Address: *DisplayBaseAddress* + 0x100 .. 0x188

Type: RO/Link List update

Buffer: Immediate

Reset: 0

Description: The DISP horizontal luma filter registers 0x00 - 0x7C are 32-bit registers containing four coefficients each (actually four subpositions of a coefficient). Each register coefficient value is given in two's complement format. Three other registers give supplementary information about filter coefficients.

K X.Y:
X: 0 - 7 is the coefficient's order in the 8 tap polyphase filter
Y: 0 - 31 is the interphase order
Two's complement form is used for K X.Y

Div Factor: Filter output division
000: Output of the filter is divided by 256
001: Output of the filter is divided by 512
010: Output of the filter is divided by 1024
011: Output of the filter is divided by 2048
100: Output of the filter is divided by 4096
others: reserved

Offset KX & KY:
DC value of the coefficient KX.N and KY.N, where N is 0 to 31 interphase value

Shift X & Y:
0: Coefficients KX.N and KY.N are directly read from register before adding DC value
1: Coefficients KX.N and KY.N are multiplied by 2 (shifted) before adding DC value

Internally the coefficients are extended to 10 bits as follows:

If '***Shift X & Y***' bit in registers corresponding to the coefficient is 1, the coefficient from register is multiplied by 2 (shifted), otherwise it is just extended to 10 bit two's complement value.

Then an offset is added to that value (two's complement value from registers 0x84 and 0x88 - '***Offset X & Y***').

At the end, in order to have the sum of coefficients equal to 1 for each subposition (0 dB attenuation at 0 Hz frequency), different values of scaling are programmable by the '***Div factor***'.

To summarize, the value of coefficients internally used by the filter is given by:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(8 + Divfactor)}}$$

The coefficients are loaded only if DISP_CTRL[30] = 1 (HFilter update enable).

DISP_LVF_COEF DISP luma vertical filter coefficients

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x8C	K 4.3 (K 0.29)	K 4.2 (K 0.30)	K 4.1 (K 0.31)	K 4.0
0x90	K 4.7 (K 0.25)	K 4.6 (K 0.26)	K 4.5 (K 0.27)	K 4.4 (K 0.28)
0x94	K 4.11 (K 0.21)	K 4.10 (K 0.22)	K 4.9 (K 0.23)	K 4.8 (K 0.24)
0x98	K 4.15 (K 0.17)	K 4.14 (K 0.18)	K 4.13 (K 0.19)	K 4.12 (K 0.20)
0x9C	K 4.19 (K 0.13)	K 4.18 (K 0.14)	K 4.17 (K 0.15)	K 4.16 (K 0.16)
0xA0	K 4.23 (K 0.9)	K 4.22 (K 0.10)	K 4.21 (K 0.11)	K 4.20 (K 0.12)
0xA4	K 4.27 (K 0.5)	K 4.26 (K 0.6)	K 4.25 (K 0.7)	K 4.24 (K 0.8)
0xA8	K 4.31 (K 0.1)	K 4.30 (K 0.2)	K 4.29 (K 0.3)	K 4.28 (K 0.4)
0xAC	K 3.3 K(1.29)	K 3.2 K(1.30)	K 3.1 K(1.31)	K 3.0
0xB0	K 3.7 K(1.25)	K 3.6 K(1.26)	K 3.5 K(1.27)	K 3.4 K(1.28)
0xB4	K 3.11 K(1.21)	K 3.10 K(1.22)	K 3.9 K(1.23)	K 3.8 K(1.24)
0xB8	K 3.15 K(1.17)	K 3.14 K(1.18)	K 3.13 K(1.19)	K 3.12 K(1.20)
0xBC	K 3.19 K(1.13)	K 3.18 K(1.14)	K 3.17 K(1.15)	K 3.16 K(1.16)
0xC0	K 3.23 K(1.9)	K 3.22 K(1.10)	K 3.21 K(1.11)	K 3.20 K(1.12)
0xC4	K 3.27 K(1.5)	K 3.26 K(1.6)	K 3.25 K(1.7)	K 3.24 K(1.8)
0xC8	K 3.31 K(1.1)	K 3.30 K(1.2)	K 3.29 K(1.3)	K 3.28 K(1.4)
0xCC	K 2.3 K(2.29)	K 2.2 K(2.30)	K 2.1 K(2.31)	K 2.0
0xD0	K 2.7 K(2.25)	K 2.6 K(2.26)	K 2.5 K(2.27)	K 2.4 K(2.28)
0xD4	K 2.11 K(2.21)	K 2.10 K(2.22)	K 2.9 K(2.23)	K 2.8 K(2.24)
0xD8	K 2.15 K(2.17)	K 2.14 K(2.18)	K 2.13 K(2.19)	K 2.12 K(2.20)
0xDC	Reserved / Div factor / Shift 2	K 0.0	K 1.0	K 2.16
0xE0	Offset K2 / Shift 1 & 3	Offset K1 & K3 / Shift 0 & 4		Offset K0 & K4

Address: *DisplayBaseAddress* + 0x18C .. 0x1E0

Type: RO/Link List update

Buffer: Immediate

Reset: 0

Description: The DISP vertical luma filter registers 0x8C - 0xD8 are 32-bit registers containing four coefficients each (actually four subpositions of a coefficient). Each register coefficient value is given in two's complement format. Two other registers give supplementary information about filter coefficients.

K X.Y

X: 0 - 4 is the coefficient's order in the 5 tap polyphase filter

Y: 0 - 31 is the interphase order

Two's complement form is used for K X.Y

Div Factor: Filter output division

00: Output of the filter is divided by 64

01: Output of the filter is divided by 128

10: Output of the filter is divided by 256

11: Output of the filter is divided by 512

Offset KX & KY

DC value of the coefficient KX.N and KY.N, where N is 0 to 31 interphase value

Shift X & Y

0: Coefficients KX.N and KY.N are directly read from register before adding DC value

1: Coefficients KX.N and KY.N are multiplied by 2 (shifted) before adding DC value

Internally the coefficients are extended to 10 bits as follows:

If '***Shift X and Y***' bit in registers corresponding to the coefficient is 1, the coefficient from register is multiplied by 2 (shifted), otherwise it is just extended to 10 bit two's complement value.

Then an offset is added to that value (two's complement value from registers 0xDC and 0xE0 - '***Offset X and Y***').

At the end, in order to have the sum of coefficients equal to 4 for each subposition (12 dB gain at 0 Hz frequency), different values of scaling are programmable by '***Div Factor***'. The gain of 4 is due to 8-bit input to 10-bit output conversion.

To summarize, the value of coefficients internally used by the filter is given by:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(6 + Divfactor)}}$$

The coefficients are loaded only if DISP_CTRL[29] = 1 (VFilter update enable).

DISP_CHF_COEF DISP chroma horizontal filter coefficients

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x00	K 7.3 (K 0.29)	K 7.2 (K 0.30)	K 7.1 (K 0.31)	K 7.0
0x04	K 7.7 (K 0.25)	K 7.6 (K 0.26)	K 7.5 (K 0.27)	K 7.4 (K 0.28)
0x08	K 7.11 (K 0.21)	K 7.10 (K 0.22)	K 7.9 (K 0.23)	K 7.8 (K 0.24)
0x0C	K 7.15 (K 0.17)	K 7.14 (K 0.18)	K 7.13 (K 0.19)	K 7.12 (K 0.20)
0x10	K 7.19 (K 0.13)	K 7.18 (K 0.14)	K 7.17 (K 0.15)	K 7.16 (K 0.16)
0x14	K 7.23 (K 0.9)	K 7.22 (K 0.10)	K 7.21 (K 0.11)	K 7.20 (K 0.12)
0x18	K 7.27 (K 0.5)	K 7.26 (K 0.6)	K 7.25 (K 0.7)	K 7.24 (K 0.8)
0x1C	K 7.31 (K 0.1)	K 7.30 (K 0.2)	K 7.29 (K 0.3)	K 7.28 (K 0.4)
0x20	K 6.3 K(1.29)	K 6.2 K(1.30)	K 6.1 K(1.31)	K 6.0
0x24	K 6.7 K(1.25)	K 6.6 K(1.26)	K 6.5 K(1.27)	K 6.4 K(1.28)
0x28	K 6.11 K(1.21)	K 6.10 K(1.22)	K 6.9 K(1.23)	K 6.8 K(1.24)
0x2C	K 6.15 K(1.17)	K 6.14 K(1.18)	K 6.13 K(1.19)	K 6.12 K(1.20)
0x30	K 6.19 K(1.13)	K 6.18 K(1.14)	K 6.17 K(1.15)	K 6.16 K(1.16)
0x34	K 6.23 K(1.9)	K 6.22 K(1.10)	K 6.21 K(1.11)	K 6.20 K(1.12)
0x38	K 6.27 K(1.5)	K 6.26 K(1.6)	K 6.25 K(1.7)	K 6.24 K(1.8)
0x3C	K 6.31 K(1.1)	K 6.30 K(1.2)	K 6.29 K(1.3)	K 6.28 K(1.4)
0x40	K 5.3 K(2.29)	K 5.2 K(2.30)	K 5.1 K(2.31)	K 5.0
0x44	K 5.7 K(2.25)	K 5.6 K(2.26)	K 5.5 K(2.27)	K 5.4 K(2.28)
0x48	K 5.11 K(2.21)	K 5.10 K(2.22)	K 5.9 K(2.23)	K 5.8 K(2.24)
0x4C	K 5.15 K(2.17)	K 5.14 K(2.18)	K 5.13 K(2.19)	K 5.12 K(2.20)
0x50	K 5.19 K(2.13)	K 5.18 K(2.14)	K 5.17 K(2.15)	K 5.16 K(2.16)
0x54	K 5.23 K(2.9)	K 5.22 K(2.10)	K 5.21 K(2.11)	K 5.20 K(2.12)
0x58	K 5.27 K(2.5)	K 5.26 K(2.6)	K 5.25 K(2.7)	K 5.24 K(2.8)
0x5C	K 5.31 K(2.1)	K 5.30 K(2.2)	K 5.29 K(2.3)	K 5.28 K(2.4)
0x60	K 4.3 K(3.29)	K 4.2 K(3.30)	K 4.1 K(3.31)	K 4.0
0x64	K 4.7 K(3.25)	K 4.6 K(3.26)	K 4.5 K(3.27)	K 4.4 K(3.28)
0x68	K 4.11 K(3.21)	K 4.10 K(3.22)	K 4.9 K(3.23)	K 4.8 K(3.24)
0x6C	K 4.15 K(3.17)	K 4.14 K(3.18)	K 4.13 K(3.19)	K 4.12 K(3.20)
0x70	K 4.19 K(3.13)	K 4.18 K(3.14)	K 4.17 K(3.15)	K 4.16 K(3.16)
0x74	K 4.23 K(3.9)	K 4.22 K(3.10)	K 4.21 K(3.11)	K 4.20 K(3.12)
0x78	K 4.27 K(3.5)	K 4.26 K(3.6)	K 4.25 K(3.7)	K 4.24 K(3.8)
0x7C	K 4.31 K(3.1)	K 4.30 K(3.2)	K 4.29 K(3.3)	K 4.28 K(3.4)
0x80	K 0.0	K 1.0	K 2.0	K 3.0
0x84	Reserved / Shift & 6	Offset K1 & K6	Reserved / Div factor / Shift 0 & 7	Offset K0 & K7
0x88	Reserved / Shift 3 & 4	Offset K3 and K4	Reserved / Shift 2 & 5	Offset K2 & K5

Address: *DisplayBaseAddress* + 0x200 .. 0x288

Type: RO/Link List update

Buffer: Immediate

Reset: 0

Description: The DISP horizontal chroma filter registers 0x00 - 0x7C are 32-bit registers containing four coefficients each (actually four subpositions of a coefficient). Each register coefficient value is given in two's complement format. Three other registers give supplementary information about filter coefficients.

K X.Y
X: 0 - 7 is the coefficient's order in the 8 tap polyphase filter
Y: 0 - 31 is the interphase order
Two's complement form is used for K X.Y

Div Factor: Filter output division
000: Output of the filter is divided by 256
001: Output of the filter is divided by 512
010: Output of the filter is divided by 1024
011: Output of the filter is divided by 2048
100: Output of the filter is divided by 4096
others: reserved

Offset KX and KY
DC value of the coefficient KX.N and KY.N, where N is 0 to 31 interphase value

Shift X and Y
0: Coefficients KX.N and KY.N are directly read from register before adding DC value
1: Coefficients KX.N and KY.N are multiplied by 2 (shifted) before adding DC value

Internally the coefficients are extended to 10 bits in the next way:

If '***Shift X and Y***' bit in registers 0x74 and 0x78 corresponding to the coefficient is 1, the coefficient from register is multiplied by 2 (shifted), otherwise it is just extended to 10 bit two's complement value.

Then an offset is added to that value (two's complement value from registers 0x84 and 0x88 - '***Offset X and Y***').

At the end, in order to have the sum of coefficients equal to 1 for each subposition (0 dB attenuation at 0 Hz frequency), different values of scaling are programmable by '***Div Factor***'.

To summarize, the value of coefficients internally used by the filter is given by:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(8 + \overline{Divfactor})}}$$

The coefficients are loaded only if DISP_CTRL[30] = 1 (HFilter update enable).

DISP_CVF_COEF DISP chroma vertical filter coefficients

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x8C	K 4.3 (K 0.29)	K 4.2 (K 0.30)	K 4.1 (K 0.31)	K 4.0
0x90	K 4.7 (K 0.25)	K 4.6 (K 0.26)	K 4.5 (K 0.27)	K 4.4 (K 0.28)
0x94	K 4.11 (K 0.21)	K 4.10 (K 0.22)	K 4.9 (K 0.23)	K 4.8 (K 0.24)
0x98	K 4.15 (K 0.17)	K 4.14 (K 0.18)	K 4.13 (K 0.19)	K 4.12 (K 0.20)
0x9C	K 4.19 (K 0.13)	K 4.18 (K 0.14)	K 4.17 (K 0.15)	K 4.16 (K 0.16)
0xA0	K 4.23 (K 0.9)	K 4.22 (K 0.10)	K 4.21 (K 0.11)	K 4.20 (K 0.12)
0xA4	K 4.27 (K 0.5)	K 4.26 (K 0.6)	K 4.25 (K 0.7)	K 4.24 (K 0.8)
0xA8	K 4.31 (K 0.1)	K 4.30 (K 0.2)	K 4.29 (K 0.3)	K 4.28 (K 0.4)
0xAC	K 3.3 K(1.29)	K 3.2 K(1.30)	K 3.1 K(1.31)	K 3.0
0xB0	K 3.7 K(1.25)	K 3.6 K(1.26)	K 3.5 K(1.27)	K 3.4 K(1.28)
0xB4	K 3.11 K(1.21)	K 3.10 K(1.22)	K 3.9 K(1.23)	K 3.8 K(1.24)
0xB8	K 3.15 K(1.17)	K 3.14 K(1.18)	K 3.13 K(1.19)	K 3.12 K(1.20)
0xBC	K 3.19 K(1.13)	K 3.18 K(1.14)	K 3.17 K(1.15)	K 3.16 K(1.16)
0xC0	K 3.23 K(1.9)	K 3.22 K(1.10)	K 3.21 K(1.11)	K 3.20 K(1.12)
0xC4	K 3.27 K(1.5)	K 3.26 K(1.6)	K 3.25 K(1.7)	K 3.24 K(1.8)
0xC8	K 3.31 K(1.1)	K 3.30 K(1.2)	K 3.29 K(1.3)	K 3.28 K(1.4)
0xCC	K 2.3 K(2.29)	K 2.2 K(2.30)	K 2.1 K(2.31)	K 2.0
0xD0	K 2.7 K(2.25)	K 2.6 K(2.26)	K 2.5 K(2.27)	K 2.4 K(2.28)
0xD4	K 2.11 K(2.21)	K 2.10 K(2.22)	K 2.9 K(2.23)	K 2.8 K(2.24)
0xD8	K 2.15 K(2.17)	K 2.14 K(2.18)	K 2.13 K(2.19)	K 2.12 K(2.20)
0xDC	Reserved / Div factor / Shift 2	K 0.0	K 1.0	K 2.16
0xE0	Offset K2 / Shift 1 & 3	Offset K1 & K3 / Shift 0 & 4		Offset K0 & K4

Address: *DisplayBaseAddress* + 0x28C .. 0x2E0

Type: RO/Link List update

Buffer: Immediate

Reset: 0

Description: The DISP vertical chroma filter registers 0x8C - 0xD8 are 32-bit registers containing four coefficients each (actually four subpositions of a coefficient). Each register coefficient value is given in two's complement format. Two other registers give supplementary information about filter coefficients.

K X.Y
X: 0 - 4 is the coefficient's order in the 5 tap polyphase filter
Y: 0 - 31 is the interphase order
two's complement form is used for K X.Y

Div factor: Filter output division
00: Output of the filter is divided by 64
01: Output of the filter is divided by 128
10: Output of the filter is divided by 256
11: Output of the filter is divided by 512

Offset KX & KY
DC value of the coefficient KX.N and KY.N, where N is 0 to 31 interphase value

Shift X & Y
0: Coefficients KX.N and KY.N are directly read from register before adding DC value
1: Coefficients KX.N and KY.N are multiplied by 2 (shifted) before adding DC value

Internally the coefficients are extended to 10 bits as follows:

If '***Shift X and Y***' bit in registers corresponding to the coefficient is 1, the coefficient from register is multiplied by 2 (shifted), otherwise it is just extended to 10 bit two's complement value.

Then an offset is added to that value (two's complement value from registers 0xDC and 0xE0 - '***Offset X and Y***').

At the end, in order to have the sum of coefficients equal to 4 for each subposition (12 dB gain at 0 Hz frequency), different values of scaling are programmable by '***Div Factor***'. The gain of 4 is due to 8-bit input to 10 bit output conversion.

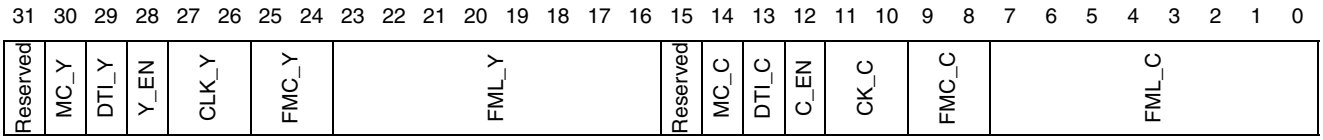
To summarize, the value of coefficients internally used by the filter is given by:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(6 + Div\overline{factor})}}$$

The coefficients are loaded only if DISP_CTRL[29] = 1 (VFilter update enable).

50.2 LMU registers

LMU_CTRL LMU control



Address: *LMUBaseAddress* + 0x00

Type: R/W

Buffer: Double buffered: Vsync

Reset: 0

Description: This register controls the LMU activity.

[31] **Reserved**

[30] **MC_Y**:
0: the LMU checks for motion by comparing individual luma pixels from the current top/bottom field with the co-located luma pixel in the previous top/bottom field.
1: the motion is calculated using a pair of horizontally adjacent luma pixels.
The comparison is always between same type fields: top field compared to previous top field, and bottom field compared to previous bottom field. When using a pixel pair, individual pixels are compared, but the highest motion result is applied for both pixels.

[29] **DTI_Y**:
0: temporal interpolation operates as described in the LMU specification.
1: this bit disables temporal interpolation for the luma pixels.

[28] **Y_EN**:
When set to 1, the luma LMU is enabled; when set to 0, the luma LMU is disabled

[27:26] **CLK_Y**:
Depending on the size of the picture being processed by the LMU, it may not be necessary for the LMU to operate at full speed. These bits allow the LMU operating speed to be controlled so that the STBus bandwidth consumed by the luma LMU can be evenly distributed over picture time.
00: divide by 1, turbo mode (no clock throttling)
01: divide by 2
10: divide by 3
11: divide by 4

[25:24] **FMC_Y**: The film-mode control bits select the method for upsampling to create the interpolated luma lines.
00: motion-adaptive de-interlacing, used for video sources
01: missing lines obtained from following field, used for film source
10: missing lines obtained from preceding field, used for film source
11: blank missing lines, simulates interlaced display

[23:16] **FML_Y**: The LMU's film mode detection algorithm compares the current field to the same field in the previous frame. The comparison starts with line 2 of the luma field and continues until the line number programmed in FML_Y. A separate comparison is made for luma and chroma portions of the picture.

[15] **Reserved**

[14] **MC_C**:
0: the LMU checks for motion by comparing individual chroma pixels from the current top/bottom field with the co-located chroma pixel in the previous top/bottom field.
1: the motion is calculated using a pair of horizontally adjacent chroma pixels.
The comparison is always between same type fields: top field compared to previous top field, and bottom field compared to previous bottom field. When using a pixel pair, individual pixels are compared, but the highest motion result is applied for both pixels. Applying the same motion value to both pixels in a pixel-pair avoids the situation where motion is detected in the Cb pixel but not in the Cr pixel.

[13] **DTI_C**: When set to 1, this bit disables temporal interpolation for the chroma pixels. When set to 0, temporal interpolation operates as described in the LMU specification.

[12] **C_EN**: Chroma LMU enable
0: Disabled 1: Enabled

[11:10] **CK_C**: Depending on the size of the picture being processed by the LMU, it may not be necessary for the LMU to operate at full speed. The bits allow the LMU operating speed to be controlled so that the STBus bandwidth consumed by the chroma LMU can be evenly distributed over picture time.
00: Divide by 1, turbo mode (no clock throttling) 10: Divide by 3
01: Divide by 2 11: Divide by 4

[9:8] **FMC_C**: The film-mode control bits select the method for upsampling to create the interpolated chroma lines.
00: motion-adaptive de-interlacing, used for video sources
01: missing lines obtained from following field, used for film source
10: missing lines obtained from preceding field, used for film source
11: blank missing lines, simulates interlaced display

[7:0] **FML_C**: The LMU's film-mode detection algorithm compares the current field to the same field in the previous frame. The comparison starts with line 2 of the chroma field and continues until the line number programmed in FML_C. A separate comparison is made for luma and chroma portions of the picture.

LMU_LMP LMU luma buffer start pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LMP																								Reserved							

Address: *LMUBaseAddress* + 0x04
Type: R/W
Buffer: Double buffered: Vsync
Reset: 0
Description: Pointer to the start of the memory LMU luma buffer, in 256-byte units.

LMU_CMP LMU chroma buffer start pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP																								Reserved							

Address: *LMUBaseAddress* + 0x08
Type: R/W
Buffer: Double buffered: Vsync
Reset: 0
Description: Pointer to the start of the memory LMU chroma buffer, in 256-byte units.

LMU_BPPL LMU number of block-pairs per line

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									BLK_PAIRS_PER_LINE						

Address: *LMUBaseAddress* + 0x0C
Type: R/W
Buffer: Double buffered: Vsync
Reset: 0
Description: This unsigned register should be programmed with a value that indicates the number of 16-pixel blocks per picture width in the display window.

LMU_CFG LMU configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													TOG	TNB	422

Address: *LMUBaseAddress* + 0x10
Type: R/W
Buffer: Double buffered: Vsync
Reset: 0
Description: This register configures the chroma input format and the type of the input field.

[31:3] **Reserved**

[2] **TOG**: toggle type of input field. When set the input field selection on the display automatically toggles on each Vsync. This only allows the field to be set once at a sequence start up. When reset, the selected field can be frozen or set every field by the application.

[1] **TNB**: top not bottom field. Indicates which field of the picture is read from memory by the display on the next Vsync. When set, the top field is selected, when reset, the bottom field is selected. If TOG is set, the selected field automatically toggles on each Vsync.

[0] **422**:
0: 420 input field 1: 422 input field

LMU_VINL LMU number of input lines

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					VINL										

Address: *LMUBaseAddress* + 0x14
Type: R/W
Buffer: Double buffered: Vsync
Reset: 0
Description: This register specifies the number of input lines to the display formatters in terms of progressive luma count.

For 4:2:0 inputs, it must be programmed with a value that is a multiple of four for interlace inputs.

For 4:2:2 inputs, it must be programmed with a value that is a multiple of two for interlace inputs.

This register is unsigned.

LMU_MRS LMU minimum space between STBus requests

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MRS															

Address: *LMUBaseAddress* + 0x18

Type: R/W

Buffer: Double buffered: Vsync

Reset: 0

Description: This register specifies the number of CLK_SYS clock cycles between two consecutive STBus requests (luma/chroma write to memory and luma/chroma read from memory).

LMU_CHK LMU maximum chunk size

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								WRC		WRY		RDC		RDY	

Address: *LMUBaseAddress* + 0x1C

Type: R/W

Buffer: Double buffered: Vsync

Reset: 0

Description: This register configures the number of STBus transactions linked together using the LCK bit. It indicates the number of packets in a message.

[31:8] **Reserved**

[7:6] **RDY**: Luma STBus read interface.
00: A message is 8 packets
01: A message is 1 packet
10: A message is 2 packets
11: A message is 4 packets

[5:4] **RDC**: Chroma STBus read interface.
00: A message is 8 packets
01: A message is 1 packet
10: A message is 2 packets
11: A message is 4 packets

[3:2] **WRY**: Luma STBus write interface.
00: A message is 8 packets
01: A message is 1 packet
10: A message is 2 packets
11: A message is 4 packets

[1:0] **WRC**: Chroma STBus write interface.
00: A message is 8 packets
01: A message is 1 packet
10: A message is 2 packets
11: A message is 4 packets

LMU_STA LMU status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CDN	YDN	CFF	YFF	YWE	CWE	YRE	CRE

Address: *LMUBaseAddress* + 0x20
Type: RO
Buffer: Immediate
Reset: Undefined
Description: This register shows the activity of the LMU and potential problems.

[31:8] **Reserved**

[7] **CDN**: Chroma LMU done for the current field

[6] **YDN**: Luma LMU done for the current field

[5] **CFF**: Chroma WR fifo to STBus is full

[4] **YFF**: Luma WR fifo to STBus is full

[3] **YWE**: STBus packet transfer error for Luma WR interface

[2] **CWE**: STBus packet transfer error for Chroma WR interface

[1] **YRE**: STBus packet transfer error for Luma RD interface

[0] **CRE**: STBus packet transfer error for Chroma RD interface

LMU_ITM LMU interrupt mask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CDN	YDN	CFF	YFF	YWE	CWE	YRE	CRE

Address: *LMUBaseAddress* + 0x24
Type: R/W
Buffer: Immediate
Reset: 0 (all interrupts disabled)
Description: Any bit set in this register enables the corresponding interrupt in the LMU_IRQ line. An interrupt is generated whenever a bit in register LMU_STA changes from 0 to 1 and the corresponding mask bit is set.

LMU_INT_STA LMU interrupt status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CDN	YDN	CFF	YFF	YWE	CWE	YRE	CRE

Address: *LMUBaseAddress* + 0x28
Type: RO
Buffer: Immediate
Reset: Undefined
Description: When a bit in register LMU_STA changes from 0 to 1, the corresponding bit in register LMU_ITS is set, independent of the state of LMU_ITM. If any set LMU_ITS bit is unmasked, the line LMU_IRQ is asserted. Reading LMU_ITS clears all bits in this register. When LMU_ITS is 0, the LMU_IRQ line returns deasserted.

LMU_AFD LMU accumulated field difference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIFF_Y								DIFF_C							

Address: *LMUBaseAddress* + 0x2C

Type: RO

Buffer: Double buffered: Vsync

Reset: 0

Description: This unsigned status register returns two 8-bit values that indicate the accumulated difference between the current field and the same field of the previous frame; bits 0 to 7 indicate the difference between chroma fields, bits 8 to 15 indicate the difference between luma fields. The accumulated difference may be used by software to detect if a received picture is from a film source.

The difference is not actually accumulated over the entire field. Instead, 16 horizontally adjacent pixels in the current field are compared to the equivalent 16 pixels of the previous frame. The magnitude of each pixel difference is accumulated in a 12-bit register. At the end of the 16-pixel sample, the upper 8 bits of the accumulated difference are stored in an 8-bit maximum difference register.

This process is repeated over the picture. At the end of each 16-pixel calculation, the current accumulated difference is compared to the stored difference, and the greater of the two is stored. This is carried out for both luma and chroma pixels, and the results are accumulated separately. At the end of the field (that is, during vertical reset), the maximum difference register for luma is assigned to field DIFF_Y of LMU_AFD, while the maximum difference register for chroma is assigned to field DIFF_C.

If LMU_CTRL.DTI_Y is set to 1, field LMU_AFD.DIFF_Y is invalid. Similarly, if bit LMU_CTRL.DTI_C of is set to 1, then field LMU_AFD.DIFF_C is invalid.

51 Compositor

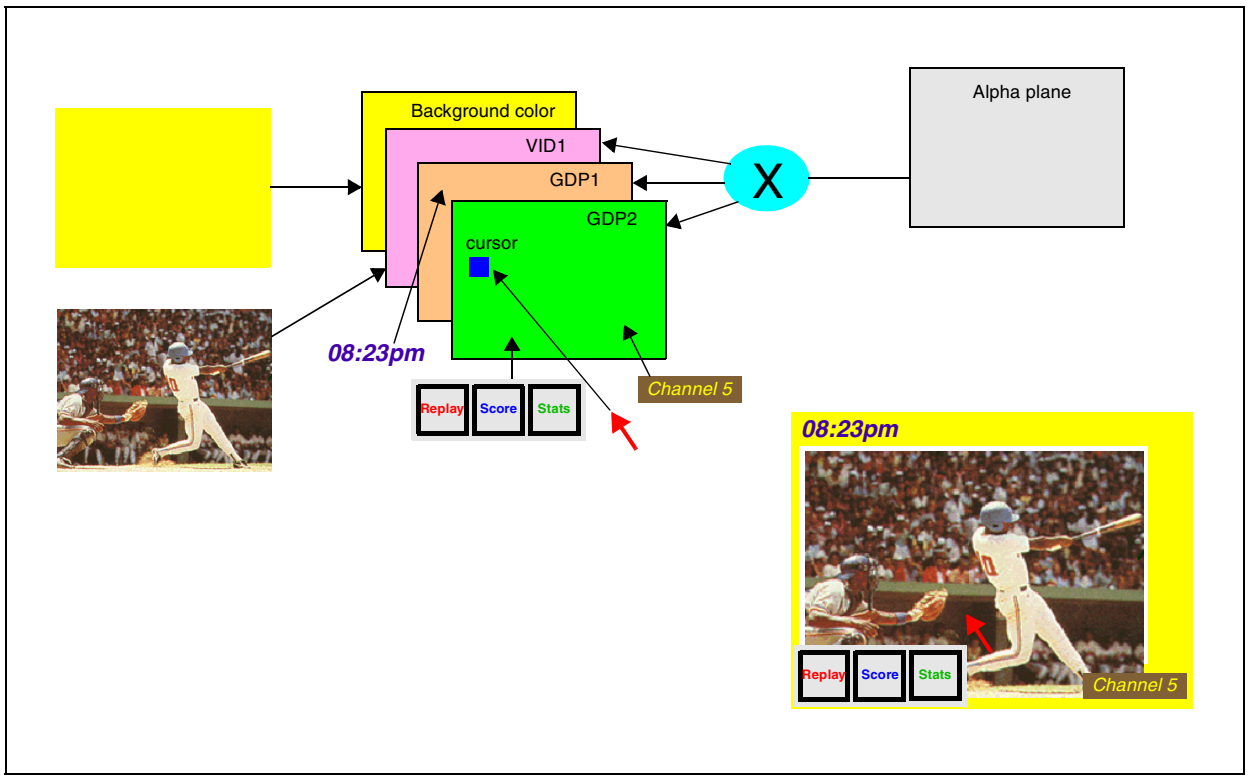
51.1 Overview

The compositor comprises two real-time, multiplane digital mixers. The main mixer (MIX1) composes up to five layers: a background color, a video plane, two graphics planes, and a cursor plane. The auxiliary mixer (MIX2) composes the video 2 plane with the graphics 2 plane.

The video planes are supplied from the main and auxiliary display processors. Pixel data for the 2D-graphics planes and cursor plane are read directly from memory.

After real time processing by the display plane pipelines, pixel data are mixed in mixer 1 or mixer 2. The output of mixer 1 supports up to full HD resolutions and is intended as the main TV display (Figure 167). The output of mixer 2 (Figure 168) supports up to full SD resolutions and is intended as an auxiliary display for applications including connection to a VCR. The mixer outputs are fed to the STx7100's output stage.

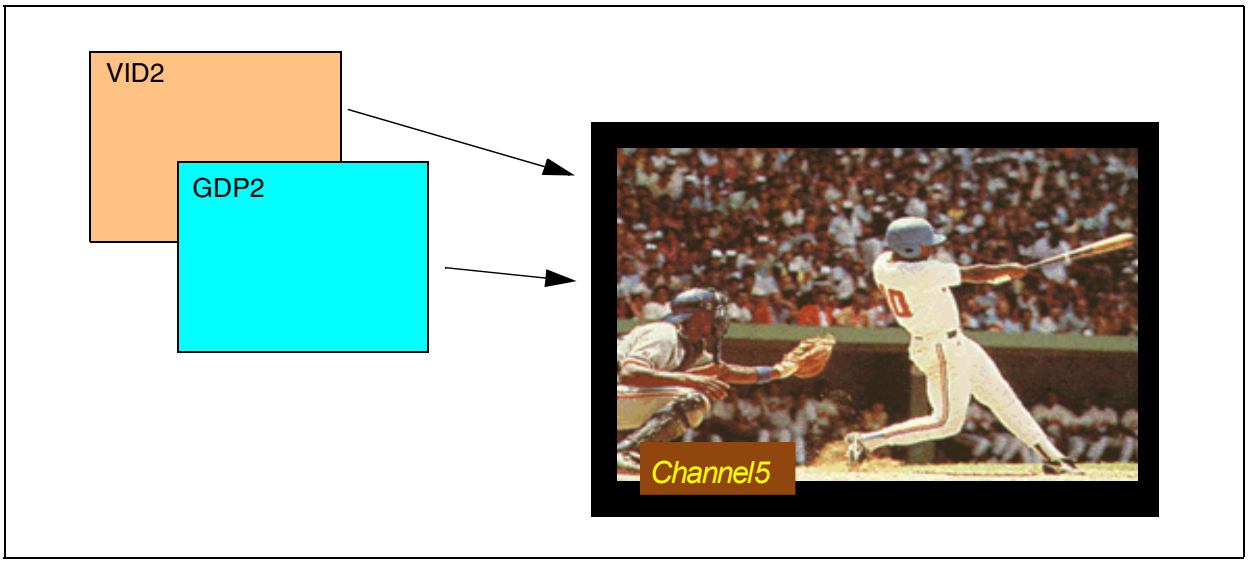
Figure 167: Mixer 1 plane



The compositor also comprises additional components that can be used to enhance the display presentation of video and graphics. These include an alpha plane attachment and a cross-bar

router. Their functions are described below. A capture pipeline is also provided for capturing main or auxiliary video streams or mixer 1 or 2 output streams and storing them in memory.

Figure 168: Mixer 2 planes



51.2 Compositor layout

Figure 169 shows a block diagram of the compositor. It presents the dataflow and memory access of all the compositor modules.

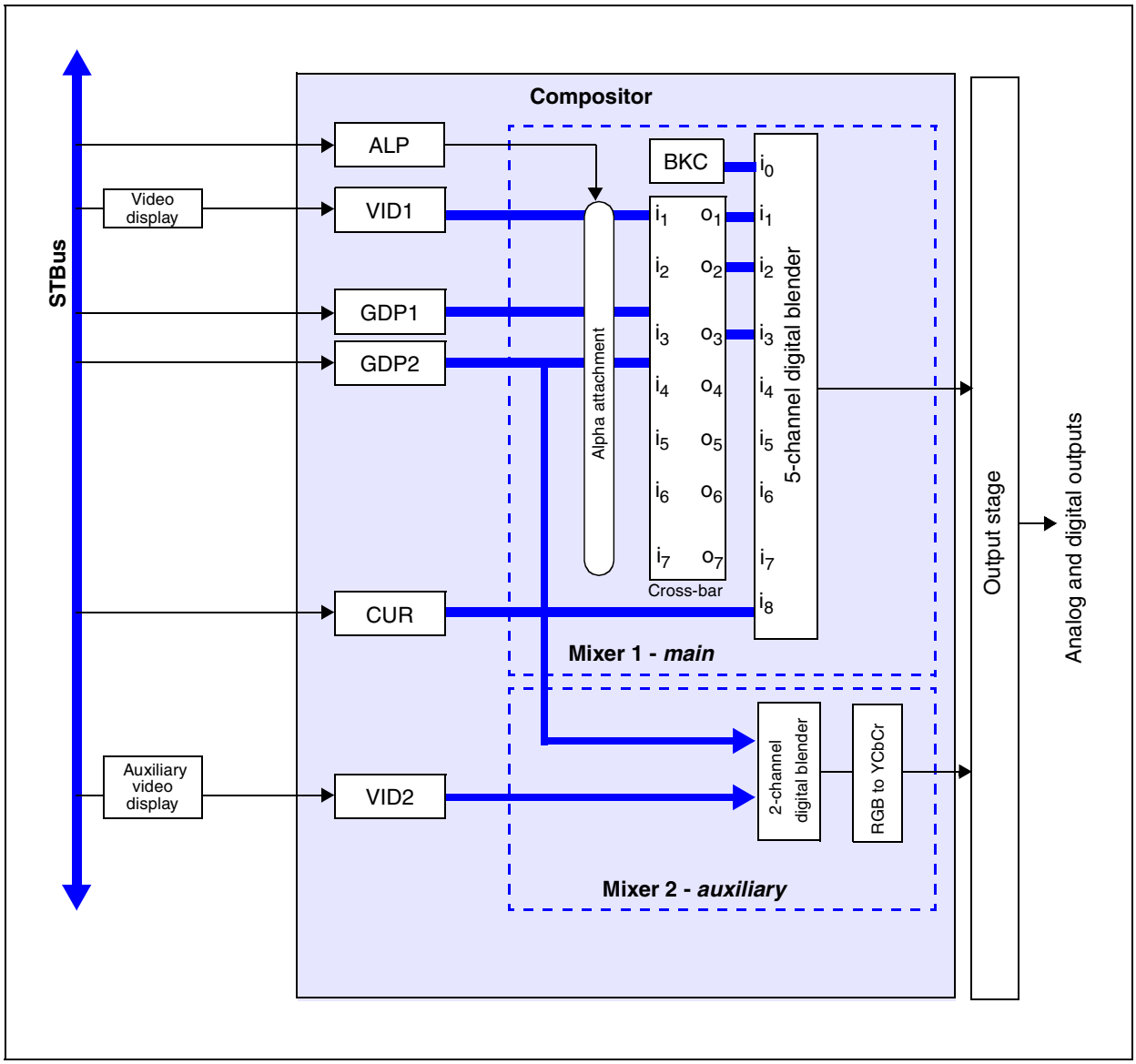
The graphics and cursor pipelines read pixel data and related control information directly from memory. The video input pipelines accept data from the main and auxiliary video display pipelines. Video and graphics data captured for the compositor data flow by the capture pipeline is written back to memory with a resolution up to 32 bits/pixel. The real-time processing performed by each pipeline is controlled by register programming.

Digital mixer 1 successively blends video layers VID1, both graphics layers (GDP1 and GDP2), the cursor layer and a background color. A cross-bar router enables the hierarchy of the GDP1 and VID1 layers to be programmed. The resulting order is background color, GDP2, (VID1 and GDP1 in programmed order) and cursor from background to foreground. Each layer can be independently enabled or disabled. The blending operates in the RGB color domain, so each layer supplies an RGB signal (3x12 bits), with transparency information that provides the weighting coefficients for the mixing operation at a given depth.

Digital mixer 2 successively blends one video layer (VID2) with one graphics layer (GDP2). For mixer 2, the priority is fixed with GDP2 in front of video.

All sub-blocks are controlled by hardware registers. All these registers can be read but not necessarily written. The graphics planes are link-list based and have their register set written through the memory (register download is controlled directly by the hardware after initialization). All other registers can be written. The registers are listed in Chapter 52 on page 554. Each plane block supports a specific set of bitmap formats. All bitmap formats are described in Section 47.8: *Local memory storage of supported graphics formats on page 458*. Each plane starts reading data from memory when it is enabled in mixer 1 or mixer 2.

Figure 169: Graphics and video block diagram



51.3 Digital mixer 1 (MIX1) - main display output

The main display output mixer mixes all the display planes and generates the main output (it cannot be used for the auxiliary output). The output can be high- or standard-definition, and is usually used for TV.

Mixer 1 is controlled by registers prefixed with GAM_MIX1_.

Each graphics and video layer can be enabled or disabled for display in register GAM_MIX1_CTRL.

Mixer 1 is composed of three independent sub-blocks: the blender, the cross-bar router, and the alpha plane attachment unit.

51.3.1 Four-channel blender

The digital mixer successively blends the four layers, from background to foreground. Each layer can be independently enabled or disabled. Blending operates in the RGB color domain, so each layer, except for the cursor, supplies an RGB signal (3 x 12 bits) with transparency information that provides the weighting coefficients for the mixing operation at a given depth.

The background color register GAM_MIX1_BKC is a 24-bit RGB register included in the mixer, and is functionally equivalent to a plane filled with solid color. This plane is always opaque with no associated alpha value. The limits of this plane are specified according to a programmable rectangular window. Outside this window, the background color is forced to the blanking color (black R = G = B = 0).

The computation unit outputs a 4:4:4 digital RGB signal. A global rectangular window for defining the active video area is provided (GAM_MIX1_AVO and GAM_MIX1_AVS). Outside this window, the mixer outputs a default blanking color (black R = G = B = 0). A window indicator signal is provided, synchronously with the RGB data bus, for external use (such as a video blanking signal).

Mixer 1 takes into account bits IGNOREONMIX1 and FORCEONMIX1 (GAM_GDPn_PPT), provided by the GDP pipelines, on any enabled GDP layer. If bit IGNOREONMIX1 is set, the current viewport is not displayed. If bit FORCEONMIX1 is set, then the GDP viewport color information can be displayed outside the active video area window, instead of the blanking color.

Note: The window indicator flag is not affected.

From an application point of view, this last feature is useful when a VBI waveform has been synthesized as a graphics object, and uses a GDP pipeline to be inserted in the analog output on a VBI line.

Two configurable signals can indicate whether the current output contains a certain amount of graphics information, or if it is composed of pure video content. In some systems, external processing operations can be applied selectively, according to the pixel video or graphic content.

51.3.2 Cross-bar router

The cross bar router is used to re-order the plane hierarchy, so that the user can program the order that the video and graphics planes appear in the display, from background to foreground. The planes that can be re-ordered include the video plane and the two graphics planes (VID1, GDP1 and GDP2). The background color and cursor layers are not affected. The depth of each layer in the hierarchy is programmed in register GAM_MIX1_CRB.

51.3.3 Alpha plane attachment unit

The alpha plane processor can combine the alpha channel provided by the alpha plane pipeline with the alpha component of either GDP1, GDP2 or VID1.

Any of these pipelines supplies RGB color information, together with two alpha components. One gives the blending weight for this current RGB pixel ($alpha_{RGB}(IN)$), whereas the other gives the weight for the intermediate pixel resulting from the blend operations of all the background layers ($alpha_{BKG}(IN)$). Let $alpha_{alpha\_plane}$ be the value provided by the alpha plane pipeline.

The alpha processor reinjects the alpha plane value on both components, using:

$alpha_{RGB}(OUT) = alpha_{RGB}(IN) \times alpha_{alpha\_plane}$

$alpha_{BKG}(OUT) = 1.0 - ((1.0 - alpha_{BKG}(IN)) \times alpha_{alpha\_plane})$

This is correct for both the VID and GDP pipelines handling either premultiplied or non-premultiplied colors.

51.4 Digital mixer 2 (MIX2) - auxiliary display output

Mixer 2 is a much simpler engine, for the auxiliary display only, with just two inputs, generic display pipeline GDP2 and video VID2. It is controlled by the registers prefixed with GAM_MIX2_.

Each graphics and video layer can be enabled or disabled for display in GAM_MIX2_CTRL.

Blending operates in the RGB color domain, from the RGB signals (3 x 12 bits) supplied by the layers, together with the associated transparency information.

There is no background color register, but a default black background color (R = G = B = 0).

The computation unit outputs a 4:4:4 digital RGB signal. A global rectangular window for defining the active video area is provided (GAM_MIX2_AVO and GAM_MIX2_AVS). Outside this window, the mixer outputs a default blanking color (black R = G = B = 0). A window indicator signal is provided, synchronously with the RGB/YCbCr data bus, for external use (such as a video blanking signal, for instance).

If GDP2 is enabled, mixer 2 takes into account the property register GAM_GDPn_PPT, bits IGNOREONMIX2 and FORCEONMIX2, provided by the GDP2 pipeline. If IGNOREONMIX2 is set, the current viewport is not displayed. If FORCEONMIX2 is set, then the GDP viewport color information can be displayed outside the active video area window, instead of the blanking color.

Note: The window indicator flag is not affected.

From an application point of view, this last feature is useful when a VBI waveform has been synthesized as a graphics object, and uses the GDP2 pipeline to be inserted in the analog output, on a VBI line.

Two configurable signals indicate whether the current output contains graphics information, or if it is composed of pure video content. In some systems, external processing operations can be applied selectively, according to the pixel video or graphics content.

51.5 Generic display pipelines (GDP1 and GDP2)

The generic display pipelines have the following features:

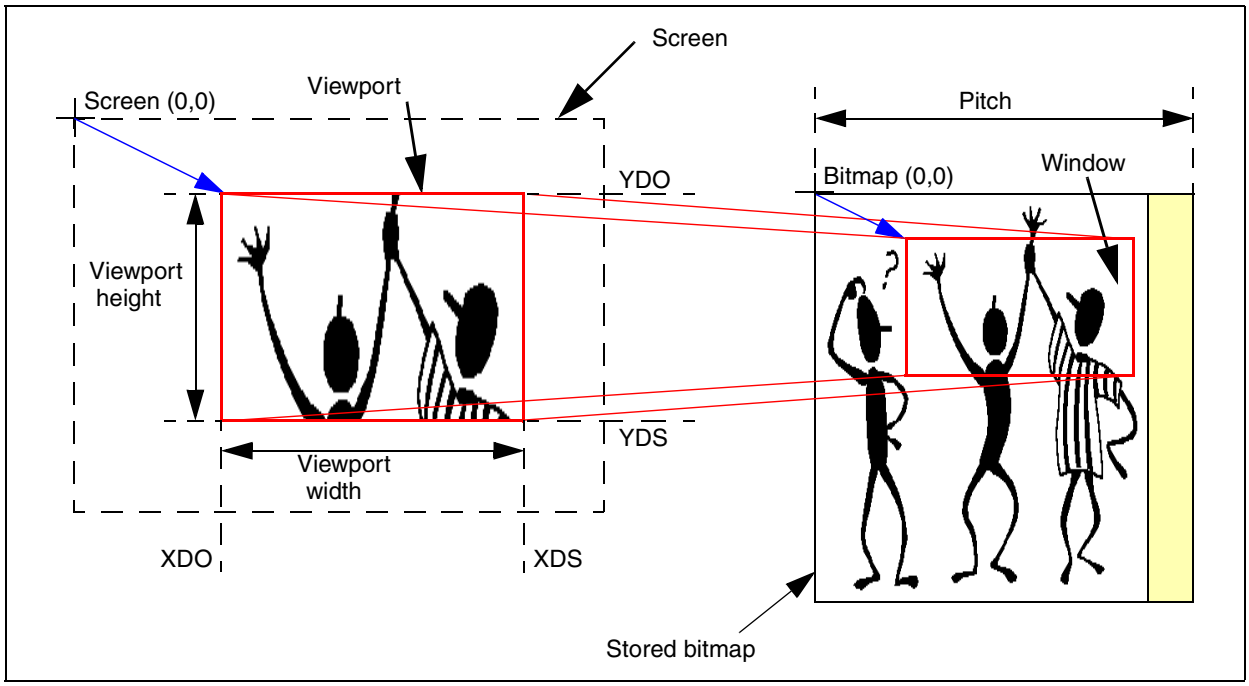
- Link-list-based display engine, for multiple viewport capabilities.
- ARGBargb formats including ARGB1555, ARGB4444, RGB565, RGB888, ARGB8565, ARGB8888.
- YCbCr4:2:2R, YCbCr888 and AYCbCr8888 format support.
- Premultiplied or non-premultiplied RGB component support.
- Little endian and big endian bitmap support.
- Color space conversion matrix, (YCbCr 601/709, chroma signed/unsigned to RGB).
- Gain and offset adjustment.
- Per-pixel alpha channel combined with per-viewport global alpha.
- 5-tap horizontal sample rate converter, for horizontal upsampling. This can be used to adapt the pixel aspect ratio. The resolution is 1/8th pixel (polyphase filter with eight subpositions).
- Color keying capability.

The output format of GDP is RGB-12-12-12 that goes to the digital mixer along with the 8-bit alpha and 8-bit (1-alpha) values. See full description of the supported graphics in Section 47.8: *Local memory storage of supported graphics formats on page 458*.

51.5.1 General description of GDPs

A GDP can handle a multiple-viewport display, using a display instruction list (link list) stored in the external memory. (A viewport is a physical rectangular area on the screen. It is defined by XDO / XDS / YDO / YDS with reference to the top-left corner of the screen.)

Figure 170: Example: a resized window attached to a viewport



One or several viewports can be attached to the GDP layer, depending on its capabilities. When several viewports are defined within a layer, only one viewport can be handled by each video scan line. In a given layer, each screen location where no viewport is defined is automatically transparent (the alpha channel to the mixer is forced to 0).

A bitmap stored in external memory must be attached to the viewport. All or part of this bitmap can be visible in the viewport; this is the bitmap window. The window is defined by the bitmap

color format, bitmap pitch, linear start address, width and height. The linear start address is the address of the bitmap pixel that is to be displayed in the top-left corner of the viewport, that is the address of the window (0,0) pixel.

Most of the time, the window and the viewport have the same size. Their sizes are different each time a resizing factor is applied to the window, so that the rescaled bitmap matches with the viewport size.

When programmed sizes are not consistent, the display is locally undefined.

The screen is described by a display link-list that must be built in the external memory. For each viewport, a node is defined that contains the viewport configuration, bitmap window settings, display options (such as global transparency, filtering mode, gain/offset adjustment). The node also contains a memory pointer to the node describing the next viewport to display.

The display link-list is generally circular. For an interlaced display, the link-list is field-based. A node must be provided for the top and the bottom fields (they can be different, particularly the start address).

Figure 171: Typical display link-list configuration for a progressive display

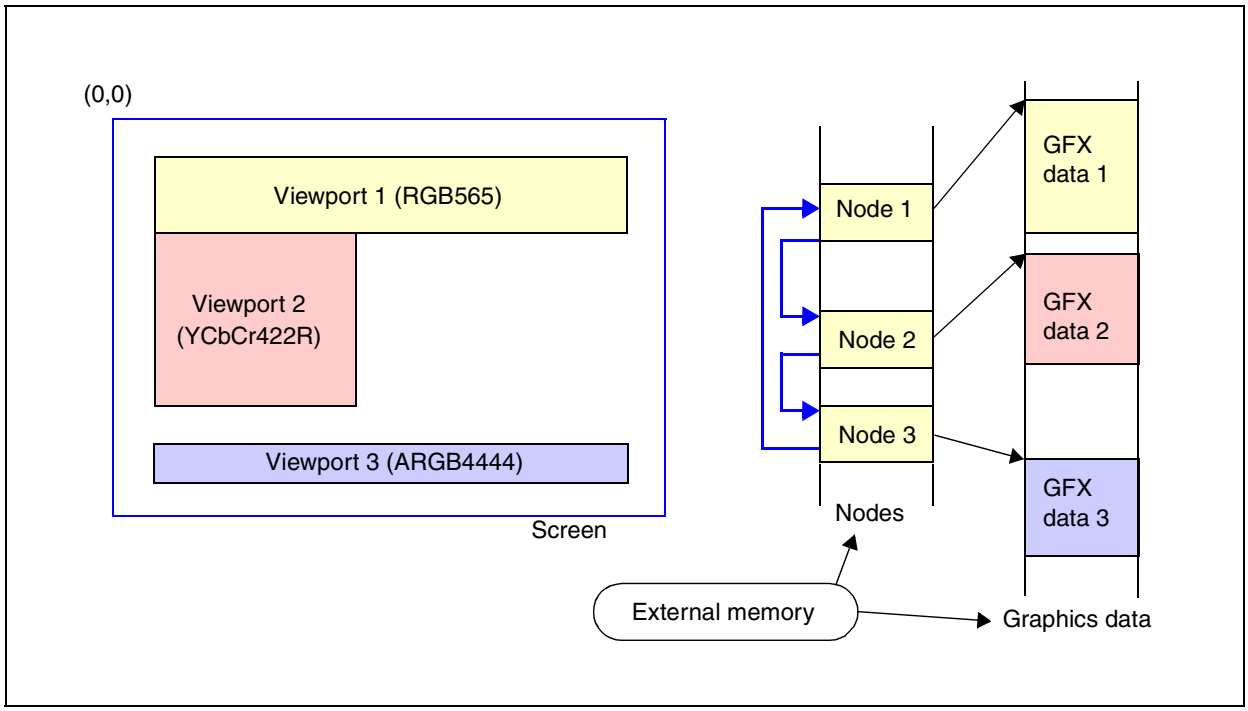
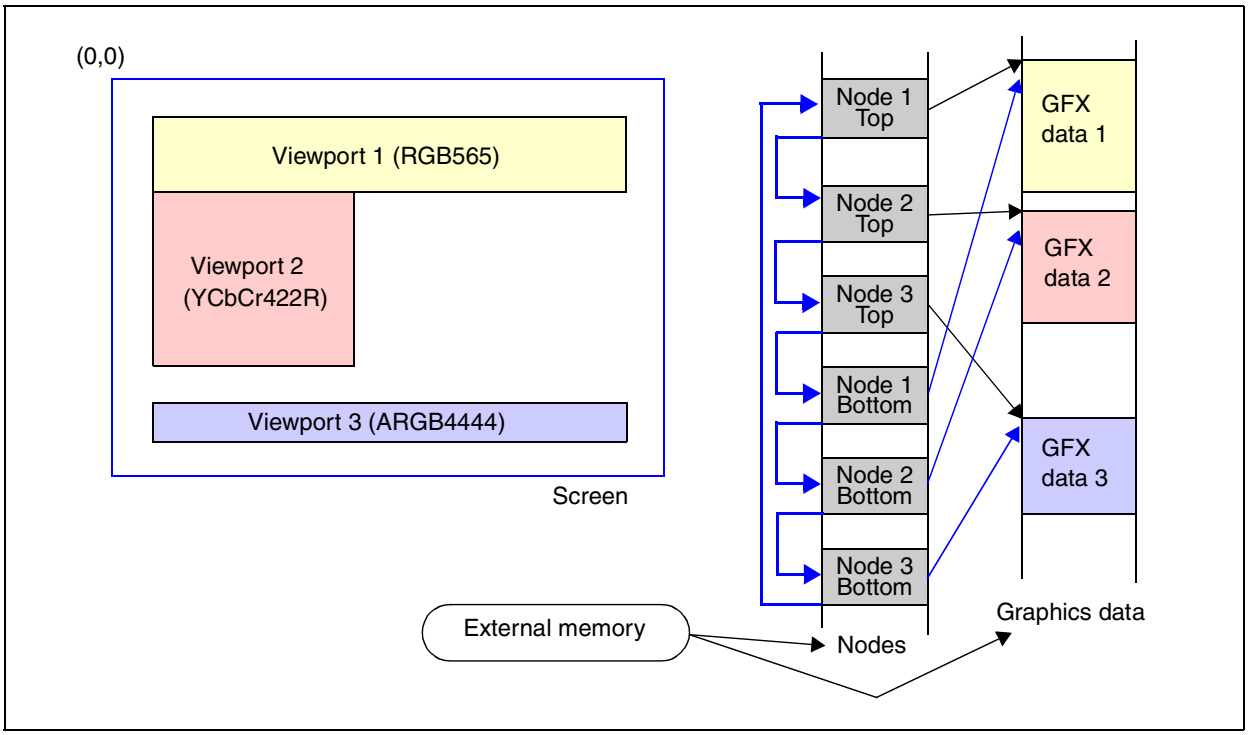


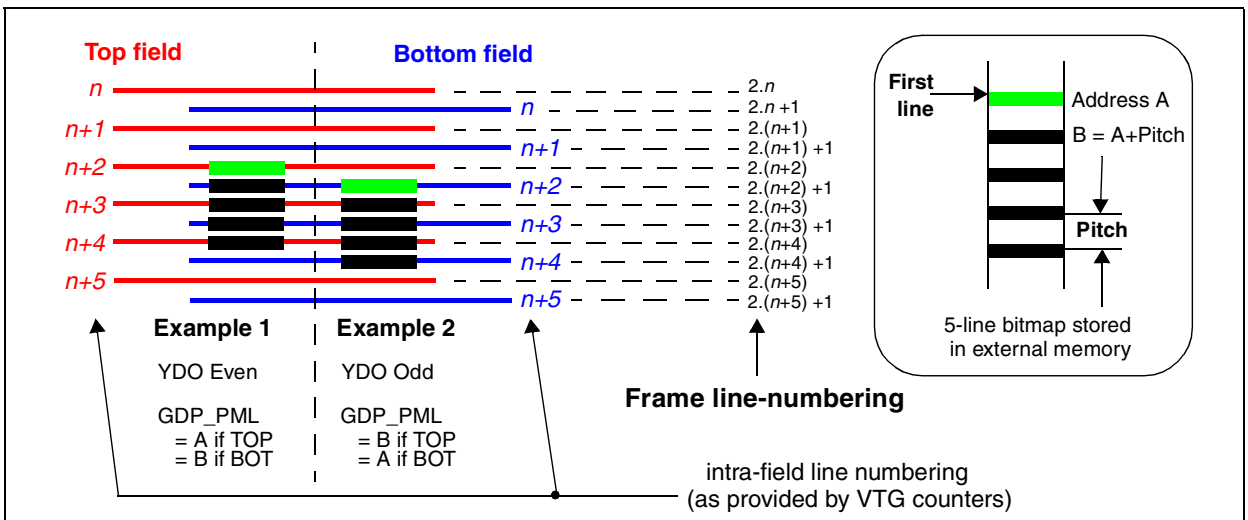
Figure 172: Typical display link-list configuration for an interlaced display



YDO and YDS are specified with respect to a frame line-numbering, even in an interlaced display.

The next figure specifies how the hardware must consider the register values, depending on the top or bottom field, so that each viewport can be vertically positioned with a one-line accuracy.

Figure 173: Line numbering convention in an interlaced display



1 - During the top field, the VTG counter must be compared to **(YDO+1 >> 1) and (YDS >> 1)** for detecting the viewport active area.

2 - During the bottom field, the VTG counter must be compared to **(YDO >> 1) and (YDS-1 >> 1)** for detecting the viewport active area.

3 - Whatever the field, top or bottom, the first bitmap line to read from memory is always at the memory location specified by the GDP_PML register. It is the responsibility of the software to program this register according to the field parity.

For the progressive display configuration, this consideration is of course trivial.

How to start the display

To start the display on a GDP, a link-list must be programmed in memory. Register GAM_GDPn_NVN must be set by the CPU with the address of the first node to be displayed within this link-list. Then the corresponding GDP enable bit in register GAM_MIX*n*_CTRL must be set. This write operation is synchronized on the next Vsync event. On that event, the GDP pipeline fetches the first node from memory, and starts to retrieve pixel data according to the display parameters specified in the node.

To obtain clean node switching, particularly in the case of two vertically adjacent viewports (the horizontal blanking interval can be quite short), the hardware uses an internal dual register bank in toggle mode for the nodes. The next node loading process is always anticipated (for node *N*), and occurs at the beginning of the first line displayed for node (*N* - 1). This mechanism is transparent to the programmer.

To stop the display, the GAM_MIX*n*_CTRL enable bit must be set back to 0. This is taken into account synchronously with the field (frame) rate.

How to modify the display parameters

Hardware and software may conflict when accessing the nodes if the software wants to modify a display parameter. It is recommended to toggle between two link-lists stored in memory each time one or more display parameters have to be modified.

For the final node of either of these link-lists, which corresponds to the lowest viewport, bit WAIT_NEXT_VSYNC in register GAM_GDPn_CTRL must be set to 1. This prevents the node anticipation process from occurring: loading is delayed until the next Vsync event.

When the hardware is working from the current displayed link-list, the software is free to make any modification in the other link-list, such as viewport parameters, viewport insertion/deletion. Once the link-list has been updated, the software simply updates the GAM_GDPn_NVN register field in the last node of the current link-list. This is a single memory write access, and thus cannot conflict with a hardware access (no partially updated parameters). As this node has been programmed not to anticipate the next node loading, the hardware waits until the next Vsync, and then correctly switches to the new link-list.

If memory update for register GAM_GDPn_NVN is synchronized in the Vsync software handler, the link-list switch occurs with a one field delay (or one frame if progressive).

For an interlaced display, this scheme can be simplified by using a single link-list and updating the top parameters while the bottom field is displayed and vice-versa.

Bandwidth considerations

In terms of the bandwidth requirement (BR), the general formula to estimate the GDP weight on a given system is the following:

BR (in Mbyte/s) = (pixel frequency in MHz) x (number of bytes per pixel) x (horizontal resampling factor)

Examples:

- 601 rate / RGB565 / x1: BR = 13.5 x 2 x 1 = 27 Mbyte/s
- PAL SQ rate / YCbCr422R / x1: BR = 14.75 x 2 x 1 = 29.5 Mbyte/s
- 601 rate / ARGB8888 / zoom out x2: BR = 13.5 x 4 x 2 = 108 Mbyte/s

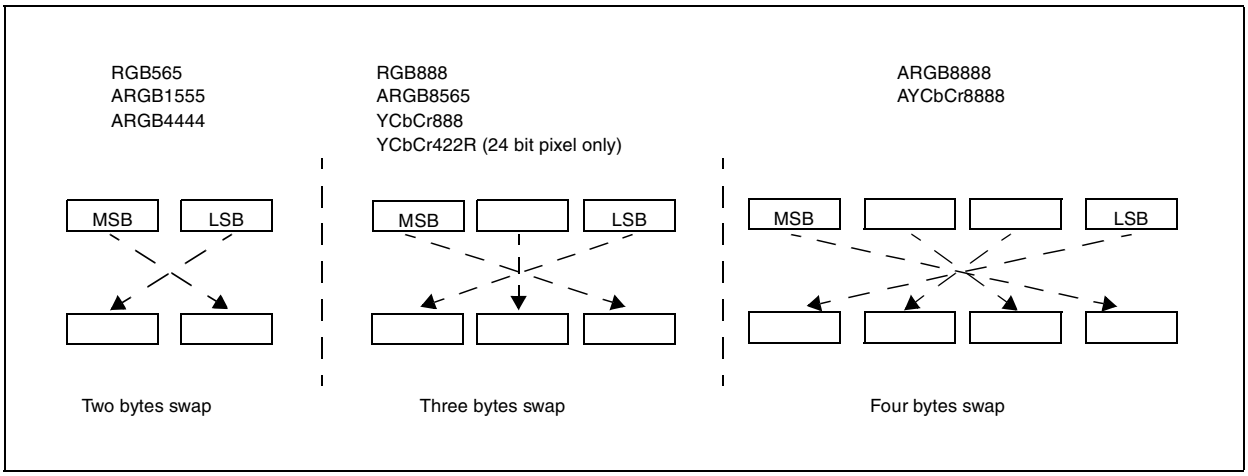
51.5.2 Description of unitary functions

Input formatter

This sub-block converts the 128-bit STBus data bus, into an internal pixel bus. The clock rate for the pixel bus is PIXCLK.

If the bitmap is big endian, the endianness conversion occurs on the original pixels as read from memory, before their mapping on the internal 32-bit pixel bus. Figure 174 shows the byte swap performed on big endian pixels, according to the different color formats. For each viewport, the endianness of the bitmap may be defined using GAM_GDPn_CTRL.BIGNOTLITTLE.

Figure 174: Endianness conversion for big endian pixels



Another bit that is not included in the viewport node (GAM_GDPn_PKZ.BIGNOTLITTLE) indicates if the nodes, filters coefficients, are stored in big or little endian. This bit should reflect the endianness of the host CPU. It performs a four-byte swap on the 32-bit word fetched when loading the nodes and the filter coefficients.

Table 163: 32-bit pixel bus format, at the formatter output

Input color format	Alpha		Color data					
	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
RGB565	128		R5/3MSB or 000		G6/2MSB or 00		B5/3MSB or 000	
RGB888	128		R8		G8		B8	
ARGB1555	GLOBAL_ALPHA_0 if A1=0 GLOBAL_ALPHA_1 if A1=1		R5/3MSB or 000		G5/3MSB or 000		B5/3MSB or 000	
ARGB8565	A8[a]		R5/3MSB or 000		G6/2MSB or 00		B5/3MSB or 000	
ARGB8888	A8[a]		R8		G8		B8	
ARGB4444	0/A4/100 (but keep 0 and 128)		R4/4MSB or 0000		G4/4MSB or 0000		B4/4MSB or 0000	
YCbCr888 YCbCr422R	128		Cr8		Y8		Cb8	
AYCbCr8888	A8[a]		Cr8		Y8		Cb8	

a. The GDP supports either 0 to 128 or 0 to 255 8-bit alpha. When a 255 range is used, the input alpha value is converted to a 0 to 128 component using the following formula: $A_{0..128} = (A_{0..255} + 1) \times 2^{-1}$.

When there is no alpha channel in the input color format, a 128 alpha value is forced, except for the ARGB1555 mode. In this case, the global alpha registers (0 and 1) are used in the input formatter block. When bit 15 of the incoming pixel (A1) is 0, the alpha output is forced to the value of GLOBAL_ALPHA_0. When bit 15 is 1, the alpha output is forced to the value of GLOBAL_ALPHA_1.

Two possibilities are provided to extend the component depth (4/5/6 bits to 8 bits): either the LSBs are filled with 0, or the MSBs are repeated, so that the full 8-bit color range can be used (see LSB_STUFFING_MODE in register GAM_GDPn_CTRL). For 4:2:2 to 4:4:4 conversion, the interpolation scheme is similar to the one used in the blitter (except for the chroma extended mode that is not supported). This is described in Chapter 47: *2D graphics processor (blitter) on page 428*.

Color space converter

If the input format is YCbCr888, AYCbCr8888 or YCbCr4:2:2R, a YCbCr to RGB color space converter is required. It uses either the 601 matrix or the 709 matrix. It is bypassed for an RGB input. In this version of the compositor, the video RGB color space is used to fit with NTSC/PAL test patterns. The output is 10:10:10 signed RGB.

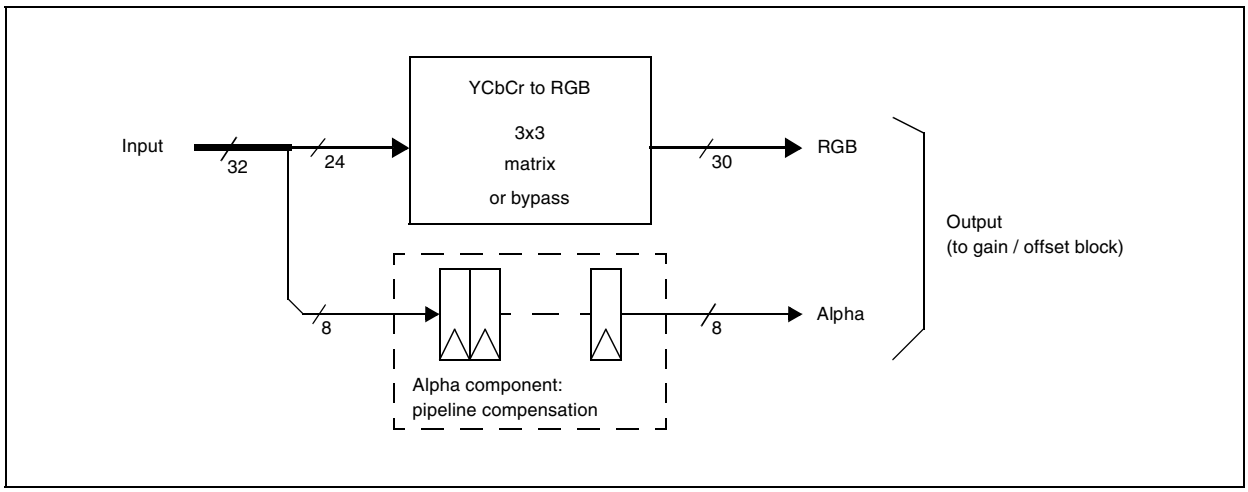
Table 164A: 601 colorimetry / floating-point matrix / digital range

YCbCr to video RGB integer matrix									
R	=	1	x Y	+	0	x (Cb - 128)	+	351/256	x (Cr - 128)
G	=	1	x Y	-	86/256	x (Cb - 128)	-	179/256	x (Cr - 128)
B	=	1	x Y	+	444/256	x (Cb - 128)	+	0	x (Cr - 128)

Table 164B: 709 colorimetry / floating-point matrix / digital range

YCbCr to video RGB integer matrix									
R	=	1	x Y	+	0	x (Cb - 128)	+	394/256	x (Cr - 128)
G	=	1	x Y	-	47/256	x (Cb - 128)	-	117/256	x (Cr - 128)
B	=	1	x Y	+	464/256	x (Cb - 128)	+	0	x (Cr - 128)

Figure 175: Color space converter block diagram



2D-resize, horizontal filter

The GDP contains a horizontal filter that can be used for upsampling of graphics displays. Downsizing and vertical resizing are not supported. The filter is based on an 5-tap sample rate converter with eight phases and is programmed using GAM_GDPn_HSRC. The SRC increment is programmed using a 2.8 format and the initial phase can be programmed to a resolution of eight subpixel positions.

The horizontal filter coefficients are stored in memory. The 40 filter coefficients are loaded from memory pointed to by GAM_GDPn_HFP and can be updated for each viewport if required. Resizing can be performed with the filter enabled for interpolation or disabled for pixel repetition. The horizontal filter can be used for upsampling and is provided to allow storage of HD resolution graphics with a lower resolution (for example, x1/4 compared with full resolution storage) and also for aspect ratio conversion of square pixel generated graphics for nonsquare pixel aspect ratio display or vice versa.

Color key

If the current pixel color is part of the key range, then the associated alpha component of this pixel is forced to 0, making this pixel totally transparent when mixed with the background layers in the digital mixer. The color components are forced to black.

Table 165: Color key block output if match

Input format	Output if color key match
ARGBargb	A = R = G = B = 0
YCbCr422R / YCbCr888 / AYCbCr8888 Chroma offset binary (unsigned)	A = 0, Y = 16, Cb = Cr = 128
YCbCr422R / YCbCr888 / AYCbCr8888 Chroma two's complement (signed)	A = 0, Y = 16, Cb = Cr = 0

The color key feature is enabled for a viewport by setting GAM_GDPn_CTRL.COLOR_KEY_EN to 1. When set, the color key operates as follows:

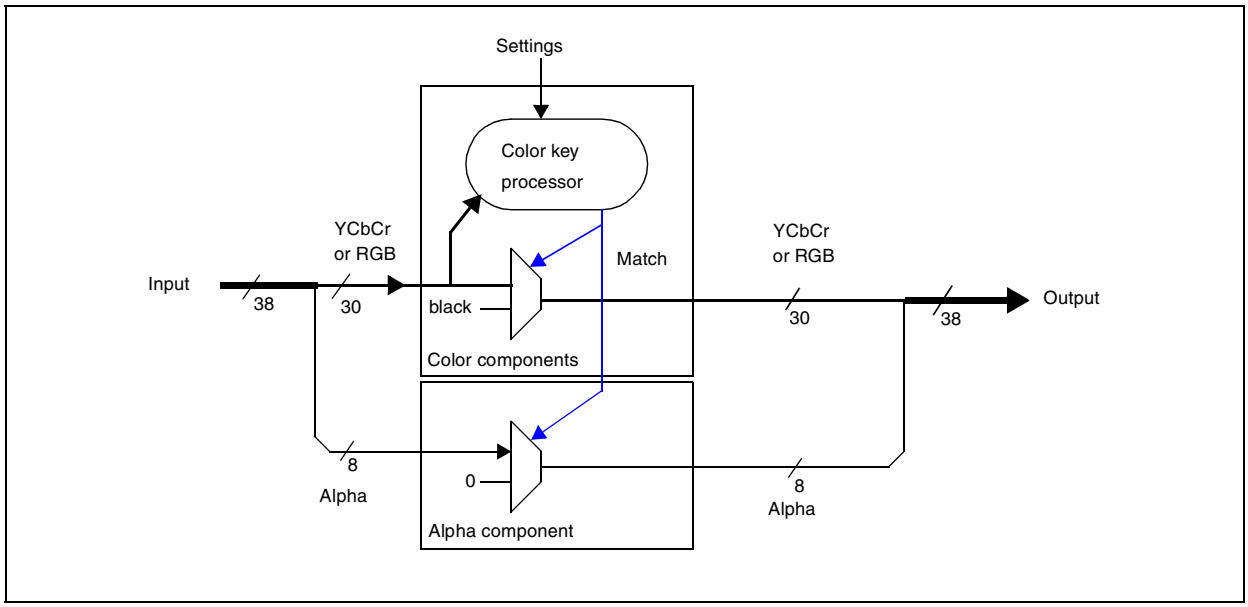
```
COLOR_KEY_MATCH = TRUE
IF
   [(Rin<Rmin) or (Rin>Rmax)] AND GAM_GDPn[21:20] = 11
OR (Rmin<=Rin<=Rmax) AND GAM_GDPn[21:20] = 01
OR GAM_GDPn[21:20] = x0
AND
   [(Gin<Gmin) or (Gin>Gmax)] AND GAM_GDPn[19:18] = 11
OR (Gmin<=Gin<=Gmax) AND GAM_GDPn[19:18] = 01
OR GAM_GDPn[19:18] = x0
AND
   [(Bin<Bmin) OR (Bin>Bmax)] AND GAM_GDPn[17:16] = 11
OR (Bmin<=Bin<=Bmax) AND GAM_GDPn[17:16] = 01
OR GAM_GDPn[17:16] = x0
```

When used on a YCbCr input the correspondence is the following: G/Y, Cb/B, Cr/R.

The values R_{MIN}, G_{MIN}, B_{MIN} are programmed in GAM_GDPn_KEY1.

The values R_{MAX}, G_{MAX} and B_{MAX} are programmed in GAM_GDPn_KEY2.

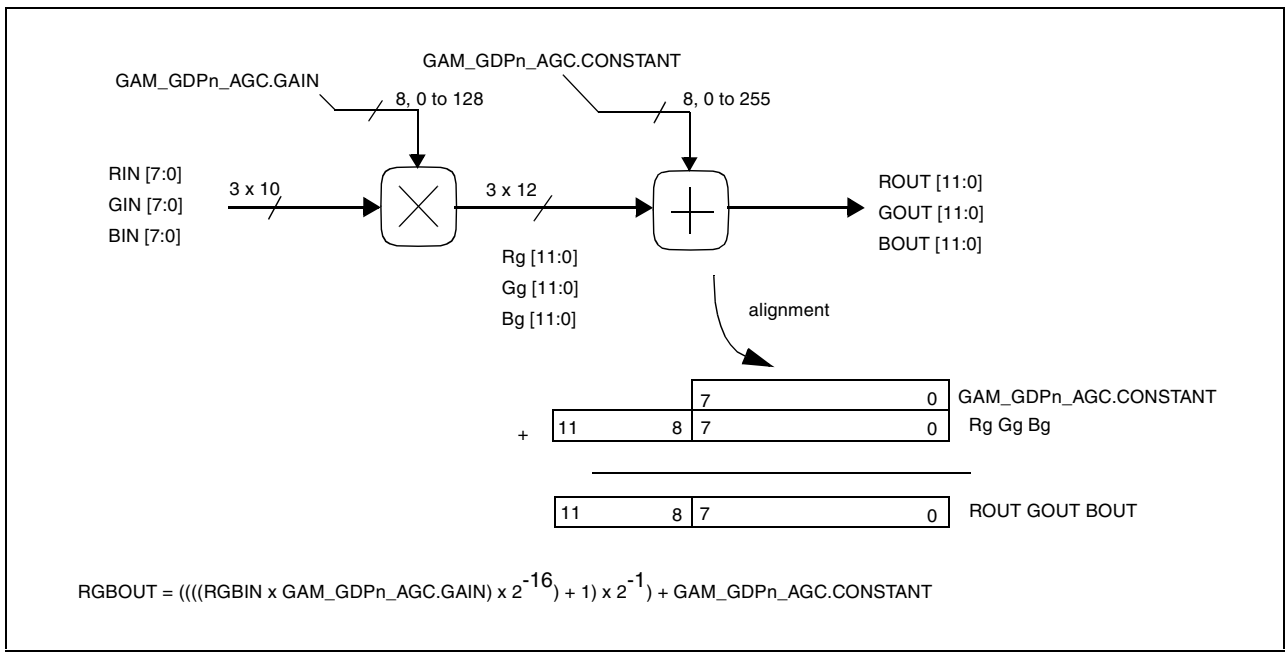
Figure 176: Color key block diagram



Gain/offset adjustment

A dynamic range adjustment is provided. The three RGB components can be multiplied by a fixed coefficient, before they are sent to the digital mixer. This is useful to avoid highly saturated colors to be mixed with video. The black level can be adjusted as well. When used simultaneously, these features act like a contrast adjustment.

Figure 177: Gain/offset diagram



Alpha out sub-block

This block provides the relative weights that are used by the digital mixer when mixing the GDP output on top of the background layers (BKG). The following C-like code shows what is output by the pipeline, depending on the input format (premultiplied or not, special case for ARGB1555), and on the ALPHA_HBORDER_EN / ALPHA_VBORDER_EN configuration bits in register GAM_GDPn_CTRL.

Naming conventions:

- $Alpha_{PIXEL}$ is the alpha component attached to the current pixel at the output of the color space converter block.
- $Alpha_{GDP}$ is the weight provided to the mixer for the GDP color components
- $Alpha_{BKG}$ is the weight provided to the mixer for the background color components

```
IF (InputFormat == ARGB1555)
GlobalAlpha = 128; // GLOBALALPHA0/1 registers used in input formatter block
ELSE
  GlobalAlpha = GlobalAlpha0_register;
IF [(FirstPixel or LastPixel) AND ALPHA_HBORDER_EN]
OR ((FirstLine OR LastLine)  AND ALPHA_VBORDER_EN))
GlobalAlpha = GlobalAlpha >> 1; // Alpha divided by 2 on viewport edges
IF (PREMULTIPLIED_FORMAT)
Alpha_GDP  = GlobalAlpha // Alpha_PIXEL already applied on color components
ELSE
  Alpha_GDP  = (((GlobalAlpha x Alpha_PIXEL) >> 6) + 1) >> 1;
  Alpha_BKG = 128 - { [((GlobalAlpha x Alpha_PIXEL) >> 6) + 1] >> 1 },
```

GLOBALALPHA0/1 registers refer to GAM_GDPn_AGC.GLOBAL_ALPHA_0 and GLOBAL_ALPHA_1.
ALPHA_VBORDER_EN, ALPHA_HBORDER_EN, and PREMULTIPLIED_FORMAT are in GAM_GDPn_CTRL.

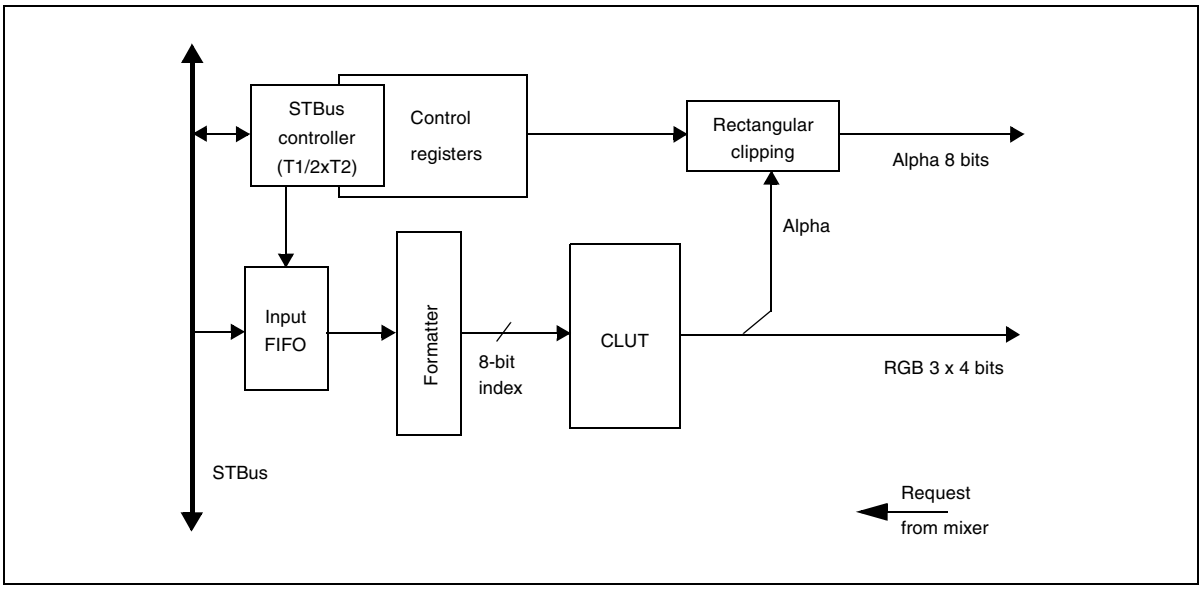
51.6 Cursor plane (CUR)

The cursor is defined as a 128x128 pixel area held in off-chip memory, in ACLUT8 format. Each cursor entry is a 16-bit ARGB4444 color + alpha value. The alpha factor of four-bits is for handling an anti-aliased cursor pattern on top of the composed output picture.

The cursor plane has the following features:

- ACLUT8 format, with ARGB4444 CLUT entries, so 256 colors can be simultaneously displayed for the cursor pattern, among 4096 colors associated with a 16-level translucency channel.
- Size is programmable up to 128x128.
- Hardware rectangular clipping window, out of which the cursor is never displayed (per-pixel clipping, so only part of the cursor can be out of this window, and consequently transparent).
- Current bitmap is specified using a pointer register to an external memory location, making cursor animation very easy.
- Programmable pitch, so that all cursor patterns can be stored in a single global bitmap.

Figure 178: CUR functional block diagram



51.6.1 General description of cursor plane

Cursor overview

The cursor pipeline can display a cursor pattern, stored in external memory in ACLUT8 format. An internal CLUT makes the color expansion from index to true-color values, using 16-bit ARGB4444 entries.

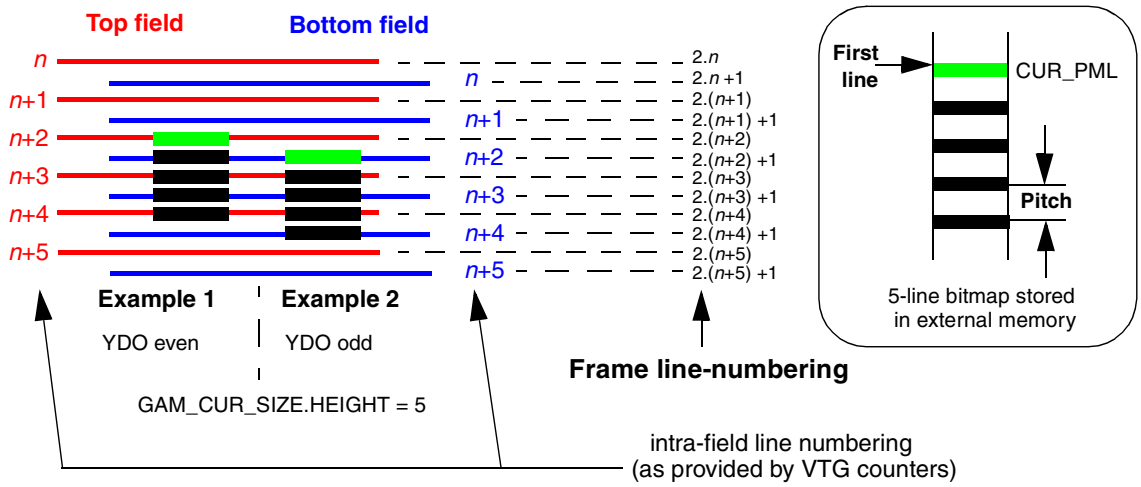
- The vertical size of the cursor pattern can range from 1 to 128 (register GAM_CUR_SIZE).
- The maximum horizontal size depends on the memory alignment of the cursor pixel data. Assume that CUR_MEM_ADDR (GAM_CUR_PML) is the memory address for pixel (0,0), with respect to a top-left origin. The maximum width is:

  MAX_CURSOR_WIDTH = 128 - (CUR_MEM_ADDR mod 16)

  When the cursor data are aligned on a 128-bit word memory, the maximum width is 128. The worst case is when (CUR_MEM_ADDR mod 16) = 15 (example: 0xAAAF); the maximum width becomes 113.
- The cursor pipeline is controlled through a register file directly accessible by the CPU. As these registers are double-buffered with internal update on Vsync, any change is taken into account during the next field to be displayed.
- The cursor pattern can be changed simply by modifying register GAM_CUR_PML (GUI cursor change, cursor animation).
- The CLUT is loaded from memory during each vertical blanking interval, if this loading process is enabled (see GAM_CUR_CTRL.REFRESH). A dedicated register provides the memory address to retrieve the CLUT data (GAM_CUR_CML).
- The YDO value (in register GAM_CUR_VPO) is specified with respect to frame line-numbering, even in an interlaced display.

  The next figure specifies how the hardware uses these register values, depending on the parity of the current field (top or bottom), so that each viewport can be vertically positioned with a one-line accuracy. Thus these registers need not be programmed according to the current displayed field. The pipeline automatically fetches the right data in memory.

Figure 179: Cursor line numbering assumption in an interlaced display



1 - During the top field, the VTG counter must be compared to ((YDO + 1) x 2^{-1}) for detecting the video line where the first bitmap line must be displayed
If YDO is even, this first bitmap line to display is at memory location CUR_PML
If YDO is odd, this first bitmap line to display is at memory location CUR_PML+PITCH

2 - During the bottom field, the VTG counter must be compared to (YDO x 2^{-1}) for detecting the video line where the first bitmap line must be displayed
If YDO is even, this first bitmap line to display is at memory location CUR_PML+PITCH
If YDO is odd, this first bitmap line to display is at memory location CUR_PML

For progressive display configuration, this consideration is not important.

How to start the cursor display

To start displaying the cursor, the corresponding enable bit in register GAM_MIX1_CTRL must be set. This write operation is synchronized on the next Vsync event. On that event, the cursor pipeline fetches the CLUT from memory, if the CLUT refresh process is enabled.

Then the cursor pipeline retrieves pixel data from memory, according to the display parameters specified in the node. A single access is made.

To stop the display, the GAM_MIX1_CTRL enable bit must be set back to 0. This is taken into account synchronously with the field (frame) rate.

Bandwidth considerations

In terms of bandwidth requirements, the cursor pipeline weight on a given system is as follows:

- During the active video scanning area, one or two requests that occur at the beginning of the video line (total: 128 bytes or less, according to the cursor width). Normally, one burst is OK, except if there is a memory page crossing (the page size is assumed to be 512 bytes).
- During the vertical blanking interval, four or five consecutive requests to refresh the CLUT, that occur just after the Vsync event (five in the case of a memory page crossing).

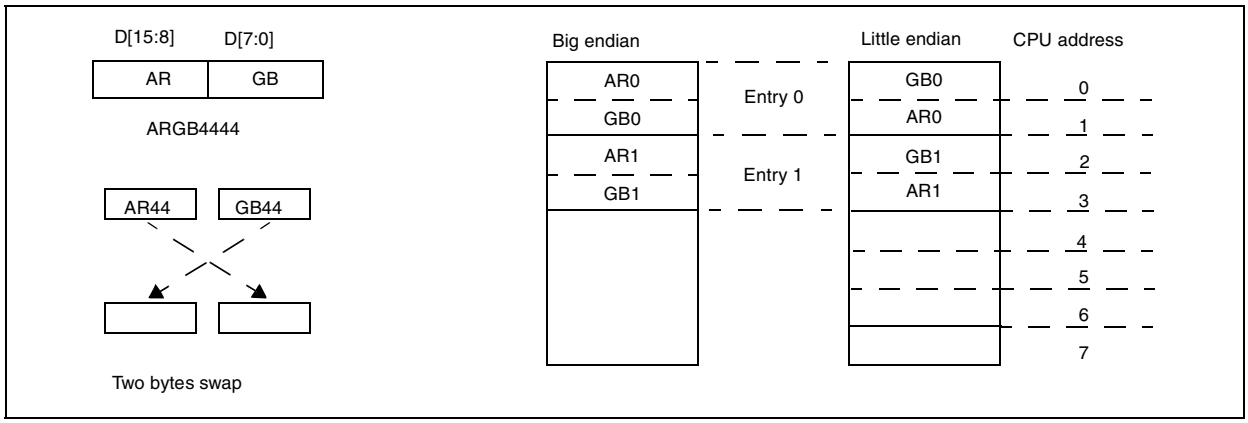
51.6.2 Unitary functions

Input formatter

There is no endianness issue concerning the bitmap, as the format is 8 bpp.

A register bit GAM_CUR_PKZ.BIGNOTLITTLE indicates if the palette is stored in big or little endian. It performs a two bytes swap on the 16-bit ARGB4444 entries.

Table 166: Endianness conversion for big endian CLUT entries



Clipping window

A rectangular clipping window can be defined. Outside this window, the cursor pattern, even if defined, is forced to be transparent (alpha = 0).

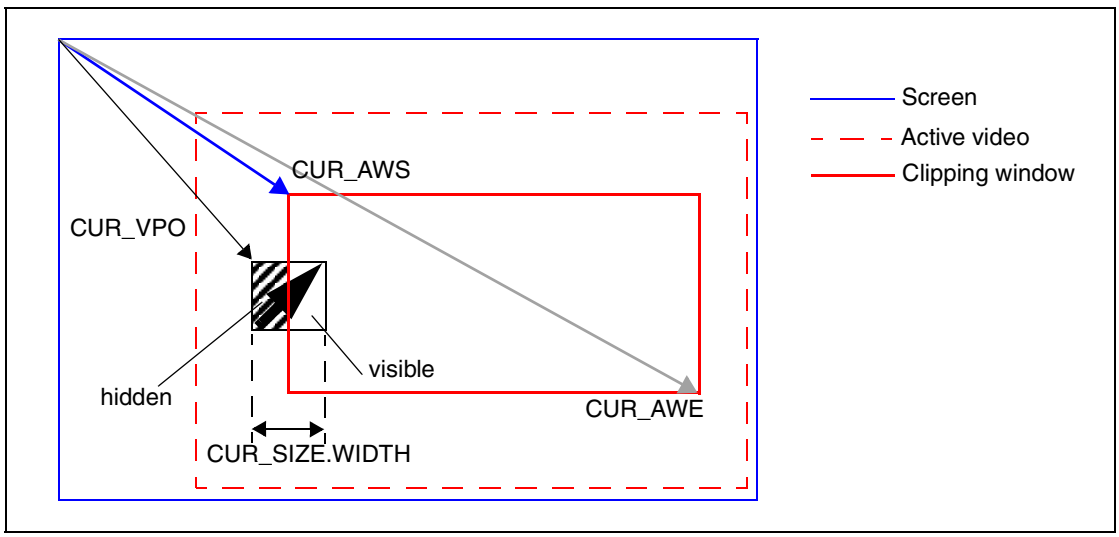
The clipping window is defined using two registers: GAM_CUR_AWS (active window start) and GAM_CUR_AWE (active window end). This is specified with respect to the frame numbering convention (such as a GDP or VID viewport).

Figure 180 illustrates the use of the hardware cursor clipping window. Bits XDO and YDO are signed (register GAM_CUR_VPO), so that the top-left corner of the cursor pattern can be positioned anywhere in the screen, even in horizontal and vertical blanking.

There is no enable bit for the hardware window: the feature is disabled by programming the window to match the screen size.

Note: The same screen look can be obtained without using the hardware window. As the address generator supports a pitch parameter (GAM_CUR_PMP) that is different from the cursor width, it is possible to use the base address GAM_CUR_CML.CLUT_MEM_ADDR and the width (GAM_CUR_SIZE.WIDTH) parameters in conjunction.

Figure 180: Cursor clipping window



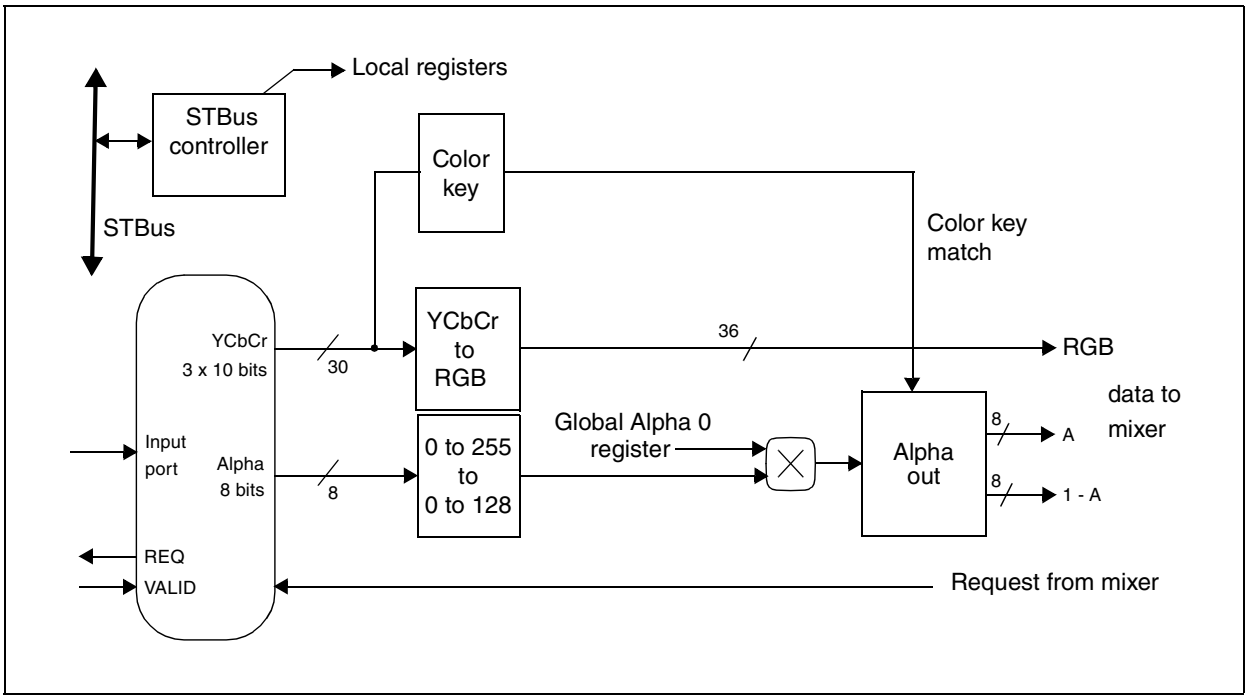
51.7 Video plug (VID)

The STx7100 compositor has two video input plugs, one main (VID1) and one auxiliary (VID2). These inputs accept 3 x 10 bit YCbCr video from the main and auxiliary display processors respectively. Figure 181 shows a functional block diagram of the video input plug. The Alpha channel from the input port is not used, so that the alpha weights provided to the mixers are purely based on the global alpha values in the GAM_VIDn_ALP registers.

The plug provides the following features:

- Color key capability,
- Global alpha blending (combined with the per pixel alpha channel, if any),
- Vertical and horizontal edge smoothing.

Figure 181: VID functional block diagram



51.7.1 Color data path

- Programmable matrix: all input components are 10-bit signed. The nominal range is:
  -448 ≤Y ≤+373
  -420 ≤Cb, Cr ≤+420
  Prior to the matrix, a (512 - GAM_VIDn_MPR1.Y_OS) offset is added, in order to recover an unsigned luminance component.

Fixed video color space matrix:

Table 167: 601 colorimetry / integer matrix / digital range

RGB to YCbCr integer matrix as implemented												
Y	=		77/256 306/1024	xR	+	150/256 601/1024	xG	+	29/256 117/1024	xB		
Cb	=	-	44/256 177/1024	xR	-	87/256 347/1024	xG	+	131/256 523/1023	xB	+	128
Cr	=		131/256 523/1024	xR	-	110/256 438/1024	xG	-	21/256 85/1024	xB	+	128

Table 168: 709 colorimetry / integer matrix / digital range

RGB to YCbCr integer matrix as implemented												
Y	=		54/256 218/1024	xR	+	183/256 732/1024	xG	+	19/256 74/1024	xB		
Cb	=	-	30/256 120/1024	xR	-	101/256 404/1024	xG	+	131/256 524/1024	xB	+	128
Cr	=		131/256 524/1024	xR	-	119/256 476/1024	xG	-	12/256 48/1024	xB	+	128

The matrices are providing 12 bit signed gamma-corrected RGB components, with a dynamic range (-2048 to 2047).

The color key processor operates in the YCbCr color space, according to the following equation:

```
COLOR_KEY_MATCH =
((CRin < CRmin) OR (CRin > CRmax)) AND (CRoutside = True) AND (EnableCR = True)
OR (CRmin <= CRin <= CRmax) AND (CRoutside = False) AND (EnableCR = True)
OR (EnableCR = False)
AND
((Yin < Ymin) OR (Yin > Ymax)) AND (Youtside = True) AND (EnableY = True)
OR ( Ymin <= Yin <= Ymax) AND (Youtside = False) AND (EnableY = True)
OR (EnableY = False)
AND
((CBin < CBmin) OR (CBin > CBmax)) AND (CBoutside = True) AND (EnableCB = True)
OR ( CBmin <= CBin <= CBmax) AND (CBoutside = False) AND (EnableCB = True)
OR (EnableCB = False)
```

*Note: The min and max registers are 8-bit registers (*GAM_VID*n*_KEY1/2*). Thus, the two LSBs of the incoming components are ignored by the color key processor.*

51.7.2 Alpha data path

The alpha data path provides a programmable global alpha value for the video layer.

The alpha out block then performs the following processing:

- Alpha borders: the alpha component can be divided by two on the first and last viewport columns (GAM_VIDn_CTRL.AB_H) and/or on the first and last viewport lines (AB_V).
- If the color key is enabled and there is a match, the alpha output is forced to 0, so that the pixel is transparent.
- Both alpha and 128-alpha are provided to the next processing blocks (digital mixer).

51.8 Alpha plane (ALP)

This is an alpha-only version of the GDP pipeline, providing an 8-bit alpha bus. Only the differences with the GDP specification are provided here.

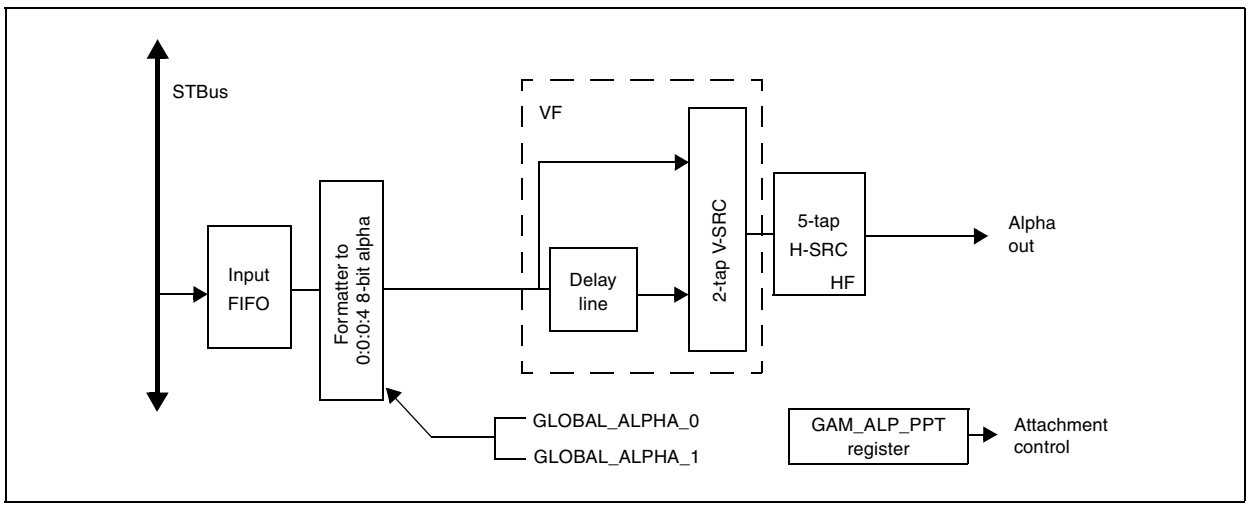
This alpha component can then be attached to a color layer (video or graphics), and is combined with the alpha component of the layer. This occurs in the digital mixer block. The features are:

- Link-list-based display engine, for multiple viewport capabilities.
- A1 and A8 formats supported (0 to 128 or 0 to 255 range for A8).
- 5-tap horizontal sample rate converter, for horizontal upsampling. This can be used to adapt the pixel aspect ratio.

  The resolution is 1/8th pixel (polyphase filter with 8 subpositions)

The ALP output format is an 8-bit alpha value, ranging from 0 to 128.

Figure 182: ALP functional block diagram



51.8.1 General description

ALP overview

ALP can handle multiple-viewport display, from a display instruction list (link list) stored in the external memory. Typically, an alpha plane viewport should be set up to match exactly the video or graphics viewport it is attached to.

The alpha plane can be used only on MIX1.

For the viewport and window definition, refer to the relevant GDP subsection. Outside a viewport, the alpha plane pipeline outputs 128.

How to start the display and modify the display parameters

This is similar to GDP; refer to the relevant GDP subsection.

Bandwidth considerations

In terms of bandwidth requirement (BR), the general formula to estimate the ALP weight on a given system is the following:

BR (in Mbyte/s) = (pixel frequency in MHz) x (number of bits per pixel) x (horizontal resampling factor) / 8

Examples:

- 601 rate / A1 / x1: BR = 13.5 x 1 x 1 / 8 = 1.69 Mbyte/s
- PAL SQ rate / A8 / x1: BR = 14.75 x 8 x 1 / 8 = 14.75 Mbyte/s
- 601 rate / A8 / zoom out x2: BR = 13.5 x 8 x 2 / 8 = 27 Mbyte/s

51.8.2 Unitary functions

Input formatter

This sub-block converts the 128-bit STBus data bus, into an internal 8-bit alpha bus. The clock rate for the pixel bus is PIXCLK.

There is no endianness issue concerning the bitmap, as the format is either 1 or 8 bpp.

Bit GAM_ALP_PKZ.BIGNOTLITTLE (that is not included in the viewport node) indicates if the nodes and filter coefficients are stored in big or little endian. This bit should reflect the endianness of the host CPU. It performs a four-byte swap on the 32-bit word fetched when loading the nodes and the filter coefficients.

Table 169: 8-bit alpha bus format, at the formatter output

Input color format	Alpha
A1	GLOBAL_ALPHA_0 if A1 = 0 GLOBAL_ALPHA_1 if A1 = 1
A8	A8[a]

a. The ALP supports either 0 to 128 or 0 to 255 8-bit alpha. In case 255 range is used, the input alpha value is converted in to a 0 to 128 component, using the following formula:
$A_{0..128} = (A_{0..255} + 1) \times 2^{-1}$
In A1 mode, when the bit is 0, then the alpha output is forced to the GLOBAL_ALPHA_0 value, when the bit is 1, the alpha output is forced to the GLOBAL_ALPHA_1 value.

The output of the formatter can be optionally complemented: (128 - Alpha) operator is applied if GAM_ALP_CTRL.REV_ALPHA_EN is set to 1.

2D-resize, vertical filter and horizontal filter

This is similar to GDP; refer to Section 51.5.1: *General description of GDPs on page 538*.

52 Compositor registers

52.1 Introduction

Register addresses are provided as *CompositorBaseAddress* + *SubsystemOffset* + register offset.

The *CompositorBaseAddress* is:

0x1920 A000.

Subsystems and *SubsystemOffset*s are shown in Table 170.

Table 170: Compositor subsystems and subsystem register offsets

Subsystem	Offset	
	Name	Value
CURSOR	*CUROffset*	0x0000
GDP1	*GDP1Offset*	0x0100
GDP2	*GDP2Offset*	0x0200
ALP	*ALPOffset*	0x0600
VID1	*VID1Offset*	0x0700
VID2	*VID2Offset*	0x0800
MIX1	*MIX1Offset*	0x0C00
MIX2	*MIX2Offset*	0x0D00

Each subsystem register block occupies a 64 x 32-bit word address range.

Table 171 shows the full list of compositor registers, in a matrix format. Each register name can be built up from the table as follows: GAM_*subsystem_function*, for example GAM_CUR_CTRL.

Table 171: Compositor register summary

Offset (0x)	Subsystem					
	CUR	**GDPn**	**VIDn**	**ALP**	**MIX1**	**MIX2**
0000	CTRL	CTRL	**CTRL**	CTRL	CTRL	CTRL
0004		AGC	ALP	AGC	BKC	
0008		HSRC		HSRC		
000C	VPO	VPO	VPO	VPO	BCO	
0010		VPS	VPS	VPS	BCS	
0014	PML	PML		PML		
0018	PMP	PMP		PMP		
001C	SIZE	SIZE		SIZE		
0020	CML	VSRC		VSRC		
0024		NVN		NVN		
0028	AWS	KEY1	KEY1		AVO	AVO
002C	AWE	KEY2	KEY2		AVS	AVS
0030		HFP		HFP		
0034		PPT		PPT	CRB	
0038					ACT	ACT
003C						
0040		HFC0		HFC0		
0044		HFC1		HFC1		
0048		HFC2		HFC2		
004C		HFC3		HFC3		
0050		HFC4		HFC4		
0054		HFC5		HFC5		
0058		HFC6		HFC6		
005C		HFC7		HFC7		
0060		HFC8		HFC8		
0064		HFC9		HFC9		
0070			BC			
0074			TINT			
0078			CSAT			
00FC	PKZ	PKZ		PKZ		

Complete descriptions of these registers are given in the following sections.

52.2 Register maps

Table 172: Mixers 1 and 2, cursor, video-plug and capture register map

Description	Register	Offset	Type
Control	GAM_MIX1_CTRL	0x0000	R/W
	GAM_MIX1_CRB	0x0034	
	GAM_MIX1_ACT	0x0038	
Background color	GAM_MIX1_BKC	0x0004	
	GAM_MIX1_BCO	0x000C	
	GAM_MIX1_BCS	0x0010	
Active video area definition	GAM_MIX1_AVO	0x0028	
	GAM_MIX1_AVS	0x002C	
Control	GAM_MIX2_CTRL	0x0000	
	GAM_MIX2_ACT	0x0038	
Active video area definition	GAM_MIX2_AVO	0x0028	
	GAM_MIX2_AVS	0x002C	
Control register	GAM_CUR_CTRL	0x0000	
Viewport offset	GAM_CUR_VPO	0x000C	
Pixmap	GAM_CUR_PML	0x0014	
	GAM_CUR_PMP	0x0018	
	GAM_CUR_SIZE	0x001C	
	GAM_CUR_CML	0x0020	
Clipping window	GAM_CUR_AWS	0x0028	
	GAM_CUR_AWE	0x002C	
Miscellaneous	GAM_CUR_PKZ	0x00FC	
Control register	GAM_VIDn_CTRL	0x0000	
Alpha blending	GAM_VIDn_ALP	0x0004	
Video viewport definition	GAM_VIDn_VPO	0x000C	
	GAM_VIDn_VPS	0x0010	
Color key control	GAM_VIDn_KEY1	0x0028	
	GAM_VIDn_KEY2	0x002C	
Picture control	GAM_VIDn_BC	0x0070	
	GAM_VIDn_TINT	0x0074	
	GAM_VIDn_CSAT	0x0078	
Reserved	-	0x0030 - 0x003C	

Table 173: Generic display pipeline register map

Description	Register	Offset	Type	128-bit word alignment	
Control register	GAM_GDPn_CTRL	0x0000	RO/LLU	Node 128-bit word 1	Memory node
Blending and dynamic range control	GAM_GDPn_AGC	0x0004			
Horizontal sample rate converter control	GAM_GDPn_HSRC	0x0008			
Viewport definition	GAM_GDPn_VPO	0x000C			
	GAM_GDPn_VPS	0x0010	RO/LLU	Node 128-bit word 2	
Pixmap-related settings	GAM_GDPn_PML	0x0014			
	GAM_GDPn_PMP	0x0018			
	GAM_GDPn_SIZE	0x001C			
Reserved	-	0x0020	-	Node 128-bit word 3	
Pointer to the next viewport node	GAM_GDPn_NVN	0x0024	RO/LLU		
Color key	GAM_GDPn_KEY1	0x0028			
	GAM_GDPn_KEY2	0x002C			
Pointer the horizontal filter coefficients	GAM_GDPn_HFP	0x0030	RO/LLU	Node 128-bit word 4	
Viewport properties	GAM_GDPn_PPT	0x0034			
Reserved		0x0038	-		
		0x003C			
Horizontal filter coefficients	GAM_GDPn_HFCn				
	GAM_GDPn_HFC0	0x0040	RO/LLU	HF coefficient 128-bit word 1	Filter coefficient structure
	GAM_GDPn_HFC1	0x0044			
	GAM_GDPn_HFC2	0x0048			
	GAM_GDPn_HFC3	0x004C			
	GAM_GDPn_HFC4	0x0050	RO/LLU	HF coefficient 128-bit word 2	
	GAM_GDPn_HFC5	0x0054			
	GAM_GDPn_HFC6	0x0058			
	GAM_GDPn_HFC7	0x005C			
	GAM_GDPn_HFC8	0x0060	RO/LLU	HF coefficient 128-bit word 3	
	GAM_GDPn_HFC9	0x0064			
Reserved	-	0x0068 - 0x006C	-		
STBus protocol/maximum packet size	GAM_GDPn_PKZ	0x00FC	R/W	-	

Table 174: Alpha plane register map

Description	Register	Offset	Type	128-bit word alignment	
Control register	GAM_ALP_CTRL	0x0000	RO/LLU	Node 128-bit word 1	Memory node
	GAM_ALP_ALP	0x0004			
Horizontal sample rate converter control	GAM_ALP_HSRC	0x0008			
Viewport definition	GAM_ALP_VPO	0x000C			
	GAM_ALP_VPS	0x0010	RO/LLU	Node 128-bit word 2	
Pixmap-related settings	GAM_ALP_PML	0x0014			
	GAM_ALP_PMP	0x0018			
	GAM_ALP_SIZE	0x001C			
Reserved	-	0x0020	-	Node 128-bit word 3	
Pointer to the next viewport node	GAM_ALP_NVN	0x0024	R/W/LLU		
Reserved	-	0x0028	-		
		0x002C			
Pointer the horizontal filter coefficients	GAM_ALP_HFP	0x0030	RO/LLU	Node 128-bit word 4	
Viewport properties	GAM_ALP_PPT	0x0034			
Reserved	-	0x0038	-		
		0x003C			
Horizontal filter coefficients	GAM_ALP_HFCn				
	GAM_ALP_HFC0	0x0040	RO/LLU	HF coefficient. 128-bit word 1	Filter coefficient structure
	GAM_ALP_HFC1	0x0044			
	GAM_ALP_HFC2	0x0048			
	GAM_ALP_HFC3	0x004C			
	GAM_ALP_HFC4	0x0050	RO/LLU	HF coefficient 128-bit word 2	
	GAM_ALP_HFC5	0x0054			
	GAM_ALP_HFC6	0x0058			
	GAM_ALP_HFC7	0x005C			
	GAM_ALP_HFC8	0x0060	RO/LLU	HF coefficient 128-bit word 3	
	GAM_ALP_HFC9	0x0064			
Reserved		0x0068 - 0x006C	-		
STBus protocol / packet maximum size	GAM_ALP_PKZ	0x00FC	R/W	Not relevant	

52.3 Register descriptions

GAM_ALP_CTRL ALP control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAIT_NEXT_VSYNC	HFILTER_UPDATE_EN	REV_ALPHA_EN	Reserved	A1_INBYTE_ORDER	A1_BYTE_START			Reserved													H_RESIZE_EN	Reserved				ALPHA_RANGE	ALPHA_FORMAT				

Address: *CompositorBaseAddress* + *ALPOffset* + 0x00

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the operating mode of the ALP pipe, for the current viewport.

[31] **WAIT_NEXT_VSYNC**
0: The next node (as specified by GAM_ALP_NVN) is immediately loaded
1: ALP pipeline waits for the next VSync event before it loads the next node

[30] **HFILTER_UPDATE_EN**: This bit is taken into account for any viewport node within a frame (field).
0: The coefficients for the H filter are not loaded
1: The coefficients for the H filter are updated from memory (see GAM_ALP_HFP)

[29] **REV_ALPHA_EN**: In reverse mode, (128-alpha) operator is applied at the pipeline input.
0: Alpha mode 1: Reverse alpha mode

[28] **Reserved**

[27] **A1_INBYTE_ORDER**: A1 format only: specifies sample ordering inside a byte
0: Screen most right sample in the MSB 1: Screen most right sample in the LSB

[26:24] **A1_BYTE_START**: A1 format only: specifies the bit location of the first alpha sample, in the first byte
000: First 1-bit alpha sample is bit 0 001: First 1-bit alpha sample is bit 1
- - - - - - - -
111: First 1-bit alpha sample is bit 7

[23:11] **Reserved**: Set to 0

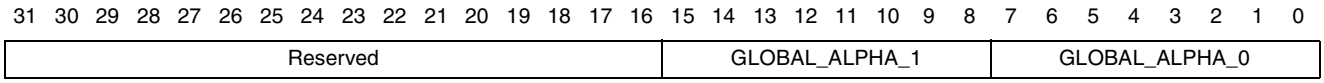
[10] **H_RESIZE_EN**: Horizontal resize enable
0: Disabled 1: Enabled

[9:6] **Reserved**

[5] **ALPHA_RANGE**: for A8 format, this bit specifies the alpha range.
0: 0 to 128 range (128 = opaque) 1: 0 to 255 range (255 = opaque)

[4:0] **ALPHA_FORMAT**: alpha format for the bitmap associated to the current viewport
11000: A1 (0x18) 11001: A8 (0x19)

GAM_ALP_ALP ALP alpha



Address: *CompositorBaseAddress* + *ALPOffset* + 0x04

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

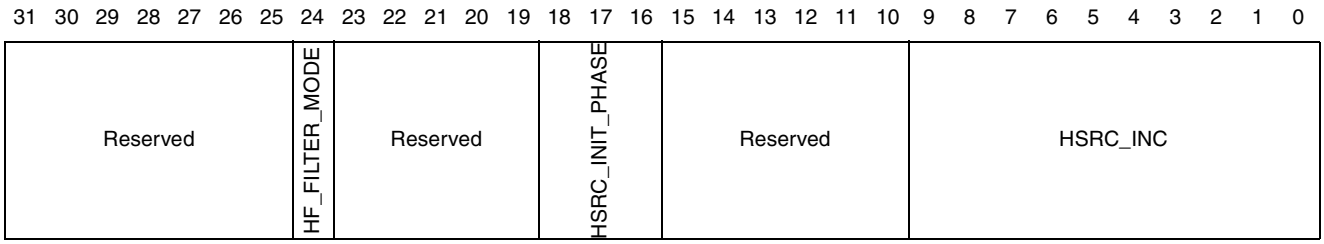
Description: This register provides the 2 8-bit alpha values to be used when A1 format is selected at the pipeline input. This register has no effect when A8 is selected.

[31:16] **Reserved**

[15:8] **GLOBAL_ALPHA_1**: For A1 format, this 8-bit value is used if the input bit is one. The register range is 0 to 128. (0: Fully transparent, 128: Fully opaque)

[7:0] **GLOBAL_ALPHA_0**: For A1 format, this 8-bit value is used if the input bit is zero. The register range is 0 to 128. (0: Fully transparent, 128: Fully opaque)

GAM_ALP_HSRC ALP horizontal sample rate converter



Address: *CompositorBaseAddress* + *ALPOffset* + 0x08

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the configuration for the horizontal sample rate converter. The alpha plane HSRC should only be programmed for upscaling.

[31:25] **Reserved**

[24] **HF_FILTER_MODE**
0: Only horizontal resizing
1: Filter is enabled

[23:19] **Reserved**

[18:16] **HSRC_INIT_PHASE**: The horizontal sample rate converter state-machine initial phase, 0 to 7

[15:10] **Reserved**

[9:0] **HSRC_INC**: The horizontal sample rate converter state-machine increment, in 2.8 format

GAM_ALP_VPO ALP viewport offset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDO											Reserved				XDO											

Address: *CompositorBaseAddress* + *ALPOffset* + 0x0C

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the x, y location of the viewport top-left pixel, with respect to the current video timebase.

[31:27] **Reserved**

[26:16] **YDO**: Y location for the first line of the viewport (top), with respect to frame numbering

[15:12] **Reserved**

[11:0] **XDO**: X location for the first pixel of the viewport (left)

GAM_ALP_VPS ALP viewport stop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDS											Reserved				XDS											

Address: *CompositorBaseAddress* + *ALPOffset* + 0x10

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the x, y location of the viewport bottom-right pixel, with respect to the current video timebase.

[31:27] **Reserved**

[26:16] **YDS**: Y location for the last line of the viewport (bottom), with respect to frame numbering

[15:12] **Reserved**

[11:0] **XDS**: X location for the last pixel of the viewport (right)

GAM_ALP_PML ALP pixmap memory location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64MB_BANK						PIXMAP_ADDR																									

Address: *CompositorBaseAddress* + *ALPOffset* + 0x14

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

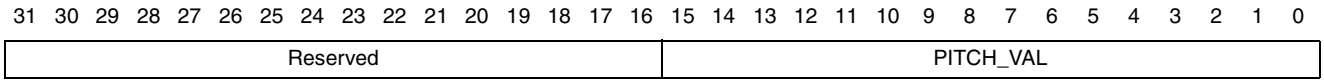
Reset: 0

Description: This register contains the memory location for the first alpha component to be read (top-left corner).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:0] **PIXMAP_ADDR**: First pixel byte address, in the selected 64 Mbyte bank.
Note: The whole bitmap to be displayed must be totally included in the same bank.

GAM_ALP_PMP ALP pixmap memory pitch

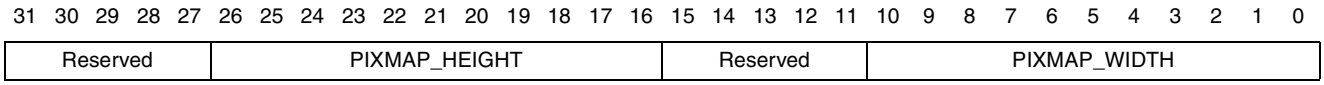


Address: *CompositorBaseAddress* + *ALPOffset* + 0x18
Type: Read/link list update
Buffer: Double-bank, automatic hardware toggle
Reset: 0
Description: This register contains the memory pitch for the alpha bitmap, as stored in the memory.

[31:16] **Reserved**

[15:0] **PITCH_VAL**: Memory pitch for the displayed pixmap
Note: the pitch is the distance inside the memory, in bytes, between two vertically adjacent pixels.

GAM_ALP_SIZE ALP pixmap size



Address: *CompositorBaseAddress* + *ALPOffset* + 0x1C
Type: Read/link list update
Buffer: Double-bank, automatic hardware toggle
Reset: 0
Description: This register provides the size of the alpha bitmap attached to the viewport.

[31:27] **Reserved**

[26:16] **PIXMAP_HEIGHT**
Pixmap height in lines, being defined as the number of lines that must be read from memory for the current field in an interlaced display.

[15:11] **Reserved**

[10:0] **PIXMAP_WIDTH**: Pixmap width in pixels

GAM_ALP_NVN ALP next viewport node

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64MB_BANK						NEXT_NODE_ADDR																						Reserved			

Address: *CompositorBaseAddress* + *ALPOffset* + 0x24

Type: R/W/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register contains a memory pointer to the next viewport node to be executed within the link list.

Note: The CPU can directly write into this register; this is required at least to enable the DMA link-list process to start.

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:4] **NEXT_NODE_ADDR**: Memory location for the next node to be loaded (the node must be fully contained in the specified bank). 4 LSBs address bits are “don’t care”, because the node structure must be aligned on a 128-bit word boundary.

[3:0] **Reserved**

GAM_ALP_HFP ALP horizontal filter pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64MB_BANK						H_FILTER_PTR																						Reserved			

Address: *CompositorBaseAddress* + *ALPOffset* + 0x30

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register contains a memory pointer to the set of filter coefficients that must be used for the horizontal sample rate converter.

A new set of coefficients may be used for each individual viewport. The coefficients are loaded only if GAM_ALP_CTRL[30] = 1 (HFILTER_UPDATE_EN).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:4] **H_FILTER_PTR**
Memory location when to retrieve the filter coefficients (10 32-bit words that must be fully contained in the specified bank). 4 LSBs address bits are “don’t care”, because the filter coefficients structure must be aligned on a 128-bit word boundary.

[3:0] **Reserved**

GAM_ALP_PPT ALP properties

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								ALPHA_ ATTACHMENT				Reserved			

Address: *CompositorBaseAddress* + *ALPOffset* + 0x34

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: Provides the mixer(s) with special information on the viewport currently displayed.

[31:8] **Reserved**

[7:4] **ALPHA_ATTACHMENT**: This field indicates the layer the alpha plane must be combined with.

0000: no attachment
0001: attach to VID1
0010: Reserved
0011: attach to GDP1
0100: attach to GDP2
0101: Reserved
0110: Reserved
0111: Reserved
1XXX: Reserved

[3:0] **Reserved**

GAM_ALP_HFCn ALP HF coefficients

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

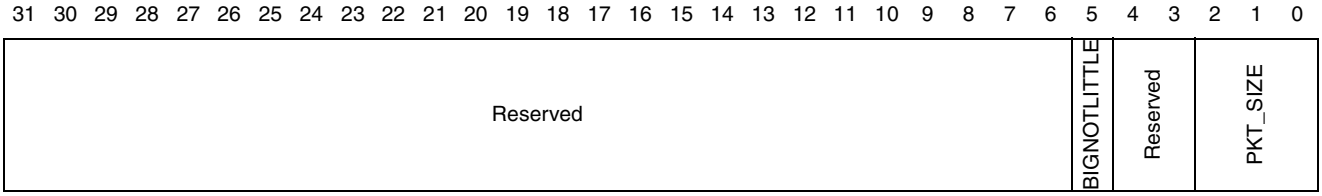
Address: *CompositorBaseAddress* + *ALPOffset* + register offset
HFC0: 0x40, HFC1: 0x44, HFC2: 0x48, HFC3: 0x4C, HFC4: 0x50, HFC5: 0x54, HFC6: 0x58, HFC7: 0x5C, HFC8: 0x60, HFC9: 0x64

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

GAM_ALP_PKZ ALP maximum packet size



Address: *CompositorBaseAddress* + *ALPOffset* + 0xFC

Type: R/W

Buffer: Immediate

Reset: 0x10

Description: This register is a 3-bit register for controlling the maximum size of a data packet during an STBus transaction. These bits should be set to 0 in the STx7100 device.

[31:6] **Reserved**

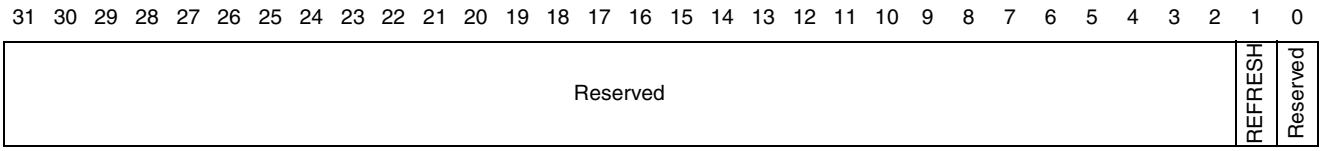
[5] **BIGNOTLITTLE**: CPU endianness
0: little endian CPU 1: big endian CPU

[4:3] **Reserved**: Set to 0

[2:0] **PKT_SIZE**: Maximum packet size during an STBus transaction
000: message size 001: 16 STBus words
010: 8 STBus words 011: 4 STBus words
100: 2 STBus words 101: 1 STBus words

This is a static register (not part of the link-list).

GAM_CUR_CTRL CUR control



Address: *CompositorBaseAddress* + *CUROffset* + 0x00

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the values for controlling the CUR core.

[31:2] **Reserved**

[1] **REFRESH**: 0: Disable cursor CLUT refresh 1: Enable cursor CLUT refresh

[0] **Reserved**

GAM_CUR_VPO Viewport offset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				YDO												Reserved			XDO												

Address: *CompositorBaseAddress* + *CUROffset* + 0x0C
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register provides the screen x, y location of the cursor pattern (upper-left pixel).

[31:28] **Reserved**

[27:16] **YDO**: Vertical start location for CUR viewport, wrt line frame-numbering

[15:13] **Reserved**

[12:0] **XDO**: Horizontal start location for CUR viewport, wrt VTG horizontal counter

GAM_CUR_PML First pixel memory location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUR_BANK_NUM						CUR_MEM_ADDR																									

Address: *CompositorBaseAddress* + *CUROffset* + 0x14
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the address for the first pixel memory location.

[31:26] **CUR_BANK_NUM**: 64 Mb bank number

[25:0] **CUR_MEM_ADDR**: First pixel memory byte address

GAM_CUR_PMP Pixmap memory pitch

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PIX_MEM_PITCH															

Address: *CompositorBaseAddress* + *CUROffset* + 0x18
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the memory pitch for the cursor bitmap.

[31:16] **Reserved**

[15:0] **PIX_MEM_PITCH**: CUR pixmap memory pitch in bytes

GAM_CUR_SIZE Pixmap memory size

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PIXMAP_HEIGHT								Reserved								PIXMAP_WIDTH							

Address: *CompositorBaseAddress* + *CUROffset* + 0x1C
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the horizontal and vertical size of the cursor pattern.

[31:24] **Reserved**

[23:16] **PIXMAP_HEIGHT**: Cursor pattern height (in pixel unit)

[15:8] **Reserved**

[7:0] **PIXMAP_WIDTH**: Cursor pattern width (in pixel units)

GAM_CUR_CML CLUT memory location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLUT_BANK_NUM								CLUT_MEM_ADDR																							

Address: *CompositorBaseAddress* + *CUROffset* + 0x20
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the memory pointer for the color look-up table (CLUT).

[31:26] **CLUT_BANK_NUM**: 64 Mb bank number

[25:0] **CLUT_MEM_ADDR**: CLUT memory byte address

GAM_CUR_AWS Active window start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					AWS_Y											Reserved				AWS_X											

Address: *CompositorBaseAddress* + *CUROffset* + 0x28
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the coordinates for the top-left location of the cursor active window.

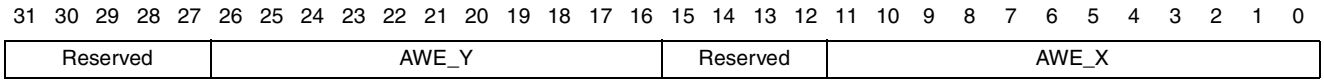
[31:27] **Reserved**

[26:16] **AWS_Y**: Vertical coordinate

[15:12] **Reserved**

[11:0] **AWS_X**: Horizontal coordinate

GAM_CUR_AWE Active window end



Address: *CompositorBaseAddress* + *CUROffset* + 0x2C

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: Contains the coordinates for the bottom-right location of the cursor active window.

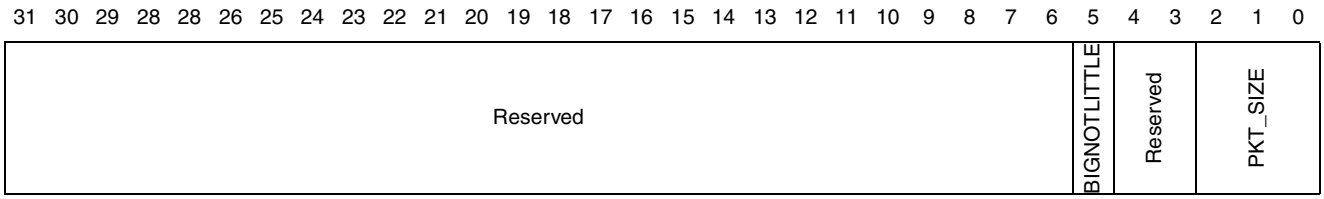
[31:27] **Reserved**

[26:16] **AWE_Y**: Vertical coordinate

[15:12] **Reserved**

[11:0] **AWE_X**: Horizontal coordinate

GAM_CUR_PKZ Packet size control



Address: *CompositorBaseAddress* + *CUROffset* + 0xFC

Type: R/W

Buffer: Immediate

Reset: 0

Description: Controls the packet size on an STBus transaction. In this device, the packet size must be set to 0.

[31:3] **Reserved**

[5] **BIGNOTLITTLE**: CLUT endianness
0: Little endian CLUT
1: Big endian CLUT

[4:3] **Reserved**

[2:0] **PKT_SIZE**: Packet size
000: Packet size = message size
001: Packet size = 16 x 128-bit words
010: Packet size = 8 x 128-bit words
011: Packet size = 4 x 128-bit words
100: Packet size = 2 x 128-bit words
101: Packet size = 1 x 128-bit words

GAM_GDPn_CTRL GDP control

31	30	29	28 27	26	25	24	23	22	21 20	19 18	17 16	15	14	13	12	11	10	9 8 7 6	5	4 3 2 1 0
WAIT_NEXT_VSYNC	HFILTER_UPDATE_EN	LSB_STUFFING_MODE	Reserved	CHROMA_FORMAT	601/709_SEL	PREMULT_FORMAT	BIGNOTLITTLE	Reserved	R/CR_COLOR_KEY_CFG	G/Y_COLOR_KEY_CFG	B/CB_COLOR_KEY_CFG	Reserved	COLOR_KEY_EN	ALPHA_VBORDER	ALPHA_HBORDER_EN	Reserved	H_RESIZE_EN	Reserved	ALPHA_RANGE	COLOR_FORMAT

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x00

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the operating mode of the GDP pipe for the current viewport display.

[31] **WAIT_NEXT_VSYNC**
0: The next node (as specified by GAM_GDPn_NVN) is immediately loaded
1: The GDP must wait for the next Vsync event before it loads the next node

[30] **HFILTER_UPDATE_EN**
This bit is taken into account for any viewport node within a frame (field)
0: Coefficients for the H filter are not loaded
1: Coefficients for the H filter are updated from memory (see GAM_GDPn_HFP)

[29] **LSB_STUFFING_MODE**
This configuration bit is used by the input formatter, in order to build the 32-bit internal pixel bus, whatever the input color format. When set, this bit preserves the full dynamic range 0.0 to 1.0, whatever the input format.
Note: The color key is affected by this setting.
0: If the number of bits per component at the input is fewer than 8, missing LSBs are filled with 0.
1: If the number of bits per component at the input is fewer than 8, missing LSBs are filled by the appropriate number of MSBs.

[28:27] **Reserved**

[26] **CHROMA_FORMAT**
0: The color space converter assumes Cb/Cr use offset binary representation
1: The color space converter assumes Cb/Cr use two's complement signed representation

[25] **601/709_SEL**
0: The color space conversion uses the 601 colorimetry
1: The color space conversion uses the 709 colorimetry

[24] **PREMULT_FORMAT**: Premultiplied format
0: RGB components are not premultiplied by the alpha component
1: RGB components are premultiplied by the per-pixel alpha component
Note: This is only meaningful for ARGB4444, ARGB8565 and ARGB8888 formats.

[23] **BIGNOTLITTLE**: 0: Little endian bitmap 1: Big endian bitmap

[23:22] **Reserved**

[21:20] **R/CR_COLOR_KEY_CFG**: R/CR color key configuration
x0: R/Cr component ignored (disabled = always match)
01: R/Cr enabled: match if (R/Cr_{min}<= R/Cr <= R/Cr_{max})
11: R/Cr enabled: match if ((R/Cr < R/Cr_{min}) or (R/Cr > R/Cr_{max}))

[19:18] **G/Y_COLOR_KEY_CFG**: G/Y color key configuration
x0: G/Y component ignored (disabled = always match)
01: G/Y enabled: match if (G/Y_{min}<= G/Y <= G/Y_{max}))
11: G/Y enabled: match if ((G/Y < G/Y_{MIN}) or (G/Y > G/Y_{MAX}))

[17:16] **B/CB_COLOR_KEY_CFG**: B/CB color key configuration
x0: B/Cb component ignored (disabled = always match)
01: B/Cb enabled: match if (B/Cb_{min}<= B/Cb <= B/Cb_{max}))
11: B/Cb enabled: match if ((B/Cb < B/Cb_{MIN}) or (B/Cb > B/Cb_{MAX}))

[15] **Reserved**

[14] **COLOR_KEY_EN**
0: Color key feature is disabled 1: Color key feature is enabled

[13] **ALPHA_VBORDER_EN**: Provides soft horizontal edges for the viewport, with flicker reduction effect.
0: Feature disabled
1: Alpha component is divided by 2 on the first line and the last line of the viewport

[12] **ALPHA_HBORDER_EN**: Provides soft vertical edges for the viewport.
0: Feature disabled
1: Alpha component is divided by 2 on the first and last columns of the viewport

[11] **Reserved**: Set to 0

[10] **H_RESIZE_EN**
0: Horizontal resize block is disabled 1: Horizontal resize block is enabled

[9:6] **Reserved**

[5] **ALPHA_RANGE**: for color format with an 8-bit alpha component (ARGB8888 / ARGB8565), this bit specifies the alpha range.
0: 0 to 128 range (128 = opaque) 1: 0 to 255 range (255 = opaque)

[4:0] **COLOR_FORMAT**: color format for the bitmap associated to the current viewport
0 0000: RGB565 (0x0) 0 0001: RGB888 (0x1)
0 0100: ARGB8565 (0x4) 0 0101: ARGB8888 (0x5)
0 0110: ARGB1555 (0x6) 0 0111: ARGB4444 (0x7)
1 0000: YCbCr888 (0x10) 1 0010: YCbCr4:2:2R (0x12)
1 0101: AYCbCr8888 (0x15)

GAM_GDPn_AGC GDP alpha gain constant

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONSTANT								GAIN								GLOBAL_ALPHA_1								GLOBAL_ALPHA_0							

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x04

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides display parameters, such as global translucency factors, black level adjustment, and dynamic range adjustment.

[31:24] **CONSTANT**

This 8-bit register is used to adjust the black level of the RGB signal at the output of the GDP block, between 0 and 25% of the total dynamic range (that is 10 bit, 0 to 1023).

A value of 0 sets the black level to 0%. A value of 255 sets the black level to 25%.

Warning: The gain register must be consistent with the black level programmed value, otherwise some saturation effects may occur.

[23:16] **GAIN**

This 8-bit register is used to adjust the dynamic range of the RGB components at the output of the GDP block, between 0 and 100%. A value of 128 corresponds to a 100% range

[15:8] **GLOBAL_ALPHA_1**

For any color format except ARGB1555, the Alpha 1 register is unused.

For ARGB1555 color format, this is the pixel transparency that is applied to the current pixel if the alpha bit within the pixel (bit 15) is 1.

The register range is 0 to 128. (0: fully transparent, 128: fully opaque)

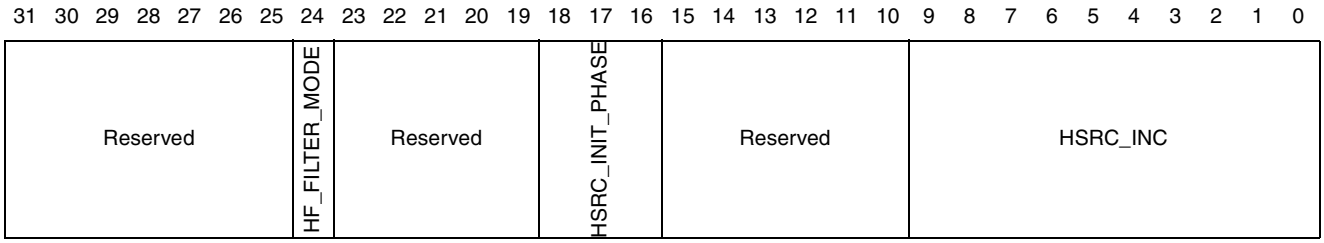
[7:0] **GLOBAL_ALPHA_0**

For any color format except ARGB1555, Alpha 0 is the global transparency associated with the viewport, that is combined with the per-pixel alpha component, if any.

For ARGB1555 color format, this is the pixel transparency that is applied to the current pixel if the alpha bit within the pixel (bit 15) is 0.

The register range is 0 to 128. (0: fully transparent, 128: fully opaque)

GAM_GDPn_HSRC GDP horizontal sample rate converter



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x08

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the configuration for the horizontal sample rate converter. The GDP*n* plane HSRC should only be programmed for upscaling.

[31:25] **Reserved**

[24] **HF_FILTER_MODE**
1: The filter is enabled, otherwise only horizontal resizing.

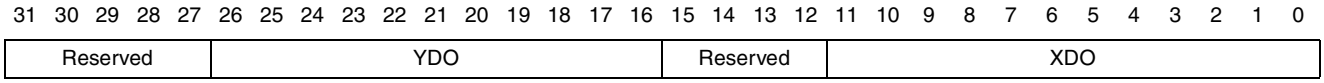
[23:19] **Reserved**

[18:16] **HSRC_INIT_PHASE**: Horizontal sample rate converter state-machine initial phase, 0 to 7.

[15:10] **Reserved**

[9:0] **HSRC_INC**: Horizontal sample rate converter state-machine increment, in 2.8 format.

GAM_GDPn_VPO GDP viewport offset



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x0C

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the x, y location of the viewport top-left pixel, with respect to the current video timebase.

[31:27] **Reserved**

[26:16] **YDO**: Y location for the first line of the viewport (top), with respect to frame numbering.

[15:12] **Reserved**

[11:0] **XDO**: X location for the first pixel of the viewport (left).

GAM_GDPn_VPS GDP viewport stop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDS											Reserved				XDS											

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x10

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the x, y location of the viewport bottom-right pixel, with respect to the current video timebase.

[31:27] **Reserved**

[26:16] **YDS**: Y location for the last line of the viewport (bottom), with respect to frame numbering

[15:12] **Reserved**

[11:0] **XDS**: X location for the last pixel of the viewport (right)

GAM_GDPn_PML GDP pixmap memory location

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64MB_bank						PIXMAP_ADDR																									

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x14

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register contains the memory location for the first pixel to be displayed (top-left corner).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:0] **PIXMAP_ADDR**: First pixel byte address, in the selected 64 Mbyte bank
Note: the whole bitmap to be displayed must be totally included into the same bank.

GAM_GDPn_PMP GDP pixmap memory pitch

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PITCH_VAL															

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x18

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

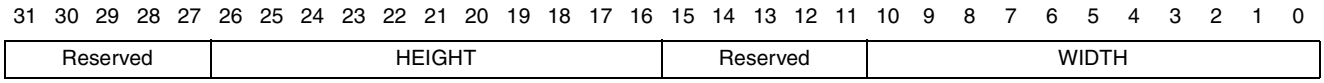
Reset: 0

Description: This register contains the memory pitch for the displayed pixmap, as stored in the memory.

[31:16] **Reserved**

[15:0] **PITCH_VAL**: Memory pitch for the displayed pixmap.
Note: the pitch is the distance inside the memory, in bytes, between two vertically adjacent pixels.

GAM_GDPn_SIZE GDP pixmap size



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x1C

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the size of the displayed pixmap attached to the viewport.

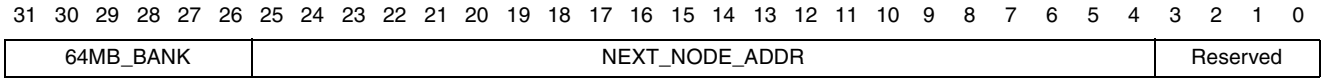
[31:27] **Reserved**

[26:16] **HEIGHT**: Pixmap height, in lines, being defined as the number of lines that must be read from memory for the current field in an interlaced display

[15:11] **Reserved**

[10:0] **WIDTH**: Pixmap width, in pixels

GAM_GDPn_NVN GDP next viewport node



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x24

Type: R/W/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register contains a memory pointer to the next viewport node to be displayed within the link list.

Note: The CPU can directly write into this register; this is required at least to enable the display link-list process to start.

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:4] **NEXT_NODE_ADDR**: 4 LSBs address bits are “don’t care”, because the node structure must be aligned on a 128-bit word boundary
Memory location for the next node to be loaded (the node must be fully contained in the specified bank.

[3:0] **Reserved**

GAM_GDPn_KEY1 Color keying - lower limit

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	R/CR_MIN	G/Y_MIN	B/CB_MIN

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x28

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register is a 24-bit register containing the values for the lower limit of the color range to be detected when using the color key feature.

[31:24] **Reserved**

[23:16] **B/CB_MIN**: Minimum value for the blue component in ARGBargb color formats; minimum value for the Cb component in YCbCr color formats

[15:8] **G/Y_MIN**: Minimum value for the green component in ARGBargb color formats; minimum value for the luminance component in YCbCr color formats

[7:0] **R/CR_MIN**: Minimum value for the red component in ARGBargb color formats; minimum value for the Cr component in YCbCr color formats

GAM_GDPn_KEY2 Color keying - upper limit

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	R/CR_MAX	G/Y_MAX	B/CB_MAX

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x2C

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register is a 24-bit register containing the values for the upper limit of the color range to be detected when using the color key feature.

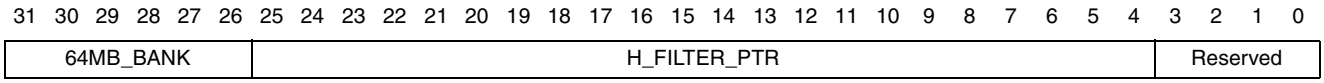
[31:24] **Reserved**

[23:16] **B/CB_MAX**: Maximum value for the blue component in ARGBargb color formats; maximum value for the Cb component in YCbCr color formats

[15:8] **G/Y_MAX**: Maximum value for the green component in ARGBargb color formats; maximum value for the luminance component in YCbCr color formats

[7:0] **R/CR_MAX**: Maximum value for the red component in ARGBargb color formats; maximum value for the Cr component in YCbCr color formats

GAM_GDPn_HFP GDP horizontal filter pointer



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x30

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register contains a memory pointer to the set of filter coefficients that must be used for the horizontal sample rate converter.

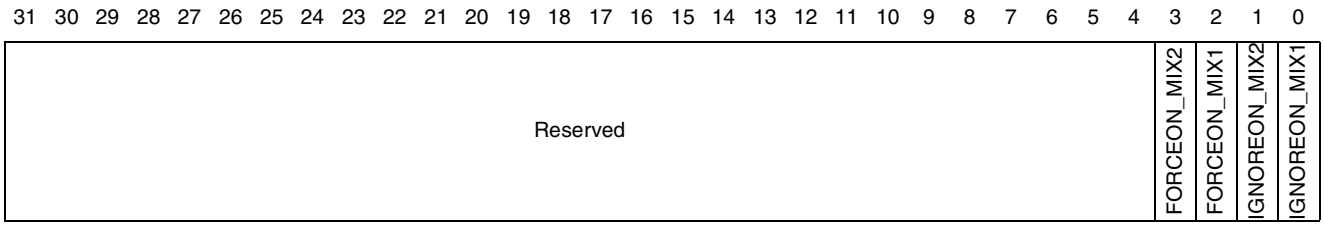
A new set of coefficients may be used for each individual viewport. The coefficients are loaded only if GAM_GDPn_CTRL[30] = 1 (HFILTER_UPDATE_EN).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:4] **H_FILTER_PTR**: 4 LSBs address bits are “don’t care”, because the filter coefficients structure must be aligned on a 128-bit word boundary.
Memory location when to retrieve the filter coefficients (ten 32-bit words that must be fully contained in the specified bank)

[3:0] **Reserved**

GAM_GDPn_PPT GDP properties



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x34

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides special information to the mixer(s), concerning the viewport currently displayed.

[31:4] **Reserved**

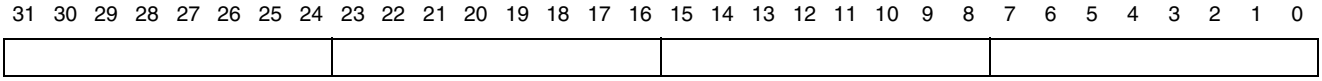
[3] **FORCEON_MIX2**: When set, this bit indicates to MIX2 that the current viewport must always be displayed, even out of the MIX2 active video area (defined with MIX2_AVO / MIX2_AVS registers).
Note: The GDP pipeline must be enabled in the MIX2_CTRL register.

[2] **FORCEON_MIX1**: When set, this bit indicates to MIX1 that the current viewport must always be displayed, even out of the MIX1 active video area (defined with MIX1_AVO / MIX1_AVS registers).
Note: The GDP pipeline must be enabled in the MIX1_CTRL register.

[1] **IGNOREON_MIX2**: When set, this bit indicates to MIX2 that the current viewport must not be displayed, although the GDP pipeline is enabled in the MIX2_CTRL register

[0] **IGNOREON_MIX1**: When set, this bit indicates to MIX1 that the current viewport must not be displayed, although the GDP pipeline is enabled in the MIX1_CTRL register

GAM_GDPn_HFCn GDP HF coefficients registers



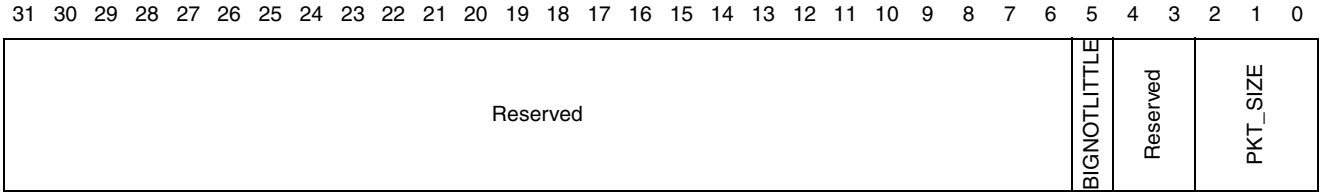
Address: *CompositorBaseAddress* + *GDPnOffset* + register offset
HFC0: 0x40, HFC1: 0x44, HFC2: 0x48, HFC3: 0x4C, HFC4: 0x50, HFC5: 0x54, HFC6: 0x58, HFC7: 0x5C, HFC8: 0x60, HFC9: 0x64

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

GAM_GDPn_PKZ GDP maximum packet size



Address: *CompositorBaseAddress* + *GDPnOffset* + 0xFC

Type: R/W

Buffer: Immediate

Reset: 0x10

Description: This register is a 3-bit register for controlling the maximum size of a data packet during an STBus transaction. These bits must be set to 0 in the STx7100 device.

[31:6] **Reserved**

[5] **BIGNOTLITTLE**: CPU endianness.
0: little endian CPU 1: big endian CPU

[4:3] **Reserved**
Must be set to 0.

[2:0] **PKT_SIZE**: Maximum packet size during an STBus transaction
000: message size 001: 16 STBus words
010: 8 STBus words 011: 4 STBus words
100: 2 STBus words 101: 1 STBus word

This is a static register (not part of the link-list).

GAM_MIX1_CTRL MIX control

31–17	15	14–10	9	8–5	4	3	2	1	0
Reserved	AP_DISP_EN	Reserved	CUR_DISP_EN	Reserved	GDP2_DISP_EN	GDP1_DISP_EN	Reserved	VID1_DISP_EN	BKC_DISP_EN

Address: *CompositorBaseAddress* + *MIX1Offset* + 0x00
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the values for controlling the MIX1 core.

[31:16] **Reserved**

[15] **AP_DISP_EN**: Alpha plane is enabled for attachment with a given layer that is enabled on the mixer output

[14:10] **Reserved**

[9] **CUR_DISP_EN**: CUR display is enabled on the mixer output

[8:5] **Reserved**

[4] **GDP2_DISP_EN**: GDP 2 display is enabled on the mixer output

[3] **GDP1_DISP_EN**: GDP 1 display is enabled on the mixer output

[2] **Reserved**

[1] **VID1_DISP_EN**: Video 1 display is enabled on the mixer output

[0] **BKC_DISP_EN**: Background color display is enabled on the mixer output

GAM_MIX1_BKC Background color

31–24	23–16	15–8	7–0
Reserved	RED_COMP	GREEN_COMP	BLUE_COMP

Address: *CompositorBaseAddress* + *MIX1Offset* + 0x04
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the background solid color components used by the MIX1 core.

[31:24] **Reserved**

[23:16] **RED_COMP**: Red component

[15:8] **GREEN_COMP**: Green component

[7:0] **BLUE_COMP**: Blue component

GAM_MIX1_BCO Background color offset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDO											Reserved				XDO											

Address: *CompositorBaseAddress* + *MIX1Offset* + 0x0C

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the X and Y coordinate values for defining the top-left corner of the background color rectangular area. Outside this window, the blanking color is displayed (black).

[31:27] **Reserved**

[26:16] **YDO**: Vertical start location of the MIX background color window, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDO**: Horizontal start location of the MIX background color window, wrt VTG horizontal counter

GAM_MIX1_BCS Background color stop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDS											Reserved				XDS											

Address: *CompositorBaseAddress* + *MIX1Offset* + 0x10

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

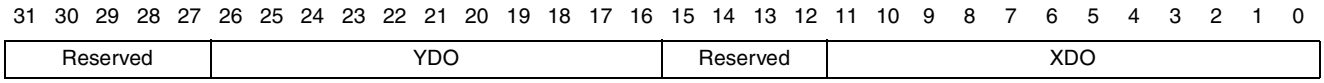
Description: This register contains the X and Y coordinate values for defining the bottom-right corner of the background color rectangular area. Outside this window, the blanking color is displayed (black).

[31:27] **Reserved**

[26:16] **YDS**: Vertical stop location of the MIX background color window, wrt line frame-numbering

[11:0] **XDS**: Horizontal stop location of the MIX background color window, wrt VTG horizontal counter

GAM_MIX1_AVO Active video offset



Address: *CompositorBaseAddress* + *MIX1Offset* + 0x28

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the X and Y coordinate values for defining the top-left corner of the active video rectangular area on the MIX1 output. Typically, it should be programmed according to the video standard currently in use.

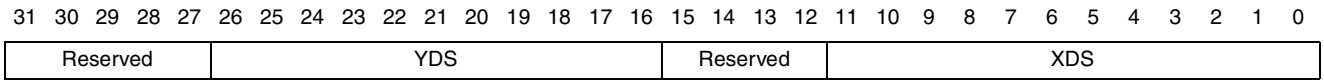
[31:27] **Reserved**

[26:16] **YDO**: Vertical start location of the MIX active video window, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDO**: Horizontal start location of the MIX active video window, wrt VTG horizontal counter

GAM_MIX1_AVS Active video stop



Address: *CompositorBaseAddress* + *MIX1Offset* + 0x2C

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the X and Y coordinate values for defining the bottom-right corner of the active video rectangular area on the MIX output. Typically, it should be programmed according to the video standard currently in use.

[31:27] **Reserved**

[26:16] **YDS**: Vertical stop location of the MIX active video window, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDS**: Horizontal stop location of the MIX active video window, wrt VTG horizontal counter

GAM_MIX1_CRB MIX 1 cross-bar control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							DEPTH3			DEPTH2			DEPTH1		

Address: *CompositorBaseAddress* + *MIX1Offset* + 0x34

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the values for controlling the mixer 7<>7 cross-bar.

[31:21] **Reserved**

[20:0] **DEPTHn**[2:0]: input identifier for depth *n* (depth 0: background color, depth 8: cursor, depth *n1* in front of depth *n2* if *n1* > *n2*)
000: Nothing displayed at depth *n*
001: VID1 displayed at depth *n*
010: Reserved
011: GDP1 displayed at depth *n*
100 to 111: Reserved

GAM_MIX1_ACT MIX active content flags control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CUR_ONVIDACTIVE	Reserved				GDP2_ONVIDACTIVE	GDP1_ONVIDACTIVE	Reserved	VID1_ONVIDACTIVE	BKC_ONVIDACTIVE	Reserved						CUR_ONGFXACTIVE	Reserved				GDP2_ONGFXACTIVE	GDP1_ONGFXACTIVE	Reserved	VID1_ONGFXACTIVE	BKC_ONGFXACTIVE

Address: *CompositorBaseAddress* + *MIX1Offset* + 0x38

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

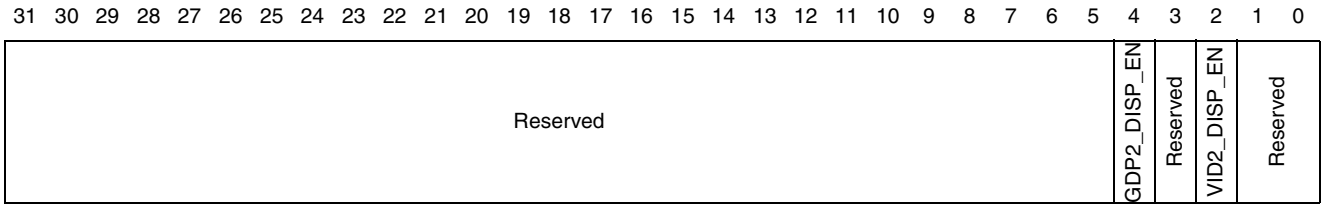
Reset: 0

Description: Defines the layers that contribute to the construction of GFXActive1 and VideoActive1 flags. A layer is considered as active for a given pixel if its associated alpha component is not zero.

XXX_ONVIDACTIVE
1: The XXX layer is taken into account for the VideoActive flag.

XXX_ONGFXACTIVE
1: The XXX layer is taken into account for the GFXActive flag.

GAM_MIX2_CTRL MIX control



Address: *CompositorBaseAddress* + *MIX2Offset* + 0x00

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the values for controlling the MIX 2 core.

[31:5] **Reserved**

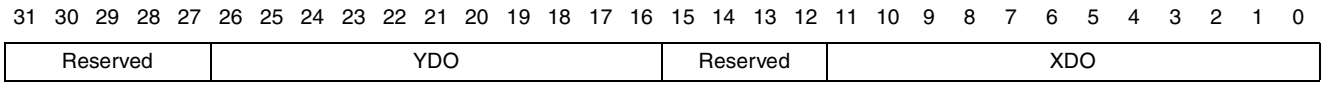
[4] **GDP2_DISP_EN**: GDP2 display enable
1: Enable GDP 2 display on the mixer output

[3] **Reserved**

[2] **VID2_DISP_EN**: VID2 display enable
1: Enable video 2 display on the mixer output

[1:0] **Reserved**

GAM_MIX2_AVO Active video offset



Address: *CompositorBaseAddress* + *MIX2Offset* + 0x28

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the X and Y coordinate values for defining the top-left corner of the active video rectangular area on the MIX2 output. Typically, it should be programmed according to the video standard currently in use.

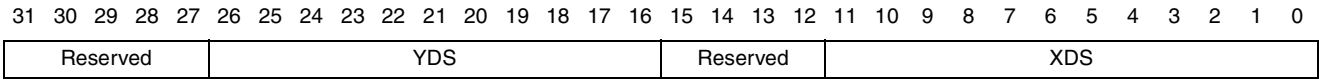
[31:27] **Reserved**

[26:16] **YDO**: Vertical start location of the MIX active video window, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDO**: Horizontal start location of the MIX active video window, wrt VTG horizontal counter

GAM_MIX2_AVS Active video stop



Address: *CompositorBaseAddress* + *MIX2Offset* + 0x2C

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the X and Y coordinate values for defining the bottom-right corner of the active video rectangular area on the MIX output. Typically, it should be programmed according to the video standard currently in use.

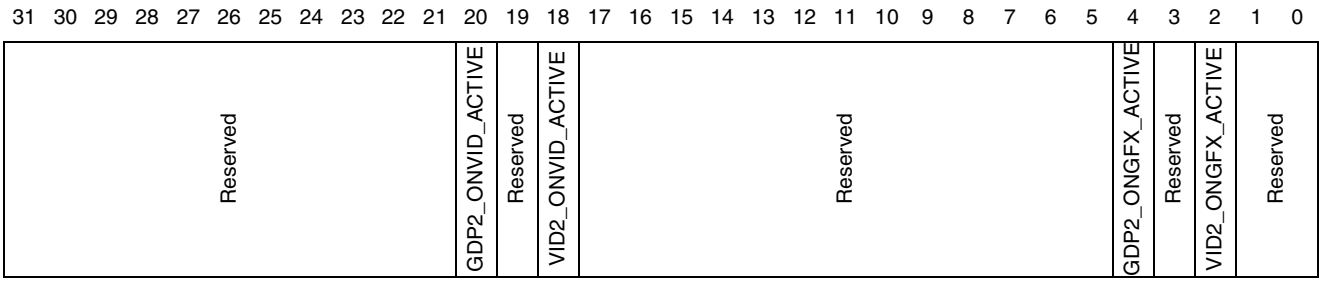
[31:27] **Reserved**

[26:16] **YDS**: Vertical stop location of the MIX active video window, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDS**: Horizontal stop location of the MIX active video window, wrt VTG horizontal counter

GAM_MIX2_ACT MIX active content flags control



Address: *CompositorBaseAddress* + *MIX2Offset* + 0x38

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: Defines the layers that contribute to the construction of GFXActive2 and VideoActive2 flags. A layer is considered as active for a given pixel if its associated alpha component is not zero.

XXX_ONVID_ACTIVE: If set to 1, the *XXX* layer is taken into account for the VideoActive flag

XXX_ONGFX_ACTIVE: If set to 1, the *XXX* layer is taken into account for the GFXActive flag

GAM_VIDn_CTRL VID control

31	30	29 28 27	26	25	24 23 22	21 20 19 18 17 16	15	14	13	12	11 10 9 8 7 6 5 4 3	2	1	0
IGNORE_MX2	IGNORE_MX1	Reserved	CFORM	709NOT601	Reserved	CKEY_CFG	Reserved	CKEY	AB_V	AB_H	Reserved	PSI_SAT_EN	PSI_TINT_EN	PSI_BC_EN

Address: *CompositorBaseAddress* + *VIDnOffset* + 0x00

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register contains the values for controlling VID core. Color key and alpha border features are not available for a video layer displayed via MIX2.

[31] **IGNORE_MX2**: When set, this bit indicates to MIX2 that the video data must be ignored when blending, even if enabled in MIX2_CTRL. (But the mixer request is generated)

[30] **IGNORE_MX1**: When set, this bit indicates to MIX1 that the video data must be ignored when blending, even if enabled in MIX1_CTRL. (But the mixer request is generated)

[29:27] **Reserved**

[26] **CFORM**: Chroma format (0=offset 128,1=signed)

[25] **709NOT601**: Colorimetry selection (0=601, 1=709)

[24:22] **Reserved**

[21:16] **CKEY_CFG**: Configuration of color key

[21:20] **VID_CTRL**[5:4]: Cr component
x0: Cr component ignored (disabled = always match)
01: Cr enabled: match if (Cr_{min} <= Cr <= Cr_{max})
11: Cr enabled: match if ((Cr < Cr_{min}) or (Cr > Cr_{max}))

[19:18] **VID_CTRL**[3:2]: Y component
x0: Y component ignored (disabled = always match)
01: Y enabled: match if (Y_{min} <= Y <= Y_{max})
11: Y enabled: match if ((Y < Y_{min}) or (Y > Y_{max}))

[17:16] **VID_CTRL**[1:0]: Cb component
x0: Cb component ignored (disabled = always match)
01: Cb enabled: match if (Cb_{min} <= Cb <= Cb_{max})
11: Cb enabled: match if ((Cb < Cb_{min}) or (Cb > Cb_{max}))

[15] **Reserved**

[14] **CKEY**: When this bit is set, the color key feature is enabled.

[13] **AB_V**: Enable AlphaVBorder: When this bit is set, the alpha component on the first and last lines of the viewport is divided by 2 (edge smoothing)

[12] **AB_H**: Enable AlphaHBorder: When this bit is set, the alpha component on the first and last columns of the viewport is divided by 2 (edge smoothing)

[11:3] **Reserved**

[2] **PSI_BC_EN**: Enable brightness and contrast correction with PSI block

[1] **PSI_TINT_EN**: Enable tint correction with PSI block

[0] **PSI_SAT_EN**: Enable chroma saturation with PSI block

GAM_VIDn_ALP Global alpha

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								GLOBAL_ALPHA_VAL							

Address: *CompositorBaseAddress* + *VIDnOffset* + 0x04
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the value for the VID global alpha.

[31:8] **Reserved**

[7:0] **GLOBAL_ALPHA_VAL**: Global blending coefficient for the video layer
The register range is 0 to 128. (0: fully transparent, 1:fully opaque)

GAM_VIDn_VPO Viewport offset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDO											Reserved				XDO											

Address: *CompositorBaseAddress* + *VIDnOffset* + 0x0C
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the X and Y coordinate values for defining the top-left corner of the video viewport.

[31:27] **Reserved**

[26:16] **YDO**: Vertical start location of the VID viewport, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDO**: Horizontal start location of the VID viewport, wrt VTG horizontal counter

GAM_VIDn_VPS Viewport stop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					YDS											Reserved				XDS											

Address: *CompositorBaseAddress* + *VIDnOffset* + 0x10
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the X and Y coordinate values for defining the bottom-right corner of the video viewport.

[31:27] **Reserved**

[26:16] **YDS**: Vertical stop location of the VID viewport, wrt line frame-numbering

[15:12] **Reserved**

[11:0] **XDS**: Horizontal stop location of the VID viewport, wrt VTG horizontal counter

GAM_VIDn_KEY1 Lower limit of the color keying range

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	CR_MIN	Y_MIN	CB_MIN

Address: *CompositorBaseAddress* + *VIDnOffset* + 0x28
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the values for the lower limit of the VID color keying range.

[31:24] **Reserved**

[23:16] **CR_MIN**: Minimum value of Cr component

[15:8] **Y_MIN**: Minimum value of Y component

[7:0] **CB_MIN**: Minimum value of Cb component

GAM_VIDn_KEY2 Upper limit of the color keying range

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	CR_MAX[7:0]	Y_MAX[7:0]	CB_MAX[7:0]

Address: *CompositorBaseAddress* + *VIDnOffset* + 0x2C
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the values for the upper limit of the VID color range.

[31:24] **Reserved**

[23:16] **CR_MAX**: Maximum value of Cr component

[15:8] **Y_MAX**: Maximum value of Y component

[7:0] **CB_MAX**: Maximum value of Cb component

GAM_VIDn_BC Brightness and contrast

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	CONTRAST	BRIGHTNESS

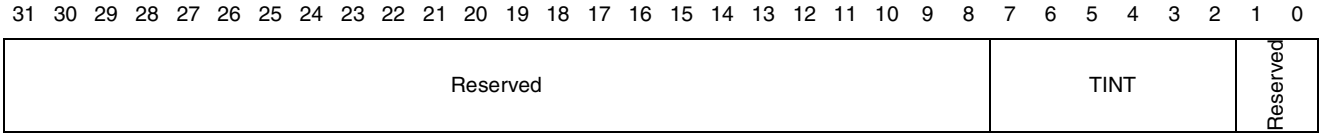
Address: *CompositorBaseAddress* + *VIDnOffset* + 0x70
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0x0000 8000
Description: Contains the values for the luminance gain and offset. Both values are coded on 8 bits and need to be adapted to the input luminance depth.

[31] **Reserved**

[15:8] **CONTRAST**: Adjust luminance dynamic range (unsigned value, 128 centered)

[7:0] **BRIGHTNESS**: Adjust luminance intensity (2's signed value)

GAM_VIDn_TINT Tint (hue/chroma phase)



Address: *CompositorBaseAddress* + *VIDnOffset* + 0x74

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0x0000 0000

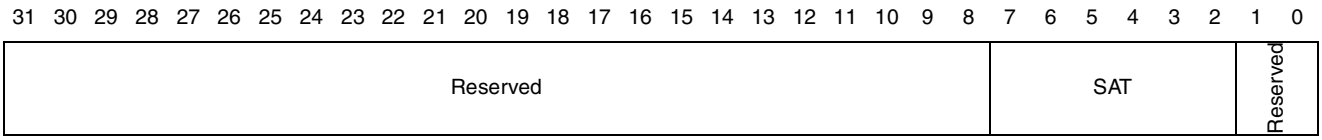
Description: This register codes the chrominance phase shift. This value is coded on 6 signed bits, the effective rotation angle t in radians will be TINTx2-6, This gives a resolution of 1/64 radian or 0.90 degrees (therefore, considering that 1 LSB~1degree is a reasonable approximation) and a maximum codable rotation of about +-π/6.

[31:8] **Reserved**

[7:2] **TINT**: Chrominance phase shift angle value (2's signed value)

[1:0] **Reserved**

GAM_VIDn_CSAT Chroma saturation



Address: *CompositorBaseAddress* + *VIDnOffset* + 0x78

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0x0000 0080

Description: This register contains the values for the chrominance gain (from 0 to 2). This value is coded on 6 bits, giving a gain step of 3% (from 0 to 1.97)

[31:8] **Reserved**

[7:2] **SAT**: Chrominance saturation value (unsigned, 32 centered)

[1:0] **Reserved**

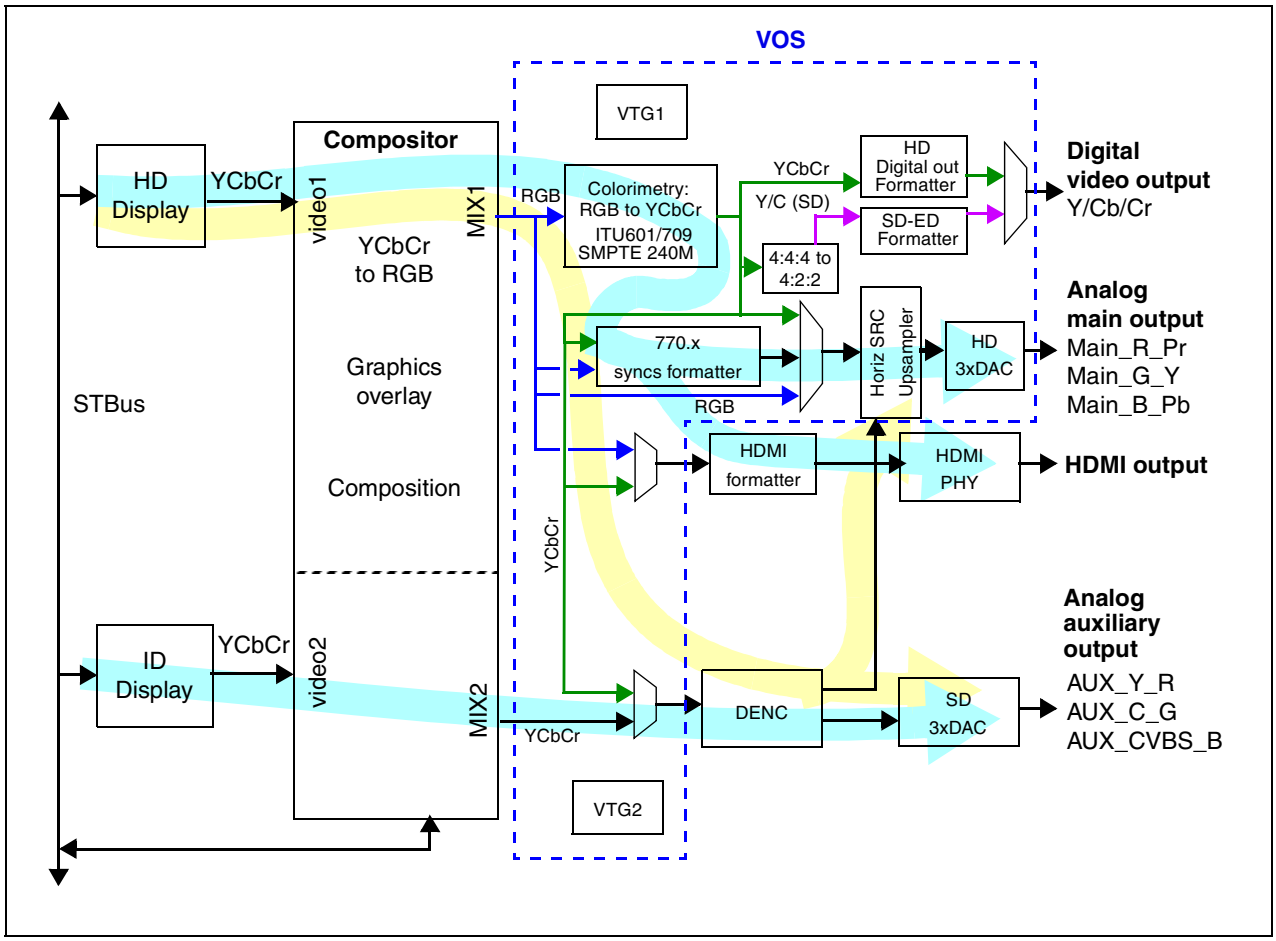
53 Video output stage (VOS)

The VOS receives video data from the main (mixer 1) and auxiliary (mixer 2) compositor outputs. This data is formatted and delivered to the DENC, HD and SD video DACs, HDMI formatter and digital video output (DVO).

53.1 Display and video output subsystem overview

Figure 183 shows the different units of the VOS.

Figure 183: Video output stage block diagram



53.1.1 Standard application: HD on main output and SD on auxiliary output

In this configuration, the two main data paths of the VOS (main and auxiliary) are used independently to display an HD video on the main output and an SD video on the auxiliary output. These two data paths are shown in light blue in the above Figure 183. In this chapter, HD means any format other than 480i (SD interlaced).

Main data path (HD)

The main data path is paced by VTG1 and uses the compositor mixer 1. The HD display pipe reads video frame buffers from memory which are delivered to the compositor mixer 1. The compositor mixer overlays and blends the video with graphics, manages the windowing, and generates RGB components. The RGB signals are either used, or passed through an RGB-to-YCbCr matrix supporting ITU-R BT601, ITU-R BT709, or SMPTE 240M colorimetry. Data then goes to HDMI and/or analog output. The HDMI comprises a digital formatting and HDCP

encryption unit followed by a physical interface (HDMI PHY). In front of the analog output, a programable formatter is provided allowing different output modes. The DACs are preceded by a sample-rate-converter (SRC) which upsamples the video signal to adapt the pixel rate to the video DAC sampling rate. This SRC has programmable coefficients. The video signal is interpolated using programmable set of coefficients designed to keep the maximum signal bandwidth. With a dedicated set of coefficients, a “constrained output” can be generated whereby the video bandwidth is cut down. For example, the analog video resolution can be degraded by 1/4, such that the full-resolution video is only available on the encrypted HDMI output.

Main Analog output

This output delivers HD format up to 3H. The analog video can be full-scale RGB, full-scale YPbPr, EIA-770.2 compatible YPbPr with embedded bi-level syncs or EIA 770.3 compatible YPbPr with embedded tri-level syncs. The video signal is reduced by 70% to conform to the EIA 770.x standards.

The following display standards are available:
SMPTE 274M (1920 x 1080i, 1125 total lines per frame), SMPTE 293M (720 x 483p, 525 total lines), SMPTE 295M (1920 x 1080i, 1250 total lines), SMPTE 296M (1280 x 720p, 750 total lines), ARIB-BS4 (525p, 750p, 1125i total lines).

Macrovision encoding is supported on this output when configured for 525p output.

Full scale YPbPr or RGB component without embedded sync information can also be generated.

Auxiliary analog output

This output delivers an SD interlaced decoded video, possibly with graphics overlay and ancillary data. For details, refer to Chapter 51: *Compositor on page 533* and Chapter 55: *Digital encoder (DENC) on page 644*.

HDMI output

This output supports 480p, 720p and 1080i formats as per HDMI specification.

Digital Video (DVO) output

This digital video output is provided mainly for debug and diagnosis. The following modes are available:

- HD (1080i or 720p): YCbCr 16-bit 4:2:2 with embedded sync (Digital SMPTE 274M, SMPTE 295M),
- SD progressive (480p): 4:2:2 with embedded sync (SMPTE 293M).

Auxiliary datapath (SD)

The auxiliary data path is paced by VTG2 and uses the compositor mixer 2. The ID display pipe reads video frame buffers from the memory which are delivered to the compositor mixer 2. The mixer overlays and blends the video with graphics, manages windowing, and generates YCbCr output components with CCIR 601 colorimetry. Those video components are then connected to the DENC, which generates Y, C and CVBS signals which are converted in analog with a triple SD DACs (auxiliary analog output).

The auxiliary analog video output signal is available either in composite (CVBS) or S-VHS (Y/C) format. The DENC supports the insertion of various VBI data on programmed lines, including close-caption and teletext.

53.1.2 Alternate application: SD on main and auxiliary outputs

An SD interlaced video can be delivered simultaneously to the HD video and SD video DACs. The data path used in this configuration is shown in yellow in Figure 183.

The data path is paced by VTG1 and uses the compositor mixer 1. The HD display pipe reads video frame buffers from the memory and feeds the compositor mixer 1 which overlays and blends the video with graphics, manages windowing, and generates RGB components, all in SD format. In this mode, VTG2, ID Display pipe and compositor mixer 2 are not used. The RGB components are then passed through an RGB to YCbCr conversion matrix (601 colorimetry) and are connected to the DENC inputs. The DENC provides 2 sets of outputs to drive both the main and auxiliary analog video DACs. Between the DENC and the HD video DACs is the upsampler (SRC), which must be configured with the correct set of coefficients to perform a x4 interpolation.

Main analog output

This output delivers an analog video in SD format, YPbPr with embedded syncs.

Auxiliary analog output

This output delivers an analog SD video in composite CVBS or S-VHS components.

HDMI output

This output delivers an interlaced SD video (480i) with pixel repetition to comply with HDMI specifications.

Digital video output

This output delivers a digital SD video (480i), YCbCr 8-bit 4:2:2 with or without embedded syncs (ITU-R BT 656 or 601).

53.2 HD DACs clocking

The HD DACs clock frequency is programmable and is set to a value high-enough relative to the video spectrum to ease the design of the external analog reconstruction filter (anti-aliasing). The frequency is set to a multiple of the display clock frequency (x2 or x4), and a programmable interpolation filter is provided to upsample the DAC input.

The main master pixel clock (generated by the clock generator B) is set to one of the following values:

- 148.5 MHz: used in applications where the main output is 1080i or 720p at 30 or 60 Hz.
- 148.5/1.001: used in applications where the main output is 1080i or 720p at 29.97 Hz or 59.94 Hz.
- 108 MHz: used in application where the main output is 480p at 59.94 Hz. The display clock in that case is 27 MHz and an x2 upsampling must be used.
- 108 MHz: used in applications where both outputs are 480i at 29.97 Hz. The display clock in that case is 13.5 MHz and an x4 upsampling must be used.
- 108 x 1.001 (108.108 MHz): used in applications where the main output is 480p at 60 Hz.

The auxiliary master pixel clock (generated by the clock generator B) is always 27 MHz.

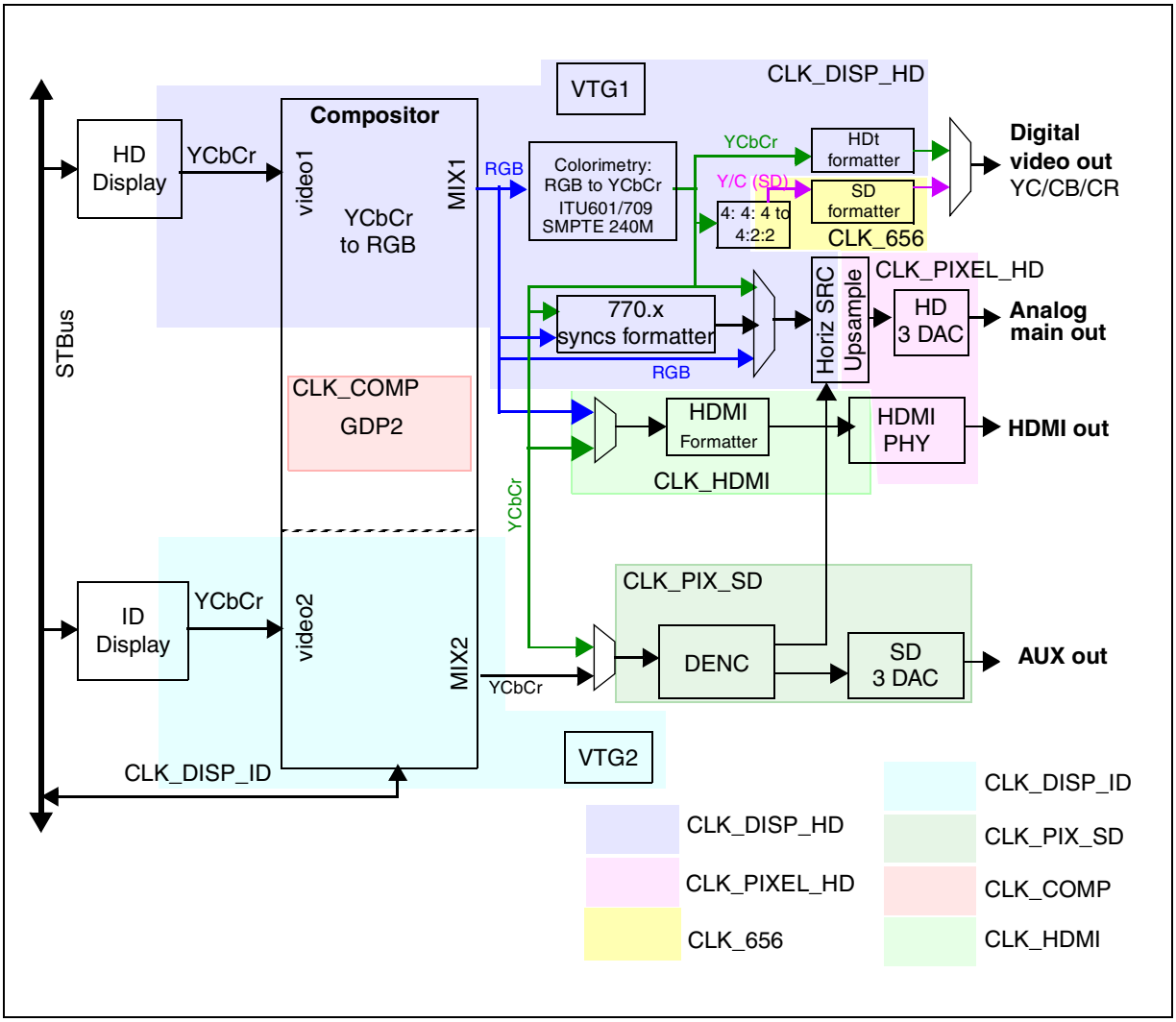
Note: 1080i and 720p are 30/60 Hz display standards; 480i and 480p are 29.97/59.94 Hz display standards. For some applications where it is preferable to keep the input rate for the display, the clock is altered by a factor of 1.001 to adapt the output rate by keeping the scanning. This is possible in 1080i, 720p and 480p formats but not in 480i format where the CVBS or C outputs, (chroma modulation) are very sensitive to clock rate and can not accept such a divergence.

The VOS implements a number of clock domains to make possible different configurations:

- CLK_PIX_HD: main master pixel clock, clocking HD DACs and TMDS block (HDMI PHY),
- CLK_DISP_HD: pixel clock of main datapath (VTG1, mixer 1, HD display, formatters),
- CLK_DISP_ID: pixel clock of auxiliary path (VTG2, mixer 2, ID display),
- CLK_COMP: GDP2 pixel clock, depends on GDP2 allocation,
- CLK_PIX_SD: 2X SD pixel clock (27 MHz) for DENC and SD DACs,
- CLK_656: 2X pixel clock for CCIR 656/SMPTE 293M formatter,
- CLK_HDMI: clock for the HDMI digital formatter.

For more details see Chapter 15: *Clocks on page 120*.

Figure 184: VOS clock domains



53.2.1 Applications

Table 175 summarizes some clock frequency values with respect to the display configurations. Related clocks (obtained by division of the same master clock) are shown with the same background color: magenta for main and blue for auxiliary.

Table 175: Clock domains by applications

Application		Clock frequencies, MHz					
		CLK_PIXEL_HD	CLK_DISP_HD	CLK_PIXEL_SD	CLK_DISP_SD	CLK_656	CLK_COMP
Main: 1080i@30 Hz or 720p@60 Hz	GDP2 on main	148.5	74.25	27	13.5	-	74.25
	GDP2 on aux	148.5	74.25	27	13.5	-	13.5
Main: 480p@59.94 Hz	GDP2 on main	108	27	27	13.5	54	27
	GDP2 on aux	108	27	27	13.5	54	13.5
Main: 1080i@29.97 Hz or 720p@59.94 Hz	GDP2 on main	~148.35	~74.176	27	13.5	-	~74.176
	GDP2 on aux	~148.35	~74.176	27	13.5	-	13.5
Main: 480p@60 Hz	GDP2 on main	108.108	27.027	27	13.5	54.054	27.027
	GDP2 on aux	108.108	27.027	27	13.5	54.054	13.5
Main: 480i@29.97 Hz	GDP2 on main	108	13.5	27	-	27	13.5

These clocks are generated by the clock generator B. For details, see Chapter 15: *Clocks on page 120*.

53.3 Video timing generators (VTG)

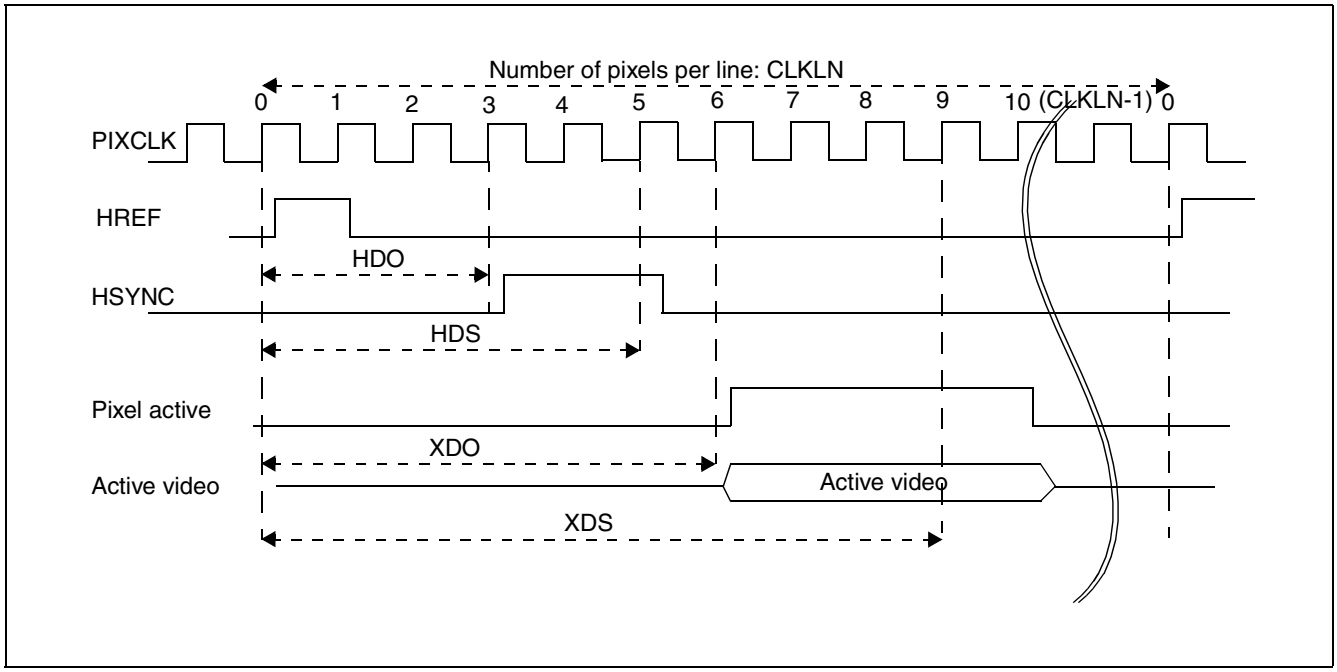
The STx7100 provides two video timing generators: VTG1 and VTG2.

The two VTGs provide horizontal and vertical video synchronization reference signals to the main and auxiliary data paths: in standard applications, VTG1 paces the main path and VTG2 paces the auxiliary path.

Each VTG provides a vertical temporal reference (V_{REF}) and an horizontal temporal reference (H_{REF}), that are used to generate VSYNC and HSYNC signals throughout the chip. In addition, the VTGs generate the VIDDIGOUTHSYNC and VIDDIGOUTVSYNC output signals. The shape and polarity of these signals are programmable through HDO/HDS and VDO/VDS registers

53.3.1 Horizontal synchronization generation

Figure 185: Horizontal synchronization generation



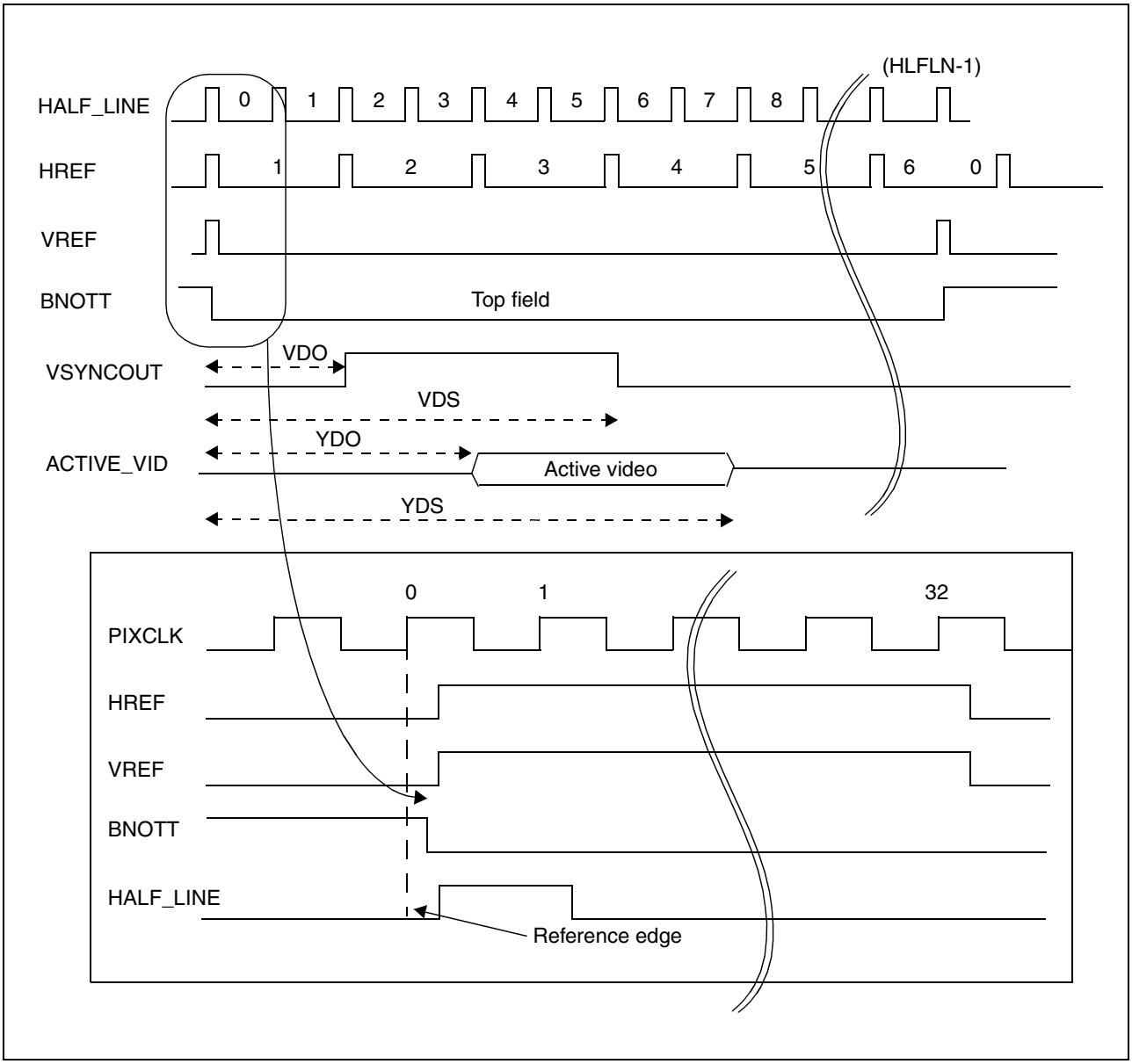
All the pixel numbers in a line are referenced to the rising edge of H_{REF}.

- CLKLN (VTGn_CLKLN register) specifies the line length in PIXCLK cycles.
- HDO (register VTGn_HDO) defines the number of PIXCLK cycles between the rising edge of the internal H_{REF} and the rising edge of H_{SYNC}.
- HDS (register VTGn_HDS) defines the number of PIXCLK cycles between the rising edge of the internal H_{REF} and the falling edge of H_{SYNC}.
- Registers XDO and XDS, defining the active video line (horizontal dimension of the active video window) are located in the mixers of the compositor.

Note: If the HDS value is less than the HDO value, the result is the generation of an active low HSYNC pulse.

53.3.2 Vertical synchronization generation (interlaced output)

Figure 186A: Vertical synchronization generation: (interlaced output)



The number of half lines per field HLFLN (or number of lines per picture) is specified in the VTGn_HLFLN register. The half-line counter is incremented with a half-line resolution. It is reset when the counter matches the value programmed into the VTGn_HLFLN register, generating VREF.

If the value programmed into the VTGn_HLFLN register is even, a progressive scan display output is generated. If the value in HLFLN is odd, an interlaced output is generated. In both modes, a BNOTT (bottom not top) signal is also generated (in progressive output BNOTT is always 0). For an interlaced picture with an even number of lines (SMPTE 295M standard) a bit is programmed in a register to force the VTG to be in interlaced mode. In this case the picture has HLFLN - 1 half lines in one field and HLFLN + 1 in the other field.

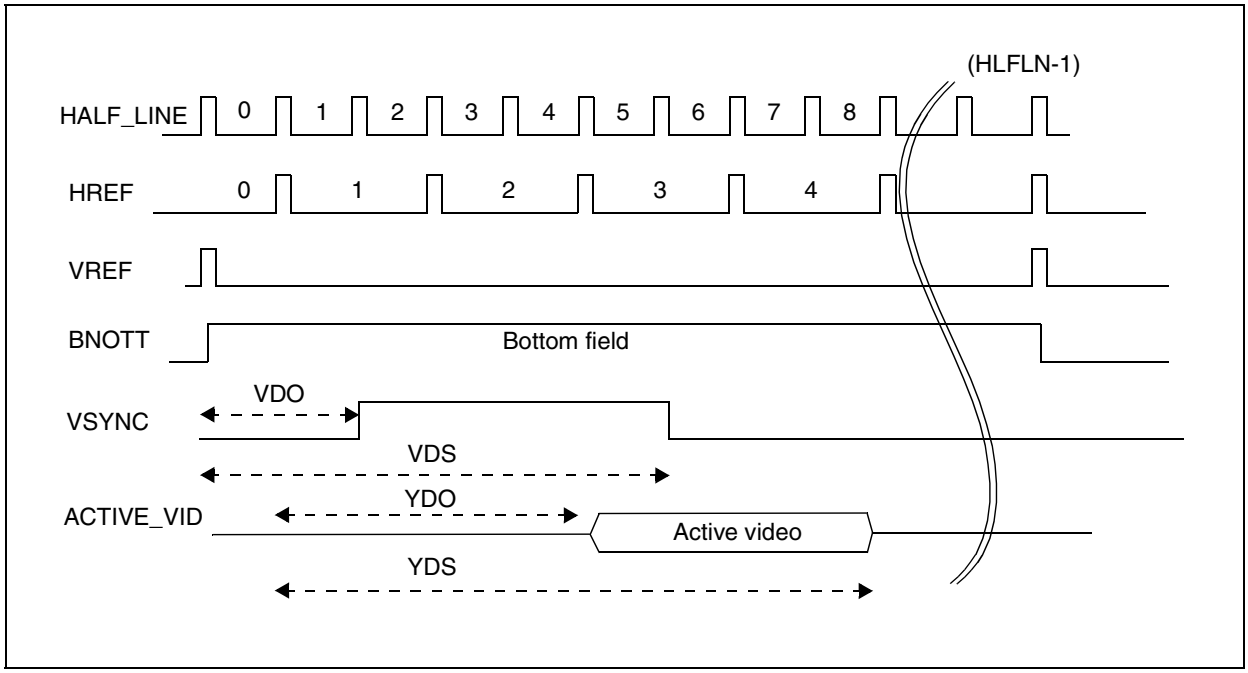
The line counter starts with 1 for VREF top and 0 for VREF bottom. It is then incremented by 1 every H_{REF}.

The registers YDO and YDS defining the vertical active window are located in the mixers of the compositor. YDO defines the first active line and YDS defines the last active line (in line units). The number of lines per field is (YDS - YDO + 1).

The VDO register (VTGn_VDO) determines the starting position of the vertical drive output pulse relative to VREF (half-line increment).

The VDS register (VTGn_VDS) determines the ending position of the vertical drive output pulse relative to VREF (half-line increment).

Figure 187: VTG Output (interlaced picture: bottom field)



Vsync positioning relative to HREF (available from cut 2.0 only)

The VHD field of the register VDS holds a 12-bit value that define the VSync position (in pixel clock cycles) relative to the HREF signal. With a 0 value, VSYNC rises one clock cycle after VREF. To get a VSYNC coincident to HSYNC, VHD must be set to HDO (or HDS) if HDO is in the first half of the line, or VHD must be set to HDO (or HDS) - CLKN/2 if HDO is in the second half of the line.

In the example of Figure 187 above, YDO = 3, YDS = 4, VDO =2 and VDS = 6.

Number of active video lines = (YDS - YDO +1) = 2

Length of the VSYNCOUT pulse = (VDS - VDO) = 4 half lines.

Note: If HDO = 0, then HSYNCOUT *and* VSYNCOUT *active edge are both generated on the same clock cycle on the line corresponding to VDO.*

53.3.3 Interrupts

VST (Vsync top) and VSB (Vsync bottom) can be generated by the VTG. These interrupts are available though the ITS (interrupt status) register if the corresponding ITM (interrupt mask) bit is set. In progressive scan, the VTG still generates VST and VSB even though they are not necessary.

A timer is provided to generate an delayed interrupt PDVS (programmable delay on vsync) *n* cycles after the rising edge of vsync. The number of cycles is a 24-bit value defined in the register VTGn_VTMR (vsync delayed interrupt timer).

53.3.4 VTG modes

The following synchronization modes are supported by the STx7100 VTGs:

- **VTG1** (Main - HD or SD output): VREF1 and HREF1 are always free running and generated from internal counters.
- **VTG2** (Auxiliary - SD output): the following configurations are possible:
  - VREF2 and HREF2 are free running based on internal counters
  - VREF2 and HREF2 slaved to VREF1 and HREF1 from VTG1. This is to slave the auxiliary video to the main video
  - VREF2 slaved to VREF1 from VTG1, HREF2 based on internal counter. This mode enables to slave VTG2 on a frame basis to a source with a different scanning format.

Figure 188 and Figure 189 show HREF and VREF generation for VTGs 1 and 2 respectively.

Figure 188: VTG1 HREF and VREF generation

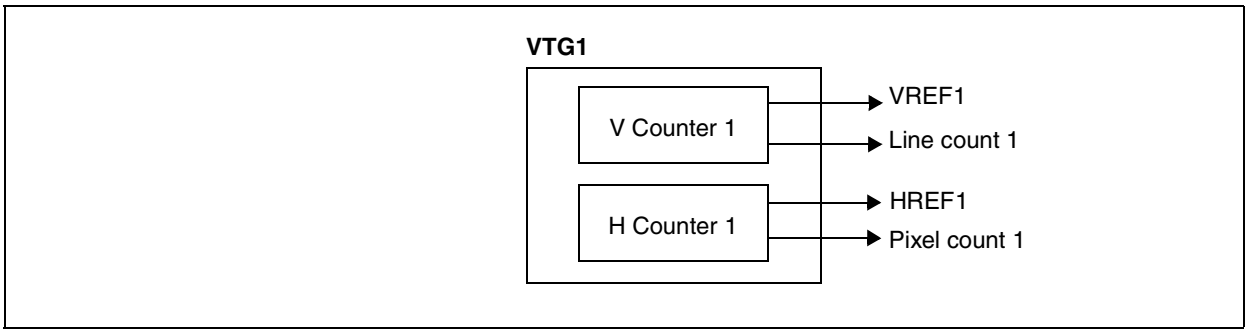
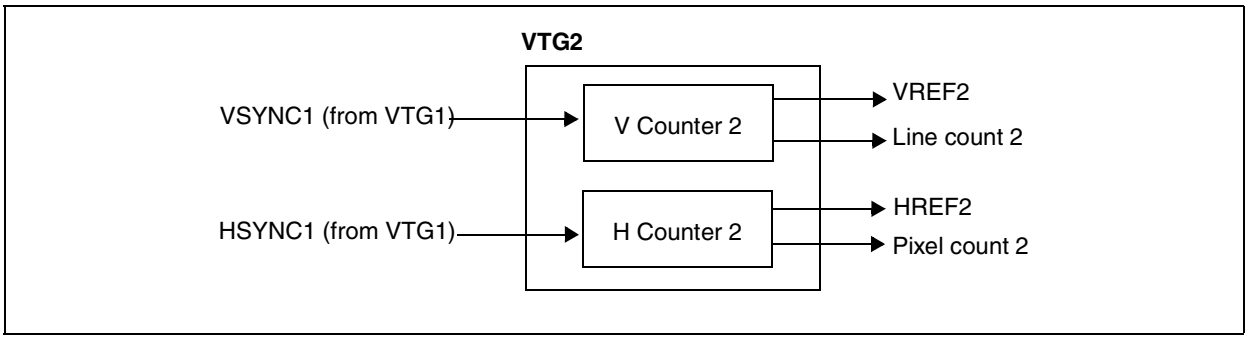


Figure 189: VTG2 HREF and VREF generation



53.3.5 Syncs generation for DVO, analog, HDMI and waveform generator

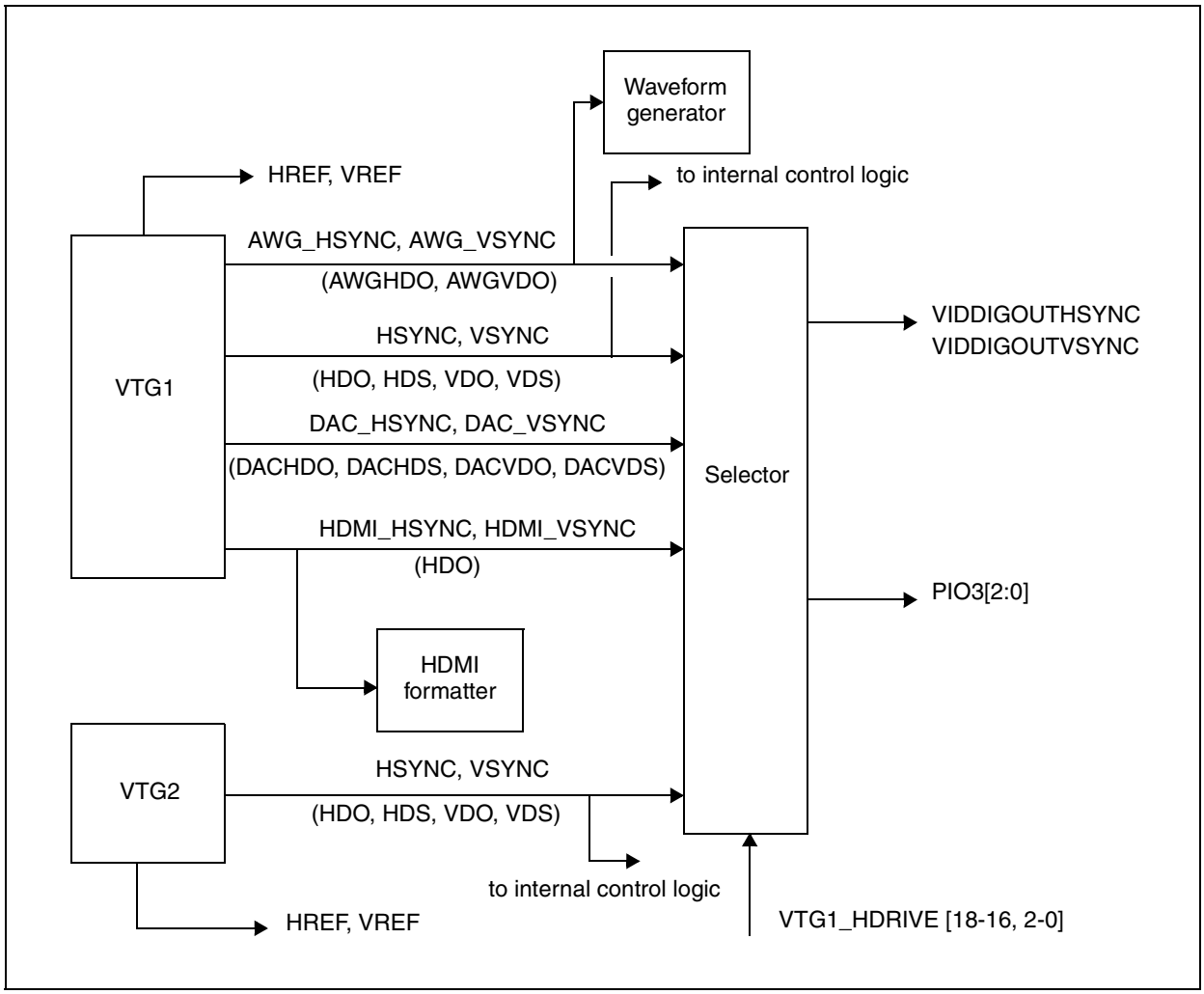
The STx7100 VTGs are able to generate several pairs of Hsync and Vsync signals with different parameters to be used by different units.

These Hsync and Vsync signals are:

- DVO Hsync and Vsync defined by VTG*n*_VDO, VTG*n*_VDS, VTG*n*_HDO and VTG*n*_HDS registers.
- Analog Hsync and Vsync that can be used with RGB analog outputs and defined by VTG1_DACVDO, VTG1_DACVDS, VTG1_DACHDO and VTG1_DACHDS registers.
- Waveform generator Hsync and Vsync defined by VTG1_AWGVDO and VTG1_AWGHDO registers.
- HDMI Hsync and Vsync which can be delayed relative to Hsync and Vsync. The delay is specified in the register VTG1_HDO.

Any of these signals can be routed to the output pins VIDDIGOUTHSYNC, VIDDIGOUTVSYNC, or PIO3(2:0). The selection is done by configuring the register VTG1_HDRIVE.

Figure 190: Hsync and Vsync generation



53.3.6 Synchronization with external video

When receiving an external digital video via the DVP (D1/ITU-R656 4:2:2), the incoming data is sampled with the incoming video clock supplied externally and not controlled by the IC. To synchronize the DVP input to the VTG, the frame buffers fullness must be monitored and frame skip or repeat operation must be performed.

53.4 RGB to YCrCb color space conversion

RGB values received by the video output stage are gamma-corrected values (sometimes called R'G'B'). It is assumed that all the video signals connected to the chip are already gamma-corrected. Therefore, the following equations do not perform any gamma correction. If a gamma correction is needed (for non-precorrected graphics downloaded from the web for example), the gamma correction must be done with the 2D graphic processor (see Chapter 47: *2D graphics processor (blitter)* *on page 428*).

Three equations are implemented:

- RGB to ITU-R BT 601 YCbCr,
- RGB to ITU-R BT 709 YCbCr,
- RGB to SMPTE 240M YCbCr.

The selection of the conversion matrix is possible with through configuration bits.

53.5 Digital video output formatter

The digital output formatter inserts proper synchronization words (EAV/SAV) into the video data flow according to selected standards and can produce:

- YCbCr, 4:2:2 on 16 bits with embedded sync: digital SMPTE 274M, SMPTE 295M, SMPTE 293M,
- YCbCr, 4:2:2 on 8 bits (8 MSBs of the DVO output) with or without embedded EAV/SAV sync: ITU-R 656.

53.6 Analog video output

The analog video output can be either:

- full-scale RGB output
- full-scale YPbPr output
- EIA-770.2 compatible YPbPr with embedded bi-level sync
- EIA 770.3 compatible YPbPr with embedded tri-level sync.

The pure video signal is reduced by 70% to conform to the EIA 770.x standards.

The YPbPr analog output handles the following raster-scanning systems:

Table 176: YPbPr analog output raster-scanning systems

Active picture	Frame	maximum PIXCLK frequency, MHz
1920 x 1080i	2200 x 1125i	74.25
1280 x 720p	1650 x 750p	
1920 x 1080i	2200 x 1250i	
720 x 483p	858 x 525p	27

The VOS configuration registers can be programmed to generate video complying with the following display standards: SMPTE 274M (1125i), SMPTE 293M (525p), SMPTE 295M (1250i), SMPTE 296M (750p) and ARIB-BS4 (525p, 750p, 1125i).

53.6.1 Analog syncs generation (bi-level and tri-level)

To generate analog waveforms compliant with standards, a 770.3 and 770.2 formatter rescales the video data and inserts synchronization pulses before digital to analog conversion.

The analog syncs waveform (bi-level and tri-level) can be defined either using a flexible syncs generator or using pre-defined waveforms. This option is defined by the bit 3 of the register DSPCFG_ANA. The synchronization pulses waveforms are defined with the configuration registers AWG_CTRL_0 and AWG_RAM[0 .. 45].

In 480p, the formatter is able to apply Macrovision protection to the video flow if required.

The programmable configuration parameters allow flexibility for slight deviations from these standards.

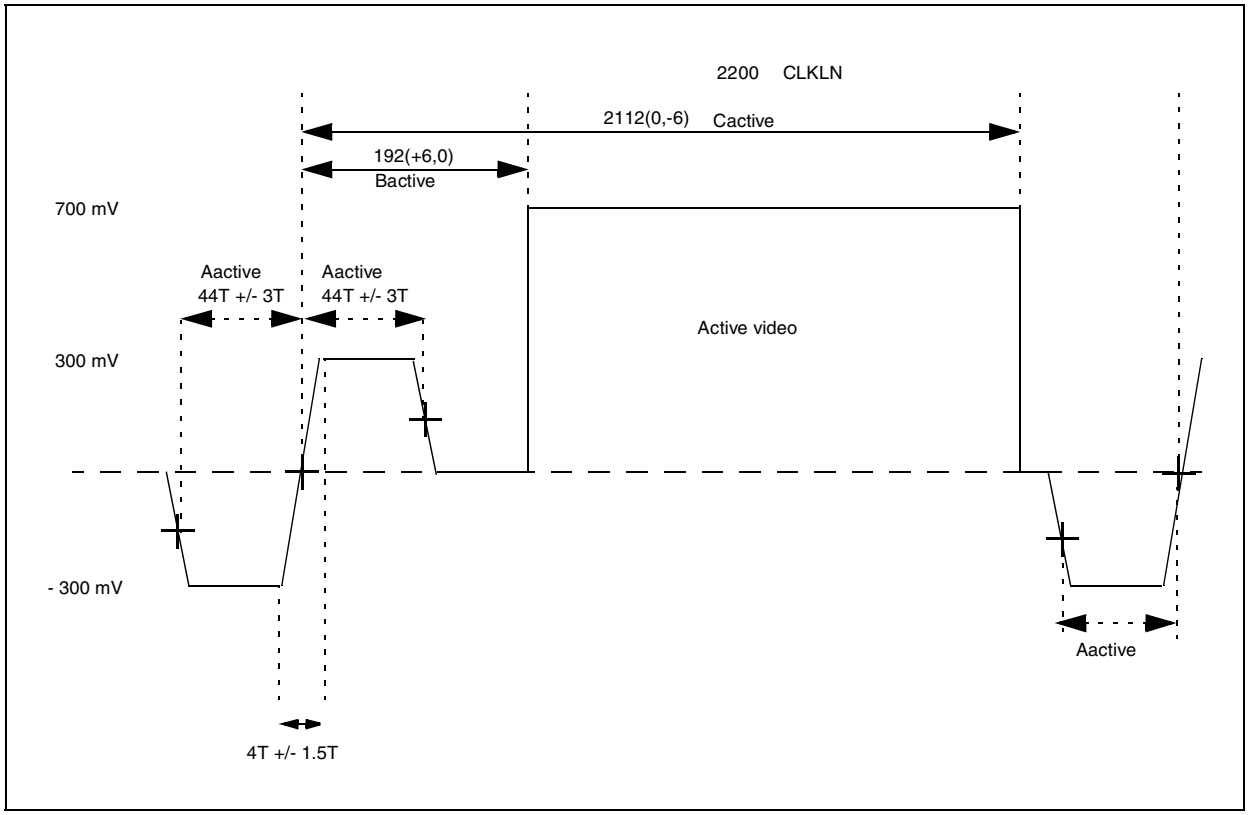
53.6.2 YPbPr waveforms - HD formats

53.6.2.1 Horizontal timing

Analog waveform (1080i)

Figure 191 describes the Y' analog waveform.

Figure 191: Y' analog waveform



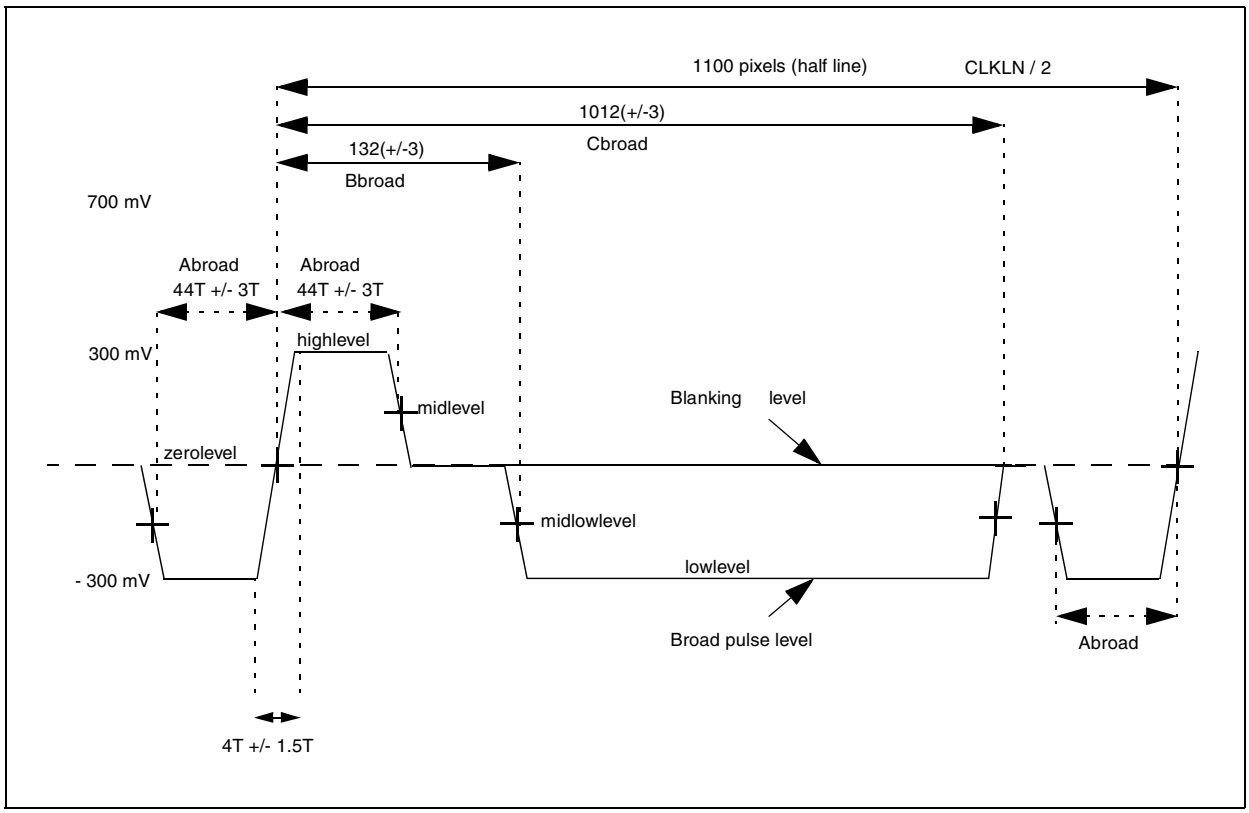
The Pb' and Pr' waveforms are similar, except for the voltage excursion (+/-350 mV).

Several raster scanning systems can be accommodated by programming *Aactive*, *Bactive*, *Cactive* and *CLKLN*.

Extra-sync pulse

With interlaced pictures, the vertical sync line may include a mid-line tri-level sync pulse. Certain vertical sync lines may therefore contain a broad pulse during the first half line and a broad pulse during the second half line. This is shown in Figure 192.

Figure 192: Vertical timing relating to analog waveform



Values *Abroad*, *Bbroad* and *Cbroad* are programmable to handle several raster scanning systems.

Aactive and *Abroad* have the same value which is programmed in register DHDO_ABROAD. The blanking waveform (waveform without the broad pulse level) only requires the values Abroad and CLKLN to be programmed.

The synchronization level is also programmable (five registers for luma and five for chroma: ZEROLEVEL, MIDLEVEL, HIGHLEVEL, MIDLOWLEVEL and LOWLEVEL).

53.6.2.2 Vertical timing

As indicated in the previous section, a field or frame is composed of three line types: lines with a broad pulse, lines with blanking and lines with active video. The number of lines of each type is programmable (broad length, blanking length and active length). Figure 193 and Figure 193 show vertical timings for different video standards.

Figure 193: SMPTE 274M AND EIA 770.3 vertical timings

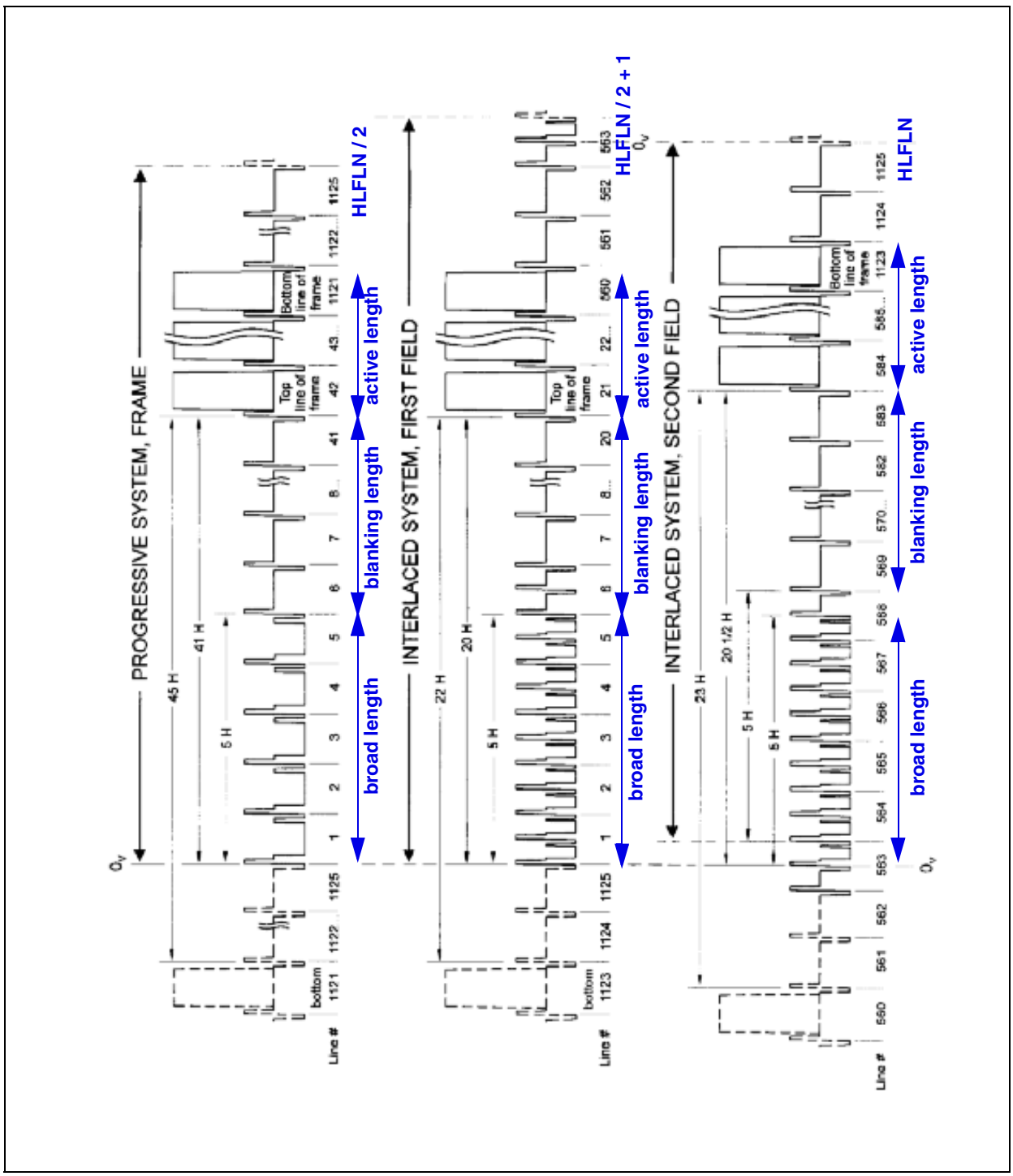
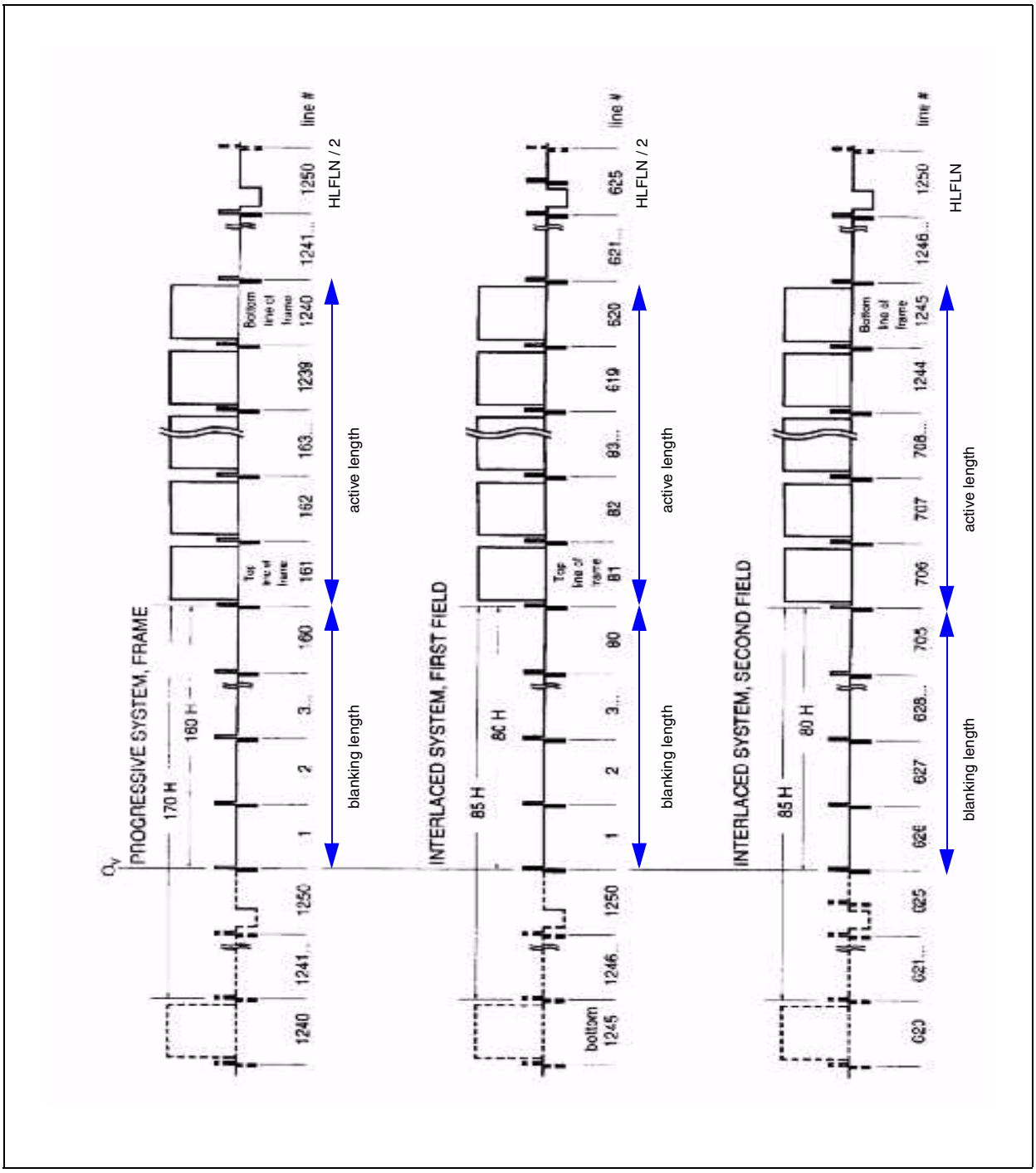


Figure 194: SMPTE 295M vertical timings



SMPTE 295M is a special case because the number of lines is even for both the interlaced and progressive systems. Moreover, there are only two special lines in interlaced and one in progressive. All other lines are either blank lines or active lines. Therefore *broadlength* must be 0.

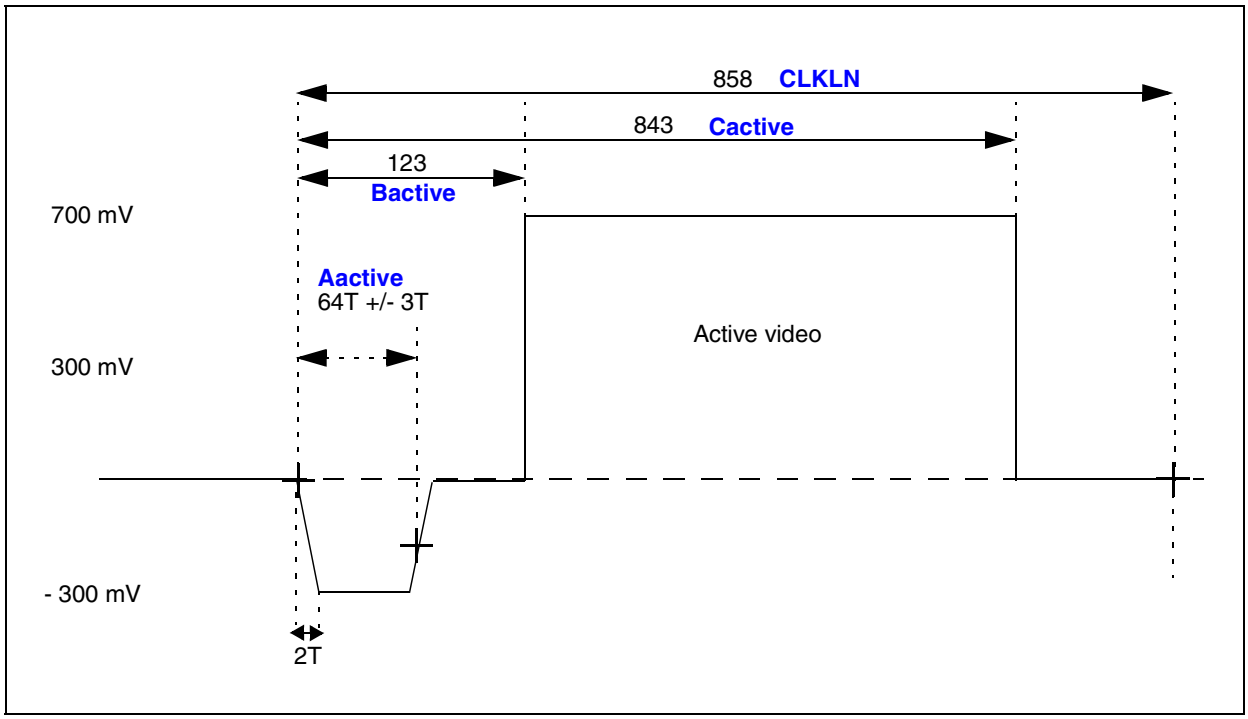
53.6.3 YPbPr waveform - SD/ED formats

53.6.3.1 Horizontal timing

SMPTE 293M is a bi-level sync pulse standard. For the control of this raster-scan system, no dedicated register is used. The registers used for EIA 770.3, SMPTE 274M and SMPTE 295M are used but not exactly in the same way - attention must be paid to their use.

Figure 195 shows the Y' analog waveform in a SMPTE 293M video standard.

Figure 195: Y' analog waveform



The P'b, P'r analog waveforms are similar, except for the voltage excursion (+/- 350 mV).

The registers *Aactive*, *Bactive*, *Cactive* and *CLKLN* must be setup accordingly.

Figure 195 and Figure 195 show SMPTE 293M vertical timing signals.

Figure 196: Vertical timing relating to analog waveform for broad lines

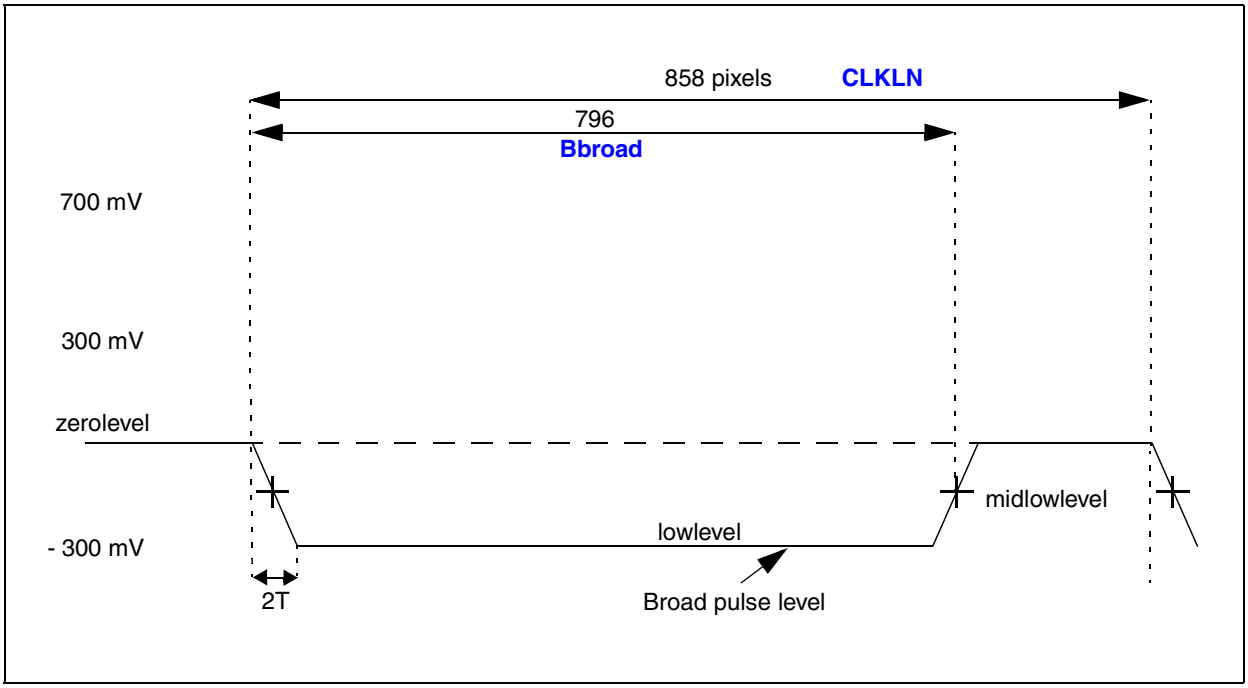
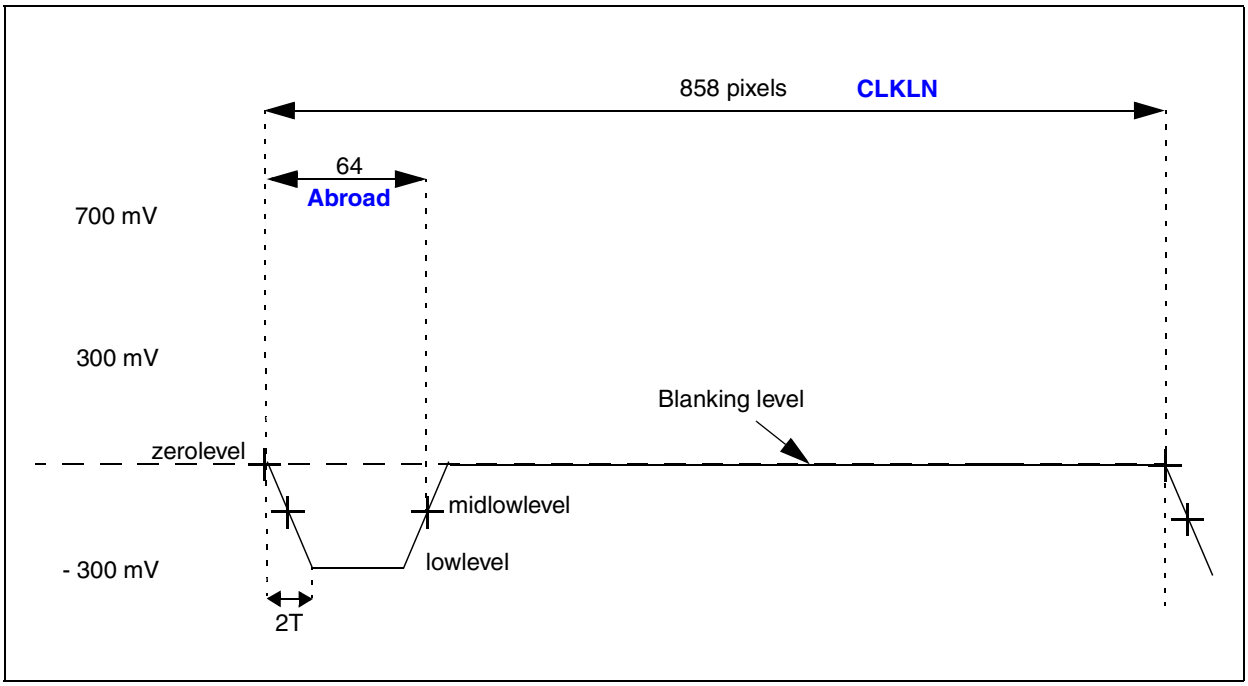


Figure 197: Vertical timing relating to analog waveform for blanking lines



Abroad, *Bbroad* and *Cbroad* values are programmable to handle several raster scanning systems.

Aactive and *Abroad* have the same value defined in the register DHDO_ABROAD. The blanking waveform only requires the programming of *Abroad* and *CLKLN*. The broad waveform only requires the programming of the values *Bbroad* and *CLKN*. The value *Cbroad* is not used with the SMPTE 293M standard.

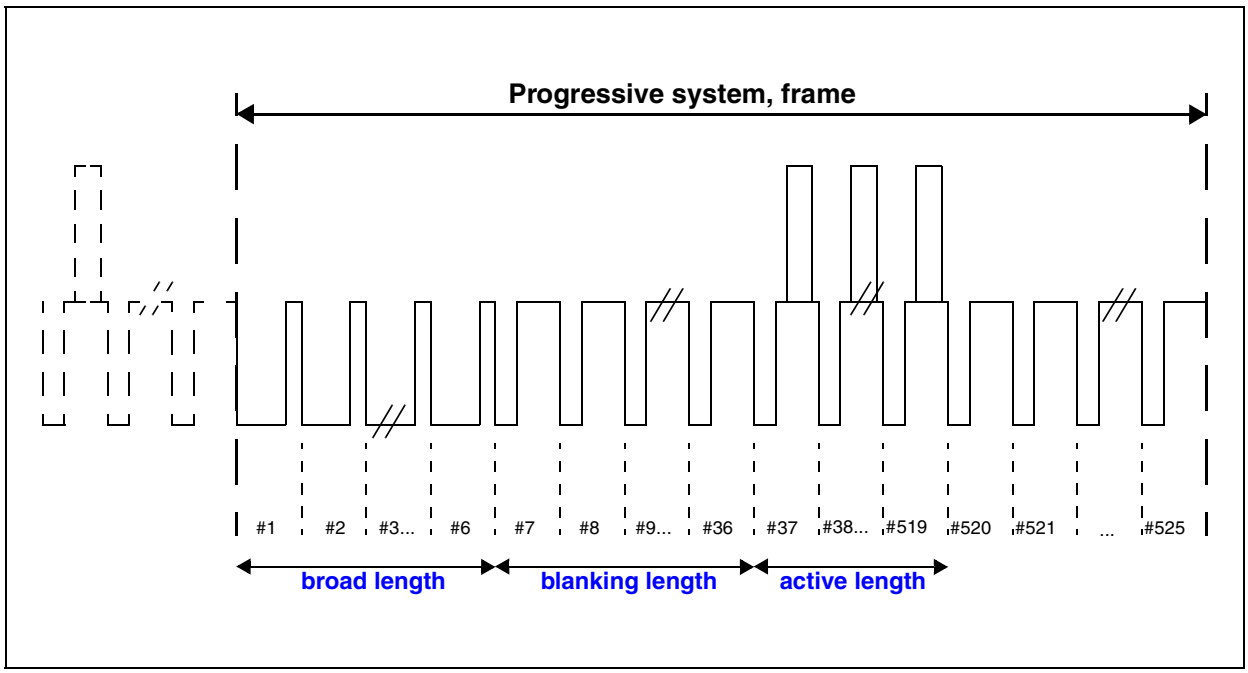
The synchronization level is also programmable (three registers for luma and three for chroma: ZEROLEVEL, MIDLOWLEVEL and LOWLEVEL).

53.6.3.2 Vertical timing

The SMPTE 293M standard describes a progressive system.

As indicated in the previous sections, a frame is composed of three line types: lines with a broad pulse, lines with blanking and lines with active video. The number of lines of each type is programmable (broad length, blanking length and active length) Figure 198 shows frame waveforms for a 525p picture.

Figure 198: 525p frame waveform



53.7 Upsampler

The purpose of the upsampler is to adapt the display pixel rate to the HD DAC clock rate. In a standard configuration, the pixel rate is defined by the clock used in the main display data path (CLK_DISP_HD) and requires an upsampling by 2 or 4.

In an alternate application where the main and auxiliary outputs are SD, the upsampler gets its data from the DENC at 27 MHz and up samples by a factor of 4. It must be noticed that the DENC itself already performs an upsampling by 2 to the original pixel rate at 13.5 MHz.

The SRC (sample rate conversion) filter used to perform the interpolation is a polyphase tapped delay line filter with 12 taps and programmable coefficients.

The upsampler provides 2 modes:

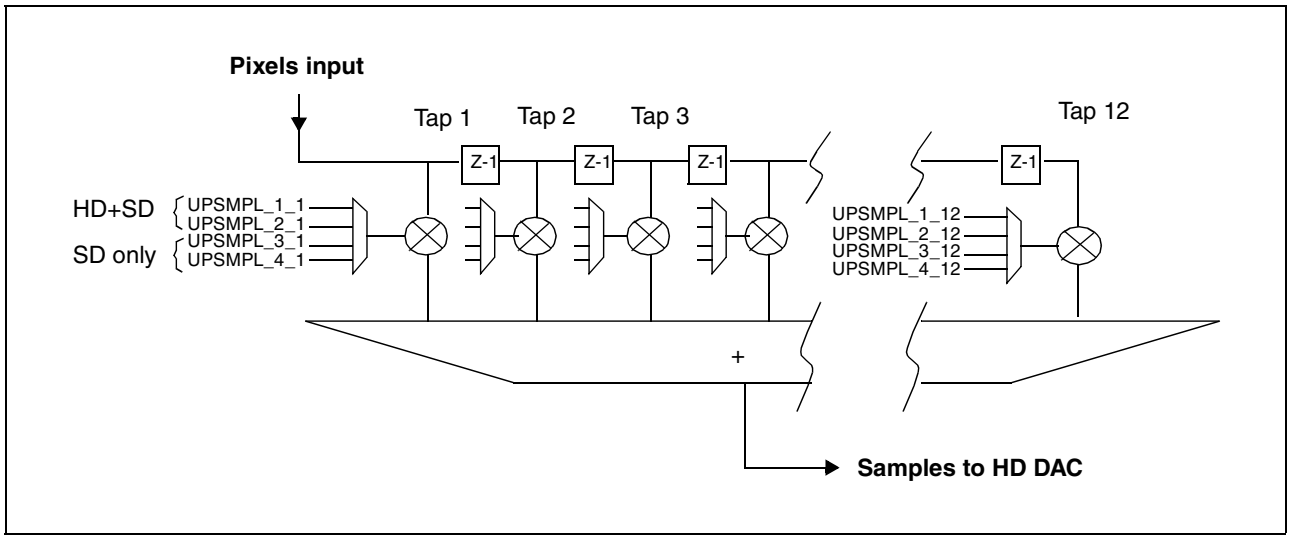
- **High quality mode**: the upsampling maintains the bandwidth of the incoming video stream. It includes an adequate sin*x*/*x* compensation filter to cope with the DAC external anti-aliasing filter.
- **Low pass mode**: the upsampling implements a 1/4 low-pass filter. This can be used for 'constrained output' mode support whereby, while the HDMI produces a full resolution encrypted digital video, the analog output only provides an output video with a degraded resolution.

STMicroelectronics can supply suggested coefficient settings for the above applications.

In addition to these modes, the user can modify coefficients to change the filter's response if required. Coefficient programming is done through the configuration registers UPSMPL_CSET*x_y*.

Figure 199 shows the SRD filter structure.

Figure 199: x2 (HD) or x4 (SD) SRC filter structure



54 Video output stage (VOS) registers

VOS register addresses are provided as *VOSBaseAddress* + offset.

The *VOSBaseAddress* is:

0x1900 5000.

AWG register addresses are provided as *AWGBaseAddress* + offset.

The *AWGBaseAddress* is:

0x1900 4800.

Table 177: VOS register summary

Register	Description	Offset	Type
Video timing generator 1			
VTG1_HDRIVE	VTG1 - HSYNC, VSYNC output enable	0x0000	R/W
VTG1_CLKLN	VTG1 - Clocks per line	0x0004	R/W
VTG1_HDO	VTG1 - HSYNC rising edge	0x0008	R/W
VTG1_HDS	VTG1 - HSYNC falling edge	0x000C	R/W
VTG1_HLFLN	VTG1 - Half line per vertical	0x0010	R/W
VTG1_VDO	VTG1 - VSYNC rising edge	0x0014	R/W
VTG1_VDS	VTG1 - VSYNC falling edge	0x0018	R/W
VTG1_AWGHDO	VTG1 - AWG HSYNC rising edge	0x0130	R/W
VTG1_AWGVDO	VTG1 - AWG VSYNC falling edge	0x0134	R/W
VTG1_DACHDO	VTG1 - DAC HSYNC rising edge	0x0138	R/W
VTG1_DACHDS	VTG1 - DAC HSYNC falling edge	0x013C	R/W
VTG1_DACVDO	VTG1 - DAC VSYNC rising edge	0x0140	AC
VTG1_DACVDS	VTG1 - DAC VSYNC falling edge	0x0144	R/W
VTG1_MODE	VTG1 - Mode of operation	0x001C	R/W
VTG1_VTMR	VTG1 - Vsync delayed interrupt timer	0x0020	R/W
VTG1_DRST	VTG1 - VTG reset	0x0024	WO
VTG1_ITM	VTG1 - IT mask	0x0028	R/W
VTG1_ITS	VTG1 - IT Status	0x002C	RO
VTG1_STA	VTG1 - Status	0x0030	RO
Video timing generator 2			
VTG2_CLKLN	VTG2 - Clocks per line	0x0038	R/W
VTG2_HDO	VTG2 - VSYNC output rising edge	0x003C	R/W
VTG2_HDS	VTG1 - HSYNC output falling edge	0x0040	R/W
VTG2_HLFLN	VTG2 - Half line per vertical	0x0044	R/W
VTG2_VDO	VTG2 - VSYNC output rising edge	0x0048	R/W
VTG2_VDS	VTG2 - VSYNC output falling edge	0x004C	R/W

Table 177: VOS register summary

Register	Description	Offset	Type
VTG2_MODE	VTG2 - Mode of operation	0x0050	R/W
VTG2_VTMR	VTG2 - Vsync delayed interrupt timer	0x0054	R/W
VTG2_DRST	VTG2 - VTG reset	0x0058	WO
VTG2_R1	VTG2 - Range 1	0x005C	R/W
VTG2_R2	VTG2 - Range 2	0x0060	R/W
VTG2_ITM	VTG2 - IT mask	0x0064	R/W
VTG2_ITS	VTG2 - IT Status	0x0068	RO
VTG2_STA	VTG2 - Status	0x006C	R/W
General VOS configuration			
DSPCFG_CLK	DSPCFG - Display and global configuration	0x0070	R/W
DSPCFG_DIG	DSPCFG - Display digital output config	0x0074	R/W
DSPCFG_ANA	DSPCFG - Display analog output config	0x0078	R/W
Main video output configuration			
DHDO_ACFG	DHDO - Analog output config	0x007C	R/W
DHDO_ABROAD	DHDO - Abroad value	0x0080	R/W
DHDO_BBROAD	DHDO - Bbroad value	0x0084	R/W
DHDO_CBROAD	DHDO - Cbroad value	0x0088	R/W
DHDO_BACTIVE	DHDO - Bactive value	0x008C	R/W
DHDO_CACTIVE	DHDO - Cactive value	0x0090	R/W
DHDO_BROADL	DHDO - Broadpulse lines number	0x0094	R/W
DHDO_BLANKL	DHDO - Blanking lines number	0x0098	R/W
DHDO_ACTIVEL	DHDO - Active lines number	0x009C	R/W
DHDO_ZEROL	DHDO - Zero level values	0x00A0	R/W
DHDO_MIDL	DHDO - Middle level values	0x00A4	R/W
DHDO_HI	DHDO - High level values	0x00A8	R/W
DHDO_MIDLO	DHDO - Mid low level values	0x00AC	R/W
DHDO_LO	DHDO - Low level values	0x00B0	R/W
DHDO_COLOR	DHDO - Main display output color space selection	0x00B4	R/W
DHDO_YMLT	DHDO - Luma multiplication factor	0x00B8	R/W
DHDO_CMLT	DHDO - Chroma multiplication factor	0x00BC	R/W
DHDO_COFF	DHDO - Chroma offset	0x00C0	R/W
DHDO_YOFF	DHDO - Luma offset	0x00C4	R/W
Digital SD video out			
C656_ACTL	C656 - Active lines	0x00C8	R/W
C656_BACT	C656 - Beginning of active video	0x00CC	R/W

Table 177: VOS register summary

Register	Description	Offset	Type
C656_BLANKL	C656 - Blanking lines	0x00D0	R/W
C656_EACT	C656 - End of active video	0x00D4	R/W
C656_PAL	C656- PAL configuration	0x00D8	R/W
HD DAC configuration			
DSPCFG_DAC	DSPCFG - HD DACs config	0x00DC	R/W
Reserved		0x00E0	-
Upsampler tap coefficients			
UPSMPL_CSET1_1	Upsampler coefficient set 1.1	0x00E4	R/W
UPSMPL_CSET1_2	Upsampler coefficient set 1.2	0x00E8	R/W
UPSMPL_CSET1_3	Upsampler coefficient set 1.3	0x00EC	R/W
UPSMPL_CSET1_4	Upsampler coefficient set 1.4	0x00F0	R/W
UPSMPL_CSET2_1	Upsampler coefficient set 2.1	0x00F4	R/W
UPSMPL_CSET2_2	Upsampler coefficient set 2.2	0x00F8	R/W
UPSMPL_CSET2_3	Upsampler coefficient set 2.3	0x00FC	R/W
UPSMPL_CSET2_4	Upsampler coefficient set 2.4	0x0100	R/W
UPSMPL_CSET3_1	Upsampler coefficient set 3.1	0x0104	R/W
UPSMPL_CSET3_2	Upsampler coefficient set 3.2	0x0108	R/W
UPSMPL_CSET3_3	Upsampler coefficient set 3.3	0x010C	R/W
UPSMPL_CSET3_4	Upsampler coefficient set 3.4	0x0110	R/W
UPSMPL_CSET4_1	Upsampler coefficient set 4.1	0x0114	R/W
UPSMPL_CSET4_2	Upsampler coefficient set 4.2	0x0118	R/W
UPSMPL_CSET4_3	Upsampler coefficient set 4.3	0x011C	R/W
UPSMPL_CSET4_4	Upsampler coefficient set 4.4	0x0120	R/W
HDMI DLL & PLL lock status			
HDMI_PHY_LCK_STA	HDMI phy lock status	0x0124	RO

Table 178: AWG register summary

Register	Description	Offset	Type
Waveform generator			
AWG_CTRL0	AWG - global control	0x0000	R/W
AWG_RAMn (n=0 to 45)	AWG - instruction RAM	0x0100 + *n* x 0x04	R/W

54.1 VTG 1

VTG1_HDRIVE HSYNC and VSYNC selection

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													DVOSYNC			Reserved							AVSYNC	Reserved						ALTSYNC		

Address: *VOSBaseAddress* + 0x0000

Type: R/W

Reset: 0

Description: This register controls the selection of the VSYNC, HSYNC and BNOTTOP signals that are delivered to the VIDDIGOUTHSYNC, VIDDIGOUTVSYNC and PIO3(2-0) pads.

[31:19] **Reserved**

[18:16] **DVOSYNC**: selects which sync signals is delivered on the VIDDIGOUTHSYNC and VIDDIGOUTVSYNC DVO pads
000: 0 on all pads
001: Main hsync and vsync
010: DAC hsync and vsync (analog)
011: Main HREF and VREF
100: Aux HSYNC and VSYNC
101: Aux HREF and VREF

[15:9] **Reserved**

[8] **AVSYNC**: selects which VREF is used by the PCM reader
0: Main VREF 1: Aux VREF

[7:3] **Reserved**

[2-0] **ALTSYNC**: selects which sync signals are delivered on the PIO alternate function outputs PIO3(2-0): VTG_HREF, VTG_VREF, VTG_BNOTTOP
000: 0 on all pads 011: Main HREF, VREF and BNOTTOP
001: Main HSYNC, VSYNC and BNOTTOP 100: Aux HSYNC, VSYNC and BNOTTOP
010: DAC HSYNC, VSYNC and BNOTTOP (analog) 101: Aux HREF, VREF and BNOTTOP

VTG1_CLKLN Clocks per line

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				CLKLN											

Address: *VOSBaseAddress* + 0x0004

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Defines the total number of pixel per line.

[31:12] **Reserved**

[11:0] **CLKLN:** Total number of pixel clocks per horizontal period (horizontal complete line length).

VTG1_HDO HSYNC rising edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					HDMIHDO			Reserved					DVOHDO			Reserved				HDO											

Address: *VOSBaseAddress* + 0x0008

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Defines the position of the internal HSYNC rising edge relative to HREF, the additional offset added to HDMI Hsync and VIDDIGOUTHSYNC relative to the internal HSYNC signal.

[31:27] **Reserved**

[26:24] **HDMIHDO:** Additional offset (in pixel clock cycle) added to the internal Hsync to generate the HDMI Hsync signal.

[23:19] **Reserved**

[18:16] **DVOHDO:** Additional offset (in pixel clock cycle) added to the internal Hsync to generate the VIDDIGOUTHSYNC signal.

[15:12] **Reserved**

[11:0] **HDO:** Determines the rising edge (in pixel clock cycle) of the internal HSYNC signal pulse relative to HREF. The HSYNC output rises from 0 to 1 level when the pixel counter is equal to VTG1_HDO.

Note: The order of VTG1_HDO and VTG1_HDS determine the polarity of the Hsync signal pin.

VTG1_HDS HSYNC falling edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				HDS											

Address: *VOSBaseAddress* + 0x000C

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

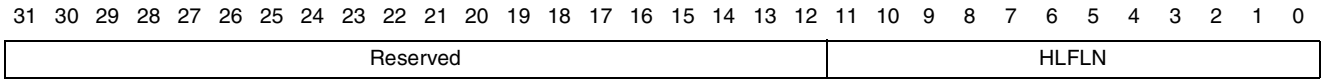
Description: Defines the position of the internal HSYNC falling edge relative to HREF.

[31:12] **Reserved**

[11:0] **HDS:** Determines the falling edge (in pixel clock cycle) of the Hsync signal pulse relative to HREF. The Hsync output falls from 1 to 0 when the pixel counter is equal to VTG1_HDS.

Note: The order of VTG1_HDO and VTG1_HDS determines the polarity of the Hsync signal pin.

VTG1_HLFLN Half line per vertical

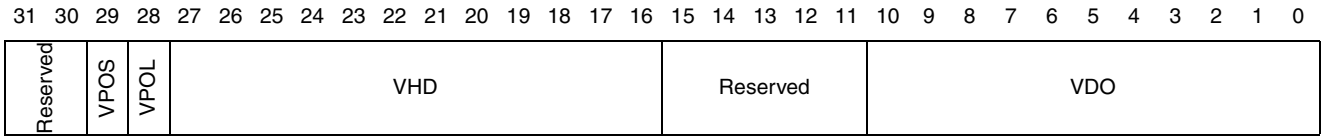


Address: *VOSBaseAddress* + 0x001D
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the number of lines per frame.

[31:12] **Reserved**

[11:0] **HLFLN:** Specifies the number of half line per vertical period in the raster display. An odd value indicates interlaced output. An even value indicates a progressive output, except if overridden by VTG1_VTGMOD.FI (for the 1250 line interlaced standard SMPTE 295M).

VTG1_VDO VSYNC rising edge



Address: *VOSBaseAddress* + 0x0014
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the VSYNC rising edge relative to VREF, the offset of VSYNC relative to HREF, the polarity of VSYNC and the position (late or early) of VSYNC relative to VREF.

[31:30] **Reserved**

[29] **VPOS:** VSYNC position (early or late) relative to VREF
0: VSYNC active edge is after VREF rising edge
1: VSYNC active edge is before VREF rising edge

[28] **VPOL:** VSYNC polarity relative to VREF - used to generate correctly the BNOTTOP signal.
0: The VSYNC synchronization is the rising edge
1: The VSYNC synchronization is the falling edge

[27:16] **VHD:** VSYNC to HREF offset - This register specifies the offset (in pixel clock cycles) of the VSYNC signal relative to HREF. A value 0 means that VSYNC is rising one clock cycle after VREF.

[15:12] **Reserved**

[11:0] **VDO:** VSYNC rising edge position relative to VREF.
Specifies the offset (in number of half-line) of the VSYNC rising edge relative to VREF. The VSYNC output rises from 0 to 1 level when the half line counter is equal to VTG1_VDO.

Note: 1. The order of VTG1_VDO and VTG1_VDS determines the polarity of the Vsync signal.

VTG1_VDS VSYNC falling edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					VDS										

Address: *VOSBaseAddress* + 0x0018
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the internal Vsync falling edge relative to VREF.

[31:11] **Reserved**

[10:0] **VDS:** Vsync falling edge position relative to VREF.
Specifies the offset (in number of half-line) of the Vsync falling edge relative to VREF. The VSYNC output rises from 0 to 1 level when the half line counter is equal to VTG1_VDO.

Note: The order of VTG1_VDO and VTG1_VDS determine the polarity of the VSYNC *signal pin.*

VTG1_AWGHDO Waveform generator hsync rising edge

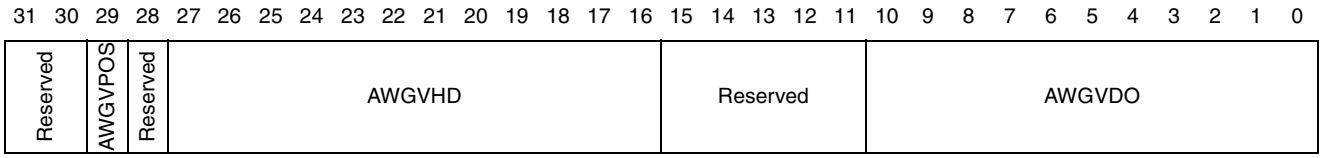
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				AWGHDO											

Address: *VOSBaseAddress* + 0x0130
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the internal AWG HSYNC rising edge relative to HREF. This sync signal is used as a reference signal by the waveform generator.

[31:12] **Reserved**

[11:0] **AWGHDO:** AWG HSYNC rising edge position (in pixel clock cycle) relative to HREF. The AWG HSYNC signal rises from 0 to 1 when the pixel counter is equal to VTG1_AWGHDO.

VTG1_AWGVDO VSYNC rising edge



Address: *VOSBaseAddress* + 0x0134

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Defines the position of the internal AWG Vsync rising edge relative to VREF and the position (late or early) of AWG Vsync relative to VREF.

[31:30] **Reserved**

[29] **AWGVPOS:** AWG Vsync position (early or late) relative to VREF
0: AWG VSYNC active edge is after VREF rising edge
1: AWG VSYNC active edge is before VREF rising edge

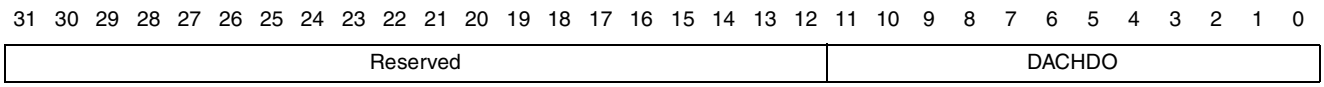
[28] **Reserved**

[27:16] **AWGVHD:** AWG VSYNC to HREF offset - This register specifies the offset (in pixel clock cycle) of the AWG VSYNC signal relative to HREF. A value 0 means that AWG VSYNC is rising one clock cycle after VREF.

[15:12] **Reserved**

[11:0] **AWGVDO:** AWG VSYNC rising edge position relative to VREF
Specifies the offset (in number of half-line) of the AWG Vsync rising edge relative to VREF. The AWG VSYNC signal rises from 0 to 1 when the half-line counter is equal to VTG1_AWGVDO.

VTG1_DACHDO Analog HSYNC rising edge



Address: *VOSBaseAddress* + 0x0138

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Defines the position of the analog Hsync rising edge relative to HREF.

[31:12] **Reserved**

[11:0] **DACHDO:** analog Hsync rising edge position (in pixel clock cycle) relative to HREF. The analog HSYNC signal rises from 0 to 1 when the pixel counter is equal to VTG1_DACHDO.

VTG1_DACHDS Analog HSYNC falling edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				DACHDS											

Address: *VOSBaseAddress* + 0x013C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the analog Hsync falling edge relative to HREF.

[31:12] **Reserved**

[11:0] **DACHDS:** Determines the falling edge (in pixel clock cycles) of the analog HSYNC signal pulse relative to HREF. The HSYNC output falls from 1to 0 when the pixel counter is equal to VTG1_DACHDS.

Note: The order of VTG1_DACHDO and VTG1_DACHDS determine the polarity of the analog Hsync signal.

VTG1_DACVDO Analog VSYNC rising edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DACVPOS	DACVPOL	DACVHD												Reserved				DACVDO											

Address: *VOSBaseAddress* + 0x0140
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the analog Vsync rising edge relative to VREF, the polarity and the position (late or early) of the analog VSYNC relative to VREF.

[31:30] **Reserved**

[29] **DACVPOS**: Analog VSYNC position (early or late) relative to VREF
0: Analog VSYNC active edge is after VREF rising edge
1: Analog VSYNC active edge is before VREF rising edge

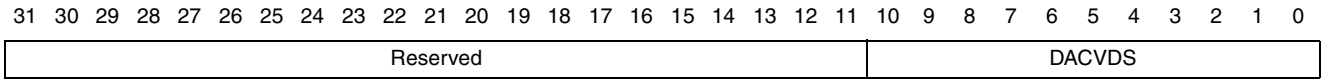
[28] **DACVPOL**: Analog Vsync polarity relative to VREF - used to generate correctly the BNOTTOP signal.
0: The VSYNC synchronization is the rising edge
1: The VSYNC synchronization is the falling edge

[27:16] **DACVHD**: Analog VSYNC to HREF offset - specifies the offset (in pixel clock cycle) of the analog VSYNC signal relative to HREF. A value 0 means that analog VSYNC is rising one clock cycle after VREF.

[15:12] **Reserved**

[11:0] **DACVDO**: Analog VSYNC rising edge position relative to VREF
Specifies the offset (in number of half-line) of the analog VSYNC rising edge relative to VREF. The analog VSYNC signal rises from 0 to 1 when the half-line counter is equal to VTG1_DACVDO.

VTG1_DACVDS Analog VSYNC falling edge



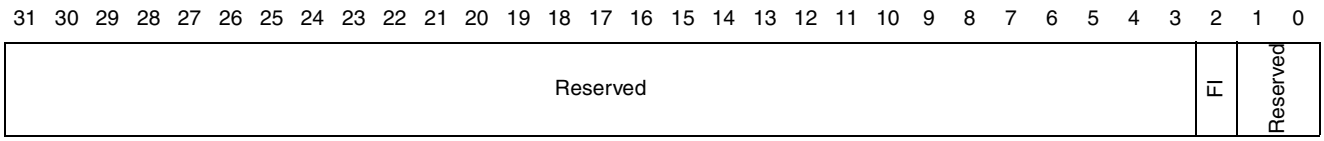
Address: *VOSBaseAddress* + 0x0144
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the analog Vsync falling edge relative to VREF.

[31:11] **Reserved**

[10:0] **DACVDS:** Vsync falling edge position relative to VREF
Specifies the offset (in number of half-line) of the analog VSYNC falling edge relative to VREF. The VSYNC output rises from 1 to 0 level when the half line counter is equal to VTG1_DACVDS.

Note: The order of VTG1_DACVDO and VTG1_DACVDS determine the polarity of the Vsync signal.

VTG1_MODE VTG mode of operations



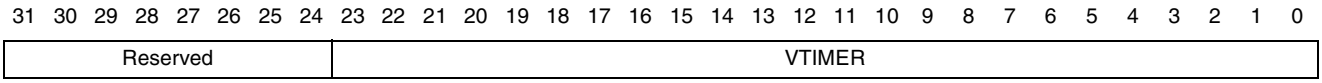
Address: *VOSBaseAddress* + 0x001C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Controls the operating mode of the VTG1.

[31:3] **Reserved**

[2] **FI**: Force interleaved
Allows VTG1 to be in interlaced mode when VTG1_HLFLN is even (typically for SMPTE 295M when line number is 1250).
0: No force 1: Force interleaved

[1:0] **Reserved**

VTG1_VTMR VSYNC delayed interrupt timer

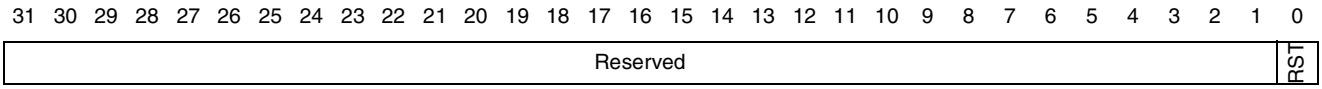


Address: *VOSBaseAddress* + 0x0020
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the delay of the PDVS interrupt.

[31:24] **Reserved**

[23:0] **VTIMER**
Holds the value, in pixel clock period units, of the delay between internal VREF and PDVS interrupt.

VTG1_DRST VTG1 raster reset

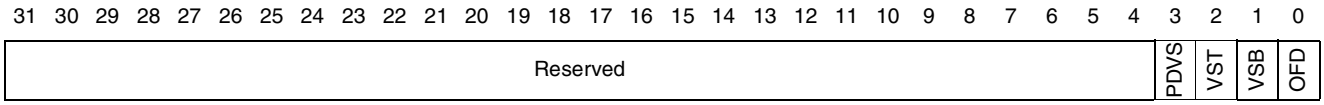


Address: *VOSBaseAddress* + 0x0024
Type: WO
Reset: 0
Description:

[31:1] **Reserved**

[0] **RST**
Writing to this bit resets the pixel counters and line counters in the raster generator. This reset is just activated once on writing. When activated, VTG1 reset also the generated VREF and HREF. For an interlaced raster, a top field is generated first.

VTG1_ITM Interrupt mask



Address: *VOSBaseAddress* + 0x0028

Type: R/W

Reset: 0

Description: Any bit set in this register enables the corresponding interrupt in the VTG1_IRQ line. An interrupt is generated whenever a bit in register VTG1_STA changes from 0 to 1 and the corresponding mask bit is set.

[31:4] **Reserved**

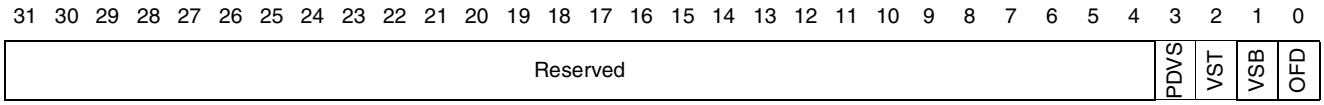
[3] **PDVS**: Programmable delay on VSync
This bit is set for a short time after VSync, with a delay programmed in the VTG1_VTMR register.

[2] **VST**: Vsync top
This bit is set at the beginning of the top field, corresponding to the falling edge of the internal BNOTT (bottom not top) signal.

[1] **VSB**: Vsync bottom
This bit is set at the beginning of the top field, corresponding to the rising edge of the internal BNOTT (bottom not top) signal.

[0] **OFD**: Output field: (only relevant in register VTG1_STA)
0: Bottom field.
1: Top field.

VTG1_ITS Interrupt status



Address: *VOSBaseAddress* + 0x002C

Type: RO

Reset: 0

Description: When a bit in register VTG1_STA changes from 0 to 1, the corresponding bit in register VTG1_ITS is set, independent of the state of VTG1_ITM. If any set VTG1_ITS bit is unmasked, the line VTG1_IRQ is asserted. Reading VTG1_ITS clears all bits in this register. When VTG1_ITS is zero, the VTG1_IRQ line returns de-asserted.

[31:4] **Reserved**

[3] **PDVS**: Programmable delay on VSync. This bit is set after VSync, with a delay programmed in VTG1_VTMR register.

[2] **VST**: Vsync top
This bit is set at the beginning of the top field, corresponding to the falling edge of the internal BNOTT (bottom not top) signal.

[1] **VSB**: Vsync bottom
This bit is set at the beginning of the top field, corresponding to the rising edge of the internal BNOTT (bottom not top) signal.

[0] **OFD**: Output field: (only relevant in register VTG1_STA)
0: Bottom field.
1: Top field.

VTG1_STA VTG1 status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												PDVS	VST	VSB	OFD

Address: *VOSBaseAddress* + 0x0030

Type: RO

Reset: 0

Description: Any change from 0 to 1 of any of these bits sets the corresponding bit of register VTG1_ITS, and can thus potentially cause an interrupt on the VTG1_IRQ line. PDVS, VST and VSB are pulses and are unlikely to be ever read as a 1.

[31:4] **Reserved**

[3] **PDVS**: Programmable delay on VSync
This bit is set after VSync, with a delay programmed in register VTG1_VTMR.

[2] **VST**: Vsync top
This bit is set at the beginning of the top field, corresponding to the falling edge of the internal BNOTT (bottom not top) signal.

[1] **VSB**: Vsync bottom
This bit is set at the beginning of the top field, corresponding to the rising edge of the internal BNOTT (bottom not top) signal.

[0] **OFD**: Output field: (only relevant in register VTG1_STA)
0: Bottom field.
1: Top field.

54.2 VTG 2

VTG2_CLKLN Clocks per line

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				CLKLN											

Address: *VOSBaseAddress* + 0x0038

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Defines the number of pixel per line.

[31:12] **Reserved**

[11:0] **CLKLN**
Specifies the total number of pixel clocks per horizontal period (complete horizontal line length).

VTG2_HDO HSYNC rising edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				HDO											

Address: *VOSBaseAddress* + 0x003C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the internal HSYNC rising edge relative to HREF.

[31:12] **Reserved**

[11:0] **HDO**: Determines the position of the Hsync rising edge relative to HREF
The Hsync signal rises from 0 to 1 level when the pixel counter is equal to VTG2_HDO.

Note: The order of VTG2_HDO and VTG2_HDS determines the polarity of the Hsync signal.

VTG2_HDS HSYNC output falling edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				HDS											

Address: *VOSBaseAddress* + 0x0040
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the position of the internal HSYNC falling edge relative to HREF.

[31:12] **Reserved**

[11:0] **HDS**: Determines the position of the Hsync falling edge relative to HREF
The Hsync output falls from 1 to 0 when the pixel counter is equal to VTG2_HDS.

Note: The order of VTG2_HDO and VTG2_HDS determines the polarity of the Hsync signal.

VTG2_HLFLN Half line per vertical

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				HLFLN											

Address: *VOSBaseAddress* + 0x0044
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the number of half-line per frame or field.

[31:12] **Reserved**

[11:0] **HLFLN**: Specifies the number of half line per vertical period in the raster display
An odd value indicates interlaced output and an even value indicates progressive output, except if overridden by VTG2_VTGMOD.FI in case of the 1250 line interlaced standard (SMPTE 295M).

VTG2_VDO VSYNC rising edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		VPOS	VPOL	VHD												Reserved					VDO										

Address: *VOSBaseAddress* + 0x0048

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Defines the position of the Vsync rising edge relative to VREF, the offset of VSYNC relative to HREF, the polarity of VSYNC and the position (late or early) of VSYNC relative to VREF.

[31:30] **Reserved**

[29] **VPOS:** VSYNC position (early or late) relative to VREF
0: VSYNC active edge is after VREF rising edge
1: VSYNC active edge is before VREF rising edge

[28] **VPOL:** VSYNC polarity relative to VREF - used to generate correctly the BNOTTOP signal.
0: The VSYNC synchronization is the rising edge
1: The VSYNC synchronization is the falling edge

[27:16] **VHD:** VSYNC to HREF offset - Specifies the offset (in pixel clock cycle) of the Vsync signal relative to HREF
0: VSYNC rises one clock cycle after VREF.

[15:12] **Reserved**

[11:0] **VDO:** VSYNC rising edge position relative to VREF.
Specifies the offset (in number of half-line) of the Vsync rising edge relative to VREF. The VSYNC output rises from 0 to 1 level when the half line counter is equal to VTG1_VDO.

Note: The order of VTG2_VDO and VTG2_VDS determines the polarity of the Vsync signal.

VTG2_VDS VSYNC falling edge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					VDS										

Address: *VOSBaseAddress* + 0x004C

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description:

[31:11] **Reserved**

[10:0] **VDS**: Determines the position (in hal-line) of the VSYNC signal relative to VREF
The VSYNC signal rises from 0 to 1 level when the half-line counter is equal to VTG2_VDO.

Note: The order of VTG2_VDO and VTG2_VDS determine the polarity of the VSYNC *signal pin.*

VTG2_VTG_MODE VTG mode of operations

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FI	SMD	

Address: *VOSBaseAddress* + 0x0050
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: This register controls the operating mode of the VTG2.

[31:3] **Reserved**

[2] **FI**: Force interleaved
This bit, when set, allows VTG2 to be in interlaced mode when VTG2_HLFLN is even (typically for SMPTE 295M when line number is 1250).
0: No force 1: Force interleaved

[1:0] **SMD**: Slave mode selection
00: VTG2 is master (Hsync and Vsync are internally generated).
01: VTG2 is slave (uses Hsync and Vsync from VTG1).
10: VTG2 is slave (uses Vsync from VTG1 and generates Hsync).

VTG2_VTMR VSYNC delayed interrupt timer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VTIMER																							

Address: *VOSBaseAddress* + 0x0054
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Defines the PVDS interrupt delay.

[31:24] **Reserved**

[23:0] **VTIMER**: Value, in pixel clock period units, of the delay between internal VREF and PDVS interrupt.

VTG2_DRST VTG2 raster reset

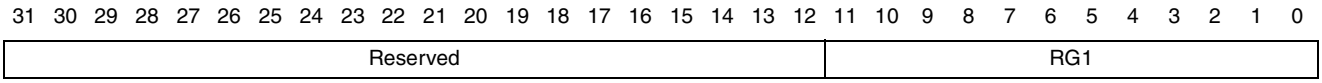
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RST

Address: *VOSBaseAddress* + 0x0058
Type: WO
Reset: 0
Description:

[31:1] **Reserved**

[0] **RST**: Reset
Resets the pixel counters and line counters in the raster generator. This reset is activated once on writing. When activated, the VTG2 reset also generates VREF and HREF. In case of an interlaced raster, a top field is generated first.

VTG2_R1 Range 1

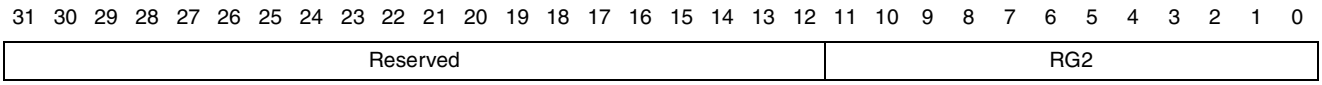


Address: *VOSBaseAddress* + 0x005C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Used in slave mode only.

[31:12] **Reserved**

[11:0] **RG1**
Holds the lower limit of pixel window for the detection of a bottom field. In slave mode, if VTG1 Vsync is detected in this range, VTG2 generate a bottom field. If VTG1 Vsync is detected out of this range, a top field is generated.

VTG2_R2 Range 2

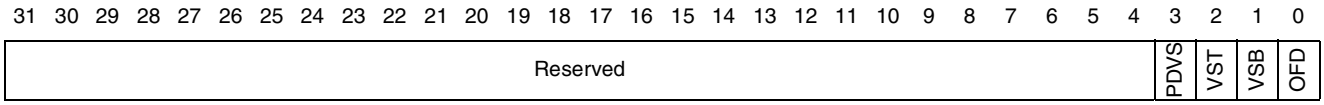


Address: *VOSBaseAddress* + 0x0060
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Used in slave mode only.

[31:12] **Reserved**

[11:0] **RG2**
Holds the upper limit of pixel window for the detection of a bottom field. In slave mode, if VTG1 Vsync is detected in this range, VTG2 generate a bottom field. If VTG1 Vsync is detected out of this range, a top field is generated.

VTG2_ITM Interrupt mask



Address: *VOSBaseAddress* + 0x0064

Type: R/W

Reset: 0

Description: Any bit set in this register enables the corresponding interrupt in the VTG2_IRQ line. An interrupt is generated whenever a bit in register VTG2_STA changes from 0 to 1 and the corresponding mask bit is set.

[31:4] **Reserved**

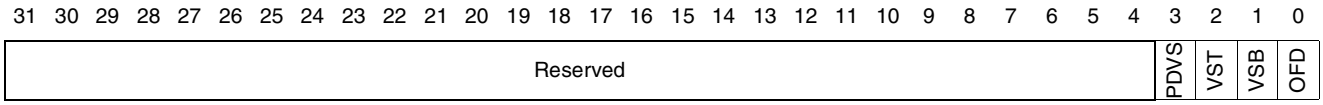
[3] **PDVS**: Programmable delay on VSync
This bit is set for a short time after VSync, with a delay programmed in VTG1_VTIMER register.

[2] **VST**: Vsync top
This bit is set for a short time at the beginning of the top field, corresponding to the falling edge of the internal BNOTT (bottom not top) signal.

[1] **VSB**: Vsync bottom
This bit is set for a short time at the beginning of the top field, corresponding to the rising edge of the internal BNOTT (bottom not top) signal.

[0] **OFD**: Output field (only relevant in register VTG2_STA)
0: Bottom filed
1: Top filled

VTG2_ITS Interrupt status



Address: *VOSBaseAddress* + 0x0068

Type: RO

Reset: 0

Description: When a bit in register VTG2_STA changes from 0 to 1, the corresponding bit in register VTG2_ITS is set, independent of the state of VTG2_ITM. If any set VTG2_ITS bit is unmasked, the line VTG2_IRQ is asserted. Reading VTG2_ITS clears all bits in this register. When VTG2_ITS is zero, the VTG2_IRQ line returns de-asserted.

[31:4] **Reserved**

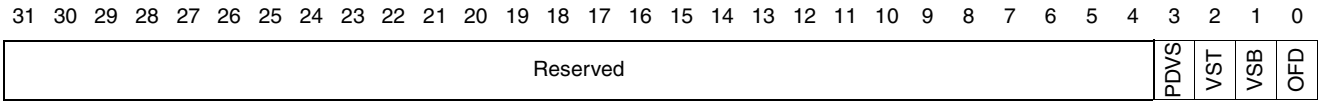
[3] **PDVS**: Programmable delay on Vsync
This bit is set with some delay after VSync. This delay is defined in VTG2_VTMR register.

[2] **VST**: Vsync top
This bit is set at the beginning of a top field, corresponding to the falling edge of the internal BNOTT (bottom not top) signal.

[1] **VSB**: Vsync bottom
This bit is set at the beginning of a bottom field, corresponding to the rising edge of the internal BNOTT (bottom not top) signal.

[0] **OFD**: Output field: (only relevant in register VTG2_STA)
0: bottom filled
1: top filled

VTG2_STA VTG2 status



Address: *VOSBaseAddress* + 0x006C

Type: R/W

Reset: 0

Description: This register contained a set of bits which represent the status of the video timing generator. any change from 0 to 1 of any of this bits sets the corresponding bit of register VTG2_ITS, and can thus potentially cause an interrupt on VTG2_IRQ line. PDVS, VST and VSB are pulses and are unlikely to be ever read as a 1.

[31:4] **Reserved**

[3] **PDVS**: Programmable delay on VSync.
This bit is set for a short time after VSync, with a delay programmed in VTG2_VTIMER register.

[2] **VST**: Vsync top
This bit is set at the beginning of a top field, corresponding to the falling edge of the internal BNOTT (bottom not top) signal.

[1] **VSB**: Vsync bottom
This bit is set at the beginning of a bottom field, corresponding to the rising edge of the internal BNOTT (bottom not top) signal.

[0] **OFD**: Output field: (only relevant in register VTG2_STA)
0: bottom filled
1: top filled

54.3 General VOS configuration

DSP_CFG_CLK Display and global configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					UPS_SEL1	DVP_SEL		HDMI_IDDQ	DVP_REF	UPS_DIV		UPS_SEL	NUP	HDMI_CFG	DESL

Address: *VOSBaseAddress* + 0x0070

Type: R/W

Reset: 0

Description: This register configures the upsampler, the HDMI serializer power-down and the DENC input.

[31:11] **Reserved**

[10] **UPS_SEL1**: Upsampler mode 1
0: UPS_SEL mode is used
1: x4 upsampling - Progressive SD source from main data path (ex: 27 MHz)

[9:8] **DVP_SEL**: Compositor to DVP datapath selection
00: Compositor output (9:0)
01: Compositor output (19:10)
10: Compositor output (29:20)

[7] **HDMI_IDDQ**: HDMI serializer power down
0: HDMI serializer is on
1: HDMI serializer is off

[6] **DVP_REF**: Reference to DVP
0: Main HREF and VREF (from VTG1) are transmitted to DVP
1: Aux HREF and VREF (from VTG2) are transmitted to DVP

[5:4] **UPS_DIV**: UPS post-filter division
The sum of programmed upsampler coefficients should be equal to the value programmed here for unitary overall filter gain.
00: 512
01: 256
10: 1024

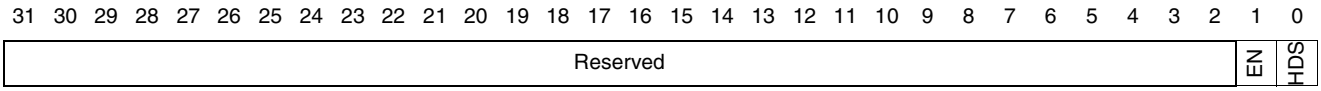
[3] **UPS_SEL**: Upsampler Mode (only when UPS_SEL1=0)
0: x2 upsampling - HD source from main data path
1: x4 upsampling - SD source from DENC

[2] **NUP**: Upsampler enable
0: Upsampler is bypassed 1: Upsampler is enabled

[1] **HDMI_CFG**: HDMI configuration
0: HDMI received RGB saturated 1: HDMI received YCbCr (601/709)

[0] **DESL**: DENC Selection
0: DENC takes auxiliary compositor output 1: DENC takes main compositor output

DSP_CFG_DIG Display digital output configuration



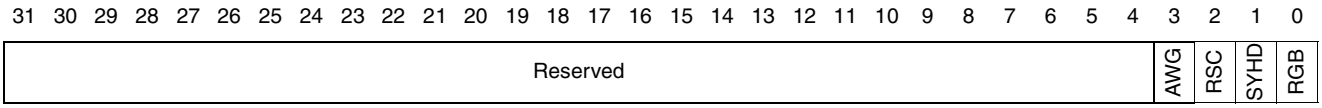
Address: *VOSBaseAddress* + 0x0074
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: This register configures the DVO digital display output.

[31:7] **Reserved**

[1] **EN**: DVO enable
0: Digital video output disabled 1: Digital video output enabled

[0] **HDS**: HD/SD selection
0: Main HD path on digital display 1: Aux SD path on digital display

DSP_CFG_ANA Main analog display output configuration



Address: *VOSBaseAddress* + 0x0078
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: This register configures the main analog display output.

[31:4] **Reserved**

[3] **AWG**: sync generation by programmable AWG or pre-defined EIA770.3
0: pre-defined EIA770.3 analog sync is used 1: user-defined AWG sync is used

[2] **RSC**: Rescale
0: No rescale.
1: Display signal is rescaled to allow insertion of sync in the DACs range.

[1] **SYHD**: Sync on HD path
0: No synchronization inserted.
1: Synchronization (EIA770.3 type) is inserted on analog outputs.

[0] **RGB**: RGB out
0: YPbPr/YCbCr is output. 1: RGB is output

54.4 Main video output configuration

DHDO_ACFG Analog output configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													P293	P295	SIC

Address: *VOSBaseAddress* + 0x007C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: This register configures the main analog display output.

[31:7] **Reserved**

[2] **P293**: Progressive SMPTE 293M
0: Any standard other than progressive SMPTE 293M (including interlaced 293M).
1: Progressive SMPTE 293M Standard.

[1] **P295**: Progressive SMPTE 295M
0: Any standard other than progressive SMPTE 295M (including interlaced 295M).
1: Progressive SMPTE 295M Standard.

[0] **SIC**: Synchronization in chroma:
0: Synchronization pulses only available on luma output.
1: Synchronization pulses also available on chroma output.

DHDO_ABROAD ABROAD value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				ABROAD											

Address: *VOSBaseAddress* + 0x0080
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **ABROAD**: Length of high and low level synchronization pulses in pixel clock units for all types of lines

DHDO_BBROAD BBROAD value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				BBROAD											

Address: *VOSBaseAddress* + 0x0084
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **BBROAD**: Length of high level and low level synchronization pulses during broad pulse lines, in pixel clock units

DHDO_CBROAD CBROAD value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				CBROAD											

Address: *VOSBaseAddress* + 0x0088
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **CBROAD**: Length of high and low level synchronization pulses during broad pulse line, in pixel clock units

DHDO_BACTIVE BACTIVE value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				BACTIVE											

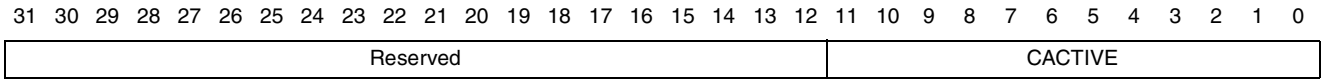
Address: *VOSBaseAddress* + 0x008C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **BACTIVE**: Length of high level plus zero level synchronization during pulses active video line, in pixel clock units

Note: (CACTIVE - BACTIVE) is the number of horizontal active pixels per line.

DHDO_CACTIVE CACTIVE value



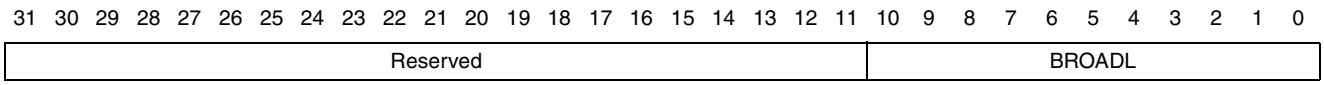
Address: *VOSBaseAddress* + 0x0090
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **CACTIVE**
Length of high level plus zero level plus active level synchronization during active video line, in pixel clock units.

Note: (CACTIVE - BACTIVE) is the number of horizontal active pixels per line.

DHDO_BROADL Broad pulse line number

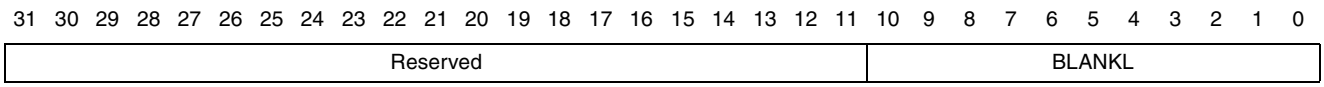


Address: *VOSBaseAddress* + 0x0094
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:11] **Reserved**

[10:0] **BROADL**
Number of broad pulse lines to be generated (in interlaced mode, these lines contain the pattern twice)

DHDO_BLANKL Blanking line number



Address: *VOSBaseAddress* + 0x0098
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: This register.

[31:11] **Reserved**

[10:0] **BLANKL**: Number of blanking lines to be generated.

DHDO_ACTIVEL Active line number

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					ACTIVEL										

Address: *VOSBaseAddress* + 0x009C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:11] **Reserved**

[10:0] **ACTIVEL**: Number of active lines to be generated.

DHDO_ZEROL Zero level values

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ZEROLC										Reserved						ZEROLL									

Address: *VOSBaseAddress* + 0x00A0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Level zero DAC entry values.

[31:26] **Reserved**

[25:16] **ZEROLC**: Zero level value for chroma

[15:10] **Reserved**

[9:0] **ZEROLL**: Zero level value for luma

DHDO_MIDL Middle level values

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MIDLC										Reserved						MIDLL									

Address: *VOSBaseAddress* + 0x00A4
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Middle level DACs entry values (middle value between zero and high level).

[31:26] **Reserved**

[25:16] **MIDLC:** Middle level value for chroma

[15:10] **Reserved**

[9:0] **MIDLL**: Middle level value for luma

DHDO_HI High level values

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						HIGHLC										Reserved						HIGHLL									

Address: *VOSBaseAddress* + 0x00A8
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Synchronization pulse high level DACs entry values.

[31:26] **Reserved**

[25:16] **HIGHLC**: High level value for chroma

[15:10] **Reserved**

[9:0] **HIGHLL**: High level value for luma

DHDO_MIDLO Mid low level values

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MIDLOWLC										Reserved						MIDLOWLL									

Address: *VOSBaseAddress* + 0x00AC
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Mid lowlevel DACs entry values (mid value between zero and low level).

[31:26] **Reserved**

[25:16] **MIDLOWLC**: Mid low level value for chroma

[15:10] **Reserved**

[9:0] **MIDLOWLL**: Mid low level value for luma

DHDO_LO Low level values

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						LOWLC										Reserved						LOWLL									

Address: *VOSBaseAddress* + 0x00B0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Synchronization pulse low level DACs entry values.

[31:26] **Reserved**

[25:16] **LOWLC**: Low level value for chroma

[15:10] **Reserved**

[9:0] **LOWLL**: Low level value for luma

DHDO_COLOR Main display output color space selection

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														240	601

Address: *VOSBaseAddress* + 0x00B4

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Allows the selection of the color space matrix to be used on RGB to YCbCr conversion before main display output.

[31:26] **Reserved**

[1] **240**: SMPTE 240M
0: ITU-R 601 or 709 is selected.
1: SMPTE 240M is selected.

[0] **601**: ITU-R 601 / 709
0: ITU-R 709 is selected.
1: ITU-R 601 is selected.

DHDO_YMLT Luma multiplication factor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						YMLT									

Address: *VOSBaseAddress* + 0x00B8

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 0

Description: Multiplication factor applied to the luma or green signal (Y/G) before the digital to analog converter to allow synchronization levels insertion.

Note: *G/Y = (YMLT x internal signal G/Y)/1024 + YOFF*

DHDO_CMLT Chroma multiplication factor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CMLT									

Address: *VOSBaseAddress* + 0x00BC

Type: R/W

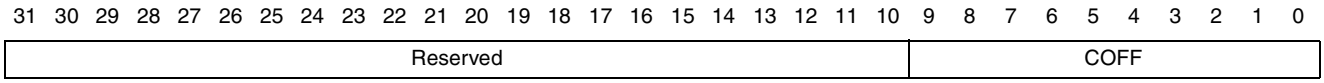
Buffer: Rebuffered on VSYNC

Reset: 0

Description: Multiplication factor applied to the chroma or red and blue signal (R/Pr and B/Pb) before the digital to analog converter to allow synchronization levels insertion.

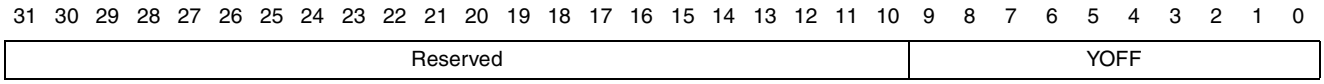
Note: *R/Pr = (CMLT x internal signal R/Cr)/1024 + COFF*
B/Pb = (CMLT x internal signal B/Cb)/1024 + COFF

DHDO_COFF Chroma offset



Address: *VOSBaseAddress* + 0x00C0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Offset applied to the chroma or red and blue signal (R/Pr and B/Pb) before the digital to analog converter to allow synchronization levels insertion.
Note: *R/Pr = (CMLT x internal signal R/Cr)/1024 + COFF*
B/Pb = (CMLT x internal signal B/Cb)/1024 + COFF

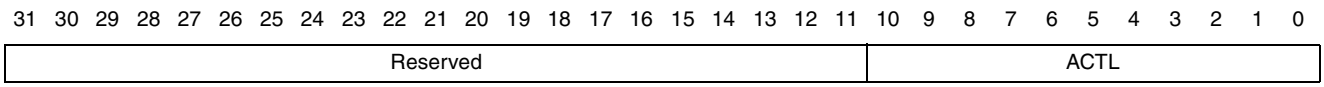
DHDO_YOFF Luma offset



Address: *VOSBaseAddress* + 0x00C4
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Offset applied to the luma or green signal (Y/G) before the digital to analog converter to allow synchronization levels insertion.
Note: *G/Y = (YMLT x internal signal G/Y)/1024 + YOFF*

54.5 Digital SD video out

C656_ACTL Active line



Address: *VOSBaseAddress* + 0x00C8
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Number of active lines

C656_BACT Beginning of active video

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					BACT										

Address: *VOSBaseAddress* + 0x00CC
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Duration in pixel clock between internal HREF and start of active video pixels

C656_BLANKL Blanking lines

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					BLANKL										

Address: *VOSBaseAddress* + 0x00D0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Number of blanking lines generated.

C656_EACT End of active video

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					EACT										

Address: *VOSBaseAddress* + 0x00D4
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Duration in pixel clocks between internal HREF and end of active video pixels.

C656_PAL PAL/NTSC selection

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															PAL

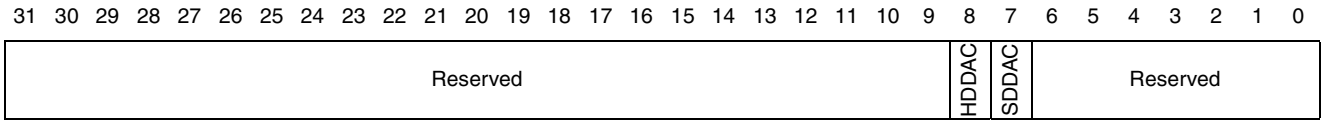
Address: *VOSBaseAddress* + 0x00D8
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: Adjusts timing between PAL and NTSC standards.

[31:1] **Reserved**

[0] **PAL**: PAL/NTSC
0: NTSC is selected.
1: PAL is selected.

54.6 HD DAC configuration

DSP_CFG_DAC HD and SD DACs configuration



Address: *VOSBaseAddress* + 0x00DC
Type: R/W
Reset: 0
Description: This register enables the HD and SD DACs.

[31:9] **Reserved**

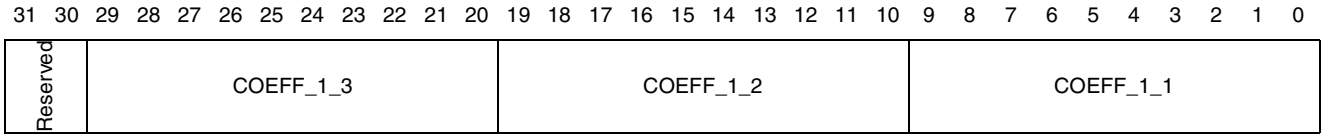
[8] **HDDAC**: HD DAC power-on
0: HD DACs are power-off (default value)
1: HD DACs are power-on

[7] **SDDAC**: SD DAC power-on
0: SD DACs are power-off (default value)
1: SD DACs are power-on

[6:0] **Reserved**

54.7 Upsampler tap coefficients

UPSMPL_CSET1_1 Upsampler coefficient set 1 - coeff tap 1, 2, 3



Address: *VOSBaseAddress* + 0x00E4
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

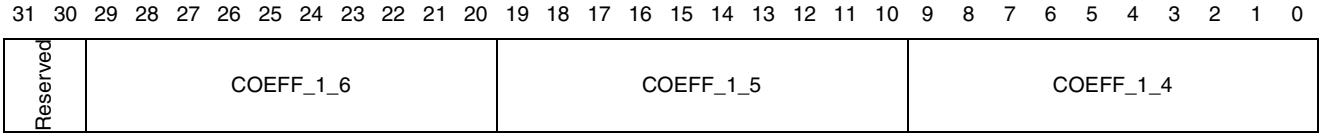
[31:30] **Reserved**

[29:20] **COEFF_1_3**: Coefficient applied to tap 3 at first phase (for x2 or x4 upsampling)

[19:10] **COEFF_1_2**: Coefficient applied to tap 2 at first phase (for x2 or x4 upsampling)

[9:0] **COEFF_1_1**: Coefficient applied to tap 1 at first phase (for x2 or x4 upsampling)

UPSMPL_CSET1_2 Upsampler coefficient set 1 - coeff tap 4, 5, 6



Address: *VOSBaseAddress* + 0x00E8
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

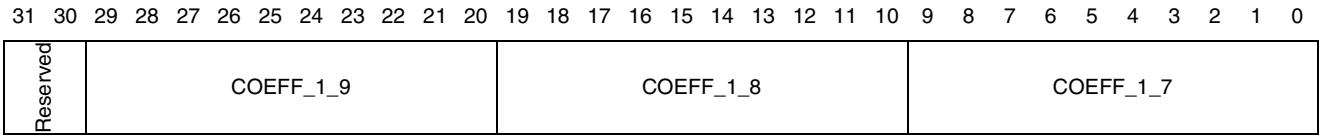
[31:30] **Reserved**

[29:20] **COEFF_1_6: Coefficient applied to tap 6 at first phase (for x2 or x4 upsampling)**

[19:10] **COEFF_1_5: Coefficient applied to tap 5 at first phase (for x2 or x4 upsampling)**

[9:0] **COEFF_1_4: Coefficient applied to tap 4 at first phase (for x2 or x4 upsampling)**

UPSMPL_CSET1_3 Upsampler coefficient set 1 - coeff tap 7, 8, 9



Address: *VOSBaseAddress* + 0x00EC
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

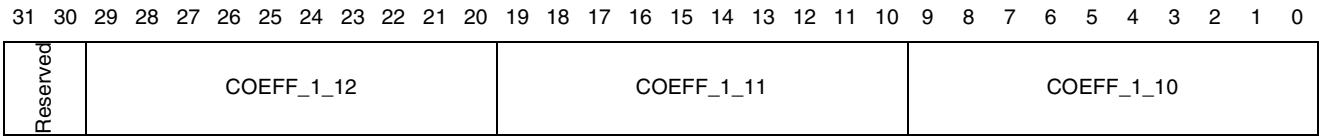
[31:30] **Reserved**

[29:20] **COEFF_1_9**: Coefficient applied to tap 9 at first phase (for x2 or x4 upsampling)

[19:10] **COEFF_1_8**: Coefficient applied to tap 8 at first phase (for x2 or x4 upsampling)

[9:0] **COEFF_1_7**: Coefficient applied to tap 7 at first phase (for x2 or x4 upsampling)

UPSMPL_CSET1_4 Upsampler coefficient set 1 - coeff tap 10, 11, 12



Address: *VOSBaseAddress* + 0x00F0
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

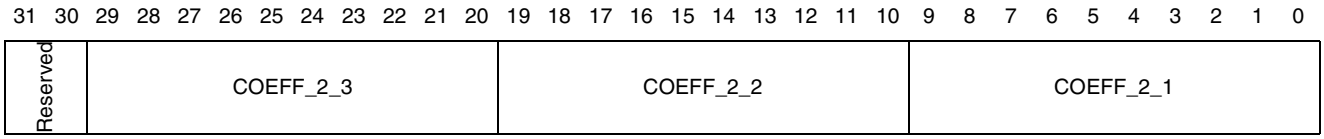
[31:30] **Reserved**

[29:20] **COEFF_1_12**: Coefficient applied to tap 10 at first phase (for x2 or x4 upsampling)

[19:10] **COEFF_1_11**: Coefficient applied to tap 11 at first phase (for x2 or x4 upsampling)

[9:0] **COEFF_1_10**: Coefficient applied to tap 12 at first phase (for x2 or x4 upsampling)

UPSMPL_CSET2_1 Upsampler coefficient set 2 - coeff tap 1, 2, 3



Address: *VOSBaseAddress* + 0x00F4
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

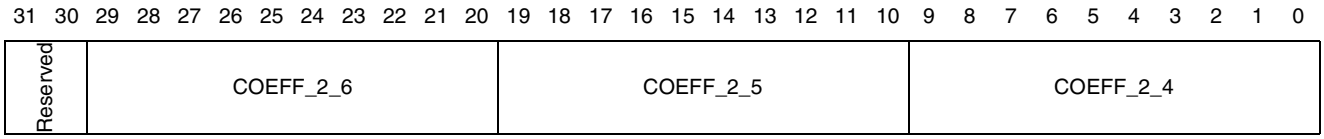
[31:30] **Reserved**

[29:20] **COEFF_2_3**: Coefficient applied to tap 3 at second phase (for x2 or x4 upsampling)

[19:10] **COEFF_2_2**: Coefficient applied to tap 2 at second phase (for x2 or x4 upsampling)

[9:0] **COEFF_2_1**: Coefficient applied to tap 1 at second phase (for x2 or x4 upsampling)

UPSMPL_CSET2_2 Upsampler coefficient set 2 - coeff tap 4, 5, 6



Address: *VOSBaseAddress* + 0x00F8
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

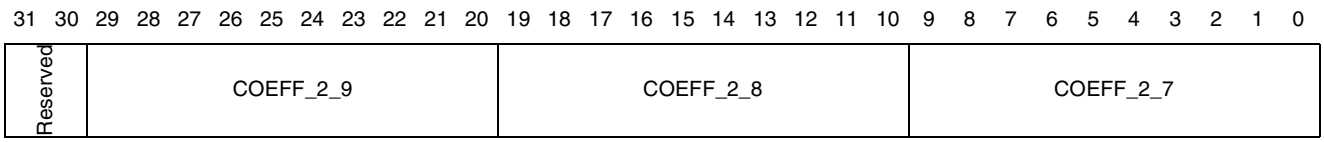
[31:30] **Reserved**

[29:20] **COEFF_2_6**: Coefficient applied to tap 6 at second phase (for x2 or x4 upsampling)

[19:10] **COEFF_2_5**: Coefficient applied to tap 5 at second phase (for x2 or x4 upsampling)

[9:0] **COEFF_2_4**: Coefficient applied to tap 4 at second phase (for x2 or x4 upsampling)

UPSMPL_CSET2_3 Upsampler coefficient set 2 - coeff tap 7, 8, 9



Address: *VOSBaseAddress* + 0x00FC
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

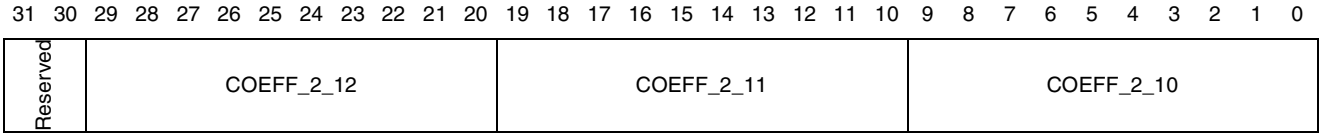
[31:30] **Reserved**

[29:20] **COEFF_2_9**: Coefficient applied to tap 9 at second phase (for x2 or x4 upsampling)

[19:10] **COEFF_2_8**: Coefficient applied to tap 8 at second phase (for x2 or x4 upsampling)

[9:0] **COEFF_2_7**: Coefficient applied to tap 7 at second phase (for x2 or x4 upsampling)

UPSMPL_CSET2_4 Upsampler coefficient set 2 - coeff tap 10, 11, 12



Address: *VOSBaseAddress* + 0x0100
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (HD x2 or SD x4).

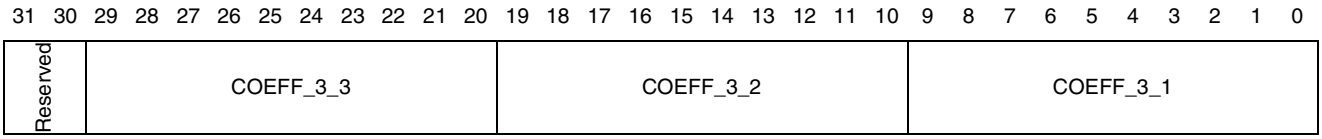
[31:30] **Reserved**

[29:20] **COEFF_2_12**: Coefficient applied to tap 12 at second phase (for x2 or x4 upsampling)

[19:10] **COEFF_2_11**: Coefficient applied to tap 11 at second phase (for x2 or x4 upsampling)

[9:0] **COEFF_2_10**: Coefficient applied to tap 10 at second phase (for x2 or x4 upsampling)

UPSMPL_CSET3_1 Upsampler coefficient set 3 - coeff tap 1, 2, 3



Address: *VOSBaseAddress* + 0x0104
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

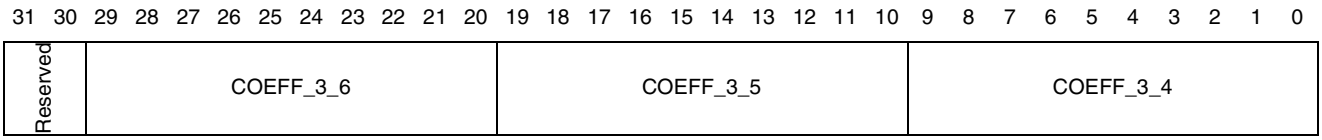
[31:30] **Reserved**

[29:20] **COEFF_3_3**: Coefficient applied to tap 3 at third phase (for x4 upsampling)

[19:10] **COEFF_3_2**: Coefficient applied to tap 2 at third phase (for x4 upsampling)

[9:0] **COEFF_3_1**: Coefficient applied to tap 1 at third phase (for x4 upsampling)

UPSMPL_CSET3_2 Upsampler coefficient set 3 - coeff tap 4, 5, 6



Address: *VOSBaseAddress* + 0x0108
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

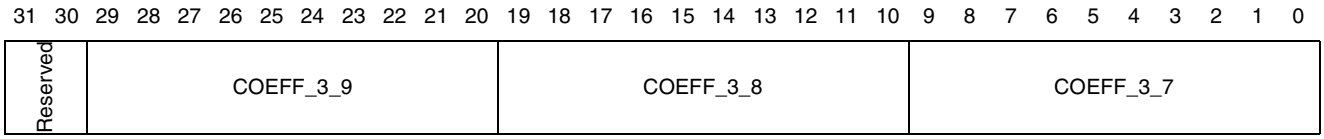
[31:30] **Reserved**

[29:20] **COEFF_3_6**: Coefficient applied to tap 6 at third phase (for x4 upsampling)

[19:10] **COEFF_3_5**: Coefficient applied to tap 5 at third phase (for x4 upsampling)

[9:0] **COEFF_3_4**: Coefficient applied to tap 4 at third phase (for x4 upsampling)

UPSMPL_CSET3_3 Upsampler coefficient set 3 - coeff tap 7, 8, 9



Address: *VOSBaseAddress* + 0x010C
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

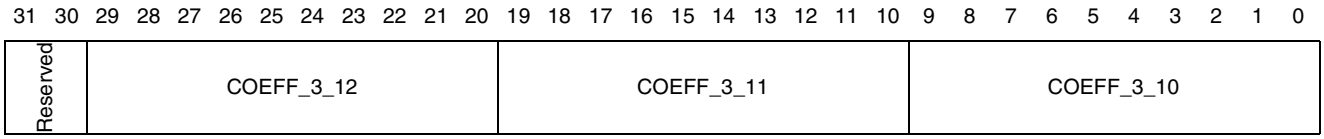
[31:30] **Reserved**

[29:20] **COEFF_3_9**: Coefficient applied to tap 9 at third phase (for x4 upsampling)

[19:10] **COEFF_3_8**: Coefficient applied to tap 8 at third phase (for x4 upsampling)

[9:0] **COEFF_3_7**: Coefficient applied to tap 7 at third phase (for x4 upsampling)

UPSMPL_CSET3_4 Upsampler coefficient set 3 - coeff tap 10, 11, 12



Address: *VOSBaseAddress* + 0x0110
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

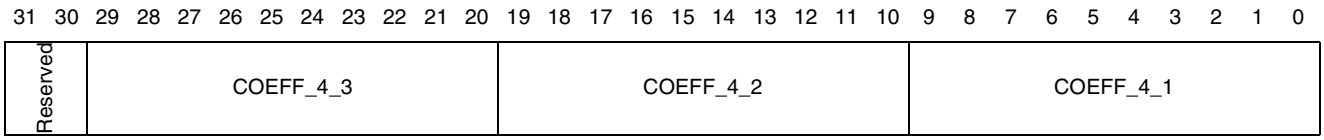
[31:30] **Reserved**

[29:20] **COEFF_3_12**: Coefficient applied to tap 12 at third phase (for x4 upsampling)

[19:10] **COEFF_3_11**: Coefficient applied to tap 11 at third phase (for x4 upsampling)

[9:0] **COEFF_3_10**: Coefficient applied to tap 10 at third phase (for x4 upsampling)

UPSMPL_CSET4_1 Upsampler coefficient set 4 - coeff tap 1, 2, 3



Address: *VOSBaseAddress* + 0x0114
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

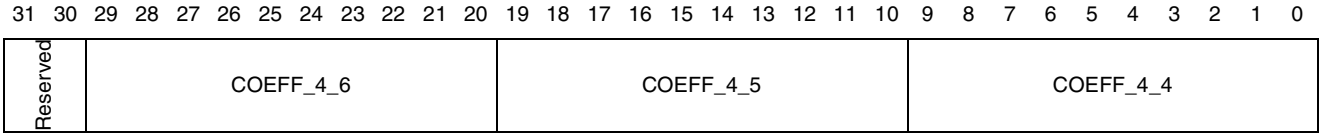
[31:30] **Reserved**

[29:20] **COEFF_4_3**: Coefficient applied to tap 3 at fourth phase (for x4 upsampling)

[19:10] **COEFF_4_2**: Coefficient applied to tap 2 at fourth phase (for x4 upsampling)

[9:0] **COEFF_4_1**: Coefficient applied to tap 1 at fourth phase (for x4 upsampling)

UPSMPL_CSET4_2 Upsampler coefficient set 1 - coeff tap 4, 5, 6



Address: *VOSBaseAddress* + 0x0118
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

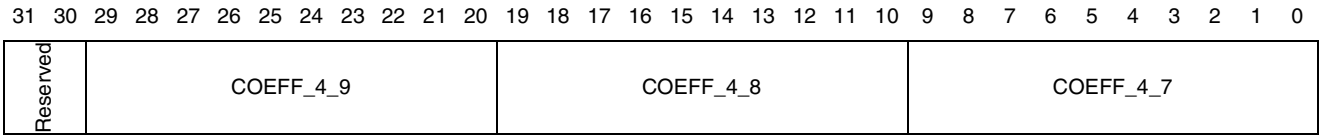
[31:30] **Reserved**

[29:20] **COEFF_4_6**: Coefficient applied to tap 6 at fourth phase (for x4 upsampling)

[19:10] **COEFF_4_5**: Coefficient applied to tap 5 at fourth phase (for x4 upsampling)

[9:0] **COEFF_4_4**: Coefficient applied to tap 4 at fourth phase (for x4 upsampling)

UPSMPL_CSET4_3 Upsampler coefficient set 4 - coeff tap 7, 8, 9



Address: *VOSBaseAddress* + 0x011C
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

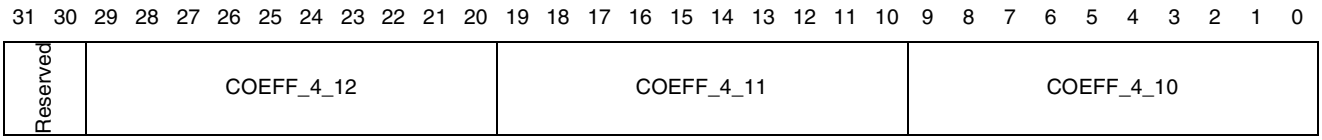
[31:30] **Reserved**

[29:20] **COEFF_4_9**: Coefficient applied to tap 9 at fourth phase (for x4 upsampling)

[19:10] **COEFF_4_8**: Coefficient applied to tap 8 at fourth phase (for x4 upsampling)

[9:0] **COEFF_4_7**: Coefficient applied to tap 7 at fourth phase (for x4 upsampling)

UPSMPL_CSET4_4 Upsampler coefficient set 4 - coeff tap 10, 11, 12



Address: *VOSBaseAddress* + 0x0120
Type: R/W
Reset: 0
Description: SRC coefficients used for upsampling (SD only).

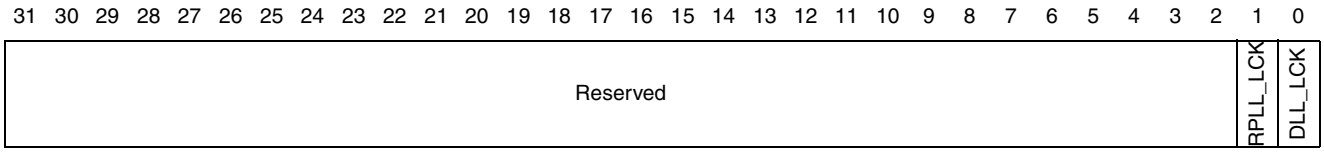
[31:30] **Reserved**

[29:20] **COEFF_4_12**: Coefficient applied to tap 12 at fourth phase (for x4 upsampling)

[19:10] **COEFF_4_11**: Coefficient applied to tap 11 at fourth phase (for x4 upsampling)

[9:0] **COEFF_4_10**: Coefficient applied to tap 10 at fourth phase (for x4 upsampling)

HDMI_PHY_LCK_STA HDMI PHY lock status



Address: *VOSBaseAddress* + 0x0124

Type: RO

Reset: 0

Description: This register provides the lock status of the HDMI rejection PLL and the lock status of the HDMI DLL serializer.

[31:2] **Reserved**

[1] **RPLL_LCK**: Lock status of the rejection PLL
0: Unlock 1: Lock

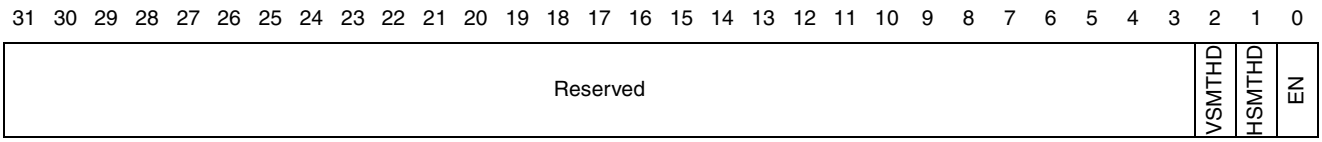
[0] **DLL_LCK**: Lock status of the HDMI PHY DLL
0: Unlock 1: Lock

The reset sequence of the HDM serializer is as follows:

1. Setup the HDMI clocks in the clock generator B.
2. Reset the rejection PLL.
3. Wait for the PLL lock and check the lock status.
4. Reset the HDMI serializer.
5. Wait for the serializer DLL lock and check the lock status.
6. Configure the HDMI.
7. reset the VTGs

54.8 Waveform generator configuration

AWG_CTRL_0 AWG control



Address: *AWGBaseAddress* + 0x0000

Type: RW

Reset: 0

Description: Enables the waveform generator and defines the method used for syncs generation.

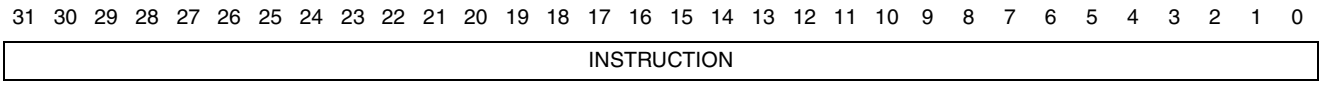
[31:3] **Reserved**

[1] **VSMTHD**: vertical synchronization method
0: Frame based synchronization
1: Field based synchronization

[1] **HSMTHD**: horizontal synchronization method
0: AWGHSYNC occurrence causes the instruction pointer to increment (has priority on any instruction effect)
1: AWGHSYNC occurrence has no effect on the instruction pointer

[0] **EN**: waveform generator enable
0: Disable - all internal counters and variables are cleared, no instruction is executed
1: Enable

AWG_RAMn AWG instruction n



Address: *AWGBaseAddress* + 0x0100 + *n* x 0x04 (*n*= 0 to 45)

Type: RW

Reset: 0

Description: Provides a memory mapped view of the instruction RAM.

55 Digital encoder (DENC)

55.1 Digital encoder overview

This is the final stage of the video pipeline of the device and is a high-performance PAL/SECAM/NTSC digital encoder referred to as the DENC. The DENC converts a 4:4:4/4:2:2 digital video stream into a standard analog baseband PAL/SECAM/NTSC signal and into RGB and YUV analog components.

The DENC can handle interlaced mode (in all standards) and non-interlaced mode in PAL and NTSC. It can perform closed-captions, CGMS, WSS, Teletext and VPS encoding and allows Macrovision^{TM} 7.01/ 6.1 copy protection. There are two sets of triple DACs, on each of which three analog output pins are available, and it is possible to output either (S-VHS(Y/C) + CVBS) or (V + Y + U) or (RGB).

As implemented in the STx7100, the encoder operates in one of several slave modes, where it locks onto incoming Hsync and OddEven signals from one of the two VTGs integrated in the STx7100. The VTGs are programmed by registers described in Chapter 53: *Video output stage (VOS) on page 588*. The DENC should therefore be programmed to use one of the slave modes based on these two signals. The VTG used by the DENC is automatically selected to be that of the mixer in the compositor whose display is being output by the DENC. An autotest mode is also provided.

The main functions are controlled using a register interface with the CPU. Each register is mapped into the least significant byte of a 32-bit register space in the STx7100. The DENC registers are described in Chapter 56 on page 665.

55.2 Data input format

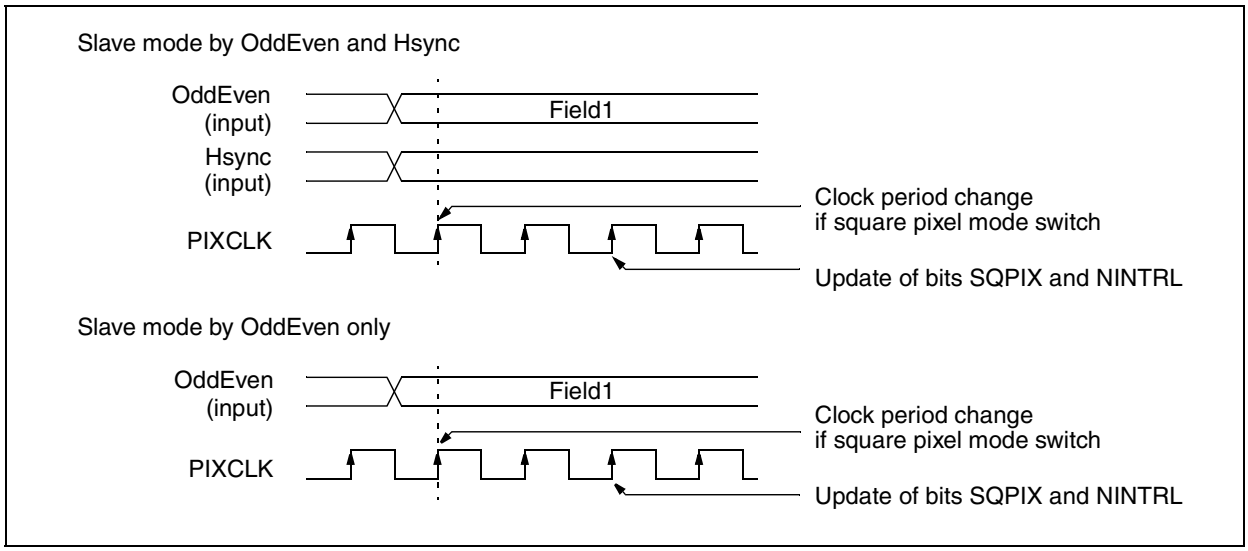
The digital input is a time-multiplexed [[Cb,Y,Cr],Y] 8-bit video stream or 4:4:4 [Y,Cb,Cr] 24-bit video stream. Input samples are latched on the rising of the clock signal PIXCLK.

The DENC is able to encode interlaced (in all standards) and non-interlaced (in PAL and NTSC) video. One bit is sufficient to automatically direct the DENC to process non-interlaced video. Update is performed internally on the first frame sync active edge following the programing of bit NINTRL in register DENC_CFG2. The non-interlaced mode is a 624/2 = 312 line mode or a 524/2 = 262 line mode, where all fields of the frame are identical.

The DENC also supports square pixel mode in PAL and NTSC modes with the appropriate clock reference. (PAL 29.5 MHz, NTSC 24.5454 MHz). Square pixel mode is programmed using bit SQPIX in DENC_CFG7.

Square pixel and/or non-interlaced modes are updated on the beginning of the frame (Figure 200).

Figure 200: Square pixel and/or non-interlaced mode switch



Note: If on-the-fly format changing is required, clock switching must be synchronized onto the start of the frame as shown in the above waveform. Internally, update of bits SQPIX and NINTRL is taken into account at the beginning of a new frame. (Bits SQPIX and NINTRL are in DENC_CFG7 and DENC_CFG2 respectively.)

55.3 Video timing

The burst sequences are internally generated, subcarrier generation being performed numerically with PIXCLK as reference. Four-frame bursts are generated for PAL or two-frame bursts for NTSC. Rise and fall times of the synchronization tips and burst envelope are internally controlled according to the relevant ITU-R and SMPTE recommendations. The six-frame subcarrier phase sequence is generated in SECAM (Section 55.9).

Figure 201 to Figure 206 show typical VBI (vertical blanking interval) waveforms.

Incoming YCbCr data may be encoded on those lines of the VBI with no line sync pulses or pre/post-equalization pulses (see figures). This mode of operation is referred to as **partial blanking** and is the default set-up. It allows the encoded waveform to keep any VBI data present in digitized form in the incoming YCbCr stream (for example, supplementary closed-caption lines). In SECAM mode, only Y data are encoded, Cr and Cb are ignored. Alternatively, the complete VBI may be **fully blanked**, so no incoming YCbCr data is encoded on these lines. Full or partial blanking is set by register bit DENC_CFG1.BLKLI.

For 525/60 systems, with the SMPTE line numbering convention:

- complete VBI consists of lines 1 to 19 and the second half of lines 263 to 282,
- partial VBI consists of lines 1 to 9 and the second half of lines 263 to 272,
- line 282 is either fully blanked or fully active.

For 625/50 systems, with the CCIR line numbering convention:

- complete VBI consists of the second half of lines 623 to 22 and lines 311 to 335,
- partial VBI consists of the second half of lines 623 to 5 and lines 311 to 318,
- line 23 is always fully active.

Figure 201: PAL-BDGHI, PAL-N typical VBI waveform, interlaced mode (ITU-R625 line numbering)

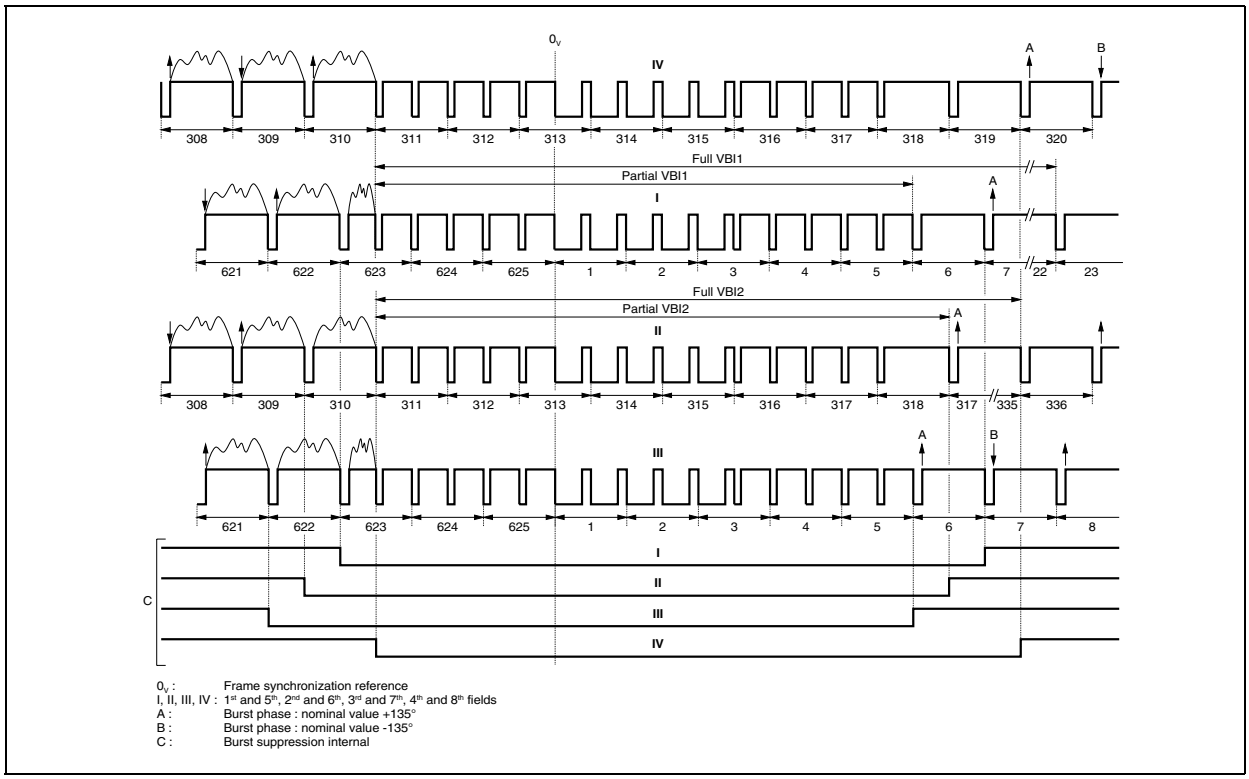


Figure 202: PAL-BDGHI, PAL-N typical VBI waveform, non-interlaced mode (CCIR-like line numbering)

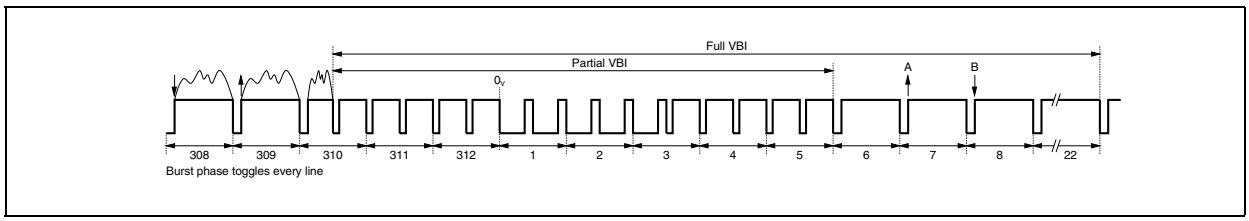


Figure 203: NTSC-M typical VBI waveforms, interlaced mode (SMPTE-525 line numbering)

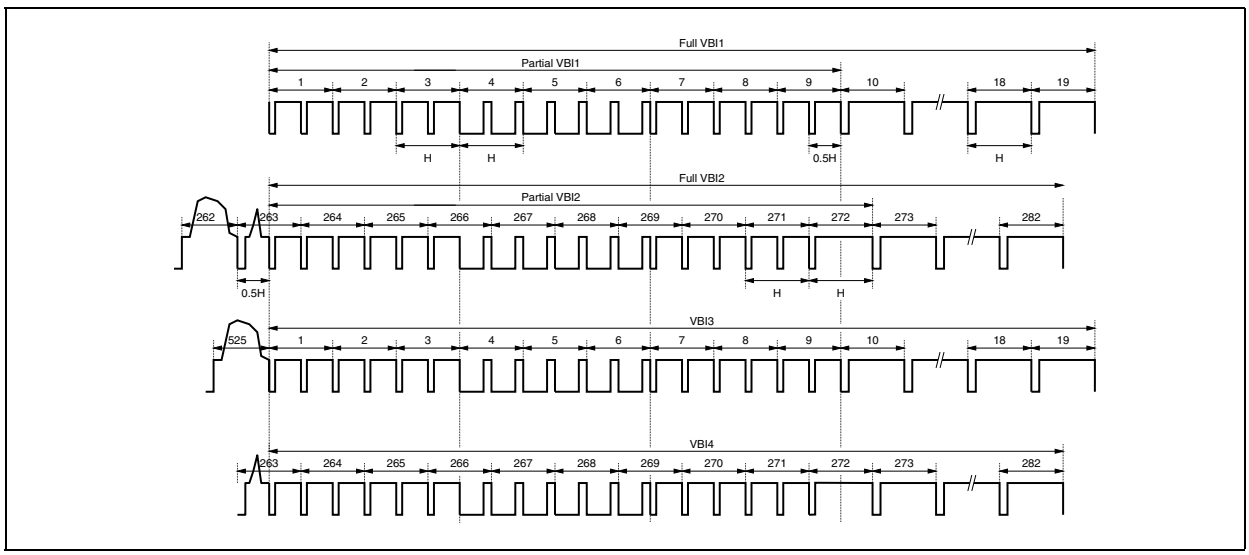


Figure 204: NTSC-M typical VBI waveforms, non-interlaced mode (SMPTE-like line numbering)

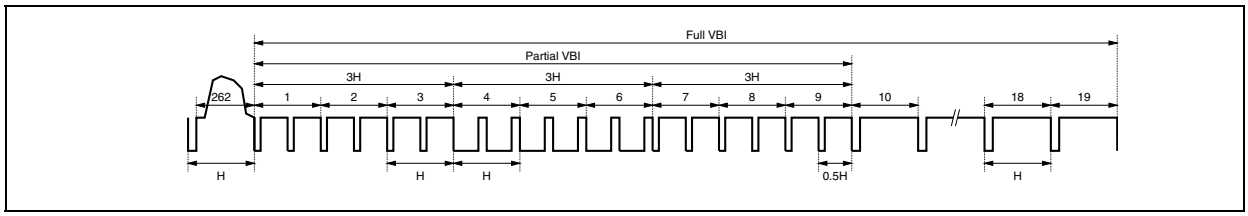


Figure 205: PAL-M typical VBI waveforms, interlaced mode (ITU-R/CCIR-525 line numbering)

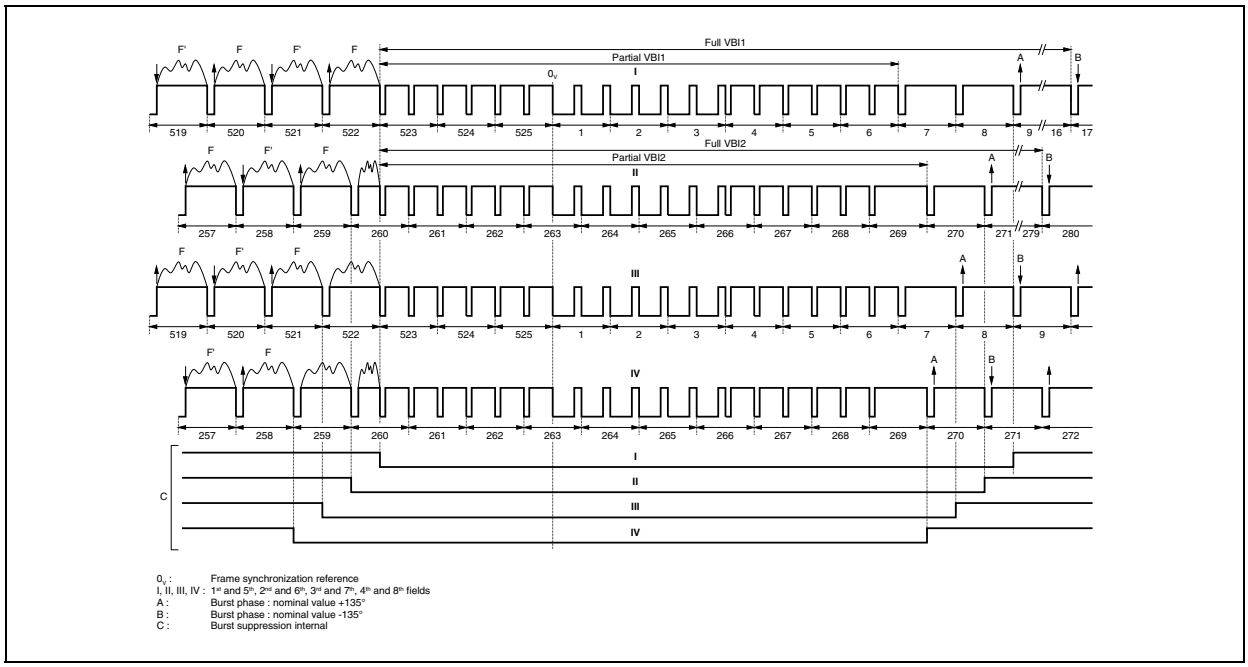
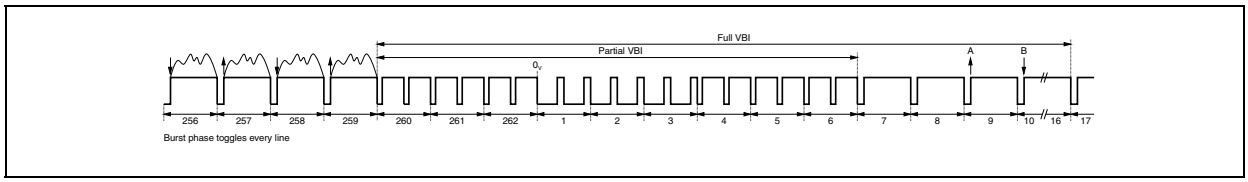


Figure 206: PAL-M typical VBI waveforms, non-interlaced mode (ITU-R/CCIR-like line numbering)



55.4 Reset procedure

A hardware reset sets the DENC in Hsync + OddEven (line-locked) slave mode; for NTSC-M, interlaced ITU-R601 encoding closed-captioning, WSS, VPS, Teletext and CGMS encoding are all disabled.

The configuration can then be customized by writing into the appropriate registers. A few registers are never reset, their contents are unknown until the first loading (see Chapter 56 on page 665).

A software reset may also be performed by setting bit SOFTRESET in register DENC_CFG6. The device responds in a similar way as after a hardware reset except that the configuration registers DENC_CFG0 to DENC_CFG8 are not altered.

55.5 Digital encoder synchronization

The following slave modes are available:

- OddEven(Vsync) + Hsync based (line-based sync),
- OddEven(Vsync)-only based (frame-based sync),

OddEven and Hsync signals come from the STx7100's video timing generators VTGn (see output stage Section 53.3: *Video timing generators (VTG) on page 592*).

55.5.1 Line-based synchronization

OddEven + Hsync based synchronization

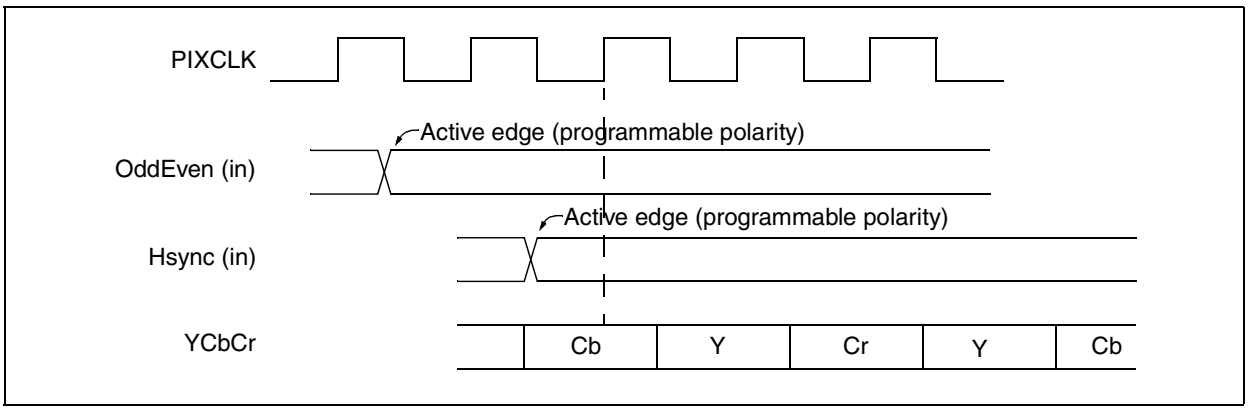
Synchronization is performed on a line-by-line basis by locking onto incoming OddEven and Hsync signals. Refer to Figure 207 for waveforms and timings. The polarities of the active edges of Hsync and OddEven are programmable and independent.

The first active edge of OddEven initializes the internal line counter but encoding of the first line does not start until an Hsync active edge is detected (at the earliest, an Hsync transition may be at the same time as OddEven). At this point, the internal sample counter is initialized and encoding of the first line starts. Then, encoding of each subsequent line is individually triggered by Hsync active edges.

The phase relationship between Hsync and the incoming YCbCr data is normally such that the first clock rising edge following the Hsync active edge samples Cb (that is, a blue chroma sample within the YCbCr stream). However, the incoming sync signals (Hsync + OddEven) may be internally delayed by up to three clock cycles to cope with different data/sync phases, using configuration bits SYNCIN_AD in DENC_CFG4. The DENC is thus fully slaved to the Hsync signal, which means that lines may contain more or fewer samples than usual.

- If the digital line is shorter than its nominal value, the sample counter is re-initialized when the early Hsync arrives and all internal synchronization signals are re-initialized.
- If the digital line is longer than its nominal value, the sample counter stops when it reaches its nominal end-of-line value and waits for the late Hsync before re-initializing.

Figure 207: Hsync + OddEven based slave mode sync signals



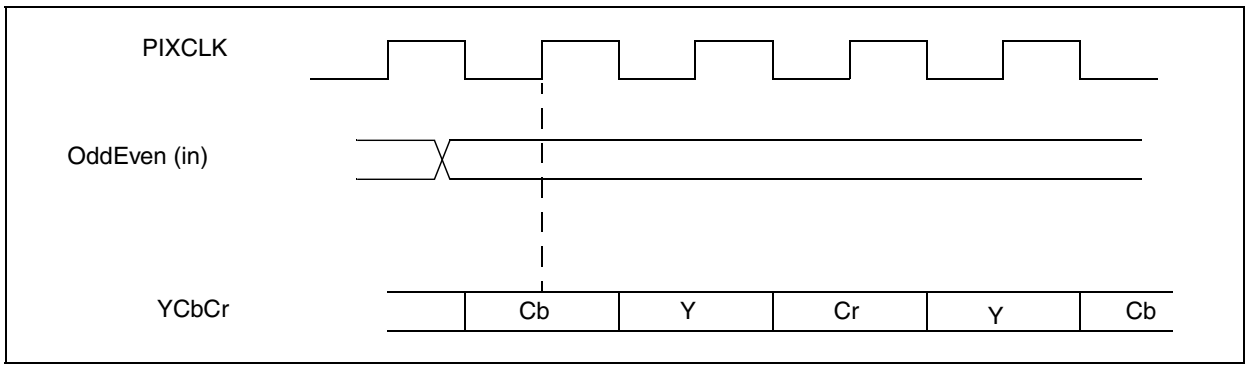
Note: This figure is valid for bits SYNCIN_AD = default.

55.5.2 Frame-based synchronization

OddEven-only based synchronization

Synchronization is performed on a frame-by-frame basis by locking onto an incoming OddEven signal. A line sync signal is derived internally. See to Figure 208 for waveforms and timings. The phase relationship between OddEven and the incoming YCbCr data is normally such that the first clock rising edge following the OddEven active edge samples Cb (that is, a blue chroma sample within the YCbCr stream). However, the incoming OddEven signal may be internally delayed by up to three clock cycles to cope with different data/sync phasing, using configuration bits SYNCIN_AD in DENC_CFG4.

Figure 208: OddEven based slave mode sync signals



Note: This figure is valid for bits SYNCIN_AD = default

The first active edge of OddEven triggers generation of the analog sync signals and encoding of the incoming video data. Since frames are supposed to be of constant duration, the next OddEven active transition is expected at a precise time after the last OddEven detected.

So once an active OddEven edge has been detected, checks are performed for the presence of the following OddEvens at the expected instants.

Encoding and analog sync generation continue unless three successive fails of these checks occur.

In this case, there are three possibilities according to the configuration programmed in registers DENC_CFG0 and DENC_CFG1:

- If FREERUN is enabled, the DENC carries on generating analog video just as though the expected OddEven edge is present. However, it resynchronizes onto the next OddEven active edge detected, whatever its location.
- If FREERUN is disabled but SYNCOK is set in the configuration registers, the DENC sets the active portion of the TV line to black level but carries on outputting the analog sync tips (on Ys and CVBS). When programmed, Macrovision™ pseudo-sync pulses and AGC pulses are also present in the analog sync waveform.
- If FREERUN is disabled and SYNCOK is not set, all analog video is at black level and no analog sync tips are output.

This mode is a frame-based sync mode, as opposed to a field-based sync mode. This means that only one edge type (rising or falling, according to programming) is of interest to the DENC; the other is ignored.

55.5.3 Autotest mode

An autotest mode is available, which causes the DENC to produce a color bar pattern, in the appropriate standard, independently from the video input.

The autotest mode is started by setting the SYNC to 7 (register DENC_CFG0). Table 179 shows the decimal values of Y, Cr and Cb corresponding to the autotest color bar.

Table 179: Autotest colors

	Y	Cr	Cb
Black	16	128	128
Blue	36	116	212
Red	64	212	100
Magenta	84	200	184
Green	112	56	72
Cyan	136	44	156
Yellow	160	140	44
White	236	128	128

The corresponding decimal output values just before the DACs are shown in Figure 209 and Figure 210. Both figures show the static values corresponding to the input values in Table 179.

Figure 209: Luminance output levels in autotest for NTSC without set-up

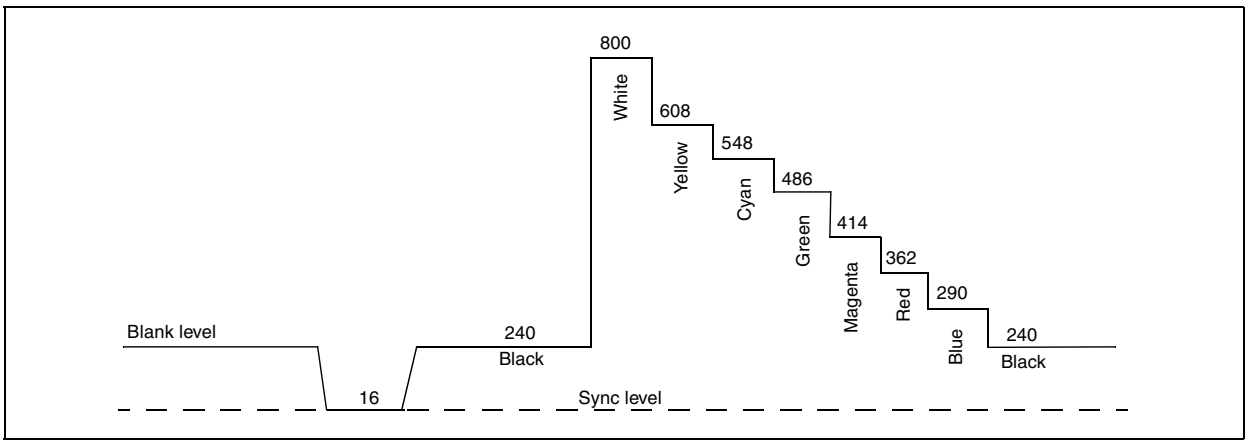
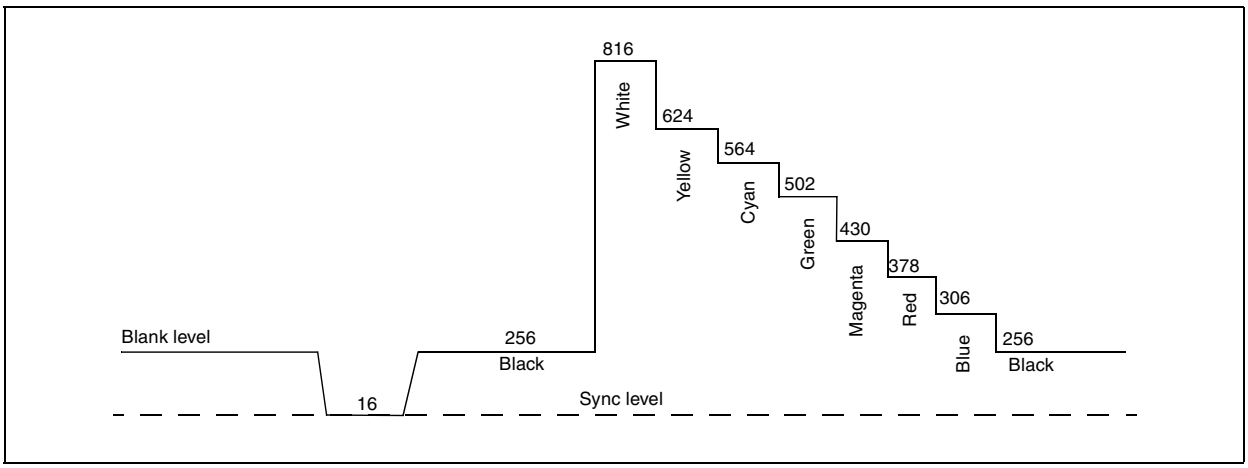


Figure 210: Luminance output levels in autotest for PAL (BGHI) and SECAM



55.6 Input demultiplexor

The incoming YCbCr data is demultiplexed into a blue-difference chroma information stream, a red-difference chroma information stream and a luma information stream. Incoming data bits are treated as blue, red or luma samples according to their relative position with respect to the sync signals in use and the contents of configuration bit SYNCIN_AD. Brightness, saturation and contrast are then performed on demultiplexed data (registers DENC_BRIGHT, DENC_CONTRAST and DENC_SATURATION).

The ITU-R601 recommendation defines the black luma level as Y = 16 and the maximum white luma level as Y = 235. Similarly, it defines 225 quantification levels for the color difference components (Cr, Cb), centered around 128. After the saturation, brightness and contrast stage, the incoming YCrCB samples can be saturated in the input multiplexer with the following rules:

For Cr or Cb samples:	Cr, Cb > 240	=> Cr, Cb saturated at 240
	Cr, Cb < 16	=> Cr, Cb saturated at 16
For Y samples:	Y > 235	=> Y saturated at 235
	Y < 16	=> Y saturated at 16

This avoids saturating the composite video codes heavily before digital-to-analog conversion in the case of erroneous or unrealistic YCbCr samples being input to the encoder. These could lead to overflow errors in the codes driving the DACs. In this way, a distorted output waveform is avoided.

However, in some applications, it may be desirable to let extreme YCbCr codes pass through the demultiplexor. This is controlled using bit MAXDYN in register DENC_CFG6. In this case, only codes 0x00 and 0xFF are overridden; if such codes are found in the active video samples, they are forced to 0x01 and 0xFE.

55.7 Subcarrier generation

A direct digital frequency synthesizer (DDFS) generates the required color subcarrier frequency using a 24-bit phase accumulator. This oscillator feeds a quadrature modulator that modulates the base-band chrominance components.

The subcarrier frequency is obtained from the following equation:

$$F_{SC} = (increment\_word / 2^{24}) \times PIXCLK$$

where *increment_word* is a 24-bit value.

Hard-wired *increment_word* values are available for each standard and can be automatically selected. Alternatively, using bit SELRST_INC in DENC_CFG5, the frequency can be fully customized by programming other values into a dedicated *increment_word* register, DENC_DFS_INC0/1/2. This allows, for instance, the encoding of NTSC-4.43 or PAL-M-4.43.

The procedure is as follows:

- Program the required increment in DENC_DFS_INC0/1/2.
- Set bit SELRST_INC to 1 in register DENC_CFG5.
- Perform a software reset using register DENC_CFG6. This sets all bits in all DENC registers except DENC_CFG*n* to their default value.
  Alternatively, set DENC_CFG8 bits PH_RST_MODE to 01. Then frequency (and phase) update is performed on the beginning of the next video line.

Warning: if a standard change occurs after the software reset, the increment value is automatically re-initialized with the hard-wired or loaded value according to bit SELRST_INC.

The reset phase of the color subcarrier can also be software-controlled by register DENC_DFS_PHASE0/1.

The subcarrier phase can be periodically reset to its nominal value to compensate for any drift introduced by the finite accuracy of the calculations. In PAL and NTSC, subcarrier phase can be adjusted every line, every eight fields, every four fields, or every two fields (DENC_CFG2 bits VALRST). If SECAM is performed, the subcarrier phase is reset every line.

55.8 Burst insertion (PAL and NTSC)

The color reference burst is inserted so as to always start with a positive zero crossing of the subcarrier sine wave. The first and last half-cycles have a reduced amplitude so that the burst envelope starts and ends smoothly.

The burst contains 9 or 10 sine cycles of 4.43361875 MHz or 3.579545 MHz (depending on the standard programmed in register DENC_CFG0) as follows:

NTSC-M	9	cycles of	3.579545	MHz
PAL-BDGHI	10	cycles of	4.43361875	MHz
PAL-M	9	cycles of	3.579545	MHz
PAL-N	9	cycles of	3.579545	MHz

The burst can be turned off (no burst insertion) by setting DENC_CFG2 bit BURSTEN to 0.

Burst insertion is performed by always starting the burst with a positive-going zero crossing. This guarantees a smooth start and end of burst with a maximum of undistorted burst cycles and can only be beneficial to chroma decoders.

This avoids an uncontrolled initial burst phase, and guarantees a start on a positive-going zero crossing with the consequence that two burst start locations are visible over successive lines, according to line parity. This is normal and explained below.

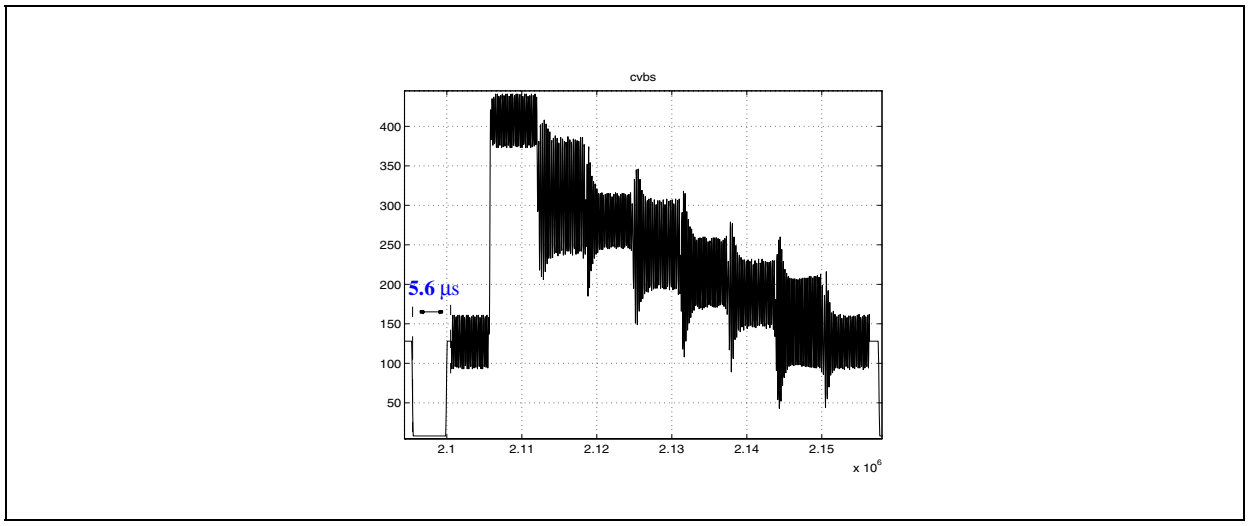
In NTSC, the relation between subcarrier frequency and line length creates a 180° subcarrier phase difference with respect to horizontal sync from one line to the next, according to line parity. So if the burst always starts with the same phase (positive-going zero crossing), the burst is inserted at time X or at time ($X + T_{NTSC}/2$) after the horizontal sync tip according to line parity, where T_{NTSC} is the duration of one cycle of the NTSC burst.

With PAL, a similar rationale holds, and again there are two possible burst start locations. The subcarrier phase difference with respect to the horizontal sync from one line to the next in this case is either 0° or 180° with the following series: A-A-B-B-A-A-... (A denotes A-type bursts and B denotes B-type bursts, A-type and B-type being 180° out of phase with respect to the horizontal sync). Two locations are thus possible, one for A-type, the other for B-type.

This assumes that the subcarrier is automatically reset periodically (VALRST in DENC_CFG2). Otherwise, the burst start drifts over several frames, within an interval of half a subcarrier's cycle. This is normal and means the burst is correctly locked onto the encoded colors. The equivalent effect with a gated burst approach would be the following: the start location would be fixed but the phase of the burst start with respect to the horizontal sync would drift.

55.9 Subcarrier insertion (SECAM)

Figure 211: SECAM color bar pattern (blue line)



Subcarrier frequency in SECAM mode depends on Cr and Cb values (frequency modulation). The color subcarrier frequency is 4 250 000 Hz for Cb = 128 (on blue lines) and 4 406 249 Hz for Cr = 128 (on red lines). Frequency clipping values are 3 900 000 Hz and 4 756 250 Hz.

The insertion point of the non-modulated subcarrier is shown in Figure 211.

In odd fields, the phase of the subcarrier follows the sequence: 0, 0, π, 0, 0, π, 0, 0, π, ... compared to a sine wave starting at the same point - 5.6 µs after horizontal sync pulse (inverted on one line out of every three and also at each frame). This sequence begins from line 1 or line 23 of the first field (see GEN_SECAM in register DENC_CFG7). Bit INV_PHI_SECAM (DENC_CFG7) allows the inversion of this sequence (π, π, 0, π, π, 0,...) in odd fields. In even fields, the sequence of the subcarrier is always inverted with respect to the odd field.

To enable SECAM mode, program a 1 in DENC_CFG7.SECAM (MSB) and then soft-reset or load DENC_CFG0.

55.10 Luminance encoding

The demultiplexed Y samples are band-limited and interpolated at PIXCLK clock rate. The resulting luminance signal is correctly scaled before insertion of any closed-captions, CGMS, VPS, Teletext or WSS data and synchronization pulses.

The interpolation filter compensates for the sin(*x*)/ *x* attenuation inherent in D/A conversion and greatly simplifies the output stage filter. See Figures 212, 213 and 214 for characteristic curves.

In addition, the luminance that is added to the chrominance to create the composite CVBS signal can be trap-filtered at 3.58 MHz (NTSC) or 4.43 MHz (PAL). This supports applications oriented towards low-end TV sets which are subject to cross-color if the digital source has a wide luminance bandwidth (for example, some DVD sources).

Note: The trap filter does not affect the S-VHS luminance output or the RGB outputs. For SECAM, enable the trap filter with 4.43 MHz cut-off frequency on the luma part of the CVBS signal (DENC_CFG3 bits ENTRAP and TRAP_4.43).

A 7.5 IRE pedestal can be programmed if needed with all standards (registers DENC_CFG1 and DENC_CFG7). This allows, in particular, encoding of Argentinian and non-Argentinian PAL-N, or Japanese NTSC (NTSC with no set-up).

The luma channel has a 19th order filter with coefficients programmable by registers DENC_LU_COEF0..9. This filter is described in Section 55.11: *Chrominance encoding on page 655*.

FLT_YS (DENC_CFG9) selects either the register or the hard-wired values for the filter coefficients.

The luma processing as well as line and field timings in SECAM mode are identical to PAL BDGHI ones.

Figure 212: Luma filtering including DAC attenuation

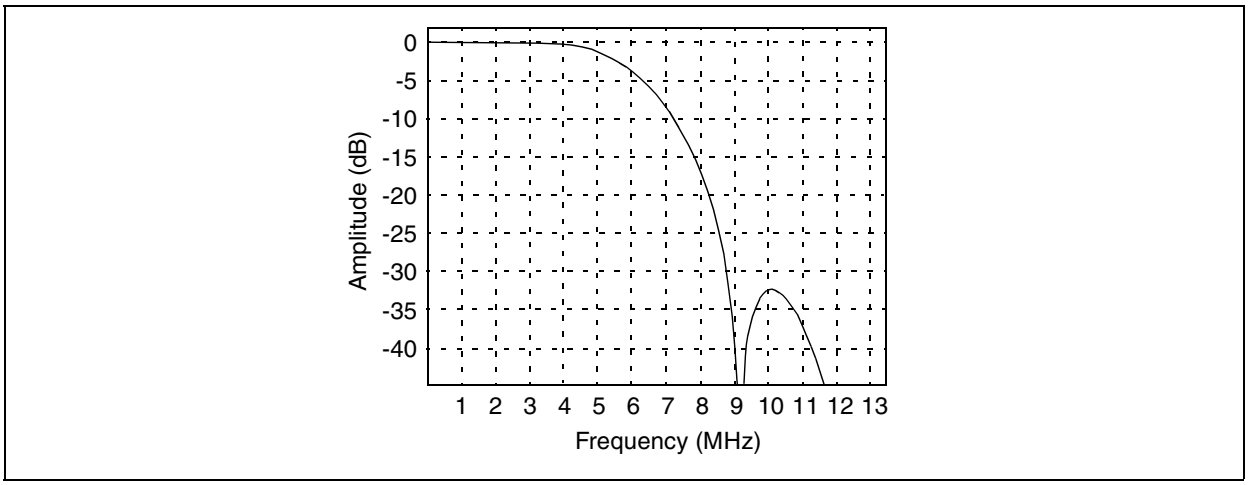


Figure 213: Luma filtering with 3.58 MHz trap, including DAC attenuation

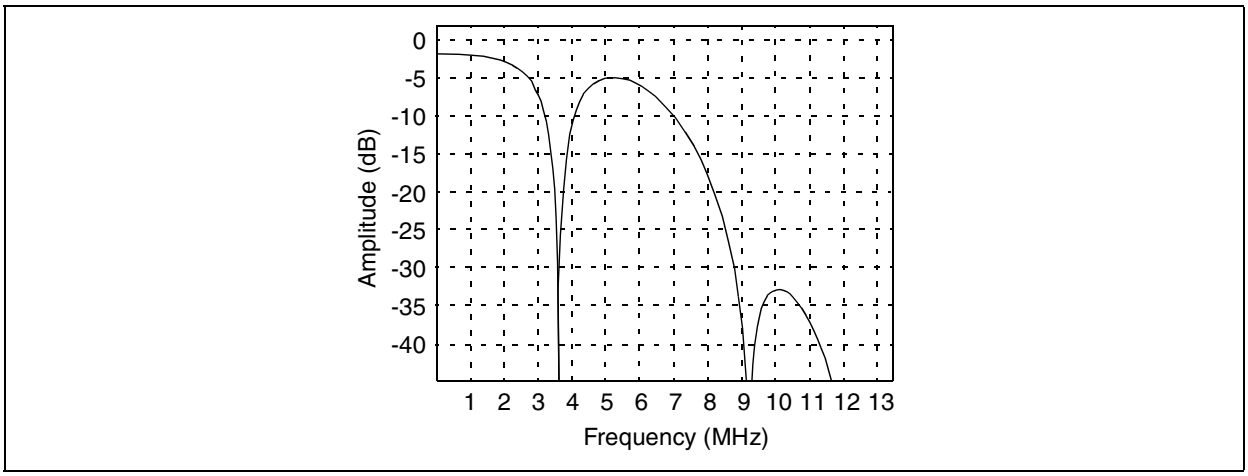
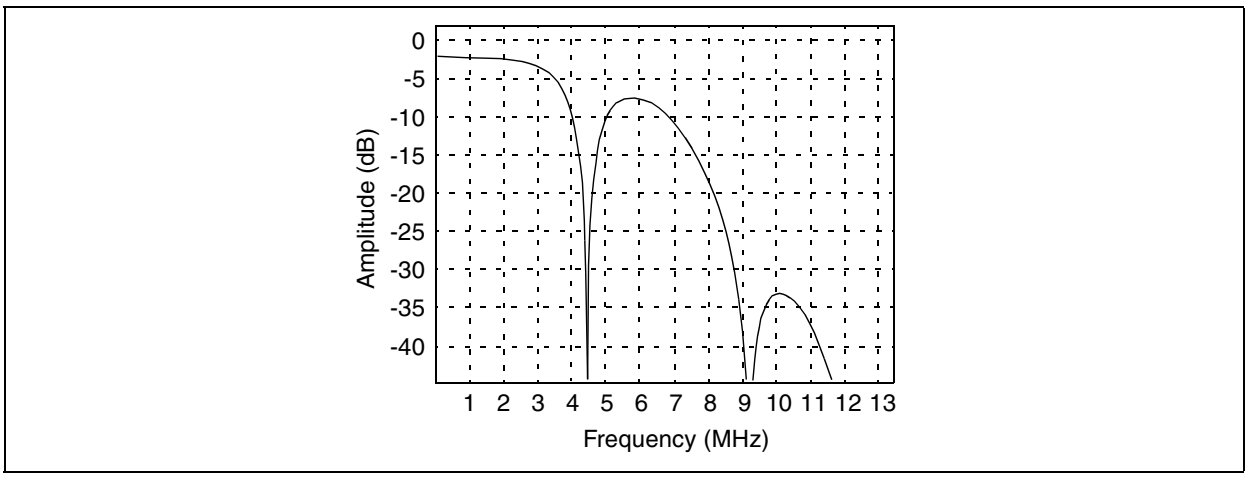


Figure 214: Luma filtering with 4.43 MHz trap, including DAC attenuation



55.11 Chrominance encoding

U, V (PAL and NTSC) and Dr, Db (SECAM) chroma components are computed from demultiplexed Cb, Cr samples. Before modulating the subcarrier, these are band-limited and interpolated at PIXCLK clock rate. This processing simplifies filtering following D/A conversion and allows more accurate encoding.

A set of four different filters is available in PAL and NTSC for chroma filtering to fit a wide variety of applications in the different standards, and include filters recommended by ITU-R 624-4 and SMPTE170-M.

The available -3 dB bandwidths are 1.1, 1.3, 1.6 or 1.9 MHz. See Figures 217, 218, 219, 220 and 221 for the various frequency responses and register DENC_CFG1 for programming. The narrower bandwidths are useful against cross-luminance artifacts, the wider bandwidths allow higher chroma contents.

Alternatively, a filter with programmable coefficients can be used (see registers DENC_CFG0 to DENC_CFG13). This is necessary when other clock frequencies are used (24.545454 and 29.5 MHz clocks in square pixel mode), or for specific applications where another frequency response is needed. The order of the chroma filter is 17, with symmetric coefficients, and it works at frequency PIXCLK (default 27 MHz). If the 4:2:2 input format is used on CVBS and S-VHS outputs, the first upsampling by 2 (6.75 to 13.5 MHz sampling frequency with 27 MHz PIXCLK) is implemented by repeating the samples (10 20 30... at 6.75 MHz give 10 10 20 20 30 30... at 13.5 MHz clock frequency). This is equivalent to filtering by $\cos(\pi \times f/(2 \times f_{CLK}))$, where f_{CLK} is the PIXCLK clock frequency. The second upsampling (13.5 to 27 MHz with 27 MHz PIXCLK) is implemented by padding with zeros.

Register bit DENC_CFCOEF0.FLT_S selects either the register or the hard-wired values for the filter coefficients.

In SECAM, 1.3 MHz low-pass and pre-emphasis filtering are performed on Dr and Db chroma components, before frequency modulation, according to ITU-R Rec624-4.

Refer to Figure 215 for the frequency response of these filters. Bell filtering is performed at the end of the frequency modulation stage.

Peak-to-peak amplitude of the modulated chrominance signal at the central frequency (4 279.7 kHz) is 22.88% of the black-white interval (22.88 IRE).

The chrominance path can be delayed, with respect to the luma path for S-VHS and CVBS outputs. DEL_EN (DENC_CFG3) selects either the default delays or the programmable delay (DENC_CFG1.DEL). The default delays are pre-programmed for the PAL, SECAM and NTSC modes in 4:2:2 and 4:4:4 format on CVBS.

Refer to Figure 216 for frequency response of the bell filter with subcarrier frequencies and clipping values.

Figure 215: SECAM chroma filtering (pre-emphasis and 1.3 MHz low pass filtering)

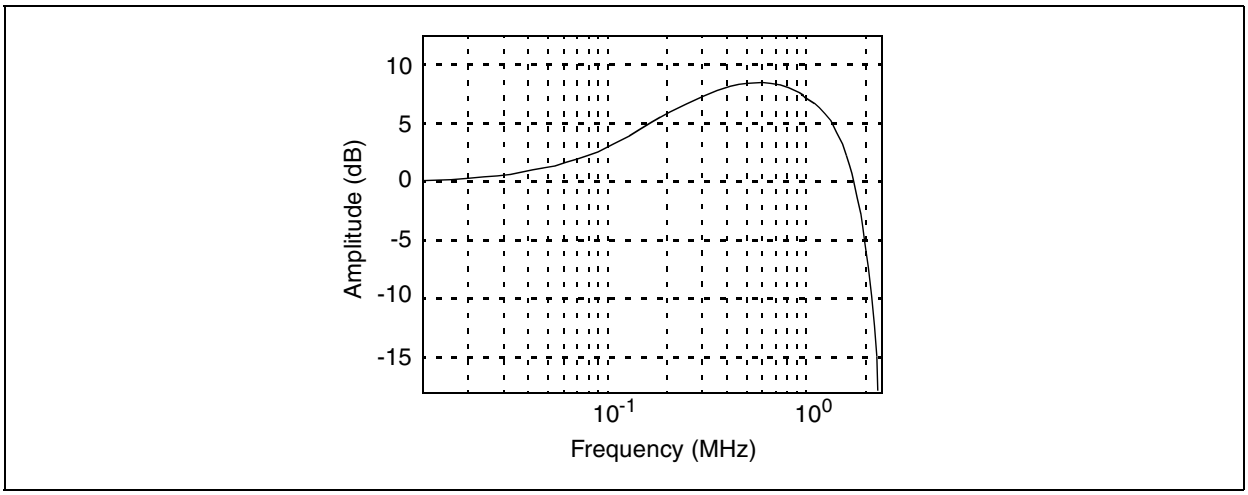


Figure 216: SECAM high-frequency subcarrier pre-emphasis (Bell filtering), including DAC attenuation

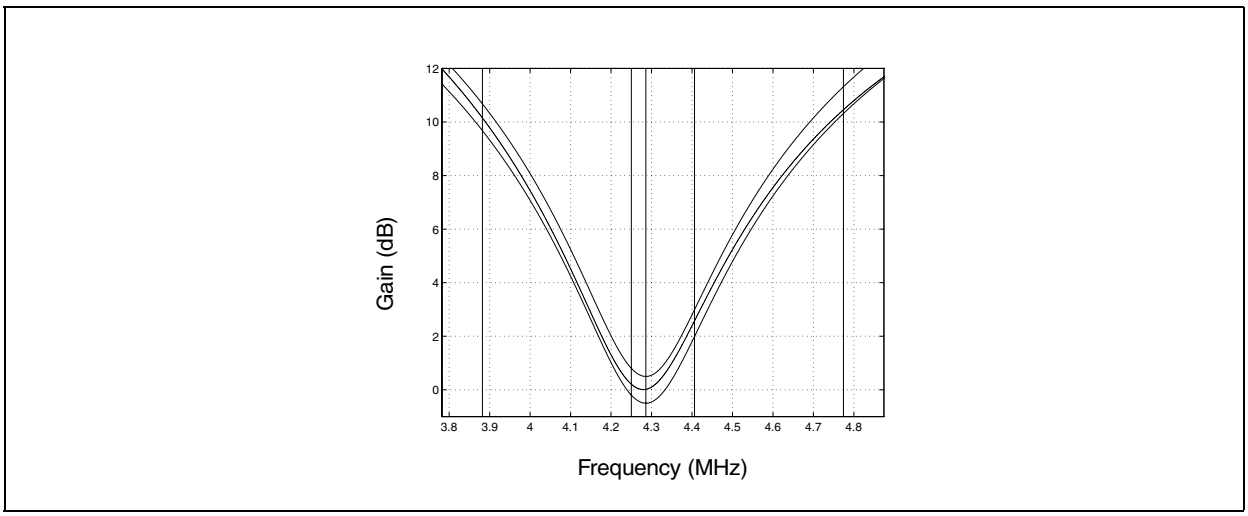
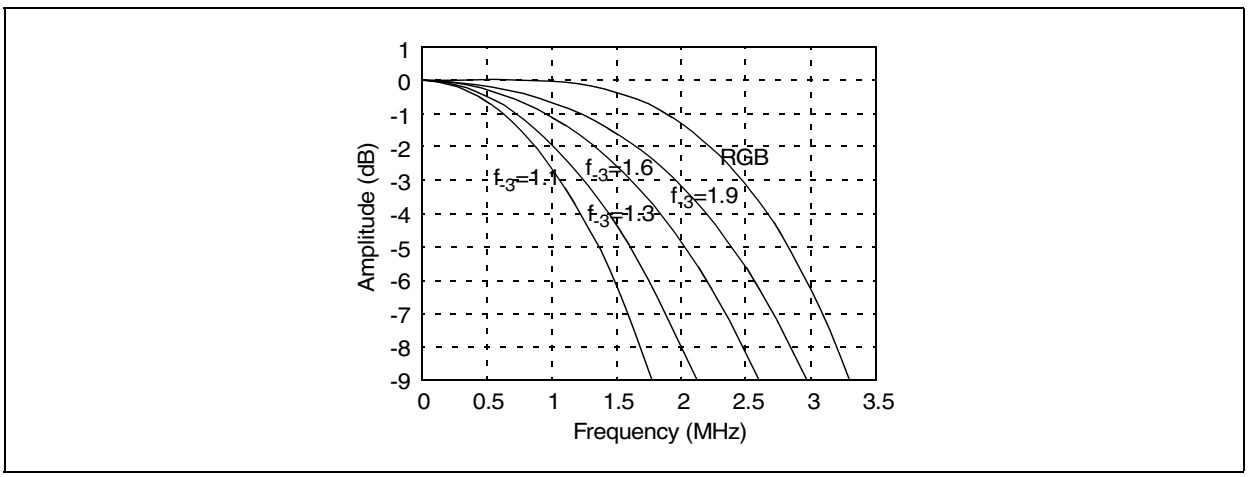


Figure 217: Various chroma filters available and RGB filter



55.12 Composite video signal generation

The composite video signal is created by adding the luminance (after trap filtering - optional in PAL and NTSC, see DENC_CFG3) and the chrominance components. A saturation function is included in the adder to avoid overflow errors should extreme luminance levels be modulated with highly saturated colors. This does not occur with natural colors but may be generated by computers or graphics engines.

A color killing function is available (DENC_CFG1.COKI), whereby the composite signal contains no chrominance, that is, replicates the trap-filtered luminance. This function does not suppress the chrominance on the S/VHS outputs, but suppressing the S-VHS chrominance is possible using bit BKDAC*n* in DENC_CFG5, where the chrominance signal is output on DAC *n*.

Figure 218: 1.1 MHz chroma filter

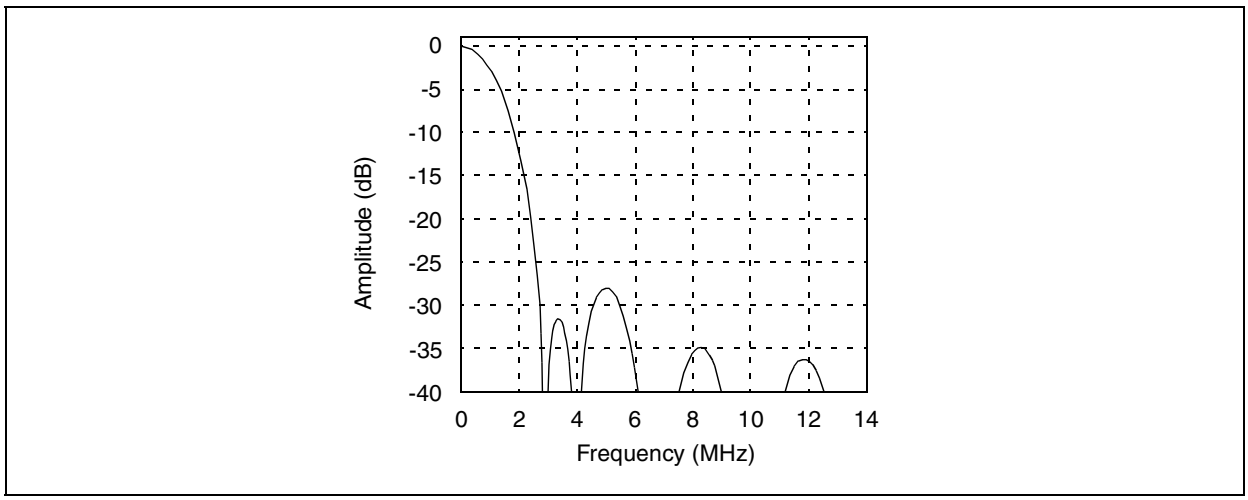


Figure 219: 1.3 MHz chroma filter

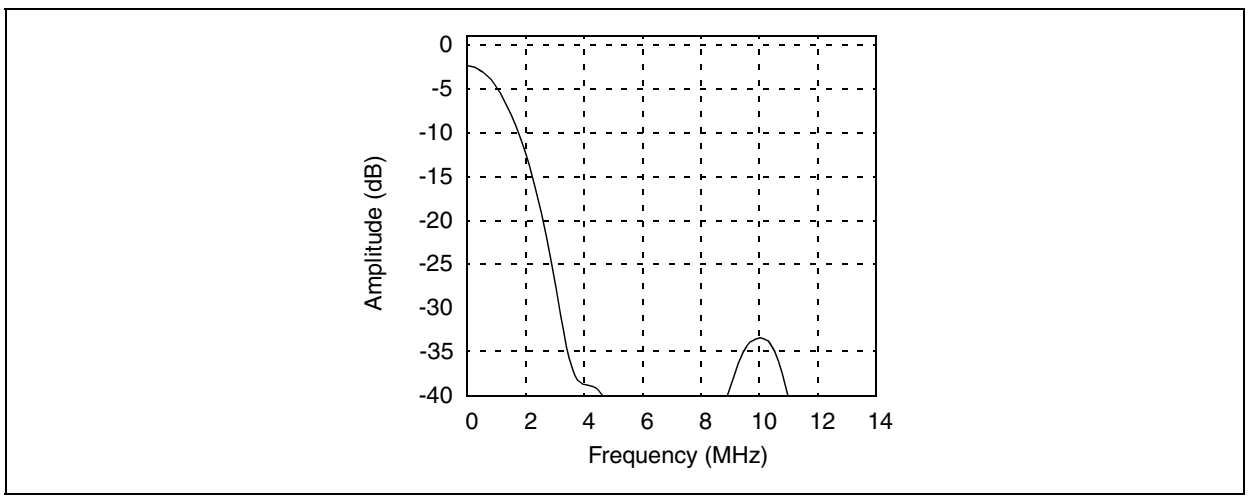


Figure 220: 1.6 MHz chroma filter

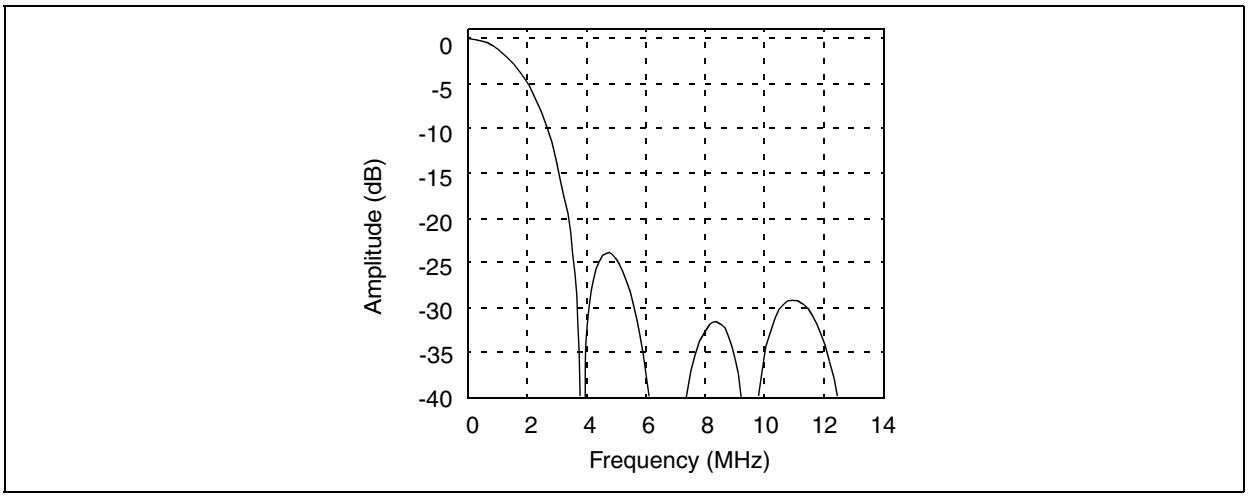
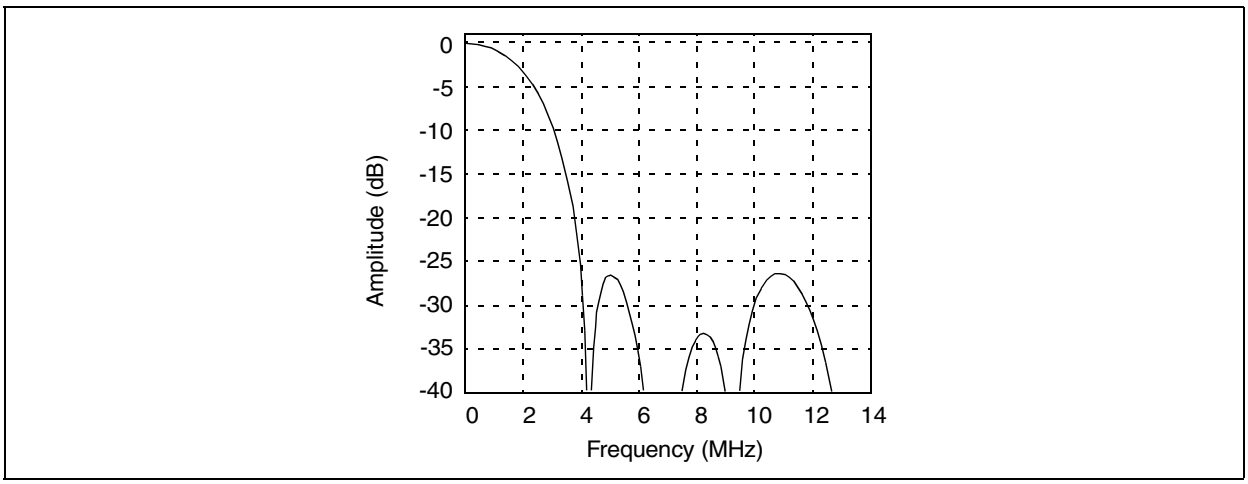


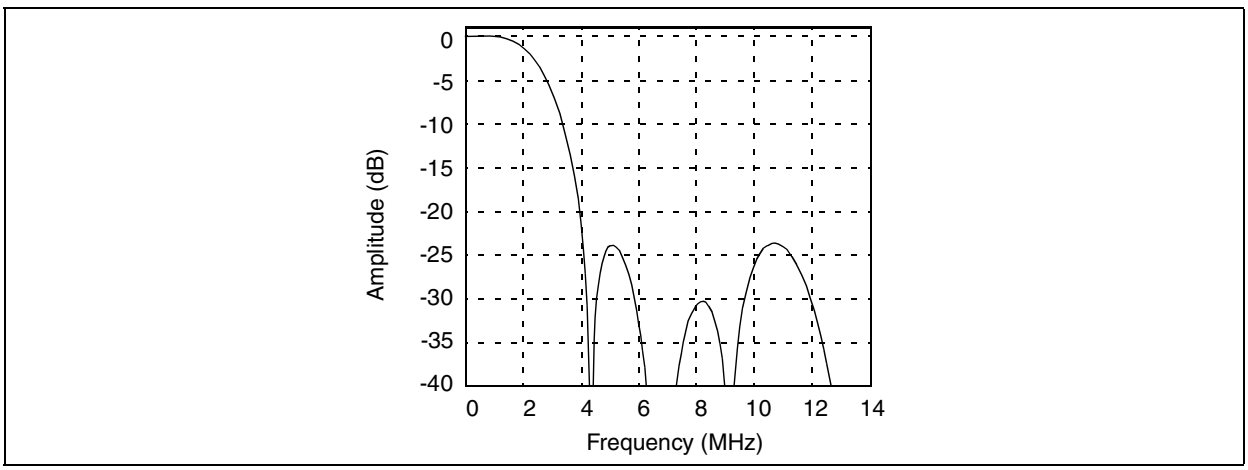
Figure 221: 1.9 MHz chroma filter



55.13 RGB and UV encoding

After demultiplexing, the Cr and Cb samples feed a four times interpolation filter. The resulting base-band chroma signal has a 2.45 MHz bandwidth (Figure 222) and is combined with the filtered luma component to generate R,G,B or U,V samples at 27 MHz.

Figure 222: RGB - chroma filtering



55.14 Closed captioning

Closed-captions (or data from an extended data service as defined by the closed-captions specification) can be encoded by the circuit. Closed-caption data is delivered to the circuit through the register interface. Two dedicated byte pairs (two bytes per field), each pair preceded by a clock run-in and start bit, can be encoded and inserted on the luminance path on a selected TV line. The clock run-in and start code are generated by the DENC.

Closed-caption data registers are double-buffered so that loading can be performed anytime, even during line 21/284 or any other selected line.

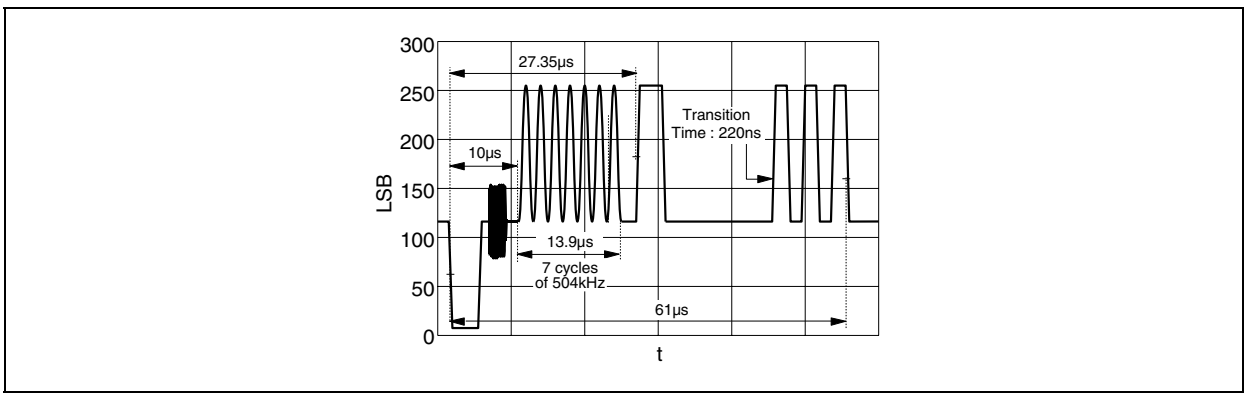
User registers DENC_CCCF1 and DENC_CCCF2 each contain the first and second byte to send (LSB first) after the start bit on the appropriate TV line, where DENC_CCCF1 refers to field 1 and DENC_CCCF2 to field 2. The TV line number where data is to be encoded is programmable using registers DENC_CCLIF1 and DENC_CCLIF2. Closed-caption data has priority over any CGMS signals programmed for the same line.

The internal clock run-in generator is based on a direct digital frequency synthesizer. The nominal instantaneous data rate is 503.496 kHz (that is, 32 times the NTSC line rate). Data low corresponds nominally to 0 IRE, data high corresponds to 50 IRE at the DAC outputs.

When closed-captioning is on (bits CC1 and CC2 in DENC_CFG1), the CPU should load the relevant registers (DENC_CCLIF1 or DENC_CCLIF2) once every frame at most (although there is in fact some margin due to the double-buffering). Two bits are set in register DENC_STA in case of attempts to load the closed-caption data registers too frequently; these can be used to regulate the loading rate.

The closed-caption encoder considers that closed-caption data has been loaded and is valid on completion of the write operation into DENC_CCCF1 for field1, or DENC_CCCF2 for field 2. If closed-caption encoding has been enabled and no new data bytes have been written into the closed-caption data registers when the closed-caption window starts on the appropriate TV line, then the circuit outputs two US-ASCII NULL characters with odd parity after the start bit.

Figure 223: Example of closed-caption waveform

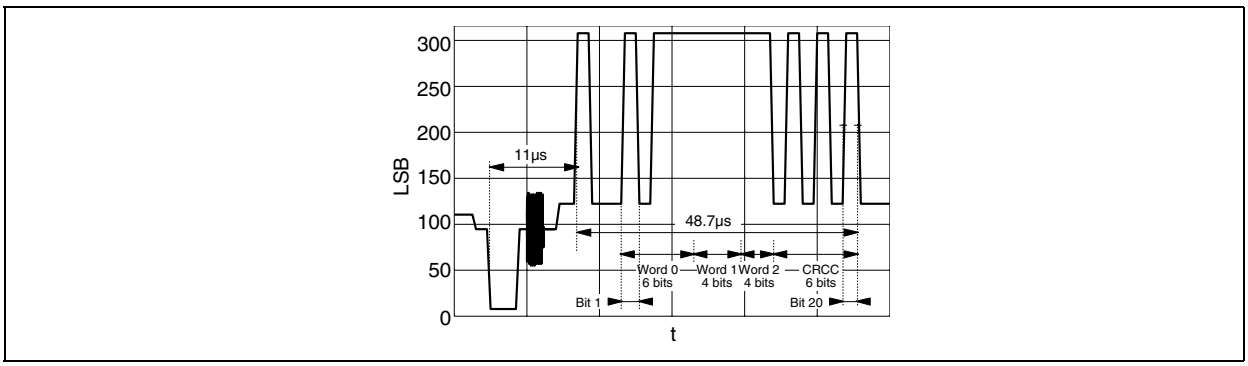


55.15 CGMS encoding

CGMS (copy generation management system, also known as VBID) is described by standard CPX-1204 of EIAJ. CGMS data can be encoded by the digital encoder.

Three bytes, containing 20 significant bits, are delivered to the chip via the register interface. Two reference bits (1 then 0) are encoded first, followed by 20 bits of CGMS data. This includes a cyclic redundancy check sequence, that is not computed by the device but is supplied as part of the 20 data bits. The reference bits are generated locally by the DENC. Figure 224 shows a typical CGMS waveform.

Figure 224: Example of CGMS waveform



CGMS encoding is enabled by setting bit ENCGMS in register DENC_CFG3. When enabled, the CGMS waveform is present once in each field, on lines 20 and 283 (SMPTE-525 line numbering).

The CGMS data register is double-buffered, so it can be loaded at any time (even during line 20/283) without any risk of corrupting CGMS data that could be in the process of being encoded. The CGMS encoder considers that new CGMS data has been loaded and is valid on completion of the write operation into register DENC_CGMS0/1/2.

55.16 WSS encoding

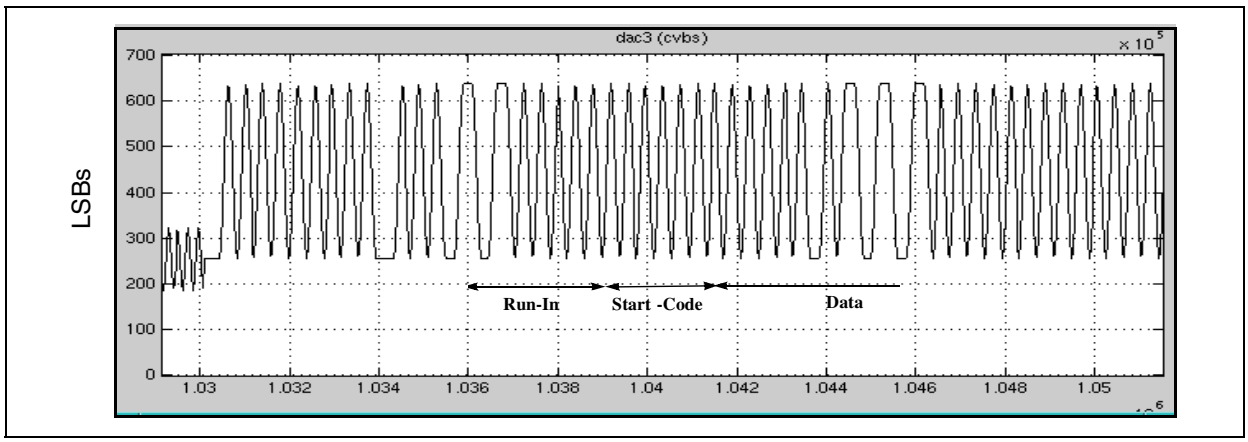
The digital encoder allows WSS (wide screen signalling) in 625-line format, complying with the ETS 300 294 standard. Two bytes are delivered to the circuit through the register interface into two dedicated registers (register DEN_WSSn).

WSS encoding is enabled using bit ENWSS in register DENC_CFG3. When WSS encoding is enabled, a waveform is present on the first half of line 23 of each frame. Data is preceded by a run-in sequence and a start code generated locally by the DENC.

55.17 VPS encoding

VPS data encoding is defined by *ETS 300 231 communication, June 1993*. VPS data can be encoded by the DENC on line 16 (CCIR) for 625-line PAL and SECAM television systems. The VPS data is delivered to the circuit using register DENC_VPS0. Data transmission is preceded by a clock run-in and a start code generated by the DENC. The clock frequency is 5 MHz. This feature is enabled by setting bit ENVPS of register DENC_CFG7. Figure 225 shows an example of a VPS waveform.

Figure 225: Example of VPS waveform



55.18 Teletext encoding

The DENC can encode Teletext according to the *CCIR/ITU-R Broadcast Teletext System B* specification (also known as World System Teletext), and NABTS (North American Basic Teletext Specification) EIA-516.

In DVB applications, Teletext data is embedded within DVB streams as MPEG data packets. The transport layer processing IC (ST40) sorts incoming data packets and stores Teletext packets in a buffer. It then passes them to the DENC on request.

Teletext is enabled by TTX_EN (DENC_CFG6). Teletext data priority over other VBI data must also be programmed using TTXNOTMV (DENC_CFG8).

Signal exchange

The DENC and the Teletext buffer exchange two signals: TTXS (Teletext synchronization) from the DENC to the Teletext buffer, and TTXD (Teletext data) from the Teletext buffer to the DENC.

Signal TTXS is a request signal generated on selected lines. In response to this signal, the Teletext buffer is expected to send Teletext bits to the DENC for insertion of a Teletext line into the analog video signal. The number of Teletext bits sent depends on the Teletext system being used (selected by register bit DENC_TTX1.TTXT_ABCD); 360 bits are sent for Teletext B - WST in PAL and SECAM, or 288 for Teletext C - NABTS in NTSC.

The duration of the TTXS window corresponds to the number of bits being sent (see Transmission protocol below).

- For Teletext B and 625 line systems, the TTXS window duration is 1402 reference clock periods (corresponding to 360 bits).
- For Teletext C and 525 line systems (NABTS), this duration is 1121 master clock periods.

Following the TTXS rising edge, the encoder expects data from the Teletext buffer after a programmable number (2 to 9) of 27 MHz master clock periods. Data is transmitted synchronously with the master clock at an average rate of 6.9375 Mbit/s according to the protocol described below. In order of transmission, it consists of: 16 clock run-in bits, 8 framing code bits and one Teletext packet of 336 or 228 bits (depending on the Teletext system being used). If more than one packet of bits (336 or 228) are transmitted, they are ignored by the DENC. By default, register bit DENC_CM_TTX.TTX_MASK_OFF masks the two bits of Teletext framing code, allowing the code to be set by the DENC according to the selected Teletext standard.

Transmission protocol

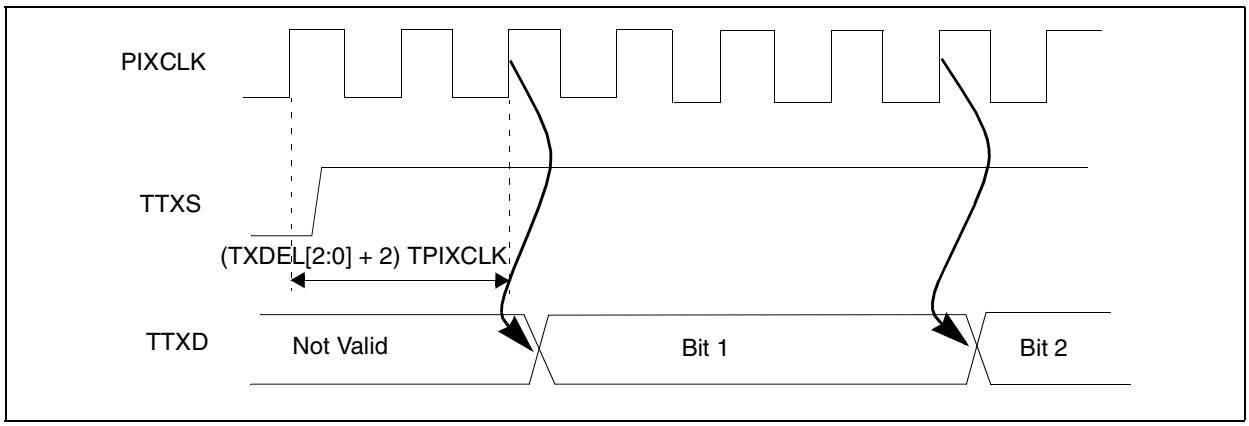
In order to transmit the Teletext data bits at an average rate of 6.9375 Mbit/s, which is about 1/3.89 times the master clock frequency, the following scheme is adopted:

The 360-bit packet is regarded as nine 37-bit sequences plus one 27-bit sequence. In every sequence, each Teletext data bit is transmitted as a succession of **four** identical samples at 27 Msample/s, except for the 10th, 19th, 28th and 37th bits of the sequence which are transmitted as a succession of **three** identical samples.

Programming TTXS rising to first valid sample

The encoder expects the Teletext buffer to clock-out the first Teletext data sample on the $(2+N)^{th}$ rising edge of the master clock following the rising edge of TTXS. Figure 226 depicts this graphically for $N = 0$.

Figure 226: TTXS rising to first valid sample delay for TXDEL[2:0] = 0



N is programmable from 0 to 7 by register bits DENC_TTX1.TTXDEL[2:0]. The value written in TXDEL[2:0] is two less than the overall delay in PIXCLK cycles, so a value of 0 for TXDEL[2:0] corresponds to an overall delay of 2 cycles, and a value of 7 corresponds to a delay of 9 cycles.

Programming teletext line selection

Five dedicated registers, DENC_TTX2-5, program Teletext encoding in various lines in the vertical blanking interval (VBI) of each field. In this way, each line in VBI can be selected independently.

Full-page teletext encoding is set by FP_TTXT (DENC_TTX1). Teletext is encoded on lines 24 to 311 and 336 to 623 (ITU-R line numbering). This is in addition to the lines already programmed in the VBI. When full-page teletext is performed, no video data is encoded (YCbCr input streams are ignored).

Teletext pulse shape

The shape and amplitude of a single Teletext pulse is shown in Figure 227. Its relative power spectral density is shown in Figure 228 and Figure 229 It is zero at frequencies above 5 MHz, as required by the World System Teletext specification.

Figure 227: Shape and amplitude of a single teletext symbol

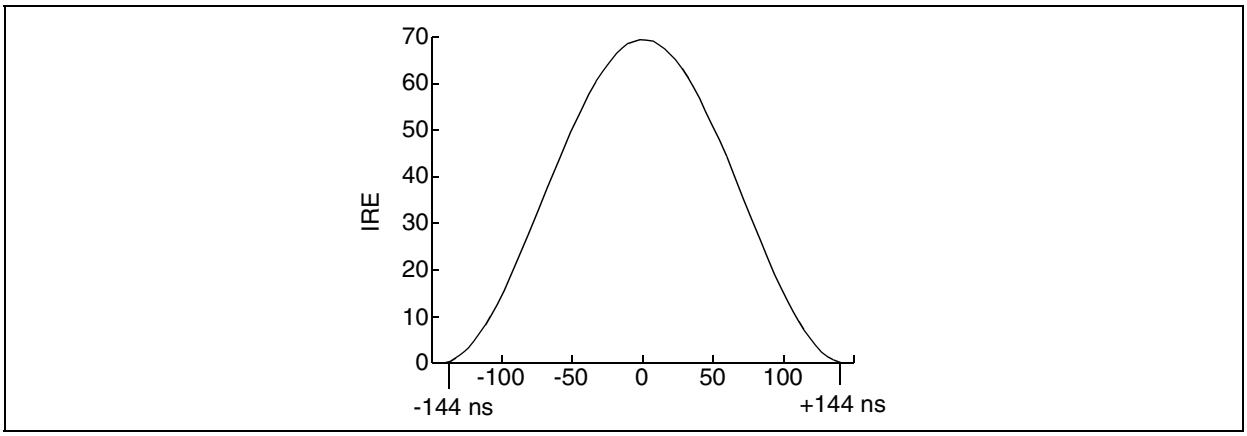


Figure 228: Linear PSD scale

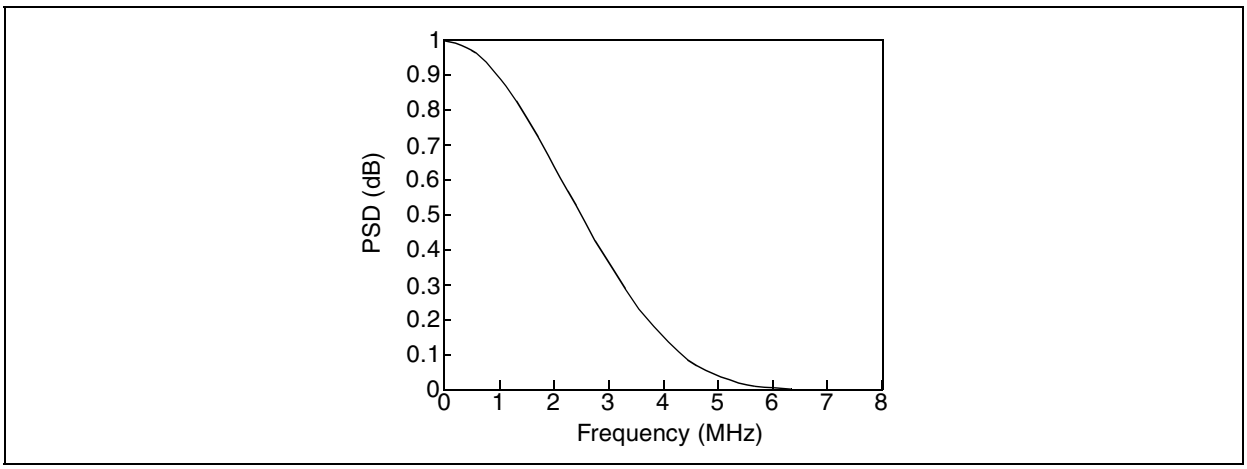
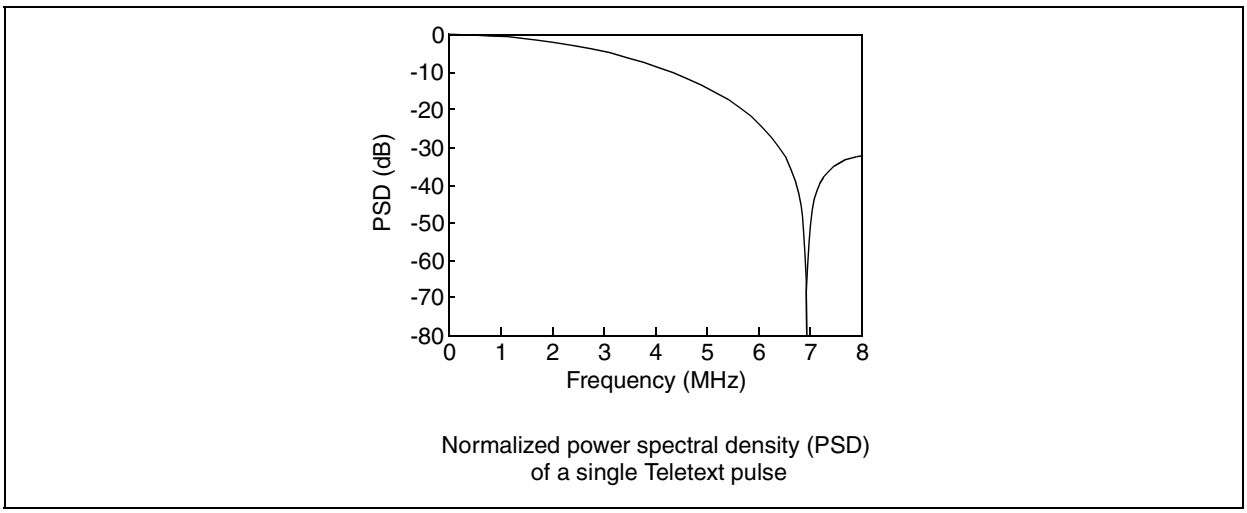


Figure 229: Logarithmic PSD scale



Normalized power spectral density (PSD)
of a single Teletext pulse

55.19 CVBS, S-VHS, RGB and UV outputs

Three out of eight video signals can be directed to three analog output pins through a 10-bit DAC operating at the reference clock frequency. The available combinations are:

S-VHS (Y/C) + CVBS, or U + Y + V, or RGB.

These combinations are controlled by bits DAC123_CFG[2:0] and DAC456_CFG[2:0] (DENC_CFG13).

The C to Y peak-to-peak amplitude ratio can be modified in both CVBS and VHS (Y/C) outputs using C_MULT (DENC_CM_TTX).

Default peak-to-peak amplitude of UV and RGB outputs is set to 70% of Y or CVBS peak-to-peak amplitude, for 100/0/100/0 color bar pattern, and can be modified using the multiplying factors in registers DENC_DAC34_MULT, DENC_DAC45_MULTand DENC_DAC6_MULT.

If DENC_CFG7 bit UV_LEV is 0 (default value), U and V outputs have 0.7 V peak-to-peak amplitude if a 100/0/100/0 color bar pattern is input. If this bit is 1, U and V outputs are those defined by ITU-R 624-4 for PAL and NTSC standards (V_{pp}/U_{pp} = 1.4). In this case, U peak-to-peak amplitude is 0.61 V (0.57 V if DENC_CFG7 bit SETUPYUV is set) and V peak-to-peak amplitude is 0.86 V (0.80 V if SETUPYUV is set). In all these cases, UV outputs can be multiplied by a factor of 0.75 to 1.22 depending on the value of DAC4_MULT (DENC_DAC34_MULT,DENC_DAC45_MULT) and DAC6_MULT (DENC_DAC6_MULT).

Any unused DAC may be independently disabled by software. In this case, its output is at neutral level (blanking for luma and composite outputs, no color for chroma output, black for RGB and UV outputs).

Due to the 3.3 V power supply used, the output swing of the DACs is about 1 V_{pp}. Therefore some external gain may be required, which, combined with the recommended output filtering stage, requires active filtering. For this active filtering stage to be very simple, it is possible to invert the DAC outputs by programming bit DACINV (DENC_CFG5). Code *N* becomes code (1024 - *N*), that is, the resulting waveform undergoes a symmetry around the mid-swing code.

56 Digital encoder (DENC) registers

Register addresses are provided as *DencBaseAddress* + offset.

The *DencBaseAddress* is:

0x1920 C000.

Each register is mapped into the least significant byte of a 32-bit register space. Undefined locations within the register space are reserved.

Where Teletext is not implemented, registers related to Teletext should not be used and bit TTX_EN (DENC_CFG6) must be left at the reset value of zero.

Table 180 summarizes the digital encoder registers by function.

Table 180: Digital encoder registers summarized by function

Function	Register	Offset	Type
Configuration and status	DENC_CFG0	0x00	R/W
	DENC_CFG1	0x04	R/W
	DENC_CFG2	0x08	R/W
	DENC_CFG3	0x0C	R/W
	DENC_CFG4	0x10	R/W
	DENC_CFG5	0x14	R/W
	DENC_CFG6	0x18	R/W
	DENC_CFG7	0x1C	R/W
	DENC_CFG8	0x20	R/W
	DENC_CFG9	0x144	R/W
	DENC_CFG10	0x170	R/W
	DENC_CFG11	0x174	R/W
	DENC_CFG12	0x178	R/W
	DENC_CFG13	0x17C	R/W
	DENC_STA	0x024	RO
Digital frequency synthesizer	DENC_DFS_INC0	0x028	R/W
	DENC_DFS_INC1	0x02C	R/W
	DENC_DFS_INC2	0x030	R/W
	DENC_DFS_PHASE0	0x034	R/W
	DENC_DFS_PHASE1	0x038	R/W
WSS	DENC_WSS_DATA1	0x03C	R/W
	DENC_WSS_DATA2	0x040	R/W
DAC inputs	DENC_DAC13_MULT	0x044	R/W
	DENC_DAC34_MULT	0x048	R/W
	DENC_DAC45_MULT	0x04C	R/W
	DENC_DAC2_MULT	0x1A8	R/W
	DENC_DAC6_MULT	0x1AC	R/W
Chip ID	DENC_HW_VER	0x060	RO
VPS data	DENC_VPS0	0x064	R/W
	DENC_VPS1	0x068	R/W
	DENC_VPS2	0x06C	R/W
	DENC_VPS3	0x070	R/W
	DENC_VPS4	0x074	R/W
	DENC_VPS5	0x078	R/W

Table 180: Digital encoder registers summarized by function

Function	Register	Offset	Type
CGMS data	DENC_CGMS0	0x07C	R/W
	DENC_CGMS1	0x080	R/W
	DENC_CGMS2	0x084	R/W
Teletext	DENC_TTX1	0x088	R/W
	DENC_TTX2	0x08C	R/W
	DENC_TTX3	0x090	R/W
	DENC_TTX4	0x094	R/W
	DENC_TTX5	0x098	R/W
	DENC_TTX_CFG	0x100	R/W
	DENC_CM_TTX	0x104	R/W
Closed caption	DENC_CCCF10	0x09C	R/W
	DENC_CCCF11	0x0A0	R/W
	DENC_CCCF20	0x0A4	R/W
	DENC_CCCF21	0x0A8	R/W
	DENC_CCLIF1	0x0AC	R/W
	DENC_CCLIF2	0x0B0	R/W
Input demultiplexor	DENC_BRIGHT	0x114	R/W
	DENC_CONTRAST	0x118	R/W
	DENC_SATURATION	0x11C	R/W
Chroma coefficient	DENC_CF_COEF0	0x120	R/W
	DENC_CF_COEF1	0x124	R/W
	DENC_CF_COEF2	0x128	R/W
	DENC_CF_COEF3	0x12C	R/W
	DENC_CF_COEF4	0x130	R/W
	DENC_CF_COEF5	0x134	R/W
	DENC_CF_COEF6	0x138	R/W
	DENC_CF_COEF7	0x13C	R/W
	DENC_CF_COEF8	0x140	R/W
Luma coefficient	DENC_LU_COEF0	0x148	R/W
	DENC_LU_COEF1	0x14C	R/W
	DENC_LU_COEF2	0x150	R/W
	DENC_LU_COEF3	0x154	R/W
	DENC_LU_COEF4	0x158	R/W
	DENC_LU_COEF5	0x15C	R/W
	DENC_LU_COEF6	0x160	R/W
	DENC_LU_COEF7	0x164	R/W
	DENC_LU_COEF8	0x168	R/W
	DENC_LU_COEF9	0x16C	R/W
Hue control	DENC_HUE	0x1A4	R/W

56.1 Configuration and status

DENC_CFG0 Configuration 0

7	6	5	4	3	2	1	0
STD		SYNC			POLH	POLV	FREERUN

Address: *DencBaseAddress* + 0x000
Type: R/W
Reset: 0x92
Description:

[7:6] **STD**: Video signal standard
00: PAL BDGHI
01: PAL N
10: NTSC M
11: PAL M
The standard on hardware reset is NTSC. Any standard modification automatically selects the right parameters for correct subcarrier generation. If bit SECAM from register DENC_CFG7 is set, STD1 and STD0 are not taken into account.

[5:3] **SYNC**: Configuration.
000: OddEven-only based slave mode (frame locked)
001: F based slave mode (frame locked)
010: OddEven + HSync based slave mode (line locked)
011 - 110: Reserved
111: Autotest mode (color bar pattern)

[2] **POLH**: Horizontal sync
Active edge of HSync selection (when input) or polarity of HSync (when output)
0: HSync is a negative pulse (128 TPIX_CLK wide) or falling edge is active
1: HSync is a positive pulse (128 TPIX_CLK wide) or rising edge is active

[1] **POLV**: Vertical sync
Active edge of OddEven selection.
0: Falling edge of OddEven flags start of field1 (odd field) or Vsync is active low
1: Rising edge of OddEven flags start of field1 (odd field) or Vsync is active high

[0] **FREERUN**: Sync
Active edge of OddEven. This bit is taken into account in OddEven-only and F-based slave modes. It is irrelevant to other sync modes.
0: Disabled
1: Enabled

DENC_CFG1 Configuration 1

7	6	5	4	3	2	1	0
BLKLI	FLT		SYNCOK	COKI	SETUP	CC	

Address: *DencBaseAddress* + 0x004

Type: R/W

Reset: 0x44

Description:

[31:8] **Reserved**

[7] **BLKLI**: Vertical blanking interval selection for active video lines area[a]
0: (partial blanking) Only the following lines inside the vertical interval are blanked:
NTSC-M: lines [1 to 9], [263(half) to 272] (525-SMPTE), PAL-M: lines [523 to 6], [260(half) to 269] (525-CCIR)
other PAL: lines [623(half) to 5], [311 to 318] (625-CCIR)
This mode preserves embedded VBI data within the incoming YCbCr, for example, teletext (lines [7 to 22] and [320 to 335]), Wide screen signalling (full line 23), video programing service (line16)and so on).
1: (full blanking) All lines inside VBI are blanked
NTSC-M: lines [1 to 19], [263(half) to 282] (525-SMPTE), PAL-M: lines [523 to 16], [260(half) to 279] (525-CCIR)
other PAL: lines [623(half) to 22], [311 to 335] (625-CCIR)

[6:5] **FLT**: U/V Chroma filter bandwidth selection
Typical applications for 3 dB bandwidth are given for each FLT bit configuration.
00: f_{-3dB} = 1.1 MHz low definition NTSC filter
01: f_{-3dB} = 1.3 MHz low definition PAL filter
10: f_{-3dB} = 1.6 MHz HD NTSC filter (ATSC compliant) and PAL M/N (ITU-R 624.4 compliant)
11: f_{-3dB} = 1.9 MHz HD PAL filter: Rec 624 - 4 for PAL BDG/I compliant

[4] **SYNCOK**: sync signal availability (analog and digital) for input synchronization loss with FREERUN inactive (0)
0: No sync output signals
1: Output syncs available on YS, CVBS and, when applicable, Hsync (if output port), OddEven (if output port), that is, same behavior as FREERUN except that video outputs are blanked in the active portion of the line

[3] **COKI**: Color killer
0: Color on
1: Color suppressed on CVBS output signal (CVBS=YS) but color still present on C output
For color suppression on chroma DAC ‘C, see register DENC_CFG5.BKDAC2.

[2] **SETUP**: Pedestal enable
0: Blanking level and black level are identical on all lines (example: Arg. PAL-N, Japan NTSC-M, PAL-BDGHI)
1: Black level is 7.5 IRE above blanking level on all lines outside VBI (ex: Paraguayan and Uruguayan PAL-N). In all cases, gain factor is adjusted to obtain the required chrominance levels.

[1:0] **CC**
00: No closed caption/extended data encoding
01: Closed caption/extended data encoding enabled in field 1 (odd)
10: Closed caption/extended data encoding enabled in field 2 (even)
11: Closed caption/extended data encoding enabled in both fields

a. BLKLI must be set to 0 when closed captions are to be encoded on the following lines:
in 525/60 system: before line 20(SMPTE) or before line 283(SMPTE)
in 625/50 system: before line 23(CCIR) or before line 336(CCIR)

DENC_CFG2 Configuration 2

7	6	5	4	3	2	1	0
NINTRL	RST_EN	BURST_EN	Reserved	SEL_RST	RSTOSC_BUF	VAL_RST	

Address: *DencBaseAddress* + 0x008

Type: R/W

Reset: 0x20

Description:

[31:8] **Reserved**

[7] **NINTRL**: Non-interlaced mode select[a]
0: Interlaced mode (625/50 or 525/60 system)
1: Non-interlaced mode (2 x 312/50 or 2 x 262/60 system)

[6] **RST_EN**: Cyclic update of DDFS phase
0: No cyclic subcarrier phase reset
1: Cyclic subcarrier phase reset depends on VAL_RST

[5] **BURST_EN**: Chrominance burst control
0: Burst is turned off on CVBS, chrominance output is not affected
1: Burst is enabled

[4] **Reserved**
Do not change from reset value.

[3] **SEL_RST**: Select reset values for direct digital frequency synthesizer
0: Hardware reset values for subcarrier oscillator phase (see register DENC_DFS_PHASE0/1 for values)
1: Loaded reset values selected (see registers DENC_DFS_PHASE0/1)

[2] **RSTOSC_BUF**: Software phase reset of DDFS (direct digital frequency synthesizer) buffer[b]
0: No reset
1: When a 0-to-1 transition occurs, either the hard-wired default phase value, or the value loaded in register DENC_DFS_PHASE0/1 (according to bit SELRST), is put to phase buffer. This value is loaded into accumulator (phase of subcarrier) when PH_RST_MODE from DENC_CFG8 is programmed, or when the standard changes or soft reset occurs.
RSTOSC_BUF is set back to 0 after the buffer is loaded.

[1:0] **VAL_RST:** Automatic reset of the oscillator[c]
00: Every line
01: Every 2nd field
10: Every 4th field
11: Every 8th field

a. NINTRL update is internally taken into account at the beginning of the next frame. In SECAM mode, only the interlaced mode is available.

b. RSTOSC_BUF is automatically set back to 0 after the buffer is loaded

c. VAL_RST is taken into account only if bit RST_EN is set. Resetting the oscillator means here forcing the value of the phase accumulator to its nominal value to avoid accumulating errors due to the finite number of bits used internally. The value to which the accumulator is reset is either the hard-wired default phase value or the value loaded in registers DENC_DFS_PHASE0/1 (according to bit SEL_RST), to which a 0°, 90°, 180°, or 270° correction is applied according to the field and line on which the reset is performed. If SECAM is performed, the oscillator is reset every line.

DENC_CFG3 Configuration 3

7	6	5	4	3	2	1	0
MAIN_TRAP_EN	TRAP_443	CGMS_EN	Reserved	MAIN_DEL_EN	Reserved		WSS_EN

Address: *DencBaseAddress* + 0x00C

Type: R/W

Reset: 0x00

Description:

[31:8] **Reserved**

[7] **MAIN_TRAP_EN**: Enable trap filter[a]
0: Trap filter disabled 1: Trap filter enabled

[6] **TRAP_443**: Trap filter frequency
TRAP_443 is taken into account only for main if bit MAIN_TRAP_EN is set. No independent selection can be made on main and auxiliary paths.
0: Select the trap filter centered around 3.58 MHz 1: select the trap filter centered around 4.43 MHz
See Section 55.10: *Luminance encoding on page 653*

[5] **CGMS_EN**[b]: CGMS encoding enable
0: Disabled 1: Enabled

[4] **Reserved**

[3] **MAIN_DEL_EN**: Enable of chroma to luma delay programming on the main 4:4:4 input
0: Disabled (digital encoder automatically sets this delay)
1: Enabled (chroma to luma delay is programmed by DEL[3:0] bits from DEN_CFG9.

[2:1] **Reserved**: Reset value = 0

[0] **WSS_EN**: WSS encoding enable
0: Disabled 1: Enabled

a. When SECAM is performed, trap filter must be enabled (set ENTRAP = 1).

b. When CGMS_EN is set to 1, closed-caption/extended-data services should not be programmed on lines 20 and 283 (525/60, SMPTE line number convention).

DENC_CFG4 Configuration 4

7	6	5	4	3	2	1	0
SYNCIN_AD		Reserved		ALINE	DEL4		

Address: *DencBaseAddress* + 0x010
Type: R/W
Reset: 0x00
Description:

[31:8] **Reserved**

[7:6] **SYNCIN_AD**: Adjustment of incoming sync signals
Used to ensure correct interpretation of incoming video samples such as Y, Cr or Cb when the encoder is slaved to incoming sync signals.
00: Nominal 01: +1 PIXCLK
10: +2 PIXCLK 11: +3 PIXCLK

[5:4] **Reserved**
Do not change from reset values.

[3] **ALINE**: Video active line duration control
0: Full digital video line encoding (720 pixels - 1440 clock cycles)
1: Active line duration follows ITU-R/SMPTE analog-standard requirements

[2:0] **DEL4**: Delay on luma path w.r.t. chroma path on YUV and RGB outputs when 4:4:4 input is used
010: + 2 pixel delay on luma 001: + 1 pixel delay on luma
000: + 0 pixel delay on luma 111: - 1 pixel delay on luma
110: - 2 pixel delay on luma
Other configurations: + 0 pixel delay on luma

DENC_CFG5 Configuration 5

7	6	5	4	3	2	1	0
SELRST_INC	BKDAC1	BKDAC2	BKDAC3	BKDAC4	BKDAC5	BKDAC6	DACINV

Address: *DencBaseAddress* + 0x014
Type: R/W
Reset: 0x00
Description:

[31:8] **Reserved**

[7] **SELRST_INC**: Digital frequency synthesizer increment
Choice of digital frequency synthesizer increment after soft-reset or when PH_RST_MODE = 01 (see register DENC_CFG8)
0: hard wired value (depending on TV standard) (default)
1: Soft (value from registers DENC_DFS_INC0/1/2)

[6:1] **BKDAC1** to **BKDAC6**: Blanking of DACs *N* (1 to 6)
0: DAC *N* in normal operation (default)
1: DAC *N* input code forced to black level (if RGB, UV or C) or blanking level (if Y or CVBS) depending on CONF_OUT of register DENC_CFG8.

[0] **DACINV**: Invert DAC codes
... to compensate for an inverting output stage in the application
0: noninverted DAC inputs (outputs) (default) 1: inverted DAC inputs (outputs)

DENC_CFG6 Configuration 6

7	6	5	4	3	2	1	0
SW_RST	Reserved					TTX_EN	MAXDYN

Address: *DencBaseAddress* + 0x018
Type: R/W
Reset: 0x10
Description:

[31:8] **Reserved**

[7] **SW_RST**: Software reset[a]
0: No reset 1: Software reset

[6:2] **Reserved**
Do not change from default reset value.

[1] **TTX_EN**: Teletext enable
0: Disabled 1: Enabled

[0] **MAXDYN**: Maximum dynamic magnitude allowed on YCbCr inputs for encoding
0: 0x10 to 0xEB for Y, 0x10 to 0xE0 for chrominance (Cr, Cb)
1: 0x01 to 0xFE for Y, Cr and Cb

a. Bit SW_RST is automatically reset after internal reset generation. Software reset is active for four PIXCLK periods. When soft reset is activated, the DENC is reset as with hardware reset except for the first nine user registers (DENC_CFG0 to DENC_CFG8).

DENC_CFG7 Configuration 7

7	6	5	4	3	2	1	0
SECAM	GEN_SECAM	INV_PHI_ SECAM	Reserved			VPS_EN	SQPIX

Address: *DencBaseAddress* + 0x01C

Type: R/W

Reset: 0x00

Description:

[31:8] **Reserved**

[7:6] **SECAM**, **GEN_SECAM**: Where the bit order is SECAM - GEN_SECAM.

00: Standard selected by bits STD1 and STD0 of DENC_CFG0[a] (PAL or NTSC)
01: Reserved
10: SECAM standard, the subcarrier phase sequence start-point is line 1
11: SECAM standard, the subcarrier phase sequence start-point is line 23
If master mode is performed in the SECAM standard, bit SECAM must be set before bits SYNC2, SYNC1 and SYNC0 of DENC_CFG0. If DENC_CFG0 is not programmed, then a soft reset must be performed after programming SECAM. In SECAM, the trap filter should always be enabled (see register DENC_CFG3).

[5] **INV_PHI_SECAM**: Inversion of subcarrier phase

0: 0, 0, π,... in odd fields and π, π, 0 in even fields[a] 1: π, π, 0 in odd fields and 0, 0, π,... in even fields

[4:2] **Reserved**

[1] **VPS_EN**: VPS encoding enable

0: Disable[a] 1: Enable

[0] **SQPIX**: Square pixel mode enable.

0: Disable[a] 1: Enable

a. Default value

DENC_CFG8 Configuration 8

7	6	5	4	3	2	1	0
PH_RST_MODE		Reserved		BLK_ALL	TTX_NOTMV	Reserved	

Address: *DencBaseAddress* + 0x020

Type: R/W

Reset: 0x20

Description:

[31:8] **Reserved**

[7:6] **PH_RST_MODE**: Subcarrier phase reset mode
In modes 01 and 10, this bit is automatically reset to 00 after oscillator reset. See Section 55.7: *Subcarrier generation* *on page 651*.

00: Disabled[a]
01: Enabled - phase is updated with value from phase buffer register (see DENC_CFG2.RSTOSC_BUF) on the beginning of the next video line. Increment is updated with hard or soft value depending on SELRST_INC value (see DENC_CFG5)
10: Enabled - phase is updated with value from DENC_DFS_INC0/1 on the next increment, updating by CFC (depending on CFC loading moment and DEN_CFG6 CFC(1:0) bits.
11: Enabled - phase is reset after detecting RST bit on CFC line, up to 9 PIX_CLK after loading of CFC's LSB.
Note: Bits PH_RST_MODE[1:0] are automatically set back to 00 after the oscillator reset has been performed in modes 01 and 10.

[5:4] **Reserved**

[3] **BLK_ALL**: Blanking of all video lines

0: Disabled[a]
1: Enabled (all inputs ignored - 0x80 instead of Cr and Cb and 0x10 instead of Y)

[2] **TTX_NOTMV**: priority of ancillary data on a VBI line.
Note: higher priority data overwrites lower priority data.

0[a]: Priority is: CGMS > Closed caption > Macrovision[b] > WSS > VPS > Teletext
1: priority is: Teletext > CGMS > Closed caption > WSS > VPS > Macrovision
Note: After reset, the default value of this bit is zero, however, for devices using teletext, this bit must then be reprogrammed to 1. This is to avoid the occurrence of teletext glitches in SECAM mode and loss of teletext data in all modes.

[1:0] **Reserved**

a. Default value

b. If there is no Macrovision programmed on the line, then VBI line blanking (when register DENC_CFG1 bit BLKLI = 1) overwrites VPS or teletext, and no VPS or teletext data is encoded.

DENC_CFG9 Configuration 9 - chroma delay and luma filter control

7	6	5	4	3	2	1	0
DEL				Reserved	PLG_DIV_Y		FLT_YS

Address: *DencBaseAddress* + 0x144

Type: R/W

Reset: 0x22

Description: This register contains the chroma path delay (referenced to luma one on S-VHS and CVBS outputs), selection of input format mode, and three bits to control the luma filter.

[31:8] **Reserved**

[7:4] **DEL**[a]: Delay on chroma path
... with reference to luma path encoding (1 pixel = 2 periods of frequency f_{PIXCLK}):
0010: -0.5 pixel delay (default for PAL/NTSC in 4:2:2 format on CVBS)
0011: -1.0 pixel delay
0100: -1.5 pixel delay
0101: -2.0 pixel delay
1100: +2.5 pixel delay
1101: +2.0 pixel delay
1110: +1.5 pixel delay
1111: +1.0 pixel delay
0000: +0.5 pixel delay (default for SECAM in 4:4:4 format on CVBS)
0110, 1000, 1010: +0.5 pixel delay
0001: 0.0 pixel delay (default for PAL/NTSC in 4:4:4 format or SECAM in 4:2:2 format on CVBS)
0111, 1001, 1011: 0.0 pixel delay

[3] **Reserved**

[2:1] **PLG_DIV_Y**
00: When sum of coefficients = 256
01: When sum of coefficients = 512 (reset value)
10: When sum of coefficients = 1024
11: When sum of coefficients = 2048

[0] **FLT_YS**
0: Use hard-wired coefficients for the luma filter
1: Use register DENC_LCOEF0..9 values, default (reset) or programmed, to determine coefficients

a. DENC_CFG3.DEL_EN selects either default or programmed delays.

DENC_CFG10 Configuration 10

7	6	5	4	3	2	1	0
Reserved				RGBSAT_EN	SECAM_IN_MUX	Reserved	

Address: *DencBaseAddress* + 0x170
Type: R/W
Buffer: Immediate
Reset: 0x48

Description:

[7:4] **Reserved**

[3] **RGBSAT_EN**: RGB format output not saturated to real colors (241 max and 15 min), but allowed to have more saturated colors (255 max and 0 min).
0: RGB outputs allowed to have more saturated colors.
1: RGB outputs saturated to real colors.

[2] **SECAM_IN_MUX**: SECAM input video select. (Refer to section Section 55.10: *Luminance encoding on page 653*) Depends on SECAM bit of Register 7. This is bit is enabled if SECAM =1.
0: Main video input is SECAM encoded.
1: Aux video input is SECAM encoded.

[1:0] **Reserved**

DENC_CFG11 Configuration 11

7	6	5	4	3	2	1	0
Reserved					MAIN_IF_DEL	Reserved	

Address: *DencBaseAddress* + 0x174
Type: R/W
Buffer: Immediate
Reset: 0x28
Description: The delay on the luma comparing to chroma in CVBS_MAIN and S-VHS_MAIN outputs. This delay is five clock cycles in rectangular pixel mode (27 MHz clock) and four clock cycles in NTSC square pixel mode (24.5454 MHz clock); see Section 55.12: *Composite video signal generation on page 657*. Set MAIN_IF_DEL to 1 to enable.

DENC_CFG12 Configuration 12

7	6	5	4	3	2	1	0
Reserved					NOTCH_EN	Reserved	

Address: *DencBaseAddress* + 0x178
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: Notch filtering on the main 4:4:4 luma input; see Section 55.10: *Luminance encoding on page 653*). Set NOTCH_EN to 1 to enable.

DENC_CFG13 Configuration 13

7	6	5	4	3	2	1	0
AUX_NOTMAIN_RGB	RGB_MAXDYN	DAC123_CFG				DAC456_CFG	

Address: *DencBaseAddress* + 0x17C

Type: R/W

Buffer: Immediate

Reset: 0x82

Description: There are two filters for chroma, one for the main and one for the auxiliary path. This should be taken into account when selecting different output configurations. For example: if configuration 001 is selected on DAC123 (Y_MAIN, C_MAIN and CVBS_AUX) and configuration 010 on DAC456 (RGB) then to have correct bandwidths, RGB should be taken from the main input to have larger (and common) chroma bandwidth on Y/C and RGB outputs. On the CVBS output, chroma bandwidth should always be narrow to reduce cross luma effects.

[7] **AUX_NOTMAIN_RGB**: Select either main or aux video as RGB
0: Main video is output as RGB.
1: Aux video is output as RGB (default).
Note: There is no auxiliary video input support

[6] **RGB_MAXDYN**: Select gain when RGB is output
0: Black to peak R/G/B swing is 560 decimal values (default).
1: Black to peak R/G/B swing is 821 decimal values. This uses the maximum dynamic range of the video DACs better. Blanking level is 64.

[5:3] **DAC123_CFG**

000: DAC1 = Y_MAIN,	DAC2 = C_MAIN,	DAC3 = CVBS_MAIN (default)
001: DAC1 = Y_MAIN,	DAC2 = C_MAIN,	DAC3 = CVBS_AUX
010: DAC1 = R,	DAC2 = G,	DAC3 = B
011: DAC1 = PR_MAIN,	DAC2 = Y_MAIN,	DAC3 = PB_MAIN
100: DAC1 = CVBS_MAIN,	DAC2 = CVBS_AUX,	DAC3 = CVBS_AUX
101: DAC1 = CVBS_MAIN,	DAC2 = CVBS_AUX,	DAC3 = CVBS_MAIN
110: DAC1 = Y_AUX,	DAC2 = C_AUX,	DAC3 = CVBS_MAIN
111: DAC1 = Y_AUX,	DAC2 = C_AUX,	DAC3 = CVBS_AUX (reserved)

Note: There is no auxiliary video input support.

[2:0] **DAC456_CFG**

000: DAC1 = Y_AUX,	DAC2 = C_AUX,	DAC3 = CVBS_AUX (reserved)
001: DAC1 = Y_AUX,	DAC2 = C_AUX,	DAC3 = CVBS_MAIN
010: DAC1 = R,	DAC2 = G,	DAC3 = B (default)
011: DAC1 = PR_AUX,	DAC2 = Y_AUX,	DAC3 = PB_AUX (reserved)
100: DAC1 = CVBS_AUX,	DAC2 = CVBS_MAIN,	DAC3 = CVBS_MAIN
101: DAC1 = CVBS_AUX,	DAC2 = CVBS_MAIN,	DAC3 = CVBS_AUX
110: DAC1 = Y_MAIN,	DAC2 = C_MAIN,	DAC3 = CVBS_AUX
111: DAC1 = Y_MAIN,	DAC2 = C_MAIN,	DAC3 = CVBS_MAIN

Note: There is no auxiliary video input support.

DENC_STA Status

7	6	5	4	3	2	1	0
HOK	ATFR	BUF2_FREE	BUF1_FREE	FIELDCT			Reserved

Address: *DencBaseAddress* + 0x024
Type: RO
Buffer: Immediate
Reset: Undefined
Description:

[7] **HOK**: Hamming decoding of frame sync flag embedded within ITU-R656 / D1 compliant YCrCb streams
0: Consecutive errors
1: A single or no error (default)
Note: Signal quality detector is issued from Hamming decoding of EAV,SAV from YCrCb

[6] **ATFR**: Frame synchronization flag
0: Encoder not synchronized (default)
1: In slave mode: encoder synchronized

[5] **BUF2_FREE**: Closed caption registers access condition for field 2
(See Section 55.14: *Closed captioning on page 659*)
Closed caption data for field 2 is buffered before being output on the relevant TV line; BUF2_FREE is reset if the buffer is temporarily unavailable. If the microcontroller can guarantee that registers DENC_CCCF2 are never written more than once between two frame reference signals, then bit BUF2_FREE will always be true (set). Otherwise, closed caption field 2 registers access might be temporarily forbidden by resetting bit BUF2_FREE until the next field 2 closed caption line occurs.
Note: This bit is false (reset) when 2 pairs of data bytes are awaiting to be encoded, and is set back immediately after one of these pairs has been encoded (so at that time, encoding of the last pair of bytes is still pending)
Reset value = 1 (access authorized)

[4] **BUF1_FREE**: Closed caption registers access condition for field 1
Same as BUF2_FREE but concerns field 1.
(*) Reset value:= 1 (access authorized)

[3:1] **FIELDCT:** Digital field identification number
000: Indicates field 1
...
111: Indicates field 8
FIELDCT[0] also represents the odd/even information (odd='0', even='1')

[0] **Reserved**

56.2 Digital frequency synthesizer

DENC_DFS_INC0/1/2 Increment for digital frequency synthesizer, bytes 0, 1, 2

DFS_INC0	23	22	21	20	19	18	17	16
0x028	D							
DFS_INC1	15	14	13	12	11	10	9	8
0x02C	D							
DFS_INC2	7	6	5	4	3	2	1	0
0x030	D							

Address: *DencBaseAddress* + 0x028, 0x02C, 0x030

Type: R/W

Reset: Undefined

Description: These three registers contain the 24-bit increment used by the DDFS, **only if** bit DENC_CFG5.SELRST_INC equals 1. They generate the phase of the subcarrier, that is, the address that is supplied to the sine ROM. It therefore customizes the synthesized subcarrier frequency: 1 LSB ~ 1.6 Hz

To validate use of these registers instead of the hard-wired values:

- Load the registers with the required value
- Set bit SELRST_INC to 1 (register DENC_CFG5)
- Perform a software reset (register DENC_CFG6)

Note: *The values loaded in* DENC_DFS_INC0/1/2 *are taken into account after a software reset,* **only if** *bit* SELRST_INC*= 1 (*register DENC_CFG5*)*
These registers are never reset and must be explicitly written, to ensure that they contain sensible information.

On hardware or software reset the DDFS is initialized with a hardwired increment, independently of DENC_DFS_INC0/1/2. These hardwired values cannot be read out of the DENC.

[23:0] **D**: Hard-wired default value	f, Frequency synthesized, MHz	Ref.Clock, MHz
0x 21 F07C: NTSC M	3.5795452	27
0x2A 098B: PAL BGHIN	4.43361875	27
0x21 F694: PAL N	3.5820558	27
0x21 E6F0: PAL M	3.57561149	27
0x25 5554: NTSC M square pixel	3.5795434	24.545454
0x26 798C: PAL BGHIN square pixel	4.43361867	29.5
0x1F 15C0: PAL N square pixel	3.58205605	29.5
0x25 4AD4: PAL M square pixel	3.57561082	24.545454

DENC_DFS_PHASE0/1 Phase DFS registers

Address: *DencBaseAddress* + 0x034, 0x038

Type: R/W

Reset: HUE_EN: Disabled (0), DFS_PHASE: Undefined; never reset and must be explicitly written to.

Description: If bit SEL_RST = 0 (for example, after a hardware reset), the phase offset used every time the DDFS is reinitialized is hard-wired. The hard-wired values cannot be read out of the DENC. These are:

0xD9 C000 for PAL BDGHI, N, M,
0x1F C000 for NTSC-M,
0x00 0000 (blue lines)
0x43 C000 (red lines) for SECAM

When SECAM = 0 (DENC_CFG7)

Static phase offset for digital frequency synthesizer (10 bits only)

DFS_PHASE0	-	-	-	o26	-	-	o23	o22
0x034	Reserved			HUE_EN	Reserved		VALUE	
DFS_PHASE1	o21	o20	o19	o18	o17	o16	o15	o14
0x038	VALUE							

Description: HUE_EN:

Enables variance in phase of the subcarrier, as programmed in DENC_HUE, during active video with respect to the phase of the subcarrier during the color burst. Once set, this bit automatically resets to 0.

0: Disabled (Default)
1: Enabled

VALUE:

Under certain circumstances (detailed below), these registers contain the 10 MSBs of the value with which the phase accumulator of the DDFS is initialized after a 0-to-1 transition of bit RSTOSC_BUF (DENC_CFG2), or after a standard change, or when cyclic phase readjustment has been programmed (see DENC_CFG2.VAL_RST). The 14 remaining LSBs loaded into the accumulator in these cases are all 0s (defining the phase reset value with a 0.35° accuracy).

To validate use of these registers instead of the hard-wired values:

- Load the registers with the required value.
- Set bit DENC_CFG2.SEL_RST to 1.
- Perform a software reset (DENC_CFG6), or set DENC_CFG2 bit RSTOSC_BUF to 1. This puts the soft-phase value into a temporary register and sets DENC_CFG8.PH_RST_MODE to put this value into an accumulator at the beginning of the next line.

When SECAM = 1 (DENC_CFG7)

Static phase offset for digital frequency synthesizer for two SECAM subcarriers (8 MSB only) (blue lines - DENC_DFS_PHASE0 **and red lines -** DENC_DFS_PHASE1**)**

DFS_PHASE0	b21	b20	b19	b18	b17	b16	b15	b14
0x034	VALUE							
DFS_PHASE1	r21	r20	r19	r18	r17	r16	r15	r14
0x038	VALUE							

Description: These registers contain the 8 bits (21 to 14) of the value with which the phase accumulator of the DDFS is initialized on **every line** in SECAM mode. The phase is calculated with 1.4° accuracy as:

(Blue lines) DENC_DFS_PHASE0 x 16384 (dec), or DENC_DFS_PHASE0 x 0x4000

(Red lines) (256 + DENC_DFS_PHASE0 + DENC_DFS_PHASE1) x 16384 (dec), or (100 + DENC_DFS_PHASE0 + DENC_DFS_PHASE1) x 0x4000

To validate use of these registers instead of the hard-wired values, follow the procedure for PAL and NTSC mode.

56.3 WSS

DEN_WSSn WSS data 1 and 2

15	14	13	12	11	10	9	8
WSS15	WSS14	WSS13	WSS12	WSS11	WSS10	WSS9	WSS8
7	6	5	4	3	2	1	0
WSS7	WSS6	WSS5	WSS4	WSS3	WSS2	WSS1	WSS0

Address: *DencBaseAddress* + 0x03C, 0x040
Type: R/W
Buffer: Double buffered
Reset: Undefined
Description:

[15] **WSS15**: Reserved: must be set to 0.

[14] **WSS14**: Helper bit
0: No helper 1: Modulated helper
A helper signal may be present only when the aspect ratio label is either 16:9 letterbox center or > 16:9 letterbox center and the number of active lines 430 lines.

[13] **WSS13**: Color coding bit
0: Standard coding 1: Motion adaptive color plus
In film mode (bit b 4 = 1), motion adaptive color plus is set to fixed color plus operation, in other words it is not motion adaptive.

[12] **WSS12**: Film bit
0: Camera mode 1: Film mode

[11:8] **WSS**[11:8]: Aspect ratio
0001: Box 14:9 center 0010: Box 14:9 top
0100: Box 16:9 top 0111: Full format 16:9 (anamorphic)
1000: Full format 4:3 1011: Box 16:9 center
1101: > Box 16:9 center 1110: Full format 4:3 (shoot and protect 14:9 center)
WSS11 is an odd parity bit.

[7] **WSS7**: CGMS-A: Generation bit
0: Copying not restricted 1: Copying restricted

[6] **WSS6**: CGMS-A: copyright bit
0: No copyright asserted or status unknown 1: Copyright asserted

[5] **WSS5**: Surround sound bit
0: No surround sound information 1: Surround sound mode

[4:3] **WSS**[4:3]: Subtitling mode
00: No open subtitles 01: Subtitles in active image area
10: Subtitles out of active image area 11: Reserved

[2] **WSS2**: Subtitles with teletext bit
0: No subtitles within teletext 1: Subtitles within teletext

[1:0] **WSS**[1:0]: Reserved: must be set to 0.

56.4 DAC inputs

DENC_DAC13_MULT DAC1 and DAC3 multiplying factors

7	6	5	4	3	2	1	0
DAC1_MULT[5:0]						DAC3_MULT[5:4]	

Address: *DencBaseAddress* + 0x044
Type: R/W
Reset: 0x82
Description:

[7:2] **DAC1_MULT[5:0]**

[1:0] **DAC3_MULT[5:4]**

DENC_DAC34_MULT DAC3 and DAC4 multiplying factors

7	6	5	4	3	2	1	0
DAC3_MULT[3:0]				DAC4_MULT[5:2]			

Address: *DencBaseAddress* + 0x048
Type: R/W
Reset: 0x08
Description:

[7:4] **DAC3_MULT[3:0]**

[3:0] **DAC4_MULT[5:2]**

DENC_DAC45_MULT DAC4 and 5 multiplying factors

7	6	5	4	3	2	1	0
DAC4_MULT[1:0]		DAC5_MULT[5:0]					

Address: *DencBaseAddress* + 0x04C
Type: R/W
Reset: 0x20
Description:

[31:8] **Reserved**

[7:6] **DAC4_MULT[1:0]**: Multiplying factor on DAC4_R_V_C digital signal before D/A converters with 0.78% step (see also DENC_DAC34_MULT)

	% of default value	
	if R, G or B (DAC123_CFG = 010)	if not R, G or B (DAC123_CFG != 010)
00 0000:	81.25	75.00
00 0001:	81.84	75.78
00 0010:	82.42	76.56
00 0011:	83.01	77.34
.. ..		
10 0000: (default)	100.00	100.00
.. ..		
11 1111:	118.16	124.22

[5:0] **DAC5_MULT[5:0]**: Multiplying factor on DAC5_G_Y digital signal before D/A converters with 0.78% step

	% of default value	
	if R, G or B (DAC123_CFG = 010)	if not R, G or B (DAC123_CFG != 010)
00 0000:	81.25	75.00
00 0001:	81.84	75.78
00 0010:	82.42	76.56
00 0011:	83.01	77.34
.. ..		
10 0000: (default)	100.00	100.00
.. ..		
11 1111:	118.16	124.22

DENC_DAC2_MULT DAC2 multiplying factors

7	6	5	4	3	2	1	0
Reserved		DAC2_MULT[5:0]					

Address: *DencBaseAddress* + 0x1A8
Type: R/W
Reset: 0x20
Description:

DENC_DAC6_MULT DAC6 multiplying factors

7	6	5	4	3	2	1	0
Reserved		DAC6_MULT[5:0]					

Address: *DencBaseAddress* + 0x1AC
Type: R/W
Reset: 0x20
Description:

56.5 Hardware ID

DENC_HW_VER DENC hardware version

7	6	5	4	3	2	1	0
CHIPID							

Address: *DencBaseAddress* + 0x060
Type: RO
Reset: 0xC0
Description: CHIPID: 1100 0000

56.6 VPS data

DENC_VPSn VPSn data registers

	7	6	5	4	3	2	1	0
DENC_VPS0: 0x64	PROGT							
DENC_VPS1: 0x68	CNTRY[1:0]		NP[5:0]					
DENC_VPS2: 0x6C	MIN						CNTRY[3:2]	
DENC_VPS3: 0x70	MON[2:0]			HR				
DENC_VPS4: 0x74	NP[7:6]		DAY					MON[3]
DENC_VPS5: 0x78	SND		Reserved		CNI			

Address: *DencBaseAddress* + register offset

Type: Read only

Reset: Undefined

Description: Refer to Section 55.17: *VPS encoding on page 661*.

DENC_VPS5: [7:6] **SND**: sound
00: Don't know 01: Mono
10: Stereo 11: Dual sound

[5:4] **Reserved**

DENC_VPS5: [3:0] **CNI**: 4 bits of CNI, reserved for enhancement of VPS

DENC_VPS4: [7:6]
DENC_VPS1: [5:0] **NP[7:0]**: Network or program CNI (Country and Network Identification)

DENC_VPS4: [5:1] **DAY**: Day, binary

DENC_VPS4: [0]
DENC_VPS3: [7:5] **MON[3:0]**: Month, binary

DENC_VPS3: [4:0] **HR**: Hour, binary

DENC_VPS2: [7:2] **MIN**: Minute, binary

DENC_VPS2: [1:0]
DENC_VPS1: [7:6] **CNTRY[3:0]**: Country, binary

DENC_VPS0: [7:0] **PROGT[7:0]**: Program type, binary

56.7 CGMS data

DENC_CGMS0/1/2 CGMS data registers (20 bits only)

CGMS0	7	6	5	4	3	2	1	0
0x07C	Reserved				B1	B2	B3	B4
CGMS1	7	6	5	4	3	2	1	0
0x080	B5	B6	B7	B8	B9	B10	B11	B12
CGMS2	7	6	5	4	3	2	1	0
0x084	B13	B14	B15	B16	B17	B18	B19	B20

Address: *DencBaseAddress* + register offset
Type: R/W
Reset: Undefined
Description:

B1 to **B3**: Word 0A

B4 to **B6**: Word 0B

B7 to **B10**: Word 1

B11 to **B14**: Word 2

B15 to **B20**: CRC (not internally computed)

56.8 Teletext

DENC_TTX1 Teletext block definition

7	6	5	4	3	2	1	0
FP_TTXT	TTXDEL2	TTXDEL1	TTXDEL0	TTX_L6	TTX_L7	TTZ_L8	TTX_L9

Address: *DencBaseAddress* + 0x088
Type: R/W
Reset: 0x40
Description:

[7] **FP_TTXT**: Full page teletext mode enable (refer to Section 55.18: *Teletext encoding on page 661*)
0: Disable
1: Enable

[6:4] **TTXDEL[2:0]**: Teletext data latency (* “100” default) (Refer to Section 55.18: *Teletext encoding on page 661*)

The encoder will clock in the first teletext data sample on the (2 + TXDL[2:0])th rising edge of the master clock following the rising edge of TTXS (teletext synchro signal, supplied by the encoder).

[3:0] **TTX_L6:L8:** See Table 181: *Teletext line selection for standards other than ITU-R and 625 line systems on page 688*

DENC_TTX2-5 Teletext block definition

7	6	5	4	3	2	1	0
TTX_L10	TTX_L11	TTX_L12	TTX_L13	TTX_L14	TTX_L15	TTZ_L16	TTX_L17
TTX_L18	TTX_L19	TTX_L20	TTX_L21	TTX_L22	TTX_L23	TTZ_L318	TTX_L319
TTX_L320	TTX_L321	TTX_L322	TTX_L323	TTX_L324	TTX_L325	TTZ_L326	TTX_L327
TTX_L328	TTX_L329	TTX_L330	TTX_L331	TTX_L332	TTX_L333	TTZ_L334	TTX_L335

Address: *DencBaseAddress* + 0x08C, 0x090, 0x94, 0x098

Type: R/W

Reset: TTX1: 0x40; TTX2-5: 0x00

Description: Each of these bits enables teletext on line x (ITU-R line numbering and 625 line systems). For other standards refer to the table below for teletext lines selection.

Table 181: Teletext line selection for standards other than ITU-R and 625 line systems

	PAL BDGHIN line (CCIR 625 line numbering)	PAL M line (CCIR 525 line numbering)	NTSC M line (SMPTE 525 line numbering)
TTX_L6	6	6	9
TTX_L7	7	7	10
TTX_L8	8	8	11
TTX_L9	9	9	12
TTX_L10	10	10	13
TTX_L11	11	11	14
TTX_L12	12	12	15
TTX_L13	13	13	16
TTX_L14	14	14	17
TTX_L15	15	15	18
TTX_L16	16	16	19
TTX_L17	17	17	20
TTX_L18	18	18	21
TTX_L19	19	19	22
TTX_L20	20	20	23
TTX_L21	21	21	24
TTX_L22	22	22	25
TTX_L23	23	23	26
TTX_L318	318	268	271
TTX_L319	319	269	272
TTX_L320	320	270	273
TTX_L321	321	271	274
TTX_L322	322	272	275
TTX_L323	323	273	276
TTX_L324	324	274	277
TTX_L325	325	275	278
TTX_L326	326	276	279
TTX_L327	327	277	280

Table 181: Teletext line selection for standards other than ITU-R and 625 line systems

	PAL BDGHIN line (CCIR 625 line numbering)	PAL M line (CCIR 525 line numbering)	NTSC M line (SMPTE 525 line numbering)
TTX_L328	328	278	281
TTX_L329	329	279	282
TTX_L330	330	280	283
TTX_L331	331	281	284
TTX_L332	332	282	285
TTX_L333	333	283	286
TTX_L334	334	284	287
TTX_L335	335	285	288

DENC_TTX_CFG Teletext configuration

7	6	5	4	3	2	1	0
TTXT100IRE	TTXT_ABCD1	TTXT_ABCD0	Reserved				

Address: *DencBaseAddress* + 0x100
Type: R/W
Buffer: Double buffered
Reset: 0x50
Description:

[31:8] **Reserved**

[7] **TTXT100IRE**: Amplitude of teletext waveform
0: 70 IRE 1: 100 IRE

[6:5] **TTX_ABCD[1:0]**: Teletext standard selection
00: Teletext A
01: Teletext B/world system teletext (625 line systems)
10: Teletext C/NABTS (525 line systems)
11: Teletext D

[4:0] **Reserved**

DENC_CM_TTX C multiplying factor and teletext

7	6	5	4	3	2	1	0
C_MULT				TTX_MASK_OFF	Reserved		BCS_MAIN_EN

Address: *DencBaseAddress* + 0x104

Type: R/W

Reset: 0x01

Description:

[31:8] **Reserved**

[7:4] **C_MULT**: Multiplying factor on C digital output (before the D/A converters) with 3.125% step
(C value compared to default)
0000: 1.000 000
0001: 1.000 001 (1.015625 Dec.)
0010: 1.000 010 (1.031250 Dec.)
0011: 1.000 011 (1.046875 Dec.)
....
1111: 1.001 111 (1.234375 Dec.)

Default peak to peak amplitude of U and V outputs corresponds to 70% of default Y or CVBS peak to peak amplitude if 100/0/100/0 color bar pattern is input. In other words, when I_{ref} is set to deliver 1 V_{PP} for CVBS on DAC3 for example (and DAC3_MULT = “1000”), when switched to U, DAC6 delivers 0.7 V_{PP}.
Default peak to peak amplitude of RGB outputs corresponds to 70% of default Y or CVBS peak to peak amplitude if 100/0/75/0 color bar pattern is input. In other words, when I_{ref} is set to deliver 1 V_{PP} for CVBS on DAC3 for example (and DAC3_MULT = “1000”), when switched to B, DAC6 delivers 0.7 V_{PP}.

[3] **TTX_MASK_OFF**: Masking of 2 bits in Teletext framing code, depending on selected teletext standard
0: Enable 1: Disable

[2:1] **Reserved**

[0] **BCS_MAIN_EN**
Brightness, contrast and saturation control by registers DENC_BRIGHT, DENC_CONTRAST and DENC_SATURATION on 4:4:4 main video input
0: Disable 1: Enable

56.9 Closed caption

DENC_CCCF1 Closed-caption characters/extended data for field 1

CCCF10	7	6	5	4	3	2	1	0
0x09C	OPC11	C117	C116	C115	C114	C113	C112	C111
CCCF11	7	6	5	4	3	2	1	0
0x0A0	OPC12	C127	C126	C125	C124	C123	C122	C121

Address: *DencBaseAddress* + 0x09C, 0x0A0

Type: R/W

Reset: Undefined (these registers are never reset)

Description: These registers have no default value, but enabling closed captions without loading these registers issues a null character.

OPC11: Odd-parity bit of US-ASCII 7-bit character C117 to C111

C117 - **C111**: First byte to encode in field 1

OPC12: Odd-parity bit of US-ASCII 7-bit character C127 to C121

C127 - **C121**: Second byte to encode in field 1

DENC_CCCF2 Closed caption characters/extended data for field 2

CCCF20	7	6	5	4	3	2	1	0
0x0A4	OPC21	C217	C216	C215	C214	C213	C212	C211
CCCF21	7	6	5	4	3	2	1	0
0x0A8	OPC22	C227	C226	C225	C224	C223	C222	C221

Address: *DencBaseAddress* + 0x0A4, 0x0A8

Type: R/W

Reset: Undefined (these registers are never reset)

Description: These registers have no default value, but enabling closed captions without loading these registers issues a null character.

[7] **OPC21**: Odd-parity bit of US-ASCII 7-bit character C217 to C211

[6:0] **C217** - **C211**: First byte to encode in field 2

[7] **OPC22**: Odd-parity bit of US-ASCII 7-bit character C227 to C221

[6:0] **C227** - **C221**: Second byte to encode in field 2

DENC_CCLIF1 Closed caption/extended data line insertion for field 1

7	6	5	4	3	2	1	0
Reserved			L14	L13	L12	L11	L10

Address: *DencBaseAddress* + 0x0AC

Type: R/W

Reset: 0x0F

Description: This register programs the TV line number of the closed caption/extended data encoded in field 1

525/60 system: (525-SMPTE line number convention)

Only lines 10 to 22 should be used for closed caption or extended data services (lines 1 to 9 contain the vertical sync pulses with equalizing pulses).

L14 to **L10**

0 0000: No line selected for closed caption encoding

0 00xx: Do not use these codes

....

i code line (i+6) (SMPTE) selected for encoding

....

1 1111: Line 37 (SMPTE) selected

625/50 system: (625-CCIR/ITU-R line number convention)

Only lines 7 to 23 should be used for closed caption or extended data services.

L14 to **L10**

0 0000: No line selected for closed caption encoding

....

i code line (i+6) (CCIR) selected for encoding (i>0)

....

1 1111: line 37 (CCIR) selected

Default value = 0 1111 line 21 (525/60, 525-SMPTE line number convention); this value also corresponds to line 21 in I625/50 system,(625-CCIR line number convention).

DENC_CCLIF2 Closed caption/extended data line insertion for field 2

7	6	5	4	3	2	1	0
Reserved			L24	L23	L22	L21	L20

Address: *DencBaseAddress* + 0x0B0

Type: R/W

Reset: 0x0F

Description: This register programs the TV line number of the closed caption/extended data encoded in field 1

525/60 system: (525-SMPTE line number convention)

Only lines 273 to 284 should be used for closed caption or extended data services (preceding lines contain the vertical sync pulses with equalizing pulses.

L24 to **L20**

0 0000:	No line selected for closed caption encoding
0 00xx:	Do not use these codes
i line (269 +i)	(SMPTE) selected for encoding
....	
0 1111:	Line 284 (SMPTE) selected for encoding
1 1111:	Line 289 (SMPTE)

Note: If CGMS is allowed on lines 20 and 283 (525/60, 525-SMPTE line number convention), closed captions should not be programmed on these lines.

625/50 system: (625-CCIR line number convention)

Only lines 319 to 336 should be used for closed caption or extended data services (preceding lines contain the vertical sync pulses with equalizing pulses).

L24 to **L20**

0 0000:	No line selected for closed caption encoding
i line (318 +i)	(CCIR) selected for encoding
....	
1 0010:	Line 336 (CCIR) selected for encoding
1 1111:	Line 349 (CCIR)

Default value:= 0 1111 line 284 (525/60, 525-SMPTE line number convention) this value also corresponds to line 333 in 625/50 system, (625-CCIR line number convention)

56.10 Input demultiplexor

DENC_BRIGHT Brightness

7	6	5	4	3	2	1	0
B							

Address: *DencBaseAddress* + 0x114

Type: R/W

Reset: 1000 0000

Description: The register contents are used by the following formula to adjust the luminance intensity of the display video image:

$$Y_{OUT} = Y_{IN} + B - 128$$

Where Y_{IN} is 8-bit input luminance and Y_{OUT} is the result of a brightness operation (still on 8 bits).

This value is saturated at 235 (16) or 254 (1) according to DENC_CFG6 bit MAXDYN, B: brightness (unsigned value with center at 128, default 128).

DENC_CONTRAST Contrast

7	6	5	4	3	2	1	0
C							

Address: *DencBaseAddress* + 0x118

Type: R/W

Reset: 0000 0000

Description: The register contents are used by the following formula to adjust the relative difference between the display image higher and lower intensity luminance values:

$$Y_{OUT} = \frac{(Y_{IN} - 128)(C + 128)}{128} + 128$$

where, Y_{IN} is 8-bit input luminance, Y_{OUT} is the result of a contrast operation (still on 8 bits).

This value is saturated at 235 (16) or 254 (1) according to DENC_CFG6 bit MAXDYN, C: contrast (two's complement value from -128 to 127, default 0).

DENC_SATURATION Saturation

7	6	5	4	3	2	1	0
S							

Address: *DencBaseAddress* + 0x11C

Type: R/W

Reset: 1000 0000

Description: The register contents are used by the following formula to adjust the color intensity of the displayed video image:

$$Crout = \frac{s(Crin - 128)}{128} + 128$$

$$Cbout = \frac{s(Cbin - 128)}{128} + 128$$

Where Cr_{IN} and Cb_{IN} are the 8-bit input chroma, Cr_{OUT} and Cb_{OUT} are the result of a saturation operation (still on 8 bits)

This value is saturated at 240 (16) or 254 (1) according to DENC_CFG6 bit MAXDYN, S: saturation value (unsigned value with centre at 128, default 128).

56.11 Chroma coefficient

DENC_CF_COEFn Chroma filter coefficient 0 to 8 (main video)

	7	6	5	4	3	2	1	0
CF_COEF0: 0x120	MAIN_FLT_S	MAIN_PLG_DIV[1:0]		CF_COEF0[4:0]				
CF_COEF1: 0x124	Reserved	CF_COEF8[8]	CF_COEF1[5:0]					
CF_COEF2: 0x128	Reserved	CF_COEF2[6:0]						
CF_COEF3: 0x12C	Reserved	CF_COEF3[6:0]						
CF_COEF4: 0x130	CF_COEF4[7:0]							
CF_COEF5: 0x134	CF_COEF5[7:0]							
CF_COEF6: 0x138	CF_COEF6[7:0]							
CF_COEF7: 0x13C	CF_COEF7[7:0]							
CF_COEF8: 0x140	CF_COEF8[7:0]							

Address: *DencBaseAddress* + 0x120 (DEN_CFCOEF0), 0x124, 0x128, 0x12C, 0x130, 0x134, 0x138, 0x13C, 0x140 (DEN_CFCOEF8)

Type: R/W

Buffer: Immediate

Reset: See below

Description: These registers contain the nine coefficients and three control bits for the chroma filter, which are described below. Values from these registers are used only when MAIN_FLT_S is set to 1.

If MAIN_FLT_S is 1 the digital encoder is programmed from MAIN_PLG_DIV[1:0] to all the chroma coefficients. When MAIN_FLT_S is set to 0 the default values of the coefficients are loaded depending on the mode selected or the type of filter selected in a particular mode with FLT (DENC_CFG1). If SECAM square pixel mode is performed, the values from these registers are loaded and their default values correspond to that standard.

The coefficients are chosen to give the required filter response for a specific application according to the symmetrical FIR filter equation:
$H(z) = C_0 + C_1z^{-1} + C_2z^{-2} + ... + C_7z^{-7} + C_8z^{-8} + C_7z^{-9} + ... + C_2z^{-14} + C_1z^{-15} + C_0z^{-16}$
The register reset (or default) values give the coefficients for the SECAM square pixel mode.
Each register value is calculated by adding an offset value to the desired coefficient value, according to the relationship: *register value* = *offset* + *actual coefficient value*.
For instance, to obtain a *coefficient value* of 5 for C4, which has an offset of 32, the register DEN_CFCOEF4 must contain the value 100101, which is the binary equivalent of 32 + 5.
Filter sampling frequency is VIDPIX_2XCLK (27 MHz, 24.545454 MHz or 29.5 MHz).

MAIN_PLG_DIV[1:0]: This value is chosen depending on the sum of all coefficients.
00: When sum of coefficients = 512
01: When sum of coefficients = 1024 (default)
10: When sum of coefficients = 2048
11: When sum of coefficients = 4096

MAIN_FLT_S
0: Use hardwired coefficients for the chroma filter.
1: Use register DENC_CF_COEFn values, default (reset) or programmed, to determine coefficients.

Reserved bits are set to 0.

The value of MAIN_PLG_DIV is chosen depending on what is the sum of all the coefficients.

- MAIN_PLG_DIV = 11 when sum of coeffs = 4096,
- MAIN_PLG_DIV = 10 when sum of coeffs = 2048,
- MAIN_PLG_DIV = 01 when sum of coeffs = 1024,default,
- MAIN_PLG_DIV = 00 when sum of coeffs = 512.

Offset change before loading to user register:

- CHROMA_MAIN_COEF0 = Actual value + 16,
- CHROMA_MAIN_COEF1 = Actual value + 32,
- CHROMA_MAIN_COEF2 = Actual value + 64,
- CHROMA_MAIN_COEF3 = Actual value + 32,
- CHROMA_MAIN_COEF4 = Actual value + 32,
- CHROMA_MAIN_COEF5 = Actual value + 32,
- CHROMA_MAIN_COEF6 = Actual value,
- CHROMA_MAIN_COEF7 = Actual value,
- CHROMA_MAIN_COEF8 = Actual value.

Default values:

- CHROMA_MAIN_COEF0[4:0] = 1 0001 means c0 = 1,
- CHROMA_MAIN_COEF1[5:0] = 10 0111 means c1 = 7,
- CHROMA_MAIN_COEF2[6:0] = 101 0100 means c2= 20,
- CHROMA_MAIN_COEF3[6:0] = 100 0111 means c3 = 39,
- CHROMA_MAIN_COEF4[7:0] = 0101 1111 means c4 = 63,
- CHROMA_MAIN_COEF5[7:0] = 0111 0111 means c5 = 87,
- CHROMA_MAIN_COEF6[7:0] = 0110 1100 means c6 = 108,
- CHROMA_MAIN_COEF7[7:0] = 0111 1011 means c7 = 123,
- CHROMA_MAIN_COEF8[8:0] = 0 1000 0000 means c8 = 128.

56.12 Luma coefficient

DENC_LU_COEF0..9 Luma filter coefficients 0 to 9

	7	6	5	4	3	2	1	0
LU_COEF0: 0x148	Reserved			LU_COEF8[8]	LU_COEF0[4:0]			
LU_COEF1: 0x14C	LU_COEF9[9:8]			LU_COEF1[5:0]				
LU_COEF2: 0x150	LU_COEF6[8]	LU_COEF2[6:0]						
LU_COEF3: 0x154	LU_COEF7[8]	LU_COEF3[6:0]						
LU_COEF4: 0x158	LU_COEF4[7:0]							
LU_COEF5: 0x15C	LU_COEF5[7:0]							
LU_COEF6: 0x160	LU_COEF6[7:0]							
LU_COEF7: 0x164	LU_COEF7[7:0]							
LU_COEF8: 0x168	LU_COEF8[7:0]							
LU_COEF9: 0x16C	LU_COEF9[7:0]							

Address: *DencBaseAddress* + register offset

Type: R/W

Reset: See table below

Description: These registers contain the 10 coefficients for the luma filter. The coefficients give the required filter response for a specific application according to the symmetrical FIR filter equation:

$$H(z) = a0 + a1z^{-1} + a2z^{-2} + a8z^{-8} + a9z^{-9} + a8z^{-10} + \ldots + a2z^{-16} + a1z^{-17} + a0z^{-18}$$

The values of the coefficients LU_COEF0 through to LU_COEF7 must be entered in two's complement form and the remainder as normal positive values.

The sampling frequency of the filter is PIXCLK (27 MHz, 24.5454 MHz or 29.5 MHz).

The control bits for this filter are in register DENC_CFG9. This register is used only when bit FLT_YS in DENC_CFG9 is set to 1.

Reserved bits are reset to 0.

Default values:

- a_0 = LUMA_COEF_0[4:0] = 0 0001 means +1,
- a_1 = LUMA_COEF_1[5:0] = 11 1111 means -1,
- a_2 = LUMA_COEF_2[6:0] = 111 0111 means -9,
- a_3 = LUMA_COEF_3[6:0] = 000 0011 means +3,
- a_4 = LUMA_COEF_4[7:0] = 0001 1111 means +31,
- a_5 = LUMA_COEF_5[7:0] = 1111 1011 means -5,
- a_6 = LUMA_COEF_6[8:0] = 1 1010 1100 means -84,
- a_7 = LUMA_COEF_7[8:0] = 0 0000 0111 means +7,
- a_8 = LUMA_COEF_8[8:0] = 1 0011 1101 means +317,
- a_9 = LUMA_COEF_9[9:0] = 01 1111 1000 means +504.

56.13 Hue control

DENC_HUE Hue control

7	6	5	4	3	2	1	0
HUE_PHS_SGN	HUE_PHS_VAL						

Address: *DencBaseAddress* + 0x1A4

Type: R/W

Buffer: Immediate

Reset: 0x00 (no phase shift)

Description: Defines the phase shift in the subcarrier during active video with respect to the phase of the subcarrier during the color burst. Once enabled by DENC_DFS_PHASE0/1.HUE_EN, phase variation may vary by +/- 22.324 degrees with increments of 0.17578127.

Note: *To make a value programmed in this register effective, immediately after programming* HUE_CTRL*, pulse* DENC_DFS_PHASE0/1.HUE_EN*.*

Once enabled, to disable any phase shift in active subcarrier with respect to burst, write the default value into this register followed by a pulse on DENC_DFS_PHASE0/1.HUE_EN.

[7] **HUE_PHS_SGN**: Sign of phase
1: Positive 0: Negative

[6:0] **HUE_PHS_VAL**: Absolute value of phase adjustment, range 1 to 127
LSB 0x01 implies 0.17578127 degrees, 0x7F imply 22.324 degrees.
1000 0000: No phase shift 0000 0000: No phase shift
1111 1111: +22.324 degrees phase 0111 1111: -22.324 degrees phase

57 High-definition multimedia interface (HDMI)

This chapter describes the HDMI frame formatter.

57.1 Glossary

HDMI	High definition multimedia interface HDMI is a licensable format designed for transmitting digital television audiovisual signals from DVD players, set-top boxes and other audiovisual sources to television sets, projectors and other video displays. HDMI can carry high quality multi-channel audio data, and all standard and high-definition consumer electronics video formats, together with bidirectional control and status signals. Content protection technology is available.
DVI	Digital visual interface
TMDS	Transition minimized differential signalling
TERC4	TMDS error reduction coding - 4 bit
SSC4	Synchronous serial controller version 4.0

57.2 Overview

The HDMI block transmits HDMI-compatible data. It primarily consists of two sub-blocks:

- **digital HDMI frame formatter**: frame formatting and encoding.
- **analog line driver**: serializes the data and transmits the differential line,

Video data synchronous to the pix clock is fed to the HDMI via three 8-bit buses, HSYNC and VSYNC lines. The video interface supports **RGB**, **YCbCr 4:4:4** and **YCbCr 4:2:2** formats only. The encryption keys generated by the HDCP block are fed to the HDMI on a separate interface. The HDMI **XORs the video data with the keys** generated by the HDCP. If the HDMI is in not authenticated, a default (programmed) value is sent during the video data interval periods; audio data is **not sent** at all; null packets and AVI info frames are sent instead.

The HDMI receives audio data via an S/PDIF interface. The HDMI does not compress or decompress audio, but simply inserts audio into appropriate fields in the HDMI frame. The audio interface performs the cycle time stamp (CTS) calculations and inserts the CTS in the frame appropriately.

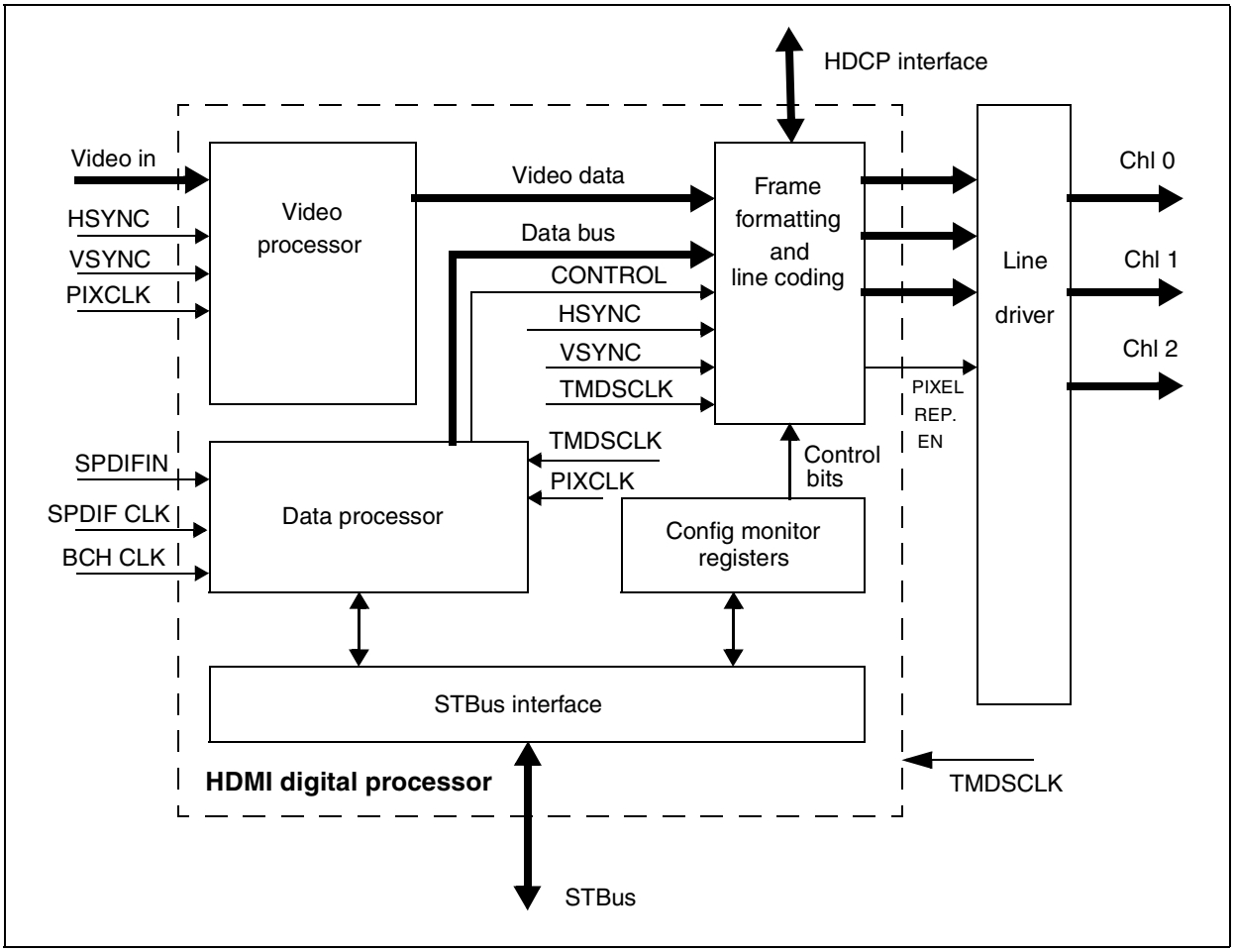
The HDMI is configured via its STBus interface. This interface is also used to transmit data for info frame during the data island period of the HDMI frame.

Control data is inserted appropriately, depending on the HDMI configuration.

The HDMI frame formatter interfaces to the line driver on three 10-bit parallel buses, which are synchronous with the TMDS clock.

The block diagram for the HDMI is shown in Figure 230 below.

Figure 230: HDMI block diagram



Each 10-bit bus carries the transition minimized differential signalling (TMDS) data. The TMDS data changes with each rising edge of the TMDS clock. The clocks have the following characteristics:

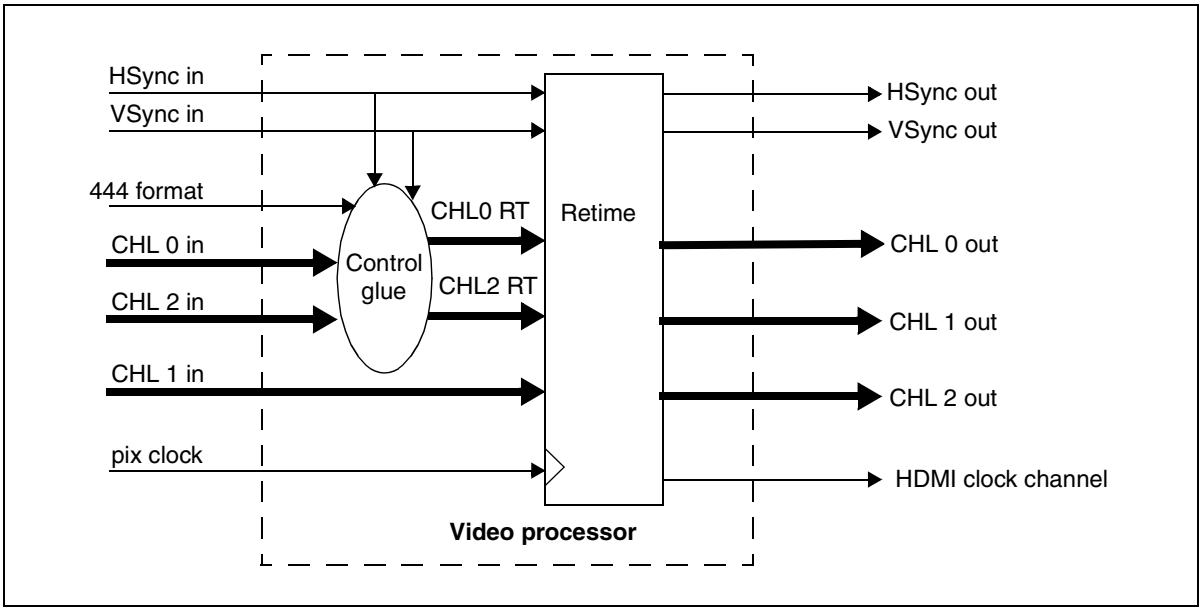
- The TMDS clock frequency is *n* x PIXCLOCK where *n* = 1 or 2.
- The TMDS and pix clocks are synchronous.
- The BCH clock frequency is *m* x TMDS clock where *m* = 2 or 4.
- The BCH clock is used to run BCH-encoding blocks of the data packets.
- Pix clock, TMDS and BCH clocks are phase aligned.

The HDMI frame formatter accepts video data synchronous to the pix clock (on every rising edge). Audio data is fed through the S/PDIF interface. The STBus interface is used to configure and monitor its status. This interface is also used to write the AVI info and any auxiliary data to be sent on the HDMI during the data island period.

57.3 Video processor

The video processor formats the video data into HDMI-compatible frame format. The supported input video data formats are RGB, 8 bits per pixel and YCbCr 4:4:4, 8 bits per pixel. A typical application of the video processor is shown in Figure 231.

Figure 231: Block diagram of the video processor



The first pixel in a frame is counted from when Hsync and Vsync are active on a rising pix clock edge. The first pixel of the active video window is defined in registers HDMI_ACTIVE_XMIN and HDMI_ACTIVE_YMIN. The maximum number of pixels per line and maximum number of lines per frame are defined by the registers HDMI_ACTIVE_XMAX and HDMI_ACTIVE_YMAX.

The retimed Hsync and Vsync (HSync out and VSync out) are used by the other sub-blocks to derive timings.

The connection of video input bus to the individual channels is as follows.

- RGB, 8 bits per pixel
  - CHL0 in connected to B
  - CHL1 in connected to G
  - CHL2 in connected to R
- YCbCR 4:4:4, 8 bits per pixel
  - CHL0 in connected to Cb
  - CHL1 in connected to Y
  - CHL2 in connected to Cr

The control glue connects CHL0 in to CHL0 RT and CHL1 in to CHL1 RT.

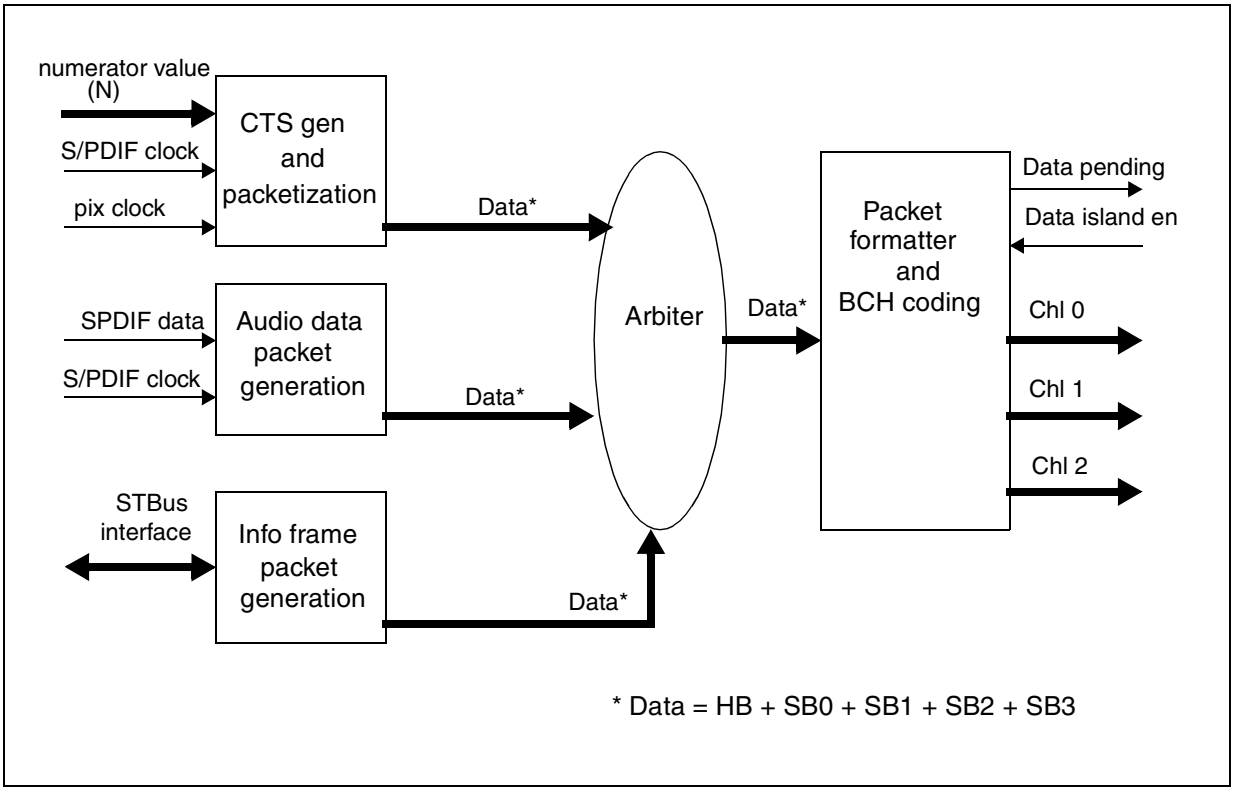
57.4 Data processor

The data processor formats and transmits data packets during data island periods. If there is no valid data, the data processor transmits a null packet. The data processor transmits the following data in the order specified during the data island period (subject to availability of the corresponding data packet):

- audio sample clock capture packet,
- audio data packet,
- general control packet,
- info frame packet,
- null packet.

Figure 232 shows the data processor block diagram.

Figure 232: Data processor



Null packets are sent in the data field when the encryption is enabled and HDMI is in not authenticated.

57.4.1 Audio sample clock capture

The CTS value generated by the HDMI source is used by the HDMI sync to regenerate the audio sample clock. The CTS is generated according to the equation

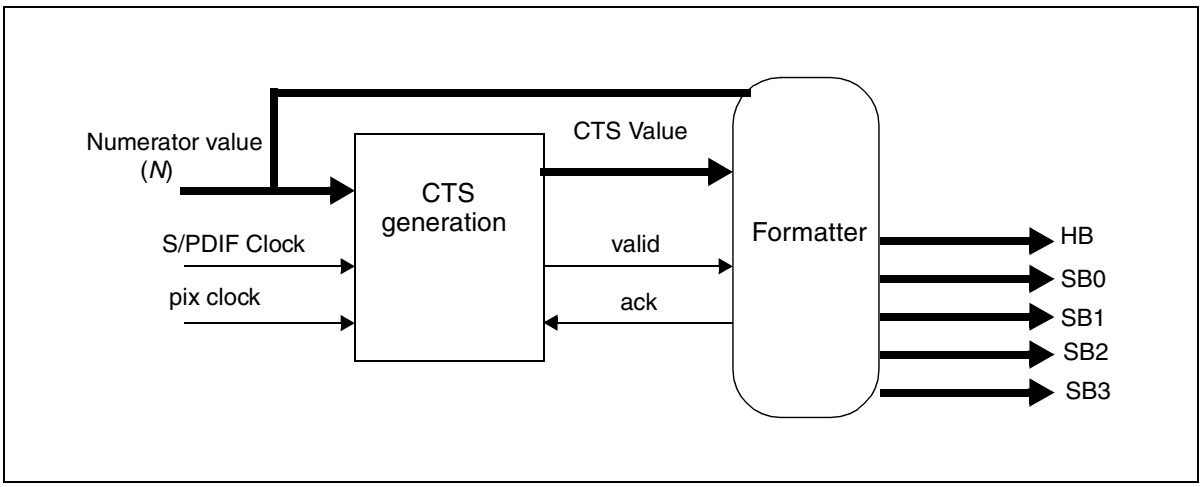
128 x F_S = pixel clock x *N*/CTS,

or

$$CTS = \frac{pixclock \times N}{128 \times Fs}$$

where F_S is the audio sample clock. In other words, CTS is generated by counting the number of pix clocks in (128 x Fs/*N*) duration. The CTS counter is a 20 bit counter, and is initialized to zero after every 128*F_S/*N* clocks. F_S and pix clock are treated as asynchronous; appropriate care is taken during synchronization.

Figure 233: CTS packet generation

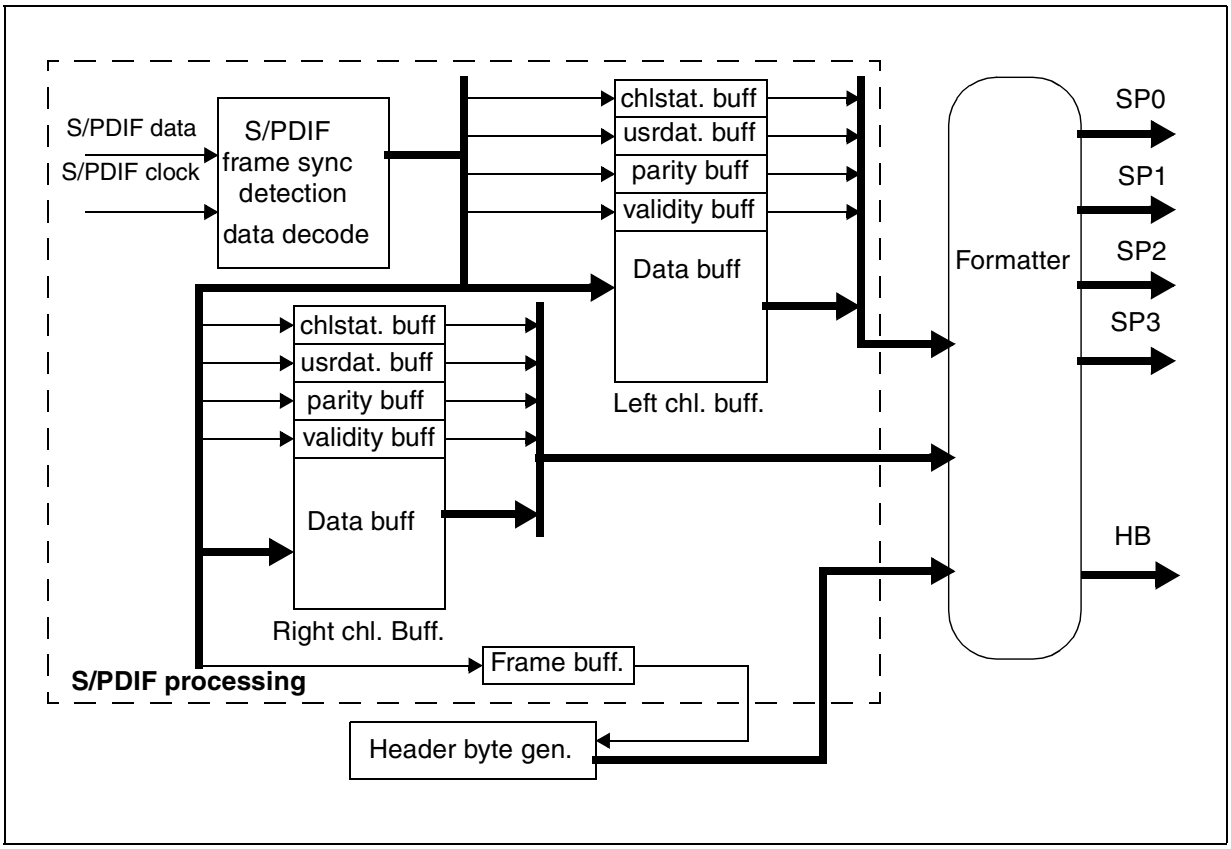


57.4.2 Audio data packetization

HDMI supports two audio channels, input on one S/PDIF input. The S/PDIF stream is synchronous to the S/PDIF clock, and is processed by the S/PDIF processor, the output of which is fed to the data formatter.

The S/PDIF processor consists of an S/PDIF frame sync detection and data decode block, FIFOs to store the received data, validity, parity, user data and channel status bits for both left and right channels.

Figure 234: Audio packet generation



57.4.3 General control packet

The general control packet is transmitted if the enable bit is set in register HDMI_GEN_CTRL_PKT_CFG. If the bit BUFF_NOT_REG is reset, the packet is formatted by hardware and the status in the bit AVMUTE is sent in the general control packet. If the bit BUFF_NOT_REG is 1, then the information in the info frame header byte register and info frame packet words is sent. This is done to cope with any future upgrades in the general control packet structure.

57.4.4 Info frame

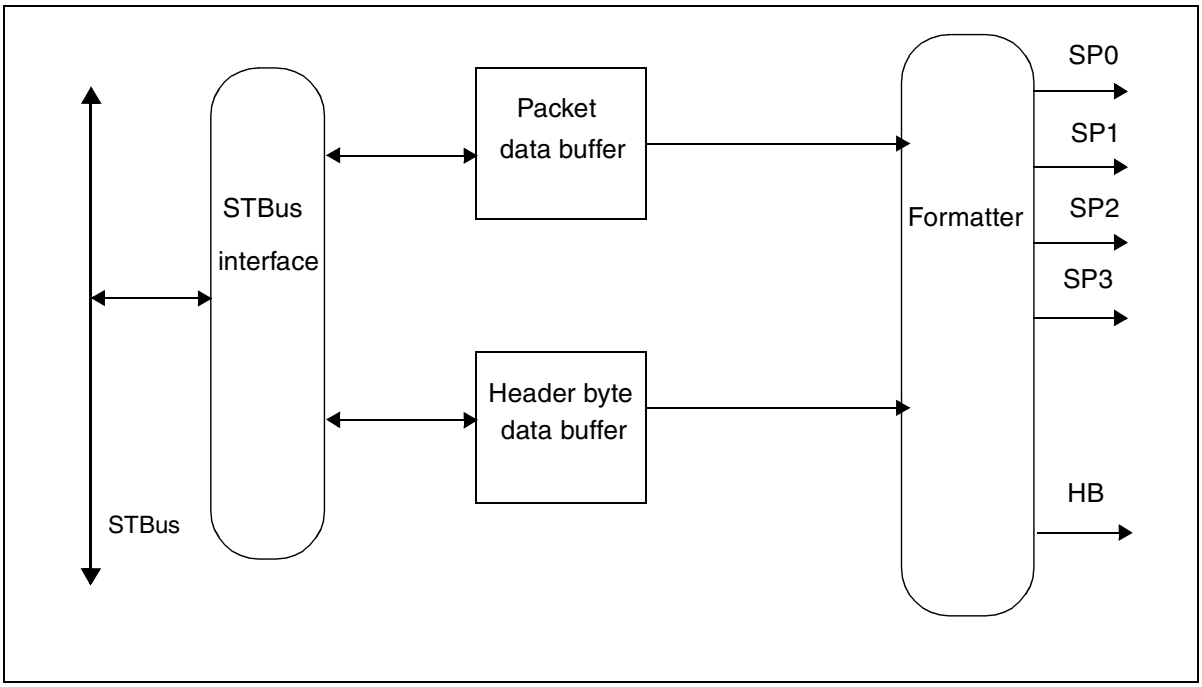
This module is responsible for sending the EIA/CEA-861B info frames on the HDMI interface. The data corresponding to the header bytes and the packet contents must be written into the appropriate buffers. The length of the info frame is inferred from the contents of the third byte of the header word. The total number of bytes transmitted is the contents of this location +1 (checksum).

The checksum is calculated by software and is written into the first byte of the packet word 0. The checksum should be calculated by the software such that the bytewise sum of all three bytes of header frame and all the valid data bytes of the infoframe packet and the checksum itself is equal to zero.

After writing the data into the header bytes word and packet words, the valid bit in the info frame configuration register is set. An interrupt is generated if enabled.

One info packet is transmitted every frame. The info packet can consist of multiple info frames depending on the sync. The maximum number of bytes that can be contained in an info packet is 255. This implies a **maximum** of nine to ten info frames in one frame. Since an interrupt is generated after transmission of each info frame, there is a maximum of ten interrupts every 16 ms.

Figure 235: Info packet generation



To reduce the dependency of the CPU, the HDMI_AVI_BUFF_EMPTY signal is bristled out, which can be used as pacing signal for DMA. The HDMI_AVI_BUFF_EMPTY is set when the data in the buffer is read by the HDMI formatter. It is reset when bit HDMI_IFRAME_CFG.EN is set.

57.4.5 Arbiter

The arbiter arbitrates between requests from different sources that wish to transmit data during the data island period of the HDMI frame. The arbiter follows a simple priority-based arbitration scheme, with CTS packets being given highest priority, followed by audio data packets, general control packets and info packet having the least priority. If there are no packet requests, the arbiter sends the data corresponding to the null packet to the packet formatter.

If the transmission of the general control packet is enabled, then the arbiter sends this packet only, between assertion of VSYNC and next 384 pixel clocks following the assertion of VSYNC. An interrupt is generated if enabled, notifying completion of general packet transmission.

57.4.6 Packet formatter and BCH coding

The data packet formatter is shown in the Figure 236 below. The module fetches the data, and performs BCH coding for the header packet and individual subpackets. It outputs data on the three 4-bit channels. If encryption is enabled and HDMI is authenticated, the four bits are encrypted with the keys from the HDCP. The data is then TERC encoded before being fed to the line driver.

The header byte and each of the subpacket bytes fetched are stored in an 8-bit flip-flop array. Each of these buffers has an BCH encoder associated with it. The BCH encoder associated with the header byte runs from the TMDS clock, while the BCH encoders associated with the subpacket buffers run at BCH clock frequency. The enable pulses are appropriately generated depending on whether the BCH clock is twice or four times the TMDS clock. The contents of the BCH encoder are loaded in the corresponding byte buffer (header byte or subpacket); the data is then given to the channel formatter.

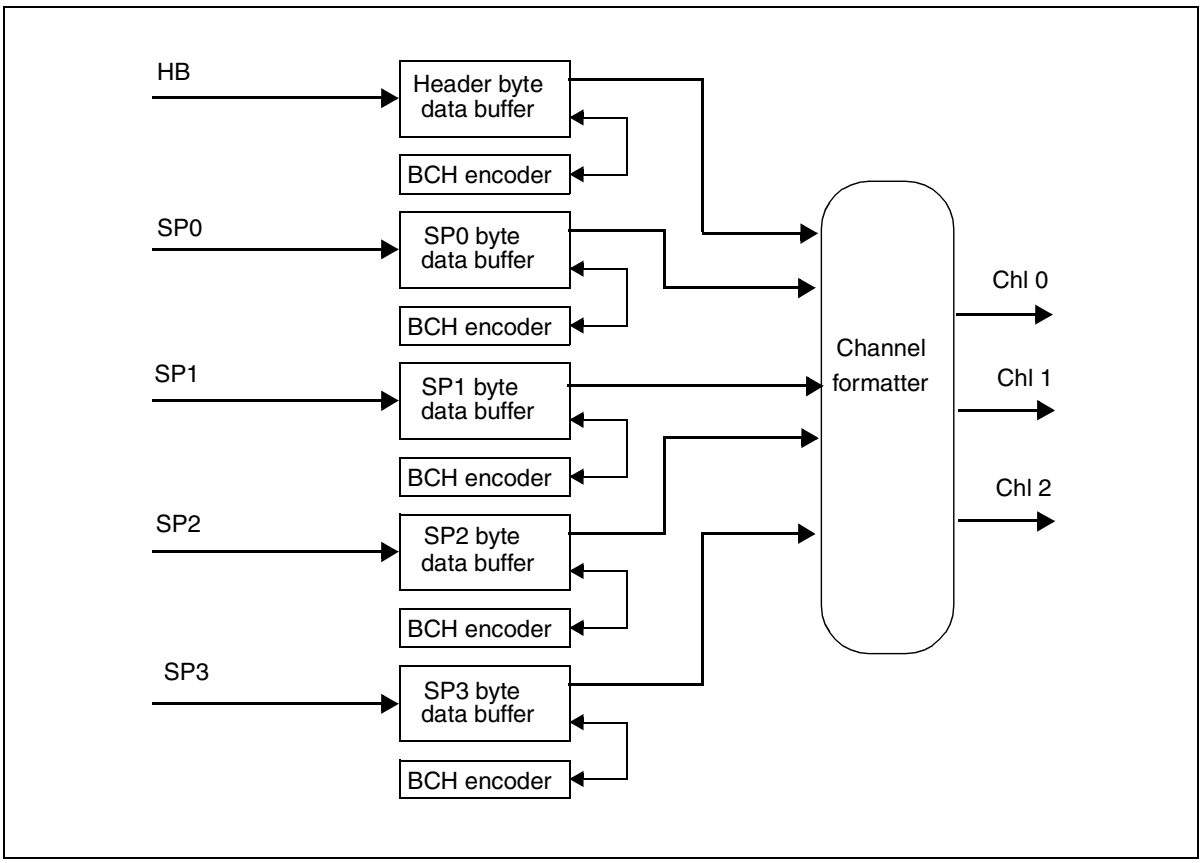
The channel formatter maps the individual bits of the of the data buffers on to the CHL0, CHL1 or CHL2 bus, as shown in Table 182 below. A new byte of the subpacket is transmitted during clock periods 5, 6, 7 and 8, shown by shaded rows.

Table 182: Byte buffer mapping

Clock period	CHL0 bits	CHL1 bits				CHL2 bits			
	4	1	2	3	4	1	2	3	4
1	HB0	SP0-0	SP1-0	SP2-0	SP3-0	SP0-1	SP1-1	SP2-1	SP3-1
2	HB1	SP0-2	SP1-2	SP2-2	SP3-2	SP0-3	SP1-3	SP2-3	SP3-3
3	HB2	SP0-4	SP1-4	SP2-4	SP3-4	SP0-5	SP1-5	SP2-5	SP3-5
4	HB3	SP0-6	SP1-6	SP2-6	SP3-6	SP0-7	SP1-7	SP2-7	SP3-7
5	HB4	SP0-0	SP1-0	SP2-0	SP3-0	SP0-1	SP1-1	SP2-1	SP3-1
6	HB5	SP0-2	SP1-2	SP2-2	SP3-2	SP0-3	SP1-3	SP2-3	SP3-3
7	HB6	SP0-4	SP1-4	SP2-4	SP3-4	SP0-5	SP1-5	SP2-5	SP3-5
8	HB7	SP0-6	SP1-6	SP2-6	SP3-6	SP0-7	SP1-7	SP2-7	SP3-7

Figure 236 shows the data packet generator.

Figure 236: Data packet generation

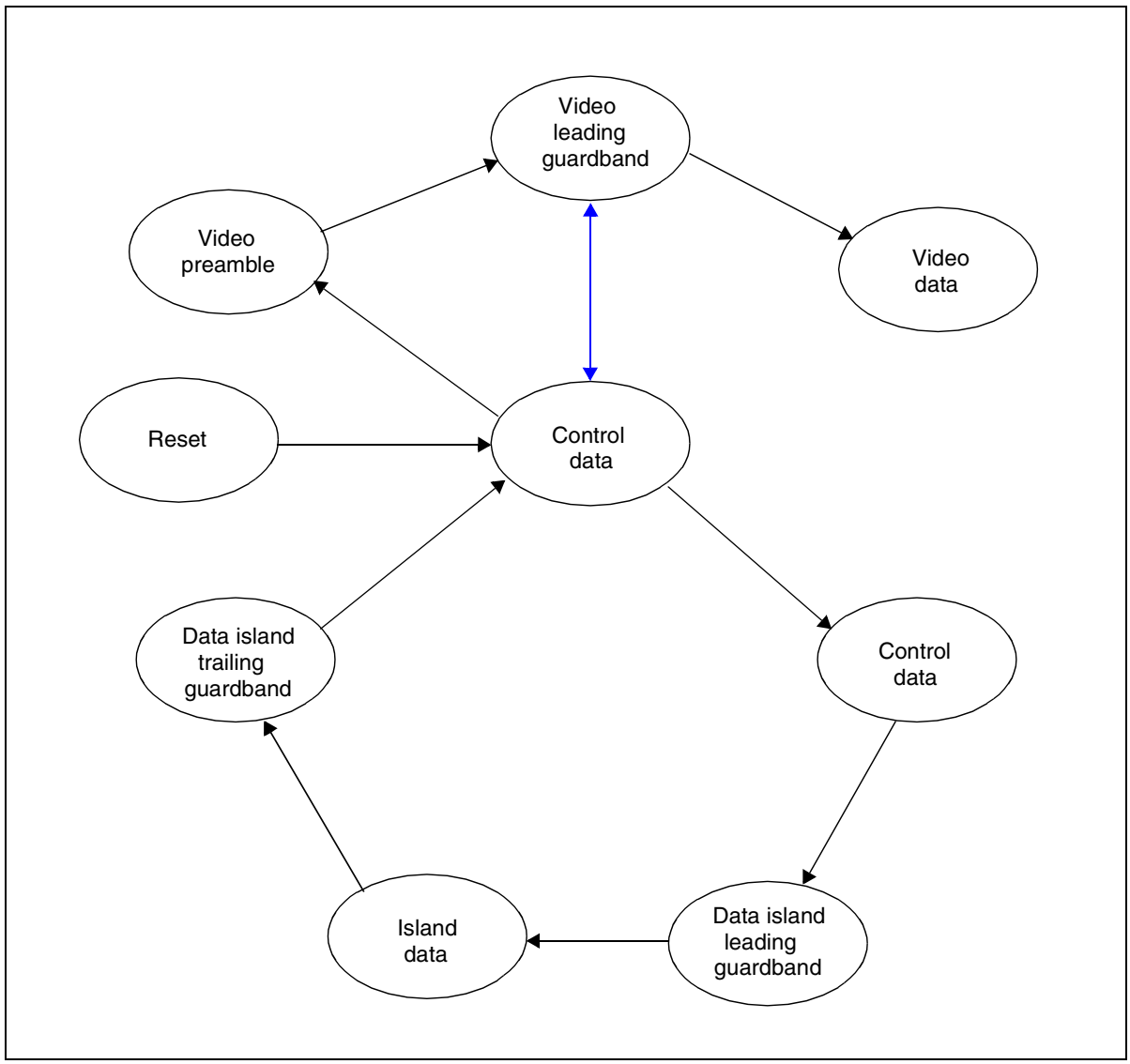


57.5 Frame formatting and line coding

57.5.1 Frame formatting

The HDMI frame is generated using a state machine as shown in Figure 237.

Figure 237: HDMI Framer FSM



On reset, the HDMI interface **does not** generate any frames. It stays in the reset state until bit HDMI_CFG.DEVICE_EN is set. The state machine moves out of the reset state only at the beginning of the frame (first pixel).

If bit HDMI_CFG.HDMI_NOT_DVI is not set, then the HDMI interface is configured as a digital visual interface (DVI), and toggles between control data and video data states only (shown by the blue line).

The extended control periods are generated as programmed in HDMI mode.

Control data variations for DVI

All the control pins CTL0, CTL1, CTL2 and CTL3 are held to logic 0 internally in DVI mode. Legacy applications where the CTL0 is toggled at even to odd edges are **not supported**.

Control data during original encryption status signalling (OESS)

OESS is used **only** in DVI, and uses only CTL3 to indicate that a fame is being encrypted. This signalling is only used when the HDMI is in the authenticated state. CTL3 must be asserted to 1 for **at least eight pixel clock periods**, starting no closer than 128 pixel clock periods from the end of vertical blank. That is, CTL3 must to asserted for at least eight pixel clocks after 128 pixel clocks after the end of VSYNC.

Control data during enhanced encryption status signalling (EESS)

EESS utilizes all four control signals. The ESS can be used in both DVI and HDMI modes. The possible patterns for control bits are shown in the Table 183.

Table 183: EESS

CTL3	CTL2	CTL1	CTL0	Encryption for the frame
1	0	0	1	Enabled
0	0	0	1	Disabled

The CTL*x* signals as listed in the table above have to be **asserted during a 16 clock window of opportunity starting at 512 pixel clocks**. That is, encryption enable CTL*x* should be asserted for 16 clock periods starting from 512 pixel clock periods after assertion of VSYNC. If the HDCP is enabled, the HDMI framer has to ensure that no data island, video data or any guard band is transmitted during a **keep out period** that starts 508 pixels after the active edge of the VSync, and ends 650 pixels past active edge of VSYNC, to facilitate frame key calculation. **No data island is allowed 58 pixel clocks** following fall of video data, to facilitate line key calculation.

The EESS is used only when the HDMI transmitter is in the authenticated state.

57.5.2 Line coding

The HDMI uses different encoding techniques during different states of the frame. The encoding schemes employed during each of the frame states are documented in the following sections.

Data control (preamble)

During the data island period, HSYNC, VSYNC and packet header bits are mapped onto bits 0, 1 and 2 respectively. Channel 1 carries CTL0 and CTL1 on bits 0 and 1, while channel 2 carries CTL2 and CTL3 on bits 0 and 1. The two bits are coded into 10 bits as shown in Table 184.

Table 184: Coding of control data

Bit 1	Bit 0	TMDS output [9:0]
0	0	0b11 0101 0100
0	1	0b00 1010 1011
1	0	0b01 0101 0100
1	1	0b10 1010 1011

Leading guard band data

During the leading guard band, HSYNC, VSYNC, 1 and 1 are mapped on to bits 0, 1, 2 and 3 respectively. These four bits are TERC4 encoded. Channel 1 carries binary data Q_OUT[9:0] = 0b01 0011 0011, and channel 2 carries binary data Q_OUT[9:0] = 0b01 0011 0011.

Data island

During the data island period, HSYNC, VSYNC and packet header bits are mapped onto bits 0, 1 and 2 respectively. Logic 0 is sent on bit 3 for the first clock, and logic 1 thereafter. The LSB 4 bits of channel 1 and channel 2 carry packet data. The four bits are TERC4 encoded as shown in Table 185.

Table 185: TERC4 encoding

Input	TERC4 Data [9:0]	Input	TERC4 Data [9:0]
0b 0000	0b10 1001 1100	0b 1000	0b10 1100 1100
0b 0001	0b10 0110 0011	0b 1001	0b01 0011 1001
0b 0010	0b10 1110 0100	0b 1010	0b01 1001 1100
0b 0011	0b10 1110 0010	0b 1011	0b10 1100 0110
0b 0100	0b01 0111 0001	0b 1100	0b10 1000 1110
0b 0101	0b01 0001 1110	0b 1101	0b10 0111 0001
0b 0110	0b01 1000 1110	0b 1110	0b01 0110 0011
0b 0111	0b01 0011 1100	0b 1111	0b10 1100 0011

Trailing guard band data

The mapping of the data pins during the trailing guard band is exactly same as leading guard band data.

Video control (preamble)

During the video control period, channel 0 carries HSYNC and VSYNC on bits 0 and 1 respectively. Channel 1 carries CTL0 and CTL1 on bits 0 and 1 respectively, while channel 2 carries CTL2 and CTL3 on bits 0 and 1 respectively. These bits are coded as shown in Table 184: *Coding of control data*. The value of the control bits in video preamble are as follows CTL0 = 1, CTL1 = 0, CTL2 = 0, and CTL3 = 0.

Leading guard band video

During the video leading guard band, channel 0 carries Q_OUT[9:0] = 0b10 1100 1100, channel 1 carries Q_OUT[9:0] = 0b01 0011 0011, and channel 2 carries Q_OUT[9:0] = 0b10 1100 1100.

Video data

During the video data period, the 8-bit input data is encoded in 10 bits, which has an approximate DC balance, as well as reduced transitions. This is done in two stages. In first stage, 9-bit transition- minimized data is produced. In the second stage, a 10th bit is added, which balances the overall DC.

57.6 HDCP interface

The HDCP block outputs keys on a 24-bit data bus. The keys are XORed with the data just before TMDS or TERC4 encoding. The 24 bits from HDCP are XORed with the CHL0, CHL1 and CHL2 data buses as shown in the table below. The LSB bit of the key is XORed with the corresponding LSB bit of the channel.

Table 186: Encryption stream mapping for video data

HDCP key in	TMDS channel
23:16	CHL2
15:8	CHL1
7:0	CHL0

Mapping of the encryption keys during the data island period is shown in Table 187.

Table 187: Encryption stream mapping for data island

HDCP key in	TERC4 bits
23:20	Unused
19	Channel 2 bit 3
18	Channel 2 bit 2
17	Channel 2 bit 1
16	Channel 2 bit 0
15:12	Unused
11	Channel 1 bit 3

HDCP key in	TERC4 bits
10	Channel 1 bit 2
9	Channel 1 bit 1
8	Channel 1 bit 0
7:3	Unused
2	Channel 0 bit 2
1:0	Unused

57.6.1 Authenticated state

When the HDMI is in the authenticated state, the HDCP module asserts signal ENCRYPTION_EN to logic 1. The HDMI sends the valid video and audio data if encryption is enabled (bit HDMI_CFG.HDCP_EN is set) and the HDMI is in authenticated state. If the HDMI is not in the authenticated state, a default value is sent (see registers HDMI_ACTIVE_VID_YMIN, HDMI_ACTIVE_VID_YMAX and HDMI_DEFAULT_CHL0_DATA). If encryption is not enabled, the valid video and audio data are sent.

57.6.2 Enhanced ri computation mode support

For enhanced ri computation, it is necessary to capture the first pixel of the active window. To facilitate this, the HDCP sets the signal RI_PIXEL_CAPTURE to high for one pix clock period when VSYNC is asserted. On detection of the signal, the HDMI captures the pixel values. Registers HDMI_ACTIVE_VID_XMIN and HDMI_ACTIVE_VID_YMIB set the location of the pixel value to be captured. The captured pixels can be read from register HDMI_CHL0_CAP_DATA. When the input video format is RGB, channel 0 data corresponds to blue, channel 1 to green and channel 2 to red. An interrupt is generated if enabled.

57.7 Hot plug detect

The HDMI sink asserts a high when E-EDID (enhanced extended display identification data) is available. The sink asserts this line only after detection of the +5 V line. The source uses hot plug to initiate reading E-EDID data. The sink can drive the hot plug detect to low for at least 100 ms to indicate a change in E-DID data. An input pin is provided to detect any changes in the hot plug status. An interrupt is generated on change of the input pin logic level, if enabled.

57.8 Clocks

See Table 69: *Video displays clocking for various application on page 129*.

57.9 Reset

The HDMI frame formatter is reset by the global asynchronous reset. The TMDS interface becomes active only after bit HDMI_CFG.DEVICE_EN is set.

57.10 Info frame programming

To program an info frame:

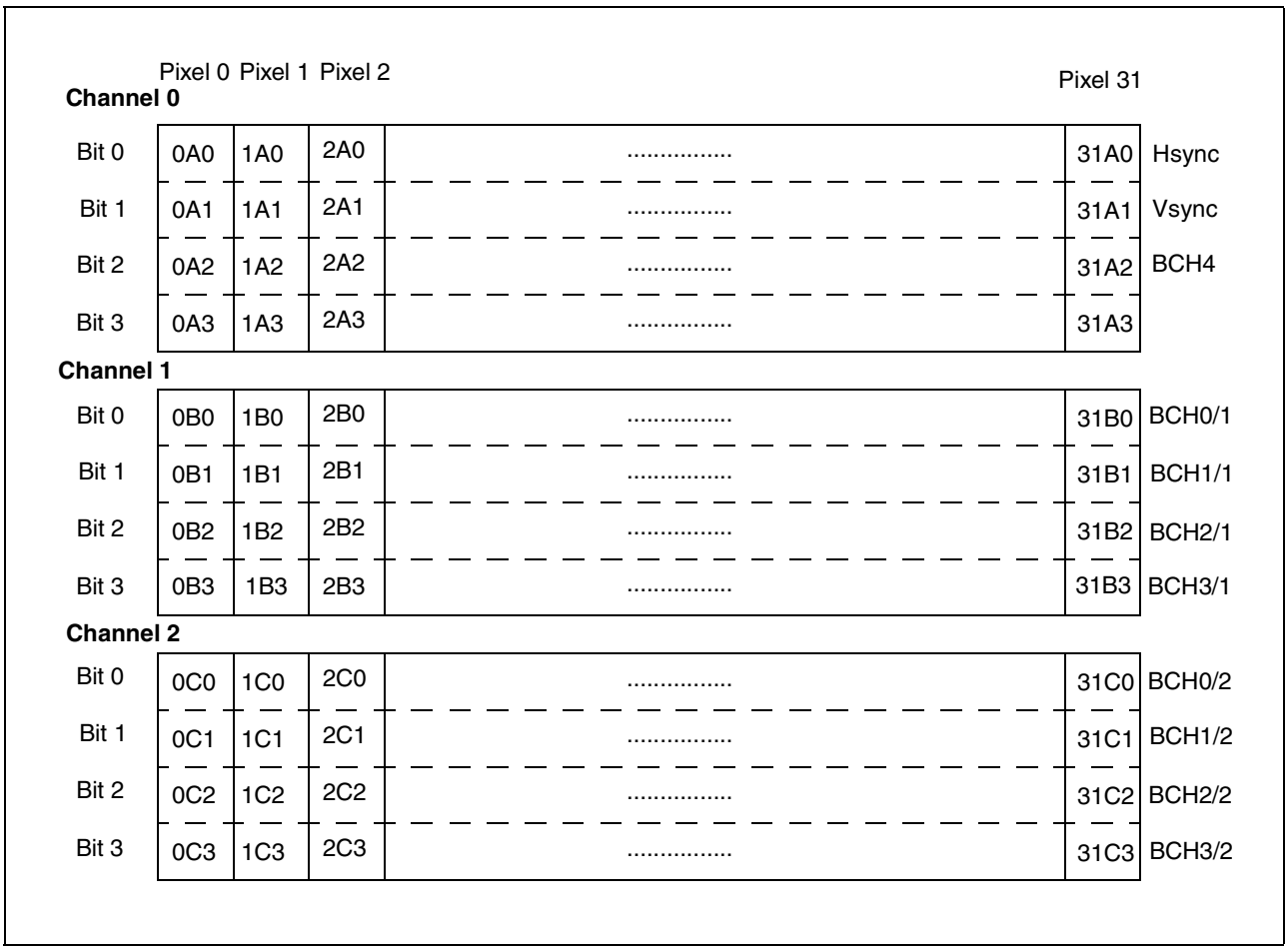
1. Check HDMI_STA.INFO_BUFF_STA is set to 1.
   Continue only if this bit is set, otherwise the processor has to wait till this bit is set by the HDMI block.
2. Assemble header bytes as a word (with LSB containing HB0) and write into register HDMI_IFRAME_HEAD_WD.
3. Assemble packet bytes as words and write to registers HDMI_IFRAME_PKT_WD[0 - 6].
   This data is transferred to the FIFO.
4. Enable the info frame by setting bit HDMI_IFRAME_CFG.EN.
   The HDMI block automatically infers the number of bytes to be transmitted from the lower nibble of the third byte in register HDMI_IFRAME_HEAD_WD and starts transmitting the info frame.
5. HDMI_IFRAME_FIFO_STA.BUSY is set once the HDMI cell starts transmitting the info frame. It simultaneously resets HDMI_IFRAME_CFG.EN. The BUSY bit is reset only after the complete info frame is transmitted.
6. HDMI_STA.INFO_BUFF_STA is set once all the data is read by the HDMI cell, not necessary transmitted.
7. An interrupt is generated (if enabled) once the info frame buffer is empty.
8. The interrupt can be cleared by writing 1 into bit HDMI_INT_CLR.CLR_IFRAME_INT.

One info packet is sent each frame. One info packet may contain multiple info frames. A maximum of 255 bytes may be carried in each info packet. This implies a maximum of 10 info frames. To facilitate transmission of an info packet every new frame, an interrupt can be generated (if enabled) on the start of a new frame. Assuming a typical frame rate of 60 Hz, transmission of 10 info frames in one frame interval implies servicing 10 interrupts every 16 ms, or one interrupt every 1.5 ms (maximum).

57.11 Data island definitions

57.11.1 Data island packet construction

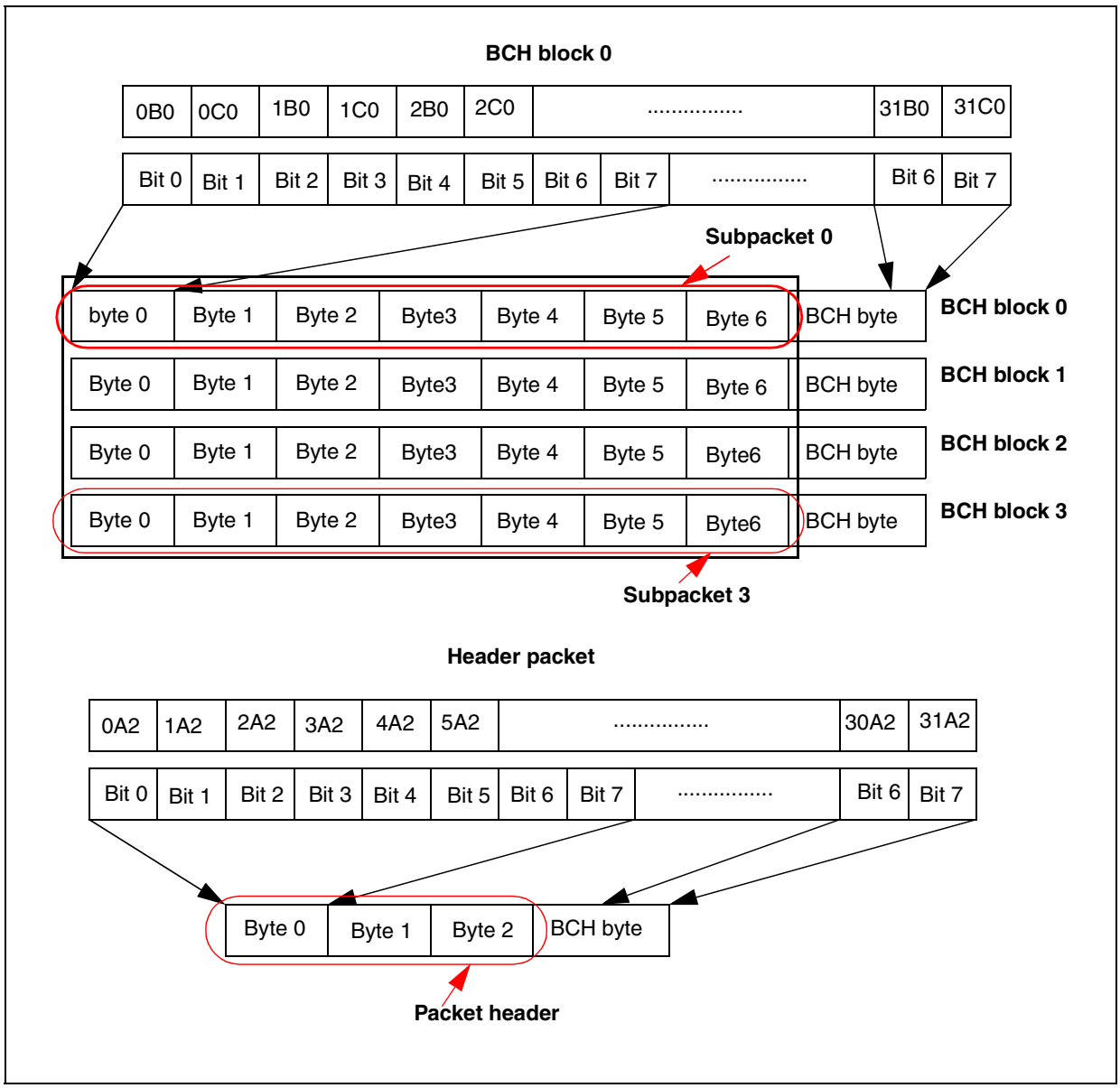
Figure 238: BCH blocks during data island



Insertion of different BCH blocks during the data island period is shown in the figure above. BCH block 0 is constructed by clubbing BCH0/1,BCH0/2 blocks together. Similarly BCH block 1,2, and 3 are constructed by clubbing BCH1/1, BCH1/2; BCH 2/1, BCH 2/2 and BCH3/1,BCH 3/2 blocks respectively.

The structure of BCH4 is also shown in Figure 239.

Figure 239: Formation of BCH blocks



57.11.2 Data island packet definitions

The packet during the data island consists of two sections, one the packet header and the other the packet body. The packet header is three bytes long, with one byte of BCH code. The packet header specifies the current type of packet. The first byte of the packet header specifies packet type, the next two bytes contain the packet specific data.

The packet body consists of four subpackets. Each subpacket is seven bytes long with one BCH byte. The data payload of the packet is carried in the packet body.

The packet header and the packet body for each of the packets types are described in the following sections.

57.11.2.1Null packet

All the three bytes of the header packet and all the bytes of each of the subpackets of a null packet contain 0x00.

57.11.2.2 Audio clock regeneration packet

The contents of the header bytes of the audio clock regeneration packet are shown in Table 188.

Table 188: Header bytes of audio clock regeneration packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
HB0	0	0	0	0	0	0	0	1
HB1	0	0	0	0	0	0	0	0
HB2	0	0	0	0	0	0	0	0

All the four subpackets contain the same information. The contents of the subpacket are shown in Table 189.

Table 189: Subpacket bytes of audio clock regeneration packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
SB0	0	0	0	0	0	0	0	0
SB1	0	0	0	0	CTS_19	CTS_18	CTS_17	CTS_16
SB2	CTS_15	CTS_14	CTS_13	CTS_12	CTS_11	CTS_10	CTS_9	CTS_8
SB3	CTS_7	CTS_6	CTS_5	CTS_4	CTS_3	CTS_2	CTS_1	CTS_0
SB4	0	0	0	0	N_19	N_18	N_17	N_16
SB5	N_15	N_14	N_13	N_12	N_11	N_10	N_9	N_8
SB6	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0

57.11.2.3 Audio sample packet

The contents of the header bytes of the audio sample packet are shown in Table 190.

Table 190: Header bytes of audio sample packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
HB0	0	0	0	0	0	0	1	0
HB1	0	0	0	layout	sample present.sp3	sample present.sp2	sample present.sp1	sample present.sp0
HB2	B.3	B.2	B.1	B.0	sample flat.sp3	sample flat.sp2	sample flat.sp1	sample flat.sp0

layout: Indicates which of two possible audio sample packet layouts are used.

sample present.sp*X*: indicates if subpacket *X* contains an audio sample

sample flat.sp*X*: Bit is set if no useful audio data is available. Valid only if sample present.sp*X* is set.

B.*X*: is set if subpacket *X* contains the first frame of the IEC60958 block.

Table 191: Subpacket bytes of audio sample packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
SB0	L.11	L.10	L.9	L.8	L.7	L.6	L.5	L.4
SB1	L.19	L.18	L.17	L.16	L.15	L.14	L.13	L.12
SB2	L.27	L.26	L.25	L.24	L.23	L.22	L.21	L.20
SB3	R.11	R.10	R.9	R.8	R.7	R.6	R.5	R.4
SB4	R.19	R.18	R.17	R.16	R.15	R.14	R.13	R.12
SB5	R.27	R.26	R.25	R.24	R.23	R.22	R.21	R.20
SB6	P_R	C_R	U_R	V_R	P_L	C_L	U_L	V_L

The possible layouts are listed in Table 192.

Table 192: Layouts for audio sample packets

Layout value	Max num channels	Samples	Subpkt 0	Subpkt 1	Subpkt 2	Subpkt 3
0	2	4	Channels 1, 2, sample 0	Channels 1, 2, sample 1	Channels 1, 2, sample 2	Channels 1, 2, sample 3
1	8	1	Channels 1, 2, sample 0	Channels 3, 4, sample 0	Channels 5, 6, sample 0	Channels 7, 8, sample 0

57.11.2.4 Info frame packet

The contents of the header bytes of the info packet are shown in Table 193.

Table 193: Header bytes of info packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
HB0	1	INFOFRAME_TYPE						
HB1	INFOFRAME_VERSION							
HB2	0	0	0	INFOFRAME_LENGTH				

The bytes in the packet body are shown in Table 194.

Table 194: Subpacket bytes of info packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
PB0	Checksum							
PB1	Data byte 1							
PB2	Data byte 2							
PB3 - PB26	.....							
PB27	Data byte 27							

The packet bytes are mapped onto individual subpackets as shown below.

- Subpacket0 SB0 to SB6 to PB0 to PB6.
- Subpacket1 SB0 to SB6 to PB7 to PB13.
- Subpacket2 SB0 to SB6 to PB14 to PB20.
- Subpacket3 SB0 to SB6 to PB21 to PB27.

57.11.2.5General control packet

The general control packet is transmitted only between an active edge of Vsync and 384 pixels following the active edge.

The contents of the header bytes of the general control packet are shown in the Table 195.

All four subpackets are identical. The contents of the subpackets are shown in Table 196.

Table 195: Header bytes of general control packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
HB0	0	0	0	0	0	0	1	1
HB1	0	0	0	0	0	0	0	0
HB2	0	0	0	0	0	0	0	0

Table 196: Subpacket bytes of general control packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
SB0	0	0	0	CLR_AVMUTE	0	0	0	SET_AVMUTE
SB1	0	0	0	0	0	0	0	0
SB2	0	0	0	0	0	0	0	0
SB3	0	0	0	0	0	0	0	0
SB4	0	0	0	0	0	0	0	0
SB5	0	0	0	0	0	0	0	0
SB6	0	0	0	0	0	0	0	0

58 HDMI registers

Register addresses are provided as *HDMIBaseAddress* + offset.

The *HDMIBaseAddress* is:

0x1900 5800.

The register address space is non-contiguous, to facilitate addition of new registers. Writes to read-only registers have no effect on the contents of these registers and reads from write only registers return all 0s.

The HDMI Frame formatter occupies an address space of 1 KBytes, 0x000 to 0x3FF. It is expected that a response request is generated for any requests within the 1 K address map.

Table 197: Register summary table

Register name	Description	Offset	Type
HDMI_CFG	Configuration	0x000	R/W
HDMI_INT_EN	Interrupt enable	0x004	
HDMI_INT_STA	Interrupt status	0x008	RO
HDMI_INT_CLR	Interrupt clear	0x00C	WO
HDMI_STA	Status	0x010	RO
HDMI_EXTS_MAX_DLY	Max. time between extended control periods	0x014	R/W
HDMI_EXTS_MIN	Duration of extended control period	0x018	
Reserved	-	0x01C - 0x0FC	
HDMI_ACTIVE_VID_XMIN	Active video window co-ordinates	0x100	
HDMI_ACTIVE_VID_XMAX		0x104	
HDMI_ACTIVE_VID_YMIN		0x108	
HDMI_ACTIVE_VID_YMAX		0x10C	
HDMI_DEFAULT_CHL0_DATA	Channel data to be sent when the HDCP is not authenticated	0x110	
HDMI_DEFAULT_CHL1_DATA		0x114	
HDMI_DEFAULT_CHL2_DATA		0x118	
HDMI_CHL0_CAP_DATA	Captured data	0x11C	RO
Reserved	-	0x120 - 0x1FC	-
HDMI_AUD_CFG	Audio configuration	0x200	R/W
Reserved	-	0x204 - 0x20C	-
HDMI_IFRAME_HEAD_WD	Header word of AVI info frame	0x210	R/W
HDMI_IFRAME_PKT_WDn	Info frame packet data word 0 - 6	0x214 - 0x22C	
HDMI_IFRAME_CFG	Configure the info frame	0x230	
Reserved	-	0x234 - 0x23C	-
HDMI_IFRAME_FIFO_STA	Info frame FIFO status	0x240	R/W
HDMI_SAMPLE_FLAT_MASK	Sample flat mask	0x244	
HDMI_GEN_CTRL_PKT_CFG	Configuration for general control packet	0x248	R/W
Reserved	-	0x24C - 0xFFC	-

See also Chapter 54: *Video output stage (VOS) registers*, *HDMI_PHY_LCK_STA* *on page 642*.

HDMI_CFG HDMI configuration

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_RST_EN	Reserved																		BCH_CLK_RAT	DLL_CFG		PIXEL_RPT	Reserved				SYNC_POL	ESS_NOT_OESS	HDCP_EN	HDMI_NOT_DVI	HDMI_EN

Address: *HDMIBaseAddress* + 0x000

Type: R/W

Reset: 0

Description: The register contents like DEVICE_EN, HDMI_NOT_DVI, HDCP_EN, ESS_NOT_OESS, 422_EN, and PIXEL_RPT are expected to take effect from the new frame onwards.

[31] **SW_RST_EN**
To enable software reset, bit SW_RST must be set to 1. This forces the state machines to reset state (except the STBus FSM) and re-initialize FIFOs (resets write and read pointers only). The soft reset execution is synchronous to appropriate clocks in which the soft reset has to be executed. This bit has to be re-set to 0 after completion of software reset.
1: Soft reset enable

[30:13] **Reserved**

[12] **BCH_CLK_RAT**: BCH clock ratio
0: BCH clock = 2 x TMDS clock 1: BCH clock = 4 x TMDS clock

[11:10] **DLL_CFG**: Configure relationship between DLL clock and TMDS clock

[9] **PIXEL_RPT**
1: Enable pixel repetition by 2

[8:5] **Reserved**
Set to 0

[4] **SYNC_POL**: Sync polarity
0: HSYNC and VSYNC active high 1: HSYNC and VSYNC active low

[3] **ESS_NOT_OESS**
This bit is valid only if HDCP_EN is set to 1.
0: OESS Enable 1: ESS Enable

[2] **HDCP_EN**
1: Enable HDCP interface

[1] **HDMI_NOT_DVI**
0: DVI compatible frame generation 1: Enable HDMI compatible frame generation

[0] **HDMI_EN**
1: Enable HDMI

HDMI_INT_EN HDMI interrupt enable

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							SPDIF_FIFO_OVRUN_INT_EN	GENCTRL_PKT_INT_EN	NEW_FRM_INT_EN	DLL_LCK_INT_EN	HOT_PLUG_INT_EN	PIX_CAP_INT_EN	INFO_FRAME_INT_EN	SW_RST_INT_EN	INT_EN

Address: *HDMIBaseAddress* + 0x004
Type: R/W
Reset: 0
Description:

[30:9] **Reserved**

[8] **SPDIF_FIFO_OVRUN_INT_EN**
1: Enable interrupt on S/PDIF FIFO overrun

[7] **GENCTRL_PKT_INT_EN**
1: Enable interrupt on general control packet

[6] **NEW_FRM_INT_EN**
1: Enable interrupt on new frame

[5] **DLL_LCK_INT_EN**
1: Enable interrupt on DLL lock

[4] **HOT_PLUG_INT_EN**
1: Enable interrupt on change in hot_plug_in logic toggle

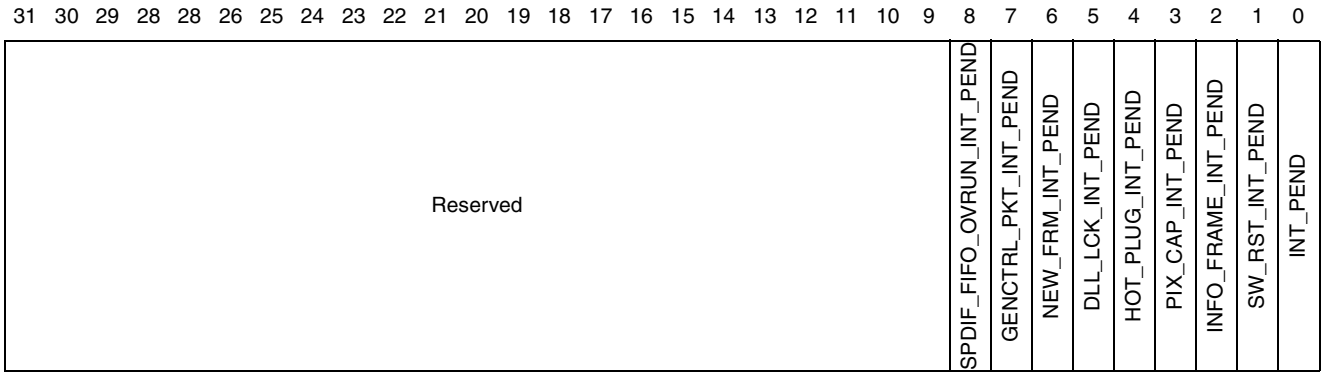
[3] **PIX_CAP_INT_EN**
1: Enable interrupt after capture of first active pixel

[2] **INFO_FRAME_INT_EN**
1: Enable interrupt after info frame transmission

[1] **SW_RST_INT_EN**
1: Enable interrupt after soft reset completion

[0] **INT_EN**
1: Global interrupt enable

HDMI_INT_STA Interrupt status



Address: *HDMIBaseAddress* + 0x008

Type: RO

Reset: 0

Description: All the bits in the interrupt status register are set only when the corresponding bit in the interrupt enable register is set and bit 0 of the interrupt enable register is set. The bit in the interrupt status can be cleared by performing a clear bit (write with the bit set to 1) operation at the corresponding location in interrupt clear register.

[31:9] **Reserved**

[8] **SPDIF_FIFO_OVRUN_INT_PEND**
1: Interrupt on S/PDIF FIFO overrun pending

[7] **GENCTRL_PKT_INT_PEND**
1: Interrupt on general control packet pending

[6] **NEW_FRM_INT_PEND**
1: Interrupt on new frame pending

[5] **DLL_LCK_INT_PEND**
1: Interrupt on DLL lock pending

[4] **HOT_PLUG_INT_PEND:** Hot plug interrupt pending
1: Any change in logic state of hot_plug_in signal is detected.

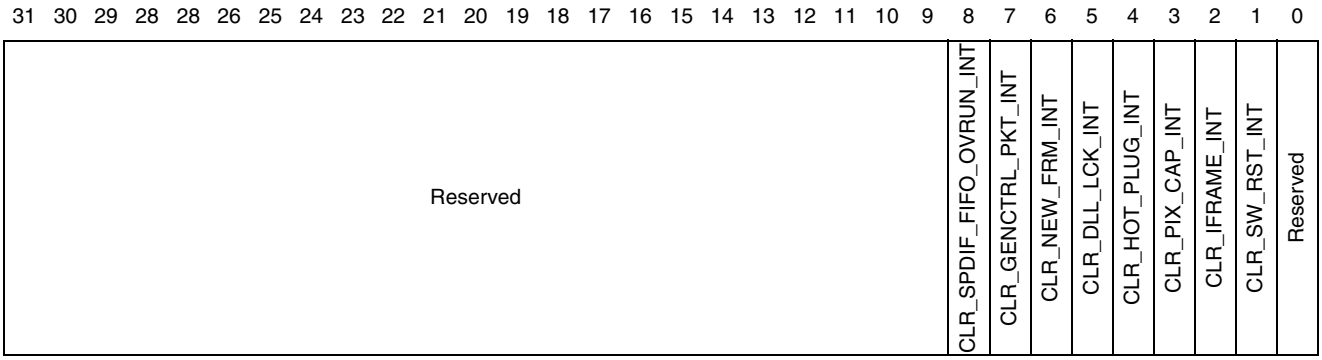
[3] **PIX_CAP_INT_PEND**
1: Pixel capture interrupt pending - set when the pixel data is captured.

[2] **INFO_FRAME_INT_PEND**
1: Info frame transmission interrupt pending
Set when the info frame buffer is empty

[1] **SW_RST_INT_PEND**
1: Soft reset completion interrupt pending - set when the software reset process is complete in all the clock domains. Once set it can be reset only clearbit operation.
Clearing this bit does not de-assert sw reset. If this bit is kept asserted, recursive soft reset interrupts should not occur.

[0] **INT_PEND**: Global interrupt pending
1: Global interrupt pending - any interrupt status bit is set to 1. Cannot be cleared by clear bit operation.

HDMI_INT_CLR Interrupt clear



Address: *HDMIBaseAddress* + 0x00C
Type: WO
Reset: 0
Description:

[31:9] **Reserved**

[8] **CLR_SPDIF_FIFO_OVRUN_INT**
1: Clear interrupt on S/PDIF FIFO overrun

[7] **CLR_GENCTRL_PKT_INT**
1: Clear interrupt on general control packet

[6] **CLR_NEW_FRM_INT**
1: Clear interrupt on new frame

[5] **CLR_DLL_LCK_INT**
1: Clear interrupt on DLL lock

[4] **CLR_HOT_PLUG_INT**
1: Clear hot plug interrupt

[3] **CLR_PIX_CAP_INT**
1: Clear pixel capture interrupt

[2] **CLR_IFRAME_INT**
1: Clear info frame transmission interrupt

[1] **CLR_SW_RST_INT**
1: Clear soft reset interrupt

[0] **Reserved**

HDMI_STA Status

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										DLL_LCK	HOT_PLUG_STS	PIX_CAP_STA	INFO_BUFF_STA	SW_RST_STA	Reserved

Address: *HDMIBaseAddress* + 0x010
Type: RO
Reset: 0
Description:

[31:6] **Reserved**

[5] **DLL_LCK**: DLL lock

[4] **HOT_PLUG_STS**: Logic level on hot plug in line

[3] **PIX_CAP_STA**
1: After pixel is stored and ri calculation enabled, is reset after reading the pixel contents.

[2] **INFO_BUFF_STA**
1: Info buffer empty.

[1] **SW_RST_STA**
1: Soft reset complete, Bit is reset when bit HDMI_CFG.SW_RST is reset.

[0] **Reserved**

HDMI_EXTS_MAX_DLY Maximum time between extended control periods

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												MAX_DLY			

Address: *HDMIBaseAddress* + 0x014
Type: R/W
Reset: 0
Description: This register holds the maximum time between extended control periods in number of VSYNC occurrences. It is expected that the maximum time between the extended control periods will not exceed 50 ms. For a frame rate of 50 or 60 Hz, the extended control period has to be sent every two frames.

0000: 16 Vsyncs
0001: 1 Vsync

Default value to be written: 0010

HDMI_EXTS_MIN Minimum duration of extended control periods

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								EXTS_MIN							

Address: *HDMIBaseAddress* + 0x018
Type: R/W
Reset: 0
Description: Specifies the minimum duration of extended control periods in TMDS clocks.
0: 256 clocks
Default value to be written: 0x32

HDMI_ACTIVE_VID_XMIN Xmin active video

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			XMIN												

Address: *HDMIBaseAddress* + 0x100
Type: R/W
Reset: 0
Description: The video data is considered active only for those pixels whose line number is greater than or equal to the contents of HDMI_ACTIVE_VID_YMIN and whose pixel number is greater than or equal to XMIN.

[31:13] **Reserved**

[12:0] **XMIN:** Xmin value in pix clocks.

HDMI_ACTIVE_VID_XMAX Xmax active video

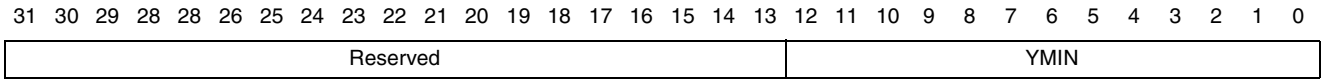
31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			XMAX												

Address: *HDMIBaseAddress* + 0x104
Type: R/W
Reset: 0
Description: Maximum number of pixels per line.

[31:13] **Reserved**

[12:0] **XMAX**: xmax value in pix clocks.

HDMI_ACTIVE_VID_YMIN Ymin active video

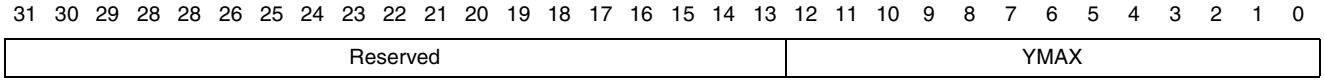


Address: *HDMIBaseAddress* + 0x108
Type: R/W
Reset: 0
Description: Line from which an active video starts.

[31:13] **Reserved**

[12:0] **YMIN**: ymin value in pix clocks.

HDMI_ACTIVE_VID_YMAX Ymax active video



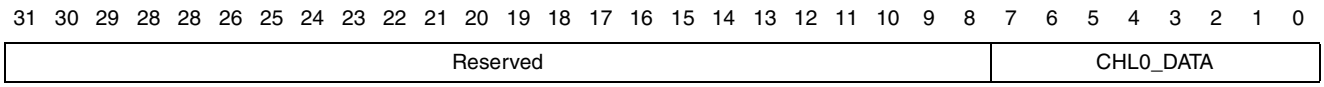
Address: *HDMIBaseAddress* + 0x10C
Type: R/W
Reset: 0
Description: Maximum number of lines in a frame.

[31:13] **Reserved**

[12:0] **YMAX**: ymax value in pix clocks.

Note: xmin, xmax, ymin and ymax must be programmed appropriately before the DEVICE_EN *bit is set. If the values of xmin, ymin, xmax and ymax are at zero, the video frame will not start.*

HDMI_DEFAULT_CHL0_DATA Default channel 0 data



Address: *HDMIBaseAddress* + 0x110
Type: R/W
Reset: 0
Description: This register stores a default value of the data that has to be transmitted on channel 0 when the encryption is enabled and the HDMI is not authenticated.

[31:8] **Reserved**

[7:0] **CHL0_DATA**: Contains the channel 0 default data

HDMI_DEFAULT_CHL1_DATA Default channel 1 data

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CHL1_DATA							

Address: *HDMIBaseAddress* + 0x114

Type: R/W

Reset: 0

Description: This register stores a default value of the data that has to be transmitted on channel 1 when the encryption is enabled and the HDMI is not authenticated.

[31:8] **Reserved**

[7:0] **CHL1_DATA**: Contains the channel 0 default data

HDMI_DEFAULT_CHL2_DATA Default channel 2 data

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CHL2_DATA							

Address: *HDMIBaseAddress* + 0x118

Type: R/W

Reset: 0

Description: This register stores a default value of the data that has to be transmitted on channel 2 when the encryption is enabled and the HDMI is not authenticated.

[31:8] **Reserved**

[7:0] **CHL2_DATA**: Contains the channel 0 default data

HDMI_CHL0_CAP_DATA Channel 0 capture data

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CHL0_DATA							

Address: *HDMIBaseAddress* + 0x11C

Type: RO

Reset: 0

Description: The captured data corresponding to channel 0 is stored here.

[31:8] **Reserved**

[7:0] **CHL2_DATA**: Contains the channel 0 captured data

HDMI_AUD_CFG Audio configuration

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								FIFO3_OVR_CLR	FIFO2_OVR_CLR	FIFO1_OVR_CLR	FIFO0_OVR_CLR	Reserved	SPDIF_DIV		LAYOUT

Address: *HDMIBaseAddress* + 0x200
Type: R/W
Reset: 0
Description:

[31:8] **Reserved**

[7] **FIFO3_OVR_CLR**
1: Clear FIFO3 overrun status

[6] **FIFO2_OVR_CLR**
1: Clear FIFO2 overrun status

[5] **FIFO1_OVR_CLR**
1: Clear FIFO1 overrun status

[4] **FIFO0_OVR_CLR**
1: Clear FIFO0 overrun status

[3] **Reserved**

[2:1] **SPDIF_DIV**
00: Divide input S/PDIF clock by 1 before using
01: Divide input S/PDIF clock by 2 before using
10: Divide input S/PDIF clock by 3 before using
11: Divide input S/PDIF clock by 4 before using

[0] **LAYOUT**
0: Two channel audio
1: Eight channel audio

HDMI_IFRAME_HEAD_WD Info frame header word

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								HB2								HB1								HB0							

Address: *HDMIBaseAddress* + 0x210
Type: R/W
Reset: 0
Description: The info frame header bytes are written into the register below.

[31:24] **Reserved**

[23:16] **HB2**: Header byte 2; bits 23:20 are always 0

[15:8] **HB1**: Header byte 1

[7:0] **HB0**: Header byte 0

HDMI_IFRAME_PKT_WDn Info frame packet word n

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PKT_WD_n*4+3								PKT_WD_n*4+2								PKT_WD_n*4+1								PKT_WD_n*4+0							

Address: *HDMIBaseAddress* + 0x214 + 4*n*

Type: R/W

Reset: 0

Description: There are seven registers (n = 0 to 6) corresponding to 28 bytes in the packet data.

[31:24] **PKT_WD_n*4+3**: Packet word 3
Contains the packet bytes.

[23:16] **PKT_WD_n*4+2**: Packet word 2
Contains the packet bytes.

[15:8] **PKT_WD_n*4+1**: Packet word 1
Contains the packet bytes.

[7:0] **PKT_WD_n*4+0**: Packet word 0
Contains the packet bytes.

HDMI_IFRAME_CFG Info frame configuration

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															EN

Address: *HDMIBaseAddress* + 0x230

Type: R/W

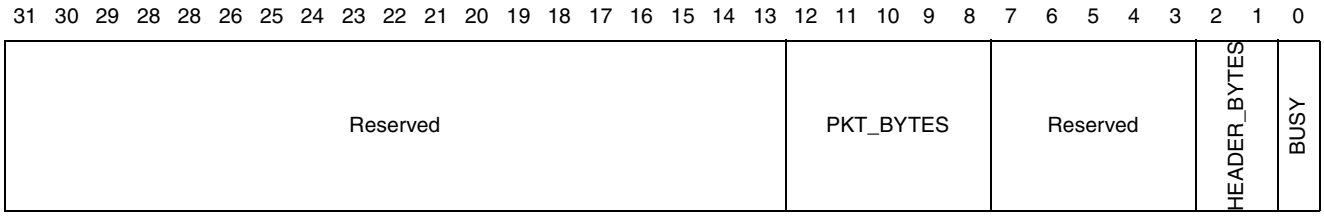
Reset: 0

Description:

[31:1] **Reserved**

[0] **EN**
1: Enable info frame with data in the FIFO

HDMI_IFRAME_FIFO_STA Info frame FIFO status



Address: *HDMIBaseAddress* + 0x240

Type: R/W

Reset: 0

Description: The status of the header bytes and the packet FIFO can be inferred by reading from this register.

[31:13] **Reserved**

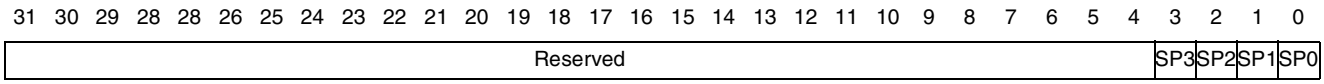
[12:8] **PKT_BYTES**: Packet bytes
0 0000: FiFO empty
0 0001: One byte in FIFO
0 0010: Two bytes in FIFO
.....
1 1100: 28 Bytes, FIFO full
1 1110: Undefined
1 1111: Undefined

[7:3] **Reserved**

[2:1] **HEADER_BYTES**
00: Empty
01: one valid byte
10: two valid bytes
11: Header byte word full

[o] **BUSY**
1: FIFO busy, Data being read from the FIFO

HDMI_SAMPLE_FLAT_MSK Sample flat mask



Address: *HDMIBaseAddress* + 0x244

Type: R/W

Reset: 0

Description: If a bit is set, then the sample flat bit for the corresponding audio subpacket in the audio sample packet is also set. Setting this bit overrides any internal generated conditions for the sample flat bit for the sub packet.

[31:4] **Reserved**

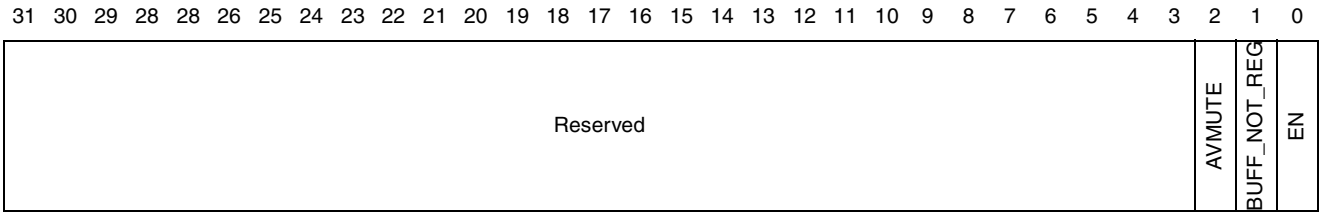
[3] **SP3**: Sample flat bit for sp3

[2] **SP2**: Sample flat bit for sp2

[1] **SP1**: Sample flat bit for sp1

[0] **SP0**: Sample flat bit for sp0

HDMI_GEN_CTRL_PKT_CFG General control packet configuration



Address: *HDMIBaseAddress* + 0x248

Type: R/W

Reset: 0

Description: Transmission of the general control packet can be enabled using this register.

The general control packet is transmitted only when bit 0 (enable) is set to 1.

If bit 1 (BUFF_NOT_REG) is reset, the general control packet is formatted by the hardware and the status in the bit2 (AVMUTE) is sent in the general control packet. If this bit is set, the contents of the AVI info buffer (both header and packet words) are sent. It is expected that all the 28 words are sent and the count value is overridden. While sending the general control packet from the buffer the rules of sending the packet like, it has to be sent only between active edge of VSYNC and 384 pixel clocks following has to be respected.

If bit 1 (AVMUTE) is at 0 then bit AVMUTE_CLR in the subpacket is set and AVMUTE_SET bit is reset. If this bit is set then AVMUTE_CLR bit is reset and AVMUTE_SET bit is set in the subpacket.

The general packet has to only transmitted between the active edge of VSYNC and 384 pixels following this edge. Once the general packet is transmitted the enable bit is reset to 0. An interrupt is generated if enabled after completion of general packet transmission.

[31:3] **Reserved**

[2] **AVMUTE**: AVMute status

[1] **BUFF_NOT_REG**
0: AVMUTE inferred from next bit
1: Data in the buffer transmitted

[0] **EN**: Enable general control packet transmission

Part 7

Audio

59 Audio subsystem

59.1 Overview

The STx7100 audio subsystem decodes and plays different standards of multichannel compressed audio streams.

The audio decoder is a 400 MHz ST231 CPU core, which basically reads encoded data from memory and writes decoded PCM data into memory. The audio ST231 is a slave to the ST40 host CPU, and can execute its code through either the EMI or LMI (system and video). The audio stream (encoded or decoded) can be received either from an external source via the digital PCM input interface or by an internal source such as the transport subsystem via the memory. Decoded audio data is output in both analog and digital formats.

For more information on the ST231, see Chapter 4: *CPUs on page 37*.

Audio outputs

The decoded audio data can be output via a stereo analog DAC on output pins AUDANAPLEFTOUT, AUDANAMLEFTOUT, AUDANAPRIGHTOUT and AUDANAMRIGHTOUT.

The STx7100 also provides a digital 10-channels PCM output on pins AUD0PCMOUT0, AUD0PCMOUT1, AUD0PCMOUT2, AUD0PCMOUT3, AUD0PCMOUT4, AUD0PCMCLK_OUT, AUD0LRCLKOUT and AUD0SCLKOUT.

An S/PDIF output on pin AUDSPDIFOUT is available to provide compressed or decoded audio PCM data.

Audio input

Stereo PCM data can be captured by the STx7100 via the input pins AUDATAIN, AUDSTRBIN and AUDLRCLKIN.

59.2 Features

The STx7100 audio decoder features are as follows:

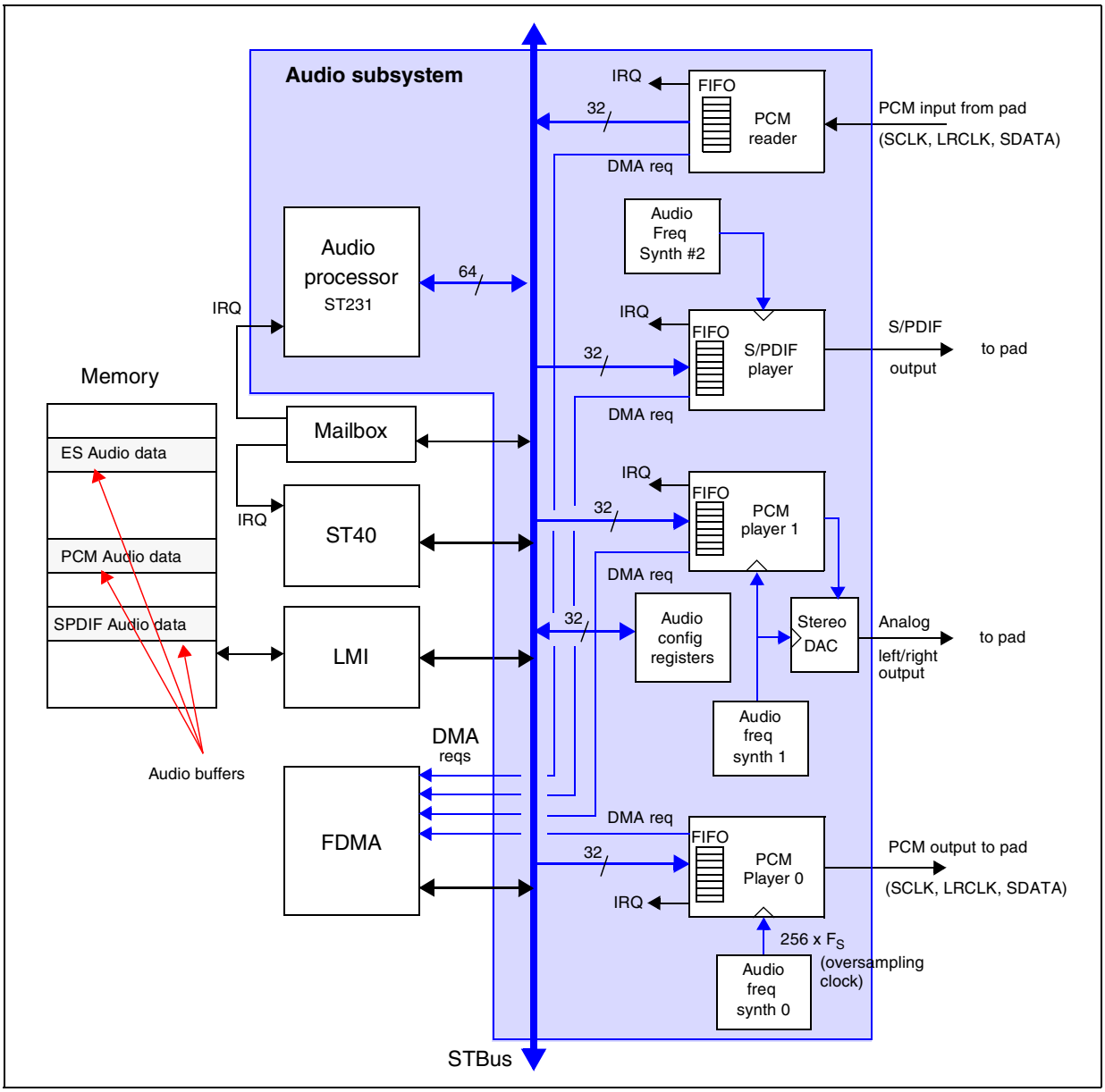
- compatible with all popular audio standards,
- PCM Mixing with internal or external source with sample rate conversion (32, 44.1, 48 KHz),
- encoded (IEC 61937) digital audio on S/PDIF output,
- 6-to-2 channel downmixing,
- PCM audio input (I^2S format),
- multichannel digital PCM output (10 channels),
- stereo analog output,
- 2-channel description channel decoding,
- post-processing (channel virtualization): Dolby Pro Logic downmix, volume control, bass redirection.

59.3 Block diagram

The audio subsystem block diagram is shown in Figure 240, and comprises the following functional units:

- An audio processor ST231 core running at 400 MHz executes the decoding algorithms, the sample-rate conversion, the post-processing and the volume control.
- A PCM reader captures data at the PCM input. The data is then stored in memory via an FDMA channel.
- Two PCM players get decoded PCM data from memory via FDMA channels. PCM player 0 delivers a decoded 10-channel audio stream to the output pins; PCM player 1 delivers a downmixed PCM data to the audio DAC.
- An S/PDIF player reads encoded data from memory via an FDMA channel.
- Three frequency synthesizers (comprising clockgen C) generate audio clocks used by the S/PDIF player, PCM players and audio DAC. Synthesizer FS0 clocks the PCM player associated to the PCM output; FS1 clocks the PCM player and the audio DAC; FS2 clocks the S/PDIF.
- A stereo 24-bit audio DAC with differential outputs provides the analog output stream.

Figure 240: Audio subsystem block diagram



59.4 Operation

Both host CPU (an ST40 core) and FDMA are used during the audio decoding process.

Since the audio decoder is a frame-decoder, the host CPU is required to control the audio processor frame-by-frame. A mailbox is used for the communication between the two processors,

The FDMA is used to build the ES buffer, feed the PCM and S/PDIF players and store in memory the data captured by the PCM reader.

59.5 Audio clocks generator (clock generator C)

The audio clocks generator generates the clocks for the two PCM players, the S/PDIF player and the audio DAC. These clocks are generated from the three channels (FS0, FS1 and FS2) of a single quad-frequency synthesizer.

The reference clock used by the synthesizer is either the clock connected to the SYSBCLKINALT pin or the 30 MHz generated from the SATA PHY. This frequency synthesizer is continuously adjusted by the clock recovery mechanism (refer to Chapter 15: *Clocks on page 120*).

The output clock is obtained by a digital algorithm controlled by the following parameters:

- MD[4:0]: coarse selector for the phase taps selection,
- PE[15:0]: fine selector for the phase taps selection,
- SD[2:0]: output divider.

There is a set of such parameters for each of the three channels.

The output frequency is given by the formula:

$$F_{out} = \frac{2^{15} \cdot F_{PLL}}{sdiv \times \left[\left(pe \cdot \left(1 + \frac{md}{32}\right)\right) - \left((pe - 2^{15}) \cdot \left(1 + \frac{md+1}{32}\right)\right)\right]}$$

with F_{PLL} equal to 27 or 30 MHz.

To avoid glitches at the frequency synthesizer output, only the MD, PE and EN_PRG parameters can be changed. The other parameters can be changed but glitches will occur.

The following registers control clock generator C:

- Register AUD_FSYN_CFG controls frequency synthesizer global parameters such as power-down, reference clock source or PLL filter selection.
- Configuration registers AUD_FSYN0_MD, AUD_FSYN0_PE, AUD_FSYN0_SDIV and AUD_FSYN0_PROG_EN setup the clock of the frequency synthesizer channel 0.
- Configuration registers AUD_FSYN1_MD, AUD_FSYN1_PE, AUD_FSYN1_SDIV and AUD_FSYN1_PROG_EN setup the clock of the frequency synthesizer channel 1.
- Configuration registers AUD_FSYN2_MD, AUD_FSYN2_PE, AUD_FSYN2_SDIV and AUD_FSYN2_PROG_EN setup the clock of the frequency synthesizer channel 2.

Changing the clock frequency

To change the clock frequency, the PROG_EN register must be set to 0 before changing the values of the registers MD, PE and SDIV. Setting the register PROG_EN to 1 enables the new setup.

59.6 Audio DAC

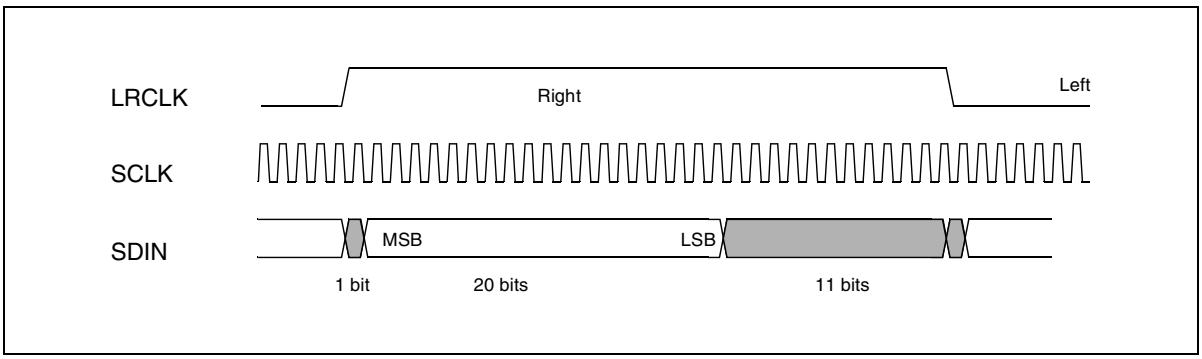
The audio DAC is a 24-bit digital-to-analog converter operating at 256 F_S (sampling frequency). The maximum sampling frequency is 96 KHz. The audio DAC clock is produced by the audio frequency synthesizer channel 1 and the input data is delivered by PCM player 1.

The DAC input is serial (I^2S format) and the DAC output is an analog differential current output.

The DAC features are as follows:

- 24 bits stereo DAC, I^2S format,
- differential output current,
- 110 dB dynamic range,
- -98 dB total harmonic distortion (THD),
- 256 F_S sampling frequency.

Figure 241: Audio DAC input format



The configuration register AUD_ADAC_CTRL controls the power-down, the reset and the mute of the audio DAC. See also Chapter 13: *Audio DAC* *on page 105*.

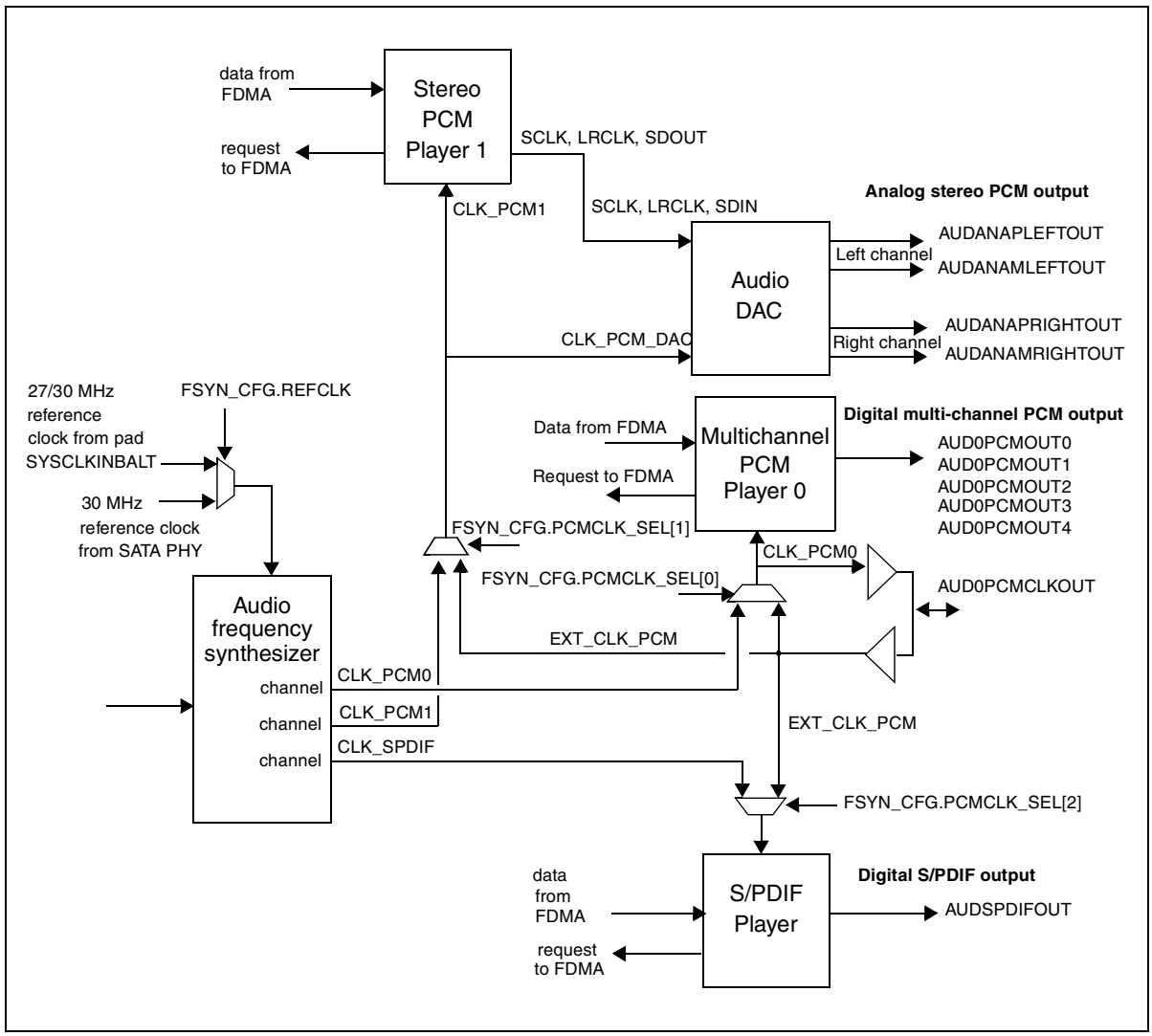
59.7 Audio output configuration

The register AUD_IO_CTRL contains bit to control the direction of the PCMOUT pins and the direction of the PCMCLKOUT pin. After reset, these pins are in high-impedance mode.

59.8 Audio DAC and frequency synthesizer connections

Figure 242 shows the connections between clock generator C, the audio peripherals and audio DAC.

Figure 242: Audio DAC & frequency synthesizer connections



59.9 PCM player 0 and 1

59.9.1 Overview

The PCM player receives audio data from the FDMA and provides outputs on serial output channels. It includes a 40 bytes internal FIFO for 10 channels. The STx7100 integrates 2 PCM players, PCM player 1 uses 2-channel only and PCM player 0 uses up to 10-channels.

Interface with the FDMA

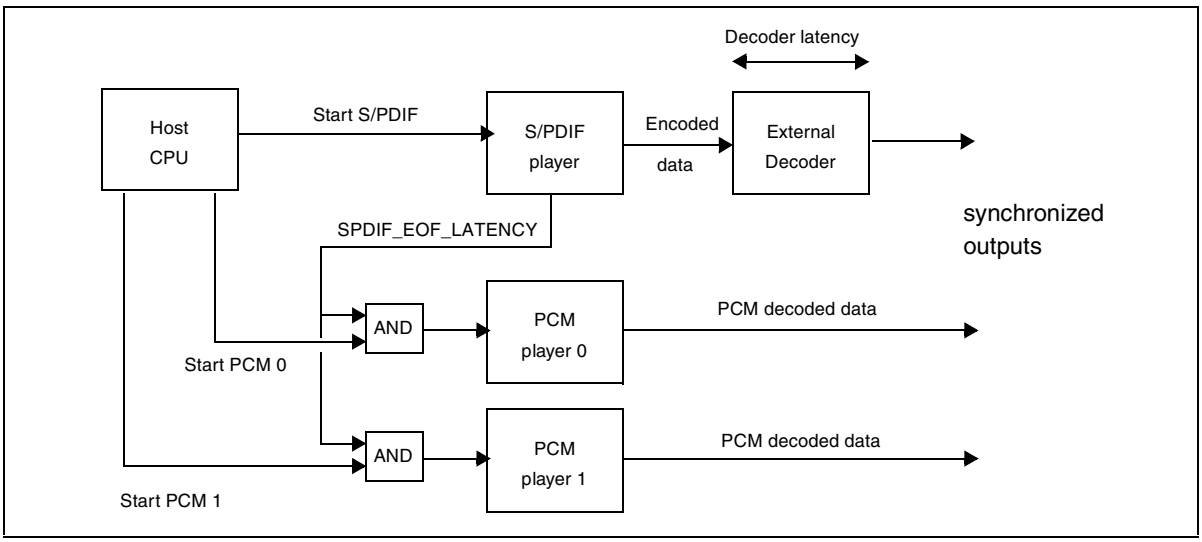
The PCM player communicates with the FDMA via a request signal connected to one of the paced-access FDMA channels (request index 26 for PCM player 0 and request index 27 for PCM player 1). When the internal FIFO is empty, a request is generated to the FDMA which will execute 10 4-bytes store operations to the PCM player 32-bits interface. The FDMA request will be de-asserted as soon as the first data is written in the FIFO.

S/PDIF latency

It may be necessary to synchronize the PCM players with the S/PDIF player to take into account the decoding time of the external decoder/amplifier connected to the S/PDIF. This synchronization is achieved by a pulse generated by the S/PDIF player and received by the PCM player that will start to produce data after receiving this pulse.

Both PCM player 0 and 1 can be synchronized with the S/PDIF player. This feature is defined in AUD_PCM_CTRL register.

Figure 243: PCM players and S/PDIF player synchronization



Interrupt

The PCM players can generate an interrupt in the following conditions: FIFO underflow or when a predefined number of bytes has been read from memory (defined in AUD_PCM_CTRL register).

The interrupts are handled via the register AUD_PCM_ITS (interrupt status), AUD_PCM_ITS_CLEAR, AUD_PCM_IT_ENABLE, AUD_PCM_IT_ENABLE_SET, AUD_PCM_IT_ENABLE_CLEAR.

Clocking

The two PCM players are clocked by two different frequency synthesizer channels (0 and 1). The synthesizer generates the audio oversampling clock from which the PCM serial clock and left-right clock are derived. The ratio between the frequency synthesizer clock frequency and the serial clock is defined in the register AUD_PCM_CTRL.

Control

The register AUD_PCM_CTRL controls the modes of operation (PCM or compressed data), the memory storage format and the clocking parameters.

Format

A number of format are supported: 16 or 32 bits per subframe, 24, 20, 18 or 16 bits data. The data alignment and the clock edge polarity can also be defined by the register AUD_PCM_FORMAT.

Status

The status of the PCM players is reported in the register AUD_PCM_FORMAT.

59.10 S/PDIF player

59.10.1 Overview

The S/PDIF player supports the IEC-61937 (compressed audio data) standard.

The S/PDIF player receives formatted audio data from the FDMA and outputs these audio data on the S/PDIF output pin. It includes an internal 8-bytes FIFO.

Interface with the FDMA

The S/PDIF player communicates with the FDMA via a request signal connected to one of the paced-access FDMA channels (request index 28). When the internal FIFO is empty, a request is generated to the FDMA which will execute 2 4-bytes store operations to the S/PDIF player 32-bits interface. The FDMA request will be de-asserted as soon as the first data is written into the FIFO.

Interrupt

The S/PDIF player can generate an interrupt in the following conditions: FIFO underflow, end of data burst, end of block, EOF latency, EOF Pd databurst or when a predefined number of bytes has been read from memory (defined in AUD_SPDIF_CTRL register).

The interrupts are handled via the register AUD_SPDIF_ITS (interrupt status), AUD_SPDIF_ITS_CLEAR, AUD_SPDIF_IT_ENABLE, AUD_SPDIF_IT_ENABLE_SET, AUD_SPDIF_IT_ENABLE_CLEAR.

Clocking

The S/PDIF player is clocked by the frequency synthesizer channel 2. The ratio between the frequency synthesizer clock frequency and the S/PDIF clock is defined in the register AUD_SPDIF_CTRL.

Control

The register AUD_SPDIF_CTRL controls the mode of operation and the clocking parameters.

Burst parameters

The burst parameters pa_pb, pc_pd, channel status bit, user and validity bits, burst length are defined in the registers AUD_SPDIF_PA_PB, AUD_SPDIF_PC_PD, AUD_SPDIF_CL1, AUD_SPDIF_CR1, AUD_SPDIF_CL2_CR2_UV, AUD_SPDIF_PAUSE_LAT, AUD_SPDIF_FRMLGTH_BURST.

Status

The register AUD_SPDIF_STA reports data related to the S/PDIF player state such as data underflow, end of burst, end of block, and end of subframe.

59.11 PCM reader

59.11.1 Functional description

The PCM reader captures an external serial I^2S input stream. After de-serialization, the data is stored into memory. The data can be either PCM or compressed data. The PCM reader includes an 8-bytes FIFO.

Interface with the FDMA

The PCM reader communicates with the FDMA via a request signal connected to one of the paced-access FDMA channels (request index 29). When the internal FIFO is empty, a request is generated to the FDMA, which executes two 4-byte load operations to the PCM reader's 32-bit interface. The FDMA request will be de-asserted as soon as the first data is read from the FIFO.

Interrupt

The PCM reader can generate an interrupt when the FIFO is in overflow.

The interrupts are handled via the register AUD_PCMIN_ITS (interrupt status), AUD_PCMIN_ITS_CLEAR, AUD_PCMIN_IT_ENABLE, AUD_PCMIN_IT_ENABLE_SET, AUD_PCMIN_IT_ENABLE_CLEAR.

Clocking

The PCM reader is clocked by the external serial clock on pin AUDSTRBIN.

Control

The register AUD_PCMIN_CTRL controls the mode of operation and the memory storage formats.

Format

The register AUD_PCMIN_FORMAT defines the number of bits per subframe, the data size and the polarity of the serial and left-right clocks.

Status

The register AUD_PCMIN_STA reports data related to the PCM reader state such as: FIFO overflow, the number of audio frames received between two vertical syncs and the state (running or stopped) of the PCM reader.

60 Audio registers

Register addresses are provided as one of
AudioConfigBaseAddress + offset;
AudioPCM0BaseAddress + offset;
AudioPCM1BaseAddress + offset;
AudioSPDIFBaseAddress + offset;
AudioPCMReaderBaseAddress + offset.

The *AudioConfigBaseAddress* is:
0x1921 0000.

The *AudioPCM0BaseAddress* is:
0x1810 1000.

The *AudioPCM1BaseAddress* is:
0x1810 1800.

The *AudioSPDIFBaseAddress* is:
0x1810 3000.

The Audio*PCMReaderBaseAddress* is:
0x1810 2000.

60.1 Summary

60.1.1 Audio configuration

Table 198: Audio configuration - register summary

Register name	Function	Offset	Type	Reset value
AUD_FSYN_CFG	Frequency synthesizer global parameters	0x000	R/W	0x0000 0000
AUD_FSYN0_MD	Frequency synthesizer channel 0 coarse selection	0x010	R/W	0x0000 0000
AUD_FSYN0_PE	Frequency synthesizer channel 0 fine selection	0x014	R/W	0x0000 0000
AUD_FSYN0_SDIV	Frequency synthesizer channel 0 output divider	0x018	R/W	0x0000 0000
AUD_FSYN0_PROG_EN	Frequency synthesizer channel 0 program enable	0x01C	R/W	0x0000 0000
AUD_FSYN1_MD	Frequency synthesizer channel 1 coarse selection	0x020	R/W	0x0000 0000
AUD_FSYN1_PE	Frequency synthesizer channel 1 fine selection	0x024	R/W	0x0000 0000
AUD_FSYN1_SDIV	Frequency synthesizer channel 1 output divider	0x028	R/W	0x0000 0000
AUD_FSYN1_PROG_EN	Frequency synthesizer channel 1 program enable	0x02C	R/W	0x0000 0000
AUD_FSYN2_MD	Frequency synthesizer channel 2 coarse selection	0x030	R/W	0x0000 0000
AUD_FSYN2_PE	Frequency synthesizer channel 2 fine selection	0x034	R/W	0x0000 0000
AUD_FSYN2_SDIV	Frequency synthesizer channel 2 output divider	0x038	R/W	0x0000 0000
AUD_FSYN2_PROG_EN	Frequency synthesizer channel 2 program enable	0x03C	R/W	0x0000 0000
AUD_ADAC_CTRL	Audio DAC control	0x100	R/W	0x0000 003F
AUD_IO_CTRL	Audio outputs control	0x200	R/W	0x0000 0000

60.1.2 PCM player 0

Table 199: PCM player 0 - register summary

Register name	Function	Offset	Type	Reset value
AUD_PCMOUT0_RST	PCM player 0 soft reset	0x000	R/W	0x0000 0000
AUD_PCMOUT0_DATA	PCM player 0 data FIFO	0x004	R/W	0x0000 0000
AUD_PCMOUT0_ITS	PCM player 0 interrupt status	0x008	RO	0x0000 0000
AUD_PCMOUT0_ITS_CLR	PCM player 0 interrupt status clear	0x00C	R/W	0x0000 0000
AUD_PCMOUT0_IT_EN	PCM player 0 interrupt enable	0x010	RO	0x0000 0000
AUD_PCMOUT0_IT_EN_SET	PCM player 0 interrupt enable set	0x014	R/W	0x0000 0000
AUD_PCMOUT0_IT_EN_CLR	PCM player 0 interrupt enable clear	0x018	R/W	0x0000 0000
AUD_PCMOUT0_CTRL	PCM player 0 control	0x01C	R/W	0x0000 0000
AUD_PCMOUT0_STA	PCM player 0 status	0x020	R/W	0x0000 0000
AUD_PCMOUT0_FMT	PCM player 0 format	0x024	R/W	0x0000 0000

60.1.3 PCM player 1

Table 200: PCM player 1 - register summary

Register name	Function	Offset	Type	Reset value
AUD_PCMOUT1_RST	PCM player 1 soft reset	0x000	R/W	0x0000 0000
AUD_PCMOUT1_DATA	PCM player 1 data FIFO	0x004	R/W	0x0000 0000
AUD_PCMOUT1_ITS	PCM player 1 interrupt status	0x008	RO	0x0000 0000
AUD_PCMOUT1_ITS_CLR	PCM player 1 interrupt status clear	0x00C	R/W	0x0000 0000
AUD_PCMOUT1_IT_EN	PCM player 1 interrupt enable	0x010	RO	0x0000 0000
AUD_PCMOUT1_IT_EN_SET	PCM player 1 interrupt enable set	0x014	R/W	0x0000 0000
AUD_PCMOUT1_IT_EN_CLR	PCM player 1 interrupt enable clear	0x018	R/W	0x0000 0000
AUD_PCMOUT1_CTRL	PCM player 1 control	0x01C	R/W	0x0000 0000
AUD_PCMOUT1_STA	PCM player 1 status	0x020	R/W	0x0000 0000
AUD_PCMOUT1_FMT	PCM player 1 format	0x024	R/W	0x0000 0000

60.1.4 S/PDIF player

Table 201: S/PDIF player - register summary

Register name	Function	Offset	Type	Reset value
AUD_SPDIF_RST	S/PDIF player soft reset	0x000	R/W	0x0000 0000
AUD_SPDIF_DATA	S/PDIF player data FIFO	0x004	R/W	0x0000 0000
AUD_SPDIF_ITS	S/PDIF player interrupt status	0x008	RO	0x0000 0000
AUD_SPDIF_ITS_CLR	S/PDIF player interrupt status clear	0x00C	R/W	0x0000 0000
AUD_SPDIF_IT_EN	S/PDIF player interrupt enable	0x010	RO	0x0000 0000
AUD_SPDIF_IT_EN_SET	S/PDIF player interrupt enable set	0x014	R/W	0x0000 0000
AUD_SPDIF_IT_EN_CLR	S/PDIF player interrupt enable clear	0x018	R/W	0x0000 0000
AUD_SPDIF_CTRL	S/PDIF player control	0x01C	R/W	0x0000 0000
AUD_SPDIF_STA	S/PDIF player status	0x020	R/W	0x0000 0000
AUD_SPDIF_PA_PB	S/PDIF player Pa, Pb burst	0x024	R/W	0x0000 0000
AUD_SPDIF_PC_PD	S/PDIF player Pc, Pd burst	0x028	R/W	0x0000 0000
AUD_SPDIF_CL1	S/PDIF player channel status for left subframes	0x02C	R/W	0x0000 0000
AUD_SPDIF_CR1	S/PDIF player channel status for right subframes	0x030	R/W	0x0000 0000
AUD_SPDIF_CL2_CR2_UV	S/PDIF player channel status for right and left subframes, user and validity bits	0x034	R/W	0x0000 0000
AUD_SPDIF_PAU_LAT	S/PDIF player pause gap length and latency	0x038	R/W	0x0000 0000
AUD_SPDIF_FRA_LEN_BST	S/PDIF player burst length	0x03C	R/W	0x0000 0000

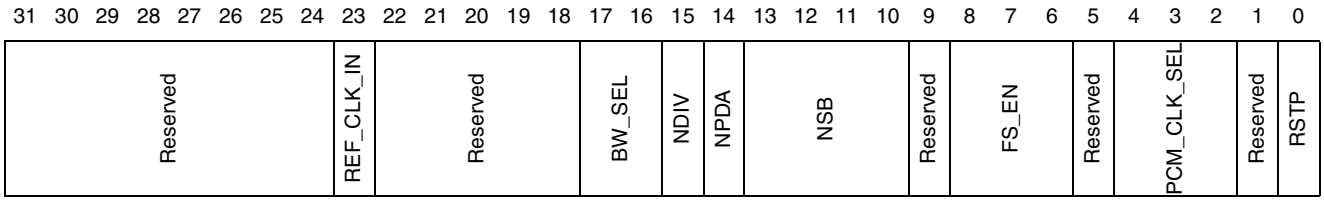
60.1.5 PCM reader

Table 202: PCM reader - register summary

Register name	Function	Offset	Type	Reset value
AUD_PCMIN_RST	PCM reader soft reset	0x000	R/W	0x0000 0000
AUD_PCMIN_DATA	PCM reader data FIFO	0x004	R/W	0x0000 0000
AUD_PCMIN_ITS	PCM reader interrupt status	0x008	RO	0x0000 0000
AUD_PCMIN_ITS_CLR	PCM reader interrupt status clear	0x00C	R/W	0x0000 0000
AUD_PCMIN_IT_EN	PCM reader interrupt enable	0x010	RO	0x0000 0000
AUD_PCMIN_IT_EN_SET	PCM reader interrupt enable set	0x014	R/W	0x0000 0000
AUD_PCMIN_IT_EN_CLR	PCM reader interrupt enable clear	0x018	R/W	0x0000 0000
AUD_PCMIN_CTRL	PCM reader control	0x01C	R/W	0x0000 0000
AUD_PCMIN_STA	PCM reader status	0x020	R/W	0x0000 0000
AUD_PCMIN_FMT	PCM reader format	0x024	R/W	0x0000 0000

60.2 Audio configuration

AUD_FSYN_CFG Frequency synthesizer global parameters



Address: *AudioBaseAddress* + 0x000

Type: R/W

Reset: 0xXX00 0000

Description: Defines the global parameters of the frequency synthesizer.

[31:24] **Reserved**

[23] **REF_CLK_IN**: Frequency synthesizer reference clock source
0: SATA PHY 30 MHz clock 1: SYSBCLKINALT clock

[22:18] **Reserved**

[17:16] **BW_SEL**: Frequency synthesizer reference clock filter
00: Very good reference 10: Bad reference
01: Good reference 11: Very bad reference

[15] **NDIV**: Frequency synthesizer reference clock frequency
0: 27/30 MHz 1: 54/60 MHz

[14] **NPDA**: Frequency synthesizer analog power down mode - active low

[13:10] **NSB**: Frequency synthesizer digital part standby mode - active low
[13] Reserved
[12] SPDIF_FSYN2
[11] PCM_FSYN1
[10] PCM_FSYN0

[9] **Reserved**

[8:6] **FS_EN**: Enable frequency synthesizers - active high
[8] SPDIF_FSYN2
[7] PCM_FSYN1
[6] PCM_FSYN0

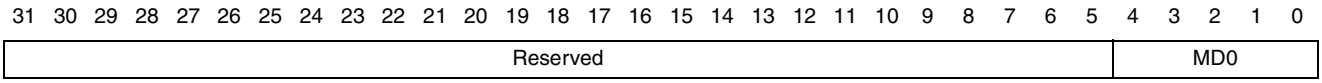
[5] **Reserved**

[4:2] **PCM_CLK_SEL**: Select the frequency synthesizer clock or the external PCM clock for each channel
[4] SPDIF_FSYN2
[3] PCM_FSYN1
[2] PCM_FSYN0
0: External clock 1: Frequency synthesizer clock

[1] **Reserved**

[0] **RSTP**: Frequency synthesizer reset - active high
0: The frequency synthesizer is running 1: Reset the frequency synthesizer

AUD_FSYN0_MD Frequency synthesizer channel 0 coarse selection



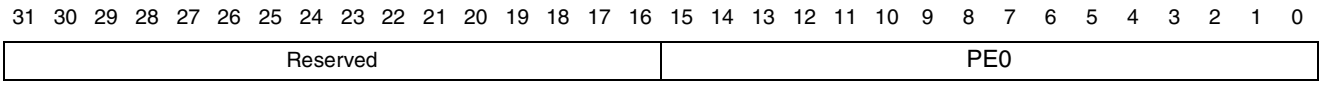
Address: *AudioBaseAddress* + 0x010

Type: R/W

Reset: 0xXXXX XX00

Description: Defines the MD parameter (coarse selection) for the frequency synthesizer channel 0. The MD parameter is defined in the range [-1 to -16] ([0x1F to 0x0F]).

AUD_FSYN0_PE Frequency synthesizer channel 0 fine selection



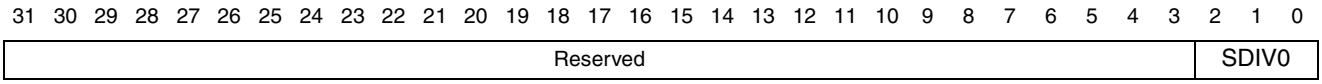
Address: *AudioBaseAddress* + 0x014

Type: R/W

Reset: 0xXXXX 0000

Description: Defines the PE parameter (fine selection) for the frequency synthesizer channel 0. The PE parameter is defined in the range [0, 2^{15} - 1].

AUD_FSYN0_SDIV Frequency synthesizer channel 0 output divider



Address: *AudioBaseAddress* + 0x018

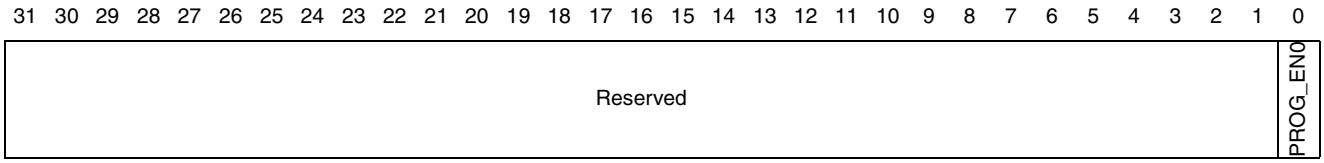
Type: R/W

Reset: 0x0

Description: Defines the SDIV parameter (output divider control) for the frequency synthesizer channel 0. The SDIV0 parameter can take the values 2, 4, 8 or 256.

000: 2
001: 4
010: 8
011 - 110: Reserved
111: 256

AUD_FSYN0_PROG_EN Frequency synthesizer channel 0 program enable



Address: *AudioBaseAddress* + 0x01C

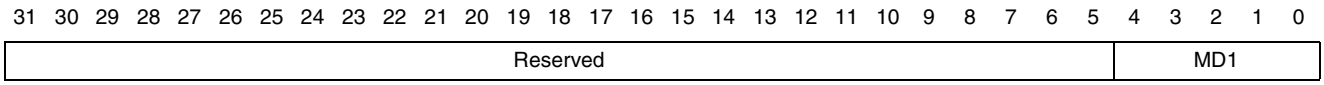
Type: R/W

Reset: 0xXXXX XXX1

Description: Defines the PROG_EN parameter (programming enable) for the frequency synthesizer channel 0. This parameter must be set to 1 to take into account a new configuration.

0: PE0 and MD0 parameters are ignored
1: PE0 and MD0 parameters are taken into account.

AUD_FSYN1_MD Frequency synthesizer channel 1 coarse selection



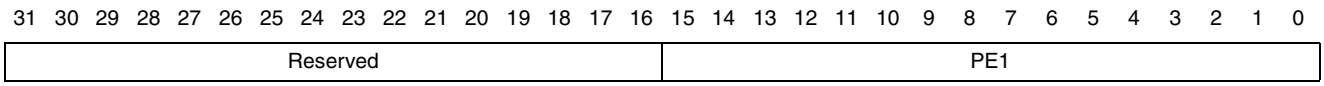
Address: *AudioBaseAddress* + 0x020

Type: R/W

Reset: 0xXXXX XX00

Description: Defines the MD parameter (coarse selection) for the frequency synthesizer channel 1. The MD parameter is defined in the range [-1, -16] ([0x1F, 0x0F]).

AUD_FSYN1_PE Frequency synthesizer channel 1 fine selection



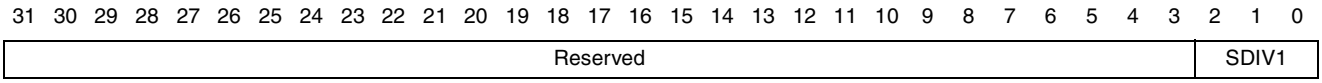
Address: *AudioBaseAddress* + 0x024

Type: R/W

Reset: 0xXXXX 0000

Description: Defines the PE parameter (fine selection) for the frequency synthesizer channel 1. The PE parameter is defined in the range [0, 2^{15} - 1].

AUD_FSYN1_SDIV Frequency synthesizer channel 1 output divider



Address: *AudioBaseAddress* + 0x028

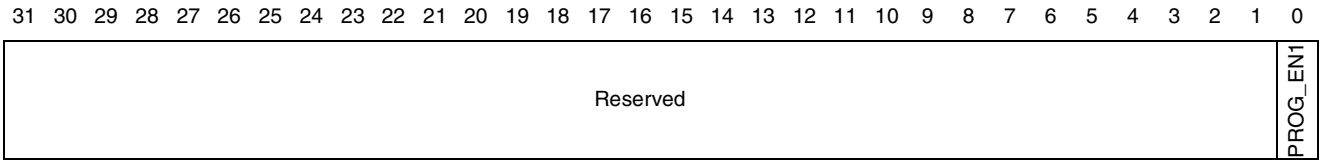
Type: R/W

Reset: 0x0xXXXX XXX0

Description: Defines the SDIV parameter (output divider control) for the frequency synthesizer channel 1. The SDIV0 parameter can take the values 2, 4, 8 or 256.

000: 2
001: 4
010: 8
011 - 110: Reserved
111: 256

AUD_FSYN1_PROG_EN Frequency synthesizer channel 1 program enable



Address: *AudioBaseAddress* + 0x02C

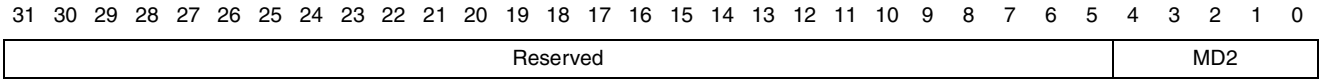
Type: R/W

Reset: 0xXXXX XXX1

Description: Defines the PROG_EN parameter (programming enable) for the frequency synthesizer channel 1. This parameter must be set to 1 to take into account a new configuration.

0: PE1 and MD1 parameters are ignored.
1: PE1 and MD1 parameters are taken into account.

AUD_FSYN2_MD Frequency synthesizer channel 2 coarse selection



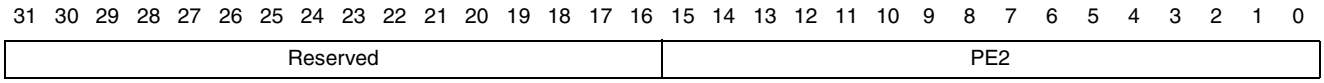
Address: *AudioBaseAddress* + 0x030

Type: R/W

Reset:

Description: Defines the MD parameter (coarse selection) for the frequency synthesizer channel 2. The MD parameter is defined in the range [-1 to -16] ([0x1F to 0x0F]).

AUD_FSYN2_PE Frequency synthesizer channel 2 fine selection



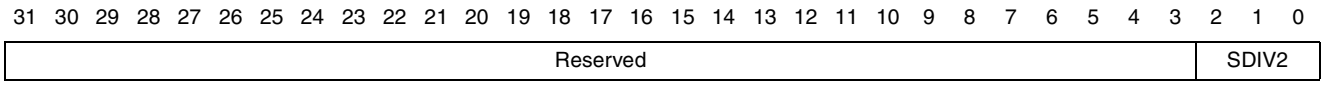
Address: *AudioBaseAddress* + 0x034

Type: R/W

Reset: 0xXXXX 0000

Description: Defines the PE parameter (fine selection) for the frequency synthesizer channel 2. The PE parameter is defined in the range [0, 2^{15} - 1].

AUD_FSYN2_SDIV Frequency synthesizer channel 2 output divider



Address: *AudioBaseAddress* + 0x038

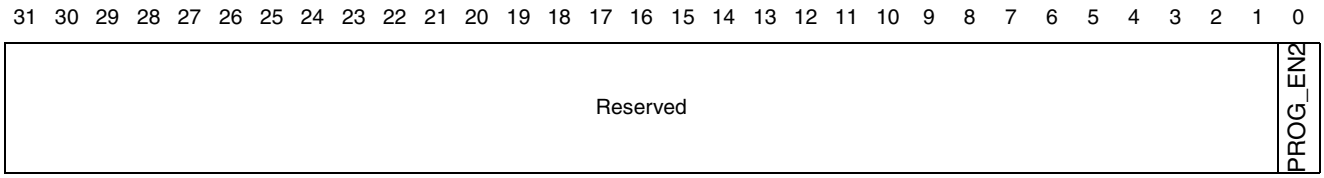
Type: R/W

Reset: 0xXXXX XXX0

Defines the SDIV parameter (output divider control) for the frequency synthesizer channel 2. The SDIV0 parameter can take the values 2, 4, 8 or 256.

000: 2
001: 4
010: 8
011 - 110: Reserved
111: 256

AUD_FSYN2_PROG_EN Frequency synthesizer channel 2 program enable



Address: *AudioBaseAddress* + 0x03C

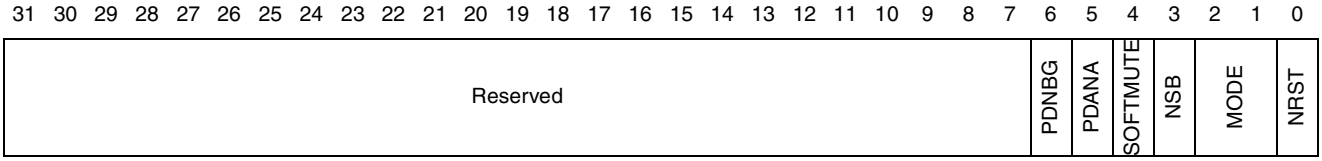
Type: R/W

Reset: 0xXXXX XXX1

Description: Defines the PROG_EN parameter (programming enable) for the frequency synthesizer channel 2. This parameter must be set to 1 to take into account a new configuration.

0: PE2 and MD2 parameters are ignored.
1: PE2 and MD2 parameters are taken into account.

AUD_ADAC_CTRL Audio DAC control



Address: *AudioBaseAddress* + 0x100
Type: R/W
Reset: 0xXXXX XX00
Description: Controls the audio DAC.

[31:7] **Reserved**

[6] **PDNBG**: DAC bandgap power down - active low

[5] **PDANA**: DAC analog part power down mode - active low

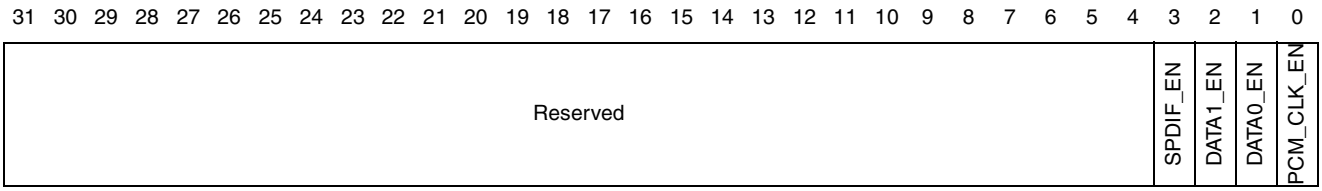
[4] **SOFTMUTE**: DAC soft mute

[3] **NSB**: DAC digital part standby mode - active low

[2:1] **MODE**: Must be set to 00

[0] **NRST**: DAC reset - active low

AUD_IO_CTRL Audio outputs control



Address: *AudioBaseAddress* + 0x200
Type: R/W
Reset: 0xXXXX XXX0
Description: Controls the direction of the audio output pins.

[31:4] **Reserved**

[3] **SPDIF_EN**: Enable AUDSPDIFOUT
0: Disable 1: Enable

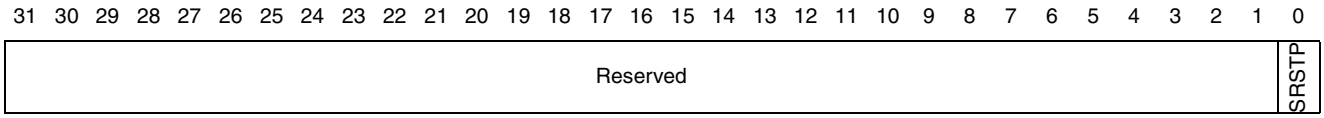
[2] **DATA1_EN**: AUDPCMOUT1, 2, 3, 4 pin direction
0: Input 1: Output

[1] **DATA0_EN**: AUDPCMOUT0 pin direction
0: Input 1: Output

[0] **PCM_CLK_EN**: AUDPCMCLK_OUT clock pin direction
0: Input 1: Output

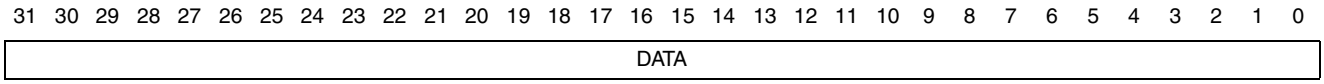
60.3 PCM player 0

AUD_PCMOUT0_RST PCM Player 0 soft reset



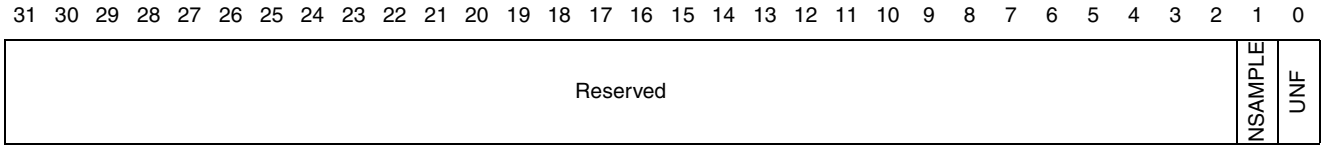
Address: *AudioPCM0BaseAddress* + 0x000
Type: R/W
Reset: 0xXXXX XXX0
Description: Soft reset the PCM player - active high.

AUD_PCMOUT0_DATA PCM player 0 data FIFO



Address: *AudioPCM0BaseAddress* + 0x004
Type: R/W
Reset: 0x0000 0000
Description: PCM player data to be stored in the FIFO.

AUD_PCMOUT0_ITS PCM player 0 interrupt status



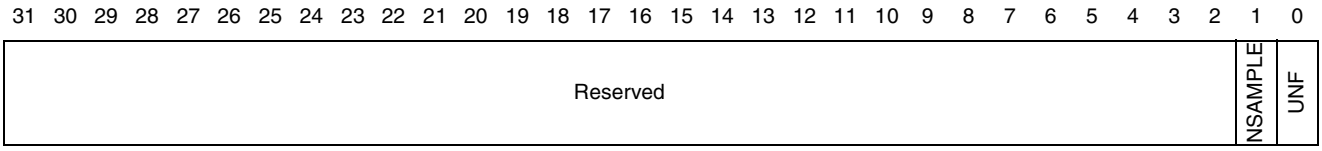
Address: *AudioPCM0BaseAddress* + 0x008
Type: RO
Reset: 0x0000 0000
Description: 1 indicates an interrupt.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: The defined number of samples has been read from memory.

[0] **UNF**
1: FIFO underflow

AUD_PCMOUT0_ITS_CLR PCM player 0 interrupt status clear



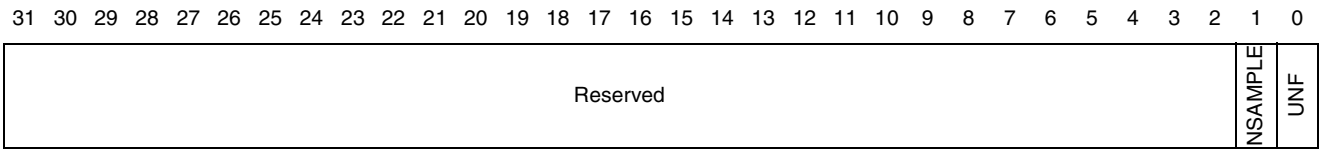
Address: *AudioPCM0BaseAddress* + 0x00C
Type: R/W
Reset: 0x0000 000C
Description: 1 clears the corresponding interrupt.

[31:29] **Reserved**: 0000

[1] **NSAMPLE**
1: Clear the interrupt generated by the defined number of samples being read from memory

[0] **UNF**
1: Clear the FIFO underflow interrupt

AUD_PCMOUT0_IT_EN PCM player 0 interrupt enable



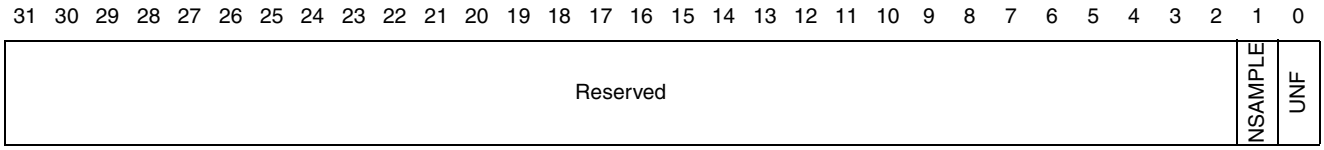
Address: *AudioPCM0BaseAddress* + 0x010
Type: RO
Reset: 0x0000 0000
Description: 1 indicates that the corresponding interrupt is enabled.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Enable interrupt when a defined number of samples is read from memory

[0] **UNF**
1: Enable FIFO underflows interrupt

AUD_PCMOUT0_IT_EN_SET PCM player 0 interrupt enable set



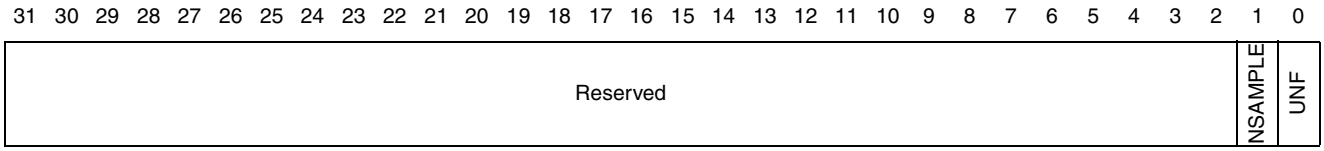
Address: *AudioPCM0BaseAddress* + 0x014
Type: R/W
Reset: 0x0000 0000
Description: 1 enables the corresponding interrupt.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Enable interrupt when a defined number of samples is read from memory.

[0] **UNF**
1: Enable FIFO underflow interrupt

AUD_PCMOUT0_IT_EN_CLR PCM player 0 interrupt enable clear



Address: *AudioPCM0BaseAddress* + 0x018
Type: R/W
Reset: 0x0000 0000
Description: 1 clears the corresponding interrupt.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Enable interrupt when a defined number of samples is read from memory.

[0] **UNF**
1: Enable FIFO underflow interrupt

AUD_PCMOUT0_CTRL PCM player 0 control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSAMPLE																			SPDIF_LAT	CLK_DIV								RND	MEM_FMT	MODE	

Address: *AudioPCM0BaseAddress* + 0x01C

Type: R/W

Reset: 0x0000 0000

Description: This register controls the PCM player.

[31:13] **NSAMPLE**: Defines the number of samples read from memory before an interrupt is generated
Each time the number of samples read by the PCM player is equal to this number, the associated bit in the status register is set to 1.

[12] **SPDIF_LAT**: S/PDIF latency
0: Ignore S/PDIF latency signal to start the PCM player
1: Wait for S/PDIF latency signal to start the PCM player

[11:4] **CLK_DIV**: Frequency synthesizer clock divide
32 slots / subframe:
0: 1 x f_{FS}, 1: 128 x f_{FS}, 6: 192 x f_{FS}, 2: 256 x f_{FS}, 3: 384 x f_{FS}, 4: 512 x f_{FS}, 6: 768 x f_{FS}
16 slots / subframe:
0: 1 x f_{FS}, 2: 128 x f_{FS}, 3: 192 x f_{FS}, 4: 256 x f_{FS}, 6: 384 x f_{FS}, 8: 512 x f_{FS}, 12: 768 x f_{FS}

[3] **RND**: Rounding
0: No rounding 1: 16 bits rounding on PCM data

[2] **MEM_FMT**: Memory storage format
0: 16 bits / 0 1: 16 bits / 16 bits

[1:0] **Mode**
00: PCM Off 10: PCM
01: PCM mute 11: CD

AUD_PCMOUT0_STA PCM player 0 status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													NSAMPLE	UNF	RUN_STOP

Address: *AudioPCM0BaseAddress* + 0x020

Type: R/W

Reset: 0x0000 0000

Description: Status of the PCM player.

[31:3] **Reserved**

[2] **NSAMPLE**
1: The predefined number of samples have been read from memory

[1] **UNF**
1: FIFO underflow

[0] **RUN_STOP**
0: PCM player stopped 1: PCM player running

AUD_PCMOUT0_FMT PCM player 0 format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DMA_REQ_TRIG_LMT				Reserved	NUM_CH			ORDER	ALIGN	PADDING	SCLK_EDGE	LR_POL	DATA_SIZE		NBIT

Address: *AudioPCM0BaseAddress* + 0x024

Type: R/W

Reset: 0x0000 0000

Description: Defines the data format.

[31:16] **Reserved**

[15:12] **DMA_REQ_TRIG_LMT**: DMA request trigger limit
A DMA request is generated as soon as the number of cells available in the FIFO is equal to this value.
Valid values: 2,4,6,8,10.
Reset value: 0000

[11] **Reserved**

[10:8] **NUM_CH**: Number of channels
Specifies the number of channels required at the output. Null data appears at the output for remaining unused channels.
001: One channel only (stereo)
010: Two channels only
011: Three channels only
100: Four channels only
101: All five channels
Reset value: 000

[7] **ORDER**: Bit ordering
0: Data is output LSBit first
1: Data is output MSBit first

[6] **ALIGN**
0: Data is left-aligned wrt to LR clock
1: Data is right-aligned wrt to LR clock

[5] **PADDING**:
0: Data is delayed by 1 clock cycle
1: Data is not delayed

[4] **SCLK_EDGE**: Active edge
0: Data is output on the rising edge of SCLK
1: Data is output on the falling edge of SCLK

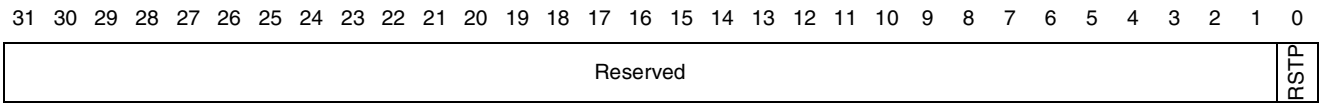
[3] **LR_POL**: Left-right clock polarity
0: Left word when LR clock is low
1: Left word when LR clock is high

[2:1] **DATA_SIZE**: Data size
00: 24 bits
01: 20 bits
10: 18 bits
11: 16 bits

[0] **NBIT**: Number of bits per subframe
0: 32 bits per subframe
1: 16 bits per subframe

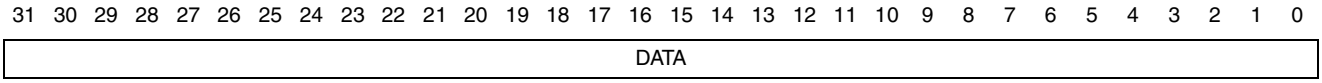
60.4 PCM player 1

AUD_PCMOUT1_RST PCM player 1 soft reset



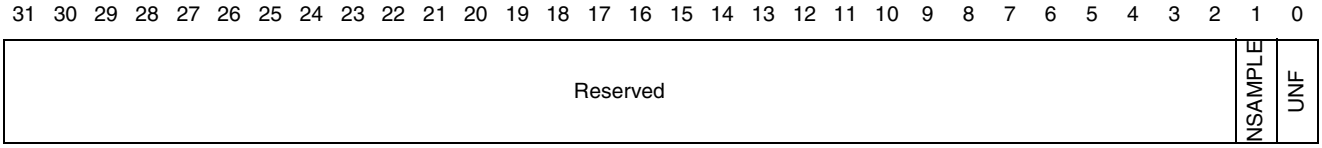
Address: *AudioPCM1BaseAddress* + 0x000
Type: R/W
Reset: 0x0000 0000
Description: Soft reset the PCM player - active high.

AUD_PCMOUT1_DATA PCM player 1 data FIFO



Address: *AudioPCM1BaseAddress* + 0x004
Type: R/W
Reset: 0x0000 0000
Description: PCM player data to be stored in the FIFO.

AUD_PCMOUT1_ITS PCM player 1 interrupt status



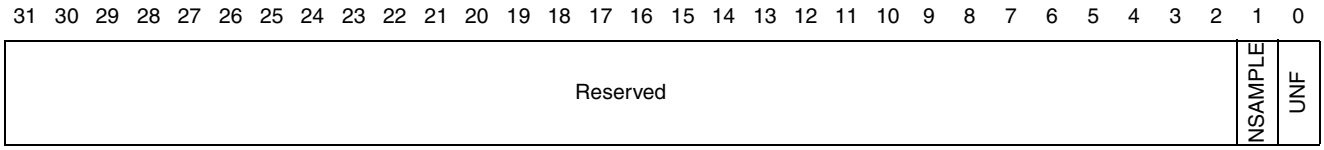
Address: *AudioPCM1BaseAddress* + 0x008
Type: RO
Reset: 0x0000 0000
Description: 1 indicates an interrupt has occurred.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: The defined number of samples has been read from memory

[0] **UNF**
1: FIFO underflow

AUD_PCMOUT1_ITS_CLR PCM player 1 interrupt status clear



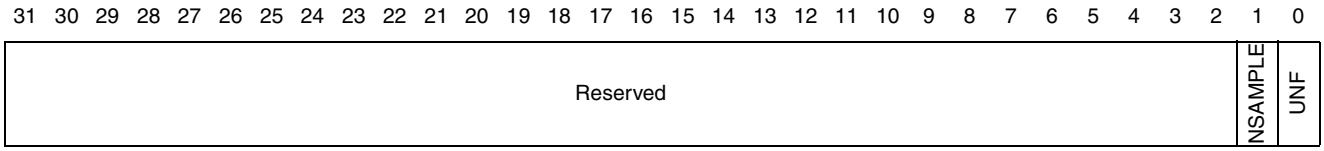
Address: *AudioPCM1BaseAddress* + 0x00C
Type: R/W
Reset: 0x0000 000C
Description: 1 clears the corresponding interrupt.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Clear the interrupt generated when a defined number of samples has been read from memory

[0] **UNF**
1: Clear the FIFO underflow interrupt

AUD_PCMOUT1_IT_EN PCM player 1 interrupt enable



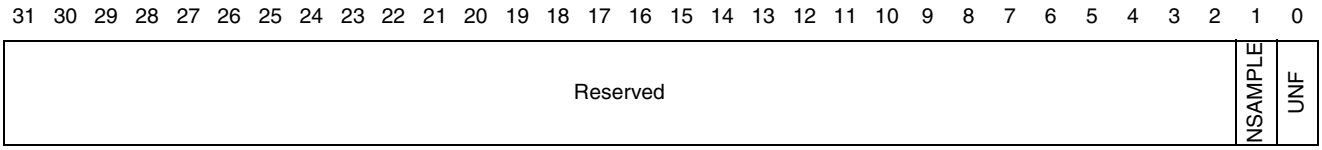
Address: *AudioPCM1BaseAddress* + 0x010
Type: RO
Reset: 0x0000 0000
Description: 1 indicates that the corresponding interrupt is enabled.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Enable interrupt for when a defined number of samples has been read from memory

[0] **UNF**
1: Enable FIFO underflow interrupt

AUD_PCMOUT1_IT_EN_SET PCm player 1 interrupt enable set



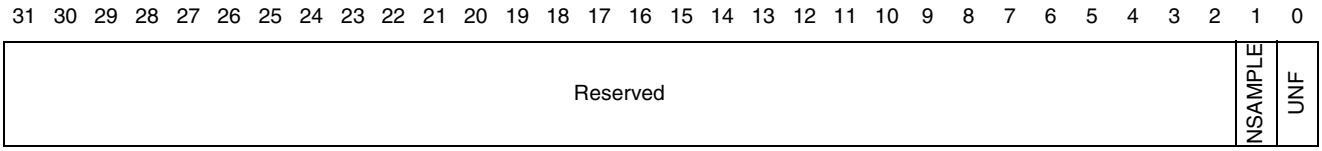
Address: *AudioPCM1BaseAddress* + 0x014
Type: R/W
Reset: 0x0000 0000
Description: 1 enable the corresponding interrupt

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Enable interrupt for when a defined number of samples has been read from memory

[0] **UNF**
1: Enable FIFO underflow interrupt

AUD_PCMOUT1_IT_EN_CLR PCM player 1 interrupt enable clear



Address: *AudioPCM1BaseAddress* + 0x018
Type: R/W
Reset: 0x0000 0000
Description: 1 clears the corresponding interrupt.

[31:2] **Reserved**: 0000

[1] **NSAMPLE**
1: Disable interrupt for when a defined number of samples has been read from memory

[0] **UNF**
1: Disable FIFO underflow interrupt

AUD_PCMOUT1_CTRL PCM player 1 control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSAMPLE																			SPDIF_LAT	CLK_DIV								RND	MEM_FMT	MODE	

Address: *AudioPCM1BaseAddress* + 0x01C

Type: R/W

Reset: 0x0000 0000

Description: Controls the PCM player

[31:13] **NSAMPLE**: Defines the number of samples to read from memory before an interrupt is generated
Each time the number of samples read by the PCM player is equal to this number, the associated bit in the status register is set to 1.

[12] **SPDIF_LAT**: S/PDIF latency
0: Ignore S/PDIF latency signal to start
1: Wait for S/PDIF latency signal to start

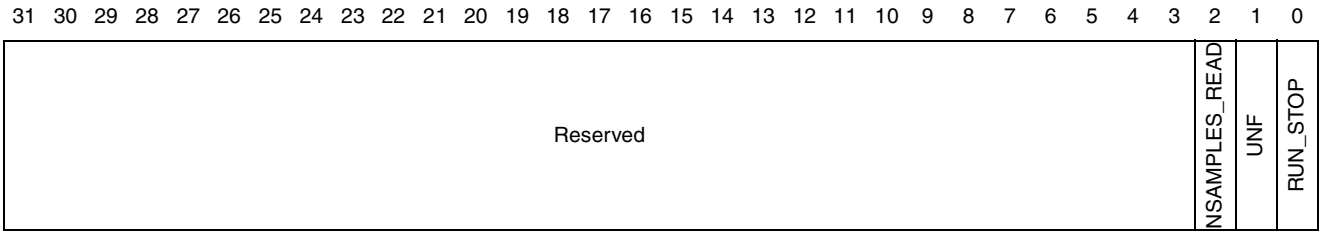
[11:4] **CLK DIV**: Frequency synthesizer clock divide
32 slots / subframe:
0: 1 x f_{FS}, 1: 128 x f_{FS}, 6: 192 x f_{FS}, 2: 256 x f_{FS}, 3: 384 x f_{FS}, 4: 512 x f_{FS}, 6: 768 x f_{FS}
16 slots / subframe:
0: 1 x f_{FS}, 2: 128 x f_{FS}, 3: 192 x f_{FS}, 4: 256 x f_{FS}, 6: 384 x f_{FS}, 8: 512 x f_{FS}, 12: 768 x f_{FS}

[3] **RND**: Rounding
0: No rounding
1: 16 bits rounding on PCM data

[2] **MEM_FMT**: Memory storage format
0: 16 bits / 0
1: 16 bits / 16 bits

[1:0] **Mode**
00: PCM off
01: PCM mute
10: PCM
11: CD

AUD_PCMOUT1_STA PCM Player 1 status



Address: *AudioPCM1BaseAddress* + 0x020

Type: R/W

Reset: 0x0000 0000

Description: Status of the PCM player.

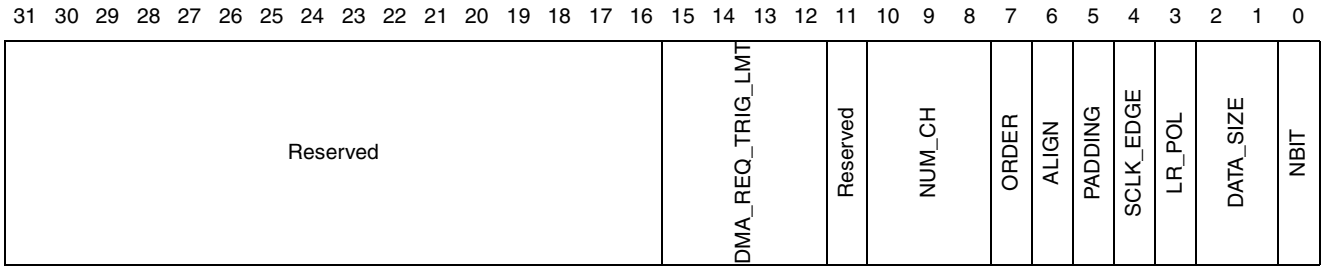
[31:3] **Reserved**

[2] **NSAMPLE**
1 Predefined number of samples have been read from memory

[1] **UNF**
1: FIFO underflow

[0] **RUN_STOP**
0: PCM player stopped
1: PCM player running

AUD_PCMOUT1_FMT PCM Player 1 data format



Address: *AudioPCM1BaseAddress* + 0x024

Type: R/W

Reset: 0x0000 0000

Description: Defines the data format.

[31:16] **Reserved**

[15:12] **DMA_REQ_TRIG_LMT**: DMA request trigger limit
A DMA request is generated as soon as the number of cells available in the FIFO is equal to this value.
Valid values: 2,4,6,8,10.
Reset value: 0000

[11] **Reserved**

[10:8] **NUM_CH**: Number of channels
Specifies the number of channels required at the output. Null data appears at the output for remaining unused channels.
001: One channel only (stereo)
010: Two channels only
011: Three channels only
100: Four channels only
101: All five channels
Reset value: 000

[7] **ORDER**: Bit ordering
0: Data is output LSBit first
1: Data is output MSBit first

[6] **ALIGN**
0: Data is left-aligned wrt to LR clock
1: Data is right-aligned wrt to LR clock

[5] **PADDING**:
0: Data is delayed by 1 clock cycle
1: Data is not delayed

[4] **SCLK_EDGE**: Active edge
0: Data is output on the rising edge of SCLK
1: Data is output on the falling edge of SCLK

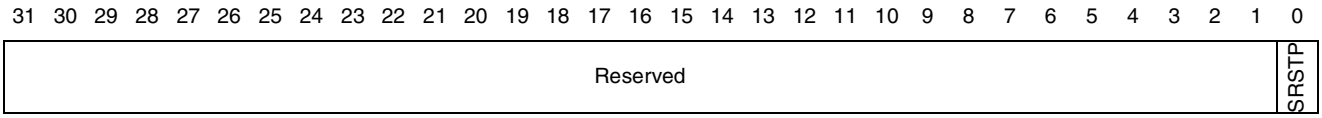
[3] **LR_POL**: Left-right clock polarity
0: Left word when LR clock is low
1: Left word when LR clock is high

[2:1] **DATA_SIZE**: Data size
00: 24 bits
01: 20 bits
10: 18 bits
11: 16 bits

[0] **NBIT**: Number of bits per subframe
0: 32 bits per subframe
1: 16 bits per subframe

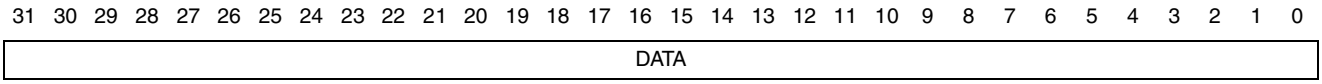
60.5 S/PDIF player

AUD_SPDIF_RST S/PDIF player soft reset



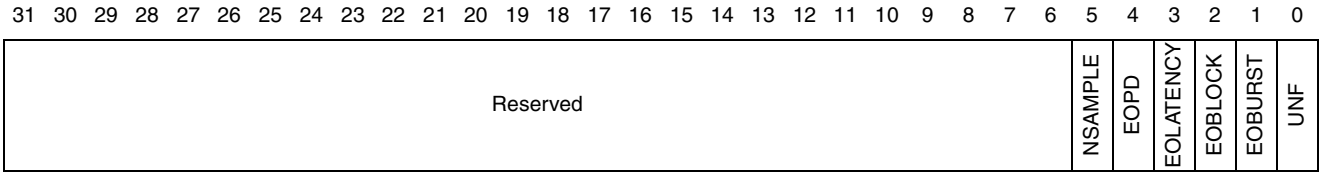
Address: *AudioSPDIFBaseAddress* + 0x000
Type: R/W
Reset: 0x0000 0000
Description: Soft reset the S/PDIF player - active high.

AUD_SPDIF_DATA S/PDIF player data FIFO



Address: *AudioSPDIFBaseAddress* + 0x004
Type: R/W
Reset: 0x0000 0000
Description: S/PDIF player data to be stored in the FIFO.

AUD_SPDIF_ITS S/PDIF player interrupt status



Address: *AudioSPDIFBaseAddress* + 0x008
Type: RO
Reset: 0x0000 0000
Description: 1 indicates an interrupt.

[31:6] **Reserved**: 0000

[5] **NSAMPLE**
1: The defined number of samples has been read from memory

[4] **EOPD**
1: End of Pd

[3] **EOLATENCY**
1: End of latency

[2] **EOBLOCK**
1: End of block

[1] **EODBURST**
1: End of data burst

[0] **UNF**
1: FIFO underflows

AUD_SPDIF_ITS_CLR S/PDIF player interrupt status clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										NSAMPLE_CLR	EOPD_CLR	EOLATENCY_CLR	EOBLK_CLR	EOBST_CLR	UNF_CLR

Address: *AudioSPDIFBaseAddress* + 0x00C

Type: R/W

Reset: 0x0000 000C

Description: 1 clears the corresponding interrupt.

[31:6] **Reserved**: 0000

[5] **NSAMPLE_CLR**
1: Clear the interrupt generated when the defined number of samples have been read from memory

[4] **EOPD_CLR**
1: Clear the end of Pd interrupt

[3] **EOLATENCY_CLR**
1: Clear the end of latency interrupt

[2] **EOBLK_CLR**
1: Clear the end of block interrupt

[1] **EOBST_CLR**
1: Clear the end of data burst interrupt

[0] **UNF_CLR**
1: Clear the FIFO underflow interrupt

AUD_SPDIF_IT_EN S/PDIF player interrupt enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										NSAMPLE	EOPD	EOLATENCY	EOBLOCK	EOBURST	UNF

Address: *AudioSPDIFBaseAddress* + 0x010

Type: RO

Reset: 0x0000 0000

Description: 1 indicates that the corresponding interrupt is enabled.

[31:6] **Reserved**: 0000

[5] **NSAMPLE**
1: Enable interrupt when a defined number of samples is read from memory

[4] **EOPD**
1: Enable end of Pd interrupt

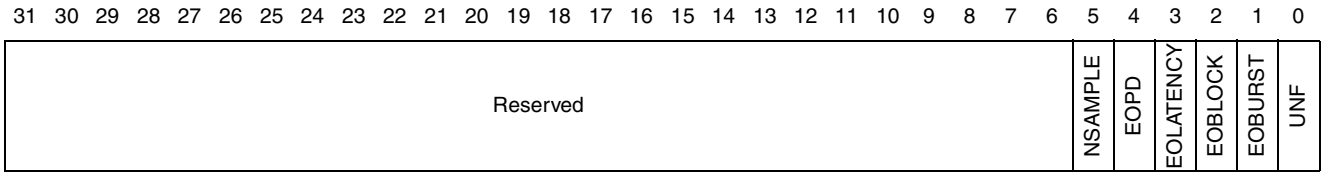
[3] **EOLATENCY**
1: Enable end of latency interrupt

[2] **EOBLOCK**
1: Enable end of block interrupt

[1] **EOBURST**
1: Enable end of data burst interrupt

[0] **UNF**
1: Enable FIFO underflow interrupt

AUD_SPDIF_IT_EN_SET S/PDIF player interrupt enable set



Address: *AudioSPDIFBaseAddress* + 0x014

Type: R/W

Reset: 0x0000 0000

Description: 1 enables the corresponding interrupt.

[31:6] **Reserved**: 0000

[5] **NSAMPLE**
1: Enable interrupt when a defined number of samples is read from memory

[4] **EOPD**
1: Enable end of Pd interrupt

[3] **EOLATENCY**
1: Enable end of latency interrupt

[2] **EOBLOCK**
1: Enable end of block interrupt

[1] **EOBURST**
1: Enable end of data burst interrupt

[0] **UNF**
1: Enable FIFO underflow interrupt

AUD_SPDIF_IT_EN_CLR S/PDIF player interrupt enable clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										NSAMPLE	EOPD	EOLATENCY	EOBLOCK	EOBURST	UNF

Address: *AudioSPDIFBaseAddress* + 0x018

Type: R/W

Reset: 0x0000 0000

Description: 1 clears the corresponding interrupt.

[31:6] **Reserved**: 0000

[5] **NSAMPLE**
1: Disable interrupt when a defined number of samples is read from memory

[4] **EOPD**
1: Disable end of Pd interrupt

[3] **EOLATENCY**
1: Disable end of latency interrupt

[2] **EOBLOCK**
1: Disable end of block interrupt

[1] **EOBURST**
1: Disable end of data burst interrupt

[0] **UNF**
1: Disable FIFO underflow interrupt

AUD_SPDIF_CTRL S/PDIF player control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMREAD																	STUFFING	Reserved	CLKDIV							RND	IDLE	MODE			

Address: *AudioSPDIFBaseAddress* + 0x01C

Type: R/W

Reset: 0x0000 0000

Description: This register controls the PCM player.

[31:15] **NSAMPLE**: Number of samples to read from memory before an interrupt is generated
Each time the number of samples read by the S/PDIF player is equal to this number, the associated bit in the status register is set to 1.

[14] **STUFFING**:
0: Stuffing done by software (encoded mode)
1: Stuffing done by hardware (encoded mode)

[13] **Reserved**:

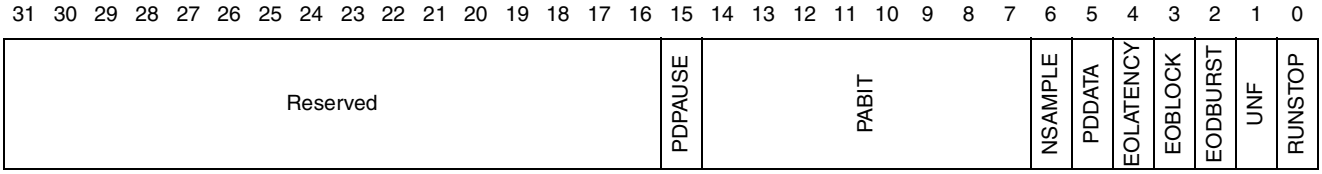
[12:5] **CLKDIV**: Frequency synthesizer clock divide
32 slots / subframe:
0: 1 x f_{FS}, 1: 128 x f_{FS}, 6: 192 x f_{FS}, 2: 256 x f_{FS}, 3: 384 x f_{FS}, 4: 512 x f_{FS}, 6: 768 x f_{FS}
16 slots / subframe:
0: 1 x f_{FS}, 2: 128 x f_{FS}, 3: 192 x f_{FS}, 4: 256 x f_{FS}, 6: 384 x f_{FS}, 8: 512 x f_{FS}, 12: 768 x f_{FS}

[4] **RND**: Rounding
0: No rounding
1: 16 bits rounding on PCM data

[3] **IDLE**: When 1, set the S/PDIF output in idle state

[2:0] **MODE**:
000: Off
001: Mute with PCM null data
010: Mute with pause burst
011: PCM
100: Encoded

AUD_SPDIF_STA S/PDIF player status



Address: *AudioSPDIFBaseAddress* + 0x020
Type: R/W
Reset: 0x0000 0000
Description: Status of the S/PDIF player.

[31:16] **Reserved**

[15] **PDPAUSE**
1: A Pd word is sent on S/PDIF output during pause-data bursts

[14:7] **PABIT**: Pa channel status bit number when Pause Data burst sent

[6] **NSAMPLE**
1: The predefined number of samples have been read from memory.

[5] **PDDATA**
1: A Pd word is sent on S/PDIF output during data bursts

[4] **EOLATENCY**
1: End of sub frame latency counter reached

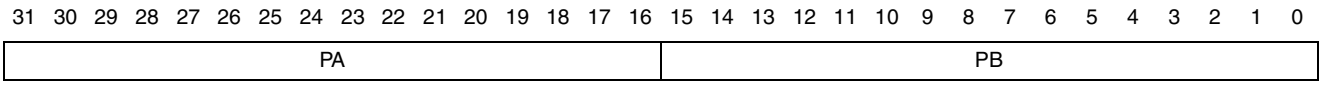
[3] **EOBLOCK**
1: End of block

[2] **EODBURST**
1: End of data burst

[1] **UNF**
1: FIFO underflow

[0] **RUNSTOP**
0: S/PDIF player stopped
1: S/PDIF player running

AUD_SPDIF_PA_PB S/PDIF player Pa, Pb burst

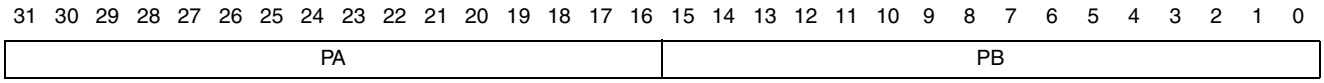


Address: *AudioSPDIFBaseAddress* + 0x024
Type: R/W
Reset: 0x0000 0000
Description: Defines the Pa and Pb parameters.

[31:16] **PA**: Pa sync word 1

[15:0] **PB**: Pb sync word 2

AUD_SPDIF_PC_PD S/PDIF player PC PD burst

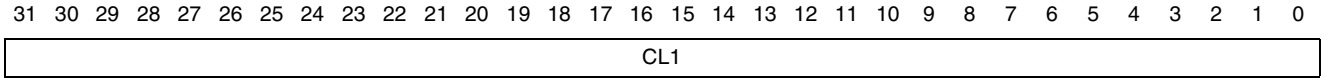


Address: *AudioSPDIFBaseAddress* + 0x028
Type: R/W
Reset: 0x0000 0000
Description: Defines the Pc and Pd parameters.

[31:16] **PC**: Encoded control word length in frames

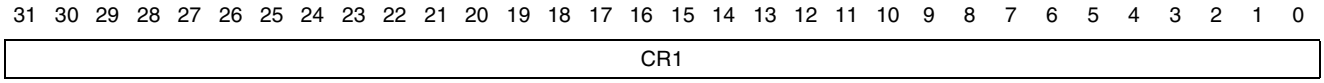
[15:0] **PD**: Encoded data length in frames

AUD_SPDIF_CL1 S/PDIF player left subframe channel status



Address: *AudioSPDIFBaseAddress* + 0x02C
Type: R/W
Reset: 0x0000 0000
Description: Defines the channel status for the left subframe (subframes 31 down to 0).

AUD_SPDIF_CR1 S/PDIF player right subframe channel status



Address: *AudioSPDIFBaseAddress* + 0x030
Type: R/W
Reset: 0x0000 0000
Description: Defines the channel status for the right subframe (subframes 31 down to 0).

AUD_SPDIF_CL2_CR2_UV S/PDIF player right/left subframe channel status and UV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												LU	LV	RU	RV	Reserved				CR2				Reserved				CL2			

Address: *AudioSPDIFBaseAddress* + 0x034

Type: R/W

Reset: 0x0000 0000

Description: Defines the channel status for the left and right subframes (subframes 35 down to 32). Also defines the user and validity bits.

[31:20] **Reserved**

[19] **LU**: User bit for left subframe

[18] **LV**: Validity bit for left subframe

[17] **RU**: User bit for right subframe

[16] **RV**: Validity bit for right subframe

[15:12] **Reserved**

[11:8] **CR2**: Channel status for left subframe - subframes 35 down to 32

[10:4] **Reserved**

[3:0] **CL2**: Channel status for left subframe - subframes 35 down to 32

AUD_SPDIF_PAU_LAT S/PDIF player pause and latency

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPD_BST																LAT															

Address: *AudioSPDIFBaseAddress* + 0x038

Type: R/W

Reset: 0x0000 0000

Description: Defines the number of pause-data bursts (including the stuffing) generated by the player before checking if the operation mode has changed or if data are available. Also defines the latency parameter.

[31:16] **NPD_BURST**: Number of pause data bursts

[15:0] **LAT**: Subframe counter maximum value for latency

AUD_SPDIF_FRA_LEN_BST S/PDIF player data and pause burst length

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBURST																PDBURST															

Address: *AudioSPDIFBaseAddress* + 0x038

Type: R/W

Reset: 0x0000 0000

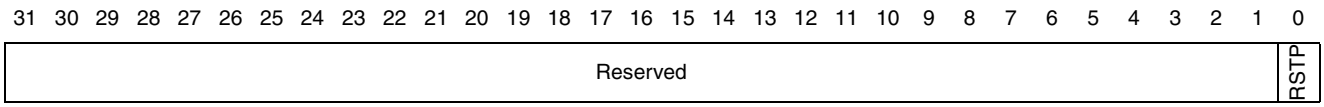
Description: Defines the length in frame units of the data and pause bursts (including the stuffing).

[31:16] **DBURST**: Data burst total length in frame units (repetition period between two data bursts)

[15:0] **PDBURST**: Pause burst total length in frame units (repetition period between two pause bursts)

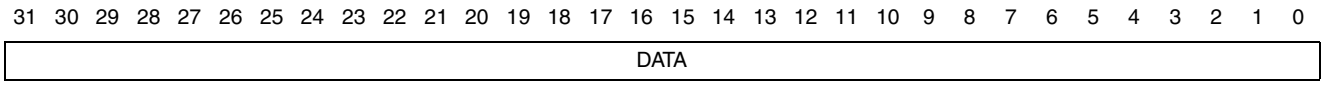
60.6 PCM reader

AUD_PCMIN_RST PCM reader soft reset



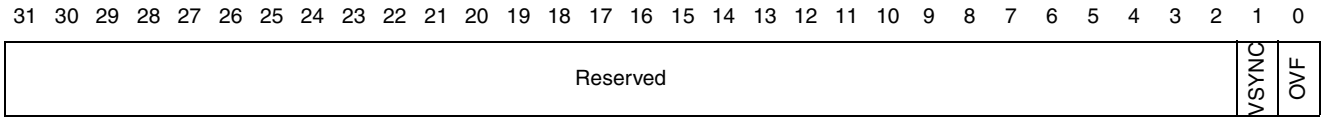
Address: *PCMReaderBaseAddress* + 0x000
Type: R/W
Reset: 0x0000 0000
Description: Soft reset the PCM player - active high.

AUD_PCMIN_DATA PCM reader data FIFO



Address: *PCMReaderBaseAddress* + 0x004
Type: R/W
Reset: 0x0000 0000
Description: PCM reader data from the FIFO.

AUD_PCMIN_ITS PCM reader interrupt status



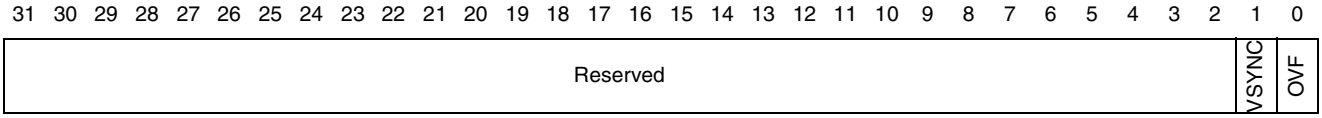
Address: *PCMReaderBaseAddress* + 0x008
Type: RO
Reset: 0x0000 0000
Description: 1 indicates an interrupt.

[31:29] **Reserved**: 0000

[1] **VSYNC**
1: A vertical sync rising edge has occurred

[0] **OVF**
1: FIFO overflow

AUD_PCMIN_ITS_CLR PCM reader interrupt status clear



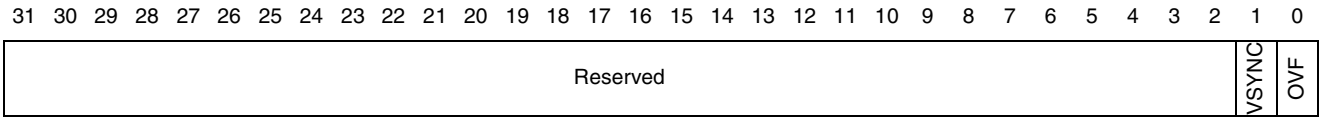
Address: *PCMReaderBaseAddress* + 0x00C
Type: R/W
Reset: 0x0000 000C
Description: 1 clears the corresponding interrupt.

[31:29] **Reserved**: 0000

[1] **VSYNC**
1: Clear the interrupt generated when a vertical sync rising edge occurs

[0] **OVF**
1: Clear the FIFO overflow interrupt

AUD_PCMIN_IT_EN PCM reader interrupt enable



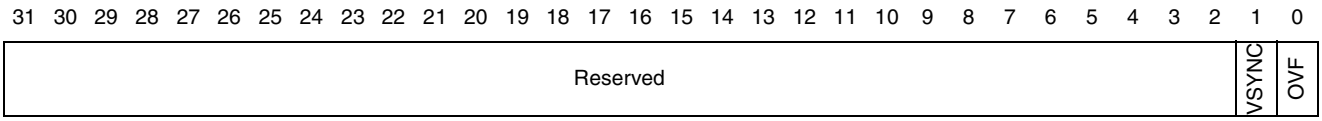
Address: *PCMReaderBaseAddress* + 0x010
Type: RO
Reset: 0x0000 0000
Description: 1 indicates that the corresponding interrupt is enabled.

[31:2] **Reserved**: 0000

[1] **VSYNC**
1: Enable vertical sync rising edge interrupt

[0] **OVF**
1: Enable FIFO overflow interrupt

AUD_PCMIN_IT_EN_SET PCM reader interrupt enable set



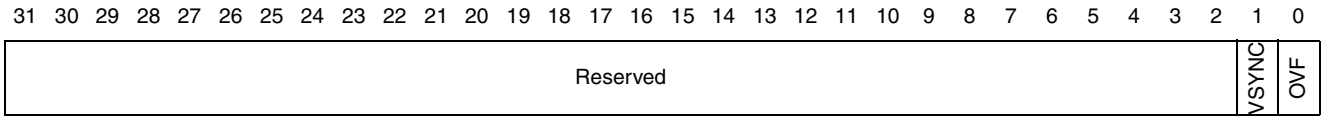
Address: *PCMReaderBaseAddress* + 0x014
Type: R/W
Reset: 0x0000 0000
Description: 1 enables the corresponding interrupt.

[31:2] **Reserved**: 0000

[1] **VSYNC**
1: Enable vertical sync rising edge interrupt

[0] **OVF**
1: Enable FIFO overflow interrupt

AUD_PCMIN_IT_EN_CLR PCM reader interrupt enable clear



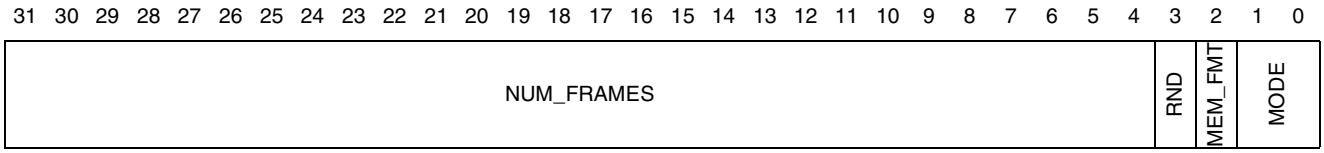
Address: *PCMReaderBaseAddress* + 0x018
Type: R/W
Reset: 0x0000 0000
Description: 1 clears the corresponding interrupt.

[31:2] **Reserved**: 0000

[1] **VSYNC**
1: Disable vertical sync rising edge interrupt

[0] **OVF**
1: Disable FIFO overflow interrupt

AUD_PCMIN_CTRL PCM reader control



Address: *PCMReaderBaseAddress* + 0x01C
Type: R/W
Reset: 0x0000 0000
Description: This register controls the PCM player.

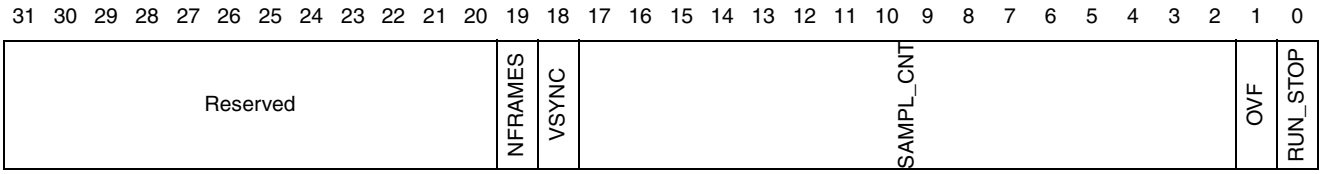
[31:4] **NUM_FRAMES**
Number of frames to be captured by the PCM reader before setting bit 19 of the status register

[3] **RND**
0: No rounding
1: 16 bits rounding on PCM data

[2] **MEM_FMT**: Memory storage format
0: 16 bits / 0
1: 16 bits / 16 bits

[1:0] **MODE**
0x: Off
10: PCM (use LR clock)
11: CD (do not use LR clock)

AUD_PCMIN_STA PCM reader status



Address: *PCMReaderBaseAddress* + 0x020

Type: R/W

Reset: 0x0000 0000

Description:

[31:20] **Reserved**

[19] **NFRAMES**
1: The predefined number of frames are captured.

[18] **VSYNC**: Vertical sync
Set to 1 on each vsync rising edge.

[17:2] **SAMPL_CNT**: Samples count
The reader counts the number of received audio frames, and on each rising edge of the vsync, the counter value is loaded in this register.

[1] **OVF**
1: FIFO overflow

[0] **RUN_STOP**
0: PCM reader stopped
1: PCM reader running

AUD_PCMIN_FMT PCM reader format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								ORDER	ALIGN	PADDING	SCLK_EDGE	LR_POL	DATA_SIZE		NBIT

Address: *PCMReaderBaseAddress* + 0x024

Type: R/W

Reset: 0x0000 0000

Description: Defines the data format.

[31:8] **Reserved**

[7] **ORDER**
0: Data is output LSB first
1: Data is output MSB first

[6] **ALIGN**
0: Data is right-aligned wrt to LR clock
1: Data is left-aligned wrt to LR clock

[5] **PADDING**
0: Data is delayed by one clock cycle
1: Data is not delayed

[4] **SCLK_EDGE**: Active edge
0: Data is output on the rising edge of SCLK
1: Data is output on the falling edge of SCLK

[3] **LR_POL**: Left-right clock polarity
0: Left word when LR clock is low
1: Left word when LR clock is high

[2:1] **DATA_SIZE**: data size
00: 24 bits
10: 18 bits
01: 20 bits
11: 16 bits

[0] **NBIT**: Number of bits per subframe
0: 32 bits per subframe
1: 16 bits per subframe

Part 8

Peripherals and interfaces

61 Asynchronous serial controller (ASC)

61.1 Overview

The asynchronous serial controller, also referred to as the UART interface, provides serial communication between the STx7100 and other microcontrollers, microprocessors or external peripherals. The STx7100 provides four ASCs, two of which are generally used by the smartcard controllers.

Parity generation, 8- or 9-bit data transfer and the number of stop bits is programmable. Parity, framing, and overrun error detection is provided to increase the reliability of data transfers. The transmission and reception of data can simply be double-buffered, or 16-deep FIFOs may be used. Handshaking is supported on both transmission and reception. For multiprocessor communication, a mechanism to distinguish the address from the data bytes is included. Testing is supported by a loop back option. A dual mode 16-bit baudrate generator provides the ASC with a separate serial clock signal.

Each ASC supports full duplex, asynchronous communication, where both the transmitter and the receiver use the same data frame format and the same baudrate. Data is transmitted on the transmit data output pin TXD and received on the receive data input pin RXD.

Each ASC can be set to operate in smartcard mode for use when interfacing to a smartcard.

61.2 Control

Register ASC_n_CTRL controls the operating mode of the ASC. It contains control and enable bits, error check selection bits, and status flags for error identification.

Serial data transmission or reception is only possible when the baudrate generator run bit (RUN) is set to 1. When the RUN bit is set to 0, TXD is 1. Setting the RUN bit to 0 immediately freezes the state of the transmitter and receiver and should only be done when the ASC is idle.

*Note: Programming the mode control field (*MODE*) to one of the reserved combinations results in unpredictable behavior.*

The ASC can be set to use either double-buffering or a 16-deep FIFO on transmission and reception.

61.2.1 Resetting the FIFOs

Registers ASC_n_TXRESET and ASC_n_RXRESET have no actual storage associated with them. A write of any value to one of these registers resets the corresponding FIFO.

61.2.2 Transmission and reception

Serial data transmission or reception is only possible when the baudrate generator run bit (RUN) is set to 1. A handshaking protocol is supported on both transmission and reception, using CTS and RTS signals.

A transmission is started by writing to the transmit buffer register ASC_n_TXBUFFER. Data transmission is either double buffered or uses a FIFO (selectable in register ASC_*n*_CTRL), therefore a new character may be written to the transmit buffer register before the transmission of the previous character is complete. This allows characters to be sent back to back without gaps.

Data reception is enabled by the receiver enable bit ASC_*n*_CTRL.RX_EN. After reception of a character has been completed, the received data and, if provided by the selected operating mode, the parity error bit, can be read from the receive buffer register, ASC_*n*_RXBUFFER.

Reception of a second character may begin before the received character has been read out of the receive buffer register. The overrun error status flag (OVERRUN_ERR) in the status register, ASC_*n*_STA, is set when the receive buffer register has not been read by the time the reception of a second character is completed. The previously received character in the receive buffer is overwritten, and the ASC_*n*_STA register is updated to reflect the reception of the new character.

The loop back option (selected by the LOOPBACK bit in register ASC_n_CTRL) internally connects the output of the transmitter shift register to the input of the receiver shift register. This may be used to test serial communication routines at an early stage without having to provide an external network.

61.3 Data frames

Data frames may be 8-bit or 9-bit, with or without parity and with or without a wake up bit. The data frame type is selected by setting the MODE bit field in the control register.

The transmitted data frame consists of three basic elements:

- start bit,
- data field (8 or 9 bits, least significant bit (LSB) first, including a parity bit or wake up bit, if selected),
- stop bits (0.5, 1, 1.5 or 2 stop bits).

61.3.1 8-bit data frames

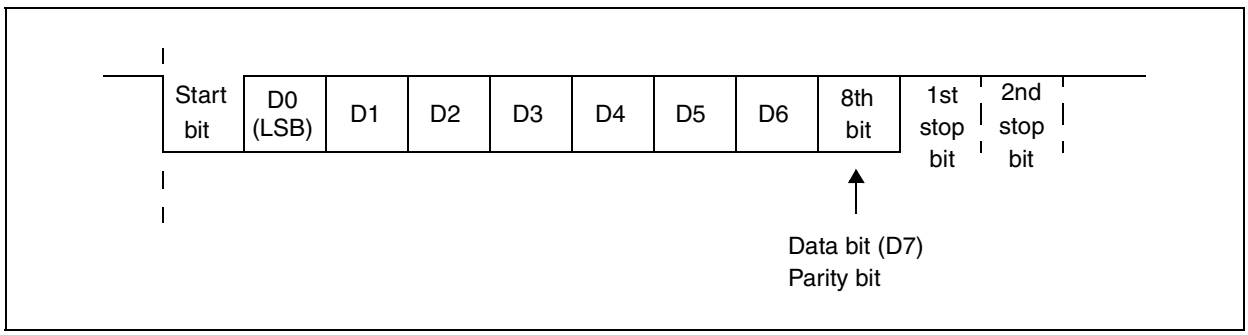
Figure 244 illustrates an 8-bit transmitted data frame. 8-bit frames may use one of the following formats:

- eight data bits D[0:7] (MODE set to 001),
- seven data bits D[0:6] plus an automatically generated parity bit (MODE set to 011).

Parity may be odd or even, depending on the bit ASC_n_CTRL.PARITYODD. If the modulo 2 sum of the seven data bits is 1, then the even parity bit is set and the odd parity bit is cleared.

In receive mode the parity error flag (PARITYERROR) is set if a wrong parity bit is received. The parity error flag is stored in the 8th bit (D7) of the ASC_n_RXBUFFER register. The parity error bit is set high if there is a parity error.

Figure 244: 8-bit Tx data frame format

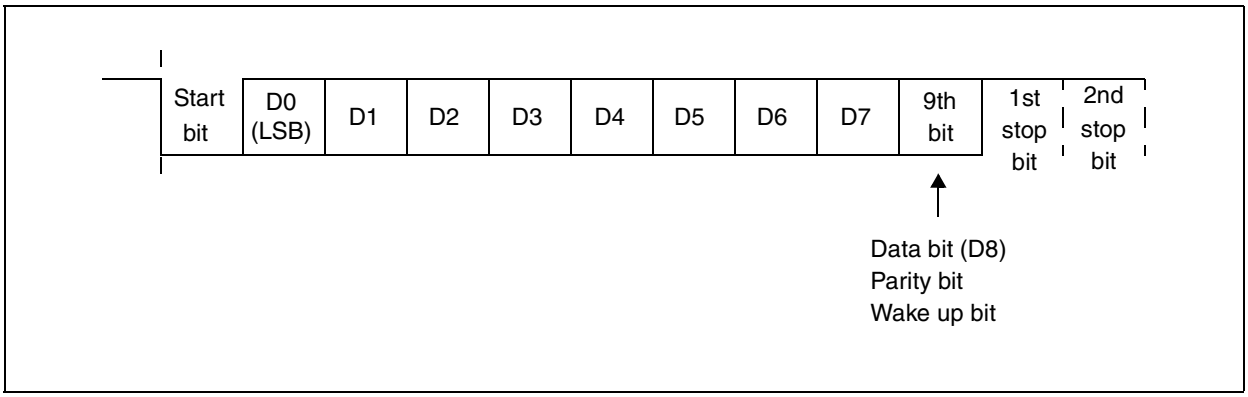


61.3.2 9-bit data frames

Figure 245 illustrates a 9-bit transmitted data frame. 9-bit data frames use of one of the following formats:

- nine data bits D[0:8] (MODE set to 100),
- eight data bits D[0:7] plus an automatically generated parity bit (MODE set to 111),
- eight data bits D[0:7] plus a wake up bit (MODE set to 101).

Figure 245: 9-bit Tx data frame format



Parity may be odd or even, depending on bit ASC_n_CTRL.PARITYODD. If the modulo 2 sum of the eight data bits is 1, then the even parity bit is set and the odd parity bit is cleared. The parity error flag (PARITYERROR) is set if a wrong parity bit is received. The parity error flag is stored in the 9th bit (D8) of the ASC_n_RXBUFFER register. The parity error bit is set high if there is a parity error.

In wake up mode, received frames are only transferred to the receive buffer register if the ninth bit (the wake up bit) is 1. If this bit is 0, no receive interrupt requests is activated and no data is transferred.

This feature may be used to control communication in multiprocessor systems. When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the additional ninth bit is 1 for an address byte and 0 for a data byte, so no slave is interrupted by a data byte. An address byte interrupts all slaves (operating in 8-bit data plus wake up bit mode), so each slave can examine the eight least significant bits (LSBs) of the received character, which is the address. The addressed slave switches to 9-bit data mode, which enables it to receive the data bytes that are coming (with the wake up bit cleared). The slaves that are not being addressed remain in 8-bit data plus wake up bit mode, ignoring the data bytes which follow.

61.4 Transmission

Transmission begins at the next baudrate clock tick, provided that the RUN bit is set and data has been loaded into the ASC_n_TX_BUFF. If bit ASC_n_CTRL.CTS_EN is set, then transmission only occurs when NOT_ASCnCTS is low.

The transmitter empty flag (TXEMPTY) indicates whether the output shift register is empty. It is set at the beginning of the last data frame bit that is transmitted, that is, during the first comms clock cycle of the first stop bit shifted out of the transmit shift register.

The loop back option (selected by bit ASC_n_CTRL.LOOPBACK) internally connects the output of the transmitter shift register to the input of the receiver shift register. This may be used to test serial communication routines at an early stage without having to provide an external network.

A transmission ends with stop bits (1 is output on TXD). When bit ASC_*n*_CTRL.SC_EN is 0, the length of these stop bits is determined by the setting of field ASC_*n*_CTRL.STOPBITS. This can either be for 0.5, 1, 1.5 or 2 baud clock periods. In smartcard mode, when bit ASC_*n*_CTRL.SC_EN is 1, the number of stop bits is determined by the value in ASC_n_ GUARDTIME.

61.4.1 Transmission with FIFOs enabled

The FIFOs are enabled by setting bit ASC_n_CTRL.FIFO_EN. The output FIFO is implemented as a 16-deep array of 9-bit vectors. Values to be transmitted are written to the output FIFO by writing to ASC_n_TX_BUFF.

Bit ASC_*n*_STA.TXFULL is set when the transmit FIFO is considered full, that is, when it contains 16 characters. Further writes to ASC_n_TXBUFFER fail to overwrite the most recent entry in the output FIFO. Bit ASC_n_STA.TXHALFEMPTY is set when the output FIFO contains eight or fewer characters.

Values are shifted out of the bottom of the output FIFO into a 9-bit output shift register in order to be transmitted. If the transmitter is idle (that is, the output shift register is empty) and something is written to the ASC_n_TXBUFFER so that the output FIFO becomes nonempty, the output shift register is immediately loaded from the output FIFO and transmission of the data in the output shift register begins at the next baudrate tick.

When the transmitter is just about to transmit the stop bits, and if the output FIFO is nonempty, the output shift register is immediately loaded from the output FIFO, and the transmission of this new data begins as soon as the current stop bit period is over (that is, the next start bit is transmitted immediately following the current stop bit period). If the output FIFO is empty at this point, the output shift register becomes empty. Thus back to back transmission of data can take place. Writing anything to ASC_n_TXRESET empties the output FIFO.

After changing the FIFO_EN bit, it is important to reset the FIFO to empty (by writing to the ASC_n_TXRESET register), or garbage may be transmitted.

61.4.2 Double buffered transmission

Double buffering is enabled and the FIFOs disabled by writing 0 to bit ASC_n_CTRL.FIFO_EN. When the transmitter is idle, the transmit data written into the transmit buffer ASC_n_TX_BUFF is immediately moved to the transmit shift register, thus freeing the transmit buffer for the next data to be sent. This is indicated by the transmit buffer empty flag (TX_HALFEMPTY) being set. The transmit buffer can be loaded with the next data while transmission of the previous data is still going on.

When the FIFOs are disabled, the TX_FULL bit is set when the buffer contains 1 character, and a write to ASC_n_TX_BUFF in this situation overwrites the contents. The TX_HALFEMPTY bit of the ASC_n_STA register is set when the output buffer is empty.

61.5 Reception

Reception is initiated by a falling edge on the data input pin RXD, provided that the RUN and RX_EN bits of the ASC_*n*_CTRL register are set.

Controlled data transfer can be achieved using the RTS handshaking signal provided by the UART. The sender checks the RTS to ensure the UART is ready to receive data. In double buffered reception RTS goes high when ASC_n_RXBUFFER is empty, in FIFO controlled operation it goes high when RXHALFFULL is zero.

The RXD pin is sampled at 16 times the rate of the selected baudrate. A majority decision of the first, second and third samples of the start bit determines the effective bit value. This avoids erroneous results that may be caused by noise.

If the detected value of the first bit of a frame is not 0, then the receive circuit is reset and waits for the next falling edge transition at the RXD pin. If the start bit is valid, that is 0, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register. For subsequent data and parity bits, the majority decision of the seventh, eighth and ninth samples in each bit time is used to determine the effective bit value. The effective values received on RXD are shifted into a 10-bit input shift register.

For 0.5 stop bits, the majority decision of the third, fourth, and fifth samples during the stop bit is used to determine the effective stop bit value. For 1 and 2 stop bits, the majority decision of the seventh, eighth, and ninth samples during the stop bits is used to determine the effective stop bit values. For 1.5 stop bits, the majority decision of the 15th, 16th, and 17th samples during the stop bits is used to determine the effective stop bit value.

Reception is stopped by clearing bit ASC_n_CTRL.RX_EN. Any currently received frame is completed, including the generation of the receive status flags. Start bits that follow this frame are not recognized.

61.5.1 Hardware error detection

To improve the safety of serial data exchange, the ASC provides three error status flags in the ASC_*n*_STA register which indicate if an error has been detected during reception of the last data frame and associated stop bits.

- The parity error bit (PARITYERROR) in the ASC_*n*_STA register is set when the parity check on the received data is incorrect.
  In FIFO operation parity errors on the buffers are ORed to yield a single parity error bit.
- The framing error bit (FRAMEERROR) in the ASC_*n*_STA register is set when the RXD pin is not 1 during the programmed number of stop bit times (see Section 61.5).
  In FIFO operation the bit remains set while at least one of the entries has a frame error.
- The overrun error bit (OVERRUNERROR) in the ASC_*n*_STA register is set when the input buffer is full and a character has not been read out of the ASC_n_RXBUFFER register before reception of a new frame is complete.

These flags are updated simultaneously with the transfer of data to the receive input buffer.

Frame and parity errors

The most significant bit (bit 9 of 0 to 9) of each input entry, records whether or not there was a frame error when that entry was received (that is, one of the effective stop bit values was 0). Bit ASC_*n*_STA.FRAME_ERR is set when the input buffer (double buffered operation), or at least one of the valid entries in the input buffering (FIFO controlled operation) has its most significant bit set.

If the mode is one where a parity bit is expected, then the next bit (bit 8 of 0 to 9) records whether there was a parity error when that entry was received. It does not contain the parity bit that was received. For 7-bit + parity data frames the parity error bit is set in both the eighth (bit 7 of 0 to 9) and the ninth (bit 8 of 0 to 9) bits. The PARITYERROR bit of ASC_*n*_STA is set when the input

buffer (double buffered operation), or at least one of the valid entries in the input buffering (FIFO controlled operation), has bit 8 set.

When receiving 8-bit data frames without parity (see Section 61.3.1 on page 775), the ninth bit of each input entry (bit 8 of 0 to 9) is undefined.

61.5.2 Input buffering modes

FIFO enabled reception

The FIFOs are enabled by setting bit ASC_n_CTRL.FIFO_EN. The input FIFO is implemented as a 16-deep array of 10-bit vectors (each 9 down to 0). If the input FIFO is empty, that is, no entries are present, bit ASC_n_STA.RX_BUFFULL is set to 0. If one or more FIFO entries are present, bit ASC_n_STA.RX_BUFFULL register is set to 1. If the input FIFO is not empty, a read from ASC_n_RX_BUFF gets the oldest entry in the input FIFO.

Bit ASC_n_STA.RX_HALFFULL is set when the input FIFO contains more than eight characters. Writing anything to ASC_n_RX_RST empties the input FIFO. As soon as the effective value of the last stop bit has been determined, the content of the input shift register is transferred to the input FIFO (except during wake up mode, in which case this happens only if the wake up bit, bit 8, is 1). The receive circuit then waits for the next falling edge transition at the RXD pin.

Bit ASC_n_STA.OVERRUN_ERR is set when the input FIFO is full and a character is loaded from the input shift register into the input FIFO. It is cleared when the ASC_n_RX_BUFF register is read.

After changing the FIFO_EN bit, it is important to reset the FIFO to empty by writing to the ASC_n_RXRESET register; otherwise the state of the FIFO pointers may be garbage.

Double buffered reception

Double buffered operation is enabled and the FIFOs disabled by writing 0 to bit ASC_*n*_CTRL.FIFO_EN. This mode can be seen as equivalent to a FIFO controlled operation with a FIFO of length 1 (the first FIFO vector is in fact used as the buffer). When the last stop bit has been received (at the end of the last programmed stop bit period) the content of the receive shift register is transferred to the receive data buffer register (ASC_n_RX_BUFF). The receive buffer full flag (RX_BUFFULL) is set, and the parity error (PARITY_ERR) and framing error (FRAME_ERR) flags are updated at the same time, after the last stop bit has been received (that is, at the end of the last stop bit programmed period), the flags are updated even if no valid stop bits have been received. The receive circuit then waits for the next falling edge transition at the RXD pin.

61.5.3 Time out mechanism

The ASC contains an 8-bit time out counter. This reloads from ASC_n_TIMEOUT whenever one or more of the following is true:

- ASC_n_RXBUFFER is read,
- the ASC is in the middle of receiving a character,
- ASC_n_TIMEOUT is written to.

If none of these conditions hold the counter decrements towards 0 at every baudrate tick.

The TIMEOUTNOTEMPTY bit of the ASC_*n*_STA register is 1 when the input FIFO is not empty and the time out counter is zero.

The TIMEOUTIDLE bit of the ASC_*n*_STA register is 1 when the input FIFO is empty and the time out counter is zero.

The effect of this is that whenever the input FIFO has got something in it, the time out counter decrements until something happens to the input FIFO. If nothing happens, and the time out counter reaches zero, the TIMEOUTNOTEMPTY bit of the ASC_*n*_STA register is set.

When the software has emptied the input FIFO, the time out counter resets and starts decrementing. If no more characters arrive, when the counter reaches zero the TIMEOUTIDLE bit of the ASC_*n*_STA register is set.

61.6 Baudrate generation

Each ASC has its own dedicated 16-bit baudrate generator with 16-bit reload capability. The baudrate generator has two possible modes of operation.

The ASC_n_BAUDRATE register is the dual function baudrate generator and reload value register. A read from this register returns the content of the counter or accumulator (depending on the mode of operation); writing to it updates the reload register.

If bit ASC_n_CTRL.RUN register is 1, then any value written in register ASC_n_BAUDRATE is immediately copied to the counter/accumulator. However, if the RUN bit is 0 when the register is written, then the counter/accumulator is not reloaded until the first comms clock cycle after the RUN bit is 1.

The baudrate generator supports two modes of operation, offering a wide range of possible values. The mode is set via bit ASC_*n*_CTRL.BAUDMODE. Mode 0 is a simple counter driven by the comms clock whereas mode 1 uses a loop back accumulator. Mode 0 is recommended for low baudrates (below 19.2 Kbaud), where its error deviation is low, and mode 1 is recommended for baudrates above 19.2 Kbytes.

61.6.1 Baudrates

The baudrate generator provides an internal oversampling clock at 16 times the external baudrate. This clock only ticks if the bit ASC_n_CTRL.RUN is set to 1. Setting this bit to 0 immediately freezes the state of the ASC's transmitter and receiver.

Mode 0

When bit ASC_n_CTRL.BAUDMODE is set to 0, the baudrate and the required reload value for a given baudrate can be determined by the following formulae:

$$BaudRate = \frac{f_{comms}}{16 \times ASCBaudRate}$$

$$ASCBaudRate = \frac{f_{comms}}{16 \times BaudRate}$$

where:

- *ASCBaudRate* represents the content of the ASC_n_BAUDRATE reload value register, taken as an unsigned 16-bit integer,
- f_{comms} is the frequency of the comms clock (clock channel PLL_CLK[2], see Chapter 15: *Clocks on page 120*).

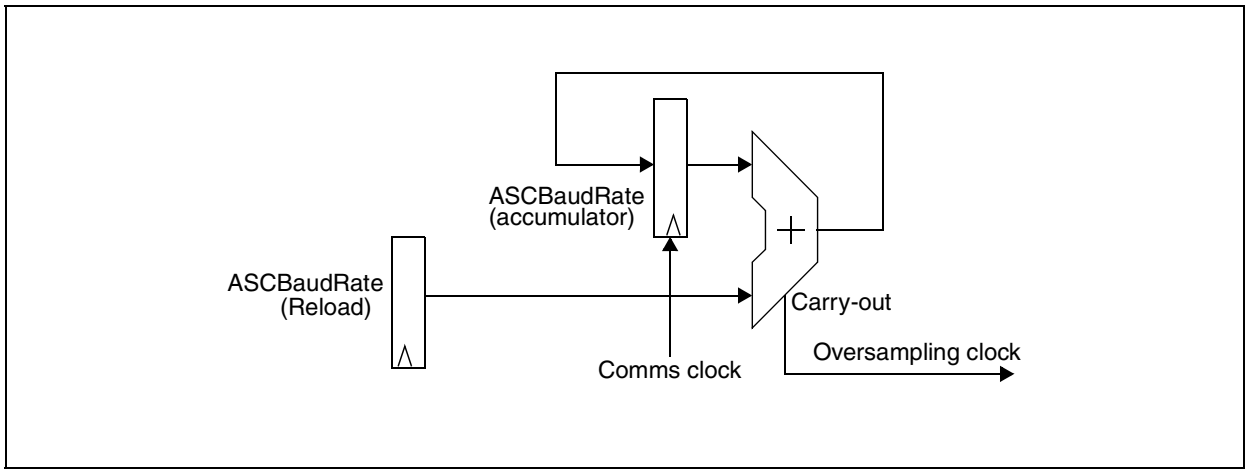
The baudrate counter is clocked by the comms clock. It counts downwards and can be started or stopped by bit ASC_*n*_CTRL.RUN. Each underflow of the timer provides one oversampling baudrate clock pulse. The counter is reloaded with the value stored in its 16-bit reload register each time it underflows.

Writes to register ASC_n_BAUDRATE update the reload register value. Reads from the ASC_n_BAUDRATE register return the current value of the counter.

Mode 1

When bit ASC_n_CTRL.BAUDMODE is set to 1, the baudrate is controlled by the circuit in Figure 246.

Figure 246: Baudrate in mode 1



The CPU writes go to ASC_n_BAUDRATE to the reload register. The CPU then reads from ASC_n_BAUDRATE and returns the value in the accumulator register. Both registers are 16 bits wide and are clocked by the comms clock (PLL_CLK[2]).

Writing a value of *ASCBaudRate* to the ASC_n_BAUDRATE register results in an average oversampling clock frequency of:

$$\frac{ASCBaudRate \times f_{comms}}{2^{16}}$$

So the baudrate is given by:

$$BaudRate = \frac{ASCBaudRate \times f_{comms}}{16 \times 2^{16}}$$

This gives good granularity, and hence low baudrate deviation errors, at high baudrate frequencies.

61.7 Interrupt control

Each ASC contains two registers that are used to control interrupts, the status register (ASC_*n*_STA) and the interrupt enable register (ASC_n_INT_EN). The status bits in the ASC_*n*_STA register show the cause of any interrupt. The interrupt enable register allows certain interrupt causes to be masked. Interrupts occur when a status bit is 1 (high) and the corresponding bit in the ASC_n_INT_EN register is 1.

The ASC interrupt signal is generated from the OR of all interrupt status bits after they have been ANDed with the corresponding enable bits in the ASC_*n*_INT_EN register, as shown in Figure 247.

The status bits cannot be reset by software because the ASC_*n*_STA register cannot be written to directly. Status bits are reset by operations performed by the interrupt handler:

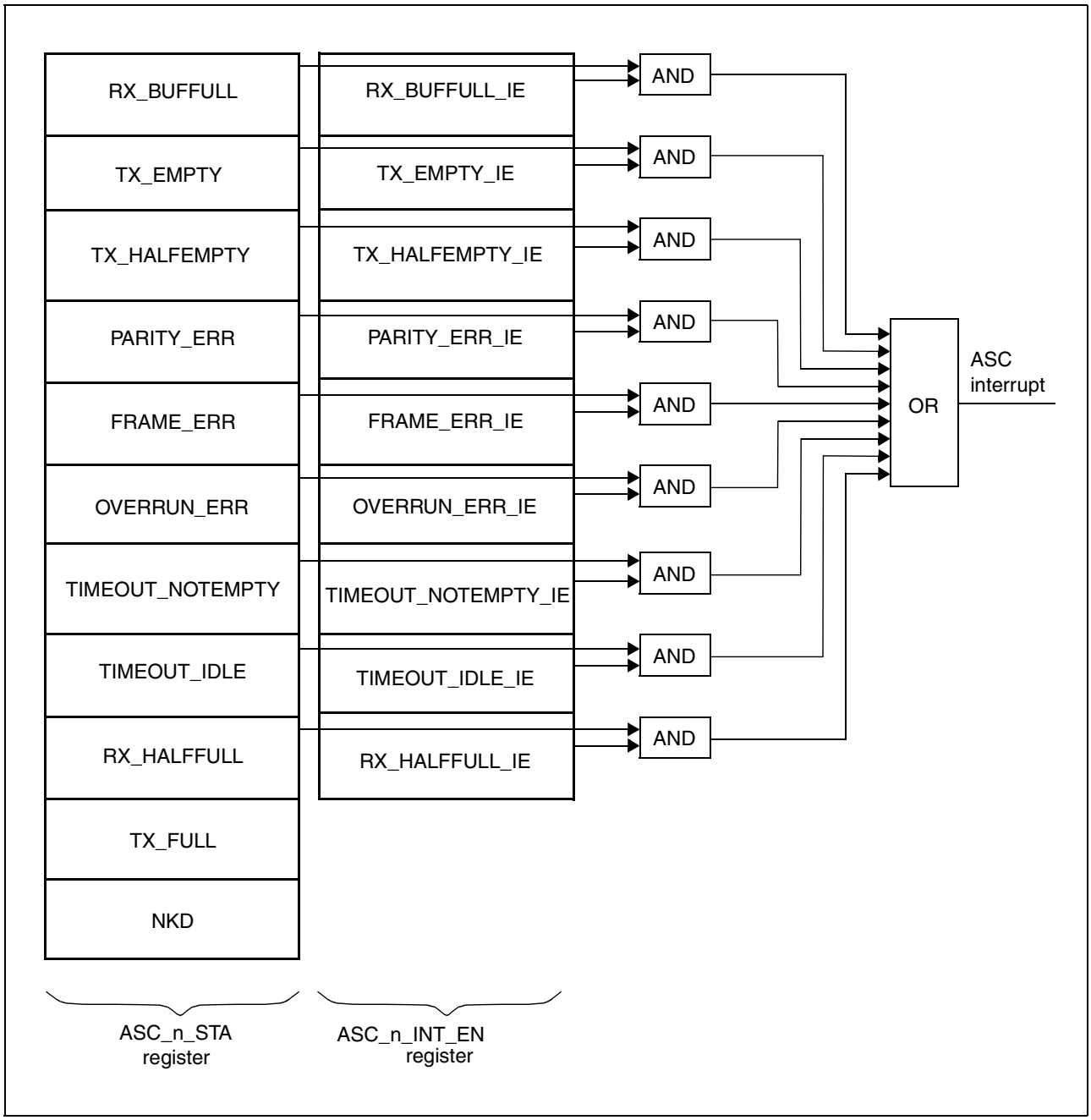
- transmitter interrupt status bits (TX_EMPTY and TX_HALFEMPTY) are reset when a character is written to the transmitter buffer,
- receiver interrupt status bit (RX_BUFFULL) is reset when a character is read from the receive buffer,
- PARITY_ERR and FRAME_ERR status bits are reset when all characters containing errors have been read from the receive input buffer,
- The OVERRUN_ERR status bit is reset when a character is read from ASC_*n*_RXBUFF.

61.7.1 Using the ASC interrupts when FIFOs are disabled (double buffered operation)

The transmitter generates two interrupts; this provides advantages for the servicing software. For normal operation (that is, other than the error interrupt) when FIFOs are disabled the ASC provides three interrupt requests to control data exchange via the serial channel:

- TX_HALFEMPTY is activated when data is moved from ASC_n_TXBUFFER to the transmit shift register,
- TX_EMPTY is activated before the last bit of a frame is transmitted,
- RX_BUFFULL is activated when the received frame is moved to ASC_*n*_RX_BUFF.

Figure 247: ASC status and interrupt registers



As shown in Figure 247, TX_HALFEMPTY is an early trigger for the reload routine, while TX_EMPTY indicates the completed transmission of the data field of the frame. Therefore, software using handshake should rely on TX_EMPTY at the end of a data block to make sure that all data has really been transmitted.

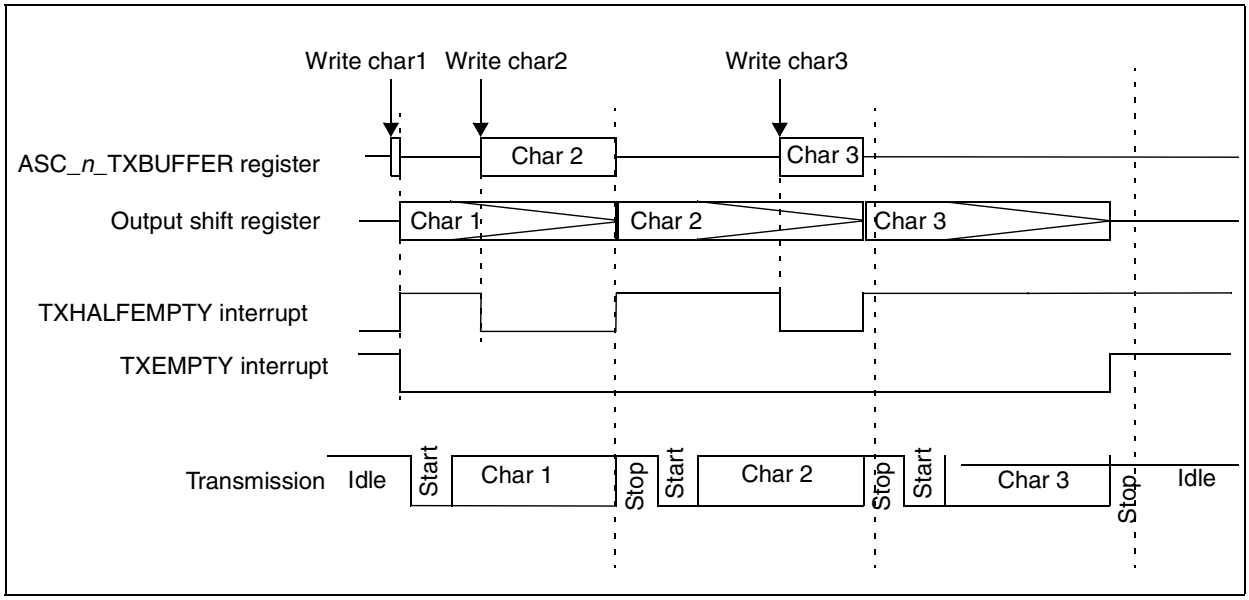
For single transfers it is sufficient to use the transmitter interrupt (TX_EMPTY), which indicates that the previously loaded data has been transmitted, except for the last bit of a frame.

For multiple back to back transfers it is necessary to load the next data before the last bit of the previous frame has been transmitted. The use of TX_EMPTY alone would leave just one stop bit time for the handler to respond to the interrupt and initiate another transmission. Using the output buffer interrupt (TX_HALFEMPTY) to signal for more data allows the service routine to load a complete frame, as ASC_n_TX_BUFF may be reloaded while the previous data is still being transmitted.

61.7.2 Using the ASC interrupts when FIFOs are enabled

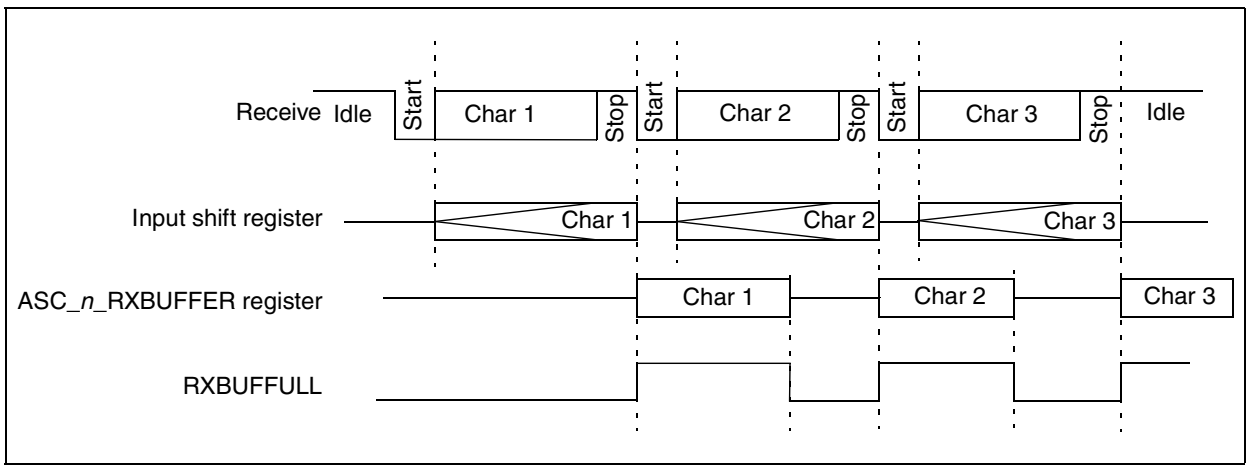
To transmit a large number of characters back to back, the driver routine initially writes 16 characters to ASC_*n*_TX_BUFF. Then, every time a TXHALFEMPTY interrupt fires, it writes eight more. When there is nothing more to send, a TXEMPTY interrupt tells the driver that everything has been transmitted.

Figure 248: ASC transmission



When receiving, the driver can use RXBUFFULL to interrupt every time a character arrives. Alternatively, if data is coming in back to back, it can use RXHALFFULL to interrupt it when there are more than eight characters in the input FIFO to read. It has as long as it takes to receive eight characters to respond to this interrupt before data overruns. If less than eight characters stream in, and no more are received for at least a time out period, the driver can be woken up by one of the two time out interrupts, TIMEOUTNOTEMPTY or TIMEOUTIDLE.

Figure 249: ASC reception



61.8 Smartcard operation

Smartcard mode is selected by setting bit ASC_n_CTRL.SC_EN to 1. In smartcard mode the RXD and TXD ports of the UART are both connected externally via a single bidirectional line to a smartcard I/O port. Characters are transferred to and from the smart card as 8-bit data frames with parity (see Section 61.3 on page 775). Handshaking between the UART and the smartcard ensures secure data transfer.

The UART supports both T=0 and T=1 protocol. In T=0 protocol, the reception of parity errors by either the UART or the smartcard is signalled by the automatic transmission of a NACK, where the receiver pulls the data line low, 0.5 baud clock periods after the end of the parity bit. The UART supports the reception and transmission of such NACKs. In T=1 protocol, this NACK behavior is not required, and any such behavior on the part of the UART can be disabled by setting the ASC_n_ CONTROL bit NACKDISABLE.

When bit ASC_n_CTRL.SC_EN is set to 0, normal UART operation occurs.

Smartcard operation complies with the ISO smartcard specification except where noted (see Section 61.8.4).

61.8.1 Control registers

ASC_N_GUARDTIME

The programmable 9-bit register ASC_n_GUARDTIME controls the time between transmitting the parity bit of a character and the start bit of any further bytes, or transmitting a NACK (no acknowledge signal, see Handshaking). During the guardtime period the UART receiver is insensitive to possible start bits and the smartcard is free to send NACKs.

The guardtime is effectively the number of stop bits to use when transmitting in smart card mode. Programming a value of 0 is undefined. Any positive value < 512 is possible.

The guardtime mentioned here is different from the guardtime mentioned in ISO7816. In fact to achieve a particular guardtime value, the guardtime should be programmed with the following value:

Guardtime = *guardtime* + 2 (mod 256)

In particular, this applies to the special case of guardtime = 255, where effectively, the number of stop bits is 1.

Note: If guardtime = 255 then any NACKs from the smart card might conflict with subsequent transmitted start bits, so it is assumed that the smart card is not sending NACKs in this case (T=1 protocol is being used for example). It is also important that the UART should be programmed in 0.5 stop bit mode, so that it does not see a subsequent start bit as a frame error (that is a NACK). So when guardtime = 255, the UART should be programmed in 0.5 stop bit mode.

Guardtime should always be set to at least two.

61.8.2 Transmission

In smartcard mode FIFOs can be either enabled or disabled. If FIFOs are disabled, the UART transmission behaves according to NDC requirements.

Handshaking

When the UART is transmitting data to the smartcard, the smartcard can NACK (not acknowledge) the transmission by pulling the line low, 0.5 baud clock periods into the guardtime period and holding it low for at least 1 baud clock period. The UART should also be programmed in 1.5 stop bit mode, and since it receives what it transmits, NACKs is detected as receive framing errors.

Behavior with FIFOs enabled

At about 1 baud clock period into the guardtime period, the UART knows whether or not the transmitted character has been NACKed. If no NACK has been received and the Tx FIFO is not empty, the next character is transmitted after the guardtime period.

If a transmitted character is NACKed by the receiving UART, the character is retransmitted as soon as the guardtime period expires (or if guardtime is two, an extra baud clock period later), and retransmission is attempted up to the number of retries set in the ASC_n_RETRIES register. If the last retry is also NACKed the Tx FIFO is emptied, putting the transmitter into an idle state, and the NKD bit is set in the ASC_n_STA register.

Emptying the FIFO causes an interrupt, which can be handled by software. The NKD bit in the ASC_*n*_STA register can be reset by writing to the ASC_*n*_TX_RST register.

All unNACKed (successfully transmitted) data is looped back into the receive FIFO. This FIFO can be read by software to determine the status of the data transmission.

Behavior with FIFOs disabled

When the smartcard mode bit is set to 1, the following operation occurs.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- If a parity error is detected during reception of a frame programmed with a 1/2 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame, that is, at the end of the 1/2 stop bit period. This is to indicate to the smartcard that the data transmitted to the UART has not been correctly received.
- The assertion of the TX_EMPTY interrupt can be delayed by programming the ASC_n_GUARDTIME register. In normal operation, TX_EMPTY is asserted when the transmit shift register is empty and no further transmit requests are outstanding.
- The receiver enable bit in the ASC_*n*_CTRL register is automatically reset after a character has been transmitted. This avoids the receiver detecting a NACK from the smartcard as a start bit.

In smartcard mode an empty transmit shift register triggers the guardtime counter to count up to the programmed value in the ASC_*n*_GUARDTIME register. TX_EMPTY is forced low during this time. When the guardtime counter reaches the programmed value TX_EMPTY is asserted high.

The de-assertion of TX_EMPTY is unaffected by smartcard mode.

61.8.3 Reception

Reception can be done with FIFOs either enabled or disabled. The behavior is the same as in normal (nonsmartcard) mode except that if a parity error occurs then, providing the transmitter is idle, and bit ASC_*n*_CTRL.NACKDISABLE is 0, the UART transmits a NACK on the TXD for one baud clock period from the end of the received stop bit. RXD is masked when transmitting a NACK, since TXD is tied to RXD and a NACK must not be seen as a start bit.

If bit ASC_*n*_CTRL.NACKDISABLE is 1 then no automatic NACK generation takes place.

61.8.4 Divergence from ISO smartcard specification

This UART does not support guardtimes of 0 or 1, and does not have any special behavior for a guardtime of 255.

62 Asynchronous serial controller (ASC) registers

The registers for each ASC are grouped in 4 Kbyte blocks, with the base of the block for ASC number *n* at the address *ASCnBaseAddress*.

Register addresses are provided as *ASCnBaseAddress* + offset.

The *ASCnBaseAddresses* are:

UART0: 0x1803 0000,
UART1: 0x1803 1000,
UART2: 0x1803 2000,
UART3: 0x1803 3000.

There is also one enable register located in the infrared blaster block. This is provided as *IRBBaseAddress* + offset.

The *IRBBaseAddress* is:

0x1811 5000.

Table 203: ASC register summary

Register	Description	Offset	Type
ASC_n_BAUDRATE	ASCn baudrate generator	0x000	R/W
ASC_n_TX_BUFF	ASCn transmit buffer	0x004	WO
ASC_n_RX_BUFF	ASCn receive buffer	0x008	RO
ASC_n_CTRL	ASCn control	0x00C	R/W
ASC_n_INT_EN	ASCn interrupt enable	0x010	R/W
ASC_n_INT_STA	ASCn interrupt status	0x014	RO
ASC_n_GUARDTIME	ASCn guard time	0x018	R/W
ASC_n_TIMEOUT	ASCn time out	0x01C	R/W
ASC_n_TX_RST	ASCn transmit FIFO reset	0x020	WO
ASC_n_RX_RST	ASCn receive FIFO reset	0x024	WO
ASC_n_RETRIES	ASCn number of retries on transmission	0x028	R/W

ASC_n_BAUDRATE ASCn baudrate generator

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RELOAD_VAL															

Address: *ASCnBaseAddress* + 0x000

Type: R/W

Reset: 1

Description: This register is the dual function baudrate generator and reload value register. A read from this register returns the content of the 16-bit counter/accumulator; writing to it updates the 16-bit reload register.

If bit ASC_n_CTRL.RUN is 1, then any value written in the ASC_n_BAUDRATE register is immediately copied to the timer. However, if the RUN bit is 0 when the register is written, then the timer is not reloaded until the first comms clock cycle after the RUN bit is 1.

The mode of operation of the baudrate generator depends on the setting of bit ASC_*n*_CTRL.BAUDMODE.

Mode 0

When bit ASC_*n*_CTRL.BAUDMODE is set to 0, the baudrate and the required reload value for a given baudrate can be determined by the following formulae:

$$BaudRate = \frac{f_{comms}}{16 \times ASCBaudRate}$$

$$ASCBaudRate = \frac{f_{comms}}{16 \times BaudRate}$$

where: *ASCBaudRate* represents the content of the ASC_n_BAUDRATE register, taken as an unsigned 16-bit integer,

f_{comms} is the frequency of the comms clock (clock channel PLL_CLK[2]).

Mode 0 should be used for all baudrates below 19.2 Kbaud.

Table 204 lists commonly used baudrates with the required reload values and the approximate deviation errors for an example baudrate with a comms clock of 60 MHz.

Table 204: Mode 0 baudrates

Baudrate	Reload value (exact)	Reload value (integer)	Reload value (hex)	Approximate deviation error (%)
38.4 K	97.656	98	0x0062	0.35
19.2 K	195.313	195	0x00C3	0.16
9600	390.625	391	0x0187	0.1
4800	781.250	781	0x030D	0.03
2400	1562.500	1563	0x061B	0.03
1200	3125.000	3125	0x0C35	0.00
600	6250.000	6250	0x186A	0.00
300	12500.000	12500	0x30D4	0.00
75	50000.000	50000	0xC350	0.00

Mode 1

When bit ASC_n_CTRL.BAUDMODE is set to 1, the baudrate is given by:

$$BaudRate = \frac{ASCBaudRate \times f_{comms}}{16 \times 2^{16}}$$

where: f_{comms} is the comms clock frequency and *ASCBaudRate* is the value written to the ASC_n_BAUDRATE register. Mode 1 should be used for baudrates of 19.2 Kbytes and above as it has a lower deviation error than mode 0 at higher frequencies.

Table 205: Mode 1 baudrates

Baudrate	Reload value (exact)	Reload value (integer)	Reload value (hex)	Approximate deviation error (%)
115200	2013.266	2013	0x07DD	0.01
96000	1677.722	1678	0x068E	0.02
38.4 K	671.089	671	0x029F	0.02
19.2 K	335.544	336	0x0150	0.14

ASC_n_TX_BUFF ASCn transmit buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TD								

Address: *ASCnBaseAddress* + 0x004

Type: WO

Reset: 0

Description: A transmission is started by writing to the transmit buffer register ASC_n_TXBUFFER. Serial data transmission is only possible when the baudrate generator bit ASC_n_CTRL.RUN is set to 1.

Data transmission is double buffered or uses a FIFO, so a new character may be written to the transmit buffer register before the transmission of the previous character is complete. This allows characters to be sent back to back without gaps.

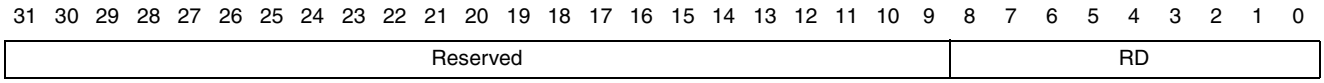
[31:9] **Reserved**

[8] **TD[8]**
Transmit buffer data D8, or parity bit, or wake up bit or undefined depending on the operating mode (the setting of field ASC_n_CTRL.MODE).
If the MODE field selects an 8-bit frame then this bit should be written as 0.

[7] **TD[7]**
Transmit buffer data D7, or parity bit depending on the operating mode (the setting of field ASC_n_CTRL.MODE).

[6:0] **TD[6:0]:** Transmit buffer data D6 to D0

ASC_n_RX_BUFF ASCn receive buffer



Address: *ASCnBaseAddress* + 0x008

Type: RO

Reset: 0

Description: Serial data reception is only possible when the baudrate generator bit ASC_n_CTRL.RUN is set to 1.

[31:9] **Reserved**

[8] **RD[8]**
Receive buffer data D8, or parity error bit, or wake up bit depending on the operating mode (the setting of field ASC_n_CTRL.MODE)
If the MODE field selects an 8-bit frame then this bit is undefined. Software should ignore this bit when reading 8-bit frames

[7] **RD[7]**
Receive buffer data D7, or parity error bit depending on the operating mode (the setting of field ASC_n_CTRL.MODE)

[6:0] **RD[6:0]**
Receive buffer data D6 to D0

ASC_n_CTRL ASCn control

31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NACK_DISABLE	BAUDMODE	CTS_EN	FIFO_EN	SC_EN	RX_EN	RUN	LOOPBACK	PARITYODD	STOPBITS		MODE		

Address: *ASCnBaseAddress* + 0x00C

Type: R/W

Reset: 0

Description: This register controls the operating mode of the UART ASCn and contains control bits for mode and error check selection, and status flags for error identification.

Programming the mode control field (MODE) to one of the reserved combinations may result in unpredictable behavior. Serial data transmission or reception is only possible when the baudrate generator run bit (RUN) is set to 1. When the RUN bit is set to 0, TXD is 1. Setting the RUN bit to 0 immediately freezes the state of the transmitter and receiver. This should only be done when the ASC is idle.

Serial data transmission or reception is only possible when the baudrate generator RUN bit is set to 1. A transmission is started by writing to the transmit buffer register ASC_n_TXBUFFER.

[31:14] **Reserved**

[13] **NACK_DISABLE:** NACKing behavior control
0: NACKing behavior in smartcard mode
1: No NACKing behavior in smartcard mode

[12] **BAUDMODE:** Baudrate generation mode
0: Baud counter decrements, ticks when it reaches 1 1: Baud counter added to itself, ticks when there is a carry

[11] **CTS_EN:** CTS enable
0: CTS ignored 1: CTS enabled

[10] **FIFO_EN:** FIFO enable:
0: FIFO disabled 1: FIFO enabled

[9] **SC_EN:** Smartcard enable
0: Smartcard mode disabled 1: Smartcard mode enabled

[8] **RX_EN:** Receiver enable bit
0: Receiver disabled 1: Receiver enabled

[7] **RUN:** Baudrate generator run bit
0: Baudrate generator disabled (ASC inactive) 1: Baudrate generator enabled

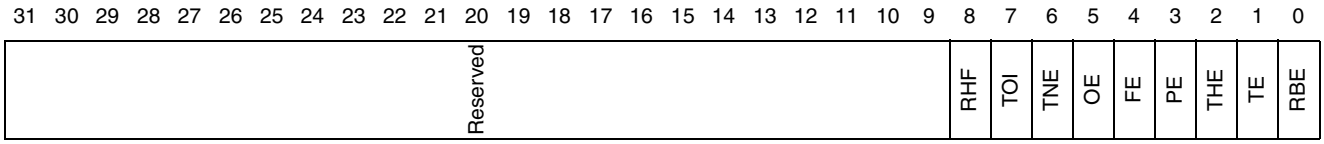
[6] **LOOPBACK:** Loopback mode enable bit
0: Standard transmit/receive mode 1: Loopback mode enabled

[5] **PARITYODD:** Parity selection
0: Even parity (parity bit set on odd number of 1's in data)
1: Odd parity (parity bit set on even number of 1's in data)

[4:3] **STOPBITS:** Number of stop bits selection
00: 0.5 stop bits 01: 1 stop bits
10: 1.5 stop bits 11: 2 stop bits

[2:0} **MODE:** ASC mode control: Mode2
000: Reserved 001: 8-bit data
010: Reserved 011: 7-bit data + parity
100: 9-bit data 101: 8-bit data + wake up bit
110: Reserved 111: 8-bit data + parity

ASC_n_INT_EN ASCn interrupt enable



Address: *ASCnBaseAddress* + 0x010

Type: R/W

Reset: 0

Description:

[31:9] **Reserved**

[8] **RHF:** Receiver FIFO is half full interrupt enable
0: Receiver FIFO is half full interrupt disable 1: Receiver FIFO is half full interrupt enable

[7] **TOI:** Time out when the receiver FIFO is empty interrupt enable
0: Time out when the input FIFO or buffer is empty interrupt disable
1: Time out when the input FIFO or buffer is empty interrupt enable

[6] **TNE:** Time out when not empty interrupt enable
0: Time out when input FIFO or buffer not empty interrupt disable
1: Time out when input FIFO or buffer not empty interrupt enable

[5] **OE:** Overrun error interrupt enable
0: Overrun error interrupt disable 1: Overrun error interrupt enable

[4] **FE:** Framing error interrupt enable
0: Framing error interrupt disable 1: Framing error interrupt enable

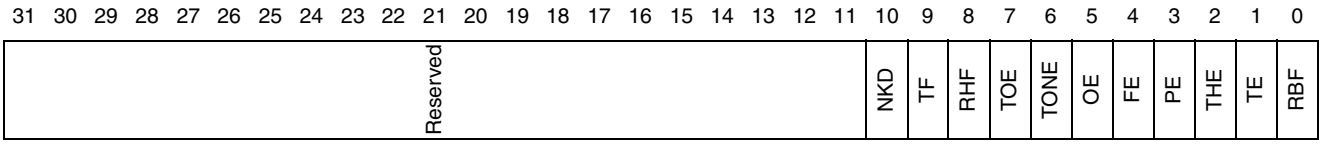
[3] **PE:** Parity error interrupt enable:
0: Parity error interrupt disable 1: Parity error interrupt enable

[2] **THE:** Transmitter buffer half empty interrupt enable
0: Transmitter buffer half empty interrupt disable 1: Transmitter buffer half empty interrupt enable

[1] **TE:** Transmitter empty interrupt enable
0: Transmitter empty interrupt disable 1: Transmitter empty interrupt enable

[0] **RBE:** Receiver buffer full interrupt enable
0: Receiver buffer full interrupt disable 1: Receiver buffer full interrupt enable

ASC_n_INT_STA ASCn interrupt status



Address: *ASCnBaseAddress* + 0x014

Type: RO

Reset: 3 (Rx buffer full and Tx buffer empty)

Description:

[31:11] **Reserved**

[10] **NKD**: Transmission failure acknowledgement by receiver
0: Data transmitted successfully
1: Data transmission unsuccessful (data NACKed by smartcard)

[9] **TF**: Transmitter FIFO or buffer is full
0: The FIFOs are enabled and the transmitter FIFO is empty or contains less than 16 characters or the FIFOs are disabled and the transmit buffer is empty
1: The FIFOs are enabled and the transmitter FIFO contains 16 characters or the FIFOs are disabled and the transmit buffer is full

[8] **RHF**: Receiver FIFO is half full
0: The receiver FIFO contains eight characters or less
1: The receiver FIFO contains more than eight characters

[7] **TOE**: Time out when the receiver FIFO or buffer is empty
0: No time out or the receiver FIFO or buffer is not empty
1: Time out when the receiver FIFO or buffer is empty

[6] **TONE**: Time out when the receiver FIFO or buffer is not empty
0: No time out or the receiver FIFO or buffer is empty
1: Time out when the receiver FIFO or buffer is not empty

[5] **OE**: Overrun error flag
0: No overrun error
1: Overrun error, that is, data received when the input buffer is full

[4] **FE**: Input frame error flag
0: No framing error 1: Framing error (stop bits not found)

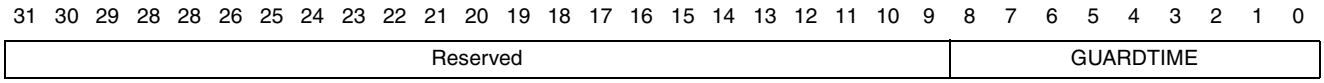
[3] **PE**: Input parity error flag:
0: No parity error 1: Parity error

[2] **THE**: Transmitter FIFO at least half empty flag or buffer empty
0: The FIFOs are enabled and the transmitter FIFO is more than half full (more than eight characters) or the FIFOs are disabled and the transmit buffer is not empty.
1: The FIFOs are enabled and the transmitter FIFO is at least half empty (eight or less characters) or the FIFOs are disabled and the transmit buffer is empty

[1] **TE**: Transmitter empty flag
0: Transmitter is not empty 1: Transmitter is empty

[0] **RBF**: Receiver FIFO not empty (FIFO operation) or buffer full (double buffered operation)
0: Receiver FIFO is empty or buffer is not full 1: Receiver FIFO is not empty or buffer is full

ASC_n_GUARDTIME ASCn guard time



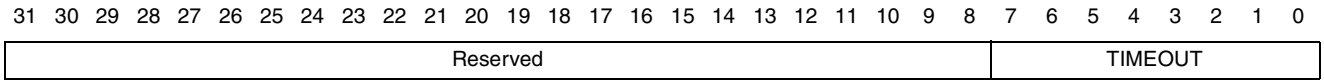
Address: *ASCnBaseAddress* + 0x018

Type: R/W

Reset: 0

Description: This register enables the delay of the assertion of the interrupt TXEMPTY by a programmable number of baud clock ticks. The value in the register is the number of baud clock ticks to delay assertion of TXEMPTY. This value must be in the range 0 to 511.

ASC_n_TIMEOUT ASCn time out



Address: *ASCnBaseAddress* + 0x01C

Type: R/W

Reset: 0

Description: The time out period in baudrate ticks. The ASC contains an 8-bit time out counter, which reloads from ASC_n_TIMEOUT when one or more of the following is true:

- ASC_n_RXBUFFER is read,
- the ASC is in the middle of receiving a character,
- ASC_n_TIMEOUT is written to.

If none of these conditions hold the counter decrements to 0 at every baudrate tick.

The TONE (time out when not empty) bit of the ASC_N_INT_STA register is 1 when the input FIFO is not empty and the time out counter is zero. The TIMEOUTIDLE bit of the ASC_N_INT_STA register is 1 when the input FIFO is empty and the time out counter is zero.

When the software has emptied the input FIFO, the time out counter resets and starts decrementing. If no more characters arrive, when the counter reaches zero the TIMEOUTIDLE bit of the ASC_N_INT_STA register is set.

ASC_n_TX_RST ASCn transmit FIFO reset

Address: *ASCnBaseAddress* + 0x020

Type: WO

Description: Reset the transmit FIFO. Registers ASC_n_TXRESET have no storage associated with them. A write of any value to these registers resets the corresponding transmitter FIFO.

ASC_n_RX_RST ASCn receive FIFO reset

Address: *ASCnBaseAddress* + 0x024

Type: WO

Description: Reset the receiver FIFO. The registers ASC_n_RXRESET have no actual storage associated with them. A write of any value to one of these registers resets the corresponding receiver FIFO.

ASC_n_RETRIES ASCn number of retries on transmission

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								NUM_RETRIES							

Address: *ASCnBaseAddress* + 0x028

Type: R/W

Reset: 1

Description: Defines the number of transmissions attempted on a piece of data before the UART discards the data. If a transmission still fails after NUM_RETRIES, the NKD bit is set in the ASC_N_INT_STA register where it can be read and acted on by software. This register does not have to be reinitialized after a NACK error.

63 Synchronous serial controller (SSC)

63.1 Overview

The synchronous serial controller (SSC) is a high-speed interface which can be used to communicate with a wide variety of serial memories, remote control receivers and other microcontrollers. There are a number of serial interface standards for these. Three SSCs are provided on the STx7100. The SSC supports all the features of the serial peripheral interface (SPI) bus and also includes additional functions for the full support of the I^2C bus. The general programmable features should also allow interface to other serial bus standards.

The SSC shares pins with the parallel input/output (PIO) ports. It supports full-duplex[1] and half-duplex synchronous communication when used in conjunction with the PIO configuration.

The SSC uses three signals:

- serial clock SCLK,
- serial data in/out MRST,
- serial data out/in MTSR[1].

To set the SSC PIOs to their alternate functions, follow this sequence:

1. Set SCL and MTSR as open drain bidirectional.
2. Set MRST as input[1].
3. Set SCL and MTSR to logic high.
4. Set all SSC registers to slave mode.
5. Only now, when the software is ready to accept data from the master, reprogram the PIO pins to their alternate output functions.

For I^2C operation, MRST and MTSR can either be externally wired together, or just the MTSR pin can be used[1]. These pins are connected to the SSC clock and data interface pins in a configuration which allows their direction to be changed when in master or slave mode (see Section 63.2.1: *Pin connection and control on page 798*). The serial clock signal is either generated by the SSC (in master mode) or received from an external master (in slave mode). The input and output data are synchronized to the serial clock.

The following features are programmable: baudrate, data width, shift direction (heading control), clock polarity and clock phase. These features allow communications with SPI compatible devices.

In the SPI standard, the device can be used as a bus master, a bus slave, or can arbitrate in a multi-master environment for control of the bus. Many of these features require software support.

The SSC also fully supports the I^2C bus standard and contains additional hardware (beyond the SPI standard) to achieve this. The extra I^2C features include:

- multi-master arbitration,
- acknowledge generation,
- start and stop condition generation and detection,
- clock stretching.

These allow software to fully implement all aspects of the standard, such as master and slave mode, multi-master mode, 10-bit addressing and fast mode.

1. On the STx7100, by default the two serial data in/out signals are multiplexed on to a single pin for I^2C mode (full-duplex mode is not supported).

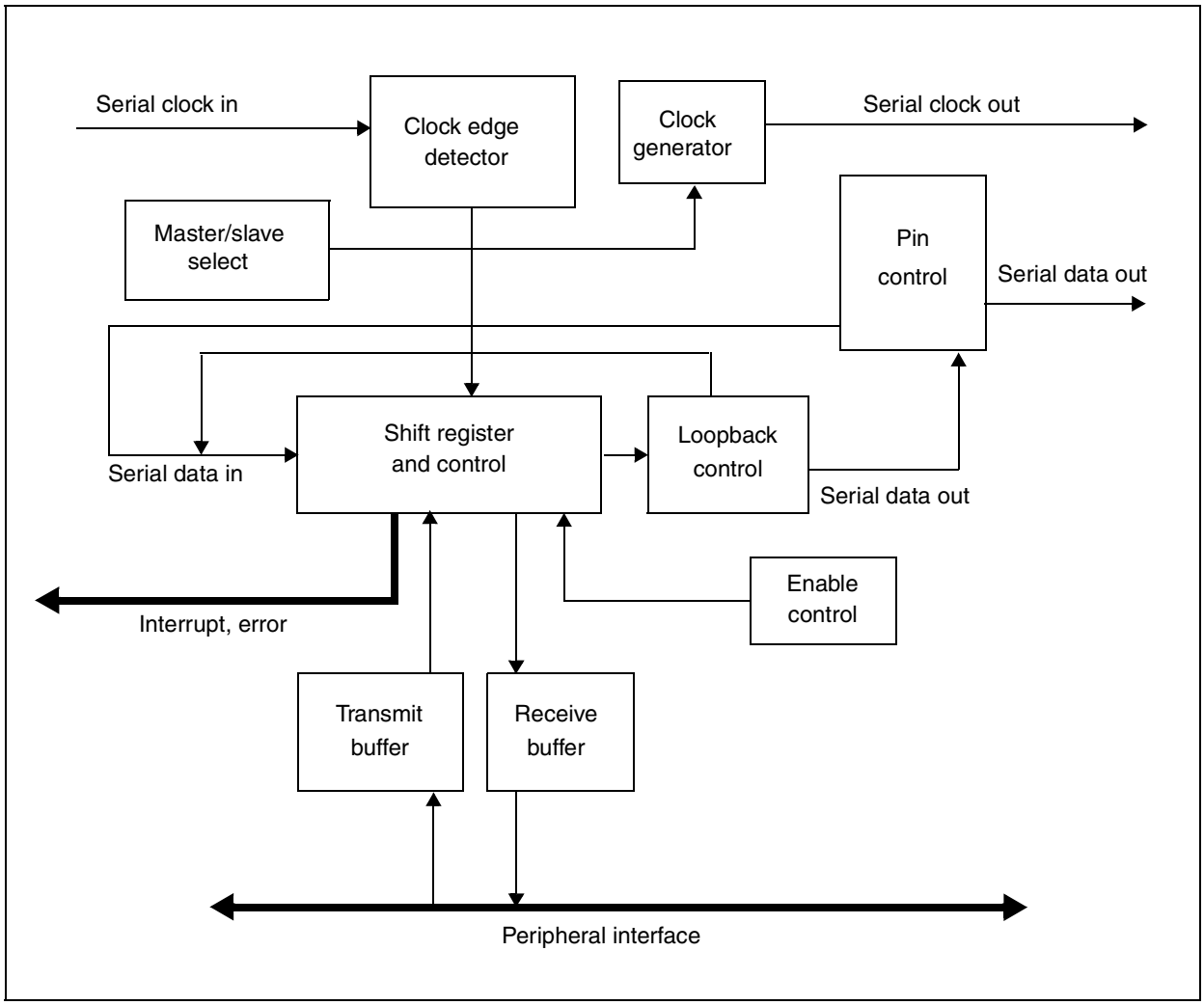
63.2 Basic operation

Control of the direction, either as input, output or bidirectional, of the SCLK, MTSR and MRST pins[1] is performed in software by configuring the PIO.

The serial clock output signal is programmable in master mode for baudrate, polarity and phase. This is described in Section 63.2.2: *Clock generation on page 799*.

The SSC works by taking the data frame (2 to 16 bits) from a transmission buffer and placing it into a shift register. It then shifts the data at the serial clock frequency out of the output pin and synchronously shifts in data coming from the input pin. The number of bits and the direction of shifting (MSB or LSB first) are programmable. This is described in Section 63.2.4: *Shift register on page 801*.

Figure 250: SSC architecture



After the data frame has been completely shifted out of the shift register, it transfers the received data frame into the receive buffer. The transmit and receive buffers are described in Section 63.2.7: *Transmit and receive buffers on page 803*. The SSC is therefore double buffered. This allows back-to-back transmission and reception of data frames up to the speed that interrupts can be serviced.

1. On the STx7100, by default the two serial data in/out signals can be multiplexed on to a single pin for I^2C mode (full-duplex mode is not supported).

The SSC can also be configured to loop the serial data output back to serial data input in order to test the device without any external connections. This is described in Section 63.2.8: *Loopback mode* *on page 803*.

The SSC can be turned on and off by setting the enable control. This is described in Section 63.2.9: *Enabling operation* *on page 803*. It can be also be set to operate as a bus master or as a bus slave device. This is described in Section 63.2.10: *Master/slave operation* *on page 803*.

The SSC generates interrupts in a variety of situations:

- when the transmission buffer is empty,
- when the receive buffer is full, and
- when an error occurs. A number of error conditions are detected. These are described in Section 63.2.11: *Error detection* *on page 804*.

There are additional hardware features which can be independently enabled in order to fully support the I^2C bus standard when used in conjunction with a suitable software driver. The additional I^2C hardware is described in Section 63.3: *I2C operation* *on page 806*.

63.2.1 Pin connection and control

To fully support the SPI standard, the interface presented at the pins is:

- a single clock pin, SCLK, which is both an input and an output;
- two data pins, MTSR, and MRST, which are either inputs or outputs depending on whether the SSC is in slave or master mode[1].

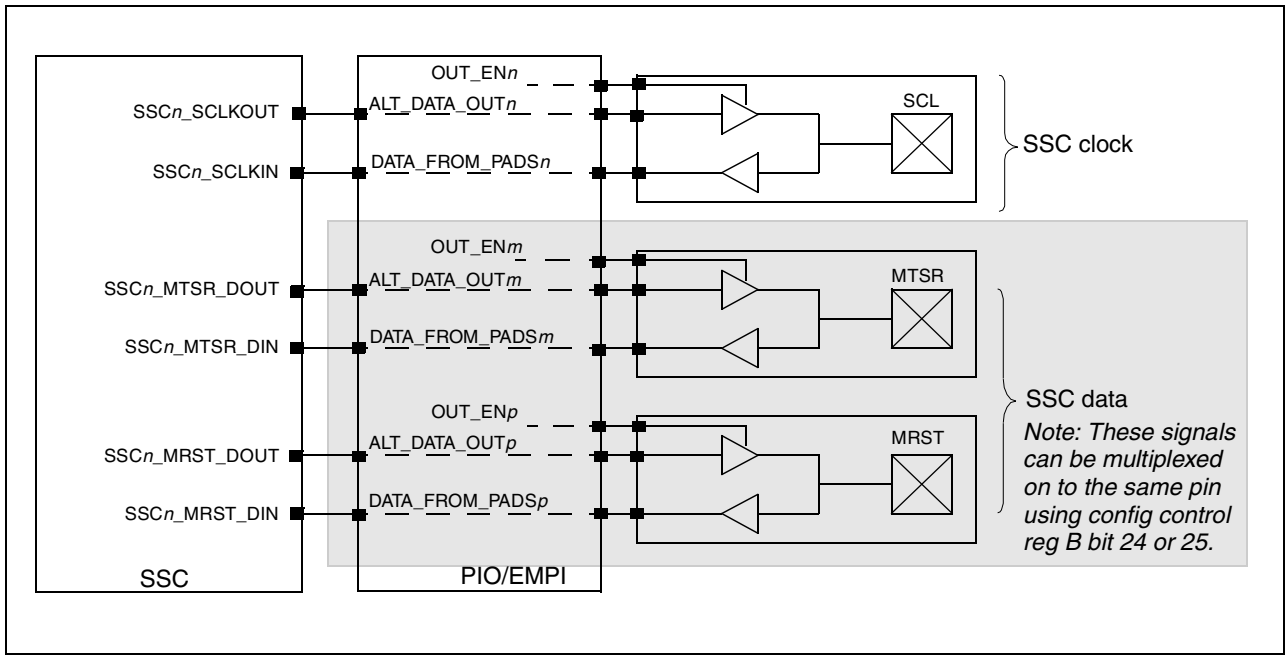
In I^2C mode only, the MTSR pin is used as an input and output. This means only the MTSR pad needs to be used on the I^2C data line. However, for backward compatibility, it is still possible to short MTSR and MRST data pins externally and achieve the same function (the MRST data output is permanently driven to a high logic value and its input is ignored[1].

These pads are provided by three bits of a standard PIO block. Their directions (input, output or bidirectional) can therefore be configured in software using the appropriate PIO settings. Consequently the SSC does not need to provide automatic control of data pad directions and does not need to provide a bidirectional clock port.

1. On the STx7100, by default the two serial data in/out signals can be multiplexed on to a single pin (full-duplex mode is not supported) for I^2C mode.

The connections between the SSC ports and the relevant PIO pins are illustrated in Figure 251. Pins are shared with PIO, and with EMPI in the GX1 (the selection between PIO and EMPI pins is made in the glue logic).

Figure 251: SSC to PIO and EMPI connections



The pad control block inside the SSC determines which of the serial data input ports is used to read data from (depending on the master or slave mode). It also determines which of the serial data output ports to write data to (depending on the master or slave mode).

The deselected serial data output port is driven to ground (except in I^2C mode when it is driven high). Therefore the user must ensure that the relevant PIO pad output enable is turned off depending on the master/slave status of the SSC.

It is up to the user to ensure that the PIO pads are configured correctly for direction and output driver type (for example, push/pull or open drain).

Throughout the rest of this document, the data in and out ports is referred to as SERIAL_DATA_OUT and SERIAL_DATA_IN, where this is assumed to be the correct pair of pins dependent on the master or slave mode of the SSC.

63.2.2 Clock generation

If the SSC is configured to be the bus master, then it generates a serial clock signal on the serial clock output port.

The clock signal can be controlled for polarity and phase and its period (baudrate) can be set to a variety of frequencies.

For I^2C operation there are a number of additional clocking features. These are described in Section 63.3: *I2C operation on page 806*.

Clock control

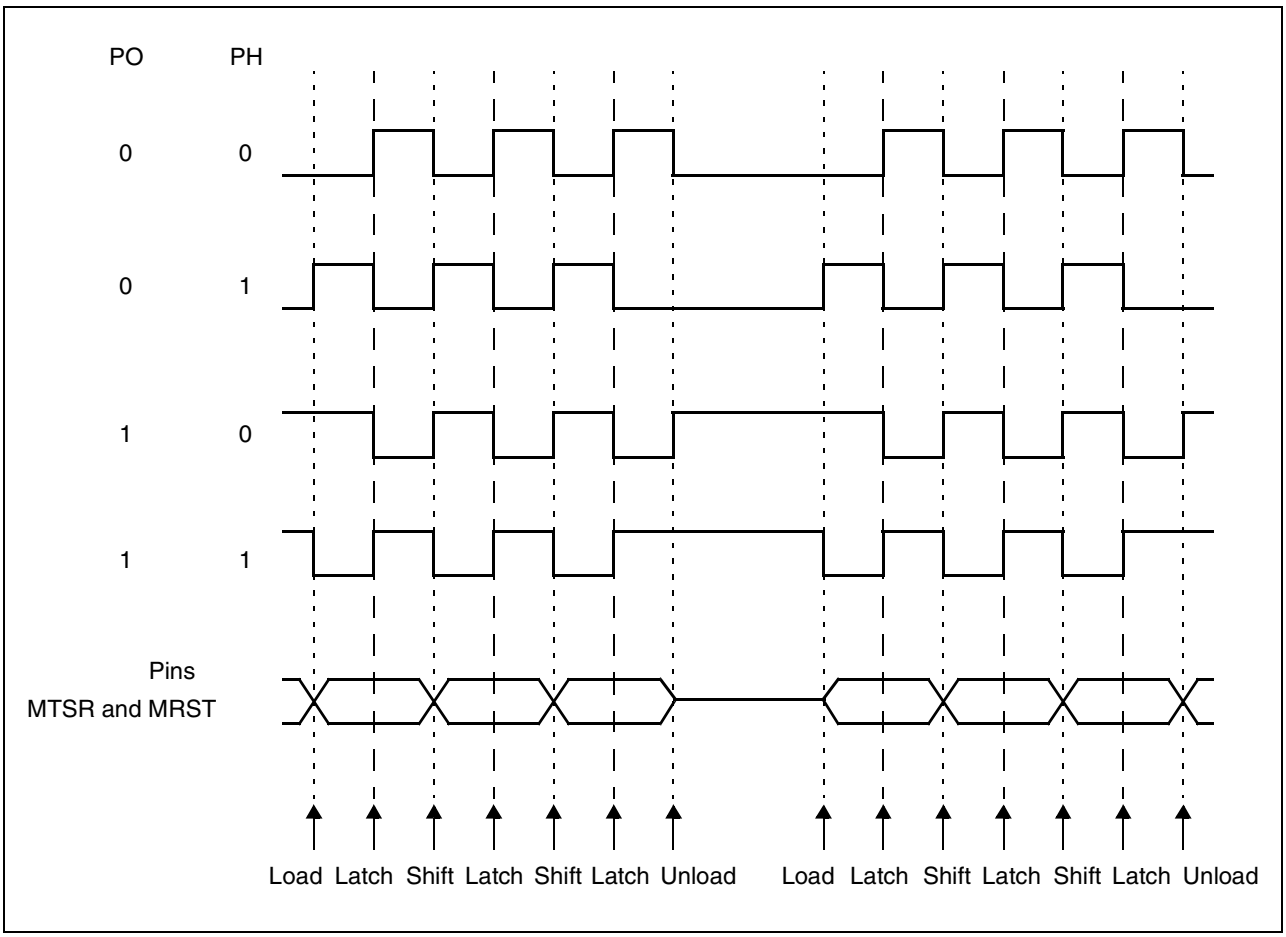
In master mode, the serial clock SCLK, is generated by the SSC according to the setting of the phase bit PH and polarity bit PO in the control register SSC*n*CON.

The polarity bit PO defines the logic level the clock idles at, that is, when the SSC is in master mode but is between transactions. A polarity bit of 1 indicates an idle level of logic 1; 0 indicates idle of logic 0.

The phase bit PH indicates whether a pulse is generated in the first or second half of the cycle. This is a pulse relative to the idle state of the clock line; so if the polarity is 0 then the pulse is positive going; if the polarity is 1 then the pulse is negative going. A phase setting of 0 causes the pulse to be in the second half of the cycle while a setting of 1 causes the pulse to occur in the first half of the cycle.

The different combinations of polarity and phase are shown in Figure 252.

Figure 252: Polarity and phase combinations



The SSC always latches incoming data in the middle of the clock period at the point shown in the diagram. With the different combinations of polarity and phase it is possible to generate or not generate a clock pulse before the first data bit is latched.

Shifting out of data occurs at the end of the clock period. At the start of the first clock period the shift register is loaded. At the end of the last clock period, the shift register is unloaded into the receive buffer.

63.2.3 Baudrate generation

The SSC can generate a range of different baudrate clocks in master mode. These are set up by programming the baudrate generator register SSCnBRG.

In write mode this register is set up to program the baudrate as defined by the following formulae:

$$Baudrate = \frac{fcomms}{2 \times SSCGBR} \qquad SSCBRG = \frac{fcomms}{2 \times Baudrate}$$

where *SSCBRG* represents the content of the baudrate generator register, as an unsigned 16-bit integer, and f_{comms} represents the comms clock frequency.

At a comms clock frequency of 60 MHz the baudrates generated are shown in Table 206.

Table 206: Baudrates and bit times for different SSCBRG reload values

Baudrate	Bit time	Reload value
Reserved. Use a reload value > 0	-	0x0000
5 MBaud	200 ns	0x0006
3.3 MBaud	300 ns	0x0009
2.5 MBaud	400 ns	0x000C
2.0 MBaud	500 ns	0x000F
1.0 MBaud	1 µs	0x001E
100 KBaud	10 µs	0x012C
10 KBaud	100 µs	0x0BB8
1.0 KBaud	1 ms	0x07D0

The value in SSCnBRG is used to load a counter at the start of each clock cycle. The counter counts down until it reaches 1 and then flips the clock to the opposite logic value. Consequently, the clock produced is twice the SSCnBRG number of comms clock cycles.

In read mode the SSCnBRG register returns the current count value. This can be used to determine how far into each half cycle the counter is.

63.2.4 Shift register

The shift register is loaded with the data in the transmit buffer at the start of a data frame. It then shifts data out of the serial output port and data in from the serial input port.

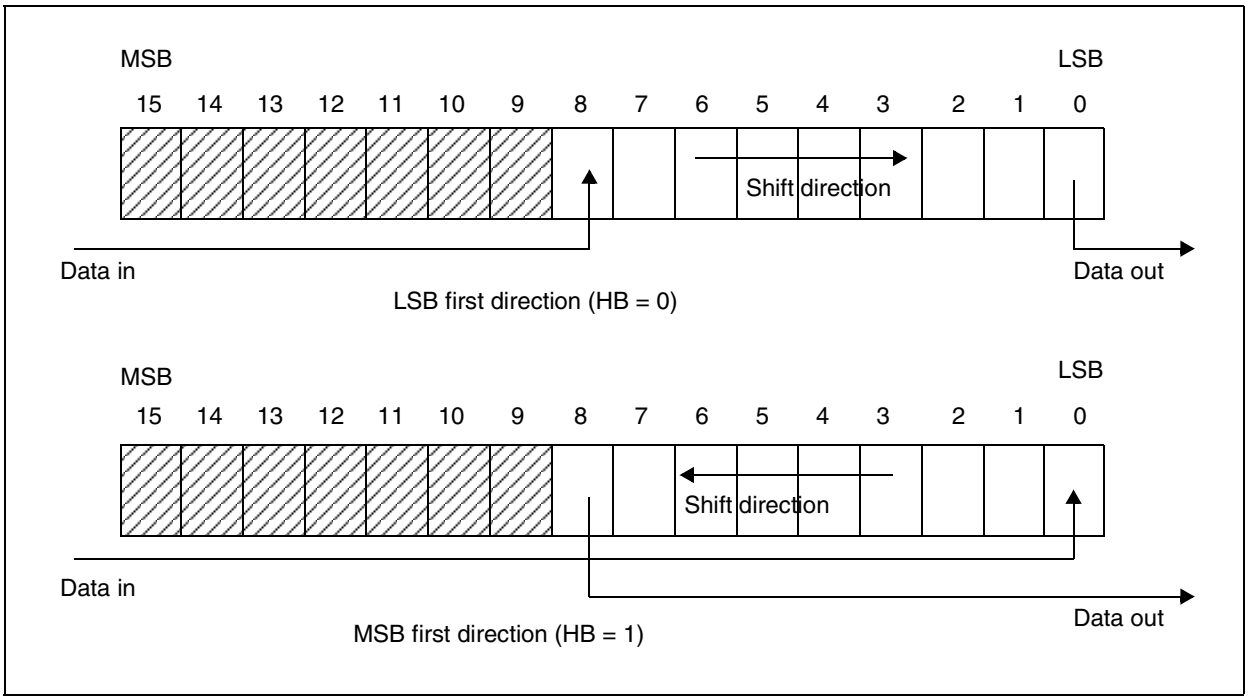
The shift register can shift out LSB first or MSB first. This is programmed by the heading control bit HB in the control register SSCnCON. A logic 1 indicates that the MSB is shifted out first and a logic 0 that the LSB shifts first.

The width of a data frame is also programmable from 2 to 16 bits. This is set by the BM bit field of the control register SSCnCON. A value of 0000 is not allowed. Subsequent values set the bit width to the value plus one; for example 0001 sets the frame width to 2 bits and 1111 sets it to 16 bits.

Note: For I²C, bit SSC*n*_CON.BM *must be programmed for a 9-bit data width.*

When shifting LSB first, data comes into the shift register at the MSB of the programmed frame width and is taken out of the LSB of the register. When shifting in MSB first, data is placed into the LSB of the register and taken out of the MSB of the programmed data width. This is shown for a 9-bit data frame in Figure 253.

Figure 253: 9-bit data frame shifting



The shift register shifts at the end of each clock cycle. The clock pulse for shifting is presented to it from the clock generator (see Section 63.2.2: *Clock generation on page 799*). This is regardless of the polarity or phase of the clock.

When a complete data frame has been shifted, the contents of the shift register (that is, all bits shifted into the register) is loaded into the receive buffer.

There are some additional controls required on the shifting operation to allow full support of the I^2C bus standard. These are described in Section 63.3: *I2C operation on page 806*.

63.2.5 Receive data sampling

The data received by the SSC is sampled after the latching edge of the input clock, the latching edge being determined by the programming of the polarity and phase bits.

The data value which is finally latched is determined by taking three data samples at the third, fourth and fifth comms clock periods after the latching data edge. The data value is determined from the predominant data value in the three samples. This gives an element of spike suppression.

63.2.6 Antiglitch filter

The antiglitch filter suppresses any pulses which have a value in microseconds of less than a programmed width. Such signals may be either high or low. The filter has two registers, SSC_NOISE_SUPP_WID and SSC_PRESCALER.

SSC_NOISE_SUPP_WID holds the value of maximum glitch width. To suppress glitches of *n* microseconds and below, the value *n* + 1 is written into the register. Writing 0x00 bypasses the antiglitch filter.

The comms clock is divided by a prescaler factor equivalent to 10 MHz, before being fed to the antiglitch filter. For example, if the comms clock is 50 MHz the prescaler division factor is 5.

63.2.7 Transmit and receive buffers

The transmit and receive buffers are used to allow the SSC to do back-to-back transfers; that is, continuous clock and data transmission.

The transmit buffer SSCnTBUF is written with the data to be sent out of the SSC. This is loaded into the shift register for transmission. Once this has been performed, the SSCnTBUF is available to be loaded again with a new data frame. This is indicated by the assertion of the transmit interrupt request status bit SSCTIR, which indicates that the transmit buffer is empty. This causes an interrupt if the transmit buffer empty interrupt is enabled, by setting the TIEN bit in the interrupt enable register SSCnIEN.

A transmission is started in master mode by a write to the transmit buffer. This starts the clock generation circuit and loads the shift register with the new data.

Continuous transfers of data are therefore possible by reloading the transmit buffer whenever the interrupt is received. The software interrupt routine has the length of time for a complete data frame in order to refill the buffer before it is next emptied. If the transmit buffer is not reloaded in time when in slave mode, a transmit error condition TE (see Section 63.2.11: *Error detection*) is generated.

The number of bits to be loaded into the transmit buffer is determined by the frame data width selected in the control register bit BM. The unused bits are ignored.

The receive buffer SSCnRBUF is loaded from the shift register when a complete data frame has been shifted in. This is indicated by the assertion of the receive interrupt request status bit RIR, which indicates that the receive buffer is full. This causes an interrupt if the receive buffer full interrupt is enabled, by setting the RIEN in the interrupt enable register.

The CPU should then read out the contents of this register before the next data frame has been received otherwise the buffer is reloaded from the shift register over the top of the previous data. This is indicated as a receive error condition RE. See Section 63.2.11: *Error detection*.

The number of bits which is loaded into the receive buffer is determined by the frame data width selected in the control register BM. The unused bits are not valid and should be ignored.

63.2.8 Loopback mode

A loopback mode is provided which connects the SERIAL_DATA_OUT to SERIAL_DATA_IN. This allows software testing to be performed without the need for an external bus device. This mode is enabled by setting the LPB bit in the control register, SSCnCON. A setting of logic 1 enables loopback, logic 0 puts the SSC into normal operation.

63.2.9 Enabling operation

The transmission and reception of data by the SSC block can be enabled or disabled by setting the EN bit in the control register, SSCnCON. A setting of logic 1 turns on the SSC block for transmission and reception. Logic 0 prevents the block from reading or writing data to the serial data input and output ports.

63.2.10 Master/slave operation

The control of a number of the features of the SSC depends on whether the block is in master or slave mode. For example, in master mode the SSC generates the serial clock signal according to the setting of baudrate, polarity and phase. In slave mode, no clock is generated and instead the assumption is made that an external device is generating the serial clock.

Master or slave mode is set by the MS bit in the control register, SSCnCON. A setting of logic 0 means the SSC is in slave mode, a setting of logic 1 puts the device into master mode.

63.2.11 Error detection

A number of different error conditions can be detected by the SSC. These are related to the mode of operation (master or slave, or both).

On detection of any of these error conditions a status flag is set in the status register, SSCnSTAT. Also, if the relevant enable bit is set in the interrupt enables register SSCnIEN, then an error interrupt is generated from the SSC.

The different error conditions are described as follows.

Transmit error

A transmit error can be generated both in master and slave mode. It indicates that a transfer has been initiated by a remote master device before a new transmit data buffer value has been written in to the SSC.

In other words, the error occurs when old transmit data is going to be transmitted. This could cause data corruption in the half-duplex open drain configuration.

The error condition is indicated by the setting of the TE bit in the status register. An interrupt is generated if the TEEN bit is set in the interrupt enables register.

The transmit error status bit (and the interrupt, if enabled) is cleared by the next write to the transmit buffer.

Receive error

A receive error can be generated in both master and slave modes. It indicates that a new data frame has been completely received into the shift register and has been loaded into the receive buffer before the existing receive buffer contents have been read out. Consequently, the receive buffer has been overwritten with new data and the old data is lost.

The error condition is indicated by the setting of the RE bit in the status register SSCnSTAT. An interrupt is generated if the REEN bit is set in the interrupt enables register.

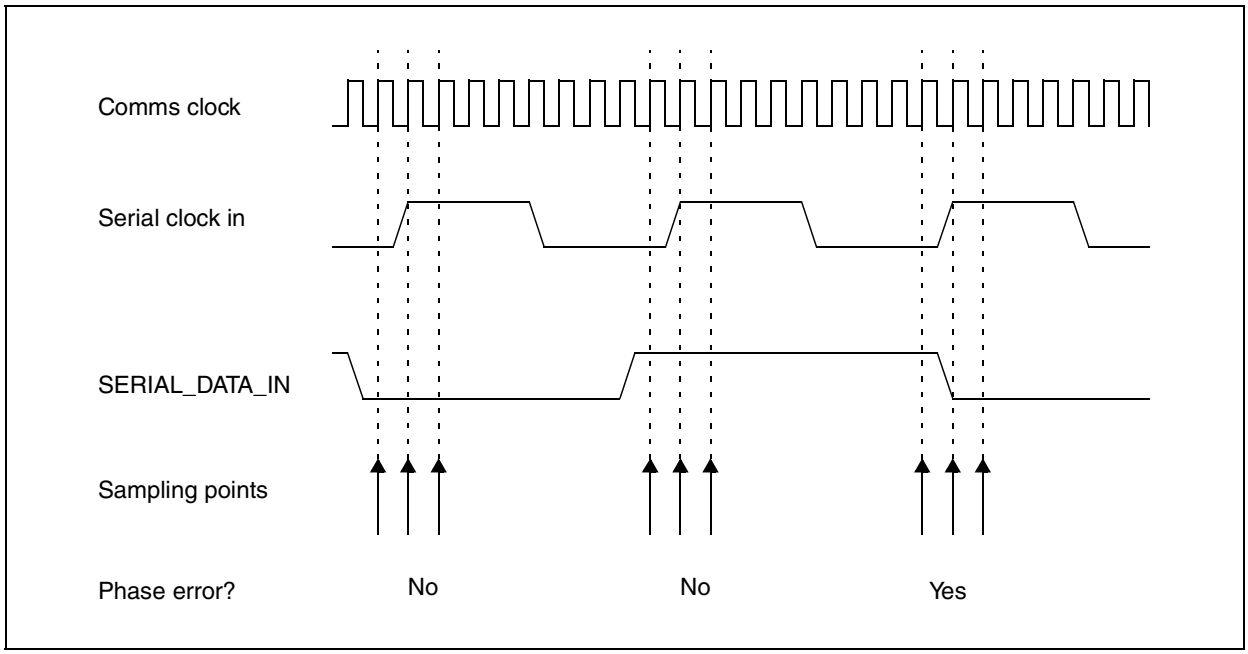
The receive error status bit (and the interrupt, if enabled) is cleared by the next read from the receive buffer.

Phase error

A phase error can be generated in master and slave modes. This indicates that the data received at the incoming data pin (MRST in master mode or MTSR in slave mode) has changed during the time from one sample before the latching clock edge and two samples after the edge.

The data at the incoming data pin is supposed to be stable around the time of the latching clock edge, hence the error condition. Each sample occurs at the comms clock frequency. The sampling scheme is shown in Figure 254.

Figure 254: Sampling scheme



The error condition is indicated by the setting of the PE bit in the status register. An interrupt is generated if the PEEN bit is set in the interrupt enables register. The phase error status bit (and the interrupt, if enabled) is cleared by the next read from the receive buffer.

63.2.12 Interrupt mechanism

The SSC can generate a variety of different interrupts. They can all be enabled or disabled independently of each other. All the enabled interrupt conditions are ORed together to generate a global interrupt signal.

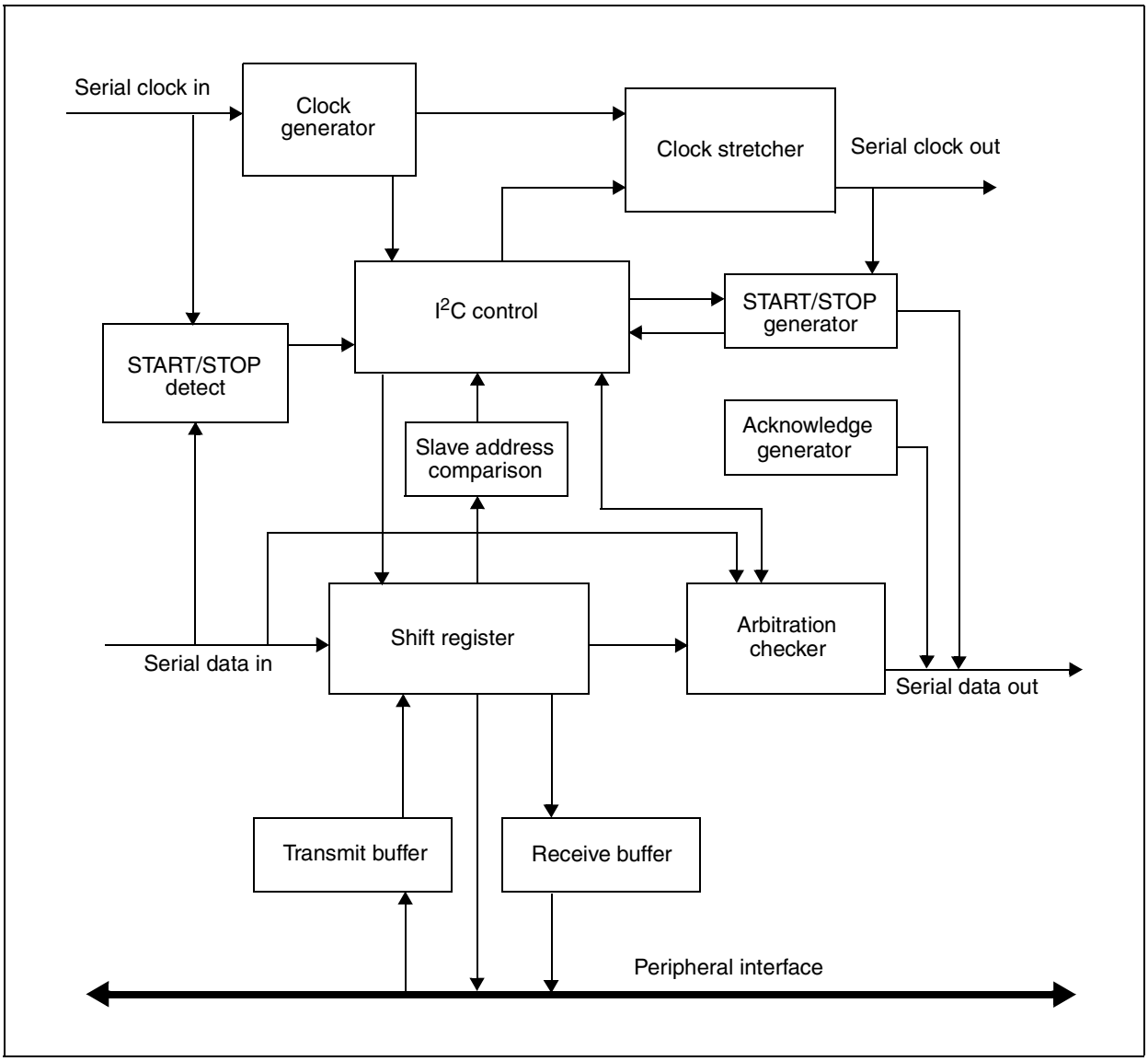
To determine which interrupt condition has occurred, a status register SSCnSTAT is provided which includes a bit for each condition. This is independent of the interrupt enables register SSCnIEN, and determines whether the condition asserts one or more of the interrupt signals.

63.3 I²C operation

This section describes the additional hardware features which are implemented in order to allow full support for the I²C bus standard.

The architecture of the I²C including all the I²C hardware additions is shown in Figure 255.

Figure 255: I²C architecture



63.3.1 I²C control

There are a number of features of the I²C-bus protocol which require special control.

- To allow slow slave devices to be accessed and to allow multiple master devices to generate a consistent clock signal, a clock synchronization mechanism is specified.
- START and STOP conditions must be recognized when in slave mode or multi-master mode. A START condition initiates the address comparison phase. A STOP condition indicates that a master has completed transmission and that the bus is now free.
- In slave mode (and in multi-master configurations), it is necessary to determine if the first byte received after a START condition is the address of the SSC. If it is, then an acknowledge must be generated in the ninth bit position.

Subsequently, an interrupt must be generated to inform the software that the SSC has been addressed as a slave device and therefore that it needs to either send data to the addressing master or to receive data from it.

In addition to normal 7-bit addressing, there is an extended 10-bit addressing mode where the address is spread over two bytes. In this mode, the SSC must compare two consecutive bytes with the incoming data after a START condition. It must also generate acknowledge bits for the first and second bytes automatically if the address matches.

The 10-bit addressing mode is further complicated by the fact that if the slave has been previously addressed for writing with the full two-byte address, the master can issue a repeated START condition and then transmit just the first address byte for a read. The slave therefore must remember that it has already been addressed and must respond.

- For the software interrupt handler to have time to service interrupts, the SSC can hold the clock line low until the software releases it. This is called clock stretching.
- In master mode the SSC must begin a transmission by generating a START condition and must end transmission by generating a STOP condition. In multi-master configurations a START condition should not be generated if the bus is already busy; that is, a START condition has already been received.
- When the SSC is receiving data from another device, it must generate acknowledge bits in the ninth bit position. However, when receiving data as a master, the last byte received must not be acknowledged. This only applies to data bytes; when operating as a slave device the SSC should always acknowledge a matching address byte; that is, the first byte after a START condition.
- In multi-master configurations, arbitration must take place because it is not possible to determine if another master is also trying to transmit to the bus; that is, the START conditions were generated within the allowed time frame.

Arbitration involves checking that the data being transmitted is the same as the data received. If this is not the case, then we have lost arbitration. The SSC must then continue to transmit a high logic level for the rest of the byte to avoid corrupting the bus.

It is also possible that, having lost arbitration, it is addressed as a slave device. So the SSC must then go into slave mode and compare the address in the normal fashion (and generate an acknowledge if it was addressed).

After the byte plus acknowledge the SSC must indicate to the software that we have lost arbitration by setting a flag.

All of these features are provided in the SSC design. They are controlled by the I^2C control block which interacts with various other modules to perform the protocols.

In order to program for I^2C mode, a separate control register SSCnI2C is provided. To perform any of the I^2C hardware features, the I^2C control bit I2CM, must be set in this register. When the I^2C control bit is set, the clock synchronization mechanism is always enabled (see Section 63.3.2: *Clock synchronization on page 808*). When the I^2C control bit is set, the START and STOP condition detection is performed. Fast mode is supported by bit 12 (I2CFSMODE) of the SSCnI2C register. In addition bits PH and PO of register SSC_nCON must be set to 1.

To program the slave address of the SSC the slave address register, SSCnSLAD must be written to with the address value. In the case of 7-bit addresses, only 7 bits should be written. For 10-bit addressing, the full 10 bits are written. The SSC then uses this register to compare the slave address transmitted after a START condition (see Section 63.3.4: *Slave address comparison on page 810*). To perform 10-bit address comparison and address acknowledge generation, the 10-bit addressing mode bit AD10 must be set in the SSCnI2C register (see Section 63.3.4: *Slave address comparison*).

The clock stretching mechanism is enabled for various interrupt conditions when the I^2C control enable bit I2CM in register SSCnI2C is set (see Section 63.3.5: *Clock stretching on page 811*).

To generate a START condition, the I²C START condition generate bit STRTG in register SSCnI2C, must be set (see Section 63.3.6: *START/STOP condition generation on page 811*). To generate a STOP condition, the I²C STOP condition generate bit STOPG, must be set (see Section 63.3.6: *START/STOP condition generation*).

To generate acknowledge bits (that is, a low data bit), after each 8 bit data byte when receiving data, the acknowledge generation bit ACKG in register SSCnI2C, must be set. When receiving data as a master, this bit must be reset to 0 before the final data byte is received, thereby signalling to the slave to stop transmitting (see Section 63.3.7: *Acknowledge bit generation on page 812*).

To indicate to the software that various situations have arisen on the I²C bus, a number of status bits are provided in the status register SSCnSTAT. In addition, some of these bits can generate interrupts if corresponding bits are set in the interrupt enable register SSCnIEN.

To indicate that the SSC has been accessed as a slave device, the addressed as slave bit AAS in register SSCnSTAT, is set. This also causes an interrupt if the AASEN bit is set in register SSCnIEN.

The interrupt occurs after the SSC has generated the address acknowledge bit. In 10-bit addressing mode, where two bytes of address are sent, the interrupt occurs after the second byte acknowledge bit; it occurs after the first byte acknowledge where only one byte is required.

Until the status bit is reset, the SSC holds the clock line low (see Section 63.3.5: *Clock stretching on page 811*). This forces the master device to wait until the software has processed the interrupt.

The status bit and the interrupt are reset by reading from the receive buffer SSCnRBUF, when the slave is being sent data, and by writing to the transmit buffer SSCnTBUF, when the SSC needs to send data.

To indicate that a STOP condition has been received, when in slave mode, the STOP condition detected bit STOP is set. This also causes an interrupt if the STOPEN bit is set in the interrupt enable register. The STOP interrupt and status bit is reset by a read of the status register SSCnSTAT.

To indicate that the SSC has lost the arbitration process, when in a multi-master configuration, the arbitration lost bit ARBL in register SSCnSTAT, is set. This also results in an interrupt if the ARBLEN bit is set in the interrupt enable register. The interrupt occurs immediately after the arbitration is lost.

Until the status bit is reset, the SSC holds the clock line low at the end of the current data frame, (see Section 63.3.5: *Clock stretching*). This forces the winning master device to wait until the software has processed the interrupt.

The interrupt and status bit is reset by a read of the status register SSCnSTAT.

To indicate that the I²C bus is busy (that is, between a START and a STOP condition), the I²C bus busy bit BUSY in register SSCnSTAT is set. This does not generate an interrupt.

63.3.2 Clock synchronization

The I²C standard defines how the serial clock signal can be stretched by slow slave devices and how a single synchronized clock is generated in a multi-master environment. The clock synchronization of all the devices is performed as follows.

All master devices start generating their low clock pulse when the external clock line goes low (this may or may not correspond with their own generated high to low transition).

They count out their low clock period and when finished attempt to pull the clock line high. However, if another master device is attempting to use a slower clock frequency, then it is holding the clock line low, or if a slave device wants to, it can extend the clock period by deliberately holding the clock low.

As the output drive is open-drain, the slower clock wins and the external clock line remains low until this device has finished counting its slow clock pulse, or until the slave device is ready to proceed. In the mean time, the quicker master device has detected a contradiction and goes into a wait state until the clock signal goes high again.

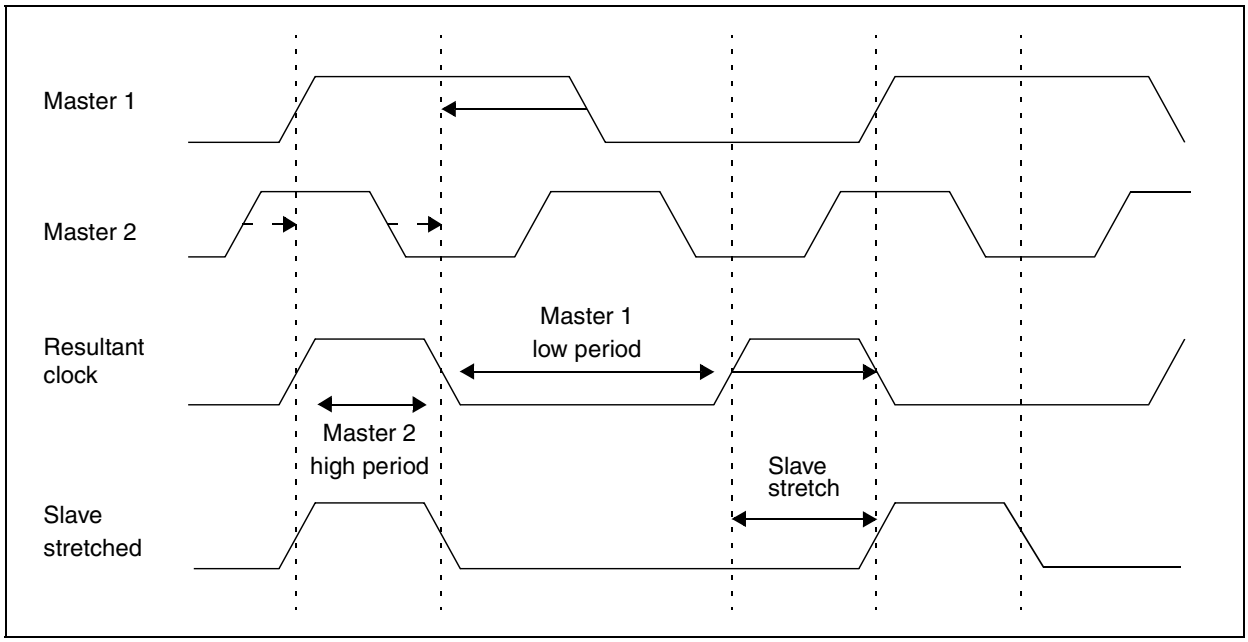
Once the external clock signal goes high, all the master devices begin counting off their high clock pulse. In this case the first master to finish counting attempts to pull the external clock line low and wins (because of the open drain line). The other master devices detect this and abort their high pulse count and switch to counting out their low clock pulse.

Consequently, the quicker master device determines the length of the high clock pulse and the slowest master or slave device determines the length of the low clock pulse.

This results in a single synchronized clock signal which all master and slave devices then use to clock their shift registers.

The synchronization and stretching mechanism is shown in Figure 256.

Figure 256: Synchronization and stretching



The SSC implements this clock synchronization mechanism when the I^2C control bit I2CM, is enabled.

63.3.3 START/STOP condition detection

START/STOP conditions are only generated by a master device. A slave device must detect the START condition and expect the next byte (or two bytes in 10-bit addressing) to be a slave address. A STOP condition is used to signal when the bus is free.

A START condition occurs when the transmit/receive data line changes from high to low during the high period of the clock line. It indicates that a master device wants control of the bus. In a single master configuration, it automatically gets control. In a multi-master configuration, it begins to transmit as part of the arbitration procedure, and may or may not get control (see Section 63.3.8: *Arbitration checking on page 812*).

A STOP condition occurs when the transmit/receive data line changes from low to high during the high period of the clock line. It indicates that a master device has relinquished control of the bus (the bus is made free a specified time after the stop condition).

An additional piece of hardware is provided on the SSC to detect START and STOP conditions. This is necessary in slave mode as detection cannot be performed in time merely by programming the PIO pads. This is because there is not sufficient time for a software interrupt

between the end of the START condition and the beginning of the data transmitted by a remote master.

START and STOP conditions are detected by sampling the data line continuously when the clock line is high. Minimum set up and hold times are measured by the counters.

The START condition is detected when data goes low (and the clock is high) and remains low for the minimum time specified by the I^2C standard.

The STOP condition is detected when data goes high (and the clock is high) and remains high for the minimum time specified by the I^2C standard.

START and STOP condition detection is enabled when the I^2C control bit I2CM is set in the I^2C control register.

When a START condition is triggered, the SSC informs the I^2C control block which then initiates the address comparison phase.

When a STOP condition is triggered, the SSC sets the STOP bit in the status register. It also generates an interrupt if the STOPDEN bit is set in the interrupt enable register.

The interrupt and the status bit are cleared when the status register is read.

63.3.4 Slave address comparison

After a START condition has been detected, the SSC goes into the address comparison phase.

It receives the first eight bits of the next byte transmitted and compares the first seven bits against the address stored in the slave address register SSCnSLAD. If they match, the address comparison block indicates this to the I^2C control block.

This generates an acknowledge bit in the next bit position and set the addressed as slave bit AAS in the status register. An interrupt is then generated after the acknowledge bit if the addressed as slave enable bit AASEN is set in the interrupt enables register.

The eighth bit of the first byte indicates whether the SSC is written to (low) or read from (high). This is used by the control block to determine if it needs to acknowledge the following data bytes (that is, when receiving data).

When 10-bit addressing mode is selected by setting the 10-bit addressing bit AD10 in register SSCnI2C, the first seven bits of the first data byte is compared against 11110nn, where *nn* is the two most significant bits of the 10-bit address stored in the slave address register.

The read/write bit then determines what to do next.

If the read/write bit is low, indicating a write, an acknowledge must be generated for the byte. The addressed as slave status bit and interrupt however are not yet asserted so, instead, the address comparator waits for the next data byte and compares this against the eight least significant bits of the slave address register.

If this matches, then the SSC is being addressed, so the second byte is acknowledged and the addressed as slave bit is set. An interrupt also occurs after the acknowledge bit if the addressed as slave interrupt enable is set.

On the other hand if the first byte sent has the read/write bit high, then the SSC only acknowledges it if it has previously been addressed and a STOP condition has not yet occurred (that is, the master has generated a repeated START condition). In this case the addressed as slave bit is set after the first byte plus acknowledge and an interrupt is generated if the interrupt enable is set. The second byte in this case is sent by the SSC as this is a read operation.

In all cases if the address does not match, then the SSC ignores further data until a STOP condition is detected.

63.3.5 Clock stretching

The I^2C standard allows slave devices to hold the clock line low if they need more time to process the data being received (see Section 63.3.2: *Clock synchronization on page 808*). The SSC takes advantage of this by inserting extended clock low periods. This is done to allow a software device driver to process the interrupt conditions when in slave mode.

The clock stretching mechanism is used in the situations listed below.

- When the SSC has been addressed as a slave device and the interrupt has been enabled. The clock stretch occurs immediately after the first byte with acknowledge, after a START condition has occurred (or in the case of 1-bit addressing this might occur after the second byte plus acknowledge). This gives the software interrupt routine time to initialize for transmission or reception of data. The clock stretch is cleared by writing 0x1FF to the transmit buffer register.
- When the SSC is in slave mode and is transmitting or receiving. The clock stretch occurs immediately after each data byte plus acknowledge. When transmitting, this allows the software interrupt routine to check that the master has acknowledged before writing the next data byte into the transmit buffer. If no acknowledge is received, then the software must stop transmitting bytes. When receiving, it allows the software to read the next data byte before the master starts to send the next one. The clock stretch is cleared by a write to the transmit buffer when transmitting and by a read from the receive buffer when receiving.
- When the SSC loses arbitration. The clock stretch occurs immediately after the current data byte and acknowledge have been performed only if the master which has lost arbitration has been addressed. This gives the software time to abort its current transmission and prepare to retry after the next STOP condition. The clock stretch is not performed if the master which has lost arbitration has not been addressed.

If a clock stretching event occurs but no relevant interrupt is enabled then the clock is stretched indefinitely. Hence it is important that the correct interrupts are always enabled.

63.3.6 START/STOP condition generation

As a master device the SSC must generate a START condition before transmission of the first byte can start. It may also generate repeated START conditions. It must complete its access to the bus with a STOP condition.

Between STOP and START conditions, the bus is free and the clock and data lines must be held high. The I^2C control block determines this and instructs the START/STOP generator to hold the lines high between transactions.

The START/STOP generator is controlled by the START condition generate bit STRTG and the STOP condition generate bit STOPG in register SSCnI2C.

The generator pulls the SERIAL_DATA_OUT line low during the high period of the clock to produce a START condition. In the case of a STOP condition it pulls the data line high.

However, a START condition is only generated if the bus is currently free (that is, the BUSY bit in the status register is low). This is to prevent the SSC from generating a START condition when another master has just generated one.

If a START condition cannot be generated because the bus is busy, then the generator forces the arbitration checker to generate an arbitration lost interrupt and prevent data from being transmitted for the next byte. The software interrupt handler is therefore informed of the aborted transmission when servicing the interrupt. Bit 11 (REPSTRT) of register SSCnSTAT shows that a repeated start condition has occurred.

To properly generate the timing waveforms of the START and STOP conditions, the SSC contains a timing counter. This ensures the minimum setup and hold times are met with some additional margin.

63.3.7 Acknowledge bit generation

For I^2C operation, it is required to both detect acknowledge bits when transmitting data, and to generate them when receiving data.

An acknowledge bit must be transmitted by the receiver at the end of every 8-bit data frame. The transmitter must verify that an acknowledge bit has been received before continuing.

An acknowledge bit is not generated by a master receiver for the last byte it wishes to receive. This “not acknowledge” is used by the slave device to determine when to stop transmission.

The acknowledge bit is generated by the receiver after the eight data bits have been transferred to it. In the ninth clock pulse, the transmitter holds the data line high and the receiver must pull the line low to acknowledge receipt. If the receiver is unable to acknowledge receipt, then the master generates a stop condition to abort the transfer.

Acknowledge bits are generated by the SSC when the acknowledge generation bit, ACKG, is set in the I^2C control register. They are only generated when receiving data.

When in master mode and receiving data the ACKG bit should be set to 0 before the last byte to be received. The SSC automatically generates acknowledge bits when addressed as a slave device.

Bit 10 of the SSCnIEN register (NACKEN) permits the setting of an interrupt on a NACK condition.

63.3.8 Arbitration checking

This situation only arises when two or more master devices generate a START condition within the minimum hold time of the bus standard. This generates a valid start condition on the bus with more than one master valid.

However, a master device cannot determine if two or more masters have generated a START condition, so arbitration is always enabled. The arbitration for which device wins control of the bus is determined by which master is the first to transmit a low data bit on the data line when the other master wants to send a high bit. This master wins control of the bus. Therefore a master which detects a different data bit on its input to that which it transmitted must switch off its output stage for the rest of the eight bit data byte, as it has lost the arbitration.

The arbitration scheme does not affect the data transmitted by the winning master. Consequently, arbitration proceeds concurrently with data transmission and the data received by the selected slave during the arbitration process. It is valid that the winning master is actually addressing the losing master and hence this device must respond as if it were a slave device.

Arbitration is implemented in hardware by comparing the transmitted and received data bits every cycle. Loss of arbitration is indicated by the setting of the ARBL arbitration lost error flag in the status register. An interrupt also occurs if the ARBLEN bit is set in the interrupt enables register.

Loss of arbitration also causes a clock stretch to be inserted if the master which has lost arbitration has been addressed. The interrupt and the clock stretch occurs immediately after the eight bits plus acknowledge. The clock stretch is cleared when the software reads the receive buffer.

64 Synchronous serial controller (SSC) registers

Register addresses are provided as *SSCnBaseAddress* + offset.

The *SSCnBaseAddresses* are:

SSC0: 0x1804 0000,
SSC1: 0x1804 1000,
SSC2: 0x1804 2000,
SSC3: 0x1804 3000.

Table 207: SSC register summary

Register	Description	Offset	Type
SSCn_BRG	SSCn baudrate generation	0x000	R/W
SSCn_TBUFF	SSCn transmit buffer	0x004	WO
SSCn_RBUFF	SSCn receive buffer	0x008	RO
SSCn_CTRL	SSCn control	0x00C	R/W
SSCn_INT_EN	SSCn interrupt enable	0x010	R/W
SSCn_STA	SSCn status	0x014	RO
SSCn_I2C_CTRL	SSCn I2C control	0x018	R/W
SSCn_SLA_ADDR	SSCn slave address	0x01C	WO
SSC_REP_START_HOLD_TIME	Programming repeated start hold time count value	0x020	R/W
SSC_START_HOLD_TIME	Programming start hold time count value	0x024	R/W
SSC_REP_START_SETUP_TIME	Programming repeated start setup time count value	0x028	R/W
SSC_DATA_SETUP_TIME	Programming data setup time count value	0x02C	R/W
SSC_STOP_SETUP_TIME	Programming stop setup time count value	0x030	R/W
SSC_BUS_FREE_TIME	Programming bus free time count value	0x034	R/W
SSC_TX_FSTAT	Transmitter FIFO status	0x038	RO
SSC_RX_FSTAT	Receiver FIFO status	0x03C	RO
SSC_RE_SCALER_BRG	Programming prescaler value for clock	0x040	R/W
SSC_CLR_STA	Clear status bits	0x080	
SSC_NOISE_SUPP_WID	Noise suppression width	0x100	R/W
SSC_PRESCALER	Clock prescaler	0x104	R/W
SSC_NOISE_SUPP_WID_DOUT	Noise suppression max output data delay width	0x108	
SSC_PRE_SCALER_DATAOUT	Prescaler data out	0x10C	

SSCn_BRG SSCn baudrate generation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRG															

Address: *SSCnBaseAddress* + 0x000
Type: R/W
Reset: 0x01
Description: This register is dual purpose. When reading, the current 16-bit counter value is returned. When a value is written to this address, the 16-bit reload register is loaded with that value.

When in slave mode, BRG must be zero.

BRG is only changed when initialization of the master is performed for a master transaction. When the SSC is master and either the addressed as slave or arbitration lost interrupts are fired then BRG must be reset to 0.

SSCn_TBUFF SSCn transmit buffer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															

Address: *SSCnBaseAddress* + 0x004
Type: WO
Reset: 0x00
Description: Transmit buffer data TD15 to TD0.

SSCn_RBUFF SSCn receive buffer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD[15:0]															

Address: *SSCnBaseAddress* + 0x008
Type: RO
Reset: 0x00
Description: Receive buffer data RD15 to RD0.

SSCn_CTRL SSCn control

15 14	13	12	11	10	9	8	7	6	5	4	3 2 1 0
Reserved	CLKST_RX_NOT_TX_EN	RX_FIFO_EN	Reserved	LPB	EN	MS	SR	PO	PH	HB	BM

Address: *SSCnBaseAddress* + 0x00C

Type: R/W

Reset: 0x00

Description:

[15:14] **Reserved**

[13] **CLKST_RX_NOT_TX_EN**
0 Enable clock stretch by transmitter 1 Enable clock stretch by receiver

[12] **RX_FIFO_EN**: Enable Rx side FIFO
0 FIFO bypassed 1 FIFO enabled

[11] **Reserved**

[10] **LPB**: SSC loopback
0: Disabled, 1: Shift register output is connected to shift register input

[9] **EN**: SSC enable
0: Transmission and reception disabled, 1: Transmission and reception enabled

[8] **MS**: SSC master select
0: Slave mode 1: Master mode

[7] **SR**: SSC software reset
0: Device is not reset 1: All functions are reset while this bit is set

[6] **PO**: SSC clock polarity control
0: Clock idles at logic 0 1: Clock idles at logic 1
Must be set in I^2C mode.

[5] **PH**: SSC clock phase control
0: Pulse in second half cycle 1: Pulse in first half cycle
Must be set in I^2C mode.

[4] **HB**: SSC heading control
0: LSB first 1: MSB first

[3:0] **BM**: SSC data width selection (reset value is illegal)
0000: Reserved, do not use this combination 0001: 2 bits
0010: 3 bits up to 1111: 16 bits

SSCn_INT_EN SSCn interrupt enable

15	14	13 12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SSCRHFI_EN	Reserved	REPSTRT_EN	NACK_EN	Reserved	ARBL_EN	STOP_EN	AAS_EN	Reserved	PE_EN	RE_EN	TE_EN	TI_EN	RI_EN

Address: *SSCnBaseAddress* + 0x010

Type: R/W

Reset: 0x00

Description: This register holds the interrupt enable bits, which can be used to mask the interrupts.

[15] **Reserved**

[14] **SSCRHFI_EN**: Receiver FIFO half full interrupt enable
1: Interrupt enabled

[13:12] **Reserved**

[11] **REPSTRT_EN**: I^2C repeated start condition interrupt enable
1: Repeated condition interrupt enabled

[10] **NACK_EN**: I^2C NACK condition interrupt enable
1: NACK condition interrupt enabled

[9] **Reserved**

[8] **ARBL_EN**: I^2C arbitration lost interrupt enable
1: Arbitration lost interrupt enabled

[7] **STOP_EN**: I^2C stop condition interrupt enable
1: Stop condition interrupt enabled

[6] **AAS_EN**: I^2C addressed as slave interrupt enable
1: Addressed as slave interrupt enabled

[5] **Reserved**

[4] **PE_EN**: Phase error interrupt enable
1: Phase error interrupt enabled

[3] **RE_EN**: Receive error interrupt enable
1: Receive error interrupt enabled

[2] **TE_EN**: Transmit error interrupt enable
1: Transmit error interrupt enabled

[1] **TI_EN**: Transmitter buffer empty interrupt enable
1: Transmitter buffer empty interrupt enabled

[0] **RI_EN**: Receiver buffer full interrupt enable
1: Receiver buffer interrupt enabled

SSCn_STA SSCn status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				REPSTRT	NACK	BUSY	ARBL	STOP	AAS	CLST	PE	RE	TE	TIR	RIR

Address: *SSCnBaseAddress* + 0x014

Type: RO

Reset: 0x02 (all active bits clear except TIR)

Description:

[15:12] **Reserved**

[11] **REPSTRT**: I^2C repeated start flag
1: I^2C repeated start condition detected

[10] **NACK**: I^2C NACK flag
1: NACK received

[9] **BUSY**: I^2C bus busy flag
1: I^2C bus busy

[8] **ARBL**: I^2C arbitration lost flag
1: Arbitration lost

[7] **STOP**: I^2C stop condition flag
1: Stop condition detected

[6] **AAS**: I^2C addressed as slave flag
1: Addressed as slave device

[5] **CLST**: I^2C clock stretch flag
1: Clock stretching in operation

[4] **PE**: Phase error flag
1: Phase error set

[3] **RE**: Receive error flag
1: Receive error set

[2] **TE**: Transmit error flag
1: Transmit error set

[1] **TIR**: Transmitter buffer empty flag
1: Transmitter buffer empty

[0] **RIR**: Receiver buffer full flag
1: Receiver buffer full

SSCn_I2C_CTRL SSCn I²C control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			I2CFSMODE	REPSTRTG	Reserved					TX_EN	AD10	ACKG	STOPG	STRTG	I2CM

Address: *SSCnBaseAddress* + 0x018

Type: R/W

Reset: 0x00

Description: To suit I²C specifications, bits PH and PO of register SSC_nCON must also be set to 1.

[15:13] **Reserved**

[12] **I2CFSMODE**: Configures standard or fast mode for I²C operation
0: Standard mode 1: Fast mode

[11] **REPSTRTG**: SSC I²C generate repeated START condition
0: Disabled 1: Enabled

[10:6] **Reserved**

[5] **TX_EN**: SSC I²C transaction enable control
0: Disabled 1: Enabled

[4] **AD10**: SSC I²C 10-bit addressing control
0: Disabled 1: Use 10 bit addressing

[3] **ACKG**: SSC I²C generate acknowledge bits
0: Disabled 1: Generate acknowledge bits when receiving

[2] **STOPG**: SSC I²C generate STOP condition
0: Disabled 1: Generate a STOP condition

[1] **STRTG**: SSC I²C generate START condition
0: Disabled 1: Generate a START condition

[0] **I2CM**: SSC I²C control
0: Disabled 1: Enable I²C features

SSCn_SLA_ADDR SSCn slave address

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						SL[9:7]			SL[6:0]						

Address: *SSCnBaseAddress* + 0x01C

Type: WO

Reset: 0x00

Description: The slave address is written into this register. If the address is a 10-bit address it is written into bits [9:0]. If the address is a 7-bit address then it is written into bits [6:0].

SSC_REP_START_HOLD_TIME Programming repeated start hold time count value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP_START_HOLD_TIME															

Address: *SSCnBaseAddress* + 0x020
Type: R/W
Reset: 0x01
Description: The value in this register corresponds to the I^2C repeated start hold time requirement.

[15:0] **REP_START_HOLD_TIME**
time = clock_period * (value in register)

SSC_START_HOLD_TIME Programming start hold time count value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START_HOLD_TIME															

Address: *SSCnBaseAddress* + 0x024
Type: R/W
Reset: 0x01
Description: The value in this register corresponds to the I^2C start hold time requirement.

[15:0] **START_HOLD_TIME**
time = clock_period * (value in register)

SSC_REP_START_SETUP_TIME Programming repeated start setup time count value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP_START_SETUP_TIME															

Address: *SSCnBaseAddress* + 0x028
Type: R/W
Reset: 0x01
Description: The value in this register corresponds to the I^2C repeated start setup time requirement.

[15:0] **REP_START_SETUP_TIME**
time = clock_period * (value in register)

SSC_DATA_SETUP_TIME Programming data setup time count value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_SETUP_TIME															

Address: *SSCnBaseAddress* + 0x02C
Type: R/W
Reset: 0x01
Description: The value in this register corresponds to the I^2C data setup time requirement.

[15:0] **DATA_SETUP_TIME**
time = clock_period * (value in register)

SSC_STOP_SETUP_TIME Programming stop setup time count value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP_SETUP_TIME															

Address: *SSCnBaseAddress* + 0x030
Type: R/W
Reset: 0x01
Description: The value in this register corresponds to the I^2C stop setup time requirement.

[15:0] **STOP_SETUP_TIME**
time = clock_period * (value in register)

SSC_BUS_FREE_TIME Programming bus free time count value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUS_FREE_TIME															

Address: *SSCnBaseAddress* + 0x034
Type: R/W
Reset: 0x01
Description: The value in this register corresponds to the I^2C bus free time requirement.

[15:0] **BUS_FREE_TIME**
time = clock_period * (value in register)

SSC_TX_FSTAT Transmitter FIFO status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													SSCTXF_STA		

Address: *SSCnBaseAddress* + 0x038
Type: RO
Reset: 0x00
Description: This register depicts the Tx FIFO status.

[15:3] **Reserved**

[2:0] **SSCTXFSTAT**: Tx FIFO status - number of words in the Tx FIFO:

000: 0 (empty)	100: 4
001: 1	101: 5
010: 2	110: 6
011: 3	111: 7 (full)

SSC_RX_FSTAT Receiver FIFO status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													SSCRXF_STA		

Address: *SSCnBaseAddress* + 0x03C
Type: RO
Reset: 0x00
Description: This register depicts the Rx FIFO status.

[15:13] **Reserved**

[2:0] **SSCRXFSTAT**: Rx FIFO status

Register contains the number of words in the Rx FIFO:

000: 0 (empty)
001: 1
010: 2
011: 3
100: 4
101: 5
110: 6
111: 7 (full)

SSC_RE_SCALER_BRG Programming prescaler value for clock

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRE_SCALER_BRG															

Address: *SSCnBaseAddress* + 0x040
Type: R/W
Reset: 0x01
Description: The value in this register is used to further pre-scale the clock generated according to the programming of the baud rate. It can be used in conjunction with the programming in register SSCn_BRG to slow down the serial clock generated in order to operate at lower frequencies.

[15:0] **PRE_SCALER_BRG**: Pre-scaling the BRG clock

SSC_CLR_STA Clear status bits

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				REPSTRT_CLR	NACK_CLR	Reserved	SSCARBL_CLR	SSCSTOP_CLR	SSCAAS_CLR	Reserved						

Address: *SSCnBaseAddress* + 0x080

Type: R/W

Reset: 0x00

Description: Clear bits in SSCn_STA.

[15:12] **Reserved**

[11] **CLR_REPSTRT**
1: Clear REPSTRT

[10] **CLR_NACK**
1: Clear SCCNACK

[9] **Reserved**

[8] **CLR_SSCARBL**
1: Clear SSC_ARBL

[7] **CLR_SSCSTOP**
1: Clear SSC_STOP

[6] **CLR_SSCAAS**
1: Clear SSC_AAS

[5:0] **Reserved**

SSC_NOISE_SUPP_WID Noise suppression width

7	6	5	4	3	2	1	0
NOISE_SUPP_WID							

Address: *SSCnBaseAddress* + 0x100

Type: R/W

Reset: 0x00

Description: The value, in microseconds, in this register determines the maximum width of noise pulses which the filter suppresses. To suppress glitches of n width, load n+1 in this register. All signal transitions whose width is less than the value in SSC_NOISE_SUPP_WID are suppressed. Writing 0x00 into this register bypasses the antiglitch filter.

SSC_PRESCALER Clock prescaler

7	6	5	4	3	2	1	0
Reserved				PRESCALE_VAL			

Address: *SSCnBaseAddress* + 0x104

Type: R/W

Reset: 0x00

Description: This register holds the prescaler division factor for glitch suppression, equivalent to 10 MHz. For example if the comms clock is 50 MHz the prescaler division factor should be 5.

SSC_NOISE_SUPP_WID_DOUT Noise suppression max output data delay width

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								NSWD							

Address: *ASCnBaseAddress* + 0x108

Type: R/W

Reset: 0

Description: Holds the maximum delay width by which output data has to be delayed.

SSC_PRE_SCALER_DATAOUT Prescaler data out

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												PSDO			

Address: *ASCnBaseAddress* + 0x10C

Type: R/W

Reset: 0

Description: Pre-scaler division factor of input system clock. The output of this section is fed to the baud rate generator to generate the serial clock.

65 CSS/CPxM

This block has the following capabilities:

- decryption for DVD-video and DVD-audio formats,
- encryption for recordable media (CPRM),
- ATAPI DVD drive authentication,
- CSS decoding.

The content protection for recorded or prerecorded media (CPRM/CPPM) defines a robust and renewable method for protecting content stored on a number of physical media types.

CPPM decoding and CPRM encoding can be used simultaneously. Encode and decode processes may be carried out simultaneously for all process types (CPPM, CPRM and CSS); however, only one device key set can be loaded at a time. This means that the key has to be changed when switching between CPPM and CPRM.

66 Digital satellite equipment control (DiSEqC) 2.0

66.1 Overview

The DiSEqC (digital satellite equipment control) bus protocol provides nonproprietary commands to enable communication between satellite receivers and satellite peripheral equipment via a coaxial cable. The DiSEqC system supports the DiSEqC v2.0 standard, which is a single master, multiple slave bidirectional system and uses a pulse width keying coded signal to perform peripheral device functions such as controlling polarization skew, remotely selecting a LNB local oscillator frequency, moving a steerable antenna to point in the required direction.

The pulse width and the bit interval are software-programmable, which makes the DiSEqC interface highly flexible to support different pulse width keying coding protocols.

This block transmits and receives the DiSEqC 2.0 compatible frames.

The processor provides the framing (run-in), address, command and the ancillary data fields of the message.

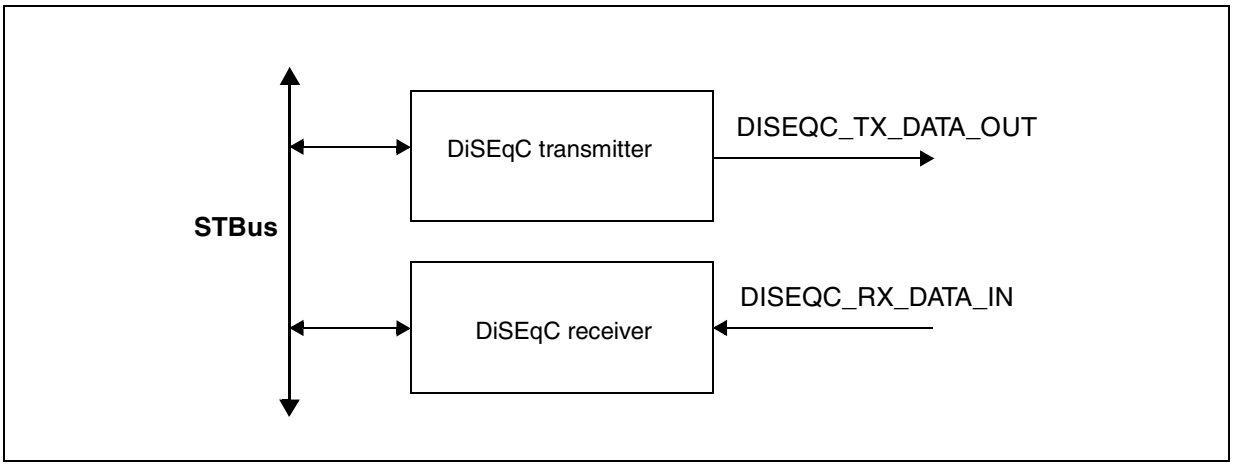
In the transmitter, the complete message including the framing, address, command and the payload data bytes are coded using programmable PWK data, and transmitted using a programmable subcarrier frequency. An interrupt is generated after the total message is transmitted.

In the receiver the payload received is stored in a local FIFO which is accessible by the processor. An interrupt is generated when pre programmed conditions are met by the module. The pulse width and symbol periods have to be programmed separately for the receive section.

Note: Subcarrier suppression for the receiver is carried out externally.

The DiSEqC block diagram is shown in Figure 257 below.

Figure 257: Block diagram of DiSEqC control module



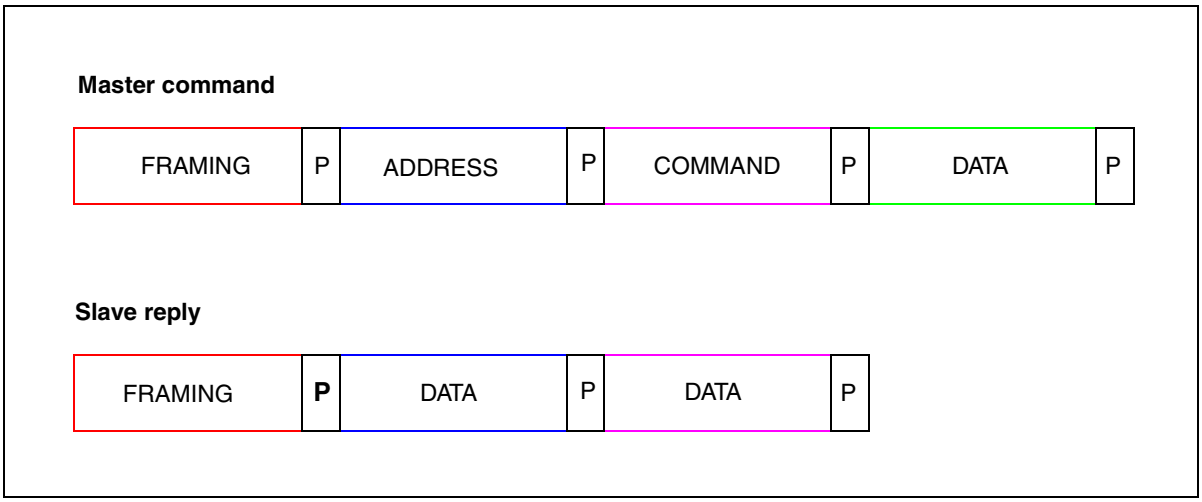
The function of the sub-blocks is explained in the following sections.

66.2 DiSEqC transmitter

The DiSEqC transmitter generates DiSEqC-compatible data messages. Each message data format is composed of a start (run-in) framer byte, a slave address byte, a command byte and ancillary data bytes which may depend upon the command structure. Each byte is followed by an odd parity check bit. The bits are transmitted as a continuous sequence until the message is complete.

A complete message frame is shown in Figure 258 below.

Figure 258: DiSEqC Frame



Each bit in a DiSEqC 2.0 message is represented using a pulse width keying coded signal. The bits are modulated by a subcarrier (square wave) before transmission. The generation of the framing byte and the parity enable or disable or selection of odd or even parity is software programmable. The address, command and data bytes required to be transmitted in the payload are also written by the processor.

A minimum programmed silence period is ensured by the transmitter between two consecutive messages.

An interrupt is asserted after a complete message is transmitted and the programmed silence period has expired.

66.2.1 Transmit data buffer

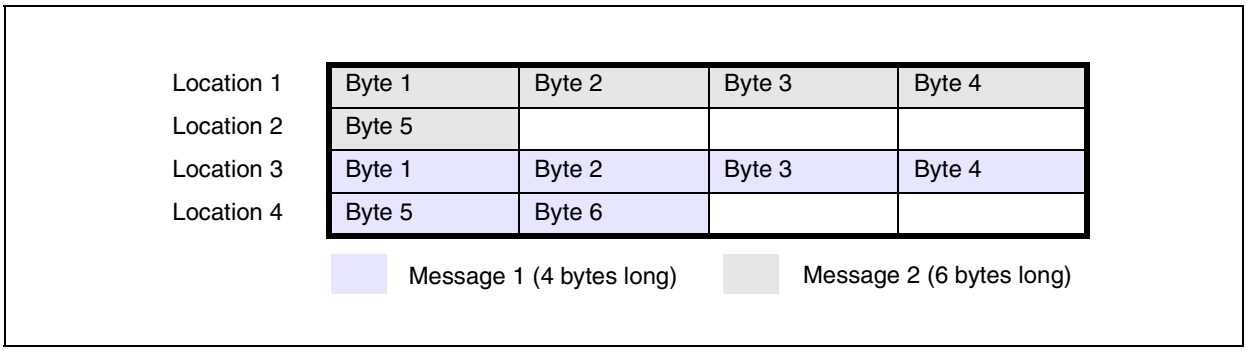
The transmission payload is stored inside the transmit data buffer register (DSQ_TX_DATA_BUFF). This register is a 32-bit buffer which holds the framing, slave address, DiSEqC commands and the ancillary data bytes to be transmitted. Software must format the message frame including the framer bytes. The buffer is quadruple buffered.

The data buffer is organized as 32 bits wide and 4 words deep. New messages are stored in the buffer word aligned on 4-byte words.

For example if there are two messages of 5 and 6 bytes in length, the first 4 bytes of the first message are stored at the first location and the remaining byte of the first message in the second location of the data buffer. The first 4 bytes of the second message are stored at the third location

and the remaining 2 bytes of the second message at the fourth location. This is shown in Figure 259.

Figure 259: Transmit data buffer organization



66.3 DiSEqC receiver

The DiSEqC receiver receives serial DiSEqC messages. The received payload bits are packed into bytes and stored in a buffer after performing PWK decoding on the data bits. The receiver checks for parity errors, symbol errors and code errors in the received message frame and reports them in the status register. An interrupt is generated if enabled.

End of message is indicated by no activity on the DiSEqC line for a programmed silence period.

The data fed to the receiver is the envelope of the received data. The subcarrier is suppressed external to the DiSEqC module.

The continuous tone and the tone burst (both modulated and unmodulated) occur at different voltage levels and are suppressed from the received DiSEqC message external to the DiSEqC module.

66.4 Noise suppression filter

A programmable noise suppression filter is used to suppress any noise pulses of a programmable width in the received DiSEqC frame. The noise suppression is programmed by writing appropriate value in noise suppression register. If a value *n* is written in this register then any pulses (HIGH or LOW) whose width is less than *n* sampling cycle period are treated as noise and will be filtered from the received data stream. Writing 0x00 in the noise suppression register disables noise suppression.

66.5 Programming the transmitter

66.5.1 Normal message transmission

Configuring the message to be transmitted

Configuration is carried out for the first message to be transmitted. For subsequent messages individual parameters can be reconfigured as required.

1. **Subcarrier period**:
   DSQ_TX_SUBCARR_DIV[0:15] and DSQ_TX_PRESCALER[0:7]
   The DSQ_TX_SUBCARR_DIV register is configured so the output is twice the required subcarrier frequency. The output is divided by 2 to generate a square wave of the requisite frequency with 50% duty cycle. For example to generate a subcarrier of 22 kHz the value to be programmed in DSQ_TX_SUBCARR_DIV must be 1511 if the value programmed in DSQ_TX_PRESCALER is 2.
2. **Symbol time**: DSQ_TX_SYMBOL_PER[0:7]
   The value represents in multiples of subcarrier periods the time used to represent each symbol in the message. For a 22 kHz carrier and a symbol time of 1.5 µs, the register is programmed with a value of 33.
3. **Silence period**: DSQ_TX_SILENCE_PER[0:15]
   Ensures a gap of the programmed period between two consecutive messages. This is an integer number denoting a number of subcarrier cycles for which a silence period is to be maintained.
4. **Symbol n on time**: DSQ_TX_SYMBOLN_ONTIME[0:7]
   This is an integer number denoting in number of subcarrier cycles of the on period of the corresponding symbol.
5. **Type of message**: DSQ_TX_MSG_CFG[0:1, 15]
   Set to 000 for a normal message type. See Section 66.5.2: *Configuring and sending tone burst messages* *on page 830* and Section 66.5.3: *Configuring and sending continuous tone messages* *on page 831* for details of other types of message.
6. **Length of message**: DSQ_TX_MSG_CFG[2:8]
   This is the number of data bytes to be sent.
7. Clear the soft reset by writing 0 in the DSQ_TX_SW_RST register.
8. Enable the transmitter by writing 1 into the DSQ_TX_EN register.

Transmitting the message

Repeat the following programming for each message to be transmitted.

1. Check if the TX_RDY bit in the DSQ_TX_STA register is 1. To start transmitting this bit should be 1. If this bit is 0, then the software must wait until this bit is set to 1.
2. Load the DSQ_TX_DATA_BUFF with the data bytes.
3. Enable the interrupt generation from the transmit section by writing into register DSQ_TX_INT_EN.
4. Start transmission of the new message by writing 1 into the DSQ_TX_START register.
   The transmission then starts transmitting frame, address, command and data fields.
   The TX_RDY bit is cleared indicating that the transmitter is busy sending the message.
   The data transmitted is automatically appended with the odd parity bit.
   This byte is PWK encoded and the resulting data bits are serially transmitted.
   The transmission continues until the message payload length as specified in DSQ_TX_MSG_CFG is reached.
   An interrupt is generated after transmitting the last bit of the message.

5. When the message length is greater than the FIFO depth and new data is written in the DSQ_TX_DATA_BUFF before the last symbol is transmitted, the remaining bytes are transmitted until the message is complete. After the last symbol is transmitted the TX_RDY status bit is asserted high and the silence period prevails on the DiSEqC line.
6. If the transmitter is configured for a new message during the silence period it starts transmitting the new message immediately after the expiry of the silence period if the DSQ_TX_START bit is set.
7. If the transmitter is not configured for a new message during the silence period it will keep the DiSEqC line idle.
8. If the transmitter is configured for transmitting a modulated or unmodulated tone burst before the expiry of the silence period of the current message (bit 1 of register DSQ_TX_MSG_CFG), if the DSQ_TX_START bit is set the transmission of the selected tone burst is started immediately after the expiry of the silence period. The tone burst is transmitted for the duration for which it is programmed (as specified in bits 9 to 15 of register DSQ_TX_MSG_CFG).

Configuring and sending the next messages

Reprogramming for a next normal message starts from the onset of the silence period of the current message (when the TXREADY status is asserted high).

1. Change the properties of the message to be transmitted by programming these registers as required:
   - **subcarrier period**: DSQ_TX_SUBCARR_DIV (bits 0 to 15) and DSQ_TX_PRESCALER (bits 0 to 7),
   - **symbol time**: DSQ_TX_SYMBOL_PER (bits 0 to 7),
   - **silence period**: DSQ_TX_SILENCE_PER (bits 0 to 15),
   - **symbol n on time**: DSQ_TX_SYMBOL*n*_ONTIME (bits 0 to 7),
   - **type of message**: DSQ_TX_MSG_CFG (bits 0 to 1 and 15),
   - **length of message**: DSQ_TX_MSG_CFG (bits 2 to 8).

   See *Configuring the message to be transmitted* on page 828 for further details.
2. Check if the TX_RDY bit in the DSQ_TX_STA register is 1. To start transmitting this bit should be 1. If this bit is 0, then the software must wait until this bit is set to 1.
3. If set then clear the soft reset by writing 0 in the DSQ_TX_SW_RST register.
4. Load the DSQ_TX_DATA_BUFF with the data bytes.
5. If disabled, enable the interrupt generation from transmit section by writing into register DSQ_TX_INT_EN.
6. Start transmission of the new message by writing 1 into the DSQ_TX_START register.

   The transmission of the new message begins immediately after the silence period of the current message has expired if the transmitter was executing the silence period of a previous message. If idle the new message transmission starts immediately after the execution of the TX_START bit.

Underrun errors

When the message length is greater than the FIFO depth an underrun condition is generated if new data is not written in the DSQ_TX_DATA_BUFF before the last symbol is transmitted. For an underrun condition the TX_UNDERRUN error bit in the DSQ_TX_STA register is set. Two possible scenarios exist in which the TX_UNDERRUN status bit must be cleared.

If the underrun interrupt is enabled and an underrun interrupt is generated:

1. On detection of an interrupt, the processor reads the DSQ_TX_INT_STA to infer the interrupt cause.
2. The processor disables the transmitter by writing 0 in the DSQ_TX_EN register.
3. The processor then clears the underrun interrupt by performing a clear bit operation on the corresponding underrun clear bit on DSQ_TX_CLR_INT_STA location.
4. The processor then writes in the DSQ_TX_DATA_BUFF to clear the underrun status from the DSQ_TX_STA register.
5. Execute a soft reset by writing a 1 in the DSQ_TX_SW_RST register to clear the DSQ_TX_DATA_BUFF.
6. Return to step 1 in *Transmitting the message* *on page 828*.

If the underrun interrupt is not enabled, an underrun interrupt is not generated:

1. The processor reads the DSQ_TX_STA register (polls) to determine the state of the DiSEqC cell to determine whether an underrun condition has occurred.
2. The processor disables the transmitter by writing 0 in the DSQ_TX_EN register.
3. The processor then writes in DSQ_TX_DATA_BUFF to clear the underrun status from the DSQ_TX_STA register.
4. Execute a soft reset by writing 1 in the DSQ_TX_SW_RST register to clear the DSQ_TX_DATA_BUFF.
5. Return to step 1 in *Transmitting the message* *on page 828*.

66.5.2 Configuring and sending tone burst messages

1. Configure the DSQ_TX_MSG_CFG register for the type of tone burst (bit 1, modulated or unmodulated) and the duration of tone burst (bits 9 to 15).
2. If disabled, enable the interrupt generation from the transmit section by writing into register DSQ_TX_INT_EN.
3. Check if the TX_RDY bit in the DSQ_TX_STA register is 1. To start transmitting a tone burst this bit should be 1. If this bit is 0, then the software must wait until this bit is set to 1.
4. If set then clear the soft reset by writing 0 in the DSQ_TX_SW_RST register.
5. Start transmission of the tone burst message by writing 1 into the DSQ_TX_START register.
6. The DSQ_TX_DATA_BUFF register does not need to be programmed during tone burst mode.
7. The transmission of the tone burst message begins immediately after the silence period of the current message has expired if the transmitter was executing the silence period of a previous message else if idle the tone burst starts immediately after the execution of the TX_START bit.

66.5.3 Configuring and sending continuous tone messages

1. Configure the DSQ_TX_MSG_CFG register for continuous tone transmission by writing 1 in bit 16 of the DSQ_TX_MSG_CFG register.
2. Check if the TX_RDY bit in the DSQ_TX_STA register is 1. To start transmitting a continuous tone this bit should be 1. If this bit is 0, then the software must wait until this bit is set to 1.
3. If set then clear the soft reset by writing 0 in the DSQ_TX_SW_RST register.
4. If the DiSEqC interface is programmed for continuous tone transmission when it is already in the middle of the silence period, the transmission of the continuous tone begins only after the expiry of the current silence period and it does not require the programming of the DSQ_TX_START register.

66.6 Programming the receiver

66.6.1 Normal message reception

Configuring the message to be received

Configuration is carried out for the first message to be received. For subsequent messages individual parameters can be reconfigured as required.

1. **Sampling clock**: DSQ_RX_SAMPLING_PER[0:15]
   Program the sampling clock. For example, to generate a sampling clock of 660 kHz from a system clock of 108 MHz, program this register with a value of 164.
2. **Silence period**: DSQ_RX_SILENCE_PER[0:15]
   Program the silence period to detect the end of message. For example, for a silence period of 12.5 µs with an RX_SAMPLING clock of 660 kHz the register needs to be programmed with a value of 8250.
3. **Minimum threshold symbol on time limits**: DSQ_RX_SYMBOLn_MIN_THOLD[0:15]
   Define the minimum threshold duration needed to determine symbol *n* on time in the received PWK coded message (for *n* = 0 and 1). The value is represented as multiples of sampling clock periods.
   For a sampling clock of 660 kHz, the register needs to be programmed with a value 265 to achieve 0.4 µs minimum threshold time to detect a symbol *n*.
4. **Maximum threshold symbol on time limits**: DSQ_RX_SYMBOLn_MAX_THOLD[0:15]
   Define the maximum threshold duration needed to determine symbol *n* on time in the received PWK coded message (for *n* = 0 and 1). The value is represented as multiples of sampling clock periods.
   For a sampling clock of 660 kHz, the register needs to be programmed with a value 529 to achieve 0.8 µs max threshold time to detect a symbol *n*.
5. If set, clear the soft reset by writing 0 in the DSQ_RX_SW_RST register.
6. Enable the receive section by writing 1 into register DSQ_RX_EN.

Receiving a message

Repeat the following for each message received:

1. Enable the interrupt generation by writing 1 into register DSQ_RX_INT_EN.
2. The receiver is enabled. The RX_BUSY status bit in register DSQ_RX_STA is set on the reception of the first symbol of a message. This bit is cleared after the expiry of the silence period of the message being currently received.
3. PWK decoding is performed on the received payload. Each byte of the received payload including the frame byte is stored in DSQ_RX_DATA_BUFF.

4. The number of bytes received is calculated and reflected in register DSQ_RX_BYTE_CNT.
5. An interrupt is generated when the last data bit of the payload is received.
6. If the message length is greater than the FIFO depth, and the received data is not read from DISEQC_RX_DATA_BUFF before the last bit is stored in the RX_BUFFER (RX_FULL), an overflow condition is generated. At overflow, the Rx overflow error bit in register DISEQC_RX_STA is set. Two possible scenarios exist in which the RX_OVERFLOW status bit has to be cleared:

   6.1 If the overflow interrupt is enabled, an overflow interrupt is generated on the PER_INTERRUPT line:

   (a) On detection of an interrupt the processor reads DISEQC_RX_INT_STA to infer the interrupt cause.
   (b) The processor disables the receiver by writing '0' in the DISEQC_RX_EN register.
   (c) The processor then clears the overflow interrupt by performing a clear bit operation on the corresponding overflow clear bit on DISEQC_RX_CLR_INT_STA location.
   (d) The processor then reads the contents of the DISEQC_RX_DATA_BUFF to clear the overflow status from the DISEQC_RX_STA register.
   (e) The processor can execute a soft reset to clear the DISEQC_RX_DATA_BUFF and to clear the overflow status from the DISEQC_RX_STA register.

   6.2 If the overflow interrupt is not enabled, an overflow interrupt is not generated on the PER_INTERRUPT line:

   (a) The processor reads the DISEQC_RX_STA register (polls) to determine the state of the diseqc cell to determine whether an overflow condition has occurred.
   (b) The processor disables the receiver by writing '0' in the DISEQC_RX_EN register.
   (c) The processor reads from DISEQC_RX_DATA_BUFF to clear the overflow status from the DISEQC_RX_STA register.
   (d) The processor can either execute a soft reset to clear the DISEQC_RX_DATA_BUFF and clear the overflow status from the DISEQC_RX_STA register.
7. If the received message length is greater than the FIFO depth, and data is read from the DISEQC_RX_DATA_BUFF before the last bit is written into the buffer, the receiver continues receiving the remaining bytes until the message is complete. After the last symbol is detected, the receiver asserts the Rx Busy status bit low, till a new message is received.
8. When the **symbol On Time** of either symbol 0 or symbol 1 does not fall within the programmed min/max threshold limits specified in the registers (DISEQC_RX_SYMB*n*_MIN_THRESH/ DISEQC_RX_SYMB*n*_MAX_THRESH where *n* = 0, 1) the message is said to have a **symbol error**.

   If at least one symbol in the received DiSEqC message violates the programmed threshold limits, the symbol error status bit in the DISEQC_RX_STA register is set. If the symbol error interrupt is enabled, a symbol error interrupt is generated on the PER_INTERRUPT line.
9. When the **number of bits** received in a DiSEqC message is not a multiple of eight, **or** the inter-separation time between bits is not equal to the value programmed in the DISEQC_RX_SYMB_PER register **or** the inter-separation time between individual bytes in a diseqc frame is greater than or equal to the value programmed in the DISEQC_RX_SYMB_PER register, the message is said to have a **code error**: the code error status bit in the DISEQC_RX_STA register is set. If the code error interrupt is enabled, a code error interrupt is generated on the PER_INTERRUPT line.
10. If the parity of the received symbols is **even** then the message is said to have a **parity error**: the parity error status bit in the DISEQC_RX_STA register is set. If the parity error interrupt is enabled, a parity error interrupt is generated on the PER_INTERRUPT line.

Configuring and receiving the next messages

1. Change the properties of the message to be received by programming these registers as required:
   - **sampling clock**: DSQ_RX_SAMPLING_PER[0:15],
   - **minimum symbol period**: DSQ_RX_MIN_SYMBOL_PER[0:15],
   - **maximum symbol period**: DSQ_RX_MAX_SYMBOL_PER[0:15],
   - **silence period**: DSQ_RX_SILENCE_PER[0:15],
   - **minimum threshold symbol on time limits**: DSQ_RX_SYMBOLN_MIN_THOLD[0:15],
   - **maximum threshold symbol on time limits**: DSQ_RX_SYMBOLN_MAX_THOLD[0:15].

See *Configuring the message to be received on page 831* for further details.

1. Check if bit DSQ_RX_STA.RX_BUSY is set to 0. To start receiving, this bit should be 0. If this bit is 1, then the software must wait until this bit is reset to 0.
2. If set then clear the soft reset by writing 0 in the DSQ_RX_SOFT_RST register.
3. If disabled, enable the interrupt generation from the receive section by writing into register DSQ_RX_INT_EN.
4. Enable reception of the new message by writing 1 into register DSQ_RX_EN.

Overflow errors

When the message length is greater than the FIFO depth an overflow condition is generated if the received data is not read from the DSQ_RX_DATA_BUFF before the last bit is stored in the RX buffer (RX_FULL). For an overflow condition the RX_OVERFLOW error bit in the DSQ_RX_STA register is set. Two possible scenarios exist in which the RX_OVERFLOW status bits must be cleared.

If the overflow interrupt is enabled, an overflow interrupt is generated:

1. On detection of an interrupt the processor reads the DSQ_RX_INT_STA to infer the interrupt cause.
2. The processor disables the receiver by writing 0 in the DSQ_RX_EN register.
3. The processor then clears the overflow interrupt by performing a clear bit operation on the corresponding overflow clear bit on DSQ_RX_CLR_INT_STA location.
4. The processor then reads the contents of the DSQ_RX_DATA_BUFF to clear the overflow status from the DSQ_RX_STA register.
5. The processor can either execute a soft reset to clear the DSQ_RX_DATA_BUFF and clear the overflow status from the DSQ_RX_STA register.

If the overflow interrupt is not enabled, an overflow interrupt is not generated:

1. The processor then reads (polls) the DSQ_RX_STA register to determine whether an overflow condition has occurred.
2. The processor disables the receiver by writing 0 in register DSQ_RX_EN.
3. The processor then reads from the DSQ_RX_DATA_BUFF to clear the overflow status from register DSQ_RX_STA.
4. The processor can either execute a soft reset to clear the DSQ_RX_DATA_BUFF and clear the overflow status from register DSQ_RX_STA.
5. When the message length is greater than the FIFO depth and data is being read from the DSQ_RX_DATA_BUFF before the last bit is written into the buffer, the remaining bytes are received until the message is complete. After the last symbol is detected the receiver asserts the RX_BUSY status bit low until no more new messages are received.

Symbol errors

When the symbol on time of either symbol 0 or symbol 1 does not fall within the minimum or maximum threshold limits specified in DSQ_RX_SYMBOLn_MIN_THOLD or DSQ_RX_SYMBOLn_MAX_THOLD the message has a symbol error.

If at least one symbol in the received DiSEqC message violates the programmed threshold limits, the SYMBOL_ERR status bit in the DSQ_RX_STA register is set. If the SYMBOL_ERR interrupt is enabled a symbol error interrupt is generated.

Code errors

A code error occurs for any of the conditions listed below.

- The number of bits received in the message is not a multiple of eight. Thus there are either more or less bits per byte in the message.
- The time between bits or bytes in the received message is not equal to the value programmed in the DSQ_RX_SYMBOL_PER register.
- The time between individual bytes in a DiSEqC frame is greater than or equal to the value programmed in the DSQ_RX_SYMBOL_PER register.

Bit DSQ_RX_STA.CODE_ERR is set. If the CODE_ERR interrupt is enabled, a code error interrupt is generated.

The entire message has a code error when one bit in the received DiSEqC message has a code error.

Parity errors

The message has a parity error when the parity of the received symbols is not odd. The PARITY_ERR status bit in the DSQ_RX_STA register is set. If the PARITY_ERR interrupt is enabled a parity error interrupt is generated.

66.6.2 Modulated / unmodulated tone burst reception

Modulated and unmodulated tone burst reception is suppressed external to the DiSEqC module. For detection of a modulated/unmodulated tone burst the external device must ensure a logic 0 at the DiSEqC receiver input.

66.6.3 continuous tone reception

Continuous tone reception is suppressed external to the DiSEqC module. For detection of a modulated/unmodulated tone burst the external device must ensure a logic 0 at the DiSEqC receiver input.

66.6.4 Subcarrier generation

The prescaler divides the system clock to get the required granularity. The prescaler clock is further divided by the subcarrier value to generate a clock of 44 kHz, which is then divided by two to get a 22 kHz subcarrier with 50% duty cycle.

For example with a system clock frequency of 133 MHz:

To generate 44 kHz the 133 MHz has to be divided by a factor
(133 * 10 ^ 6) / (44 * 10 ^ 3) = 3022.72.

If the division factor is varied from 3020 to 3024, the % variation in the subcarrier clock (22 kHz) is documented in Table 208 for a system clock frequency of 133 MHz.

Table 208: Subcarrier frequency variation for a system clock of 133 MHz.

Prescalar value	Subcarrier value	Division value	Subcarrier freq (Hz)	Offset (Hz)	% Variation
2,5,10	1510,604,302	3020	22019.8	*+19.8*	0.900
3	1007	3021	22012.5	+12.5	0.057
2	1511	3022	22005.3	+5.3	0.024
1	3023	3023	21998.0	- 2.0	0.009
7,14,28,56	432,216,108,54	3024	21990.7	- 9.3	0.042

Since the allowed variation in subcarrier frequency is 20%, optimum division choices are shown in shaded boxes in Table 208.

67 DiSEqC registers

Register addresses are provided as *DiSEqCBaseAddress* + offset.

The *DiSEqCBaseAddress* is:

0x1806 8000.

Table 209: DiSEqC register summary

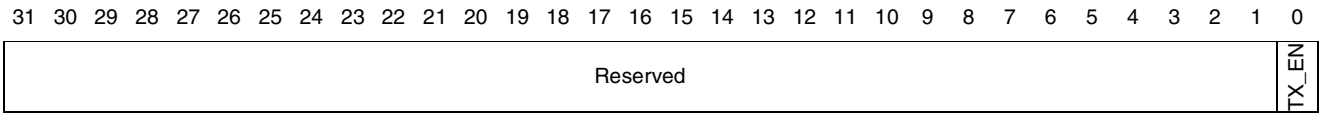
Register	Description	Offset	Type
Transmitter			
DSQ_TX_EN	Enable message transmission	0x000	R/W
DSQ_TX_MSG_CFG	Transmitter message configuration	0x004	R/W
DSQ_TX_PRESCALER	Transmitter prescaler value	0x008	R/W
DSQ_TX_SUBCARR_DIV	Transmitter subcarrier divider	0x00C	R/W
DSQ_TX_SILENCE_PER	Transmitter silence period	0x010	R/W
DSQ_TX_DATA_BUFF	Transmitter data buffer	0x014	WO
DSQ_TX_SYMB_PER	Transmitter symbol period	0x018	R/W
DSQ_TX_SYMB0_ONTIME	Symbol 0 on-time period	0x01C	R/W
DSQ_TX_SYMB1_ONTIME	Symbol 1 on-time period	0x020	R/W
DSQ_TX_SW_RST	Transmitter soft reset	0x024	R/W
DSQ_TX_START	Start message transmission	0x028	R/W
DSQ_TX_INT_EN	Transmitter interrupt enable	0x02C	R/W
DSQ_TX_INT_STA	Transmitter interrupt status	0x030	RO
DSQ_TX_STA	Transmitter status	0x034	RO
DSQ_TX_CLR_INT_STA	Clear transmitter interrupt status	0x038	WO
Reserved		0x03C	-
DSQ_TX_AGC	Transmitter automatic gain control	0x040	R/W
Receiver			
DSQ_RX_EN	Receiver enable	0x080	R/W
DSQ_RX_SAMPLING_PER	Receiver sampling period	0x084	R/W
DSQ_RX_BYTE_CNT	Receiver byte count	0x088	RO
DSQ_RX_SILENCE_PER	Receiver silence period	0x08C	R/W
DSQ_RX_DATA_BUFF	Received data buffer	0x090	RO
DSQ_RX_SYMB0_MIN_THOLD	Receiver minimum threshold for symbol 0 on-time	0x094	R/W
DSQ_RX_SYMB0_MAX_THOLD	Receiver minimum threshold for symbol 0 on-time	0x09C	R/W
DSQ_RX_SYMB1_MIN_THOLD	Receiver maximum threshold for symbol 1 on-time	0x098	R/W
DSQ_RX_SYMB1_MAX_THOLD	Receiver maximum threshold for symbol 1 on-time	0x0A0	R/W
DSQ_RX_SW_RST	Receiver soft reset	0x0A4	R/W
DSQ_RX_INT_EN	Receiver interrupt enable	0x0A8	R/W
DSQ_RX_INT_STA	Receiver interrupt status	0x0AC	RO
DSQ_RX_STA	Receiver status	0x0B0	RO
DSQ_RX_CLR_INT_STA	Clear receiver interrupt status	0x0B4	WO
DSQ_RX_TIMEOUT	Receiver timeout	0x0B8	-
DSQ_RX_NSUPP_WID	Receiver noise suppression width	0x0BC	R/W

Table 209: DiSEqC register summary

Register	Description	Offset	Type
DSQ_POL_INV	Receiver polarity invert	0x0C0	R/W
DSQ_RX_SUBCARR_NSUPP_EN	Receiver subcarrier noise suppression enable	0x0C4	R/W

67.1 Transmitter

DSQ_TX_EN Enable message transmission



Address: *DiSEqCBaseAddress* + 0x000

Type: R/W

Reset: 0

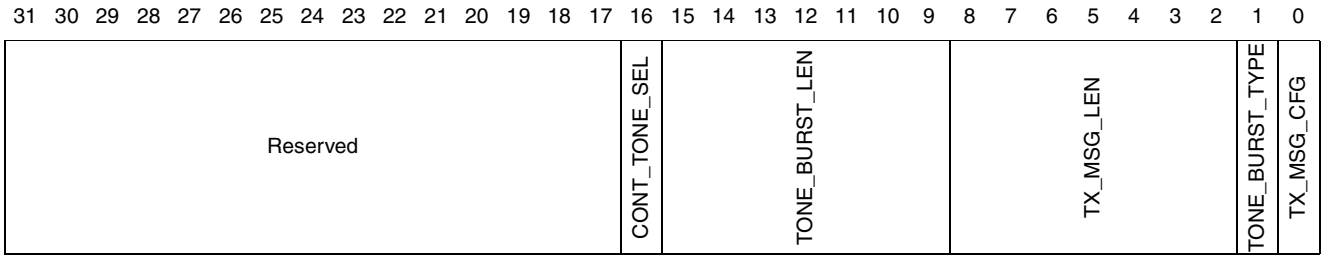
Description: Bit 0 is used to enable or disable the transmission of DiSEqC messages. Writing a 1 into the enable bit of this register enables the transmission. Writing 0 disables transmission.

Writing inside register DSQ_TX_EN enables the transmitter but does not start transmission.

[31:1] **Reserved**

[0] **TX_EN**
0: Disable transmitter
1: Enable transmitter

DSQ_TX_MSG_CFG Transmitter message configuration



Address: *DiSEqCBaseAddress* + 0x004

Type: R/W

Reset: 0

Description: Bit 0 of this register selects whether the message to be transmitted on the DiSEqC bus is a normal message or a tone burst (required for backward compatibility). Writing 1 into the LSB bit of this register enables the generation of tone burst.

Bit 1 of this register selects one of the two possible types of tone burst. The default tone burst is the unmodulated tone burst. However to generate a modulated tone burst a value of 1 has to be written in this bit. Bit 1 of this register has significance only if bit 0 of this register has a value of 1.

For normal messages the number of bytes to be transmitted in the message is specified by the remaining 7 bits of this register (bits 2 to 8). This ensures support of a maximum message payload length of 127 bytes (1 header byte + 1 address byte + 1 command byte + 124 data bytes).

Bits 9 to 15 specify the duration of the tone burst as an integer multiple of symbol periods.

Bit 16 selects whether a continuous tone has to be sent. To transmit a continuous tone a value of 1 is written at this location.

[31:17] **Reserved**

[16] **CONT_TONE_SEL**: Continuous tone select
0: No continuous tone 1: Continuous tone selected

[15:9] **TONE_BURST_LEN:** tone burst duration in integer multiple of Tx symbol period.
000 0001: Tone burst length is 1 symbol period 000 1001: Tone burst length is 9 symbol periods
111 1111: Tone burst length is 127 symbol periods 000 0000: Tone burst length is 128 symbol periods

[8:2] **TX_MSG_LEN:** total message length.
000 0011: Message length is 3 000 0100: Message length is 4
000 0101: Message length is 5 111 1111: Message length is 127

[1] **TONE_BURST_TYPE**
0: Unmodulated tone burst message 1: Modulated tone burst message

[0] **TX_MSG_CFG**
0: Normal message 1: Tone burst message

DSQ_TX_PRESCALER Transmitter prescaler value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PRESCALAR_VAL							

Address: *DiSEqCBaseAddress* + 0x008

Type: R/W

Reset: 0

Description: This 8-bit register is used to select the prescaler value for clock division. To get a clock division by a value *n* the prescaler register must be loaded with the value *n*.

[31:8] **Reserved**

[7:0] **PRESCALAR_VAL**: Prescalar value
A value of 0 sets clock division to 256.

DSQ_TX_SUBCARR_DIV Transmitter subcarrier divider

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SUBCARR_VAL															

Address: *DiSEqCBaseAddress* + 0x00C

Type: R/W

Reset: 0

Description: The output of prescaler is fed to the subcarrier generation unit. The subcarrier of the required frequency is generated by dividing the output of the prescaler by an appropriate value. To divide the output of the prescaler by a value *m* the subcarrier generation register is to be loaded with the value *m*. The time period of the subcarrier thus generated is given by the following formula:

$F_{(subcarrier)} = F_{(system\ clock\ frequency)} / (2n \times m)$

Where:

- *n* is the value loaded into the DSQ_TX_PRESCALER, and
- *m* is the value loaded into DSQ_TX_SUBCARR_DIV.

Refer to Section 66.6.4: *Subcarrier generation on page 834* for more details on subcarrier generation.

For achieving a subcarrier with 50% duty cycle the output of the subcarrier generation unit is divided by two. Hence if the desired subcarrier division ratio is *n* the value to be programmed in DSQ_TX_SUBCARR_DIV register must be *n*/2.

A value of 0 ensures the prescaler output division by 65536.

DSQ_TX_SILENCE_PER Transmitter silence period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TX_SILENCE_PER															

Address: *DiSEqCBaseAddress* + 0x010

Type: R/W

Reset: 0

Description: This register specifies the intermessage separation time.This number is specified as integer multiples of subcarrier cycles of silence after each message.

DSQ_TX_DATA_BUFF Transmitter data buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAYLOAD																															

Address: *DiSEqCBaseAddress* + 0x014

Type: WO

Buffer: Quadruple

Reset: 0

Description: The transmission payload is stored inside this buffer.This register is a 32-bit buffer which holds the framing, slave address, DiSEqC commands and the ancillary data bytes to be transmitted. It is the responsibility of software to format the message frame including the framer bytes.

Note: *The data buffer is organized as 32 bits wide and 4 words deep. Every new message has to be stored in the buffer word aligned on 4 byte words. See Section 66.2.1: Transmit data buffer on page 826.*

DSQ_TX_SYMB_PER Transmitter symbol period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SYMBOL_PER							

Address: *DiSEqCBaseAddress* + 0x018

Type: R/W

Reset: 0

Description: To ensure flexibility of variable symbol times this register is used to define the duration of the symbol in the PWK coded message.The value represents the symbol period in multiples of subcarrier cycles.

For a 22 kHz carrier and a symbol time of 1.5 µs, the register needs to be programmed with a value 33.This register need not be programmed for every message.

DSQ_TX_SYMBn_ONTIME Symbol n on-time period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SYMBOLn_ONTIME							

Address: *DiSEqCBaseAddress* + 0x01C (symbol 0), 0x020 (symbol 1)

Type: R/W

Reset: 0

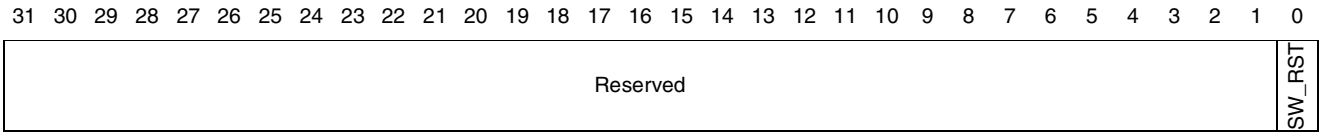
Description: This register is used to define the on-time period of symbol *n* in the PWK coded message.The value is represented in multiples of subcarrier periods. This register need not be programmed every message.

For a 22 kHz carrier and 0.5 µs of desired symbol *n* on time the register needs to be programmed with a value 11.

[31:8] **Reserved**

[7:0] **SYMBOLn_ONTIME**
Number of subcarrier cycles used to represent symbol *n* ON time where *n* = 1 or 2.

DSQ_TX_SW_RST Transmitter soft reset



Address: *DiSEqCBaseAddress* + 0x024

Type: R/W

Reset: 0

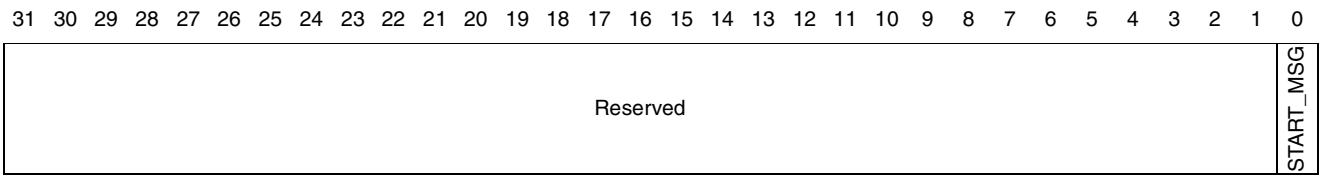
Description: This is a single bit register which initializes the contents of the data buffer without changing the transmitter configuration.The contents of the message config, silence period, symbol on times, prescaler and subcarrier registers are not cleared.

Writing 1 into the register enables soft reset and disable transmission. This bit is cleared to disable soft reset and restart transmission.

[31:1] **Reserved**

[0] **SW_RST**: Software reset
0: Inactive 1: Active

DSQ_TX_START Start message transmission



Address: *DiSEqCBaseAddress* + 0x028

Type: R/W

Reset: 0

Description: This is a 1-bit register which starts the transmission of the DiSEqC message. The transmission of the message begins only when bit 0 of this register is set.

Message transmission can start only when the transmitter is enabled (that is a value of 1 is written in the LSB of register DSQ_TX_EN).

Once set, the bit is cleared after the message transmission on the DiSEqC line has started.

DSQ_TX_INT_EN Transmitter interrupt enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									TX_BUFF_FULL	TX_BUFF_HFULL	TX_BUFF_EMPTY	TX_RDY	TX_UNDRUN_ERR	TX_SW_RST	TX_INT_EN

Address: *DiSEqCBaseAddress* + 0x02C

Type: R/W

Reset: 0

Description: To enable generation of interrupts, set the appropriate the interrupt enable bit.

[31: 7] **Reserved**

[6] **TX_BUFF_FULL**: Tx data buffer full interrupt
0: Disabled 1: Enabled

[5] **TX_BUFF_HFULL**: Tx data buffer half-full interrupt
0: Disabled 1: Enabled

[4] **TX_BUFF_EMPTY**: Tx data buffer empty interrupt
0: Disabled 1: Enabled

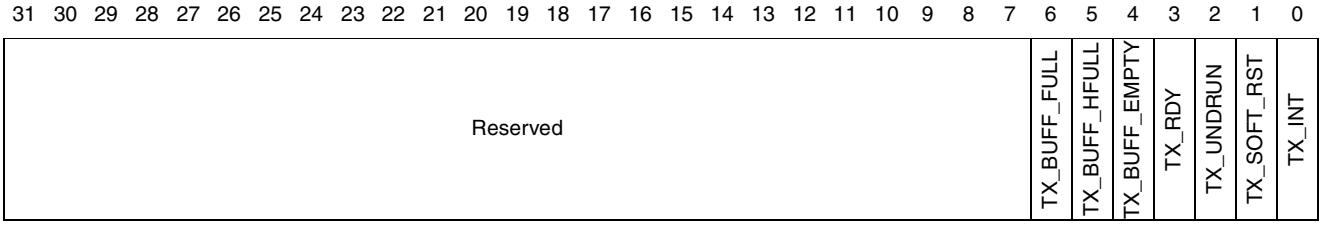
[3] **TX_RDY**: Transmitter ready interrupt
0: Disabled 1: Enabled

[2] **TX_UNDRUN_ERR**: Underrun error interrupt
0: Disabled 1: Enabled

[1] **TX_SW_RST**: Software reset
0: Disabled 1: Enabled

[0] **TX_INT_EN**: Global Tx interrupt
0: Disabled 1: Enabled

DSQ_TX_INT_STA Transmitter interrupt status



Address: *DiSEqCBaseAddress* + 0x030

Type: RO

Reset: 0

Description: This register gives the status of the interrupts for the transmitter and shows various states of the DSQ_TX_DATA_BUFF.

[31:7] **Reserved**

[6] **TX_BUFF_FULL**: Interrupt when Tx buffer is full

[5] **TX_BUFF_HFULL**: Interrupt when Tx buffer is at least half-empty

[4] **TX_BUFF_EMPTY**: Interrupt when Tx buffer is empty

[3] **TX_RDY**: Interrupt when the transmitter is ready sending message

[2] **TX_UNDRUN**: Interrupt due to Tx buffer underrun

[1] **TX_SW_RST**: Interrupt due to soft reset

[0] **TX_INT**: Interrupt pending

DSQ_TX_STA Transmitter status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						TX_FIFO_STA			TX_BUFF_FULL	TX_BUFF_HFULL	TX_BUFF_EMPTY	TX_RDY	TX_UNDER_RUN	TX_SW_RST	TX_INT_PEND

Address: *DiSEqCBaseAddress* + 0x034

Type: RO

Reset: 0 (except for TX_RDY which is 1)

Description: This register is used to depict the true status of the DiSEqC transmitter.The processor reads the transmitter's state from the corresponding bits in this register.The interrupt status can be cleared by writing into the clear interrupt location but the bits in the transmitter status register are cleared by the transmitter state alone.

The status of the last transmitted message can be inferred by reading the bits of the DSQ_TX_STA register.

[31:10] **Reserved**

[9:7] **TX_FIFO_STA**

000: FIFO full, not empty
001: One word empty in FIFO
010: Two words empty in FIFO
011: Three words empty in FIFO
100: Four words empty in FIFO

The Tx FIFO status bits depict the true status of the DSQ_TX_DATA_BUFF.Hence the bits are cleared based upon the status of the DSQ_TX_DATA_BUFF.

[6] **TX_BUFF_FULL**: Tx buffer is full

The Tx buffer full bit in the status register is set when the DSQ_TX_DATA_BUFF is full.This bit is cleared when there is less than four words left into DSQ_TX_DATA_BUFF.

[5] **TX_BUFF_HFULL**: Tx buffer is at least half-full

The Tx buffer half full bit in the status register is set when there are at least two words in the DSQ_TX_DATA_BUFF.This bit is cleared when there is at most one word left in DSQ_TX_DATA_BUFF.

[4] **TX_BUFF_EMPTY**: Tx buffer is empty. The Tx buffer empty bit in the status register is set at the beginning of transmission of the last byte of the last word in the DSQ_TX_DATA_BUFF. This bit is cleared when there is at least one word written into DSQ_TX_DATA_BUFF.

[3] **TX_RDY**: Transmitter is ready to send messages

The TX_RDY bit in the status register is set once the Tx is ready to send a new message.On reset this bit is set to one. This bit is cleared immediately after the first symbol of the programmed message is transmitted on the DiSEqC line.

The TX_RDY status indicates that the transmitter can be programmed for message transmission thus allowing the flexibility of programming even during the silence period to allow transmission of back to back messages.

[2] **TX_UNDRUN**: Tx buffer underrun has occurred

The TX_UNDERRUN bit in the status register is set once an underrun condition occurs. This bit is cleared by writing data inside the DSQ_TX_DATA_BUFF or by executing a soft reset operation that is writing 1 in the DSQ_TX_SW_RST register.

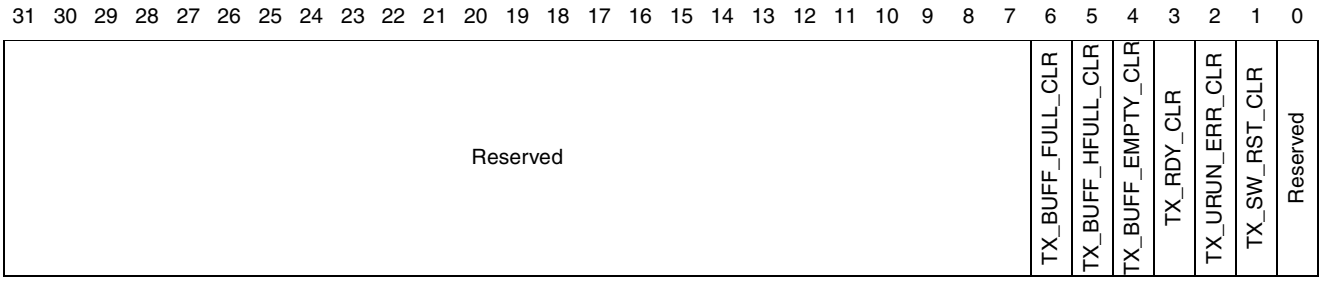
[1] **TX_SW_RST**: Transmitter has been soft reset

The TX_SOFT reset bit in the status register is set after the execution of the soft reset is complete.This bit can be cleared by writing a 0 in the DSQ_TX_SW_RST bit 0.

[0] **TX_INT_PEND**: Tx interrupt is pending for acknowledgement

The Tx interrupt pending status is set if any interrupt is pending in the DSQ_TX_INT_STA register. This bit is cleared when all the interrupt status bits in the DSQ_TX_INT_STA are cleared (serviced).

DSQ_TX_CLR_INT_STA Clear transmitter interrupt status



Address: *DiSEqCBaseAddress* + 0x038

Type: WO

Reset: 0

Description: The bits in the DSQ_TX_INT_STA register can be cleared by performing a clear bit operation at the corresponding bit locations.

Clearing the interrupt status bit in the DSQ_TX_INT_STA register does not alter the state of the DSQ_TX_STA register.The bits in the DSQ_TX_STA register are cleared only when the specific device condition causing the interrupt is removed.

[31:7] **Reserved**

[6] **TX_BUFF_FULL_CLR**
0: Tx buffer full interrupt not cleared
1: Tx buffer full interrupt cleared.

[5] **TX_BUFF_HFULL_CLR**
0: Tx buffer half full interrupt not cleared
1: Tx buffer half full interrupt cleared.

[4] **TX_BUFF_EMPTY_CLR**
0: Tx buffer empty interrupt not cleared
1: Tx buffer empty interrupt cleared

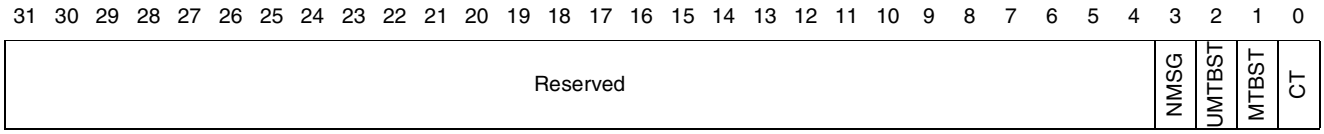
[3] **TX_RDY_CLR**
0: Tx message ready interrupt not cleared
1: Tx message ready interrupt cleared

[2] **TX_URUN_ERR_CLR**
0: Tx. under run error interrupt not cleared
1: Tx. under run error interrupt cleared

[1] **TX_SW_RST_CLR**
0: Soft reset interrupt not cleared
1: Soft reset interrupt cleared.

[0] **Reserved**

DSQ_TX_AGC Transmitter automatic gain control



Address: *DiSEqCBaseAddress* + 0x040

Type: R/W

Reset:

Description: Defines when the gain of the front end AGC has to be changed. These register bits are bristled at the output ports as defined in Section 8.16.8: *DiSEqC 2.0 on page 88*.

[31:4] **Reserved**

[3] **NMSG**: Normal message AGC
0: No change in normal message gain control
1: Change in normal message gain control

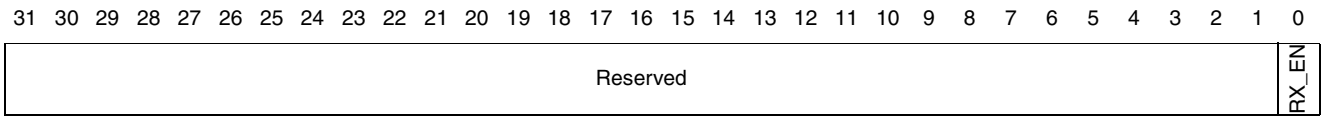
[2] **UMTBST**: Unmodulated tone burst AGC
0: No change in unmodulated tone burst gain control
1: Change in unmodulated tone burst gain control

[1] **MTBST**: Modulated tone burst AGC
0: No change in Mod. Tone Burst Gain Control
1: Change in Mod. Tone Burst Gain Control

[0] **CT**: Continuous tone AGC
0: No change in continuous tone gain control
1: Change in continuous tone gain control

67.2 Receiver

DSQ_RX_EN Receiver enable



Address: *DiSEqCBaseAddress* + 0x080

Type: R/W

Reset: 0

Description: Bit 0 is used to enable or disable the reception of DiSEqC messages. Writing 1 into the enable bit of this register enables reception. Writing 0 disables reception.

DSQ_RX_SAMPLING_PER Receiver sampling period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RX_SAMPLING_PER															

Address: *DiSEqCBaseAddress* + 0x084

Type: R/W

Reset: 0

Description: Selects the value for clock division to generate the sampling clock. To get a clock division by a value *n* the register must be loaded with the value *n*.

[31:16] **Reserved**

[15:0] **RX_SAMPLING_PER**
0: Ensures the clock division by 65535.

DSQ_RX_BYTE_CNT Receiver byte count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								RX_BYTE_CNT							

Address: *DiSEqCBaseAddress* + 0x088

Type: RO

Reset: 0

Description: Specifies the number of bytes received in the DiSEqC message. This is an 8-bit register so the maximum length of the received message supported is 256. When a message consisting of *n* bytes is received the register holds a value *n*.The byte count includes the framing byte also.

[31:8] **Reserved**

[7:0] **RX_BYTE_CNT**: Specifies the length of the received payload

DSQ_RX_SILENCE_PER Receiver silence period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RX_SILENCE_PER															

Address: *DiSEqCBaseAddress* + 0x08C

Type: R/W

Reset: 0

Description: This register specifies the time in number of sampling clock cycles to detect an end of a received message. If no activity occurs on the receive line during this time the receiver assumes an end of message has occurred.

[31:16] **Reserved**

[15:0] **RX_SILENCE_PER**: Specifies the silence period to detect an end of message.

DSQ_RX_DATA_BUFF Received data buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_PAYLOAD																															

Address: *DiSEqCBaseAddress* + 0x090
Type: RO
Buffer: Quadruple
Reset: 0
Description: Holds received payload information.

DSQ_RX_SYMBn_MIN_THOLD Receiver minimum threshold for symbol n on-time

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								MIN_SYMBn_TIME							

Address: *DiSEqCBaseAddress* + 0x094 (symbol 0), 0x09C (symbol 1)
Type: R/W
Reset: 0
Description: Defines the minimum threshold duration for determining symbol *n* on time in the received PWK coded message. The value is represented as multiples of sampling clock periods.

For a sampling clock of 660 kHz, the register needs to be programmed with a value 265 to achieve 0.4 µs min threshold time to detect a symbol *n*.

[31:8] **Reserved**

[7:0] **MIN_SYMBn_TIME**: Minimum threshold time for symbol *n*, where *n* = 0 or 1

DSQ_RX_SYMBn_MAX_THOLD Receiver maximum threshold for symbol n on-time

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								MAX_SYMBn_TIME							

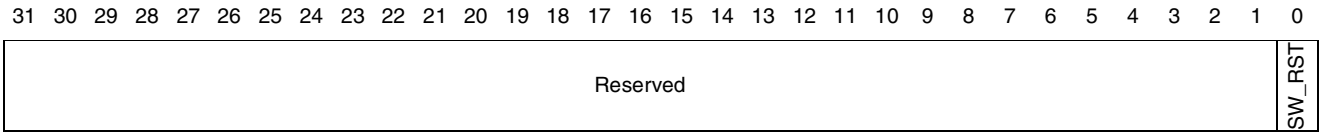
Address: *DiSEqCBaseAddress* + 0x098 (symbol 0), 0x0A0 (symbol 1)
Type: R/W
Reset: 0
Description: Defines the maximum threshold duration for determining symbol *n* on time in the received PWK coded message. The value is represented as multiples of sampling clock periods.

For a sampling clock of 660 kHz, the register needs to be programmed with a value 529 to achieve 0.8 µs max threshold time to detect a symbol *n*.

[31:8] **Reserved**

[7:0] **MAX_SYMBn_TIME**: Maximum threshold time for symbol *n*, where *n* = 0 or 1

DSQ_RX_SW_RST Receiver soft reset



Address: *DiSEqCBaseAddress* + 0x0A4

Type: R/W

Reset: 0

Description: This is a single bit register. This is used to clear the receive data buffer without changing the receiver configuration. The contents of the receiver message configuration registers, receiver sampling period register, silence period register, min and max threshold times for symbol 0 and symbol 1 register values are not cleared. Writing a 1 into this register enables soft reset.

Enabling soft reset disables reception. To restart reception, the soft reset bit must be cleared by writing 0 to SW_RST.

DSQ_RX_INT_EN Receiver interrupt enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SYMBOL_ERR	PARITY_ERR	CODE_ERR	LAST_BYTE_REC	RX_BUFF_FULL	RX_BUFF_HFULL	RX_OVF_ERR	RX_INT_EN

Address: *DiSEqCBaseAddress* + 0x0A8

Type: R/W

Reset: 0

To enable generation of interrupts the corresponding interrupt enable bit in this register must be set. An interrupt is generated if an interrupt condition occurs and the interrupt enable bit is set. Refer to Section 66.6.1: *Normal message reception* *on page 831* for descriptions of code errors, parity errors and symbol errors.

[31:8] **Reserved**

[7] **SYMBOL_ERR**: Symbol error interrupt
0: Disabled 1: Enabled

[6] **PARITY_ERR**: Parity error interrupt
0: Disabled 1: Enabled

[5] **CODE_ERR**: Code error interrupt
0: Disabled 1: Enabled

[4] **LAST_BYTE_REC**: Last byte received interrupt
0: Disabled 1: Enabled

[3] **RX_BUFF_FULL**: Rx buffer full interrupt
0: Disabled 1: Enabled

[2] **RX_BUFF_HFULL**: Rx buffer half full interrupt
0: Disabled 1: Enabled

[1] **RX_OVF_ERR**: Receive overflow error interrupt
0: Disabled 1: Enabled

[0] **RX_INT_EN**: Receive interrupt
0: Disabled 1: Enabled

DSQ_RX_INT_STA Receiver interrupt status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SYMBOL_ERR	PARITY_ERR	CODE_ERR	LAST_BYTE_RECEIVED	RX_BUFF_FULL	RX_BUFF_HFULL	RX_OVF	RX_INT

Address: *DiSEqCBaseAddress* + 0x0AC

Type: RO

Reset: 0

Description: Depicts the status of the interrupts for the receiver, various conditions of the DSQ_RX_DATA_BUFF and interrupt status due to the error conditions detected on the received DiSEqC message.

[31:8] **Reserved**

[7] **SYMBOL_ERR**: Interrupt due to symbol error in received message

[6] **PARITY_ERR**: Interrupt due to parity error in received message

[5] **CODE_ERR**: Interrupt due to code error in received message

[4] **LAST_BYTE_RECEIVED**: Last message byte receive interrupt

[3] **RX_BUFF_FULL**: Rx buffer full interrupt

[2] **RX_BUFF_HFULL**: Rx buffer half-full interrupt

[1] **RX_OVF**: Rx overflow interrupt

[0] **RX_INT**: Rx interrupt active
1: Interrupt active 0: Interrupt not active

DSQ_RX_STA Receiver status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				RX_FIFIO_STA			RX_BUSY	SYMBOL_ERR	PARITY_ERR	CODE_ERR	LAST_BYTE_RECEIVED	RX_BUFF_FULL	RX_BUFF_HFULL	RX_OVF	RX_INT_PEND

Address: *DiSEqCBaseAddress* + 0x0B0

Type: RO

Reset: 0

Description: This register is depicts the true status of the DiSEqC receiver. The processor can read the receiver's state from this register. The bits in this register are set when the receiver starts receiving the message and are updated when the reception of the next message starts. On the start of reception of a new message status bit 8 is set to indicate that the receiver is busy.

The status of the last received message can be inferred by reading the bits of the DSQ_RX_STA register.

[31:10] **Reserved**

[11:9] **RX_FIFO_STA**: Receiver FIFO status

000: FIFO empty, not full
001: One word in FIFO
010: Two words in FIFO
011: Three words in FIFO
100: Four words in FIFO

The Rx FIFO status bits depict the true status of the DSQ_RX_DATA_BUFF. Hence the bits are cleared based upon the status of the DSQ_RX_DATA_BUFF.

[8] **RX_BUSY**: Receiver busy

0: Receiver is idle
1: Receiver is busy and receiving message

The RX_BUSY bit is set once the receiver starts receiving a new message. This bit is cleared when the last symbol of the current message is detected on the DiSEqC line.

[7] **SYMBOL_ERR**: Symbol error

0: No symbol error
1: Symbol error detected in the received payload

[6] **PARITY_ERR**: Parity error

0: No parity error
1: Parity error detected in the received payload

[5] .**CODE_ERR**: Code error

0: No code error
1: Code error detected in the received payload

[4] **LAST_BYTE_RECEIVED**: Last byte received

0: End of message has not occurred
1: End of message has occurred and last byte received

[3] **RX_BUFF_FULL**: Receive buffer full

0: Rx buffer is more than half empty
1: Rx buffer is half full

The Rx buffer full bit in the status register is set at the beginning of reception of the last byte of the last word in the DSQ_RX_DATA_BUFF.This bit is cleared when at least one word is read from the DSQ_RX_DATA_BUFF.

[2] **RX_BUFF_HFULL**: Receive buffer half-full

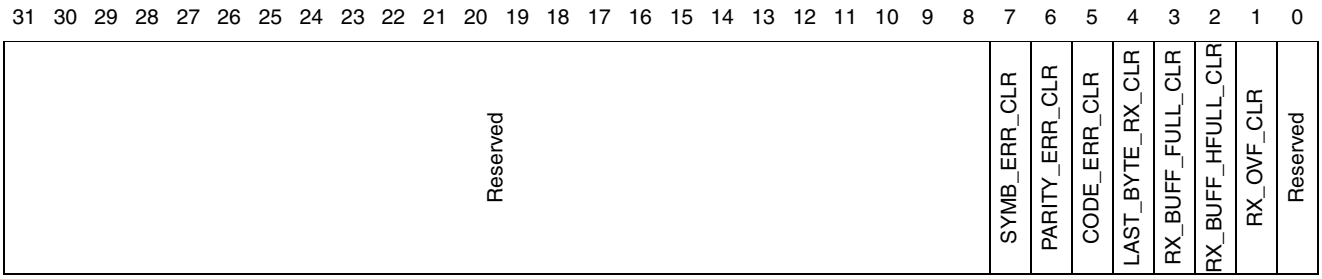
0: At least one word can be received
1: Rx buffer is full.

The Rx buffer half full bit in the status register is set when there are at least two words in the DSQ_RX_DATA_BUFF.This bit is cleared when there is at least one word remaining in the DSQ_RX_DATA_BUFF.

[1] **RX_OVF**: Receiver buffer overflow
0: Rx buffer overflow has not occurred 1: Rx buffer overflow has occurred
The Rx overflow bit in the status register is set once an overflow condition occurs. This bit is cleared by reading data from the DSQ_RX_DATA_BUFF or by executing a soft reset, that is writing 1 in the DSQ_RX_SW_RST register.

[0] **RX_INT_PEND**: Receiver interrupt pending
0: Interrupt not pending 1: Interrupt pending
The Rx interrupt pending status is set if any interrupt is pending in the DSQ_RX_INT_STA register. This bit is cleared when all the interrupt status bits in the DSQ_RX_INT_STA register are cleared (serviced).

DSQ_RX_CLR_INT_STA Clear receiver interrupt status



Address: *DiSEqCBaseAddress* + 0x0B4
Type: WO
Reset: 0
Description: The bits in the DSQ_RX_INT_STA register can be cleared by performing a clear bit operation at appropriate locations.To clear a corresponding interrupt a clear bit operation has to be performed at the corresponding bit location.

Clearing the interrupt through DSQ_RX_CLR_INT_STA (clear interrupt status location) only clears the corresponding interrupt status bit in the DSQ_RX_INT_STA register. The bits in the DSQ_RX_STA register are cleared only when the device condition causing the interrupt is removed.

[31:8] **Reserved**

[7] **SYMB_ERR_CLR:** Symbol error interrupt clear

[6] **PARITY_ERR_CLR:** Parity error interrupt clear

[5] **CODE_ERR_CLR:** Code error interrupt clear

[4] **LAST_BYTE_RX_CLR:** Last byte received interrupt clear

[3] **RX_BUFF_FULL_CLR:** Rx buffer full interrupt clear

[2] **RX_BUFF_HFULL_CLR:** Rx buffer half full interrupt clear

[1] **RX_OVF_CLR:** Rx overflow interrupt error clear

[0] **Reserved**

DSQ_RX_TIMEOUT Receiver timeout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TIME_OUT_PER															

Address: *DiSEqCBaseAddress* + 0x0B8

Type: R/W

Reset: 0

Description: Sets the maximum time interval which the receiver may consider a valid bit separation time in a message.

The value programmed is an integer multiple of sampling cycles.

DSQ_RX_NSUPP_WID Receiver noise suppression width

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																NOISE_SUPP_WID															

Address: *DiSEqCBaseAddress* + 0x0BC

Type: R/W

Reset: 0

Description: This register programs the maximum width of noise pulses which are to be suppressed. To suppress a noise pulse of width *n* sampling clock periods, this register is programmed with the value *n*.

[31:16] **Reserved**

[15:0] **NOISE_SUPP_WID**: Maximum width of noise pulses to be suppressed
Programming a value 0x0000 in the noise suppression width disables noise suppression.

DSQ_POL_INV Receiver polarity invert

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															POL_INV

Address: *DiSEqCBaseAddress* + 0x0C0

Type: R/W

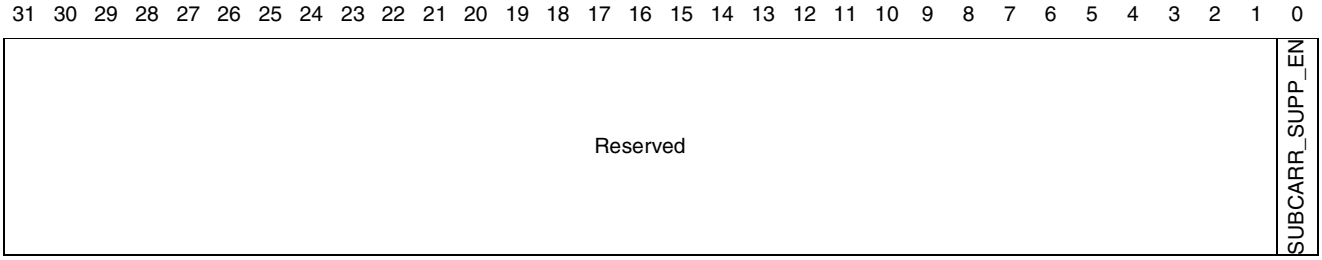
Reset: 0

Description: This register inverts the polarity of the bits in the received DiSEqC frame.

[31:1] **Reserved**

[0] **POL_INV**
0: No polarity inversion 1: Polarity inversion

DSQ_RX_SUBCARR_NSUPP_EN Receiver subcarrier noise suppression enable



Address: *DiSEqCBaseAddress* + 0x0C4

Type: R/W

Reset: 0

Description: This register allows programmable suppression of the subcarrier on the received envelope of the DiSEqC signal. Setting a value of 1 allows suppression of the subcarrier external to the DiSEqC module.

Setting a value of 0 disables the external suppression of the subcarrier. This register programs the maximum width of noise pulses to be suppressed. To suppress a noise pulse of width *n* sampling clock periods, this register is programmed with the value *n*.

[31:1] **Reserved**

[0] **SUBCARR_SUPP_EN**

0: Subcarrier suppression disabled

1: Subcarrier suppression enabled

68 High-bandwidth digital content protection (HDCP)

68.1 Functional description

The HDCP cipher is used in various ways through software, to provide key and messaging information which is used to authenticate an HDMI-compliant video channel. Additionally, signals from the video transmitter instruct the HDCP when to generate encryption keys, which are used by the transmitter to encrypt outgoing video data.

Device keys are stored in local RAM.

The HDCP cipher logic itself is wrapped up by a control state machine.

68.1.1 Video receiver authentication

A simple set of commands are defined, which are all that is required by software to generate the necessary messaging information for channel authentication. The software is responsible for ordering the commands; no hardware ordering checks are performed. The software must ensure that commands are only issued after the AUTHENTICATED bit in HDCP_CTRL has been deasserted.

Commands are issued from software by writing to the HDCP_AUTH_CMD register.

Command summary

Table 210: Command summary

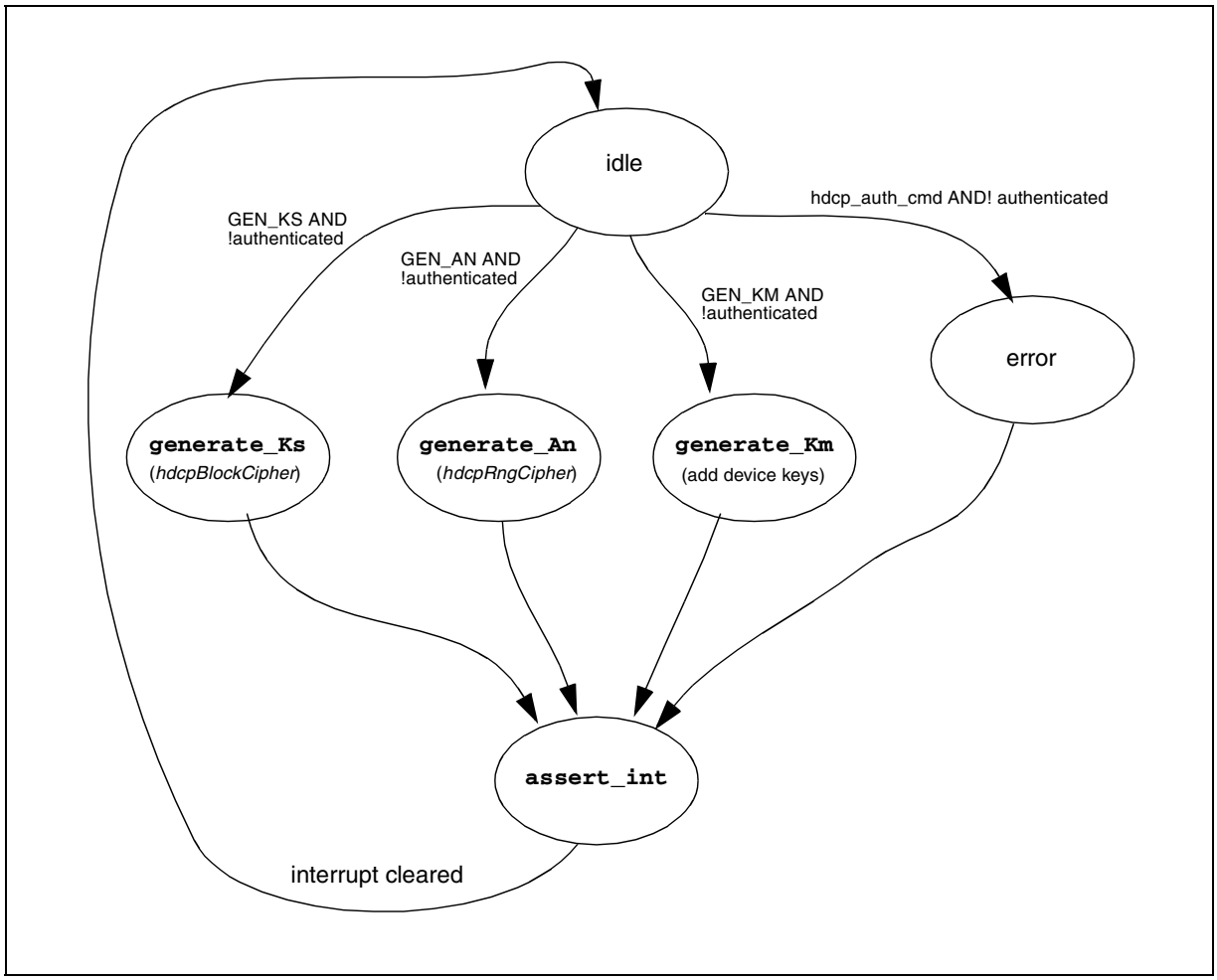
Code	Command	Function
0x01	**generate_An**	Perform **hdcpRngCipher** function to generate 64-bit pseudo-random value, *An*, and store in the AN register.
0x02	**generate_Km**	Using 40-bit key selection vector in KSV register, calculate shared secret, *Km*, from locally stored device keys and store 56-bit result internally. *Note: Device keys should be written in encrypted format; after writing an IV value to seed the internal on-the-fly decryption of these keys. Km is not readable by software.*
0x03/0x83	**generate_Ks**	perform **hdcpBlockCipher**, using *Km* and *RPTI*/*An* from previous commands to calculate the session key, Ks (and also M_0 and R_0, into data register). *Note: Here, bit 7 of HDCP_AUTH_CMD corresponds to the repeater bit*

Whenever a command has completed successfully, O_INT is asserted and HDCP_AUTH_CMD_DONE is set in HDCP_STA. Upon completion, any command that expects a result output will find it by reading the appropriate register. The status bit and the interrupt are cleared by writing 1 to HDCP_AUTH_CMD_DONE.

If any command is issued when bit HDCP_CTRL.AUTHENTICATED is set, bit HDCP_STA.HDCP_AUTH_CMD_ERR is set, and the interrupt is asserted. The status bit and the interrupt are cleared by writing 1 to HDCP_AUTH_CMD_ERR.

Figure 260 shows the modes of operation used by the HDCP Cipher hardware when carrying out authentication commands.

Figure 260: HDCP authentication command flow abstraction



68.1.2 Video encryption key generation and control

The HDCP block is required to generate a unique key at intervals specified by control signals from the HDMI, in order to be combined with outgoing data on the transmitter interface. Only four operations are ever used by the HDCP cipher: **`hdcpBlockCipher`**, **`hdcpStreamCipher`**, **`hdcpReKeyCipher`** and **`hdcpRngCipher`**.

Figure 262: HDCP data encryption flow abstraction shows the real-time flow taken when encrypting data on the interface.

Note: 1 I_VSYNC *is considered active on its rising edge.*

2 Frame key calculation does not begin until 528 cycles after the rising edge of I_VSYNC*.*

3 The HDCP will only examine the contents of the ENC_EN *and* AV_MUTE *bits of the* HDCP_CTRL *register 528 cycles after the rising edge of* I_VSYNC*.*

4 I_DATA_ISLAND_ACTIVE *and* I_VIDEO_DATA_ACTIVE *must not be asserted until at least 108 cycles after commencement of frame key calculation (frame key calculation is started by assertion of* V_SYNC *+ 528 cycles).*

5 After I_VIDEO_DATA_ACTIVE *has been deasserted,* I_DATA_ISLAND_ACTIVE *must not be asserted for at least 57 cycles (during which line key is calculated).*

6 If both I_VSYNC *and* I_HSYNC*, are asserted simultaneously,* I_VSYNC *takes priority.*

68.1.2.1 O_ENC_EN

This signal is asserted by HDCP when it is authenticated, the ENC_EN register is set, and the AV_MUTE register is not set.

68.1.2.2 O_RX_PIXEL_CAPTURE

Ri is updated every 128th frame, starting with the 128th frame, unless specified otherwise by the I_RATE register. Frames are counted as those for which encryption is **enabled**, **unless in advance cipher mode** (indicated by the assertion of bit HDCP_CTRL.AC_MODE) when all frames are counted. The O_RX_PIXEL_CAPTURE output is asserted for one frame when the value of *Ri* has been updated (this occurs 108 TMDS cycles after the rising edge of V_SYNC, and remains asserted until the end of the frame). This allows the HDMI to update its value of *Ri* as indicated by *HDCP Specification Rev 1.1, Digital Content Protection LLC*, section *Enhanced Ri Computation Mode*.

68.2 Software usage

Software drivers should perform the following functions in order to program and use the HDCP Block. See also Figure 261: Software flow.

1. **Generate a pseudo-random value**
   After reset, to begin authentication, calculate a value for *An*. This is done by issuing **`hdcp_cmd`** 0x01 and awaiting the interrupt to indicate completion.
2. **Device key encryption**
   The algorithm for encrypting the device private keys is delivered separately. Note that the IV value seeds both encryption and decryption algorithms, and is a random constant for a given device-key set.
3. **Load device keys**
   Write in the 56-bit IV value in order to seed the decryption of the device keys. Write in the 40 56-bit device keys into the block at the appropriate addresses.

*Note: 1 The order of loading **must** be observed if correct decryption of these keys is to occur: they must be loaded in starting at key 0, low word then high word, up to key 39.*

4. **Generate the shared secret value**
   Calculate *Km* value from previously loaded device keys. This is done by firstly loading the *Ksv* value into the appropriate register and then issuing **`hdcp_cmd`** 0x02 and awaiting the interrupt to indicate completion.
5. **Generate a session key**
   Calculate *Ks* (and *M* and *R*) values by issuing **`hdcp_cmd`** 0x03/0x83 (depending on whether the receiver is a repeater) and awaiting the interrupt to indicate completion.

At this point the software **must** set bit HDCP_CTRL.AUTHENTICATED. This initiates the generation of the first frame key. The block is now ready to start output of encryption keys, and will generate *new_Line* and *frame* keys according to the control signals from the HDMI.

By default, *R* is updated every 128 frames. If a more frequent update is required, software should set the required update time in HDCP_I_RATE register at a safe juncture (for example upon re-authentication).

If at any time software wishes to temporarily disable encryption, it may do so (starting with the next frame) by disabling the ENC_EN register (or AV_MUTE if in EESS mode).

If at any time software wishes to de-authenticate abruptly (for example on loss of sync on the HDMI interface), or restart a failed authentication mid-way through, then it must unset bit HDCP_CTRL.AUTHENTICATED, and then restart the authentication process.

Figure 261: Software flow

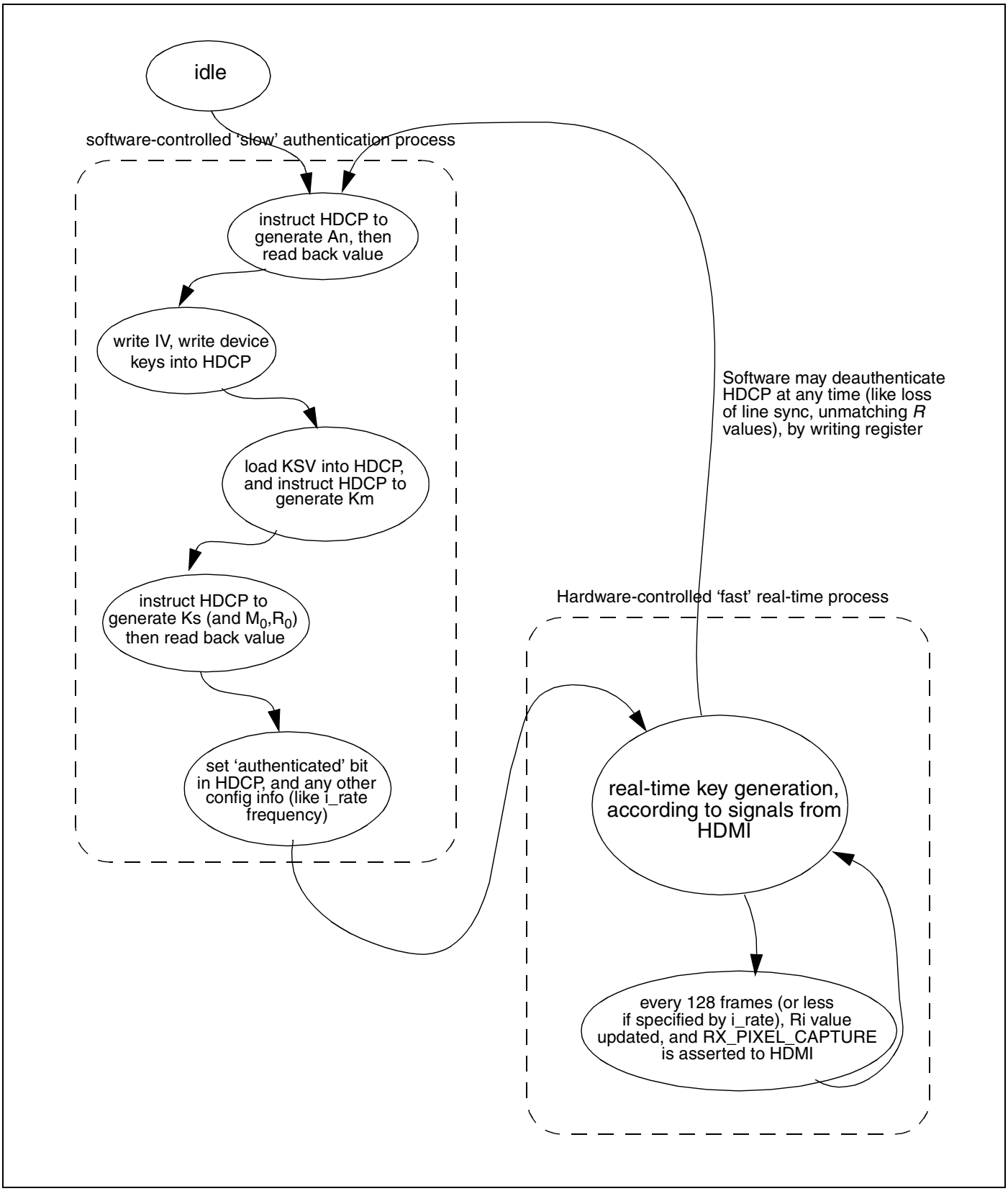
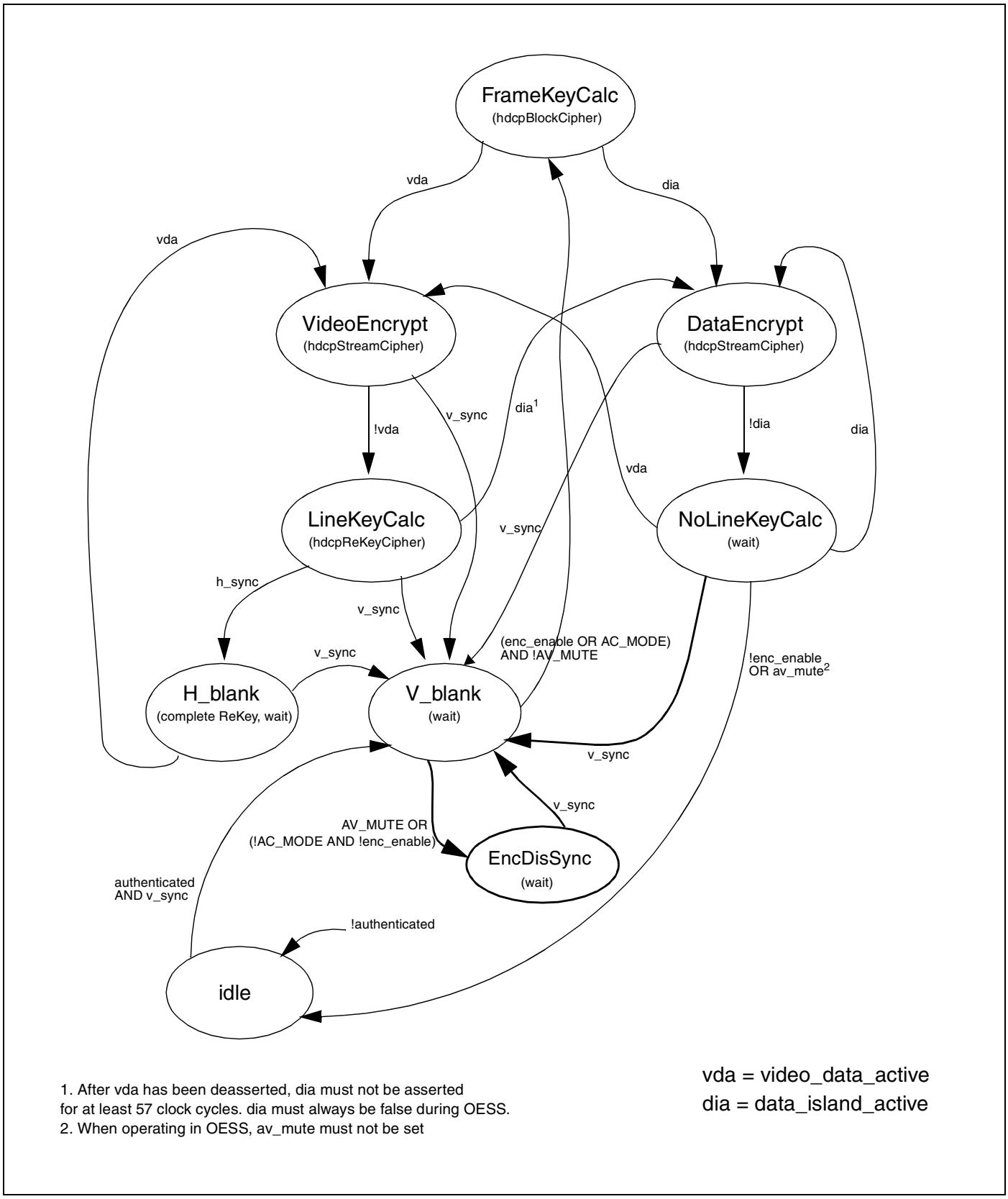


Figure 262: HDCP data encryption flow abstraction



69 HDCP registers

Register addresses are provided as *HDCPBaseAddress* + offset.

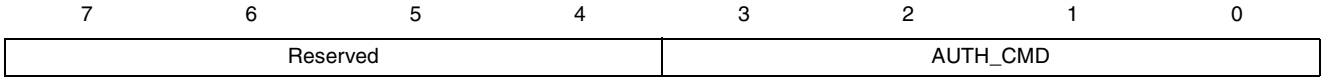
The *HDCPBaseAddress* is:

0x1900 5400.

Table 211: HDCP register summary

Register	Description	Offset	Type
HDCP_AUTH_CMD	Authorization command	0x0000	R/W
HDCP_CTRL	Real time control	0x0004	R/W
HDCP_I_RATE	I_rate	0x0008	R/W
HDCP_STA	Status	0x000C	R/W
HDCP_KSV/IV_0	Shared secret/Initialization vector LSB	0x0010	R/W
HDCP_KSV/IV_1	Shared secret/Initialization vector MSB	0x0014	R/W
HDCP_AN_0	Pseudo-random value LSB	0x0018	RO
HDCP_AN_1	Pseudo-random value MSB	0x001C	RO
HDCP_KS_0	Session key LSB	0x0020	RO
HDCP_KS_1	Session key MSB	0x0024	RO
HDCP_MI_0	Mi LSB	0x0028	RO
HDCP_MI_1	Mi MSB	0x002C	RO
HDCP_RI	16-bit response value	0x0030	RO
Reserved		0x0034 - 003C	-
DEVKEY_*n*_LO_WORD	Devkey_0[31:0]	0x0040+0x8**n*	
DEVKEY_*n*_HI_WORD	Devkey_0[55:32], mapped to [23:0] of register	0x0044+0x8**n*	
...	...	...	-
DEVKEY_39_LO_WORD	Devkey_39[31:0]	0x0178	
DEVKEY_39_HI_WORD	Devkey_39[55:32], mapped to [23:0] of register	0x017C	
Reserved		0x0180 - 01FF	-

HDCP_AUTH_CMD Authorization command

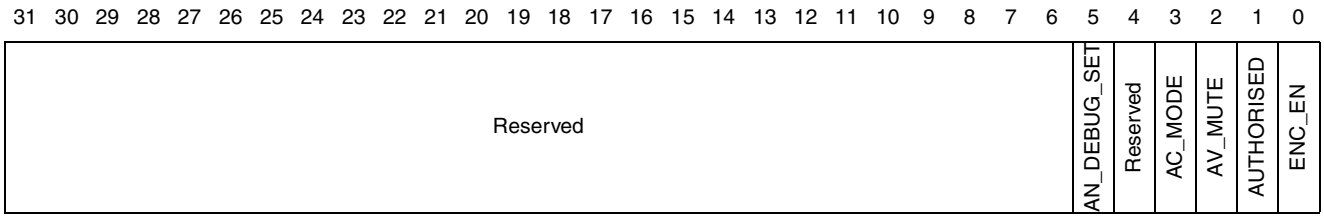


Address: *HDCPBaseAddress* + 0x0000
Type: R/W
Reset: 0
Description:

[31:4] **Reserved**

[3:0] **AUTH_CMD**
0x01: **`generate_An`**
0x02: **`generate_Km`**
0x03/0x83: **`generate_Ks`** (Bit 7 indicates repeater)
Other: Reserved

HDCP_CTRL Real time control



Address: *HDCPBaseAddress* + 0x0004
Type: R/W
Reset: 0
Description:

[31:3] **Reserved**

[5] **AN_DEBUG_SET**
Gated with CFG_DEBUG_ENABLE antifuse.
1: The value of An written into the An_0/An_1 registers is used instead of the pseudo-randomly-generated one.

[4] **Reserved**

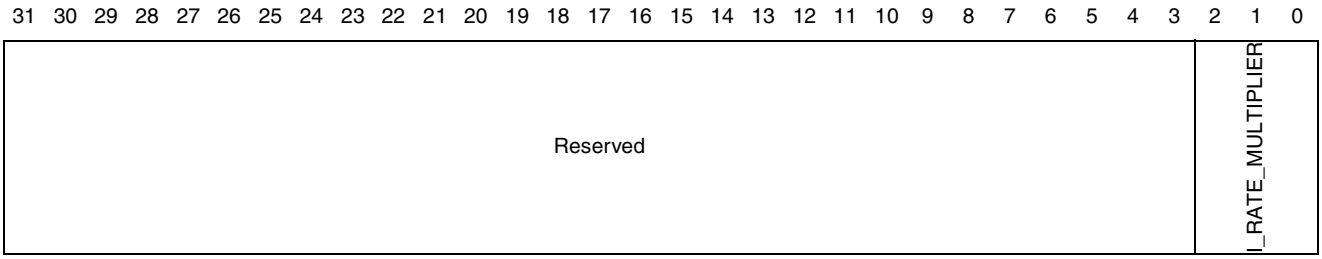
[3] **AC_MODE**
1: Enable advance cipher mode

[2] **AV_MUTE**: Encryption
0: Enable 1: Temporarily disable

[1] **AUTHORISED**
0: No (disable encryption) 1: Yes (enable encryption)

[0] **ENC_EN**
0: Disable 1: Enable

HDCP_I_RATE I rate multiplier



Address: *HDCPBaseAddress* + 0x0008

Type: R/W

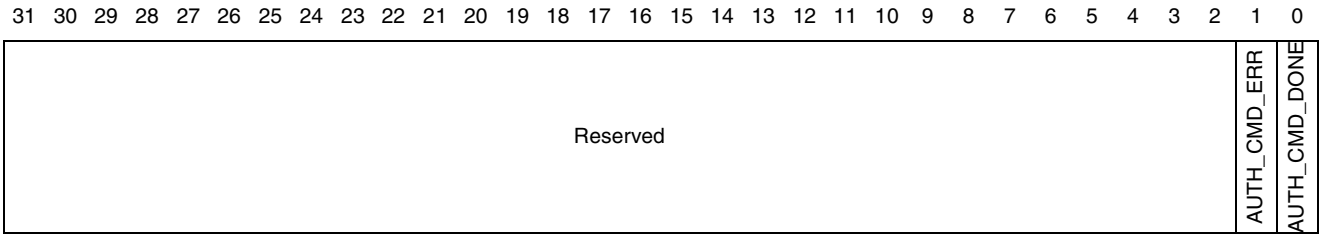
Reset: 0

Description:

[31:3] **Reserved**

[2:0] **I_RATE_MULTIPLIER**
1: Update RI every '16 x I_RATE_MULTIPLIER' encrypted frames, starting with the '16 x I_RATE_MULTIPLIER'th encrypted frame.
0: Default to every 128th encrypted frame.

HDCP_STA Status



Address: *HDCPBaseAddress* + 0x000C

Type: R/W

Reset: 0

Description:

[31:2] **Reserved**

[1] **AUTH_CMD_ERR**: Authorization CMD error
1: Indicates previous **`hdcp_cmd`** has not completed. Once set, this bit and the interrupt are cleared when overwritten with the same value (Logic1).
0: No error state.

[0] **AUTH_CMD_DONE**: Authorization CMD error
1: Indicates previous **`hdcp_cmd`** has completed successfully. Once set, this bit and the interrupt are cleared when overwritten with the same value (Logic1). 0: previous **`hdcp_cmd`** not completed

HDCP_IVKSV_n Initialization vector/key selection vector

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
IV_0	IV[31:0]			
IV_1	Reserved	IV[55:32]		
KSV_0	KSV[31:0]			
KSV_1	Reserved			KSV[39:32]

Address: *HDCPBaseAddress* + 0x0010 (IVKSV_0); + 0x0014 (IVKSV_1)
Type: R/W
Reset: 0
Description:

IV: Before writing device keys, write the initialization value IV, used to seed decryption of the device keys.

IV_0: [31:1] **IV[31:0]**

IV_1: [31:24] **Reserved**

[23:0] **IV[55:32]**

KSV: After device key loading, write the KSV value.

KSV_0: [31:0] **KSV[31:0]**

KSV_1: [31:8] **Reserved**

[7:0] **KSV[55:32]**

HDCP_AN_n Pseudo-random value

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
AN_0	AN[31:0]			
AN_1	AN[63:32]			

Address: *HDCPBaseAddress* + 0x0018 (AN_0); + 0x001C (AN_1)
Type: RO
Reset: 0
Description: 64-bit pseudo-random value

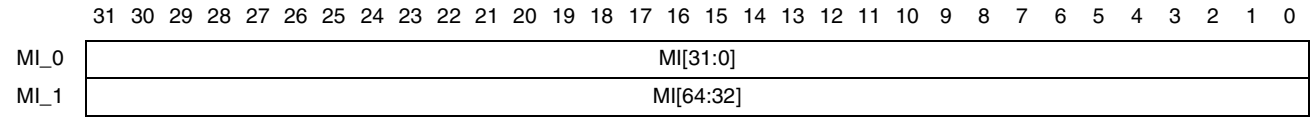
HDCP_KS_n Session key

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
KS_0	KS[31:0]			
KS_1	Reserved	KS[55:32]		

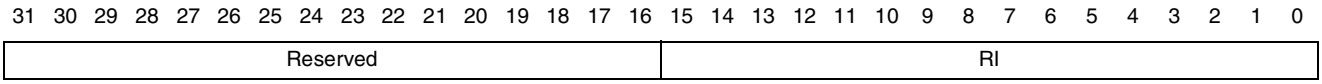
Address: *HDCPBaseAddress* + 0x0020 (KS_0); + 0x0024 (KS_1)
Type: RO
Reset: 0
Description: Session key

HDCP_MI Mi

Address: *HDCPBaseAddress* + 0x0028 (MI_0); + 0x002C (MI_1)
Type: RO
Reset: 0
Description:



HDCP_RI 16-bit response



Address: *HDCPBaseAddress* + 0x0030
Type: RO
Reset: 0
Description: 16-bit response

70 Infrared transmitter/receiver

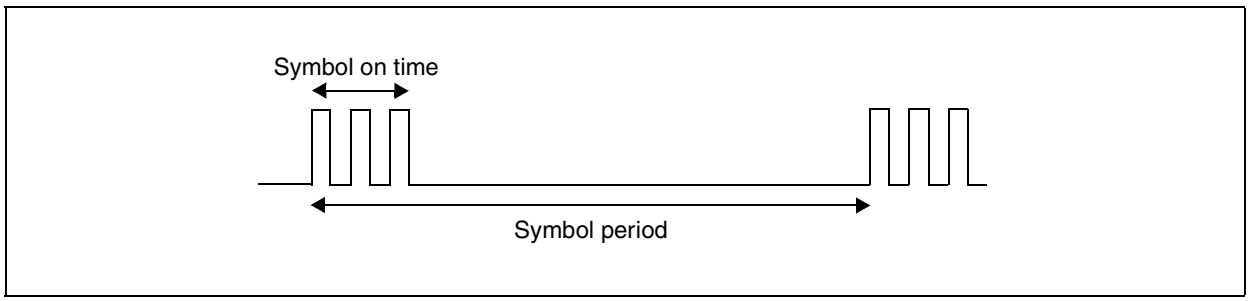
70.1 Overview

The infrared (IR) transmitter/receiver is an ST40 peripheral. For each symbol transmitted, the software driver determines the symbol period and the symbol on time of the IR pulse, and transfers these parameters into a eight-word deep FIFO. The IR transmitter/receiver then generates coded symbols using an internally generated subcarrier clock.

The parameters symbol period and symbol on time are illustrated in Figure 263.

The incoming signal must be detected, and the subcarrier must be suppressed, externally. Only the symbol envelope can be used by the IR and UHF processors. It is sampled at 10 MHz and the sample values are transferred into the input buffer in microseconds.

Figure 263: IR transmitter/receiver symbol



70.2 Functional description

The IR transmitter/receiver transmits infrared data and receives both IR and UHF data. The IR and UHF receivers are independent and identical, except that the IR receiver does not use the noise filter. Both receivers are simultaneously active. The IR transmitter/receiver supports RC (remote control) codes only.

Figure 264 shows the IR transmitter/receiver block diagram in a typical circuit configuration with input demodulating and output buffering (open drain).

In the transmitter there are two programmable dividers to generate the prescaled clock and the subcarrier clock. The subcarrier clock sets the resolution for the transmitted data. Both receivers contain a sampling rate clock, which samples the incoming data, and is programmed to 10 MHz.

FIFOs buffer both the transmitter output and the receivers' inputs to avoid timing problems with the CPU. Interrupts can be set on the FIFOs' levels to prevent input data overrun and output data underrun.

The two receivers each have one input pin (RC_IRDA_DATA_IN and IRDA_ASC_DATA_IN), and the transmitter has two output pins (RC_IRDA_DATA_OUT driven directly and IRDA_ASC_DATA_OUT inverted as open drain).

There are six 8-word FIFOs: two in the RC transmitter and two in each RC receiver. The eighth word in each FIFO is used internally and is not accessible. Therefore a FIFO is empty when there are seven empty words and full when it contains seven words. At all times, the fullness level of the FIFO is given in its corresponding status register.

Each submodule pair of FIFOs, for symbol period and symbol on time, should be treated as a set and must be consecutively accessed for read or for write. They share a common pointer which is incremented only when they have been accessed correctly. Repeated reads on one FIFO always give the same data, and repeated writes always overwrite the previous data.

Figure 264 shows the complete system, and Figure 265, the receiver subsystem.

Figure 264: IR/UHF transmitter/receiver block diagram and implementation

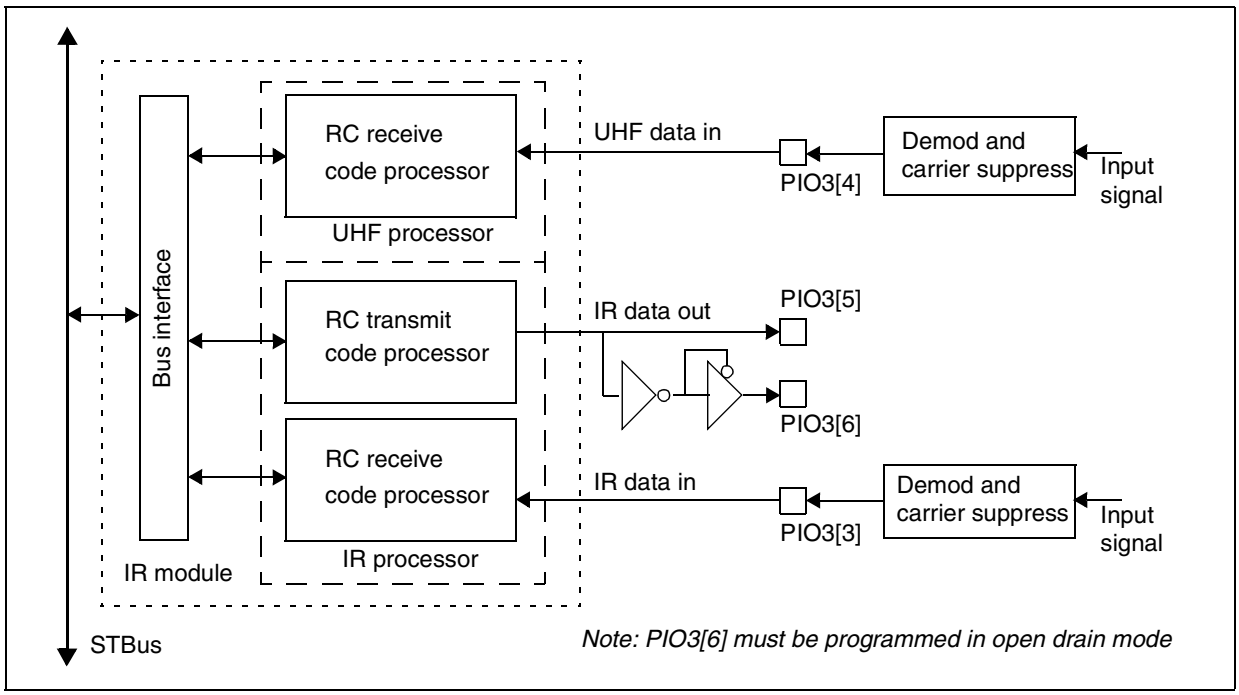
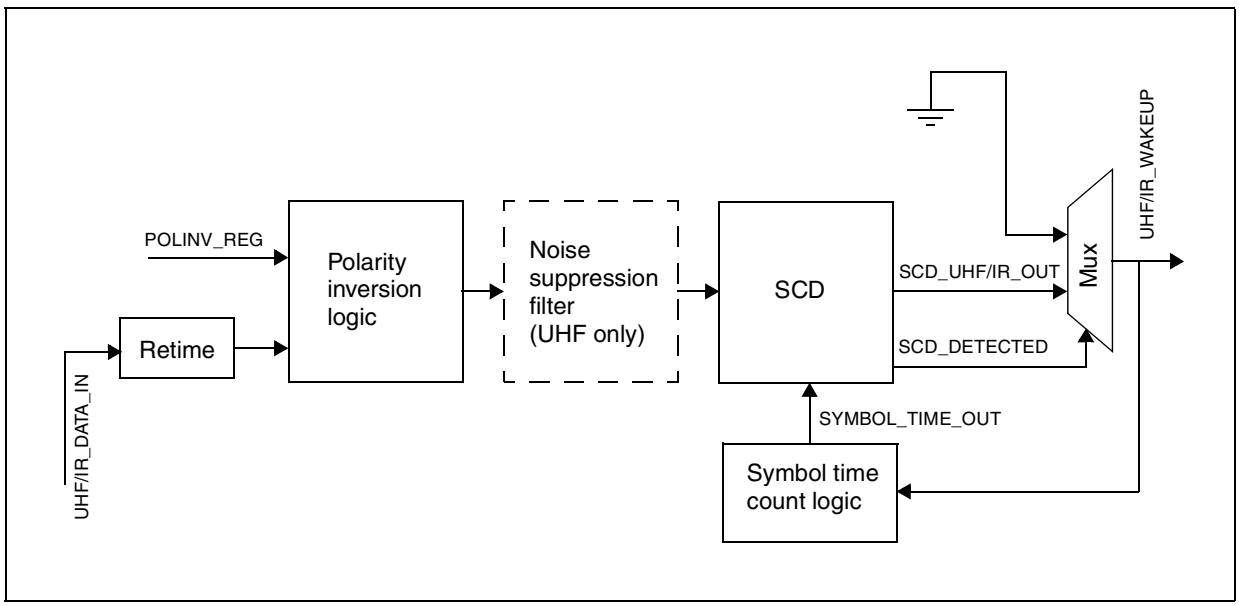


Figure 265: IR/UHF receiver block diagram



70.2.1 RC transmit code processor

RC codes are generated by programming the transmit frequency and writing the symbol information into a FIFO, which is then read internally and the data processed to provide a serial PWM data stream. The transmit interrupt is set on a preselected FIFO level. An interrupt and a flag in the status register indicate an underrun condition (that is, an empty FIFO). RC data transmission is disabled by setting bit IRB_TX_EN_IR.TX_EN to 0.

The transmit interrupt is set by register IRB_TX_INT_EN, on one of three FIFO levels:

- when seven words are empty (buffer is empty),
- when four words are empty (buffer is half full),
- when at least one word is empty.

The transmit interrupt is cleared automatically when new data is written to registers IRB_TX_SYMB_PER and IRB_TX_ON_TIME. Register bits IRB_TX_INT_STA[4:1] give the FIFO's fullness status.

The frequency of the subcarrier is set by programming registers IRB_TX_PRESCALER and IRB_TX_SUBCARR.

The symbol period, in subcarrier cycles, is programmed in register IRB_TX_SYMB_PER and the on time of the IR pulse is written to register IRB_TX_ON_TIME. These two registers are eight-word FIFOs. They must be programmed sequentially as a pair to increment the write pointer and be ready for the next data. Transmission is enabled by setting bit IRB_TX_EN.TX_EN to 1. If new data is not written before the last symbol in the buffer is transmitted, no RC codes are generated. The output is driven to logic 0 and bit IRB_TX_INT_STA.UNDERRUN is set.

Before data can be transmitted, the underrun condition must be cleared as follows:

1. Disable the transmission by writing 0 to register IRB_TX_IR_EN.
2. Load at least one block of data into IRB_TX_SYMB_PER_IR and IRB_TX_ON_TIME_IR.
3. Clear the TX_UNDERRUN status bit by writing 1 to register IRB_TX_CLR_UNDERRUN_IR.

Transmission is resumed by writing 1 to register IRB_TX_IR_EN.

70.2.2 RC receive code processor

This section describes the UHF data and the IR data receivers. They are independent and identical except that the noise suppression filter is programmable in the UHF receiver, and is not used in the IR receiver. The 10 MHz sampling clock is common to both receivers and is set by register IRB_SAMPLE_RATE_COMM. This register is programmed with the value 10 for a 100 MHz infrared transmitter/receiver system clock.

Each receiver processes the incoming RC code symbol envelope and stores the values symbol period and symbol on time (in µs) in a eight-word FIFO buffer, until the data can be read by the microcontroller.

The receive interrupt is set by register IRB_RX_INT_EN to one of the following three FIFO levels:

- at least one word is available to be read,
- four or more words are available to be read (FIFO half full),
- seven words are available to be read (FIFO full).

The interrupt is cleared when registers IRB_RX_SYMB_PER and IRB_RX_ON_TIME have been read. They must be read consecutively, as a pair, to increment the FIFO read pointer. Bits 4 and 5 of register IRB_RX_INT_STA give the fullness level of the FIFO.

If the FIFO is full and has not been read before the arrival of new data, then this data is lost and a receive overrun flag is set in the status register IRB_RX_INT_STA. No new data is written to the FIFO while this condition exists.

To reset the overrun flag:

1. Read at least one word from each of the receive FIFO registers, IRB_RX_SYMB_PER and IRB_RX_ON_TIME.
2. Clear the RX_OVERRUN_STA bit by writing 0x01 to register IRB_RX_CLR.

The last symbol is detected using a time out condition whose value is stored (in µs) in register IRB_IRDA_RX_MAX_SYMB_PER. If no pulse has been received during this time then the last word in the FIFO IRB_RX_MAX_SYMB_PER has a value 0xFFFF. If the value of bit IRB_RX_INT_EN.LAST_SYMB_INT_EN is 1, then an interrupt is triggered and the status register IRB_RX_INT_STA.LAST_SMB_INT is set. The interrupt and its status bit are cleared automatically when the last value in the FIFO has been read.

When IRB_RX_INT_EN.INT_EN is set to 0 then both the FIFO level interrupt and the last symbol interrupt are inhibited.

RC data reception can be disabled by setting IRB_RX_INT_EN.INT_EN to 0. However, both receivers are normally always enabled.

The polarity of input RC_DATA can be inverted by setting bit 0 in one of the polarity invert registers IRB_POL_INV.

70.2.3 Noise suppression filter

This filter is turned off in the IR receiver and is programmable in the UHF receiver using register IRB_RX_NOISE_SUPP_WID_UHF. Any pulses, either high or low, having a value in microseconds of less than the programmed width, are assumed to be noise and, therefore, suppressed.

The noise suppression filter can be disabled by writing 0x00 to register IRB_RX_NOISE_SUPP_WIDTH_UHF.

70.3 Start code detector

The start code detector detects any programmable start code on the RC-Rx input. The length of the start code (number of bits), the minimum, nominal and maximum time of the input signal and the start code are all programmable.

The start code detector works on a time unit called symbol time. The value of the input is expected to be constant during the symbol period. If the value of the input violates the symbol period, then the start code detect resets the start code detection process and restarts searching for start codes.

The minimum and nominal symbol periods are programmed in two different registers, IRB_SCD_SYMB_MIN_TIME and IRB_SCD_SYMB_NOM_TIME. For the symbol to be valid, it has to be constant for at least the minimum symbol period. The symbol is registered as 0 or 1 only after expiry of the nominal symbol time. The registering of the symbol is done by the clock, which aligns itself to changes in the input data. This periodic phase alignment ensures that wrong start codes are not detected and that valid start codes do not go undetected.

The maximum number of symbols in a start code sequence is 32. The start code detector can be programmed to detect a start code which is smaller than 32 symbols.

70.3.1 Generation of subcarrier

The methodology of sub carrier generation is documented in this section. Configuration of different registers to generate sub-carrier is discussed with an example. The pre-scaler divides the system clock (60 MHz) to get the required granularity.

For example lets assume a 40 kHz clock with 50% duty cycle is to be generated.

To get the 40 kHz clock, the programmable counters have to be programmed to generate a clock of 25us clock period and 12.5 µs on time. This is done because the time period and on time of the sub-carrier are now programmable.

To achieve the above time period the pre-scaler can be configured for divide by 6. The out put of the pre-scaler will be 60/6 = 10 MHz. This clock gives a granularity of 0.1us in sub carrier generation. To configure the pre-scaler in divide by 6mode, write '6' in 'pre-scaler register.

Register IRB_TX_SUBCARR_IR has to be programmed with an appropriate value so that the required time period for the sub-carrier clock cycle is achieved. To get a clock period of 25 µs, write 250 to IRB_TX_SUBCARR_IR. This generates a clock of period 25 µs, or 40 kHz.

To get a 50% duty cycle, the subcarrier has to be high for 12.5 µs, by writing 125 to IRB_TX_SUBCARR_WID_IR.

70.3.2 Signalling rates and pulse duration specification for IrDA

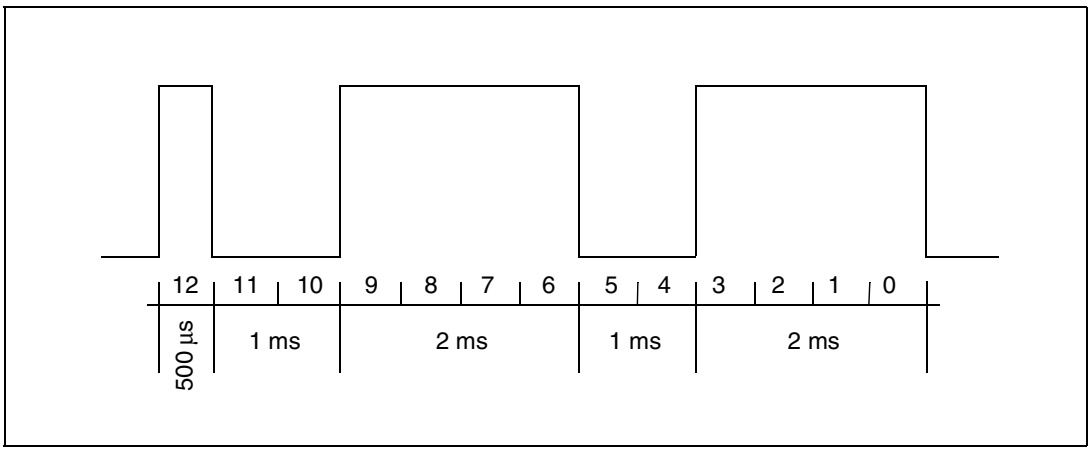
Table 212: Signalling rate and pulse duration specification

Signalling rate (kb/s)	Tolerance % of rate (+/-)	Pulse duration (µs)		
		Min	Nom	Max
9.6	0.87	1.41	19.53	22.13
19.2	0.87	1.41	9.77	11.07
38.4	0.87	1.41	4.88	5.96
57.6	0.87	1.41	3.26	4.34
115.2	0.87	1.41	1.63	2.23

70.3.3 Start code detection example

As an example, suppose the start code detector is programmed with the start code as shown in Figure 266.

Figure 266: Start code example



Here, the nominal symbol duration is 500 µs and there are 13 symbols (denoted as 12 to 0). Since the data is shifted through the shift register, the data bit first received is the MSB.

1. Since the SCD operates on a 100 MHz clock, program the prescaler to 100 (0x64). The sampling clock is 1 MHz, and sampling resolution is 1µs.
2. Program 450 (µs) in register IRB_SCD_SYMB_MIN_TIME, 500 (µs) in IRB_SCD_SYMB_NOM_TIME and 550 (µs) in IRB_SCD_SYMB_MAX_TIME.
3. Program IRB_SCD_CODE with 0b1 0011 1100 1111 and IRB_SCD_CODE_LEN with 13 (ox0E) corresponding to 13 symbols to be detected.
4. Start the start code detection by setting the enable and re-search bits in IRB_SCD_CFG to 1.

The start code detectors checks for the minimum symbol time of each register and the sequence in which symbols are received. If the symbol time is not respected by the input UHF (if noisy) the start code detection is re-initialized.

Constraints on SCD registers

The following relationships need to be respected:

- SCD_SYMB_MAX_TIME <= (SCD_SYMBOL_NOM_TIME * 1.5)
- SCD_SYMB_MIN_TIME >= (SCD_SYMBOL_NOM_TIME * 0.5)
- (SCD_SYMBOL_MAX_TIME - SCD_SYMBOL_NOM_TIME) >= 4
- (SCD_SYMBOL_NOM_TIME - SCD_SYMBOL_MIN_TIME) >= 4
- Start code should not be equal to reset value (all zeros).
- IRB_SCD_CODE_LEN 0x00 corresponds to 32-bit standard and alternate start codes.
- If there is only one start code to be detected, the registers for normal and alternate start codes must be programmed with identical values, as do the IRB_SCD_CODE_LEN values.

Noise recovery

IRB_SCD_NOISE_RECOV is provided to overcome possible issues in detecting a valid start code in cases where noise preceding the start code has the same logic level as that of a start code LSB; in such cases the SCD logic may fail to detect a valid start if this register is not enabled. For example, considering the start code of Figure 266 (0b1 0011 1100 1111) noise at logic level 1 must be ignored.

Example: Start code is 13-bit, 0b1 0011 1100 1111:

IRB_SCD_NOISE_RECOV

.EN = 1 - noise recovery feature is enabled.

.LOGIC_LEV = 1 - the logical value of first symbol in start code is 1.

.NCSSLV = 00001 as there is change in logic value after first symbol of the start code.

71 Infrared transmitter/receiver registers

This section describes the RC transmitter and receiver registers, the RC and UHF receiver and control registers and the noise suppression registers of the IR transmitter/receiver. Although the IR RC receiver and UHF RC receiver registers are held at different addresses, their register descriptions are identical and are only given once for each pair of registers. Registers are suffixed with _IR and _UHF as appropriate.

Register addresses are provided as *IRBBaseAddress* + offset.

The *IRBBaseAddress* is:

0x1801 8000.

Table 213: Infrared transmitter/receiver register summary

Register	Description	Offset		Type
		IR	UHF	
RC transmitter				
IRB_TX_PRESCALER	Clock prescaler	0x00	-	R/W
IRB_TX_SUBCARR	Subcarrier frequency programming	0x04	-	R/W
IRB_TX_SYMB_PER	Symbol time programming	0x08	-	WO
IRB_TX_ON_TIME	Symbol on time programming	0x0C	-	RO
IRB_TX_INT_EN	Transmit interrupt enable	0x10	-	R/W
IRB_TX_INT_STA	Transmit interrupt status	0x14	-	RO
IRB_TX_EN	RC transmit enable	0x18	-	R/W
IRB_TX_CLR	Transmit interrupt clear	0x1C	-	WO
IRB_TX_SUBCARR	Subcarrier frequency programming	0x20	-	R/W
IRB_TX_STA	Transmit status	0x24	-	RO
RC receiver				
IRB_RX_ON_TIME	Received pulse time capture	0x40	0x80	RO
IRB_RX_SYMB_PER	Received symbol period capture	0x44	0x84	RO
IRB_RX_INT_EN	Receive interrupt enable	0x48	0x88	R/W
IRB_RX_INT_STA	Receive interrupt status	0x4C	0x8C	RO
IRB_RX_EN	RC receive enable	0x50	0x90	R/W
IRB_RX_MAX_SYMB_PER	Maximum RC symbol period	0x54	0x94	R/W
IRB_RX_CLR	Receive interrupt clear	0x58	0x98	WO
Noise suppression				
IRB_RX_NOISE_SUPP_WID	Noise suppression width	0x5C	0x9C	R/W
I/O control				
RC_IO_SEL	I/O select RC or IrDA	0x60		R/W

Table 213: Infrared transmitter/receiver register summary

Register	Description	Offset		Type
		IR	UHF	
Reverse polarity				
IRB_POL_INV	Reverse polarity data	0x68	0xA8	R/W
Receive status and clock				
IRB_RX_STA	Receive status and interrupt clear	0x6C	0xAC	RO
IRB_SAMPLE_RATE_COMM	Sampling frequency division	0x64		R/W
IRB_CLK_SEL	Clock select configuration	0x70		R/W
IRB_CLK_SEL_STA	Clock select status	0x74		RO
IrDA Interface				
IRB_IRDA_BAUD_RATE_GEN	Baud rate generation for IR	0xC0	-	R/W
IRB_IRDA_BAUD_GEN_EN	Baud rate generation enable for IR	0xC4	-	R/W
IRB_IRDA_TX_EN	Transmit enable for IR	0xC8	-	R/W
IRB_IRDA_RX_EN	Receive enable for IR	0xCC	-	R/W
IRB_IRDA_ASC_CTRL	Asynchronous data control	0xD0	-	R/W
IRB_IRDA_RX_PULSE_STA	Receive pulse status for IR	0xD4	-	RO
IRB_IRDA_RX_SAMPLE_RATE	Receive sampling rate for IR	0xD8	-	WO
IRB_IRDA_RX_MAX_SYMB_PER	Receive maximum symbol period for IR	0xDC	-	WO
Start code detector				
IRB_SCD_CFG	SCD configuration	0x200		R/W
IRB_SCD_STA	SCD status	0x204		RO
IRB_SCD_CODE	Start code to be detected	0x208		R/W
IRB_SCD_CODE_LEN	Start code length	0x20C		R/W
IRB_SCD_SYMB_MIN_TIME	SCD minimum symbol time	0x210		R/W
IRB_SCD_SYMB_MAX_TIME	SCD maximum symbol time	0x214		R/W
IRB_SCD_SYMB_NOM_TIME	SCD nominal symbol time	0x218		R/W
IRB_SCD_PRESCALAR	SCD prescalar value	0x21C		R/W
IRB_SCD_INT_EN	SCD detect interrupt enable	0x220		R/W
IRB_SCD_INT_CLR	SCD detect interrupt clear	0x224		WO
IRB_SCD_INT_STA	SCD detect interrupt status	0x22C		RO
IRB_SCD_NOISE_RECOV	SCD noise recovery configuration	0x228		R/W
IRB_SCD_ALT_CODE	Alternative start code to be detected	0x230		R/W

71.1 RC transmitter registers

IRB_TX_PRESCALER Clock prescaler

7	6	5	4	3	2	1	0
PRESCALE_VAL							

Address: *IRBBaseAddress* + 0x00
Type: R/W
Reset: 0
Description: Selects the value of the prescaler for clock division. The prescaled clock frequency is obtained by dividing the system clock frequency by PRESCALE_VAL. It determines the transmit subcarrier resolution, see IRB_TX_SUBCARR_IR.

IRB_TX_SUBCARR Subcarrier frequency programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBCARR_VAL															

Address: *IRBBaseAddress* + 0x04
Type: R/W
Reset: 0
Description: Determines the RC transmit subcarrier frequency. The prescaled clock frequency divided by (SUBCARR_VAL x 2) gives the subcarrier frequency, which has a 50% duty cycle.

IRB_TX_SYMB_PER Symbol time programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_SYMB_TIME_VAL															

Address: *IRBBaseAddress* + 0x08
Type: WO
Buffer: 8-word buffered
Reset: 0
Description: Gives the symbol time (symbol period) in periods of the subcarrier clock. It must be programmed sequentially with register IRB_TX_ON_TIME.

IRB_TX_ON_TIME Symbol on time programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX_ON_TIME_VAL															

Address: *IRBBaseAddress* + 0x0C
Type: RO
Buffer: 8-word buffered
Reset: 0
Description: Gives the symbol on time (pulse duration) in periods of the subcarrier clock.
Note: *Registers* IRB_TX_SYMB_PER *and* IRB_TX_ON_TIME *act as a single register set. They must be programmed sequentially as a pair to latch in the data.*

IRB_TX_INT_EN Transmit interrupt enable

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											1WD	HALF_EMPTY	EMPTY	UNDERRUN	INT_EN

Address: *IRBBaseAddress* + 0x10
Type: R/W
Reset: 0
Description:

[15:5] **Reserved**

[4] **1WD**
1: Interrupt enable on at least one word empty in FIFO

[3] **HALF_EMPTY**
1: Interrupt enable on FIFO half empty

[2] **EMPTY**
1: Interrupt enable on FIFO empty

[1] **UNDERRUN**
1: Enable interrupt on underrun

[0] **INT_EN**: Interrupt enable
1: Global transmit interrupt enable

IRB_TX_INT_STA Transmit interrupt status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											1WD	HALF_EMPTY	EMPTY	UNDERRUN	INT_PEND

Address: *IRBBaseAddress* + 0x14
Type: RO
Reset: 0
Description: This register is also updated when data is written into registers IRB_TX_SYMB_PER and IRB_TX_IR_ON_TIME.

[15:5] **Reserved**

[4] **1WD**
1: At least one word empty interrupt pending

[3] **HALF_EMPTY**
1: FIFO half empty interrupt pending

[2] **EMPTY**
1: FIFO Empty interrupt pending

[1] **UNDERRUN**
1: Underrun interrupt pending

[0] **INT_PEND**: Interrupt pending
1: Global interrupt pending

IRB_TX_EN RC transmit enable

7	6	5	4	3	2	1	0
Reserved							TX_EN

Address: *IRBBaseAddress* + 0x18

Type: R/W

Reset: 0

Description: Enables the RC transmit processor. When it is set to 1 and there is data in the transmit FIFO, then the RC processor is transmitting.

IRB_TX_CLR Transmit interrupt clear

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											1WD	HALF_EMPTY	EMPTY	UNDERRUN	Reserved

Address: *IRBBaseAddress* + 0x1C

Type: WO

Reset: 0

Description:

[15:5] **Reserved**

[4] **1WD**
1: Clear interrupt: at least one word empty in FIFO

[3] **HALF_EMPTY**
1: Clear interrupt: FIFO half-empty

[2] **EMPTY**
1: Clear interrupt: FIFO empty

[1] **UNDERRUN**
1: Clear interrupt: underrun

[0] **Reserved**

IRB_TX_SUBCARR_WID Subcarrier frequency programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBCARR_WID_VAL															

Address: *IRBBaseAddress* + 0x20

Type: R/W

Reset: 0

Description: The pulse width of the subcarrier generated is programmed into this register. Loading a value *k* into this register keeps the subcarrier high for *n* * *k* comms clock cycles. Where *n* is the value loaded into the IRB_TX_PRESCALER register. Software has to ensure that the value written in this register is less than that written in register IRB_TX_SUBCARR. If the condition is not met, the subcarrier is not generated.

IRB_TX_STA Transmit status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TX_FIFO_LEVEL			Reserved			1WD	HALF_EMPTY	EMPTY	UNDERRUN	Reserved

Address: *IRBBaseAddress* + 0x24
Type: RO
Reset: 0x1C
Description:

[11:15] **Reserved**

[10:8] **TX_FIFO_LEVEL**
000: FIFO empty
001: 1 block in FIFO
010: 2 blocks in FIFO
011: 3 blocks in FIFO
100: 4 blocks in FIFO
101: 5 blocks in FIFO
110: 6 blocks in FIFO
111: 7 blocks in FIFO (full)

[7:5] **Reserved**

[4] **1WD**
1: At least one word empty in FIFO

[3] **HALF_EMPTY**
1: FIFO half empty

[2] **EMPTY**
1: FIFO empty

[1] **UNDERRUN**
1: FIFO underrun

[0] **Reserved**

Clearing an interrupt **does not** clear the corresponding status flag. The status reflects the true transmit status.

71.2 RC receiver registers

If not explicitly stated the following registers are common to both the RC IR receiver and the RC UHF receiver. The first address given is the RC IR receiver (IR). The registers are distinguished by the suffix _IR for the IR receiver and _UHF for the UHF receiver.

IRB_RX_ON_TIME Received pulse time capture

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_ONTIME_VAL															

Address: *IRBBaseAddress* + 0x40 (IR) *IRBBaseAddress* + 0x80 (UHF)

Type: RO

Buffer: 8-word buffered

Reset: 0

Description: Detected duration of the received RC pulse in microseconds. Must be read sequentially with register IRB_RX_SYMB_PER.

IRB_RX_SYMB_PER Received symbol period capture

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_SYMB_TIME_VAL															

Address: *IRBBaseAddress* + 0x44 (IR) *IRBBaseAddress* + 0x84 (UHF)

Type: RO

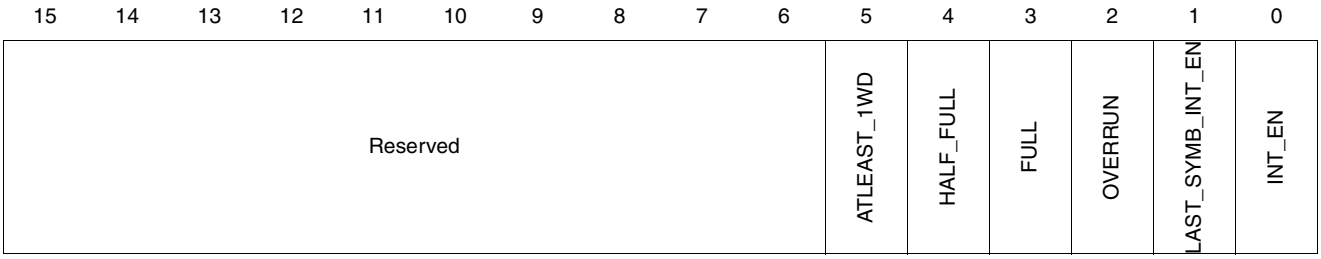
Buffer: 8-word buffered

Reset: 0

Description: Detected time between the start of two successive received RC pulses, in microseconds.

Note: Registers IRB_RX_SYMB_PER *and* IRB_RX_IR_ON_TIME *act as a register set. A new value can only be read after reading both registers sequentially.*

IRB_RX_INT_EN Receive interrupt enable



Address: *IRBBaseAddress* + 0x48 (IR) *IRBBaseAddress* + 0x88 (UHF)

Type: R/W

Reset: 0

Description:

[15:6] **Reserved**

[5] **ATLEAST_1WD**
1: Enable interrupt on at least one word in FIFO

[4] **HALF_FULL**
1: Enable interrupt on FIFO half-full

[3] **FULL**
1: Enable interrupt on FIFO full

[2] **OVERRUN**
1: Enable interrupt on overrun

[1] **LAST_SYMB_INT_EN**
1: Enable interrupt on last symbol receive

[0] **INT_EN**
1: Enable global receive interrupt

IRB_RX_INT_STA Receive interrupt status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ATLEAST_1WD	HALF_FULL	FULL	OVERRUN	LAST_SYMB_INT	INT

Address: *IRBBaseAddress* + 0x4C (IR) *IRBBaseAddress* + 0x8C (UHF)
Type: RO
Reset: 0
Description:

[15:6] **Reserved**

[5] **ATLEAST_1WD**
1: At least one word in FIFO interrupt pending

[4] **HALF_FULL**
1: Half-full interrupt pending

[3] **FULL**
1: FIFO full interrupt pending

[2] **OVERRUN**
1: FIFO overrun pending

[1] **LAST_SYMB_INT**
1: Last symbol receive interrupt pending

[0] **INT**
Global receive interrupt pending

IRB_RX_EN RC receive enable

7	6	5	4	3	2	1	0
Reserved							RX_EN

Address: *IRBBaseAddress* + 0x50 (IR) *IRBBaseAddress* + 0x90 (UHF)
Type: R/W
Reset: 0
Description: 1: The RC receive section is enabled to read incoming data.

IRB_RX_MAX_SYMB_PER Maximum RC symbol period

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_MAX_SYMB_TIME															

Address: *IRBBaseAddress* + 0x54 (IR) *IRBBaseAddress* + 0x94 (UHF)
Type: R/W
Reset: 0
Description: Sets the maximum symbol period (in microseconds) which is necessary to define the time out for recognizing the end of the symbol stream.

IRB_RX_CLR Receive interrupt clear

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ATLEAST_1WD	HALF_FULL	FULL	OVERRUN	LAST_SYMB_INT_EN	INT_EN

Address: *IRBBaseAddress* + 0x58 (IR) *IRBBaseAddress* + 0x98 (UHF)

Type: WO

Reset: 0

Description: Set to 1 as part of the procedure for clearing flags in register IRB_RX_INT_STA. No new data is written into the receive FIFO while these bits are set.

[15:6] **Reserved**

[5] **ATLEAST_1WD**
1: Clear interrupt: at least one word in FIFO

[4] **HALF_FULL**
1: Clear interrupt: FIFO half-full

[3] **FULL**
1: Clear interrupt: FIFO full

[2] **OVERRUN**
1: Clear interrupt: FIFO overrun

[1] **LAST_SYMB_INT**
1: Clear interrupt: last symbol receive

[0] **Reserved**

71.3 Noise suppression

IRB_RX_NOISE_SUPP_WID Noise suppression width

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOISE_SUPP_WID															

Address: *IRBBaseAddress* + 0x5C (IR) *IRBBaseAddress* + 0x9C (UHF)

Type: R/W

Reset: 0

Description: Determines the maximum width of noise pulses, in microseconds, which the filter suppresses.

71.4 I/O control

RC_IO_SEL I/O select RC or IrDA

7	6	5	4	3	2	1	0
Reserved							IO_SEL

Address: *IRBBaseAddress* + 0x60
Type: R/W
Reset: 0
Description: Selects the data type on RC_IRDA_DATA_OUT and RC_IRDA_DATA_IN pins.
0: RC data
1: IrDA data
This register is present only in the receive interface for IR signals.

71.5 Reverse polarity

The two IRB input pins (IRB_IR_IN {PIO3 bit 0} and IRB_UHF_IN {PIO5 bit 1}) are inverted internally from high to low. To account for this, IRB_IR_IN and IRB_UHF_IN should be configured as PIO inputs and the bits in the POLINV registers set to 1.

IRB_POL_INV Reverse polarity data

7	6	5	4	3	2	1	0
Reserved							POLARITY

Address: *IRBBaseAddress* + 0x68 (IR) *IRBBaseAddress* + 0xA8 (UHF)
Type: R/W
Reset: 0
Description:

[7:1] **Reserved**

[0] **POLARITY**
0: No polarity inversion 1: Polarity of IR data inverted
This bit should always be set to 1

71.6 Receive status and clock

IRB_RX_STA Receive status and interrupt clear

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RX_FIFO_LEVEL			Reserved		AT_LEAST_1WD	HALF_FULL	FULL	OVERRUN	LAST_SYMB	Reserved

Address: *IRBBaseAddress* + 0x6C (IR) *IRBBaseAddress* + 0xAC (UHF)

Type: RO

Reset: 0

Description:

[15:11] **Reserved**

[10:8] **RX_FIFO_LEVEL**

000: FIFO empty
001: 1 block FIFO
010: 2 blocks in FIFO
011: 3 blocks in FIFO
100: 4 blocks in FIFO
101: 5 blocks in FIFO
110: 6 blocks in FIFO
111: 7 blocks in FIFO (full)

[7:6] **Reserved**

[5] **AT_LEAST_1WD**: At least one word
1: Clear interrupt: at least one word in FIFO

[4] **HALF_FULL**
1: Clear interrupt: FIFO half full

[3] **FULL**
1: Clear interrupt FIFO full

[2] **OVERRUN**
1: Clear interrupt: FIFO overrun

[1] **LAST_SYMB**: Last symbol
1: Clear interrupt: last symbol receive

[0] **Reserved**

Note: Clearing the interrupt does not clear the status. To clear the status, appropriate actions (such as reading the data from the FIFO) have to be performed.

IRB_SAMPLE_RATE_COMM Sampling frequency division

7	6	5	4	3	2	1	0
Reserved				N			

Address: *IRBBaseAddress* + 0x64

Type: R/W

Reset: 0

Description: Programs the sampling rate for the RC codes receive section. A 4-bit counter with auto reload feature generates the sampling frequency. This counter is configured such that the output of this counter is 10 MHz.

The clock is divided by *N*.

This is a common register for both IR and UHF receivers.

IRB_CLK_SEL Clock select configuration

7	6	5	4	3	2	1	0
Reserved							CLK_SEL

Address: *IRBBaseAddress* + 0x70

Type: R/W

Reset: 0

Description: Used to select if the receive sections (both RC and UHF) are clocked by the system clock or the 27 MHz clock. In low power mode it is expected that the system clock is switched off. The noise suppression filter is clocked by 27 MHz clock to ensure the filtering takes place on the received signal even in the low power mode.

[7:1] **Reserved**: Set to 0

[0] **CLK_SEL**
0: System clock 1: 27 MHz clock

IRB_CLK_SEL_STA Clock select status

7	6	5	4	3	2	1	0
Reserved							CLK_STA

Address: *IRBBaseAddress* + 0x74

Type: RO

Reset: 0

Description: Used to infer if the receive section is clocked by the system clock or the 27 MHz clock. After changing the clocking mode by programming register IRB_CLK_SEL, software has to read this register to see if the programmed clock change has happened.

[7:6] **Reserved**

[0] **CLK_STA**
0: System clock 1: 27 MHz clock

71.7 IrDA Interface

IRB_IRDA_BAUD_RATE_GEN Baud rate generation for IR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASCBAUD															

Address: *IRBBaseAddress* + 0xC0
Type: R/W
Reset:
Description: The baud rate generation is exactly same as in ASC. It has a 16-bit counter with auto-reload capability. This register holds the value ASCBAUD to be loaded into the counter.

ASCBAUD can be calculated by the formula:

ASCBAUD = f_{CPU} / (16* *baudrate*)

where f_{CPU} is the CPU clock frequency. For a CPU clock frequency of 100 MHz, the values to be loaded into BAUD_RATE_GEN_IRDA are shown in Table 214 below.

Table 214: Bit fields in interrupt enable register

Baud rate	Reload value (to 3 dec places)	Reload value (Integer)	Reload value (Hex)	Approximate deviation
9600	651.042	651	0x28B	0.01%
19200	325.521	326	0x146	0.15%
34.8 k	179.600	180	0x0B4	0.22%
57.6 k	108.507	109	0x06D	0.45%
115.2 k	54.250	54	0x036	0.46%

IRB_IRDA_BAUD_GEN_EN Baud rate generation enable for IR

7	6	5	4	3	2	1	0
Reserved							BRG_EN

Address: *IRBBaseAddress* + 0xC4
Type: R/W
Reset: 0
Description: 1: The baud rate generator is enabled.

IRB_IRDA_TX_EN Transmit enable for IR

7	6	5	4	3	2	1	0
Reserved							IRTX_EN

Address: *IRBBaseAddress* + 0xC8
Type: R/W
Reset: 0
Description: 1: The IrDA transmit section is enabled.

IRB_IRDA_RX_EN Receive enable for IR

7	6	5	4	3	2	1	0
Reserved							IRRX_EN

Address: *IRBBaseAddress* + 0xCC
Type: R/W
Reset: 0
Description: 1: The IrDA receive section is enabled.

IRB_IRDA_ASC_CTRL Asynchronous data control

7	6	5	4	3	2	1	0
Reserved							ASC_CTRL

Address: *IRBBaseAddress* + 0xD0
Type: R/W
Reset: 0
Description: Controls the data on pins IRDA_ASC_TX_DATA_OUT and IRDA_ASC_RX_DATA_IN.
0: IrDA data is available on pins IRDA_ASC_TX_DATA_OUT and IRDA_ASC_RX_DATA_IN.
1: Asynchronous data is available on pins IRDA_ASC_TX_DATA_OUT and IRDA_ASC_RX_DATA_IN.

IRB_IRDA_RX_PULSE_STA Receive pulse status for IR

7	6	5	4	3	2	1	0
Reserved							P_STA

Address: *IRBBaseAddress* + 0xD4
Type: RO
Reset: 0
Description: Set to one if there is pulse width violation of IrDA input signal from the infrared detector. This bit is set to 0 when it is read.

IRB_IRDA_RX_SAMPLE_RATE Receive sampling rate for IR

7	6	5	4	3	2	1	0
Reserved							N

Address: *IRBBaseAddress* + 0xD8
Type: WO
Reset:
Description: The sampling frequency of the IrDA receive signal is selected by programming this register. If f_{IRB} is the module clock frequency, then this register must be programmed with a value *N* such that f_{IRB}/N = 10 MHz.

IRB_IRDA_RX_MAX_SYMB_PER Receive maximum symbol period for IR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M															

Address: *IRBBaseAddress* + 0xDC
Type: WO
Reset:
Description: The maximum symbol time for which the IR pulse should be high is programmed in this register. If a value *M* is written in this register, then the maximum pulse duration is *M*/10 µs. If an IR pulse greater than this time is detected then the IR pulse is neglected.

71.8 SCD

IRB_SCD_CFG SCD configuration

7	6	5	4	3	2	1	0
Reserved				RST_SFT	SW_RST	RE_SEARCH	EN

Address: *IRBBaseAddress* + 0x200
Type: R/W
Reset: 0
Description:

[7:4] **Reserved**

[3] **RST_SFT**
Reset shift register only. SCD_STA is not affected.

[2] **SW_RST**
Reset all counters and shift register.

[1] **RE_SEARCH**
1: Start a new search
Asserting RE_SEARCH while start code detection is in progress has no effect. The purpose of this bit is to provide software a capability to force a restart when SCD has already detected a start code and symbol-time-out does not occur.

[0] **EN**
0: Bypass SCD, UHF sent to UHF_OUT
1: Enable start code detection

IRB_SCD_STA SCD status

7	6	5	4	3	2	1	0
Reserved						ALT	DETECT

Address: *IRBBaseAddress* + 0x204
Type: RO
Reset: 0
Description:

[7:2] **Reserved**

[1] **ALT**
1: Alternative code detected

[0] **DETECT**
1: Start code detected, UHF sent to UHF_OUT

IRB_SCD_CODE Start code to be detected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODE																															

Address: *IRBBaseAddress* + 0x208
Type: R/W
Reset: 0
Description:

IRB_SCD_CODE_LEN Start code length

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			ALT_CODE_LEN					Reserved			CODE_LEN				

Address: *IRBBaseAddress* + 0x20C
Type: R/W
Reset: 0
Description: Length of the start code in symbols.

[31:13] **Reserved**

[12:8] **ALT_CODE_LEN**: Alternative start code length

[7:5] **Reserved**

[4:0] **CODE_LEN**: Start code length

IRB_SCD_SYMB_MIN_TIME SCD minimum symbol time

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIN _TIME																															

Address: *IRBBaseAddress* + 0x210

Type: R/W

Reset: 0

Description: Minimum time of a symbol. If any symbol is shorter than this value, the SCD process is re-initialized. The symbol time counting is done by a clock (enable pulse) output from the pre-scaler. If the minimum time of the symbol is *n* pre-scaler clock periods, the SCD_SYMB_MIN_TIME register should be programmed with a value of (*n*-1). For example, if a value 0xF is written into this register, the symbol minimum time is 16 pre-scaler clock periods.

IRB_SCD_SYMB_MAX_TIME SCD maximum symbol time

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAX_SYMB_TIME																															

Address: *IRBBaseAddress* + 0x214

Type: R/W

Reset: 0

Description: Maximum time of a symbol. Any changes in the input data are allowed only between symbol minimum time and symbol maximum time. The symbol time counting is done by a clock (enable pulse) output from the pre-scaler. If the maximum time of the symbol is *n* pre-scaler clock periods, the SCD_SYMB_MAX_TIME register should be programmed with a value of (*n*-1). For example, if a value 0xF is written into this register, the symbol maximum time is 16 pre-scaler clock periods.

IRB_SCD_SYMB_NOM_TIME SCD nominal symbol time

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOM_TIME																															

Address: *IRBBaseAddress* + 0x218

Type: R/W

Reset: 0

Description: Nominal time for a symbol. This value is used by SCD to register a new symbol for when consecutive identical symbols are received. The symbol time counting is done by a clock (enable pulse) output from the pre-scaler. If the SCD nominal time is *n* pre-scaler clock periods, the SCD_SYMB_NOM_TIME register should be programmed with a value of (*n*-1).For example, if a value 0xF is written into this register, the symbol nominal time is 16 pre-scaler clock periods.

IRB_SCD_PRESCALAR SCD prescalar value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRE_SCALAR															

Address: *IRBBaseAddress* + 0x21C
Type: R/W
Reset: 0x01
Description: Prescalar division value

IRB_SCD_INT_EN SCD detect interrupt enable

7	6	5	4	3	2	1	0
Reserved							SCD_INT_EN

Address: *IRBBaseAddress* + 0x220
Type: R/W
Reset: 0
Description: Enable interrupt on SCD detected
0: Disable interrupt
1: Enable interrupt

IRB_SCD_INT_CLR SCD detect interrupt clear

7	6	5	4	3	2	1	0
Reserved							SCD_INT_CLR

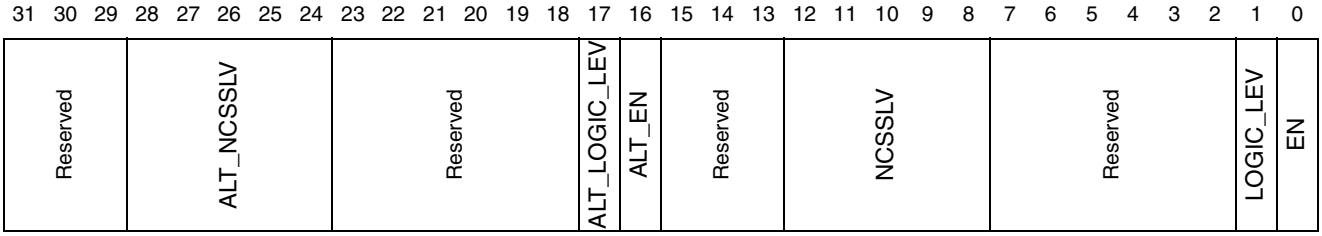
Address: *IRBBaseAddress* + 0x
Type: WO
Reset: 0
Description: Clear SCD-detected interrupt. This register clears the interrupt only when the SCD is functioning on the interconnect clock.
1: Clear interrupt

IRB_SCD_INT_STA SCD detect interrupt status

7	6	5	4	3	2	1	0
Reserved							SCD_INT_STA

Address: *IRBBaseAddress* + 0x22C
Type: RO
Reset: 0
Description: Status of SCD detected interrupt.
0: No pending interrupt
1: Pending interrupt

IRB_SCD_NOISE_RECOV SCD noise recovery configuration



Address: *IRBBaseAddress* + 0x228
Type: R/W
Reset: 0
Description:

[31:29] **Reserved**

[28:24] **ALT_NCSSLV**: Number of contiguous symbols at same logical value as first symbol, for alternate code
0x00: Noise recovery disabled
0x01
0x02
...
0x1E
0x1F

[23:18] **Reserved**

[17] **ALT_LOGIC_LEV**: Logic level for alternate code
0: Alt code starts with 0 1: Alt code starts with 1

[16] **ALT_EN**: Enable noise recovery for alternate code
0: Noise recovery disabled 1: Noise recovery enabled

[15:13] **Reserved**

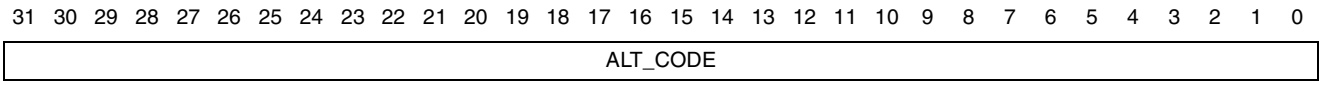
[12:8] **NCSSLV**: Number of contiguous symbols at same logical value as first symbol

[7:2] **Reserved**

[1] **LOGIC_LEV**: Logic level
0: Code starts with 0 1: Code starts with 1

[0] **EN**: Enable noise recovery
0: Noise recovery disabled 1: Noise recovery enabled

IRB_SCD_ALT_CODE Alternative start code to be detected



Address: *IRBBaseAddress* + 0x230
Type: R/W
Reset: 0
Description:

72 Modem analog front-end (MAFE) interface

72.1 Overview

The modem analog front end interface (MAFE) is an integrated interface to an analog front end (AFE) for a modem such as the STLC7550.

In this chapter, the term sample is a 16-bit data object that is transferred to or from the modem through the MAFE, and the term sample period is the time from the start of one sample to the start of the next.

The MAFE simultaneously transmits samples into and out of the AFE. It typically operates at a rate of 9600 samples/second, giving a typical sample period of 100 µs. That is, every 100 µs, one sample is transmitted and another received through the MAFE.

The MAFE receives its system clock signal (SCLK) from the AFE. The SCLK frequency is typically 256 ticks/sample period, or 2.56 MHz. The first 16 ticks of the 256 tick sample period are used to exchange a 16-bit sample pair (1 bit per tick).

The MAFE uses one DMA to transfer samples from a transmit memory buffer to the AFE, and simultaneously uses a second DMA to receive samples from the AFE and write them into the receive memory buffer. The software driver is woken up every time a simultaneous transfer is completed, that is, every time a transmit memory buffer has been emptied and a receive memory buffer has been filled. For example, if each memory buffer contains 100 samples, the software is woken up every 10 ms (100 x 100 µs). This is more stringent for handshake signals, where the buffer size could be as low as a few samples, for example, four.

The software modem has two pairs of pointers (that is, four pointers) that point to two pairs of transmit/receive buffers. The modem and the MAFE alternately switch between the two pairs of pointers. While the MAFE transmits and receives using one pair of buffers, the software modem processes the information in the other pair. Using the above example for a buffer containing 100 samples, the software has 10 ms to wake up and then process one pair of transmit/receive buffers before they are required again by the MAFE.

72.2 Using the MAFE to connect to a modem

The following table lists the pins that are used by the MAFE to connect a modem:

Table 215: MAFE pins

Name	Type	Function name (alternate)	Function description
PIO2[3]	O	MAFE_HC1	Indicates to the AFE that a control/status exchange is to take place.
PIO2[4]	O	MAFE_DOUT	Line for serially transmitting samples to the AFE.
PIO2[5]	I	MAFE_DIN	Line for serially receiving samples from the AFE.
PIO2[6]	I	MAFE_FS	Signal from the AFE indicating the start of a sampling period. This is latched on falling edges of SCLK. For normal operation it should not remain high for more than 16 SCLK cycles, and there should be at least 20 SCLK ticks between consecutive rising edges of F_S.
PIO2[7]	I	MAFE_SCLK	Modem system clock. The frequency should be less than half of the device system clock.

72.3 Software

The MAFE software manages the data exchange between the software modem and MAFE, and handles the control/status exchange.

72.3.1 Data exchange

When the MAFE exchanges data, the software:

1. disables all interrupts,
2. sets the buffer size, for example, 100 samples (for handshake response times, the buffer size could be as low as a few samples, for example 4),
3. sets up both pairs of memory pointers in the MAFE (this is probably not changed again),
4. enables status (complete) interrupt,
5. sets the control (run) bit,
6. deschedules.

The MAFE then processes a buffer load of samples (that is, it transmits 100 samples and receives 100 samples). When this is complete, the MAFE sets the status (complete) bit, causing the software to be woken up. The software then:

7. processes the receive memory buffer and fills the next transmit memory,
8. confirms that there has been no overflow (that is, failure to finish the software processing of a buffer before that buffer has started to be overwritten again),
9. confirms that there have been no memory latency problems during the exchange of the previous buffer, by reading the status (missed) bit,
10. if there are no problems, the software writes to the MOD_ACK register and deschedules.

72.3.2 Control/status exchange

For a control/status exchange, the software writes to register MOD_CTRL to enable the status interrupt (CTRL_EMPTY), and then deschedules.

When the software wakes up, it reads the modem status and disables the status interrupt (CTRL_EMPTY) again.

73 Modem analog front-end (MAFE) interface registers

Register addresses are provided as *ModemBaseAddress* + offset.

The *ModemBaseAddress* is:

0x1805 8000.

Table 216: MAFE interface register summary

Register	Description	Offset	Type
MOD_CTRL_1	Control 1	0x00	R/W
MOD_STA	Status 1	0x04	RO
MOD_INT_EN	Interrupt enable	0x08	R/W
MOD_ACK	Acknowledge	0x0C	WO
MOD_BUFF_SIZE	Buffer size	0x10	R/W
MOD_CTRL_2	Control 2	0x14	WO
MOD_STA	Status 2	0x18	RO
MOD_RECEIVE0_PTR	Receive memory buffer 0 start address	0x20	R/W
MOD_RECEIVE1_PTR	Receive memory buffer 1 start address	0x24	R/W
MOD_TX0_PTR	Transmit memory buffer 0 start address	0x28	R/W
MOD_TX1_PTR	Transmit memory buffer 1 start address	0x2C	R/W

MOD_CTRL_1 Control 1

7	6	5	4	3	2	1	0
Reserved						START	RUN

Address: *ModemBaseAddress* + 0x00

Type: R/W

Reset: Undefined

Description:

[7:2] **Reserved**

[1] **START**

Indicates which of the two pairs of memory buffer pointers it should start off using:

0: Indicates RECEIVE0_POINTER and TRANSMIT0_POINTER.

1: Indicates RECEIVE1_POINTER and TRANSMIT1_POINTER.

[0] **RUN**

1: The MAFE interface is to start exchanging data with the AFE.

0: The MAFE interface stops after completing the exchange of the current buffer load of samples.

MOD_STA Status 1

7	6	5	4	3	2	1	0
Reserved		MISSED	OVERFLOW	LAST	CTRL_EMPTY	COMPLETE	IDLE

Address: *ModemBaseAddress* + 0x04
Type: RO
Reset: Undefined
Description:

[7:6] **Reserved**

[5] **MISSED**
1: Indicates that the memory latency is too high, causing a sample to be missed (the MAFE interface is exchanging samples faster than they can be read from/written to the memory buffers).
Cleared by writing to MOD_ACK.

[4] **OVERFLOW**
1: Indicates that overflow has occurred (the MAFE interface has completed the exchange of another buffer load of samples before the software has got round to acknowledging the previous buffer load).
Cleared by writing to MOD_ACK.

[3] **LAST**: indicates the last pair of buffer pointers used by the DMA.

[2] **CTRL_EMPTY**
Set to 0 by writing to MOD_MAFE_CTRL. Set to 1 when the MAFE interface has completed the control/status exchange.

[1] **COMPLETE**
Set to 1 when a buffer load of samples has been exchanged. Cleared by writing to MOD_ACK register.

[0] **IDLE**
0: The MAFE interface is exchanging data with the AFE.
1: The RUN bit is low and the MAFE interface is not exchanging data. After the software clears the RUN bit, the MAFE interface only goes idle when it has finished exchanging the current buffer load of samples.

MOD_INT_EN Interrupt enable

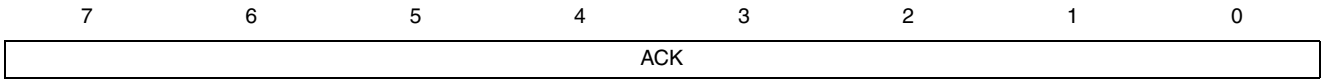
7	6	5	4	3	2	1	0
Reserved					INT_EN2	INT_EN1	INT_EN0

Address: *ModemBaseAddress* + 0x08
Type: R/W
Reset: Undefined
Description:

[7:3] **Reserved**

[2:0] **INT_EN[2:0]**
Enables interrupts connected to MOD_MAFE_STA[2:0].
1: Indicates that the corresponding interrupt is enabled.

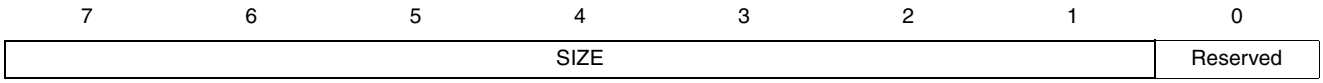
MOD_ACK Acknowledge



Address: *ModemBaseAddress* + 0x0C
Type: WO
Reset: Undefined
Description:

[7:0] **ACK**: Acknowledge
Clears the overflow, missed and complete flags in register MOD_MAFE_STA.

MOD_BUFF_SIZE Buffer size

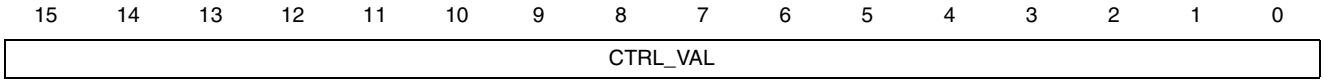


Address: *ModemBaseAddress* + 0x10
Type: R/W
Reset: Undefined
Description:

[7:1] **SIZE**: Buffer size (the number of 16-bit samples in a buffer)
This value must be a multiple of two.

[0] **Reserved**

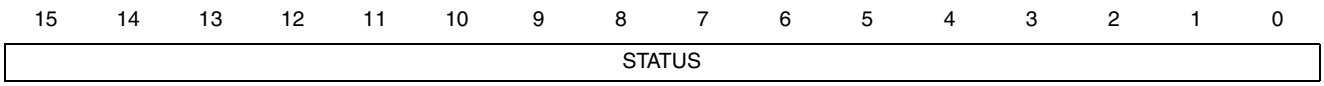
MOD_CTRL_2 Control 2



Address: *ModemBaseAddress* + 0x14
Type: WO
Reset: Undefined
Description:

[15:0] **CTRL_VAL**: Control value to send out to the MAFE interface

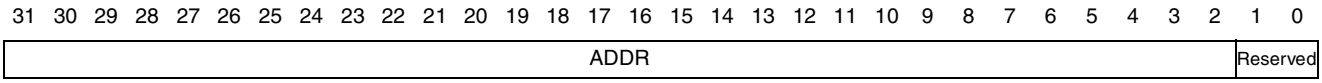
MOD_STA Status 2



Address: *ModemBaseAddress* + 0x18
Type: RO
Reset: Undefined
Description:

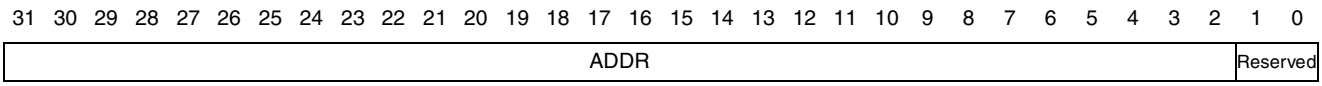
[15:0] **STATUS**: Status value received from the MAFE interface.

MOD_RECEIVE0_PTR Receive memory buffer 0 start address



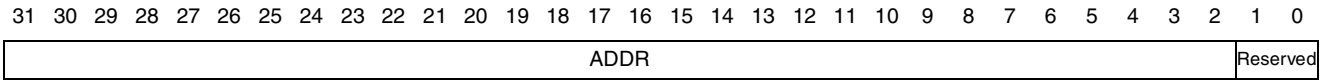
Address: *ModemBaseAddress* + 0x20
Type: R/W
Reset: Undefined
Description: Start address of RECEIVE_MEM_BUFF_0.

MOD_RECEIVE1_PTR Receive memory buffer 1 start address



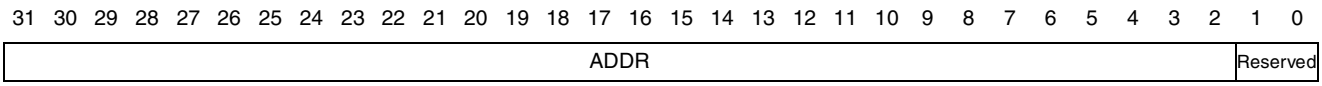
Address: *ModemBaseAddress* + 0x24
Type: R/W
Reset: Undefined
Description: Start address of RECEIVE_MEM_BUFF_1.

MOD_TX0_PTR Transmit memory buffer 0 start address



Address: *ModemBaseAddress* + 0x28
Type: R/W
Reset: Undefined
Description: Start address of TX_MEM_BUFF_0

MOD_TX1_PTR Transmit memory buffer 1 start address



Address: *ModemBaseAddress* + 0x2C
Type: R/W
Reset: Undefined
Description: Start address of TX_MEM_BUFF_1.

74 Direct access arrangement modem (DAA)

See Silicon Laboratories Inc. document *32.4 MHz Differential-Link Interface DAA - Embedded System-Side DAA Module Specification*.

75 Programmable descrambler (PDES)

The STx7100 programmable descrambler is responsible for descrambling broadcast content received in transport streams under the control of the PTI. Descrambling is performed on a packet by packet basis with the possibility to change the descrambling cipher between each packet if necessary. Packet payloads are streamed from the PTI descrambled and returned to the PTI for further processing. The following descrambling ciphers are supported:

- DVB CSA,
- ICAM 2.1,
- FAST-I,
- DES ECB, DES ECB DVS 042, DES CBC, DES CBC DVS 042, DES OFB, DES CTS,
- Multi-2 ECB, Multi-2 ECB DVS 042, Multi-2 CBC, Multi-2 CBC DVS 042, Multi-2 OFB, Multi-2 CTS,
- AES ECB, AES ECB DVS 042, AES CBC, AES CBC DVS 042, AES OFB, AES CTS, AES DTV.

76 PWM and counter

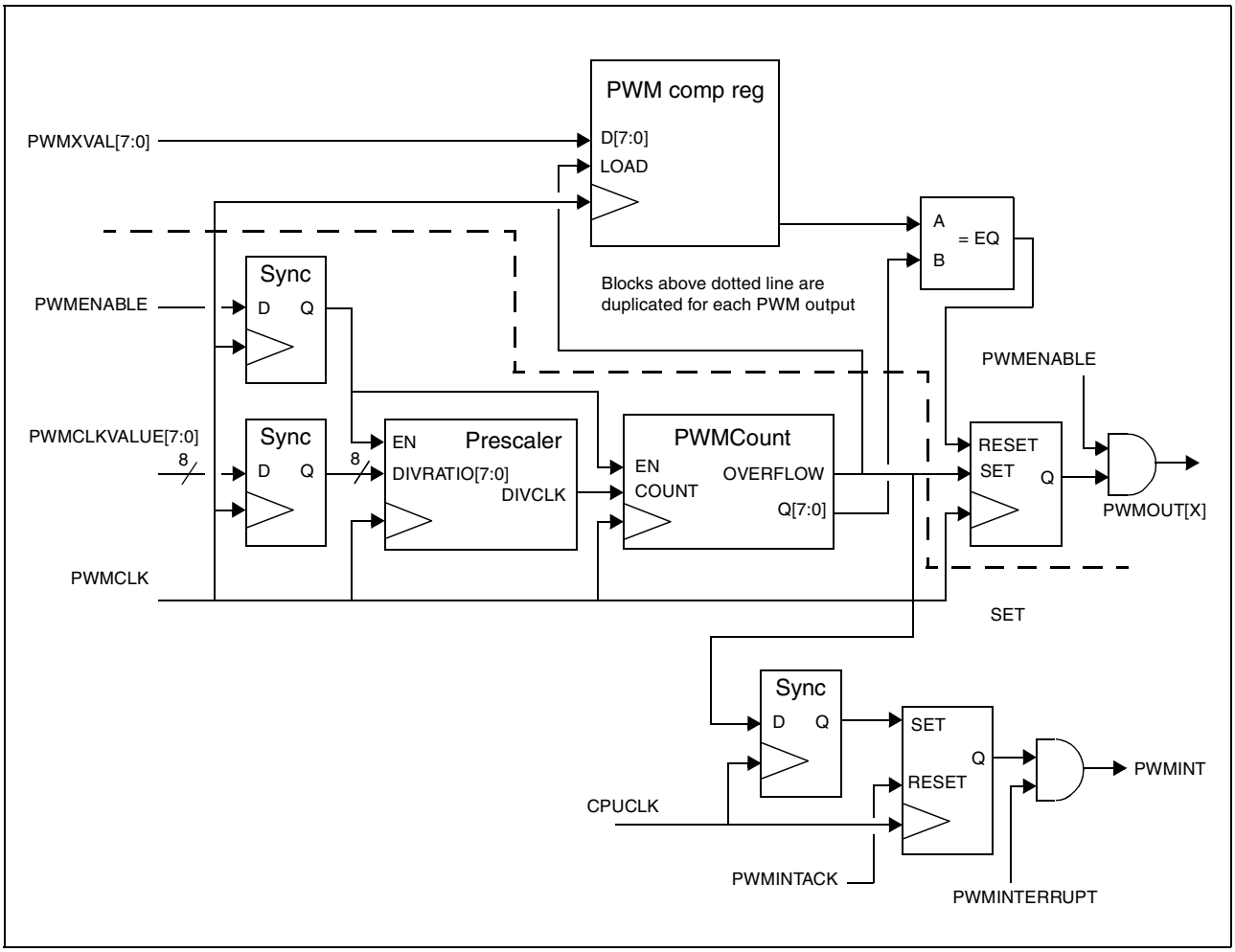
The STx7100 includes an independent PWM-timer module with two identical channels, each containing one programmable timer.

The module includes the following functions:

- generates very low PWM frequencies (typically 411 Hz to 105 kHz for a 100 MHz master clock),
- enables generation of PWM waveforms,
- enables generation of an interrupt on a periodic basis with little software intervention, with a wide periodic range - from a few microseconds to over 100 ms.

There are two completely independent counters with associated prescalers and duty cycle control, each usable either to generate an interrupt or generate a PWM waveform (or both if desired - interrupt and PWM periods are identical).

Figure 267: PWM block diagram



The PWM capture (decoder) **inputs** are PWM_CAPTUREIN0 and PWM_CAPTUREIN1. The encoder **outputs** are PWM_OUT0, PWM_OUT1, PWM_COMPAREOUT0 and PWM_COMPAREOUT1 (see Section 7.2: *Alternative functions on page 54*); the **interrupt requests** (routed to the ILC) are all made through a single signal, PWM_INT. Register PWM_INT_STA indicates which event which caused the interrupt. The module is clocked by two independent clocks, one for capture inputs/timers and one for PWM outputs.

Each capture input can be programmed to detect rising-edge, falling-edge, both edges or neither edge (disabled) by means of register PWM*n*_CPT_EDGE.

76.1 Programmable PWM function

The system clock (100 MHz nominal) is first prescaled (divided) by a factor of 1 to 512 according to control bits PWM*n*_VAL, and triggers an 8-bit counter (from 0 to 255). The counter and prescaler may be stopped by writing 0 to PWM_CTRL.PWM_EN. While disabled, the value in the counter may be read or written from register PWM*n*_CNT. Every 256 counts, the counter triggers the output block to start new pulses.

The prescaler consists of a modulo counter counting from 0 to PWM_CTRL.PWM_CLK_VAL[7:0] and then rolling over to zero (see Figure 267).

For example:
CLK_VAL =

0x00: divide by 1 (that is, generated clock = system clock),
0x02: divide by 3 (modulo-3 count, from 0 to 2),
0x0F: divide by 16 (modulo-16 count, from 0 to 15),
0xFF: divide by 256 (modulo-256 count, from 0 to 255),

When PWM_CTRL.PWM_EN is low, the prescaler is disabled and reset. It starts upon the next rising edge following the setting of the enable bit.

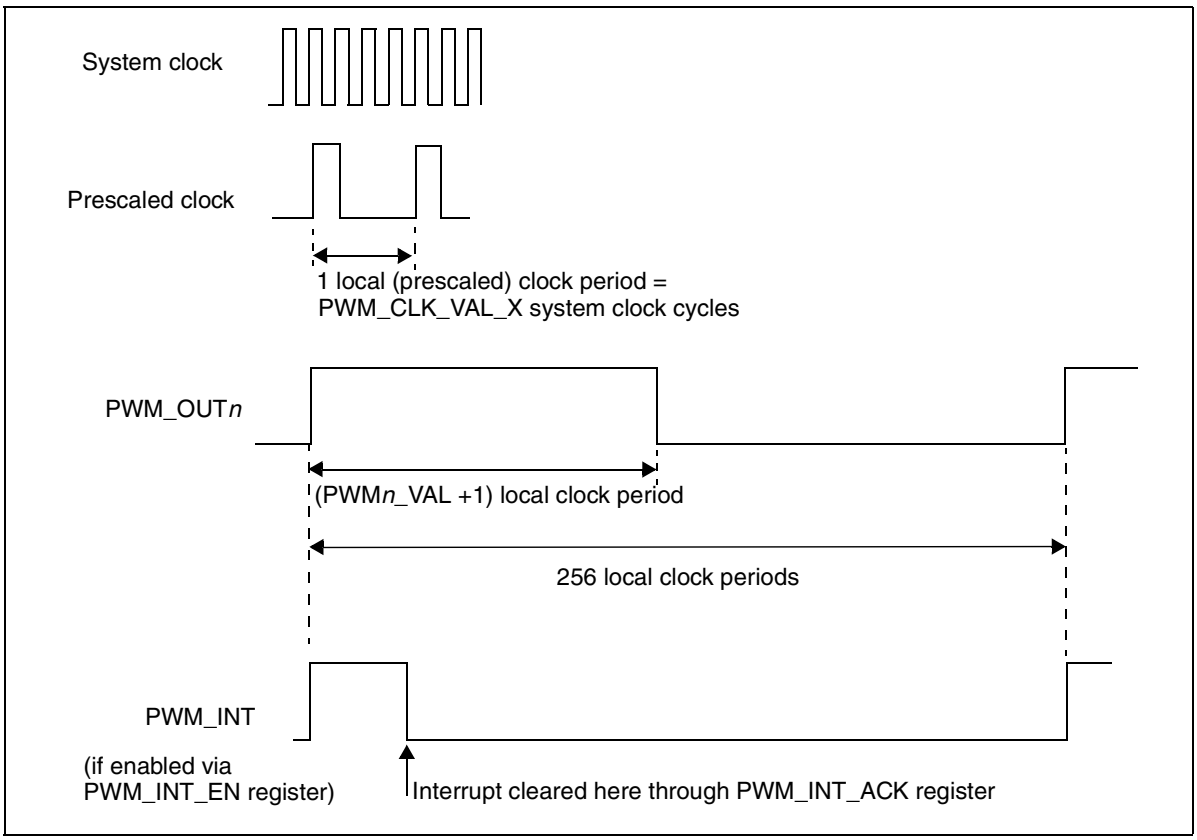
A new PWM pulse is started (PWM_OUT*n* rises to logic 1) every time the 0-to-255 counter rolls over. It returns to logic 0 after the number of cycles programmed in register PWM*n*_VAL + 1. Therefore PWM*n*_VAL controls the duty cycle of the PWM signal. For example, if the value programmed is 127 (that is, half the maximum possible), the resulting output is a 50% duty cycle waveform; if the value programmed is the maximum (255) the pulse will last for all the 256 cycles and the resulting output is constantly high. The length of the pulse is updated only upon the last count, so that the pulse currently executing always finishes before a pulse of different width is output. Following reset, PWM_OUT*n* is low.

Figure 268 below shows the generated waveforms.

76.2 Periodic interrupt generation

Similarly, an interrupt request is raised (PWM_TMR_INT*n* goes high) when the 0-to-255 rolls over, provided bit PWM_INT_EN.EN is set. The interrupt request remains active until cleared by a write access to the corresponding bit in register PWM_INT_ACK (or the interrupt gets disabled). The status of the interrupt is available to software via register PWM_INT_STA.

Figure 268: PWM-timer periodic interrupt generation waveforms



76.3 Capture and compare

The Capture and compare module contains a 32-bit counter, *CaptureCount*, and four 32-bit registers: two (PWM*n*_CPT_VAL) to capture the value of the counter *CaptureValn* and two, PWM*n*_CMP_VAL, to compare with PWM*n*_CPT_VAL, causing an interrupt when the two values are equal. The capture and compare module shares the PWM clock source, which can be prescaled using register PWM_CTRL. This register also contains the capture enable bit, CPT_EN, which is used to enable the counter.

76.3.1 Capture function

It is possible to select which edge to trigger on (a a capture event) by setting PWM*n*_CPT_EDGE. The detection of a capture event for capture register *n* results in the current value of *CaptureCount* being loaded into PWM*n*_CPT_VAL*,* and the CPT*n*_INT bit being set in register PWM_INT_STA. Interrupts are enabled through register PWM_INT_EN. Bits PWM_INT_ACK.CPT*n*_INT are used to reset PWM_INT_STA.CPT*n*_INT.

Note: The capture signals are synchronized to the system clock domain.

76.3.2 Compare function

There are two compare registers, PWM*n*_CMP_VAL, each of which can generate an interrupt when the values of *CaptureCount* and PWM*n*_CMP_VAL are equal.

77 PWM and counter registers

The base address for the PWM-timer, referred to as *PWMTimerBaseAddress*, is:
0x1801 0000.

Note: The PWM-timer is software compatible with PWM modules present on earlier ST MPEG decoders in the sense that a routine running on such PWM modules and exploiting only PWM features (not capture of interrupts) should be transposable as is to run on this PWM-timer.

Table 217: PWM-timer register summary

Register	Description	Offset	Type
PWM0_VAL	PWM 0 reload value	0x00	R/W
PWM1_VAL	PWM 1 reload value	0x04	
PWM0_CPT_VAL	PWM 0 capture value	0x10	RO
PWM1_CPT_VAL	PWM 1 capture value	0x14	
PWM0_CMP_VAL	PWM 0 compare value	0x20	R/W
PWM1_CMP_VAL	PWM 1 compare value	0x24	
PWM0_CPT_EDGE	PWM 0 capture edge control	0x30	
PWM1_CPT_EDGE	PWM 1 capture edge control	0x34	
PWM0_CMP_OUT_VAL	PWM 0 compare output value	0x40	
PWM1_CMP_OUT_VAL	PWM 1 compare output value	0x44	
PWM_CTRL	PWM control	0x50	
PWM_INT_EN	PWM interrupt enable	0x54	
PWM_INT_STA	PWM interrupt status	0x58	RO
PWM_INT_ACK	PWM interrupt acknowledge	0x5C	WO
PWM_CNT	PWM count	0x60	R/W
PWM_CPT_CMP_CNT	PWM capture/compare counter	0x64	

PWM0_VAL PWM 0 reload value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PWM0_VAL							

Address: *PWMTimerBaseAddress* + 0x00
Type: R/W
Reset:
Description: PWM0 reload value, defining the duty cycle of output PWM0_OUT as follows:
PWM0_VAL + 1 is the number of local (prescaled) clock cycles for which PWM0_OUT is high in a period of 256 local (prescaled) clock cycles.

PWM1_VAL PWM 1 reload value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PWM1_VAL							

Address: *PWMTimerBaseAddress* + 0x08

Type: R/W

Reset:

Description: PWM1 reload value, defining the duty cycle of output PWM1_OUT as follows:

PWM1VAL + 1 is the number of local (prescaled) clock cycles for which PWM1_OUT is high in a period of 256 local (prescaled) clock cycles.

PWM0_CPT_VAL PWM 0 capture value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT_VAL																															

Address: *PWMTimerBaseAddress* + 0x

Type: RO

Reset:

Description: Value of capture counter 0 when a capture event occurs.

PWM1_CPT_VAL PWM 1 capture value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT_VAL																															

Address: *PWMTimerBaseAddress* + 0x

Type: RO

Reset:

Description: Value of capture counter 1 when a capture event occurs.

PWM0_CMP_VAL PWM 0 compare value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_VAL																															

Address: *PWMTimerBaseAddress* + 0x

Type: R/W

Reset:

Description: When the values of PWM0_CPT_VAL and this register are equal, an interrupt is triggered.

PWM1_CMP_VAL PWM 1 compare value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_VAL																															

Address: *PWMTimerBaseAddress* + 0x
Type: R/W
Reset:
Description: When the values of PWM1_CPT_VAL and this register are equal, an interrupt is triggered.

PWM0_CPT_EDGE PWM 0 capture edge control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														CE	

Address: *PWMTimerBaseAddress* + 0x
Type: R/W
Reset:
Description: Controls the edge used for the capture of the timer in register PWM0_CPT_VAL.
00: Disabled
01: Rising edge
10: Falling edge
11: Rising or falling edge

PWM1_CPT_EDGE PWM 1 capture edge control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														CE	

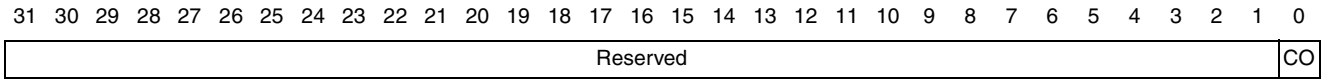
Address: *PWMTimerBaseAddress* + 0x
Type: R/W
Reset:
Description: Controls the edge used for the capture of the timer in register PWM1_CPT_VAL.
00: Disabled
01: Rising edge
10: Falling edge
11: Rising or falling edge

PWM0_CMP_OUT_VAL PWM 0 compare output value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															CO

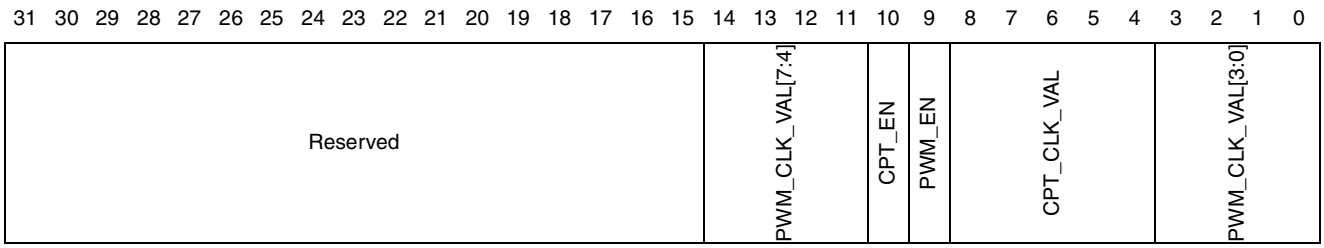
Address: *PWMTimerBaseAddress* + 0x
Type: R/W
Reset:
Description: On the next compare event, this value is output on the COMPAREOUT0 signal.

PWM1_CMP_OUT_VAL PWM 1 compare output value



Address: *PWMTimerBaseAddress* + 0x
Type: R/W
Reset:
Description: On the next compare event, this value is output on the COMPAREOUT1 signal.

PWM_CTRL PWM control



Address: *PWMTimerBaseAddress* + 0x50
Type: R/W
Reset:
Description:

[31:15] **Reserved**

[14:11] **PWM_CLK_VAL[7:4]**
High order bits of the parameter that defines the period of the local prescaled clock for the PWM-timer. The local clock enable signal is generated upon the prescale counter reaching PWM_CLK_VAL[7:0]. See

[10] **CPT_EN**
0: Disable capture
1: Enable capture

[9] **PWM_EN**
0: Prescale counter is cleared and PWM counter is stopped
1: Prescale counter and PWM counter are enabled

[8:4] **CPT_CLK_VAL**: Capture counter clock prescale value

[3:0] **PWM_CLK_VAL[3:0]**

PWM_INT_EN PWM interrupt enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									CMP1_INT_EN	CMP0_INT_EN	Reserved		CPT1_INT_EN	CPT0_INT_EN	EN

Address: *PWMTimerBaseAddress* + 0x54

Type: R/W

Reset:

Description:

[31:7] **Reserved**

[6] **CMP1_INT_EN**: Enable compare 1 interrupt

[5] **CMP0_INT_EN**: Enable compare 0 interrupt

[4:3] **Reserved**

[2] **CPT1_INT_EN**: Enable capture 1 interrupt

[1] **CPT0_INT_EN**: Enable capture 0 interrupt

[0] **EN**: PWM-timer interrupt enable
0: Interrupts from the PWM-timer are disabled.
1: The PWM-timer generates a high level interrupt as counter 0 rolls over.

PWM_INT_STA PWM interrupt status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									CMP1_INT	CMP0_INT	Reserved		CPT1_INT	CPT0_INT	PWM_INT

Address: *PWMTimerBaseAddress* + 0x58

Type: RO

Reset:

Description:

[31:7] **Reserved**

[6] **CMP1_INT**: Compare 1 interrupt

[5] **CMP0_INT**: Compare 0 interrupt

[4:3] **Reserved**

[2] **CPT1_INT**: Capture 1 interrupt

[1] **CPT0_INT**: Capture 0 interrupt

[0] **PWM_INT**: PWM counter interrupt

PWM_INT_ACK PWM interrupt acknowledge

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									CMP1_INT	CMP0_INT	Reserved			CPT1_INT	CPT0_INT	PWM_INT

Address: *PWMTimerBaseAddress* + 0x5C

Type: WO

Reset:

Description: Write 1 to clear interrupt.

[31:2] **Reserved**

[6] **CMP1_INT**: Compare 1 interrupt acknowledge

[5] **CMP0_INT**: Compare 0 interrupt acknowledge

[4:3] **Reserved**

[2] **CPT1_INT**: Capture 1 interrupt acknowledge

[1] **CPT0_INT**: Capture 0 interrupt acknowledge

[0] **PWM_INT**: PWM counter interrupt acknowledge

PWM_CNT PWM count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PWM_CNT							

Address: *PWMTimerBaseAddress* + 0x60

Type: R/W (see text)

Reset:

Description:

[31:8] **Reserved**

[7:0] **PWM_CNT**: Direct access to the PWM counter
Write access (to preset a value for example) is only possible when the PWM-timer is disabled (PWM_CTRL.PWM_EN = 0).

PWM_CPT_CMP_CNT PWM capture/compare counter

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT_CMP_CNT																															

Address: *PWMTimerBaseAddress* + 0x

Type: R/W

Reset:

Description: Counter used in capture and compare mode. Unlike the PWM counter, this can be accessed when the counter is enabled.

78 Serial ATA (SATA) subsystem

78.1 Glossary

ATA	AT attachment
SATA	Serial ATA interface
ATAPI	AT attachment packet interface
FIS	Frame information structure
DWORD	SATA work size, 32 bits
AHB	Advanced high performance bus
DMA	Direct memory access
PHY	SATA physical layer
LL	Logical link layer
TL	Transport layer
BIU	Bus interface unit
IPF	Interrupt pending flag
ICM	Interconnect matrix

78.2 References

The SATA interface is based on a host controller block from Synopsys, Inc. For full details, see the following Synopsys documents:

- *DesignWare SATA Host Core Databook*
- *DesignWare DW_ahb_dmac Databook*

The interface is compliant with the serial ATA specifications for SATA II. See:

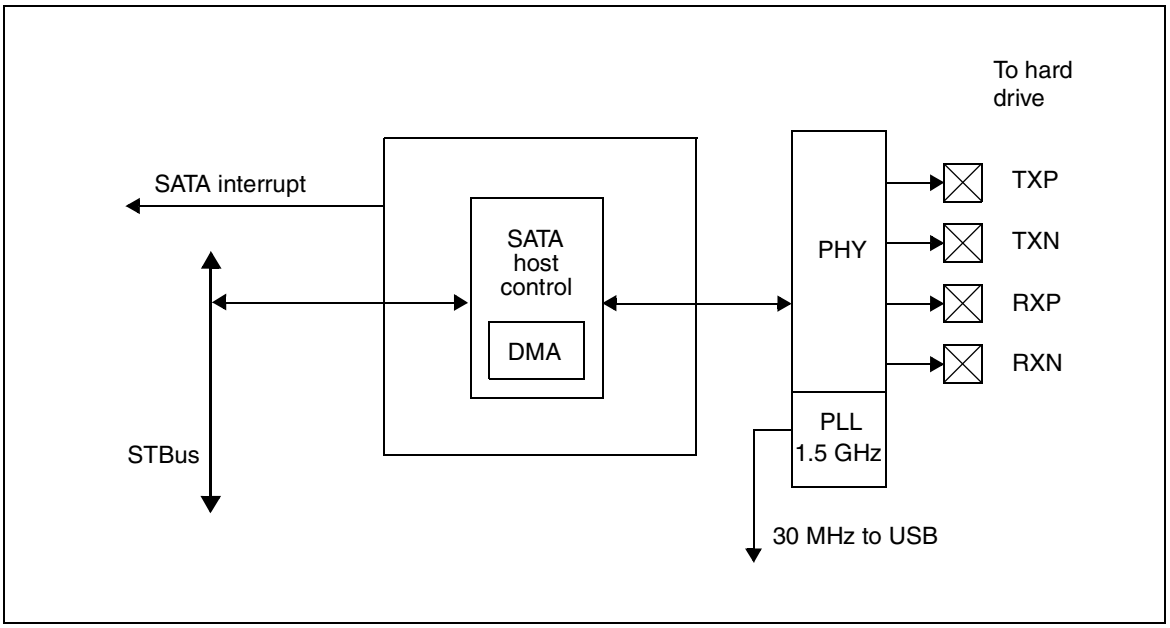
- Serial ATA II: *Extensions to serial ATA 1.0a specification,*
- Serial ATA 1.0a: *Serial ATA 1.0a specification.*

Both these specifications are available from www.serialata.org.

PCB layout recommendations for SATA signals are given in Chapter 14: *PCB layout recommendations on page 108*.

78.3 Overview

Figure 269: SATA host block diagram



The STx7100 integrates a single serial ATA (SATA) host and PHY for a glueless interface to a SATA hard disk drive for DVR applications. The SATA host interface features:

- integrated PHY, configured to perform 8 - 10 bit encode/decode, with 16-bit decoded data interfaces to the SATA controller,
- 1.5 GHz PLL,
- SATA controller with integrated DMA engine.

78.3.1 External interface

The SATA host signals are described in Table 218.

Table 218: SATA interface signals

Signal	Dir	Function
SATAXTAL1[a]	I/O	In oscillator mode, input pin for 30 MHz quartz. REFCLKN differential input when BYPASS_OSC is high
SATAXTAL2[a]	I/O	In oscillator mode, input pin for 30 MHz quartz. REFCLKP differential input when BYPASS_OSC is high
SATATXP	O	Positive differential output for serialized data on transmit (high when data bit at 1)
SATATXN	O	Negative differential output for serialized data on transmit (low when data bit at 1)
SATARXN	I	Negative differential input for serialized data on receive (low when data bit at 1)
SATARXP	I	Positive differential input for serialized data on receive (high when data bit at 1)
SATAVSSREF	I/O	External reference resistor is connected between this pin and VSSREF to provide a resistor reference for the compensation block.

a. Not brought out to a pin on the STx7100.

78.3.2 DMA data transfers

The embedded host controller is coupled to the STBus via a local DMA controller that provides bulk data transfers between the SATA host and the external DDR memory. The data transfer sequence is listed below.

1. The SATA host initializes and enables the DMA controller for a given transfer.
2. The SATA host issues a bulk transfer command to the DMA controller.
3. The command is executed and SATA host is notified that data is ready to be sent or received.
4. During a read operation, data is read from the external HDD and written to the external DDR memory via the host controller.
5. During a write operation, a DMA activate or a DMA setup, the SATA host sends data to the external HDD.
6. During such a transfer the host controller is in a BUSY state
7. The host controller returns to IDLE when all of the data block has been transmitted or received.

78.3.3 SATA host controller

The SATA host controller operates in three modes.

- **Transmit**: the DMA writes to the host controller when the host controller's transmit FIFO has space available. After receiving a write request from the DMA controller, the SATA host controller receives data from the STBus in one block chunks. While a data block is being received the SATA host controller is in a BUSY state.
- **Receive**: The DMA reads from the host controller when the host controller's receive FIFO starts to fill. The STBus is requested and the DMA controller transfers data from the SATA host controller to the STBus in one block chunks. While a data block is being received the SATA host controller is in a BUSY state.
- **PIO mode**: in SATA PIO mode data is read from and written to the STBus.

78.4 Low power management

Power states are controlled by the host driver and the STx7100. The SATA host interface supports the following three power states.

- **PHY ready**: the PHY and the main PLL are both active. The interface is synchronized and ready to receive and send data.
- **Partial**: the PHY is powered but is in a reduced power state. Both signals on the interface are in a neutral state. The exit latency from this state is not longer than 10 µs. High selects partial power management mode. The cable interface is quiet (no differential transitions). The SRC_CLK is running at 30 MHz.
- **Slumber**: the PHY is powered but in a reduced power state. Both signals on the interface are in a neutral state. The exit latency from this state is not longer than 10 ms. High selects low power management mode. The cable interface is quiet (no differential transitions) and the PHY is shut down. The SRC_CLK runs at 30 MHz.

78.5 Interrupt management

The SATA host interface outputs two interrupts.

- **DMA interrupt request (SATA_DMA)**: This signal is asserted when any unmasked DMA interrupt status register is set.
- **Host controller interrupt request (SATA_HC)**: This signal is asserted when any unmasked SATA host interrupt pending register is set.

79 Serial ATA (SATA) DMA

79.1 Configuration Parameters

Table 219: SATA DMA parameters

Label	Parameter definition
Global SATA DMA configuration	
Number of AHB master interfaces	**Parameter name**: DMAH_NUM_MASTER_INT **Values**: 1 to 4 **Default Value**: 1 **Dependencies**: None **Description**: Creates the specified number of AHB master interfaces. A channel source or destination device can be programmed to be on any of the configured AHB layers attached to the AHB master interface. This setting determines if a master interface signal set is present on the I/O or not. AHB master interface 1 signals are always present.
Number of DMA channels	**Parameter name**: DMAH_NUM_CHANNELS **Values**: 1 to 8 **Default Value**: 1 **Dependencies**: None **Description** : Creates the specified number of SATA DMA channels. Each channel is uni-directional and transfers data from the channel source to the channel destination. The channel source and destination AHB layer, system address, and handshaking interface are under software control
Number of handshaking interfaces	**Parameter name**: DMAH_NUM_HS_INT **Values**: 0 to 16 **Default Value**: 2 **Dependencies**: None **Description:** Creates the specified number of handshaking interfaces. You can program the SATA DMA to assign a handshaking interface for each channel source and destination (refer toSADMA_CFG0 on *page 1027*). If 0 is selected, then no hardware handshaking signals are present on the I/O.
IN number	**Parameter name**: DMAH_ID_NUM **Values**: 0x0 to 0XFFFF FFFF **Default Value**: 0x0 **Dependencies**: None **Description:** This 32-bit value is hardwired and read back by a read to the SATA DMA ID Register (DmaldReg).
Interrupts are active low?	**Parameter name**: DMAH_INTR_POL **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description:** When set to1, the interrupt polarity for all interrupt signals on the interface become active-low

Table 219: SATA DMA parameters

Label	Parameter definition
Interrupt pins to appear as outputs	**Parameter name**: DMAH_INTR_IO **Values**: ALL (0), TYPE (1), or COMBINED (2) **Default Value**: COMBINED (2) **Dependencies**: None **Description**: Selects which interrupt-related signals appear as outputs on the design. **All**: Contents of all five Interrupt Status registers appear as output signals; 5 * DMAH_NUM_CHANNELS bits wide. **Type**: All internal interrupts are combined by type, and appear on int_flag(_n) 5-bit bus. bit[4] – Error interrupt bit[3] – Destination transaction complete interrupt bit[2] – Source transaction complete interrupt bit[1] – Block complete interrupt bit[0] – Transfer complete interrupt Also, the bitwise OR of all bits of the int_flag(_n) bus is driven onto the int_combined(_n) single bit output. **Combined**: Bitwise OR of all bits of the int_flag(_n) bus is driven onto the int_combined(_n) single bit output.
Add encoded parameters Parameter	**Parameter name**: DMAH_ADD_ENCODED_PARAMS **Values**: True (1) or False (0) **Default Value**: True (1) **Dependencies**: None **Description**: Adding the encoded parameters gives firmware an easy and quick way of identifying the DesignWare component within an I/O memory map. Some critical design-time options determine how a driver should interact with the peripheral. There is a minimal area overhead when you include these parameters. Additionally, this option allows a self-configurable single driver to be developed for each component.
Allow the bus burst length to be limited to a programmable maximum value?	**Parameter name**: DMAH_MABRST **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description**: When set to 1, you can limit the maximum bus burst length to a value under software control by writing to the channel configuration register; this is a global parameter for all of the configured master interfaces. Setting this option to 0 allows for some logic optimization. *Note: When set to 1, you can limit the maximum bus burst length to a value under software control by writing to the channel configuration register; this is a global parameter for all of the configured master interfaces. Setting this option to 0 allows for some logic optimization.*
Allow the slave interface to return an error response when an illegal access is attemprted?	**Parameter name**: DMAH_RETURN_ERR_RESP **Values**: True (1) or False (0) **Default Value**: True (1) **Dependencies**: None **Description**: Setting this option to 1 allows the slave interface to return an error response on the hresp bus when an illegal access is attempted over the AHB slave interface. For a list of illegal accesses, refer to *Section 79.2.2: Illegal register access on page 921*. Setting this option to 0 results in an OKAY response being returned on the hresp bus for all of the illegal accesses.

Table 219: SATA DMA parameters

Label	Parameter definition
Configuration of bus layers	
Global bus configuration	
System is big endian?	**Parameter name**: DMAH_BIG_ENDIAN **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description**: The AHB master interfaces and AHB slave interfaces can be configured to exist in either a big- or littleendian system. All AHB master interfaces and the AHB slave interface have the same endianness. When set to 1, all master interfaces and the slave interface become big-endian.
Slave interface data bus width	**Parameter name**: DMAH_S_HDATA_WIDTH **Values**: 32, 64, 128, 256 bits **Default Value**: 32 **Dependencies**: None **Description**: Specifies the data bus width for the AHB slave interface.
Layer-specific configuration	
Is Layer 1 bus-Lite?	**Parameter name**: DMAH_M1_AHB_LITE **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: DMAH_NUM_MASTER_INT >= 1 **Description**: Select True (1) if master interface 1 is the only master interface on this bus layer. If this is the case, then this is an bus-Lite layer.
Master 1 interface data bus width	**Parameter name**: DMAH_M1_HDATA_WIDTH **Values**: 32, 64, 128, or 256 bits **Default Value**: 32 bits **Dependencies**: DMAH_NUM_MASTER_INT >= 1 **Description**: AHB master 1 interface data bus width.
Is Layer 2 bus-Lite?	**Parameter name**: DMAH_M2_AHB_LITE **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: Option is available if DMAH_NUM_MASTER_INT >= 2. **Description**: Select True (1) if master interface 2 is the only master interface on this bus layer. If this is the case, then this is an bus-Lite layer
Master 2 interface databus width	**Parameter name**: DMAH_M2_HDATA_WIDTH **Values**: 32, 64, 128, or 256 bits **Default Value**: 32 bits **Dependencies**: Option is available if DMAH_NUM_MASTER_INT >= 2. **Description**: AHB master 2 interface data bus width.
Is Layer 3 bus-Lite?	**Parameter name**: DMAH_M3_AHB_LITE **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: Option is available if DMAH_NUM_MASTER_INT >= 3 **Description**: Select Yes (1) if master interface 3 is the only master interface on this bus layer. If this is the case, then this is an bus-Lite layer.
Master 3 interface databus width	**Parameter name**: DMAH_M3_HDATA_WIDTH **Values**: 32, 64, 128, or 256 bits **Default Value**: 32 bits **Dependencies**: Option is available if DMAH_NUM_MASTER_INT >= 3. **Description**: AHB master 3 interface data bus width.

Table 219: SATA DMA parameters

Label	Parameter definition
Is Layer 4 bus-Lite?	**Parameter name**: DMAH_M4_AHB_LITE **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: Option is available if DMAH_NUM_MASTER_INT >= 4 **Description**: Select Yes (1) if master interface 4is the only master interface on this bus layer. If this is the case, then this is an bus-Lite layer
Master 3 interface databus width	**Parameter name**: DMAH_M4_HDATA_WIDTH **Values**: 32, 64, 128, or 256 bits **Default Value**: 32 bits **Dependencies**: Option is available if DMAH_NUM_MASTER_INT >= 4 **Description**: AHB master 4 interface data bus width.
Channel configuration	
Channel x configuration, where x = 0 t- 7 (STx7100 has one channel only, channel 0)	
Channel x FIFO depth in bytes	**Parameter name**: DMAH_CHx_FIFO_DEPTH **Values**: 8, 16, 32, 64, 128 bytes **Default Value**: 16 bytes **Dependencies**: None **Description**: Channel x FIFO depth in bytes.
Maximum value of burst transaction size	**Parameter name**: DMAH_CHx_MAX_MULT_SIZE **Values**: 4, 8, 16, 32, 64, 128, and 256 **Default Value**: 8 **Dependencies**: None **Description**: Maximum value of burst transaction size that can be programmed (SADMA_CTRL0.SRC_MSIZE and SADMA_CTRL0.DEST_MSIZE). Limiting DMAH_CHx_MAX_MULT_SIZE to a maximum value will allow for some logic optimization of the implementation.
Maximum block size in source transfer widths	**Parameter name**: DMAH_CHx_MAX_BLK_SIZE **Values**: 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047, 4095 **Default Value**: 31 **Dependencies**: None **Description**: The description of this parameter is dependent on what is assigned as the flow controller. SATA DMA flow controller: Maximum block size, in multiples of source transfer width, that can be programmed. A programmed value greater than this will result in erroneous behavior. Source/destination assigned as flow controller: In this case, the blocks can be greater than DMAH_CHx_MAX_BLK_SIZE in size, but the logic that keeps track of the size of a block saturates at DMAH_CHx_MAX_BLK_SIZE. This does not result in erroneous behavior, but a readback by software of the block size is incorrect when the block size exceeds the saturated value. This parameter is also used to limit the gather and scatter count registers to a maximum value of DMAH_CHx_MAX_BLK_SIZE. Limiting DMAH_CHx_MAX_BLK_SIZE to a maximum value allows some logic optimization of the implementation.
Hardcode channel x's flow control device to allow for logic optimization	**Parameter name**: DMAH_CHx_FC **Values**: DMA_FC_ONLY (0), SRC_FC_ONLY (1), DST_FC_ONLY (2), or ANY_FC (3) **Default Value**: DMA_FC_ONLY(0) **Dependencies**: None **Description**: Hardcodes the flow control peripheral for the channel. If ANY_FC is selected, then the flow control device is not hardcoded, and software selects the flow control device for a DMA transfer. Hardcoding the flow control device allows some logic optimization of the implementation.

Table 219: SATA DMA parameters

Label	Parameter definition
Include logic to enable channel or bus locking on channel x?	**Parameter name**: DMAH_CHx_LOCK_EN **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description**: If set to 1, includes logic to enable channel or bus-level locking on channel x. When set to 1, then software can program the SATA DMA to assert hlock over the DMA transfer, block transfer, or transaction. When set to 1, then software can also program the SATA DMA to lock the arbitration for the master bus interface over the DMA transfer, block transfer, or transaction. Disabling this option allows some logic optimization of the implementation.
Hardcode the master interface attached to the source of channel x	**Parameter name**: DMAH_CHx_SMS **Values**: MASTER_1 (0), MASTER_2 (1), MASTER_3 (2), MASTER_4 (3), NO_HARDCODE (4) **Default Value**: MASTER_1 (0) **Dependencies**: DMAH_NUM_MASTER_INT > 1 **Description**: Hardcode the AHB master interface attached to the source of channel x. If this is not hardcoded, then software can program the source of channel x to be attached to any of the configured layers. Hardcoding this value allows some logic optimization of the implementation.
Hardcode the master interface attached to the destination of channel x	**Parameter name**: DMAH_CHx_DMS **Values**: MASTER_1 (0), MASTER_2 (1), MASTER_3 (2), MASTER_4 (3), NO_HARDCODE (4) **Default Value**: MASTER_1 (0) **Dependencies**: DMAH_NUM_MASTER_INT > 1 **Description**: Hardcode the AHB master interface attached to the channel x destination. If this is not hardcoded, then software can program the destination of channel x to be attached to any of the configured layers. Hardcoding this value allows some logic optimization of the implementation.
Hardcode channel x's source transfer width	**Parameter name**: DMAH_CHx_STW **Values**: BYTE (8), HALFWORD (16), WORD (32), TWO_WORD (64), FOUR_WORD (128), EIGHT_WORD (256), or NO HARDCODE (0) **Default Value**: WORD (32) **Dependencies**: None **Description**: Hardcode the source transfer width for transfers from the source of channel x. If this is not hardcoded, then software can program the source transfer width. Hardcoding the source transfer width allows some logic optimization of the implementation.
Hardcode channel x's destination transfer width	**Parameter name**: DMAH_CHx_DTW **Values**: BYTE (8), HALFWORD (16), WORD (32), TWO_WORD (64), FOUR_WORD (128), EIGHT_WORD (256), or NO HARDCODE (0 **Default Value**: WORD (32) **Dependencies**: None **Description**: Hardcode the destination transfer width for transfers from the destination of channel x. If this is not hardcoded, then software can program the destination transfer width. Hardcoding the destination transfer width allows some logic optimization of the implementation.
Can the source of channel x return a non-OK response on hresp?	**Parameter name**: DMAH_CHx_SRC_NON_OK **Values**: True (1) or False (0) **Default Value**: True (1) **Dependencies**: None **Description**: Set this parameter to 1 if the source peripheral attached to channel x can return a non-OK response on hresp, such as a SPLIT, RETRY, or ERROR response. If set to 0, then hardware assumes that only OK responses are returned from the source peripheral attached to the channel. Setting this to parameter to 0 allows some logic optimization of the implementation.

Table 219: SATA DMA parameters

Label	Parameter definition
Can the destination of channel x return a non-OK response on hresp?	**Parameter name**: DMAH_CHx_DST_NON_OK **Values**: True (1) or False (0) **Default Value**: True (1) **Dependencies**: None **Description**: Set this parameter to 1 if the destination peripheral attached to channel x can return a non-OK response on hresp, such as a SPLIT, RETRY, or ERROR response. If set to 0, then hardware assumes that only OK responses are returned from the destination peripheral attached to the channel. Setting this to parameter to 0 allows some logic optimization of the implementation.
Include logic to enable multi-block DMA transfers on channel x?	**Parameter name**: DMAH_CHx_MULTI_BLK_EN **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description**: Includes or excludes logic to enable multi-block DMA transfers on channel x. If this option is set to 0, then hardware hardwires channel x to perform only single block transfers. Setting this parameter to 0 allows some logic optimization of the implementation.
Hardcode Channel x LLP register to 0?	**Parameter name**: DMAH_CHx_HC_LLP **Values**: True (1) or False (0) **Default Value**: True (1) **Dependencies**: (DMAH_CHx_MULTI_BLK_TYPE == 0 or DMAH_CHx_MULTI_BLK_TYPE >= 4) **Description**: If set to 1, hardcodes channel x Linked List Pointer register to 0 (SADMA_LLP0.LOC == 0), which disables the following features: 1. Multi-block support through block chaining 2. Source and destination status fetch 3. Control and source/destination status write back Multi-block DMA transfers can still be enabled on channel x through auto-reloading of channel registers. Hardcoding this value allows some logic optimization of the implementation.
Hardcode the master interface attached to the LLP peripheral of channel x	**Parameter name**: DMAH_CHx_LMS **Values**: MASTER_1 (0), MASTER_2 (1), MASTER_3 (2), MASTER_4 (3), NO_HARDCODE (4) **Default Value**: MASTER_1 (0) **Dependencies**: DMAH_CHx_HC_LLP == 0 and DMAH_NUM_MASTER_INT > 1 and (DMAH_CHx_MULTI_BLK_TYPE == 0 or DMAH_CHx_MULTI_BLK_TYPE >= 4) **Description**: Hardcode the AHB master interface attached to the peripheral that stores the LLI information (Linked List Item). If this is not hardcoded, then software can program the peripheral that stores the LLI information of channel x to be attached to any of the configured layers. Hardcoding this value allows some logic optimization of the implementation.

Table 219: SATA DMA parameters

Label	Parameter definition
Choose type of multi-blocks to be supported?	**Parameter name**: DMAH_CHx_MULTI_BLK_TYPE **Values**: NO_HARDCODE (0): Allow all types of multi-support CONT_RELOAD (1): Allow only multi-block transfers where SAR0 is contiguous; DAR0 and CTL0 are reloaded from their initial values. RELOAD_CONT (2): Allow only multi-block transfers where SAR0 and CTL0 are reloaded from their initial values; DAR0 is contiguous. RELOAD_RELOAD (3): Allow only multi-block transfers where SAR0, DAR0, and CTL0 are reloaded from their initial values. CONT_LLP (4): Allow only multi-block transfers where SAR0 is contiguous; DAR0, CTL0, and LLP0 are loaded from the next linked list item. RELOAD_LLP (5): Allow only multi-block transfers where SAR0 is reloaded from it's initial value; DAR0, CTL0, and LLP0 are loaded from the next linked list item. LLP_CONT (6): Allow only multi-block transfers where SAR0, CTL0, and LLP0 are loaded from the next linked list item. LLP_RELOAD (7): Allow only multi-block transfers where SAR0, CTL0, and LLP0 are loaded from the next linked list item; DAR0 is reloaded from it's initial values. LLP_LLP (8): Allow only multi-block transfers where SAR0, DAR0, CTL0, and LLP0 are loaded from the next linked list item. **Default Value**: NO_HARDCODE **Dependencies**: DMAH_CHx_MULTI_BLK_EN = 1 **Description**: This parameter allows you to hardcode the type of multi-block transfers that SATA DMA can perform. This results in some logic optimization of the implementation.
Fetch status from destination of channel x?	**Parameter name**: DMAH_CHx_STAT_DST **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: DMAH_CHx_HC_LLP==0 **Description**: Include or exclude logic to fetch a status register from the destination peripheral of channel x and write this status information to system memory at the end of each block transfer. Disabling this feature allows some logic optimization of the implementation.
Fetch status from source of channel x?	**Parameter name**: DMAH_CHx_STAT_SRC **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: DMAH_CHx_HC_LLP == 0 **Description**: Include or exclude logic to fetch a status register from source peripheral of channel x and write this status information to memory at end of each block transfer. Disabling this feature allows some logic optimization of the implementation.
Can the LLP peripheral of channel x return a non-OK response on hresp?	**Parameter name**: DMAH_CHx_LLP_NON_OK **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: DMAH_CHx_HC_LLP == 0 and (DMAH_CHx_MULTI_BLK_TYPE == 0 or DMAH_CHx_MULTI_BLK_TYPE >= 4) **Description**: Set this parameter to 1 if the LLP peripheral attached to channel x can return a non-OK response on hresp, such as a SPLIT, RETRY, or ERROR response. If set to 0, then hardware assumes that only OK responses are returned from the LLP peripheral attached to the channel. Setting this parameter to 0 allows some logic optimization of the implementation.

Table 219: SATA DMA parameters

Label	Parameter definition
Include logic to enable control register writeback after each block transfer?	**Parameter name**: DMAH_CHx_CTL_WB_EN **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: DMAH_CHx_HC_LLP == 0 and DMAH_CHx_STAT_SRC = 0 and DMAH_CHx_STAT_DST = 0 **Description**: Include or exclude logic to enable writeback of the control register at the end of every block transfer. Disabling this feature allows some logic optimization of the implementation.
Include logic to enable 'scatter' feature on channel x?	**Parameter name**: DMAH_CHx_DST_SCA_EN **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description**: Includes or excludes logic to enable the scatter feature on channel x. Setting this option to 0 allows some logic optimization of the implementation.
Include logic to enable the 'gather' feature on channel x?	**Parameter name**: DMAH_CHx_SRC_GAT_EN **Values**: True (1) or False (0) **Default Value**: False (0) **Dependencies**: None **Description**:
	Parameter name: **Values**: **Default Value**: **Dependencies**: None **Description**: Includes or excludes logic to enable the gather feature on channel x. Setting this option to 0 allows some logic optimization of the implementation.
	Parameter name: **Values**: **Default Value**: **Dependencies**: None **Description**:

79.2 Programming Interface

This section includes information on how to program the SATA DMA, grouped in the following categories:

- Register Access
- Illegal Register Access
- SATA DMA Transfer Types
- Programming a Channel
- Disabling a Channel Prior to Transfer Completion
- Memory Map of SATA DMA
- Register Descriptions
- Channel Registers
- Interrupt Registers
- Software Handshaking Registers
- Miscellaneous SATA DMA Registers

Note: There are references to both software and hardware parameters throughout this chapter. The software parameters are the field names in each register description table and are prefixed by the register name; for example, the Block Transfer Size field in the Control Register is designated as "SADMA_CTRL0.BLOCK_TS."

79.2.1 Register access

All registers are aligned to a 64-bit boundary and are 64 bits wide. In general, the upper 32 bits of a register are reserved. A write to reserved bits within the register is ignored. A read from reserved bits in the register reads back 0. To avoid address aliasing, do one of the following:

1. The SATA DMA should not be allocated more than 1 Kb of address space in the system memory map. If it is, then addresses selected above 1 Kb from the base address are aliased to an address within the 1 Kb space, and a transfer takes place involving this register.
2. Software should not attempt to access non-register locations when hsel is asserted.

Note: The hsel signal is asserted by the system decoder when the address on the bus is within the system address assigned for SATA DMA.

79.2.2 Illegal register access

An illegal access can be any of the following:

1. An bus transfer of hsize greater than 64 is attempted.
2. HSEL is asserted, but the address does not decode to a valid address.
3. A write to the SADMA_ ... SAR0, DAR0, LLP0, CTRL0, SSTATx, DSTATx, SSTATAR0, DSTATARx, SGR0, or DSR0 registers occurs when the channel is enabled.
4. A read from the ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr is attempted.
5. A write to the StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr is attempted.
6. A write to the StatusInt register is attempted.
7. A write to either the DmaIdReg or DMA Component ID Register register is attempted.

The response to an illegal access is configured using the configuration parameter

DMAH_RETURN_ERR_RESP. When DMAH_RETURN_ERR_RESP is set to True, an illegal access (read/write) returns an error response.

If DMAH_RETURN_ERR_RESP is set to False, an OKAY response is returned, a read reads back 0x0, and a write is ignored.

79.2.3 SATA DMA transfer types

A DMA transfer may consist of single or multi-block transfers. On successive blocks of a multi-block transfer, the SAR0/DAR0 register in the SATA DMA is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the SADMA_CTRL0 register in the SATA DMA is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using Linked Lists is the multi-block method of choice. On successive blocks, the SADMA_LLP0 register in the SATA DMA is re-programmed using block chaining with linked lists

A block descriptor consists of six registers: SADMA_... SAR0, DAR0, LLP0, CTRL0, SSTATx, and DSTATx. The first four registers, along with the SADMA_CFG0 register, are used by the SATA DMA to set up and describe the block transfer.

Note: The term Link List Item (LLI) and block descriptor are synonymous.

79.2.3.1 Multi-block transfers

Multi-block transfers are enabled by setting the configuration parameter, DMAH_CHX_MULTI_BLK_EN to True.

Block chaining using linked lists

To enable multi-block transfers using block chaining, you must set the configuration parameter DMAH_CHx_MULTI_BLK_EN to True and the DMAH_CHx_HC_LLP parameter to False.

In this case, the SATA DMA re-programs the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

SATA DMA block chaining uses a Linked List Pointer register (SADMA_LLP0) that stores the address in memory of the next linked list item. Each LLI contains the corresponding block descriptors:

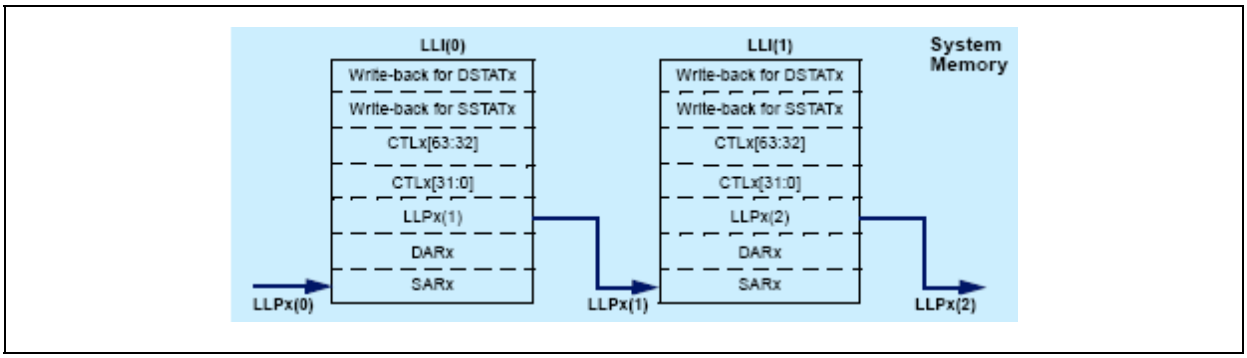
1. SADMA_SAR0
2. SADMA_DAR0
3. SADMA_LLP0
4. SADMA_CTRL0
5. SADMA_SSTATx
6. SADMA_DSTATx

To set up block chaining, you program a sequence of Linked Lists in memory.

The SADMA_... SAR0, DAR0, LLPx, and CTRL0 registers are fetched from system memory on an LLI update. If configuration parameter DMAH_CHx_CTL_WB_EN = True, then the updated contents of the SADMA_... CTRL0, SSTATx, and DSTATx regregisters are written back to

memory on block completion. *Figure 270* and *Figure 271* show how to use chained linked lists in memory to define multi-block transfers using block chaining.

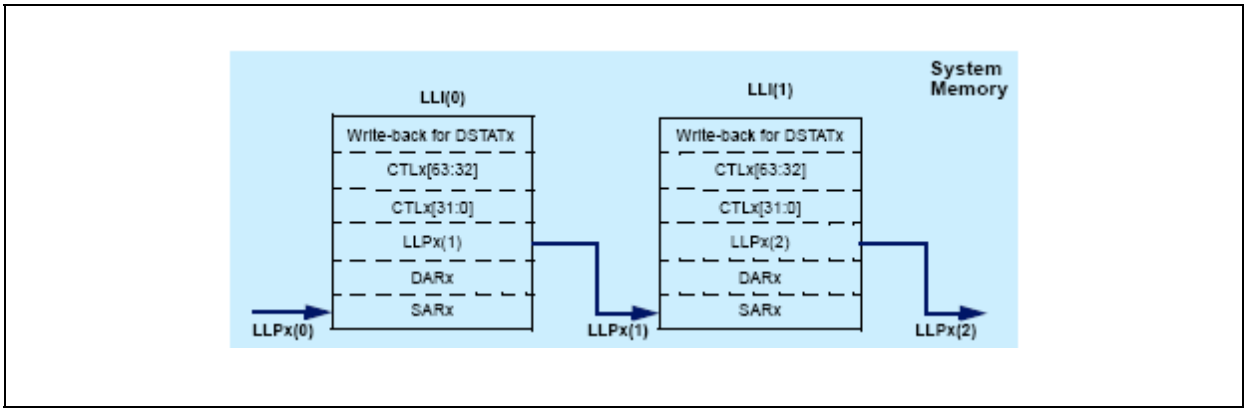
Figure 270: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True



It is assumed that no allocation is made in system memory for the source status when the configuration parameter DMAH_CHx_STAT_SRC is set to False. If this parameter is False, then the order of a Linked List item is as follows:

1. SAR0
2. DAR0
3. LLP0
4. CTRL0
5. DSTATx

Figure 271: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to False



Note: So as not to confuse the SADMA_... SAR0, DAR0, LLPx, CTRL0, STATx, and DSTATx register locations of the LLI with the corresponding SATA DMA memory mapped register locations, the LLI register locations are prefixed with LLI; that is, LLI.SAR0, LLI.DAR0, LLI.LLPx, LLI.SADMA_CTRL0, LLI.SSTATx, and LLI.DSTATx.

Figure 272 on page 926 and *Figure 273 on page 926* show the mapping of a Linked List Item stored in memory to the channel registers block descriptor.

Rows 6 through 10 of *Table 220: Programming of transfer types and channel register update method on page 924* show the required values of LLPx, SADMA_CTRL0, and SADMA_CFG0 for multi-block DMA transfers using block chaining.

Note: For rows 6 through 10 of Table 220, the LLI.SADMA_CTRL0, LLI.LLPx, LLI.SAR0, and LLI.DAR0 register locations of the LLI are always affected at the start of every block transfer. The LLI.LLPx and LLI.SADMA_CTRL0 are always used to reprogram the SATA DMA LLPx and SADMA_CTRL0 registers. However, depending on the Table 220 row number, the LLI.SAR0/ LLI.DAR0 address may or may not be used to reprogram the SATA DMA SAR0/DAR0 registers.

Table 220: Programming of transfer types and channel register update method

Transfer Type	LLP LOC =0	LLP_ SRC_E N (SADM A_CTR L0)	RELOA D _SRC (SADM A_CFG 0)	LLP_ DST_E N (SADM A_CTR L0)	RELOA D _DST (SADM A_CFG 0)	SADMA_ CTRL0, LLPx Update Method	SAR0 Update Method	DAR0 Update Method	Write Back
1. Single-block Yes 0 0 0 0 None, user None None No or last transfer of reprograms (single) (single) multi-block.	Yes	0	0	0	0	None, user reprogra mmes	None (single)	None (single)	No
2. Auto-reload Yes 0 0 0 1 SADMA_CTRL0 , LLPx Con- Auto-No multi- block are reloaded tiguous Reloadtransfer with from initial contiguous SAR values.	Yes	0	0	0	1	SADMA_ CTRL0, LLPx are reloaded from initial values	Con- tiguous	Auto reload	No
3. Auto-reload Yes 0 1 0 0 SADMA_CTRL0 , LLPx Auto- Con-No multi- block are reloaded Reload tiguous transfer with from initial contiguous DAR. values	Yes	0	1	0	0	SADMA_ CTRL0, LLPx are reloaded from initial values	Auto reload	Con- tiguous	No
4. Auto-reload Yes 0 1 0 1 SADMA_CTRL0 , LLPx Auto- Auto-No multi- block are reloaded reload Reload transfer from initial values	Yes	0	1	0	1	SADMA_ CTRL0, LLPx are reloaded from initial values	Auto reload	Auto reload	No
5. Single-block No 0 0 0 0 None, user None None Yes or last transfer of reprograms (single) (single) multi-block.	No	0	0	0	0	None, user reprogra mmes	None (single)	None (single)	Yes
6. Linked list No 0 0 1 0 SADMA_CTRL0 , LLPx Con- Linked Yes multi- block loaded from tiguous List transfer with next Linked contiguous SAR List item.	No	0	0	1	0	SADMA_ CTRL0, LLPx are loaded from next Linked List item	Con- tiguous	Linked list	Yes

Table 220: Programming of transfer types and channel register update method

Transfer Type	LLP LOC =0	LLP_ SRC_E N (SADM A_CTR L0)	RELOA D _SRC (SADM A_CFG 0)	LLP_ DST_E N (SADM A_CTR L0)	RELOA D _DST (SADM A_CFG 0)	SADMA_ CTRL0, LLPx Update Method	SAR0 Update Method	DAR0 Update Method	Write Back
7. Linked list No 0 1 1 0 SADMA_CTRL0 , LLPx Auto-Linked Yes multi-block loaded from Reload List transfer with next Linked auto-reload SAR List item.	No	0	1	1	0	SADMA_ CTRL0, LLPx are loaded from next Linked List item	Auto reload	Linked list	Yes
8. Linked list No 1 0 0 0 SADMA_CTRL0 , LLPx Linked Con-Yes multi-block loaded from List tiguous transfer with next Linked contiguous DAR List item.	No	1	0	0	0	SADMA_ CTRL0, LLPx are loaded from next Linked List item	Linked list	Con-tiguous	Yes
9. Linked list No 1 0 0 1 SADMA_CTRL0 , LLPx Linked Auto-Yes multi-block loaded from List Reload transfer with next Linked auto-reload DAR List item.	No	1	0	0	1	SADMA_ CTRL0, LLPx are loaded from next Linked List item	Linked list	Auto reload	Yes
10. Linked list No 1 0 1 0 SADMA_CTRL0 , LLPx Linked Linked Yes multi-block loaded from List List transfer next Linked List item.	No	1	0	1	0	SADMA_ CTRL0, LLPx are loaded from next Linked List item	Linked list	Linked list	Yes

Note: Write back: This column assumes that the configuration parameter DMAH_CHx_CTL_WB_EN = True. If DMAH_CHx_CTL_WB_EN = False, then there is never writeback of the control and status registers regardless of transfer type, and all rows of this column are “No”.

Figure 272: Mapping of Block Descriptor (LLI) in Memory to Channel Registers when DMAH_CHx_STAT_SRC is set to True

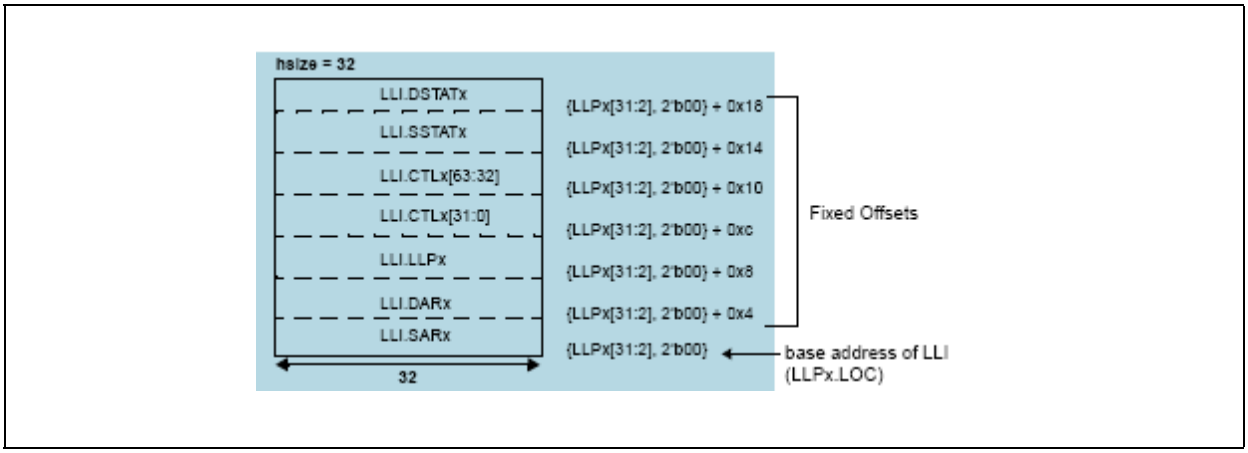
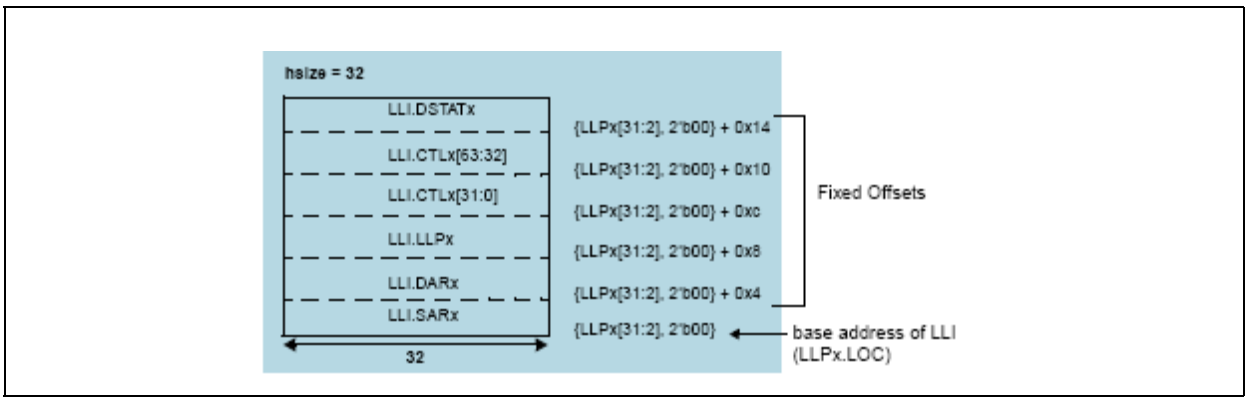


Figure 273: Mapping of Block Descriptor (LLI) in Memory to Channel Registers when DMAH_CHx_STAT_SRC is set to false



Note: Throughout this databook, there are descriptions about fetching the LLI.SADMA_CTRL0 register from the location pointed to by the LLPx register. This exact location is the LLI base address (stored in LLPx register) plus the fixed offset. For example, in Figure 272, the location of the LLI.SADMA_CTRL0 register is LLPx.LOC + 0xc.

Note: Referring to Table 220: Programming of transfer types and channel register update method on page 924, if the Write Back column entry is "Yes" and the configuration parameter DMAH_CHx_CTL_WB_EN = True, then the SADMA_CTRL0[63:32] register is always written to system memory (to LLI.SADMA_CTRL0[63:32]) at the end of every block transfer.

The source status is fetched and written to system memory at the end of every block transfer if the Write Back column entry is "Yes," DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and SADMA_CFG0.SS_UPD_EN is enabled.

The destination status is fetched and written to system memory at the end of every block transfer if the Write Back column entry is "Yes," DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and SADMA_CFG0.DS_UPD_EN is enabled.

79.2.3.2 Auto-reloading of channel registers

During auto-reloading, the channel registers are reloaded with their initial values at the completion of each block and the new values used for the new block. Depending on the row number in *Table 220: Programming of transfer types and channel register update method on*

page 924, some or all of the SAR0, DAR0, and SADMA_CTRL0 channel registers are reloaded from their initial value at the start of a block transfer.

79.2.3.3 Contiguous address between blocks

In this case, the address between successive blocks is selected as a continuation from the end of the previous block.

Enabling the source or destination address to be contiguous between blocks is a function of the SADMA_CTRL0.LLP_SRC_EN, SADMA_CFG0.RELOAD_SRC, SADMA_CTRL0.LLP_DST_EN, and SADMA_CFG0.RELOAD_DST registers (see Table 220).

Note: You cannot select both SAR0 and DAR0 updates to be contiguous. If you want this functionality, you should increase the size of the Block Transfer (SADMA_CTRL0.BLOCK_TS), or if this is at the maximum value, use Row 10 of Table 220 and set up the LLI.SAR0 address of the block descriptor to be equal to the end SAR0 address of the previous block. Similarly, set up the LLI.DAR0 address of the block descriptor to be equal to the end DAR0 address of the previous block. For more information, refer to Multi-block transfer with linked list for source and linked list for destination (row 10) on page 930.

79.2.3.4 Suspension of transfers between blocks

At the end of every block transfer, an end-of-block interrupt is asserted if:

1. Interrupts are enabled, SADMA_CTRL0.INT_EN = 1, and
2. The channel block interrupt is unmasked, MaskBlock[n] = 1, where n is the channel number.

Note: The block-complete interrupt is generated at the completion of the block transfer to the destination.

For rows 6, 8, and 10 of Table 220: Programming of transfer types and channel register update method on page 924, the DMA transfer does not stall between blocktransfers. For example, at the end-of-block N, the SATA DMA automatically proceedsto block N +1.

For rows 2, 3, 4, 7, and 9 of Table 220 (SAR0 and/or DAR0 auto-reloaded between block transfers), the DMA transfer automatically stalls after the end-of-block interrupt is asserted, if the end-of-block interrupt is enabled and unmasked.

The SATA DMA does not proceed to the next block transfer until a write to the ClearBlock[n] block interrupt clear register, done by software to clear the channel block-complete interrupt, is detected by hardware.

For rows 2, 3, 4, 7, and 9 of Table 220 (SAR0 and/or DAR0 auto-reloaded between block transfers), the DMA transfer does not stall if either:

- Interrupts are disabled, SADMA_CTRL0.INT_EN = 0, or
- The channel block interrupt is masked, MaskBlock[n] = 0, where n is the channel number.

Channel suspension between blocks is used to ensure that the end-of-block ISR (interrupt service routine) of the next-to-last block is serviced before the start of the final block commences. This ensures that the ISR has cleared the SADMA_CFG0.RELOAD_SRC and/or SADMA_CFG0.RELOAD_DST bits before completion of the final block. The reload bits SADMA_CFG0.RELOAD_SRC and/or SADMA_CFG0.RELOAD_DST should be cleared in the end-of-block ISR for the next-to-last block transfer.

79.2.3.5 Ending multi-block transfers

All multi-block transfers must end as shown in either Row 1 or Row 5 of Table 220. At the end of every block transfer, the SATA DMA samples the row number, and if the SATA DMA is in the Row 1 or Row 5 state, then the previous block transferred was the last block and the DMA transfer is terminated.

Note: Row 1 and Row 5 are used for single-block transfers or terminating multi-block transfers. Ending in the Row 5 state enables status fetch and write-back for the last block. Ending in the Row 1 state disables status fetch and write-back for the last block.

For rows 2, 3, and 4 of *Table 220*, (LLPx = 0 and SADMA_CFG0.RELOAD_SRC and/or SADMA_CFG0.RELOAD_DST is set), multi-block DMA transfers continue until both the SADMA_CFG0.RELOAD_SRC and SADMA_CFG0.RELOAD_DST registers are cleared by software. They should be programmed to zero in the end-of-block interrupt service routine that services the next-to-last block transfer; this puts the SATA DMA into the Row 1 state.

For rows 6, 8, and 10 of *Table 220: Programming of transfer types and channel register update method on page 924* (both SADMA_CFG0.RELOAD_SRC and SADMA_CFG0.RELOAD_DST cleared), the user must set up the last block descriptor in memory so that both LLI.SADMA_CTRL0.LLP_SRC_EN and LLI.SADMA_CTRL0.LLP_DST_EN are zero. If the LLI.LLPx register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the LLI.LLPx register of the last block descriptor in memory is zero, then the DMA transfer is terminated in Row 1.

For rows 7 and 9, the end-of-block interrupt service routine that services the next-to-last block transfer should clear the SADMA_CFG0.RELOAD_SRC and SADMA_CFG0.RELOAD_DST reload bits. The last block descriptor in memory should be set up so that both the LLI.SADMA_CTRL0.LLP_SRC_EN and LLI.SADMA_CTRL0.LLP_DST_EN registers are zero. If the LLI.LLPx register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the LLI.LLPx register of the last block descriptor in memory is zero, then the DMA transfer is terminated in Row 1.

Note: The only allowed transitions between the rows of Table 220 are from any row into Row 1 or Row 5. As already stated, a transition into row 1 or row 5 is used to terminate the DMA transfer; all other transitions between rows are not allowed. Software must ensure that illegal transitions between rows do not occur between blocks of a multi-block transfer. For example, if block N is in row 10, then the only allowed rows for block N +1 are rows 10, 5, or 1.

79.2.4 Programming a channel

Three registers – LLPx, SADMA_CTRL0, and SADMA_CFG0 – need to be programmed to determine whether single- or multi-block transfers occur, and which type of multi-block transfer is used. The different transfer types are shown in *Table 220*.

The SATA DMA can be programmed to fetch status from the source or destination peripheral; this status is stored in the SSTATx and DSTATx registers. When the SATA DMA is programmed to fetch this status from the source or destination peripheral, it writes this status and the contents of the SADMA_CTRL0 register back to memory at the end of a block transfer. The Write Back column of *Table 220* shows when this occurs.

The “Update Method” columns indicate where the values of SAR0, DAR0, SADMA_CTRL0, and LLPx are obtained for the next block transfer when multi-block SATA DMA transfers are enabled.

Note: In Table 220, all other combinations of LLPx.LOC = 0, SADMA_CTRL0.LLP_SRC_EN, SADMA_CFG0.RELOAD_SRC, SADMA_CTRL0.LLP_DST_EN, and SADMA_CFG0.RELOAD_DST are illegal, and will cause indeterminate or erroneous behavior.

79.2.4.1 Programming examples

- Single-block Transfer (Row 1)
- Multi-Block Transfer with Linked List for Source and Linked List for Destination (Row 10)
- Multi-Block Transfer with Source Address Auto-Reloaded and Destination Address Auto-Reloaded (Row 4)
- Multi-Block Transfer with Source Address Auto-Reloaded and Linked List Destination Address (Row 7)
- Multi-Block Transfer with Source Address Auto-Reloaded and Contiguous Destination Address (Row 3)
- Multi-Block DMA Transfer with Linked List for Source and Contiguous Destination Address (Row 8)

Single-block transfer (row 1)

This section describes a single-block transfer, Row 1 in *Table 220*.

Note: Row 5 in Table 220 is also a single-block transfer with write-back of control and status information enabled at the end of the single-block transfer.

1. Read the Channel Enable register to choose a free (disabled) channel; refer to ChEnReg.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: *SADMA_CLEAR_TFR on page 1039*, *SADMA_CLEAR_BLOCK*, *SADMA_CLEAR_SRC_TRAN*, *SADMA_CLEAR_DST_TRAN*, and *SADMA_CLEAR_ERR*. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:

   3.1 Write the starting source address in the *SADMA_SAR0* register (see *page 1022*)

   3.2 Write the starting destination address in the DAR0 register (see *page 1022*).

   3.3 Program SADMA_CTRL0 and SADMA_CFG0 according to Row 1, as shown in *Table 220*.

   3.4 Write the control information for the DMA transfer in the SADMA_CTRL0 register (see *page 1023*). For example, in the register, you can program the following:

   i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the SADMA_CTRL0 register. NOT! *Table 242: SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field.

   ii. Set up the transfer characteristics, such as:

   Transfer width for the source in the SRC_TR_WIDTH field. *Table 241: SADMA_... CTRL0.SRC_MSIZE and DEST_MSIZE decoding on page 1026* lists the decoding for this field.

   Transfer width for the destination in the DST_TR_WIDTH field. *Table 241* lists the decoding for this field.

   Source master layer in the SMS field where the source resides.

   Destination master layer in the DMS field where the destination resides.

   Incrementing/decrementing or fixed address for the source in the SINC field.

   Incrementing/decrementing or fixed address for the destination in the DINC field.

   3.5 Write the channel configuration information into the SADMA_CFG0 register (see *page 1027*).

   i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.

   This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests. Writing a 1 activates the software handshaking interface to handle source and destination requests.

   If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral; this requires programming the SRC_PER and DEST_PER bits, respectively.
4. After the SATA DMA-selected channel has been programmed, enable the channel by writing a 1 to the ChEnReg.CH_EN bit. Ensure that bit 0 of the DmaCfgReg register is enabled.
5. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripherals). The SATA DMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
6. Once the transfer completes, hardware sets the interrupts and disables the channel. At this time, you can respond to either the Block Complete or Transfer Complete interrupts, or poll

for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, the software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled.

Multi-block transfer with linked list for source and linked list for destination (row 10)

Note: This type of multi-block transfer can only be enabled when the either of the following parameters is set:

DMAH_CHx_MULTI_BLK_TYPE = NO_HARDCODE

or

DMAH_CHx_MULTI_BLK_TYPE = LLP_LLP

1. Read the Channel Enable register (see *SADMA_CH_EN on page 1042*) to choose a free (disabled) channel.
2. Set up the chain of Linked List Items (otherwise known as block descriptors) in memory. Write the control information in the LLI.SADMA_CTRL0 register location of the block descriptor for each LLI in memory (see *Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True on page 923*). For example, in the register, you can program the following:
   2.1 Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the SADMA_CTRL0 register. *Table 243: SADMA_CTRL0.TT_FC field decoding on page 1027* lists the decoding for this field.
   2.2 Set up the transfer characteristics, such as:
      v Transfer width for the source in the SRC_TR_WIDTH field. *Table 241: SADMA_...CTRL0.SRC_MSIZE and DEST_MSIZE decoding on page 1026* lists the decoding for this field.
      vi Transfer width for the destination in the DST_TR_WIDTH field. *Table 241 on page 1026* lists the decoding for this field.
      vii Source master layer in the SMS field where the source resides.
      viii Destination master layer in the DMS field where the destination resides.
      ix Incrementing/decrementing or fixed address for the source in the SINC field.
      x Incrementing/decrementing or fixed address for the destination in the DINC field.
3. Write the channel configuration information into the SADMA_CFG0 register (see *page 1027*).
   3.1 Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
   This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
   3.2 If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
4. Make sure that the LLI.SADMA_CTRL0 register locations of all LLI entries in memory (except the last) are set as shown in Row 10 of *Table 220*. The LLI.SADMA_CTRL0 register of the last Linked List Item must be set as described in Row 1 or Row 5 of *Table 220: Programming of transfer types and channel register update method on page 924*. *Table 270: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True on page 923* shows a Linked List example with two list items.

5. Make sure that the LLI.SAR0/LLI.DAR0 register locations of all LLI entries in memory point to the start source/destination block address preceding that LLI fetch.
6. If parameter DMAH_CHx_CTL_WB_EN = True, ensure that the LLI.SADMA_CTRL0.DONE field of the LLI.SADMA_CTRL0 register locations of all LLI entries in memory is cleared.
7. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: SADMA_... CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, and CLEAR_ERR (*page 1039*). Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
8. Program the SADMA_CTRL0 and SADMA_CFG0 registers according to Row 10, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*.
9. Finally, enable the channel by writing a 1 to the ChEnReg.CH_EN bit; the transfer is performed.
10. The SATA DMA fetches the first LLI from the location pointed to by LLPx(0).

Note: The LLI.SAR0, LLI. DAR0, LLI.LLPx, and LLI.SADMA_CTRL0 registers are fetched. The SATA DMA automatically reprograms the SADMA_SAR0, SADMA_DAR0, SADMA_LLP0, and SADMA_CTRL0 channel registers from the SADMA_LLPx(0).

11. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripheral). The SATA DMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
12. Once the block of data is transferred, the source status information is fetched from the location pointed to by the SSTATAR0 register and stored in the SSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True and SADMA_CFG0.SS_UPD_EN is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of *Table 220*.

    The destination status information is fetched from the location pointed to by the DSTATARx register and stored in the DSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True and SADMA_CFG0.DS_UPD_EN is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of *Table 220*.
13. If DMAH_CHx_CTL_WB_EN = True, then the SADMA_CTRL0[63:32] register is written out to system memory. For conditions under which the SADMA_CTRL0[63:32] register is written out to system memory, refer to the Write Back column of *Table 220*. The SADMA_CTRL0[63:32] register is written out to the same location on the same layer (SADMA_LLP0.LMS) where it was originally fetched; that is, the location of the SADMA_CTRL0 register of the linked list item fetched prior to the start of the block transfer. Only the second word of the SADMA_CTRL0 register is written out – SADMA_CTRL0[63:32] – because only the SADMA_CTRL0.BLOCK_TS and SADMA_CTRL0.DONE fields have been updated by the SATA DMA hardware. Additionally, the SADMA_CTRL0.DONE bit is asserted to indicate block completion. Therefore, software can poll the LLI.SADMA_CTRL0.DONE bit of the SADMA_CTRL0 register in the LLI to ascertain when a block transfer has completed.

Note: Do not poll the SADMA_CTRL0.DONE bit in the SATA DMA memory map; instead, poll the LLI.SADMA_CTRL0.DONE bit in the LLI for that block. If the polled LLI.SADMA_CTRL0.DONE bit is asserted, then this block transfer has completed. This LLI.SADMA_CTRL0.DONE bit was cleared at the start of the transfer (Step 7).

14. The SSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and SADMA_CFG0.SS_UPD_EN is enabled. It is

written to the SSTATx register location of the LLI pointed to by the previously saved SADMA_LLP0.LOC register.

The DSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and SADMA_CFG0.DS_UPD_EN is enabled. It is written to the DSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register.

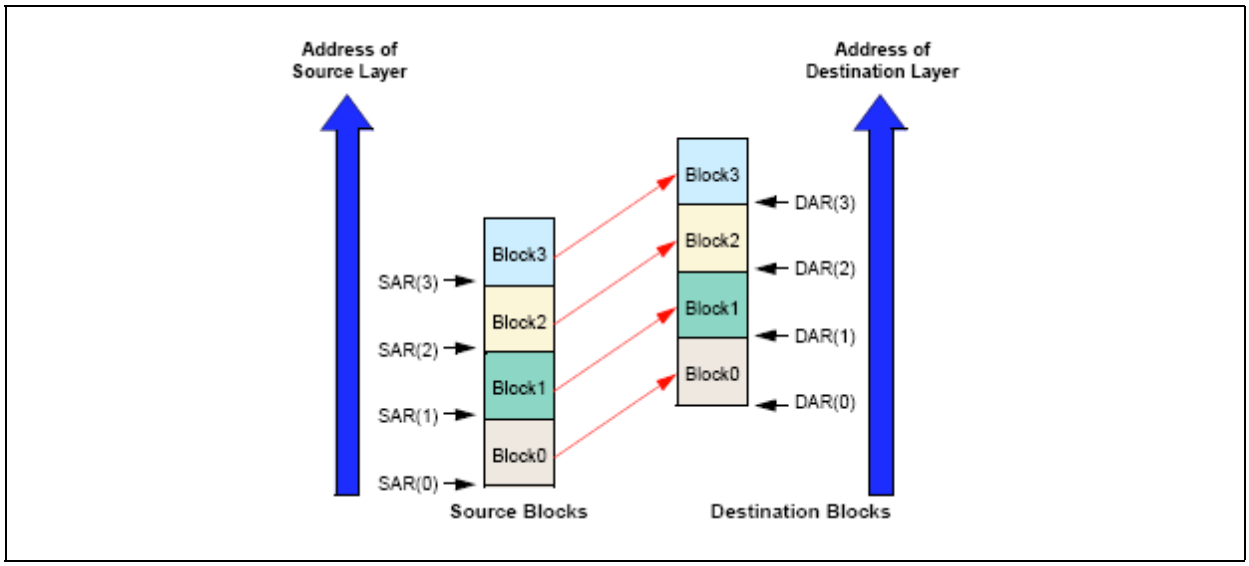
The end-of-block interrupt, int_block, is generated after the write-back of the control and status registers has completed.

Note: The write-back location for the control and status registers is the LLIpointed to by the previous value of the LLPx.LOC register, not the LLI pointed to by the current value of the LLPx.LOC register.

15. The SATA DMA does not wait for the block interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by the current LLPx register and automatically reprograms the SAR0, DAR0, LLPx, and SADMA_CTRL0 channel registers. The DMA transfer continues until the SATA DMA determines that the SADMA_CTRL0 and LLPx registers at the end of a block transfer match the ones described in Row 1 or Row 5 of *Table 220* (as discussed earlier). The SATA DMA then knows that the previously transferred block was the last block in the DMA transfer.

    The DMA transfer might look like that shown in *Figure 274*.

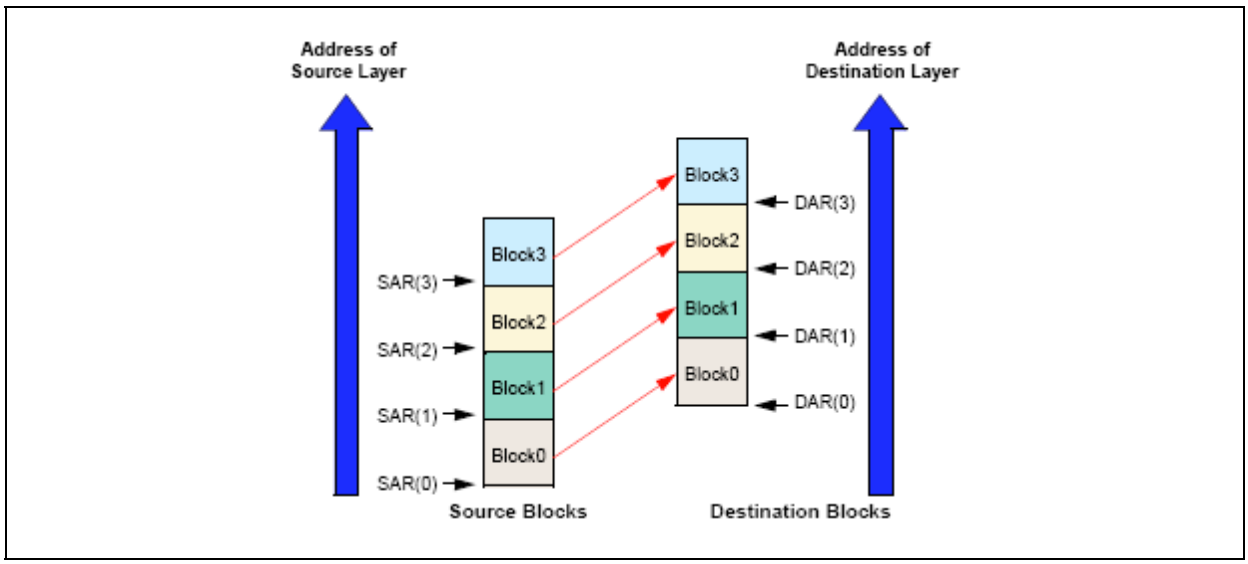
Figure 274: Multi-Block with Linked Address for Source and Destination



If the user needs to execute a DMA transfer where the source and destination address are contiguous, but where the amount of data to be transferred is greater than the maximum block

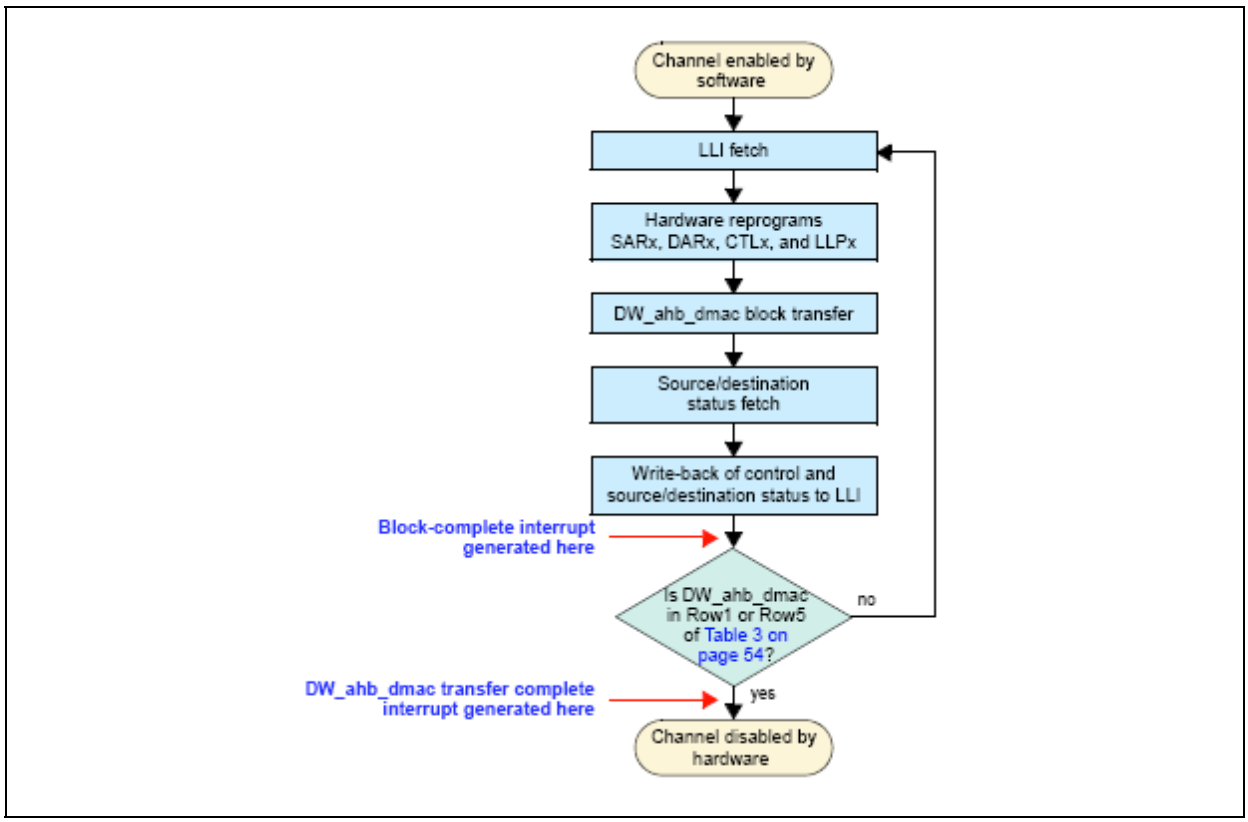
size SADMA_CTRL0.BLOCK_TS, then this can be achieved using the type of multi-block transfer shown in *Figure 275*.

Figure 275: Multi-Block with Linked Address for Source and Destination where SAR0 and DAR0 Between Successive Blocks are Contiguous



The DMA transfer flow is shown in *Figure 276*.

Figure 276: DMA Transfer Flow for Source and Destination Linked List Address



Multi-block transfer with source address auto-reloaded and destination address auto-reloaded (row 4)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:
DMAH_CHx_MULTI_BLK_TYPE = NO_HARDCODE

or
DMAH_CHx_MULTI_BLK_TYPE = RELOAD_RELOAD

1. Read the Channel Enable register (see *SADMA_CH_EN on page 1042*) to choose an available(disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: SADMA_... CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, AND CLEAR_ERR (*page 1039*). Reading the Interrupt Raw Status and Interrupt Status registers (*page 1032*) confirms that all interrupts have been cleared.
3. Program the following channel registers:
   - 3.1 Write the starting source address in the SADMA_SAR0 register (see *page 1022*)
   - 3.2 Write the starting destination address in the SADMA_DAR0 register (see *page 1022*).
   - 3.3 Program SADMA_CTRL0 and SADMA_CFG0 according to Row 4, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*.
   - 3.4 Write the control information for the DMA transfer in the SADMA_CTRL0 register. For example, in the register, you can program the following:
     - i Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the SADMA_CTRL0 register. *Table 243: SADMA_CTRL0.TT_FC field decoding on page 1027* lists the decoding for this field.
     - ii Set up the transfer characteristics, such as:
       - Transfer width for the source in the SRC_TR_WIDTH field; *Table 242: SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field.
       - Transfer width for the destination in the DST_TR_WIDTH field; *Table 242* lists the decoding for this field.
       - Source master layer in the SMS field where the source resides.
       - Destination master layer in the DMS field where the destination resides.
       - Incrementing/decrementing or fixed address for the source in the SINC field.
       - Incrementing/decrementing or fixed address for the destination in the DINC field.

3.5 Write the channel configuration information into the SADMA_CFG0 register (see *page 1027*). Ensure that the reload bits, SADMA_CFG0.RELOAD_SRC and SADMA_CFG0.RELOAD_DST, are enabled.

i Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.

ii If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.

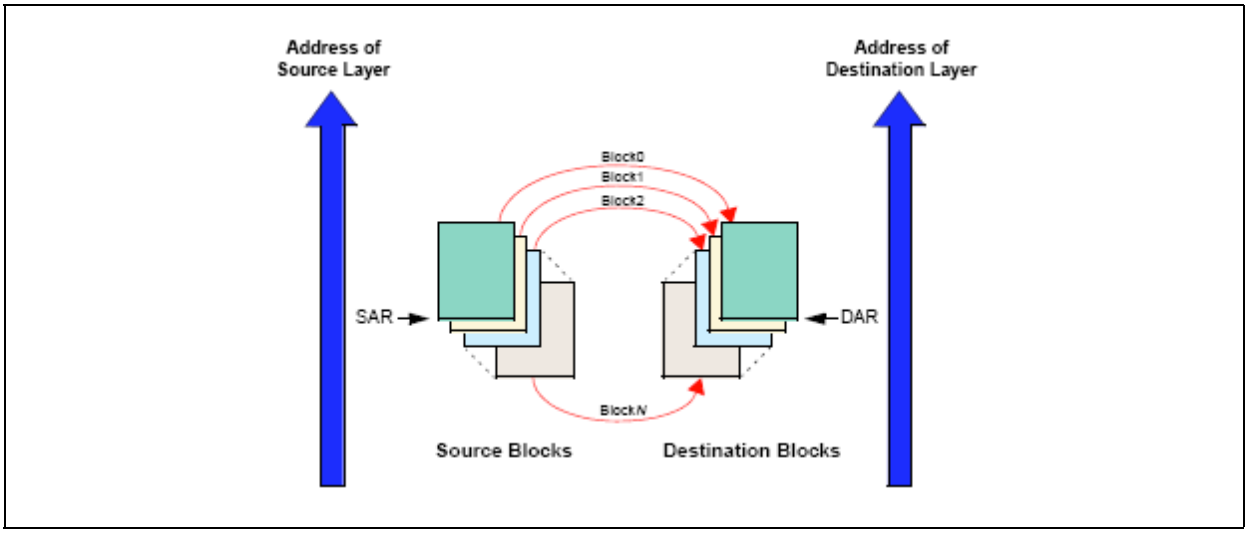
4. After the SATA DMA selected channel has been programmed, enable the channel by writing a 1 to the ChEnReg.CH_EN bit. Ensure that bit 0 of the DmaCfgReg register is enabled.
5. Source and destination request single and burst SATA DMA transactions to transfer the block of data (assuming non-memory peripherals). The SATA DMA acknowledges on completion of each burst/single transaction and carries out the block transfer.
6. When the block transfer has completed, the SATA DMA reloads the SAR0, DAR0, and SADMA_CTRL0 registers. Hardware sets the block-complete interrupt. The SATA DMA then samples the row number, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*. If the SATA DMA is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RawTfr[n], where n is the channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled. If the SATA DMA is not in Row 1, the next step is performed.
7. The DMA transfer proceeds as follows:

7.1 If interrupts are enabled (SADMA_CTRL0x.INT_EN = 1) and the block-complete interrupt is un-masked (MaskBlock[x] = 1'b1, where x is the channel number) hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the reload bits in the SADMA_CFG0.RELOAD_SRC and SADMA_CFG0.RELOAD_DST registers. This puts the SATA DMA into Row 1, as shown in *Table 220*. If the next block is not the last block in the DMA transfer, then the reload bits should remain enabled to keep the SATA DMA in Row 4.

7.2 If interrupts are disabled (SADMA_CTRL0.INT_EN = 0) or the block-complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it immediately starts the next block transfer. In this case, software must clear the reload bits in the SADMA_CFG0.RELOAD_SRC and SADMA_CFG0.RELOAD_DST registers to put the SATA DMA into Row 1 of *Table 220* before the last block of the DMA transfer has completed.

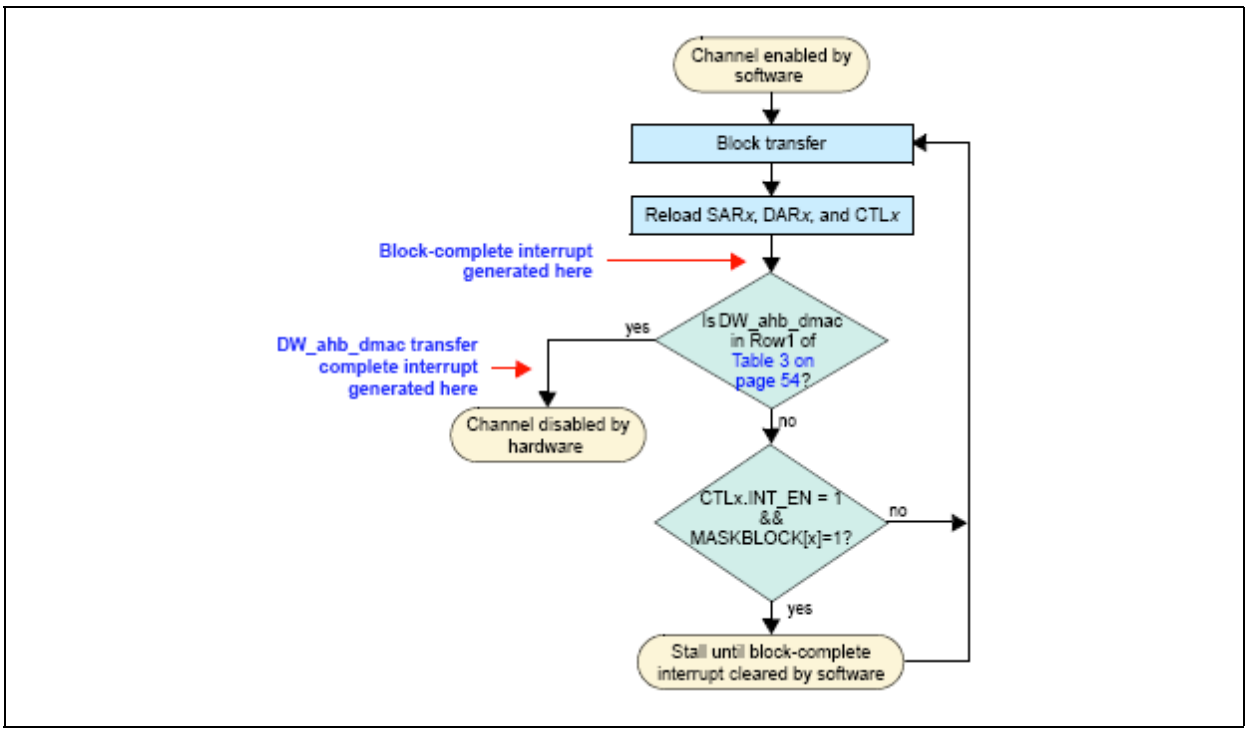
The transfer is similar to that shown in *Figure 277*.

Figure 277: Multi-block DMA transfer with source and destination address auto-reloaded



The DMA transfer is shown in *Figure 277*.

Figure 278: DMA Transfer Flow for Source and Destination Address Auto-Reloaded



Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

DMAH_CHx_MULTI_BLK_TYPE = 0

or

DMAH_CHx_MULTI_BLK_TYPE = RELOAD_LLP

1. Read the Channel Enable register (see *SADMA_CH_EN on page 1042*) in order to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as block descriptors) in memory. Write the control information in the LLI.SADMA_CTRL0 register location of the block descriptor for each LLI in memory (see *Figure 270: Multi-Block Transfer Using Linked Lists When*

DMAH_CHx_STAT_SRC Set to True on page 923). For example, in the register you can program the following:

2.1 Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control peripheral by programming the TT_FC of the SADMA_CTRL0 register. *Table 243: SADMA_CTRL0.TT_FC field decoding on page 1027* lists the decoding for this field.

2.2 Set up the transfer characteristics, such as:

i Transfer width for the source in the SRC_TR_WIDTH field. *Table 242: SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field.

ii Transfer width for the destination in the DST_TR_WIDTH field. *Table 242: SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field.

iii Source master layer in the SMS field where the source residest.

iv Destination master layer in the DMS field where the destination resides.

v Incrementing/decrementing or fixed address for the source in the SINC field.

vi Incrementing/decrementing or fixed address for the destination in the DINC field.

3. Write the starting source address in the SADMA_SAR0 register (see *page 1022*).

Note: The values in the LLI.SAR0 register locations of each of the Linked List Items (LLIs) set up in memory, although fetched during an LLI fetch, are not used.

4. Write the channel configuration information into the SADMA_CFG0 register (see *page 1027*).

4.1 Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC/HS_SEL_DST bits. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface source/destination requests.

4.2 If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral; this requires programming the SRC_PER and DEST_PER bits, respectively.

5. Make sure that the LLI.SADMA_CTRL0 register locations of all LLIs in memory (except the last) are set as shown in Row 7 of *Table 220: Programming of transfer types and channel register update method on page 924*, while the LLI.SADMA_CTRL0 register of the last Linked List item must be set as described in Row 1 or Row 5. *Table 270: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True on page 923* shows a Linked List example with two list items.
6. Ensure that the LLI.LLPx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Ensure that the LLI.DAR0 register location of all LLIs in memory point to the start destination block address preceding that LLI fetch.
8. If DMAH_CHx_CTL_WB_EN = True, ensure that the LLI.SADMA_CTRL0.DONE fields of the LLI.SADMA_CTRL0 register locations of all LLIs in memory are cleared.
9. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: SADMA_... CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, and CLEAR_ERR (*page 1039*). Reading the Interrupt Raw Status and Interrupt Status registers (*page 1032*) confirms that all interrupts have been cleared.
10. Program the SADMA_CTRL0 and SADMA_CFG0 registers according to Row 7, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*.
11. Finally, enable the channel by writing a 1 to the CH_EN.CH_EN bit; the transfer is performed. Ensure that bit 0 of the SADMA_DMA_CFG register is enabled.
12. The SATA DMA fetches the first LLI from the location pointed to by LLPx(0).

Note: The LLI.SAR0, LLI.DAR0, LLI. LLPx, and LLI.SADMA_CTRL0 registers are fetched. The LLI.SAR0 register – although fetched – is not used.

13. Source and destination request single and burst SATA DMA transactions to transfer the block of data (assuming non-memory peripherals). The SATA DMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
14. Once the block of data is transferred, the source status information is fetched from the location pointed to by the SSTATAR0 register and stored in the SSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and SADMA_CFG0.SS_UPD_EN is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of *Table 220*.
    The destination status information is fetched from the location pointed to by the DSTATARx register and stored in the DSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and SADMA_CFG0.DS_UPD_EN is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of *Table 220*.
15. If DMAH_CHx_CTL_WB_EN = True, then the SADMA_CTRL0[63:32] register is written out to system memory. For conditions under which the SADMA_CTRL0[63:32] register is written out to system memory, refer to the Write Back column of *Table 220*. The SADMA_CTRL0[63:32] register is written out to the same location on the same layer (SADMA_LLP0.LMS) where it was originally fetched; that is, the location of the SADMA_CTRL0 register of the linked list item fetched prior to the start of the block transfer. Only the second word of the SADMA_CTRL0 register is written out – SADMA_CTRL0[63:32] – because only the SADMA_CTRL0.BLOCK_TS and SADMA_CTRL0.DONE fields have been updated by hardware within the SATA DMA. The LLI.SADMA_CTRL0.DONE bit is asserted to indicate block completion. Therefore, software can poll the LLI.SADMA_CTRL0.DONE bit field of the SADMA_CTRL0 register in the LLI to ascertain when a block transfer has completed.

Note: Do not poll the SADMA_CTRL0.DONE bit in the SATA DMA memory map. Instead, poll the LLI.SADMA_CTRL0.DONE bit in the LLI for that block. If the polled LLI.SADMA_CTRL0.DONE bit is asserted, then this block transfer has completed. This LLI.SADMA_CTRL0.DONE bit was cleared at the start of the transfer (Step 8).

16. The SSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and SADMA_CFG0.SS_UPD_EN is enabled. It is written to the SSTATx register location of the LLI pointed to by the previously saved SADMA_LLP0.LOC register.
    The DSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and SADMA_CFG0.DS_UPD_EN is enabled. It is written to the DSTATx register location of the LLI pointed to by the previously saved SADMA_LLPx.LOC register.
    The end-of-block interrupt, int_block, is generated after the write-back of the control and status registers has completed.

Note: The write-back location for the control and status registers is the LLIpointed to by the previous value of the SADMA_*LLP0.LOC register, not the LLI pointed to by the current value of the* SADMA_*LLP0.LOC register.*

17. The SATA DMA reloads the SAR0 register from the initial value. Hardware sets the block-complete interrupt. The SATA DMA samples the row number, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*. If the SATA DMA is in Row 1 or Row 5, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled. If the SATA DMA is not in Row 1 or Row 5 as shown in *Table 220*, the following steps are performed.

18. The DMA transfer proceeds as follows:

    18.1 If interrupts are enabled (SADMA_CTRL0.INT_EN = 1) and the block-complete interrupt is unmasked (MaskBlock[x] = 1'b1, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the SADMA_CFG0.RELOAD_SRC source reload bit. This puts the SATA DMA into Row 1, as shown in *Table 220* If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the SATA DMA in Row 7, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*.

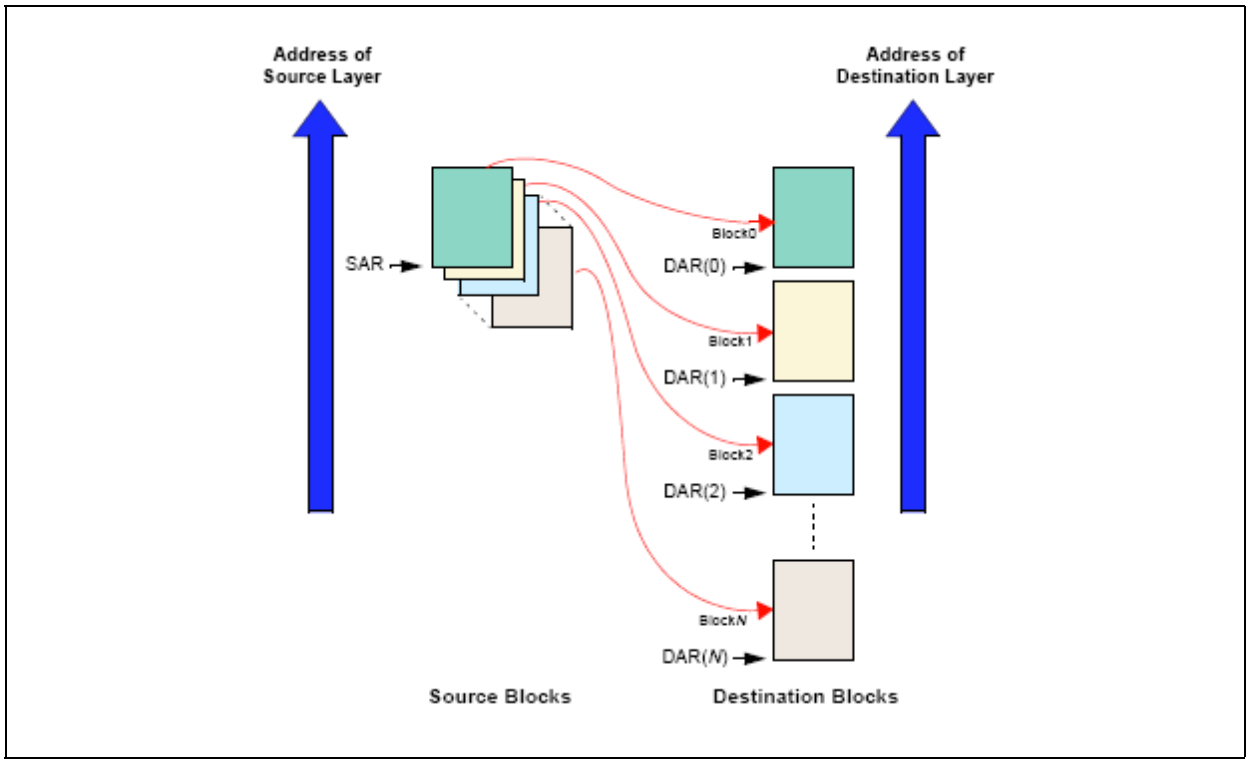
    18.2 interrupts are disabled (SADMA_CTRL0.INT_EN = 0) or the block-complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it immediately starts the next block transfer. In this case, software must clear the source reload bit, SADMA_CFG0.RELOAD_SRC in order to put the device into Row 1 of *Table 220* before the last block of the DMA transfer has completed.

19. The SATA DMA fetches the next LLI from memory location pointed to by the SADMA_LLP0 register and automatically reprograms the SADMA_DAR0, SADMA_CTRL0, and SADMA_LLP0 channel registers.

Note: The SAR0 is not re-programmed, since the reloaded value is used for the next DMA block transfer. If the next block is the last block of the DMA transfer, then the SADMA_CTRL0 and SADMA_LLP0 registers just fetched from the LLI should match Row 1 or Row 5 of Table 220

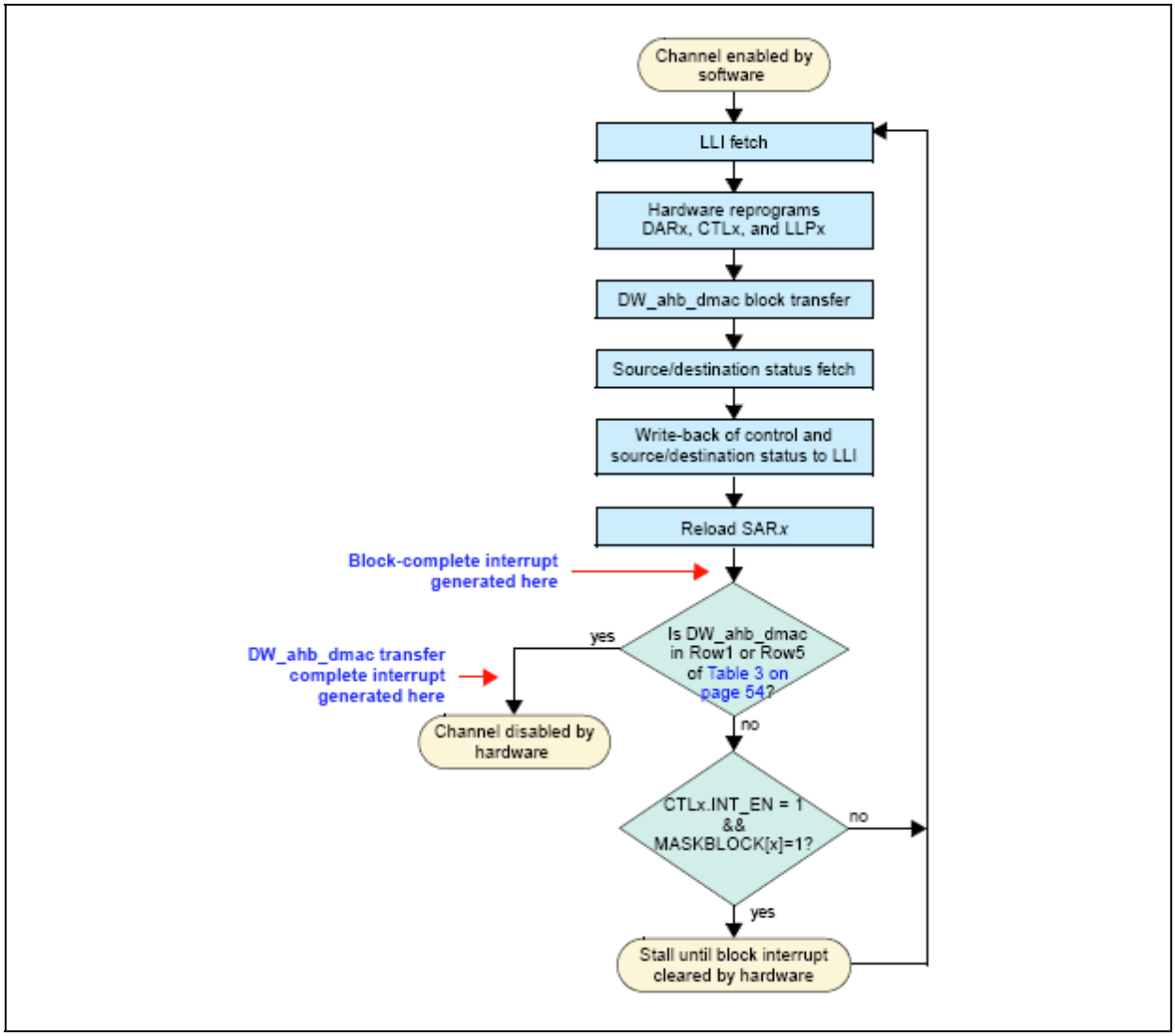
The DMA transfer might look like that shown in *Table 279*.

Figure 279: Multi-Block DMA Transfer with Source Address Auto-Reloaded and Linked List Destination Address



The DMA transfer flow is shown in *Table 280*.

Figure 280: DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address



Multi-block transfer with source address auto-reloaded and contiguous destination address (row 3)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

DMAH_CHx_MULTI_BLK_TYPE = 0

or

DMAH_CHx_MULTI_BLK_TYPE = RELOAD_CONT

1. Read the Channel Enable register (*SADMA_CH_EN on page 1042*) to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: SADMA_... CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, AND CLEAR_ERR (*page 1039*). Reading the Interrupt Raw Status and Interrupt Status registers (*page 1034*) confirms that all interrupts have been cleared.

3. Program the following channel registers:
   3.1 Write the starting source address in the SADMA_SAR0 register for channel (see *page 1022*)
   3.2 Write the starting destination address in the SADMA_DAR0 register (see *page 1022*).
   3.3 Program SADMA_CTRL0 and SADMA_CFG0 according to Row 3, shown in *Table 220*.

3.4 Write the control information for the DMA transfer in the SADMA_CTRL0 register (see *page 1023*). For example, in the register, you can program the following:

i Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the SADMA_CTRL0 register. *Table 243: SADMA_CTRL0.TT_FC field decoding on page 1027* lists the decoding for this field.

ii Set up the transfer characteristics, such as:

- Transfer width for the source in the SRC_TR_WIDTH field. *Table 242: SADMA_...CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field.
- Transfer width for the destination in the DST_TR_WIDTH field. *Table 242* lists the decoding for this field.
- Source master layer in the SMS field where the source resides.
- Destination master layer in the DMS field where the destination resides.
- Incrementing/decrementing or fixed address for the source in the SINC field.
- Incrementing/decrementing or fixed address for the destination in the DINC field.

3.5 Write the channel configuration information into the SADMA_CFG0 register (see *page 1023*).

i Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.

ii If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.

4. After the SATA DMA channel has been programmed, enable the channel by writing a 1 to the ChEnReg.CH_EN bit. Ensure that bit 0 of the DmaCfgReg register is enabled.
5. Source and destination request single and burst SATA DMA transactions to transfer the block of data (assuming non-memory peripherals). The SATA DMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
6. When the block transfer has completed, the SATA DMA reloads the SAR0 register; the DAR0 register remains unchanged. Hardware sets the block-complete interrupt. The SATA DMA then samples the row number, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*. If the SATA DMA is in Row 1, then the DMA transfer has completed. Hardware sets the transfer-complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled. If the SATA DMA is not in Row 1, the next step is performed.
7. The DMA transfer proceeds as follows:

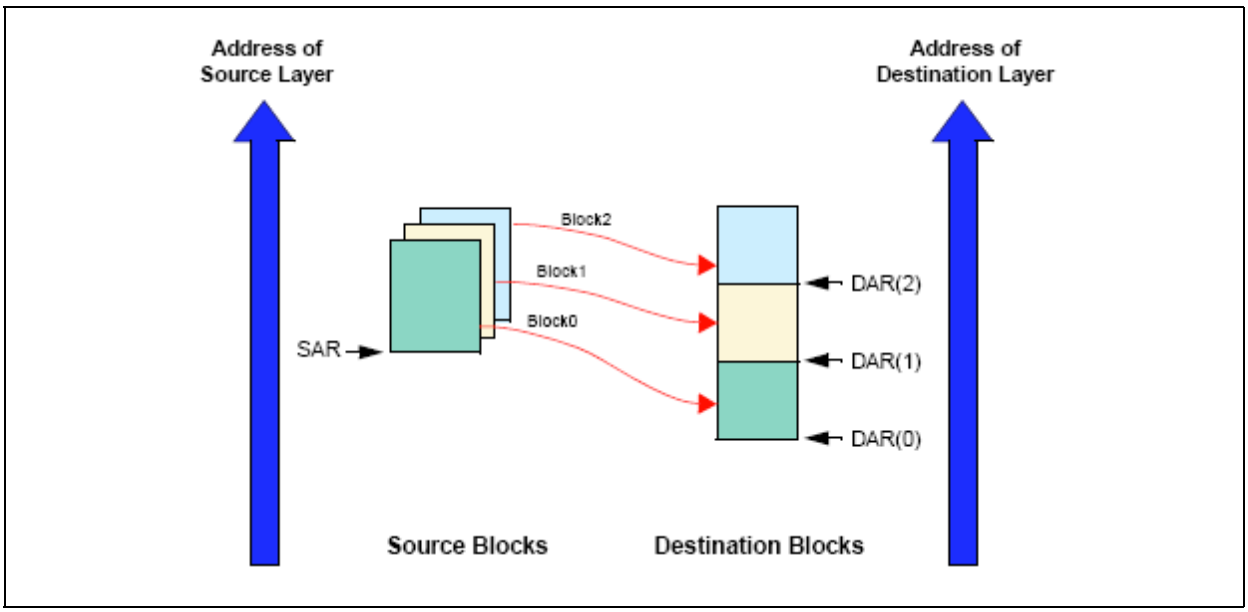
7.1 If interrupts are enabled (SADMA_CTRL0.INT_EN = 1) and the block-complete interrupt is un-masked (MaskBlock[x] = 1'b1, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine)

should clear the source reload bit, SADMA_CFG0.RELOAD_SRC. This puts the SATA DMA into Row 1, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*. If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the SATA DMA in Row 4.

7.2 If interrupts are disabled (SADMA_CTRL0.INT_EN = 0) or the block-complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it starts the next block transfer immediately. In this case, software must clear the source reload bit, SADMA_CFG0.RELOAD_SRC, to put the device into Row 1 of *Table 220* before the last block of the DMA transfer has completed.

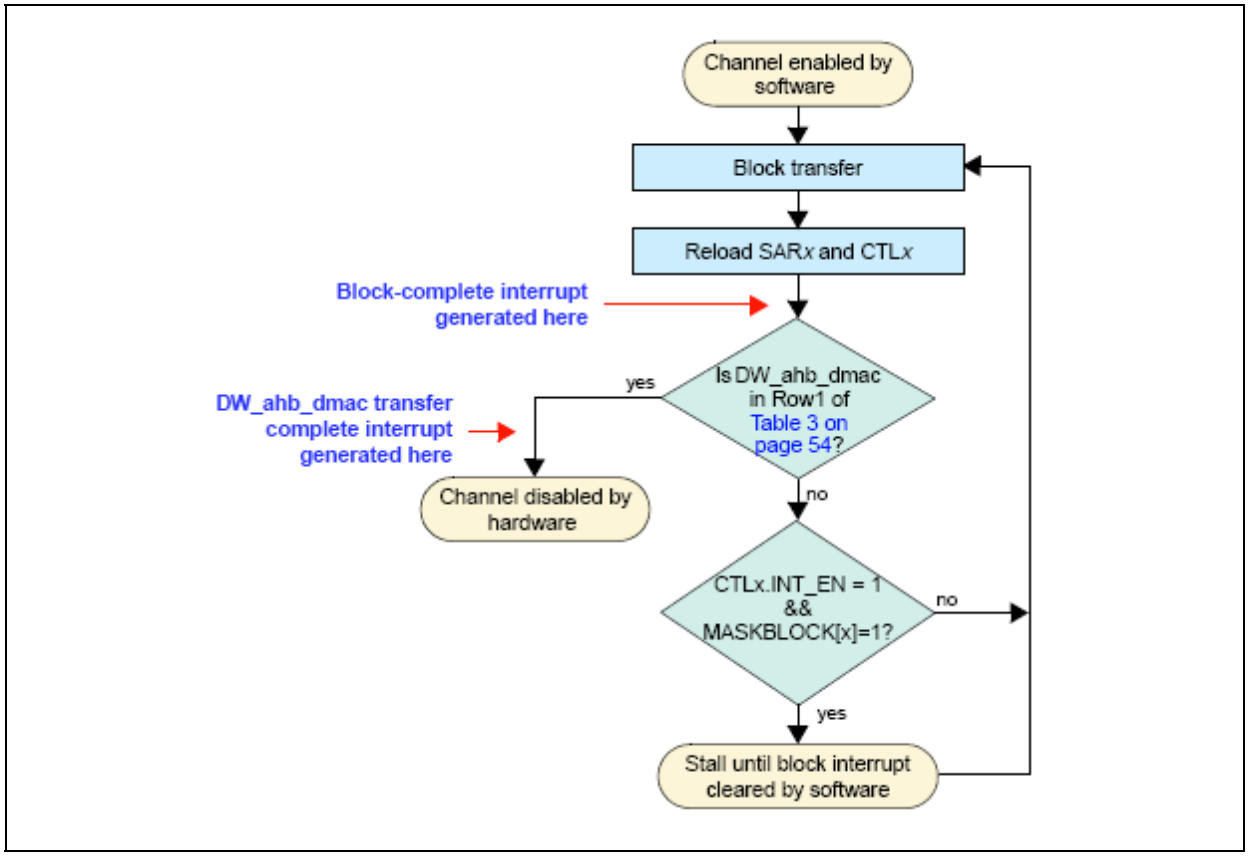
The transfer is similar to that shown in *Figure 281*.

Figure 281: Multi-Block DMA Transfer with Source Address Auto-Reloaded and Contiguous Destination Address



The DMA transfer flow is shown in *Figure 281*.

Figure 282: DMA Transfer Flow for Source Address Auto-Reloaded and Contiguous Destination Address



Multi-block DMA transfer with linked list for source and contiguous destination address (row 8)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:
DMAH_CHx_MULTI_BLK_TYPE = 0
or
DMAH_CHx_MULTI_BLK_TYPE = LLP_CONT

1. Read the Channel Enable register (see *SADMA_CH_EN on page 1042*) to choose a free (disabled) channel.
2. Set up the linked list in memory. Write the control information in the LLI. SADMA_CTRL0 register location of the block descriptor for each LLI in memory (see *Figure 270: Multi-Block*

Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True on page 923). For example, in the register, you can program the following:

2.1 Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the SADMA_CTRL0 register. *SADMA_CTRL0.TT_FC field decoding on page 1027* lists the decoding for this field.

2.2 Set up the transfer characteristics, such as:

i Transfer width for the source in the SRC_TR_WIDTH field. *Table 242: SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field.

ii Transfer width for the destination in the DST_TR_WIDTH field. *Table 242* lists the decoding for this field.

iii Source master layer in the SMS field where the source resides.

iv Destination master layer in the DMS field where the destination resides.

v Incrementing/decrementing or fixed address for the source in the SINC field.

vi Incrementing/decrementing or fixed address for the destination in the DINC field.

3. Write the starting destination address in the SADMA_DAR0 register (see *page 1022*).

Note: The values in the LLI.DAR0 register location of each Linked List Item (LLI) in memory, although fetched during an LLI fetch, are not used.

4. Write the channel configuration information into the SADMA_CFG0 register (see *page 1027*).

i Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC/HS_SEL_DST bits. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.

ii If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripherals. This requires programming the SRC_PER and DEST_PER bits, respectively.

5. Ensure that all LLI.SADMA_CTRL0 register locations of the LLI (except the last) are set as shown in Row 8 of *Table 220: Programming of transfer types and channel register update method on page 924*, while the LLI.SADMA_CTRL0 register of the last Linked List item must be set as described in Row 1 or Row 5 of *Table 220*. *Figure 270: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True on page 923* shows a Linked List example with two list items.
6. Ensure that the LLI.LLPx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Ensure that the LLI.SAR0 register location of all LLIs in memory point to the start source block address preceding that LLI fetch.
8. If DMAH_CHx_CTL_WB_EN = True, ensure that the LLI.SADMA_CTRL0.DONE fields of the LLI.SADMA_CTRL0 register locations of all LLIs in memory are cleared.
9. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: SADMA_... CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, and CLEAR_ERR (*page 1039*). Reading the Interrupt Raw Status and interrupt Status registers (*page 1032*) confirms that all interrupts have been cleared.
10. Program the SADMA_CTRL0 and SADMA_CFG0 registers according to Row 8, as shown in *Table 220: Programming of transfer types and channel register update method on page 924*.

11. Finally, enable the channel by writing a 1 to the CH_EN.CH_EN bit; the transfer is performed. Ensure that bit 0 of the SADMA_DMA_CFG register is enabled.
12. The SATA DMA fetches the first LLI from the location pointed to by SADMA_LLP0(0).

Note: The SADMA_... LLI.SAR0, LLI.DAR0, LLI.LLPx, and LLI.SADMA_CTRL0 registers are fetched.

The LLI.DAR0 register location of the LLI – although fetched – is not used.

The DAR0 register in the SATA DMA remains unchanged.

13. Source and destination request single and burst SATA DMA transactions to transfer the block of data (assuming non-memory peripherals). The SATA DMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
14. Once the block of data is transferred, the source status information is fetched from the location pointed to by the SSTATAR0 register and stored in the SADMA_SSTAT0 register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True and SADMA_CFG0.SS_UPD_EN is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of *Table 220: Programming of transfer types and channel register update method on page 924*.

    The destination status information is fetched from the location pointed to by the DSTATARx register and stored in the DSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True and SADMA_CFG0.DS_UPD_EN is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of *Table 220*.
15. If DMAH_CHx_CTL_WB_EN = True then the SADMA_CTRL0[63:32] register is written out to system memory. For conditions under which the SADMA_CTRL0[63:32] register is written out to system memory, refer to the Write Back column of *Table 220*.

    The SADMA_CTRL0[63:32] register is written out to the same location on the same layer (SADMA_LLP0.LMS) where it was originally fetched; that is, the location of the SADMA_CTRL0 register of the linked list item fetched prior to the start of the block transfer. Only the second word of the SADMA_CTRL0 register is written out, SADMA_CTRL0[63:32], because only the SADMA_CTRL0.BLOCK_TS and SADMA_CTRL0.DONE fields have been updated by hardware within the SATA DMA. Additionally, the SADMA_CTRL0.DONE bit is asserted to indicate block completion. Therefore, software can poll the LLI.SADMA_CTRL0.DONE bit field of the SADMA_CTRL0 register in the LLI to ascertain when a block transfer has completed.

Note: Do not poll the SADMA_CTRL0.DONE bit in the SATA DMA memory map. Instead, poll the LLI.SADMA_CTRL0.DONE bit in the LLI for that block. If the polled LLI.SADMA_CTRL0.DONE bit is asserted, then this block transfer has completed. This LLI.SADMA_CTRL0.DONE bit was cleared at the start of the transfer (Step 8).

16. The SSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True and SADMA_CFG0.SS_UPD_EN is enabled. It is written to the SSTATx register location of the LLI pointed to by the previously saved SADMA_LLP0.LOC register.

    The DSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True and SADMA_CFG0.DS_UPD_EN is enabled. It is written to the DSTATx register location of the LLI pointed to by the previously saved SADMA_LLP0.LOC register.

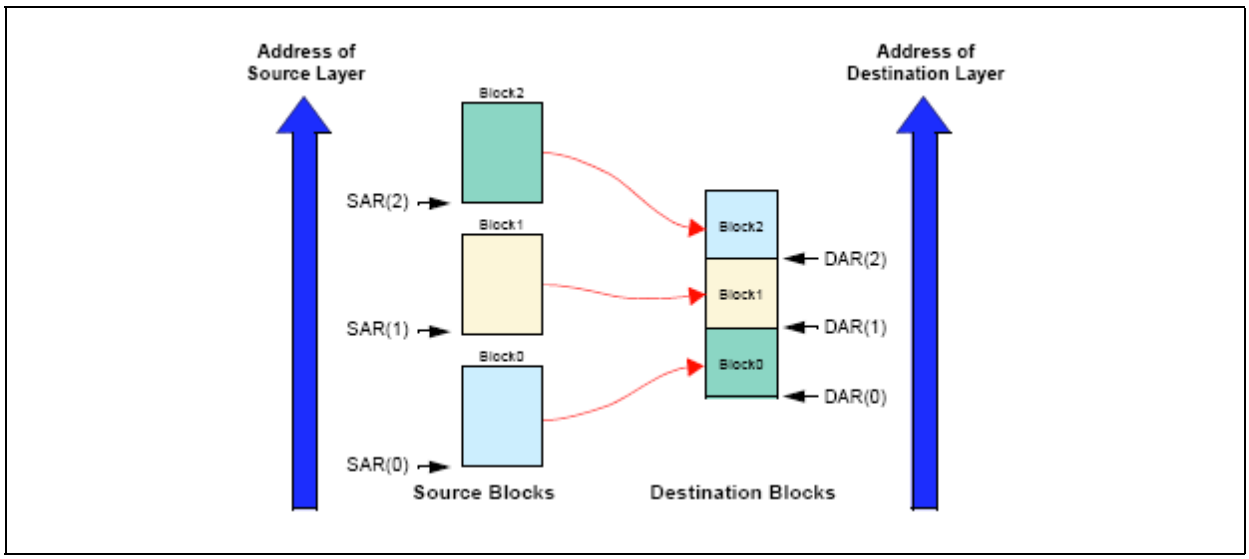
    The end-of-block interrupt, int_block, is generated after the write-back of the control and status registers has completed.

Note: The write-back location for the control and status registers is the LLIpointed to by the previous value of the LLPx.LOC register, not the LLIpointed to by the current value of the LLPx.LOC register.

17. The SATA DMA does not wait for the block interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by current LLPx register and automatically reprograms the SADMA_... SAR0, SADMA_CTRL0, and LLP0 channel registers. The DAR0 register is left unchanged. The DMA transfer continues until the SATA DMA samples that the SADMA_CTRL0 and LLPx registers at the end of a block transfer match. The SATA DMA then knows that the previously transferred block was the last block in the DMA transfer.

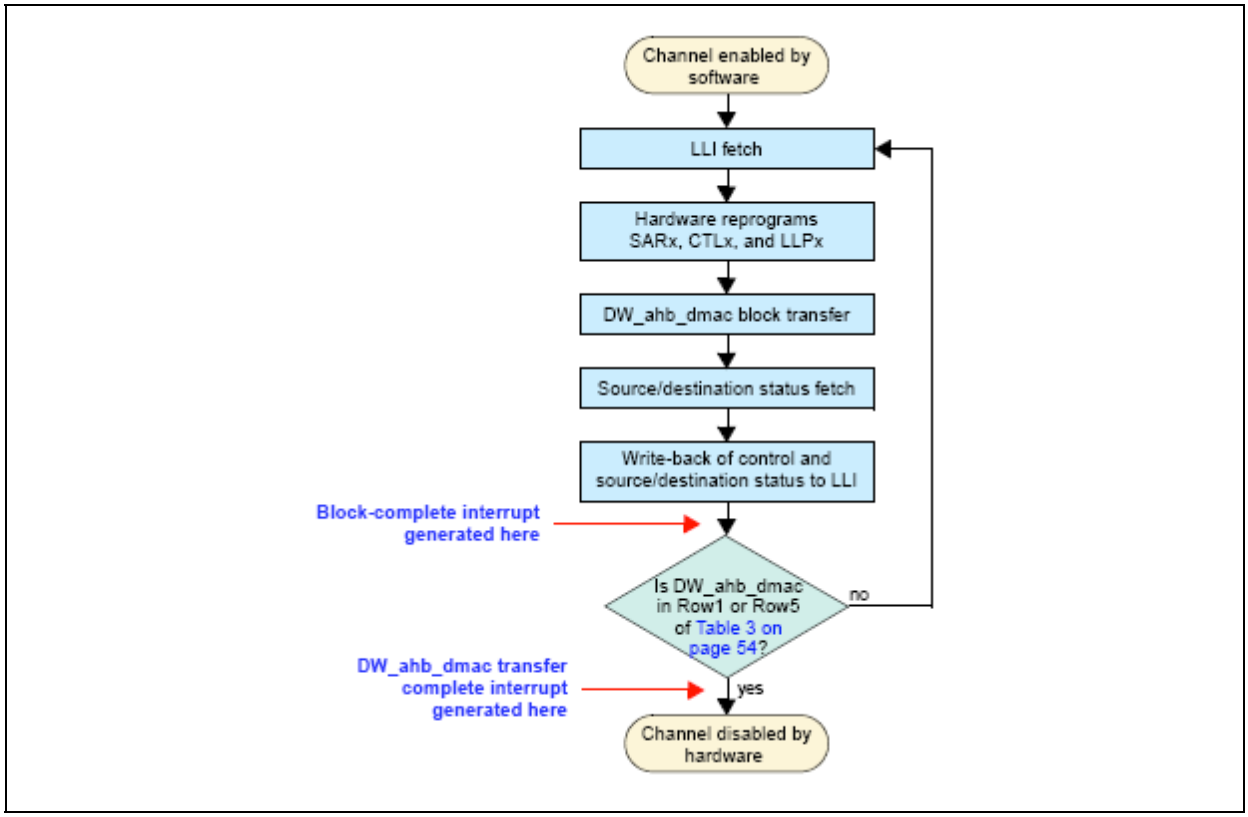
The SATA DMA transfer might look like that shown in *Figure 283*. Note that the destination address is decrementing.

Figure 283: Multi-Block DMA Transfer with Linked List Source Address and Contiguous Destination Address



The DMA transfer flow is shown in *Figure 284*.

Figure 284: DMA Transfer Flow for Source Address Auto-Reloaded and Contiguous Destination Address



79.2.5 Disabling a channel prior to transfer completion

Under normal operation, software enables a channel by writing a 1 to the channel enable register, ChEnReg.CH_EN, and hardware disables a channel on transfer completion by clearing the ChEnReg.CH_EN register bit.

The recommended way for software to disable a channel without losing data is to use the CH_SUSP bit in conjunction with the FIFO_EMPTY bit in the Channel Configuration Register (SADMA_CFG0).

1. If software wishes to disable a channel prior to the DMA transfer completion, then it can set the SADMA_CFG0.CH_SUSP bit to tell the SATA DMA to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
2. Software can now poll the SADMA_CFG0.FIFO_EMPTY bit until it indicates that the channel FIFO is empty.
3. The ChEnReg.CH_EN bit can then be cleared by software once the channel FIFO is empty.

When SADMA_CTRL0.SRC_TR_WIDTH < SADMA_CTRL0.DST_TR_WIDTH and the SADMA_CFG0.CH_SUSP bit is high, the SADMA_CFG0.FIFO_EMPTY is asserted once the contents of the FIFO do not permit a single word of SADMA_CTRL0.DST_TR_WIDTH to be formed. However, there may still be data in the channel FIFO, but not enough to form a single transfer of SADMA_CTRL0.DST_TR_WIDTH. In this scenario, once the channel is disabled, the remaining data in the channel FIFO is not transferred to the destination peripheral.

It is permissible to remove the channel from the suspension state by writing a 0 to the SADMA_CFG0.CH_SUSP register. The DMA transfer completes in the normal manner.

Note: If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.

79.2.5.1 Abnormal transfer termination

A SATA DMA DMA transfer may be terminated abruptly by software by clearing the channel enable bit, ChEnReg.CH_EN. You must not assume that the channel is disabled immediately after the ChEnReg.The CH_EN bit is cleared over the AHB slave interface. Consider this as a request to disable the channel. You must poll ChEnReg.CH_EN and confirm that the channel is disabled by reading back 0. A case where the channel is not disabled after a channel disable request is where either the source or destination has received a split or retry response. The SATA DMA must keep re-attempting the transfer to the system HADDR that originally received the split or retry response until an OKAY response is returned; to do otherwise is an bus protocol violation.

Software may terminate all channels abruptly by clearing the global enable bit in the SATA DMA Configuration Register (DmaCfgReg[0]). Again, you must not assume that all channels are disabled immediately after the DmaCfgReg[0] is cleared over the AHB slave interface. Consider this as a request to disable all channels. You must poll ChEnReg and confirm that all channels are disabled by reading back 0.

Note: 1 If the channel enable bit is cleared while there is data in the channel FIFO, this data is not sent to the destination peripheral and is not present when the channel is re-enabled. For read-sensitive source peripherals, such as a source FIFO, this data is therefore lost. When the source is not a read-sensitive device (such as memory), disabling a channel without waiting for the channel FIFO to empty may be acceptable, since the data is available from the source peripheral upon request and is not lost.

2 If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.

79.3 Functional Description

This section describes the functional details of the SATA DMA component. The topics are as follows:

79.3.1 Setup/Operation of SATA DMA Transfers

Section 79.2.4: Programming a channel on page 928 describes how to program the SATA DMA in order to perform DMA transfers. This section discusses how a single block transfer, made up of transactions, is actually performed. The relevant settings of the SATA DMA are also discussed here.

79.3.2 Block Flow Controller and Transfer Type

The device that controls the length of a block is known as the flow controller. Either the SATA DMA, the source peripheral, or the destination peripheral must be assigned as the flow controller.

If the block size is known prior to when the channel is enabled, then the SATA DMA should be programmed as the flow controller. The block size should be programmed into the SADMA_CTRL0.BLOCK_TS field.

If the block size is unknown when the SATA DMA channel is enabled, either the source or destination peripheral must be the flow controller.

The SADMA_CTRL0.TT_FC field indicates the transfer type and flow controller for that channel. *Table 221* lists valid transfer types and flow controller combinations.

Table 221: Transfer Types and Flow Control Combinations

Transfer Type	Flow Controller
Memory to Memory	SATA DMA
Memory to Peripheral	SATA DMA
Memory to Peripheral	Peripheral
Peripheral to Memory	SATA DMA
Peripheral to Memory	Peripheral
Peripheral to Peripheral	SATA DMA
Peripheral to Peripheral	Source Peripheral
Peripheral to Peripheral	Destination Peripheral

As an example, the SATA DMA can be programmed as the flow controller when a DMA block must be transferred from a receive DW_apb_ssi peripheral to memory. In a block transfer, software programs the DW_apb_ssi register – CTRLR1.NDF – with the number of source data items minus 1. Software then programs the SADMA_CTRL0.BLOCK_TS register with the same value and programs the SATA DMA as the flow controller.

The DW_apb_ssi has no built-in intelligence to signal block completion to the SATA DMA; this is not required in this case because software knows the block size prior to enabling the channel.

As another example, a peripheral can be a block flow controller when a DMA block must be transferred from an Ethernet controller to memory. In this case, the size of an ethernet packet may not be known prior to enabling the SATA DMA channel. Therefore, the ethernet controller needs built-in intelligence to indicate to the SATA DMA when a block transfer has completed.

79.3.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or burst transactions. The operation of the handshaking interface is different and depends on whether the peripheral or the SATA DMA is the flow controller.

The peripheral uses the handshaking interface to indicate to the SATA DMA that it is ready to transfer or accept data over the bus bus.

A non-memory peripheral can request a DMA transfer through the SATA DMA using one of two types of handshaking interfaces:

- Hardware
- Software

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

Note: Throughout the remainder of this document, references to both source and destination hardware handshaking interfaces assume an active-high interface (refer to SADMA_CFG0.SRC(DST)_HS_POL bits in the Channel Configuration register, SADMA_CFG0 on page 1027). When active-low handshaking interfaces are used, then the active level and edge are reversed from that of an active-high interface.

The type of handshaking interface depends on whether the peripheral is a flow controller or not; descriptions are contained in the following sections:

- *Section 79.3.6: Handshaking Interface – Peripheral Is Not Flow Controller on page 954*
- *Section 79.3.6.3: Hardware Handshaking – Peripheral Is Not Flow Controller on page 955*
- *Section 79.3.6.4: Software Handshaking – Peripheral Is Not Flow Controller on page 960*

- *Section 79.3.7: Handshaking Interface – Peripheral Is Flow Controller on page 961*
- *Section 79.3.7.1: Hardware Handshaking – Peripheral Is Flow Controller on page 962*

Note: Source and destination peripherals can independently select the handshaking interface type; that is, hardware or software handshaking. For more information, refer to the CFG0.HS_SEL_SRC and CFG0.HS_SEL_DST parameters in the SADMA_CFG0 on page 1027.

79.3.4 Basic Definitions

Note: In this chapter and the following equations, references to SADMA_CTRL0.SRC_MSIZE, SADMA_CTRL0.DEST_MSIZE, SADMA_CTRL0.SRC_TR_WIDTH, and SADMA_CTRL0.DST_TR_WIDTH refer to the decoded values of the parameters; for example, SADMA_CTRL0.SRC_MSIZE = 3'b001 decodes to 4, and SADMA_CTRL0.SRC_TR_WIDTH = 3'b010 decodes to 32 bits.

The following definitions are used in this chapter:

- Source single transaction size in bytes
  src_single_size_bytes = SADMA_CTRL0.SRC_TR_WIDTH/8 (1)
- Source burst transaction size in bytes
  src_burst_size_bytes = SADMA_CTRL0.SRC_MSIZE * src_single_size_bytes (2)
- Destination single transaction size in bytes
  dst_single_size_bytes = SADMA_CTRL0.DST_TR_WIDTH/8 (3)
- Destination burst transaction size in bytes
  dst_burst_size_bytes = SADMA_CTRL0.DEST_MSIZE * dst_single_size_bytes (4)
- Block size in bytes:
  SATA DMA is flow controller – With the SATA DMA as the flow controller, the processor programs the SATA DMA with the number of data items (block size) of source transfer width (SADMA_CTRL0.SRC_TR_WIDTH) to be transferred by the SATA DMA in a block transfer; this is programmed into the SADMA_CTRL0.BLOCK_TS field. Therefore, the total number of bytes to be transferred in a block is:
  blk_size_bytes_dma = SADMA_CTRL0.BLOCK_TS * src_single_size_bytes (5)
- Source peripheral is block flow controller
  blk_size_bytes_src = (Number of source burst transactions in block * src_burst_size_bytes) + (Number of source single transactions in block * src_single_size_bytes) (6)
- Destination peripheral is block flow controller
  blk_size_bytes_dst = (Number of destination burst transactions in block * dst_burst_size_bytes) + (Number off destination single transactions in block * dst_single_size_bytes) (7)

79.3.5 Memory Peripherals

There is no handshaking interface with the SATA DMA, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request.

The alternative to not having a transaction-level handshaking interface is to allow the SATA DMA to attempt bus transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these bus transfers, it inserts wait states onto the bus (by de-asserting hready) until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking interface, the peripheral can signal to the SATA DMA that it is ready to transmit or receive data, and then the SATA DMA can access the peripheral without the peripheral inserting wait states onto the bus.

Note: If a channel is used exclusively for memory-to-memory DMA transfers – that is, no transaction-level handshaking on the source or destination side – then set DMAH_MAX_MULT_SIZE to 4 in order to achieve logic optimization.

79.3.6 Handshaking Interface – Peripheral Is Not Flow Controller

When the peripheral is not the flow controller, the SATA DMA tries to efficiently transfer the data using as little of the bus bandwidth as possible. Generally, the SATA DMA tries to transfer the data using burst transactions and, where possible, fill or empty the channel FIFO in single bus bursts – provided that the software has not limited the bus burst length; refer to *Example 3 on page 968*. The SATA DMA can also lock the arbitration for the master bus interface so that a channel is permanently granted the master bus interface. Additionally, the SATA DMA can assert the bus hlock signal to lock the DW_ahb system arbiter. For more information, refer to *Section 79.3.11.1: Locked DMA Transfers on page 1002*.

Before describing the handshaking interface operation when the peripheral is not the flow controller, the following sections define the terms “Single Transaction Region” and “Early-Terminated Burst Transaction.”

79.3.6.1 Single Transaction Region

There are cases where a DMA block transfer cannot complete using only burst transactions. Typically this occurs when the block size is not a multiple of the burst transaction length; for more information, refer to *Section : Example 4 on page 971* and *Section : Example 5 on page 974*. In these cases, the block transfer uses burst transactions up to the point where the amount of data left to complete the block is less than the amount of data in a burst transaction. At this point, the SATA DMA samples the “single” status flag and completes the block transfer using single transactions; again refer to *Section : Example 4 on page 971* and *Section : Example 5 on page 974*.

The peripheral asserts a single status flag to indicate to the SATA DMA that there is enough data or space to complete a single transaction from or to the source/destination peripheral.

Note: For hardware handshaking, the single status flag is a signal on the hardware handshaking interface; refer to Section 79.3.6.3: Hardware Handshaking – Peripheral Is Not Flow Controller on page 955. For software handshaking, the single status flag is one of the software handshaking interface registers; refer to Section 79.3.6.4: Software Handshaking – Peripheral Is Not Flow Controller on page 960.

The Single Transaction Region is the time interval where the SATA DMA uses single transactions to complete the block transfer; burst transactions are exclusively used outside this region.

Note: Burst transactions can also be used in this region; for more information, refer to “Section 79.3.6.2: Early-Terminated Burst Transaction on page 955

The Single Transaction Region applies to only a peripheral that is not the flow controller. The precise definition of when this region is entered is dependent on what acts as the flow controller:

- The SATA DMA is the flow controller – The source peripheral enters the Single Transaction Region when the number of bytes left to complete in the source block transfer is less than src_burst_size_bytes. If:

  blk_size_bytes/src_burst_size_bytes = integer (8)

  then the source never enters this region, and the source block uses only burst transactions.

  The destination peripheral enters the Single Transaction Region when the number of bytes left to complete in the destination block transfer is less than dst_burst_size_bytes. If:

  blk_size_bytes/dst_burst_size_bytes = integer (9)

  then the destination never enters this region, and the destination block uses only burst transactions.

Note: The above conditions cause a peripheral to enter the Single Transaction Region. When the peripheral is outside the Single Transaction Region, then the SATA DMA responds to only burst

transaction requests. Whether the peripheral knows that it is in the Single Transaction Region or not, it must always generate burst requests outside the Single Transaction Region, or the DMA block transfer stalls. Once in the Single Transaction Region, the SATA DMA can complete the block transfer using single transactions.

- Either the source or destination peripheral is the flow controller – The destination or source peripheral enters the Single Transaction Region when the flow control peripheral – that is, the source or destination – signals the last transaction in the block and when the amount of data left to be transferred in the destination/source block is less than that which is specified by dst_burst_size_bytes/src_burst_size_bytes.

79.3.6.2 Early-Terminated Burst Transaction

When a source or destination peripheral is in the Single Transaction Region, a burst transaction can still be requested. However, src_burst_size_bytes/ dst_burst_size_bytes is greater than the number of bytes left to complete in the source/destination block transfer at the time that the burst transaction is triggered. In this case, the burst transaction is started and “early-terminated” at block completion without transferring the programmed amount of data – that is, src_burst_size_bytes or dst_burst_size_bytes – but only the amount required to complete the block transfer. An Early-Terminated Burst Transaction occurs between the SATA DMA and the peripheral only when the peripheral is not the flow controller.

79.3.6.3 Hardware Handshaking – Peripheral Is Not Flow Controller

Figure 285 illustrates the hardware handshaking interface between a peripheral – whether a destination or source – and the SATA DMA when the peripheral is not the flow controller.

Figure 285: Hardware Handshaking Interface

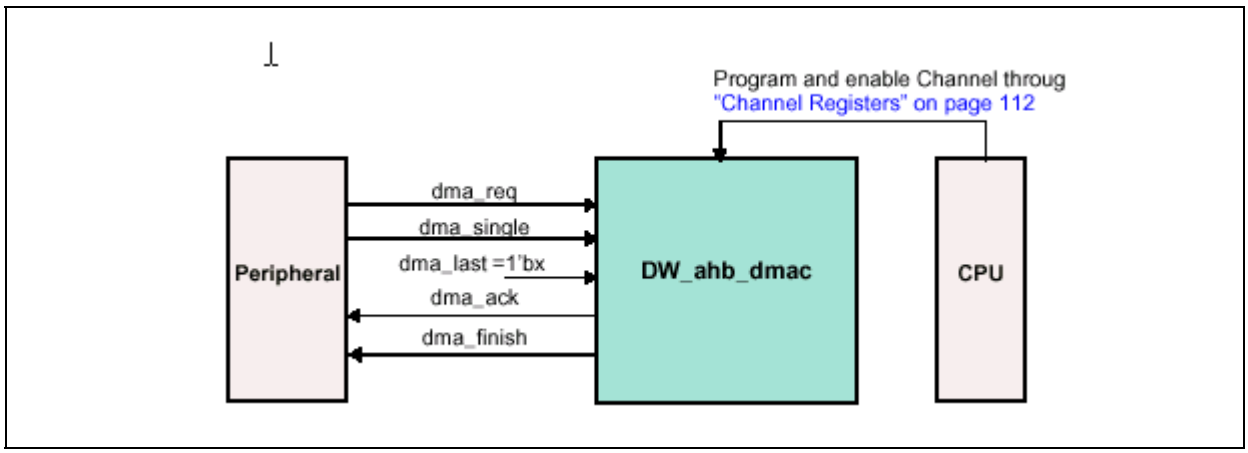


Table 222 describes the hardware handshaking signals in the case where the peripheral is not the flow controller; that is, where either the SATA DMA or the other peripheral is the flow

controller.Signal polarity can be programmed using the SADMA_CFG0.SRC_HS_POL and SADMA_CFG0.DST_HS_POL fields.

Table 222: Hardware Handshaking Interface

Signal	Direction	Description
dma_ack	Output	SATA DMA acknowledge signal to peripheral. The dma_ack signal is asserted after the data phase of the last bus transfer in the current transaction – single or burst – to the peripheral that has completed. For a single transaction, dma_ack remains asserted until the peripheral de-asserts dma_single; dma_ack is de-asserted one hclk cycle later. For a burst transaction, dma_ack remains asserted until the peripheral de-asserts dma_req; dma_ack is de-asserted one hclk cycle later.
dma_finish	Output	SATA DMA asserts dma_finish to signal block completion. This has the same timing as dma_ack and forms a handshaking loop with dma_req if the last transaction in the block was a burst transaction, or with dma_single if the last transaction in the block was a single transaction. There is an exception to the above timing definition when dma_finish interfaces with a source peripheral when the destination peripheral is the flow controller; for details, refer to*Section : Example 7 on page 978* Case 1b.
dma_last	Input	Since the peripheral is not the flow controller, dma_last is not sampled by the SATA DMA and this signal is ignored.
dma_req	Input	Burst transaction request from peripheral. The SATA DMA always interprets the dma_req signal as a burst transaction request, regardless of the level of dma_single. The SATA DMA uses rising-edge detection of the dma_req signal in order to identify a burst request on the channel. Once asserted by the peripheral, dma_req should remain asserted until the SATA DMA asserts dma_ack. Upon receiving the dma_ack signal from the SATA DMA to indicate the burst transaction is complete, the peripheral should de-assert the burst request signal, dma_req. Once dma_req is de-asserted by the peripheral, the SATA DMA de-asserts dma_ack. If an active edge of dma_req is detected in the Single Transaction Region, then the block is completed using an Early-Terminated Burst Transaction.
dma_single	Input	Single transfer status. The dma_single signal is a status signal that is asserted by a destination peripheral when it can accept at least one destination data item; otherwise it is cleared. For a source peripheral, the dma_single signal is again a status signal and is asserted by a source peripheral when it can transmit at least one source data item; otherwise it is cleared. Once asserted, dma_single must remain asserted until dma_ack is asserted, at which time the peripheral should de-assert dma_single. This signal is sampled by the SATA DMA only in the Single Transaction Region of the block transfer. Outside of this region, dma_single is ignored and all transactions are burst transactions.

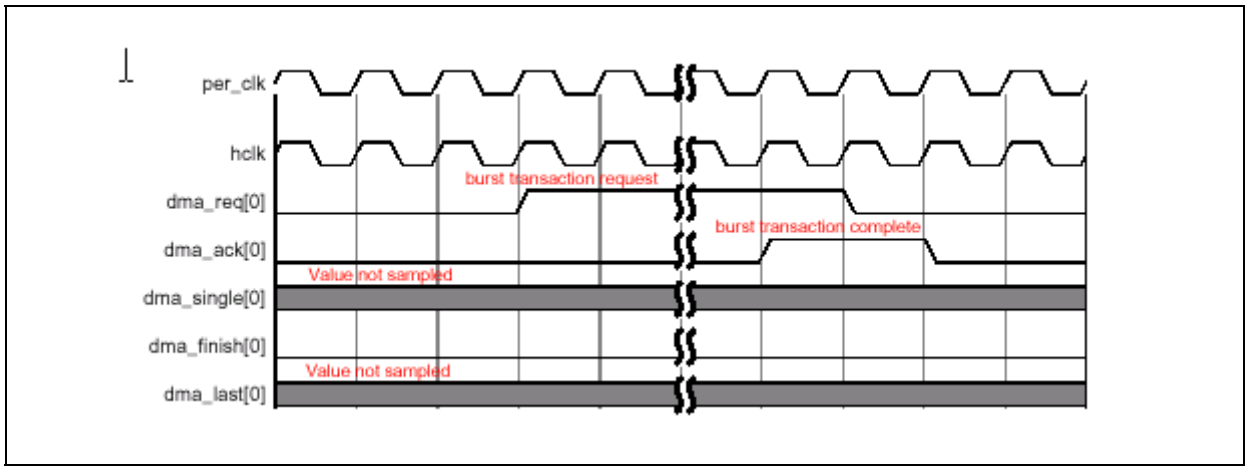
Figure 286 shows the timing diagram of a burst transaction where the peripheral clock, per_clk, equals hclk. In this example, the peripheral is outside the Single Transaction Region, and therefore the SATA DMA does not sample dma_single[0].

The handshaking loop is as follows:

dma_req asserted by peripheral

- dma_ack asserted by SATA DMA
- dma_req de-asserted by peripheral
- dma_ack de-asserted by SATA DMA.

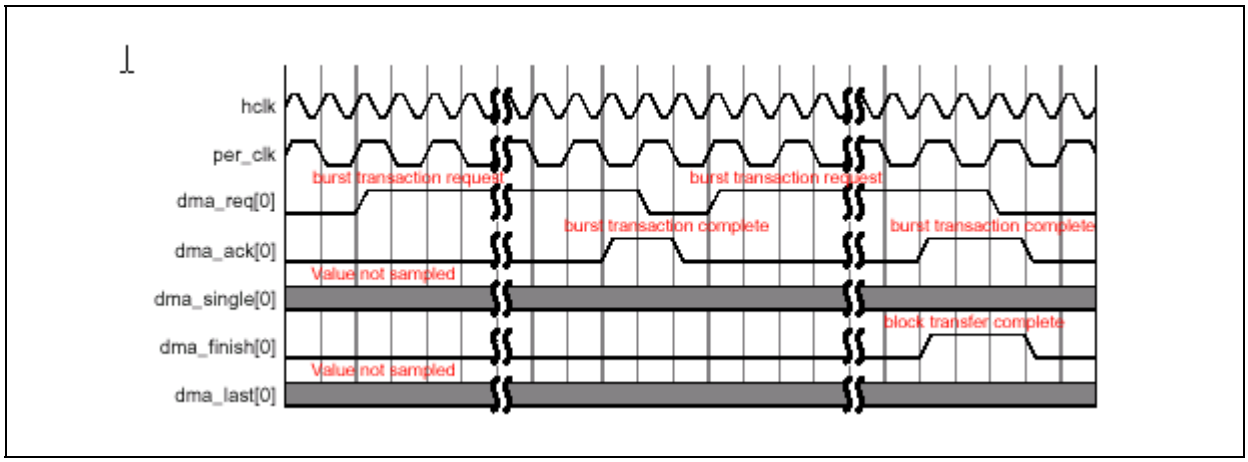
Figure 286: Burst Transaction – pclk = hclk



Note: The per_clk signal is equal to hclk if the peripheral is an AHB peripheral; it is equal to pclk if the peripheral is an APB peripheral. The burst transaction request signal, dma_req, and the single status signal, dma_single, are generated in the peripheral of per_clk and sampled by hclk in the SATA DMA. The acknowledge signal, dma_ack, is generated in the SATA DMA of hclk and sampled in the peripheral by per_clk. The handshaking mechanism between the SATA DMA and the peripheral supports quasi-synchronous clocks; that is, hclk and per_clk must be phase-aligned, and the hclk frequency must be a multiple of the per_clk frequency.

Figure 287 shows two back-to-back burst transactions at the end of a block transfer where the hclk frequency is twice the pclk frequency; the peripheral is an APB peripheral. The second burst transaction terminates the block, and dma_finish[0] is asserted to indicate block completion.

Figure 287: Back-to-Back Burst Transactions – hclk = 2*per_clk



There are two things to note when designing the hardware handshaking interface:

- Once asserted, the dma_req burst request signal must remain asserted until the corresponding dma_ack signal is received, even if the condition that generates dma_req in the peripheral is False.
- The dma_req signal should be de-asserted when dma_ack is asserted, even if the condition that generates dma_req in the peripheral is True.

Figure 288 shows a single transaction that occurs in the Single Transaction Region. The handshaking loop is as follows:

dma_single asserted by peripheral

- dma_ack asserted by SATA DMA
- dma_single de-asserted by peripheral
- dma_ack de-asserted by SATA DMA

Figure 288: Single Transaction

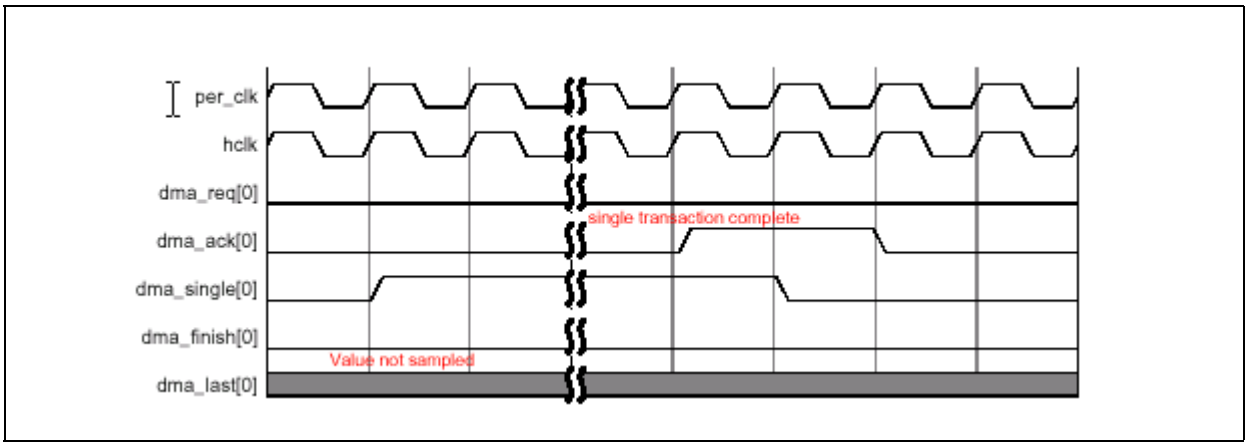
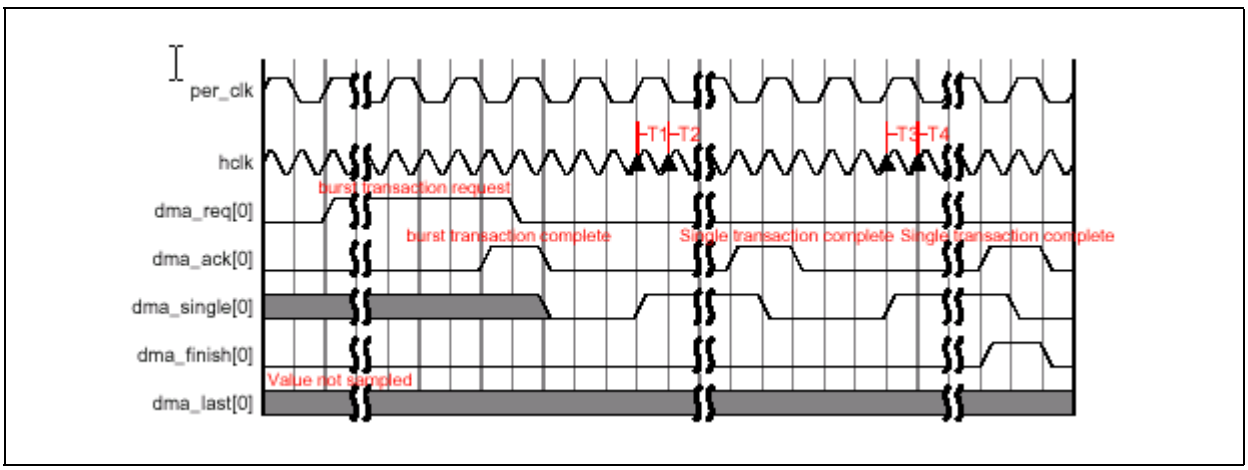


Figure 289 shows a burst transaction, followed by two back-to-back single transactions, where the hclk frequency is twice the per_clk frequency; handshaking interface 0 is used. After the first burst transaction, the peripheral enters the Single Transaction Region and the SATA DMA starts sampling dma_single[0]. On hclk edges T2 and T4, the SATA DMA samples that dma_single[0] is asserted and performs single transactions. The second single transaction terminates the block transfer; dma_finish[0] is asserted to indicate block completion.

Figure 289: Burst Followed by Back-to-Back Single Transactions



In the Single Transaction Region, if an active edge on dma_req and dma_single occur on the same cycle – or if the active edge on dma_single occurs only one cycle before an active edge on dma_req – then the burst transaction takes precedence over the single transaction, and the block would be completed using an Early-Terminated Burst Transaction. If SATA DMA samples that dma_req[0] was asserted on hclk cycles T1-T2 or T3-T4 in *Figure 289*, then the burst request takes precedence. In *Figure 290*, a rising edge on dma_req[0] at time T4 takes precedence over the rising edge on dma_single[0] at time T3.

Figure 290: Early-Terminated Burst Transaction

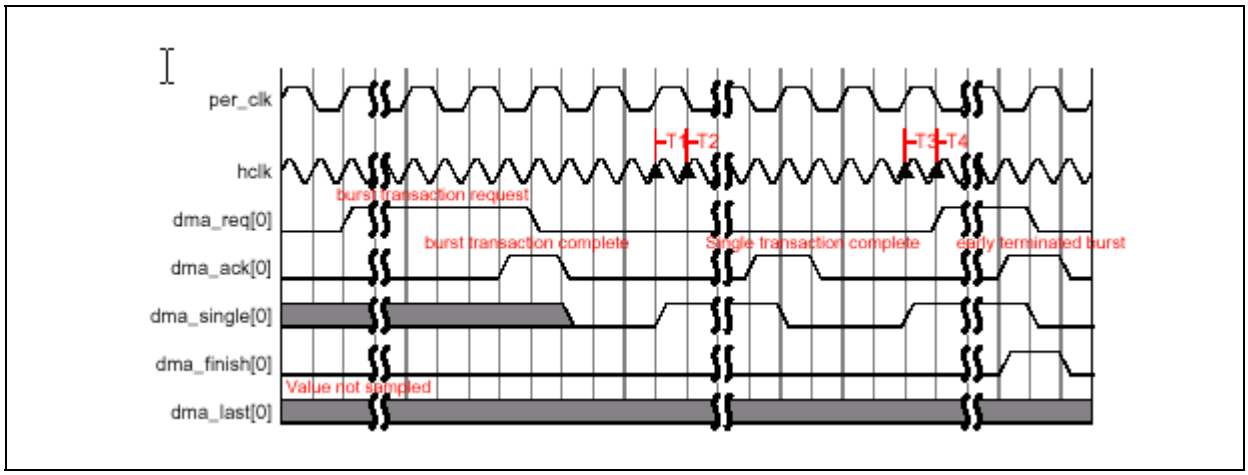
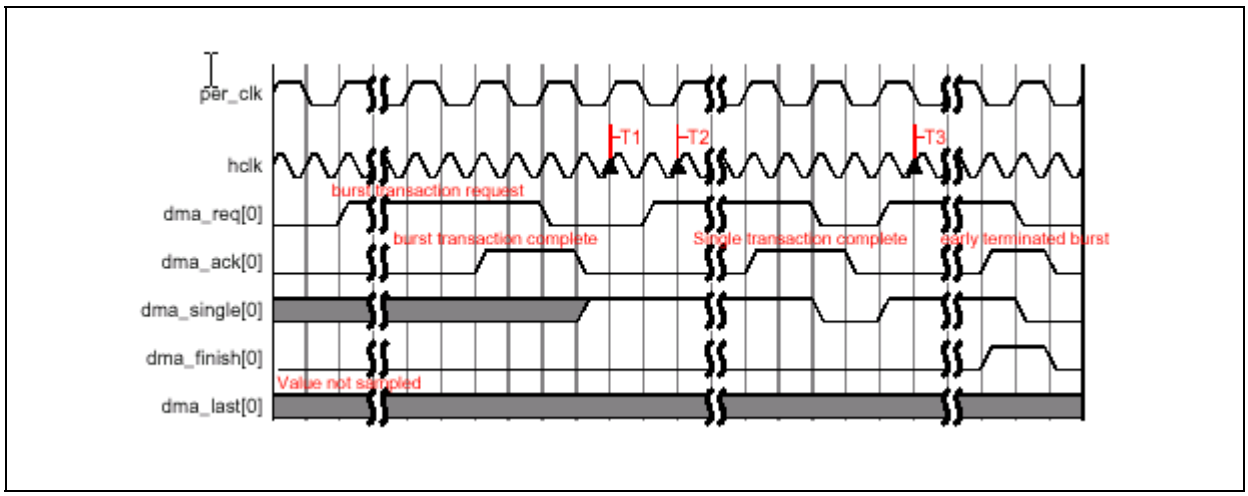


Figure 291 shows a burst transaction followed by a single transaction followed by a burst transaction at the end of a block. After the first burst transaction completes the peripheral is in the Single Transaction Region and SATA DMA samples that dma_single[0] is asserted at T1. dma_req[0] is triggered in the middle of this single transaction at time 'T2'. This burst transaction request is ignored and is not serviced. An active edge on dma_req[0] is re-generated and sampled by SATA DMA at time T3. This burst transaction completes the block transfer using an Early-Terminated Burst Transaction.

Figure 291: Burst Transaction ignored During Active Single Transaction



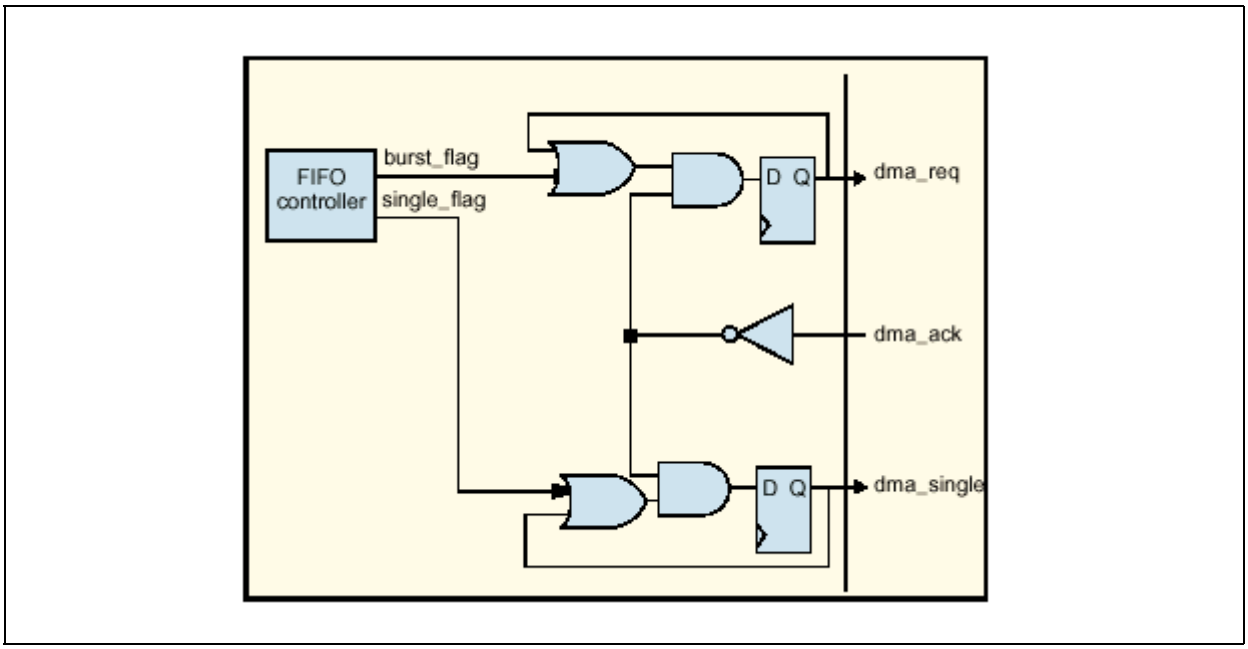
Generating dma_req and dma_single Hardware Handshaking Signals

Figure 292 illustrates a suggested method of generating dma_req and dma_single for a source peripheral when the peripheral is not the flow controller. The single_flag signal in *Figure 292* is asserted when the source FIFO has at least one source data item in the FIFO. The burst_flag signal in *Figure 292* is asserted when the source FIFO contains data items greater than or equal to some watermark-level number of data items in it.

Note: Figure 292 shows dma_req and dma_single being de-asserted when dma_ack is asserted. It also shows how, once asserted, dma_req and dma_single remain asserted until dma_ack is asserted. The example assumes active-high handshaking.

The destination peripheral dma_req and dma_single signals can be generated in a similar fashion, but in this case the single_flag signal in *Figure 292* is asserted when the destination FIFO has at least one free location. The burst_flag signal in *Figure 292* is asserted when the destination FIFO contains free locations greater than or equal to some watermark-level number.

Figure 292: Generation of dma_req and dma_single by Source



79.3.6.4 Software Handshaking – Peripheral Is Not Flow Controller

When the peripheral is not the flow controller, then the last transaction registers – LstSrcReg and LstDstReg – are not used, and the values in these registers are ignored.

Operation – Peripheral Not In Single Transaction Region

Writing a 1 to the ReqSrcReg[x]/ReqDstReg[x] register is always interpreted as a burst transaction request, where x is the channel number. However, in order for a burst transaction request to start, software must write a 1 to the SglReqSrcReg[x]/ SglReqDstReg[x] register.

You can write a 1 to the SglReqSrcReg[x]/SglReqDstReg[x] and ReqSrcReg[x]/ReqDstReg[x] registers in any order, but both registers must be asserted in order to initiate a burst transaction. Upon completion of the burst transaction, the hardware clears the SglReqSrcReg[x]/ SglReqDstReg[x] and ReqSrcReg[x]/ReqDstReg[x] registers.

Operation – Peripheral In Single Transaction Region

Writing a 1 to the SglReqSrcReg/SglReqDstReg initiates a single transaction. Upon completion of the single transaction, both the SglReqSrcReg/SglReqDstReg and ReqSrcReg/ReqDstReg bits are cleared by hardware. Therefore, writing a 1 to the ReqSrcReg/ReqDstReg is ignored while a single transaction has been initiated, and the requested burst transaction is not serviced.

Again, writing a 1 to the ReqSrcReg/ReqDstReg register is always a burst transaction request. However, in order for a burst transaction request to start, the corresponding channel bit in the SglReqSrcReg/SglReqDstReg must be asserted. Therefore, to ensure that a burst transaction is serviced in this region, you must write a 1 to the ReqSrcReg/ReqDstReg before writing a 1 to the SglReqSrcReg/SglReqDstReg register.

If the programming order is reversed, a single transaction is started instead of a burst transaction. The hardware clears both the ReqSrcReg/ReqDstReg and the SglReqSrcReg/ SglReqDstReg registers after the burst transaction request completes. When a burst transaction is initiated in the Single Transaction Region, then the block completes using an Early-Terminated Burst Transaction.

Software can poll the relevant channel bit in the SglReqSrcReg/SglReqDstReg and ReqSrcReg/ ReqDstReg registers. When both are 0, then either the requested burst or single transaction has completed. Alternatively, the IntSrcTran or IntDstTran interrupts can be enabled and unmasked in order to generate an interrupt when the requested source or destination transaction has completed.

Note: The transaction-complete interrupts are triggered when both single and burst transactions are complete. The same transaction-complete interrupt is used for both single and burst transactions.

79.3.6.5 Single Transactions – Peripheral Is Not Flow Controller

When the source peripheral is not flow controller, the source peripheral can hardcode dma_single to an inactive level (hardware handshaking), or software will never need to initiate single transactions from the source (software handshaking). This can happen if either of the following is true:

- Block size is a multiple of the burst transaction length.
  - If SATA DMA is flow controller
    blk_size_bytes_dma/src_burst_size_bytes = integer
  - If the destination peripheral is flow controller
    blk_size_bytes_dst/src_burst_size_bytes = integer or
- Block size is not a multiple of the burst transaction length, but the peripheral can dynamically adjust the watermark level that triggers a burst request in order to enable block completion.

When the destination peripheral is not a flow controller, then the destination peripheral may hardcode dma_single to an inactive level (hardware handshaking), or software will never need to initiate single transactions to the destination (software handshaking). This can happen when any of the following are true:

- Block size is a multiple of the burst transaction length.
  - If SATA DMA is flow controller:
    block_size_bytes_dma/dst_burst_size_bytes = integer or
  - If the source peripheral is flow controller:
    block_size_bytes_src/dst_burst_size_bytes = integer or
- The destination peripheral can dynamically adjust the watermark level upwards so that a burst request is triggered in order to enable a destination block completion or
- It is guaranteed that data at some point will be extracted from the destination FIFO in the “Single transaction region” in order to trigger a burst transaction.

Note: The destination peripheral requires data to be extracted in order to empty the destination FIFO below a watermark level for triggering a destination burst request. If it is guaranteed that data at some point will be extracted from the destination FIFO in the Single Transaction Region in order to trigger a dma_req, then in this case, the dma_single signal from the destination can be tied to an inactive level, and the destination block completes with an Early-Terminated Burst Transaction.

If none of the above are true, then a series of burst transactions followed by single transactions is needed to complete the source/destination block transfer; for more information, refer to *Section : Example 4 on page 971* and *Section : Example 5 on page 974*.

79.3.7 Handshaking Interface – Peripheral Is Flow Controller

When the peripheral is the flow controller, it controls the length of the block and must communicate to the SATA DMA when the block transfer is complete. The peripheral does this by telling the SATA DMA that the current transaction – burst or single – is the last transaction in the block. When the peripheral is the flow controller and the block size is not a multiple of the SADMA_CTRL0.SRC_MSIZE/SADMA_CTRL0.DEST_MSIZE, then the peripheral must use single transactions to complete a block transfer.

Note: Since the peripheral can terminate the block on a single transaction, there is no notion of a Single Transaction Region such as there is when the peripheral is not the flow controller.

When the peripheral is the flow controller, it indicates directly to SATA DMA which type of transaction – single or burst to perform. Where possible, the SATA DMA uses the maximum possible bus burst length. It can also lock the arbitration for the master bus so that a channel is permanently granted the master bus interface. The SATA DMA can also assert the hlock signal to lock the DW_ahb system arbiter. Refer to *Section 79.3.11.1: Locked DMA Transfers on page 1002*.

79.3.7.1 Hardware Handshaking – Peripheral Is Flow Controller

Figure 293 shows the hardware handshaking interface between a destination or source peripheral and the SATA DMA when the peripheral is the flow controller.

Figure 293: Hardware Handshaking Interface

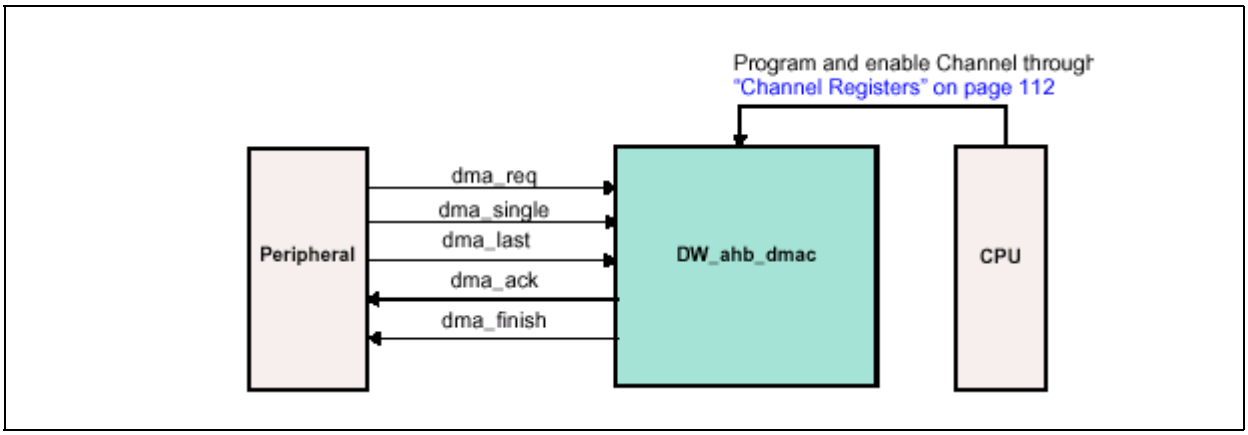


Table 223 describes the hardware handshaking interface signals operation in the case where the peripheral is flow controller. Timing diagrams are illustrated in *Figure 294* and *Figure 295*.

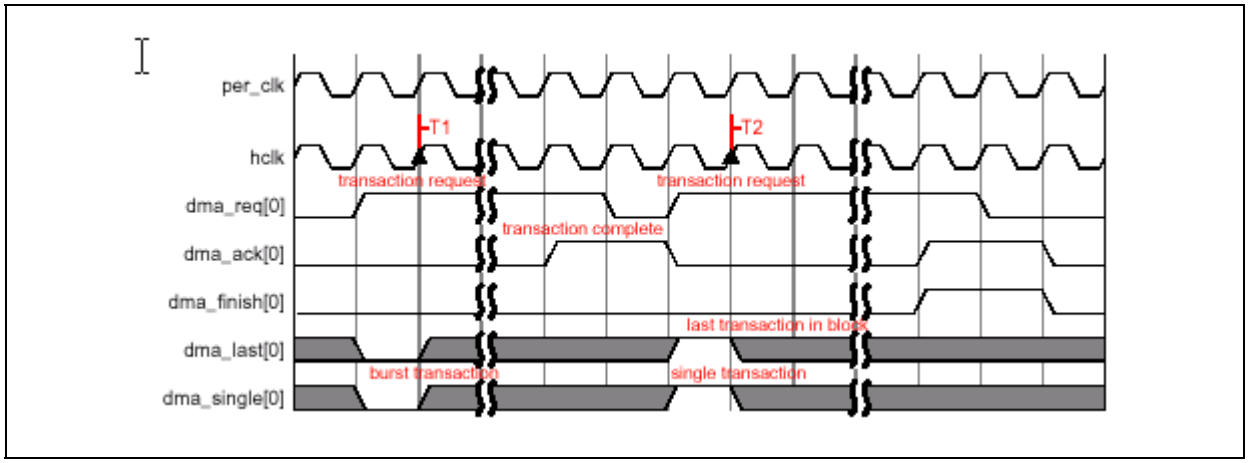
Table 223: Hardware Handshaking Interface

Signal	Direction	Description
dma_ack	Output	SATA DMA acknowledge signal to the peripheral. This is asserted after the data phase of the last bus transfer in the current transaction (single or burst) to the peripheral has completed. It forms a handshaking loop with dma_req and remains asserted until the peripheral de-asserts dma_req (de-asserted one hclk cycle later).
dma_finish	Output	SATA DMA block transfer complete signal. The SATA DMA asserts dma_finish in order to signal block completion. This uses the same timing as dma_ack and forms a handshaking loop with dma_req.
dma_last	Input	Last transaction in block. When the peripheral is the flow controller, it asserts dma_last on the same cycle as dma_req is asserted in order to signal that this transaction request is the last in the block; the block transfer is complete after this transaction is complete. If dma_single is high in the same cycle, then the last transaction is a single transaction. If dma_single is low in the same cycle, then the last transaction is a burst transaction.
dma_req	Input	Transaction request from peripheral. A rising edge on dma_req initiates a transaction request. The type of transaction – single or burst – is qualified by dma_single. Once dma_req is asserted, it must remain asserted until dma_ack is asserted. When the peripheral that is driving dma_req determines that dma_ack is asserted, it must de-assert dma_req.
dma_single	Input	Single or burst transaction request. If dma_single is de-asserted in the same clock cycle as a rising edge on dma_req, a burst transaction is requested by the peripheral. If asserted, the peripheral requests a single transaction.

The following timing diagrams assume that handshaking interface 0 is active-high.

Figure 294 shows a burst transaction followed by a single transaction, where the single transaction is the last in the block. On clock edge T1, SATA DMA samples that dma_req[0] is asserted, dma_single[0] is de-asserted, and dma_last[0] is de-asserted. This is a request for a burst transaction, which is not the last transaction in the block.

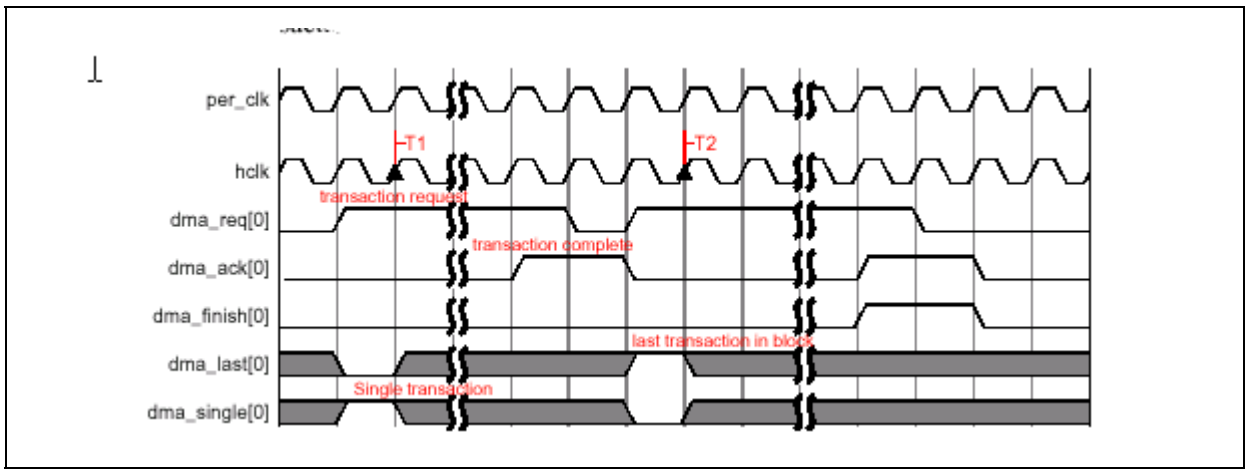
Figure 294: Burst Transaction Followed by Single Transaction that Terminates Block



On clock edge T2, the SATA DMA samples that dma_req[0], dma_single[0], and dma_last[0] are all asserted. This is a request for a single transaction, which is the last transaction in the block. The dma_last[0] and dma_single[0] signals need only be valid on the same clock cycle that dma_req is generated.

Similarly, *Figure 295* shows a single transaction followed by a burst transaction, where the burst transaction is the last transaction in the block.

Figure 295: Single Transaction Followed by Burst Transaction that Terminates Block



79.3.7.2 Single Transactions – Peripheral is Flow Controller

When the source peripheral is the flow controller, then it can hardcode dma_single to an inactive level (hardware handshaking), or software will never need to initiate single transactions from the source (software handshaking). This occurs when:

block_size_bytes_src/src_burst_size_bytes = integer (10)

When the destination peripheral is flow controller then the destination peripheral may hardcode dma_single to an inactive level (hardware handshaking) or software will never need to initiate single transactions to the destination (software handshaking) when:

block_size_bytes_dst/dst_burst_size_bytes = integer (11)

79.3.8 Setting Up Transfers

Transfers are set up by programming fields of the SADMA_CTRL0 and SADMA_CFG0 registers for that channel. A single block is made up of numerous transactions – single and burst – which are in turn composed of bus transfers. A peripheral requests a transaction through the handshaking interface to the SATA DMA; for more information, refer to *Section 79.3.3: Handshaking Interface on page 952*.

The operation of the handshaking interface is different and depends on what is acting as the flow controller.

Table 224 lists the parameters that are investigated in the following examples. The effects of these parameters on the flow of the block transfer are highlighted. In addition to the software parameters, it includes the channel FIFO depth, DMAH_CHx_FIFO_DEPTH, which is configurable only in coreConsultant.

For definitions of the register parameters, refer to*Chapter 81: Serial ATA (SATA) registers*, *Section 81.1: DMA controller on page 1022*; for definitions of SATA DMA parameters, refer to *Section 79.1: Configuration Parameters on page 913*.

Table 224: Parameters Used in Transfer Examples

Parameter	Description
DMAH_CHx_FIFO_DEPTH	Channel x FIFO depth in bytes
SADMA_CTRL0.TT_FC	Transfer type and flow control
SADMA_CTRL0.BLOCK_TS	Block transfer size
SADMA_CTRL0.SRC_TR_WIDTH	Source transfer width
SADMA_CTRL0.DST_TR_WIDTH	Destination transfer width
SADMA_CTRL0.SRC_MSIZE	Source burst transaction length
SADMA_CTRL0.DEST_MSIZE	Destination burst transaction length
SADMA_CFG0.MAX_ABRST	Maximum bus burst length
SADMA_CFG0.FIFO_MODE	FIFO mode select
SADMA_CFG0.FCMODE	Flow-control mode

79.3.8.1 Transfer Operation

The following examples show the effect of different settings of each parameter from *Table 224* on a DMA block transfer. In all examples, it is assumed that no bus bursts are early-terminated by the DW_ahb system arbiter, unless otherwise stated. Example 1 through Example 8 use hardware handshaking on both the source and destination side. Example 9 through Example 11 use software handshaking on both the source and destination side.

The following is a brief description of each of the examples:

- *Example 1 on page 966* – Block transfer when the SATA DMA is the flow controller.
- *Example 2 on page 967* – Effect of DMAH_CHx_FIFO_DEPTH on a block transfer.
- *Example 3 on page 968* – Effect of maximum bus burst length, SADMA_CFG0.MAX_ABRST, on a block transfer.
- *Example 4 on page 971* – Block transfer when the SATA DMA is the flow controller and the source peripheral enters the Single Transaction Region.
- *Example 5 on page 974* – Block transfer when the SATA DMA is the flow controller and the destination peripheral enters the Single Transaction Region. Also demonstrates channel FIFO flushing to the destination peripheral at the end of a block transfer.
- *Example 6* – Effect of SADMA_CFG0.FIFO_MODE on a block transfer.
- *Example 7 on page 978* – Block transfer when the destination peripheral is the flow controller and data pre-fetching from the source is enabled; SADMA_CFG0.FCMODE = 0.
- *Example 8 on page 985* – Block transfer when the destination peripheral is the flow controller and data pre-fetching from the source is disabled; SADMA_CFG0.FCMODE = 1.
- *Example 9 on page 987* – Block transfer when the SATA DMA is the flow controller and software handshaking is used on both the source and destination side.

- *Example 10 on page 989* – Block transfer when the SATA DMA is the flow controller and software handshaking is used on both the source and destination side; the source enters the Single Transaction Region.
- *Example 11 on page 991*– Block transfer when the source peripheral is the flow controller and software handshaking is used on both the source and destination side; the destination peripheral enters the Single Transaction Region. The SATA DMA is programmed with the number of data items that are to be transferred for each burst transaction request, SADMA_CTRL0.SRC_MSIZE /SADMA_CTRL0.DEST_MSIZE. Similarly, the width of each data item in the transaction is set by the SADMA_CTRL0.SRC_TR_WIDTH and SADMA_CTRL0.DST_TR_WIDTH fields.

Example 1

Scenario: Example block transfer when the SATA DMA is the flow controller. This example is the same for both software and hardware handshaking interfaces. *Table 225* lists the DMA parameters for this example.

Table 225: Parameters in Transfer Operation – Example 1

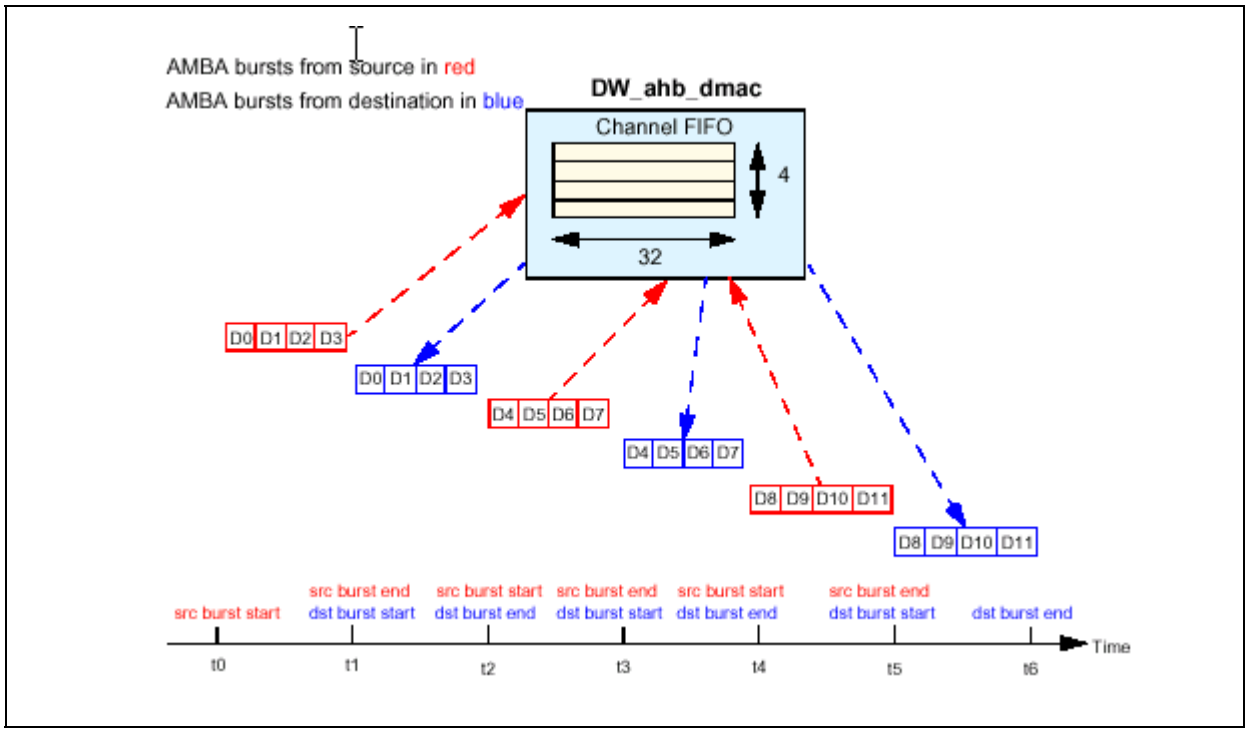
Parameter	Description
SADMA_CTRL0.TT_FC = 3'b011	Peripheral-to-peripheral transfer with SATA DMA as flow controller
SADMA_CTRL0.BLOCK_TS = 12	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b010	32 bit

Table 226: Parameters in Transfer Operation – Example 1 (Continued)

Parameter	Description
SADMA_CTRL0.SRC_MSIZE = 3'b 001	Source burst transaction length = 4
SADMA_CTRL0.DEST_MSIZE = 3'b 001	Destination burst transaction length = 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 16 bytes	–

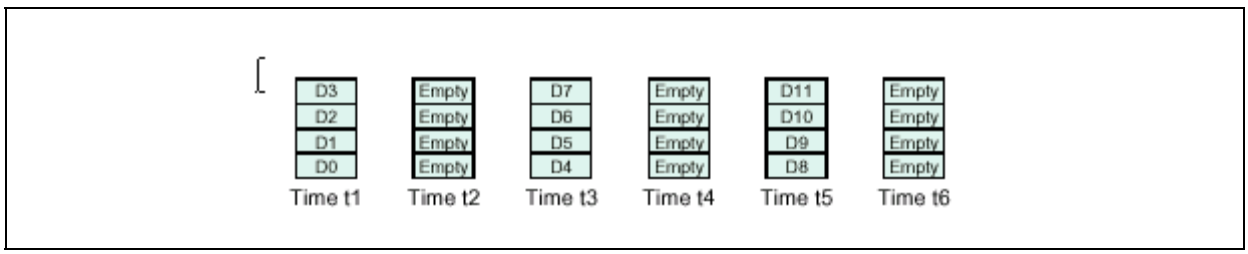
Using equation (5), a total of 48 bytes are transferred in the block; that is, blk_size_bytes_dma = 48. As shown in *Figure 296*, this block transfer consists of 3 bursts of length 4 from the source, interleaved with 3 bursts, again of length 4, to the destination.

Figure 296: Breakdown of Block Transfer



The channel FIFO is alternatively filled by an bus burst from the source and emptied by an bus burst to the destination until the block transfer has completed, as shown in *Figure 297*.

Figure 297: Channel FIFO Contents at Times Indicated in *Figure 296*



Burst transactions are completed in one bus burst. Additionally, because (8) and (9) are both true, neither the source or destination peripherals enter their Single Transaction Region at any stage throughout the DMA transfer and the block transfer from the source and to the destination consists of burst transactions only.

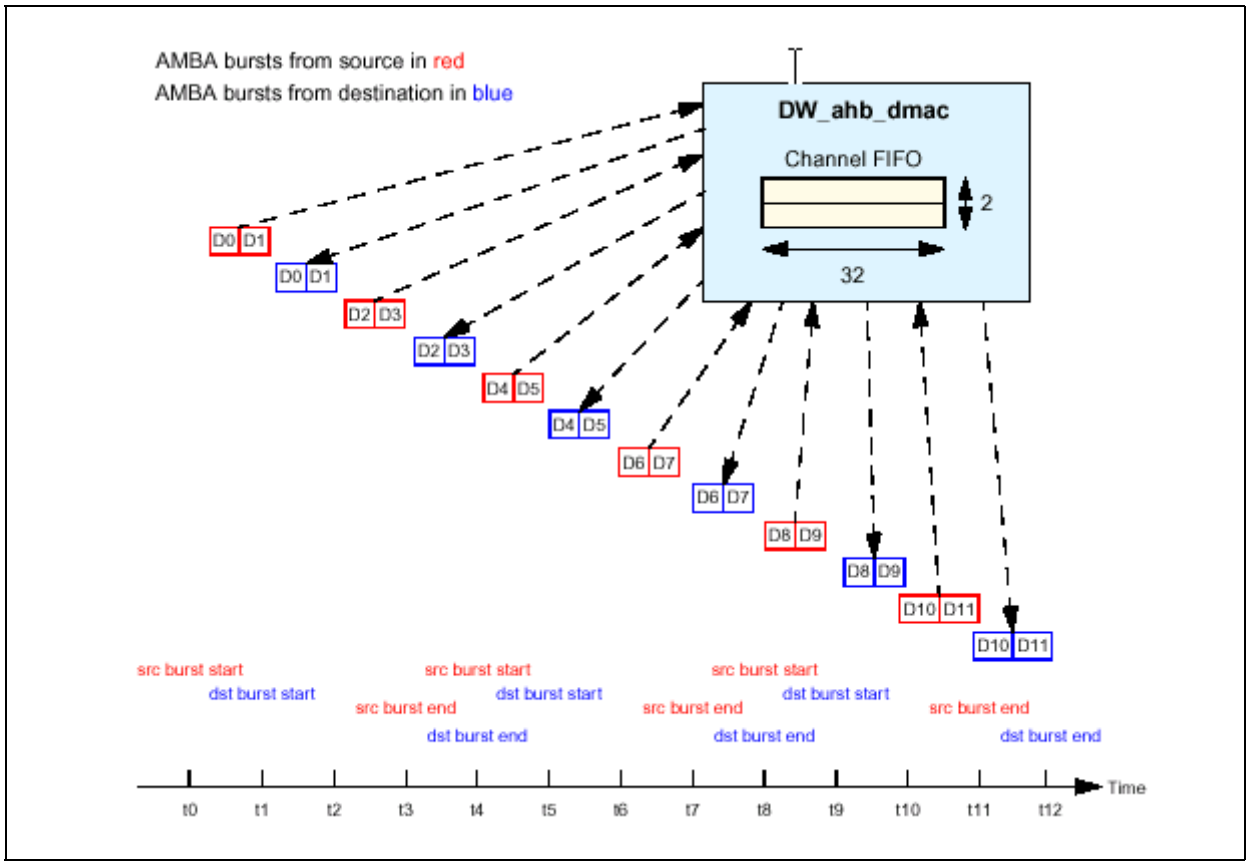
Example 2

Scenario: Effect of DMAH_CHx_FIFO_DEPTH on block transfers. This example is the same for both software and hardware handshaking interfaces.

In this example, the coreConsultant DMAH_CHx_FIFO_DEPTH parameter is changed to 8 bytes, and all other parameters are left unchanged from Example 1, *Table 225: Parameters in Transfer Operation – Example 1 on page 966*.

Example 1 shows the source and destination burst transactions completing in a single bus burst. In general, a burst transaction may take multiple bus bursts to complete. With the DMAH_CHx_FIFO_DEPTH parameter set to 8 bytes instead of 16 bytes, the block transfer would look like that shown in *Figure 298*.

Figure 298: Breakdown of Block Transfer for DMAH_CH_FIFO_DEPTH=8



The block transfer consists of 6 bus bursts of length 2 from the source, interleaved with 6 bus bursts – again of length 2 – to the destination, as shown in *Figure 298*. The channel FIFO is alternatively filled by an bus burst from the source and emptied by an bus burst to the destination, until the block transfer has completed. In this example, a transfer of each source or destination burst transaction is made up of two bus bursts, each of length 2.

Therefore, Example 2 has twice the number of bus bursts per block than Example 1.

Recommendation: To allow a burst transaction to complete in a single bus burst, the SATA DMA channel FIFO depth should be large enough to accept an amount of data equal to an entire burst transaction. Therefore, in order to allow both source and destination burst transactions to complete in a one bus burst:

DMAH_CHx_FIFO_DEPTH >= max(src_burst_size_bytes, dst_burst_size_bytes)(12)

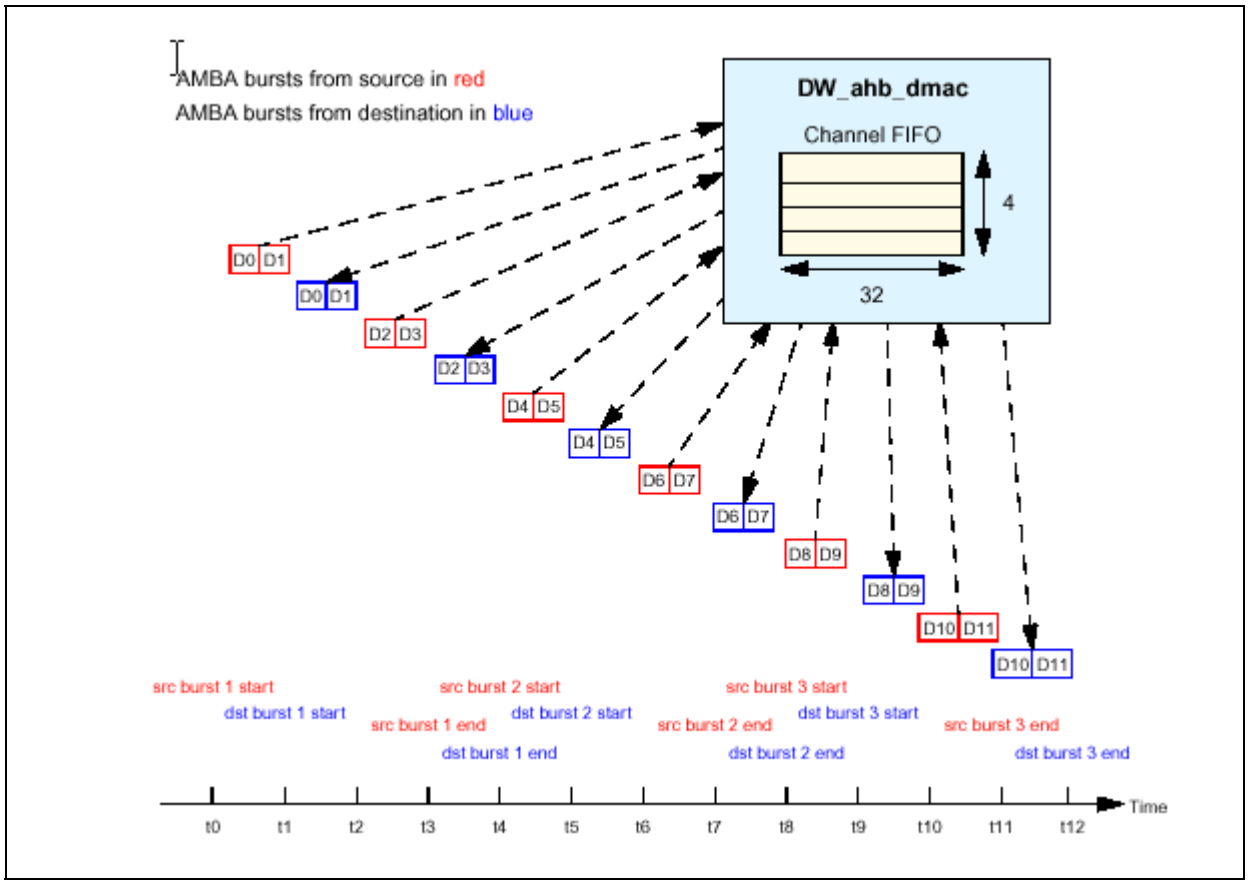
Adhering to the above recommendation results in a reduced number of bus bursts per block, which in turn results in improved bus utilization and lower latency for block transfers.

Example 3

Scenario: Effect of the maximum bus burst length, SADMA_CFG0. MAX_ABRST. This example is the same for both software and hardware handshaking interfaces.

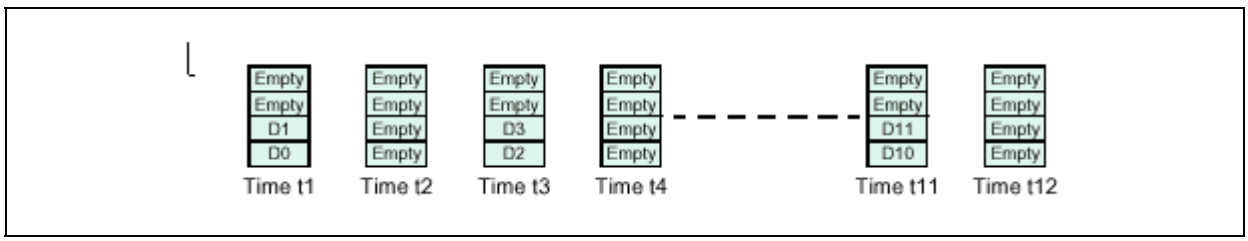
If the parameter SADMA_CFG0. MAX_ABRST = 2 and all other parameters are left unchanged from Example 1, *Table 225: Parameters in Transfer Operation – Example 1 on page 966*, then the block transfer would look like that shown in *Figure 299*.

Figure 299: Breakdown of Block Transfer where max_abrst = 2, Case 1



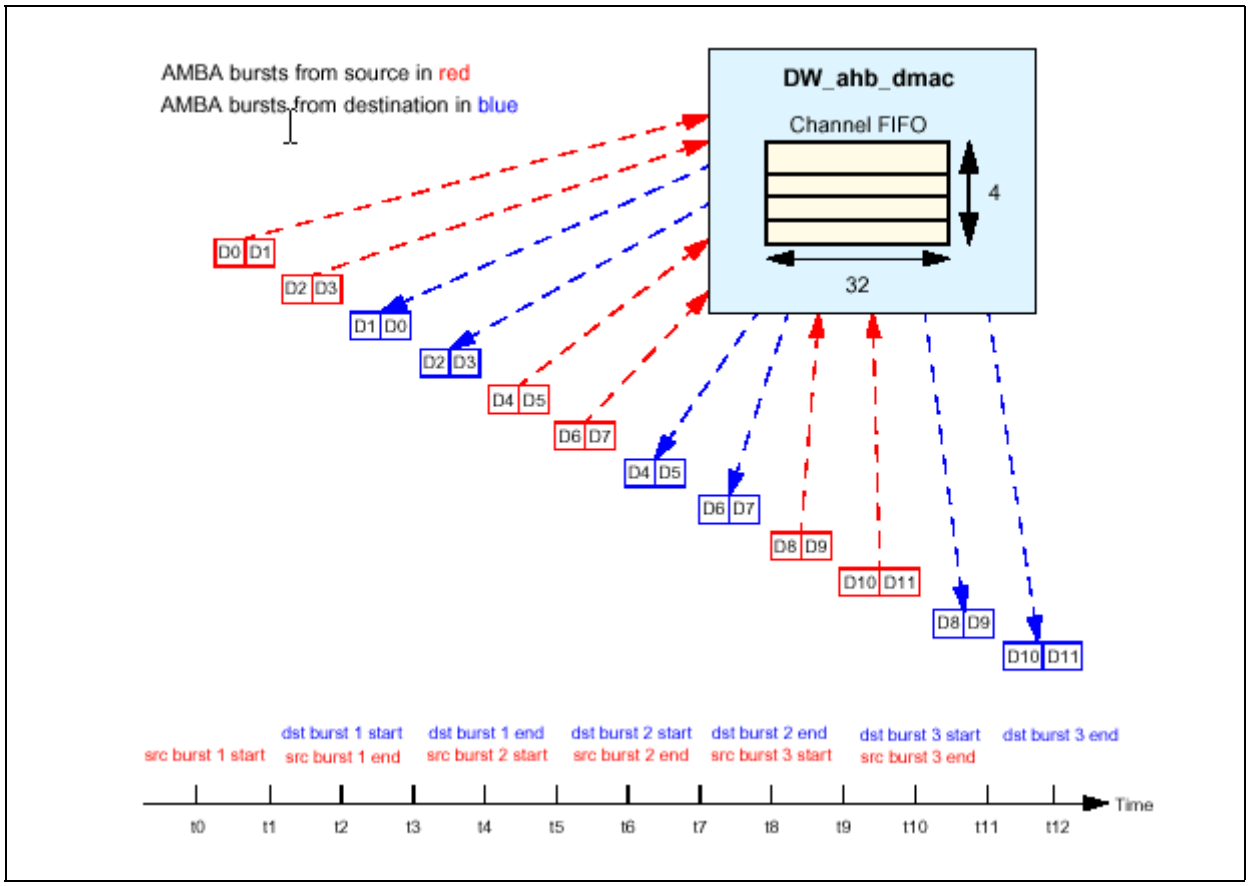
The channel FIFO is alternatively half filled by an bus burst from the source, and then emptied by an bus burst to the destination until the block transfer has completed; this is illustrated in *Figure 300*.

Figure 300: Channel FIFO Contents at Times Indicated in *Figure 299*



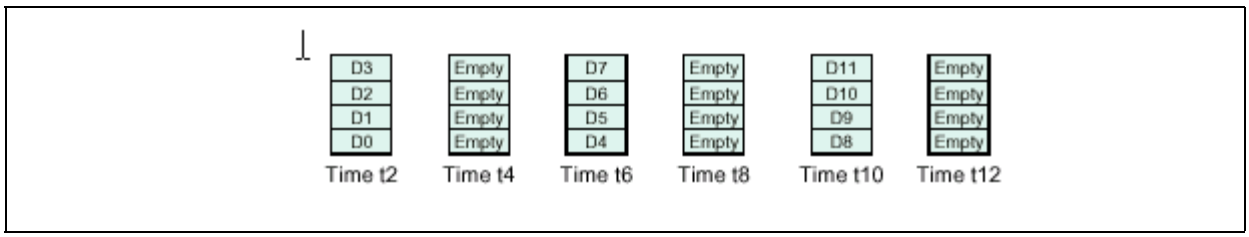
In this example block transfer, each source or destination burst transaction is made up of two bus bursts, each of length 2. As *Figure 300* illustrates, the top two channel FIFO locations are redundant for this block transfer. However, this is not the general case. The block transfer could proceed as indicated in *Figure 301*.

Figure 301: Breakdown of Block Transfer where max_abrst = 2, Case 2



This depends on the timing of the source and destination transaction requests, relative to each other. *Figure 302* illustrates the channel FIFO status for *Figure 301*.

Figure 302: Channel FIFO Contents at Times Indicated in *Figure 301*



Recommendation: To allow a burst transaction to complete in a single bus burst, the following should be true:

SADMA_CFG0.MAX_ABRST >= max(src_burst_size_bytes, dst_burst_size_bytes)

Adhering to the above recommendation results in a reduced number of bus bursts per block, which in turn results in improved bus utilization and lower latency for block transfers. Limiting a bus burst to a maximum length prevents the SATA DMA from saturating the AHB bus when the DW_ahb system arbiter is configured to only allow changing of the grant signals to bus masters at the end of an undefined length burst. It also prevents a channel from saturating a SATA DMA master bus interface. For more information, refer to *Section 79.3.12: Arbitration for AHB Master Interface on page 1003*.

Example 4

Scenario: Source peripheral enters Single Transaction Region; the SATA DMA is the flow controller.

This example is the same for both hardware and software handshaking and demonstrates how a block from the source can be completed using a series of single transactions. It also demonstrates how the watermark level that triggers a burst request in the source peripheral can be dynamically adjusted so that the block transfer from the source completes with an Early-Terminated Burst Transaction. *Table 227* lists the parameters used in this example.

Table 227: Parameters in Transfer Operation – Example 4

Parameter	Comment
SADMA_CTRL0.TT_FC = 3'b011	Peripheral-to-peripheral transfer with SATA DMA as flow controller
SADMA_CTRL0.BLOCK_TS = 12	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.SRC_MSIZE = 3'b010	Source burst transaction length = 8
SADMA_CTRL0.DEST_MSIZE = 3'b001	Destination burst transaction length = 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 16 bytes	–

In this case, BLOCK_TS is not a multiple of the source burst transaction length, SADMA_CTRL0.SRC_MSIZE, so near the end of a block transfer from the source, the amount of data left to be transferred is less than src_burst_size_bytes.

In this example, the block size is a multiple of the destination burst transaction length:

blk_size_bytes_dma/dst_burst_size_bytes = 48/8 = integer

The destination block is made up of 3 burst transactions to the destination and does not enter the Single Transaction Region.

The block size is not a multiple of the source burst transaction length:

blk_size_bytes_dma/src_burst_size_bytes = 48/32 != integer

Consider the case where the watermark level that triggers a source burst request in the source peripheral is equal to SADMA_CTRL0.SRC_MSIZE = 8; that is, 8 entries or more need to be in the source peripheral FIFO in order to trigger a burst request. *Figure 303* shows how this block transfer is broken into burst and single transactions, and bus bursts and single transfers.

Figure 303: Breakdown of Block Transfer

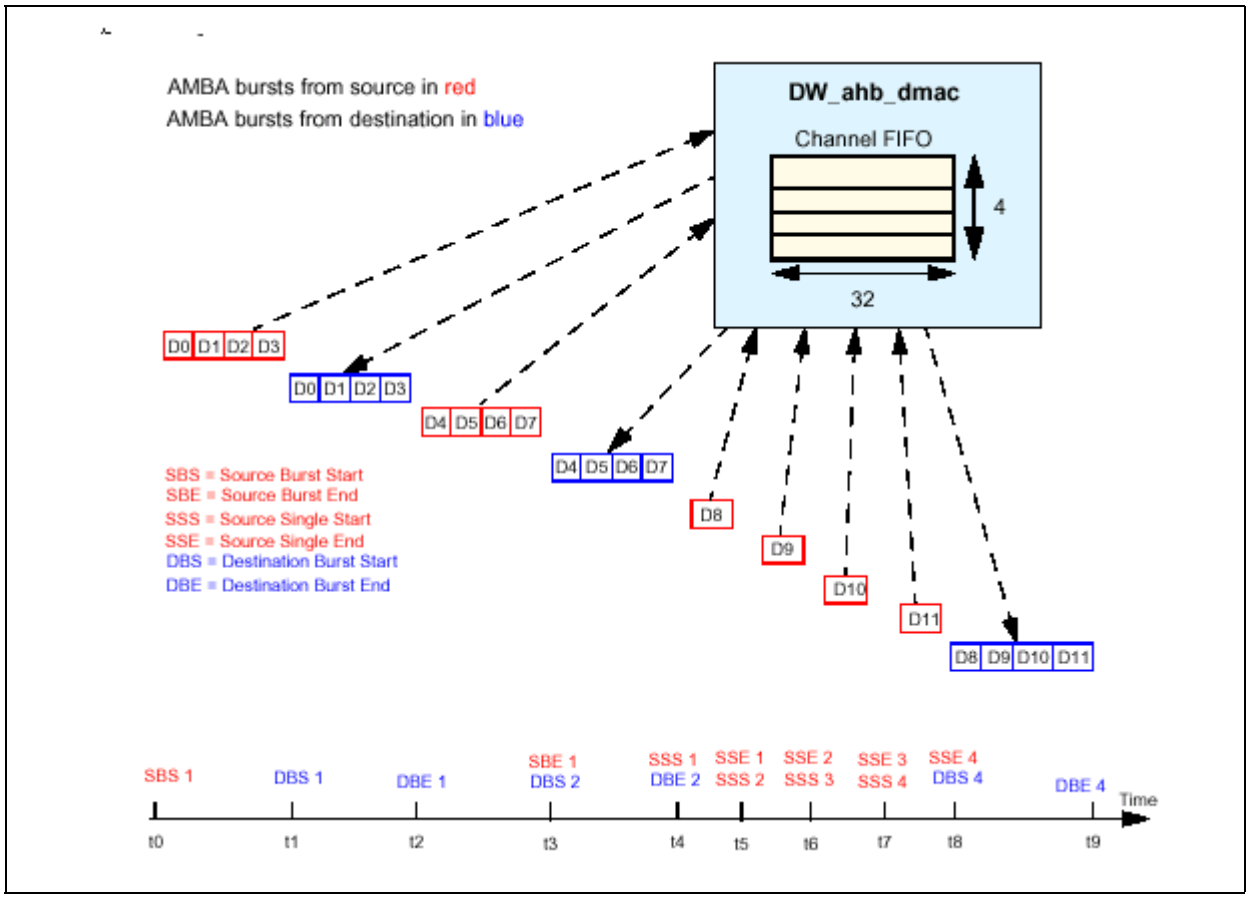
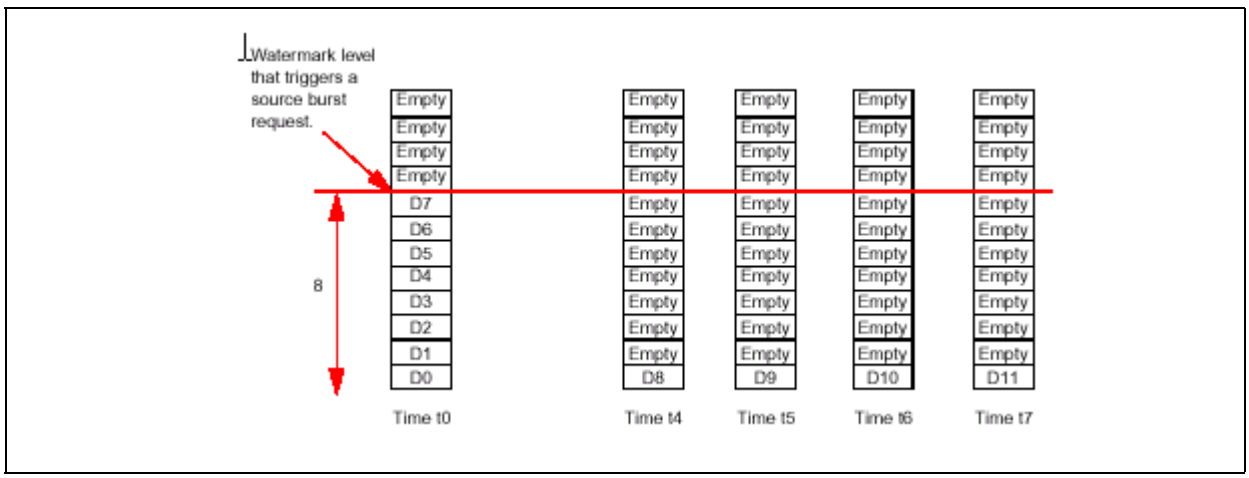


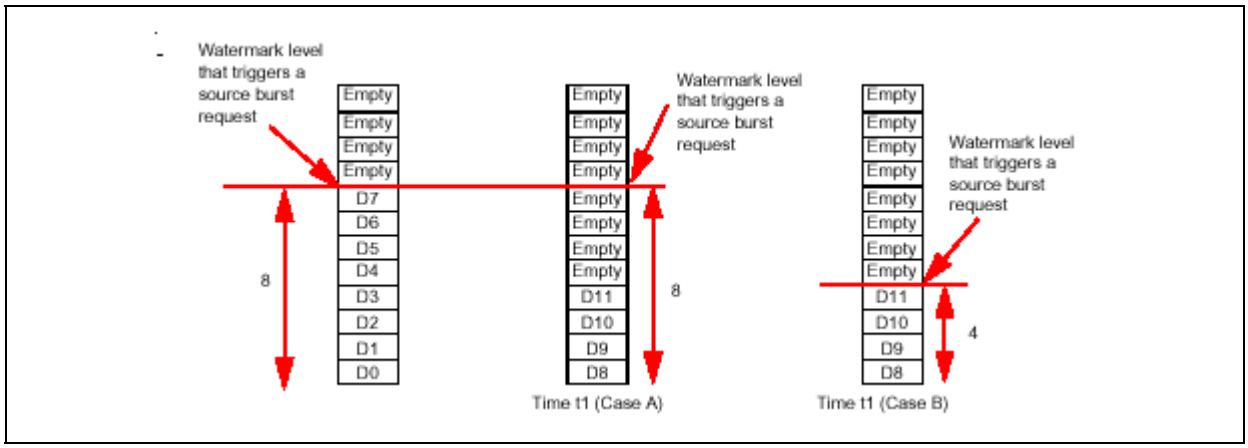
Figure 304 shows the status of the source FIFO at various times throughout the source block transfer.

Figure 304: Source FIFO Contents at Time Indicated in *Figure 303*



As shown in *Figure 305*, if the SATA DMA does not perform single transactions, the source FIFO contains 4 entries at time t1. However, the source has no more data to send. Therefore, if the watermark level remains at 8 (at time t1, Case A in *Figure 305*), the watermark level is never reached and a new burst request is never triggered.

Figure 305: Source FIFO Contents where Watermark Level is Dynamically Adjusted



The source peripheral, not knowing the length of a block and only able to request burst transactions, sits and waits for the FIFO level to reach a watermark level before requesting a new burst transaction request. This region, where the amount of data left to transfer in the source block is less than src_burst_size_bytes, is known as the Single Transaction Region.

In the Single Transaction Region, the SATA DMA performs single transactions from the source peripheral until the source block transfer has completed. In this example, the SATA DMA completes the source block transfer using 4 single transactions from the source.

Now consider Case B in *Figure 305* where the source peripheral can dynamically adjust the watermark level that triggers a burst transaction request near the end of a block. After the first source burst transaction completes, the source peripheral recognizes that it has only 4 data items left to complete in the block and adjusts the FIFO watermark level that triggers a burst transaction to 4. This triggers a burst request, and the block completes using a burst transaction. However, SADMA_CTRL0.SRC_MSIZE = 8, and there are only 4 data items left to transfer in the source block. The SATA DMA terminates the last source burst transaction early and fetches only 4 of the 8 data items in the last source burst transaction. This is called an Early-Terminated Burst Transaction.

Observation: Under certain conditions it is possible to hardcode dma_single from the source peripheral to an inactive level (hardware handshaking). Under the same conditions, it is possible for software to complete a source block transfer without initiating single transactions from the source. Refer to *Section 79.3.6.5: Single Transactions – Peripheral Is Not Flow Controller on page 961*.

Example 5

Scenario: The destination peripheral enters the Single Transaction Region while the SATA DMA is the flow controller. This example also demonstrates how the SATA DMA channel FIFO is flushed at the end of a block transfer to the destination; this example is the same for both hardware and software handshaking.

Consider the case with the parameters set to values listed in *Table 228*.

Table 228: Parameters in Transfer Operation – Example 5

Parameter	Comment
SADMA_CTRL0.TT_FC = 3'b011	Peripheral-to-peripheral transfer with SATA DMA as flow controller
SADMA_CTRL0.BLOCK_TS = 44	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b000	8 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b 011	64bit
SADMA_CTRL0.SRC_MSIZE = 3'b001	Source burst transaction length = 4
SADMA_CTRL0.DEST_MSIZE = 3'b001	Destination burst transaction length = 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 32 bytes	–

In this example, the block size is a multiple of the source burst transaction length:

blk_size_bytes_dma/src_burst_size_bytes = (44 * 1)/4 = 11 = integer

The source block transfer is completed using only burst transactions, and the source does not enter the Single Transaction Region.

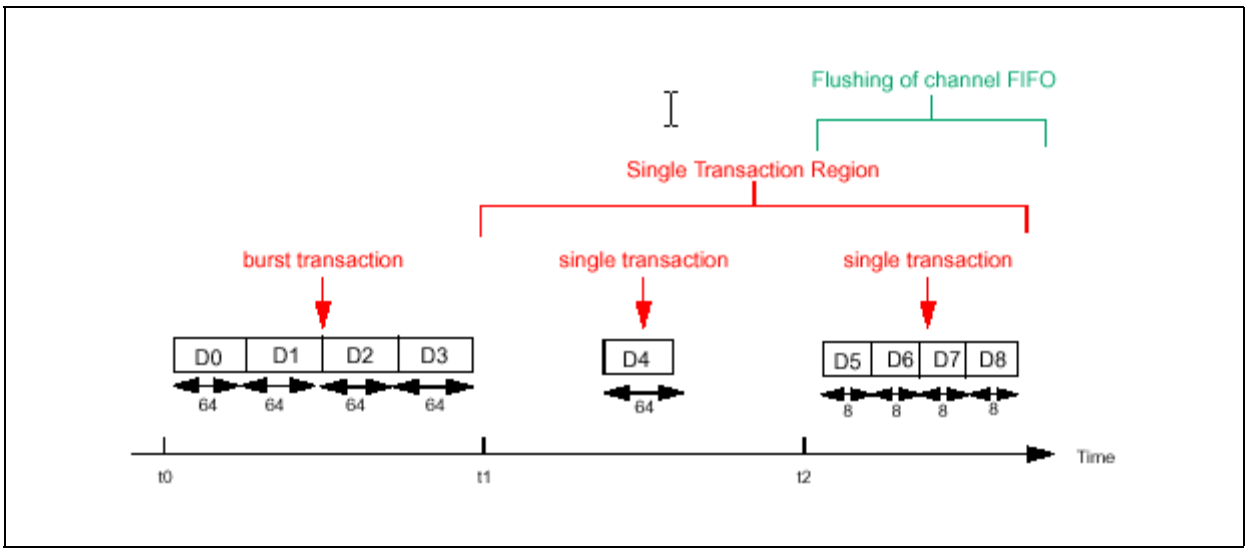
The block size is not a multiple of the destination burst transaction length:

blk_size_bytes_dma/dst_burst_size_bytes 44/32 != integer

So near the end of the block transfer to the destination, the amount of data left to be transferred is less than dst_burst_size_bytes and the destination enters the Single Transaction Region.

Figure 306 shows one way in which the block transfer to the destination can occur.

Figure 306: Block Transfer to Destination



After the first 32 bytes (dst_burst_size_bytes = 32) of the destination burst transaction have been transferred to the destination, there are 12 bytes (blk_size_bytes_dma dst_burst_size_bytes = 44 -32) bytes left to transfer. This is less then the amount of data that is transferred in a destination burst (dst_burst_size_bytes = 32). Therefore, the destination peripheral enters the

Single Transaction Region where the SATA DMA can complete a block transfer to the destination using single transactions.

Note: In the Single Transaction Region, asserting dma_single initiates a single transaction for hardware handshaking. Writing a 1 to the relevant channel bit of the SglReqDstReg register initiates a single transaction for software handshaking.

Note: The destination peripheral, not knowing the length of a block and only able to request burst transactions, sits and waits for the FIFO to fall below a watermark level before requesting a new burst transaction request.

At time 't2' in *Figure 306* a single transaction to the destination has been completed. There are now only 4 bytes (12 - dst_single_size_bytes = 12 - 8) left to transfer in the destination block. But SADMA_CTRL0.DST_TR_WIDTH implies 64-bit bus transfers to the destination (dst_single_size_bytes = 8 byte); therefore, the SATA DMA cannot form a single word of the specified SADMA_CTRL0.DST_TR_WIDTH.

The SATA DMA channel FIFO has 4 bytes in it that must be flushed to the destination. The SATA DMA switches into a "FIFO flush mode," where the block transfer to the destination is completed by changing the bus transfer width to the destination to be equal to that of the SADMA_CTRL0.SRC_TR_WIDTH; that is, byte bus transfers in this example. Thus the last single transaction in the destination block is made up of an bus burst of length 4 and SADMA_CTRL0.SRC_TR_WIDTH width.

When the SATA DMA is in FIFO flush mode, the address on haddr is incremented by the value of hsize on the bus; that is, SADMA_CTRL0.SRC_TR_WIDTH, and not SADMA_CTRL0.DST_TR_WIDTH. In cases where the DAR0 is selected to be contiguous between blocks (refer to *Table 220* and *Section 79.2.4.1: Programming examples on page 928*), the DAR0 will need re-alignment at the start of the next block as it is aligned to SADMA_CTRL0.SRC_TR_WIDTH and not SADMA_CTRL0.DST_TR_WIDTH at the end of the previous block. This is handled by hardware. Refer to *Hardware re-alignment of SAR/DAR registers on page 1023*.

In general, channel FIFO flushing to the destination occurs if all three of the following are true:

- SATA DMA or the Source peripheral are flow control peripherals
- SADMA_CTRL0.DST_TR_WIDTH > SADMA_CTRL0.SRC_TR_WIDTH
- Flow control device:
  - If SATA DMA is flow controller:
    blk_size_bytes_dma/dst_single_size_bytes != integer
  - If source is flow controller:
    blk_size_bytes_src/dst_single_size_bytes != integer

Note: When not in FIFO flush mode, a single transaction is mapped to a single bus transfer. However, in FIFO flush mode, a single transaction is mapped to multiple bus transfers of SADMA_CTRL0.SRC_TR_WIDTH width. The cumulative total of data transferred to the destination in FIFO flush mode is less than dst_single_size_bytes.

In the above example, a burst request is not generated in the Single Transaction Region. If a burst request were generated at time t1 in *Figure 306*, then the burst transaction would proceed until there was not enough data left in the destination block to form a single data item of SADMA_CTRL0.DST_TR_WIDTH width. The burst transaction would then be early- terminated. In this example, only one data item of the four requested (decoded value of DEST_MIZE = 4) would be transferred to the destination in the burst transaction. This is referred to as an Early-Terminated Burst Transaction. If a burst request were generated at time t2 in *Figure 306*, then the destination block would be completed (four byte transfers to the destination to flush the SATA DMA channel FIFO) and this burst request would again be early-terminated at the end of the destination block.

Observation: If the source transfer width, SADMA_CTRL0.SRC_TR_WIDTH in the channel control register (SADMA_CTRL0), is less than the destination transfer width

(SADMA_CTRL0.DST_TR_WIDTH), then the FIFO may need to be flushed at the end of the block transfer. This is done by setting the bus transfer width of the last few bus transfers of the block to the destination so that it is equal to SADMA_CTRL0.SRC_TR_WIDTH and not the programmed SADMA_CTRL0.DST_TR_WIDTH.

Example 6

Scenario: In all examples presented so far, none of the bus bursts have been early- terminated by the DW_ahb system arbiter. Referring to Example 1, the bus transfers on the source and destination side look somewhat symmetric. In the examples presented so far, where the bus bursts are not early-terminated by the DW_ahb system arbiter, the traffic profile on the bus bus would be the same, regardless of the value of SADMA_CFG0.FIFO_MODE. This example, however, considers the effect of SADMA_CFG0.FIFO_MODE; it is the same for both hardware and software handshaking.

SADMA_CFG0.FIFO_MODE: Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.

0 = Space/data available for single bus transfer of the specified transfer width.

1 = Space/data available is greater than or equal to half the FIFO depth for destination transfers and less than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.

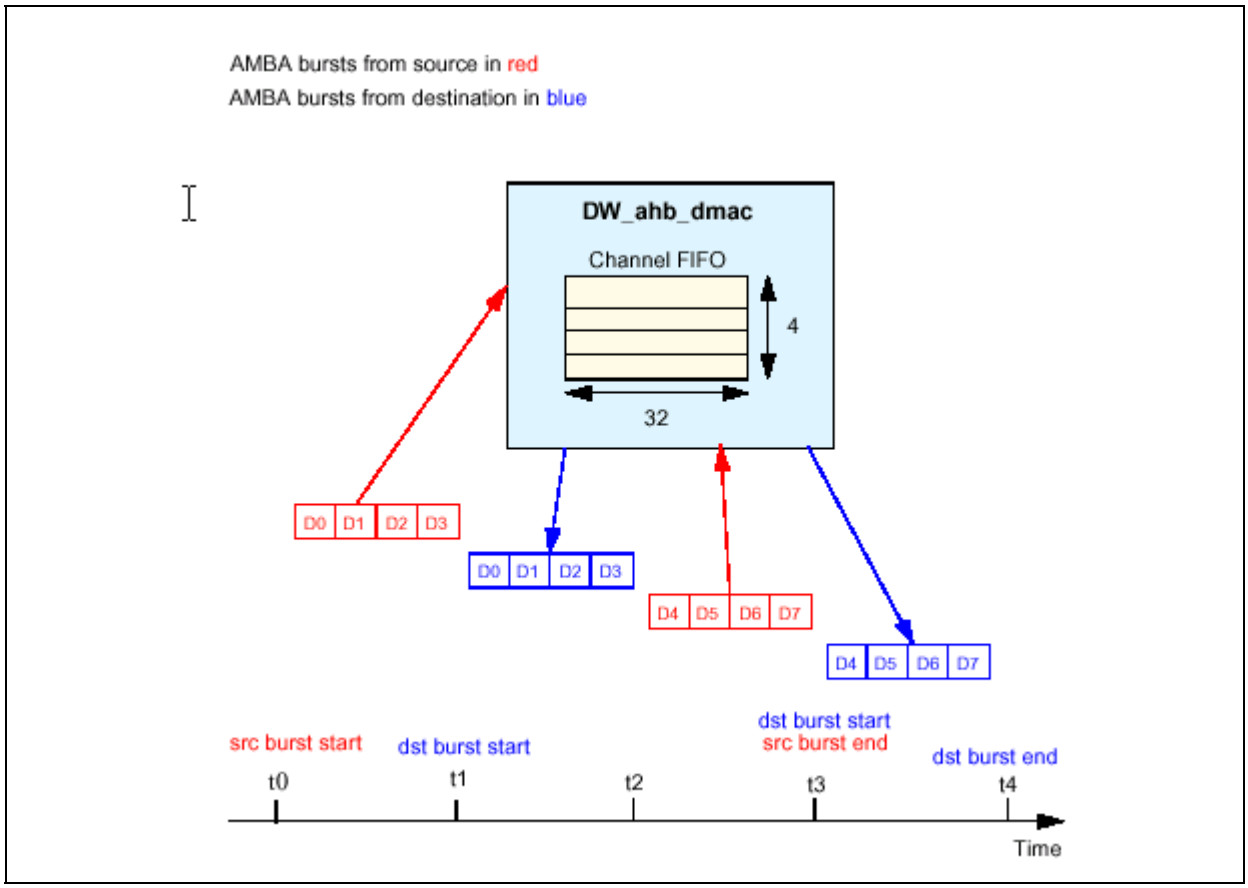
Table 229 lists the parameters used in this example.

Table 229: Parameters in Transfer Operation – Example 6

Parameter	Comment
SADMA_CTRL0.TT_FC = 3'b011	Peripheral-to-peripheral transfer with SATA DMA as flow controller
SADMA_CTRL0.BLOCK_TS = 32	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.SRC_MSIZE = 3'b010	Decoded value = 8
SADMA_CTRL0.DEST_MSIZE = 3'b001	Decoded value = 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 16 bytes	–

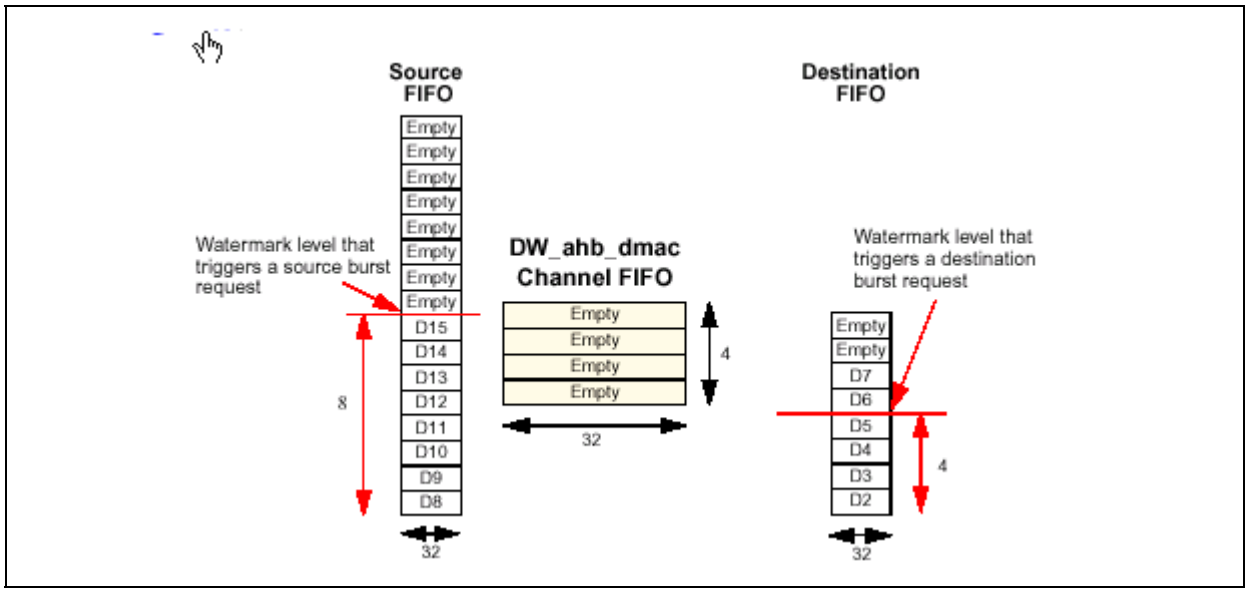
The block transfer may proceed by alternately filling and emptying the SATA DMA channel FIFO. Up to time t4, the transfer might proceed like that shown in *Figure 307*.

Figure 307: Block Transfer Up to Time "t4"



At time t4, the src, channel, and destination FIFO's might look like that shown in *Figure 308*.

Figure 308: Source, SATA DMA Channel and Destination FIFOs at Time 't4' in *Figure 305*

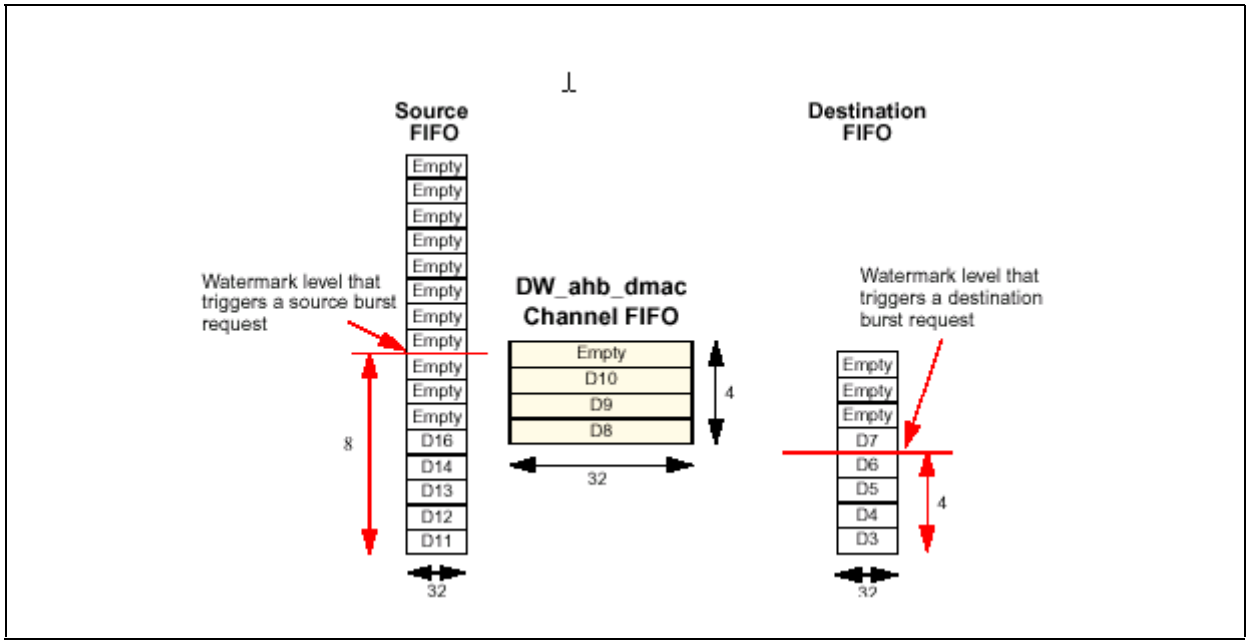


At time t4, a source burst transaction is requested, and the SATA DMA attempts a burst of length 4. Suppose that this bus burst is early-burst terminated after three bus transfers. The FIFO status's after this bus burst might look like that shown in *Figure 309*.

Referring to *Figure 309*, notice that a burst request from the destination is not triggered, since the destination FIFO contents are above the watermark level. The SATA DMA has space for one data item in the channel FIFO. So, does the SATA DMA initiate a single bus transfer from the source peripheral to fill the channel FIFO?

The answer depends on the value of SADMA_CFG0.FIFO_MODE. If SADMA_CFG0.FIFO_MODE = 0, then the SATA DMA attempts to perform a single bus transfer in order to fill the channel FIFO. If SADMA_CFG0.FIFO_MODE = 1, then the SATA DMA waits until the channel FIFO is less than half-full before initiating an bus burst from the source, as illustrated in *Figure 309*.

Figure 309: FIFO Status After Early-Terminated bus Burst



Observation: When SADMA_CFG0.FIFO_MODE = 1, the number of bus bursts per block is less than when SADMA_CFG0.FIFO_MODE = 0 and, hence, the bus utilization will improve. This setting favors longer bus bursts. However, the latency of DMA transfers may increase when SADMA_CFG0.FIFO_MODE = 1, since the SATA DMA waits for the channel FIFO contents to be less than half the FIFO depth for source transfers, or greater than or equal to half the FIFO depth for destination transfers. Therefore, system bus occupancy and usage can be improved by delaying the servicing of multiple requests until there is sufficient data/space available in the FIFO to generate an bus burst (rather than multiple single bus transfers); this comes at the expense of transfer latency. For reduced block transfer latency, set SADMA_CFG0.FIFO_MODE = 0. For improved bus utilization, set SADMA_CFG0.FIFO_MODE = 1.

Example 7

Scenario: Example block transfer when the destination is the flow controller; the effect of data pre-fetching (SADMA_CFG0.FCMODE = 0) and possible data loss. This example uses a hardware handshaking interface, but the same scenario can be explained using a software handshaking interface.

Data pre-fetching is when data is fetched from the source before the

destination requests it.

Note: Flow Control Mode. Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.

0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled.

1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode the amount of data transferred from the source is limited such that it is guaranteed

to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

Table 230 lists the parameters used in this example.

Table 230: Parameters in Transfer Operation – Example 7

Parameter	Description
SADMA_CTRL0.TT_FC=3'b111	Peripheral-to-peripheral transfer with destination as flow controller
SADMA_CTRL0.BLOCK_TS = x	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.DEST_TR_WIDTH=3'b010	32 bit
SADMA_CTRL0.SRC_MSIZE = 3'b010	Decoded value = 8
SADMA_CTRL0.DEST_MSIZE = 3'b001	Decoded value = 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 32	–
SADMA_CFG0.FCMODE = 0	Data pre-fetching enabled
SADMA_CFG0.SRC_PER = 0	Source assigned handshaking interface 0
SADMA_CFG0.DEST_PER = 1	Destination assigned handshaking interface 1
SADMA_CFG0.MAX_ABRST = 7	–

Consider a case where the destination block is made up of one burst transaction, following by one single transaction.

blk_size_bytes_dst = dst_burst_size_bytes = 16 + 4 = 20 bytes

There are a number of different cases that can arise when SADMA_CFG0.FCMODE = 0:

1. When the destination peripheral signals a last transaction, there is enough data in the SATA DMA channel FIFO to complete the last transaction to the destination. Therefore, the SATA DMA stops transferring data from the source, and the block transfer from the source completes; any surplus data that has been fetched from the source is effectively lost. Two cases arise when the destination peripheral signals the last transaction in a block:

- Case 1a: Active burst transaction request on source side
- Case 1b: No active burst request on source side

2. When the destination peripheral signals a last transaction, there is not enough data in the channel FIFO to complete the last transaction to the destination. The SATA DMA fetches just enough data to complete the block transfer. Two cases arise.

- Case 2a: Source enters Single Transaction Region when destination peripheral signals last transaction.
- Case 2b: Source does not enter Single Transaction Region when destination peripheral signals last transaction.

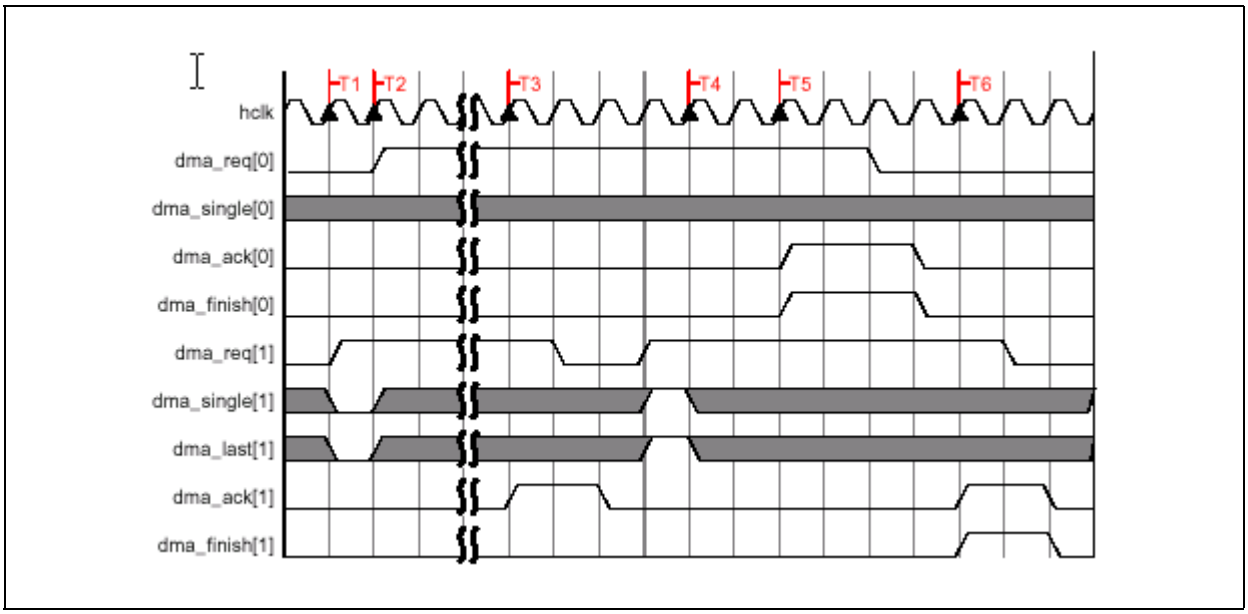
Setting SADMA_CFG0.FCMODE is pertinent only when the destination peripheral is the flow controller. When SADMA_CFG0.FCMODE = 0 scenarios arise where not all the data that has been pre-fetched from the source is required to complete the block transfer to the destination. This excess data is not transferred to the destination peripheral and is effectively lost for a read sensitive source peripheral. In this example we will assume a read sensitive source peripheral.

Case 1a and Case 1b highlight instances where data pre-fetching is enabled and data is lost. Case 2a and Case 2b highlight instances were data pre-fetching is enabled, but no data loss occurs.

In this example, handshaking interface 0 is assigned to the source peripheral, and handshaking interface 1 is assigned to the destination peripheral.

Consider the block transfer shown in *Figure 310*, where the destination is the flow controller and data pre-fetching is enabled (SADMA_CFG0.FCMODE = 0).

Figure 310: Data Loss when Pre-Fetching is Enabled



The source requests a burst transaction a time T2. The destination requests a burst transaction at time T1 and completes this burst request at time T3. At time T4 the destination requests a single transaction, which is to be the last in the block transfer.

Suppose that at time T4 the SATA DMA has fetched 7 words from the source and written 4 words to the destination. Therefore, the SATA DMA has three words in the channel FIFO, but the destination requires only one word to complete the block transfer.

The SATA DMA, recognizing that it has enough data in the channel FIFO to complete the block transfer to the destination, fetches no more data from the source and early-terminates the source burst transaction (only 7 of the 8 data items in the source burst transaction have been fetched from the source) – Early-Terminated Burst Transaction. The SATA DMA asserts dma_finish[0] to the source at time T5, and this has the same timing as dma_ack[0], as shown in *Figure 310*.

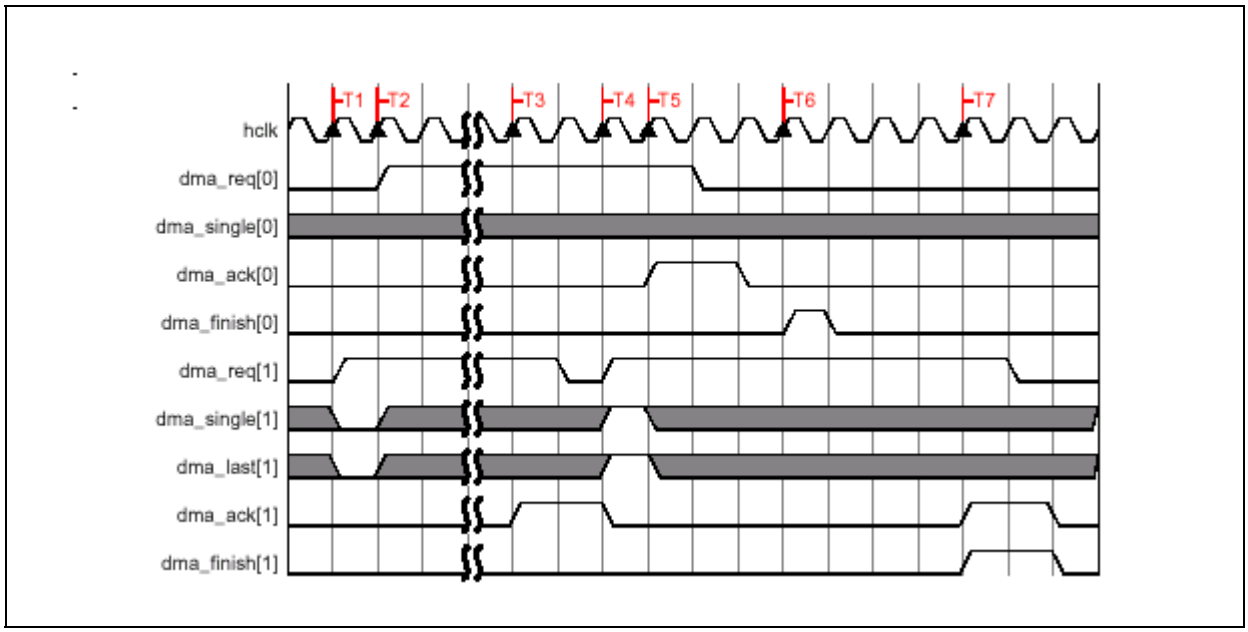
At time T6, the last single transaction to the destination has completed, which has removed one of the remaining three data words in the channel FIFO. At this time, both the source and destination block transfers have completed, and there remain two data words in the channel FIFO that have been fetched from the source. These two data words are lost because they do not form the start of the next block transfer for multi-block transfers, since the channel FIFO is cleared between blocks for multi-block transfers.

Note: There is an exception to Case 1a. If the last bus transfer to the source received a SPLIT/RETRY response over the bus bus, then dma_finish is not asserted until the bus transfer that received the SPLIT/RETRY response is retried and an OKAY response is received over the hresp bus bus. To do otherwise would be a violation of the bus protocol. This additional word that is fetched is effectively lost, since it is not transferred to the destination.

Case 1b – Timing exception on dma_finish to the source when data pre-fetching is enabled

Consider the block transfer shown in *Figure 311*, where the destination is the flow controller and data pre-fetching is enabled (SADMA_CFG0.FCMODE = 0).

Figure 311: Timing Exception on dma_finish to Source Peripheral



The source requests a burst transaction at time T2 and completes the burst transaction at time T5. The destination requests a burst transaction at time T1 and completes this burst request at time T3. At time T4, the destination requests a single transaction, which is to be the last in the block transfer. At time T5, the SATA DMA has completed the burst transaction from the source.

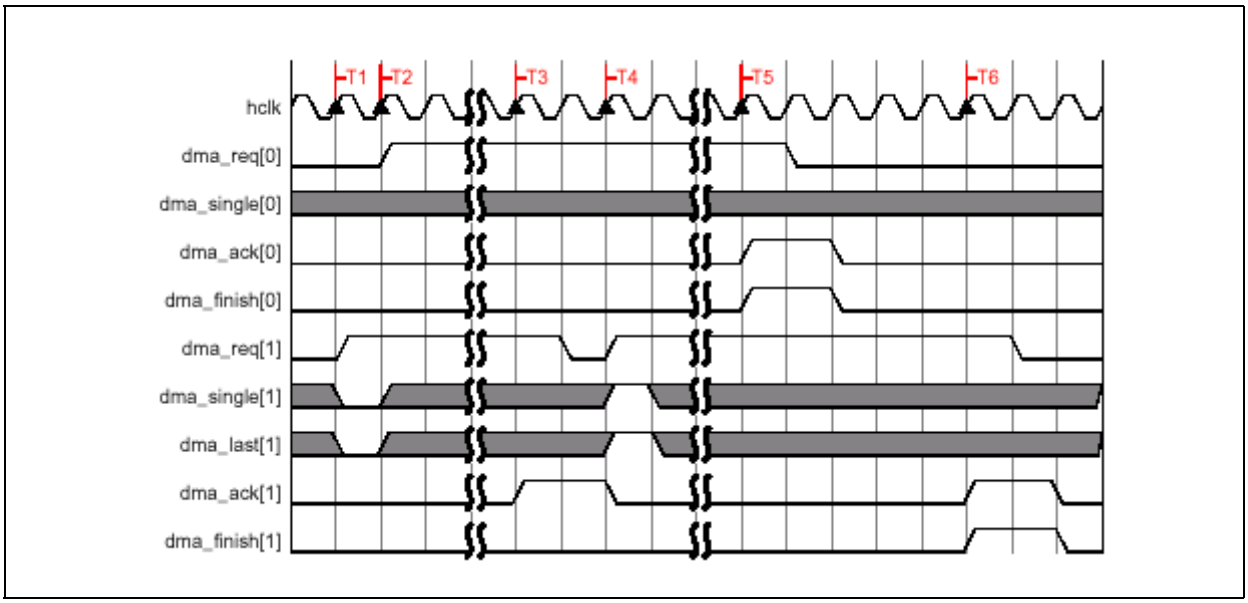
At time T5, the SATA DMA has fetched eight words from the source and written four words to the destination, which means that the SATA DMA has four words in the channel FIFO. However, the destination requires only one word to complete the block transfer. The SATA DMA, recognizing that it has enough data in the channel FIFO to complete the block transfer to the destination, fetches no more data from the source and signals a source block completion by asserting dma_finish[0] for a single cycle at time T6. Since there is no active transaction on the source side – that is, the previous source burst transaction has completed and there has been no new burst request from the source – the dma_finish[0] cannot form a handshaking loop with dma_req[0] (there is no active burst request) and therefore is asserted for only a single cycle.

Similar to Case 1a, when both the source and destination block transfers have completed at time T7, there are three data items left in the channel FIFO that are effectively lost.

Case 2a – Data pre-fetching enabled but no data loss. Source enters Single Transaction Region when destination signals last transaction.

Consider the block transfer as shown in *Figure 312*, where the destination is a flow controller and data pre-fetching is enabled (SADMA_CFG0.FCMODE = 0). The transfer parameters are the same as case 1a, *Table 230: Parameters in Transfer Operation – Example 7 on page 979*.

Figure 312: Case of No Data Loss When Pre-Fetching is Enabled



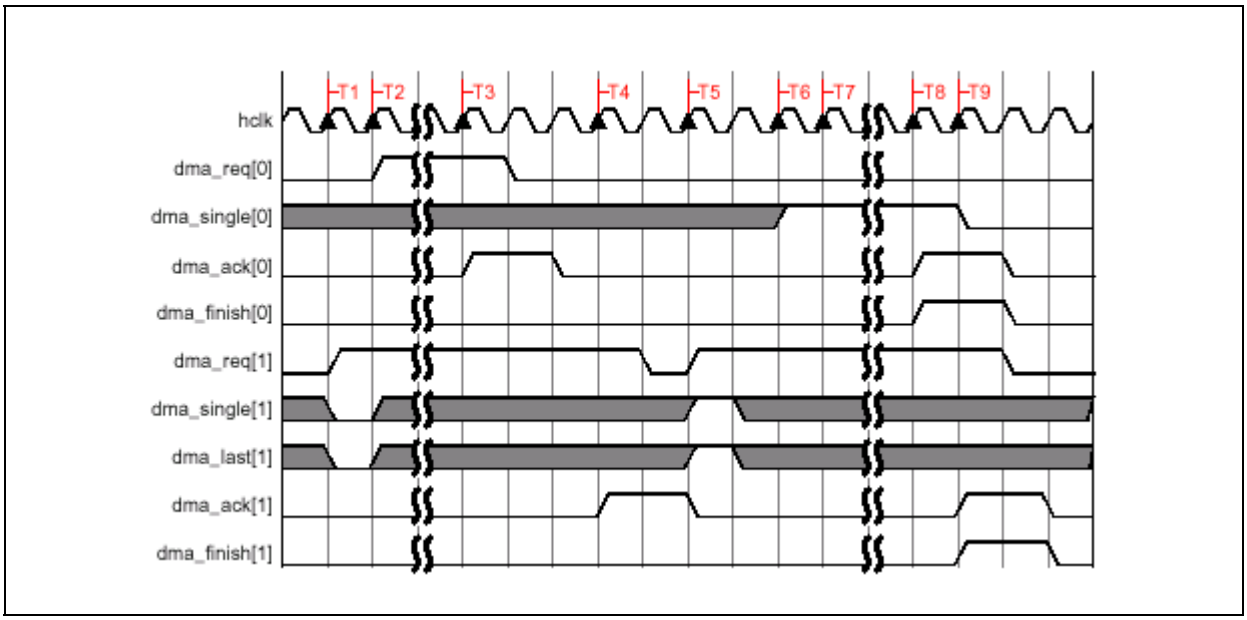
In this scenario, when dma_last[1] is asserted by the destination peripheral at time T4, there is not enough data in the channel FIFO to complete the last single transaction. Assume that the SATA DMA has fetched four data items from the source peripheral at time T4. In this case, the SATA DMA fetches one more data item from the source peripheral and then early terminates the source burst using an Early-Terminated Burst Transaction. The SATA DMA signals block completion to the source by asserting dma_finish[0] at T5, which forms a handshaking loop with dma_req[0]. In this case, there is no data loss, and all data that was fetched from the source has been transferred to the destination.

Consider the case where the transfer parameters are as given in *Table 231*.

Table 231: Transfer Parameters

Parameter	Description
SADMA_CTRL0.TT_FC=3'b111	Peripheral-to-peripheral transfer with destination as flow controller
SADMA_CTRL0.BLOCK_TS = x	
SADMA_CTRL0.SRC_TR_WIDTH = 3'b010	32 bit
SADMA_CTRL0.DEST_TR_WIDTH=3'b010	32 bit
SADMA_CTRL0.SRC_MSIZE = 3'b001	Decoded value = 4
SADMA_CTRL0.DEST_MSIZE = 3'b001	Decoded value = 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 32	–
SADMA_CFG0.FCMODE = 0	Data pre-fetching enabled
SADMA_CFG0.SRC_PER = 0	Source assigned handshaking interface 0
SADMA_CFG0.DEST_PER = 1	Destination assigned handshaking interface 1
SADMA_CFG0.MAX_ABRST = 7	–

Figure 313: Source Enters Single Transaction Region when Destination Asserts dma_last[1]



As illustrated in *Figure 312*, the source requests a burst transaction at time T2 and completes the burst transaction at time T3. The destination requests a burst transaction at time T1 and completes this burst request at time T4. The destination requests a last single transaction at time T5; the channel FIFO is empty at this time. The amount of data left to complete a source block transfer, 4 bytes, is less than src_burst_size_bytes = 4 * 4 = 16 bytes. Therefore, the source enters the Single Transaction Region at time T6. At time T7, the SATA DMA samples that dma_single[0] from the source peripheral is asserted and initiates a single transaction.

Note: If an active edge on dma_req[0] was triggered at time T6 or T7, a source burst transaction would have taken precedence over the source single transaction. Upon completion of the source block, this burst transaction would have been early-terminated using an Early-Terminated Burst Transaction.

When this transaction completes at time T8, the SATA DMA recognizes that enough data has been fetched from the source peripheral to complete the block transfer to the destination. The SATA DMA asserts dma_finish[0] to the source peripheral at time T8. This has the same timing as dma_ack[0]. The destination block transfer completes as previously described. No data loss occurs.

Case 2b – Data pre-fetching enabled but no data loss.

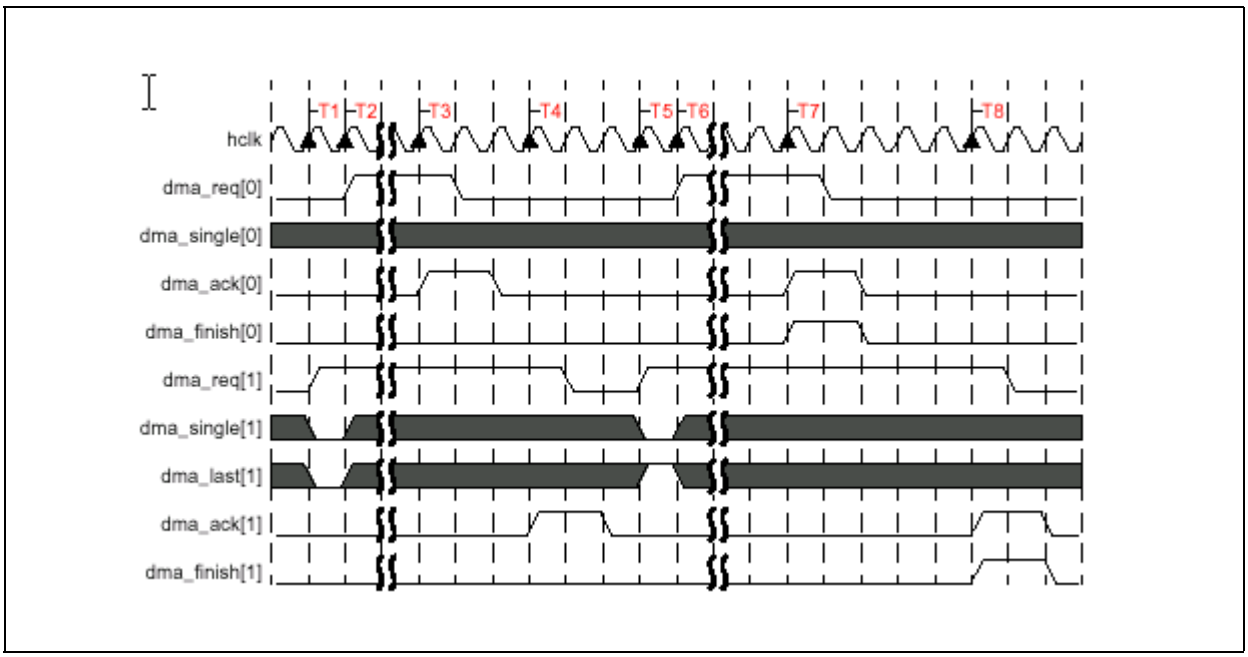
In this case, the source does not enter the Single Transaction Region when the destination signals the last transaction. This case uses the parameters listed in *Table 231*.

The destination block is made up of two burst transactions.

blk_size_bytes_dst = 2 * (4 * 4) = 32 bytes

As illustrated in *Figure 312*, the source requests a burst transaction a time T2 and completes the burst transaction at time T3. The destination requests a burst transaction at time T1 and completes this burst request at time T4.

Figure 314: Case where Source Does Not Enter Single transaction region when Destination Asserts dma_last[1]



At time T5 the destination peripheral requests the last burst transaction in the block transfer to the destination. At this point, the channel FIFO is empty. The number of bytes that must be fetched from the source peripheral to complete the block transfer to the destination is equal to 4 * 4 = 16 bytes. Since 16 bytes is not less than src_msize_bytes (16 bytes), the source does not enter the Single Transaction Region. The SATA DMA waits for a burst request from the source peripheral, which occurs at time T6. Upon completion of this burst request at time T7, the SATA DMA signals a source block transfer completion and asserts dma_finish[0]. Upon completion of the last destination burst transaction at time T8, the SATA DMA signal a destination block transfer completion and asserts dma_finish[1] to the destination.

Note that when data pre-fetching is enabled, SADMA_CFG0.FCMODE = 0, then the maximum amount of data that can be lost depends on whether the last transaction in the block transfer to the destination is a single transaction or burst transaction. In the worst case scenario, the SATA DMA has pre-fetched enough data from the source to fill the channel FIFO when the last transaction is signalled by the destination peripheral.

The maximum amount of data that can be lost is:

- Last transaction in block transfer is a single transaction:
  DMAH_CHx_FIFO_DEPTH - dst_single_size_bytes [refer to equation (1)]
- Last transaction in block transfer is a burst transaction:
  DMAH_CHx_FIFO_DEPTH - dst_burst_size_bytes [refer to equation (2)]
  If this equation is <= 0 then no data is lost.

Thus, if the last transaction in the block is a burst transaction and equation (2) is less than zero, then no data can be lost when SADMA_CFG0.FCMODE = 0. There is one exception to this, as outlined in Example 8.

Enabling data pre-fetching may reduce the latency of the DMA transfer when the destination is flow controller.

Observation: For a source peripheral which is not read sensitive (such as memory) then data pre-fetching should be enabled, SADMA_CFG0.FCMODE = 0, to reduce the transfer latency when the destination is flow controller. If the source peripheral is a read sensitive device (such as a source FIFO) then data pre-fetching should be disabled, SADMA_CFG0.FCMODE = 1, when the destination peripheral is flow controller.

Example 8

Scenario: Data loss when destination is flow controller and data pre-fetching is disabled; SADMA_CFG0.FCMODE = 1.

This example uses a hardware handshaking interface, but the same scenario can be explained using a software handshaking interface. Two scenarios arise when SADMA_CFG0.FCMODE = 1:

- SADMA_CTRL0.SRC_TR_WIDTH <= SADMA_CTRL0.DST_TR_WIDTH
- SADMA_CTRL0.SRC_TR_WIDTH > SADMA_CTRL0.DST_TR_WIDTH

Case 1 – SADMA_CTRL0.SRC_TR_WIDTH <= SADMA_CTRL0.DST_TR_WIDTH

In this case, the SATA DMA controls the transfer of data from the source, such that at any time there is at most enough data to complete the current transaction – single or burst – to the destination. If there is currently no active transaction to the destination, then the channel FIFO is empty and no data is pre-fetched from the source, even if the source has an active transaction request. If both the source and destination are requesting, then the SATA DMA fetches only enough data from the source to complete the current destination transaction, and no more. Therefore, there can never be any data loss.

Case 2 – SADMA_CTRL0.SRC_TR_WIDTH > SADMA_CTRL0.DST_TR_WIDTH

In this example, assume the parameters in *Table 232: Parameters in Transfer Operation – Example 7, Case 2b on page 985*.

Table 232: Parameters in Transfer Operation – Example 7, Case 2b

Parameter	Description
SADMA_CFG0.FCMODE = 1	Data pre-fetching disabled
SADMA_CTRL0.BLOCK_TS = x	–
SADMA_CTRL0.SRC_MSIZE = 3'b001	Decoded value = 4
SADMA_CTRL0.DEST_MSIZE = 3'b010	Decoded value = 8
SADMA_CTRL0.SRC_TR_WIDTH = 3'b011	64-bit
SADMA_CTRL0.DST_TR_WIDTH=3'b010	32-bit
SADMA_CTRL0.TT_FC=3'b111	Peripheral to Peripheral transfer with destination as flow controller
DMAH_CHx_FIFO_DEPTH = 32	–
SADMA_CFG0.SRC_PER = 0	Source assigned handshaking interface 0
SADMA_CFG0.DEST_PER = 1	Destination assigned handshaking interface 1
SADMA_CFG0.MAX_ABRST = 8	–

Consider the case where the destination block is made up of a burst transaction, followed by one single transaction:

blk_size_bytes_dst = (8 * 4) + 4 = 36 bytes

src_single_size_bytes =

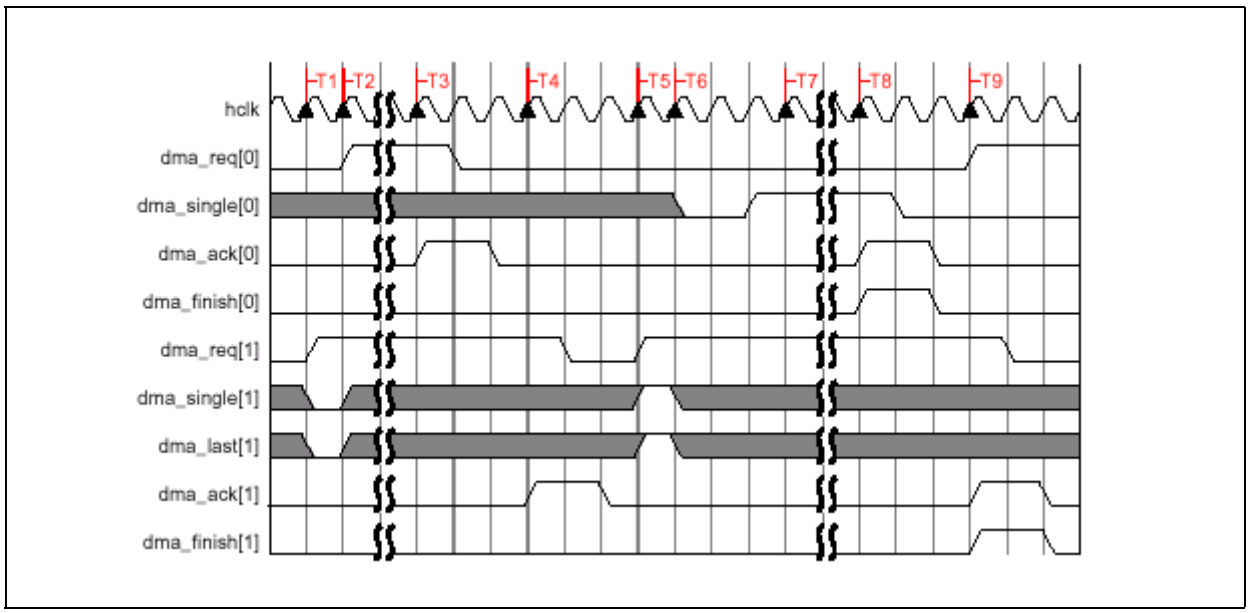
dst_single_size_bytes =

src_burst_size_bytes = 4 * 8 = 32

dst_burst_size_bytes = 8 * 4 = 32

As illustrated in *Figure 315*, the source requests a burst transaction at time T2, and completes the burst transaction at time T3. The destination requests a burst transaction at time T1 and completes this burst request at time T4. The destination requests a last single transaction at time T5. The channel FIFO is empty at this time.

Figure 315: Data Loss when Data Pre-Fetching is Disabled



The amount of data left to complete a source block transfer, 4 bytes, is less than src_burst_size_bytes (32 bytes). Therefore, the source enters the Single Transaction Region. At time T7, the SATA DMA samples that dma_single[0] is asserted and initiates a single transaction from the source.

The SATA DMA fetches a single source data item from the source peripheral and stores the 8 bytes in the channel FIFO; this single transaction completes at time T8. The source block also completes at time T8. However, the destination requires only four of these 8 bytes to complete the block transfer to the destination (the block transfer to destination completes at time T9). The remaining four bytes are lost. Thus, when the destination is the flow controller, data loss occurs when SADMA_CFG0.FCMODE = 1 if both of the following are true:

- SRC_TR_WITDT > SADMA_CTRL0.DST_TR_WIDTH
- . blk_size_bytes_dst/src_single_size_bytes != integer.

The amount of data lost is:

src_single_size_bytes - dst_single_size_bytes [refer to equations (2) and (3)]

Observation: Data loss can occur when the destination is the flow controller, even if data pre-fetching is disabled, SADMA_CFG0.FCMODE = 1.

Example 9

Scenario: This scenario demonstrates how to use software handshaking on both the source and destination sides when the SATA DMA is the flow controller. The DW_apb_ssi is used as the source and destination peripherals; for more information on the DW_apb_ssi, refer to the DesignWare DW_apb_ssi Databook. This example uses the parameters listed in *Table 233: Parameters in Transfer Operation – Example 9 on page 987*.

Table 233: Parameters in Transfer Operation – Example 9

Parameter	Description
SADMA_CTRL0.TT_FC = 3'b011	Peripheral-to-peripheral with SATA DMA as flow controller
SADMA_CTRL0.BLOCK_TS = 8	
SADMA_CTRL0.SRC_TR_WIDTH = 3'b001	16 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b001	16 bit
SADMA_CTRL0.SRC_MSIZE = 3'b 001,	Source burst transaction of length 4
SADMA_CTRL0.DEST_MSIZE = 3'b 010	Destination burst transaction of length 8
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 16 bytes	–
SADMA_CFG0.HS_SEL_SRC = 1	Source software handshaking
SADMA_CFG0.HS_SEL_DST = 1	Destination software handshaking

All SSI parameters are prefixed with "SSI".

The following are some definitions of DW_apb_ssi parameters that are used in this example:

- SSI. TXFTLR: Transmit FIFO Threshold – Controls the level of entries (or below) at which the DW_apb_ssi transmit FIFO controller triggers an ssi_txe_intr interrupt.
- SSI.TXFLR: Transmit FIFO Level – Contains the number of valid data entries in the DW_apb_ssi transmit FIFO.
- SSI.IMR.TXEIM– Writing a 0 to this field masks the ssi_txe_intr interrupt.
- SSI. RXFTLR: Receive FIFO Threshold – Controls the level of entries (or below) at which the DW_apb_ssi transmit FIFO controller triggers an ssi_rxf_intr interrupt.
- SSI.RXFLR: Receive FIFO Level – Contains the number of valid data entries in the DW_apb_ssi transmit FIFO.
- SSI.IMR.RXFIM – Writing a 0 to this field masks the ssi_rxf_intr interrupt.

In this example:

SSI. TXFTLR = SADMA_CTRL0.DEST_MSIZE

SSI.RXFTLR + 1 = SADMA_CTRL0.SRC_MSIZE

The block transfer takes place on channel 0.

It is assumed that at the beginning of the block transfer that the transaction complete interrupts, IntDstTran and IntSrcTran, are enabled and unmasked. For example:

MaskSrcTran[0] =

MaskDstTran[0] =

SADMA_CTRL0.INT_EN =

The block size is a multiple of the source burst transaction length:

- blk_size_bytes_dma/src_burst_size_bytes = 16/8 = 2 = integer

It is also a multiple of the destination burst transaction length:

- blk_size_bytes_dma/dst_burst_size_bytes 16/16 = 1 = integer

This block consists of two source burst transactions and one destination burst transaction. Neither the source nor destination enter the Single Transaction Region. The block transfer proceeds as follows:

1. SSI.TXFLR is initially equal to 0 so that an ssi_txe_intr interrupt is generated by the DW_apb_ssi. The Interrupt Service Routine (ISR) for this interrupt writes hex 0101 to both the ReqDstRegand SglReqDstReg registers; the order is not important. This generates a destination burst transaction request. Before exiting, this ISR software should write a 0 to the SSI.IMR.TXEIM register in order to mask any further ssi_txe_intr interrupts, since this is a level-sensitive interrupt.
2. When the DW_apb_ssi receive FIFO contains greater than or equal to SSI.RXFTLR + 1 half-words, an ssi_rxf_intr interrupt is generated by the DW_apb_ssi. The Interrupt Service Routine (ISR) for this interrupt writes hex 0101 to both the ReqSrcReg and SglReqSrcReg registers; the order is not important. This generates a source burst transaction request. Before exiting this ISR, software should write a 0 to the SSI.IMR.RXFIM register in order to mask any further the ssi_rxf_intr interrupts, since this is a level-sensitive interrupt.
3. Upon completion of the source burst transaction, the SATA DMA clears ReqSrcReg[0] and SglReqSrcReg[0], and generates an IntSrcTran interrupt. The ISR for this interrupt should write a 1 to the SSI.IMR.RXFIM register in order to unmask the generation of ssi_rxf_intr interrupts.
4. Same as step 2.
5. Same as step 3.
6. Upon completion of the destination burst transaction, the SATA DMA clears ReqDstReg[0] and SglReqDstReg[0], and generates an IntDstTran interrupt. The ISR for this interrupt should write a 1 to the SSI.IMR.TXEIM register to unmask the generation of ssi_rxf_intr interrupts.

Note: An alternative to using interrupts is for software to poll the DW_apb_ssi FIFO levels, SSI.TXFLR/ SSI.RXFLR, until they equal SADMA_CTRL0.DEST_MSIZE/SADMA_CTRL0.SRC_MIZE in place of ssi_txe_intr/ssi_rxf_intr interrupts. Also, in place of IntSrcTran/IntDstTran interrupts, software could poll the ReqSrcReg[0]/ReqDstReg[0] registers until cleared by hardware.

Example 10

Scenario: This scenario demonstrates how to use software handshaking on both the source and destination side when SATA DMA is flow controller and the source peripheral enter the Single Transaction Region. The DW_apb_ssi is used as the source and destination peripherals.

This example uses the parameters listed in *Table 234: Parameters in Transfer Operation – Example 10 on page 989*.

Table 234: Parameters in Transfer Operation – Example 10

Parameter	Description
SADMA_CTRL0.TT_FC = 3'b011	Peripheral-to-peripheral SATA DMA as flow controller
SADMA_CTRL0.BLOCK_TS = 12	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b00	16 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b001	16 bit
SADMA_CTRL0.SRC_MSIZE = 3'b 010	Source burst transaction of length 8
SADMA_CTRL0.DEST_MSIZE = 3'b 001	Destination burst transaction of length 4
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 8 bytes	–
SADMA_CFG0.HS_SEL_SRC = 1	Source software handshaking
SADMA_CFG0.HS_SEL_DST = 1	Destination software handshaking

The block size is not a multiple of the source burst transaction length:

blk_size_bytes_dma/src_burst_size_bytes = 24/16 != integer

Therefore, the block transfer from the source will enter the Single Transaction Region near the end of a block.

The block size is a multiple of the destination burst transaction length:

blk_size_bytes_dma/dst_burst_size_bytes 24/8 = 3 = integer

Therefore, the block transfer to the destination consists of 3 burst transaction and the destination does not enter the Single Transaction Region.

In this example:

SSI. TXFTLR = SADMA_CTRL0.DEST_MSIZE

The ssi_rxf_intr Interrupt Service Routine (ISR) is:

1. Read the SSI. RXFTLR register.

   If SSI. RXFTLR = 0

   1.1 Writes hex 0101 to the SglReqSrcReg register

   1.2 write 7 (SADMA_CTRL0.SRC_MSIZE -1) into the SSI. RXFTLR register. This will trigger a new ssi_rxf_intr interrupt when the source FIFO has greater than or equal to 8 data items.

   else (SSI. RXFTLR = 7)

   1.3 Write 0 to the SSI.IMR.RXFIM register in order to mask any further ssi_rxf_intr interrupts in the ISR for the ssi_rxf_intr interrupt.

   1.4 Writes hex 0101 to the ReqSrcReg register.

The IntSrcTran Interrupt Service Routine (ISR) is:

1. Write 0 into the SSI. RXFTLR register, which triggers an ssi_rxf_intr interrupt when a single data item is in the DW_apb_ssi receive FIFO.
2. Write a 1 to the SSI.IMR.RXFIM register in order to unmask the generation of ssi_rxf_intr interrupts.

The DMA block transfer might proceed as follows:

1. Software writes a value of 0 into the SSI. RXFTLR register, which triggers an ssi_rxf_intr interrupt when a single data item is in the DW_apb_ssi receive FIFO.
2. SSI.TXFLR is initially equal to 0 so a ssi_txe_intr interrupt is generated by DW_apb_ssi. The Interrupt Service Routine (ISR) for this interrupt writes hex 0101 to both the ReqDstReg and SglReqDstReg; the order is not important. This generates a destination burst transaction request. Before exiting this ISR, software should write a 0 to the SSI.IMR.TXEIM register in order to mask any further ssi_txe_intr interrupts, since this is a level-sensitive interrupt.
3. When the DW_apb_ssi receive FIFO contains one source data item an ssi_rxf_intr interrupt is generated by the DW_apb_ssi. The ssi_rxf_intr ISR is called with SSI. RXFTLR = 0.
4. When the DW_apb_ssi receive FIFO contains greater than or equal to SSI.RXFTLR + 1 = 8 half-words, an ssi_rxf_intr interrupt is generated by the DW_apb_ssi. The ssi_rxf_intr ISR is called with SSI. RXFTLR = 7. This generates a source burst transaction request as the SglReqSrcReg has already been written to in step 3.
5. On completion of the destination burst transaction, the SATA DMA clears ReqDstReg[0] and SglReqDstReg[0], and generates an IntDstTran interrupt. The ISR for this interrupt should write a 1 to the SSI.IMR.TXEIM register in order to unmask the generation of ssi_txe_intr interrupts.
6. Same as step 2, except the ssi_txe_intr interrupt is generated when the DW_apb_ssi transmit FIFO drops to or below the watermark level.
7. Same as step 5.
8. Upon completion of the source burst transaction, the SATA DMA clears ReqSrcReg[0] and SglReqSrcReg[0], and generates an IntSrcTran interrupt.

Note: The source peripheral has entered the Single Transaction Region, since there are only 8 bytes left to complete the source block, but src_burst_size_bytes = 16 bytes.

9. Same as *6*.
10. Same as *3*.
11. The SATA DMA performs a single transaction from the source. Upon completion of the source single transaction, the SATA DMA clears ReqSrcReg[0] and SglReqSrcReg[0], and generates an IntSrcTran interrupt.
12. Steps *10* and *11* and performed a further 3 times. The block transfer from the source is now complete.
13. Same as step *7*. The block transfer to the destination is now complete.

Note: The block transfer could proceed by polling the SSI.TXFLR/SSI.RXFLR registers in place of ssi_txe_intr/ssi_rxf_intr interrupts. Also, in place of IntSrcTran/IntDstTran interrupts, software could poll the ReqSrcReg[0]/ReqDstReg[0] registers.

When the ssi_rxf_intr ISR has no knowledge of when the source peripheral is in or outside the Single Transaction Region, as for the above example, then the ssi_rxf_intr ISR has to be invariant to this. If the ssi_rxf_intr ISR knows when the Single Transaction Region has been entered then it can dynamically adjusting the threshold level that triggers a source burst transaction as explained in *Example 4* *on page 971*.

The source block will then complete on an Early-Terminated Burst Transaction. The ssi_rxf_intr and IntSrcTran Interrupt Service Routines in this case would be:

The ssi_rxf_intr Interrupt Service Routine (ISR) is:

1. Write hex 0101 to the ReqSrcReg register followed by a write of hex 0101 to the SglReqSrcReg register. This generates a source burst transaction request.
2. Before exiting this ISR, software should write a 0 to the SSI.IMR.RXFIM register in order to mask any further the ssi_rxf_intr interrupts, since this is a level-sensitive interrupt.

The IntSrcTran Interrupt Service Routine (ISR) is:

1. If the source has entered the Single Transaction Region after this burst transaction then the ISR writes a value of 3 into the SSI.RXFTLR register. This triggers an ssi_rxf_intr interrupt when four data items are in the DW_apb_ssi receive FIFO.
2. Write a 1 to the SSI.IMR.TXEIM register in order to unmask the generation of ssi_txe_intr interrupts.

Note: Knowing SADMA_CTRL0.BLOCK_TS and the number of source burst transactions completed, or by reading SADMA_CTRL0.BLOCK_TS – which reads the total number of data items read from the source peripheral up to this time – software can calculate when the amount of data left to fetch in the source block transfer is less than SADMA_CTRL0.SRC_MSIZE. Therefore, software can calculate when the source peripheral has entered the Single Transaction Region.

Example 11

Scenario: This scenario demonstrates how to use software handshaking on both the source and destination side when the source peripheral is the flow controller and the destination peripheral enters the Single Transaction Region. The DW_apb_ssi is used as the destination peripheral. This example uses the parameters listed in *Table 235*.

Table 235: Parameters in Transfer Operation – Example 11

Parameter	Description
SADMA_CTRL0.TT_FC = 3'b101	Peripheral-to-peripheral with source as flow controller
SADMA_CTRL0.BLOCK_TS = x	–
SADMA_CTRL0.SRC_TR_WIDTH = 3'b001	16 bit
SADMA_CTRL0.DST_TR_WIDTH = 3'b001	16 bit
SADMA_CTRL0.SRC_MSIZE = 3'b 001,	Source burst transaction length = 4
SADMA_CTRL0.DEST_MSIZE = 3'b 010	Destination burst transaction length = 8
SADMA_CFG0.MAX_ABRST = 1'b 0	No limit on maximum bus burst length
DMAH_CHx_FIFO_DEPTH = 16 bytes	–
SADMA_CFG0.HS_SEL_SRC = 1	Source software handshaking
SADMA_CFG0.HS_SEL_DST = 1	Destination software handshaking

In this example, it is assumed that the source peripheral generates an interrupt when the FIFO level is greater than or equal to some watermark level. For the purposes of this example, assume that this watermark level that triggers an interrupt named src_burst_intr is equal to SADMA_CTRL0.SRC_MSIZE. Assume that this interrupt is level-sensitive and can be masked by writing to a software register in the source peripheral.

Consider the case where the source block is made up of a three-burst transaction:

blk_size_bytes_src = 3 * src_burst_size_bytes = 3 * 8 = 24 bytes

The block size is not a multiple of the destination burst transaction length:

blk_size_bytes_src/dst_burst_size_bytes 24/16 != integer

Therefore, the block transfer to the destination will enter the Single Transaction Region near the end of the block.

Block transfer when:

1. It is guaranteed that data at some point will be extracted from the destination FIFO in the “Single transaction region” in order to trigger a burst transaction OR
2. When the watermark level that triggers a burst transaction to the destination can be dynamically adjusted near the end of block.

The DMA block transfer might proceed as follows:

1. Program SSI. TXFTLR = SADMA_CTRL0.DEST_MSIZE
2. SSI.TXFLR is initially 0, so an ssi_txe_intr interrupt is generated by the DW_apb_ssi. The Interrupt Service Routine (ISR) for this interrupt writes hex 0101 to both the ReqDstReg and SglReqDstReg; the order is not important. This generates a destination burst transaction request. Before exiting this ISR, software should write a 0 to the SSI.IMR.TXEIM register in order to mask any further the ssi_txe_intr interrupts, since this is a level-sensitive interrupt.
3. The source peripheral generates an interrupt when the watermark level is reached or exceeded. The Interrupt Service Routine (ISR) for this interrupt writes hex 0100 to the SglReqSrcReg and LstSrcReg registers, followed by hex 0101 to the ReqSrcReg. This generates a source burst transaction request, which is not the last in the block. Software should now mask the src_burst_intr interrupt.
4. Upon completion of the source burst transaction, the SATA DMA clears ReqSrcReg[0] and generates an IntSrcTran interrupt. The ISR for this interrupt should unmask the generation of the src_burst_intr interrupt in the source peripheral.
5. Repeat 3.
6. Repeat 4.
7. Upon completion of the destination burst transaction, the SATA DMA clears ReqDstReg[0] and SglReqDstReg[0], and generates an IntDstTran interrupt. The ISR for this interrupt should write a 1 to the SSI.IMR.TXEIM register in order to unmask generation of ssi_rxf_intr interrupts.
8. The source peripheral generates an interrupt when the watermark level is reached or exceeded. The Interrupt Service Routine (ISR) for this interrupt writes hex 0100 to the SglReqSrcReg register, followed by hex 0101 to the LstSrcReg register, followed by hex 0101 to the ReqSrcReg register. This generates a source burst transaction request, which is the last in the block. Software should now mask the src_burst_intr interrupt.
9. Repeat 4. The block transfer from the source is now complete.
10. The block transfer to the destination may proceed as follows:

    9.1 If it is guaranteed that data at some point will be extracted from the destination FIFO near the end of a block in order to trigger a burst transaction an ssi_txe_intr interrupt will be generated by the destination DW_apb_ssi peripheral. The Interrupt Service Routine (ISR) for this interrupt writes hex 0101 to the ReqDstReg register, followed by a write hex 0101 to the SglReqDstReg. The order is important. This generates a destination burst transaction request, which is an Early-Terminated Burst Transaction. Before exiting this ISR, software should write a 0 to the SSI.IMR.TXEIM register in order to mask any further the ssi_txe_intr interrupts, since this is a level-sensitive interrupt. The block transfer to the destination is now complete.

    9.2 If it is not guaranteed that data at some point will be extracted from the destination FIFO near the end of a block in order to trigger a burst transaction and if software can determine that the destination has entered the Single Transaction Region then it can write a value of 4 into the SSI.TXFTLR register. This triggers an ssi_txe_intr interrupt when four free locations are present in the DW_apb_ssi transmit FIFO. On receipt of this interrupt, software can then write hex 0101 to the ReqDstReg register, followed by a write of hex 0101 to the SglReqDstReg register. The order is important. This generates a destination burst transaction request, which is an Early-Terminated Burst Transaction. Hardware clears ReqDstReg[0] and SglReqDstReg[0] upon block

completion, which occurs after four data items have been transferred to the destination in this burst transaction and an IntDstTran interrupt is generated. The block transfer to the destination is now complete.

Note: Software can determine when it has entered the Single Transaction Region since it knows that a last-burst transaction has been requested by the source, and can calculate the number of bytes left to complete in the destination block. It can then compare this to dst_burst_size_bytes. Refer to Section 79.3.6.1: Single Transaction Region on page 954.

Block transfer when:

1. It is NOT guaranteed that data at some point will be extracted from the destination FIFO in the “Single transaction region” in order to trigger a burst transaction AND
2. When the watermark level that triggers a burst transaction to the destination CANNOT be dynamically adjusted near the end of block.

The ssi_txe_intr and IntDstTran Interrupt Service Routines in this case would be:

The ssi_txe_intr Interrupt Service Routine (ISR) is:

1. Read the SSI. TXFTLR register.

   If SSI. TXFTLR = 1

   1.1 Write hex 0101 to the SglReqDstReg register

   1.2 Write 8 (SADMA_CTRL0.DEST_MSIZE) into the SSI. TXFTLR register. This will trigger a new ssi_txe_intr interrupt when the destination FIFO has less than or equal to 8 data items.

   else (SSI. TXFTLR = 8)

   1.3 Write 0 to the SSI.IMR.TXFIM register in order to mask any further ssi_txe_intr interrupts in the ISR for the ssi_txe_intr interrupt.

   1.4 Write hex 0101 to the ReqDstReg register.

The IntDstTran Interrupt Service Routine (ISR) is:

1. Write 1 into the SSI. TXFTLR register, which triggers an ssi_txe_intr interrupt when a single free location is available in the DW_apb_ssi transmit FIFO.
2. Write a 1 to the SSI.IMR.TXFIM register in order to unmask the generation of ssi_txe_intr interrupts.

The DMA block transfer might proceed as follows:

1. Software writes a value of 1 into the SSI. TXFTLR register, which triggers an ssi_txe_intr interrupt when a single free location is in the DW_apb_ssi transmit FIFO.
2. SSI.TXFLR is initially 0, so an ssi_txe_intr interrupt is generated by the DW_apb_ssi. The ssi_txe_intr ISR is called with SSI. TXFTLR = 1.
3. SSI.TXFLR is initially 0 so an ssi_txe_intr interrupt is generated by the DW_apb_ssi. The ssi_txe_intr ISR is called with SSI. TXFTLR = 8. A destination burst transaction request is generated as the SglReqDstReg register has been written to in 2.
4. The source peripheral generates an interrupt when the watermark level is reached or exceeded. The Interrupt Service Routine (ISR) for this interrupt writes hex 0100 to the SglReqSrcReg and LstSrcReg registers, followed by a write of hex 0101 to the ReqSrcReg. This generates a source burst transaction request, which is not the last in the block. Software should now mask the src_burst_intr interrupt.
5. Upon completion of the source burst transaction, the SATA DMA clears ReqSrcReg[0] and generates an IntSrcTran interrupt. The ISR for this interrupt should unmask generation of the src_burst_intr interrupt in the source peripheral.
6. Repeat *4*.
7. Repeat *5*.

8. Upon completion of the destination burst transaction, the SATA DMA clears ReqDstReg[0] and SglReqDstReg[0], and generates an IntDstTran interrupt.
9. The source peripheral generates an interrupt when the watermark level is reached or exceeded. The Interrupt Service Routine (ISR) for this interrupt writes hex 0100 to the SglReqSrcReg register, followed by a write of hex 0101 to the LstSrcReg register, followed by a write of hex 0101 to the ReqSrcReg register. This generates a source burst transaction request, which is the last in the block. Software should now mask the src_burst_intr interrupt.
10. Repeat *5*.The block transfer from the source is now complete.
11. When a free location exists in the destination FIFO a ssi_txe_intr interrupt is generated by the DW_apb_ssi. The ssi_txe_intr ISR is called with SSI.
    TXFTLR = 1.
12. The SATA DMA performs a single transaction to the destination. Upon completion of the destination single transaction, the SATA DMA clears ReqDstReg[0] and SglReqDstReg[0], and generates an IntDstTran interrupt.
13. Repeat *11* and *12* a further 4 times. The block transfer to the destination is now complete.

Note: Example 11 outlines a SATA DMA block transfer using software handshaking when interrupts are used to control the block transfer. The same could be achieved using register polling.

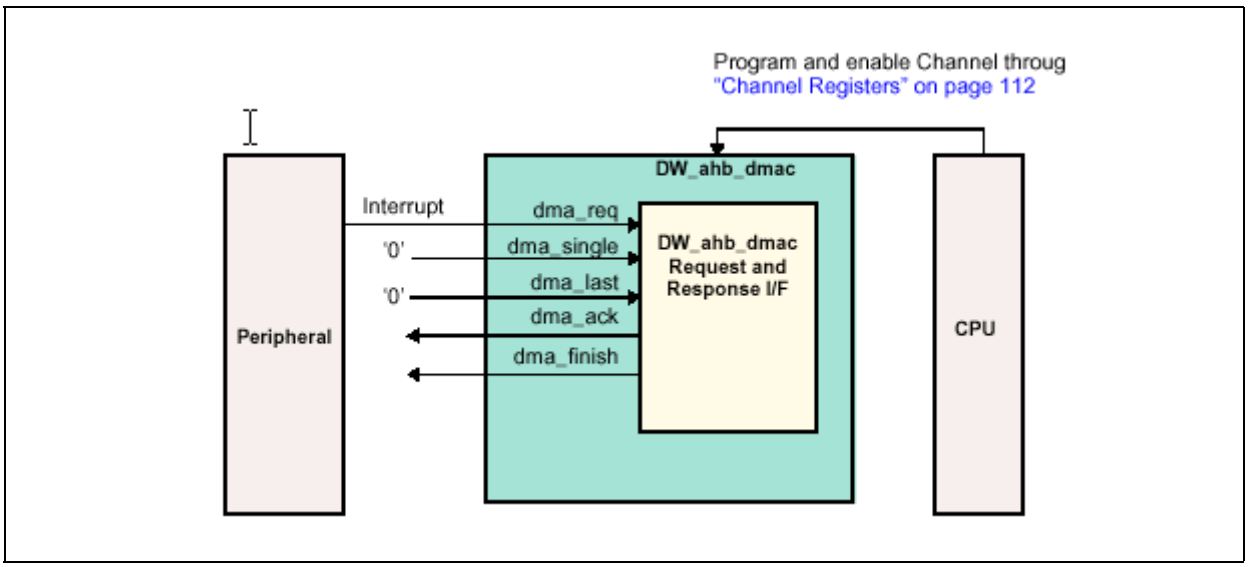
79.3.8.2 Peripheral Interrupt Request Interface

The interface illustrated in *Figure 316* is a simplified version of the hardware handshaking interface. In this mode:

- The interrupt line from the peripheral is tied to the dma_req input.
- The dma_single input is tied low.
- All other interface signals are ignored.

This interface can be used where the slave peripheral does not have hardware handshaking signals. To the SATA DMA, this is the same *Section 79.3.6: Handshaking Interface – Peripheral Is Not Flow Controller on page 954*.

Figure 316: Transaction Request Through Peripheral Interrupt



The peripheral can never be the flow controller, since it cannot connect to the dma_last signal. The interrupt line from the peripheral is tied to the dma_req line, as shown in *Figure 316*. The timing of the interrupt line from the peripheral must be the same as the dma_req line, as discussed in *Section 79.3.6: Handshaking Interface – Peripheral Is Not Flow Controller on page 954*.

Since the dma_ack line is not sampled by the peripheral, the handshaking loop is as follows:

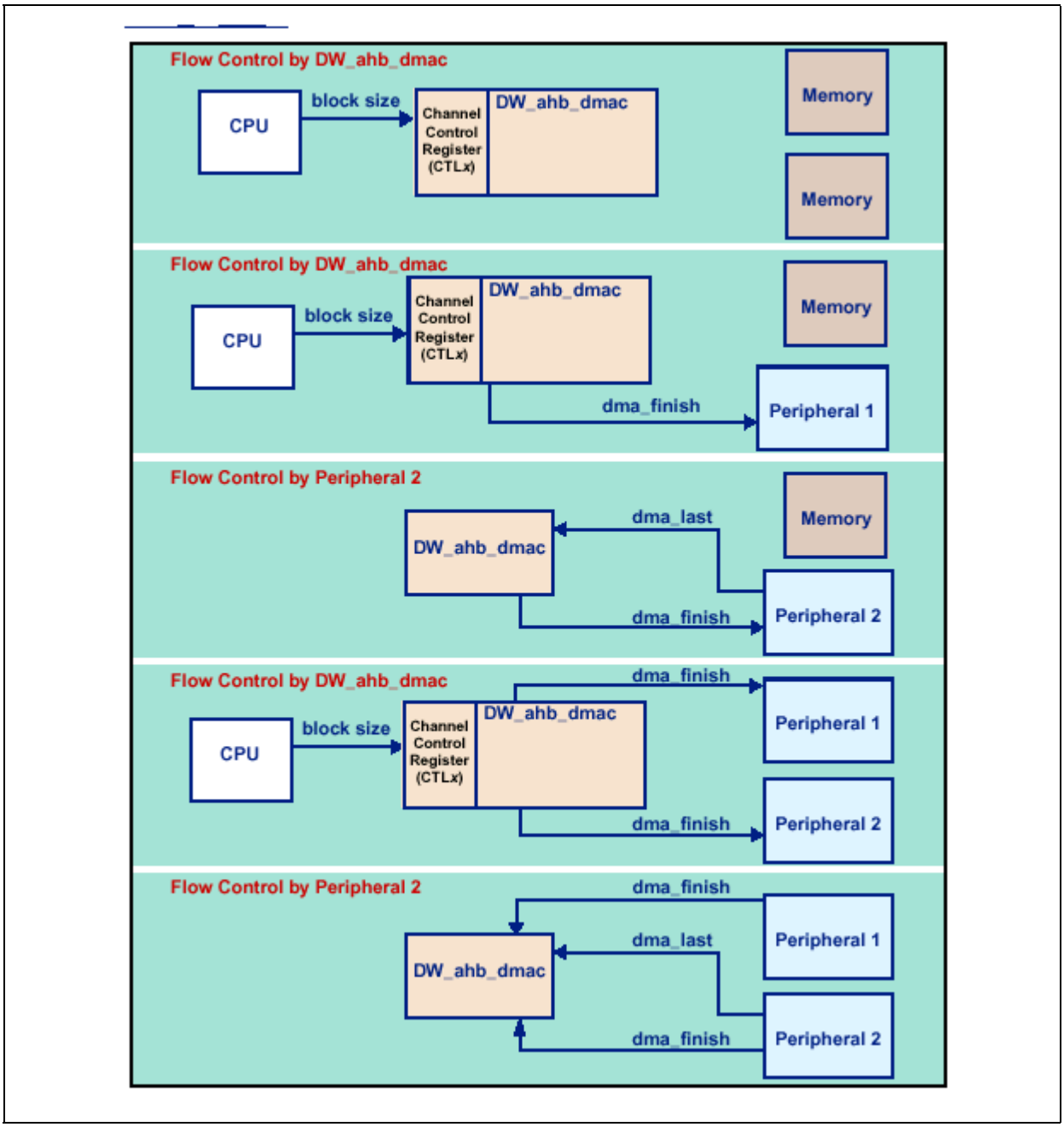
1. Peripheral generates an interrupt that asserts dma_req.
2. SATA DMA completes the burst transaction and generates an end-of-burst transaction interrupt, IntSrcTran/IntDstTran. Interrupts must be enabled and the transaction complete interrupt unmasked.
3. The interrupt service routine clears the interrupt in the peripheral so that the dma_req is de-asserted.

Notice that dma_single is hardcoded to an inactive level. For conditions where the source/ destination peripheral can tie dma_single to an inactive level, refer to *Section 79.3.6.5: Single Transactions – Peripheral Is Not Flow Controller on page 961*.

79.3.9 Flow Control Configurations

Figure 317 indicates five different flow control configurations using hardware handshaking interfaces (a simplified version of the interface is shown). These scenarios can also be used for software handshaking, which uses software registers instead of signals.

Figure 317: Flow Control Configurations



79.3.10 Peripheral Burst Transaction Requests

For a source FIFO, an active edge is triggered on dma_req when the source FIFO exceeds some watermark level. For a destination FIFO, an active edge is triggered on dma_req when the destination FIFO drops below some watermark level.

This section investigates the optimal settings of these watermark levels on the source/ destination peripherals and their relationship to source transaction length, SADMA_CTRL0.SRC_MSIZE, and destination transaction length, SADMA_CTRL0.DEST_MSIZE, respectively. For demonstration purposes, a receive DW_apb_ssi is used as a source peripheral, and a transmit DW_apb_ssi is used as a destination peripheral.

Note: Throughout this section DW_apb_ssi-related parameters are prefixed with "SSI". SATA DMA-related parameters are prefixed with "DMA".

79.3.10.1Transmit Watermark Level and Transmit FIFO Underflow

During DW_apb_ssi serial transfers, DW_apb_ssi transmit FIFO requests are made to the SATA DMA whenever the number of entries in the DW_apb_ssi transmit FIFO is less than or equal to the DW_apb_ssi Transmit Data Level Register (SSI.DMATDLR) value. This is known as the watermark level. The SATA DMA responds by writing a burst of data to the DW_apb_ssi transmit FIFO buffer, of length DMA.SADMA_CTRL0.DEST_MSIZE.

Data should be fetched from the SATA DMA often enough for the DW_apb_ssi transmit FIFO to continuously perform serial transfers; that is, when the DW_apb_ssi transmit FIFO begins to empty, another burst transaction request should be triggered.

Otherwise the DW_apb_ssi transmit FIFO runs out of data (underflow). To prevent this condition, the user must set the watermark level correctly.

79.3.10.2Choosing the Transmit Watermark Level

Consider an example with the following assumption:

DMA.SADMA_CTRL0.DEST_MSIZE = SSI_TX_FIFO_DEPTH - SSI.DMATDLR

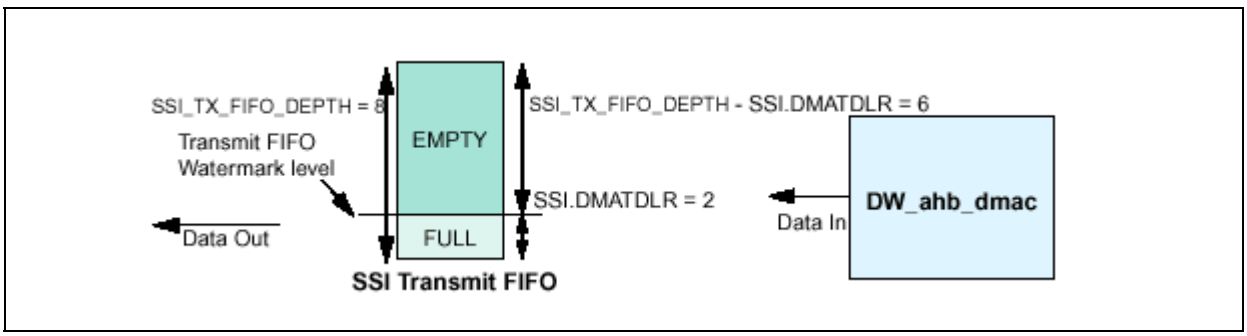
Note: SSI_TX_FIFO_DEPTH is the DW_apb_ssi transmit FIFO depth. SSI.DMATDLR controls the level at which a SATA DMA destination burst request is made by the DW_apb_ssi transmit logic. It is equal to the watermark level; that is, a destination burst request is generated (active-edge of dma_req triggered) when the number of valid data entries in the DW_apb_ssi transmit FIFO is equal to or below this field value.

In this situation, the number of data items to be transferred in a SATA DMA burst is equal to the empty space in the DW_apb_ssi transmit FIFO. Consider two different watermark level settings.

79.3.10.3Case 1: SSI.DMATDLR = 2

Figure 318 illustrates the watermark levels in Case 1 where SSI.DMATDLR = 2.

Figure 318: Case 1 Watermark Levels where SSI.DMATDLR = 2



Case 1 uses the parameters listed in *Table 236*.

Table 236: Transmit Watermark Level – Case 1

Parameter	Comment
SSI.DMATDLR = 2	DW_apb_ssi transmit FIFO watermark level
DMA.SADMA_CTRL0.DEST_MSIZE = SSI_TX_FIFO_DEPTH - SSI.DMATDLR = 6	DMA.SADMA_CTRL0.DEST_MSIZE is equal to the empty space in the transmit FIFO at the time the burst request is made.
SSI _TX_FIFO_DEPTH = 8	DW_apb_ssi transmit FIFO depth
DMA.SADMA_CTRL0.BLOCK_TS = 30	Block size

The number of burst transactions that are needed equals the block size divided by the number of data items per burst:

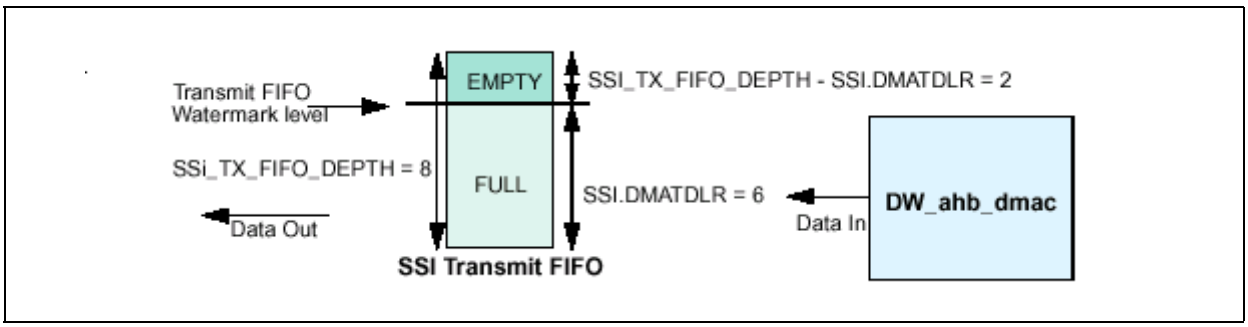
DMA.SADMA_CTRL0.BLOCK_TS/DMA.SADMA_CTRL0.DEST_MSIZE = 30/6 = 5

The number of burst transactions in the SATA DMA block transfer is 5, but the watermark level, SSI.DMATDLR, is quite low. Therefore, the probability of an SSI underflow is high where the SSI serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the SATA DMA has not had time to service the SATA DMA request before the DW_apb_ssi transmit FIFO becomes empty.

Case 2: SSI.DMATDLR = 6

Figure 319 illustrates the watermark levels in Case 2 where SSI.DMATDLR = 6.

Figure 319: Case 2 Watermark Levels where SSI.DMATDLR = 6



Case 2 uses the parameters listed in *Table 237*.

Table 237: Transmit Watermark Level – Case 2

Parameter	Description
SSI.DMATDLR = 6	DW_apb_ssi transmit FIFO watermark level
DMA.SADMA_CTRL0.DEST_MSIZE = SSI_TX_FIFO_DEPTH - SSI.DMATDLR = 2	DMA.SADMA_CTRL0.DEST_MSIZE is equal to the empty space in the transmit FIFO at the time the burst request is made.
SSI_TX_FIFO_DEPTH = 8	DW_apb_ssi transmit FIFO depth
DMA.SADMA_CTRL0.BLOCK_TS = 30	Block size

The number of burst transactions in the block are:

DMA.SADMA_CTRL0.BLOCK_TS/DMA.SADMA_CTRL0.DEST_MSIZE = 30/2 = 15

In this block transfer, there are fifteen destination burst transactions in a DMA block transfer, but the watermark level, SSI.DMATDLR, is high. Therefore, the probability of an SSI underflow is low because the SATA DMA has plenty of time to service the destination burst transaction request before the SSI transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This potentially provides a greater amount of bus bursts per block and a worse bus utilization than the former case.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the SSI transmits data to the rate at which the SATA DMA can respond to destination burst requests.

For example, promoting the channel to the highest-priority channel in the SATA DMA, and promoting the SATA DMA master interface to the highest- priority master in the bus layer, increases the rate at which the SATA DMA can respond to burst transaction requests. This in turn allows the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

79.3.10.4 Selecting SADMA_CTRL0.DEST_MSIZE and Transmit FIFO Overflow

As can be seen from *Figure 319: Case 2 Watermark Levels where SSI.DMATDLR = 6 on page 998*, programming DMA.SADMA_CTRL0.DEST_MSIZE to a value greater than the watermark level that triggers the SATA DMA request may cause overflow when there is not enough space in the DW_apb_ssi transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow:

DMA.SADMA_CTRL0.DEST_MSIZE <= SSI_TX_FIFO_DEPTH - SSI.DMATDLR (13)

In Case 2: SSI.DMATDLR = 6, the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.SADMA_CTRL0.DEST_MSIZE. Thus, the transmit FIFO may be full, but not overflowed, upon completion of the burst transaction.

Therefore, for optimal operation, DMA.SADMA_CTRL0.DEST_MSIZE should be set at the FIFO level that triggers a transmit SATA DMA request; that is:

DMA.SADMA_CTRL0.DEST_MSIZE = SSI_TX_FIFO_DEPTH - SSI.DMATDLR (14)

This is the setting used in *Figure 307: Block Transfer Up to Time "t4" on page 977*.

Adhering to equation (14) reduces the number of SATA DMA bursts needed for a

block transfer, and this in turn improves bus bus utilization.

Note: The DW_apb_ssi transmit FIFO is not full at the end of a SATA DMA burst transaction if the SSI has successfully transmitted one data item or more on the SSI serial transmit line before the end of the burst transaction.

79.3.10.5 Receive Watermark Level and Receive FIFO Overflow

During DW_apb_ssi serial transfers, DW_apb_ssi receive FIFO requests are made to the SATA DMA whenever the number of entries in the DW_apb_ssi receive FIFO is at or above the SATA DMA Receive Data Level Register; that is, SSI.DMARDLR+1. This is known as the watermark level. The SATA DMA responds by reading a burst of data from the receive FIFO buffer of length DMA.SADMA_CTRL0.SRC_MSIZE.

Note: SSI.DMARDLR controls the level at which a source burst request is made by the receive logic. When the number of valid data entries in the DW_apb_ssi receive FIFO is equal to or greater than the watermark level (DMARDL+1) a source burst request is generated (active-edge of dma_req triggered).

Data should be fetched by the SATA DMA often enough for the DW_apb_ssi receive FIFO to accept serial transfers continuously; that is, when the DW_apb_ssi receive FIFO begins to fill, another burst transaction request should be triggered. Otherwise, the DW_apb_ssi receive FIFO fills with data (overflow). To prevent this condition, the user must correctly set the watermark level.

79.3.10.6Choosing the Receive Watermark level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, SSI.DMARDLR+1, should be set to minimize the probability of overflow, as shown in *Figure 320: SSI Receive FIFO on page 1000*. It is a trade-off between the number of burst transactions required per block versus the probability of an overflow occurring.

79.3.10.7Selecting SADMA_CTRL0.SRC_MSIZE and Receive FIFO Underflow

As can be seen in *Figure 320*, programming a source burst transaction length greater than the watermark level may cause underflow when there is not enough data to service the source burst request. Therefore, the following equation must be adhered to avoid underflow:

DMA.SADMA_CTRL0.SRC_MSIZE <= SSI.DMARDLR + 1 (15)

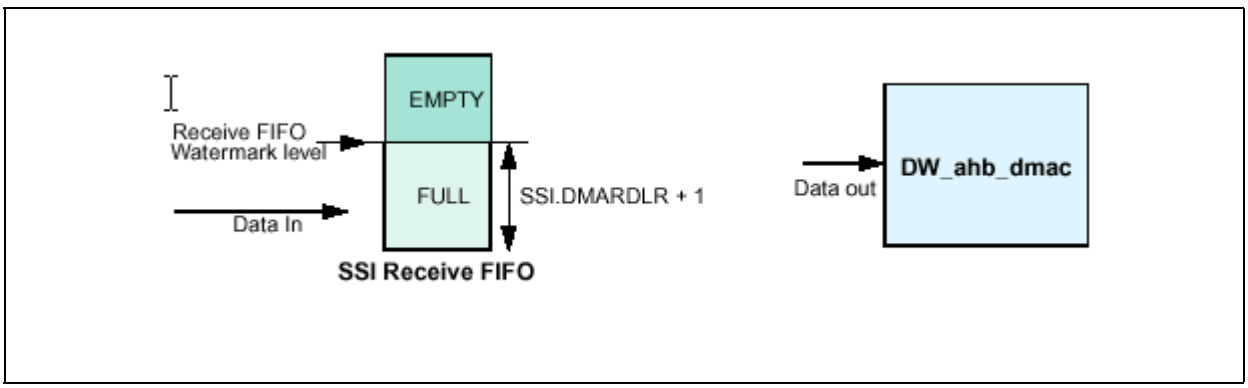
If the number of data items in the DW_apb_ssi receive FIFO is equal to the source burst length at the time the source burst request is made – DMA.SADMA_CTRL0.SRC_MSIZE – the DW_apb_ssi receive FIFO may be emptied, but not underflowed, at the completion of the source burst transaction. For optimal operation, DMA.SADMA_CTRL0.SRC_MSIZE should be set at the watermark level; that is:

DMA.SADMA_CTRL0.SRC_MSIZE = SSI.DMARDLR + 1 (16)

Adhering to equation (16) reduces the number of burst transactions in a block transfer, and this in turn can improve bus bus utilization.

Note: The DW_apb_ssi receive FIFO is not empty at the end of the source burst transaction if the SSI has successfully received one data item or more on the SSI serial receive line before the end of the burst, as illustrated in Figure 320.

Figure 320: SSI Receive FIFO



79.3.11 Generating Requests for the AHB Master Bus Interface

Each channel has a source state machine and destination state machine running in parallel. These state machines generate the request inputs to the arbiter, which arbitrates for the master bus interface (one arbiter per master bus interface).

When the source/destination state machine is granted control of the master bus interface, and when the master bus interface is granted control of the external AHB bus, then bus transfers between the peripheral and the SATA DMA (on behalf of the granted state machine) can take place.

bus transfers from the source peripheral or to the destination peripheral cannot proceed until the channel FIFO is ready. For burst transaction requests and for transfers involving memory peripherals, the criterion for “FIFO readiness” is controlled by the FIFO_MODE field of the SADMA_CFG0 register.

The definition of FIFO readiness is the same for:

- Single transactions
- Burst transactions where SADMA_CFG0.FIFO_MODE = 0
- Transfers involving memory peripherals where SADMA_CFG0.FIFO_MODE = 0

The channel FIFO is deemed ready when the space/data available is sufficient to complete a single bus transfer of the specified transfer width. FIFO readiness for source transfers occurs when the channel FIFO contains enough room to accept at least a single transfer of SADMA_CTRL0.SRC_TR_WIDTH width. FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of SADMA_CTRL0.DST_TR_WIDTH width.

Note: An exception to FIFO readiness for destination transfers occurs in "FIFO flush mode." In FIFO flush mode, FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of SADMA_CTRL0.SRC_TR_WIDTH width (and not SADMA_CTRL0.DST_TR_WIDTH width as is the normal case). For an explanation of FIFO flush mode, refer to Section : Example 5 on page 974.

When SADMA_CFG0.FIFO_MODE = 1, then the criterion for FIFO readiness for burst transaction requests and transfers involving memory peripherals is as follows:

- A FIFO is ready for a source bus burst transfer when the FIFO is less than half empty.
- A FIFO is ready for a destination bus burst transfer when the FIFO is greater than or equal to half full.

Exceptions to this "readiness" occur. During these exceptions, a value of SADMA_CTRL0.FIFO_MODE = 0 is assumed. The following are the exceptions:

- Near the end of a burst transaction or block transfer – The channel source state machine does not wait for the channel FIFO to be less than half empty if the number of source data items left to complete the source burst transaction or source block transfer is less than DMAH_CHx_FIFO_DEPTH/2. Similarly, the channel destination state machine does not wait for the channel FIFO to be greater than or equal to half full, if the number of destination data items left to complete the destination burst transaction or destination block transfer is less than DMAH_CHx_FIFO_DEPTH/2.
- In FIFO flush mode – For an explanation of FIFO flush mode, refer to *Section : Example 5 on page 974*.
- When a channel is suspended – The destination state machine does not wait for the FIFO to become half empty to flush the FIFO, regardless of the value of the FIFO_MODE field.
- After receipt of a split/retry response from a source or destination – The bus protocol requires that after an AHB master receives a split/retry response, it must re-issue the transfer that received the split/retry before attempting any other transfer. Therefore, a transfer is re-issued to the same address that returned the split/retry, regardless of FIFO_MODE, when the SATA DMA is next granted the AHB bus. This is repeated until an OKAY response is received on the bus hresp bus.

When the source/destination peripheral is not memory, the source/destination state machine waits for a single/burst transaction request. Upon receipt of a transaction request and only if the channel FIFO is 'ready' for source/destination bus transfers, a request for the master bus interface is made by the source/destination state machine.

Note: There is one exception to this, which occurs when the destination peripheral is the flow controller and SADMA_CFG0.FCMODE = 1 (data pre-fetching is disabled). Then the source state machine does not generate a request for the master bus interface (even if the FIFO is 'ready' for source transfers and has received a source transaction request) until the destination requests new data. Refer to Section : Example 8 on page 985.

When the source/destination peripheral is memory, the source/destination state machine must wait until the channel FIFO is 'ready.' A request is then made for the master bus interface. There is no handshaking mechanism employed between a memory peripheral and the SATA DMA.

79.3.11.1 Locked DMA Transfers

It is possible to program the SATA DMA for:

- Bus locking – Asserts the bus hlock signal.
- Channel locking – Locks the arbitration for the AHB master interface, which grants ownership of the master bus interface to one of the requesting channel state machines (source or destination).

Bus and channel locking can proceed for the duration of a DMA transfer, a block transfer, or a single or burst transaction.

Bus Locking

If the LOCK_B bit in the channel configuration register (SADMA_CFG0) is set, then the bus hlock signal is asserted for the duration specified in the LOCK_B_L field.

Channel Locking

If the LOCK_CH field is set, then the arbitration for the master bus interface is exclusively reserved for that channel's source and destination peripherals for the duration specified in the LOCK_CH_L field.

If bus locking is activated for a certain duration, then it follows that the channel is also automatically locked for that duration. Three cases arise:

- SADMA_CFG0.LOCK_B = 0 – Programmed values of SADMA_CFG0.LOCK_CH and SADMA_CFG0.LOCK_CH_L are used.
- SADMA_CFG0.LOCK_B = 1 and SADMA_CFG0.LOCK_CH = 0 – DMA transfer proceeds as if SADMA_CFG0.LOCK_CH = 1 and SADMA_CFG0.LOCK_CH_L = SADMA_CFG0.LOCK_B_L. The programmed values of SADMA_CFG0.LOCK_CH and SADMA_CFG0.LOCK_CH_L are ignored.
- SADMA_CFG0.LOCK_B = 1 and SADMA_CFG0.LOCK_CH = 1 – Two cases arise:
  - SADMA_CFG0.LOCK_B_L <= SADMA_CFG0.LOCK_CH_L – In this case, the DMA transfer proceeds as if SADMA_CFG0.LOCK_CH_L = SADMA_CFG0. LOCK_B_L and the programmed value of SADMA_CFG0.LOCK_CH_L is ignored. Thus, if bus locking is enabled over the DMA transfer level, then channel locking is enabled over the DMA transfer level, regardless of the programmed value of SADMA_CFG0.LOCK_CH_L
  - SADMA_CFG0.LOCK_B_L > SADMA_CFG0.LOCK_CH_L – The programmed value of SADMA_CFG0.LOCK_CH_L is used. Thus, if bus locking is enabled over the DMA block transfer level and channel locking is enabled over the DMA transfer level, then channel locking is performed over the DMA transfer level.

Locking Levels

If locking is enabled for a channel, then locking of the AHB master bus interface at a programmed locking transfer level is activated when the channel is first granted the AHB master bus interface at the start of that locking transfer level. It continues until the locking transfer level has completed; that is, if channel 0 has enabled channel level locking at the block transfer level, then this channel locks the master bus interface when it is first granted the master bus interface at the start of the block transfer, and continues to lock the master bus interface until the block transfer has completed.

Source and destination block transfers occur successively in time, and a new source block cannot commence until the previous destination block has completed. When both source and destination are on the same bus layer, then block level locking is terminated on completion of the block to the destination. If they are on separate layers, then block-level locking is terminated on completion of the block on that layer—when the source block on the source bus layer completes, and when the destination block on the destination bus layer completes. The same is true for DMA transfer-level locking.

Transaction-level locking is different due to the fact that source and destination transactions occur independently in time, and the number of source and destination transactions in a DMA block or DMA transfer do not have to match. When the source and destination are on the same AHB layer, then transaction-level locking is cleared at the end of a source or destination transaction only if the opposing peripheral is not currently in the middle of a transaction.

For example, if locking is enabled at the transaction level and an end-of-source transaction is signaled, then this disables locking only if one if the following is true:

- The destination is on a different bus layer
- The destination is on the same bus layer, but the channel is not currently in the middle of a transaction to the destination peripheral.

The same rules apply when an end-of-destination transaction is signalled.

If channel-level or bus-level locking is enabled for a channel at the transaction level, and either the source or destination of the channel is a memory device, then the locking is ignored and the channel proceeds as if locking (bus or channel) is disabled.

Note: Since there is no notion of a transaction level for a memory peripheral, then transaction-level locking is not allowed when either source or destination is memory.

79.3.12 Arbitration for AHB Master Interface

Each SATA DMA channel has two request lines that request ownership of a particular master bus interface: channel source and channel destination request lines.

Source and destination arbitrate separately for the bus. Once a source/destination state machine gains ownership of the master bus interface and the master bus interface has ownership of the AHB bus, then bus transfers can proceed between the peripheral and SATA DMA. *Figure 321* illustrates the arbitration flow of the master bus interface.

An arbitration scheme decides which of the request lines (2 * DMAH_NUM_CHANNELS) is granted the particular master bus interface. Each channel has a programmable priority. A request for the master bus interface can be made at any time, but is granted only after the current bus transfer (burst or single) has completed. Therefore, if the master interface is transfering data for a lower priority channel and a higher priority channel requests for service, then the master interface will complete the the current burst for the lower priority channel before switching over to transfer data for the higher priority channel.

To prevent a channel from saturating the master bus interface, it can be given a maximum bus burst length (MAX_ABRST field in SADMA_CFG0 register) at channel setup time. This also prevents the master bus interface from saturating the AHB bus where the system arbiter cannot change the grant lines until the end of an undefined length burst.

The following is the interface arbitration scheme employed when no channel has locked (Channel Locking) the arbitration for the master bus interface:

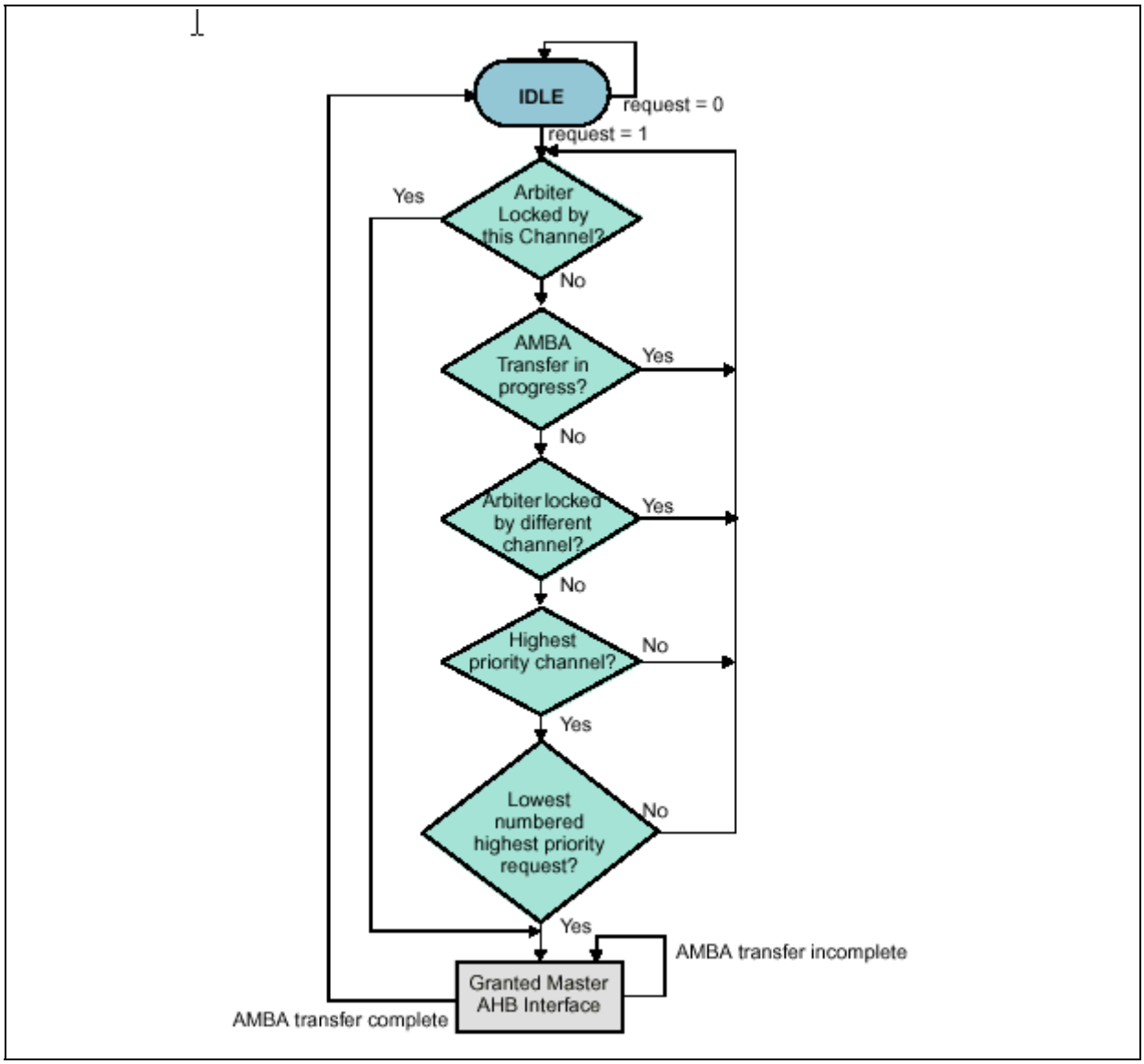
- If only one request line is active at the highest priority level, then that request with the highest priority wins ownership of the AHB master bus interface. It is not necessary for the priority levels to be unique.

  If more than one request is active at the highest requesting priority, then these competing requests proceed to a second tier of arbitration.
- If equal priority requests occur, then the lower-numbered channel is granted.

  In other words, if a peripheral request attached to Channel 7 and a peripheral request attached to Channel 8 have the same priority, then the peripheral attached to Channel 7 is granted first.

Note: A channel source is granted before the destination if both have their request lines asserted when a grant decision is been made. A channel source and channel destination inherit their channel priority and therefore always have the same priority.

Figure 321: Arbitration Flow for Master Bus Interface



79.3.13 Scatter/gather

Scatter is relevant to a destination transfer. The destination bus address is incremented/decremented by a programmed amount – the scatter increment – when a scatter boundary is reached. *Figure 322* shows an example destination scatter transfer. The destination address is incremented/decremented by the value stored in the destination scatter increment (DSR0.DSI) field multiplied by the number of bytes in a single bus transfer to the destination ((decoded value of SADMA_CTRL0.DST_TR_WIDTH)/8) when a scatter boundary is reached. The number of destination transfers between successive scatter boundaries is programmed into the Destination Scatter Count, DSC, field of the DSRx register.

Scatter is enabled by writing a 1 to the SADMA_CTRL0.DST_SCATTER_EN field. The SADMA_CTRL0.DINC field determines if the address is incremented, decremented, or remains fixed when a scatter boundary is reached. If the SADMA_CTRL0.DINC field indicates a fixed-address control throughout a DMA transfer, then the SADMA_CTRL0.DST_SCATTER_EN field is ignored, and the scatter feature is automatically disabled.

Gather is relevant to a source transfer. The source bus address is incremented/decremented by a programmed amount when a gather boundary is reached. The number of source transfers between successive gather boundaries is programmed into the Source Gather Count (SGR0.SGC) field. The source address is incremented/decremented by the value stored in the

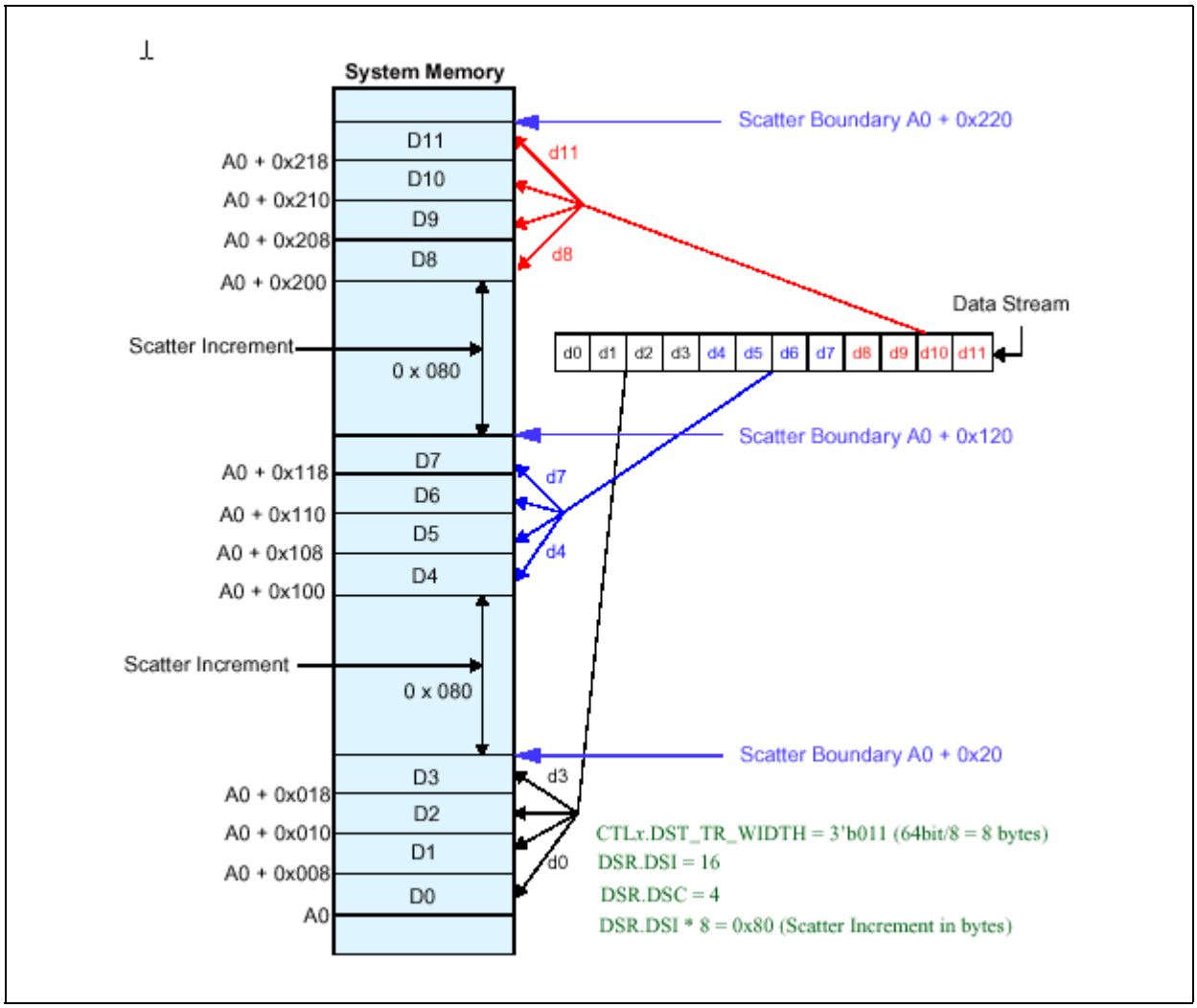
source gather increment (SGR0.SGI) field multiplied by the number of bytes in a single bus transfer from the source ((decoded value of SADMA_CTRL0.SRC_TR_WIDTH)/8) when a gather boundary is reached.

Gather is enabled by writing a '1' to the SADMA_CTRL0.SRC_GATHER_EN field. The SADMA_CTRL0.SINC field determines if the address is incremented, decremented, or remains fixed when a gather boundary is reached. If the SADMA_CTRL0.SINC field indicates a fixed-address control throughout a DMA transfer, then the SADMA_CTRL0.SRC_GATHER_EN field is ignored, and the gather feature is automatically disabled.

Note: For multi-block transfers, the counters that keep track of the number of transfers left to reach a gather/scatter boundary are re-initialized to the source gather count (SGR0.SGC) and destination scatter count (DSC), respectively, at the start of each block transfer.

Figure 322 shows an example of a destination scatter transfer:

Figure 322: Example of Destination Scatter Transfer



As an example of gather increment, consider the following:

SRC_TR_WIDTH = 3'b 010 (32 bit)

SGR.SGC = 0x04 (source gather count)

SADMA_CTRL0.SRC_GATHER_EN = 1 (source gather enabled)

SAR0 = A0 (starting source address)

Figure 323 shows a source gather when SGR.SGI = 0x0:

Figure 323: Source Gather when SGR.SGI = 0x0

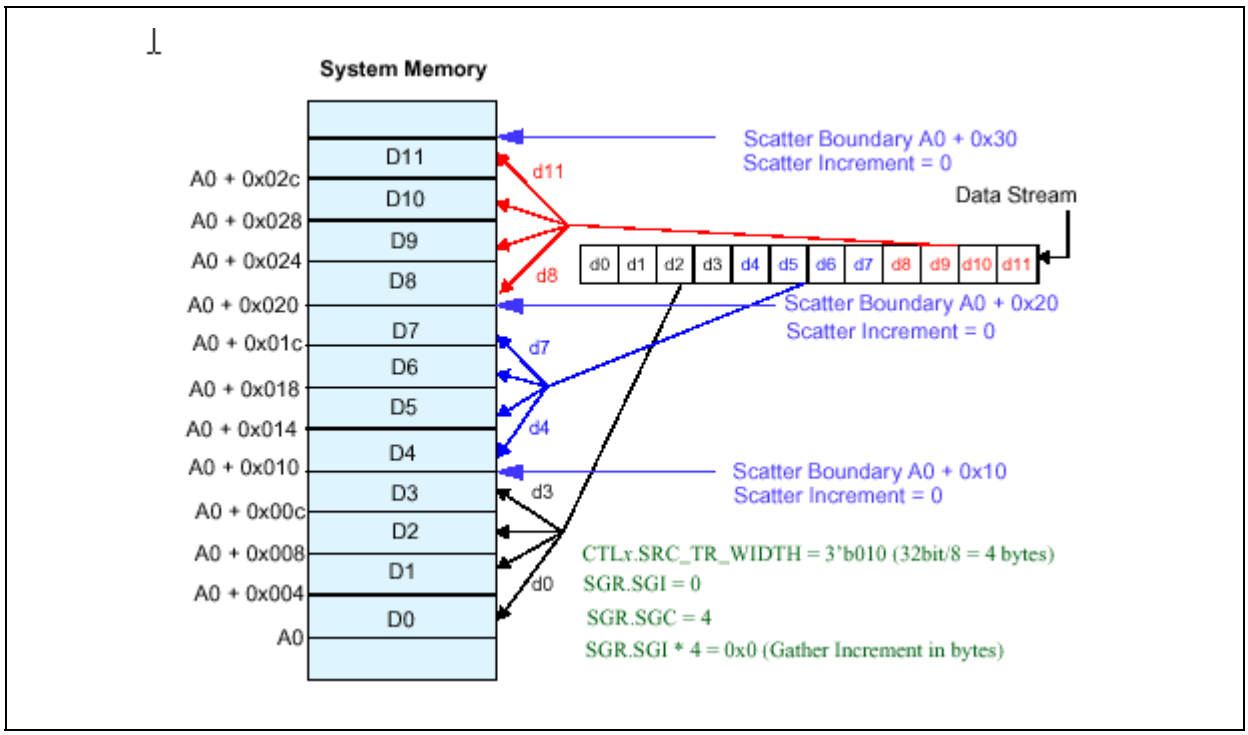
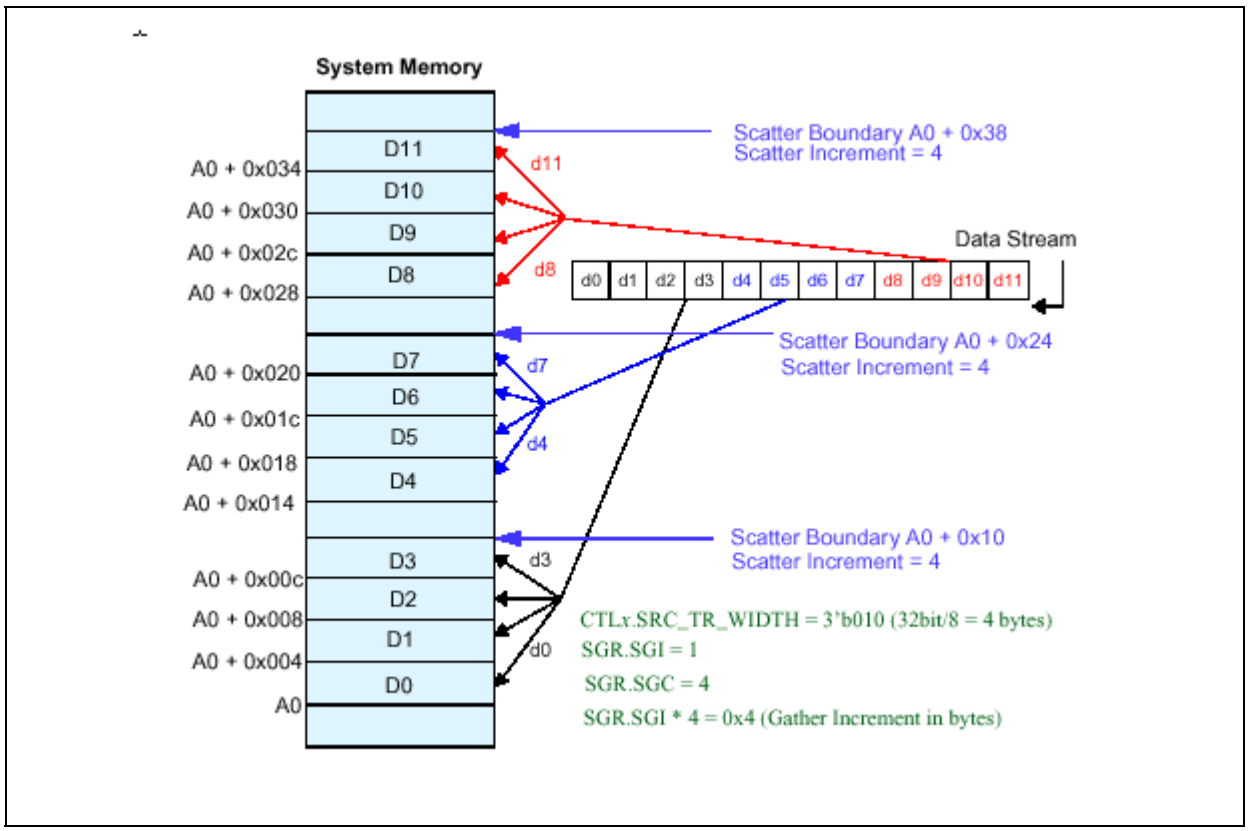


Figure 324 shows a source gather when SGR.SGI = 0x01:

Figure 324: Source Gather when SGR.SGI = 0x1



In general, if the starting address is A0 and SADMA_CTRL0.SINC = 2'b00 (increment source address control) then the transfer will be:

A0, AO + TWB, A0 + 2*TWB .....(A0 + (SGR.SGC-1)*TWB) <-scatter_increment->

(A0 + (SGR.SGC*TWB) + (SGR.SGI *TWB))

where TWB is the transfer width in bytes, decoded value of SADMA_CTRL0.SRC_TR_WIDTH/8 = src_single_size_bytes.

80 Serial ATA (SATA) host

80.1 Overview

The SATA host implements the Serial Advanced Technology Attachment (ATA) storage interface for physical storage devices.

This chapter provides an overview of the SATA host. The topics are:

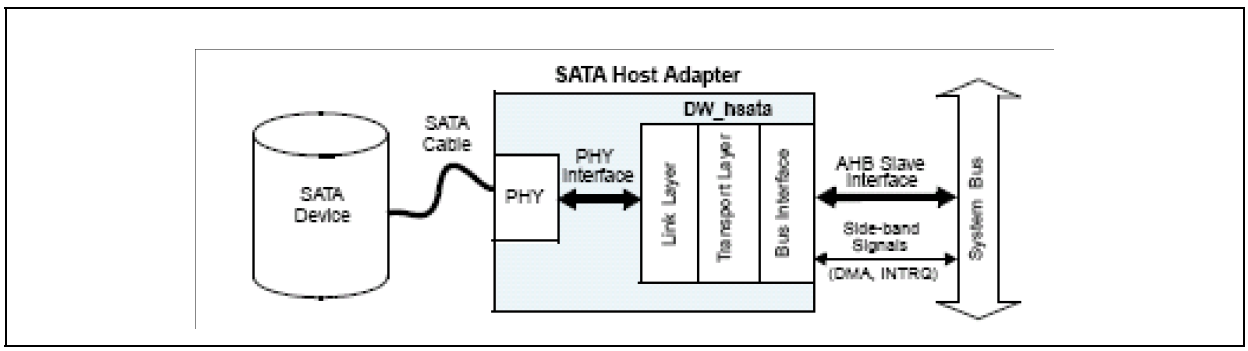
- System Overview and Block Diagram,
- General Product Description,
- Features List,
- Standards Compliance.

80.1.1 System Overview and Block Diagram

The SATA host consists of three main functional blocks: Bus interface, Transport Layer, and Link Layer. Together with the physical layer (PHY) it forms a complete Serial ATA (SATA) host adapter interface.

Figure 1 shows how the SATA host fits into the SATA system.

Figure 325: SATA system block diagram



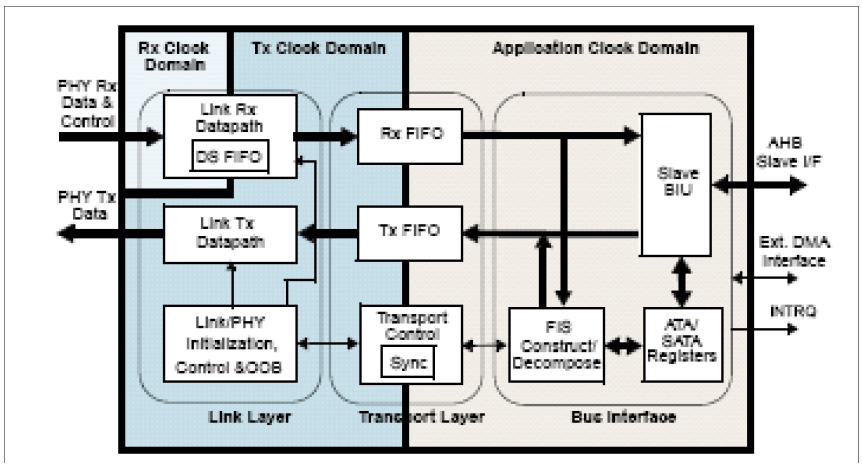
80.1.2 General product description

SATA is a half-duplex system. Either a receive or transmit operation is performed between the two agents (host and device) at any given time, but not both. This is true only for data frames transfer. Control traffic (primitives) is full duplex to maintain receiver synchronization. For example, SYNC primitives are sent continuously while both host and device sides are in their idle states.

80.1.2.1 Block diagram

Figure 326 provides a high-level view of the SATA host architecture, followed by a brief description.

Figure 326: DW_HSATA block diagram



The SATA host operates primarily in three clock domains: receive (Rx), transmit (Tx), and application. However, in some systems, an additional clock is supported for Rx OOB signal detection. The application clock is provided by the system bus; its frequency is application-specific. Rx and Tx clocks are generated in the PHY and are 150, 75, or 37.5MHz for Gen1 (300, 150, or 75MHz for Gen2), depending on the PHY data width. Most of the Link Layer (both receive and transmit data paths) and part of the Transport Layer operates in the Tx clock domain. Rx clock is a recovered clock from the PHY and is used for clocking data into the SATA host and for performing optional 8b/10b decoding, dropping ALIGNs and aligning data. Finally, the optional Rx OOB clock can be set by the user. See Link Layer Overview for more details.

Note: All SATA host clocks are asynchronous to each other in general, but the Rx clock cannot exceed the Tx clock by more than 350 ppm. Synopsys, Inc. July 28, 2004

The Bus Interface block provides a configurable AHB slave interface, which is used to connect to the system bus. Host application software (driver) accesses SATA host using a set of ATA, SATA and SATA host-specific registers. DMA flow control is implemented with several handshaking signals compatible with the DW_ahb_dmac DMA controller.

The following subsections describe the basic SATA host functionality in terms of Rx/Tx data, control paths and layers.

80.1.2.2 Internal communication paths

The following subsections describe the internal SATA host communication paths.

Receive path

The following list highlights the Receive Path functionality:

- Link Layer optionally detects Rx OOB signalling sequences from the device and initializes the system.
- Link Layer receives SATA frames and primitives from the PHY sent by the device.
- Link Layer transmits primitives using backchannel to control the data flow.
- Link Layer optionally performs 10B/8B decoding, data alignment, descrambling, deframing and CRC checking on the frame FIS (frame payload), stripping SOF, EOF and other control primitives. A data-stream FIFO (DS FIFO) is used to cross data from the Rx to the Tx clock domain.

- Link Layer passes FIS to the Transport Layer Rx FIFO. The Rx FIFO is a two-clock 33-bit wide FIFO. Data is written in the Tx clock domain and is read in the application clock domain, thus providing clock-crossing function for receive data. The Rx FIFO depth is a configurable parameter.
- Transport Layer decodes FIS type from the least-significant byte of the FIS first DWORD.
- Transport Layer updates ATA/SATA registers if it is a register type FIS. Some FIS types are used for control only (example: DMA activate FIS).
- Transport Layer passes data from the Rx FIFO to the system bus using DMA or PIO modes using the Bus Interface (example: Data FIS). In PIO mode, application software performs series of reads from the Data register. In DMA mode, Bus Interface requests transaction from the DMA controller, which in turn generates read access to the RxFIFO.

Transmit path

The following list highlights the Transmit Path functionality:

- Link Layer optionally generates Tx OOB signalling sequences to the device and initializes the system.
- Transport Layer receives data from the system bus either via DMA channel or from the ATA/SATA registers after application software writes to them. In PIO mode, application software performs series of writes to the Data register. In DMA mode, Bus Interface requests transaction from the DMA controller, which in turn generates write access to the TxFIFO.
- Transport Layer constructs the appropriate FIS and passes it to the Link Layer via Tx FIFO. The Tx FIFO is a two-clock 33-bit wide FIFO. Data is written in the application clock domain and read in the Tx clock domain, thus providing clock-crossing function for transmit data.
- Link Layer receives FIS from the Tx FIFO, calculates CRC, frames the data by inserting SOF, EOF and other flow control primitives, scrambles Repeat Primitives and data, optionally performs 8B10B encoding and multiplexes the data out to the PHY at the correct data width.
- Link Layer receives flow control and status primitives from the device on the backchannel and passes relevant information and status to the Transport Layer.

Control path

The following list highlights the Control Path functionality:

- Link control implements Link Layer and initialization state machines and interfaces to the PHY and to the Transport Layer, as well as controls data movement in both directions. Link detects all the PHY and Link Layer errors and passes them to the Transport Layer. In addition, the Link Layer controls PHY interface power management.
- Transport control implements SATA Transport Layer state machine and interfaces to the Link Layer and to the bus interface. Various synchronizing modules provides clock-crossing functionality for all control signals and data. The Transport Layer detects all the PHY/Link Layer/Transport Layer errors and passes them to the Bus Interface.
- Bus Interface control (not shown in the block diagram) provides Rx/Tx FIFO control functions, error handling, ATA/SATA register control, power management, DMA flow control, PHY control/status, interrupt control.

Bus interface overview

The Bus Interface connects the Transport Layer to the system bus through the AHB slave interface and provides the following functions:

- FIS decomposition/construction - these are Transport Layer functions, which include reception of the FIS from the RxFIFO and placing the contents into the ATA registers; forming the FIS from the ATA registers for transmission via TxFIFO.
- Interrupt control - ATA master (Device 0) emulation and other SATA host-specific interrupts.

- DMA control - provides DMA controller external handshaking signals and FIFOs flow control.
- Error control - gathers errors from the PHY, Link Layer, and Transport Layer and updates corresponding ATA and SATA error registers.
- Host power management control and status.
- PHY/Link Layer status/control - monitors interface state and provides PHY
- Register control - implements ATA/ATAPI, SATA and SATA host registers.

AHB slave interface has the following features:

- Compliant with ARM AMBATM specification rev. 2.0.
- Supports little-endian or big-endian byte ordering.
- Supports 16, 32, 64, 128, 256-bit data bus width;
- Supports 8 and 16-bit bus transfers with 16-bit data bus. Supports 8, 16 and 32-bit bus transfers with 32-bit or wider data bus.
- Supports AHB master BUSY transfer and early burst termination.
- Supports burst transfers in DMA mode (registers and PIO mode accesses are done using either single transfer (SINGLE) or unspecified-length incrementing burst (INCR) that has a burst length one).
- Supports OKAY or ERROR response (RETRY or SPLIT are not supported).

Bus configuration is selected in coreConsultant prior to synthesis. DMA transfers are initiated by the SATA host based on the state of the Rx/TxFIFO using several handshake signals. These signals are compatible with DW_ahb_dmac DMA controller.

80.1.2.3 Transport layer overview

The Transport Layer constructs Frame Information Structures (FIS) for transmission and decomposes received FIS. The following paragraph outlines the Transport Layer functions.

Transport layer FIS construction

The following list describes how an FIS is constructed:

- Gathers FIS content based on the type of FIS requested by the host application layer software.
- Places FIS content in the proper order.
- Notifies the Link Layer of required frame transmission and passes FIS to Link.
- Manages TxFIFO flow, notifies Link of required flow control via TxFIFO flags.
- Receives frame receipt acknowledge from Link.
- Reports good transmission or errors to requesting higher layer.

Transport layer FIS decomposition

The following list describes how an FIS is decomposed:

- Receives the FIS from the Link Layer.
- Determines FIS type.
- Distributes the FIS content to the locations indicated by the FIS type.
- Reports good reception or errors to the application layer.

80.1.2.4 Link layer overview

The Link Layer controls initialization between the Link Layer, PHY and a connected device. In addition, the Link Layer optionally generates and detects OOB signalling, transmits and receives frames, transmits primitives based on control signals from the Transport Layer and PHY, and receives primitives from the PHY layer that are used to control the Transport and Link Layers.

The Link Layer also controls power management via control of the PHY and PHY interface. A summary of the main Link Layer features are described in the following sections.

Initialization

The following list describes the Link Layer initialization process:

- Negotiates with its peer Link Layer and PHY to bring the system to an initialized state ready to transmit and receive data.
- Optionally transmits OOB sequences to the PHY, or instructs the PHY to generate them. These sequences are then forwarded to the remote device PHY per SATA specifications, causing device PHY initialization.
- Optionally receives OOB sequences or condition detection signals from the PHY, causing advancement in the initialization routine. When OOB detection is performed by the PHY, the Link Layer detects these conditions via phy_cominit and phy_comwake signals from the PHY.
- Once it has been determined that the remote device is ready, the Link Layer transitions to normal operation.

Power management

The following list describes the Link Layer power management process:

- Monitors power management signals from the Transport Layer and PHY.
- When power modes are enabled, disables normal data transmission on the Tx interface and prevents internal Rx data reception until a wake-up request from the Transport Layer or a remote device via the PHY is seen. Re-enables PHY interface when Transport Layer or remote device request wake-up.

Frame transmission

The following list describes the Link Layer frame transmission process:

- Negotiates with its peer Link Layer to transmit a frame, resolves arbitration conflicts if both host and device request transmission.
- Inserts frame envelope around Transport Layer data (i.e. SOF, CRC, EOF).
- Receives data in the form of DWORDs from the Transport Layer.
- Calculates CRC on Transport Layer data.
- Transforms (scrambles) data and Repeat Primitive DWORDs in such a way to distribute the potential EMI emissions over a broader range.
- Optionally performs 8b/10b encoding.
- Transmits frame.
- Provides frame flow control in response to status from the TxFIFO or the peer Link Layer.
- Receives frame receipt acknowledge from peer Link Layer.
- Reports good transmission or Link/PHY Layer errors to Transport Layer.

Frame receipt

The following list describes the Link Layer frame receipt process:

- Acknowledges to the peer Link Layer readiness to receive a frame.
- Receives data in the form of optionally encoded characters from the PHY layer.
- Optional decodes the encoded 8b/10b character stream into aligned DWORDs of data.
- Performs data alignment/realignment.
- Drops Repeat Primitive Data.
- De-scrambles the control and data DWORDs received from a peer Link Layer.
- Removes the envelope around frames (i.e. SOF, CRC, EOF Primitives).

- Calculates CRC on the received DWORDs.
- Compares the calculated CRC to the received CRC.
- Provides frame flow control in response to the status from the Rx FIFO or the peer Link Layer.
- Reports good reception status or Link/PHY Layer errors to Transport Layer and the peer Link Layer.

80.1.2.5 Physical layer overview

Note: Physical layer (PHY) is not part of the SATA host. However, the PHY/Link interface has been designed to be configurable to control and function with almost any PHY. (Some glue logic might be required for some PHYs). Also see PHY DWORD Latency Budget Requirements.

In general, a SATA PHY provides the following services:

- Transmit a 1.5 Gb/sec differential NRZ serial stream at specified voltage levels.
- Provide a 100 Ohm matched termination (differential) at the transmitter.
- Serialize a 8/10, 16/20 or 32/40 bit wide parallel input from the Link for transmission.
- Receive a 1.5 Gb/sec +350/-2650ppm differential NRZ serial stream.
- Provide a 100 Ohm matched termination (differential) at the receiver.
- Extract data and clock from the serial stream.
- Deserialize the serial stream.
- Detect the K28.5 comma character and forward it to the Link Layer along with the 8/10, 16/20 or 32/40 bit wide parallel output.
- Accept and forward proper power-on/initialization sequences to device.
- Interact with power management modes by way of Link Layer control.
- Provide proper transmitter and receiver differential pair impedances.
- Handle the input data rate frequency variation due to a spread spectrum transmitter clock (Nominal +0/-0.5% slow frequency variation, over a 33.33 µs up/down triangular wave period).

80.1.3 Features list

The following are features of dwc_sata_host:

- Supports SATA 1.5 Gbps Generation 1 speed
- Designed for easy upgrade to Generation 2 speed (3Gbps)
- Compliant with SATA specification and applicable Serial ATA II Extension specification
- Highly configurable PHY interface
- Provides additional user defined PHY status and control ports
- Optional OOB signalling detection and generation
- Gen2 speed negotiation when Tx OOB signalling is selected
- Optional 8b/10b encoding/decoding Synopsys, Inc. July 28, 2004
- Optional data alignment circuitry
- Supports power management features
- Supports BIST loopback modes
- Supports single SATA device
- ATA/ATAPI master-only emulation
- Configurable AMBA AHB Slave interface
- DMA flow-control and interrupt side-band signals

80.1.4 Standards compliance

The SATA host complies with the Serial ATA II specification. Information about Serial ATA can be found at the following website:

`http://serialata.org`

The SATA specification can be found in the ANSI T13 Committee document 1532D Volume 3 (ATA/ATAPI-7 V3) at:

`http://www.t13.org`

The Serial ATA Specification describes the following:

- Defines how a disk drive communicates with a disk controller
- Describes how a cable should plug into a connector
- Shows connector pin assignments as well as other similar interoperability points

The Serial ATA II Specification does not discuss system design.

80.2 Configurable parameters

Thia section describes the configuration parameters for the SATA host that you can set in coreConsultant.

80.2.1 Parameter descriptions

Table 238 describes the configuration parameters for the SATA host.

Table 238: SATA host Configuration Parameters

Label	Parameter definition
AHB Data bus width	Selects AHB data bus width **Parameter name**: AHB_DATA_WIDTH **Values**: 16, 32, 64, 128, 256 **Default** Value: 32 **Dependencies**: None
AHB bus endianness	Selects endianness for the AHB interface **Parameter Name**: BIG_ENDIAN **Legal** Values: Little, Big **Default Value**: Little **Dependencies**: None
AHB ERROR response	Selects whether the SATA host slave interface returns an ERROR response (if True) or OKAY response (if False) on the hresp bus when an illegal access is attempted over the AHB slave interface (see “AHB Error Conditions” on page 40). **Parameter name**: RETURN_ERR_RESP **Values**: True, False **Default value**: Fals Dependencies: None
BIST loopback checking depth	Selects the depth of error checking for BIST mode **Parameter name**: BIST_MODE **Values**: FIS (BIST errors counted one per FIS) DWORD (BIST errors counted one per DWORD) **Default value**: FIS **Dependencies**: None

Table 238: SATA host Configuration Parameters

Label	Parameter definition
ID Number	Selects the ID code value that will be hardwired and read back by software from the IDR register. The ID value is a 32-bit number. **Parameter name**: HSATA_ID_NUM **Values**: 0 to 232-1 **Default value**: 0 **Dependencies**: None
Rx FIFO depth (DWORDs)	RxFIFO depth parameter in DWORDs **Parameter name**: RXFIFO_DEPTH **Values**: 64, 128, 256, 512, 1024, 2048 **Default value**: 128 (512 bytes) **Dependencies**: None
Rx FIFO memory type	Memory location selection **Parameter name**: RX_MEM_SELECT **Values**: External (user-supplied memory) Internal (DesignWare memory instantiation) **Default value**: External **Dependencies**: Only changeable to Internal if (RXFIFO_DEPTH <= 256)
Rx FIFO memory read port	Memory read port type **Parameter name**: RX_MEM_MODE **Values**: Async (Asynchronous) Sync (Synchronous) D**efault value**: Async **Dependencies**: Only changeable to Sync if (RXFIFO_DEPTH>256 or RX_MEM_SELECT==External)
Tx FIFO depth (DWORDs)	TxFIFO depth parameter in DWORDs **Parameter name**: TXFIFO_DEPTH Values: 32, 64, 128, 256, 512, 1024, 2048 **Default value**: 128 (512 bytes) **Dependencies**: None
Tx FIFO memory type	Memory location selection **Parameter name**: TX_MEM_SELECT **Values**: External (user-supplied memory) Internal (DesignWare memory instantiation) Default value: External Dependencies: Only changeable to Internal if (TXFIFO_DEPTH <= 256)
Tx FIFO memory read port	Tx FIFO Memory read port type. **Parameter name**: TX_MEM_MODE **Values**: Async (Asynchronous) Sync (Synchronous) **Default value**: Async **Dependencies**: Only changeable to Sync if (TXFIFO_DEPTH>256 or TX_MEM_SELECT==External)
Scan test clock mode	Scan testing uses separate clocks or one single clock. When a single clock is used, all false paths between clock domains are disabled. **Parameter name**: SCAN_CLOCK_MODE **Values**: Separate (clocks used) **Single** (clock used) **Default value**: Separate (clocks used) **Dependencies**: None

Table 238: SATA host Configuration Parameters

Label	Parameter definition
Test mode lock-up latches	Clock domain crossing lock-up latches inserted in design before clock boundary crossings, for test purposes only. Used when only one scan chain is used for testing and the synthesis tool does not insert its own lock-up latches. **Parameter name**: LATCH_MODE **Values**: Exclude (No lock-up latches) Include (Lock-up latches inserted) **Default value**: Exclude (NoLatches) **Dependencies**: Only changeable to Include if (SCAN_CLOCK_MODE == Single)
User-defined PHY status width	Number of optional, user-defined PHY status port bits that map to a register that can be read by the system software via the PHYSR register. When this is set to 0, no status ports or registers are included in the design. **Parameter name**: PHY_STAT_W **Values**: 0 to 32 **Default value**: 0 **Dependencies**: None
User-defined PHY control width	Number of optional, user-defined PHY control port bits that map to the PHYCR register and that can be written by the system software. When this is set to 0, no control ports or registers are included in the design. **Parameter name**: PHY_CTRL_W **Values**: 0 to 32 **Default value**: 0 **Dependencies**: None
User-defined PHY control default power on value	When the PHY control port (phy_ctrl) is used, this sets the default power on value that is present on its output after an asynchronous reset of the hclk clock domain. **Parameter name**: PHY_CTRL_DEF **Values**: 0x0 to 0xFFFFFFFF **Default value**: 0x0 **Dependencies**: Only changeable when (PHY_CTRL_W > 0)
PHY reset mode	Determines whether the PHY Reset port is active-high or active-low and sets the port name accordingly. **Parameter name**: PHY_RST_MODE **Values**: Low (PHY Reset port is (phy_reset_n) active-low) High (PHY Reset port is (phy_reset) active-high) **Default value**: High **Dependencies**: None
PHY data width	PHY Rx and Tx data width in bytes. Actual bit width is dependent on the ENCODE_MODE parameter, that is: when PHY_DATA_WIDTH == 2 and ENCODE_MODE == Exclude, the actual data bit width is 16. When ENCODE_MODE == Include, the actual data bit width is 20. **Parameter name**: PHY_DATA_WIDTH **Values**: 1, 2, 4 **Default value**: 2 **Dependencies**: None
Tx OOB sequence generation	When the PHY does not generate Tx OOB signalling sequences, this can be achieved in the Link Layer by including this circuit. **Parameter name**: TX_OOB_MODE **Values**: Exclude (No Tx OOB generation circuit implemented) Include (Tx OOB generation circuit is implemented) **Default value**: Include **Dependencies**: None

Table 238: SATA host Configuration Parameters

Label	Parameter definition
Rx OOB Sequence Detection	When the PHY does not detect Rx OOB signalling sequences, this can be achieved in the Link Layer by including this circuit. **Parameter name**: RX_OOB_MODE **Values**: Exclude (No Rx OOB detection circuit implemented) Include (Rx OOB detection circuit is implemented) **Default value**: Include **Dependencies**: Can only be set to Include if TX_OOB_MODE is also set to Include, due to speed negotiation requirements.
Rx OOB Clock Mode	Rx OOB Detection clock uses Rx Clock (clk_rbc0) or uses a separate clock. See Link Layer Rx OOB Sequence Detection for details. **Parameter name**: RXOOB_CLK_MODE **Values**: RxClock (Rx OOB Detection occurs in clk_rbc0 clock domain and no additional clock port is available) Separate (Rx OOB Detection occurs in clk_rxoob clock domain, which is included via the clk_rxoob clock port) **Default value**: Separate (clk_rxoob port is included in design) **Dependencies**: Only enabled when RX_OOB_MODE == Include.
Rx OOB Clock Frequency	Specifies the Rx OOB sampling clock frequency in MHz. **Parameter name**: RXOOB_CLK_FREQ **Values**: 30 -limited by synthesis technology library **Default value**: 30 **Dependencies**: Only enabled when RX_OOB_MODE == Include and RXOOB_CLK_MODE == Separate.
Rx Data Alignment	When PHY_DATA_WIDTH is 2 or 4 bytes and data can be misaligned or contain bubbles in the words or DWORDs, setting this value to Misaligned enables data alignment circuitry in the design. When set to Aligned, this circuitry is not present in the design. **Parameter name**: ALIGN_MODE **Values**: Misaligned (Rx Data can be misaligned and/or have bubbles at the byte level) Aligned (Rx Data always aligned with K characters always in least significant byte and no bubbles at the byte level) **Default value**: Aligned (Rx Data always aligned) **Dependencies**: Not enabled and not relevant when PHY_DATA_WIDTH == 1
8b/10b Encoding/ Decoding	When 8b/10b encoding and decoding is not performed by the PHY, a DesignWare Foundation 8b/10b encoder and decoder can be included in the design. When excluded, the logic is not implemented. This parameter affect Rx and Tx data bit widths. See “Phy Data Width” parameter for details. **Parameter name**: ENCODE_MODE **Values**: Exclude (No 8b/10b encoding/decoding included in the design) Include (b/10b encoding/decoding is included in the design) **Default value**: Exclude (No 8b/10b encoding/decoding) **Dependencies**: None

81 Serial ATA (SATA) registers

Register addresses are provided as *SATABaseAddress* + offset.

The *SATABaseAddress* is:

0x1920 9000.

Note: The areas not allocated are reserved and must not be accessed.

There are a few additional SATA registers described in Chapter 21: *System configuration registers on page 171*.

Table 239: DMA controller register summary

Register	Description	Offset	Default	Type
Channel				
SADMA_SAR0	Channel 0 source address	0x400	0x0000 0000	R/W
SADMA_DAR0	Channel 0 destination address	0x408		
SADMA_DAR0	Reserved	-	-	-
SADMA_CTRL0_LSB	Channel 0 control LSB	0x418	0x0030 4825[a]	R/W
SADMA_CTRL0_MSB	Channel 0 control MSB	0x41C	0x0000 0002[a]	
SADMA_SSTAT0	Reserved	-	-	-
SADMA_DSTAT0				
SADMA_SSTATAR0				
SADMA_DSTATAR0				
SADMA_CFG0_LSB	Channel 0 configuration LSB	0x440	0x0000 0C00	R/W
SADMA_CFG0_MSB	Channel 0 configuration MSB	0x444	0x0000 0004	
SADMA_SGR0	Reserved	-	-	-
SADMA_DSR0				
Interrupt				
SADMA_RAW_TFR	Raw status for INTTFR interrupt	0x6C0	0x0000 0000	RO
SADMA_RAW_BLOCK	Raw status for INTBLOCK interrupt	0x6C8		
SADMA_RAW_SRC_TRAN	Raw status for INTSRCTRAN interrupt	0x6D0		
SADMA_RAW_DST_TRAN	Raw status for INTDSTTRAN interrupt	0x6D8		
SADMA_RAW_ERR	Raw status for INTERR interrupt	0x6E0		
SADMA_TFR_STA	Status for INTTFR interrupt	0x6E8		
SADMA_BLOCK_STA	Status for INTBLOCK interrupt	0x6F0		
SADMA_SRC_TRAN_STA	Status for INTSRCTRAN interrupt	0x6F8		
SADMA_DST_TRAN_STA	Status for INTDSTTRAN interrupt	0x700		
SADMA_ERR_STA	Status for INTERR interrupt	0x708		
SADMA_MASK_TFR	Mask for INTTFR interrupt	0x710	0x0000 0000	R/W
SADMA_MASK_BLK	Mask for INTBLOCK interrupt	0x718		
SADMA_CLEAR_SRC_TRAN	Mask for INTSRCTRAN interrupt	0x720		
SADMA_CLEAR_DST_TRAN	Mask for INTDSTTRAN interrupt	0x728		
SADMA_MASK_ERR	Mask for INTERR interrupt	0x730		

Table 239: DMA controller register summary

Register	Description	Offset	Default	Type
SADMA_CLEAR_TFR	Clear for INTTFR interrupt	0x738	0x0000 0000	WO
SADMA_CLEAR_BLOCK	Clear for INTBLOCK interrupt	0x740		
SADMA_CLEAR_SRC_TRAN	Clear for INTSRCTRAN interrupt	0x748		
SADMA_CLEAR_DST_TRAN	Clear for INTDSTTRAN interrupt	0x750		
SADMA_CLEAR_ERR	Clear for INTERR interrupt	0x758		
SADMA_STATUS_INT	Status for each interrupt type	0x760	0x0000 0000	RO
Software handshaking				
SADMA_REQSRC	Reserved	-	-	-
SADMA_REQDST				
SADMA_SGLREQSRC				
SADMA_SGLREQDST				
SADMA_LSTSRC				
SADMA_LSTDST				
Miscellaneous				
SADMA_DMA_CFG	DMAC configuration	0x798	0x0000 0000	R/W
SADMA_CH_EN	DMAC channel enable	0x7A0		
Reserved		0x7A8 - 0x7F8	-	-

a. Fixed values which must not be changed.

Table 240: SATA host controller register summary

Register	Offset	Description	Default	Width (bits)	Type
Shadow ATA/ATAPI: CDR - command block, CLR - control block					
SATA_CDR0	0x800	PIO mode: data Read only for PIO read/receive operation, write only for PIO write/transmit operation.	0xXXXX	16	RO/WO
		DMA mode: FIFO location Read only for DMA read/receive operation, write only for DMA write/ transmit operation.	Undefined	16/32	RO/WO
SATA_CDR1	0x804	Error	0xFF	8	RO
		Feature (current value)	0x00	8	WO
		Feature expanded (previous value)	0x00		
SATA_CDR2	0x808	Sector count (current value)	0xFF	8	R/W
		Sector count expanded (previous value)			
SATA_CDR3	0x80C	Sector number (current value)			
		Sector number expanded (previous value)			
SATA_CDR4	0x810	Cylinder low (current value)			
		Cylinder low expanded (previous value)			
SATA_CDR5	0x814	Cylinder high (current value)			
		Cylinder high expanded (previous value)			
SATA_CDR6	0x818	Device/head	0xEF		

Table 240: SATA host controller register summary

Register	Offset	Description	Default	Width (bits)	Type
SATA_CDR7	0x81C	Status	0x7F[a]	8	RO
		Command	0x00	8	WO
SATA_CLR0	0x820	Alternative status	0x7F	8	RO
		Device control	0x00	8	WO
SATA					
SATA_SCR0	0x824	SStatus	0x0000 0000	32	RO
SATA_SCR1	0x828	SError	0x0000 0000	32	R/W
SATA_SCR2	0x82C	SControl			
SATA_SCR3	0x830	SActive			
SATA_SCR4	0x834	SNotification			
Reserved	0x838 - 0x860	-	-	-	-
SATA host					
SATA_FPTAGR	0x864	First party DMA tag	0x0000 0000	32	RO
SATA_FPBOR	0x868	First party DMA buffer offset			
SATA_FPTCR	0x86C	First party DMA transfer count			
SATA_DMACR	0x870	DMA control	0x0000 0000	32	R/W
SATA_DBTSR	0x874	DMA burst transaction size	0x0008 0010		
SATA_INTPR	0x878	Interrupt pending	0x0000 0000		
SATA_INTMR	0x87C	Interrupt mask			
SATA_ERRMR	0x880	Error mask			
SATA_LLCR	0x884	Link layer control	0x0000 0007		
SATA_PHYCR	0x888	PHY control	0x0003 801C[b]		
SATA_PHYSR	0x88C	PHY status	0x0000 0000	32	RO
Reserved	0x890 - 0x8F4	-	-	-	-
SATA_ VERSIONR	0x8F8	SATA host version	0x3130 302A[c]	32	RO
SATA_IDR	0x8FC	SATA host ID	0x100A 0208[c]	32	
Reserved	0x900 - 0xBFF	Access results in bus error response	-	-	-

a. Status (CDR7) and Alternative status (CLR0) registers are set to 0x7F on power-up, then 0x80 when device presence is detected via PHY READY condition.

b. For cut 1.*x*, this value must be changed to 0x0013 704A for correct operation; however, for cut 2 onwards, the default value must not be changed.

c. Fixed value which must not be changed.

For more details on programming these registers, see the Synopsys documents listed in Chapter 78: *Serial ATA (SATA) subsystem on page 909*.

81.1 DMA controller

81.1.1 Channel

SADMA_SAR0 Channel 0 source address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAR																															

Address: *SATABaseAddress* + 0x400

Type: R/W

Reset: 0x0000 0000

Description: The starting bus source address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the source address of the current bus transfer. If this is not already the case, hardware aligns this address to the source transfer width, SADMA_CTRL0.SRC_TR_WIDTH field. Refer to Section : *Hardware re-alignment of SAR/DAR registers* on page 1023.

For information on how the SAR0 is updated at the start of each SATA DMA block for multi-block transfers, refer to Table 220: *Programming of transfer types and channel register update method* on page 924.

[31:0] **SAR**: Current Source Address of DMA transfer
Updated after each source bus transfer. The *SINC* field in the SADMA_CTRL0 register determines whether the address increments, decrements, or is left unchanged on every source bus transfer throughout the block transfer.

SADMA_DAR0 Channel 0 destination address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAR																															

Address: *SATABaseAddress* + 0x408

Type: R/W

Reset: 0x0000 0000

Description: The starting bus destination address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the destination address of the current bus transfer. If this is not already the case, hardware aligns this address to the destination transfer width, SADMA_CTRL0.*DST_TR_WIDTH* field. Refer to Section : *Hardware re-alignment of SAR/DAR registers* on page 1023.

For information on how the DAR0 is updated at the start of each SATA DMA block for multi-block transfers, refer to Table 220: *Programming of transfer types and channel register update method* on page 924 and Section 79.2.4.1: *Programming examples* on page 928.

[31:0] **DAR**: Current Destination address of DMA transfer
Updated after each destination bus transfer. The DINC field in the SADMA_CTRL0 register determines whether the address increments, decrements or is left unchanged on every destination bus transfer throughout the block transfer.

Hardware re-alignment of SAR/DAR registers

Hardware aligns the value of the SADMA_SAR0 and SADMA_DAR0 registers in the direction of the address control field — SADMA_CTRL0.SINC and SADMA_CTRL0.DINC — when the current values of and DAR0 are not aligned to the programmed transfer width, SADMA_SADMA_CTRL0.SRC_TR_WIDTH and SADMA_SADMA_CTRL0.DST_TR_WIDTH.

The SADMA_SAR0 and SADMA_DAR0 registers are aligned upwards for an incrementing address control and downwards for a decrementing address control.

If the SADMA_CTRL0.SINC and SADMA_CTRL0.DINC fields indicate a fixed FIFO mode addressing scheme – _CTRL0.SINC/_CTRL0.DINC = 2'b1x – then the _CTRL0.SINC[0] and _CTRL0.DINC[0] registers control the direction of the address alignment if the current SAR0 and DAR0 addresses are not aligned to the transfer width.

```
SADMA_CTRL0.SINC[1:0]/SADMA_CTRL0.DINC[1:0] = 2'b10 : Align SARx/DARx upwards
SADMA_CTRL0.SINC[1:0]/SADMA_CTRL0.DINC[1:0] = 2'b11 : Align SARx/DARx
downwards
```

SADMA_CTRL0_LSB Channel 0 control LSB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LLP_SRC_EN	LLP_DST_EN	SMS		DMS		TT_FC			Reserved	DST_SCATTER_EN	SRC_GATHER_EN	SRC_MSIZE			DEST_MSIZE			SINC		DINC		SRC_TR_WIDTH			DST_TR_WIDTH			INT_EN

Address: *SATABaseAddress* + 0x418

Type: R/W

Reset: 0x0030 4825

Description: This register contains fields that control the DMA transfer.

The _CTRL0 register is part of the block descriptor (linked list item – LLI) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled. For information about the behavior of this register between blocks, refer to Section 79.2.3.1: *Multi-block transfers on page 922*.

If status write-back is enabled, the upper word of the control register, _CTRL0[63:32], is written to the control register location of the LLI in system memory at the end of the block transfer.

Note: This register must be programmed before enabling the channel.

[31:29] **Reserved**

[28] **LLP_SRC_EN**
Block chaining is enabled on the source side only if the LLP_SRC_EN field is high and LLPx.LOC is non-zero; for more information, see Section : *Block chaining using linked lists on page 922*.
This field does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN (page 918) is not selected or if DMAH_CHx_HC_LLP is selected. In this case the read-back value is always 0.

[27] **LLP_DST_EN**
Block chaining is enabled on the destination side only if the LLP_DST_EN field is high and LLPx.LOC is non-zero. For more information, see Section : *Block chaining using linked lists on page 922*.
This field does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN (page 918) is not selected or if DMAH_CHx_HC_LLP is selected. In this case, the read-back value is always 0.

[26:25] **SMS**: Source master select
Identifies the Master Interface layer from which the source device (peripheral or memory) is accessed.
00: AHB master 1 10: AHB master 3
01: AHB master 2 11: AHB master 4
The maximum value of this field that can be read back is DMAH_NUM_MASTER_INT – 1. This field does not exist if the configuration parameter DMAH_CHx_SMS (page 917) is hardcoded; in this case, the read-back value is always the hardcoded value.

[24:23] **DMS**: Destination master select
Identifies the the Master Interface layer where the destination device (peripheral or memory) resides.
00: AHB master 1 10: AHB master 3
01: AHB master 2 11: AHB master 4
The maximum value of this field that can be read back is DMAH_NUM_MASTER_INT – 1. This field does not exist if the configuration parameter DMAH_CHx_DMS (page 917) is hardcoded; in this case, the read-back value is always the hardcoded value.

[22:20] **TT_FC**: Transfer type and flow control
The following transfer types are supported:
• Memory to Memory
• Memory to Peripheral
• Peripheral to Memory
• Peripheral to Peripheral
Flow control can be assigned to the SATA DMA block, the source peripheral, or the destination peripheral. Table 243: *SADMA_CTRL0.TT_FC field decoding on page 1027* lists the decoding for this field. For more information on transfer types and flow control, refer to Section 79.3.1: *Setup/Operation of SATA DMA Transfers on page 951*.
If the configuration parameter DMAH_CHx_FC (page 916) is set to DMA_FC_ONLY, then TT_FC[2] does not exist and TT_FC[2] always reads back 0. If DMAH_CHx_FC is set to SRC_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b10. If DMAH_CHx_FC is set to DST_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b11.
The reset value is configuration-dependent:
TT_FC[0] = 1'b0
TT_FC[1] = DMAH_CHx_FC[1] &
(!DMAH_CHx_FC[0])
TT_FC[2] = DMAH_CHx_FC[1] ^
DMAH_CHx_FC[0]

[19] **Reserved**

[18] **DST_SCATTER_EN**: Destination scatter enable
0: Scatter disabled 1: Scatter enabled
Scatter on the destination side is applicable only when the SADMA_CTRL0.DINC bit indicates an incrementing or decrementing address control.
This field does not exist if DMAH_CHx_DST_SCA_EN (page 920) is not selected; in this case, the read-back value is always 0.

[17] **SRC_GATHER_EN**: Source gather enable
0: Gather disabled 1: Gather enabled
Gather on the source side is applicable only when the SADMA_CTRL0.SINC bit indicates an incrementing or decrementing address control.
This field does not exist if DMAH_CHx_SRC_GAT_EN (page 920) is not selected; in this case, the read-back value is always 0.

[16:14] **SRC_MSIZE**: Source burst transaction length
Number of data items, each of width SADMA_CTRL0.SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 241: *SADMA_... CTRL0.SRC_MSIZE and DEST_MSIZE decoding on page 1026* lists the decoding for this field; refer to "Section 79.3.10.6: *Choosing the Receive Watermark level on page 1000*.
All remaining bits in this field do not exist and read back as 0.

[13:11] **DEST_MSIZE**: Destination burst transaction length
Number of data items, each of width SADMA_CTRL0.DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 241: *SADMA_... CTRL0.SRC_MSIZE and DEST_MSIZE decoding on page 1026* lists the decoding for this field; refer to Section 79.3.10.6: *Choosing the Receive Watermark level on page 1000*.
All surplus bits in this field do not exist and read back as 0.

[10:9] **SINC**: Source address Increment
Indicates whether to increment or decrement the source address on every source bus transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to “No change.”
00: Increment 1x = No change
01: Decrement
Note: Incrementing or decrementing is done for alignment to the next SADMA_CTRL0.SRC_TR_WIDTH boundary.

[8:7] **DINC**: Destination address increment
Indicates whether to increment or decrement the destination address on every destination bus transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to “No change.”
00: Increment 1x = No change
01: Decrement
Note: Incrementing or decrementing is done for alignment to the next SADMA_CTRL0.DST_TR_WIDTH boundary.

[6:4] **SRC_TR_WIDTH**: Source transfer width
Table 242: *SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field. Mapped to AHB bus “hsize.” For a nonmemory peripheral, typically the peripheral (source) FIFO width.
This value must be less than or equal to DMAH_Mx_HDATA_WIDTH, where x is the bus layer 1 to 4 where the source resides.
This field does not exist if the parameter DMAH_CHx_STW (page 917) is hardcoded. In this case, the read-back value is always the hardcoded source transfer width, DMAH_CHx_STW.

[3:1] **DST_TR_WIDTH**: Destination transfer width
Table 242: *SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* lists the decoding for this field. Mapped to AHB bus “hsize.” For a non-memory peripheral, typically rgw peripheral (destination) FIFO width.
This value must be less than or equal to DMAH_Mk_HDATA_WIDTH, where k
is the bus layer 1 to 4 where the destination resides.
This field does not exist if DMAH_CHx_DTW (page 917) is hardcoded. In this case, the read-back value is always the hardcoded destination transfer width, DMAH_CHx_DTW.

[0] **INT_EN**: Interrupt enable
1: All interrupt-generating sources are enabled.

SADMA_CTRL0_MSB Channel 0 control MSB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			DONE	BLOCK_TS											

Address: *SATABaseAddress* + 0x41C

Type: R/W

Reset: 0x0000 0002

Description:

[31:13] **Reserved**

[12] **DONE**

If status write-back is enabled, the upper word of the control register, _CTRL0_MSB[31:0], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set.

Software can poll the LLI _CTRL0.DONE bit to see when a block transfer is complete. The LLI _CTRL0.DONE bit should be cleared when the linked lists are set up in memory prior to enabling the channel.

For more information, refer to Section 79.2.3.1: *Multi-block transfers on page 922*.

[11:*b*] **Reserved**: Returns 0 on a read

$b = \log_2(\text{DMAH\_CH0\_MAX\_BLK\_SIZE} + 1) + 31$

[*b*:0] **BLOCK_TS**: Block transfer size

When the SATA DMA block is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size.

The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single bus beat.

The width of the single transaction is determined by _CTRL0.SRC_TR_WIDTH. For further information on setting this field, refer to Table 242: *SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding on page 1027* and Section 79.3.8.1: *Transfer Operation on page 965*.

Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller. When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at DMAH_CHx_MAX_BLK_SIZE, but the actual block size can be greater. Refer to Table 219: *SATA DMA parameters on page 913*.

$b = \log_2(\text{DMAH\_CHx\_MAX\_BLK\_SIZE} + 1) + 31$

Table 241: SADMA_... CTRL0.SRC_MSIZE and DEST_MSIZE decoding

SADMA_CTRL0.SRC_MSIZE / SADMA_CTRL0.DEST_MSIZE	Number of data items to be transferred (of width SADMA_CTRL0.SRC_TR_WIDTH or SADMA_CTRL0.DST_TR_WIDTH)
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

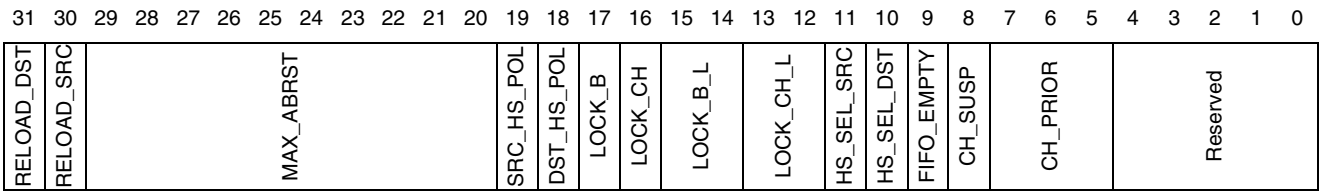
Table 242: SADMA_... CTRL0.SRC_TR_WIDTH and CTRL0.DST_TR_WIDTH decoding

SADMA_CTRL0.SRC_TR_WIDTH / SADMA_CTRL0.DST_TR_WIDTH	Size (bits)
000	8
001	16
010	32
011	64
100	128
101	256
11x	256

Table 243: SADMA_CTRL0.TT_FC field decoding

SADMA_CTRL0.TT_FC Field	Transfer type	Flow controller
000	Memory to memory	SATA DMA block
001	Memory to peripheral	SATA DMA block
010	Peripheral to memory	SATA DMA block
011	Peripheral to peripheral	SATA DMA block
100	Peripheral to memory	Peripheral
101	Peripheral to peripheral	Source peripheral
110	Memory to peripheral	Peripheral
111	Peripheral to peripheral	Destination peripheral

SADMA_CFG0_LSB Channel 0 configuration LSB



Address: *SATABaseAddress* + 0x440

Type: R/W

Reset: 0x0000 0C00

Description: This register contains fields that configure the DMA transfer. The channel configuration register remains fixed for all blocks of a multi-block transfer.

Note: This register must be programmed before enabling the channel.

[31] **RELOAD_DST**: Automatic destination reload
The DARx register can be automatically reloaded from its initial value at the end of every block for multiblock transfers. A new block transfer is then initiated. Refer to Table 220: *Programming of transfer types and channel register update method on page 924* for conditions under which this occurs.
This register does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN (page 918) is not selected. In this case, the readback value is always zero.

[30] **RELOAD_SRC**: Automatic source reload
The SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. Refer to Table 220: *Programming of transfer types and channel register update method on page 924* for conditions under which this occurs.
This field does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN (page 918) is not selected. In this case, the readback value is always zero.

[29:20] **MAX_ABRST**: Maximum bus burst length
Maximum bus burst length that is used for DMA transfers on this channel.
A value of 0 indicates that software is not limiting the maximum bus burst length for DMA transfers on this channel.
This field does not exist if the configuration parameter DMAH_MABRST (page 914) is not selected. In this case, the read-back value is always zero, and the maximum bus burst length cannot be limited by software.

[19] **SRC_HS_POL**: Source handshaking interface polarity
0: Active high 1: Active low
For information on this, refer to Section 79.3.3: *Handshaking Interface on page 952*.

[18] **DST_HS_POL**: Destination handshaking interface polarity
0: Active high 1: Active low
For information on this, refer to Section 79.3.3: *Handshaking Interface on page 952*.

[17] **LOCK_B**: Bus lock
When active, the bus bus master signal hlock is asserted for the duration specified in CFGx.LOCK_B_L. For more information, refer to Section 79.3.11.1: *Locked DMA Transfers on page 1002*.
This field does not exist if the configuration parameter DMAH_CHx_LOCK_EN is set to False (page 917). In this case, the read-back value is always zero.

[16] **LOCK_CH**: Channel lock
When the channel is granted control of the master bus interface and if the CFGx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CFGx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CFGx.LOCK_CH_L.
This field does not exist if the configuration parameter DMAH_CHx_LOCK_EN is set to False (page 917). In this case, the read-back value is always zero.

[15:14] **LOCK_B_L**: Bus lock level
Indicates the duration over which CFGx.LOCK_B bit applies.
00: Over complete DMA transfer
01: Over complete SATA DMA block transfer
1x = Over complete DMA transaction
This field does not exist if the parameter DMAH_CHx_LOCK_EN is set to False (page 917). In this case, the read-back value is always zero.

[13:12] **LOCK_CH_L**: Channel lock level
Indicates the duration over which CFGx.LOCK_CH bit applies.
00: Over complete DMA transfer
01: Over complete SATA DMA block transfer
1x = Over complete DMA transaction
This field does not exist if the configuration parameter DMAH_CHx_LOCK_EN is set to False (page 917). In this case, the read-back value is always zero.

[11] **HS_SEL_SRC**: Source software or hardware handshaking
Select. This register selects which of the handshaking interfaces, hardware or software, is active for source requests on this channel.
0: Hardware handshaking interface. Software-initiated transaction requests are ignored.
1: Software handshaking interface. Hardware-initiated transaction requests are ignored.
If the source peripheral is memory, then this bit is ignored.

[10] **HS_SEL_DST**: Destination software or hardware handshaking select
This register selects which of the handshaking interfaces, hardware or software, is active for destination requests on this channel.
0: Hardware handshaking interface. Software-initiated transaction requests are ignored.
1: Software handshaking interface. Hardware Initiated transaction requests are ignored.
If the destination peripheral is memory, then this bit is ignored.

[9] **FIFO_EMPTY**
Indicates if there is data left in the channel's FIFO. Can be used in conjunction with CFGx.CH_SUSP to cleanly disable a channel. For more information, refer to "Section 79.2.5: *Disabling a channel prior to transfer completion on page 949*.
0: Channel's FIFO not empty 1: Channel's FIFO empty

[8] **CH_SUSP**: Channel suspend
Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data.
0: Not Suspended
1: Suspend. Suspend DMA transfer from the source.
For more information, refer to "Section 79.2.5: *Disabling a channel prior to transfer completion on page 949*

[7:5] **CH_PRIOR**: Channel priority
A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the range: 0 to (DMAH_NUM_CHANNELS – 1)
A programmed value outside this range will cause erroneous behavior.

[4:0] **Reserved**

SADMA_CFG0_MSB Channel 0 configuration MSB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	DEST_PER				SRC_PER				SS_UPD_EN	DS_UPD_EN	PROT_CTRL			FIFO_MODE	FCMODE

Address: *SATABaseAddress* + 0x444
Type: R/W
Reset: 0x0000 0004
Description:

[31:15] **Reserved**

[14:11] **DEST_PER**
Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the destination of channel x if the CFGx.HS_SEL_DST field is 0. Otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface via the assigned hardware handshaking interface.
Note: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.

[10:7] **SRC_PER**
Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the source of channel x if the CFGx.HS_SEL_SRC field is 0. Otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface via the assigned hardware handshaking interface.
Note: For correct SATA DMA block operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.

[6] **SS_UPD_EN**: Source status update enable
Source status information is only fetched from the location pointed to by the SSTATARx register, stored in the SSTATx register and written out to the SSTATx location of the LLI (refer to Figure 272: *Mapping of Block Descriptor (LLI) in Memory to Channel Registers when DMAH_CHx_STAT_SRC is set to True on page 926*) if SS_UPD_EN is high.
Note: This enable is only applicable if DMAH_CHx_STAT_SRC (page 919) is set to True.
This field does not exist if the configuration parameter DMAH_CHx_STAT_SRC is set to False. In this case, the read-back value is always zero.

[5] **DS_UPD_EN**: Destination status update enable
Destination status information is only fetched from the location pointed to by the DSTATARx register, stored in the DSTATx register and written out to the DSTATx location of the LLI (refer to Figure 272: *Mapping of Block Descriptor (LLI) in Memory to Channel Registers when DMAH_CHx_STAT_SRC is set to True on page 926*) if DS_UPD_EN is high.
This field does not exist if the configuration parameter DMAH_CHx_STAT_DST is set to False. In this case, the read-back value is always zero.

[4:2] **PROT_CTRL**: Protection control
The default value of HPROT indicates a non-cached, nonbuffered, privileged data access. The reset value is used to indicate such an access.
HPROT[0] is tied high as all transfers are data accesses as there are no opcode fetches.
There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals.
Table 244: *PROT_CTRL field to HPROT Mapping on page 1030* shows the mapping of bits in this field to the HPROT[3:1] bus.

[1] **FIFO_MODE**: FIFO mode select
Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.
0: Space/data available for single bus transfer of the specified transfer width.
1: Space/data available is greater than or equal to half the FIFO depth for destination transfers and less than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.

[0] **FCMODE**: Flow control mode
Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.
0: Source transaction requests are serviced when they occur. Data pre-fetching is enabled.
1: Source transaction requests are not serviced until a destination transaction request occurs. In this mode the amount of data transferred from the source is limited such that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

Table 244: PROT_CTRL field to HPROT Mapping

1‘b1	HPROT[0]
CFG*x.PROT_CTRL*[1]	HPROT[1]
CFG*x.PROT_CTRL*[2] ->	HPROT[2]
CFG*x.PROT_CTRL*[3] ->	HPROT[3]

81.1.2 Interrupts

The following sections describe the registers pertaining to interrupts, their status, and how to clear them. For each channel, there are five types of interrupt sources:

- INT_TFR: DMA transfer complete interrupt
  This interrupt is generated on DMA transfer completion to the destination peripheral.
- INT_BLOCK: Block transfer complete interrupt
  This interrupt is generated on SATA DMA block transfer completion to the destination peripheral.
- INT_SRC_TRAN: Source transaction complete interrupt
  This interrupt is generated after completion of the last bus transfer of the requested single/ burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the source side

Note: If the source for a channel is memory, then that channel will never generate a IntSrcTran interrupt and hence the corresponding bit in this field will not be set.

- INT_DST_TRAN: Destination transaction complete interrupt
  This interrupt is generated after completion of the last bus transfer of the requested single/ burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the destination side

Note: If the destination for a channel is memory, then that channel will never generate the IntDstTran interrupt and hence the corresponding bit in this field will not be set.

- INT_ERR: Error interrupt
  This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.

There are several groups of interrupt-related registers:

- RAW_TFR, RAW_BLOCK, RAW_SRC_TRAN, RAW_DST_TRAN, RAW_ERR
- STATUS_TFR, STATUS_BLOCK, STATUS_SRC_TRAN, STATUS_DST_TRAN, STATUS_ERR
- MASK_TFR, MASK_BLOCK, MASK_SRC_TRAN, MASK_DST_TRAN, MASK_ERR
- CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, CLEAR_ERR
- STATUS_INT

When a channel has been enabled to generate interrupts, the following is true:

- Interrupt events are stored in the raw status registers.
- The contents of the raw status registers are masked with the contents of the mask registers.
- The masked interrupts are stored in the status registers.
- The contents of the status registers are used to drive the INT_* port signals.
- **Writing** to the appropriate bit in the clear registers clears an interrupt in the raw status registers and the status registers on the same clock cycle.

The contents of each of the five status registers is ORed to produce a single bit for each interrupt type in the Combined Status Register: STATUS_INT.

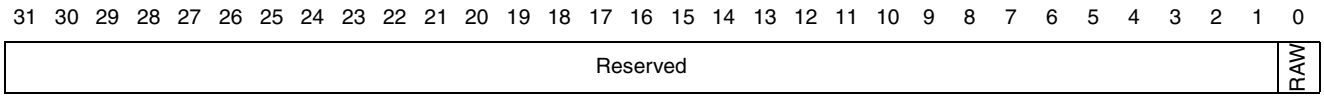
Note: The SDMA_CTRL0.INT_EN bit must be set for the enabled channel to generate any interrupts.

RAW_TFR, RAW_BLOCK, RAW_SRC_TRAN, RAW_DST_TRAN, RAW_ERR

Interrupt events are stored in these raw interrupt status registers before masking: RAW_TFR, RAW_BLOCK, RAW_SRC_TRAN, RAW_DST_TRAN, RAW_ERR.

Each bit in these registers is cleared by writing a 1 to the corresponding location in the CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, CLEAR_ERR registers.

SADMA_RAW_TFR Raw status for INTTFR interrupt



Address: *SATABaseAddress* + 0x6C0

Type: RO

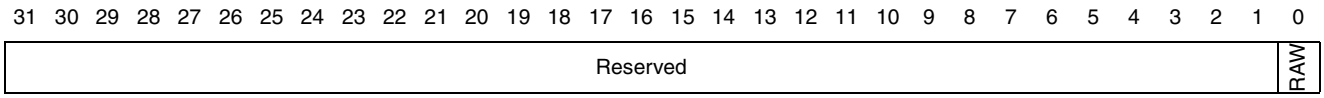
Reset: 0x0000 0000

Description:

[31:1] **Reserved**

[0] **RAW**: Raw interrupt status
Raw transfer complete

SADMA_RAW_BLOCK Raw status for INTBLOCK interrupt



Address: *SATABaseAddress* + 0x6C8

Type: RO

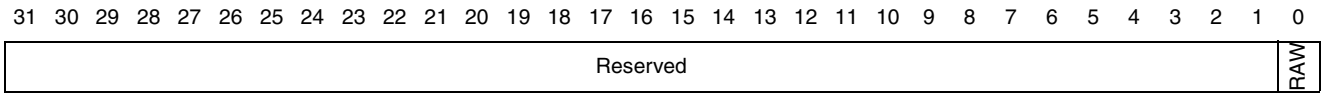
Reset: 0x0000 0000

Description:

[31:1] **Reserved**

[0] **RAW**: Raw interrupt status
Raw transfer complete

SADMA_RAW_SRC_TRAN Raw status for INTSRCTRAN interrupt



Address: *SATABaseAddress* + 0x6D0

Type: RO

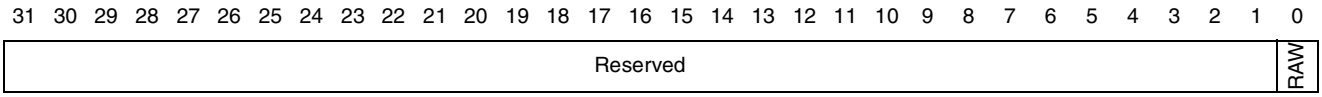
Reset: 0x0000 0000

Description:

[31:1] **Reserved**

[0] **RAW**: Raw interrupt status
Raw transfer complete

SADMA_RAW_DST_TRAN Raw status for INTDSTTRAN interrupt

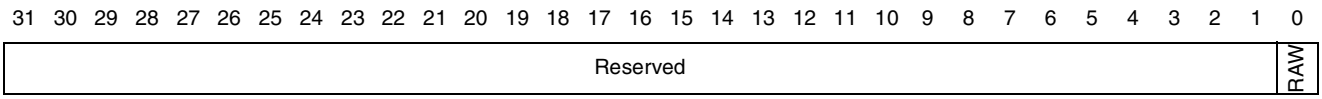


Address: *SATABaseAddress* + 0x6D8
Type: RO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **RAW**: Raw interrupt status
Raw transfer complete

SADMA_RAW_ERR Raw status for INTERR interrupt



Address: *SATABaseAddress* + 0x6E0
Type: RO
Reset: 0x0000 0000
Description:

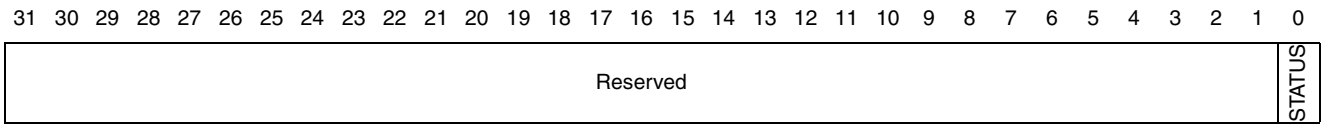
[31:1] **Reserved**

[0] **RAW**: Raw interrupt status
Raw transfer complete

STATUS_TFR, STATUS_BLOCK, STATUS_SRC_TRAN, STATUS_DST_TRAN, STATUS_ERR

All interrupt events from all channels are stored in these interrupt status registers after masking: STATUS_TFR, STATUS_BLOCK, STATUS_SRC_TRAN, STATUS_DST_TRAN, and STATUS_ERR.The contents of these register are used to generate the interrupt signals leaving the SATA DMA block.

SADMA_TFR_STA Status for INTTFR interrupt

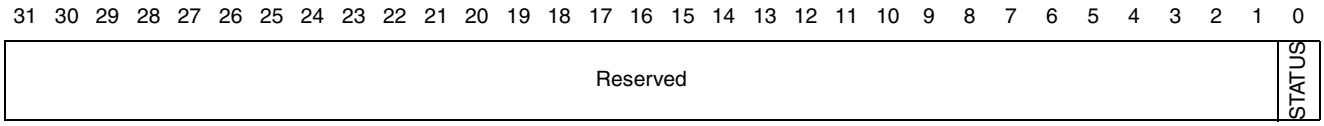


Address: *SATABaseAddress* + 0x6E8
Type: RO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **STATUS**: Interrupt status

SADMA_BLOCK_STA Status for INTBLOCK interrupt

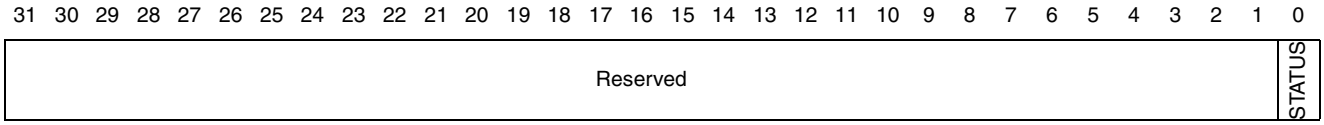


Address: *SATABaseAddress* + 0x6F0
Type: RO
Reset: 0x0000 0000
Description:

[31:24] **Reserved**

[0] **STATUS**: Interrupt status

SADMA_SRC_TRAN_STA Status for INTSRCTRAN interrupt

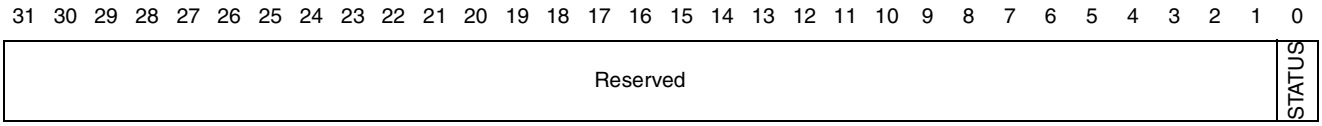


Address: *SATABaseAddress* + 0x6F8
Type: RO
Reset: 0x0000 0000
Description:

[31:24] **Reserved**

[0] **STATUS**: Interrupt status

SADMA_DST_TRAN_STA Status for INTDSTTRAN interrupt

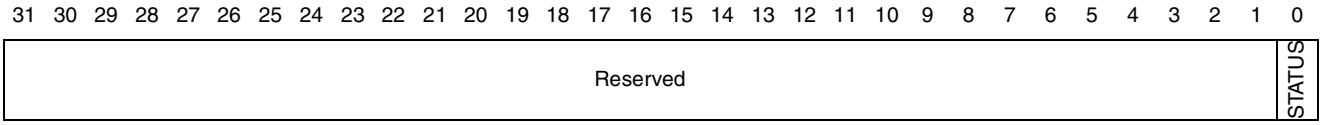


Address: *SATABaseAddress* + 0x700
Type: RO
Reset: 0x0000 0000
Description:

[31:24] **Reserved**

[0] **STATUS**: Interrupt status

SADMA_ERR_STA Status for INTERR interrupt



Address: *SATABaseAddress* + 0x708
Type: RO
Reset: 0x0000 0000
Description:

[31:24] **Reserved**

[0] **STATUS**: Interrupt status

MASK_TFR, MASK_BLOCK, MASK_SRC_TRAN, MASK_DST_TRAN, MASK_ERR

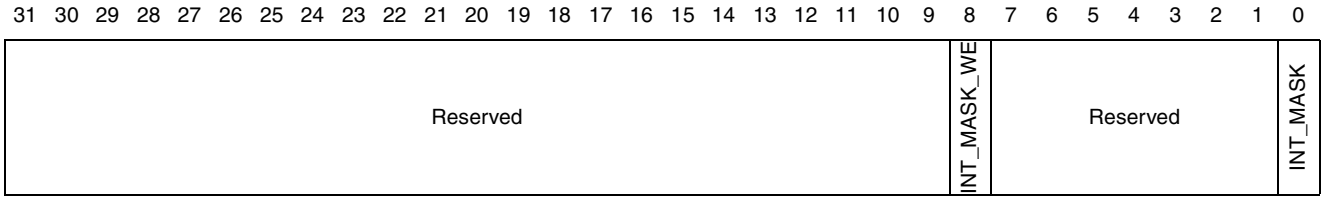
The contents of the raw status registers are masked with the contents of the mask registers: MASK_TFR, MASK_BLOCK, MASK_SRC_TRAN, MASK_DST_TRAN, MASK_ERR.

A channel's INT_MASK bit will only be written if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same bus write transfer. This allows software to set a mask bit without performing a read-modified write operation.

For example, writing hex 01x1 to the MASK_TFR register writes a 1 into MASK_TFR[0], while MASK_TFR[7:1] remains unchanged. Writing 0x00*xx* leaves MASK_TFR[7:0] unchanged.

Writing a 1 to any bit in these registers unmasks the corresponding interrupt, thus allowing the SATA DMA block to set the appropriate bit in the status registers and INT_* port signals.

SADMA_MASK_TFR Mask for INTTFR interrupt



Address: *SATABaseAddress* + 0x710

Type: R/W

Reset: 0x0000 0000

Description:

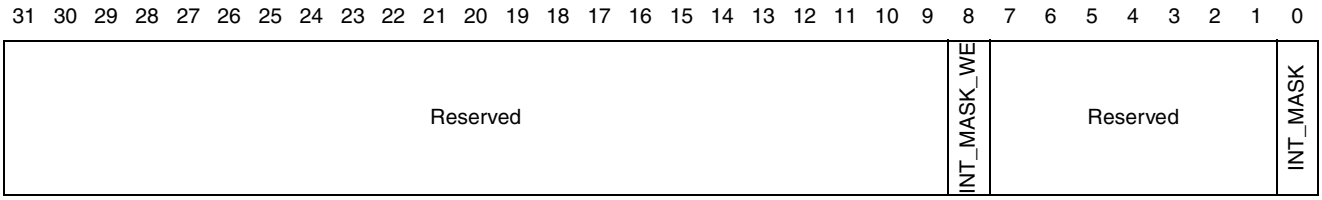
[31:9] **Reserved**

[8] **INT_MASK_WE**: Interrupt mask write enable (WO)
0: Write disabled 1: Write enabled

[7:1] **Reserved**

[0] INT_MASK: Interrupt mask (R/W)
0: Masked 1: Unmasked

SADMA_MASK_BLK Mask for INTBLOCK interrupt



Address: *SATABaseAddress* + 0x718
Type: R/W
Reset: 0x0000 0000
Description:

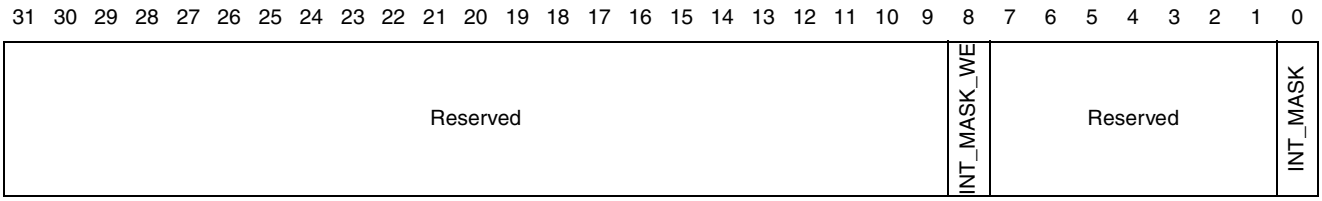
[31:9] **Reserved**

[8] **INT_MASK_WE**: Interrupt mask write enable (WO)
0: Write disabled 1: Write enabled

[7:1] **Reserved**

[0] INT_MASK: Interrupt mask (R/W)
0: Masked 1: Unmasked

SADMA_SRC_TRAN Mask for INTSRCTRAN interrupt



Address: *SATABaseAddress* + 0x720
Type: R/W
Reset: 0x0000 0000
Description:

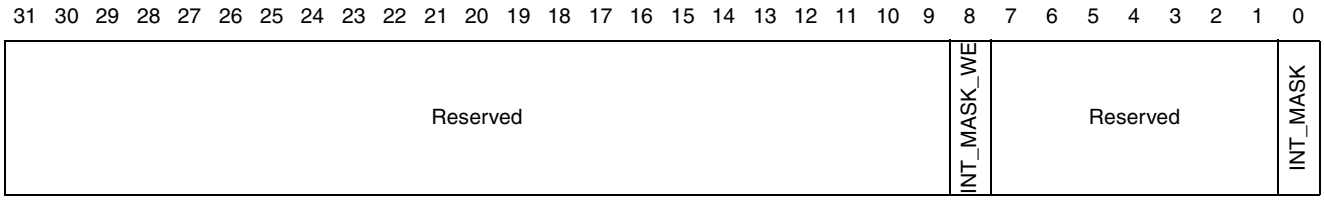
[31:9] **Reserved**

[8] **INT_MASK_WE**: Interrupt mask write enable (WO)
0: Write disabled 1: Write enabled

[7:1] **Reserved**

[0] INT_MASK: Interrupt mask (R/W)
0: Masked 1: Unmasked

SADMA_DST_TRAN Mask for INTDSTTRAN interrupt



Address: *SATABaseAddress* + 0x728
Type: R/W
Reset: 0x0000 0000
Description:

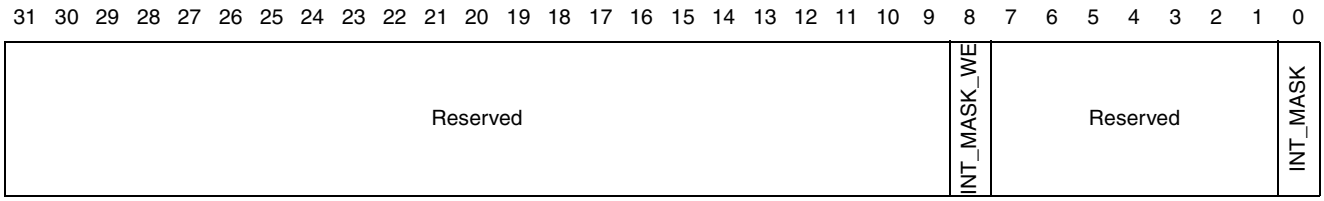
[31:9] **Reserved**

[8] **INT_MASK_WE**: Interrupt mask write enable (WO)
0: Write disabled 1: Write enabled

[7:1] **Reserved**

[0] INT_MASK: Interrupt mask (R/W)
0: Masked 1: Unmasked

SADMA_MASK_ERR Mask for INTERR interrupt



Address: *SATABaseAddress* + 0x730
Type: R/W
Reset: 0x0000 0000
Description:

[31:9] **Reserved**

[8] **INT_MASK_WE**: Interrupt mask write enable (WO)
0: Write disabled 1: Write enabled

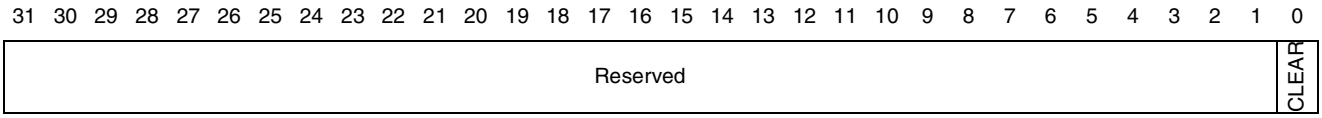
[7:1] **Reserved**

[0] INT_MASK: Interrupt mask (R/W)
0: Masked 1: Unmasked

CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, CLEAR_ERR

Each bit in the raw status and status registers is cleared on the same cycle by writing a 1 to the corresponding location in the clear registers: CLEAR_TFR, CLEAR_BLOCK, CLEAR_SRC_TRAN, CLEAR_DST_TRAN, CLEAR_ERR. Writing a 0 has no effect. These registers are not readable.

SADMA_CLEAR_TFR Clear for INTTFR interrupt

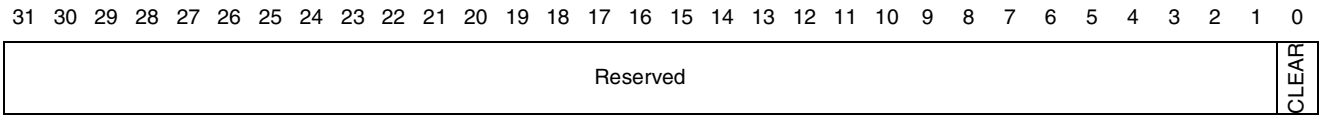


Address: *SATABaseAddress* + 0x738
Type: WO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **CLEAR**: Interrupt clear.
0: No effect 1: Clear interrupt

SADMA_CLEAR_BLOCK Clear for INTBLOCK interrupt

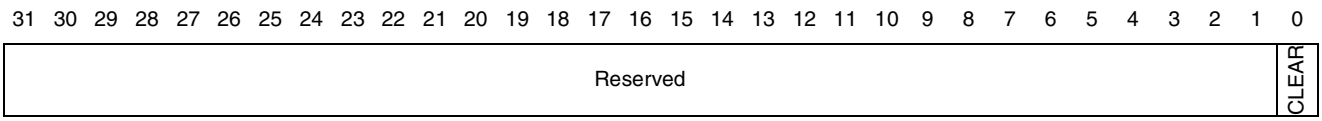


Address: *SATABaseAddress* + 0x740
Type: WO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **CLEAR**: Interrupt clear.
0: No effect 1: Clear interrupt

SADMA_CLEAR_SRC_TRAN Clear for INTSRCTRAN interrupt

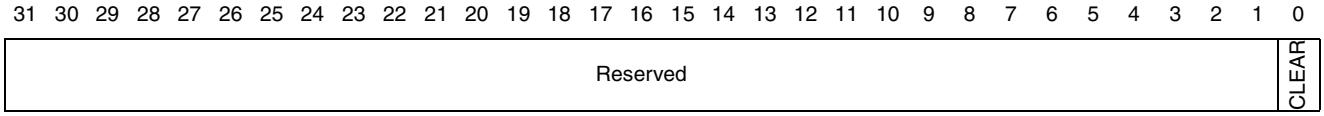


Address: *SATABaseAddress* + 0x748
Type: WO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **CLEAR**: Interrupt clear.
0: No effect 1: Clear interrupt

SADMA_CLEAR_DST_TRAN Clear for INTDSTTRAN interrupt

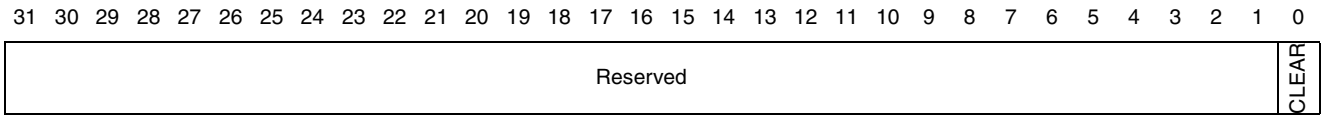


Address: *SATABaseAddress* + 0x750
Type: WO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **CLEAR**: Interrupt clear.
0: No effect 1: Clear interrupt

SADMA_CLEAR_ERR Clear for INTERR interrupt

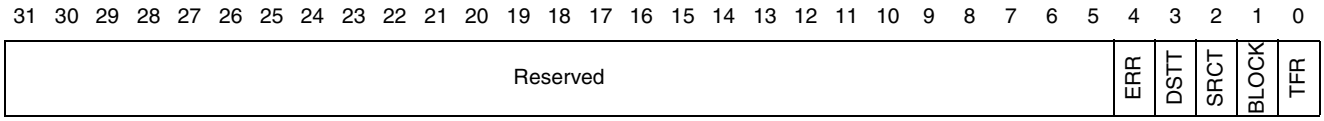


Address: *SATABaseAddress* + 0x758
Type: WO
Reset: 0x0000 0000
Description:

[31:1] **Reserved**

[0] **CLEAR**: Interrupt clear.
0: No effect 1: Clear interrupt

SADMA_STATUS_INT Status for each interrupt type



Address: *SATABaseAddress* + 0x760
Type: RO
Reset: 0x0000 0000
Description: The contents of each of the five Status Registers (STATUS_TFR, STATUS_BLOCK, STATUS_SRC_TRAN, STATUS_DST_TRAN, STATUS_ERR) are ORed to produce a single bit per interrupt type in the combined status register (STATUS_INT).
Description:

[31:5] **Reserved**

[4] **ERR**: OR of the contents of STATUS_ERR register

[3] **DSTT**: OR of the contents of STATUS_DST register

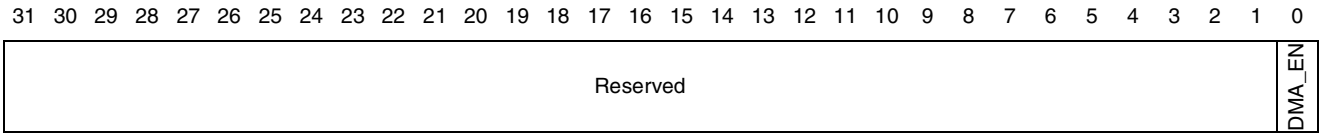
[2] **SRCT**: OR of the contents of STATUS_SRC_TRAN register

[1] **BLOCK**: OR of the contents of STATUS_BLOCK register

[0] **TFR**: OR of the contents of STATUS_TFR register

81.1.3 Miscellaneous

SADMA_DMA_CFG DMAC configuration



Address: *SATABaseAddress* + 0x798

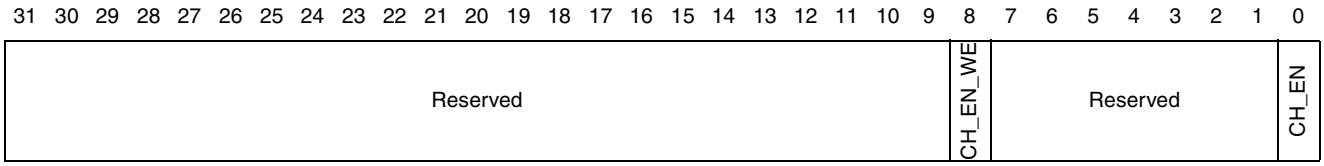
Type: R/W

Reset: 0x0000 0000

Description: Used to enable the SATA DMA block, which must be done before any channel activity can begin..

[31:0] **DMA_EN**: SATA DMA block enable
0: SATA DMA block disabled
1: SATA DMA block enabled

SADMA_CH_EN DMAC channel enable



Address: *SATABaseAddress* + 0x7A0

Type: R/W

Reset: 0x0000 0000

Description: If software needs to set up a new channel, then it can read this register in order to find out which channels are currently inactive; it can then enable an inactive channel with the required priority.

[31:9] **Reserved**

[8] **CH_EN_WE**: Channel enable write enable

[7:1] **Reserved**

[0] **CH_EN**: Enables/disables the channel
Setting this bit enables a channel, clearing this bit disables the channel.
0: Disable the Channel
1: Enable the Channel
The ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last bus transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer.

All bits of this register are cleared to 0 when the global SATA DMA block channel enable bit, DMA_CFG_REG[0], is 0. When the global channel enable bit is 0, then a write to the CH_EN_REG register is ignored and a read will always read back 0.

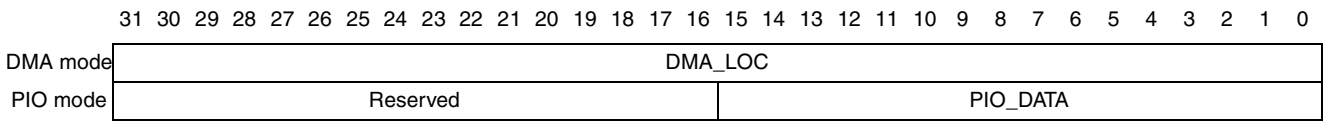
The channel enable bit, CH_EN.CH_EN, is written only if the corresponding channel write enable bit, CH_EN.CH_EN_WE, is asserted on the same bus write transfer. For example, writing hex 01x1 writes a 1 into CH_EN_REG[0], while CH_EN_REG[7:1] remains unchanged. Writing 0x00xx leaves CH_EN_REG[7:0] unchanged. Note that a read-modified write is not required.

For information on software disabling a channel by writing 0 to CH_EN.CH_EN, refer to “Section 79.2.5: *Disabling a channel prior to transfer completion* *on page 949*.

81.2 SATA host controller

81.2.1 Shadow ATA/ATAPI

SATA_CDR0 PIO data/DMA location



Address: *SATABaseAddress* + 0x800

Type: RO/WO

Reset: Undefined

Description: Used to transfer data from host-to-device and from device-tohost in PIO or DMA modes. Read-only during read/receive operation, and Write-only during write/transmit operation.

[31:0] **DMA_LOC**
This location can only be accessed by the host software or DMA controller when the SATA host is in the corresponding DMA mode:
- read (when the Data FIS is being received) or
- write (when the DMA Activate or DMA Setup FIS is received),
and the corresponding DMA handshake signal is asserted (dma_req or dma_single).
Both single and burst 32-bit or 16-bit bus transfers are supported in this mode.
NOTE: the same transfer size (either 16 or 32 bits) should be maintained during the whole DMA transfer.

[15:0] **PIO_DATA**
This register can only be accessed by the host software when the SATA host is in the corresponding PIO mode:
- read, when the PIO Setup FIS with D=1 is followed by the Data FIS, or
- write, when the PIO Setup FIS with D=0 is received.
During PIO read, the software performs a series of reads from this location, during PIO write - a series of writes to this location.
Only single 16-bit bus transfers are supported in this mode.

SATA_CDR1 Error/feature

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								ERROR/FEATURE/FEATURE_EXP							

Address: *SATABaseAddress* + 0x804

Type: RO/WO

Reset: 0xFF (error), 0x00 (feature)

Description: This location is used as one of the following:

- Error register when read;
- Feature/ Feature Expanded registers when written. Implemented as two-byte FIFO.

[31:8] **Reserved**

[7:0] **ERROR**

Error/diagnostic information from the device.

FEATURE

Current value of the Feature register. Determines the specific function of the SET FEATURES commands.

FEATURE_EXP: Feature expanded

Previous value of the Feature register (used for 48-bit addressing). It is pushed from the Feature register every time CDR1 is written.

SATA_CDR2 Sector count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SEC_CNT/SEC_CNT_EXP							

Address: *SATABaseAddress* + 0x808

Type: R/W

Reset: 0xFF

Description: These two 8-bit registers contain the number of sectors for Read/Write AT/ATAPI commands or command-specific parameters on some non-Read/Write commands. Implemented as two-byte FIFO.

[31:8] **Reserved**

[7:0] **SEC_CNT**

Current value of the CDR2 register when written. Can be read when Device Control register HOB bit is cleared CLR0.HOB=0).

SEC_CNT_EXP

Previous value of the CDR2 register (used for 48-bit addressing) pushed from the SEC_CNT register every time CDR2 is written. This value can be read when Device Control register HOB bit is set (CLR0.HOB=1).

SATA_CDR3 Sector number

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SEC_NUM/SEC_NUM_EXP							

Address: *SATABaseAddress* + 0x80C

Type: R/W

Reset: 0xFF

Description: These two 8-bit registers contain starting sector number (CHS mode) or LBA low value (LBA mode bits [7:0], [31:24]). Implemented as two-byte FIFO.

[31:8] **Reserved**

[7:0] **SEC_NUM**
Current value of the CDR3 when written. Can be read when CLR0.HOB=0. Contains LBA [7:0] bits.

SEC_NUM_EXP
Previous value of the CDR3 pushed from the Secnum every time CDR3 is written. Can be read when CLR0.HOB=1. Contains LBA [31:24] bits. Used for 48-bit addressing.

SATA_CDR4 Cylinder low

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CYLLOW_EXP/CYLLOW							

Address: *SATABaseAddress* + 0x810

Type: R/W

Reset: 0xFF

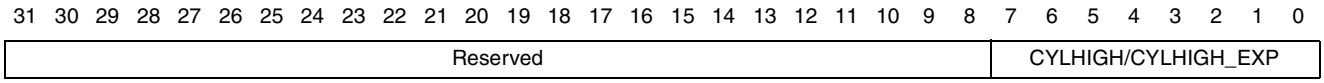
Description: These two 8-bit registers contain cylinder number low byte (CHS mode) or LBA mid value (LBA mode bits [15:8], [39:32]). Implemented as two-byte FIFO.

[31:8] **Reserved**

[7:0] **CYLLOW**
Current value of the CDR4 when written. Can be read when CLR0.HOB=0. Contains LBA [15:8] bits.

CYLLOW_EXP
Previous value of the CDR4 pushed from the Cyllow every time CDR4 is written. Can be read when CLR0.HOB=1. Contains LBA [39:32] bits. Used for 48-bit addressing.

SATA_CDR5 Cylinder high



Address: *SATABaseAddress* + 0x814

Type: R/W

Reset: 0xFF

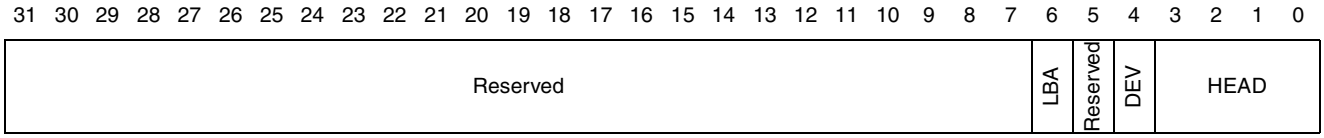
Description: These two 8-bit registers contain the cylinder-number high byte (CHS mode) or the LBA high value (LBA mode bits [23:16], [47:40]). Implemented as two-byte FIFO.

[31:8] **Reserved**

[7:0] **CYLHIGH**
Current value of the CDR5 when written. Can be read when CLR0.HOB=0. Contains LBA [23:16] bits.

[7:0] **CYLHIGH_EXP**
Previous value of the CDR5 pushed from the Cylhigh every time CDR5 is written. Can be read when CLR0.HOB=1. Contains LBA [47:40] bits. Used for 48-bit addressing.

SATA_CDR6 Device/head



Address: *SATABaseAddress* + 0x818

Type: R/W

Reset: 0xEF

Description: This read/write 8-bit register selects the device and contains command-dependent information.

[31:7] **Reserved**

[6] **LBA**: Logical Block Addressing:
0: CHS Mode
1: LBA Mode

[5] **Reserved**

[4] **DEV**: Device select
0: Device 0 (Master)
1: Device 1 (Slave)
This bit should always be cleared (Dev=0) since SATA host implements Master-only emulation. If this bit is set, any access to Command or Control register is ignored.
NOTE: this bit is not updated when Register FIS is received from the device, i.e. device can not change the state of this bit.
This bit is cleared by one of the following conditions:
- power-up;
- SControl[0]=1 (COMRESET);
- device signals COMINIT;
- SRST bit set in the Device Control register;
- EXECUTE DEVICE DIAGNOSTIC command written to the Command register.

[3:0] **HEAD**: Head number or other command-dependent information.

SATA_CDR7 Status/command

	31:8	7	6	5	4	3	2	1	0
Read	Reserved	BSY	DRDY	DWF	SERV	DRQ	Reserved		ERR
		STATUS							
Write	Reserved	COMMAND							

Address: *SATABaseAddress* + 0x

Type: RO (status)/WO (command)

Reset: 0x7F[1] (status)/0x00 (command)

Description: This location is used as one of the following:

- Command register when written,
- Status register when read.

[31:8] **Reserved**

[7:0] **COMMAND** (write)
Contains command code for device to execute. A write to this register sets the BSY bit in the Status register (BSY=1). This register is written last to initiate command execution.
Command Register FIS is sent to the device every time this register is written and both BSY and DRQ bits of the Status register are cleared.

[7:0] **STATUS** (read)
Status is a read-only 8-bit register is read-only and can be written only by the device with either the Register or the Set Device Bits FIS. Read access to the Status register clears the IPF, which negates the ATA interrupt request signal (intrq) to the system bus. Reset occurs on power-on.

[7] **BSY**: Busy
Indicates the device is busy when set. All other Command Block registers are invalid. When BSY and DRQ bits are cleared by the device, the host software can access any of the Command Block registers. This bit is cleared on power-on and set in the following cases:
- Device presence is detected via PHY READY
- signal assertion;
- SControl[0]=1 (COMRESET condition);
-Device signals COMINIT;
-A new command is written to the Command
-register when BSY=0;
-DEVICE RESET command is written to the
-Command register;
-SRST bit is set in the Device Control register.

[6] **DRDY**: Device ready
Device is ready when high and is able to execute a command.
Stays high once set unless catastrophic error.

[5] **DWF**: Drive write fault (command dependent in AT/ATAPI-4)

[4] **SERV**: Service
Used in overlap and queued commands (command dependent in AT/ATAPI-4).

[3] **DRQ**: Data Request
Asserted when device is ready to transfer data.

[2:1] **Reserved**

[0] **ERR**: Error when high (ERROR register contains further information).

1. Set to 0x7F on power-up, then 0x80 when device presence is detected via PHY READY condition.

SATA_CLR0 Alternative status/device control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								HOB	Reserved				SRST	NIEN	Reserved

Address: *SATABaseAddress* + 0x820

Type: RO (alt status)/WO (device control)

Reset: 0x7F[1] (alt status)/0x00 (device control)

Description: This location is used as one of the following:

- device control register when written,
- alternative status register when read.

Contains the same value as the Status register (CDR7), except the IPF is not cleared when the CLR0 is read.

[31:8] **Reserved**

[7] **HOB**
Used to read expanded registers (CDR2 - CDR5) when set. For example:
0: CDR2 Seccnt register is read. 1: CDR2 Seccnt_exp register is read

[6:3] **Reserved**

[2] **SRST**: Soft reset
0: Device is not reset 1: Device is reset
Control register FIS is transmitted to the device every time the state of this bit is changed.

[1] **NIEN**: Interrupt enable
0: Enable (intrq is asserted if IPF=1);
1: Disable (intrq is negated).

[0] **Reserved**

1. Set to 0x7F on power-up, then 0x80 when device presence is detected via PHY READY condition.

81.2.2 SATA

SATA_SCR0 SStatus

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IPM				SPD				DET			

Address: *SATABaseAddress* + 0x824

Type: RO

Reset: 0x0000 0000

Description: Contains the current state of the interface and host adapter. Updated continuously and asynchronously by the host adapter. Writes to this register result in bus error response. Resets on power-on.

[31:12] **Reserved**

[11:8] **IPM**: Current interface owner management state
0000: Device not present or communication not established
0001: Interface in active state
0010: Interface in PARTIAL power management state
0110: Interface in SLUMBER power management state
All other values: Reserved.

[7:4] **SPD**: Negotiated interface communication speed established
0000: No negotiated speed (device not present or communication not established)
0001: Generation 1 communication rate negotiated
0010: Generation 2 communication rate negotiated
All other values: Reserved.

[3:0] **DET**: Interface device detection and Phy state
0000: No device detected and PHY communication is not established
0001: Device presence detected but PHY communication not established (PHY COMWAKE signal is detected).
0011: Device presence detected and PHY communication established (PHY READY signal is detected).
0100: Phy in offline mode as a result of the interface being disabled or running in a BIST loopback mode
All other values: Reserved.

SATA_SCR1 SError

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DIAG_A	DIAG_X	DIAG_F	DIAG_T	DIAG_S	DIAG_H	DIAG_C	DIAG_D	DIAG_B	DIAG_W	DIAG_I	DIAG_N	Reserved				ERR_E	ERR_P	ERR_C	ERR_T	Reserved						ERR_M	ERR_I

Address: *SATABaseAddress* + 0x828

Type: R/W

Reset: 0x0000 0000

Description: This register contains supplemental SATA interface error information to complement the error information available in the Shadow Error register. The register represents all the detected errors accumulated since the last time the SError register was cleared. See “Transport Layer” chapter for more details. The set bits in the SError register indicate that the corresponding error condition became true one or more times since the last time this bit was cleared. The set bits in the SError register are explicitly cleared by a write operation to the register, or a reset operation (power-on or COMRESET). The value written to clear the set error bits should have ones encoded in the bit positions corresponding to the bits that are to be cleared.

Description:

[31:28] **Reserved**

[27] **DIAG_A**: Port Selector Presence detected
This bit is set when the Phy PORTSELECT signal is detected.

[26] **DIAG_X**: Exchanged error
This bit is set when the Phy COMINIT signal is detected.

[25] **DIAG_F**: Unrecognized FIS type
This bit is set when the Transport Layer receives a FIS with good CRC, but unrecognized FIS type.

[24] **DIAG_T**: Transport state transition error
This bit is set when the Transport Layer detects one of the following conditions:
- Wrong sequence of received FISes,
- PIO count mismatch between the PIO Setup FIS and the following Data FIS,
- Odd PIO/DMA byte count or DMA buffer offset,
- Wrong non-data FIS length (Received Data FIS length is not checked),
- RxFIFO overrun as a result of the 20 dword latency violation by the device.

[23] **DIAG_S**: Link sequence (illegal transition) error
This bit is set when the Link Layer detects an erroneous Link state machine transition.

[22] **DIAG_H**: Handshake error
This bit is set when the Link Layer
receives one or more R_ERRp handshake responses
form the device after frame transmission.

[21] **DIAG_C**: CRC error
This bit is set when the Link Layer detects CRC error in the received frame.

[20] **DIAG_D**: Disparity error
This bit is set when the Link Layer detects incorrect disparity in the received data or primitives.

[19] **DIAG_B**: 10b to 8b decoder error
This bit is set when the Link Layer detects 10b to 8b decoder error in the received data or primitives.

[18] **DIAG_W**: Comm Wake
This bit is set when Phy COMWAKE signal is detected.

[17] **DIAG_I**: Phy internal error
This bit is set when Phy internal error condition is detected.

[16] **DIAG_N**: PhyRdy change
This bit is set when Phy READY signal state is changed.

[15:12] **Reserved**

[11] **ERR_E**: Internal host adapter error
This bit is set when ERROR response is generated on the AHB bus, or dma_finish_tx is asserted, but no data had been written during DMA write operation.

[10] **ERR_P**: Protocol error
This bit is set when any of the following error conditions is detected: Transport state transition error (DIAG_T), Unrecognized FIS type (DIAG_F), or Link sequence error (DIAG_S).

[9] **ERR_C**: Non-recovered persistent communication or data integrity error
This bit is set when PHY READY signal is negated (Phy Not Ready condition) due to the loss of communication with the device or problems with interface, but not after the transition from active to PARTIAL or SLUMBER power management state.

[8] **ERR_T**: Non-recovered transient data integrity error
This bit is set if any of the 10b to 8b decoder error (DIAG_B), CRC error (DIAG_C), Disparity error (DIAG_D), Handshake error (DIAG_H), or Transport transition error (DIAG_T) is detected in the Data FIS, or non-data FIS transmission was not recovered after 3 retries.

[7:2] **Reserved**

[1] **ERR_M**: Recovered communication error
This bit is set when PHY READY condition is detected after interface initialization, but not after transition from PARTIAL or SLUMBER power management state to active state.

[0] **ERR_I**: Recovered data integrity error
This bit is set when nondata FIS transmission was unsuccessful (i.e. DIAG_H error was detected), but was recovered after 1 to 3 retries.

SATA_SCR2 SControl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PMP				SPM				IPM				SPD				DET			

Address: *SATABaseAddress* + 0x82C

Type: R/W

Reset: 0x0000 0000

Description: Provides control for the SATA interface. Write operations to the SControl register result in an action being taken by the host adapter or interface. Read operations from the register return the last value written to it.

[31:20] **Reserved**

[19:16] **PMP**: Port Multiplier Port
Represents the 4-bit value that will be placed in the PM Port field of all transmitted FISes.

[15:12] **SPM**: Select power management
Selects a power management state. A non-zero value written to this field will cause the power management state specified to be initiated. A value written to this field is treated as a one-shot. It is read as 0000b.
0000b - No power management state transition requested.
0001b - Transition to PARTIAL power management state initiated.
0010b - Transition to SLUMBER power management state initiated.
0100b - Transition to the active power management state initiated.
All other values reserved.

[11:8] **IPM**

This field represents the enabled interface power management states that can be invoked with the SATA interface power management capabilities:

0000b - No interface power management state restrictions
0001b - Transitions to the PARTIAL state disabled
0010b - Transitions to the SLUMBER state disabled
0011b - Transitions to both the PARTIAL and SLUMBER states disabled
All other values reserved.

[7:4] **SPD**

This field represents the highest-allowed communication speed that the interface is allowed to negotiate when the speed is established:

0000b - No speed negotiation restrictions
0001b - Limit speed negotiation to a rate not greater than Generation 1 communication rate
0010b - Limit speed negotiation to a rate not greater than Generation 2 communication rate
All other values reserved.

[3:0] **DET**

This field controls the host adapter device detection and interface initialization:

0000b - No device detection or initialization action requested

0001b - Perform interface communication initialization sequence to establish communication. This is functionally equivalent to a hard reset and results in the interface being reset and communication reinitialized (COMRESET condition). Upon a write to the SControl register that sets DET to 0001b, the host interface shall transition to a HP1:HR_Reset state and shall remain in that state until DET field bit 0 is cleared by a subsequent write to the SControl register.

0100b - Disable SATA interface and put Phy in offline mode.
All other values reserved.

SATA_SCR3 SActive

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCR3																															

Address: *SATABaseAddress* + 0x830

Type: R/W

Reset: 0x0000 0000

Description: Used for native SATA command queuing; contains the information returned in the SActive field of the set device bits FIS.

The host software can set bits in the SActive register by a write operation to this register. The value written to the set bits should have ones encoded in the bit positions corresponding to the bits that are to be set. Bits in the SActive register can not be cleared as a result of a register write operation by the host, and host software cannot clear bits in the SActive register.

Set bits in the SActive register are cleared as a result of data returned by the device in the SActive field of the Set Device Bits FIS. The value returned in this field will have ones encoded in the bit positions corresponding to the bits that are to be cleared. The device cannot set bits in this register.

All bits in the SActive register are cleared upon issuing a hard reset (COMRESET) signal or as a result of issuing a software reset by setting SRST bit of the Device Control register.

For the native command queuing protocol, the SActive value represents the set of outstanding queued commands that have not completed successfully yet. The value is bit-significant and each bit position represents the status of a pending queued command with a corresponding TAG value.

SATA_SCR4 SNotification

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																NOTIFY															

Address: *SATABaseAddress* + 0x834

Type: R/W

Reset: 0x0000 0000

Description: Used to notify host software which devices have sent a set device bits FIS with notification bit set. When a set device bits FIS with notification bit set to 1 is received, the bit corresponding to the value of the PM port field in the FIS is set, and an interrupt is generated if the I bit in the FIS is set and interrupt is enabled.

Set bits in the SNotification register are explicitly cleared by a write operation to the SNotification register, or a power-on reset. The register is not cleared due to a COMRESET condition. The value written to clear set bits should have ones encoded in the bit positions corresponding to the bits that are to be cleared.

[31:16] **Reserved**

[15:0] **NOTIFY**
This field represents whether a particular device with the corresponding PM Port number has sent a Set Device Bits FIS to the host with the Notification bit set.

81.2.3 SATA host

SATA_FPTAGR First party DMA tag

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											TAG					

Address: *SATABaseAddress* + 0x864

Type: RO

Reset: 0x0000 0000

Description: Contains the command 5-bit TAG value, which is updated every time a new DMA Setup FIS is received. The DMA controller uses this value to identify the buffer region in the host system memory selected for the data transfer. Write access to this location results in the bus error response.

[31:5] **Reserved**

[4:0] **TAG:** First-party DMA TAG value
Updated every time a new DMA Setup FIS is received from the device.

SATA_FPBOR First party DMA buffer offset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPBOR																															

Address: *SATABaseAddress* + 0x868

Type: RO

Reset: 0x0000 0000

Description: Contains the DMA buffer offset value, which is updated every time a new DMA Setup FIS is received. The device uses the offset to transfer DMA data out of order. Bits 1 and 0 should always be cleared (32-bit-aligned offset). A write access to this location results in the bus error response.

SATA_FPTCR First party DMA transfer count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPTCR																															

Address: *SATABaseAddress* + 0x86C

Type: RO

Reset: 0x0000 0000

Description: Contains the number of bytes that will be transferred. It is updated every time a new DMA Setup FIS is received. Bit 0 should always be cleared (even number of bytes). A write access to this location results in the bus error response.

SATA_DMACR DMA control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														RXCHEN	TXCHEN

Address: *SATABaseAddress* + 0x870

Type: R/W

Reset: 0x0000 0000

Description: Status of the DMA transmit or receive channel. Application software must set either of these bits prior to issuing a corresponding DMA command to the device. Power-on or COMRESET condition clears this register.

[31:2] **Reserved**

[1] **RXCHEN**
0: DMA receive channel is disabled
1: DMA receive channel is enabled and ready for transfer

[0] **TXCHEN**
0: DMA transmit channel is disabled
1: DMA transmit channel is enabled and ready for transfer

SATA_DBTSR DMA burst tranaction size

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														MRD		Reserved														MWR	

Address: *SATABaseAddress* + 0x874

Type: R/W

Reset: 0x0008 0010

Description: Used to set RxFIFO “pop almost empty” and TxFIFO “push almost full” thresholds to the burst transaction size (in dwords) for a DMA read or write operation. The SATA host generates corresponding request signals (DMA_REQ_RX or DMA_REQ_TX) to the DMA controller as follows:

- DMA_REQ_RX is asserted when RxFIFO contains enough data for the burst transaction of MRD size;

- DMA_REQ_TX is asserted when TxFIFO contains enough free space for the burst transaction of MWR size.

Power-up or COMRESET condition initializes this register to the value shown below.

The MRD/ MWR field can only be written if the corresponding RXCHEN/ TXCHEN bit of the DMACR register is cleared.

Note: 1 *The DMA burst transaction size must never exceed these values, otherwise an ERROR response is generated by the slave interface if either the RxFIFO empty or the TxFIFO full condition is detected during a DMA bus transfer. Host software must ensure that the DMA controller is programmed with the same values prior to enabling a channel for transfer.*

2 *For 16-bit DMA transfers, MRD and MWR values should be adjusted by dividing the burst size (number of beats in the burst) by 2 and rounding up if the value is odd. For example, if the read burst size is 7 words, then the MRD value should be 4.*

[31:m^a] **Reserved**

[(m-1):16] **MRD**
This field is used to set the RxFIFO “pop almost empty” flag to the maximum burst size in dwords for the DMA read operation. Can be written if DMACR.RXCHEN=0, otherwise, the write to this field is ignored.
Valid range: 1 to (RXFIFO_DEPTHraf^b-1)
Note: MRD=0 might result in a bus error response during DMA read transaction. Upper boundary is derived from the fact that the device will stop sending data when the host generates HOLDp raf dwords from the RxFIFO full condition. This might result in possible lock condition (dma_req_rx is never generated) if this value is exceeded.
Defaults to 8 dwords on reset (32-23-1=8).

[15:k^c] **Reserved**

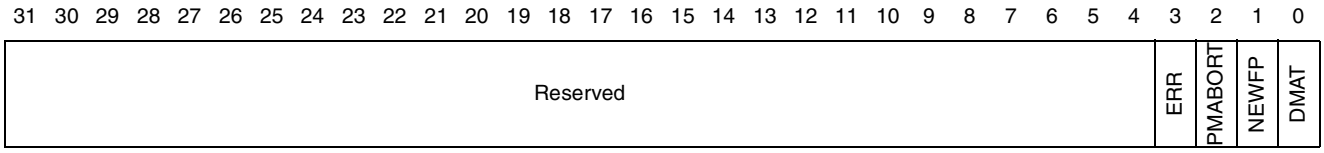
[(k-1:0] **MWR**
This field is used to set the TxFIFO “push almost full” flag to the maximum burst size in dwords for the DMA write operation. Can be written if DMACR.TXCHEN=0, otherwise, the write to this field is ignored.
Valid range: 1 to (TXFIFO_DEPTH1)
Note: MWR=0 might result in bus error response during DMA write transaction. Upper boundary is determined by the TxFIFO address width.
Defaults to 16 dwords on reset (32/2=16).

a. m - Rx FIFO RAM address width ($m = \text{Log}_2$(RXFIFO_DEPTH))

b. raf = 23

c. k - Tx FIFO RAM address width ($k = \text{Log}_2$(TXFIFO_DEPTH))

SATA_INTPR Interrupt pending



Address: *SATABaseAddress* + 0x878

Type: R/W

Reset: 0x0000 0000

Description: Contains all SATA host interrupt events before masking. The bits are set by an interrupt event. All the interrupt bits together with the IPF ATA interrupt flag are ORed to generate the SATA host intrq output. The set bits in the INTPR register can be cleared by a write operation to the register, or a reset operation (power-on or COMRESET). The value written to clear set bits should have ones encoded in the bit positions corresponding to the bits that are to be cleared.

[31:4] **Reserved**

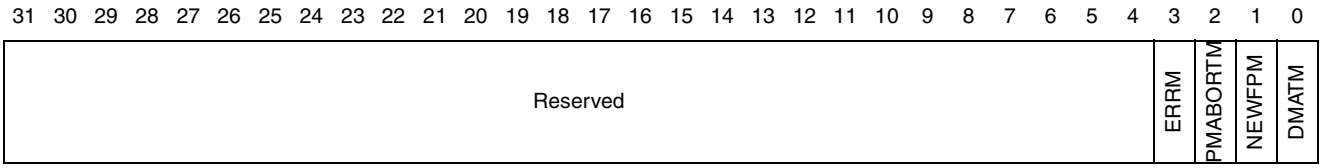
[3] **ERR**
Set when any of the bits in the SError register is set and the corresponding bit in the ERRMR register is set.

[2] **PMABORT**
Set when the link layer detects a power mode abort condition (power mode is aborted by the device requesting a frame transmission).
Note: this bit must be cleared explicitly by software before issuing a power management request to the interface.

[1] **NEWFP**
Set when a DMA Setup FIS is received from the device without errors.

[0] **DMAT**
Set when DMATp is received from the device during Data FIS transmission.

SATA_INTMR Interrupt mask



Address: *SATABaseAddress* + 0x87C

Type: R/W

Reset: 0x0000 0000

Description: Used to mask or enable corresponding interrupt events in the INTPR register. An interrupt event is masked if the bit is cleared and is enabled if set. If any of the INTPR bits are set or if IPF is set and enabled (unmasked), the INTRQ output is asserted. A COMRESET condition clears this register.

[31:4] **Reserved**

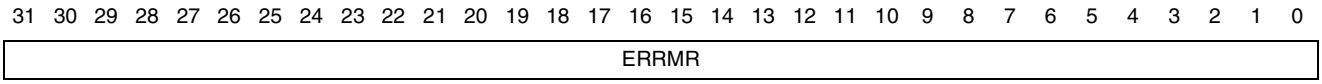
[3] **ERRM**
0: ERR interrupt is masked
1: ERR interrupt is enabled

[2] **PMABORTM**
0: PMABORT interrupt is masked
1: PMABORT interrupt is enabled

[1] **NEWFPM**
0: NEWFP interrupt is masked
1: NEWFP interrupt is enabled

[0] **DMATM**
0: DMAT interrupt is masked
1: DMAT interrupt is enabled

SATA_ERRMR Error mask



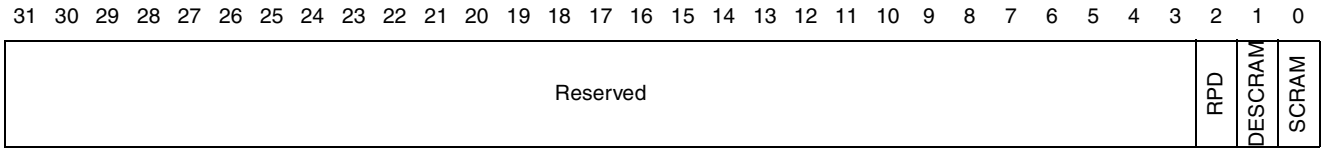
Address: *SATABaseAddress* + 0x880

Type: R/W

Reset: 0x0000 0000

Description: Used to mask or enable corresponding bits of the SError register prior to setting the ERR bit of the INTPR. This allows driver software to select the SError bits that can cause the interrupt output intrq to be asserted. The INTPR ERR bit is set if any of the SError bits are set and the corresponding ERRMR bit is set. Clearing the ERRMR bit would mask the corresponding SError bit from setting the INTRP ERR bit. COMRESET condition clears this register.

SATA_LLCR Link layer control



Address: *SATABaseAddress* + 0x884

Type: R/W

Reset: 0x0000 0007

Description: Provides Link Layer (LL) control capability for the host software. Power-on or COMRESET condition sets these bits.

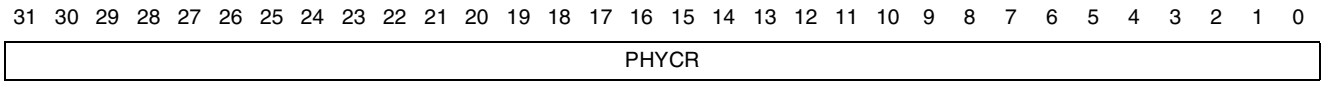
[31:3] **Reserved**

[2] **RPD**
0: Repeat primitive drop function disabled 1: Repeat primitive drop function enabled

[1] **DESCRAM**
0: Descrambler disabled 1: Descrambler enabled

[0] **SCRAM**
0: Scrambler disabled 1: Scrambler enabled

SATA_PHYCR PHY control



Address: *SATABaseAddress* + 0x888

Type: R/W

Reset: 0x0003 801C[1]

Description: Bits of this register are connected to the corresponding bits of the PHY_CONTROL output port. The width is set by the PHY_CTRL_W parameter (valid range: 0 to 32). The remaining bits are reserved: reads return zeros, writes have no effect. If the width is set to zero, then this location is reserved.

Note: This location supports only 16 or 32-bit transfer sizes for write accesses; 8- bit write accesses are ignored.

1. For cut 1.*x*, this value must be changed to 0x0013 704A for correct operation; however, for cut 2 onwards, the default value must not be changed.

SATA_PHYSR PHY status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYSR																															

Address: *SATABaseAddress* + 0x88C

Type: RO

Reset: 0x0000 0000

Description: Used to monitor Phy status. The bits of this register reflect the state of the corresponding bits of the phy_status input port. The width is set by the PHY_STAT_W parameter (valid range: 0 to 32). The remaining bits are reserved: reads return zeros, writes have no effect. If the width is set to zero, then this location is reserved.

Note: This location supports only 16 or 32-bit transfer sizes for write accesses; 8-bit write accesses are ignored.

SATA_VERSIONR SATA host version

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSIONR																															

Address: *SATABaseAddress* + 0x8F8

Type: RO

Reset: HSATA_VERSION_NUM, 0x3130 302A[1]

Description: Contains the hard-coded SATA host component version value set by the HSATA_VERSION_NUM parameter. The value represents an ASCII code of the version number. For example, version 1.00* is coded as 0x3130 302A. Writing to this register results in a bus error response.

SATA_IDR SATA host ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR																															

Address: *SATABaseAddress* + 0x

Type: RO

Reset: HSATA_ID_NUM, 0x100A 0208[2]

Description: Contains the hard-coded SATA host identification value set by the HSATA_ID_NUM parameter. Writing to this register results in a bus error response.

1. Fixed value which must not be changed.
2. Fixed value which must not be changed.

82 Smartcard interface

82.1 Overview

The smartcard interface supports asynchronous protocol smartcards as defined in the ISO7816-3 standard. Limited support for synchronous smartcards can be provided in software by using the PIO bits to provide the clock, reset, and I/O functions on the interface to the card. Two smartcard interfaces are supported on the STx7100. Each of these interfaces is able to automatically detect and removes power to the smartcard when smartcard removal is detected.

The UART function of the smartcard interface is provided by a UART (ASC). UART ASC0 can be used by smartcard0 and ASC1 can be used by smartcard1.

Each ASC used by a smartcard interface must be configured as eight data bits plus parity, 0.5 or 1.5 stop bits, with smartcard mode enabled. A 16-bit counter, the smartcard clock generator, divides down either the comms clock, or an external clock connected to a pin shared with a PIO bit, to provide the clock to the smartcard. PIO bits in conjunction with software are used to provide the rest of the functions required to interface to the smartcard. The inverse signalling convention, as defined in ISO7816-3, is handled in software, inverted data and most significant bit first. For more information, see Chapter 61: *Asynchronous serial controller (ASC)* *on page 774* and Chapter 25: *Programmable I/O (PIO) ports* *on page 216*.

82.2 External interface

The smartcard pin functions are described in the table below (which may be read in conjunction with the PIO assignment table given in Chapter 8: *Connections* *on page 59*.

Table 245: Smartcard interface pins

Pins	In/Out[a]	Function
SC0_CLKOUT (PIO0 [3]) SC1_CLKOUT (PIO1 [3])	Out, open drain for 5 V cards	Clock for smartcard
SC1_EXT_CLK (PIO1 [2])	In	Optional external clock input to smartcard clock divider #1
SC0_DATA_OUT (PIO0 [0]) SC1_DATA_OUT (PIO1 [0])	Out, open drain driver	Serial data output. Open drain drive
SC0_DATA_IN (PIO0 [1]) SC1_DATA_IN (PIO1 [1])	In	Serial data input
SC0_DIR (PIO0 [6]) SC1_DIR (PIO1 [6])	Out	Indicates if the smartcard is operating in serial data output (open drain drive) mode or serial data input mode.
SC_COND_VCC (PIO0 [5])	Out	Supply voltage enable/disable
SC_DETECT (PIO0 [7])	In	Smartcard detection

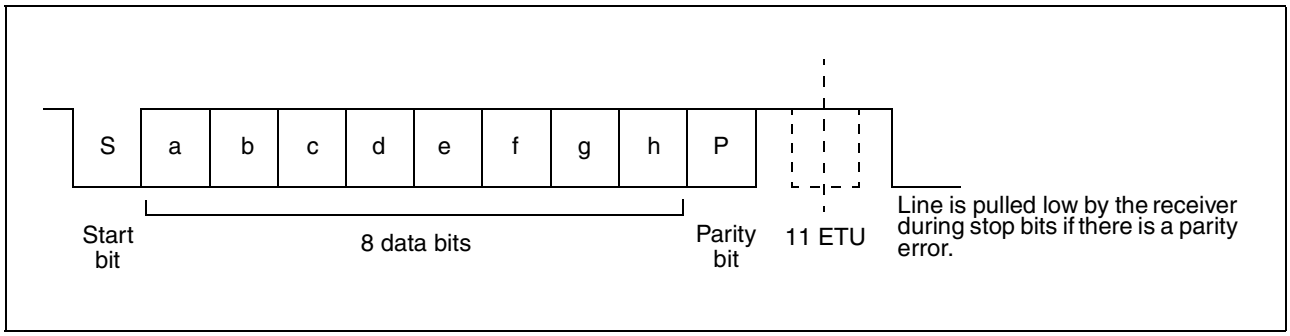
a. *The indicated directions are not set by default. They must be programmed in the PIO configuration registers, since smartcard interfaces are multiplexed as PIO alternate functions.*

All smartcard interface signals are provided by alternate functions of the PIO pins. The UART*n*_TXD data signal is connected to the SC*n*_DATA pin with the correct driver type and the clock generator is connected to the SC*n*_CLK pin.

The ISO standard defines the bit times for the asynchronous protocol in ETUs, which are related to the clock frequency received by the card. One bit time = one ETU.

The ASC transmitter output and receiver input must be connected together externally. For the transmission of data from the STx7100 to the smartcard, the ASC must be set up in smartcard mode.

Figure 327: ISO 7816-3 asynchronous protocol



82.3 Smartcard clock generator

The smartcard clock generator provides a clock signal to the smartcard. The smartcard uses this clock to derive the baudrate clock for the serial I/O between the smartcard and another UART. The clock is also used for the CPU in the card, if there is one present.

Operation of the smartcard interface requires that the clock rate to the card is adjusted while the CPU in the card is running code, so that the baudrate can be changed or the performance of the card can be increased. The protocols that govern the negotiation of these clock rates and the altering of the clock rate are detailed in the ISO7816-3 standard. The clock is used as the comms clock for the smartcard, so updates to the clock rate must be synchronized with the clock to the smartcard. This means the clock high or low pulse widths must not be shorter than either the old or new programmed value.

The source of the smartcard clock generator (that is, glitch free divider) can be set to the 100 MHz system clock or, alternatively, an external reference. For the first smartcard interface this alternate source is a dedicated programmable frequency synthesizer. Refer to Chapter 15: *Clocks* *on page 120* for frequency programming details. For the second smartcard interface this alternate source is SC1_EXTCLKIN on PIO1[2].

Two registers control the period of the generated clock and the running of the clock.

- The SCI_n_CLKVAL register determines the smartcard clock frequency. The value given in the register is multiplied by two to give the division factor of the input clock frequency. The divider is updated with the new value for the divider ratio on the next rising or falling edge of the output clock.
- The SCI_n_CLKCON register controls the source of the clock and determines whether the smartcard clock output is enabled. The programmable divider and the output are reset when the ENABLE bit is set to 0.

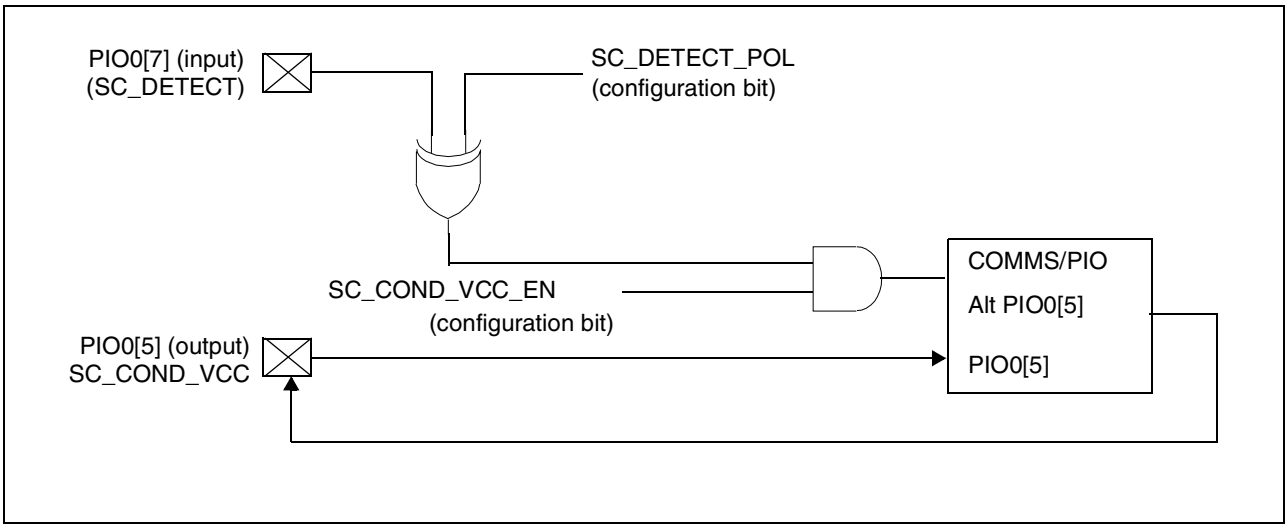
82.4 Smartcard removal and power control

A smartcard VCC control signal (‘SC_COND_VCC’ alternate out on PIO0) may be directly generated from the smartcard detect input (‘SC_DETECT’ alternate input on PIO0) under configuration bit control. The figure below illustrates the required functionality. It is still possible to read by software PIO bit 7 (SC_DETECT) like a normal PIO at the same time. The polarity of the detect is also programmable. This functionality is selected by two configuration bits part of the top level configuration registers, see Chapter 21: *System configuration registers* *on page 171*.

In this way, SC_COND_VCC can be directly generated from the interrupt input used as smartcard detect, so that power is automatically shut off when smartcard removal is detected.

A single smartcard detect input and a single conditional VCC control signal are provided. These signals can be used in conjunction with either the first or the second smartcard interface.

Figure 328: Smartcard VCC control



83 Smartcard interface registers

Register addresses are provided as *SmartcardnBaseAddress* + offset.

The *SmartcardnBaseAddresses* are:

SCG0: 0x1804 8000,

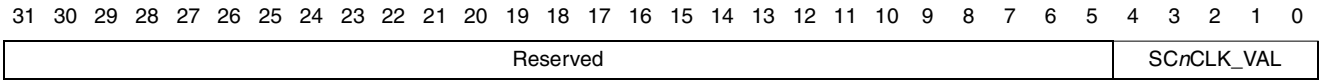
SCG1: 0x1804 9000.

See also Chapter 21: *System configuration registers on page 171*.

Table 246: Smartcard register summary

Register	Description	Offset	Type
SCI_n_CLK_VAL	Smartcard n clock	0x00	R/W
SCI_n_CLK_CTRL	Smartcard n clock control	0x04	R/W

SCI_n_CLK_VAL Smartcard n clock



Address: *SmartcardnBaseAddress* + 0x00

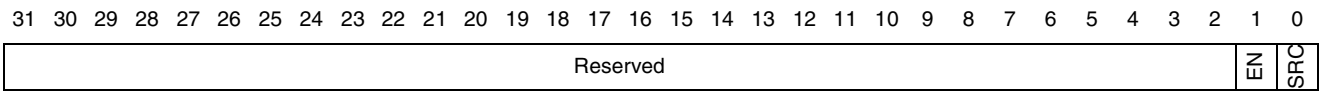
Type: R/W

Reset: 0

Description: This register determines the smartcard clock frequency. The 5-bit value given in the register is multiplied by 2 to give the division factor of the input clock frequency. For example, if SCnCLKVAL= 8 then the input clock frequency is divided by 16. The value “zero” must not be written into this register.

The divider is updated with the new value for the divider ratio on the next rising or falling edge of the output clock.

SCI_n_CLK_CTRL Smartcard n clock control



Address: *SmartcardnBaseAddress* + 0x04

Type: R/W

Reset: 0

Description: This register controls the source of the clock and determines whether the smartcard clock output is enabled. The programmable divider and the output are reset when bit ENABLE is set to 0.

[31:2] **Reserved**
Write 0.

[1] **EN:** Smartcard clock generator enable bit
0: Stop clock, set output low and reset divider 1: Enable clock generator

[0] **SOURCE:** Selects source of smartcard clock
0: Selects global clock 1: Selects external pin

84 Test access port (TAP)

The STx7100 TAP conforms to IEEE standard 1149.1, and includes device ID information. Pins are listed in Table 247. TCK can be stopped in either logic state.

Table 247: TAP pins

Pin	In/out	Pull up/down	Description
TDI	In	Up	Test data input
TDO	Out		Test data output
TMS	In	Up	Test mode select
TCK	In	Up	Test clock
TRST	In	Up	Test logic reset

The instruction register is five bits long, with no parity. The pattern 0 0001 is loaded into the register during the capture-IR state.

There are four defined public instructions, see Table 248. All other instruction codes are reserved.

Table 248: Instruction codes

Instruction code [a]					Instruction	Selected register
0	0	0	0	0	**extest**	BOUNDARY_SCAN
0	0	0	1	0	**idcode**	IDENTIFICATION
0	0	0	1	1	**sample/preload**	BOUNDARY_SCAN
1	1	1	1	1	**bypass**	BYPASS

a. MSB... LSB; LSB closest to TDO.

There are three test data registers; BYPASS, BOUNDARY_SCAN and IDENTIFICATION. These registers operate according to IEEE 1149.1. The operation of the BOUNDARY_SCAN register is defined in the BSDL description.

The identification code is 0x*n*0D4 4041, where *n* is the four-bit silicon revision number.

Table 249: Identification code

Bit 31							Bit 0 [a]
Mask rev	**Division code**		**Device code**		**STMicroelectronics manufacturers ID**		**b**
n	D	4	2	4	0	4	1

a. Closest to TDO

b. Defined as 1 in IEEE 1149.1 standard

See also Chapter 21: *System configuration registers on page 171*.

85 USB 2.0 host (USBH)

85.1 Overview

This chapter describes the functional operation of the universal serial bus host (USBH) interface module.

The interface works with an embedded microcore. It is compliant with both the EHCI and OHCI (USB 2.0 and USB 1.1) bus control standards, supporting low, full and high speed, isochronous, bulk, interrupt and control transfers. It supports up to sixteen endpoints and initiates DMA transactions on the STBus to access memory.

There are four main areas of a USB system:

- the client software and USB driver,
- host controller driver (HCD),
- host controller (HC),
- USB device.

The client software, USB driver and host controller driver are implemented in software. The host controller and USB device are implemented in hardware.

The USBH provides the features listed below.

- **USB 2.0 EHCI host controller**
  - Compliant with EHCI and USB 2.0 specifications (see Section 85.1.1).
  - High speed (480 Mbit/s) transmissions.
  - Microframe caching.
  - USB 2.0 PING and SPLIT transactions.
  - Power management capabilities.
  - Port router interfaces to USB 1.1 host controller.
- **USB 1.1 OHCI host controller**
  - Compliant with OHCI specification (see Section 85.1.1).
  - Low speed (1.5 Mbit/s) and full speed (12 Mbit/s) transmissions.
  - Power management capabilities.

PCB Layout recommendations for USB2.0 signals are given in Chapter 14: *PCB layout recommendations* *on page 108*.

85.1.1 References

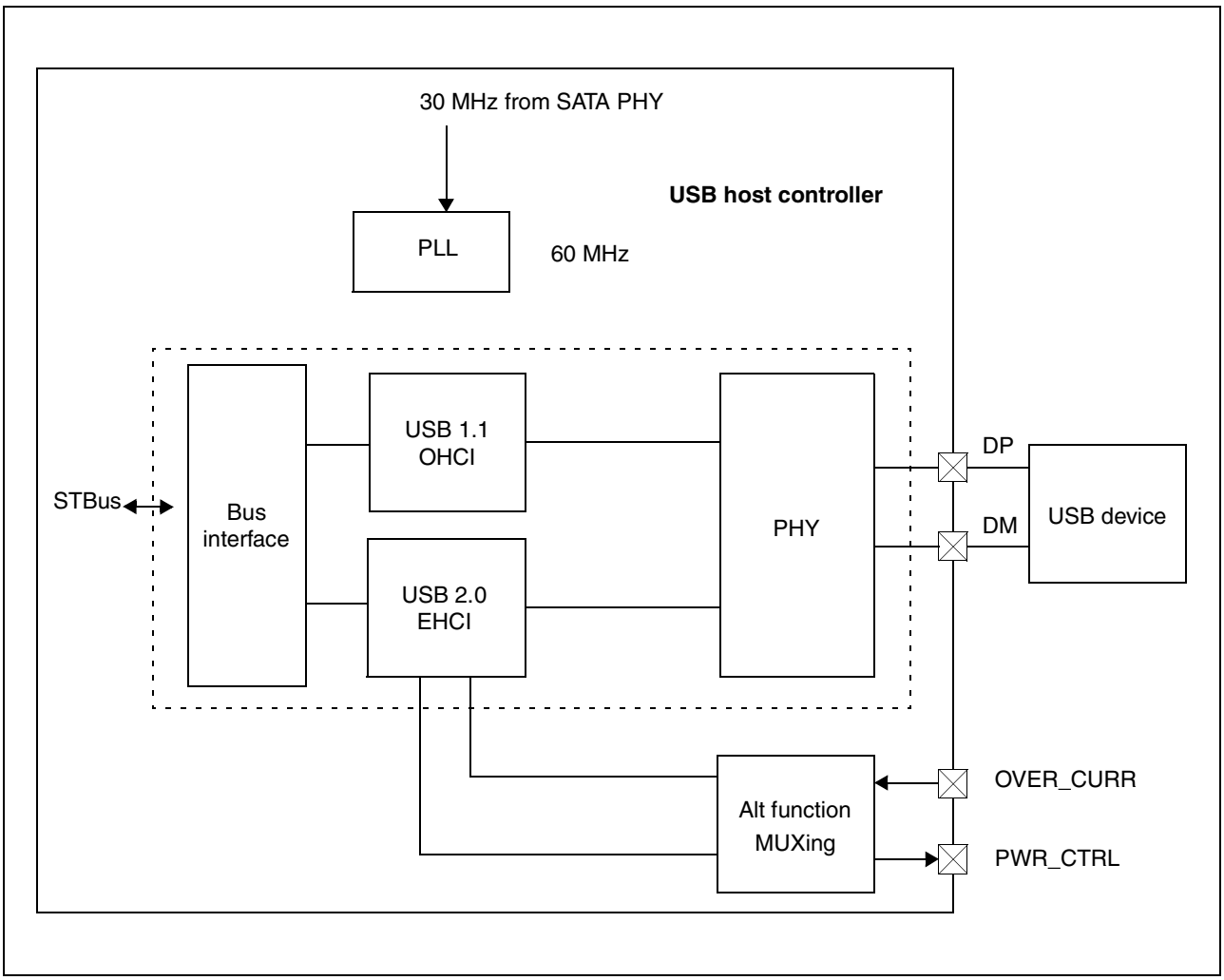
For further details of USB functionality the references below can be downloaded from www.usb.org.

- *Universal Serial Bus Specification, revision 2.0,*
- *Enhanced Host Controller Interface Specification for Universal Serial Bus, revision 1.0,*
- *Universal Serial Bus Specification, revision 1.1,*
- *OpenHCI Open Host Controller Interface Specification for USB, revision 1.0a.*

85.2 Operation

The USB connects devices with the host. The USB 2.0 host controller includes one high-speed mode host controller and one USB 1.1 host controller (see Figure 329). The high-speed host controller implements an EHCI interface. It is used for all high-speed communications to high-speed mode devices connected to the root ports of the USB 2.0 host controller. This allows communications to full- and low-speed devices connected to the root ports of the USB 2.0 host controller to be provided by the companion USB 1.1 host controller.

Figure 329: USB host controller



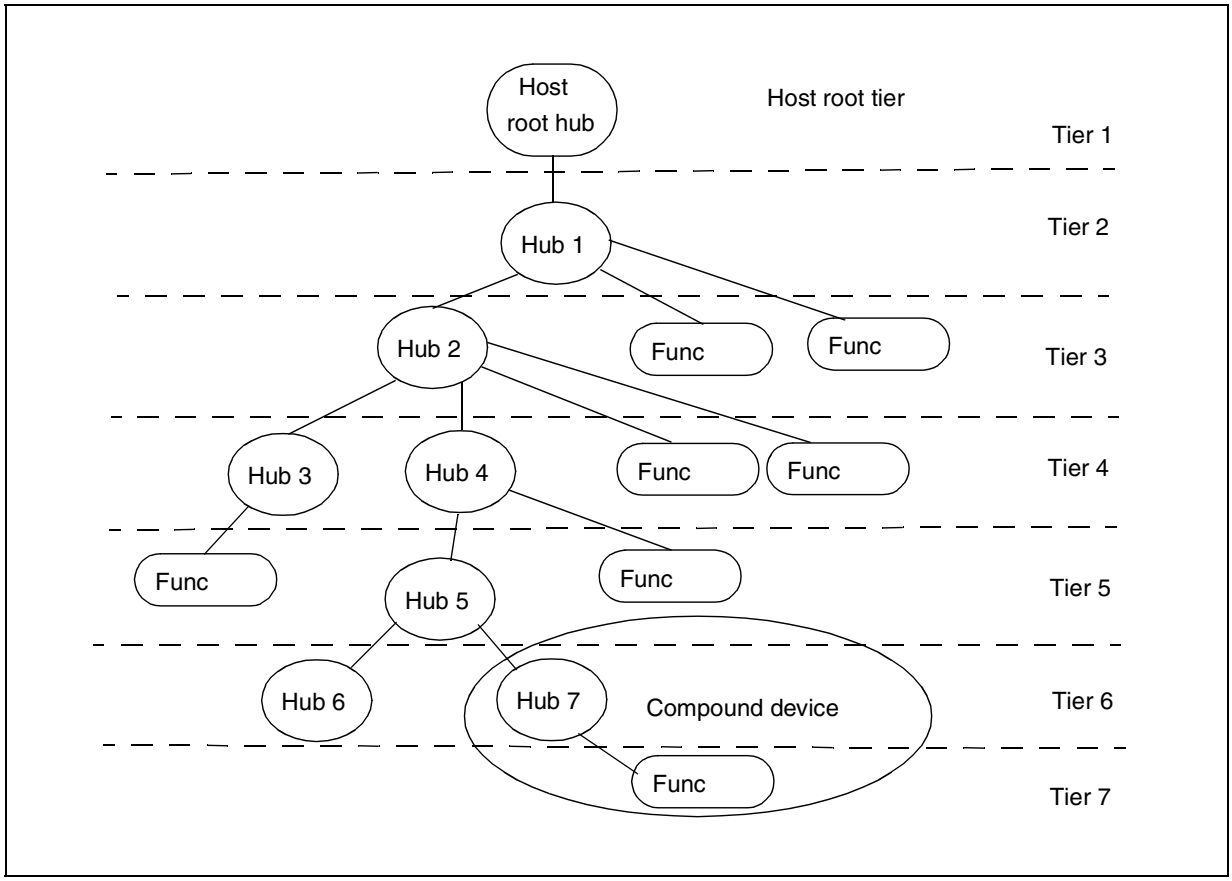
The USB 1.1 host controller has no knowledge of the high-speed mode host controller. High-speed devices are routed to and controlled by the USB 2.0 host controller. When running and configured, the USB 2.0 HC is the default owner of all the root ports. The USB 2.0 HC and its driver initially detect all devices attached. If the attached device is not a high-speed device, the USB 2.0 HC driver releases ownership of the port and control of the device to a companion host controller. For that port, enumeration starts over from the initial attach detect point and the device is enumerated under the USB 1.1 HC. Otherwise, the USB 2.0 HC retains ownership of the port and the device completes enumeration under the USB 2.0 HC.

85.3 Bus topology

The USB physical interconnect has a tiered star topology. A hub is at the center of each star. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or function.The system can support up to 127 nodes, although a performance limitation must be accepted as the depth of the tier hierarchy increases

In USB 2.0, the maximum number of tiers allowed is seven (including the root tier). Note that in seven tiers, five nonroot hubs maximum can be supported in a communication path between the host and any device. A compound device occupies two tiers. Therefore, it cannot be enabled if attached at tier level seven. Only functions can be enabled in tier seven bus protocol.

Figure 330: Bus topology



The USB is a polled bus and the host controller initiates all data transfers. All bus transactions on the USB 1.1 and most bus transactions on the USB 2.0 involve the transmission of up to three packets. Bus transactions of four packets manage data transfers between the USB 2.0 host and full- or low-speed devices.

Each transaction begins when the host controller, on a scheduled basis, sends a USB packet describing:

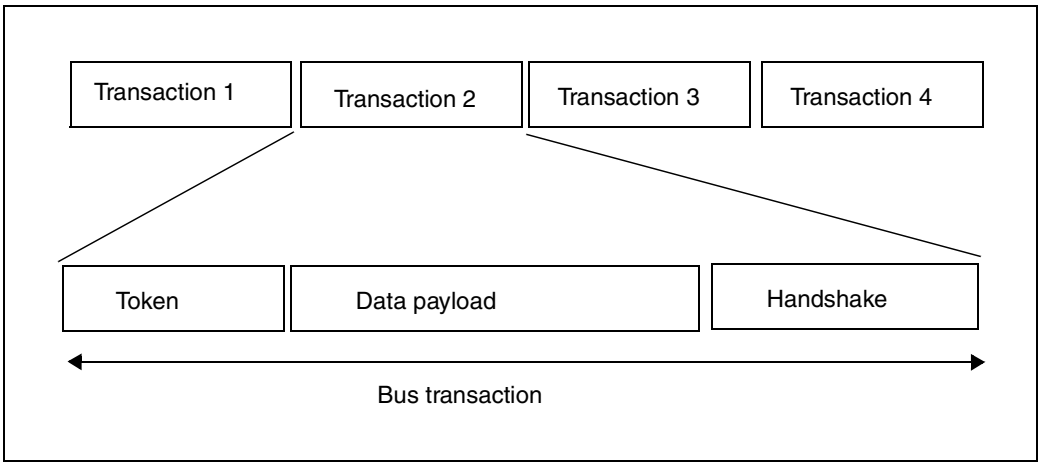
- the type and direction of transaction,
- the USB device address,
- endpoint number.

This packet is referred to as the token packet.

The USB device that is addressed selects itself by decoding the appropriate address fields. In a given transaction, data is transferred either from the host to a device or from a device to the host. The direction of data transfer is specified in the token packet. The source of the transaction then sends a data packet or indicates it has no data to transfer. The destination, in general, responds with a handshake packet indicating whether the transfer was successful.

This is the template for all data transfer over the USB, regardless of the data type. Figure 331 describes the process.

Figure 331: USB packet description



USB protocol generation is handled in the HC by various state machines. The host controller driver (HCD) provides the specific instructions for the type of data to send and the target address. The HC provides all the protocol layer application of packets when sending data to the peripheral, and also the stripping off of protocol packets when receiving data from the peripheral.

The major functional blocks of the USB system are:

- client software and USB device driver (application software driver),
- host controller driver (HC software driver),
- host controller interface (bus wrapper layer between HCD and HC),
- host controller,
- USB device (peripheral application).

The host controller driver and host controller work in tandem to transfer data between client software and a USB device. Data is translated between shared-memory data structures at the client software end to USB signal protocols at the USB device end.

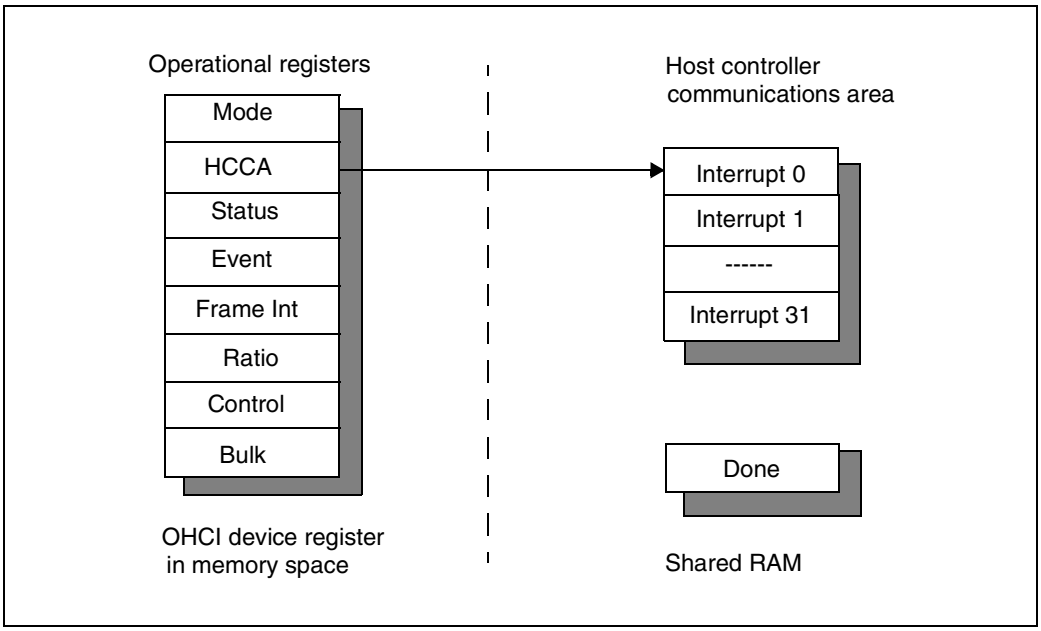
85.4 Data transfers

The USB 2.0 host manages all interrupt, bulk and control transfer types using a simple buffer queue. The queue provides automatic, in-order streaming of data transfers. Software can asynchronously add data buffers to a queue and maintain streaming. USB-defined short packet semantics are fully supported on all processing boundary conditions without software intervention.

Split transactions for USB 1.1 full- and low-speed non-isochronous endpoints are managed with the same registers. The split transaction protocol is managed as a simple extension to the high-speed execution model.

High-speed and full-speed isochronous transfers are managed using dedicated (and different) registers. These data structures are optimized for variability of the data payload and time-oriented characteristics of the isochronous transfer type.

Figure 332: USB 1.1 communications channels



86 USB 2.0 host (USBH) registers

There are two sets of registers implemented in the in the USB 2.0 host: OHCI (USB 1.1) and EHCI (USB 2).

Register addresses are provided as either

OHCIBaseAddress + offset,

EHCIBaseAddress + offset, or

The *OHCIBufferBaseAddress* is:

0x191F FC00.

The *EHCIBaseAddress* is:

0x191F FE00.

Registers in Table 250: *USB 1.1 host register summary* are detailed in the *OpenHCI Open Host Controller Interface Specification for USB*, revis*ion 1.0a*.

Registers in Table 251: *USB 2.0 host register summary* are detailed in the *Enhanced Host Controller Interface Specification for Universal Serial Bus, revision 1.0*.

There are a few additional USB registers described in Chapter 21: *System configuration registers* *on page 171*.

All registers are 32-bit.

Table 250: USB 1.1 host register summary

Register	Description	Offset	Type (HCD)	Type (HC)
OHCI_HC_REV	Version number of HC	0x00	RO	RO
OHCI_HC_CTRL	Operating modes for HC	0x04	R/W	R/W
OHCI_HC_CMD_STA	Shows status and receives commands from HCD	0x08	R/W	R/W
OHCI_HC_INT_STA	Status of events causing interrupts	0x0C	R/W	R/W
OHCI_HC_INT_EN	Enables interrupt generation for various events	0x10	R/W	RO
OHCI_HC_INT_DISABLE	Enables interrupt generation for various events	0x14	R/W	RO
OHCI_HC_HCCA	HC communication area	0x18	R/W	RO
OHCI_HC_PER_CURRENTED	Current endpoint descriptor	0x1C	RO	R/W
OHCI_HC_CTRL_HEADED	First endpoint descriptor of control list	0x20	R/W	RO
OHCI_HC_CTRL_CURRENTED	Current endpoint descriptor of control list	0x24	R/W	R/W
OHCI_HC_BULK_HEADED	First endpoint descriptor of bulk list	0x28	R/W	RO
OHCI_HC_BULK_CURRENTED	Current endpoint descriptor of bulk list	0x2C	R/W	R/W
OHCI_HC_DONE_HEAD	Last completed transfer descriptor	0x30	RO	R/W
OHCI_HC_FM_INTERVAL	Frame time interval	0x34	R/W	RO
OHCI_HC_FM_REMAINING	Time remaining in the frame	0x38	RO	R/W
OHCI_HC_FM_NUMBER	Frame number	0x3C	RO	R/W
OHCI_HC_PERIC_START	Earliest time HC can process periodic list	0x40	R/W	RO
OHCI_HC_LS_THOLD	Packet transfer threshold	0x44	R/W	RO
OHCI_HC_RHDESCRIPTORA	Description A of root hub	0x48	R/W	RO

Table 250: USB 1.1 host register summary

Register	Description	Offset	Type (HCD)	Type (HC)
OHCI_HC_RHDESCRIPTORB	Description B of root hub	0x4C	R/W	RO
OHCI_HC_RH_STA	Hub status/change	0x50	R/W, RO, WO	R/W, RO
OHCI_HC_RHPRT_STA_1	Control and report port events	0x54	R/W	R/W

Table 251: USB 2.0 host register summary

Register	Description	Offset	Type
EHCI_HCAPBASE	Capability register	0x00	RO
EHCI_HCSPARAMS	Host controller structural parameters	0x04	RO[a]
EHCI_HCCPARAMS	Host controller capability parameters	0x08	RO
EHCI_USBCMD	USB EHCI command	0x10	R/W, RO
EHCI_USBSTS	USB EHCI status	0x14	R/WC, RO
EHCI_USB_INT_EN	USB EHCI interrupt enable	0x18	R/W
EHCI_FRINDEX	USB EHCI frame index	0x1C	R/W
EHCI_CTRLDS_SEG	Control data structure segment	0x20	R/W
EHCI_PER_ICLISTBASE	Periodic frame list base address	0x24	R/W
EHCI_ASYNCLIST_ADDR	Next asynchronous list address	0x28	R/W
EHCI_CFG_FLAG	Configure flag register	0x50	R/WC, R/W, RO
EHCI_PORTSC_0	Port 0 status and control	0x54	R/W
EHCI_INSNREG00	Programmable microframe base value	0x90	R/W
EHCI_INSNREG01	Programmable packet buffer out/in threshold	0x94	R/W
EHCI_INSNREG02	Programmable packet buffer length (reserved)	0x98	R/W
EHCI_INSNREG03	Break memory transfer	0x9C	R/W
EHCI_INSNREG04	Debug - reserved	0xA0	R/W
EHCI_INSNREG05	UTMI control and status	0xA4	R/W

a. May be written when the WRT_RDONLY bit is set.

Part 9

End notes

87 Register index

88 Revision history

Reference	Change
Version A	July 05
	Initial version

89 Licenses

Supply of this product does not convey a license under the relevant intellectual property of the companies mentioned in this chapter nor imply any right to use this intellectual property in any finished end-user or ready to use final product. An independent license for such use is required and can be obtained by contacting the company or companies concerned.

Once the license is obtained, a copy must be sent to STMicroelectronics.

The details of all the features requiring licenses are not provided within the datasheet and register manual. They are provided only after a copy of the license has been received by STMicroelectronics.

The features requiring licenses are:

Dolby® Digital EX, Pro Logic® II, MLP Lossless™

Dolby Digital, Pro Logic and MLP Lossless are intellectual properties of Dolby Labs. The Dolby Digital, Pro Logic or MLP Lossless license allows the use of the corresponding decoder embedded in the STx7100.

Two types of license exist: S license must be obtained for samples (up to 25 units). P license must be obtained for production.

For all details, contact Dolby Labs at:
Dolby Labs, 100 Potrero Avenue, San Francisco, CA 94103, USA
www.dolby.com

CSS

CSS DVD Copy Protection is intellectual property of Matsushita Electronics Industrial Co. The CSS DVD Copy Protection license allows the use of the CSS decryption cell embedded in the STx7100.

For all details, contact Matsushita at:
Matsushita Electronics Industrial Co. LTD, CSS Interim License Organization, 1006 Kadoma, Kadoma-Shi, Osaka 571-8503 JAPAN

Macrovision®

Macrovision Anti-Copy System for DVD is intellectual property of Macrovision Corporation. The Macrovision license allows the use of the Macrovision feature embedded in STx7100.

For all details, contact Macrovision at:
Macrovision Corp., 1341 Orlean Drive, Sunnyvale, CA 94089 USA
www.macrovision.com

TruSurround XT™, CircleSurround™ II

The CircleSurround II, TruSurround XT and SRS technology rights incorporated in this chip are owned by SRS Labs, a U.S. corporation and licensed to ST Microelectronics. Purchasers of this chip, who use the SRS technology incorporated herein, must sign a license for use of the chip and display of the SRS trademarks. Any product using the SRS technology incorporated in this chip must be sent to SRS Labs for review. Circle Surround II, TruSurround XT and SRS are protected under U.S. and foreign patents issued and/or pending.

Neither the purchase of this chip, nor the corresponding sale of audio enhancement equipment conveys the right to sell commercialized recordings made with any SRS technology. SRS requires all set makers to comply with all rules and regulations as outlined in the SRS Trademark Usage Manual.

For all details, contact SRS Labs at:
SRS Labs Inc., 2909 Daimler Street, Santa Ana, CA 92705 USA
www.srslabs.com

DTS® (ES, 96/24, Neo)

DTS ES, DTS 96/24 and DTS Neo are intellectual properties of Digital Theater Systems Inc. The DTS licenses allow the use of the corresponding decoder executed in the STx7100.

For all details, contact Digital Theater Systems Inc at:
DTS, 5171 Clareton Drive, Agoura Hills, CA 91301, USA
www.dtsonline.com

CPRM/CPPM

CPRM/CPPM technology is intellectual property of 4C Entity. The CPRM/CPPM license allows the use of the CPRM / CPPM technology embedded in the STx7100.

For all details, contact 4C Entity at:
4C Entity, LLC, 225 B Cochrane Circle, Morgan Hill, CA 95037, USA
www.4centity.com

Windows Media® (audio [WMA] and video [WMV])

Windows Media is intellectual property of Microsoft Corporation. The Windows Media license allows the use of the Windows Media audio and video decoders executed in the STx7100.

For all details, contact Microsoft at:
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA
www.microsoft.com

USB (Universal serial bus)

USB is intellectual property of USB-IF. The USB license gives the right to display the certified USB logo in conjunction with a product when the product has passed USB-IF compliance testing for product quality.

For all details, contact USB-IF:
USB Implementers Forum, Inc., 5440 SW Westgate Drive, Suite 217, Portland, OR 97221, USA
www.usb.org

HDMI™ (High-definition multimedia interface™)

HDMI and High-definition multimedia interface are intellectual properties of HDMI Licensing, LLC. The HDMI license allows the use of HDMI in the STx7100.

For all details, contact HDMI Licensing, LLC at:
1060 E. Arques Avenue, Suite 100, Sunnyvale, CA 94085, USA

www.hdmi.org

HDCP

HDCP is an intellectual property of Digital Content Protection, LLC. The HDCP license allows the use of HDCP in the STx7100.

For all details, contact Digital Content Protection, LLC at:
C/O Intel Corporation, Stephen Balogh, JF2-55, 2111 NE 25th Ave, Hillsboro, OR 97124

www.digital-cp.com

Confidential