STi7710

Single-chip, low-cost HD set-top box decoder

ADVANCE DATA

Features

The STi7710 is a single-chip, high-definition MPEG decoder including:

- **CPU core**
- **Transport filtering and descrambling**
- **Video decoder**
- **Graphics engine**
- **Dual display**
- **Audio decoder**

The STi7710 also features the following embedded interfaces:

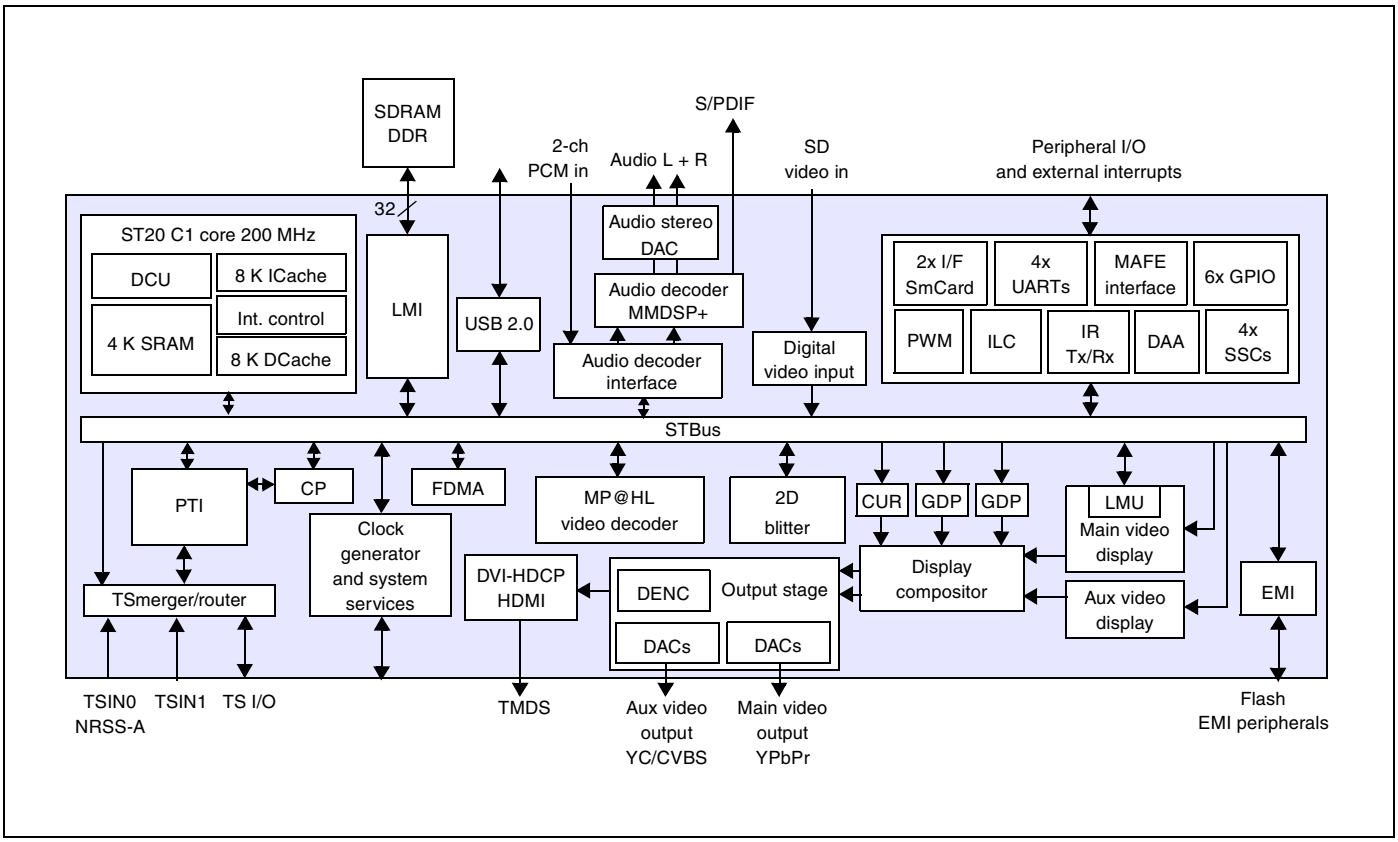
- **USB 2.0 for hard-disk drive support**
- **DVI/HDMI** (Digital Visual Interface and High-Definition Multimedia Interface™)
- **Digital audio and video auxiliary input**
- **Low-cost modem solution**

Processor subsystem

- **Enhanced ST20 32-bit VL RISC CPU**
  - 200 MHz 8-Kbyte ICache, 8-Kbyte DCache, 4-Kbyte SRAM
  - diagnostic controller unit (DCU)
  - 16 level interrupt controller

Transport stream subsystem

- **TS merger/router**
  - 2 serial/parallel inputs
  - 1 bidirectional interface
  - merging of 3 transport streams
  - software transport stream support
  - NRSS-A module interface
- **Programmable transport interface (PTI)**
  - transport stream demux: DVB, DIRECTV, ATSC, OpenCable, DCII, BS4
  - integrated DES, DVB and Multi2 encryption descramblers



Video/graphics subsystem

- **MPEG-2 MP@HL video decoder core**
- **SD (ITU-R BT 601/656) digital video input**
- **Displays**
  - 1 HD display, multiformat capable (1080I, 720P, 480P, 480I)
    - analog HD output RGB or YPbPr
    - HDMI encoded output
  - 1 standard-definition analog display output: YPbPr, or YC and CVBS
- **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
- **Compositor and video processor**
  - 5 channel mixer for high definition output
  - independent 2-channel mixer for SD output
  - 2 graphic display planes
  - hardware cursor
  - picture up-conversion hardware
  - linear/nonlinear resizing and format conversions
  - horizontal and vertical filtering
- **Copy protection**
  - HDMI / HDCP copy protection hardware
  - Macrovision™ copy protection for 480I and 480P output

Audio subsystem

- **Digital audio decoder**
  - MPEG-1 layer I/II, MP3, MPEG-2 layer II, MPEG-2 AAC and AC3 Dolby® Digital
  - SRS® TruSurround XT™,TruBass® and FOCUS™
  - PCM mixing with internal or external source and sample rate conversion
  - 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**
  - 16-bit interface supporting ROM, Flash, SFlash, SRAM
  - access in 5 banks
- **Local memory interface**
  - 32 bit DDR interface
- **USB 2.0 host interface**
- **Hard-disk drive support**
  - record and playback
  - pause and time shifting
  - watch and record
- **On-chip peripherals**
  - 4 ASCs (UARTs) with Tx and Rx FIFOS, 2 of which can be used in smartcard interfaces
  - 2 smartcard interfaces and clock generators (improved to reduce external circuitry)
  - 4 SSCs for I²C/SPI master slaves interfaces
  - 2 PWM outputs
  - teletext serializer and DMA module
  - 6 banks of general purpose I/O, 5 V tolerant
  - SiLabs line-side (DAA) interface
  - modem analog front end (MAFE) interface
  - infrared transmitter/receiver supporting RC5, RC6 and RECS80 codes
  - dual noise filters for IR inputs with programmable active levels
  - interrupt level controller and external interrupts, 5 V tolerant
  - low power / RTC / watchdog controller
  - integrated VCXO
  - UHF input interface
- **Flexible multichannel DMA**

Services

- **JTAG/TAP interface**
- **DCU toolset support**

Package

- **27 x 27 PGBA, 420 + 36 balls, 1 mm pitch**

Contents

1 Introduction

1.1 STi7710 applications

The STi7710 is a new generation of high-definition set-top box decoder chip, and provides very high performance for low-cost HD systems.

Based on the STBus (Omega2) 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 STi7710 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 down sampled display intended for a VCR. 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 or via integrated stereo audio DACs.

Digitized NTSC or PAL programs can also be input to the STi7710 for reformatting and display.

The STi7710 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 STi7710 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 and Multi2.

The STi7710 embeds a 200 MHz ST20 CPU for applications and device control and makes possible a unified memory architecture by providing a high bandwidth DDR DRAM interface. A second memory bus is also provided for Flash memory storing resident software and for connection of peripherals.

An external hard-disk drive (HDD) can be connected either to the EMI or as an expansion drive via the USB 2.0 port.

The STi7710 is supported by STMicroelectronics' STAPI software, and provides a compatible next-generation architecture for applications currently running on the STi5517 and STi7020.

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

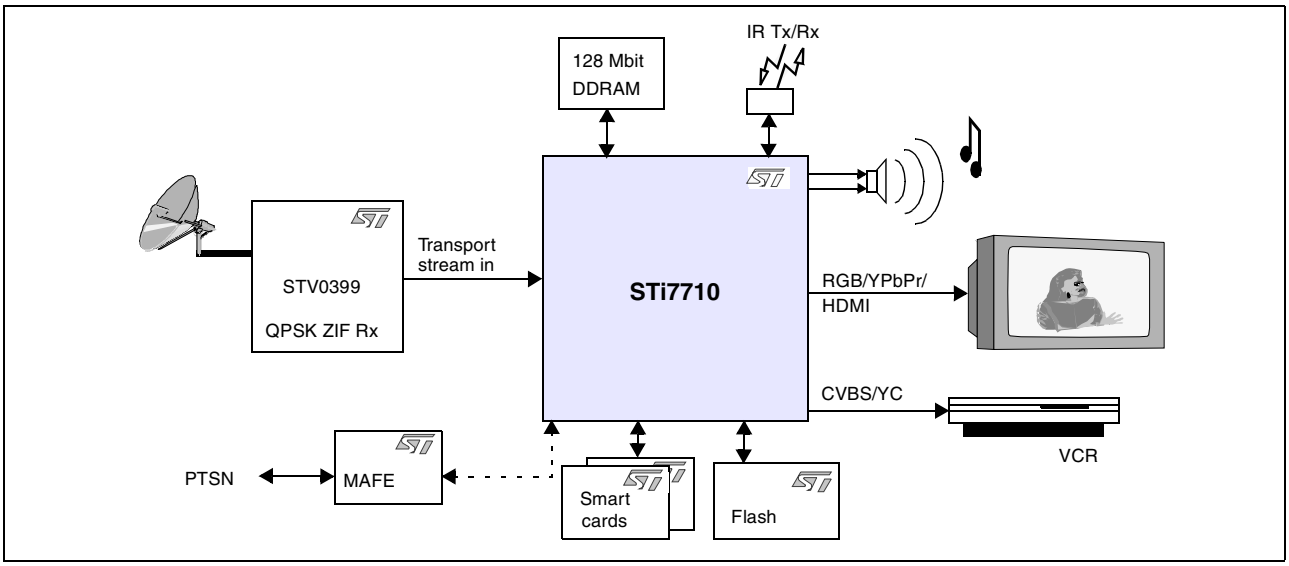


Figure 2: Low-cost cable HD set-top box

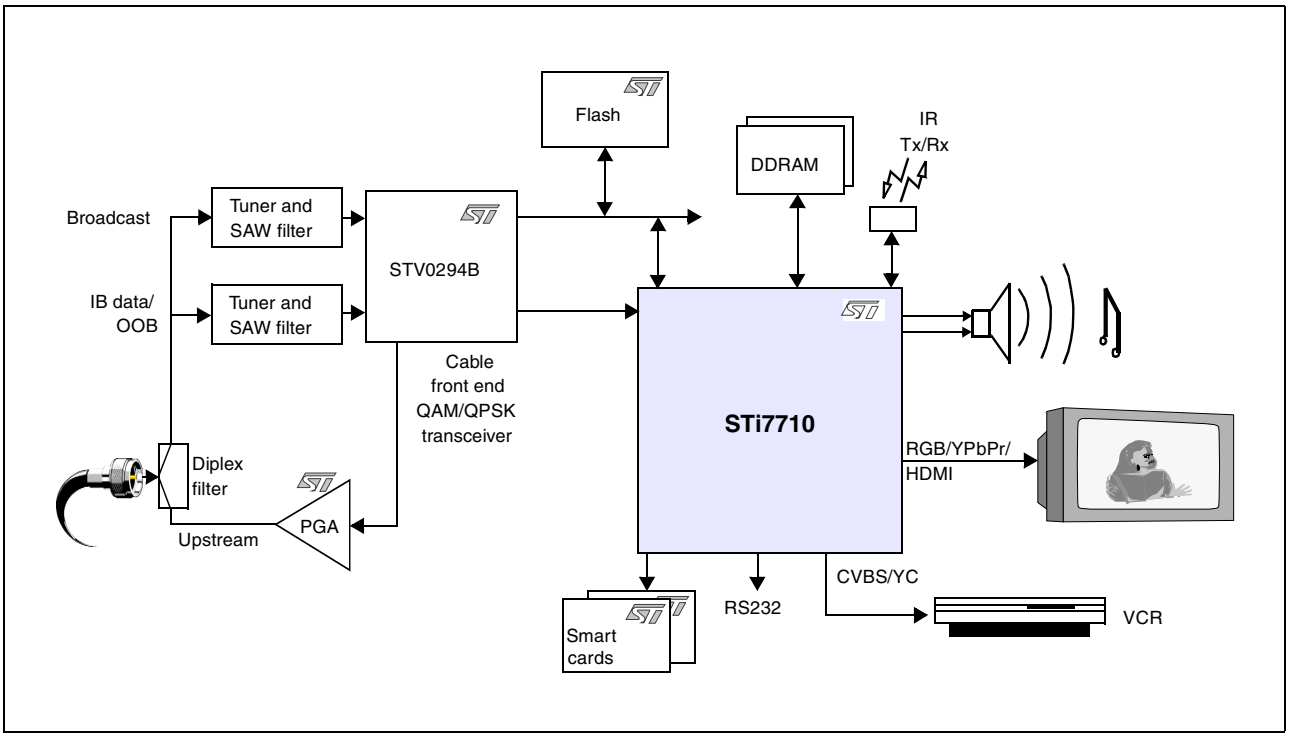
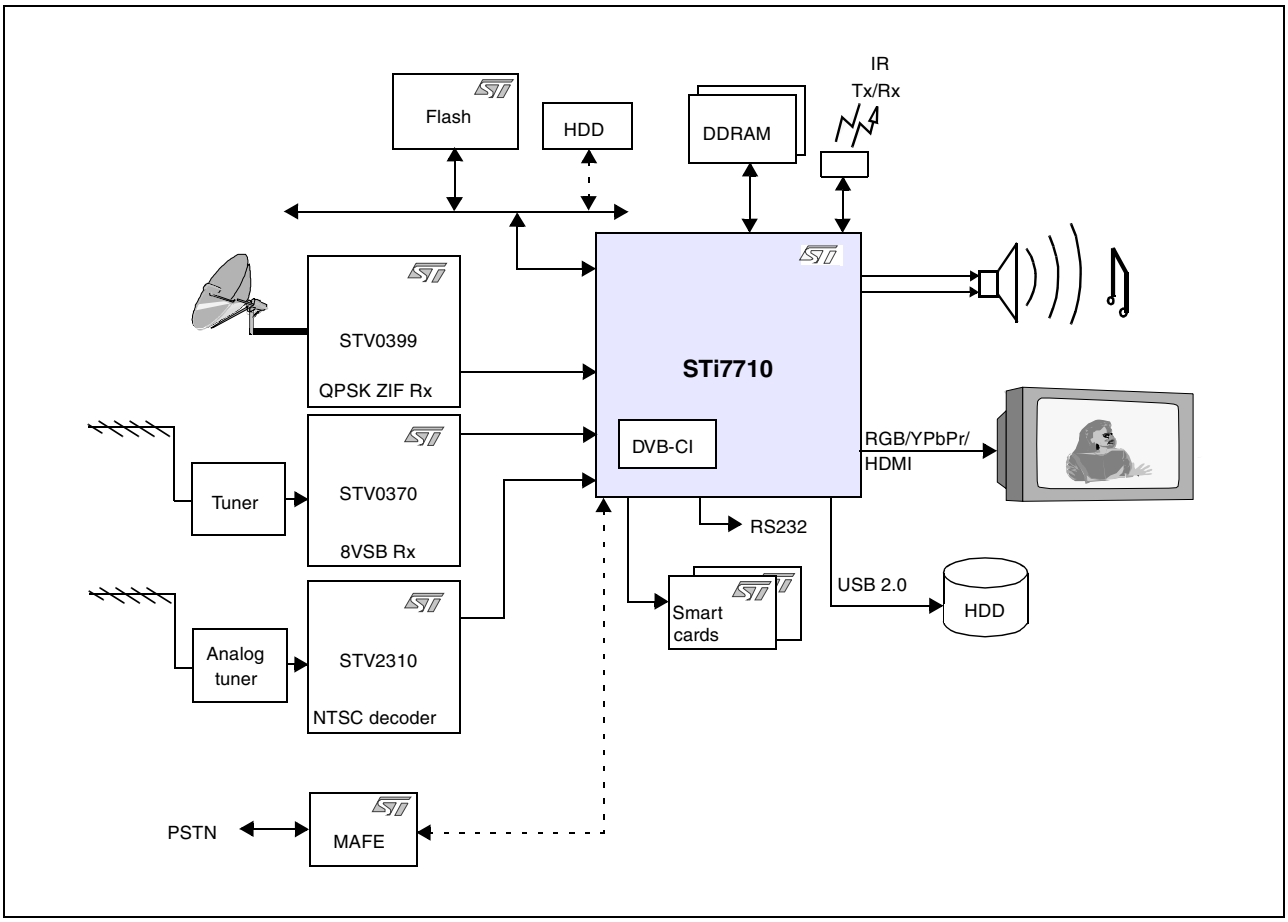


Figure 3: Low-cost dual satellite and terrestrial HD set-top box with HDD



2 Architecture

2.1 Architecture overview

The STi7710 is designed around the well-proven STBus (Omega2) interconnect.

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 interface (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 video decoder decodes HD or SD video streams. Audio decoding and PCM mixing is performed by the MMDSP and output via S/PDIF or through integrated 24-bit stereo DACs

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 YPbPr analog component or composite video on a separate interface for connection to a VCR.

A digital video input interface allows the STi7710 to receive SD uncompressed 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 curser plane and dual display compositor. Graphics buffers are created, stored in and displayed from buffers in DDR SDRAM.

The STi7710 embeds an ST20-C1 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 move and stream data transfers with minimal CPU intervention.

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

The STi7710 also integrates a range of peripherals, system services and a clock generator module with embedded VCXO (actually programmable frequency synthesizers able to replace the classical external VCXO) to significantly reduce external component cost.

2.2 Processor core

The STi7710 integrates a 200 MHz ST20-C105 processor core that is composed of the ST20C1+ CPU, a diagnostic controller unit (for low intrusion, real-time debugging), memory (8 Kbyte instruction cache, 8 Kbyte data cache and 4 Kbyte SRAM) and a 16 priority-level interrupt controller.

2.3 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 5 separate banks are available, each capable of its own strobe timing configuration and each with its own chip select signal. Two banks provide PC-Card compatible strobes for implementing a DVB-CI or CableCard (POD) module interface.

2.4 Local memory interface (LMI)

The LMI is a 32-bit, high-bandwidth memory interface that enables a unified data memory architecture through the use of DDR SDRAM. CPU instructions can also be placed here. It can operate in a 16-bit or 32-bit configuration and has a target operating frequency of 200 MHz for a peak bandwidth of 1.6 Gbyte/s. In a 32-bit configuration, the LMI supports one x32 device or two x16 devices. 64-Mbit, 128-Mbit or 256-Mbit DDR devices can be used providing a maximum capacity of 64 Mbytes.

To get the maximum transfer efficiency from the interface, the LMI includes optimizations in the control of the page structure of DDR DRAMs.

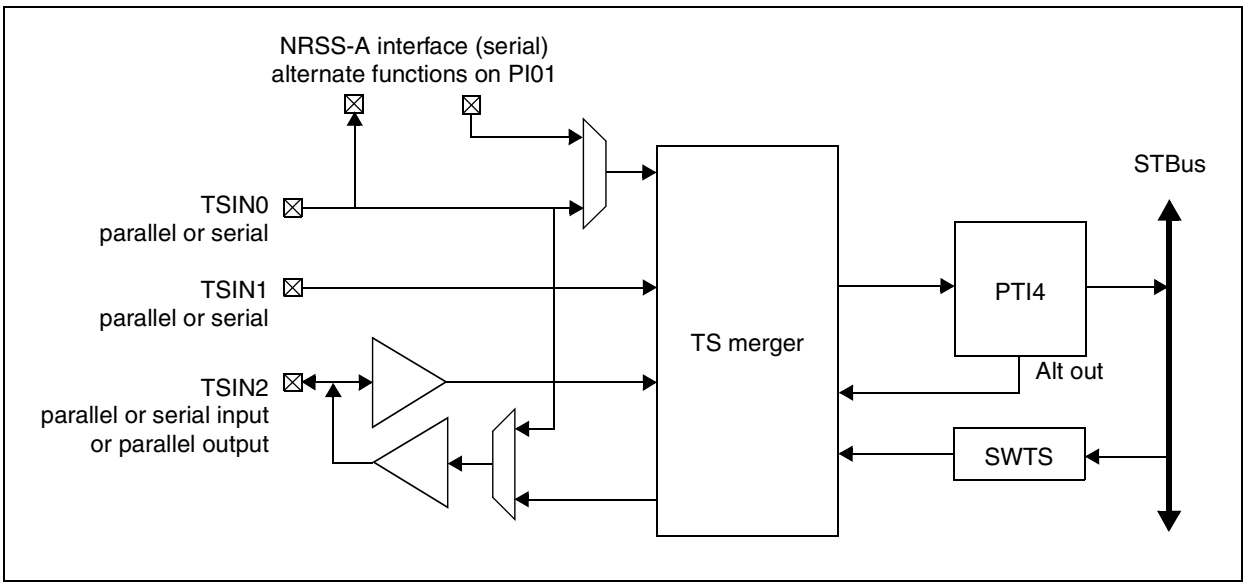
2.5 Transport subsystem

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

2.5.1 Transport stream input/output

Transport streams are input to the STi7710 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 4: 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 STi7710 via the bidirectional interface when configured as an output. This allows transport streams to be sent to a D-VCR via an IEEE1394 link layer controller, EMI or USB 2.0 port. Routing can be concurrent to two outputs.

When TS0 is operating in serial mode, an NRSS-A interface is available for routing serial transport 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 transport stream and forwarding this to the PTI allowing the PTI to process three independently sourced transport streams at the same time.

2.5.2 Programmable transport interface (PTI)

The PTI performs PID filtering, demultiplexing, descrambling, and data filtering on up to three transport streams at the same time up to an aggregate rate of 100 Mbit/s. The PTI extracts PCRs with time stamps and makes them available to the CPU for clock recovery and audio/video synchronization.

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 48 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,
- Fast-I,
- NDS specific streams can also be supported by the integration of ICAM functionality.

The PTI has a 48 x 16 byte section filter core. Four 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.

2.6 MPEG-2 video decoder

The STi7710's video decoder is a single-stream decoder for either SD or HD streams. The decoder is fully compliant with ISO/IEC13818-2 MP@ML and MP@HL formats.

A stream is decoded picture by picture from an elementary stream buffer. Decoding, reconstruction and prediction buffers are set up by the 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.

Pictures can be reconstructed with decimation for VCR recording to reduce memory bandwidth while keeping full resolution pictures for anchor frames and main display.

2.7 Digital video input

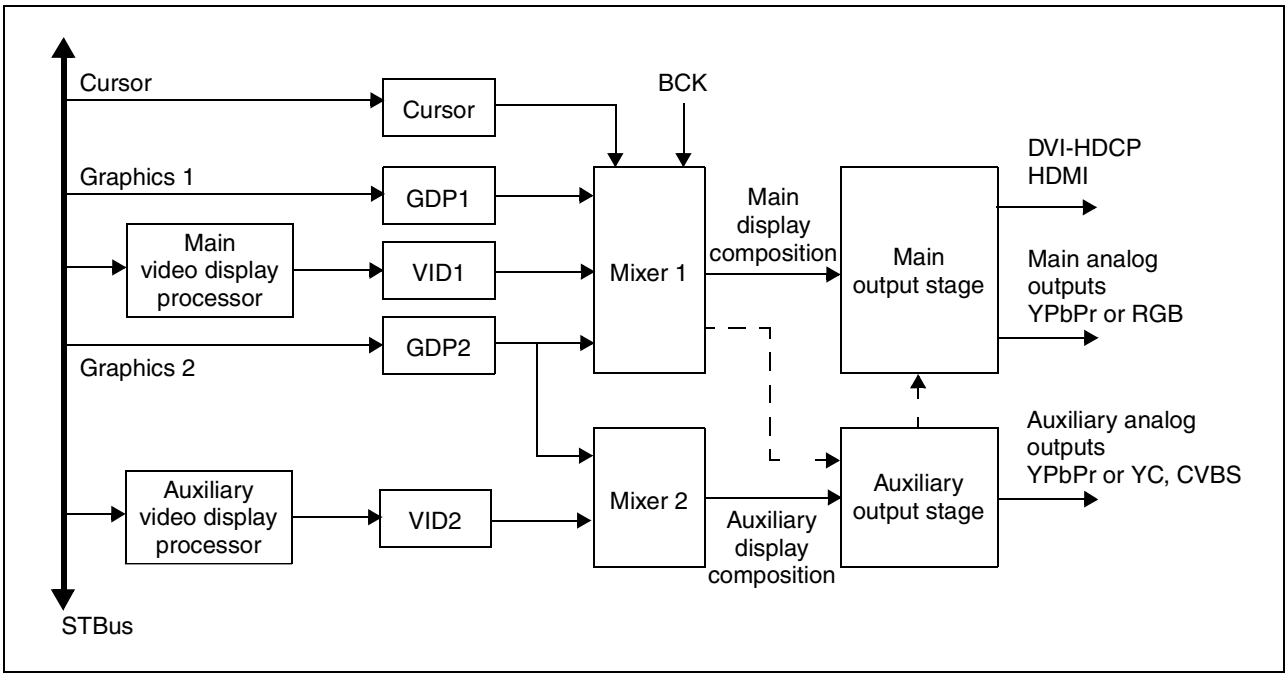
Digital SD video data can be input to the STi7710 via an 8-bit digital video input port.

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, auxiliary data embedded in the stream can be extracted to a separate buffer.

2.8 Video display processors

The STi7710 displays video using the main and auxiliary display processors (Figure 5). The video may have been decoded and reconstructed by the MPEG decoder or acquired via the digital video interface.

Figure 5: Display, composition and output



The same video is displayed via both 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 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 line size is limited to 720 pixels on the auxiliary-display processor and is intended to output video for VCR recording.

2.9 Graphics display

2.9.1 Graphics layers

The STi7710 has two independent and identical graphics layers known as generic display pipelines (GDPs) (see Figure 5). 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 eight subpositions).
- Color keying capability.

2.9.2 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 four bits handles an antialiased cursor pattern on top of the composed output picture. The curser-plane features are:

- 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.

2.10 Display compositor

The graphic compositor consists of a 5-layer digital mixer (MIX1) intended for the main TV display (Figure 6) and a two-layer digital mixer (MIX2) intended as an auxiliary display for applications including connection to a VCR (Figure 7).

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

Note: GDP2 can only be used by one mixer (MIX1 or MIX2) for a given application.

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.

The auxiliary mixer MIX2 mixes the auxiliary video display with the GDP2 graphics layers (provided they are not being used by the main mixer MIX1).

Figure 6: MIX1 planes

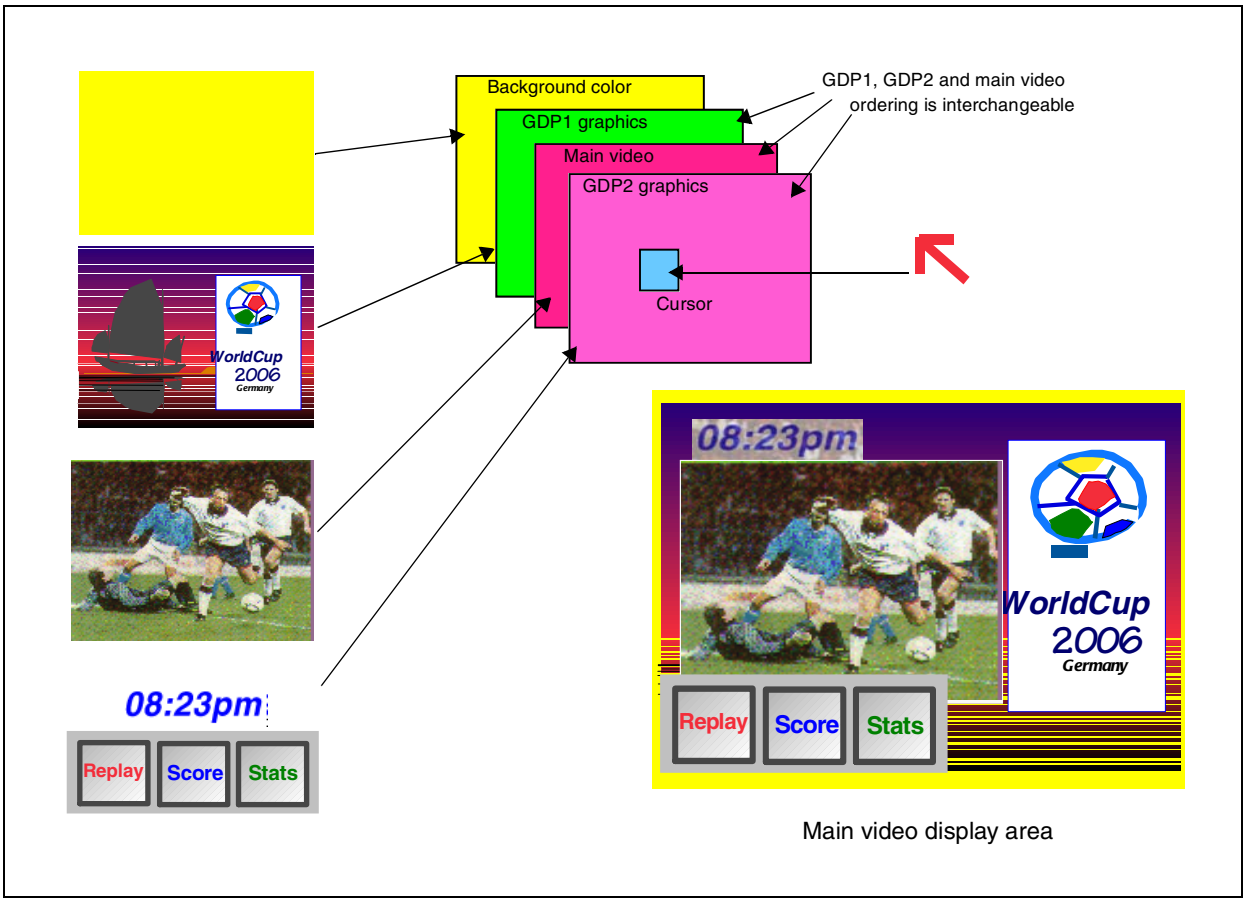
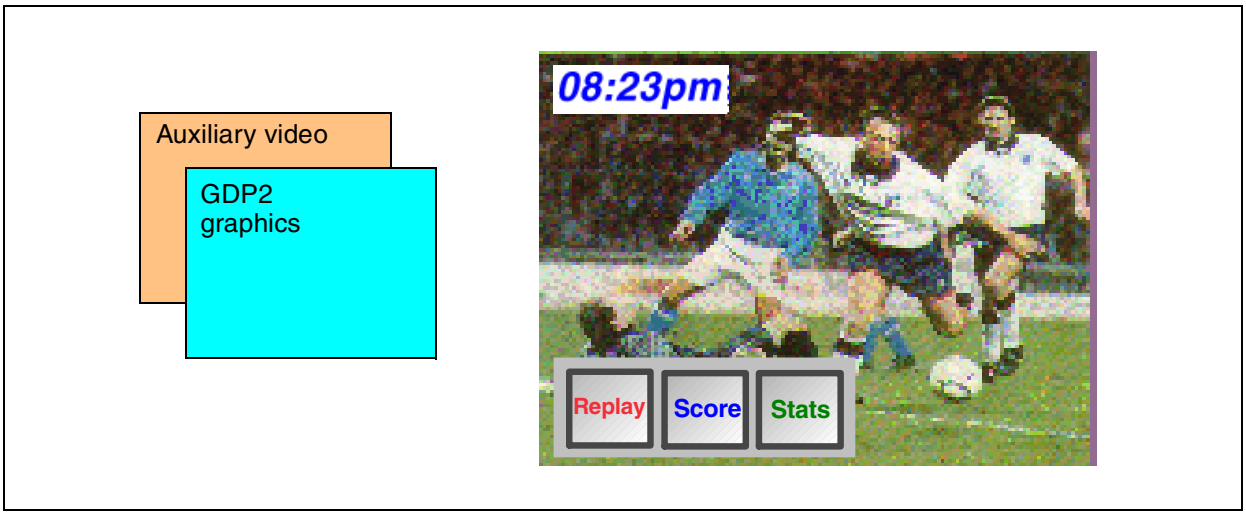


Figure 7: MIX2 planes



2.11 Main display output stage

The display composition from MIX1 can be output on any of the main display output interfaces (Figure 5). 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 BS4 SD and HD formats (480i - sometimes called 525i, 480p - sometimes called 525p, 750p, 1125i) 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 and 480p and CGMS encoding.

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

2.12 Auxiliary display output stage

The display composition from MIX2 is output on the auxiliary display output interface (Figure 5).

This interface contains a digital encoder that encodes the output from the auxiliary mixer into a standard analog baseband PAL/NTSC signal and into YPbPr components.

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 on which it is possible to output either (S-VHS(Y/C) + CVBS), YPbPr or RGB.

2.13 2D blitter

The 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 features of the STi7710's blitter are as follows:

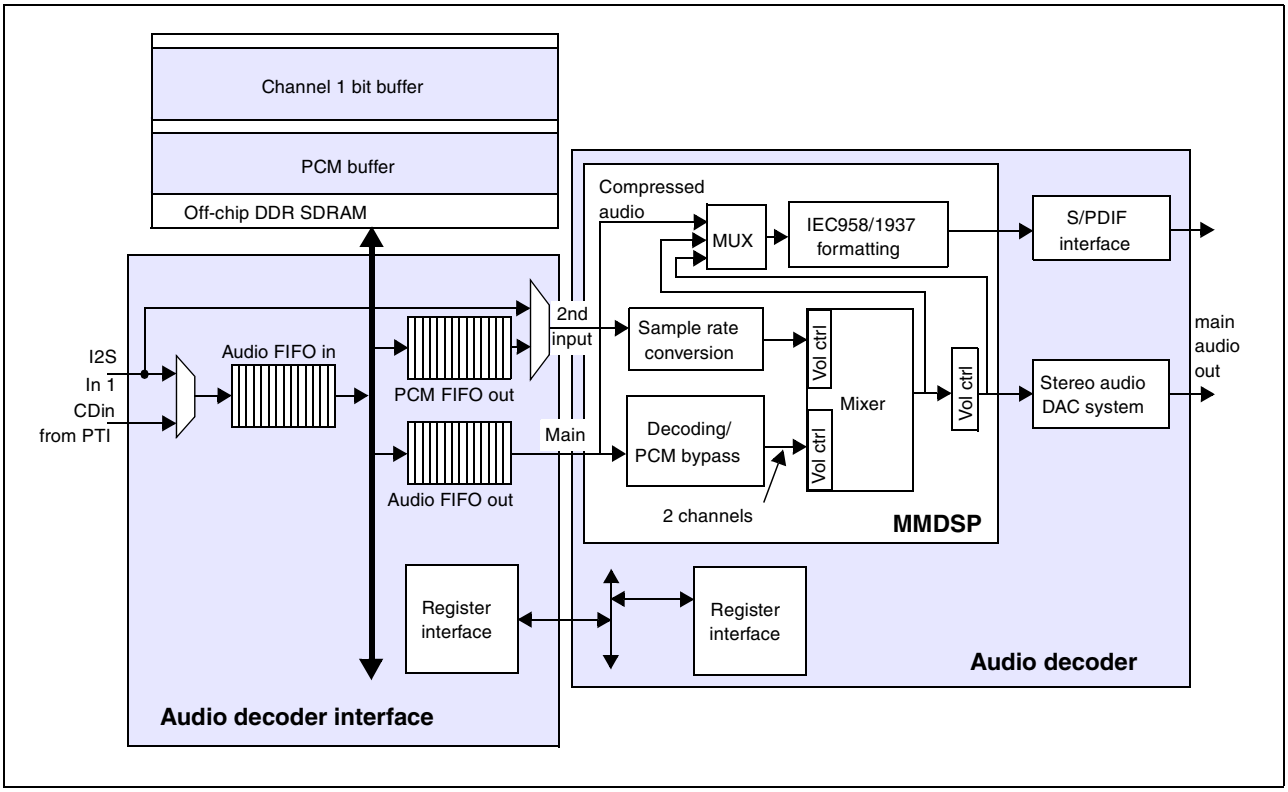
- Solid color fill of rectangular window.
- Solid color shade (fill + alpha blending).
- One source copy, with one or several operators enabled (color format conversion, 2D scaling).
- Two-source copy with alpha blending or logical operation between them.
- 4:2:2 raster/macroblock and 4:2:0 macroblock 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 ACLUTn) 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.

2.14 Audio subsystem

The audio subsystem comprises the audio controller, the MMDSP 24-bit audio digital signal processor and the audio output interfaces. The audio subsystem is illustrated in Figure 8.

Figure 8: Audio decoder interface block diagram



2.14.1 Audio decoder interface

The audio decoder interface receives, buffers and reformats audio data. It handles up to three audio data flows concurrently.

- It receives a raw PCM stream from an external source via the PCM input interface or receives a compressed data stream from an internal source such as the PTI and stores this in a memory buffer via DMA. This is used to buffer and play the main audio for the digital or analog program.
- It receives a PCM file or stream from a memory buffer via DMA and delivers this to the MMDSP's 2nd input for sample rate conversion and mixing with the main audio. The 2nd input could also receive a PCM stream directly from the PCM input interface. This is used to play PCM data which is mixed with the main audio.

2.14.2 Audio decoder

The audio decoder receives the compressed audio (PES or ES) or PCM audio stream on its main input. For compressed data it performs PES parsing, PTS extraction and multichannel decoding. and downmixing. At the same time it formats the multichannel compressed data stream for output over the S/PDIF interface for external decoding. Decoded multichannel audio is downmixed before emerging as downmixed stereo or Dolby Pro Logic® encoded audio. True Surround XT® post processing may be applied. PCM audio is passed straight through.

Decoding of MPEG-1 layers I, II, MP3, Dolby® Digital and AAC stereo are supported at sample rates of 32 kHz, 44.1 kHz and 48 kHz.

The decoded/downmixed channels may be mixed with PCM files from the 2nd input which the audio decoder has sample rate converted.

The audio decoder is also capable of programmable tone generation for dish alignment.

The audio decoder outputs the following.

- Main 2-channel output: The left and right decoded/downmixed channels (L/Lt, R/Rt) with PCM file mixing.
- S/PDIF output: A digital audio stream selected from one of the following:
  - IEC61937 formatted compressed audio received on the audio decoder's main input,
  - IEC60958 formatted version of the main 2-channel output.

2.14.3 Audio output interfaces

The audio output interfaces from the STi7710 are as follows.

- S/PDIF interface. This outputs the S/PDIF output from the audio decoder.
- Analog stereo output from integrated 24-bit DACs. This outputs the main 2-channel output from the audio decoder.

2.15 FDMA controller

The STi7710 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. These channels are suitable for transfers with internal or external peripherals and for audio and video stream transfers within the STi7710.
- Two external pacing signals for paced transfers to and from external peripherals.

2.16 Interfaces

2.16.1 USB

The STi7710 has an integrated USB host controller with one host port. The USB host interface is partially compliant with OHCI/EHCI rev 1.0 and USB rev 2.0, allowing connection to a hard-disk drive only. All speeds up to 480 Mbit/s are supported.

2.16.2 Modem

Standard solutions are available for V22bis software modems and V34/V90 controllerless modems ported to the STi7710 architecture.

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

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

2.16.3 Internal peripherals

The STi7710 has many dedicated internal peripherals for digital TV receiver applications, including:

- 2 smartcard controllers
- 4 ASCs (UARTs), two of which are generally used by the smartcard controllers
- Teletext serializer and DMA
- 4 SSCs for I^2C/SPI master/slave interfaces
- 6 GPIO ports (5V tolerant)
- 2 PWM modules:
  - the first (“PWM4”, also used as ST20-C1 time slicer), usable as timer or to control one PWM output
  - a second, dual module (“PWM-Timer2”) with each channel usable as a timer or to control a PWM output
- a multi-channel, infrared blaster/decoder interface module
- an interrupt level controller with 4 external interrupt inputs (5 V tolerant)

2.16.4 Smartcard interfaces

Both STi7710 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 a card’s insertion or removal is detected.

2.17 Clock generation

All system clocks are generated on chip using the clock generator module.

VCXO functionality has been integrated using a special purpose frequency synthesizer, removing the need for an external varactor diode or VCXO module. However support for an external VCXO module is also available.

A bank of digital frequency synthesizers is also provided for specific precise clock generation purposes, including generation of a low jitter PCM audio clock, a smartcard clock, and an external auxiliary clock.

2.18 System services

The STi7710 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 blaster,
- Real-time clock,
- JTAG boundary scan,
- Diagnostic control / support for DCU toolset.

3 Pin list and alternative functions

Signal names are prefixed by NOT_ or N_ if they are active low; otherwise they are active high.

All digital pads are 3.3 V capable. 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. I/O and power supplies pertaining to the same type of interface have been grouped under common keys.

3.1 Tri-state control and pull-up resistors

The following digital pads can be tri-stated:

- all programmable I/Os, under software control.
- all EMI pins during hardware reset and when an external master requests control of the EMI bus (refer to EMI chapter)

The following digital pads have internal weak pull-up resistors:

- all PIOs (pull-up can be disabled under SW control),
- TDI, TRST, TMS and TCK on the JTAG interface.

All IOs are 3.3 V tolerant as inputs or tri-stated outputs, and 3.3 V capable as outputs.

PIOs are 3.3 V capable as outputs, but 5 V tolerant as inputs or tri-stated outputs.

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	PIO4_4	PIO4_6	PIO5_0	PIO5_3	PIO5_6	LMI_DATA_8	LMI_DATA_10	LMI_notDQS_1	LMI_DATA_13	LMI_DATA_15	LMI_DATA_25	LMI_DATA_27	LMII_DATA_28
B	PIO3_7	PIO4_1	PIO4_5	PIO5_2	PIO5_5	LMI_DATA_9	LMI_DATA_11	LMI_DATA_12	LMI_DATA_14	LMI_DATA_24	LMI_DATA_26	LMI_notDQS_3	LMI_DATA_29
C	PIO3_4	PIO4_0	PIO4_3	PIO4_7	PIO5_4	PIO5_7	LMI_notDQM_1	VDD_RING	LMI_notDQM3	LMI_ADDR_10	LMI_ADDR_8	LMI_ADDR_6	LMI_ADDR_14
D	PIO3_0	PIO3_3	PIO3_5	PIO4_2	PIO5_1	VDD_RING	LMI_CLK	LMI_notCLK	LMI_CLKEN	LMI_ADDR_9	LMI_ADDR_7	LMI_ADDR_11	LMI_ADDR_13
E	PIO2_4	PIO2_7	PIO3_2	PIO3_6	VDDE3V3_RING	VDD_RING	VDD_RING	VDDE2V5_RING	GNDE2V5_RING	GNDE2V5_RING	VDDE2V5_RING	VDD_RING	GNDE2V5_RING
F	PIO2_0	PIO2_3	PIO2_6	PIO3_1	VDDE3V3_RING								
G	PIO1_4	PIO1_7	PIO2_2	PIO2_5	VDDE3V3_RING								
H	PIO1_1	PIO1_3	PIO1_6	PIO2_1	GNDE3V3_RING								
J	PIO0_5	PIO0_7	PIO1_2	PIO1_5	GNDE3V3_RING								
K	PIO0_2	PIO0_3	PIO0_6	PIO1_0	VDDE3V3_RING								
L	DAA_C1A	PIO0_0	PIO0_1	PIO0_4	VDD_RING						GND_RING	GND_RING	GND_RING
M	DAA_C2A	AF_VDD1	AF_VDD0	VDD_RING	VDD_RING						GND_RING	GND_RING	GND_RING
N	TS_DATA2_7	AF_VDD4	AF_VDD3	AF_VDD2	VDDE3V3_RING						GND_RING	GND_RING	GND_RING
P	TS_DATA2_3	TS_DATA2_4	TS_DATA2_5	TS_DATA2_6	GNDE3V3_RING						GND_RING	GND_RING	GND_RING
R	TS_DATA2_2	TS_DATA2_1	TS_DATA2_0	TSPKTERR1	VDD_RING						GND_RING	GND_RING	GND_RING
T	TSPKTERR0	TSVALID1	TSVALID0	TSPKTCLK1	VDD_RING						GND_RING	GND_RING	GND_RING
U	TSPKTCLK0	TSBYTECLK1	TSBYTECLK0	TS_DATA0_6	VDDE3V3_RING								
V	TS_DATA0_7	TS_DATA0_5	TS_DATA0_3	TS_DATA0_1	VDDE3V3_RING								
W	TS_DATA0_4	TS_DATA0_2	TSVALID2	TSBYTECLK2	VDDE3V3_RING								
Y	TS_DATA0_0	TSPKTERR2	TS_DATA1_7	TS_DATA1_4	VDDE3V3_RING								
AA	TSPKTCLK2	TS_DATA1_6	TS_DATA1_3	TS_DATA1_0	VDDE3V3_RING								
AB	TS_DATA1_5	TS_DATA1_2	EMI_PRTSZ	EMI_DATA_14	VDDE3V3_RING	VDD_RING	VDD_RING	VDD_RING	VDD_RING	VDD_RING	GNDE3V3_RING	VDDE3V3_RING	GND_HDMI_RING
AC	TS_DATA1_1	EMI_WAIT	EMI_DATA_13	EMI_DATA_7	EMI_DATA_3	EMI_RDnWR	EMI_nCS_2	EMI_nBAA	EMI_ADDR_22	EMI_ADDR_17	EMI_ADDR_12	EMI_ADDR_8	EMI_ADDR_3
AD	EMI_DATA_15	EMI_DATA_12	EMI_DATA_8	EMI_DATA_4	EMI_DATA_0	EMI_nCS_3	EMI_nBE_1	EMI_CLKF	EMI_ADDR_19	EMI_ADDR_15	EMI_ADDR_11	EMI_ADDR_7	EMI_ADDR_4
AE	EMI_DATA_11	EMI_DATA_9	EMI_DATA_5	EMI_DATA_1	EMI_nCS_4	EMI_nCS_0	EMI_nLBA	EMI_ADDR_21	EMI_ADDR_18	EMI_ADDR_14	EMI_ADDR_10	EMI_BUS_REQ	EMI_ADDR_2
AF	EMI_DATA_10	EMI_DATA_6	EMI_DATA_2	EMI_nOE	EMI_nCS_1	EMI_nBE_0	EMI_ADDR_23	EMI_ADDR_20	EMI_ADDR_16	EMI_ADDR_13	EMI_ADDR_9	EMI_ADDR_6	EMI_ADDR_5
	1	2	3	4	5	6	7	8	9	10	11	12	13

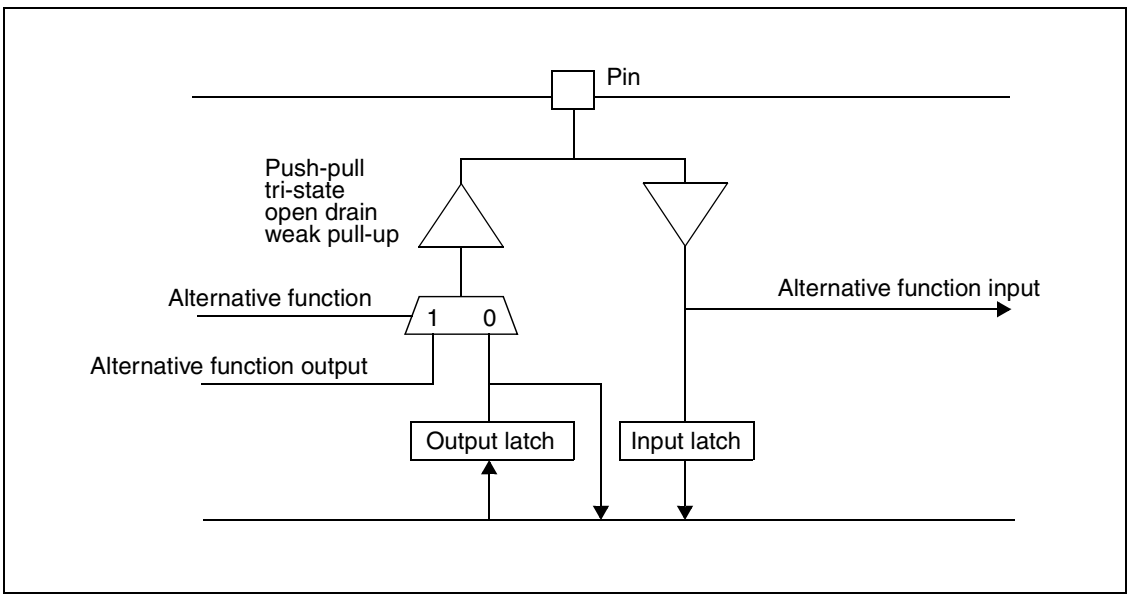
	14	15	16	17	18	19	20	21	22	23	24	25	26
A	LMI_DATA_30	LMI_notDQS_0	LMI_DATA_1	LMI_DATA_3	LMI_DATA_5	LMI_DATA_7	LMI_DATA_17	LMI_DATA_19	LMI_DATA_20	LMI_DATA_22	EXT_IRQ2	DCU_TRIGGER OUT	CG_DLL_VDD
B	LMI_DATA_31	LMI_DATA_0	LMI_DATA_2	LMI_DATA_4	LMI_DATA_6	LMI_DATA_16	LMI_DATA_18	LMI_notDQS_2	LMI_DATA_21	LMI_DATA_23	PLL_LMI_VDDD	CG_DLL_VSS	EXT_DMA_REQ2
C	VDDE2V5_RING	LMI_ADDR_5	LMI_ADDR_3	LMI_notBANK_1	LMI_notCS_0	LMI_notRAS	LMI_notDQM_0	LMI_VREF	LMI_notDQM_2	PLL_LMI_GNDD	DCU_TRIGGERIN	EXT_DMA_REQ1	TCK
D	LMI_ADDR_12	LMI_ADDR_4	LMI_ADDR_2	LMI_notBANK_0	LMI_notCS_1	LMI_notCAS	LMI_RDNOTWR	PLL_LMI_VCCA	PLL_LMI_GNDA	EXT_IRQ3	EXT_nRESET_OUT	TDO	PLL_USB_GNDA
E	VDD_RING	VDDE2V5_RING	GNDE2V5_RING	VDDE2V5_RING	VDD_RING	GNDE2V5_RING	VDDE2V5_RING	VDD_RING	GNDE2V5_RING	NOT_RESET	PIXCLK	TRST	PLL_USB_VDDA
F									GNDE2V5_RING	AUXCLK OUT	TDI	PLL_USB_GNDD	PLL_USB_VDDD
G									GNDE2V5_RING	TMS	USB_RREF	USB_VSSB	USBDN
H									VDDE_DIG2_RING	USB_VDD3	USB_VDDL	USB_VDDC	USBDP
J									GNDE_DIG2_RING	USB_VSSL	USB_VDDB	XTAL1	FS0_GNDD
K									GNDE_DIG2_RING	USB_VSSC	XTAL2	FS2_VDDD	FS0_VDDD
L	GND_RING	GND_RING	GND_RING						VDD_USB_RING	FS0_VCCA	FS0_GNDA	FS2_GNDD	FS1_VDDD
M	GND_RING	GND_RING	GND_RING						GNDE_USB_RING	FS1_VCCA	FS1_GNDA	FS2_VCCA	FS1_GNDD
N	GND_RING	GND_RING	GND_RING						GND_ANA_RING	VBG	SUBANA	IREF	FS2_GNDA
P	GND_RING	GND_RING	GND_RING						GND_ANA_RING	OUTMR	OUTPR	VDD3	VSS3
R	GND_RING	GND_RING	GND_RING						VDD_ANA_RING	VDDA	GNDA	OUTML	OUTPL
T	GND_RING	GND_RING	GND_RING						VDDE_ANA_RING	VCCA1	IDUMPV	VOUT	VCCA_GREEN
U									GNDE_ANA_RING	VCCA_BLUE	GNDA1	MASS_QUIET	REXT
V									GND_HDMI_RING	VCCA_RED	VCCA2	IDUMPW	WOUT
W									VDDE_DIG3_RING	IDUMPU	UOUT	GNDA2	AUXY
Y									VDD_DIG3_RING	ALRCLKIN	ACVBS	SD_GNDAS_REXT	SD_REXT
AA									GNDE_DIG3_RING	PCMCLK	ASIN	AUXC	SD_AGND
AB	GNDE_HDMI_RING	GNDE_HDMI_RING	GNDE_HDMI_RING	GNDE_HDMI_RING	VDD_HDMI_RING	VDDE_HDMI_RING	VDDE_HDMI_RING	GNDE_DIG3_RING	GNDE_DIG3_RING	DHSO	AUDIO_PRO-D_TEST	LRCLK	SD_AVDD
AC	DYCIO_13	GNDE_HDMI_RING	HDMI_VDDC2	HDMI_VDDC1	HDMI_VDDCK	HDMI_VDDC0	PLL_TMD_GNDA	PLL_TMD_VDDA	DYCIO_5	DYCIO_1	DVSO	SPDIFO	ASTRB
AD	DYCIO_14	DYCIO_10	COMPENSATION_PAD	HDMI_VSSC2	HDMI_VSSC1	HDMI_VSSP	HDMI_VDDP	HDMI_VDDX	DYCIO_9	DYCIO_6	DYCIO_2	DYCIO_0	SCLK
AE	DYCIO_15	DYCIO_11	HDMI_VDDSL	HDMI_VSSSL	HDMI_VSSD	HDMI_VDDD	HDMI_VSSCK	HDMI_VSSC0	HDMI_VSSX	PLL_TMD_VDDD	DYCIO_7	DYCIO_3	NOT_CLAMP
AF	EMI_ADDR_1	DYCIO_12	HDMI_TX2P	HDMI_TX2N	HDMI_TX1P	HDMI_TX1N	HDMI_TXCKP	HDMI_TXCKN	HDMI_TX0P	HDMI_TX0N	PLL_TMD_GNDD	DYCIO_8	DYCIO_4
	14	15	16	17	18	19	20	21	22	23	24	25	26

3.2 Alternative functions

To improve flexibility and to allow the STi7710 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.

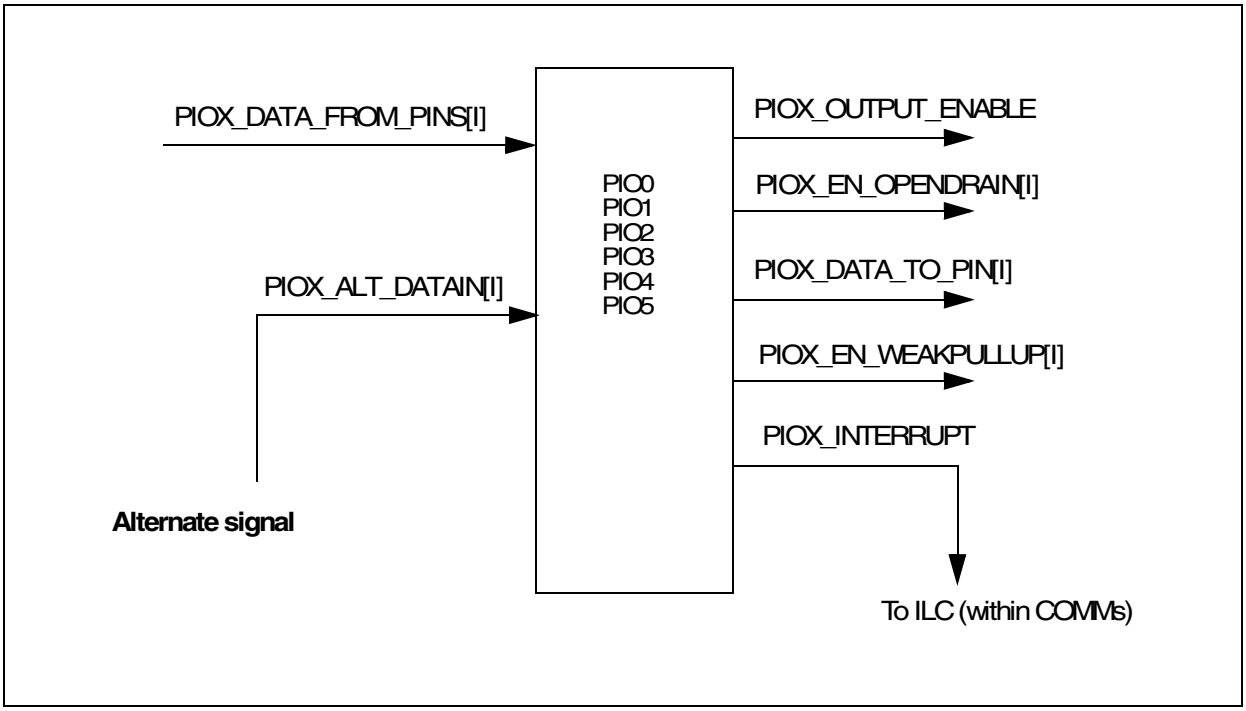
Inputs connected to the alternative function input are permanently connected to the input pin. The output signal from a peripheral is only connected when the PIO bit is configured into either push-pull or open drain driver alternative function mode.

Figure 9: I/O port pins



3.2.1 PIOs

Figure 10: PIO alternate functions



3.2.2 Assignment

The following figure illustrates the assignment of the alternate function and the notation used for the tables of the next paragraph.

Figure 11: Alternate function assignment

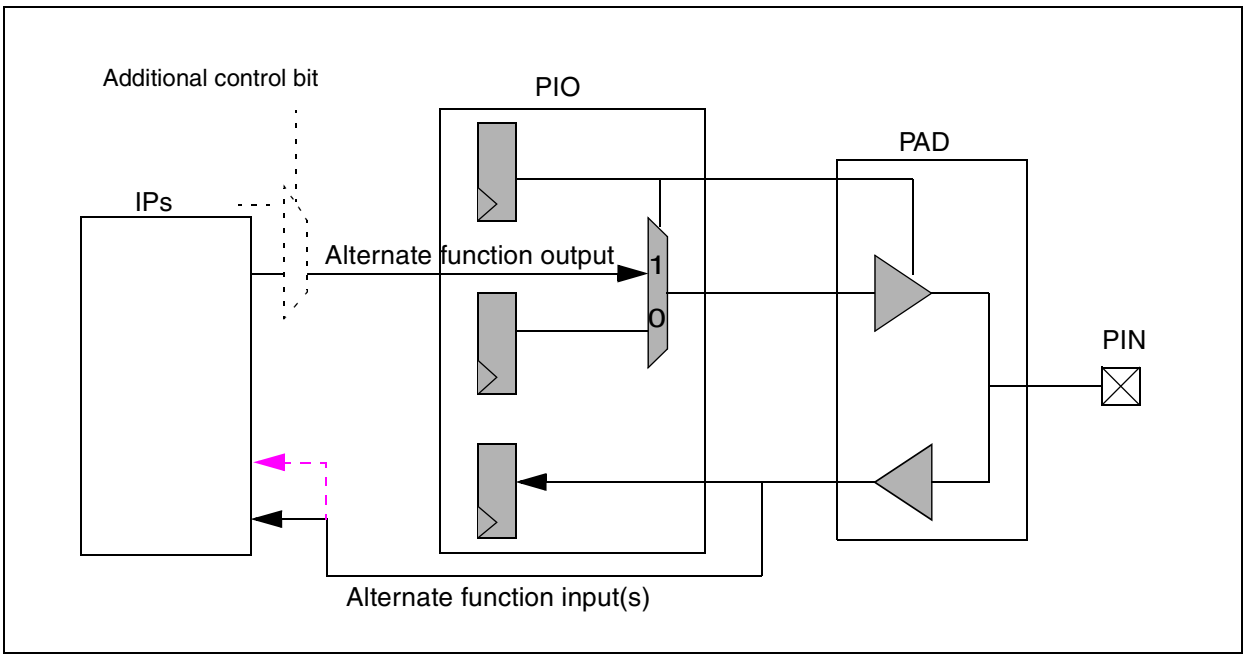


Table 1 to *Table 6* show the mapping of the alternative functions. Unless otherwise stated, the default after reset for alternative functions is the PIO port or pin function.

Some PIOs used as inputs may be shared with more than one alternate function (this corresponds to sub-columns in the ‘alt function input’ column in the tables that follow. Similarly, some PIOs may be shared by several alternate functions: in that case selection of the proper alternate function to drive the PIO is under control of additional configuration bits.

At reset, all PIO pads are in input mode with internal pull-up, and are 5 V tolerant.

PIO 0

Table 1: PI0 0 alternate functions

Port 0 Bit	PIO pin functions	
	Alternate function input	**Alternate function output**
Bit 0		SC0_DATAOUT (UART_0_TXD)
Bit 1	SC0_DATAIN (UART_0_RXD)	‘0’
Bit 2	-	‘0’
Bit 3		SC0_CLKOUT (SCCLKGEN_0_CLK_OUT) [Note 2].
Bit 4	UART_0_CTS	‘0’
Bit 5		SC_COND_VCC
Bit 6		SC0_DIR (UART_0_NOTOE)
Bit 7	SC_DETECT	UART_0_RTS

Note: 1 PIO0 alternate functions are primarily intended for smart card interfacing. However if no smart card is used on PIO0, all signals to use PIO0 as classical UART interface are also available.

2 SC0_CLKOUT (output on PIO0[3]) can be derived from a dedicated frequency synthesizer (typ.36.864 MHz) or from the 100 MHz system clock.

PIO 1

Table 2: PI0 1 alternate functions

Port 1 Bit	PIO pin functions	
	Alternate function input	**Alternate function output**
Bit 0		SC1_DATAOUT (UART_1_TXD)
Bit 1	SC1_DATAIN (UART_1_RXD)	‘0’
Bit 2	SC1_EXTCLKIN (SCCLKGEN_1_EXT_CLOCKIN)	‘0’
Bit 3		SC1_CLKOUT (SCCLKGEN_1_CLK_OUT) [Note 2]
Bit 4	UART_1_CTS	NRSS_A_CLK_OUT
Bit 5	NRSS_A_DATA_IN	UART_1_RTS
Bit 6	LONG_TIME_OUT_RST	SC1_DIR (UART_1_NOTOE)
Bit 7		NRSS_A_DATA_OUT

Note: 1 UART alternate function of PIO1 is primarily for Smart Card Interfacing - usage as classical UART is possible but exclusive with NRSS-A interfacing.

2 SC1_CLKOUT can be derived from SC1_EXTCLKIN or from the 100 MHz system clock.

PIO 2

Table 3: PI0 2 alternate functions

Port 2 Bit	PIO pin functions			
	Alternate function input		Alternate function output	
			SSC0_MUX_SEL = 0	SSC0_MUX_SEL = 1
Bit 0	SSC0_SCL (SSC0_SERIAL_CLOCK_IN)		SSC0_SCL (SSC0_SERIAL_CLOCK_OUT)	
Bit 1	SSC0_MTSR (SSC0_SERIAL_DATA_IN_MTSR)	SSC0_MRST (SSC0_SERIAL_DATA_IN_MRST)	SSC0_MTSR (SSC0_SERIAL_DATA_OUT_MTSR)	SSC0_MRST (SSC0_SERIAL_DATA_OUT_MRST)
Bit 2	EXT_IRQ_0		'0'	
Bit 3			MAFE_HC1	
Bit 4			MAFE_DOUT	
Bit 5	MAFE_DIN		ILC_REMOTE_INTERRUPT_OUT[2]	
Bit 6	MAFE_FS		ILC_REMOTE_INTERRUPT_OUT[1]	
Bit 7	MAFE_SCLK		ILC_REMOTE_INTERRUPT_OUT[0]	

*Note: 1 SSC0 being intended for I^2C support, MTSR and MRST are multiplexed. This is done under control of a configuration bit "*SSC0_MUX_SEL *(normally, '1' when this SSC is master, '0' when it is slave).*

2 Usage of MAFE_* and ILC_REMOTE_INTERRUPT_OUT*[*] as alternate functions of PIO2 bits [5:7] is exclusive.*

PIO 3

Table 4: PIO 3 alternate functions

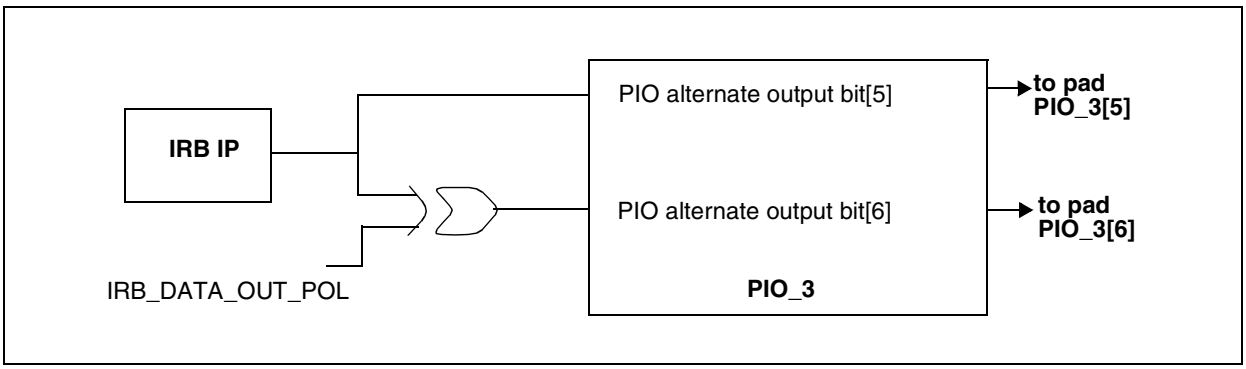
Port 3 Bit	PIO pin functions			
	Alternate function input		Alternate function output	
			SSC1_MUX_SEL = 0	SSC1_MUX_SEL = 1
Bit 0	SSC1_SCL (SSC1_SERIAL_CLOCK_IN)		SSC1_SCL (SSC1_SERIAL_CLOCK_OUT)	
Bit 1	SSC1_MTSR (SSC1_SERIAL_DATA_IN_MTSR)	SSC1_MRST (SSC1_SERIAL_DATA_IN_MRST)	SSC1_MTSR (SSC1_SERIAL_DATA_OUT_MTSR)	SSC1_MRST (SSC1_SERIAL_DATA_OUT_MRST)
Bit 2	EXT_IRQ_1		'0'	
Bit 3	IRB_IR_IN		'0'	
Bit 4	IRB_UHF_IN		'0'	
Bit 5			IRB_IR_DATA_OUT	
Bit 6			IRB_IR_DATA_OUT_OD	
Bit 7	PWM_CAPTURE0		EMI_BUS_GNT	

Note: 1 SSC1 being intended for I^2C support, MTSR and MRST are multiplexed. This is done under control of a configuration bit 'ssc1_mux_sel (normal, '1' when this SSC is master, '0' when it is slave).

2 Usages of EMI_BUS_GNT and PWM_CAPTURE0 as alternate functions of PIO2 bit [7] are all mutually exclusive.

3 *PIO3[5], when used as alternate function output 'IRB_IR_DATA_OUT', is normally programmed to behave as normal push-pull output, whilst PIO3[6] is normally programmed to behave as an open-drain output (using PIO configuration bits). Both may be required at the same time on the application. The polarity of IRB_DATA_OUT_OD is user programmable using configuration bit* 1RB_DATA_OUT_OD_POL. *See Figure 12 below.*

Figure 12: IRB_DATA_OUT_OD polarity configuration



PIO 4

Table 5: PIO4 alternate functions

Port 4 Bit	PIO pin functions			
	Alternate function input		Alternate function output	
			SSC2_MUX_SEL = 0	SSC2_MUX_SEL = 1
Bit 0	SSC2_SCL (SSC2_SERIAL_CLOCK_IN)		SSC2_SCL (SSC2_SERIAL_CLOCK_OUT)	
Bit 1	SSC2_MTSR (SSC2_SERIAL_DATA_IN_MTSR)	SSC2_MRST (SSC2_SERIAL_DATA_IN_MRST)	SSC2_MTSR (SSC2_SERIAL_DATA_OUT_MTSR)	SSC2_MRST (SSC2_SERIAL_DATA_OUT_MRST)
Bit 2	SSC2_MRST (SSC2_SERIAL_DATA_IN_MRST)		SSC2_MRST (SSC2_SERIAL_DATA_OUT_MRST)	
Bit 3	UART2_RXD		'0'	
Bit 4			UART2_TXD	
Bit 5	UART2_CTS		PWM_OUT_B	
Bit 6			UART2_RTS	
Bit 7			C1_PWM_OUT0	

Note: 1 When SSC2 is intended for I^2C support, MTSR and MRST may be multiplexed. This is done under control of a configuration bit 'ssc2_mux_sel (normally, '1' when this SSC is master, '0' when it is slave). However PIO4 retains the ability to separate MTSR from MRST for full duplex communication support.

2 C1_PWM_OUT0 is the PWM output of the PWM module present in the ST20-C1 macro-cell, PWM_OUT_B is the second PWM output of the PWM-TImer2 (also routed to internal ILC and normally used as long period timer to produce periodic interrupt requests).

3 Since 'SSC2_SERIAL_DATA_IN_MRST' can come either from pad PIO_4[1] or from pad PIO_4[2], a mux controlled by 'SSC2_MUX_SEL' is required in the COMMs wrapper to perform this selection.

PIO 5

Table 6: PIO5 alternate functions

Port 5 Bit	PIO pin functions			
	Alternate function input		Alternate function output	
			SSC3_MUX_SEL = 0	SSC3_MUX_SEL = 1
Bit 0			UART3_TXD	
Bit 1	UART3_RXD		'0'	
Bit 2	UART3_CTS		'0'	
Bit 3			UART3_RTS	
Bit 4			PWM_OUT_A	
Bit 5	SSS3_SCL (SSC3_SERIAL_CLOCK_IN)		SSC3_SCL (SSC3_SERIAL_CLOCK_OUT)	
Bit 6	SSC3_MTSR (SSC3_SERIAL_DATA_IN_MTSR)	SSC3_MRST (SSC3_SERIAL_DATA_IN_MRST)	SSC3_MTSR (SSC3_SERIAL_DATA_OUT_MTSR)	SSC3_MRST (SSC3_SERIAL_DATA_OUT_MRST)
Bit 7	SSC3_MRST (SSC3_SERIAL_DATA_IN_MRST)		SSC3_MRST (SSC3_SERIAL_DATA_OUT_MRST)	

Note: 1 When SSC3 is intended for I^2C support, MTSR and MRST may be multiplexed. This is done under control of a configuration bit 'ssc3_mux_sel (normally, '1' when this SSC is master, '0' when it is slave). However PIO5 retains the ability to separate MTSR from MRST for full duplex communication support.

2 PWM_OUT_A is the main PWM output of the PWM-TImer2,

3 Since 'SSC3_SERIAL_DATA_IN_MRST' can come either from pad PIO_5[6] or from pad PIO_5[7], a mux controlled by 'SSC3_MUX_SEL' is required in the COMMs wrapper to perform this selection.

4 Connections

This chapter indicates the STi7710 ball functions and details which type of I/O they connect to internally.

The I/O column uses the following key:

In/Out/In Out/PS: Digital input pin/output pin/bidirectional pin (in functional mode).

Ana: Analog connection.

PU/PD: Integrated weak pull-up/pull-down resistor on the input pin.

PUC/PDC: Pull-up or pull-down capable input pin (that is, pull-up/pull-down can be enabled or disabled by software).

OD: Open drain behavior.

ODC: Open drain capable (that is, open drain behavior can be enabled/disabled by software).

xxmA: Drive strength for output pins: 2 mA, 4 mA or 8 mA. This is the current that the I/O can continuously drive, peak currents during I/O switching are much higher.

5VT: 5 V tolerant digital I/O. All digital I/Os are 3.3 V capable, only a selected set can sustain 5 V when input or when tri-stated (note they sustain, but cannot drive, 5 V).

All digital input pins integrate a Schmitt trigger.

Power supplies dedicated to a particular block or function appear in the table describing pins pertaining to that function, other (that is, shared) power supplies are all grouped in a shared Table 7 and Table 8

Table 7: Digital power supplies

Ball position	Name	Function	I/O
C8	VDD_RING	1.2 V power supply for the digital core. All those nets are shorted internally via a ring on the BGA substrate.	PS
D6			
E6			
E7			
E12			
E14			
E18			
E21			
L5			
M4			
M5			
R5			
T5			
AB6			
AB7			
AB8			
AB9			
AB10			

Table 7: Digital power supplies

Ball position	Name	Function	I/O
L11	GND_RING	Ground for the digital core. All those nets are shorted internally via a ring on the BGA substrate.	PS
M11			
N11			
P11			
R11			
T11			
L12			
M12			
N12			
P12			
R12			
T12			
L13			
M13			
N13			
P13			
R13			
T13			
L14			
M14			
N14			
P14			
R14			
T14			
L15			
M15			
N15			
P15			
R15			
T15			
L16			
M16			
N16			
P16			
R16			
T16			

Table 7: Digital power supplies

Ball position	Name	Function	I/O
E5	VDDE3V3_RING	3.3 V power supply for the digital I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
F5			
G5			
K5			
N5			
U5			
V5			
Y5			
W5			
AA5			
AB5			
AB12			
H5	GNDE3V3_RING	Ground for the digital I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
J5			
P5			
AB11			
H22	VDDE_DIG2_RING	3.3 V power supply for the digital I/Os. All those nets are shorted internally via a ring on the BGA substrate. Same role as VDDE3V3_RING for different set of I/Os.	PS
J22	GNDE_DIG2_RING	Ground for the digital I/Os. All those nets are shorted internally via a ring on the BGA substrate.Same role as GNDE3V3_RING for different set of I/Os.	PS
K22			
Y22	VDD_DIG3_RING	1.2 V power supply for the digital core. All those nets are shorted internally via a ring on the BGA substrate.Same role as VDD_RING.	PS
W22	VDDE_DIG3_RING	3.3 V power supply for the digital I/Os. All those nets are shorted internally via a ring on the BGA substrate. Same role as VDDE3V3_RING for different set of I/Os.	PS
AA22	GNDE_DIG3_RING	Ground for the digital I/Os. All those nets are shorted internally via a ring on the BGA substrate. Same role as GNDE3V3_RING for different set of I/Os.	PS
AB21			
AB22			
E8	VDDE2V5_RING	2.5 V power supply dedicated to LMI I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
E11			
E15			
E17			
E20			
C14			

Table 7: Digital power supplies

Ball position	Name	Function	I/O
E9	GNDE2V5_RING	Ground for the LMI I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
E10			
E13			
E16			
E19			
E22			
F22			
G22			
M3	AF_VDD0	Independent 3 v 3 power supplies for internal AntiFuse blocks.	PS
M2	AF_VDD1		
N4	AF_VDD2		
N3	AF_VDD3		
N2	AF_VDD4		

Table 8: Shared analog power supplies

Ball position	Name	Function	I/O
AB18	VDD_HDMI_RING	1.2 V power supply for the HDMI PHY and rejection PLL. All those nets are shorted internally via a ring on the BGA substrate.	PS
AB13	GND_HDMI_RING	Ground for the HDMI PHY and rejection PLL. All those nets are shorted internally via a ring on the BGA substrate.	PS
V22			
AB19	VDDE_HDMI_RING	3.3 V power supply for the HDMI I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
AB20			
AB14	GNDE_HDMI_RING	Ground for the HDMI I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
AB15			
AB16			
AB17			
AC15			
R22	VDD_ANA_RING	1.2 V power supply used by DACs and frequency synthesizers. All those nets are shorted internally via a ring on the BGA substrate.	PS
N22	GND_ANA_RING	Ground used by DACs and frequency synthesizers. All those nets are shorted internally via a ring on the BGA substrate.	PS
P22			
T22	VDDE_ANA_RING	3.3 V power supply used by crystal oscillator and DAC I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
U22	GNDE_ANA_RING	Ground used by crystal oscillator and DAC I/Os. All those nets are shorted internally via a ring on the BGA substrate.	PS
L22	VDD_USB_RING	1.2 V power supply for USB PHY and USB PLL.	PS
M22	GNDE_USB_RING	Ground for USB I/Os.	PS

Table 9: System services

Ball position	Name	Function	I/O	Reset	
				State	Value
E23	NOT_RESET	Active low system reset	In	I	0
D24	EXT_nRESET_OUT	Combination of internal reset sources (including watchdog)	Out, OD, 4mA	O	0
C24	DCU_TRIGGERIN	External trigger input to ST20_C1	In	I	0
A25	DCU_TRIGGEROUT	Signal to trigger external debug circuitry	Out, 4mA	O	0
A24	EXT_IRQ2	External Interrupt source	In	I	L
D23	EXT_IRQ3	External Interrupt source	In	I	L

Note: EXT_IRQ0 and EXT_IRQ1 are on PIO's.

Table 10: Clocks

Ball position	Name	Function	I/O	Reset	
				State	Value
J25	XTAL1	27 MHz crystal connection or external clock input	Ana, specific	I	0
K24	XTAL2	27 MHz crystal connection	Ana, specific	O	
F23	AUXCLKOUT	Auxiliary clock output (copy of any of the internally generated clocks)	Out, 4mA	O	0
E24	PIXCLK	Pixel clock output (for the digital video output)	Out, 4mA	O	1

Table 11: Frequency synthesizers/PLLs

Ball position	Name	Function	I/O
K26	FS0_VDDD	Adjust frequency synthesizer 1.2 V digital power	PS
J26	FS0_GNDD	Adjust frequency synthesizer digital ground	PS
L23	FS0_VCCA	Adjust frequency synthesizer 1.2 V analog power	PS
L24	FS0_GNDA	Adjust frequency synthesizer analog ground	PS
L26	FS1_VDDD	QUAD frequency synthesizer 1.2 V digital power	PS
M26	FS1_GNDD	QUAD frequency synthesizer digital ground	PS
M23	FS1_VCCA	QUAD frequency synthesizer 1.2 V analog power	PS
M24	FS1_GNDA	QUAD frequency synthesizer analog ground	PS
K25	FS2_VDDD	Audio frequency synthesizer 1.2 V digital power	PS
L25	FS2_GNDD	Audio frequency synthesizer digital ground	PS
M25	FS2_VCCA	Audio frequency synthesizer 1.2 V analog power	PS
N26	FS2_GNDA	Audio frequency synthesizer analog ground	PS

Table 11: Frequency synthesizers/PLLs

Ball position	Name	Function	I/O
PLL dedicated to LMI			
B24	PLL_LMI_VDDD	LMI PLL 1.2 V digital supply	PS
C23	PLL_LMI_GNDD	LMI PLL digital ground	PS
D21	PLL_LMI_VCCA	LMI PLL 1.2 V analog supply	PS
D22	PLL_LMI_GNDA	LMI PLL analog ground	PS
PLL dedicated to USB			
F26	PLL_USB_VDDD	USB PLL 1.2 V digital supply	PS
F25	PLL_USB_GNDD	USB PLL digital ground	PS
E26	PLL_USB_VDDA	USB PLL 3.3 V analog supply	PS
D26	PLL_USB_GNDA	USB PLL analog ground	PS
Rejection PLL dedicated to TMDS			
AE23	PLL_TMD_VDDD	TMDS (HDMI PHY) 1.2 V digital supply	PS
AF24	PLL_TMD_GNDD	TMDS (HDMI PHY) digital ground	PS
AC21	PLL_TMD_VDDA	TMDS (HDMI PHY) 1.2 V analog supply	PS
AC20	PLL_TMD_GNDA	TMDS (HDMI PHY) analog ground	PS
Clockgen DLL power pads			
A26	CG_DLL_VDD	Clockgen DLL power 1.2 V	PS
B25	CG_DLL_VSS	Clockgen DLL ground	PS

Table 12: Test access port (JTAG)

Ball position	Name	Function	I/O	Reset	
				State	Value
F24	TDI	JTAG test data input	In	I	0
D25	TDO	JTAG test data output	Out, 2mA	O	Z
G23	TMS	JTAG test mode select	In	I	0
C26	TCK	JTAG test clock	In	I	0
E25	TRST	JTAG test logic reset (active low as per JTAG specification)	In	I	0

Table 13: Transport stream

Ball position	Name	Function	I/O	Reset	
				State	Value
V1	TSDATA0[7]	MPEG transport stream TS0 data input	In	I	U
U4	TSDATA0[6]				
V2	TSDATA0[5]				
W1	TSDATA0[4]				
V3	TSDATA0[3]				
W2	TSDATA0[2]				
V4	TSDATA0[1]				
Y1	TSDATA0[0]				
U3	TSBYTECLK0	MPEG transport stream TS0 clock	In	I	U

Table 13: Transport stream

Ball position	Name	Function	I/O	Reset	
				State	Value
U1	TSPKTCLK0	MPEG transport stream TS0 packet clock (packet sync)	In	I	U
T1	TSPKTERR0	MPEG transport stream TS0 error indicator for the transport packet	In	I	U
T3	TSVALID0	MPEG transport stream TS0 indicates valid data	In	I	U
Y3	TSDATA1[7]	MPEG transport stream TS1 data input	In	I	U
AA2	TSDATA1[6]				
AB1	TSDATA1[5]				
Y4	TSDATA1[4]				
AA3	TSDATA1[3]				
AB2	TSDATA1[2]				
AC1	TSDATA1[1]				
AA4	TSDATA1[0]				
See above: TSDATA [7:0]	DYCI[7:0]	Digital YCbCr input (ITU-R 656/ ‘D1’)			
U2	TSBYTECLK1	MPEG transport stream TS1 clock	In	I	U
	DCKI	27 Mhz pixel clock input (ITU-R 656/ ‘D1’)			
T4	TSPKTCLK1	MPEG transport stream TS1 packet clock	In	I	U
R4	TSPKTERR1	MPEG transport stream TS1 error indicator for the transport packet	In	I	U
	DHSI	Horizontal sync input			
T2	TSVALID1	MPEG transport stream TS1 indicates valid data	In	I	U
	DVSI	Vertical sync input			
N1	TSDATA2[7]	MPEG transport stream TS2 data input or output	InOut, 4mA	I	Z
P4	TSDATA2[6]				
P3	TSDATA2[5]				
P2	TSDATA2[4]				
P1	TSDATA2[3]				
R1	TSDATA2[2]				
R2	TSDATA2[1]				
R3	TSDATA2[0]				
W4	TSBYTECLK2	MPEG transport stream TS2 clock input or output	InOut, 4mA	I	Z
AA1	TSPKTCLK2	MPEG transport stream TS2 packet clock input or output	InOut, 4mA		
Y2	TSPKTERR2	MPEG transport stream TS2 error indicator input or output	InOut, 4mA		
W3	TSVALID2	MPEG transport stream TS2 indicates valid data input or output	InOut, 4mA		

Note: Digitized video input / "D1 input" (YcbCr + clock + optional syncs) is shared with TS1 - usage is mutually exclusive.

Table 14: Digitized video output

Ball position	Name	Function	I/O	Reset	
				State	Value
AB23	DHSO	Horizontal sync output	Out, 4mA	O	0
AC24	DVSO	Vertical sync output	Out, 4mA	O	0
AE14	DYCIO[15]	Digital YCbCr In HD mode, output format is 4:2:2, with DYCIO[15:8] - Y output DYCIO[7:0] - Multiplexed Cb/CR output In SD mode, output format is 4:2:2, with DYCIO[15:8] - Y/Cb/Cr output DYCIO[7:0] - not used	Out, 4mA	O	0
AD14	DYCIO[14]				
AC14	DYCIO[13]				
AF15	DYCIO[12]				
AE15	DYCIO[11]				
AD15	DYCIO[10]				
AD22	DYCIO[9]				
AF25	DYCIO[8]				
AE24	DYCIO[7]				
AD23	DYCIO[6]				
AC22	DYCIO[5]				
AF26	DYCIO[4]				
AE25	DYCIO[3]				
AD24	DYCIO[2]				
AC23	DYCIO[1]				
AD25	DYCIO[0]				
AE26	NOT_CLAMP	Active low clamp signal	Out, 4mA	O	0

Functional video outputs of the STi7710 are through the SD and HD DACs and the HDMI output.

Table 15: Analog video output

Ball position	Name	Function	I/O
Auxiliary (SD) triple DAC			
W26	AUX_Y_R	Y (S-VHS) or Red Component SD Video Output	Ana
AA25	AUX_C_G	C (S-VHS) or Green Component SD Video Output	Ana
Y24	AUX_CVBS_B	Composite CVBS or Blue Component SD Video ouput	Ana
AB26	SD_AVDD	3.3 V analog power supply	PS
AA26	SD_AGND	Analog ground supply	PS
Y26	SD_REXT	Connect an external precision resistor, nom. 10K (1%), between these 2 ports with SD_GNDAS_REXT also tied to Analog Ground.	Ana
Y25	SD_GNDAS_REXT		Ana
Main (typ. HD) triple DAC			
W24	R_Pr_Cr	First component of Main (HD) Analog Video Output (RGB or YPbPr or YCbCr)	Ana
T25	G_Y_Y	Second component of Main (HD) Analog Video Output (RGB or YPbPr or YCbCr)	Ana
V26	B_Pb_Cb	Third component of Main (HD) Analog Video Output (RGB or YPbPr or YCbCr)	Ana
W23	IDUMPR	Ground - current return path for the DAC red outputs	PS
T24	IDUMPG	Ground - current return path for the DAC green outputs	PS
V25	IDUMPB	Ground - current return path for the DAC blue outputs	PS
V24	VCCA2	3.3 V power supply for digital blocks	PS
W25	GNDA2	Ground for digital blocks	PS
U23	VCCA_BLUE	3.3 V analog blue matrix power supply	PS
T26	VCCA_GREEN	3.3 V analog green matrix power supply	PS
V23	VCCA_RED	3.3 V analog red matrix power supply	PS
T23	VCCA1	3.3 V analog power supply for DAC biasing	PS
U24	GNDA1	Analog ground for DAC biasing	PS
U26	REXT	Connect an external precision resistor, nom. 2.4K (1%), between these 2 ports with MASS_QUIET also tied to clean Analog Ground.	Ana
U25	MASS_QUIET		PS

Table 16: Audio digital interface

Ball position	Name	Function	I/O	Reset	
				State	Value
AA24	ASIN	Serial audio input channel (I^2S interface)	In	I	H
AC26	ASTRB	Strobe for serial audio input channel (I^2S interface)	In	I	1
Y23	ALRCLKIN	Left/right channel select input (I^2S interface)	In	I	0
AC25	SPDIFO	IEC60958 - IEC61937 digital output (S/PDIF)	Out, 4mA	O	Z
AA23	PCMCLK	PCM clock in/out. If input can be used in place of PCM clock generated by internal FS. If output can be used to observe internal PCM clock.	InOut, 4mA	O	H
AD26	SCLK	clock output	Out, 4mA	O	0

Table 16: Audio digital interface

Ball position	Name	Function	I/O	Reset	
				State	Value
AB25	LRCLK	left/right clock output	Out, 4mA	O	0
AB24		Reserved			

Table 17: Audio analog interface

Ball position	Name	Function	I/O
P24	OUTPR	Right differential analog output +	Ana
P23	OUTMR	Right differential analog output -	Ana
R26	OUTPL	Left differential analog output +	Ana
R25	OUTML	Left differential analog output -	Ana
R23	VDDA	3.3 V analog power supply	PS
R24	GNDA	Analog ground	PS
P25	VDD3	3.3 V power supply for digital to analog interface	PS
P26	VSS3	Ground for digital to analog interface	PS
N24	SUBANA	Ring isolation - connect to ground	PS
N25	IREF	Output current reference	Ana
N23	VBG	Output band gap voltage	Ana

Table 18: Programmable input/output

Ball position	Name	Function	I/O	Reset	
				State	Value
J2	PIO0[7]	Programmable input/output pins	InOut, PUC, ODC, 4 mA, 5 VT	I	H
K3	PIO0[6]				
J1	PIO0[5]				
L4	PIO0[4]				
K2	PIO0[3]				
K1	PIO0[2]				
L3	PIO0[1]				
L2	PIO0[0]				
G2	PIO1[7]	Programmable input/output pins	InOut, PUC, ODC, 4 mA, 5 VT	I	H
H3	PIO1[6]				
J4	PIO1[5]				
G1	PIO1[4]				
H2	PIO1[3]				
J3	PIO1[2]				
H1	PIO1[1]				
K4	PIO1[0]				

Table 18: Programmable input/output

Ball position	Name	Function	I/O	Reset	
				State	Value
E2	PIO2[7]	Programmable input/output pins	InOut, PUC, ODC, 4 mA, 5 VT	I	H
F3	PIO2[6]				
G4	PIO2[5]				
E1	PIO2[4]				
F2	PIO2[3]				
G3	PIO2[2]				
H4	PIO2[1]				
F1	PIO2[0]				
B1	PIO3[7]	Programmable input/output pins	InOut, PUC, ODC, 4 mA, 5 VT	I	H
E4	PIO3[6]				
D3	PIO3[5]				
C1	PIO3[4]				
D2	PIO3[3]				
E3	PIO3[2]				
F4	PIO3[1]				
D1	PIO3[0]				
C4	PIO4[7]	Programmable input/output pins	InOut, PUC, ODC, 4 mA, 5 VT	I	H
A2	PIO4[6]				
B3	PIO4[5]				
A1	PIO4[4]				
C3	PIO4[3]				
D4	PIO4[2]				
B2	PIO4[1]				
C2	PIO4[0]				
C6	PIO5[7]	Programmable input/output pins	InOut, PUC, ODC, 4 mA, 5 VT	I	H
A5	PIO5[6]				
B5	PIO5[5]				
C5	PIO5[4]				
A4	PIO5[3]				
B4	PIO5[2]				
D5	PIO5[1]				
A3	PIO5[0]				

Note: At reset, pads that are in input mode with internal pull-up are all 5 V tolerant

Please refer to Chapter 3: *Pin list and alternative functions on page 31* for alternate functions.

Table 19: Local memory interface (LMI) for DDR SDRAM

Ball position	Name	Function	I/O	Reset	
				State	Value
C13	LMI_ADDR[14]	LMI page/column address bus	Out	O	0
D13	LMI_ADDR[13]				
D14	LMI_ADDR[12]				
D12	LMI_ADDR[11]				
C10	LMI_ADDR[10]				
D10	LMI_ADDR[9]				
C11	LMI_ADDR[8]				
D11	LMI_ADDR[7]				
C12	LMI_ADDR[6]				
C15	LMI_ADDR[5]				
D15	LMI_ADDR[4]				
C16	LMI_ADDR[3]				
D16	LMI_ADDR[2]				
B14	LMI_DATA[31]	LMI 32 bits data bus	InOut	I	Z
A14	LMI_DATA[30]				
B13	LMI_DATA[29]				
A13	LMI_DATA[28]				
A12	LMI_DATA[27]				
B11	LMI_DATA[26]				
A11	LMI_DATA[25]				
B10	LMI_DATA[24]				
B23	LMI_DATA[23]				
A23	LMI_DATA[22]				
B22	LMI_DATA[21]				
A22	LMI_DATA[20]				
A21	LMI_DATA[19]				
B20	LMI_DATA[18]				
A20	LMI_DATA[17]				
B19	LMI_DATA[16]				
A10	LMI_DATA[15]				
B9	LMI_DATA[14]				
A9	LMI_DATA[13]				
B8	LMI_DATA[12]				
B7	LMI_DATA[11]				
A7	LMI_DATA[10]				
B6	LMI_DATA[9]				
A6	LMI_DATA[8]				
A19	LMI_DATA[7]				
B18	LMI_DATA[6]				

Table 19: Local memory interface (LMI) for DDR SDRAM

Ball position	Name	Function	I/O	Reset	
				State	Value
A18	LMI_DATA[5]	LMI 32 bits data bus	InOut	I	Z
B17	LMI_DATA[4]				
A17	LMI_DATA[3]				
B16	LMI_DATA[2]				
A16	LMI_DATA[1]				
B15	LMI_DATA[0]				
B12	LMI_notDQS[3]	DDR data strobes	Out	I	0
B21	LMI_notDQS[2]				
A8	LMI_notDQS[1]				
A15	LMI_notDQS[0]				
C9	LMI_notDQM[3]	DDR data masks	Out	O	0
C22	LMI_notDQM[2]				
C7	LMI_notDQM[1]				
C20	LMI_notDQM[0]				
C17	LMI_notBANK[1]	Bank select	Out	O	0
D17	LMI_notBANK[0]				
D20	LMI_RDNOTWR	Read, not write strobe	Out	O	1
D18	LMI_notCS[1]	Chip select	Out	O	1
C18	LMI_notCS[0]				
C19	LMI_notRAS	Row address strobe	Out	O	1
D19	LMI_notCAS	Column address strobe	Out	O	1
D9	LMI_CLKEN	Clock enable	Out	O	0
D7	LMI_CLK	Clock	Out	O	1
D8	LMI_notCLK	Clock	Out	O	1
C21	LMI_VREF	DDR reference voltage	PS	I	Z

Table 20: External memory interface (EMI)

Ball position	Name	Function	I/O	Reset	
				State	Value
AF7	EMI_ADDR[23]	Address bus Bit[1] connects to address LSB of 16-bit memory devices. For 8-bit memory devices, also use EMI_ADDR[0] (muxed with EMI_nBE[1], see below).	Out, 4 mA	O	Z
AC9	EMI_ADDR[22]				
AE8	EMI_ADDR[21]				
AF8	EMI_ADDR[20]				
AD9	EMI_ADDR[19]				
AE9	EMI_ADDR[18]				
AC10	EMI_ADDR[17]				
AF9	EMI_ADDR[16]				
AD10	EMI_ADDR[15]				
AE10	EMI_ADDR[14]				
AF10	EMI_ADDR[13]				
AC11	EMI_ADDR[12]				
AD11	EMI_ADDR[11]				
AE11	EMI_ADDR[10]				
AF11	EMI_ADDR[9]				
AC12	EMI_ADDR[8]				
AD12	EMI_ADDR[7]				
AF12	EMI_ADDR[6]				
AF13	EMI_ADDR[5]				
AD13	EMI_ADDR[4]				
AC13	EMI_ADDR[3]				
AE13	EMI_ADDR[2]				
AF14	EMI_ADDR[1]				
AD1	EMI_DATA[15]	Data bus	InOut, 4 mA	I	Z
AB4	EMI_DATA[14]				
AC3	EMI_DATA[13]				
AD2	EMI_DATA[12]				
AE1	EMI_DATA[11]				
AF1	EMI_DATA[10]				
AE2	EMI_DATA[9]				
AD3	EMI_DATA[8]				
AC4	EMI_DATA[7]				
AF2	EMI_DATA[6]				
AE3	EMI_DATA[5]				
AD4	EMI_DATA[4]				
AC5	EMI_DATA[3]				
AF3	EMI_DATA[2]				
AE4	EMI_DATA[1]				
AD5	EMI_DATA[0]				
AF4	EMI_nOE	Output enable strobe	Out, 4 mA	O	Z
	PCC_OE#	PC card interface - output enable			

Table 20: External memory interface (EMI)

Ball position	Name	Function	I/O	Reset	
				State	Value
AF6	EMI_nBE[0]	Low order byte enable strobe	Out, 4 mA	O	Z
	PCC_IOWR#	PC card Interface - IO write			
AD7	EMI_nBE[1]	High order byte enable strobe for 16 bit memory devices	Out, 4 mA	O	Z
	EMI_ADDR[0]	Address LSB for 8 bit memory devices			
AC6	EMI_RDnWR	Read / notWrite strobe	Out, 4 mA	O	Z
AE7	EMI_nLBA	Load burst address for burst flash	Out, 4 mA	O	Z
	PCC_WE#	PC card interface - write enable			
AE5	EMI_nCS[4]	Chip select per EMI bank	Out, 4 mA	O	Z
AD6	EMI_nCS[3]				
AC7	EMI_nCS[2]				
AF5	EMI_nCS[1]				
AE6	EMI_nCS[0]				
AC2	EMI_WAIT	Extends the current read/write cycle	In	I	0
AB3	EMI_PRTSZ_8NOT16	Bus width for Boot ROM or flash (static) Tie low for 16 bit boot memory device, high for 8 bit	In	I	0
AD8	EMI_CLKF	Burst flash clock	Out, 4 mA	O	0
AC8	EMI_nBAA	Burst address advance for AMD burst flash - NC if ST or Intel burst flash is used.	Out, 4 mA	O	L
	PCC_IORD#	PC card interface			
	SEL_LONG_RESET	If pulled up at hardware reset, the internal reset will be stretched by about 200 ms. If pulled low, it is not affected.			
AE12	EMI_BUS_REQ	External host bus mastership request	In	I	0

Note: 1 Pins EMI_nOE, EMI_nBE[0], EMI_nBE[1], EMI_nLBA, EMI_nBAA are shared with other signals in some configurations. The alternate function appears under the main function in the above table.

2 All EMI outputs are tri-stated during reset and in case of external EMI bus mastership request.

Table 21: USB2.0 memory interface

Ball position	Name	Function	I/O
H26	USBDP	USB2 data+	Ana
G26	USBDN	USB2 data-	
G24	USB_RREF	Reference resistor connection	

Table 21: USB2.0 memory interface

Ball position	Name	Function	I/O
J23	USB_VSSL	Ground supply for deserialization flip flop	PS
H24	USB_VDDL	1.2 V power supply for deserialization flip flop	
G25	USB_VSSB	Ground supply for buffers	
J24	USB_VDDB	1.2 V power supply for buffers	
K23	USB_VSSC	Ground supply for DLL	
H25	USB_VDDC	1.2 V power supply for DLL	
H23	USB_VDD3	3.3 V power supply for USB1.1 FS compliance	

Table 22: External DMA pacing

Ball position	Name	Function	I/O	Reset	
				State	Value
C25	EXT_DMA_REQ1	External DMA request to FDMA	In	I	0
B26	EXT_DMA_REQ2			I	0

Table 23: TMDS pads

Ball position	Name	Function	I/O
AF23	TX0N	Differential pair for channel 0	Ana
AF22	TX0P		
AF19	TX1N	Differential pair for channel 1	
AF18	TX1P		
AF17	TX2N	Differential pair for channel 2	
AF16	TX2P		
AF21	TXCKN	Differential pair for pixel clock	
AF20	TXCKP		
AC19	HDMI_VDDC0	1.2 V power supply for the output buffers on each of the data channels and clock channel	PS
AC17	HDMI_VDDC1		
AC16	HDMI_VDDC2		
AC18	HDMI_VDDCK		
AE21	HDMI_VSSC0	Ground for the output buffers on each of the data channels and clock channel	
AD18	HDMI_VSSC1		
AD17	HDMI_VSSC2		
AE20	HDMI_VSSCK		
AD20	HDMI_VDDP	1.2 V power supply for the serializer and the DLL	
AD21	HDMI_VDDX		
AD19	HDMI_VSSP	Ground for the serializer and the DLL	
AE22	HDMI_VSSX		
AE16	HDMI_VDDSL	1.2 V power supply for the deserializer	
AE19	HDMI_VDDD		

Table 23: TMDS pads

Ball position	Name	Function	I/O
AE17	HDMI_VSSSL	Ground for the deserializer	PS
AE18	HDMI_VSSD		
AD16	COMPENSATION_ PAD	Used to compensate for process drift in the HDMI PHY. Connected to 3.3 V via 50 Ω resistor.	Ana (specific)

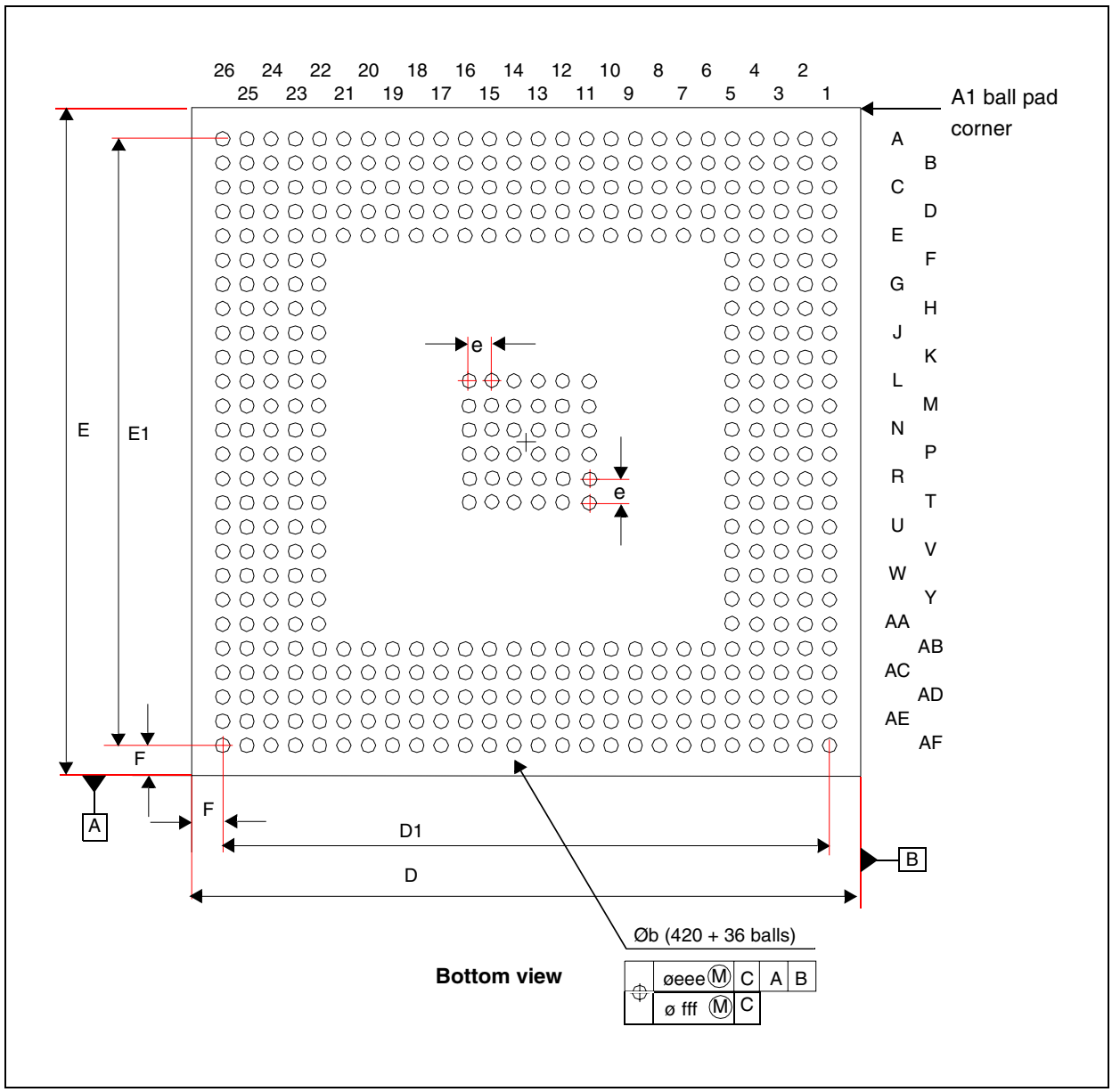
Table 24: DAA pad

Ball position	Name	Function	I/O
L1	DAA_C2A	Differential direst access arrangement mechanism for telephone connection	Ana (specific)
M1	DAA_C1A		

5 Package specifications

The STi7710 is packaged in a 420 + 36 pin plastic ball grid array (PBGA).

Figure 13: 456-pin BGA package



Note: 1 The terminal A1 corner is identified on the top surface by using a corner chamfer, ink or metallized markings, or other feature of package body or integral heatslug. There may also be a distinguishing feature on the bottom surface of the package to identify the terminal A1 corner.

2 The exact shape of each corner may vary.

Table 25 gives the values of the dimensions marked in Figure 14.

Figure 14: Package dimension

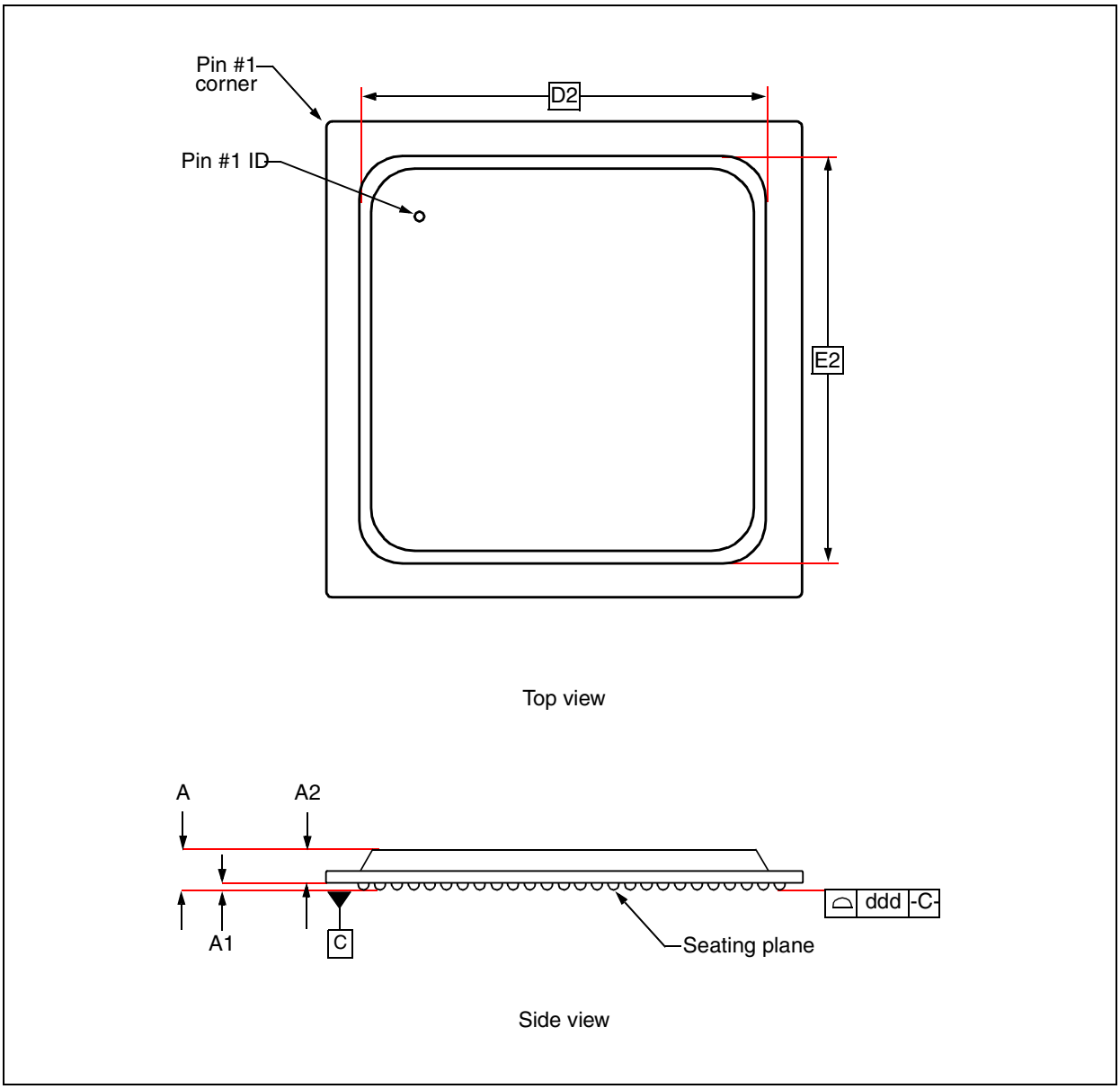


Table 25: Package dimensions, mm

Dimension	Minimum	Typical	Maximum
A			2.44
A1	0.4	0.5	0.6
A2		1.73	1.9
b	0.5	0.6	0.7
D	26.80	27.00	27.20
D1		25.00	
D2			
E	26.80	27.00	27.20
E1		25.00	
E2			
e		1.00	
F		1.00	
ddd			0.15
eee			0.25
fff			0.10

Reference document: JEDEC standard no 95 section 4.14 (Ball Grid Array Package Design Guide)

6 Configuration registers

The following functions are enabled and disabled using the configuration registers:

- alternative assignments on pins,
- smart card modes
- PIO alternate functions
- IRB modes
- DAA enable and configure
- Global power down

Addresses are provided as *TopConfigAddress* + offset.

The *TopConfigAddress* is:

0x2006 8000.

Table 26: Communications glue register summary

Register	Description	Offset	Type
COMMS_CFG_1	PIO alternate functions	0x04	R/W
COMMS_CFG_2	Smart card insertion	0x08	R/W
COMMS_CFG_3	DAA/IRB	0x0C	R/W
COMMS_PWR_CTRL	Communications power control	0x10	R/W

COMMS_CFG_1 PIO alternate functions

31	30	29	28	27	26	25	24	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																												SSC3_MUX_SEL	SSC2_MUX_SEL	SSC1_MUX_SEL	SSC0_MUX_SEL

Address: *TopConfigAddress* + 0x04

Type: R/W

Reset: 0

Description:

[31:4] **Reserved**

[3] **SSC3_MUX_SEL**
1: SSC3_MRST replaces SSC3_MTSR on the corresponding PIO alternate function.
0: the default assignment is applied.

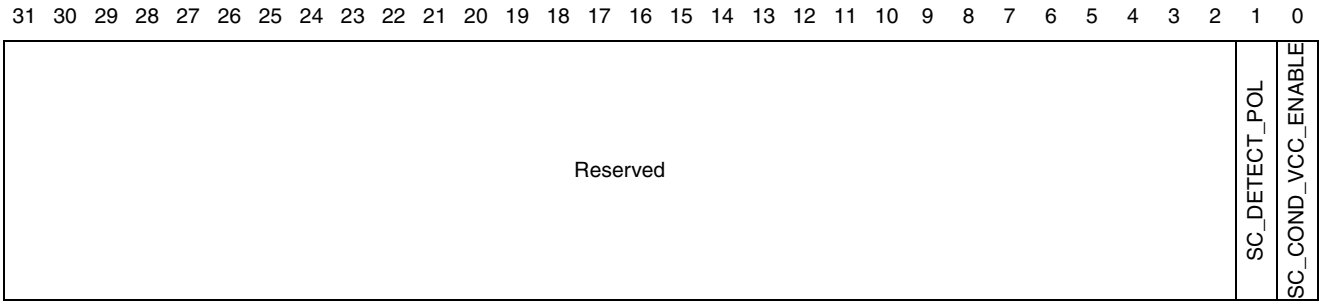
[2] **SSC2_MUX_SEL**
1: SSC2_MRST replaces SSC2_MTSR on the corresponding PIO alternate function.
0: the default assignment is applied.

[1] **SSC1_MUX_SEL**
1: SSC1_MRST replaces SSC1_MTSR on the corresponding PIO alternate function.
0: the default assignment is applied.

[0] **SSC0_MUX_SEL**
1: SSC0_MRST replaces SSC0_MTSR on the corresponding PIO alternate function.
0: the default assignment is applied.

Note: See also the PIO alternate function assignment, Figure 3: Pin list and alternative functions on page 31.

COMMS_CFG_2 Smart card insertion



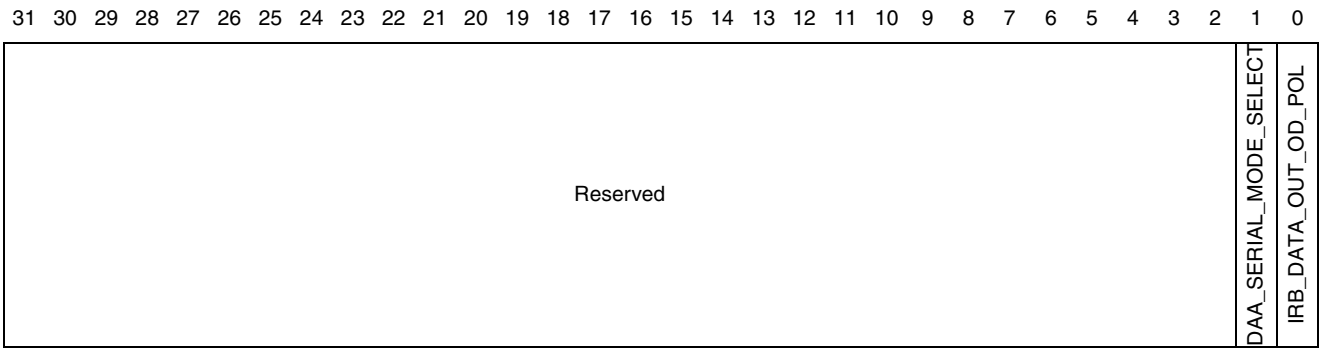
Address: *TopConfigAddress* + 0x08
Type: R/W
Reset: 0
Description:

[31:2] **Reserved**

[1] **SC_DETECT_POL**: Selection of polarity of input signal SC_DETECT.
If bit SC_COND_VCC_ENABLE is set:
1: Output pin SC_NOT_SETVCC is a copy of NOT(SC_DETECT).
0: Output pin SC_NOT_SETVCC is a copy of SC_DETECT.
If SC_COND_VCC_ENABLE is cleared then this bit has no effect.

[0] **SC_COND_VCC_ENABLE**
Enable control of NDS smart card VCC upon detection of smart card removal or insertion.
1: alternate PIO output pin SC_COND_VCC is controlled according to input SC_DETECT.
0: alternate PIO output pin SC_COND_VCC is tied to a constant value.

COMMS_CFG_3 DAA/IRB



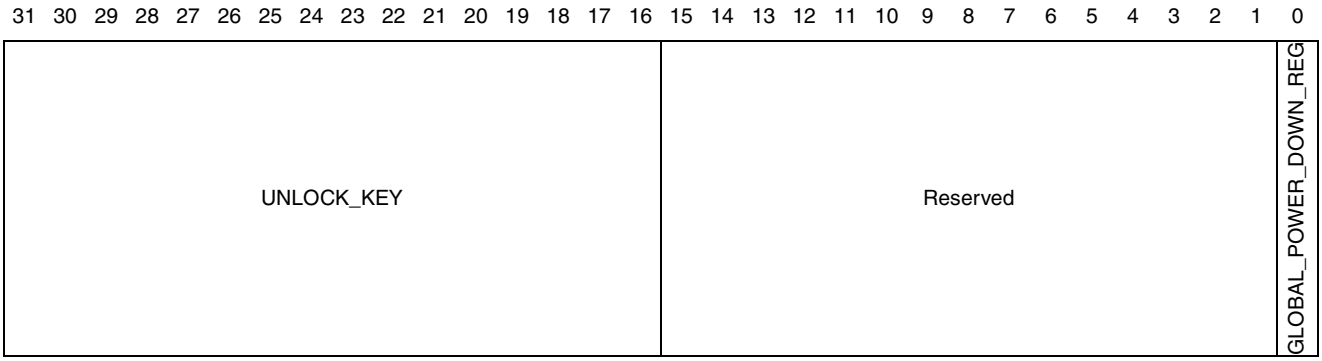
Address: *TopConfigAddress* + 0x000C
Type: R/W
Reset: 0
Description:

[31:2] **Reserved**

[1] **DAA_SERIAL_MODE_SELECT**

[0] **IRB_DATA_OUT_OD_POL**
Selection of polarity of IRB output signal routed as alternate function to PIO_3[6] (normally configured as open-drain).
1: IRB_DATA_OUT_OD has same polarity as IRB_DATA_OUT.
0: polarity of IRB_DATA_OUT_OD is inverted.

COMMS_PWR_CTRL Communications power control



Address: *TopConfigAddress* + 0x10

Type: Special

Reset: 0

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:1] **Reserved**

[0] **GLOBAL_POWER_DOWN_REG**
1: All clocks are switched to power down mode (their speed is divided by 1024).
This bit is automatically cleared upon wake up by interrupt from IRB.

7 Clocks

7.1 Microarchitecture

The ClockGen essentially consists of configuration registers and clock division. It is possible to switch any clock off, except the low power control clock (CLK_LPC) which permanently runs at 46.875 kHz (48 MHz/1024).

A power down mode is supported, whereby clocks have dramatically decreased speeds (see Chapter 9 on page 93). The intent is to cut power consumption significantly whilst still being able to access all blocks. In particular, the CPU still runs, and is able to reprogram the ClockGen to exit power saving mode.

7.2 Overview of operation

The root clock, CLK200_ROOT, runs at 200 MHz. This, together with a derived 100 MHz sub-clock is distributed throughout the chip through a balanced clock tree.

Timing synchronization throughout the chip is assured by use of a DLL, which compares clocks, and tunes the phase of CLK200_ROOT.

7.3 Clock requirements

The table below shows all STi7710 clocks, with source and target frequencies.

Table 27: Clock target frequencies and usages

Clock name	Source	Target frequency (MHz)	Sub-system connected
TSBYTECLK[2:0]	Pins: TSBYTECLK0, TSBYTECLK1, TSBYTECLK2		TSmerger
TCK	Pin TCK	50	Test mode controller (including JTAG)
CLK_LPC	ClockGen Division from FS1.4	0.046875 (46.875 kHz)	Low power controller
CLK_LMI_2X	ClockGen PLL output	400	LMI PADLOGIC
CLK_ST20	ClockGen Division from PLL	200	ST20-C105 CPU LMI Interconnect FDMA HDdisplay IDdisplay
CLK_SYS	ClockGen Division from PLL	100	Interconnect EMI DVP HDdisplay + LMU IDdisplay Blitter Compositor DENC MPEG decoder AUDIO PTI TSmerger FDMA HDMI + HDCP
CLK_USB	USB PLL	60	USB2.0
CLK_USB1	ClockGen. Division from FS1.4	48	USB1.1/1.0
CLK_PIX_2X	Pin DCKI	27	DVP DENC (PIXCLK_FROM_PAD)
CLK_PIXEL_HD / CLK_DAC_HD	FS0.4 output with (bypassable) cleaning by noise rejection PLL	148.5 or 108	Video HD DAC Rejection PLL
CLK_PIPE	ClockGen FS1.2 output	148.5 (nom)	HDdisplay (clk_pipe) IDdisplay (clk_pipe) MPEG decoder (clk_vd)
CLK_DISP_HD	ClockGen Programmable division from FS0.4	74.25 or 54	HDdisplay (clk_disp) Compositor (main_ck) VOS (formatter, VTG1, RGVtoYCbCr)
CLK_DISP_ID	ClockGen Programmable division from FS1.1	13.5 (max)	IDdisplay (clk_disp) Compositor (aux_ck) VOS

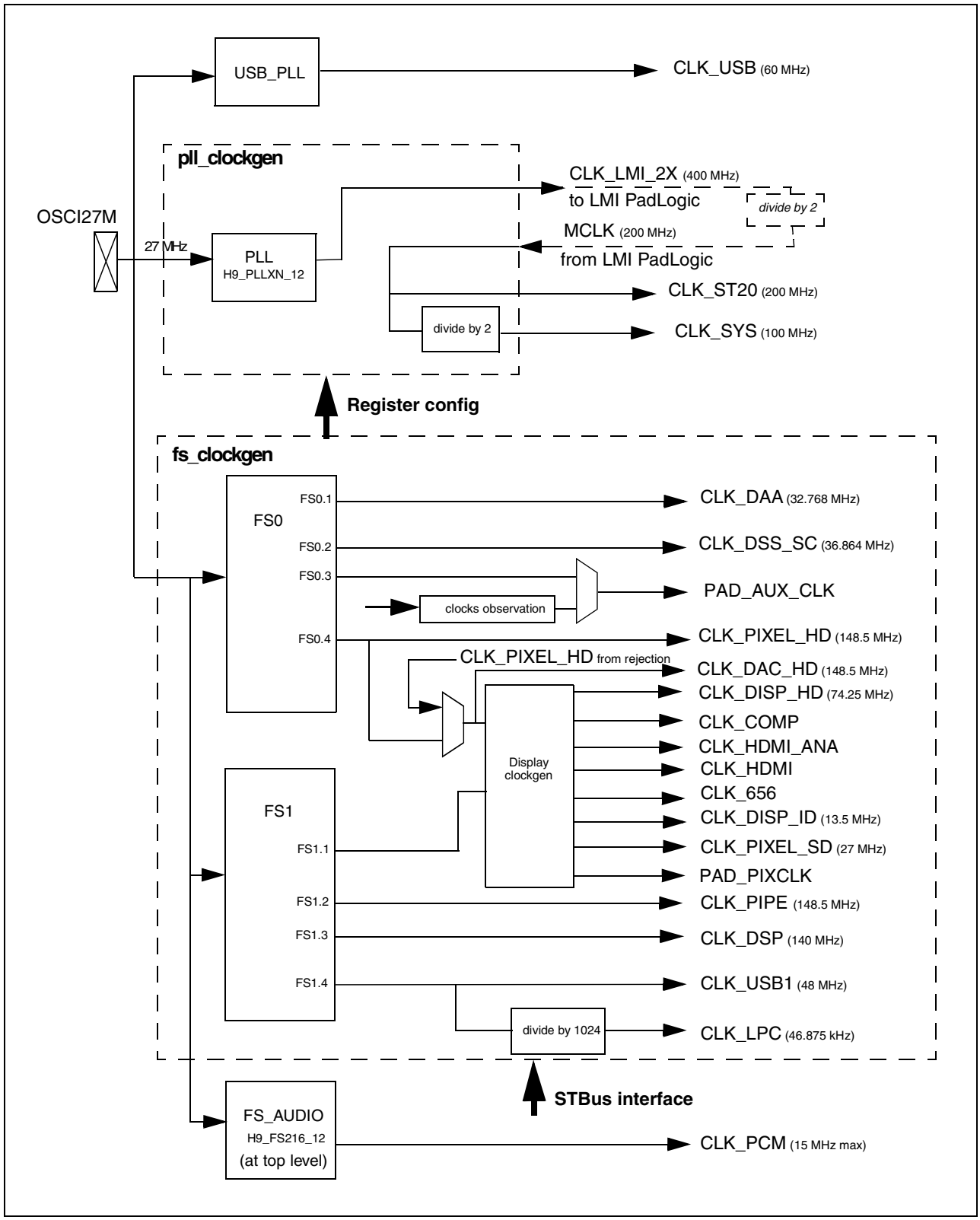
Table 27: Clock target frequencies and usages

Clock name	Source	Target frequency (MHz)	Sub-system connected
CLK_656	ClockGen Programmable division from FS0.4	54	VOS
CLK_HDMI / CLK_HDMI_ANA	ClockGen Programmable division from FS0.4	74.25 or 27	Analog HDMI(TMDS_clock) Digital HDMI(TMDS_clock)
CLK_COMP	ClockGen Programmable division from either FS0.4 or FS1.1 (depending on application)	74.25, 27 or 13.5	Compositor (gdp2_ck)
CLK_PIXEL_SD	ClockGen Programmable division from either FS1.1 or FS0.4	27	PTI (TIMER_CLK) DENC (PIX_CLK) Comms (TTX_CLOCK_27M)
PAD_CLK_PCM	Pin PCMCLK	15 (max)	AUDIO
CLK_PCM	Frequency synthesizer FS_Audio	15 (max)	AUDIO Digital HDMI (CKAU)
CLK_I2S	Pin ASTRB	15 (max)	AUDIO
CLK_DSP	ClockGen FS1.3 output	166.66	AUDIO
CLK_DAA	ClockGen FS0.1 output	32.768	Comms
CLK_DSS_SC	ClockGen FS0.2 output	36.864	Comms
PAD_AUX_CLK	ClockGen FS0.3 output or observation of one of the internally generated clocks	27 or more	to PAD CLKOUT
PAD_PIXCLK	ClockGen Copy of either CLK_DISP_HD or CLK_656	74.25 (max)	to PAD PIXCLK

Note: Adequate video clock frequencies according to target applications are detailed in Chapter 58: Video output stage (VOS) on page 542. Values mentioned here are indicative.

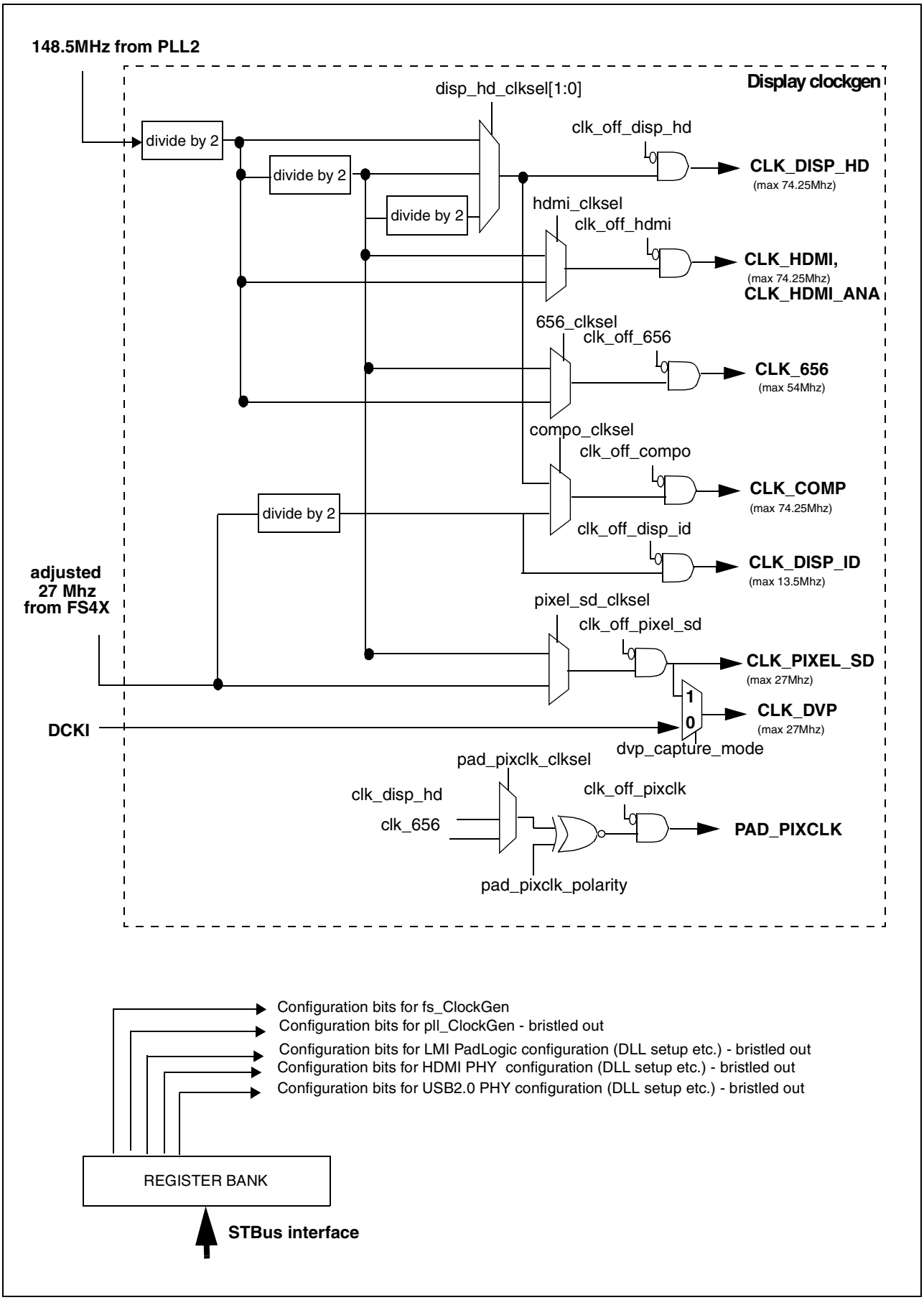
7.4 Clock generation overview

Figure 15: Clock generation



Note: Power down and clock disabling features are not shown - for details see Chapter 9: Power down and standby modes on page 93 and ClockGen registers FS_CLOCKGEN_CFG_x (in this diagram PAD_AUX_CLK*).*

Figure 16: Clock generation - video display hardware



8 Clocks registers

Addresses are provided as *ClocksBaseAddress* + offset.

The *ClocksBaseAddress* is:

0x3800 0000.

Almost all write accesses to clock generator registers are protected by a key (bit field UNLOCK_KEY). This is to make sure that any accidental writes to these registers do not make the chip hang or behave abnormally or uncontrollably.

- Almost all clock generator registers have their 16 MSBs defined as write-only.
- Writes to these registers are only accepted if the 16 MSBs are set to the correct value (the “key” that unlocks access), which is defined as 0x5AA5. Any other writes are considered invalid and have no effect.

Since most clock registers implement key protection, their read/write type is mixed. Unless otherwise stated, all bits are type read/write, write protected by their UNLOCK_KEY; UNLOCK_KEY bits are write only.

Unless otherwise stated, all bits reset to 0.

Table 28: Clocks register summary

Register	Description	Offset	Type
PLL_CLOCKGEN_CTRL_0	Clock generator PLL circuitry configuration	0x00	R/W[a]
PLL_CLOCKGEN_CTRL_1	Clock generator PLL configuration 1	0x04	R/W[a]
PLL_CLOCKGEN_CTRL_2	Clock generator PLL configuration 2	0x08	R/W[a]
PLL_CONFIG_0	Control of analog PLL I/Os 0	0x0C	R/W[a]
PLL_CONFIG_1	Control of analog PLL I/Os 1	0x10	R/W[a]
PLL_CONFIG_2	Control of analog PLL I/Os 2	0x14	R/W[a]
Reserved		0x 18 to 0x 2F	-
FS_0_CTRL	FS0.n configuration bits	0x30	R/W[a]
FS_0_1_CONFIG_0	FS0.1 configuration bits 0	0x34	R/W[a]
FS_0_1_CONFIG_1	FS0.1 configuration bits 1	0x38	R/W[a]
FS_0_2_CONFIG_0	FS0.2 configuration bits 0	0x3C	R/W[a]
FS_0_2_CONFIG_1	FS0.2 configuration bits 1	0x40	R/W[a]
FS_0_3_CONFIG_0	FS0.3 configuration 0	0x44	R/W[a]
FS_0_3_CONFIG_1	FS0.3 configuration bits 1	0x48	R/W[a]
FS_0_4_CONFIG_0	FS0.4 configuration bits 0	0x4C	R/W[a]
FS_0_4_CONFIG_1	FS0.4 configuration bits 1	0x50	R/W[a]
Reserved		0x 54 to 0x 5F	-
FS_1_CTRL	FS1.n configuration bits	0x60	R/W[a]
FS_1_1_CONFIG_0	FS1.1 configuration bits 0	0x64	R/W[a]

Table 28: Clocks register summary

Register	Description	Offset	Type
FS_1_1_CONFIG_1	FS1.1 configuration bits 1	0x68	R/W[a]
FS_1_2_CONFIG_0	FS1.2 configuration bits 0	0x6C	R/W[a]
FS_1_2_CONFIG_1	FS1.2 configuration bits 1	0x70	R/W[a]
FS_1_3_CONFIG_0	FS1.3 configuration bits 0	0x74	R/W[a]
FS_1_3_CONFIG_1	FS1.3 configuration bits 1	0x78	R/W[a]
FS_1_4_CONFIG_0	FS1.4 configuration bits 0	0x7C	R/W[a]
FS_1_4_CONFIG_1	FS1.4 configuration bits 1	0x80	R/W[a]
Reserved		0x 84 to 0x 8F	-
PLL2_CONFIG_0	Analog PLL I/O control 0	0x90	R/W[a]
PLL2_CONFIG_1	Analog PLL I/O control 1	0x94	R/W[a]
PLL2_CONFIG_2	Analog PLL I/O control 2	0x98	R/W[a]
Reserved		0x 9C to 0x BF	-
FS_CLOCKGEN_CFG_0	Individual clocks' power off control	0xC0	R/W[a]
FS_CLOCKGEN_CFG_1	Individual clocks' power down control	0xC4	R/W[a]
FS_CLOCKGEN_CFG_2	Display clock generator configuration	0xC8	R/W[a]
Reserved		0x CCto 0x EF	-
REF_MAX	Reference counter maximum value	0xF0	R/W
CMD	Reference counter command	0xF4	R/W
CPT_PCM	PCM counter value	0xFC	RO
CPT_HD	HD video counter value	0xF8	RO

a. Locked by a write key to avoid spurious writes crashing the system.

PLL_CLOCKGEN_CTRL_0 Clock generator PLL circuitry 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
UNLOCK_KEY																Reserved						DLL3_AUTORST_POWERDOWN	DLL3_AUTORST_XTAL	DLL3_IN_POLARITY	DLL3_RSTN	LMI_ASYNC_MODE	SWITCH_TO_LMIPL_CLK	DLL3_BYPASS	POWER_DOWN_PLLCLKS	CLK_SYS_OFF	CLK_ST20_OFF

Address: *ClocksBaseAddress* + 0x00

Type: R/W (locked)

Reset: 0x0000 0140

Description: Configuration of digital circuitry around the PLL generating 100, 200 and 400 MHz.

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:10] **Reserved**

[9] **DLL3_AUTORST_POWERDOWN**
1: reset DLL3 during power down mode.

[8] **DLL3_AUTORST_XTAL**:
1: reset DLL3 during start up.

[7] **DLL3_IN_POLARITY**:
0: delay element input = MCLK.
1: delay element input = not(MCLK).

[6] **DLL3_RSTN**: DLL3 reset signal, active low.
Reset: 1

[5] **LMI_ASYNC_MODE**:
1: LMI is used in asynchronous mode and CLK_ST20 comes from FS4X PAD_AUXCLK output.

[4] **SWITCH_TO_LMIPL_CLK**
0: CLK_ST20 and CLK_SYS are generated from a slow clock (27 Mhz) from the crystal-controlled oscillator. Toggle this bit from 0 to 1 when the LMI PadLogic has been configured and DLLs in the LMI PadLogic have locked.
1: CLK_ST20 and CLK_SYS are generated from their normal source, a division of the reference 400 MHz, and run at their normal speed, 200 Mhz and 100 MHz respectively.

[3] **DLL3_BYPASS**
1: DLL3 of the pll_ClockGen is bypassed. Use for debug.

[2] **POWER_DOWN_PLLCLKS**
1: The frequency of 200 and 100 MHz clocks is divided by 1024 to run at 195 and 97.5 kHz.

[1] **CLK_SYS_OFF**
1: 100 MHz clock is switched off over complete chip.

[0] **CLK_ST20_OFF**
1: 200 MHz clock is switched off over complete chip.

PLL_CLOCKGEN_CTRL_1 Clock generator PLL configuration 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
UNLOCK_KEY																DLL3_CMD[8:0]									DLL3_LOCK_STATUS	DLL3_LOCK_CTRL[3:0]				DLL3_CLEARD	DLL3_SETD

Address: *ClocksBaseAddress* + 0x04

Type: R/W (locked)

Reset: 0

Description: Configuration of the Clock generator DLL.

[31:16] WO **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:7] RO **DLL3_CMD[8:0]**: Image of the delay command: reflects the amount of delay introduced by the delay element (0x00 = minimum delay).

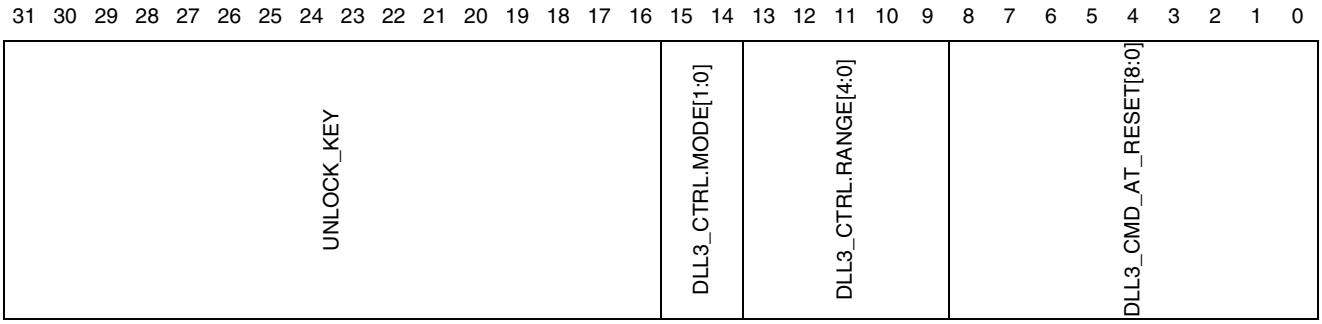
[6] RO **DLL3_LOCK_STATUS**: Lock status of the DLL (asserted high).

[5:2] R/W **DLL3_LOCK_CTRL[3:0]**: DLL lock control.
Defines the number of consecutive up/down pulses that trigger the lock status.
0001 to 1010: Lock goes high when DLL3_LOCK_CTRL+1 consecutive up or down pulses have been detected.
All other values: Lock goes high when 5 consecutive pulses have been detected.

[1] R/W **DLL3_CLEARD**: Clear all inputs to the internal delay element. Debug use only. Keep at 0 for normal operation.

[0] R/W **DLL3_SETD**: Set all inputs to the internal delay element. Debug use only. Keep at 0 for normal operation.

PLL_CLOCKGEN_CTRL_2 Clock generator PLL configuration 2



Address: *ClocksBaseAddress* + 0x08

Type: R/W (locked)

Reset: 0

Description: Configuration of the DLL used in pll_ClockGen.

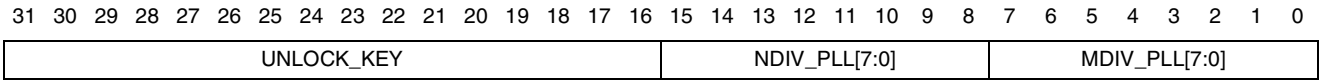
[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:14] **DLL3_CTRL.MODE[1:0]**: Defines the operating mode of the DLL:
00: Stand-alone mode. Delay is calculated by the DLL.
01: Delay command is forced by the user, DLL_CMD[8:0] = range [4:0] AND “1000”
10: Delay command is adjusted by the user by adding range [4:0] to DLL_CMD[8:0].
11: Reserved

[13:9] **DLL3_CTRL.RANGE[4:0]**: User defined range. Allowed values from 0 to 19 (natural binary codes). These bits enable to set the upper part of the delay element command (bits [8:4]) according to bits DLL3_CTRL.MODE[1:0] described above.

[8:0] **DLL3_CMD_AT_RESET[8:0]**: Start up command given to the delay. Allowed range from 0 to 400 (natural binary).

PLL_CONFIG_0 Control of analog PLL I/Os 0



Address: *ClocksBaseAddress* + 0x0C

Type: R/W (locked)

Reset: 0x0000 C71A

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:8] **NDIV_PLL[7:0]**: ‘*n*’ feedback divider ratio (1 to 256).
Reset: 0xC7 (*n* = 200).

[7:0] **MDIV_PLL[7:0]**: PLL ‘*m*’ pre-divider ratio (1 to 256).
Reset: 0x1A (*m* = 27).

Note: The reset value of this register enables a 400 MHz reference clock to be generated internally at reset.

PLL_CONFIG_1 Control of analog PLL I/Os 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
UNLOCK_KEY																Reserved			SETUP_PLL[9]	SETUP_PLL[8:6]			SETUP_PLL[5]	SETUP_PLL[4]	SETUP_PLL[3:0]				PDIV_PLL[2:0]		

Address: *ClocksBaseAddress* + 0x10

Type: R/W (locked)

Reset:

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:13] **Reserved**

[12] **SETUP_PLL[9]**: Reserved. Keep at 0.

[11:9] **SETUP_PLL[8:6]**: PLL lock detection threshold.
Reset: 111.

[8] **SETUP_PLL[5]**: PLL lock detector reset.

[7] **SETUP_PLL[4]**: Reserved. Keep at 0.

[6:3] **SETUP_PLL[3:0]**: PLL charge pump current control.
Reset value = 0x0E40
Setup_pll[3:0]: Reset=1000
Pdiv_pll[3:0]: Reset = 0000

[2:0] **PDIV_PLL[2:0]**: '2^p' post-divider ratio (1 to 32).

A 400 Mhz clock must be present at reset for this register.

PLL_CONFIG_2 Control of analog PLL I/Os 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
UNLOCK_KEY																Reserved							PLL_LOCK_STATUS	Reserved					NRST	POFF	Reserved

Address: *ClocksBaseAddress* + 0x14

Type: R/W (locked)

Reset: 0

Description:

[31:16] WO **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:9] - **Reserved**

[8] RO **PLL_LOCK_STATUS**
1: PLL is locked, meaning that the PLL's PFD phase error is less than the threshold selected through SETUP_PLL[8:6].

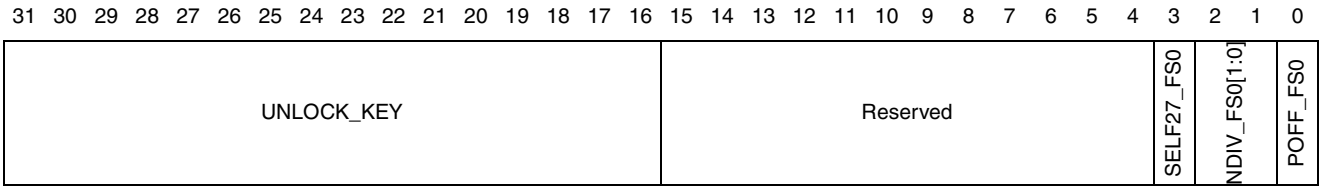
[7:3] - **Reserved**

[2] R/W **NRST**
0: PLL is bypassed: output clock = input reference clock.
Reset: 1.

[1] R/W **POFF**
1: PLL is powered off.

[0] **Reserved**

FS_0_CTRL FS0.n configuration bits



Address: *ClocksBaseAddress* + 0x30

Type: R/W (locked)

Reset: 0x0000 0004

Description: Configuration bits common to FS0.1, FS0.2, FS0.3 and FS0.4

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

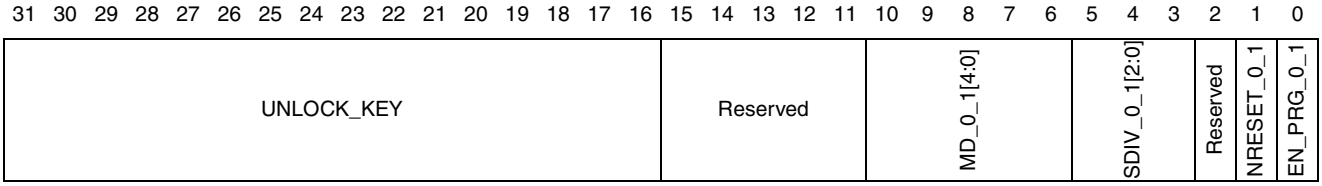
[15:4] **Reserved**

[3] **SELF27_FS0**: Set to 0 if internal reference frequency of PLL is 13.5 MHz, or to 1 if it is 27 MHz (recommended in STi7710).
Reset: 1.

[2:1] **NDIV_FS0[1:0]**: Input divider control:
00: input frequency divided by 1,
01: input frequency divided by 2,
10: input frequency divided by 4,
11: output of divider set to 0.

[0] **POFF_FS0**: Power off control.
1: The analog part of this quad-FS is stopped.

FS_0_1_CONFIG_0 FS0.1 configuration bits 0



Address: *ClocksBaseAddress* + 0x34

Type: R/W (locked)

Reset: 0x0000 0696

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_0_1[4:0]**: Coarse tuning of generated frequency.

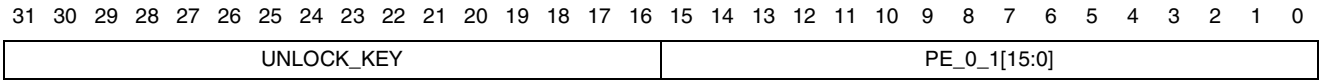
[5:3] **SDIV_0_1[2:0]**: Programming of FS0.1 output divider.
Reset: 0x2.

[2] **Reserved**

[1] **NRESET_0_1**
0: The digital algorithm used to generate FS0.1 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_0_1**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_0_1_CONFIG_1 FS0.1 configuration bits 1



Address: *ClocksBaseAddress* + 0x38

Type: R/W (locked)

Reset: 0x0000 5100

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_0_1[15:0]**: Fine tuning of generated frequency.

FS_0_2_CONFIG_0 FS0.2 configuration bits 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
UNLOCK_KEY																Reserved					MD_0_2[4:0]					SDIV_0_2[2:0]			Reserved	NRESET_0_2	EN_PRG_0_2

Address: *ClocksBaseAddress* + 0x3C

Type: R/W (locked)

Reset: 0x0000 05D6

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_0_2[4:0]**: Coarse tuning of generated frequency.

[5:3] **SDIV_0_2[2:0]**: Programming of FS0.2 output divider.
Reset: 0x2.

[2] **Reserved**

[1] **NRESET_0_2**
0: The digital algorithm used to generate FS0.2 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use
Reset: 1.

[0] **EN_PRG_0_2**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_0_2_CONFIG_1 FS0.2 configuration bits 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
UNLOCK_KEY																PE_0_2[15:0]															

Address: *ClocksBaseAddress* + 0x40

Type: R/W (locked)

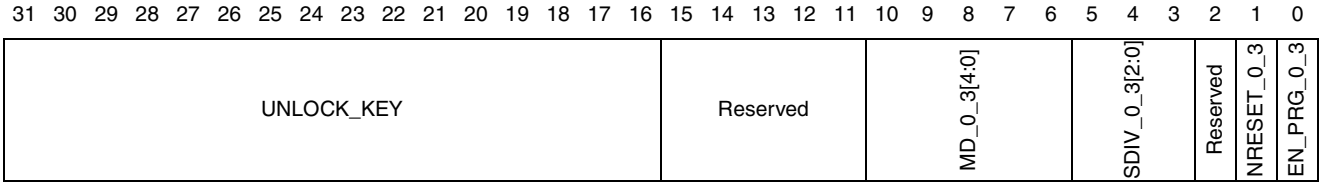
Reset: 0x4800

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_0_2[15:0]**: Fine tuning of generated frequency.

FS_0_3_CONFIG_0 FS0.3 configuration 0



Address: *ClocksBaseAddress* + 0x44

Type: R/W (locked)

Reset: 0x0000 0446

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_0_3[4:0]**: Coarse tuning of generated frequency.

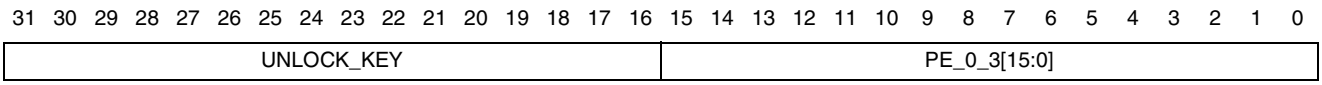
[5:3] **SDIV_0_3[2:0]**: Programming of FS0.3 output divider.

[2] **Reserved**

[1] **NRESET_0_3**
0: The digital algorithm used to generate FS0.3 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_0_3**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_0_3_CONFIG_1 FS0.3 configuration bits 1



Address: *ClocksBaseAddress* + 0x48

Type: R/W (locked)

Reset: 0x0000 5C29

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_0_3[15:0]**: Fine tuning of generated frequency.

FS_0_4_CONFIG_0 FS0.4 configuration bits 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
UNLOCK_KEY																Reserved					MD_0_4[4:0]					SDIV_0_4[2:0]			Reserved	NRESET_0_4	EN_PRG_0_4

Address: *ClocksBaseAddress* + 0x4C

Type: R/W (locked)

Reset: 0x0000 05C6

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_0_4[4:0]**: Coarse tuning of generated frequency.

[5:3] **SDIV_0_4[2:0]**: Programming of FS0.4 output divider.

[2] **Reserved**

[1] **NRESET_0_4**
0: The digital algorithm used to generate FS0.4 clock is stopped; PE and MD keep their value and an output clock is still generated; only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_0_4**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_0_4_CONFIG_1 FS0.4 configuration bits 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
UNLOCK_KEY																PE_0_4[15:0]															

Address: *ClocksBaseAddress* + 0x50

Type: R/W (locked)

Reset: 0x0000 5D17

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_0_4[15:0]**: Fine tuning of generated frequency.

FS_1_CTRL FS1.n configuration 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
UNLOCK_KEY																Reserved												SELF27_FS1	NDIV_FS1[1:0]		POFF_FS1

Address: *ClocksBaseAddress* + 0x60

Type: R/W (locked)

Reset: 0x0000 0008

Description: Configuration bits common to FS1.1, FS1.2, FS1.3 and FS1.4.

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:4] **Reserved**

[3] **SELF27_FS1**: Set to 0 if internal reference frequency of PLL is 13.5 MHz, to 1 if it is 27 MHz. Reset: 1

[2:1] **NDIV_FS1[1:0]**: Input divider control:
00: input frequency divided by 1,
01: input frequency divided by 2,
10: input frequency divided by 4,
11: output of divider set to 0.

[0] **POFF_FS1**: Power off control.
1: The analog part of this quad-FS is stopped.

FS_1_1_CONFIG_0 FS1.1 configuration bits 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
UNLOCK_KEY																Reserved					MD_1_1[4:0]					SDIV_1_1[2:0]			Reserved	NRESET_1_1	EN_PRG_1_1

Address: *ClocksBaseAddress* + 0x64

Type: R/W (locked)

Reset: 0x0000 03DE

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_1_1[4:0]**: Coarse selection bus controlling FS1_1 clock generation. Acts on choice of phase taps. Reset: 0x0F.

[5:3] **SDIV_1_1[2:0]**: Programming of FS1.1 output divider. Reset: 0x3.

[2] **Reserved**

[1] **NRESET_1_1**
0: The digital algorithm used to generate FS1.1 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_1_1**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_1_1_CONFIG_1 FS1.1 configuration bits 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
UNLOCK_KEY																PE_1_1[15:0]															

Address: *ClocksBaseAddress* + 0x68

Type: R/W (locked)

Reset: 0

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_1_1[15:0]**: Fine selection bus controlling FS1.1 clock generation. Acts on choice of phase taps.

FS_1_2_CONFIG_0 FS1.2 configuration bits 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
UNLOCK_KEY																Reserved					MD_1_2[4:0]					SDIV_1_2[2:0]			Reserved	NRESET_1_2	EN_PRG_1_2

Address: *ClocksBaseAddress* + 0x6C

Type: R/W (locked)

Reset: 0x0000 0006

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_1_2[4:0]**: Coarse selection bus controlling FS1.2 clock generation. Acts on choice of phase taps.

[5:3] **SDIV_1_2[2:0]**: Programming of FS1.2 output divider.

[2] **Reserved**

[1] **NRESET_1_2**
0: The digital algorithm used to generate FS1.2 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_1_2**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_1_2_CONFIG_1 FS1.2 configuration bits 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
UNLOCK_KEY																PE_1_2[15:0]															

Address: *ClocksBaseAddress* + 0x70

Type: R/W (locked)

Reset: 0

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.
Reset: 0x5D17.

[15:0] **PE_1_2[15:0]**: Fine selection bus controlling FS1.2 clock generation. Acts on choice of phase taps.

FS_1_3_CONFIG_0 FS1.3 configuration bits 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
UNLOCK_KEY																Reserved					MD_1_3[4:0]					SDIV_1_3[2:0]			Reserved	NRESET_1_3	EN_PRG_1_3

Address: *ClocksBaseAddress* + 0x74

Type: R/W (locked)

Reset: 0x0000 0506

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_1_3[4:0]**: Coarse selection bus controlling FS1.3 clock generation. Acts on choice of phase taps. Reset: 0x14.

[5:3] **SDIV_1_3[2:0]**: Programming of FS1.3 output divider.

[2] **Reserved**

[1] **NRESET_1_3**:
0: The digital algorithm used to generate FS1.3 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_1_3**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_1_3_CONFIG_1 FS1.3 configuration bits 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
UNLOCK_KEY																PE_1_3[15:0]															

Address: *ClocksBaseAddress* + 0x78

Type: R/W (locked)

Reset: 0x0000 21B0

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_1_3[15:0]**: Fine selection bus controlling FS1.3 clock generation. Acts on choice of phase taps. Reset: 0x21B0.

FS_1_4_CONFIG_0 FS1.4 configuration bits 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
UNLOCK_KEY																Reserved					MD_1_4[4:0]					SDIV_1_4[2:0]			Reserved	NRESET_1_4	EN_PRG_1_4

Address: *ClocksBaseAddress* + 0x7C

Type: R/W (locked)

Reset: 0x0000 0496

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:11] **Reserved**

[10:6] **MD_1_4[4:0]**: Coarse selection bus controlling FS1.4 clock generation. Acts on choice of phase taps. Reset: 0x12.

[5:3] **SDIV_1_4[2:0]**: Programming of FS1.4 output divider. Reset: 0x2.

[2] **Reserved**

[1] **NRESET_1_4**
0: The digital algorithm used to generate FS1.4 clock is stopped; PE and MD keep their value and an output clock is still generated, only internal registers are cleared. Debug and diagnosis use.
1: The digital algorithm runs normally. Normal use.
Reset: 1.

[0] **EN_PRG_1_4**: Set once MD and PE have been programmed to make new frequency setting effective.

FS_1_4_CONFIG_1 FS1.4 configuration bits 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
UNLOCK_KEY																PE_1_4[15:0]															

Address: *ClocksBaseAddress* + 0x80

Type: R/W (locked)

Reset: 0x0000 8000

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:0] **PE_1_4[15:0]**: Fine selection bus controlling FS1_4 clock generation. Acts on choice of phase taps. Reset: 0x8000.

PLL2_CONFIG_0 Analog PLL I/O control 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
UNLOCK_KEY																NDIV_PLL[7:0]								MDIV_PLL[7:0]							

Address: *ClocksBaseAddress* + 0x90
Type: R/W (locked)
Reset: 0
Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:8] **NDIV_PLL[7:0]**: '*n*' feedback divider ratio (1 to 256).
Reset: 0 (*n* = 1).

[7:0] **MDIV_PLL[7:0]**: PLL '*m*' pre-divider ratio (1 to 256).
Reset: 0 (*m* = 1).

Note: A 148.5 Mhz clock must be present at reset for this register.

PLL2_CONFIG_1 Analog PLL I/O control 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
UNLOCK_KEY																Reserved			SETUP_PLL[9]	SETUP_PLL[8:6]			SETUP_PLL[5]	SETUP_PLL[4]	SETUP_PLL[3:0]				PDIV_PLL[2:0]		

Address: *ClocksBaseAddress* + 0x94
Type: R/W (locked)
Reset: 0x0000 0901
Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:13] **Reserved**

[12] **SETUP_PLL[9]**: Reserved. Keep at 0.

[11:9] **SETUP_PLL[8:6]**: PLL lock detection threshold.
Reset: 0x5.

[8] **SETUP_PLL[5]**: PLL lock detector reset.

[7] **SETUP_PLL[4]**: Reserved. Keep at 0.

[6:3] **SETUP_PLL[3:0]**: PLL charge pump current control.

[2:0] **PDIV_PLL[2:0]**: '2^p' post-divider ratio (1 to 32).
Reset: 1 (*p* = 1).

Note: A 148.5 Mhz clock must be present at reset for this register.

PLL2_CONFIG_2 Analog PLL I/O control 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
UNLOCK_KEY																Reserved							PLL_LOCK_STATUS	Reserved					NRST	POFF	FBENABLE

Address: *ClocksBaseAddress* + 0x98

Type: R/W (locked)

Reset: 0x0000 0004

Description:

[31:16] WO **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:9] - **Reserved**

[8] RO **PLL_LOCK_STATUS**
1: PLL is locked, meaning that the PLL's PFD phase error is less than the threshold selected through SETUP_PLL[8:6].

[7:3] - **Reserved**

[2] R/W **NRST**
0: PLL is bypassed: output clock = input reference clock.
Reset: 1.

[1] R/W **POFF**
1: PLL is powered off.

[0] R/W **FBENABLE**: Keep at 0. (1 corresponds to different PLL application). Provided as error proof backup only.

Note: A 148.5 Mhz clock must be present at reset for this register.

FS_CLOCKGEN_CFG_0 Individual clocks' power off 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
UNLOCK_KEY																CLK_SPARE_1	CLK_OFF_PIXCLK	CLK_OFF_USB1	CLK_OFF_DSP	CLK_OFF_PIPE	CLK_SPARE_0	CLK_OFF_PIXEL_SD	CLK_OFF_DISP_ID	CLK_OFF_COMPO	CLK_OFF_656	CLK_OFF_HDMI	CLK_OFF_DISP_HD	CLK_OFF_PIXEL_HD	CLK_OFF_AUX	CLK_OFF_DSS	CLK_OFF_DAA

Address: *ClocksBaseAddress* + 0xC0

Type: R/W (locked)

Reset: 0

Description: When bits are 1, switch off corresponding clock over complete chip.

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15] **CLK_SPARE_1** :spare bit for Clock Control, reserved.

[14] **CLK_OFF_PIXCLK**: When 1, PAD_PIXCLK is switched off on PAD output.

[13] **CLK_OFF_USB1**: CLK_USB1

[12] **CLK_OFF_DSP**: CLK_DSP

[11] **CLK_OFF_PIPE**: CLK_PIPE

[10] **CLK_SPARE_0** : spare bit for Clock Control, reserved.

[9] **CLK_OFF_PIXEL_SD**: CLK_PIXEL_SD

[8] **CLK_OFF_DISP_ID**: CLK_DISP_ID

[7] **CLK_OFF_COMPO**: CLK_COMP

[6] **CLK_OFF_656**: CLK_656

[5] **CLK_OFF_HDMI**: CLK_HDMI

[4] **CLK_OFF_DISP_HD**: CLK_DISP_HD

[3] **CLK_OFF_PIXEL_HD**: CLK_PIXEL_HD

[2] **CLK_OFF_AUX**: PAD_AUX_CLK

[1] **CLK_OFF_DSS**: CLK_DSS_SC

[0] **CLK_OFF_DAA**: CLK_DAA

FS_CLOCKGEN_CFG_1 Individual clocks' power down control

31:16	15	14	13	12	11	10	9	8:4	3	2	1	0
UNLOCK_KEY	CLOCK_SPARE_3	CLOCK_SPARE_4	POWER_DOWN_USB1	POWER_DOWN_DSP	POWER_DOWN_PIPE	CLOCK_SPARE_2	POWER_DOWN_PIXEL_SD	RESERVED	POWER_DOWN_PIXEL_HD	POWER_DOWN_AUX	POWER_DOWN_DSS	POWER_DOWN_DAA

Address: *ClocksBaseAddress* + 0xC4

Type: R/W (locked)

Reset: 0

Description: When bits are 1, divide corresponding clock frequency by 1024.

The power down mode provides dramatically reduces power consumption of functional sub-systems by cutting their clock frequency by 1024. They are still clocked, so the chip remains as operational as possible.

Entry into/exit from power down mode is glitch free.

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15] **CLOCK_SPARE_3**: spare bit for clock control, reserved.

[14] **CLOCK_SPARE_4**: spare bit for clock control, reserved.

[13] **POWER_DOWN_USB1**: CLK_USB1

[12] **POWER_DOWN_DSP**: CLK_DSP

[11] **POWER_DOWN_PIPE**: CLK_PIPE

[10] **CLOCK_SPARE_2**: spare bit for clock control, reserved.

[9] **POWER_DOWN_PIXEL_SD**: CLK_PIXEL_SD

[8:4] **Reserved**

[3] **POWER_DOWN_PIXEL_HD**: CLK_PIXEL_HD

[2] **POWER_DOWN_AUX**: PAD_AUX_CLK

[1] **POWER_DOWN_DSS**: CLK_DSS_SC

[0] **POWER_DOWN_DAA**: CLK_DAA

FS_CLOCKGEN_CFG_2 Display clock generator 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
UNLOCK_KEY																Reserved	HDMI_ANA_POL	PLL2_BYPASS	CLK_OUT_SEL				CLK_OBS	PAD_PIXCLK_POLARITY	PAD_PIXCLK_CLKSEL	PIXEL_SD_CLKSEL	COMPO_CLKSEL	656_CLKSEL	HDMI_CLKSEL	DISP_HD_CLKSEL	

Address: *ClocksBaseAddress* + 0xC8

Type: R/W (locked)

Reset: 0

Description:

Note: *Typically for this register,* CLK_PIXEL_HD *= 148.5 MHz, or 108 MHz nominal with MPEG clock recovery applied; clk27 denotes the internal nominal 27 MHz MPEG reference clock, that is, with MPEG clock recovery applied.*

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15] **Reserved**

[14] **HDMI_ANA_POL**
0: Clock normal.
1: Clock inverted.

[13] **PLL2_BYPASS**
0: Select clock from rejection PLL2.
1: PLL2 is bypassed (output clock = FS0.4 clock).

[12:9] **CLK_OUT_SEL**: Select which clock is observed on CLKOUT pad

0000: CLK_PIXEL_HD
0001: CLK_DISP_HD
0010: CLK_COMP
0011: CLK_HDMI
0100: CLK_656
0101: CLK_DISP_ID
0110: CLK_PIXEL_SD
0111: PAD_PIXCLK
1000: CLK_PIPE
1001: CLK_DSP
1010: CLK_USB1
1011: CLK_USB
1100: AUD_FSYN_F108

[8] **CLK_OBS**
1: Allow clock observation on CLKOUT pad.

[7] **PAD_PIXCLK_POLARITY**: Toggle the polarity of PIXCLK supplied to chip output pad.

[6] **PAD_PIXCLK_CLKSEL**
0: PAD_PIXCLK = CLK_DISP_HD as defined with bits [1:0].
1: PAD_PIXCLK = CLK_656 as defined with bits [2].

[5] **PIXEL_SD_CLKSEL**
0: CLK_PIXEL_SD = CLK27.
1: CLK_PIXEL_SD = CLK_PIXEL_HD / 4.

[4] **COMPO_CLKSEL**
0: CLK_COMP = CLK_DISP_HD as defined with bits [1:0].
1: CLK_COMP = clk27 / 2.

[3] **656_CLKSEL**
0: CLK_656 = CLK_PIXEL_HD / 2.
1: CLK_656 = CLK_PIXEL_HD / 4.

[2] **HDMI_CLKSEL**:
0: CLK_HDMI = CLK_PIXEL_HD / 2.
1: CLK_HDMI = CLK_PIXEL_HD / 4.

[1:0] **DISP_HD_CLKSEL**
00: Reserved, do not use.
01: CLK_DISP_HD = CLK_PIXEL_HD / 2.
10: CLK_DISP_HD = CLK_PIXEL_HD / 4.
11: CLK_DISP_HD = CLK_PIXEL_HD / 8.
Reset: 01.

REF_MAX Reference counter maximum value

Address: *ClocksBaseAddress* + 0xF0
Type: R/W
Reset: 0
Description: This register holds the maximum value of the reference counter for HD video clock recovery.
Note: Writes are not gated by an unlock key.

CMD Reference counter 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: *ClocksBaseAddress* + 0xF4
Type: R/W
Reset: 0
Description:

[31:2] **Reserved**

[1] **INT**
1: the reference counter has reached interrupt condition; When written to, interrupt is cleared.

[0] **LD**: When set to 1, load the value defined in the reference counter maximum value register REF_MAX.

Note: Writes are not gated by an unlock key.

CPT_PCM PCM counter 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_PCM																															

Address: *ClocksBaseAddress* + 0xFC
Type: RO
Reset: 0
Description: This register holds the value captured at the output of the PCM counter.

CPT_HD HD video counter 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_HD																															

Address: *ClocksBaseAddress* + 0xF8
Type: RO
Reset: 0
Description: This register holds the value captured at the output of the HD counter.

9 Power down and standby modes

9.1 Definition of low-power modes

In **power down mode**, some or almost all clocks are slowed by a factor of 1024. This reduces power consumption dramatically, yet still enables some software to run, and enables the chip to be woken up quickly without having to reboot.

Certain precautions must be taken to achieve this, in particular, DDR SDRAM must be put into self-refresh mode prior to entering power down mode, and at wake up, no access to DDR may be performed until the DDR interface has been restored back to its normal mode.

In **standby mode**, some or all clocks can be completely switched off. This can save power in applications where some functionality clocked by a dedicated clock is not needed. It also enables troubleshooting where digital signals are suspected of causing crosstalk on analog signals. Exit from standby modes is described below, and applies to all cases where the 100 MHz and 200 MHz main system clocks are not completely switched off; whenever these clocks are switched off, it is necessary to hard reset the device to wake it up.

Note: The terms 'power down' and 'standby' as defined above apply only to the STi7710 within the context of this datasheet.

The remainder of this chapter describes the low power controller, how to enter and exit low power modes, and includes a block diagram of the low power module. See also Chapter 7: *Clocks on page 64* and Chapter 8: *Clocks registers on page 69*.

9.2 Low power controller (LPC)

The low power controller provides a number of features to reduce power consumption in standby mode. These features are:

- a low power alarm counter to force the chip into power down mode for a fixed period of time,
- wake up from power down mode when either,
  - an external interrupt is triggered,
  - the low power alarm counter reaches zero,
  - activity on the UHF or IRB input signals.

In addition, the LPC has the following capabilities.

- It provides a separate timer to allow a real-time clock to be implemented. The timer keeps track of real time even during power down and standby modes.
- It provides a watchdog function. This function uses the same low power alarm counter to ensure that it can only be used as a watchdog timer or a low power wake up timer. When configured as a watchdog timer, and when the counter reaches 1, a reset signal is generated which is fed to the reset generator causing a chip reset (see Figure 18: *Combined reset generation on page 101*).

The watchdog timer uses an internal counter to generate a reset signal if the counter counts down to 1. Once the counter reaches the value 1, the NOT_WD_OUT pin is asserted. If the WD_ENABLE_REG bit is set high then the WD_NOT_RESET pin is asserted and WD_FLAG is set high. The WD_FLAG can be read from the CPU to determine whether a reset has been generated by the watchdog or not.

9.3 Entering low-power modes

By programming the ClockGen registers

Enter power down and standby modes

Both power down and standby modes may be entered by configuring the ClockGen registers; see Chapter 10: *Power down and standby modes registers* *on page 97*.

By using the global power down control bit

Enter power down mode

Global power down mode (all clocks slowed) may be entered by configuring the communications power control register.

By using the low power controller (LPC) module

Enter power down mode

Global power down mode (all clocks slowed) may be entered by using the low power timer register.

This usage of the LPA Counter is not compatible with use as WatchDog Timer.

9.4 Leaving low-power modes

By programming the ClockGen registers

Leave power down and standby modes

If power down or standby mode was entered by programming the ClockGen configuration registers; leaving this mode can be achieved simply by clearing the appropriate bits.

By using the LPC

Leave power down mode

If power down was entered by using the LPA, when the LPA counter reaches zero, it releases the global power down command and the ClockGen exits power down mode. The duration of the countdown is user-programmable, and can be any time from a few milliseconds up to 271 days, since the LPA counter is 40 bits clocked at 46.87 kHz nominal.

Upon detection of activity on the UHF or IRB inputs

UHF or IRB inputs generate an interrupt to the ILC, which, assuming it has first been programmed accordingly, treats it as a wake up request that it routes to the LPC.

Leave power down mode

If power down was entered by using the global power down control bit or the LPC, the LPC clears the global power down command that goes to the ClockGen and normal speed is restored.

Leave power down mode and standby mode

If power down or stand by mode was entered by programming the ClockGen registers, the configuration bits in the ClockGen must be set back to their normal value. This can be done by an interrupt servicing routine. However, since DDR is not operational, this routine must be stored in internal SRAM (or external flash, but this is not very efficient). The interrupt going to the ILC upon detection of IRB/UHF activity can be treated as a normal interrupt.

Note: During power down mode, at least some clocks run 1024 times slower than normal; this will affect the speed of interrupt handling.

Servicing the interrupt consists of restoring the ClockGen configuration.

Upon any selected interrupt

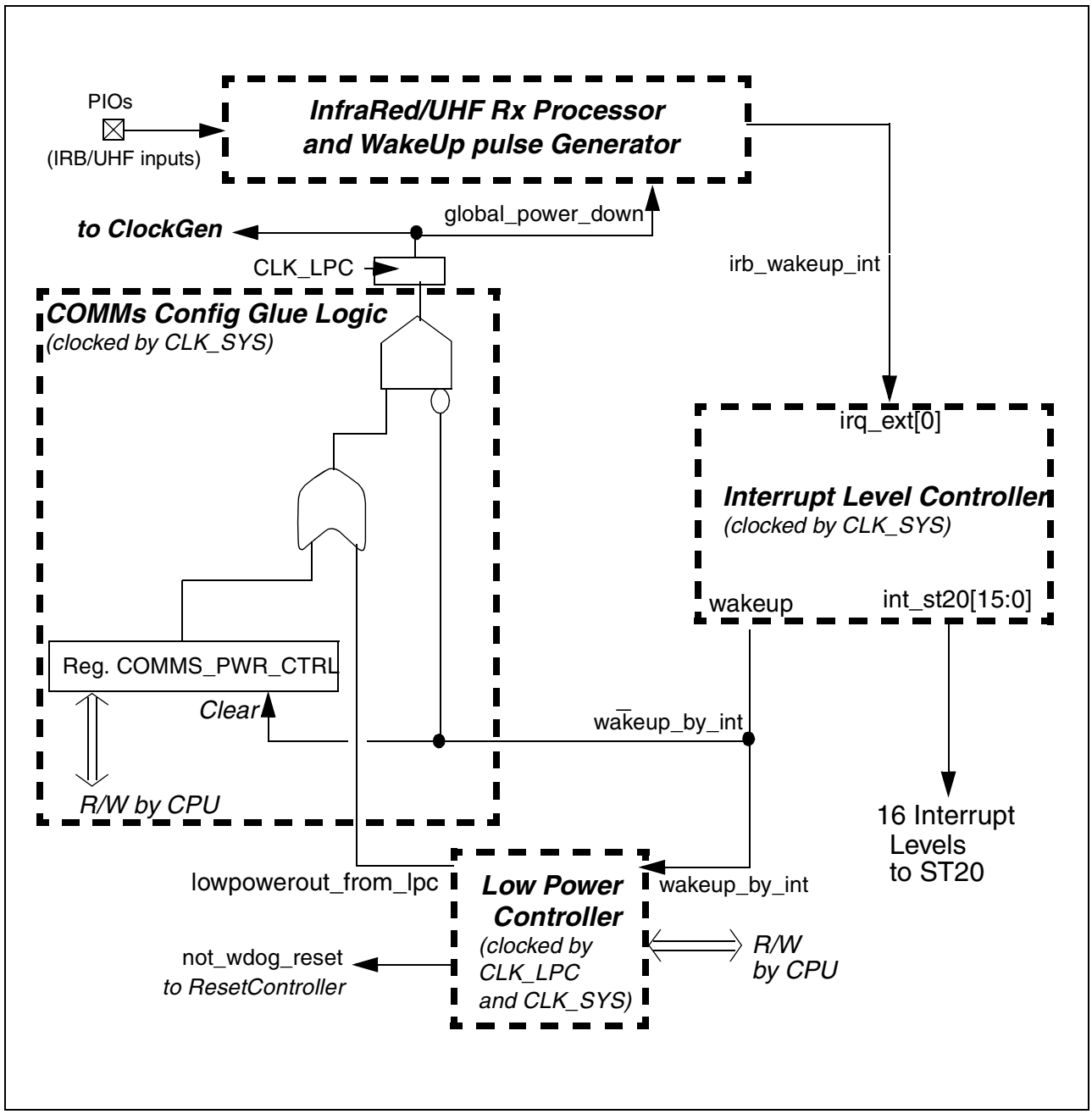
Leave power down and standby modes

This is simply an extension of the UHF/IRB case above. As long as the interrupt handler is not located in DDR, then any source can be programmed to trigger an interrupt routine that will wake up the chip. The interrupt handler may use different methods, depending on how the low power mode was entered:

- programming the ClockGen registers: clear the appropriate bits,
- using the global power down control bit: clear bit,
- using the low power controller (LPC) module: clear the LPA counter.

9.5 Low power module hardware

Figure 17: Power down mamagement hardware



10 Power down and standby modes registers

Addresses are provided as *LPCBaseAddress* + offset.

The *LPCBaseAddress* is:

0x2000 8000.

Unless otherwise stated, all bits reset to 0.

See also Chapter 6: *Configuration registers on page 60*.

Table 29: Low-power controller register summary

Register	Description	Offset	Type
LPC_LPT	Low-power timer	0x400	R/W
LPC_LPTSTART	Low-power timer start	0x408	WO
Reserved		0x409 to 0x40F	-
LPC_ALARM	Low-power alarm	0x410, 0x414	R/W
LPC_ALARMSTART	Low-power alarm start	0x418	WO
Reserved		0x419 to 0x50F	-
LPC_WDEN	Watchdog enable	0x510	R/W
LPC_WDFLAG	Watchdog flag	0x514	RO

10.1 Low-power controller register descriptions

LPC_LPT Low-power timer

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
LPTIMER[31:0]																															
LPTIMER[63:32]																															

Address: *LPCBaseAddress* + 0x400 (LSBs) and 0x404 (MSBs)

Type: R/W

Reset: Undefined

Description: These registers are the least significant and most significant words of the low-power timer register. These enable the least significant or most significant word to be written independently without affecting other words.

When either word of the register is written, the low-power timer is stopped and the new value in LPC_LPT is available to be written to the low-power timer.

LPC_LPTSTART Low-power timer start

7	6	5	4	3	2	1	0
Reserved							TIMERSTART

Address: *LPCBaseAddress* + 0x408

Type: WO

Description: A write of any value to this register starts the low-power timer counter. The counter is stopped and this register is reset if either word of LPC_LPT is written. Setting this register to zero does not stop the timer.

LPC_ALARM Low-power alarm

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
LPALARM[31:0]																															
Reserved																								LPALARM[39:32]							

Address: *LPCBaseAddress* + 0x410 (LSBs) and 0x414 (MSBs)

Type: R/W

Reset: Undefined

Description: These registers are the least significant and most significant words of the low-power alarm. They are used to program the alarm register.

LPC_ALARMSTART Low-power alarm start

7	6	5	4	3	2	1	0
Reserved							ALARMSTART

Address: *LPCBaseAddress* + 0x418

Type: WO

Description: Any write to this register starts the low-power alarm counter. The counter is stopped and this register is reset if either word of register LPC_ALARM is written.

LPC_WDEN Watchdog enable

7	6	5	4	3	2	1	0
Reserved							ENABLE

Address: *LPCBaseAddress* + 0x510
Type: R/W
Reset: Undefined
Description: Setting this register enables the low-power alarm counter to be used as a watchdog timer.

[0] **ENABLE**:
0: alarm 1: watchdog

LPC_WDFLAG Watchdog flag

7	6	5	4	3	2	1	0
Reserved							WDFLAG

Address: *LPCBaseAddress* + 0x514
Type: RO
Reset: Undefined
Description: This register is set when a watchdog reset occurs.

[0] **WDFLAG**:
0: no watchdog reset has occurred. 1: a watchdog reset has occurred.

11 Resets

11.1 Combined system reset generation

The STi7710 generates its internal global reset by combining an active low external hardware reset pin with various internal potential reset sources. This combined reset is made available to the external world via output pin EXT_NRESET_OUT, this can be used to reset the application (including for example Flash, Smart Cards and so on).

In addition, it is possible to stretch the combined reset by about 200 ms, thereby generating a fairly long reset for the application, under control of an hardware pin SEL_LONG_RESET or a configuration bit SEL_SW_LONG_RESET. The hardware pin is used to determine behavior upon external hardware reset (cold reset), the software bit is used to determine behavior after reset emanating from an alternative internal reset source (warm reset). The hardware pin SEL_LONG_RESET is muxed with output pin EMI_*n*BAA and is sampled upon hardware reset (at which time EMI_*n*BAA is tri-stated), the selected SEL_LONG_RESET value has to be set via a pull-up or pull-down resistor.

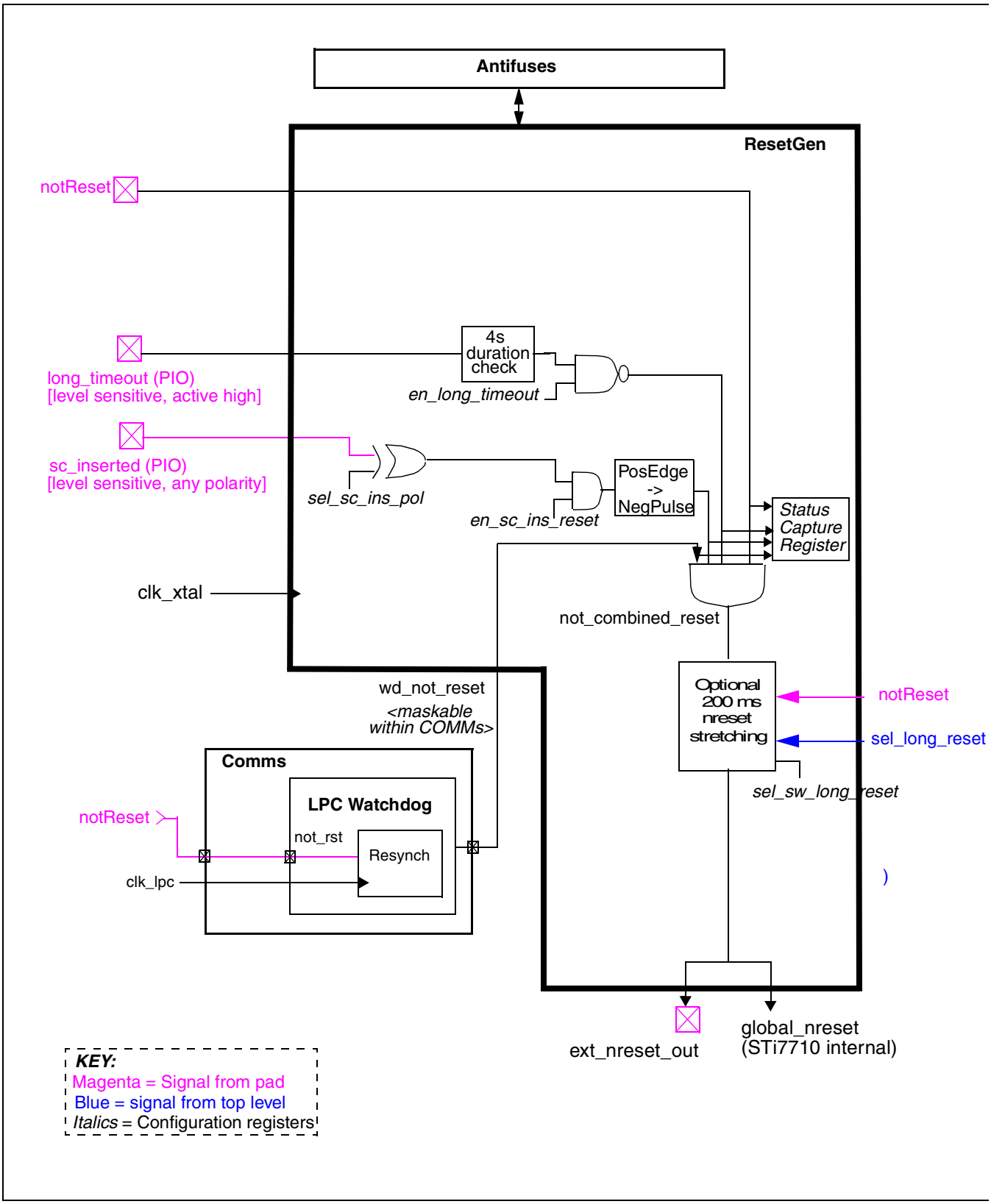
Possible reset sources are:

- hardware reset: when asserted, asynchronously asserts combined reset
- AntiFuse Data Ready: maintains reset asserted until high
- Long Time Out: assert combined reset when an external PIO is asserted (high) for more than about 4s. (The intent of this is to enable the user to reset the box by pressing a button on the STB for a few seconds). The LongTimeOut input is an alternate function of a PIO, refer to PIO assignments.
- Smart Card insertion: assert combined reset according to high level or low level of an external PIO (which connects externally to a mechanical switch that detects smart card resinsertion). Refer to PIO assignments.
- Watch Dog: assert combined reset when the watchdog counter that may be programmed in COMMs has elapsed

The Reset status register captures the state of all reset sources. This register is clearable by **hard reset only** so that when the STB restarts for any reason it is possible to know what caused the reset by reading these bits.

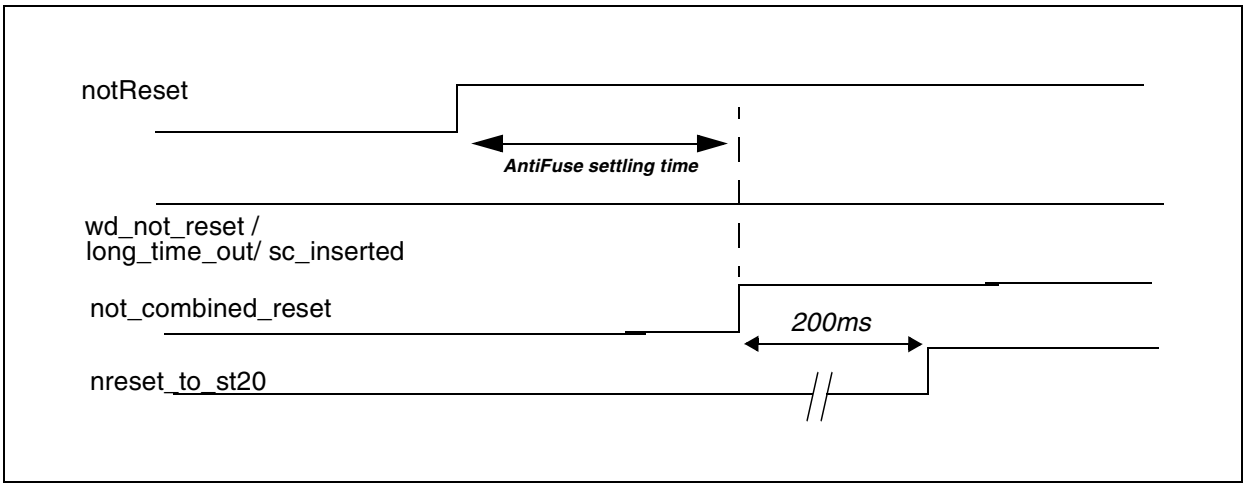
Figure 18, Figure 19A and Figure 19B illustrate reset generation and resulting waveforms.

Figure 18: Combined reset generation



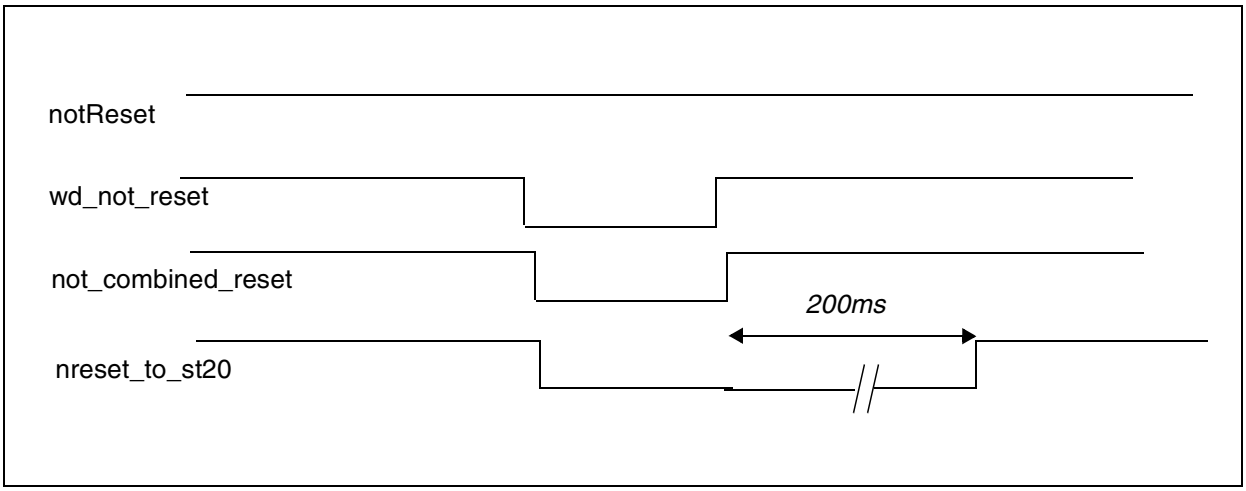
External Reset with SEL_LONG_RESET = ‘1’

Figure 19A: External reset action



Alternative source (for example internal watchdog) reset - with SEL_SW_LONG_RESET = ‘1’

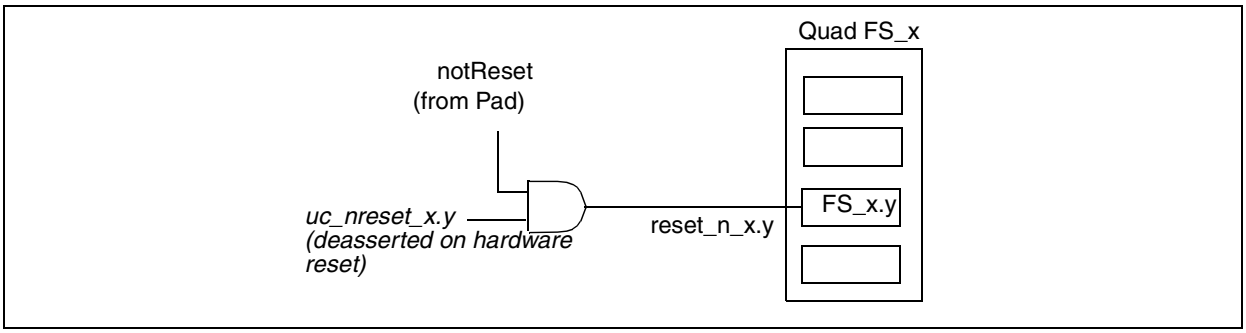
Figure 19B: Alternative source reset action



11.2 Frequency synthesizers reset

The frequency synthesizers are reset by the hardware reset; it is also possible to reset them by software: hardware and software reset are combined internally.

Figure 20: Frequency synthesizers reset



12 Resets registers

Addresses are provided as *ClocksBaseAddress* + offset.

The *ClocksBaseAddress* is:

0x3800 0000.

Some write accesses to reset registers are protected by a key (bit field UNLOCK_KEY). This is to make sure that any accidental writes to these registers do not make the chip hang or behave abnormally or uncontrollably:

- The registers have their 16 MSBs defined as write-only.
- Writes to these registers are only accepted if the 16 MSBs are set to the correct value (the “key” that unlocks access), which is defined as 0x5AA5. Any other writes are considered invalid and have no effect.

These registers’ their read/write type is mixed. Unless otherwise stated, all bits are type read/write, write protected by their UNLOCK_KEY; UNLOCK_KEY bits are write only.

Unless otherwise stated, all bits reset to 0.

Table 30: Resets register summary

Register	Description	Offset	Type
RESET_CTRL_0	Reset control 0	0xD0	R/W[a]
RESET_CTRL_1	Reset control 1	0xD4	R/W[a]
RESET_STATUS	Reset status	0xD8	RO

a. Locked by a write key to avoid spurious writes crashing the system.

RESET_CTRL_0 Reset control 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
UNLOCK_KEY																Reserved												SEL_SW_LONG_RESET	EN_LONG_TIMEOUT	SEL_SC_INS_POL	EN_SC_INS_RESET

Address: *ClocksBaseAddress* + 0xD0

Type: R/W (locked)

Reset: 0

Description:

[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:4] **Reserved**

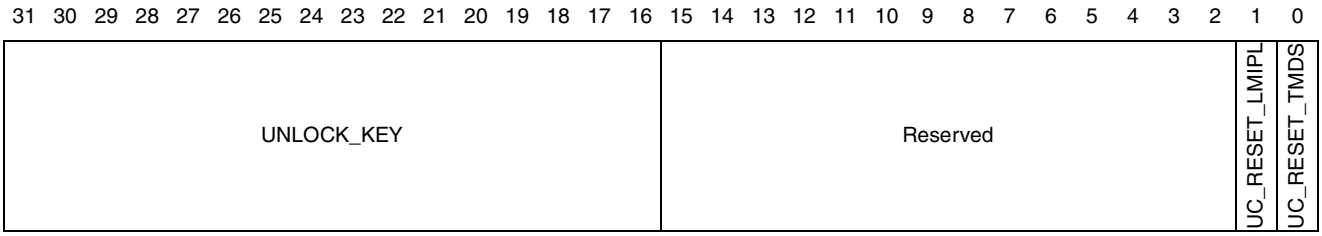
[3] **SEL_SW_LONG_RESET**
0: Soft resets have normal duration.
1**:** Soft resets last over 200 ms.

[2] **EN_LONG_TIMEOUT**
0: Disable long time out reset.
1: Enable long time out reset:
if PIO1.6 used as alternate input is asserted for more than about 4 s then a global system reset, quivalent to a hardware reset, is generated internally.

[1] **SEL_SC_INS_POL**: Select polarity of smart card insertion signal
0: SC_INSERTED high will mean smart card is present.
1: SC_INSERTED low will mean smart card is present.

[0] **EN_SC_INS_RESET**
1**:** Smart card insertion will cause a global system reset.

RESET_CTRL_1 Reset control 1



Address: *ClocksBaseAddress* + 0xD4

Type: R/W (locked)

Reset: 0

Description:

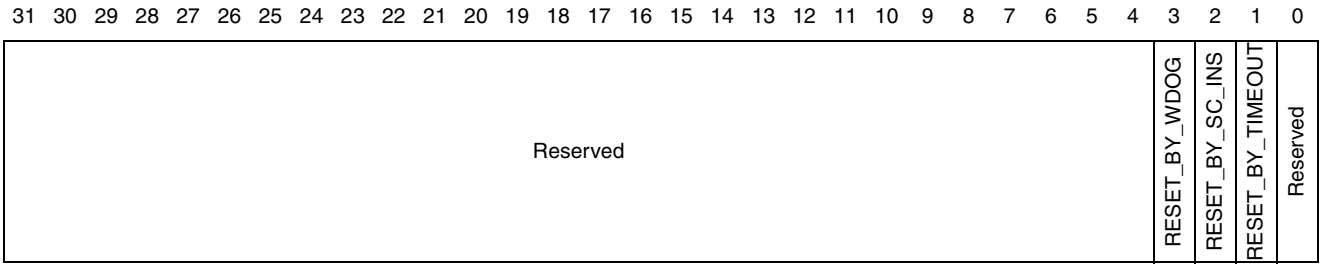
[31:16] **UNLOCK_KEY**: Must be written 0x5AA5 simultaneously with other bit writes to allow write access to other bits of this register, otherwise writes are ignored. Reads are not gated by UNLOCK_KEY. Returns 0 on read.

[15:2] **Reserved**

[1] **UC_RESET_LMIPL**
1: Send a reset to the LMI padlogic.

[0] **UC_RESET_TMDS**
1: Send a reset to the TMDS macrocell.

RESET_STATUS Reset status



Address: *ClocksBaseAddress* + 0xD8

Type: RO

Reset: 0

Description:

Note: RESET_STATUS is cleared by hardware (cold) reset. On warm reset, all status bits are updated (overwritten). This means that there is no need to clear the status bits after having read them; after a new reset, a subsequent read will indicate the single source that caused the warm reset.

[31:4] **Reserved**

[3] **RESET_BY_WDOG**
1: Watch Dog has caused a soft reset since last read.

[2] **RESET_BY_SC_INS**
1: Smart Card insertion has caused a soft reset since last read.

[1] **RESET_BY_TIMEOUT**
1: TimeOut has caused a soft reset since last read.

[0] **Reserved**

13 ST20-C1 central processing unit

The STi7710 uses an ST20-C105 core. For details, see the *ST20-C1 Core Instruction Set Reference Manual* (ADCS 7473696).

14 Interrupt system

14.1 Introduction

The STi7710 interrupt subsystem supports 16 prioritized interrupt levels routed to the ST20-C105, plus 3 levels routed to external pins (muxed on PIOs) for optional external use.

Overall there are 54 hardware interrupt sources: 49 internal sources (that is, generated by hardware blocks of the STi7710), plus 5 'external' interrupt sources: 2 from dedicated input pins, 2 shared with PIOs and an additional one generated upon detection of activity on the remote control (IRB) interface. The 49 internal sources are mapped on 64 interrupt numbers, 15 of which are reserved (see Section 14.4 on page 111).

The interrupt system allows an on-chip module or external interrupt pin to interrupt an active process so that an interrupt handling process can be run. Interrupts are signalled by one of the following:

- a signal on an external interrupt pin,
- a signal from an internal peripheral or subsystem,
- software asserting an interrupt in the pending register.

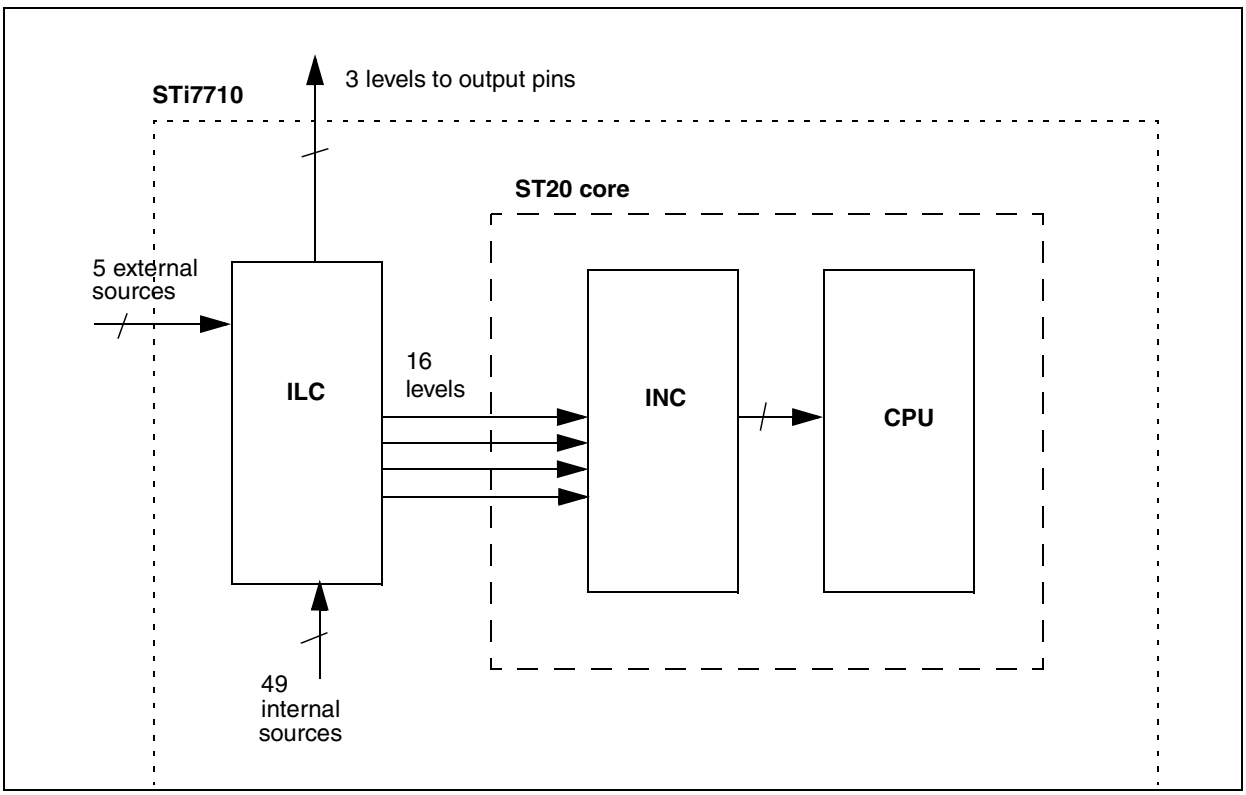
Interrupts are implemented by on-chip **interrupt level controller** (**ILC**) and **interrupt controller** (**INC**). It is important to understand the distinction between ILC and INC.

The ILC:

- allows any internal or external hardware interrupt to be assigned to any interrupt level,
- supports interrupt level sharing by different interrupt sources,
- multiplexes the 54 incoming interrupt sources onto the sixteen programmable interrupt level inputs of the INC. This multiplexing is controlled by software.

The INC (embedded in the ST20-C105 core) deals with the 16 interrupt levels to interrupt the CPU appropriately.

Figure 21: Interrupt system



14.2 Interrupt controller (INC)

The INC supports 16 prioritized interrupts as inputs, and manages the pending interrupts. This allows nested pre-emptive interrupts for real-time system design. Interrupt level 15 has the highest priority and interrupt level 0 has the lowest.

Each of the sixteen interrupt levels of the INC can be programmed with an interrupt trigger mode, using register INC_MODEn. The trigger mode can be set to be high or low level, or rising edge, falling edge or any edge sensitive.

Note: Due to a limitation in the current implementation of the INC, it is recommended to use only the edge-triggered mode.

Each of the sixteen interrupt levels can be programmed to be enabled or disabled by register INC_MASK. The default state of INC_MASK is all interrupt levels disabled. A corresponding level bit is set in register INC_PENDING if the interrupt signal from the ILC matches its trigger condition. If this is the highest numerical bit set in register INC_PENDING, the CPU executes the interrupt handler associated with that level by register INC_WDESCn and the INC_PENDING bit is then reset. If the level bit set in INC_PENDING is not the highest priority bit set, the bit remains set until it is the highest priority bit, then the CPU executes the associated interrupt handler for that level. Note the CPU only executes the interrupt handler and then clears the INC_PENDING bit if it is enabled in INC_MASK. Software can write to INC_PENDING to generate a software interrupt on any of the sixteen interrupt-levels.

Programming of registers INC_MASK, INC_PENDING and INC_MODEn is supported via the operating system run time library functions of OS20.

The INC also contains register INC_EXEC used by the INC logic to keep a record of which interrupt-level handler is currently executing on the CPU (or was previously executing before being pre-empted by a high-priority process, for low-priority interrupts) and which levels have been pre-empted by higher priority interrupt levels. This register can be read by user software, if required, but the register must never be written to as its behavior is undefined.

14.2.1 Interrupt vector table

The INC contains a table of pointers to interrupt handlers. There are 16 interrupt handlers, each controlled by a work-space descriptor register INC_WDESCn[15:0]. The table of pointer values contains a work-space pointer for each interrupt level.

Registers INC_WDESCn access the code, data and interrupt-save area of the interrupt handler. The position of INC_WDESCn in the interrupt table sets the priority of the interrupt.

The operating system run time library OS20 supports the setting and programming of the vector table.

14.2.2 Interrupt handlers

At any interruptible point in its execution, the CPU can receive an interrupt request from the INC. The CPU immediately acknowledges the request.

In response to receiving an interrupt, the CPU performs a procedure call to the process in the vector table. The state of the interrupted process is stored in the work-space of the interrupt handler. Each interrupt level has its own work-space.

For details of how to set up the exception handlers, refer to the *ST20-C1 User Manual*.

14.2.3 Interrupt latency

The interrupt latency depends on the type of data being accessed, and the position in memory of the interrupt handler and the interrupted process. This allows a trade-off of between fast internal SRAM memory and interrupt latency.

14.2.4 Pre-emption and interrupt priority

Each interrupt channel has an implied priority fixed by its place in the interrupt vector table. All interrupts cause scheduled processes of any priority to be suspended and the interrupt handler to be started. Once an interrupt has been sent from the controller to the CPU, the controller keeps a record of the current executing interrupt priority in register INC_EXEC. This is only cleared when the interrupt handler executes a return from interrupt (**iret**) instruction. Interrupts of a lower priority arriving are blocked by the INC until the interrupt priority is low enough for the routine to execute. An interrupt of a higher priority than the currently executing handler is passed to the CPU and causes the current handler to be suspended until the higher priority interrupt is serviced. In this way, interrupts can be nested and a higher priority interrupt always pre-empts a lower priority one.

Note: Deep nesting and the placing of frequent interrupts at high priority can result in systems where low-priority interrupts are never serviced or CPU time is consumed in nesting interrupt priorities instead of executing the interrupt handlers.

14.2.5 Restrictions on interrupt handlers

For optimum interrupt handling, the following restrictions are placed on interrupt handlers.

- Interrupt handlers must not deschedule.
- Interrupt handlers must not execute communication instructions. However, they may communicate with other processes through shared variables using the semaphore signal to synchronize.
- Interrupt handlers must not cause program traps. However, they may be trapped by a scheduler trap.

14.3 Interrupt level controller (ILC)

The ILC multiplexes 46 internal and 5 external interrupt source signals onto the 16 interrupt level inputs of the INC (which are the 16 ‘local’ interrupt levels at ILC output, levels 0x0 to 0xF in register ILC_PRIORITYn). In this way, it gives programmable control of the priority of the interrupt sources and extends the number of possible interrupts. In addition, interrupt steering allows routing of a further 3 levels to external pins for optional use by an external processor or for debug (which are the 3 ‘remote’ or ’external’ interrupt levels at ILC output, levels 0x8000 to 0x8002 in register ILC_PRIORITYn).

The incoming interrupt signals can be generated by on-chip subsystems or received from external pins. Software assigns a signal *n* to one of the 16 interrupt levels by writing the priority of the required input in register ILC_PRIORITYn. Each of the 51 interrupt sources of the ILC can be selectively enabled or disabled at source, by writing to register ILC_ENn. This is in addition to the individual masking of the 16 levels in the INC. Masking disables the interrupt source from generating an interrupt, without disabling all other interrupt sources mapped onto that interrupt level. This would be the case if register INC_MASK in the INC was used.

All internal interrupts are assumed to be level-sensitive and active high.

Each external interrupt source can be used to trigger an interrupt and can be programmed to trigger on rising or falling (or either) edge, or on the high or low logic level of the incoming interrupt source signal. This is controlled by writing to registers ILC_MODEn.

The default state of registers ILC_MODEn is no trigger, so these registers must be programmed if external interrupts are to be enabled. The default state of the enable registers is low, so these registers need to be programmed before internal interrupts can be serviced.

14.3.1 Setting trigger modes

Registers ILC_INPUT_INTn can be used to indicate the current logic state of all the interrupt sources. These registers are just buffered versions of the interrupt source signals before the trigger mode detection stage. They do not latch the signal, as ILC_STAn do, for interrupt sources defined with an edge-sensitive trigger mode. ILC_STAn is more useful because of this feature, as it can be read by the interrupt handler software routine to determine which interrupt sources have triggered.

For example, if the interrupt source is external and provides a pulse, the ILC has the interrupt source trigger mode set to be rising edge. On a rising edge, the corresponding bit in ILC_STAn is set high and remains set until explicitly cleared by the interrupt handler routine writing to the corresponding bit in ILC_CLR_STAn. However, if the pulse is short, by the time the interrupt handler is executed and has read ILC_INPUT_INTn, the pulse may have returned to a logic low and the bit would be read as zero. Thus the cause of interrupt can not be determined if more than one interrupt source is multiplexed onto the interrupt level.

So now, using ILC_STAn, interrupt sources of different types, including edge sensitive, may be multiplexed onto the same interrupt level in the INC.

14.3.2 CPU wake-up control

The ILC also has two registers mapped into its register address space that have no connection with normal interrupt operation. These registers control wake up of the CPU by an external interrupt pin when it has been put into low-power mode by the low-power controller module. Register ILC_WAKEUP_POL controls whether the 5 external interrupt sources (4 input pins plus wake-up pulse generated by activity on IRB) are active high or low to wake up the CPU. The setting of this register has no effect on the triggering of the external interrupt pins in the ILC. ILC_WAKEUP_EN is a mask register to enable or disable the external interrupt pins from waking up the CPU. Again, this has no effect on the masking of these interrupts in the ILC.

Thus, each level input of the INC responds to zero or more of the system interrupt sources. The ILC asserts interrupt output level *p* high when one or more of the input source interrupts with programmed priority equal to *p* are high. It is level sensitive. The ILC is effectively a programmable OR gate, which means that it is only possible to multiplex interrupt sources onto the same level in the INC if all the interrupt sources are of trigger type active high level. This is not an issue for on-chip module interrupt sources, as they are all of the active high level trigger type.

This means that for the external interrupt source pins, care must be taken if the interrupt signal source is not an active high level type. For example, if some external logic is connected to one of these pins and generates a pulse to request an interrupt, the trigger mode needs to be set to be rising or falling edge (in the INC level to which this interrupt source is mapped by the ILC). This then means that no other interrupt source can be mapped to this same level and that one level is reserved just for this external interrupt source.

In addition, the default state of ILC_PRIORITYn is zero, which means that, until programmed otherwise, all interrupt sources are mapped to interrupt level zero. This is not an issue for internal interrupt sources, as the interrupt has to be enabled in software from within the module. However it can cause problems for the external interrupt pins, that are connected directly into the ILC. For this reason, any unused external interrupt pins should be tied low to ground and not to Vdd. Otherwise, if interrupt level zero is used, spurious interrupts may occur.

Where two or more system interrupts are assigned to one interrupt handler routine, the software routine is able to ascertain the source of an interrupt by reading from ILC_INPUT_INTn, and examining the bits that correspond to the system interrupts assigned to that handler.

14.4 ILC interrupt mapping

Table 31 lists all internal interrupts and their mapping to the ILC. There are 64 interrupt numbers, of which 49 are used and 15 reserved. Internal interrupts are requested by functions integrated with the STi7710. A complete description of how each individual interrupt source is enabled and selected is included in the section covering the block that triggers the interrupt.

Table 31: Assignments at ILC input for internal interrupts

Interrupt name	Internal interrupt source number	Origin
PIO_0 interrupt	0	From Compare function of PIO Bank 0.
PIO_1 interrupt	1	From Compare function of PIO Bank 1.
PIO_2 interrupt	2	From Compare function of PIO Bank 2.
PIO_3 interrupt	3	From Compare function of PIO Bank 3.
PIO_4 interrupt	4	From Compare function of PIO Bank 4
PIO_5 interrupt	5	From Compare function of PIO Bank 5.
UART_0 interrupt	6	Depends on Tx/Rx status of UART 0.
UART_1 interrupt	7	Depends on Tx/Rx status of UART 1.
UART_2 interrupt	8	Depends on Tx/Rx status of UART 2.
UART_3 interrupt	9	Depends on Tx/Rx status of UART 3.
SSC_0 interrupt	10	Depends on Tx/Rx status of SSC 0.
SSC_1 interrupt	11	Depends on Tx/Rx status of SSC 1.
SSC_2 interrupt	12	Depends on Tx/Rx status of SSC 2.
SSC_3 interrupt	13	Depends on Tx/Rx status of SSC 3.
IRB interrupt	14	Depends on Tx/Rx status of Infrared Blaster.
TTXT interrupt	15	Used by the Teletext Interface to signal that a Teletext Out data transfer has completed.
C1_PWM interrupt	16	Fired by the PWM embedded in the C105 core when its timer function has elapsed.
DAA interrupt	17	Depends on DAA Tx/Rx status
PWM-Timer2 interrupt-A	18	Fired by the first timer of the programmable PWM-Timer2.
PWM-Timer2 interrupt-B	19	Fired by the first timer of the programmable PWM-Timer2.
MAFE interrupt	20	Depends on Tx/Rx status of MAFE interface.
AUD_CD_IRQ	21	From ADSC audio controller, Channel #o interrupt
AUD_PCM_IRQ	22	From ADSC audio controller, PCM buffer interrupt
AUD_MDEC_IRQ	23	From MMDSP audio decoder, main decoder interrupt
Reserved	24 - 25	-
LMU_IRQ	26	From LMU (Linear Median Unit) Deinterlacer
O_MAILBOX_FLAG	27	From FDMA
O_GP_OUTPUTS(0)	28	
BA_C1LMI_IRQ	29	From internal STBus analyzer, pattern matching interrupt. Debug and diagnosis usage.
DVP_O_IRQ	30	From DVP (D1 / ITU-R 656) input
EHCI_INT	31	From USB Host Controller
OHCI_INT	32	
HDCP_IRQ	33	From HDCP

Table 31: Assignments at ILC input for internal interrupts

Interrupt name	Internal interrupt source number	Origin
VOS_IRQ[0]	34	From Video Output Stage
VOS_IRQ[1]	35	
HDMI_IRQ	36	From HDMI
PTI_IRQ	37	From PTI
Reserved	38 - 41	-
DES_INTERRUPT	42	From DES encryption unit.
BA_FDMAPTI_IRQ	43	From internal STBus analyzer, pattern matching interrupt. Debug and diagnosis usage.
DAA_RING_DETECT	44	From DAA
Reserved	45	-
GLH_IRQ	46	From HD MPEG decoder
BLT_REG_IRQ	47	From Blitter engine
Reserved	48 - 52	-
MPEG_CLK_RECOV_IRQ	53	From MPEG Clock Recovery Unit
Reserved	54 - 59	-
UART0_LINE_ERROR	60	Flags error on UART 0 line
UART1_LINE_ERROR	61	Flags error on UART 1 line
UART2_LINE_ERROR	62	Flags error on UART 2 line
UART3_LINE_ERROR	63	Flags error on UART 3 line

Note: All internal interrupts routed to the ILC are active at high level (or have been pre-processed to comply with this rule).

Table 32 describes the assignment of external interrupts to the ILC. External interrupts originate from sources external to the STi7710. They can come directly through a hardware interrupt line, or indirectly when the interrupt is generated internally upon detection of a specific external activity.

Table 32: Assignment at ILC input for external interrupts

Interrupt name	External interrupt source number	Origin
IRB_WAKEUP_INT	0	From wake-up pulse generator, triggered by detection of activity on the Remote Control (IRB) interface.
EXT_IRQ_0	1	External interrupt, multiplexed on PIO (alternate input).
EXT_IRQ_1	2	External interrupt, multiplexed on PIO (alternate input).
EXT_IRQ_2	3	External interrupt, dedicated input pin.
EXT_IRQ_3	4	External interrupt, dedicated input pin.
-	5-7	External interrupt, not bristled out; clamped to logic 0 internally.

Note: The ILC can be programmed to accommodate level-triggered or edge-triggered EXTERNAL interrupts of any polarity.

15 Interrupt system registers

Interrupt controller (**INC**) register addresses are provided as *INCBaseAddress* + offset.

The *INCBaseAddress* is:

0x3000 4000.

Interrupt level controller (**ILC**) addresses are provided as *ILCBaseAddress* + offset.

The *ILCBaseAddress* is:

0x2000 0000.

Table 33: INC register summary

Register	Description	Offset	Type
INC_REV_ID	INC revision and identification code	0x00	RO
INC_LEVELS	Number of interrupt levels supported	0x04	RO
INC_PENDING	Interrupt level pending status	0x10	R/W
INC_SET_PENDING	Set interrupt level pending status bits	0x14	WO
INC_CLR_PENDING	Clear interrupt level pending status bits	0x18	WO
INC_MASK	Interrupt level mask	0x20	R/W
INC_SET_MASK	Set interrupt level mask bits	0x24	WO
INC_CLR_MASK	Clear interrupt level mask bits	0x28	WO
INC_GLOBAL_MASK	Global interrupt mask	0x30	R/W
INC_EXEC	Executing interrupt level status	0x40	R/W[a]
INC_SET_EXEC	Set executing interrupt level status bits	0x44	WO[a]
INC_CLR_EXEC	Clear executing interrupt level status bits	0x48	WO[a]
INC_WDESC0	WDESC for interrupt level 0	0x100	R/W
INC_WDESC1	WDESC for interrupt level 1	0x104	R/W
INC_WDESC2	WDESC for interrupt level 2	0x108	R/W
INC_WDESC3	WDESC for interrupt level 3	0x10C	R/W
INC_WDESC4	WDESC for interrupt level 4	0x110	R/W
INC_WDESC5	WDESC for interrupt level 5	0x114	R/W
INC_WDESC6	WDESC for interrupt level 6	0x118	R/W
INC_WDESC7	WDESC for interrupt level 7	0x11C	R/W
INC_WDESC8	WDESC for interrupt level 8	0x120	R/W
INC_WDESC9	WDESC for interrupt level 9	0x124	R/W
INC_WDESC10	WDESC for interrupt level 10	0x128	R/W
INC_WDESC11	WDESC for interrupt level 11	0x12C	R/W
INC_WDESC12	WDESC for interrupt level 12	0x130	R/W
INC_WDESC13	WDESC for interrupt level 13	0x134	R/W
INC_WDESC14	WDESC for interrupt level 14	0x138	R/W

Table 33: INC register summary

Register	Description	Offset	Type
INC_WDESC15	WDESC for interrupt level 15	0x13C	R/W
INC_MODE0	Trigger mode for interrupt level 0	0x180	R/W
INC_MODE1	Trigger mode for interrupt level 1	0x184	R/W
INC_MODE2	Trigger mode for interrupt level 2	0x188	R/W
INC_MODE3	Trigger mode for interrupt level 3	0x18C	R/W
INC_MODE4	Trigger mode for interrupt level 4	0x190	R/W
INC_MODE5	Trigger mode for interrupt level 5	0x194	R/W
INC_MODE6	Trigger mode for interrupt level 6	0x198	R/W
INC_MODE7	Trigger mode for interrupt level 7	0x19C	R/W
INC_MODE8	Trigger mode for interrupt level 8	0x1A0	R/W
INC_MODE9	Trigger mode for interrupt level 9	0x1A4	R/W
INC_MODE10	Trigger mode for interrupt level 10	0x1A8	R/W
INC_MODE11	Trigger mode for interrupt level 11	0x1AC	R/W
INC_MODE12	Trigger mode for interrupt level 12	0x1B0	R/W
INC_MODE13	Trigger mode for interrupt level 13	0x1B4	R/W
INC_MODE14	Trigger mode for interrupt level 14	0x1B8	R/W
INC_MODE15	Trigger mode for interrupt level 15	0x1BC	R/W

a. Do not write to these registers, which are intended for OS20 use only.

Table 34: ILC register summary

Register	Description	Offset	Type
ILC_INPUT_INTn	Input interrupts 0 to 2	0x080 to 0x088	RO
ILC_STAn	Status 0 to 2	0x200 to 0x208	RO
ILC_CLR_STAn	Clear status locations 0 to 2	0x280 to 0x288	WO
ILC_ENn	Interrupt enable 0 to 2	0x400 to 0x408	R/W
ILC_CLR_ENn	Clear enable locations 0 to 2	0x480 to 0x488	WO
ILC_SET_ENn	Set enable locations 0 to 2	0x500 to 0x508	WO
ILC_WAKEUP_EN	Wake up enable 0 to 2	0x600 to 0x608	R/W
ILC_WAKEUP_POL	Wake up polarity 0 to 2	0x680 to 0x688	R/W
ILC_PRIORITYn	Interrupt priority 0 to 63	0x800 to 0x9F8	R/W
ILC_EXT_PRIORITYn, ILC_MODEn	ILC priority 0 to 7, ILC mode 0 to 7	0xA00 to 0xA3C	R/W

Note: All addresses not listed are reserved and must not be written to.

15.1 INC register descriptions

INC_REV_ID Interrupt source state

31	30	29	28	27	26	25	24	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_ID																															

Address: *INCBaseAddress* + 0x00

Type: RO

Reset: Undefined

Description: This register shows the identification and revision state code of the interrupt controller.

[31:0] **REV_ID**: revision and version number

INC_LEVELS Interrupt levels supported

31	30	29	28	27	26	25	24	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_LEVEL			

Address: *INCBaseAddress* + 0x04

Type: RO

Reset: 0x0F

Description: This register shows the number of interrupt levels the interrupt controller can support. It allows OS20 software to determine the size of the interrupt controller at run-time.

[31:4] **Reserved**

[3:0] **INC_LEVEL**: number of interrupt levels supported

INC_PENDING Interrupt pending 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																PENDINGn															

Address: *INCBaseAddress* + 0x10

Type: R/W

Reset: 0xXXXX 0000

Description: This register shows if an interrupt level, INT_LEVEL[n] (where n = 0 to 15), request is pending as a result of a trigger condition being met. That is, the INT_LEVEL[n] signal matches the trigger condition programmed into INC_MODE[n]. All bits are independent, so that several bits can be set at the same time. Another triggering condition while the corresponding bit is set has no effect. The triggering condition is independent of the INC_MASK register. The most significant set bit, corresponding to the highest priority interrupt level, is reset when the interrupt controller is committed to launching an interrupt process on the CPU.

[31:16] Reserved

[15:0] **PENDINGn**: interrupt *n* pending.
Bit *n* corresponding to interrupt level INT_LEVEL[*n*] is set to 1 when the interrupt level signal matches the programed trigger mode condition set in INC_MODEn.
Individual bits can be set to 1 to explicitly trigger an interrupt request.

INC_CLR_PENDING Clear interrupt pending 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																CLRn															

Address: *INCBaseAddress* + 0x18

Type: WO

Reset: 0xXXXX 0000

Description: This write-only register is used to clear interrupt pending status bits in INC_PENDING corresponding to interrupt levels, INT_LEVEL[n] (where n = 0 to 15), which have been set as a result of a trigger condition event.

[31:16] Reserved

[15:0] **CLRn**: clear interrupt *n* pending status. Writing a 1 to bit *n* clears the corresponding status bit in INC_PENDING. Writing a 0 has no effect.

INC_SET_PENDING Set interrupt pending 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																SETn															

Address: *INCBaseAddress* + 0x14

Type: WO

Reset: 0xXXXX 0000

Description: This write-only register is used to set interrupt pending status bits in INC_PENDING corresponding to interrupt levels, INT_LEVEL[*n*] (where *n* = 0 to 15), which is used to explicitly trigger an interrupt response.

[31:16] Reserved

[15:0] **SETn**: set interrupt *n* pending status. Writing a 1 to bit[*n*] sets the corresponding status bit in INC_PENDING, thus generating a pending interrupt request on level[*n*]. Writing 0 has no effect.

INC_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
Reserved																MASKn															

Address: *INCBaseAddress* + 0x20

Type: R/W

Reset: 0xXXXX 0000

Description: This register is used to enable the various interrupt levels, INT_LEVEL[*n*] (where n = 0 to 15), to generate an interrupt to the CPU.

[31:16] Reserved

[15:0] **MASKn**: mask interrupt level *n*. Setting a bit[*n*] corresponding to interrupt level, INT_LEVEL[*n*], to 0 masks a pending interrupt in INC_PENDING to generate an interrupt to the CPU.

INC_CLR_MASK Clear 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																CLRMn															

Address: *INCBaseAddress* + 0x28

Type: WO

Reset: 0xXXXX 0000

Description: This write-only register is used to clear the interrupt mask bits in INC_MASK corresponding to interrupt sources, INT_LEVEL[n] (where n = 0 to 15).

[31:16] Reserved

[15:0] **CLRMn**: clear interrupt level *n* mask. Writing a 1 to bit[*n*] clears the corresponding mask bit in INC_MASK.
Writing a 0 has no effect.

INC_SET_MASK Set 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																SETMn															

Address: *INCBaseAddress* + 0x24

Type: WO

Reset: 0xXXXX 0000

Description: This write-only register is used to set the interrupt mask bits in INC_MASK corresponding to interrupt sources, INT_LEVEL[*n*] (where n = 0 to 15).

[31:16] Reserved

[15:0] **SETMn**: set interrupt level *n* mask. Writing a 1 to bit[*n*] sets the corresponding mask bit in INC_MASK. Writing a 0 has no effect.

INC_GLOBAL_MASK Interrupt global 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. Reset undefined																															GLB_MASK

Address: *INCBaseAddress* + 0x30

Type: R/W

Reset: 0xXXXX XXX0

Description: This register has a similar function to INC_MASK, it is used to enable all interrupt levels.

[31:1] Reserved. Reset undefined.

[0] **GLB_MASK**: global interrupt mask. Setting this bit to 1 enables all interrupt levels. Reset: 0

INC_EXEC Interrupt executing 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																EXECn															

Address: *INCBaseAddress* + 0x40

Type: R/W

Reset: 0xXXXX 0000

Description: This register keeps track of the currently executing and pre-empted interrupts. It allows the CPU to determine the highest executing interrupt level and also if previous interrupts have themselves been interrupted. The MSB (highest interrupt priority level) set is cleared when a **Done** response is returned from the CPU. The protocol, used between the interrupt controller and the CPU ensures that the correct bit is cleared.

[31:16] **Reserved**

[15:0] **EXECn**: interrupt *n* executing. A bit[*n*] corresponding to interrupt level, INT_LEVEL[n], is set to 1 when the CPU starts running code for that interrupt. Individual bits can be written to 1.

Note: Do not write to this register. This register is intended for OS20 use only.

INC_CLR_EXEC Clear interrupt executing 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																CLRn															

Address: *INCBaseAddress* + 0x48

Type: WO

Reset: 0xXXXX 0000

Description: This register is used to clear interrupt executing status bits in INC_EXEC corresponding to interrupt levels, INT_LEVEL[*n*] (where n = 0 to 15).

[31:16] Reserved

[15:0] **CLRn**: clear interrupt *n* executing status. Writing a 1 to bit[*n*] clears the corresponding status bit in INC_EXEC. Writing a 0 has no effect.

Note: This register is intended for OS20 use only and is not supported by the STi7710. Do not write anything to this register other than 0x0000.

INC_SET_EXEC Set interrupt executing 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																SETn															

Address: *INCBaseAddress* + 0x44

Type: WO

Reset: 0xXXXX 0000

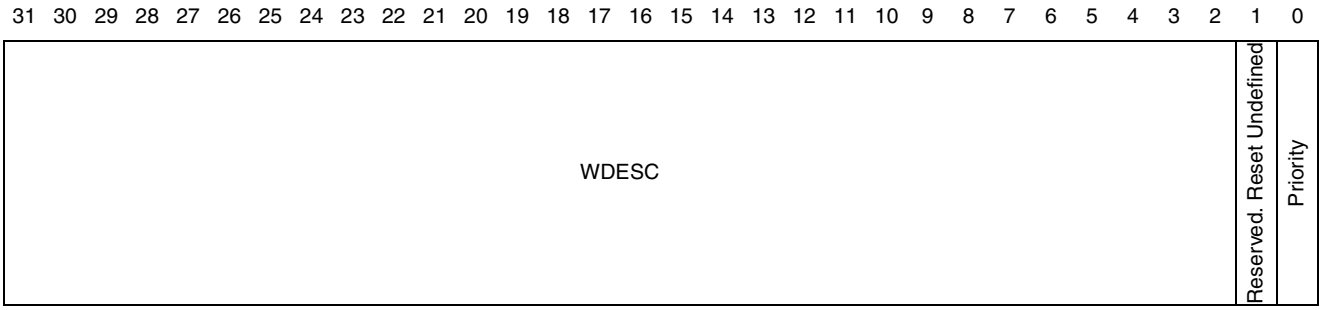
Description: This register is used to set interrupt executing status bits in INC_EXEC corresponding to interrupt levels, INT_LEVEL[n] (where n = 0 to 15).

[31:16] Reserved

[15:0] **SETn**: set interrupt *n* executing status. Writing a 1 to bit[*n*] sets the corresponding status bit in INC_EXEC, thus generating a pending interrupt request on level[*n*]. Writing a 0 has no effect.

Note: This register is intended for OS20 use only and is not supported by the STi7710. Do not write anything to this register other than 0x0000.

INC_WDESCn Interrupt handler work-space descriptor



Address: *INCBaseAddress* + 0x100 to 0x13C

Type: R/W

Reset: 0xXXXX XXX0

Description: This register set with one register for each interrupt level, INT_LEVEL[n] (where n = 0 to 15) is programmed with the address value of the interrupt handler work-space pointer.

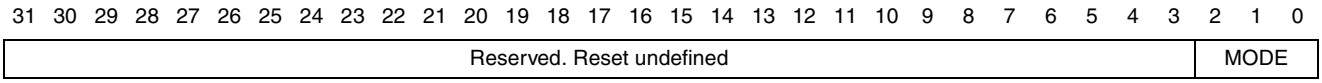
Note: This register is normally managed by OS20, because the interrupt controller does not check for valid HANDLERWPTR.

[31:2] **WDESC**: work-space descriptor. Used to set the address of the interrupt handler work-space pointer, HANDLERWPTR. Reset undefined.

[1] Reserved. Reset undefined.

[0] **PRIORITY**: CPU task priority. Used to determine the CPU task priority.
0: high priority (default). 1: low priority.

INC_MODEn Interrupt trigger mode



Address: *INCBaseAddress* + 0x180 to 0x1BC

Type: R/W

Reset: 0xXXXX XXX0

Description: This register set is used to configure the trigger mode for each of the interrupt levels, INT_LEVEL[n] (where n = 0 to 15).

[31:3] Reserved. Reset undefined.

[2:0] **MODE**: interrupt source trigger mode. Used to set the trigger mode for INT_LEVEL[n].
000: no trigger (default). 001: high level.
010: low level. 011: rising edge.
100: falling edge. 101: any edge.
Others: no trigger.

Note: Level triggering differs from edge triggering in that if the input is held at the triggering level, a continuous stream of interrupts is generated.

15.2 ILC registers

ILC_INPUT_INTn Input interrupts 0 to 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
0x088	Reserved	INT[71:64]
0x084	INT[63:32]	
0x080	INT[31:0]	

Address: *ILCBaseAddress* + 0x080 (ILC_INPUT_INT0), 0x084, 0x88 (ILC_INPUT_INT2)
Type: RO
Reset: 0x00
Description: The synchronized version of all the interrupt numbers can be read from this register.

ILC_STAn Interrupt status 0 to 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
0x208	Reserved	INT[71:64]
0x204	INT[63:32]	
0x200	INT[31:0]	

Address: *ILCBaseAddress* + 0x200, 0x204, 0x208
Type: RO
Reset: 0x00
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.

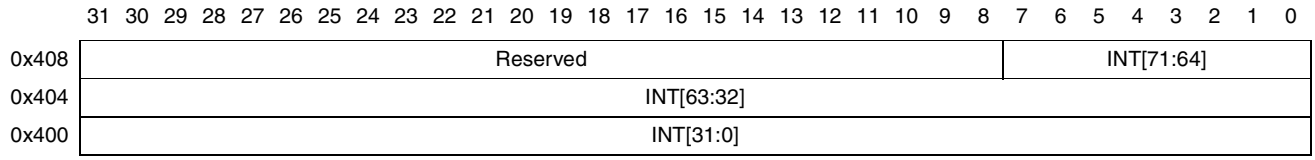
- If the corresponding interrupt number is internal (synchronous), then the interrupt number should be at logic 1.
- If the corresponding interrupt number is external (asynchronous), then the interrupt number should match the programmed trigger condition.

ILC_CLR_STAn Clear status locations 0 to 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
0x288	Reserved	INT[71:64]
0x284	INT[63:32]	
0x280	INT[31:0]	

Address: *ILCBaseAddress* + 0x280, 0x284, 0x288
Type: WO
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 performed on this location. A clear bit operation is a write operation with logic 1 on the data bus corresponding to the locations which have to be cleared.

ILC_ENn Interrupt enable 0 to 2



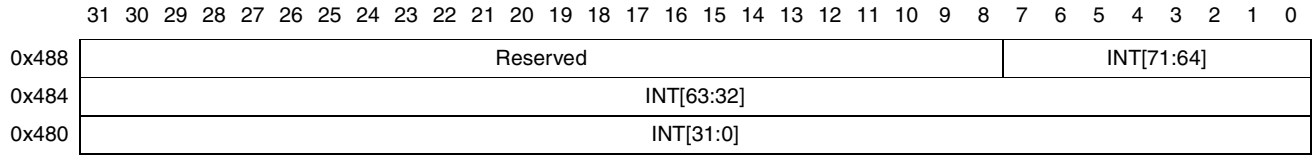
Address: *ILCBaseAddress* + 0x400, 0x404, 0x408

Type: R/W

Reset: 0x00

Description: Interrupt generation from an interrupt number is enabled only if the corresponding bit in the enable register is set to logic 1.

ILC_CLR_ENn Clear enable locations 0 to 2



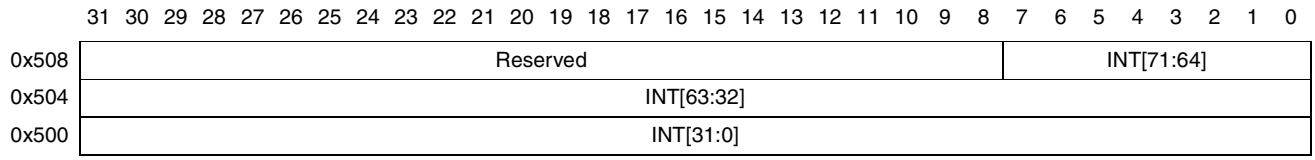
Address: *ILCBaseAddress* + 0x480, 0x484, 0x488

Type: WO

Reset: Undefined

Description: Any bit in the enable register can be cleared by performing a clear bit operation on the appropriate location.

ILC_SET_ENn Set enable locations 0 to 2



Address: *ILCBaseAddress* + 0x500, 0x504, 0x508

Type: WO

Reset: Undefined

Description: Any bit in the enable register can be set by performing a set bit operation on the appropriate location (a set bit operation is a write operation with logic 1 on the data bus corresponding to the location which has to be set).

ILC_WAKEUP_EN Wake up 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																											EXT[4:0]					

Address: *ILCBaseAddress* + 0x600

Type: R/W

Reset: 0x00

Description: This register is used to enable 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 1.

ILC_WAKEUP_POL Wake up level

31	30	29	28	27	26	25	24	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																											EXT[4:0]					

Address: *ILCBaseAddress* + 0x680

Type: R/W

Reset: 0x01

Description: This register is 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, a wake up by interrupt is generated only if the corresponding external interrupt is at 1. Writing 0 generates the wake up by interrupt when corresponding external interrupt pin is at 0. The contents of this register are set to logic 1 on reset.

ILC_PRIORITYn Interrupt priority 0 to 63

31	30	29	28	27	26	25	24	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																PRIORITYn[15:0]															

Address: ILCBaseAddress + (see below)

Table 35: Internal interrupt priority registers

ILC_PRIORITY0	0x800	ILC_PRIORITY16	0x880	ILC_PRIORITY32	0x900	ILC_PRIORITY48	0x980
ILC_PRIORITY1	0x808	ILC_PRIORITY17	0x888	ILC_PRIORITY33	0x908	ILC_PRIORITY49	0x988
ILC_PRIORITY2	0x810	ILC_PRIORITY18	0x890	ILC_PRIORITY34	0x910	ILC_PRIORITY50	0x990
ILC_PRIORITY3	0x818	ILC_PRIORITY19	0x898	ILC_PRIORITY35	0x918	ILC_PRIORITY51	0x998
ILC_PRIORITY4	0x820	ILC_PRIORITY20	0x8A0	ILC_PRIORITY36	0x920	ILC_PRIORITY52	0x9A0
ILC_PRIORITY5	0x828	ILC_PRIORITY21	0x8A8	ILC_PRIORITY37	0x928	ILC_PRIORITY53	0x9A8
ILC_PRIORITY6	0x830	ILC_PRIORITY22	0x8B0	ILC_PRIORITY38	0x930	ILC_PRIORITY54	0x9B0
ILC_PRIORITY7	0x838	ILC_PRIORITY23	0x8B8	ILC_PRIORITY39	0x938	ILC_PRIORITY55	0x9B8
ILC_PRIORITY8	0x840	ILC_PRIORITY24	0x8C0	ILC_PRIORITY40	0x940	ILC_PRIORITY56	0x9C0
ILC_PRIORITY9	0x848	ILC_PRIORITY25	0x8C8	ILC_PRIORITY41	0x948	ILC_PRIORITY57	0x9C8
ILC_PRIORITY10	0x850	ILC_PRIORITY26	0x8D0	ILC_PRIORITY42	0x950	ILC_PRIORITY58	0x9D0
ILC_PRIORITY11	0x858	ILC_PRIORITY27	0x8D8	ILC_PRIORITY43	0x958	ILC_PRIORITY59	0x9D8
ILC_PRIORITY12	0x860	ILC_PRIORITY28	0x8E0	ILC_PRIORITY44	0x960	ILC_PRIORITY60	0x9E0
ILC_PRIORITY13	0x868	ILC_PRIORITY29	0x8E8	ILC_PRIORITY45	0x968	ILC_PRIORITY61	0x9E8
ILC_PRIORITY14	0x870	ILC_PRIORITY30	0x8F0	ILC_PRIORITY46	0x970	ILC_PRIORITY62	0x9F0
ILC_PRIORITY15	0x878	ILC_PRIORITY31	0x8F8	ILC_PRIORITY47	0x978	ILC_PRIORITY63	0x9F8

Table 36: External interrupt priority registers

ILC_PRIORITY64	0xA00	ILC_PRIORITY66	0xA10	ILC_PRIORITY68	0xA20	ILC_PRIORITY69	0xA30
ILC_PRIORITY65	0xA08	ILC_PRIORITY67	0xA18	ILC_PRIORITY69	0xA28	ILC_PRIORITY70	0xA38

Type: R/W

Reset: 0x00

Description: The priority register assigns the interrupt number to one of the available interrupt levels. The assignment is done by writing the appropriate word into the priority register.

To map an interrupt number on the interrupt level of a local processor (ST20 core), a value between 0x0000 to 0x7FFF needs to be written. For example, the ST20 core has 16 interrupt levels, values between 0x0000 to 0x000F are valid. Other values from 0x0010 to 0x7FFF are invalid.

To map an interrupt number on the interrupt level of an external processor, a value between 0x8000 to 0xFFFF has to be written. The ILC can only map on to four interrupt levels of an external processor, therefore the values between 0x8000 (SYSITRQ[0]) to 0x8003 (SYSITRQ[3]) are valid. Other values from 0x8007 to 0xFFFF are invalid.

For example:

If bit 15 is set to 1, bits 0 and 1 are used to determine which of the four external interrupts is to be set.

If bit 15 is set to 0, bits 0 to 3 are used to determine which of the 16 interrupt levels is to be set.

Bits 14:4 must be set to 0 at all times

ILC_MODEn ILC mode 0 to 7

31	30	29	28	27	26	25	24	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																													MODEn		

Address: *ILCBaseAddress* + (see below)

Table 37: ILC mode registers

External interrupts	
ILC_MODE0	0xA04
ILC_MODE1	0xA0C
ILC_MODE2	0xA14
ILC_MODE3	0xA1C

Unused interrupt lines	
ILC_MODE4	0xA24
ILC_MODE5	0xA2C
ILC_MODE6	0xA34
ILC_MODE7	0xA3C

Type: R/W

Reset: 0x00

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 below.

[31:3] **Reserved**

[2:0] **MODEn**:

0x00: No trigger mode
0x01: High level trigger mode
0x02: Lowlevel trigger mode
0x03: Rising edge trigger mode
0x04: Falling edge trigger mode
0x05: Any edge trigger mode
0x06: No trigger mode
0x07: No trigger mode

16 Memory map

The memory space is divided into four main regions:

- Region 3: Addresses which map onto the External Memory Interface
- Region 2: Addresses which map onto the peripheral configuration registers. This region is split into two areas:
  - 512 Mbytes (addresses from 0x0000 0000 to 0x2FFF FFFF): maps on-chip peripherals accessible by any initiator (not just the ST20).
  - 512 Mbytes (addresses from 0x3000 0000 to 0x3FFF FFFF): maps peripherals accessible from the ST20 only. This area comprises of two zones:
    - from 0x3000 0000 up to 0x307F FFFF - ST20-C105 internal core peripherals
    - from 0x3080 0000 up to 0x3FFF FFFF - other on-chip peripherals)
- Region 1: off-chip SDRAM connected to the Local Memory Interface (LMI).
- Region 0: Addresses which map onto the internal ST20-C105 SRAM. Only the first 8 kbytes are used (addresses from 0x8000 0000 - 0x8000 1FFF).

Table 38: Top level memory map

Regions	Address Range	size (byte)	Region Type	Sub-region type
3	0x7FFF FFFF - 0x4800 0000	768 M	**EMI Memory Space**	EMI
	0x47FF FFFF - 0x4000 0000	256 M		EMI - ST20 Boot Entry
2	0x3FFF FFFF - 0x3800 0000	128 M	**On-chip peripherals configuration registers** (access from ST20 only)	(see Table 39)
	0x37FF FFFF - 0x3000 5000	~128 M (-20 K)	unused	unused
	0x3000 4FFF - 0x3000 4000	4 K	**ST20 C105 internal core peripherals** (access from ST20 only)	Interrupt Controller
	0x3000 3FFF - 0x3000 3000	4 K		PWM4 Timer
	0x3000 2FFF - 0x3000 2000	4 K		I-Cache Config
	0x3000 1FFF - 0x3000 1000	4 K		D-Cache Config
	ox3000 0FFF - 0x3000 0000	4 K		DCU
	0x2FFF FFFF - 0x2010 7000	~255 M	unused	unused
	0x2010 6FFF - 0x2010 4000	12 K	**On-Chip peripherals control** (no access restriction)	Interconnect Config (see Table 39)
	0x2010 3FFF - 0x2010 3000	4 K		Video Output Stage Control (see Table 40)
	0x2010 2FFF - 0x2010 2000	4 K		EMI Config
	0x2010 1FFF - 0x2010 1000	4 K		TS merger (SWTS data)
	0x2010 0FFF - 0x2010 0000	4 K		Audio compressed data (AUDIO_CD_FIFO)
	0x200F FFFF - 0x2000 0000	1 M		COMMs registers (see Table 41)
	0x1FFF FFFF - 0x0000 0000	512 M	unused	unused
1	0xFFFF FFFF - 0xCD00 0000	832 M	unused	unused
	0xCCFF FFFF - 0xCC00 0000	16 M	**LMI Control**	LMI config registers
	0xCBFF FFFF - 0xC000 0000	192 M	**LMI Memory Space**	192 MBytes of local memory (SDRAM)
0	0xBFFF FFFF - 0x8000 1000	~1 G (-4 K)	Reserved	RESERVED
	0x8000 0FFF - 0x8000 0000	4 K	**Internal SRAM**	ST20-C105 SRAM

Table 39: Region 2 - On-chip peripherals address map

Address Range		Size (Bytes)	Function
Start	End		
0x3821 5000	0x3821 5FFF	4 K	Compositor
0x3821 4000	0x3821 4FFF	4 K	DENC
0x3821 0000	0x3821 3FFF	16 K	Audio data controller config
0x3820 4000	0x3820 4FFF	4 K	LMU
0x3820 3000	0x3820 3FFF	4 K	ID Display (Intermediate definition display)
0x3820 2000	0x3820 2FFF	4 K	HD Disp
0x3820 1000	0x3820 1FFF	4 K	Sub_micro System Config Glue Logic
0x3820 0000	0x3820 0FFF	4 K	DVP
0x3810 0000	0x381F FFFF	1 M	USB
0x3805 0000	0x3805 7FFF	32 K	FDMA
0x3804 0000	0x3804 0FFF	4 K	Reserved
0x3803 0000	0x3803 7FFF	32 K	
0x3802 0000	0x3802 0FFF	4 K	TS Merger Config.
0x3801 0000	0x3801 FFFF	64 K	PTI
0x3800 3000	0x3800 3FFF	4 K	Blitter
0x3800 2000	0x3800 2FFF	4 K	MPEG Video decoder
0x3800 1000	0x3800 1FFF	4 K	Sub_transport System Config Glue Logic
0x3800 0000	0x3800 0FFF	4 K	Clock Generator config register

Table 40 describes the VOS configuration register map:

Table 40: Region 2 - VOS configuration address map

Address Range		size (bytes)	Function	Group
Start	End			
0x2010 3600	0x2010 37FF	512	HDCP	VOS configuration
0x2010 3200	0x2010 35FF	1 K	HDMI	
0x2010 3000	0x2010 31FF	512	VOS	

Table 41 describes the Communications address map:

Table 41: Region 2 - Communication Peripherals address map

Address Range		size (Kb)	Function	Group
Start	End			
0x2006 9FFF	0x200F FFFF	608	Reserved	Reserved
0x2006 9000	0x2006 9FFF	28	Reserved	COMMs_Glue
0x2006 8000	0x2006 8FFF	4	COMMs_Glue_Config	
0x2006 1000	0x2006 7FFF	28	Reserved	TTXT
0x2006 0000	0x2006 0FFF	4	TTXT0	
0x2005 9000	0x2005 FFFF	28	Reserved	MAFE
0x2005 8000	0x2005 8FFF	4	MAFE0	
0x2004 A000	0x2005 7FFF	56	Reserved	SCCLKGEN
0x2004 9000	0x2004 9FFF	4	SCCLKGEN1	
0x2004 8000	0x2004 8FFF	4	SCCLKGEN0	
0x2004 4000	0x2004 7FFF	16	Reserved	Synchronous serial controller
0x2004 3000	0x2004 3FFF	4	SSC3	
0x2004 2000	0x2004 2FFF	4	SSC2	
0x2004 1000	0x2004 1FFF	4	SSC1	
0x2004 0000	0x2004 0FFF	4	SSC0	
0x2003 4000	0x2003 FFFF	56	Reserved	UART
0x2003 3000	0x2003 3FFF	4	UART3	
0x2003 2000	0x2003 2FFF	4	UART2	
0x2003 1000	0x2003 1FFF	4	UART1	
0x2003 0000	0x2003 0FFF	4	UART0	
0x2002 6000	0x2002 FFFF	40	Reserved	PIO
0x2002 5000	0x2002 5FFF	4	PIO5	
0x2002 4000	0x2002 4FFF	4	PIO4	
0x2002 3000	0x2002 3FFF	4	PIO3	
0x2002 2000	0x2002 2FFF	4	PIO2	
0x2002 1000	0x2002 1FFF	4	PIO1	
0x2002 0000	0x2002 0FFF	4	PIO0	
0x2001 9000	0x2001 FFFF	28	Reserved	IR
0x2001 8000	0x2001 8FFF	4	IRB	
0x2001 0FFF	0x2001 7FFF	28	Reserved	PWM-Timer2
0x2001 0000	0x2001 0FFF	4	PWM-Timer2	
0x2000 9000	0x2000 FFFF	28	Reserved	LPC
0x2000 8000	0x2000 8FFF	4	LPC	
0x2000 1000	0x2000 7FFF	28	Reserved	ILC
0x2000 0000	0x2000 0FFF	4	ILC3	

17 Cache

17.1 Glossary

Bank a bank of memory refers to one quarter of region 3, the EMI region, that is, 256 Mbytes.

Block a block of memory refers to one 512 Kbyte chunk of memory.

Cache a memory device that speeds up access to the next level of memory hierarchy.

Coherency if the cache and the next level of the memory hierarchy have the same data stored for a given memory address, then they are coherent.

Dirty a dirty line in the cache has been written to since it was last loaded from memory and is thus different from the same location in the next level of memory.

Direct-Mapped in a direct-mapped cache, each location in memory can only be cached in one place in the cache. This means that implementation is simpler and operation faster but that misses are more likely.

Establish the operation where data is written in to the cache storage arrays

Flush to write to the next level of memory any data in the cache that is incoherent.

Hit an access is a hit when the request can be serviced wholly within the cache.

Invalidate to set the cache to a state where the next access cannot be a hit.

Line the line is the part of the address used to locate the tag and data in the RAMs. A cache line is a collection of words, all of which are moved in and out of the cache at once.

Miss an access is a miss when the request cannot be serviced without reference to the next level of memory.

Refill the process of reading the contents of a line that is not in the cache from the next level of memory and updating the cache with that data.

Region a region of memory refers to one quarter of the total memory space, that is,1 Gbyte.

Tag the tag is the top part of the address that is compares to determine a hit or a miss. This shows which location in memory the data for a given line is cached for.

Valid a valid line is one for which there is a Tag stored in the cache

Victim the victim of a cache miss is that data in the same line that was stored in the cache

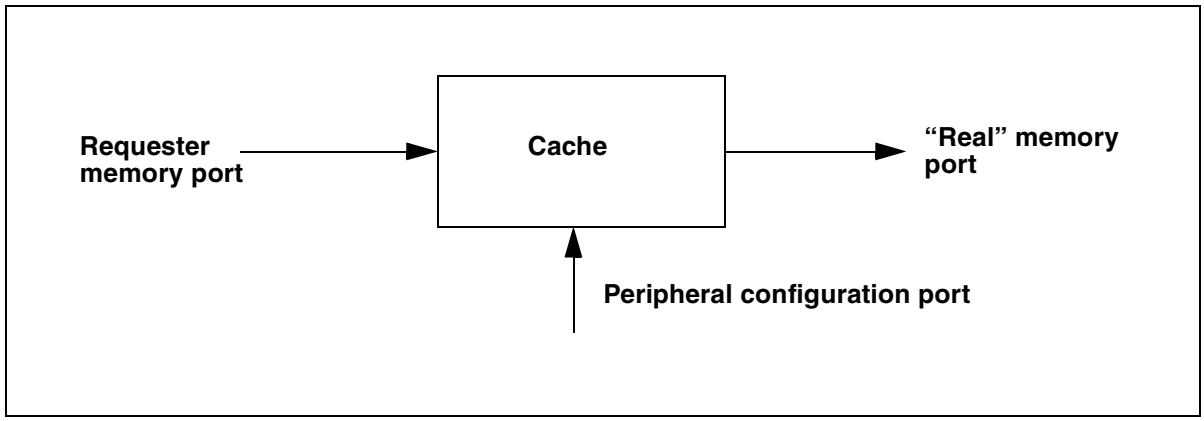
Word the word is the part of the address that specifies which word within a line is being accessed.

Writeback the process of taking the data that is stored in the cache for a given line and writing it to the next level of memory before the refill overwrites it.

17.2 Cache introduction

This cache is common to several ST20-based chips. Once set up, it is functionally transparent. However, its timing characteristics differ from those of direct memory access: accesses are on average faster. The cache also behaves like a retiming stage in certain circumstances, passing through a request verbatim. It does this for device accesses and also for certain areas of memory, which are configured by the cache's peripheral port.

Figure 22: Cache interfaces



The non-cacheable areas are those that contain peripherals, shared memory or DMA sources and destinations. The configuration of this cacheability is described in Section 18.

This cache differs from older ST20 caches in a number of functional ways:

- the registers for configuration are internal,
- it checks cacheability internally and it cannot become an SRAM.

It also differs structurally: there is no distinction between "cache" and "refill engine" as there was previously: both of these functions are combined. The specification is as generic as possible in order to define an architecture that is process-invariant.

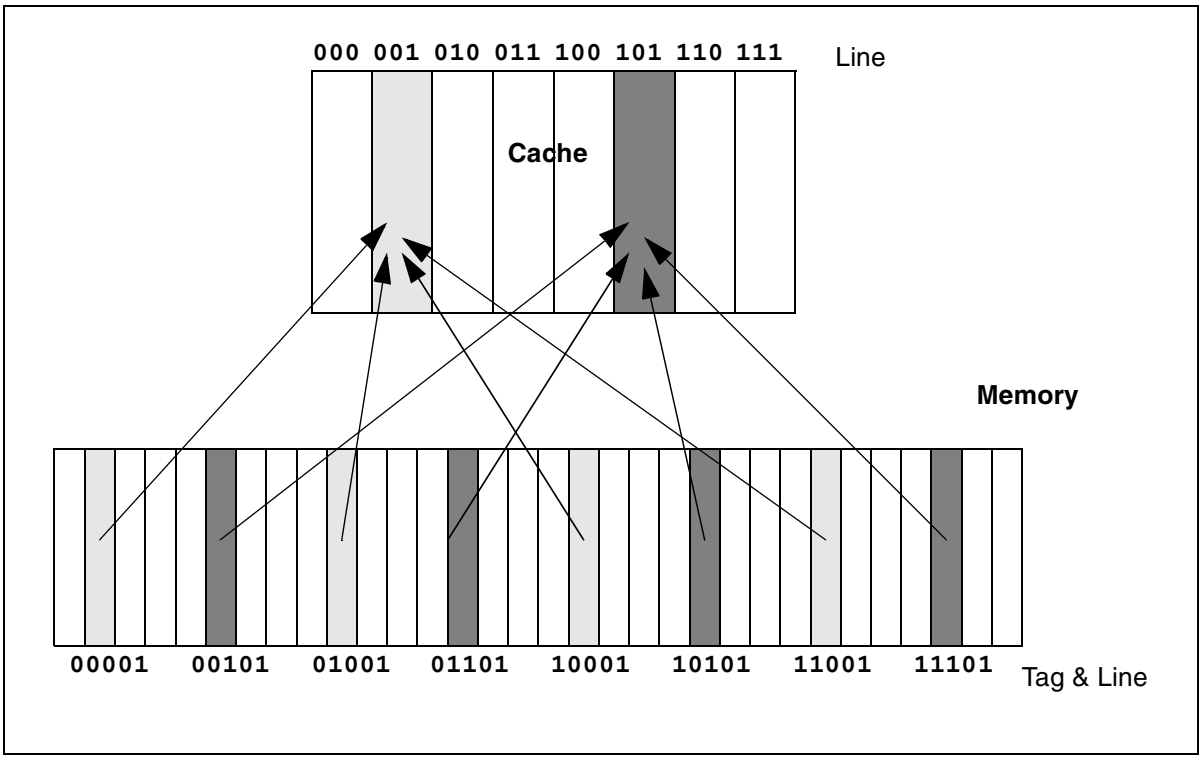
For the purposes of caching, the address space of memory is divided into lines. Each line contains four words in the instance of this cache. If any one of those four words in any given line are cached, then the other three are also cached: the line is the base "unit" of caching. To cache every line in memory would be impossible, so a hashing function is used to squeeze the total address space into 4, 8, 16 or 32 Kbytes (depending on its size configuration) of data storage within this instance of the cache. The hashing function simply ignores (for the moment) the top bits of the address. This way, any *x* Kbytes (where *x*=4,8,16 or 32) contiguous block of memory anywhere in the address space can be contained completely within the cache.

The cache also keeps a record, for each line, of what the top bits of the address were. That way, when there is another access to the cache, the cache compares those upper address bits with those of the new address and if they are the same, then we know that we have this memory location cached, and do not have to access the "real" memory.

This kind of cache is called direct-mapped (DM), because each memory location is mapped to only one location in the cache. Its miss rate is higher than other types of cache, but its simplicity allows higher clock frequencies and reduced silicon area.

The correspondence of memory locations to cache locations is shown below:

Figure 23: Direct mapping caching



17.3 Function

17.3.1 Brief overview

The cache's main purpose is performance. It is a peripheral block that outputs configuration wires to the memory cache and operates a write-back system.

A DM cache is one where any given memory location can only correspond to one line in the cache, so no mapping decisions need to be made. A cache line is 4 words (128 bits) wide. The cache is write-back rather than write-through.

For a general background on different cache types, see *Computer Organization & Design by Patterson and Hennessy, Chapter 7*.

Table 42: Cache port list

Signal name	I/O	Timing	Description	Logical grouping
CFG_SIZE[1:0]	in	no	Cache size Configuration: 00: 4 K cache 01: 8 K cache 10: 16 K cache 11: 32 K Ccahe	
CP_REQ	in	middle	Request of memory access: it is cached or not depending on the internal configuration	Cache port
CP_GNT	out	late	Access granted; it could be a default grant	
CP_VLD	out	early	Access completed; in case of read, data is valid on the return path	
CP_RNW	in	middle	Read not Write access	
CP_DEVACC	in	middle	Access is to be transferred to the memory port verbatim	
CP_BE[3:0]	in	middle	Byte enables	
CP_ADD[31:2]	in	middle	Address of the access	
CP_DATA[31:0]	in	middle	Data to be written	
CP_PASS[3:0]	in	middle	Bus ignored if the access is cacheable and passed through verbatim to the output if it is not	
CP_R_DATA[31:0]	out	early	Return data either from cache or from the memory port	
MP_REQ	out	early	Request of access to the external memory/ peripheral	Memory port
MP_GNT	in	middle	Access granted from the external arbiter	
MP_VLD	in	middle	Access completed; in case of read, data is valid on the return path	
MP_RNW	out	early	Read not Write access	
MP_BURST	out	early	Burst access request from Refill engine	
MP_DEVACC	out	early	Not cached access to peripherals	
MP_BE[3:0]	out	early	Byte enables	
MP_ADD[31:2]	out	early	Address of the access	
MP_DATA[31:0]	out	early	Data to be written to the external memory/ peripheral	
MP_PASS[3:0]	out	early	Retimed version of the input bus when there is a non-cached access and it's all set to "0" when cache access is made	
MP_R_DATA[31:0]	in	middle	Return data from the external memory/ peripheral	
PP_REQ	in	middle	Request of access to internal configuration registers	Programming port
PP_VLD	out	late	Write/Read access has been honored and back data is available	
PP_RNW	in	middle	Read not Write access	
PP_ADD[5:2]	in	middle	Address of the access to the internal register	
PP_DATA[15:0]	in	middle	Data to be written into internal registers	
PP_R_DATA[15:0]	out	late	Data read from internal registers	

Table 42: Cache port list

Signal name	I/O	Timing	Description	Logical grouping
CLOCK	*in*	N/A	*System Clock (60-120 MHz)*	Globals
RESET	in	async	Asynchronous reset for flops, not RAMs	
TST_LOGICENABLE	in	middle	Test control that allows controllability of RAM output and TAG comparator	TAP controls
TST_RUNBIST	in	middle	Begins the BIST operation on the embedded RAMs	
TST_SCANEN	in	middle	Enable scan shift registers	
TST_SCANIN[1:0]	in	middle	Scan shift registers input	Scan chains
TST_SCANOUT[1:0]	out	middle	Scan shift registers output	

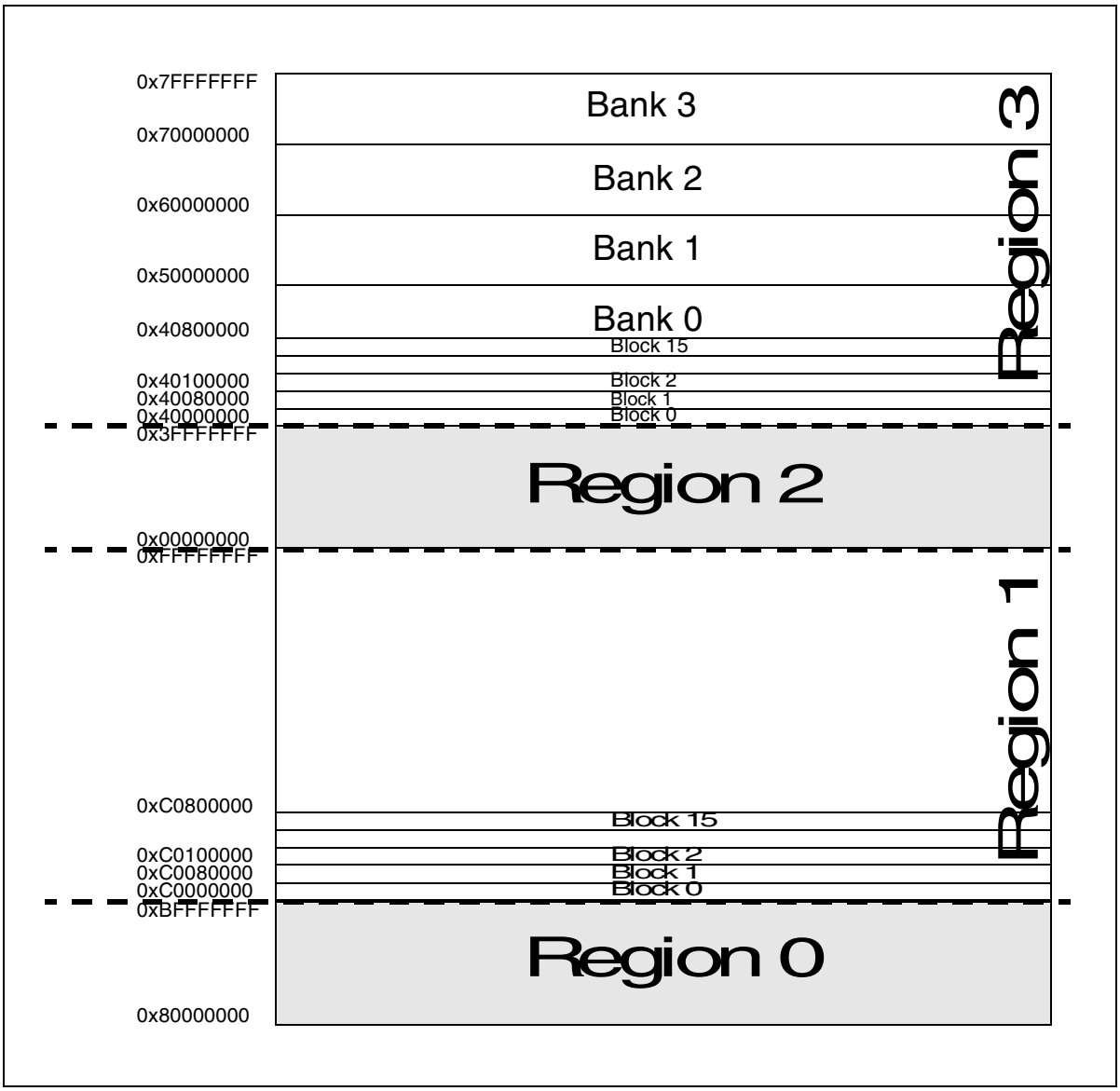
17.3.2 Interfaces

The DM cache has three main ports: the first allows the system to configure the cacheability map of the device writing in its configuration registers; the cache is seen as normal peripheral block and so it's accessed through an RG protocol (the grant is not allowed to be a default grant).

The second port is the one from the CPU memory arbiter: all accesses pass through this port and are checked for cached data or forwarded to the memory port. Valids are not given until at least the cycle after a request is granted. The protocol implemented is a full RGV one.

The third port allows for request to real memory or external devices/peripherals: either those requests determined to be non-cacheable or cache refill and write back. The burst signal indicates that the refill controller is making four consecutive accesses.

Figure 24: Cacheability memory map



17.3.3 Architecture

The ST20 cache architecture diagram below shows the major functional components required to implement the direct-mapped cache. It is not a physical microarchitecture, but a logical representation.

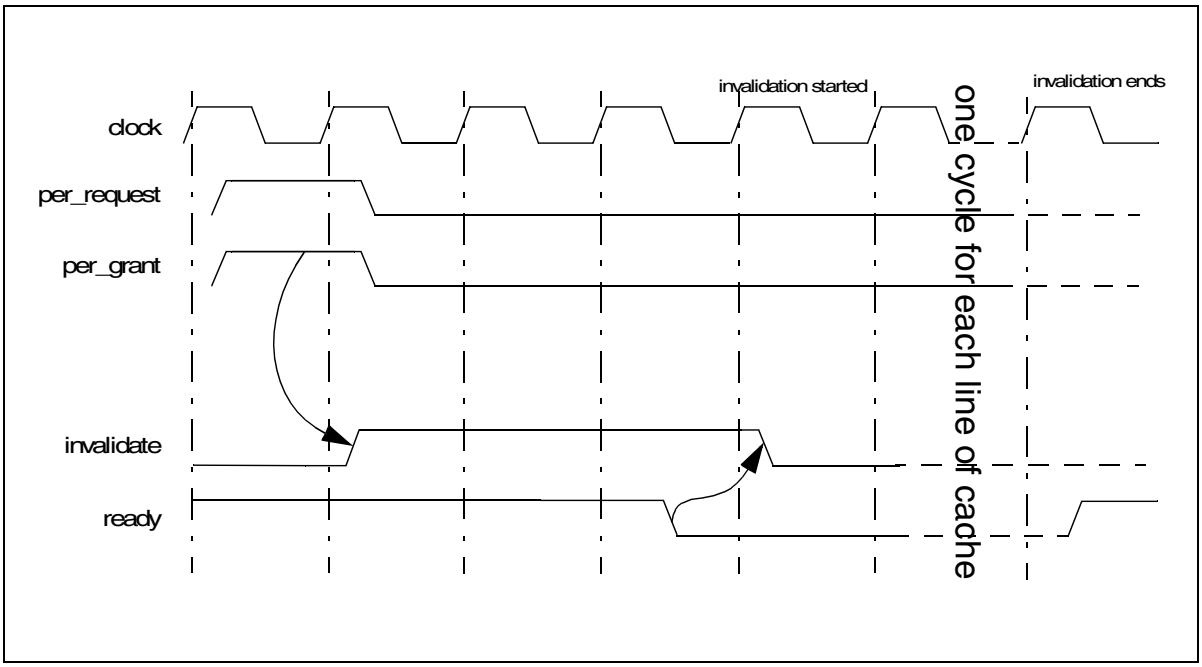
The refill controller arbitrates between its own requests to update the TAG and Data RAMs and requests coming from the input to the cache. The write buffer and retime buffer (PW Buffer) both may keep a copy of a new access, however the write buffer only updates when it has written its contents in to the Data Array. The retime buffer is there to pass on any requests that turn out to be non-cacheable and is therefore loaded on every access.

Protocols

On the memory input, there can be more than one request being dealt with in the cache at once, but valids are always kept in order. On both the input and output memory ports, valids are not allowed to occur in the same cycle as their own grant. The burst signal on the output signifies that

the current request is part of a four-word burst. These bursts must only vary the bottom two bits of the address, but those bits can vary in any order.

Figure 25: Example: Invalidate cache protocol

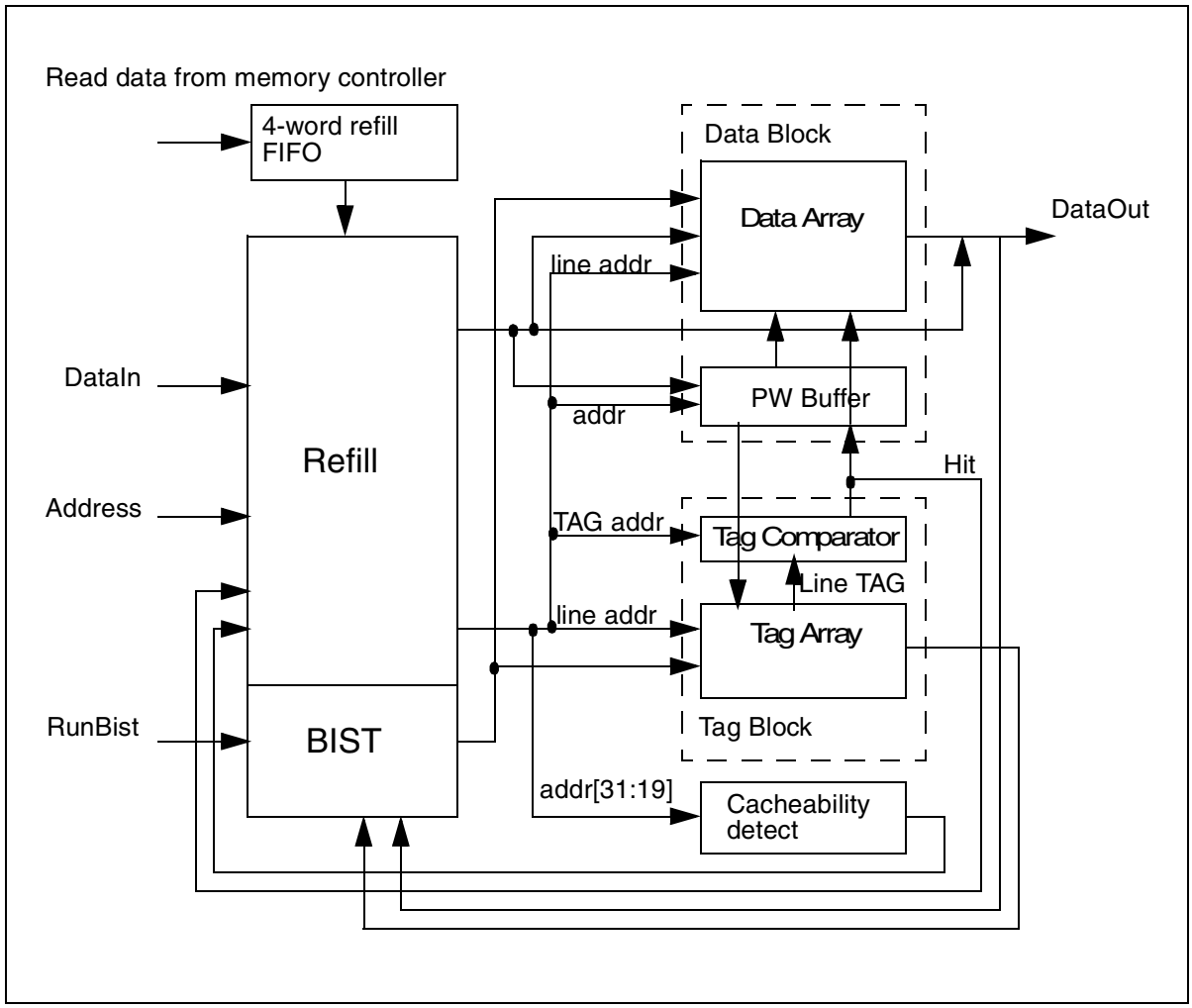


Reset initiates an invalidate process on the cache. Invalidate (and flush) can also be triggered by writing to the respective peripheral register. The protocol for the invalidate (and similarly, flush) operation is illustrated below, with the ready signal from the Status register. To cause a flush or invalidate, write to the appropriate register (FLUSH or INVALIDATE) and then poll the STATUS register until the FLUSH (or INVALIDATE, as appropriate) bit goes low. When it has gone low, this indicates that the process has actually begun and the cache will not process any memory requests until it has finished. The ready signal is also low. When ready returns to the high state, the process has finished and the cache begins processing requests again.

Overview of operation

The block has a peripheral interface used to configure its behavior and to allow invalidate and flush operations. This may be accessed through the non-critical command bus or other peripheral interface bus. The block also contains the functionality of the cache refill and cache blocks (as generated by WebGen), although the cache size may vary.

Figure 26: Architecture



This block performs address decoding on its cache memory input port to decide whether the memory access it is presented with is cacheable, then either passes the request on to its output memory port, or performs a cache access. The cacheability is configured through the peripheral port and the configuration and detection of this are kept together to allow easy modification of the system.

Any memory requests that should have a different destination than either the cache or the memory controller should be filtered out before the cache. For example, an high-speed internal RAM may have its requests filtered from the stream before they reach the cache.

If a device access is made to a location that would otherwise be a hit, the cache ignores it and therefore becomes incoherent. The compiler/assembler/programmer must make sure that their code does not introduce this problem.

The address input (CP_ADD[31:2]) has further meaning in the cache, in that it can be divided into fields that vary according to cache size:

Table 43: Address fields

Address bits (4K)	Address bits (8K)	Address bits (16K)	Address bits (32K)	Field	Description
[31:12]	[31:13]	[31:14]	[31:15]	TAG	Compared with TAG on selected line to determine hit or miss
[11:4]	[12:4]	[13:4]	[14:4]	Line address	Selects which cache line to be accessed
[3:2]	[3:2]	[3:2]	[3:2]	Word address	Selects the word within the cache line

Non-cacheable accesses

Non-cacheable accesses can be pipelined together (that is, the cache can pass another request through while it is waiting for the first non-cacheable access) but the cache cannot service a cacheable access until all outstanding non-cacheable accesses have completed because the order of valids must be maintained, and a fast cacheable hit access (see below) might violate this. The cache has an implementation-dependent limit on the number of outstanding valids it can cope with on its output, but this limit is invisible to the user, except for a stalling of the requester when it is reached. As soon as valids are received back from the interconnect and the cache moves away from its limit, normal operations should resume immediately. Non-cacheable accesses pass byteenables through unchanged, and have the burst signal low.

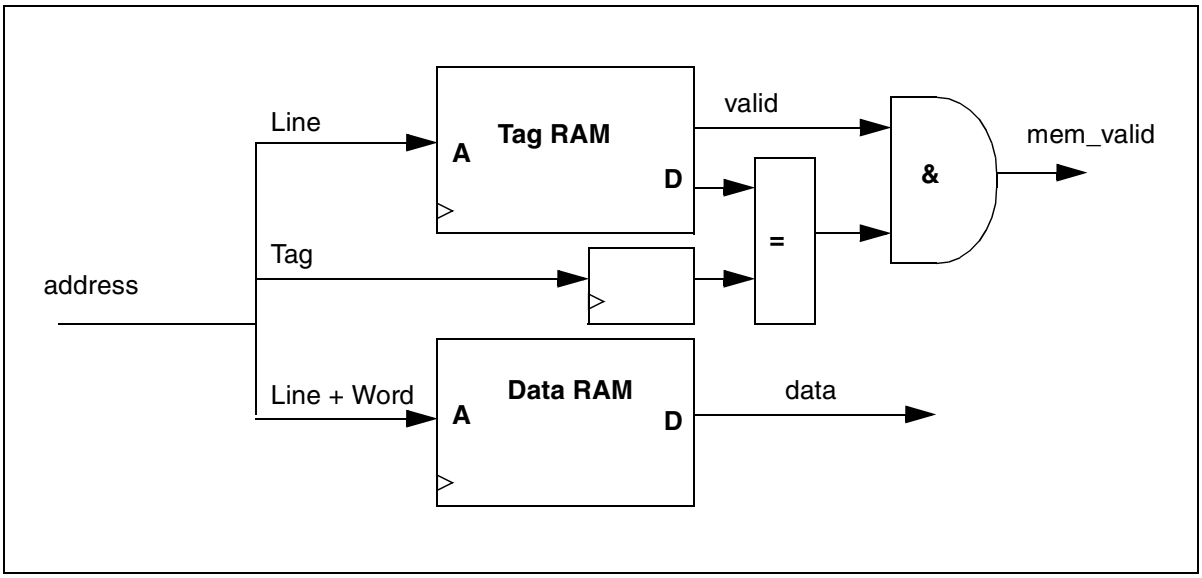
Cacheable accesses

On a cacheable request entering the cache, the address is split into three fields: tag, line and word. The line field is used to address two RAM arrays: one of which contains Data and the other Tags. By looking up the line of a new request and comparing the tag in the array with the new request's tag we can determine whether the cache can service the request without going to main memory (that is, we have a “hit”).

After reset, the cache is initialized to a state where every access will be a “miss” (not a hit). This is done by having another bit in each Tag RAM location called “valid”. If this bit is low then the tag returned by the Tag RAM is ignored, as there is no data at all in the cache for that line. An invalid line always gives a miss. The invalidate process which occurs after reset, or when triggered from the peripheral port, goes through every Tag RAM location, setting this valid bit low.

Read hit (from idle)

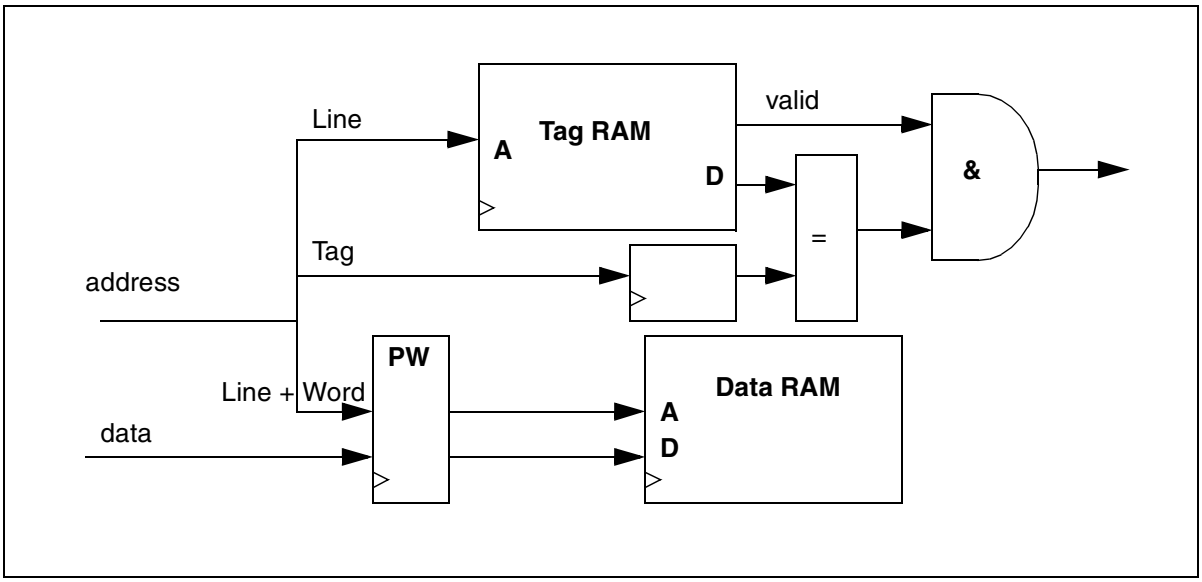
Figure 27: Read hit



If, however, we have a hit, and the hit is to a valid line, then we know that the contents of the Data RAM for this line are relevant. If we are doing a read, we can return the data for that line and word immediately. To speed up cache hits, we always address the Data RAM array with the line and word from the new request whenever we get a read access, so that we can send back the data read out to the requester on the next cycle, if it turns out to be a hit (which we also discover on the next cycle).

Write hit

Figure 28: Write hit

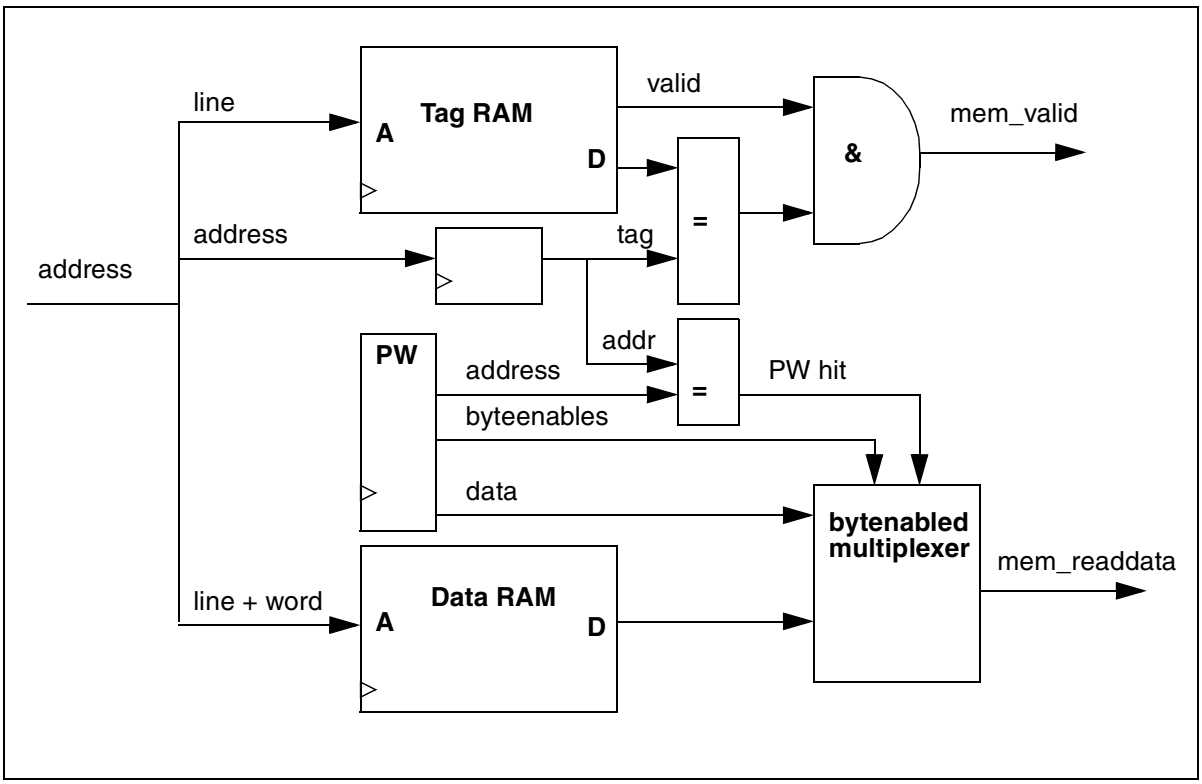


The case of a hit when writing is more complicated. We cannot put the write into the Data RAM until after we know if it is a hit or not, and we cannot afford to wait an extra cycle to find that out. The solution is to add a pipeline stage for writes, called the Pending Write buffer (PW). When we have an access, we still read the Tag RAM, but for the Data RAM we store the address and data in the PW. This data can be written in to the Data RAM the next time we have a write access (or idle), since we have just said the Data RAM is not used in the first cycle of a write access (or when the cache is idle).

Read hit with pending write

The Pending Write will wait during a read hit access, as the read hit keeps the Data RAM busy. This poses the problem of what happens if one of these reads is to the same location that the PW is to. To cope with this situation, the PW's address (Tag, Line and word) is also compared with the new request's address at the same time as the Tag array is checked. If there is a PW hit, then the PW byteenables should be used to select data from the PW when '1' and from the Data array when '0'.

Figure 29: Read hit with pending write



Miss operation

There are two reasons why the cache may not have the data requested inside and a "miss" can occur: when there is no Tag at all stored for the line requested (invalid tag) or when the Tag stored for the Line is different to the one requested.

If the cache line of the new request is invalid, then there will not be any information in that line of the Data RAM. However, if the miss occurred because of a different Tag to the existing one in the cache line, then the information in the Data RAM may have been updated since it was last read from memory. If this is the case, it should be written back to memory before the new request's line overwrites it. An extra bit is stored alongside the data in the RAM for this purpose: the "dirty" bit. There is one of these for each line and the cache sets these to '0' (clean) when the data is loaded in to the cache, and to '1' (dirty) when a cache location is written to by the requester (write hit or miss). To take advantage of burst accesses to memory, a whole line is read or written at one time during a miss.

If there is a read miss and there is a pending write to the same line, we need to make sure that this pending data is written back. This is achieved by making sure that the pending write is established in the cache before the data is written back to memory. An extra cycle is taken to establish *any* PW whenever there is a read miss. After this, the cache will then read the dirty bit for that line and if the line is dirty and needs writing back to main memory, it will trigger the Writeback.

Writeback and Refill

At this point there are two things that need to be done: the writeback (read the line from Data RAM array then write it to memory) and the refill (read the line from memory and write it to the Data RAM array). Although these at first appear sequential, they are actually two processes contending for two resources. The resources are the single memory port (to send requests) and the Data RAM array. Also, the Data RAM must be given to the writeback process first, if it is required. Both writeback and refill accesses to main memory always have the burst signal high.

This concurrence allows the two operations to happen in parallel: one can be accessing the Data RAM while the other accesses the main memory. The Refill requests memory reads and then waits for the writeback to take its fixed 4 cycles with the Data RAM. If the memory sends back data and valids before the Data RAM is available, then the cache needs to buffer these. The writeback similarly does its four word reads from the Data RAM then waits for the refill to finish with the memory port. The system cannot deadlock because only one of the two blocks will ever be waiting: either the refill is faster with the memory or the writeback is faster with the Data RAM, but these are mutually exclusive.

Once the refill has finished, if the access was a write, the write access needs to be written over the top of the refilled data. The cache takes an extra cycle to do this before it is free to service the next request. The writeback operation may still be ongoing (if grants are taking a long time to come back on the memory port) but this will only stall the cache if another miss occurs and the refill engine needs to access the memory port again: it will have to wait until the previous writeback is completed.

Other operation

The cache also has another pipeline stage at its input: the RG buffer. This is because the cache may not know if it will be busy until late on in a clock cycle, so the RG buffer keeps a copy of the last granted access in case the cache cannot service it, however, if the cache is not busy there is no cycle delay added.

The throughput of the cache can be one hit every cycle (with the valid returning the cycle after the request is granted), however a miss will take considerably longer. Accesses determined to be non-cacheable will be passed on in the next cycle, retimed. The retiming allows the fast clock cycle time to be reachable. Therefore, there will be an extra cycle of delay to any such requests before they go to the memory output port.

The general definition of cacheable bank and blocks extended to all regions with this new design will allow better re-use of the block even with CPU whit unsigned addresses (the map structure is highly symmetric).

17.4 Implementation

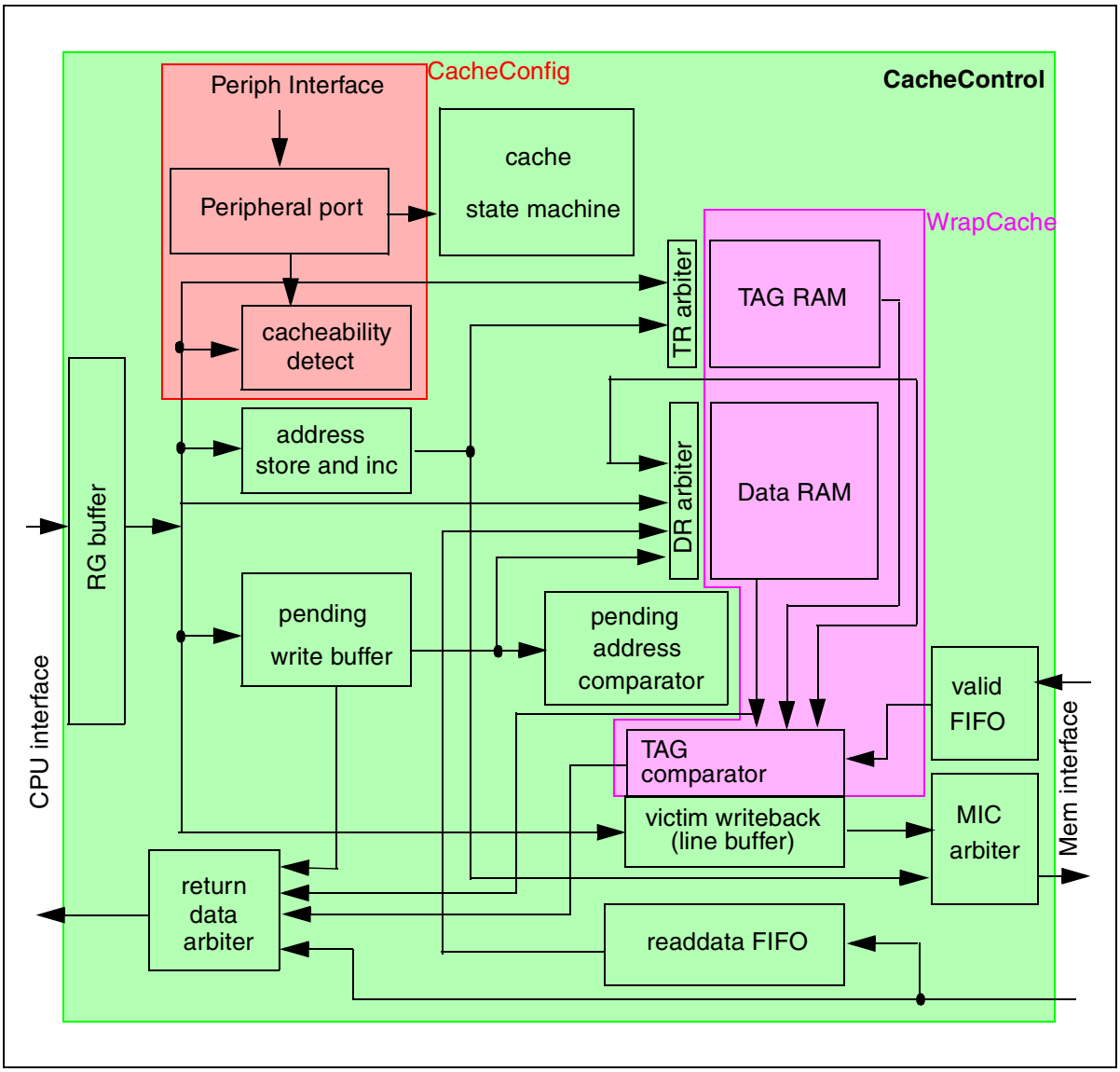
17.4.1 Microarchitecture

The new design has been developed in order to integrate the Unicad SPS6 RAMs, a unique BIST engine and to include the circuitry previously located inside Docker's RAM.

A few other considerations have been taken into account during the codification of the new behavior, to work with some ideosyncrasies of the old RAMs, like the last readdata maintained on the output until next read. Even write accesses to the same address do not change the output; this feature is only used on the Tag RAM. The Unicad RAMs do not have this property, so this is implemented within the glue logic.

The schematic below shows the design hierarchy:

Figure 30: Microarchitecture: decomposition



The separation of the “CacheConfig” block is quite clean: the configuration registers and the decoding of the address can be changed quite easily without changing the interface of the block. The interface would only change if the function of the flush and invalidate commands was to differ or be extended in some way, or if more address bits were to be decoded (that is, smaller chunks than 512 kbytes were required).

The CacheConfig block simply receives an address from bits 31 down to 16 from the control block, and returns a single signal, “cacheable” that is active high. The only other interface is the flush/invalidate/ready.

Most blocks communicate with either the control state machine or the victim writeback which contains a secondary (slave) state machine. These state machines are described below.

Refill state machine

The are two main state machines in the cache that can be respectively roughly described as “refill” and “writeback”. The refill state machine is the master, and it will trigger the writeback into operation, when appropriate. Both of these state machines have both Moore (simple, state-derived) and Mealy (complex, state and input derived) outputs. The refill state diagram is shown below:

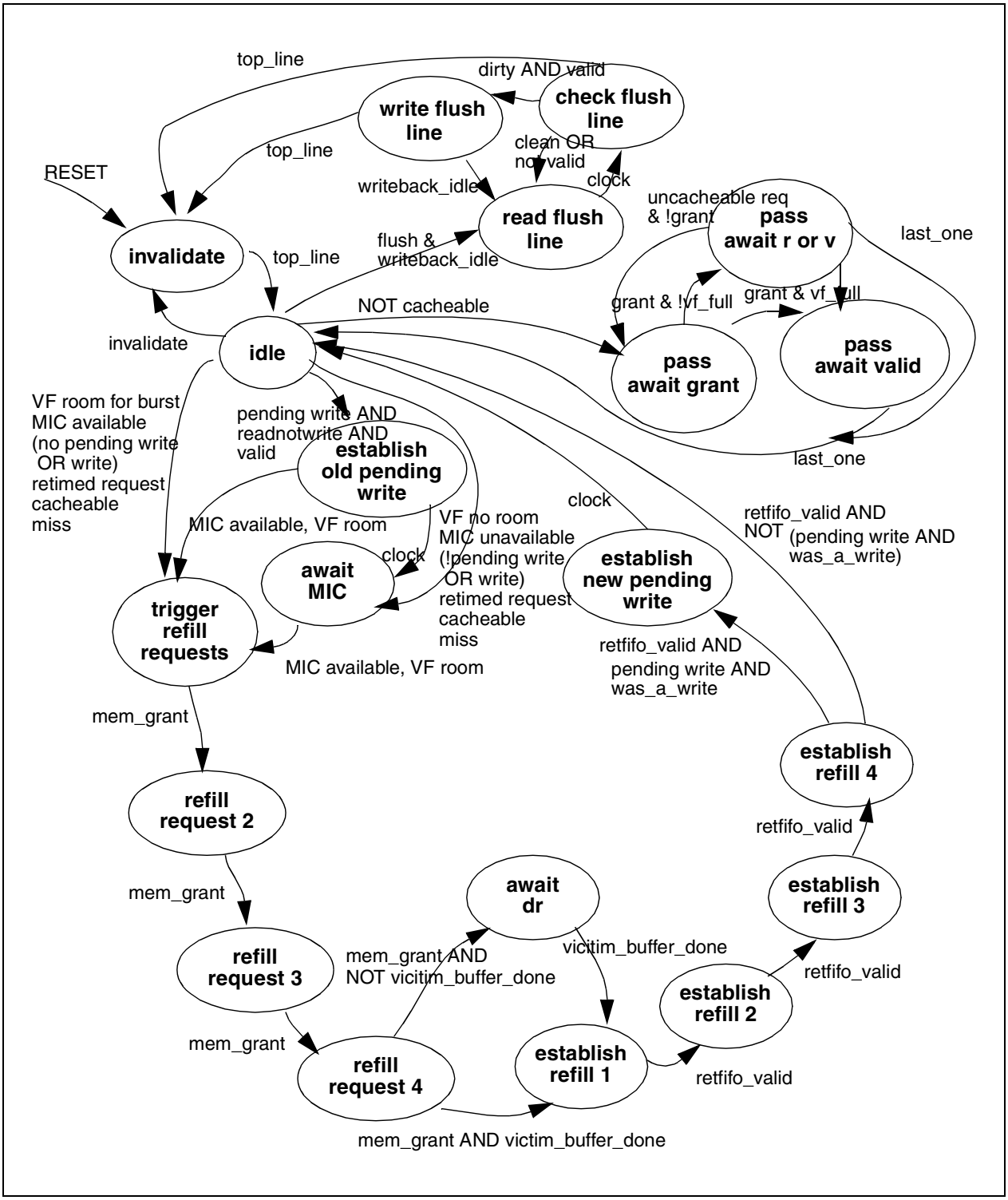


Figure 31: Main refill state machine

This state machine has the following inputs:

Table 44: Refill state machine inputs

FSM input	Source	Description
TR_HITVALID	Tag RAM comparator	The hit signal (gated with valid)
WAS_A_REQUEST	Address store	Retimed version of address
WAS_A_READNOTWRITE	Address store	Retimed version of readnotwrite
WRITEBACK_IDLE	Victim writeback FSM	Indicates when the writeback FSM is ready to be triggered
VF_ROOM_FOR_A_BURST	Valid FIFO	Whether the valid FIFO has enough storage to cope with a burst of 4 requests to memory
CACHEABLE	Cacheability detect	Whether the request is to be cached (retimed in cacheability detect block)
PENDING_WRITE_VALID	Pending write buffer	Is there something in the pending write buffer?
MEM_RETFIFO_VALID	Return data FIFO	If there is some return data form the MIC waiting in the return data FIFO
VICTIM_BUFFER_DONE	Victim writeback FSM	Indicates when the writeback FSM has finished with the Data RAM
NO_OUTSTANDING_VALIDS	Valid FIFO	Active if there are no valids to be waited for on the MIC interface
INVALIDATE	Peripheral port	User is requesting an invalidate operation
FLUSH	Peripheral port	User is requesting a flush operation
LAST_ONE	Valid FIFO	Active if there we are getting the last valid expected from the MIC this cycle
TOP_LINE	Address store	Goes high when the line part of the address is all ‘1’s
DIRTY	Data RAM	The dirty line bit from the Data RAM (retimed in DR)
REFILL_MEM_GRANT	MIC arbiter	Active when the MIC arbiter is passing on a refill request to the MIC
THRO_GRANT	MIC arbiter	Active when the MIC arbiter is passing on a pass-through request to the MIC
TR_READ_VALID	Tag RAM	Whether the stored tag was valid (retimed in TR)
MEM_NOTWB_VALID	Valid FIFO	Valid signal from MIC but, only active when the corresponding request was not from the Writeback (that is, Refill or pass-through and thus dealt with by *this* state machine)
WAS_AN_IDLE	Refill FSM	Internal: Keeps a record of whether we were in a state where we could accept a new request last cycle
MIC_AVAIL_TO_REFILL	MIC arbiter	Active if the MIC arbiter can service refill requests, or if we have to wait
VF_FULL	Valid FIFO	Active when the Valid FIFO cannot cope with more requests therefore the FSM has to wait

There are a number of Mealy outputs from this state machine:

Table 45: Refill FSM Mealy outputs

Mealy output	Destination	Description
SEND_REFILL_REQS (aka SET_AWAIT_MISS_WORD)	MIC arbiter, Refill FSM	Tells the MIC arbiter that we will want to send refill requests next cycle (also used internally to set “awaiting_miss_word” high)
CACHE_BUSY (aka REFILL_IN_PROGRESS)	DR Arbiter	High when not in idle state or when in idle but moving to a different state next cycle
GRANT_TO_BUFFER	RG Buffer	The late grant to the CPU/DCU that is retimed by the RG Buffer
LOAD_ADDRESS	Address Store	Tells the address store to load the current address
LOAD_PENDING	Pending write	Tells the PW to load the current request for writing
KILL_PENDING (KILL_PW)	Pending write	Tells the PW to throw away the current pending write because it was non-cacheable
HIT_VALID	Return data arbiter	A validated version of the hit line that can be passed back to the CPU/DCU as mem_valid if necessary
LOAD_VICTIM_BUFFER	Writeback FSM	The trigger to the writeback FSM, telling to check if the line is dirty, and if so perform a writeback operation
WORD_INCREMENT	Address store	This means that we are going on to the next word in wither our requests to the MIC or our requests to the DR. “11” should wrap to “00”.
REFILL_FIFO_REMOVE (aka UPDATE_WITH_REFILL)	Return data FIFO, DR arbiter	Takes the current top of the FIFO and puts it into the Data RAM
THRO_REQUEST	MIC arbiter	Sends a request for a pass-through access
STORE_VICTIM_TAG	Writeback FSM	Tells the Writeback FSM to store the output of the TR as it is the Tag of the victim
ALLOW_DR_ACCESS	DR arbiter	Stops writes to data RAM when low

There are also Moore outputs, derived solely from the state:

Table 46: Refill FSM Moore outputs

Moore output	Destination	Description
ESTABLISH_PENDING_WRITE	DR arbiter	Tells the arbiter to write the PW into the DR
USE_STORED_ACCESS	DR arbiter	Tells the arbiter to give priority to the address data from the address store
LINE_INCREMENT	Address store	Increments the line in the address store: used by flush and invalidate to go through each line
AWAITING_REFILL_DATA	Readdata FIFO	Tells the FIFO to load the next non-writeback readdata when the valid comes form the MIC
AWAITING_THROUGH	Return data arbiter	Sets the arbiter to pass the valid and readdata form MIC straight through to the requester
REFILL_BURST	MIC arbiter	Lets the arbiter know that this is the last access of a burst when it is low during request
REFILL_MEM_REQUEST	MIC arbiter	Requests the refill from MIC
SET_VALIDBIT	TR arbiter	Tells the arbiter to update the valid bit
UPDATE_TAG	TR arbiter	Informs the arbiter that there is a new Tag to be written to the TR
READY	Peripheral port	See the description of invalidate and flush in the overview of operation specification section

Table 46: Refill FSM Moore outputs

Moore output	Destination	Description
FLUSH_REQUEST	TR arbiter, Victim writeback	Used during flush operation, to read the Tag
LP_SM_NEEDS_CLK	Low power	Used in top-level calculation of cache clock control

Finally, there is some further state in the block, closely associated with the state machine itself:

Table 47: Auxiliary state outputs

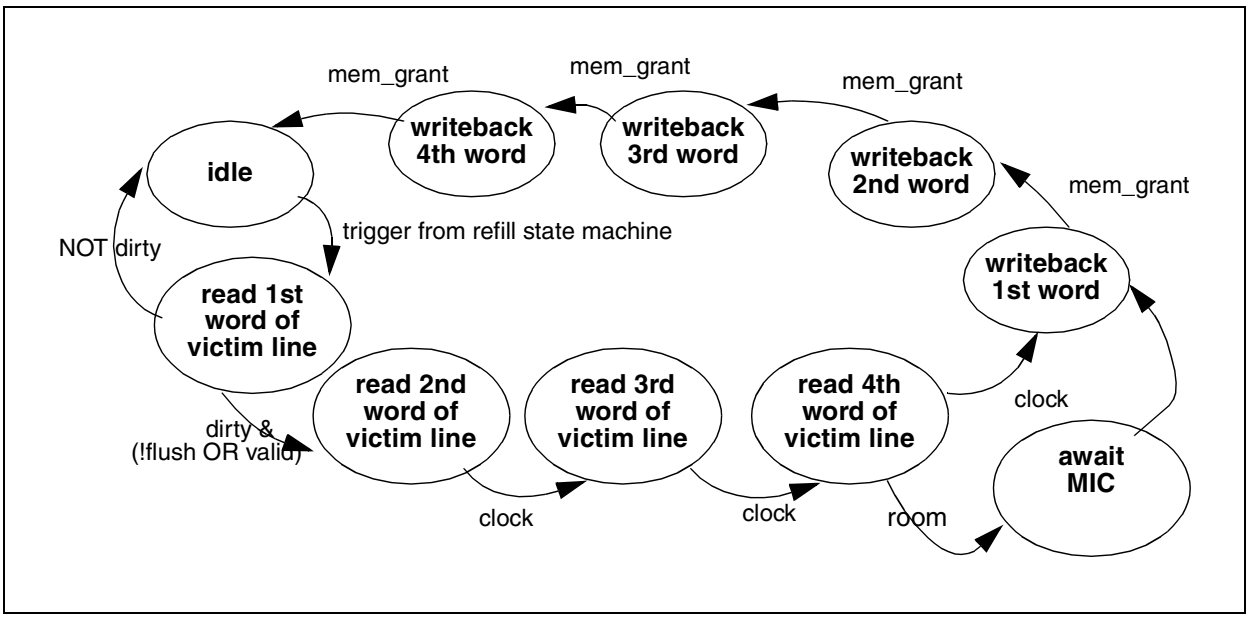
State output	Destination	Description
AWAITING_MISS_WORD	Return data arbiter	Set high by the send_refill_reqs Mealy output of the FSM, and reset to low by the first mem_notwb_valid received from the MIC: this tells the return data arbiter to pass the MIC valid straight through to the requester.

This state machine therefore controls most of the operation of the cache, arbitrating with the Writback FSM for the Data RAM and MIC interface.

Writeback state machine

The Victim Writeback state machine is show in the following diagram:

Figure 32: Victim write-back state machine



The state machine has these inputs:

Table 48: Writeback state machine inputs

FSM input	Source	Description
LOAD_VICTIM_BUFFER	Refill FSM	Triggers the loading of the victim buffer, and subsequent writeback operation if the victim was dirty
DR_DIRTY_LINE	Data RAM	If the line is dirty, we need to read all of it and write it back. If it is not, the FSM returns to idle
WRITEBACK_GRANT	MIC arbiter	The writeback's request to MIC has been granted: the valids are not looked at by the state machine
ROOM_FOR_A_BURST	Valid FIFO	If the valid FIFO can cope with another four (a burst) accesses then this signal is high
TR_READ_VALID	Tag RAM	Used during flush to determine whether to writeback or not
FLUSH_REQUEST	Refill FSM	Used during flush to tell this FSM if it should examine the valid during the read 1st word state

The writeback state machine has the following Mealy outputs:

Table 49: Writeback FSM Mealy outputs

Mealy output	Destination	Description
VICTIM_REQUEST	DR arbiter	Request to read the victim line from the DR
SEND_WRITEBACK_REQS	MIC arbiter	Means that there will be a writeback request in the next cycle
NEXT_DONE	Refill FSM	Signals when the Victim FSM is finished with the DR and the refill can put its new line in: see victim_buffer_done in "further state" below

And it has these Moore outputs:

Table 50: Writeback FSM Moore outputs

Moore output	Destination	Description
WRITEBACK_IDLE	Refill FSM	Tells the refill FSM when this FSM is idle
WRITEBACK_REQUEST	MIC arbiter	Requests the writback to go to the MIC
WRITEBACK_BURST	MIC arbiter	Lets the arbiter know that this is the last access of a burst when it is low during request
WRITEBACK_WORD	MIC arbiter, DR arbiter	The word within the line that is being requested to/from the MIC/DR

There is also further state in this block too, which is:

Table 51: Further state in block with Writeback FSM

State output	Destination	Description
VICTIM_BUFFER_DONE (aka DONE_REG)	Refill FSM	Tells the refill when the Writeback FSM has finished with the Data RAM
WRITEBACK_TAG (aka VICTIM_TAG_REG)	MIC arbiter	Writeback copy of the victim Tag
WRITEBACK_LINE (aka LINE_REG)	MIC arbiter	Writeback copy of the requested line (same as victim line)

This block only operates when triggered by the Refill state machine when there is a cache miss or a flush operation is performed and the writeback is required.

17.4.2 Critical paths

The most critical path, is **CLOCK** to **CP_VLD**. This is the generation of the valid to the CPU slice after having checked if the access hit the cache or not. This path includes the read data from the TAG memory, the multiplexer for test enhancement, the tag comparator and the gating logic if access was cacheable or not; it is strongly affected by the setup time on the CPU slice. The main constraints are the access time of the Ram (3.31 nS) and the output delay required by the core C2 (3.64 nS).

18 Cache registers

Register addresses are provided as *CacheBaseAddress* + offset, where *CacheBaseAddress* is either *DCacheBaseAddress* or *ICacheBaseAddress*.

The *DCacheBaseAddress* is:

0x3000 1000,

The *ICacheBaseAddress* is:

0x3000 2000,

The registers operate the cacheability based on a generic configurable memory map as shown in Figure 24: *Cacheability memory map* *on page 135*.

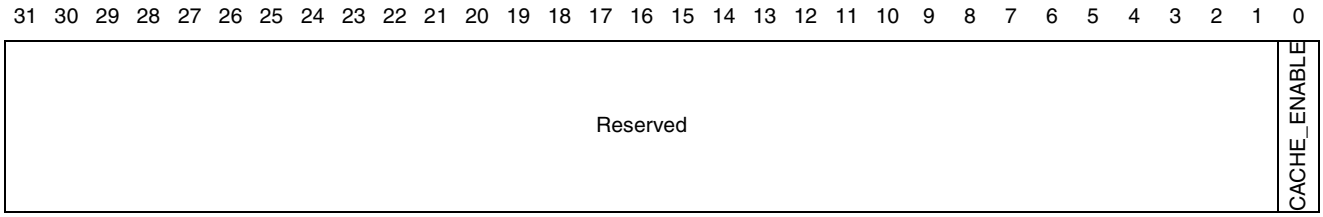
The registers may be programmed at start-up, or configured dynamically; if they are to be changed, then use the following procedure:

1. Wait until all outstanding read and write requests to the cache are complete; it is not necessary to wait for its output to be free or that invalidate or flush operations are complete, unless these are holding up an outstanding request on its input.
2. Ensure that no more requests will be made on the memory port input of the cache while:
3. Make the changes required, using peripheral port requests
4. Wait for one clock cycle before resuming requests to the cache input memory port

Table 52: Register summary table

Register	Description	Offset	Type	Bits
CACHE_ENABLE	Global cacheing enable	0x00	RW	0:0
Reserved	-	0x04-0x0C	-	-
INVALIDATE	Start invalidate	0x10	W	0:0
FLUSH	Start flush	0x14	W	0:0
STATUS	Monitor status	0x18	R	2:0
Reserved	-	ox1C	-	-
REGION_0_ENABLE	Configure region 0 cacheability	0x20	RW	0:0
Reserved	-	0x24	-	-
REGION_1_BLOCK_ENABLE	Configure region 1 cacheability	0x28	RW	15:0
REGION 1_TOP_ENABLE		0x2C	RW	0:0
REGION_2_ENABLE	Configure region 2 cacheability	0x30	RW	
Reserved		0x34	-	-
REGION_3_BLOCK_ENABLE	Configure region 3 cacheability	0x38	RW	15:0
REGION_3_BANK_ENABLE		0x3C	RW	3:0
Reserved	-	0x3F	-	-

CACHE_ENABLE Global cacheing enable

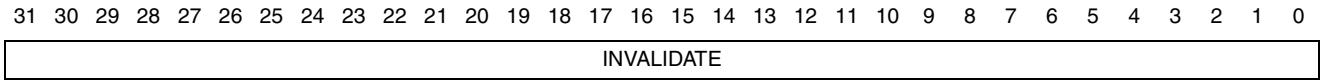


Address: *CacheBaseAddress* + 0x00
Type: R/W
Reset: 0
Description:

[31:16] **Reserved**

[0] **CACHE_ENABLE**
1: Enabled regions are cacheable according to the other configuration registers
0: No access is cacheable

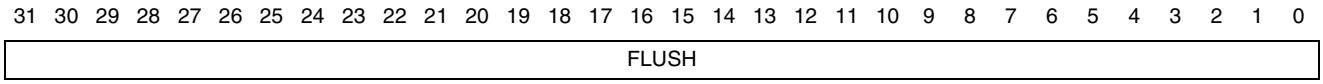
INVALIDATE Start invalidate



Address: *CacheBaseAddress* + 0x10
Type: WO
Reset: 0
Description:

[31:0] **INVALIDATE**
When accessed, begins the invalidate process (internal invalidate signal goes high)

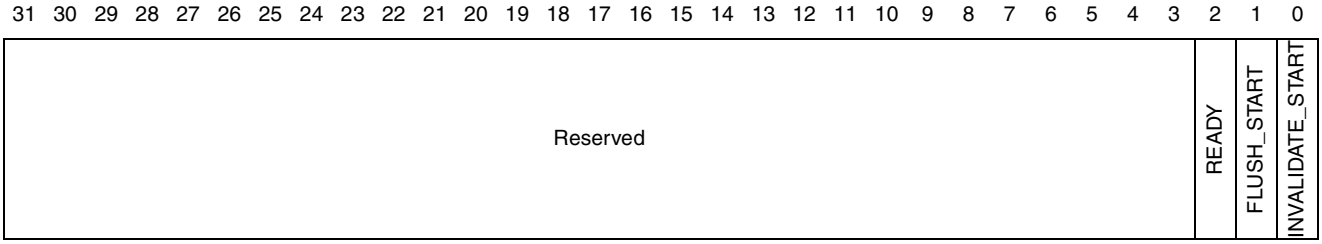
FLUSH Start flush



Address: *CacheBaseAddress* + 0x14
Type: WO
Reset: 0
Description:

[31:0] **FLUSH**
When accessed, begins the flush process (internal flush signal goes high)

STATUS Monitor status



Address: *CacheBaseAddress* + 0x18
Type: RO
Reset: 0
Description:

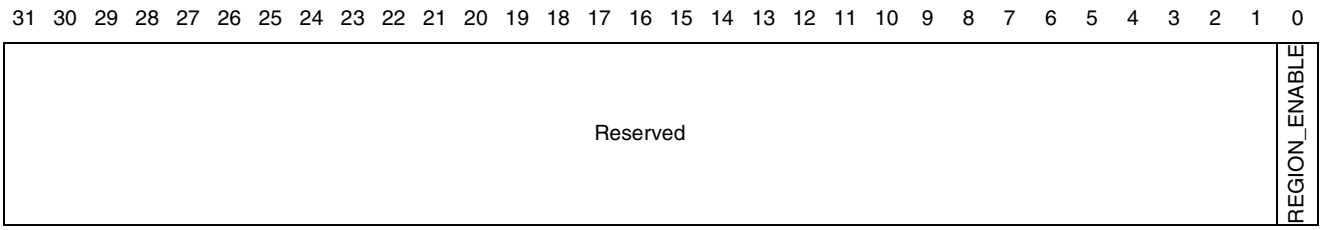
[31:3] **Reserved**

[2] **READY:** Cache status
1: ready; any flush or invalidate process has finished

[1] **FLUSH_START**: Flush process status
Reset when READY = '0', the flush process is complete when READY returns to '1'

[0] **INVALIDATE_START**: Invalidate process status
Reset when READY = '0', the flush process is complete when READY returns to '1'.
An invalidate is also performed on reset.

REGION_0_ENABLE Configure region 0 cacheability



Address: *CacheBaseAddress* + 0x20
Type: R/W
Reset: 0
Description:

[31:1] **Reserved**

[0] **REGION_ENABLE**
1: All of Region 0 (0x80000000 - 0xBFFFFFFF) is cacheable
0: this area is not cached

REGION_1_BLOCK ENABLE Configure region 1 cacheability (block)

31	30	29	28	27	26	25	24	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																BLOCK15	BLOCK14	BLOCK13	BLOCK12	BLOCK11	BLOCK10	BLOCK9	BLOCK8	BLOCK7	BLOCK6	BLOCK5	BLOCK4	BLOCK3	BLOCK2	BLOCK1	BLOCK0

Address: *CacheBaseAddress* + 0x28
Type: R/W
Reset: 0
Description:

[31:16] **Reserved**

[15] **BLOCK15**
1: Block 15 (0xC0780000 - 0xC07FFFFF) uses the cache
0: Not cacheable

[14] **BLOCK14**
1: Block 14 (0xC0700000 - 0xC077FFFF) uses the cache
0: Not cacheable

[13] **BLOCK13**
1: Block 13 (0xC0680000 - 0xC06FFFFF) uses the cache
0: Not cacheable

[12] **BLOCK12**
1: Block 12 (0xC0600000 - 0xC067FFFF) uses the cache
0: Not cacheable

[11] **BLOCK11**
1: Block 11 (0xC0580000 - 0xC05FFFFF) uses the cache
0: Not cacheable

[10] **BLOCK10**
1: Block 10 (0xC0500000 - 0xC057FFFF) uses the cache
0: Not cacheable

[9] **BLOCK9**
1: Block 9 (0xC0480000 - 0xC04FFFFF) uses the cache
0: Not cacheable

[8] **BLOCK8**
1: Block 8 (0xC0400000 - 0xC047FFFF) uses the cache
0: Not cacheable

[7] **BLOCK7**
1: Block 7 (0xC0380000 - 0xC03FFFFF) uses the cache
0: Not cacheable

[6] **BLOCK6**
1: Block 6 (0xC0300000 - 0xC037FFFF) uses the cache
0: Not cacheable

[5] **BLOCK5**
1: Block 5 (0xC0280000 - 0xC02FFFFF) uses the cache
0: Not cacheable

[4] **BLOCK4**
1: Block 4 (0xC0200000 - 0xC027FFFF) uses the cache
0: Not cacheable

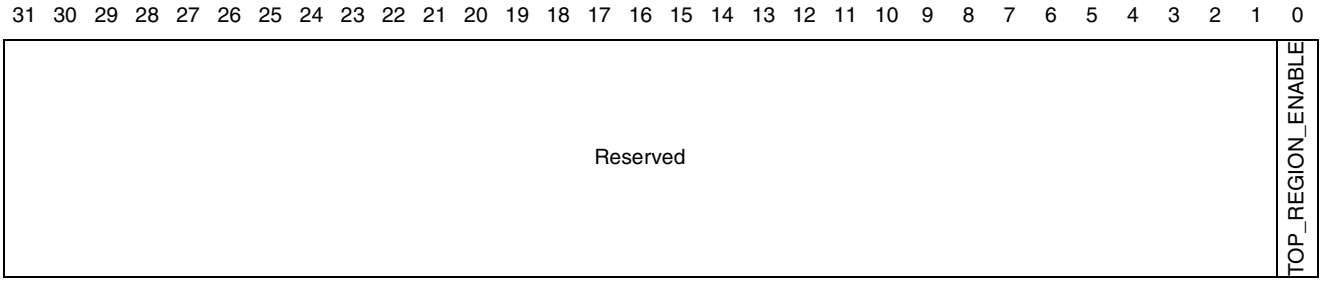
[3] **BLOCK3**
1: Block 3 (0xC0180000 - 0xC01FFFFF) uses the cache
0: Not cacheable

[2] **BLOCK2**
1: Block 2 (0xC0100000 - 0xC017FFFF) uses the cache
0: Not cacheable

[1] **BLOCK1**
1: Block 1 (0xC0080000 - 0xC00FFFFF) uses the cache
0: Not cacheable

[0] **BLOCK0**
1: Block 0 (0xC0000000 - 0xC007FFFF) uses the cache
0: Not cacheable

REGION_1_TOP_ENABLE Configure region 1 cacheability (top)

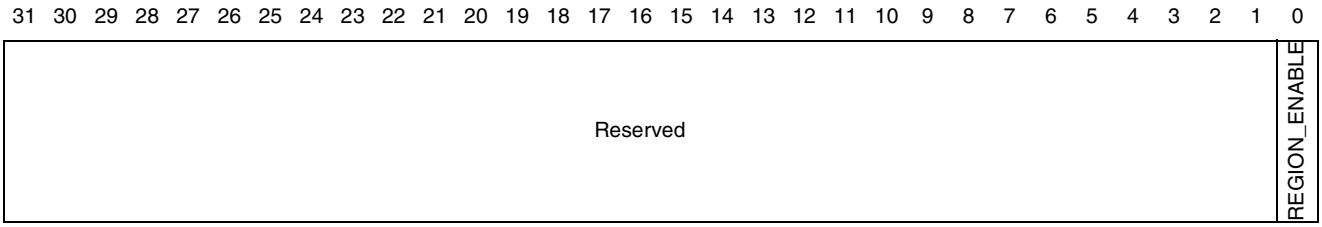


Address: *CacheBaseAddress* + 0x2C
Type: R/W
Reset: 0
Description:

[31:16] **Reserved**

[0] **TOP_REGION_ENABLE**
1: all of top Region 1(0xC0800000 - 0xFFFFFFFF) is cacheable
0: this area is not cached

REGION_2_ENABLE Configure region 2 cacheability



Address: *CacheBaseAddress* + 0x30
Type: R/W
Reset: 0
Description:

[31:1] **Reserved**

[0] **REGION_ENABLE**
1: all of Region 2 (0x00000000 - 0x3FFFFFFF) is cacheable
0: this area is not cached

REGION_3_BLOCK_ENABLE Configure region 3 cacheability (block)

31	30	29	28	27	26	25	24	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																BLOCK15	BLOCK14	BLOCK13	BLOCK12	BLOCK11	BLOCK10	BLOCK9	BLOCK8	BLOCK7	BLOCK6	BLOCK5	BLOCK4	BLOCK3	BLOCK2	BLOCK1	BLOCK0

Address: *CacheBaseAddress* + 0x38
Type: R/W
Reset: 0
Description:

[31:16] **Reserved**

[15] **BLOCK15**
1: Block 15 (0x40780000 - 0x407FFFFF) uses the cache
0: Not cacheable

[14] **BLOCK14**
1: Block 14 (0x40700000 - 0x4077FFFF) uses the cache
0: Not cacheable

[13] **BLOCK13**
1: Block 13 (0x40680000 - 0x406FFFFF) uses the cache
0: Not cacheable

[12] **BLOCK12**
1: Block 12 (0x40600000 - 0x4067FFFF) uses the cache
0: Not cacheable

[11] **BLOCK11**
1: Block 11 (0x40580000 - 0x405FFFFF) uses the cache
0: Not cacheable

[10] **BLOCK10**
1: Block 10 (0x40500000 - 0x4057FFFF) uses the cache
0: Not cacheable

[9] **BLOCK9**
1: Block 9 (0x40480000 - 0x404FFFFF) uses the cache
0: Not cacheable

[8] **BLOCK8**
1: Block 8 (0x40400000 - 0x4047FFFF) uses the cache
0: Not cacheable

[7] **BLOCK7**
1: Block 7 (0x40380000 - 0x403FFFFF) uses the cache
0: Not cacheable

[6] **BLOCK6**
1: Block 6 (0x40300000 - 0x4037FFFF) uses the cache
0: Not cacheable

[5] **BLOCK5**
1: Block 5 (0x40280000 - 0x402FFFFF) uses the cache
0: Not cacheable

[4] **BLOCK4**
1: Block 4 (0x40200000 - 0x4027FFFF) uses the cache
0: Not cacheable

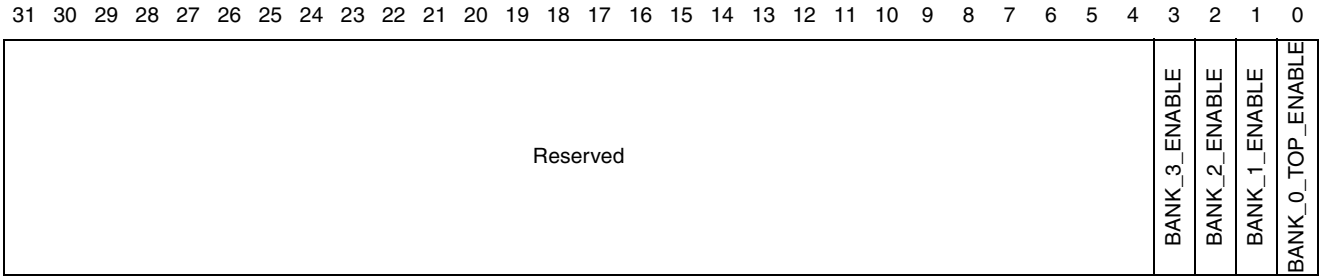
[3] **BLOCK3**
1: Block 3 (0x40180000 - 0x401FFFFF) uses the cache
0: Not cacheable

[2] **BLOCK2**
1: Block 2 (0x40100000 - 0x4017FFFF) uses the cache
0: Not cacheable

[1] **BLOCK1**
1: Block 1 (0x40080000 - 0x400FFFFF) uses the cache
0: Not cacheable

[0] **BLOCK0**
1: Block 0 (0x40000000 - 0x4007FFFF) uses the cache
0: Not cacheable

REGION_3_BANK_ENABLE Configure region 3 cacheability (Bank)



Address: *CacheBaseAddress* + 0x3C
Type: R/W
Reset: 0
Description:

[31:4] **Reserved**

[3] **BANK_3_ENABLE**
1: all of bank 3 (0x70000000 - 0x7FFFFFFF) is cacheable
0: this area is not cached

[2] **BANK_2_ENABLE**
1: all of bank 2 (0x60000000 - 0x6FFFFFFF) is cacheable
0: this area is not cached

[1] **BANK_1_ENABLE**
1: all of bank 1 (0x50000000 - 0x5FFFFFFF) is cacheable
0: this area is not cached

[0] **BANK_0_TOP_ENABLE**
1: top of bank 0 (0x40800000 - 0x4FFFFFFF) is cacheable
0: this area is not cached

19 External memory interface (EMI)

19.1 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.

19.2 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 STi7710 does not support SDRAM on its EMI interface.

19.3 EMI interface pins

The external pins are described in Table 53.

Table 53: 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.

19.4 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 58: *EMI configuration register summary on page 170*.

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_CONFIGDATA0) 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.

19.5 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.

19.5.1 Bank programming

Refer to Section 19.8: *Chip select allocation/bank configuration on page 166* for full bank programming details.

19.5.2 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_xxxCLKSEL associated with that memory type.

The external clocks and associated clock dividers are then enabled by a write of 1 to the register EMI_CLKENABLE. Once enabled, any attempt to reprogram the clock ratios may have undefined effects.

19.6 Default/reset configuration

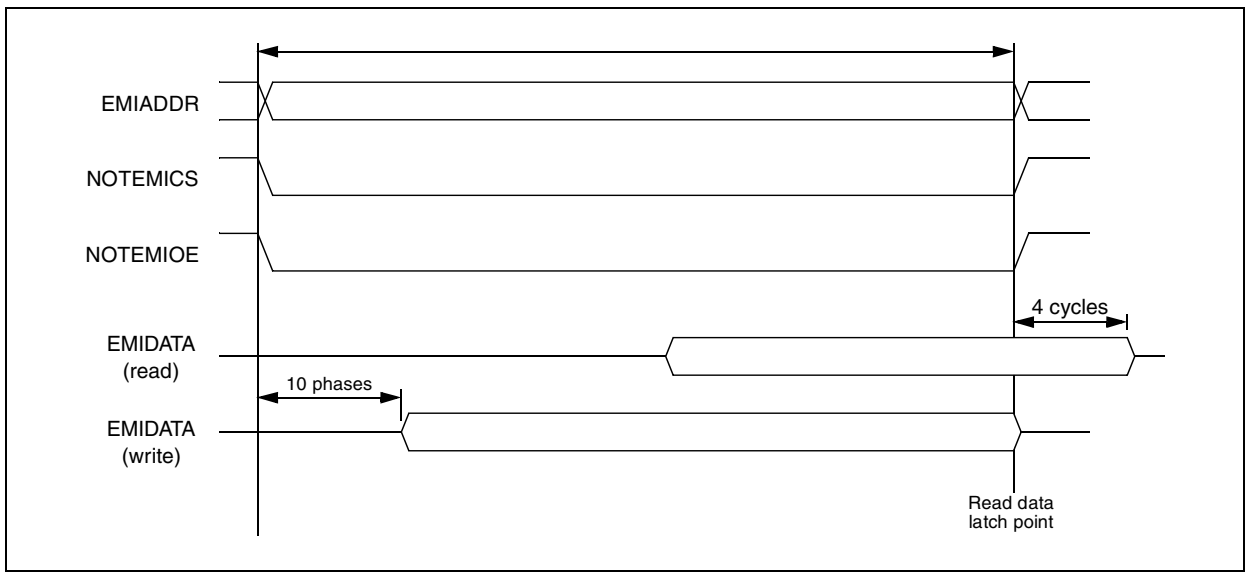
Following reset, a default configuration setting is loaded into all five banks. This allows the EMI to access data from a slow ROM or FLASH memory. The default settings are detailed in Table 54.

Table 54: 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 33: Default asynchronous configuration



19.7 Peripheral interface (with synchronous flash memory support)

19.7.1 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 19.8 on page 166 *for specific STi7710 settings.*

Figure 34 shows a generic access cycle and the allowable values for each timing field.

Figure 34: Generic access cycle

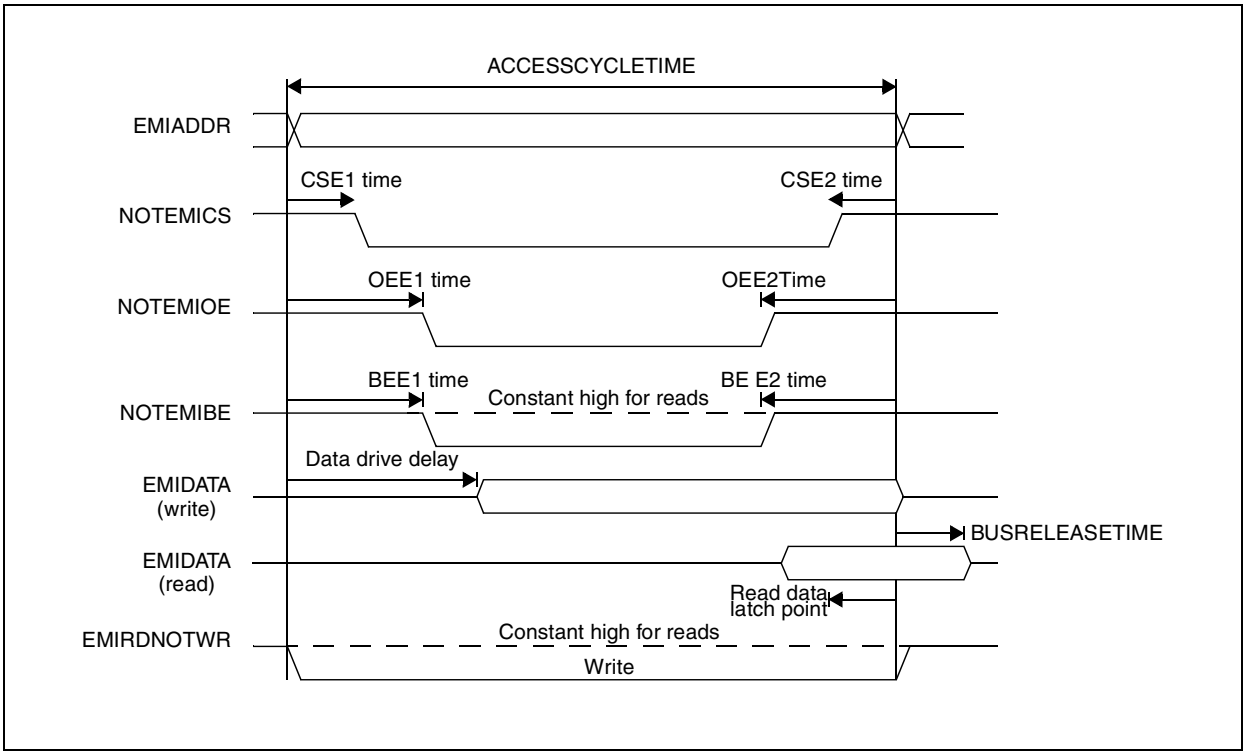


Table 55: Strobe timing parameters for peripheral

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 56: 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

19.7.2 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 57 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 57: 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.

19.7.3 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_FLASHCLKSEL.

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.

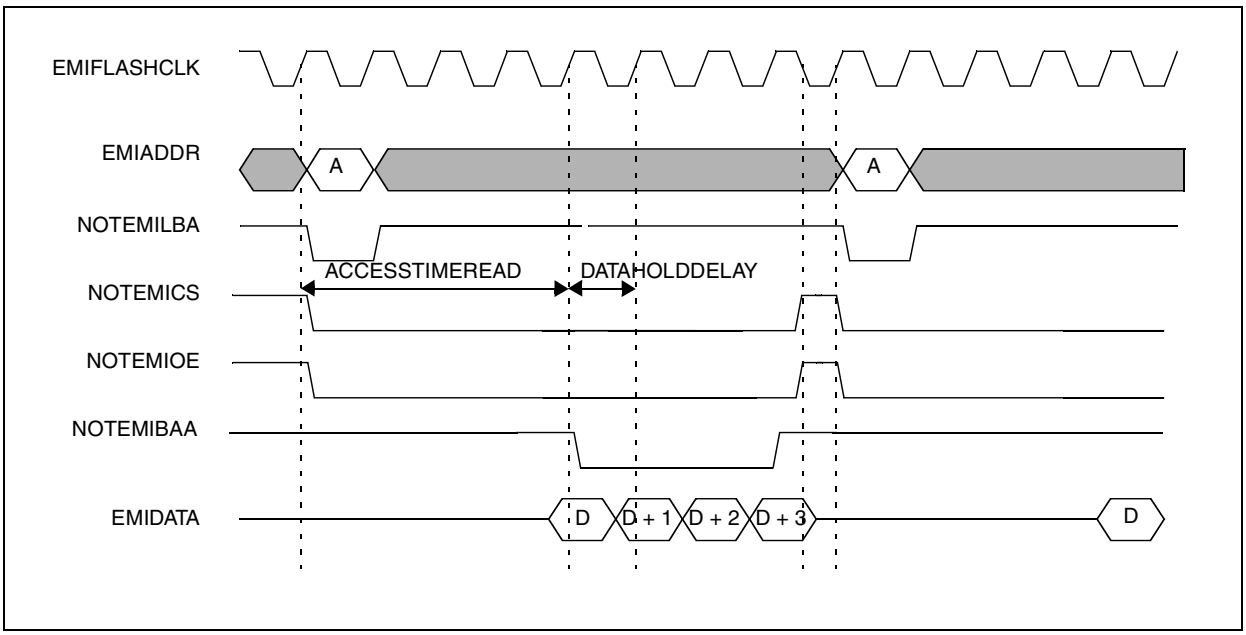
4 Interleaved burst mode is not supported by the EMI due to the implementation of multiple reads only using synchronous burst mode (feature provided by all three families of Flash chips adopted).

The EMI supports a asynchronous single write.

The asynchronous single read/write uses the same protocol as that of the normal peripheral interface.

Figure 35 shows a typical burst access with burst length of four words.

Figure 35: 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.

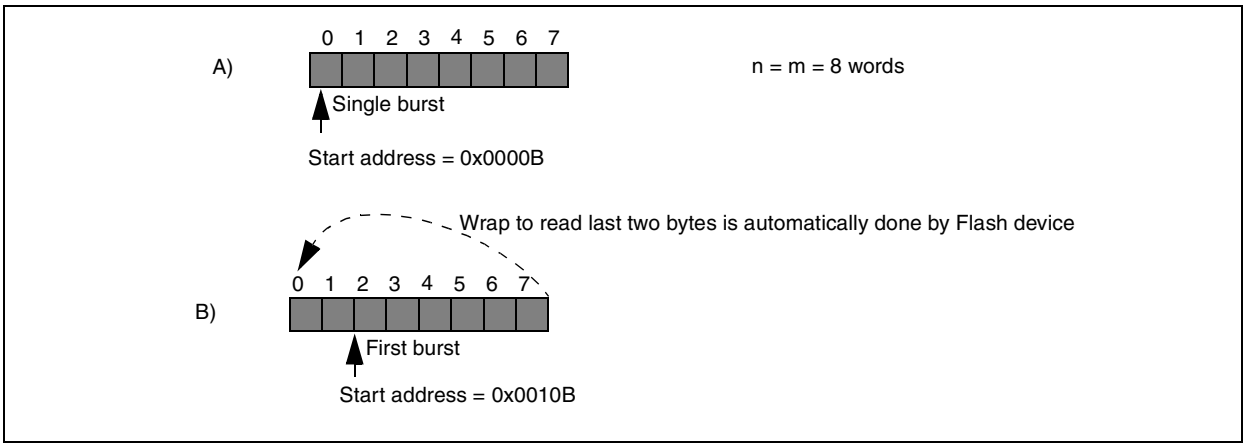
19.7.4 Burst interrupt and burst reiteration

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 36 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 36: 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.

19.7.5 Synchronous burst enable

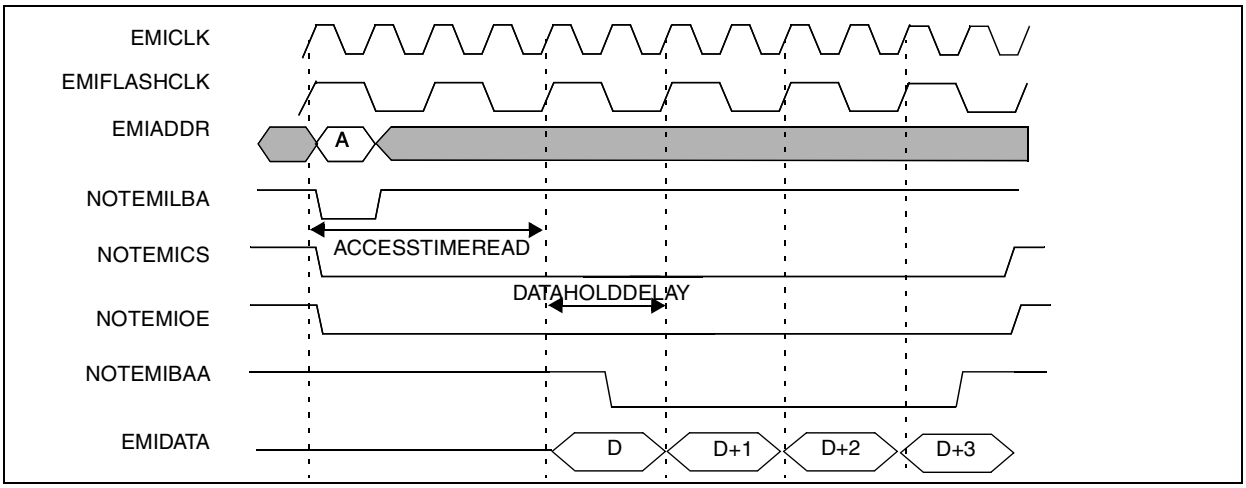
This operation is controlled by software and must only be performed when all other configuration registers in the EMI have been programmed.

Table 57 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.

19.7.6 Support for lower clock rates

Many SFlash devices operate in the 30 to 50 MHz clock range (Table 57) 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 37: Half speed EMI SFlash clock



19.7.7 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.

19.7.8 Use of Flash memories in different banks with contiguous memory spaces

As shown in Table 57 on page 162, 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.

19.8 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 STi7710 provides five chip selects at its outputs, one per bank.

19.9 Address bus extension to low order bits

The STi7710 EMI is able to use either 8- or 16-bit wide memory devices. Selection is done through field PORTSIZE of registers EMI_CONFIGDATA0, see Section 20.2.1: *Configuration register formats for peripherals on page 175*. 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.

19.10 PC Card interface

The STi7710 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 STi7710 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 STi7710 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 will be 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.

19.11 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 STi7710 and an external host.

Arbitration is always performed by the STi7710, 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 STi7710 input EMIREQ.
2. The STi7710 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 STi7710 *n* turn de-asserts signal EMIGNT and takes back ownership of the EMI bus.

Whilst pin EMIREQ is a pure input to the STi7710, 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* STi7710 *EMI access (application dependent).*

During reset, the STi7710 does not drive the EMI bus, so it is also possible to hold off the STi7710 and boot an external host from the EMI bus by holding off the hardware reset supplied to the STi7710.

19.12 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.

20 EMI registers

20.1 Overview

Addresses are provided as

EMIConfigBaseAddress + offset,
EMIBufferBaseAddress + offset, or
EMIBankBaseAddress + offset.

The *EMIConfigBaseAddress* is:

0x2010 2000.

The *EMIBufferBaseAddress* is:

0x2010 2800.

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).

Table 58: EMI configuration register summary

Register	Description	Offset from EMIConfigBaseAddress	Type
EMI_VCR	Reserved for version control EMI register	0x000	RO
EMI_STATUSCFG	Status register (configuration flags update)	0x010	RO
EMI_STATUSLOCK	Lock register (configuration flags lock)	0x018	WO
EMI_LOCK	Lock register	0x020	WO
EMI_GENCFG	General purpose configuration register set	0x028	R/W
Reserved	-	0x030 to 0x048	-
EMI_FLASHCLKSEL	Select clock speed for Flash devices	0x050	WO
Reserved	-	0x058	-
EMI_CLKENABLE	Enable clock generation for different devices	0x068	WO
Reserved	-	0x070 to 0x0F8	-
EMI_BANK0	Bank0 configuration register set (EMI_CONFIGDATA[0:3] + reserved space)	0x100 to 0x138	R/W
EMI_BANK1	Bank1 configuration register set (EMI_CONFIGDATA[0:3] + reserved space)	0x140 to 0x178	R/W
EMI_BANK2	Bank2 configuration register set (EMI_CONFIGDATA[0:3] + reserved space)	0x180 to 0x1B8	R/W
EMI_BANK3	Bank3 configuration register set (EMI_CONFIGDATA[0:3] + reserved space)	0x1C0 to 0x1F8	R/W
EMI_BANK4	Bank4 configuration register set (EMI_CONFIGDATA[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 59.

Table 59: EMI_BANKn register organization

Register	Offset from EMIBankBaseAddress
EMI_CONFIGDATA0	0x00
EMI_CONFIGDATA1	0x08
EMI_CONFIGDATA2	0x10
EMI_CONFIGDATA3	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 60: EMI buffer register summary

Register	Description	Offset from EMIBufferBaseAddress	Type
EMIB_BANK0_TOP_ADD	External memory bank 0 address bits [27:22]	0x0000	R/W
EMIB_BANK1_TOP_ADD	External memory bank 1 address bits [27:22]	0x0010	R/W
EMIB_BANK2_TOP_ADD	External memory bank 2 address bits [27:22]	0x0020	R/W
EMIB_BANK3_TOP_ADD	External memory bank 3 address bits [27:22]	0x0030	R/W
EMIB_BANK4_TOP_ADD	External memory bank 4 address bits [27:22]	0x0040	R/W
EMIB_BANK5_TOP_ADD	“Virtual” memory bank 5 address bits [27:22]	0x0050	R/W
EMIB_BANK_EN	Total number of enabled banks	0x0060	R/W

20.2 EMI register descriptions

EMI_STATUSCFG 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_STATUSLOCK 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_LOCKED			

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_LOCK 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_CONFIGDATA[0:3] may only be read. Subsequent writes to these registers are ignored.

EMI_GENCFG EMI general purpose

31	30	29	28	27	26	25	24	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_PCCB4	EN_PCCB3	MW_RETIME	Reserved	

Address: *EMIConfigBaseAddress* + 0x0028

Type: R/W

Reset: 0x00

Description: Used to propagate general purpose outputs.

[31:5] **Reserved**

[4] **EN_PCCB4**: Enable PC card bank 4
When set, during access to Bank4 some regularEMI signals (NOTEMIOE, NOTEMILBA, NOTEMIBAA, NOTEMIBE[0]) are replaced by PC Card specific signals (PCMCIA_OE#, PCMCIA_WE#, PCMCIA_IORD#, PCMCIA_IOWR#)

[3] **EN_PCCB3**: Enable PC card bank 3
When set, during access to Bank3 some regularEMI signals (NOTEMIOE, NOTEMILBA, NOTEMIBAA, NOTEMIBE[0]) will be replaced by PC Card specific signals (PCMCIA_OE#, PCMCIA_WE#, PCMCIA_IORD#, PCMCIA_IOWR#)

[2] **EWAIT_RETIME**: determines the number of retime stages (1 if set, 2 if cleared) for signal EMITREADYORWAIT. 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_SDRAMNOPGEN EMI nop generate

7	6	5	4	3	2	1	0
Reserved							NOPGEN

Address: *EMIConfigBaseAddress* + 0x0030

Type: WO

Reset: Undefined

Description: Reserved

EMI_FLASHCLKSEL 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_CLKENABLE EMI clock enable

7	6	5	4	3	2	1	0
Reserved							CLOCK_ENABLE

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.

20.2.1 Configuration register formats for peripherals

The following is a summary of the configuration register formats for peripherals.

EMI_CONFIGDATA0 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.

00000: End of access cycle	00001: 1 cycle
00010: 2 cycles	00011: 3 cycles
00100: 4 cycles	00101: 5 cycles
00110: 6 cycles	00111: 7 cycles
01000: 8 cycles	01001: 9 cycles
01010: 10 cycles	01011: 11 cycles
01100: 12 cycles	01101: 13 cycles
01110: 14 cycles	01111: 15 cycles
10000: 16 cycles	Other: Reserved

[19:15] **DATADRIVEDELAY**: 0 to 31 phases

[14:11] **BUSRELEASETIME**: 0 to15 cycles

[10:9] **CSACTIVE**: See Table 56 on page 161.

[8:7] **OEACTIVE**: See Table 56 on page 161.

[6:5] **BEACTIVE**: See Table 56 on page 161.

[4:3] **PORTSIZE**
00: Reserved 01: Reserved
10: 16-bit 11: 8-bit

[2:0] **DEVICETYPE**
001: Normal peripheral or 100: Burst Flash

EMI_CONFIGDATA1 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) timewrite for CS, BE, OE are expressed in system clock phases
1: they 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_CONFIGDATA2 EMI configuration data 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCLE not PHWR	ACCESSTIMEWRITE							CSE1TIMEWRITE				CSE2TIMEWRITE			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEE1TIMEWRITE				OEE2TIMEWRITE				BEE1TIMEWRITE				BEE2TIMEWRITE			

Address: *EMIBankBaseAddress* + 0x10

Type: R/W

Reset: 0x00

Description:

[31] **CYCLENOTPHWR**: Change measure unit for e1/e2 time accesses from phases to cycles:
0: the e1(e2) timewrite for CS, BE, OE are expressed in system clock phases
1: they are expressed in cycles.

[30:24] **ACCESSTIMEWRITE**: 2 to 127 cycles: value 0 and 1 are reserved

[23:20] **CSE1TIMEWRITE**: Falling edge of CS. 0 to 15 phases/cycles after start of access cycle

[19:16] **CSE2TIMEWRITE**: Rising edge of CS. 0 to 15 phases/cycles before end of access cycle

[15:12] **OEE1TIMEWRITE** (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] **OEE2TIMEWRITE** (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] **BEE1TIMEWRITE**: Rising edge of BE. 0 to 15 phases/cycles after start of access cycle

[3:0] **BEE2TIMEWRITE**: Falling edge of BE. 0 to 15 phases/cycles before end of access cycle

EMI_CONFIGDATA3 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_CONFIGDATA0, EMI_CONFIGDATA1, EMI_CONFIGDATA2 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_CONFIGDATA3 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

20.3 EMI buffer register descriptions

All the registers described in this section are nonvolatile

EMIB_BANK0_TOP_ADD 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_ADD 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_ADD 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_ADD 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_ADD 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_ADD 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_ENABLED							

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_ENABLED**:

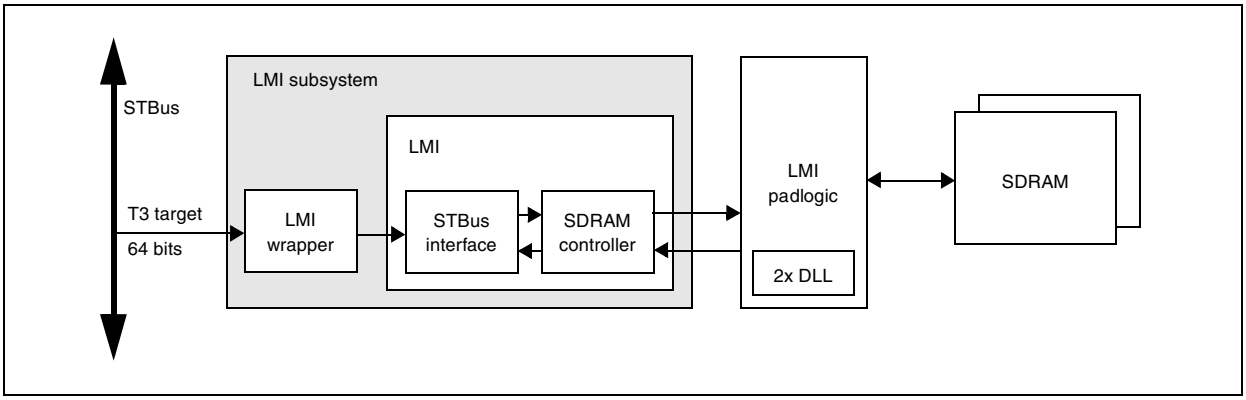
001: Bank 0 only enabled
010: Banks 0 to 1 enabled
011: Banks 0 to 2 enabled
100: Banks 0 to 3 enabled
101: All banks with associated ChipSelect enabled (0 to 4)
110: All banks enabled - including “shadow” bank 5 which has no associted ChipSelect.

21 Local memory interface (LMI)

21.1 Introduction

The local memory interface (LMI) provides the interface between the STBus and the external main-memory subsystem through the LMI glue/padlogic. The LMI core comprises an STBus port and an SDRAM controller. The key features of the main-memory array, SDRAM controller and STBus port are presented here.

Figure 38: LMI subsystem



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).

STBus port

- Two 4-deep-in-order queue for requests and responses, respectively: supports pipelining of up to 8 outstanding transactions on the STBus.
- One control block (16-Mbyte space) containing the LMI module's VCR and SDRAM control registers,
- 127 data blocks (16 Mbytes each) with access routed to the external memory.

21.2 SDRAM interface

21.2.1 Main memory configuration

The main memory is organized in rows. The data bus width can be programmed to 16 or 32 bits using bit BW (LMI_MIM). 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 registers 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.

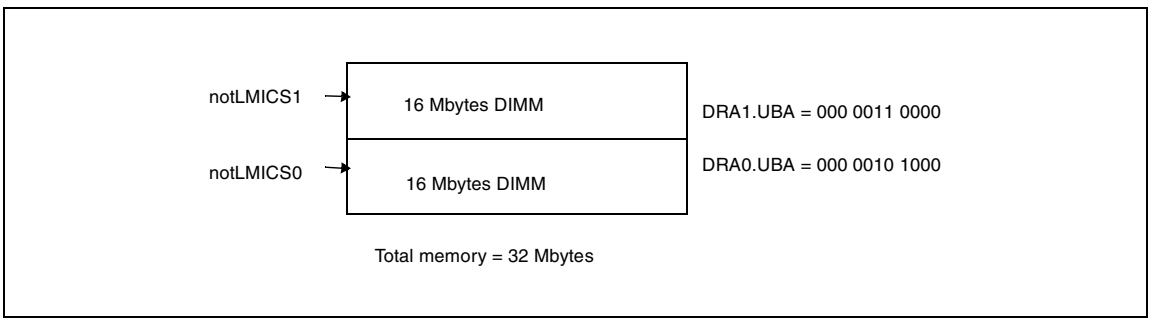
The upper boundary address of each row is defined in LMI_COC.UBA. The request address [31:21] is compared to UBA[31:21] to determine which notLMICS (chip selection) signal is to be asserted. A notLMICS signal is applied to all SDRAM devices on the same row.

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 = LMI_SDRA1.UBA must be programmed. 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.

Figure 39 depicts a 32-bit wide, 32 Mbyte main memory subsystem. It is assumed LMI_VCR.BOT_MB = 0x04.

Figure 39: Main memory configuration example



Memory configuration can be little endian or big endian. The LMI is independent of endianness when the external bus width is 32 bits. When the external bus width is 32-bit, memory interface bit LMI_MIM.ENDIAN (read only) indicates the endianness of the system.

21.2.2 SDRAM interface pins

The external pins are described in Table 61.

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 21.2.4 for details.

Table 61: SDRAM interface pins

Name	I/O	Size (pins)	Description
LMICLK	Output	1	SDRAM clock output: 66, 100, 133, 166 or 200 MHz.
notLMICLK	Output	1	LMICLK and notLMICLK are differential clock outputs to DDR SDRAM.
LMICLKEN	Output	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.
notLMICS[1:0]	Output	2	Chip select. Performs the function of selecting the particular SDRAM components during the active state.
notLMIWE	Output	1	Write enable signal. notLMIWE is asserted during writes to SDRAM.
LMIADD[12:0]	Output	13	Row and column address.
LMIBKSEL[1:0]	Output	2	Bank address.
LMIDATA[31:0]	I/O	32	Memory data.
LMIDATASTROBE[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.
LMIDATAMASK[3:0]	Output	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.
notLMIRAS	Output	1	Row address strobe.
notLMICAS	Output	1	Column address strobe.
LMIVREF	Input	1	Input reference voltage

21.2.3 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 62, Table 63 and Table 64) 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 62 and Table 63 can be bypassed enabling the bank remapping feature (see fields ENABLE_BA[1:0] of registers LMI_COC).

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, although the STi7710 LMI does not issue either read-with auto-precharge or write-with auto-precharge commands.

16-bit data bus interface

Table 62: 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]

32-bit data bus interface

Table 63: 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]

Note: AP pin: The LMI uses bit LMI_MIM.BY32AP to determine if bit LMIADD8 is used to indicate the PRE and PALL commands.

21.2.4 Initializing SDRAM devices

The driver software must initialize an SDRAM device after power-on reset. The operating system bootup code or driver software to initialize the SDRAM should not read or write the SDRAM before completing initialization.

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 needs to 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 (row [*n*]).

Example: issuing MRS command to row 0

Software does a dummy write to LMI_SDMR0, the physical address must be arranged in the following way:

- A[31:20] = 0000 1111 1000r,
- A[16:3] contains the value to be written to the SDRAM's mode register,
- DATA[63:0] is ignored since LMI_SDMR0 is a write-only virtual register,
- A[12:3] is copied to LMIADD[9:0], A[18:15] to LMIADD[13:10] and A[14:13] to LMIBKSEL[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 000. Normal SDRAM operation can then start.

Note: Software must program register LMI_MIM before writing to LMI_SDMR[0:1].

21.2.5 Operations

The SDRAM controller supports most DDR SDRAM commands. The following truth table lists all commands supported.

Table 64: 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: 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.

21.2.6 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 Trfc clock (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.

21.2.7 Power management

The LMI provides one power management mechanism.

When the LMI receives STBY_REQ from the power-down management unit (PMU), the LMI prepares the SDRAM rows to enter 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 LMI asserts STBY_ACK to the PMU. The memory clock (LMICLK) can now be stopped. The clock controlling the padlogic flip-flops can be switched off when STBY_REQ (provided by the PMU) and STBY_ACK (provided by the LMI core) are both asserted (these have to be combined to act as the enable signal of a gated clock cell outside the LMI core).

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. PMU de-asserts STBY_REQ. 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 count down 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.

21.2.8 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.

21.2.9 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.

21.2.10 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.

22 LMI registers

Register addresses are provided as *LMIBaseAddress* + offset or *MicroSystemGlueConfigBaseAddress* + offset

The *LMIBaseAddress* is:

0xCC00 0000.

The *MicroSystemGlueConfigBaseAddress* is:

0x3820 1000.

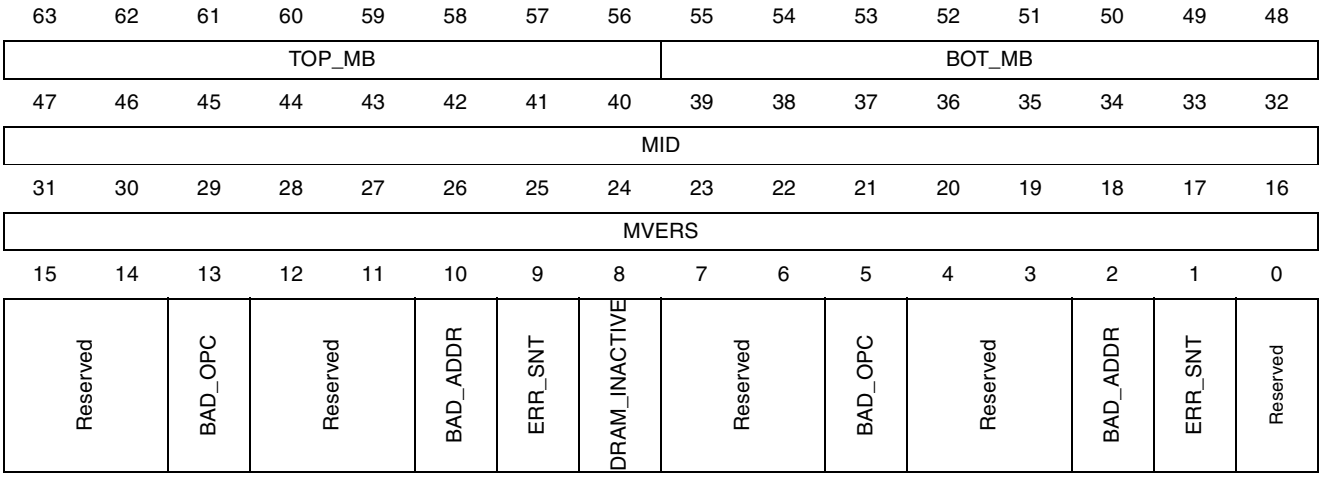
Table 65: LMI register summary

Register	Description	Offset	Type
LMI_VCR	Version control	0x0000	R/W
LMI_MIM	Memory interface mode	0x0008	R/W
LMI_SCR	SDRAM control	0x0010	R/W
LMI_STR	SDRAM timing	0x0018	R/W
LMI_PBS	Reserved	0x0020	-
LMI_COC	Padlogic control	0x0028	R/W
LMI_SDRA[0:1]	SDRAM row attribute	0x0030, 0x0038	R/W
LMI_CIC	Padlogic status	0x0040	R/W
LMI_SDMR[0:1]	SDRAM mode	0x0048, 0x0050	WO

Table 66: LMI glue register summary

Register	Description	Offset	Type

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. Used to identify 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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DRI											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				BY32AP	DIMM	DRE	ENDIAN	BW		Reserved				DT	DCE

Address: *LMIBaseAddress* + 0x0008

Type: R/W

Reset: 0x061A 0080

Description: This register specifies the configuration of the DRAM interface.

[63:28] **Reserved**

[27:16] **DRI**: DRAM refresh interval. Determines the maximum number of memory clock cycles between row refreshes, when enabled.

[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 row
1: Data output delayed by 1 LMICLK cycle and 1 LMICLK cycle added to CAS latency

[9] **DRE**: DRAM refresh enable. Enable refresh mechanism.

[8] **ENDIAN**: memory endianness (read only). Indicates whether the memory configuration is little or big endian. This bit only affects 16-bit and 32-bit bus width modes. For 64-bit external interface, the endianness is transparent to the LMI.
0: little endian memory configuration 1: big endian memory configuration

[7:6] **BW**: bus width. Indicates the data bus width of the LMI SDRAM interface.
00: 16 01: 32
10, 11: Reserved

[5:2] **Reserved**

[1] **DT**: DRAM type (read only). Specifies the SDRAM type
0: Reserved 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 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
Reserved				CST											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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.

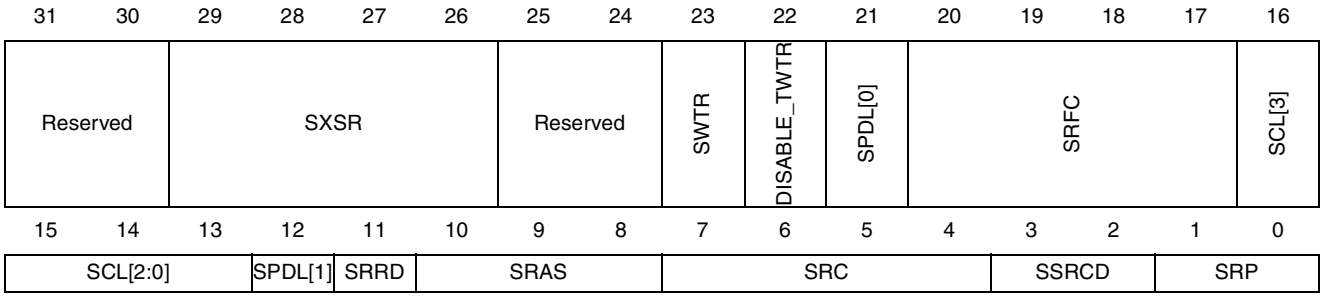
[15:7] **Reserved**

[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*] number of 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*] number of 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



Address: *LMIBaseAddress* + 0x0018

Type: R/W

Reset: 0

Description:

[63:30] **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] **Reserved**

[23] **SWTR**: T_{WTR}, write to read interruption. T_{WTR} parameter: last data-in (during write) to read command.
0: 1 clock 1: 2 clocks

[22] **DISABLE_TWTR**: disable the write to read interruption.
0: enable write to read interruption. 1: disable write to read interruption.

[21] **SPDL[0]**: see SDPL field

[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	10: 2 clocks	01: 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] **SRCD**: T_{RCD}, RAS to CAS delay. Controls the number of LMICLKs from a row activate command to a column command (for the same bank).
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

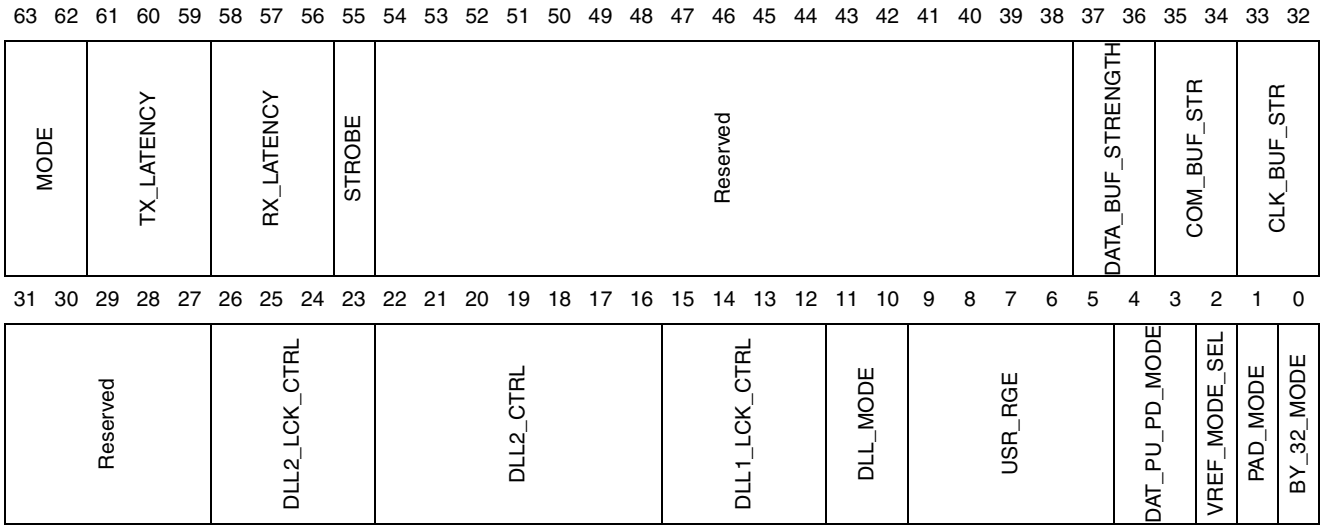
[1:0] **SRP**: T_{RP}, RAS precharge to ACT command. Controls the number of LMICLKs for RAS precharge to ACT command (for the same bank).
00: 2 clocks of RAS precharge 01: 3 clocks of RAS precharge
10: 4 clocks of RAS precharge 11: 5 clocks of RAS precharge

LMI_PBS Reserved

31	30	29	28	27	26	25	24	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																															

Address: *LMIBaseAddress* + 0x0020
Reset: 0

LMI_COC Padlogic control



Address: *LMIBaseAddress* + 0x0028

Type: R/W

Reset: 0

Description: This register sets various parameters that control padlogic functional behavior.

[63:55] LMI Async Bridge Control
This part of the register controls the asynchronous bridge configuration within the LMI.

[63:62] **MODE**
00 - synch/bypass
01 - semisynch/no retime FF
10 - semisynch/1 retime FF
11 - semisynch/2 retime FF

[61:59] **TX_LATENCY**
000 - Latency of 8 cycles
001 - Latency of 7 cycles
…

[58:56] **RX_LATENCY**
000 - Latency of 8 cycles
001 - Latency of 7 cycles
…

[55] **STROBE**: Strobe signal to validate configuration data.

[54:38] **Reserved**

[37:32] Padlogic Pin Buffer Strength
This part of the register sets the output buffer drive strength for the SSTL2 pads used in the DDR padlogic. Each two-bit control field has the following encoding:
00 - 5pf 01 - 15pf
10 - 25pf 11 - 35pf

[37:36] **DATA_BUF_STRENGTH**: Data buffer strength
This field selects the drive strength for the DQ, DM, and DQS DDR pins.

[35:34] **COM_BUF_STR**: Command buffer strength
This field selects the drive strength for the CKE, CS, RAS, CAS, WE, A, and BA DDR pins.

[33:32] **CLK_BUF_STR**: Clock buffer strength
This field selects the drive strength for the CK and CK# DDR pins.

[31:27] **Reserved**

[26:23] **DLL2_LCK_CTRL**: DLL2 lock control
See the DLL1 lock control field for a description.

[22:16] **DLL2_CTRL**: DLL2 control
See the DLL1 control field for a description.

[15:12] **DLL1_LCK_CTRL**: DLL1 lock control
Defines the number of consecutive up/down pulses trigger the lock status.
0000 - LOCK goes high when 5 consecutive up or down pulses are detected.
0001 - LOCK goes high when 2 consecutive up or down pulses are detected. Lock generation is very sensitive to jitter.
0010 - LOCK goes high when 3 consecutive up or down pulses are detected.
0011 - LOCK goes high when 4 consecutive up or down pulses are detected.
0100 - LOCK goes high when 5 consecutive up or down pulses are detected.
0101 - LOCK goes high when 6 consecutive up or down pulses are detected.
0110 - LOCK goes high when 7 consecutive up or down pulses are detected.
0111 - LOCK goes high when 8 consecutive up or down pulses are detected.
1000 - LOCK goes high when 9 consecutive up or down pulses are detected.
1001 - LOCK goes high when 10 consecutive up or down pulses are detected.
1010 - LOCK goes high when 11 consecutive up or down pulses are detected.
others - LOCK goes high when 5 consecutive up or down pulses are detected.

[11:5] DLL1 CONTROL
This field is divided into two sub-fields. For more detailed information, refer to the Pangea DLL specification.

[11:10] **DLL_MODE**
00 - Standalone mode. The delay command is calculated by the DLL.
01 - Delay command forced by the user. DLL_command = range[4:0] & “1000”.
10 - Delay command adjusted by the user.
DLL_command is created by adding range[4:0] to DLL_command[8:4] .
11 - Test mode. Delay element command and
DLL_command are forced to range[4:0] & “1000”.

[9:5] **USR_RGE**: User defined range[4:0]. Allowed values: 0 to 19.
Range[4:0] is used to set the upper part of the delay element command (bits 8 to 4) according to the DLL_mode.

[4:3] **DAT_PU_PD_MODE**: Data pull-up/pull-down mode.
This field selects the internal pad pull-up/pull-down configuration for the DQ, DM, and DQS pins.
Note: When the output buffer is enabled or tst_scan_mode is a 1, the pull-up, pull-down resistors are disabled.
00 - Pull-up and pull-down are disabled
01 - 50 k pull-down enabled (output must be disabled)
10 - 50 k pull-up enabled (output must be disabled)
11 - Forbidden

[2] **VREF_MODE_SEL**: VREF mode select
When clear, VREF is generated internally for each pad. When set, VREF must be supplied externally.

[1] **PAD_MODE**
When clear, the pad operates in SSTL receive mode.
When set, the pad operates in LVTTL receive mode.

[0] **BY_32_MODE**
When clear, each byte is captured by its respective strobe. When set, data is captured only by DQS0 for use with X32 DDR devices that have only one strobe.

LMI_SDRA[0:1] SDRAM row attribute

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

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

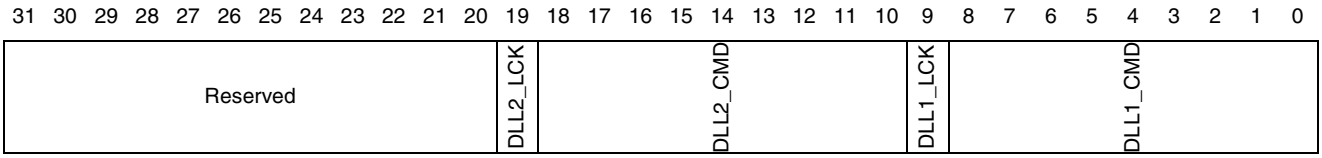
[11:8] **SPLIT**: SDRAM device address split for each bank. Defines the split of row and column address bits for a given bank within an SDRAM device.
0000: 11x8 0001: 11x9
0010: 11x10 0011: Reserved
0100: 12x8 0101: 12x9
0110: 12x10 0111: Reserved
1000: 13x8 1001: 13x9
1010: 13x10 1011: 13x11
11*nn*: Reserved

[7:2] **Reserved**

[1] **ENABLE_BA1**: 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] **ENABLE_BA0**: 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 Padlogic status



Address: *LMIBaseAddress* + 0x0040

Type: R/W

Reset: 0

Description: This register reports DLL command status information.

[31:20] **Reserved**

[19] **DLL2_LCK**: DLL2_lock
This bit is set when DLL2 has reached a steady locked state.

[18:10] **DLL2_CMD**: DLL2 command
This field reports the command currently being generated by DLL2.

[9] **DLL1_LCK**: DLL1 lock
This bit is set when DLL1 has reached a steady locked state.

[8:0] **DLL1_CMD**: DLL1 command
This field reports the command currently being generated by DLL1.

LMI_SDMR[0:1] SDRAM mode

Address: *LMIBaseAddress* + 0x0048, 0x0050

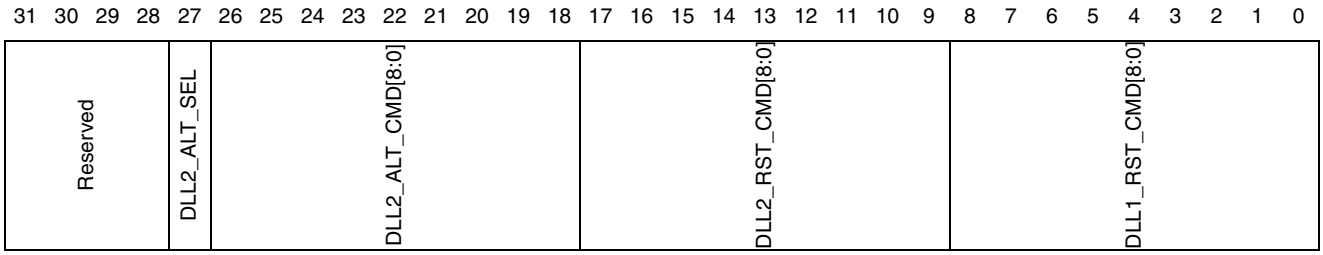
Type: WO

Description: These registers 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 row of SDRAM devices. The value on physical address A[12:3] is copied to LMIADD[9:0] pins, A[14:13] is output to LMIBKSEL[1:0] and A[18:15] is driven to LMIADD[13:10]. The values on data pins are undefined and are ignored by the SDRAM devices. In response to the mode register set command, a DDR SDRAM device then latches LMIADD[11:0] and LMIBKSEL[1:0] into its mode register. A read to these registers returns an undefined value.

Refer to the DDR SDRAM manufacturers' 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.

LMI_SYSTEM_GLUE LMI System glue



Address: *MicroSystemGlueConfigBaseAddress + 0x00*

Type: R/W

Reset:

Description:

[31:28] **Reserved**

[27] **DLL2_ALT_SEL**: selects the alternate command for DQS delay element

[26:18] **DLL2_ALT_CMD[8:0]**: alternate command for DQS delay element override

[17:9] **DLL2_RST_CMD[8:0]**: command applied to DLL2 at LMI PadLogic reset

[8:0] **DLL1_RST_CMD[8:0]**: command applied to DLL1 qt PadLogic reset

23 Test access port (TAP)

The STi7710 TAP conforms to IEEE standard 1149.1, and includes Device ID information. Pins are listed in Table 67. TCK can be stopped in either logic state.

Table 67: 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 5 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 68. All other instruction codes are reserved.

Table 68: 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.

24 Test access port (TAP) registers

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*D42 1041, where *n* is the four-bit silicon revision number.

Table 69: Identification code

Bit 31							**Bit 0** [a]
Mask rev	**ST20 family**		**Variant**		**STMicroelectronics manufacturers ID**		**b**
n	D	4	2	1	0	4	1

a. Closest to TDO

b. Defined as 1 in IEEE 1149.1 standard

25 Diagnostic controller (DCU)

25.1 Overview

The STi7710 diagnostic controller unit (DCU) is used to boot the CPU and to control and monitor all of the systems on the chip, via the standard IEEE 1194.1 test access port. The DCU includes on-chip hardware with ICE (in-circuit emulation) and LSA (logic state analyzer) features to facilitate verification and debugging of software running on the on-chip CPU in real time. It is an independent hardware module with a private link from the host to support real-time diagnostics.

The DCU has the following features.

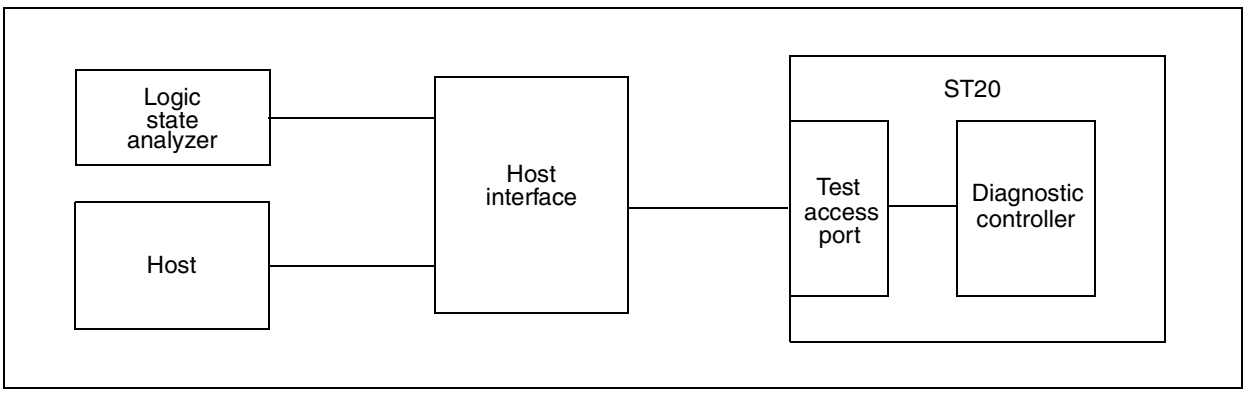
- Unified compare blocks, each of which is capable of breakpoint, breakcount, breakrange, inverse breakrange, watchrange, inverse watchrange, datawatch or watchcount operations. The DCU on STi7710 contains four compare blocks.
- Four capture blocks. Each capture block can store the value of the instruction pointer, workspace pointer, data address or write data value when a particular event occurs (for example, match from a compare block).
- A jump trace which can store:
  - instruction pointer to the source of the code jump,
  - instruction pointer to the destination of the code jump,
  - value in a capture block,
  - cycle count.

  The events which cause these to be stored are selected from: Code jump, context change (change between threads), event from a compare block and store of a capture block. The DCU jump trace stores the information in a compressed format.

25.2 Diagnostic hardware

The on-chip diagnostic controller assists in debugging, while either reducing or eliminating the intrusion into the target code space, the CPU utilization and impact on the application. As shown in Figure 40, the DCU and TAP provide a means of connecting a diagnostic host to a target board with a suitable JTAG port connector and interface.

Figure 40: Debugging hardware



The diagnostic controller provides the following facilities for debugging from a host:

- control of target CPU and subsystems including CPU boot,
- hardware breakpoint, watchpoint, datawatch and single instruction step,
- complex trigger sequencing and choice of subsequent actions,
- nonintrusive jump trace and instruction pointer profiling,
- access to the memory of the target while the device is powered up, regardless of the state of the CPU,
- full debugging of ROM code.

When running multitasking code on the target, one or more processes can be single stepped or stopped while others continue running in real time. In this case, the running threads can be interrupted by incoming hardware interrupts, with a low latency.

The host can communicate with the DCU via a private link, using the five standard test pins.

Target software allows access to the diagnostic facilities and access through the DCU to the host memory.

A logic state analyzer can be connected to the DCUTRIGGERIN and DCUTRIGGEROUT pins. The response to DCUTRIGGERIN and the events that cause a TRIGGEROUT signal can be controlled by the host or by target software.

The diagnostic controller provides debugging facilities with much less impact on the software and target performance. In particular:

- nonintrusive attachment to the host system,
- no intrusion into the performance of the CPU or any subsystems,
- no intrusion into the code space, so the application builder does not need to add a debugging kernel,
- no intrusion into any on-chip functional modules, including any communications facilities,
- no functional external connection pins are used.

The connections between the diagnostic controller and other on-chip modules and external hardware may vary between ST20 variants.

25.3 Access features

25.3.1 Access to target memory and peripheral registers from the host

Full read and write access to the entire on-chip and external memory space, and the register space, is available via the TAP. This is independent of the state of the CPU.

25.3.2 Access from the target CPU process

The CPU itself can program its own diagnostic controller. Further access may be explicitly prevented by the lock mechanism so that the application being debugged cannot interfere with the breakpoint and watchpoint settings. When the breakpoint or watchpoint match occurs, then the diagnostic controller may release the lock according to settings in the DCU_CONTROL register.

25.3.3 Access to host memory from target

If the target CPU accesses any address in the top half of the DCU memory space, these accesses are mapped on to host memory via the TAP as target initiated PEEK and POKE messages. Peek and poke accesses are specifically enabled by separate property bits.

25.4 Software debugging features

25.4.1 Control of the target CPU including boot

Various state information about the target CPU may be monitored, and the CPU may be controlled from the diagnostic controller via the TAP. The control of the CPU extends to stalling, forcing a trap and booting.

25.4.2 Nonintrusive IPTR profiling

A copy of the IPTR is visible as a read-only register in the diagnostic controller. This register may be read at any time. Reading this register is not intrusive on the CPU or its memory space.

25.4.3 Events

Support is provided by the diagnostic controller to trigger actions when certain predefined events occur.

Table 70: Software debugging events

Event	Action
Breakpoint	The function of the breakpoint is to break before the instruction is executed, but only if it really was going to be executed. A 32-bit comparator is used to compare the breakpoint register against the instruction pointer of the next instruction to be executed. The matched instruction is not executed and the CPU state, including all CPU registers, is defined as at the start of the instruction. The previous instruction is run to completion.
Breakpoint range	The function of a breakpoint range is equivalent to any single breakpoint but where the breakpoint address can be anywhere within a range of addresses bound by lower and upper register values.
Watchpoint	The function of a watchpoint is to trigger after a memory access is made to an address within the range specified by a pair of 32-bit registers. The CPU pipeline architecture allows for the CPU to continue execution of instructions without necessarily waiting for a write access to complete. So, by the time a watchpoint violation has been detected, the CPU may have executed a number of instructions after the instruction which caused the violation. If the subsequent action is to stall the CPU or to take a hardware trap, then the last instruction executed before the stall or trap may not be the instruction which caused the violation.
Datawatch	The function of a datawatch is to trigger after a data value specified in one 32-bit register is written to a memory word address specified in another 32-bit register. The subsequent action is equivalent to a watchpoint.

Following a watchpoint match, or any other condition detectable by the diagnostic controller, the subsequent action may be programmed to do one of the following:

- stall the CPU, that is, inhibit further instructions from being executed by the CPU,
- wait until the end of the current instruction, then signal a hardware trap,
- signal an immediate hardware trap,
- continue without intrusion.

In addition, the diagnostic controller may take any combination of the following actions:

- signal on TRIGGEROUT to a logic state analyzer,
- send a triggered message via the TAP to the host,
- unlock access by the target CPU.

25.4.4 Hardware single instruction step

The function of single stepping one CPU instruction is performed by using a breakpoint range over the code to be single stepped. The DCU includes a mechanism to prevent the breakpoint trap handler single stepping itself. By selecting an inverse range, the effect of single stepping one high level instruction can be achieved.

25.4.5 Jump trace

Jump tracing monitors code jumps, where a jump is any change in execution flow from the stream of consecutive instructions stored in memory. A jump may be caused by a program instruction, an interrupt or a trap.

When the jump occurs, a 32-bit DCU register is loaded with the origin of the jump. This value points to the instruction which would have been executed next if the jump had not occurred. The CPU may not have completed the instruction prior to the change in flow. The diagnostic controller can be set to trace the origin of each jump, the destination, or both.

The DCU copies the details of each jump to a rolling trace buffer in memory. The trace buffer may be located in host memory, but using target memory has less impact on performance. The tracing facility has two modes.

- Low intrusion: In this mode, the DCU uses dead memory cycles to write the trace into the buffer. This means that the CPU is not delayed, but some trace information may be lost.
- Complete trace: In this mode, the CPU is stalled on every jump to ensure the data can be written to the buffer. This means that no trace information is lost, but the CPU performance is affected.

25.4.6 Logic state analyzer (LSA) support

Two signals, TRIGGERIN and TRIGGEROUT, are provided to support diagnostics with an external LSA. The action by the DCU on receiving a TRIGGERIN signal is programmable. The selection of internal events which trigger a TRIGGEROUT signal is also programmable.

25.4.7 Trigger combinations and sequences

Complex trigger conditions can be programmed. For example, the fifth time that breakpoint 3 is encountered or to enable a watchpoint when a breakpoint occurs.

There is no software intrusion imposed by this mechanism.

25.5 Controlling the diagnostic controller

This section gives a summary of host communications with the diagnostic controller.

The diagnostic controller has direct access to:

- the instruction pointer,
- a selection of CPU state control signals,
- the memory bus,
- memory-mapped peripheral configuration registers.

This access does not depend on the state of the CPU. Access to nonmemory-mapped peripheral configuration registers is via the CPU, and for this the CPU must be active and running the appropriate handler.

The host can give two commands to the diagnostic controller: **peek** and **poke**. **peek** reads memory locations or configuration registers, and **poke** writes to memory locations or configuration registers. The diagnostic controller responds to a **peek** command with a PEEKED message, giving the contents of the peeked addresses.

The diagnostic controller has registers, which are accessed from the host using **peek** and **poke** commands. The registers are used to control breakpoints, watchpoints, datawatch, tracing and other facilities.

The target CPU can also access these registers using the normal **load** and **store** instructions, so the target software running on the CPU can program its own diagnostic controller. A lock is provided to prevent CPU access, which can be released by the diagnostic controller when a breakpoint or watchpoint match occurs.

In addition, the target CPU can peek and poke the host via the diagnostic controller by reading or writing addresses in the top half of the memory space of the diagnostic controller. This facility can be disabled.

Various different types of CPU events can be selected as trigger events. When a trigger event occurs, the diagnostic controller can send a triggered message.

The four types of message are summarized in Table 71 below. The messages are distinguished by the two least significant bits of the message header byte.

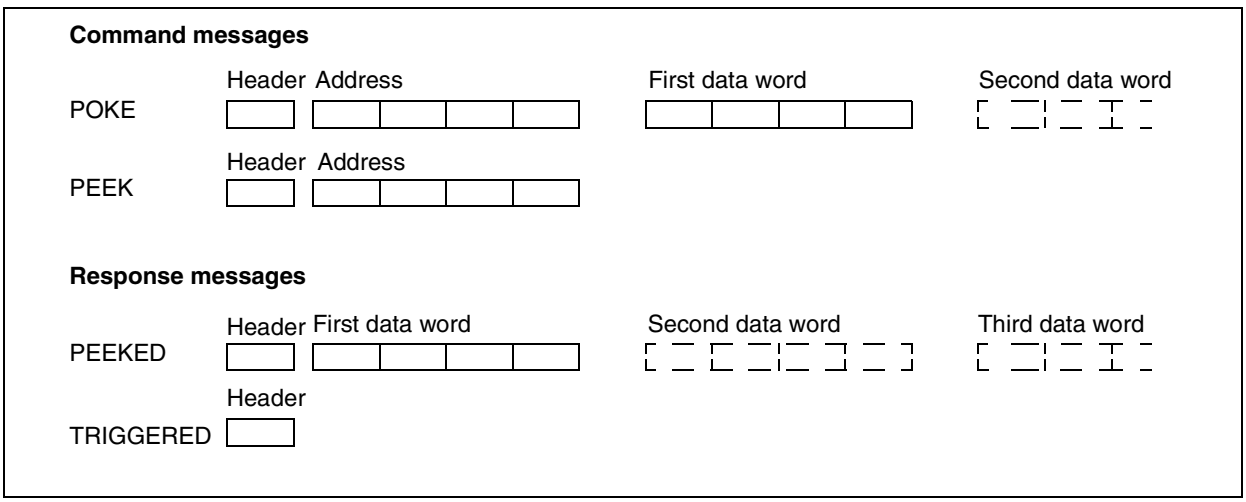
Table 71: Diagnostic controller message types

Message type	Direction	Bit 1	Bit 0	Meaning
POKE	Command	0	0	Write to one or more addresses
PEEK	Command	0	1	Read from one or more addresses
PEEKED	Opposite of **peek** command	1	0	The result of a **peek** command
TRIGGERED	DCU to host	1	1	A trigger event has occurred

Messages may be initiated from either the host or the target. Target initiated messages, which constitute asynchronous or unsolicited messages, can be enabled by a property bit.

Messages are composed of a header byte followed by zero or more data bytes, depending on the type of message. The formats for the four message types are shown in Figure 41.

Figure 41: Message formats



25.6 Peeking and poking the host from the target

The target CPU can peek and poke the host via the diagnostic controller. This is done by reading or writing a single word to a block of addresses within the DCU register block. The DCU then sends a PEEK or POKE message to the host. After a host PEEK, the target CPU waits until the host responds with a peeked message, which the DCU returns to the CPU as memory read data.

Peeking and poking the host from the target can be enabled or disabled. After reset, these bits are cleared, so peek and poke from the target are disabled.

26 Diagnostic controller (DCU) registers

All the functions of the DCU are controlled by values in its registers. The base address of the DCU is programmable depending on the value of the *DCUBaseAddress* input.

The addresses given in the following tables are offsets from this DCU base address. The register allocation allows for up to 32 blocks (any except jump trace). For any particular implementation, the value in the capability register and sequence configuration register can be used to calculate the address of all registers. Table 72: *DCU registers summary* shows addresses, assuming three compare blocks, two capture blocks, one TRIGGER_IN block, one jump trace and one work space range compare block, with full connectivity of sequencing.

Addresses are provided as *DCUBaseAddress* + offset.

The *DCUBaseAddress* is:

0x3000 0000.

In the table below, *n* is the number of trigger_in blocks defined in DCU_CAPABILITY.

Table 72: DCU registers summary

Register	Description	Offset	Type
General registers			
DCU_CAPABILITY	Capability	0x000	RO
DCU_CONTROL	Control	0x004	R/W
DCU_SIGNALLING	Signalling	0x008	R/W
DCU_STATUS	Status	0x00C	RO
DCU_TRIGGER_IN	Trigger in status	0x010	RO
DCU_COMPARE_STATUS	Compare status	0x014	RO
DCU_SEQUENCING_CONFIGURATION	Sequencing configuration	0x040	RO
DCU_TRIGGER_IN_PROPERTIES	Trigger in properties	0x080 + 4 x *n*	R/W
Jump trace registers			
DCU_JUMPTRACE_PROPERTIES	Jump trace properties	0x100	R/W
DCU_JUMPTRACE_FROM_LPTR	Jump trace from LPTR	0x104	RO
DCU_JUMPTRACE_TO_LPTR	Jump trace to LPTR	0x108	RO
DCU_JUMPTRACE_LAST_LPTR	Jump trace last LPTR	0x10C	RO
DCU_JUMPTRACE_LAST_CAPTURE0	Jump trace last capture0	0x110	RO
DCU_JUMPTRACE_BYTES	Jump trace bytes	0x114	RO
DCU_JUMPTRACE_START_ADDRESS	Jump trace start address	0x118	R/W
DCU_JUMPTRACE_END_ADDRESS	Jump trace end address	0x11C	R/W
DCU_JUMPTRACE_ADDRESS	Jump trace address	0x120	R/W
DCU_COMPAREn_PROPERTIES	Compare properties	0x200 + 0x10 x *n*	R/W
DCU_COMPAREn_VALUE1	Compare value1	0x204 + 0x10 x *n*	R/W
DCU_COMPAREn_VALUE2	Compare value2	0x208 + 0x10 x *n*	R/W
Capture registers			
DCU_CAPTUREn_PROPERTIES	Capture properties	0x400 + 0x08 x *n*	R/W
DCU_CAPTUREn_VALUE	Capture	0x404 + 0x08 x *n*	RO
Sequencing registers			
DCU_SEQUENCING_ENABLE	Sequencing enable	0x500 + 0x08 x *n*	R/W
DCU_SEQUENCING_DISABLE	Sequencing disable	0x504 + 0x08 x *n*	R/W
Work space range enable registers			
DCU_WRANGE_EN_ONLY_IN_RGE	Wrange enable only in-range	0x600 + 0x10 x *n*	R/W
DCU_WRANGE_EN_ONLY_OUT_RGE	Wrange enable only out of range	0x604 + 0x10 x *n*	R/W
DCU_WRANGE_LOWER	Wrange lower	0x608 + 0x10 x *n*	R/W
DCU_WRANGE_UPPER	Wrange upper	0x60C + 0x10 x *n*	R/W

26.1 General registers

Note: All unused register bits have value zero when read and should be written with zero.

DCU_CAPABILITY Capability

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

Address: *DCUBaseAddress* + 0x000

Type: RO

Reset: See table

Description: The capability register provides information about the hardware configuration of a particular implementation of the DCU to allow software to see what hardware is available.

[31:29] **MAJOR_VERSION**
DCU major version number — Reset (depends on implementation): 1

[28:26] **MINOR_VERSION**
DCU minor version number — Reset (depends on implementation): 0

[25:21] **Reserved**

[20:16] **NUM_W_RANGE**
The number of work space range detect blocks — Reset (depends on implementation): 1

[15:11] **NUM_CAPTURE**
Number of capture blocks — Reset (depends on implementation): 2

[10:6] **NUM_COMPARE**
Number of compare blocks — Reset (depends on implementation): 3

[5] **JTR_SUPPORT**
1 if jump trace supported, 0 if not. — Reset (depends on implementation): 1

[4:0] **NUM_TRIG_IN**
The number of TRIGGER_IN blocks — Reset (depends on implementation): 1

DCU_CONTROL 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
DCU_CONTROL_OUT																IN_TRAP_LOCK	IN_TRAP	BYTE_ENABLES [3:0]				DEVICE_ACCESS	DISABLE	TARGET_PEEK_ENABLE	TARGET_POKE_ENABLE	WRITE_LOCK	HOSTED_BUSY	TRIGGER	SINGLE_STEP	CPU_TRAP	CPU_STALL

Address: *DCUBaseAddress* + 0x004

Type: R/W

Reset: 0 (except BYTE_ENABLES: 1111 and CPU_STALL: depends on HOST_CONNECTED)

Description:

[31:16] **DCU_CONTROL_OUT:** General purpose outputs for functions like CPU reset, subsystem reset and stall
Output pins from the DCU reflect whatever value is written to DCU control out. The connection of these pins are product specific and include controls required to reset or stall the product during debugging.

[15] **IN_TRAP_LOCK:** Disables IN_TRAP from being cleared automatically on an IPTR jump
This bit is set automatically on entering the diagnostics trap handler. It is cleared by the user at the end of the trap handler to re-enable the DCU functions on exiting the trap handler. Interrupts need to be turned off before clearing this bit to ensure that the first jump with this bit clear is the trap return.

[14] **IN_TRAP:** Disables the DCU functions
This is automatically set when the CPU invokes the diagnostics trap handler and is automatically cleared on the next NEW_IPTR_LOADED that IN_TRAP_LOCK is not set. The return from the trap handler.

[13:10] **BYTE_ENABLES:** Selects which bytes of host memory access are to be written/read
Host accesses are only to the bytes of the word which are specified in this field. To do a part word access, this field must have only the required bits set. To do a full word host access, all bits must be set.

[9] **DEVICE_ACCESS:** Causes DCU memory accesses to be device accesses (to bypass the cache)

[8] **DISABLE_GROUPING:** Disables the instruction fetch in the CPU from grouping instructions

[7] **TARGET_PEEK_ENABLE**
Enables reads to the hosted memory area to be automatically translated to peeks to the host when not executing the diagnostics trap handler (this feature is always enabled while IN_TRAP is high). While not enabled, reads from hosted addresses return undefined data.

[6] **TARGET_POKE_ENABLE**
Enables writes to the hosted memory area to be automatically translated to pokes to the host when not executing the diagnostics trap handler (this feature is always enabled while IN_TRAP is high). While not enabled, writes to hosted addresses are ignored.

[5] **WRITE_LOCK**
When set, CPU write accesses to the DCU register space have no effect except when the CPU is executing the diagnostics trap handler (while IN_TRAP is high).

[4] **HOSTED_BUSY**
High if the host interface is in use, indicates that an access to hosted memory would cause the CPU to hang and wait for the hosted memory interface to become available. This is also high when the host is not connected. No behavior on write to this bit.

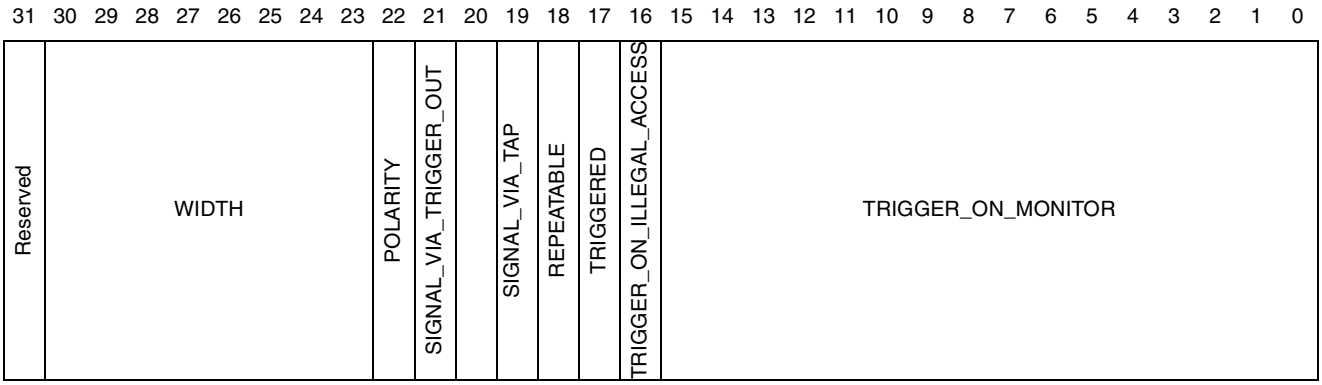
[3] **TRIGGER: s**oftware input to the signalling scheme (see *DCU_SIGNALLING* *on page 213*)
This can be used for sending trigger messages to the host or via TRIGGER_OUT.

[2] **SINGLE_STEP**
While high, the DCU requests a TRAP_AT_NEXT_INSTRUCTION for every INSTRUCTION_STARTED or NEW_IPTR_LOADED seen (unless IN_TRAP is set), causing a hardware single instruction step. Can be disabled by WRANGE enable.

[1] **CPU_TRAP:** diagnostics trap at the next interrupt point
1: sets a diagnostics trap at the next interrupt point. This bit automatically clears itself immediately.

[0] **CPU_STALL:** Stalls the CPU at the next interrupt point and causes it to remain stalled until bit is cleared
The reset value depends on the signal HOST_CONNECTED, so if a host is connected at the end of reset, this is high and stalls the CPU for a boot from DCU otherwise it resets low and so does not stall the CPU as it comes out of reset.

DCU_SIGNALLING Signalling



Address: *DCUBaseAddress* + 0x008

Type: R/W

Reset: See table

Description: The trigger on inputs of the signalling register consist of the monitor inputs which are enabled, trigger from TRIGGER_IN block, jump trace full, compare matched or software trigger from the control register.

[31] **Reserved**

[30:23] **WIDTH**
This field specifies the number of extra clock cycles to hold TRIGGEROUT high when it is being used to generate pulses (0: Single cycle pulse).
Reset: 0x00

[22] **POLARITY**
If this bit is high, TRIGGEROUT is inverted (active low).
Reset: 0

[21:20] **SIGNAL_VIA_TRIGGER_OUT**
00: Don't signal via TRIGGEROUT.
01: TRIGGEROUT pulses high for one cycle on any new TRIGGER_ON condition.
10: TRIGGEROUT is a level, the OR of any enabled conditions.
11: TRIGGEROUT is the same value as TRIGGERED.
Reset: 00

[19] **SIGNAL_VIA_TAP**
If set, then rising edge of the triggered bit causes a triggered message to be sent to the host.
Reset: 0

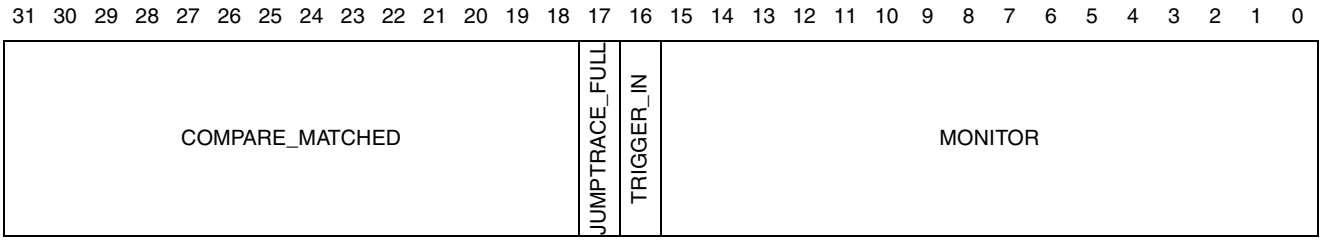
[18] **REPEATABLE**
If this is set then triggered is automatically cleared the cycle after it is set, otherwise it remains set (that is, for multiple triggers rather than one shot).
Reset: 0

[17] **TRIGGERED**
Reflects that one of the TRIGGER_ON events that signalling is sensitive to has occurred.
This gets set when any of the TRIGGER_ON inputs goes high. If repeatable is set then it is cleared automatically the next cycle, otherwise it remains set until it is cleared by a write of 0 to this register bit (a write of 1 is ignored).
Reset: 0

[16] **TRIGGER_ON_ILLEGAL_ACCESS**
If set, detects an access to registers or hosted memory when not enabled via the control register.
Reset: 0

[15:0] **TRIGGER_ON_MONITOR**
This mask selects the monitor bits to which signalling is sensitive. Triggered becomes set following the rising of any monitor for which the mask value is set.
Reset: 0x0000

DCU_STATUS Status



Address: *DCUBaseAddress* + 0x00C

Type: RO

Reset: See table

Description: The TRIGGER_IN, JUMPTRACE_FULL and COMPARE_MATCHED bits of the status register are identical to the corresponding TRIGGERED, FULL and MATCHED bits in the TRIGGER_IN, jump trace and compare properties registers. They are included in this status register to simplify polling the DCU to find the cause of a trap or stall. For typical implementations of the DCU this provides enough information. For implementations where there are more than one TRIGGER_IN block or more than 14 compare blocks, TRIGGER_IN status and compare status are provided as all of these bits don't fit in the status register.

[31:18] **COMPARE_MATCHED [*n*]**
Reflects the value of each of the match bits in the compare blocks. The number of bits depends on how many compare blocks there are.
Reset: 0 (*x n*)

[17] **JUMPTRACE_FULL**
High if JUMPTRACE_FULL bit is set.
Reset: 0

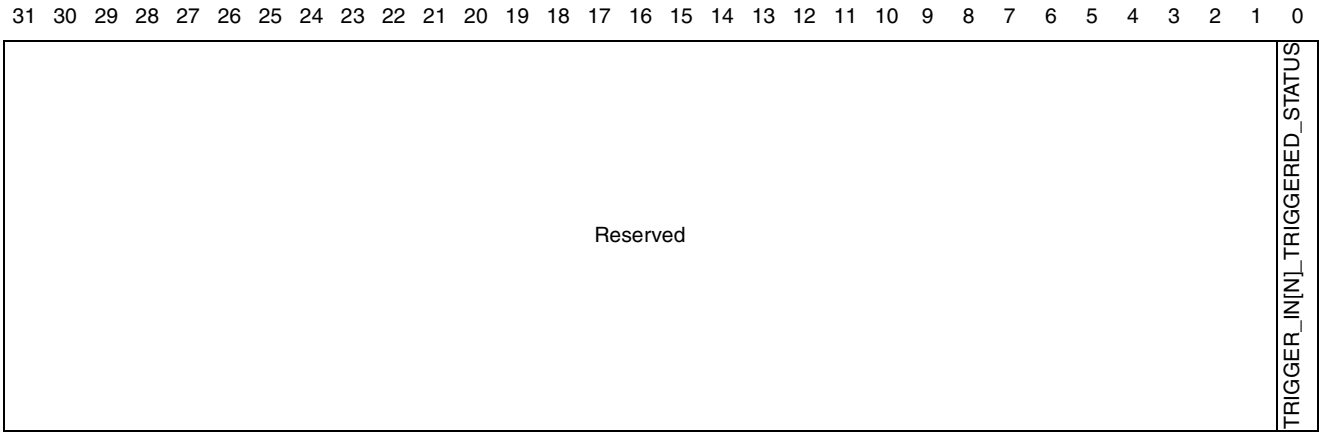
[16] **TRIGGER_IN**
Value of TRIGGERIN[0] input.
Reset: 0

[15:0] **MONITOR**
Value of bits on monitor input.
Bits 3:0 contain a decoded CPU run state, which is
different between the ST20-C1 and ST20-C2. The idle encoding is the same.

MONITOR[0]: C1 = User	C2 = Booting
MONITOR[1]: C1 = Trap	C2 = Halted
MONITOR[2]: C1 = Idle	C2 = Idle
MONITOR[3]: C1 = Interrupt	C2 = Running

Bits 15:4 are general purpose and product specific.
Reset: 0000000

DCU_TRIGGER_IN Trigger in status

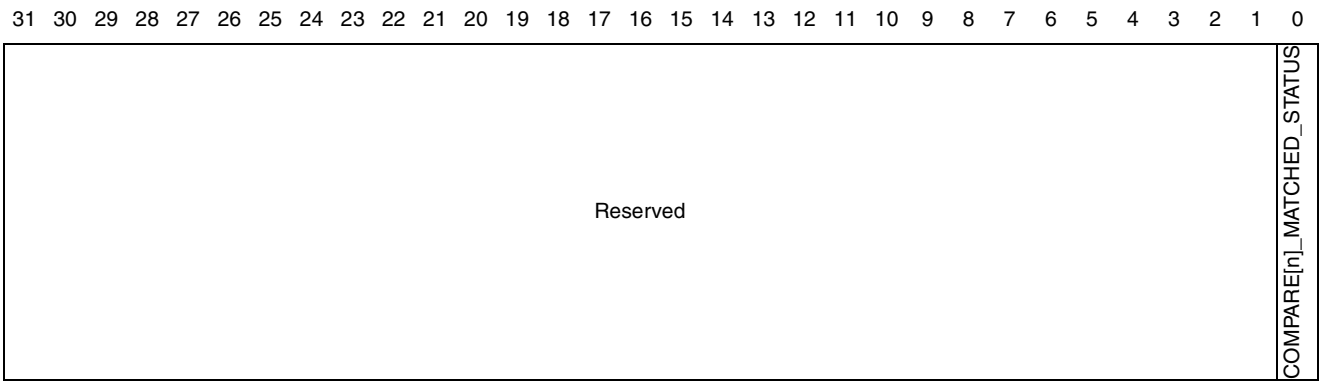


Address: *DCUBaseAddress* + 0x010
Type: RO
Reset: 0
Description:

[31:1] **Reserved**

[0] **TRIGGER_IN[n]_TRIGGERED_STATUS**
Reflects the value of the triggered bits of all the TRIGGER_IN blocks.

DCU_COMPARE_STATUS Compare status



Address: *DCUBaseAddress* + 0x014
Type: RO
Reset: 0
Description:

[31:1] **Reserved**

[0] **COMPARE[n]_MATCHED_STATUS**
Reflects the value of the matched bits of all the compare blocks.

DCU_SEQUENCING_CONFIGURATION Sequencing 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					COMPARE_EVENTS					JUMPTRACE_EVENTS	TRIGGER_IN_EVENTS					CAPTURE					COMPARE					JUMPTRACE	TRIGGER_IN			

Address: *DCUBaseAddress* + 0x040

Type: RO

Reset: See below

Description: The sequencing configuration register provides information about the hardware configuration of a particular implementation of the DCU to allow software to find the register and bit allocations for the sequencing and wrange blocks.

[31:27] **Reserved**

[26:22] **COMPARE_EVENTS**
Number of compare blocks that contribute to events detected by the sequencing blocks.
Reset (depends on implementation): 3

[21] **JUMPTRACE_EVENTS**
Number of jump trace blocks that contribute to events detected by the sequencing blocks.
Reset (depends on implementation): 1

[20:16] **TRIGGER_IN_EVENTS**
Number of TRIGGER_IN blocks that contribute to events detected by the sequencing blocks.
Reset (depends on implementation): 1

[15:11] **CAPTURE**
Number of capture blocks that can be controlled by sequencing and work space range compare.
Reset (depends on implementation): 2

[10:6] **COMPARE**
Number of compare blocks that can be controlled by sequencing and work space range compare.
Reset (depends on implementation): 3

[5] **JUMPTRACE**
Number of jump trace blocks that can be controlled by sequencing and work space range compare.
Reset (depends on implementation): 1

[4:0] **TRIGGER_IN**
Number of TRIGGER_IN blocks that can be controlled by sequencing and work space range compare.
Reset (depends on implementation): 1

DCU_TRIGGER_IN_PROPERTIES Trigger in 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																									TRIGGER_TYPE		TRIGGER_ON_TRIGGER_IN	TRAP_ON_TRIGGER_IN	STALL_ON_TRIGGER_IN	CAPTURED	ENABLE

Address: *DCUBaseAddress* + (0x080 + 4 x *n*)
n is the number of trigger_in blocks defined in DCU_CAPABILITY.

Type: R/W

Reset: 0

Description:

[31:7] **Reserved**

[6:5] **TRIGGER_TYPE**
00: Trigger on high level
01: Trigger on high level
10: Trigger on rising edge
11: Trigger on falling edge

[4] **TRIGGER_ON_TRIGGER_IN**
If set will cause a trigger out when trigger_in is seen. Depends on setting of signalling register.

[3] **TRAP_ON_TRIGGER_IN**
If set will cause the CPU to take a diagnosis trap when TRIGGER_IN is seen.

[2] **STALL_ON_TRIGGER_IN**
If set, will cause the CPU to stall when trigger_in is seen.

[1] **CAPTURED**
Is set when trigger_in is seen. If Stall or Trap are set then it will stay set until cleared by the host or the CPU, otherwise it will clear automatically at the next cycle (Write of '1' is ignored).

[0] **ENABLE**
Enable this block.

26.2 Jump trace registers

DCU_JUMPTRACE_PROPERTIES Jump trace properties

Address: *DCUBaseAddress* + 0x100
Type: R/W
Reset: See table for details
Description:

DCU_JUMPTRACE_FROM_LPTR Jump trace from LPTR

Address: *DCUBaseAddress* + 0x104
Type: RO
Reset: Not reset
Description: IPTR before jump.

DCU_JUMPTRACE_TO_LPTR Jump trace to LPTR

Address: *DCUBaseAddress* + 0x108
Type: RO
Reset: Not reset
Description: Contains the current value of IPTR.

DCU_JUMPTRACE_LAST_LPTR Jump trace last LPTR

Address: *DCUBaseAddress* + 0x10C
Type: RO
Reset: No hard reset, 1 for soft reset
Description: Contains the value of the last IPTR written to the jump trace.

DCU_JUMPTRACE_LAST_CAPTURE0 Jump trace last capture0

Address: *DCUBaseAddress* + 0x110
Type: RO
Reset: No hard reset, 1 for soft reset
Description: Contains the value of the last capture0 written to the jump trace.

DCU_JUMPTRACE_BYTES Jump trace bytes

Address: *DCUBaseAddress* + 0x114
Type: RO
Reset: No hard reset, 0 for soft reset
Description: Contains the value of the last IPTR written to the jump trace.

DCU_JUMPTRACE_START_ADDRESS Jump trace start address

Address: *DCUBaseAddress* + 0x118
Type: R/W
Reset: No hard reset, 0 for soft reset
Description: Start of rolling jump trace buffer address.

DCU_JUMPTRACE_END_ADDRESS Jump trace end address

Address: *DCUBaseAddress* + 0x11C
Type: R/W
Reset: No hard reset, 0 for soft reset
Description: End of rolling jump trace buffer address, (jump trace buffer includes this address).

DCU_JUMPTRACE_ADDRESS Jump trace address

Address: *DCUBaseAddress* + 0x120
Type: R/W
Reset: No hard reset, 0 for soft reset
Description: Current rolling jump trace buffer address.

26.3 Compare registers

DCU_COMPAREn_PROPERTIES Compare properties

Address: *DCUBaseAddress* + (0x200 + 0x10 x *n*)
Type: R/W
Reset: See table
Description:

DCU_COMPAREn_VALUE1 Compare value1

Address: *DCUBaseAddress* + (0x204 + 0x10 x *n*)
Type: R/W
Reset: 0
Description: This register holds the first value which is used in the comparison operations.

DCU_COMPAREn_VALUE2 Compare value2

Address: *DCUBaseAddress* + (0x208 + 0x10 x *n*)
Type: R/W
Reset: 0
Description: This register holds the first value which is used in the comparison operations.

26.4 Capture registers

DCU_CAPTUREn_PROPERTIES Capture properties

Address: *DCUBaseAddress* + (0x400 + 0x08 x *n*)
Type: R/W
Reset: 0
Description:

DCU_CAPTUREn_VALUE Capture

Address: *DCUBaseAddress* + (0x404 + 0x08 x *n*)
Type: RO
Reset: Not reset
Description: Contains the value which has been captured.

26.5 Sequencing registers

DCU_SEQUENCING_ENABLE Sequencing enable

Address: *DCUBaseAddress* + (0x500 + 0x08 x *n*)
Type: R/W
Reset: 0
Description:

DCU_SEQUENCING_DISABLE Sequencing disable

Address: *DCUBaseAddress* + (0x504 + 8 x *n*)
Type: R/W
Reset: 0
Description: The positioning of the bits in these registers is implementation dependant. For any implementation, the positioning can be found by examining the sequencing configuration register, see *DCU_SEQUENCING_CONFIGURATION* *on page 216*.

26.6 Work space range enable registers

DCU_WRANGE_EN_ONLY_IN_RGE Wrange enable only in-range

Address: *DCUBaseAddress* + (0x600 + 0x10 x *n*)
Type: R/W
Reset: 0
Description:

DCU_WRANGE_EN_ONLY_OUT_RGE Wrange enable only out of range

Address: *DCUBaseAddress* + (0x604 + 0x10 x *n*)
Type: R/W
Reset: 0
Description: The positioning of the bits in these registers is implementation dependant. For any implementation, the positioning can be found by examining the sequencing configuration register. See *DCU_SEQUENCING_CONFIGURATION* *on page 216*.

[31:8] **Reserved**

[7] **CAPTURE1:** Only enable capture1 block if not in range.

[6] **CAPTURE0:** Only enable capture0 block if not in range.

[5] **COMPARE2:** Only enable compare2 block if not in range.

[4] **COMPARE1:** Only enable compare1 block if not in range.

[3] **COMPARE0:** Only enable compare0 block if not in range.

[2] **JUMPTRACE:** Only enable jump trace block if not in range.

[1] **TRIGGER_IN:** Only enable trigger in block if not in range.

[0] **SINGLE_STEP:** Only enable single step if work space pointer is not in range (that is, WPTR < lower reg or WPTR ≥ upper reg).

DCU_WRANGE_LOWER Wrange lower

Address: *DCUBaseAddress* + (0x608 + 0x10 x *n*)
Type: R/W
Reset: 0
Description: Contains the lower work space pointer value for the range comparison.

DCU_WRANGE_UPPER Wrange upper

Address: *DCUBaseAddress* + (0x60C + 0x10 x *n*)
Type: R/W
Reset: 0
Description: Contains the upper work space pointer value for the range comparison.

27 Modem analog front-end interface (MAFEIF)

27.1 Overview

The modem analog front end interface (MAFEIF) 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 MAFEIF, and the term sample period is the time from the start of one sample to the start of the next.

The MAFEIF 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 MAFEIF.

The MAFEIF 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 MAFEIF 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 MAFEIF alternately switch between the two pairs of pointers. While the MAFEIF 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 MAFEIF.

27.2 Using the MAFEIF to connect to a modem

The following table lists the pins that are used by the MAFEIF to connect a modem:

Table 73: MAFEIF pins

Name	Type	Function name (alternate)	Function description
PIO2[0]	O	MAFE_HC1	Indicates to the AFE that a control/status exchange is to take place.
PIO2[1]	O	MAFE_DOUT	Line for serially transmitting samples to the AFE.
PIO2[2]	I	MAFE_DIN	Line for serially receiving samples from the AFE.
PIO2[3]	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 Fs.
PIO2[4]	I	MAFE_SCLK	Modem system clock. The frequency should be less than half of the device system clock.

27.3 Software

The MAFEIF software manages the data exchange between the software modem and MAFEIF, and handles the control/status exchange.

27.3.1 Data exchange

When the MAFEIF 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 MAFEIF (this is probably not changed again),
4. enables status (complete) interrupt,
5. sets the control (run) bit,
6. deschedules.

The MAFEIF then processes a buffer load of samples (that is, it transmits 100 samples and receives 100 samples). When this is complete, the MAFEIF 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.

27.3.2 Control/status exchange

For a control/status exchange, the software writes to the MOD_CONTROL register 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.

28 Modem analog front-end interface (MAFEIF) registers

Addresses are provided as *ModemBaseAddress* + offset.

The *ModemBaseAddress* is:

0x2005 8000.

Table 74: MAFEIF register summary

Register	Description	Offset	Type
MOD_CONTROL_1	Control 1	0x00	R/W
MOD_STATUS	Status 1	0x04	RO
MOD_INT_ENABLE	Interrupt enable	0x08	R/W
MOD_ACK	Acknowledge	0x0C	WO
MOD_BUFFER_SIZE	Buffer size	0x10	R/W
MOD_CTRL_2	Control 2	0x14	WO
MOD_STATUS	Status 2	0x18	RO
MOD_RECEIVE0_POINTER	Receive memory buffer 0 start address	0x20	R/W
MOD_RECEIVE1_POINTER	Receive memory buffer 1 start address	0x24	R/W
MOD_TRANSMIT0_POINTER	Transmit memory buffer 0 start address	0x28	R/W
MOD_TRANSMIT1_POINTER	Transmit memory buffer 1 start address	0x2C	R/W

MOD_CONTROL_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 MAFEIF is to start exchanging data with the AFE.
0: The MAFEIF stops after completing the exchange of the current buffer load of samples.

MOD_STATUS 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 MAFEIF 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 MAFEIF 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 MAFEIF 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 MAFEIF is exchanging data with the AFE.
1: The RUN bit is low and the MAFEIF is not exchanging data. After the software clears the RUN bit, the MAFEIF only goes idle when it has finished exchanging the current buffer load of samples.

MOD_INT_ENABLE Interrupt enable

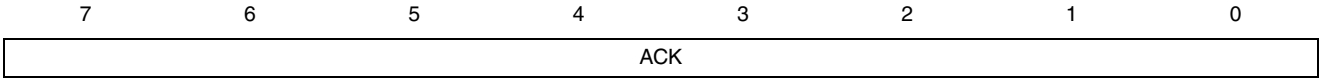
7	6	5	4	3	2	1	0
Reserved					INTENABLE2	INTENABLE1	INTENABLE0

Address: *ModemBaseAddress* + 0x08
Type: R/W
Reset: Undefined
Description:

[7:3] **Reserved**

[2:0] **INTENABLE[2:0]**
Enables interrupts connected to MOD_MAFE_STATUS[2:0].
1: Indicates that the corresponding interrupt is enabled.

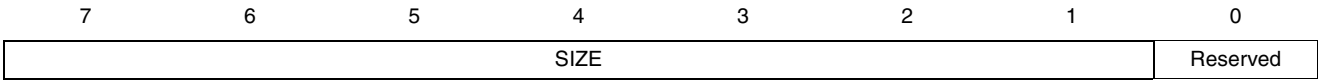
MOD_ACK Acknowledge



Address: *ModemBaseAddress* + 0x0C
Type: WO
Reset: Undefined
Description:

[7:0] **ACK:** acknowledge
Clears the overflow, missed and complete flags in the MOD_MAFE_STATUS register.

MOD_BUFFER_SIZE Buffer size

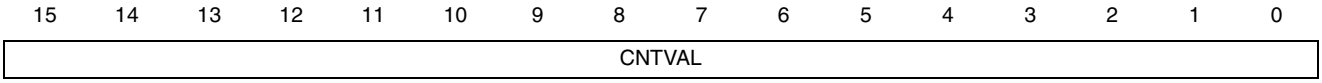


Address: *ModemBaseAddress* + 0x10
Type: R/W
Reset: Undefined
Description:

[7:1] **SIZE:** buffer size (the number of 16-bit samples in a buffer)
This value must be a multiple of two.

[0] **Reserved**

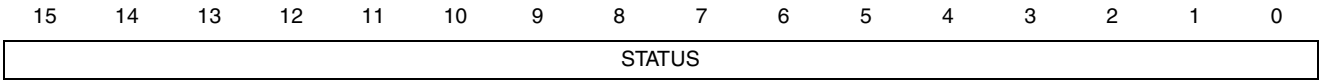
MOD_CTRL_2 Control 2



Address: *ModemBaseAddress* + 0x14
Type: WO
Reset: Undefined
Description:

[15:0] **CNTVAL:** control value to send out to the MAFEIF.

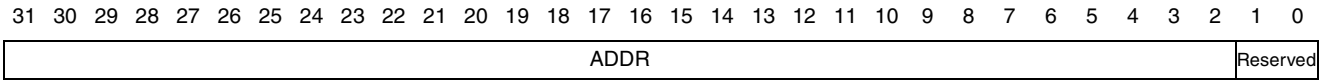
MOD_STATUS Status 2



Address: *ModemBaseAddress* + 0x18
Type: RO
Reset: Undefined
Description:

[15:0] **STATUS:** status value received from the MAFEIF.

MOD_RECEIVE0_POINTER Receive memory buffer 0 start address

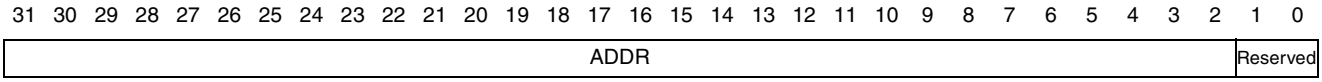


Address: *ModemBaseAddress* + 0x20
Type: R/W
Reset: Undefined
Description:

[31:2] **ADDR: s**tart address of the RECEIVE_MEMORY_BUFFER_0.

[1:0] **Reserved**

MOD_RECEIVE1_POINTER Receive memory buffer 1 start address

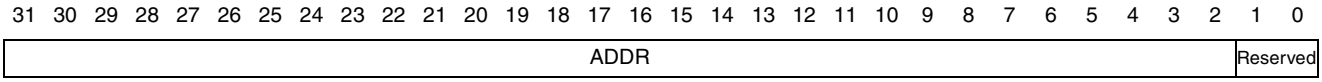


Address: *ModemBaseAddress* + 0x24
Type: R/W
Reset: Undefined
Description:

[31:2] **ADDR:** start address of the RECEIVE_MEMORY_BUFFER_1.

[1:0] **Reserved**

MOD_TRANSMIT0_POINTER Transmit memory buffer 0 start address

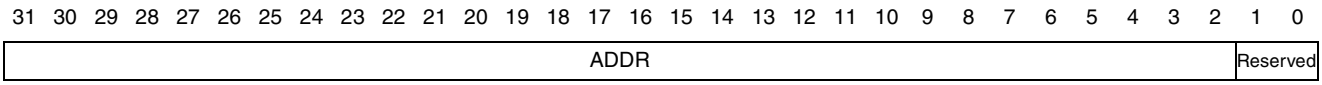


Address: *ModemBaseAddress* + 0x28
Type: R/W
Reset: Undefined
Description:

[31:2] **ADDR**: start address of the TRANSMIT_MEMORY_BUFFER_0.

[1:0] **Reserved**

MOD_TRANSMIT1_POINTER Transmit memory buffer 1 start address



Address: *ModemBaseAddress* + 0x2C
Type: R/W
Reset: Undefined
Description:

[31:2] **ADDR:** start address of the TRANSMIT_MEMORY_BUFFER_1.

[1:0] **Reserved**

29 Infrared transmitter/receiver

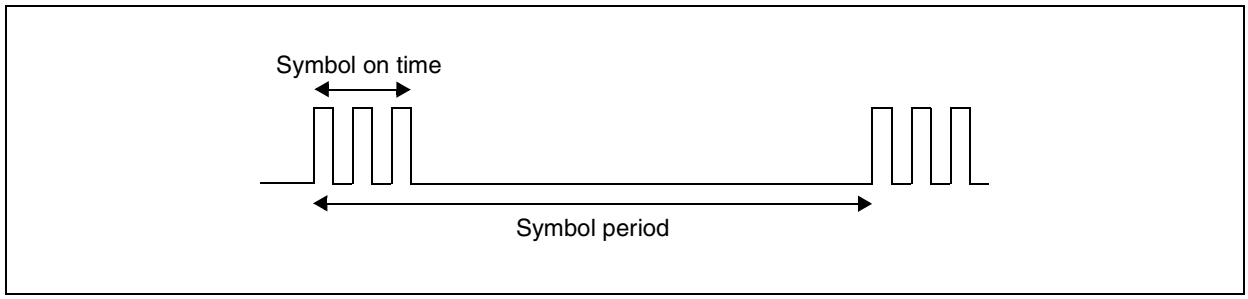
29.1 Overview

The infrared (IR) transmitter/receiver is an ST20 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 42.

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 42: IR transmitter/receiver symbol



29.2 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 43 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.

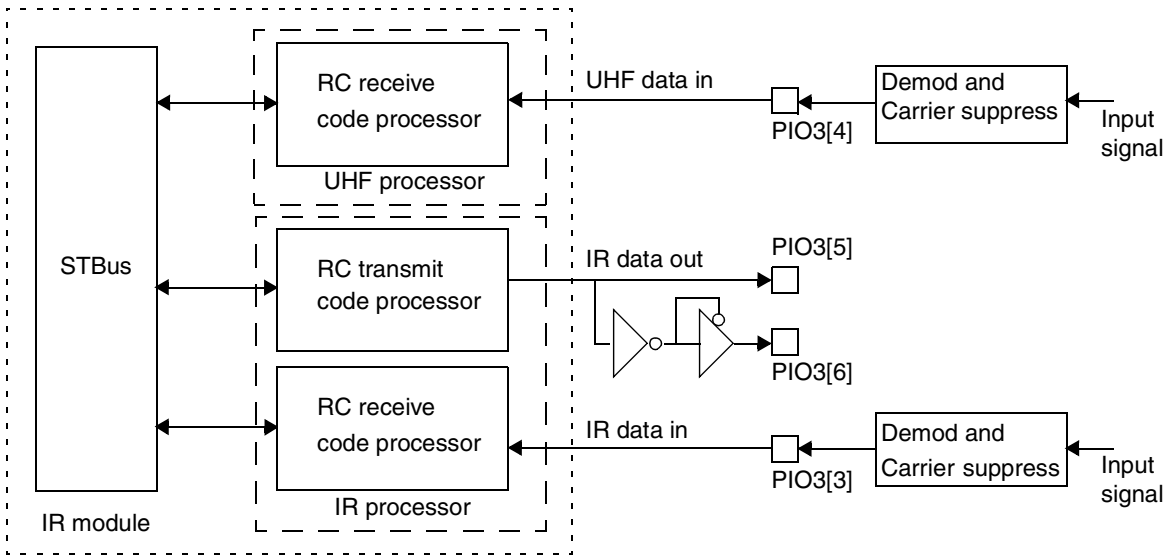
FIFOs buffer both the transmitter output and the receivers' inputs to avoid timing problems with the CPU. Interrupts can be set on the FIFOs' levels to prevent input data overrun and output data underrun.

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 to the STBus. 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 43: IR transmitter/receiver block diagram and implementation



Note: PIO3[6] must be programmed in open drain mode

29.2.1 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 indicates an underrun condition (that is, an empty FIFO). RC data transmission is disabled by setting bit 0 of register IRB_TX_EN_IR to 0.

The transmit interrupt is set by register IRB_TX_INT_EN_IR, 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 the registers IRB_TX_SYM_PERIOD_IR and IRB_TX_ON_TIME_IR. Register bits IRB_TX_INT_STATUS_IR[5:4] give the FIFO's fullness status.

The frequency of the subcarrier is set by programming the registers IRB_TX_PRE_SCALER_IR and IRB_TX_SUB_CARRIER_IR.

The symbol period, in subcarrier cycles, is programmed in the register IRB_TX_SYM_PERIOD_IR and the on time of the IR pulse is written to the register IRB_TX_ON_TIME_IR. 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 register IRB_TX_EN_IR bit 0 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 the register IRB_TX_INT_STATUS_IR bit 1 is set.

Before data can be transmitted, the underrun condition must be cleared as in the procedure below.

1. Disable the transmission by writing 0 to register IRB_TX_EN_IR.
2. Load at least one block of data into IRB_TX_SYM_PERIOD_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_EN_IR.

29.2.2 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 microseconds) 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 the registers IRB_RX_SYM_PERIOD 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 the register IRB_RX_INT_STATUS 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_STATUS. No new data is written to the FIFO while this condition exists. To reset the overrun flag, the operations below must be performed:

1. Read at least one word from each of the receive FIFO registers, IRB_RX_SYM_PERIOD and IRB_RX_ON_TIME.
2. Clear the RXOVERRUNSTATUS bit by writing 0x01 to register IRB_RX_CLR_OVERRUN.

The last symbol is detected using a time out condition whose value is stored in microseconds in register IRB_RX_MAX_SYM_PERIOD. If no pulse has been received during this time then the last word in the FIFO IRB_RX_SYM_PERIOD has a value 0xFFFF. If the value of register IRB_RX_INT_EN bit 1 (LASTSYMBOLIRQENABLE bit) is 1, then an interrupt is triggered and the status register IRB_RX_INT_STATUS bit 1 is set. The interrupt and its status bit are cleared automatically when the last value in the FIFO has been read.

When register IRB_RX_INT_EN bit 0 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 register IRB_RX_EN bit 0 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.

29.2.3 Noise suppression filter

This filter is turned off in the IR receiver and is programmable in the UHF receiver using register IRB_RX_NOISE_SUPPRESS_WIDTH_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_SUPPRESS_WIDTH_UHF.

30 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.

Addresses are provided as *IRBBaseAddress* + offset.

The *IRBBaseAddress* is:

0x2001 8000.

Table 75: Infrared transmitter/receiver register summary

Register	Description	Offset		Type
		IR	UHF	
RC transmitter				
IRB_TX_PRE_SCALER_IR	Clock prescaler	0x00	-	R/W
IRB_TX_SUB_CARRIER_IR	Subcarrier frequency programming	0x04	-	R/W
IRB_TX_SYM_PERIOD_IR	Symbol time programming	0x08	-	WO
IRB_TX_ON_TIME_IR	Symbol on time programming	0x0C	-	WO
IRB_TX_INT_EN_IR	Transmit interrupt enable	0x10	-	R/W
IRB_TX_INT_STATUS_IR	Transmit Interrupt status	0x14	-	RO
IRB_TX_EN_IR	RC transmit enable	0x18	-	R/W
IRB_TX_CLR_UNDERRUN_IR	Underrun status reset	0x1C	-	WO
IRB_TX_SUB_CARRIER_IR	Subcarrier frequency programming	0x20	-	R/W
IRB_TX_STATUS_IR	Transmit status	0x24	-	RO
RC receiver				
IRB_RX_ON_TIME	Received pulse time capture	0x40	0x80	R/W
IRB_RX_SYM_PERIOD	Received symbol period capture	0x44	0x84	RO
IRB_RX_INT_EN	Receive interrupt enable	0x48	0x88	R/W
IRB_RX_INT_STATUS	Receive Interrupt status	0x4C	0x8C	RO
IRB_RX_EN	RC receive enable	0x50	0x90	R/W
IRB_RX_MAX_SYM_PERIOD	Maximum RC symbol period	0x54	0x94	R/W
IRB_RX_CLR_OVERRUN	Overrun status reset	0x58	0x98	WO
Noise suppression				
IRB_RX_NOISE_SUPPRESS_WIDTH	Noise suppression width	0x5C	0x9C	R/W
I/O control				
RC_IRDA_CONTROL	RC or IrDA I/O control	0x60		R/W

Table 75: Infrared transmitter/receiver register summary

Register	Description	Offset		Type
		IR	UHF	
Reverse polarity				
IRB_POLINV_REG	Reverse polarity data	0x68	0xA8	R/W
Receive status and clock				
IRB_RX_STATUS	Receive status for IR	0x6C	0xAC	RO
IRB_SAMPLE_RATE_COMM	Sampling frequency division for UHF and IR frequencies	0x64	-	R/W
IRB_CLOCK_SELECT	Clock select configuration	0x70	0x70	R/W
IRB_CLOCK_SELECT_STATUS	Clock select status	0x74	0x74	RO
IrDA Interface				
IRB_BAUD_RATE_GEN_IRDA	Baud rate generation for IR	0xC0	-	R/W
IRB_BAUD_GEN_ENABLE_IRDA	IBaud rate generation enable for IR	0xC4	-	R/W
IRB_TX_ENABLE_IRDA	Transmit enable for IR	0xC8	-	R/W
IRB_RX_ENABLE_IRDA	Receive enable for IR	0xCC	-	R/W
IRB_IRDA_ASC_CONTROL	Asynchronous data control	0xD0	-	R/W
IRB_RX_PULSE_STATUS_IRDA	IReceive pulse status for IR	0xD4	-	RO
IRB_RX_SAMPLING_RATE_IRDA	Receive sampling rate for IR	0xD8	-	WO
RX_MAX_SYMB_TIME_IRDA	Receive maximum symbol time for IR	0xDC	-	WO

30.1 RC transmitter registers

IRB_TX_PRE_SCALER_IR Clock prescaler

7	6	5	4	3	2	1	0
PRESCALEVAL							

Address: *IRBBaseAddress* + 0x00

Type: R/W

Reset: 0

Description: This register selects the value of the prescaler for clock division. The prescaled clock frequency is obtained by dividing the system clock frequency by PRESCALEVAL. It determines the transmit subcarrier resolution, see IRB_TX_SUB_CARRIER_IR.

IRB_TX_SUB_CARRIER_IR Subcarrier frequency programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBCARRIERVAL															

Address: *IRBBaseAddress* + 0x04
Type: R/W
Reset: 0
Description: This register determines the RC transmit subcarrier frequency. The prescaled clock frequency divided by (SUBCARRIERVAL x 2) gives the subcarrier frequency, which has a 50% duty cycle.

IRB_TX_SYM_PERIOD_IR Symbol time programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSYMBOLTIMEVAL															

Address: *IRBBaseAddress* + 0x08
Type: WO
Reset: 0
Description: The value in this register gives the symbol time (symbol period) in periods of the subcarrier clock. It must be programmed sequentially with the register IRB_TX_ON_TIME_IR. This register is quadruple buffered.

IRB_TX_ON_TIME_IR Symbol on time programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXONTIMEVAL															

Address: *IRBBaseAddress* + 0x0C
Type: WO
Reset: 0
Description: The value in this register gives the symbol on time (pulse duration) in periods of the subcarrier clock. This register is quadruple buffered.
Note: *The registers* IRB_TX_SYM_PERIOD_IR *and* IRB_TX_ON_TIME_IR *act as a single register set. They must be programmed sequentially as a pair to latch in the data.*

IRB_TX_INT_EN_IR Transmit interrupt enable

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ONE_WORD	HALF_EMPTY	EMPTY	UNDERRUN	EN_INT

Address: *IRBBaseAddress* + 0x10

Type: R/W

Reset: 0

Description:

[15:5] **Reserved**
Set to logic 0

[4] **ONE_WORD**
1: Interrupt enable on at least one word empty in FIFO

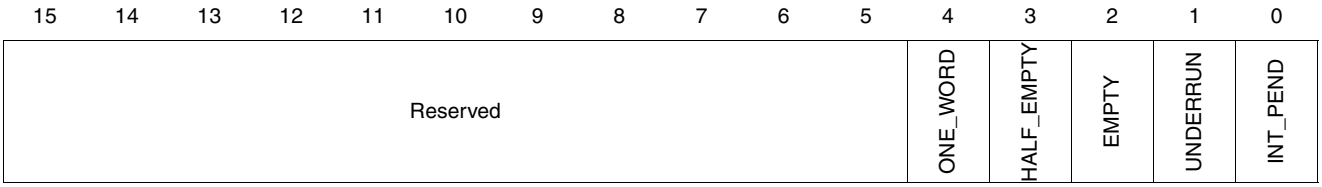
[3] **HALF_EMPTY**
1: Interrupt enable on FIFO half empty

[2] **EMPTY**
1: Interrupt enable on FIFO empty

[1] **UNDERRUN**
1: Enable interrupt on underrun

[0] **EN_INT**: Interrupt enable
1: Global Tx. interrupt enable

IRB_TX_INT_STATUS_IR Transmit Interrupt status



Address: *IRBBaseAddress* + 0x14

Type: RO

Reset: 0

Description: This register is also updated when data is written into the registers IRB_TX_SYM_PERIOD_IR and IRB_TX_IR_ON_TIME_IR.

[15:5] **Reserved**
Set to logic 0

[4] **ONE_WORD**
1: Atleast one word empty interrupt pending

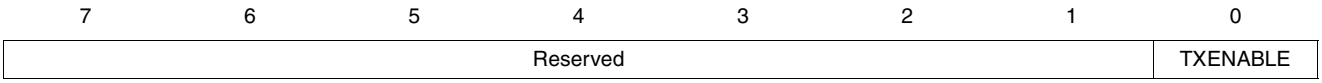
[3] **HALF_EMPTY**
1: FIFO half empty interrupt pending

[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_IR RC transmit enable



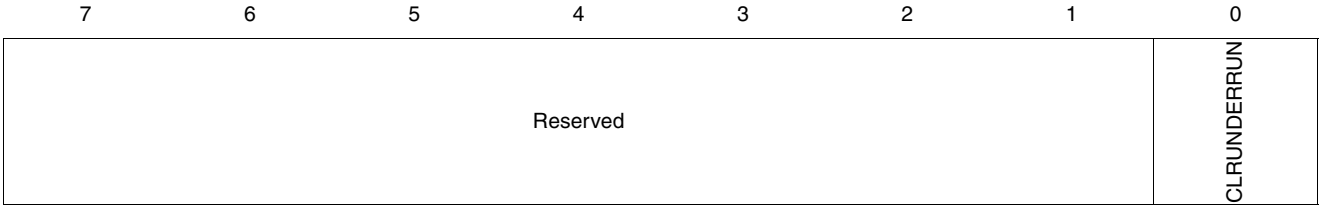
Address: *IRBBaseAddress* + 0x18

Type: R/W

Reset: 0

Description: This register 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_UNDERRUN_IR Underrun status reset



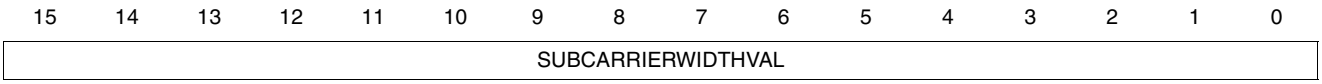
Address: *IRBBaseAddress* + 0x1C

Type: WO

Reset: 0

Description: Set to 1 as part of the procedure for clearing the underrun flag in the register IRB_TX_INT_STATUS_IR. No data is transmitted until this flag has been cleared.

IRB_TX_SUB_CARRIER_WIDTH_IR Subcarrier frequency programming



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_PRE_SCALER_IR register. Software has to ensure that the value written into this register is less than that written in the register IRB_TX_SUB_CARRIER_IR. If the condition is not met, the subcarrier is not generated.

IRB_TX_STATUS_IR Transmit status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TX_FIFO_LEVEL			Reserved			ONE_WORD	HALF_EMPTY	EMPTY	UNDER_RUN	Reserved

Address: *IRBBaseAddress* + 0x24
Type: RO
Reset: 0x1C
Description:

[11:15] **Reserved**

[10:8] **TX_FIFO_LEVEL**
000 = FIFO empty
001 = One block FIFO
010 = Two blocks in FIFO
011 = Three blocks in FIFO
100 = Four blocks in FIFO
101 = Five blocks in FIFO
110 = Six blocks in FIFO
111 = Seven blocks in FIFO (FULL)

[7:5] **Reserved**

[4] **ONE_WORD**
1 - At least one word empty in FIFO

[3] **HALF_EMPTY**
1 - FIFO half empty

[2] **EMPTY**
1 - FIFO empty

[1] **UNDER_RUN**
1 - FIFO underrun

[0] **Reserved**

Clearing an interrupt **does not** clear the corresponding status flag. The status reflects the true transmit status.

30.2 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
RXONTIMEVAL															

Address: *IRBBaseAddress* + 0x40 (IR) *IRBBaseAddress* + 0x80 (UHF)

Type: RO

Reset: 0

Description: The value in this register is the detected duration (in microseconds) of the received RC pulse. It must be read sequentially with register IRB_RX_SYM_PERIOD. The register is quadruple buffered.

IRB_RX_SYM_PERIOD Received symbol period capture

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSYMBOLTIMEVAL															

Address: *IRBBaseAddress* + 0x44 (IR) *IRBBaseAddress* + 0x84 (UHF)

Type: RO

Reset: 0

Description: This register holds the detected time (in micro seconds) between the start of two successive received RC pulses. It is quadruple buffered.

Note: The registers IRB_RX_SYM_PERIOD *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

7 6	5 4	3 2	1	0
Reserved	RXFIFOIRQ[1:0]	Reserved	LASTSYMBOLIRQENABLE	RXIRQENABLE

Address: *IRBBaseAddress* + 0x48 (IR) *IRBBaseAddress* + 0x88 (UHF)
Type: R/W
Reset: 0
Description: To inhibit all these interrupts the RXIRQENABLE bit (register bit 0) must be set to 0.

[7:6] **Reserved:** Set to logic 0

[5:4] **RXFIFOIRQ[1:0]:** Select the receive FIFO fullness interrupt
00: Invalid 01: One word available for read
10: Two words available for read (half full) 11: Three words available for read (FIFO full)

[3:2] **Reserved:** Set to logic 0

[1] **LASTSYMBOLIRQENABLE:** Select interrupt enable/disable on last symbol
1: Generate interrupt on last symbol received

[0] **RXIRQENABLE:** Select the receive interrupt enable/disable
0: Interrupt disable 1: Interrupt enable

IRB_RX_INT_STATUS Receive Interrupt status

7 6	5 4	3	2	1	0
Reserved	RXFIFOSTATUS	Reserved	RXOVERRUNSTATUS	LASTSYMBOLIRQSTATUS	RXIRQSTATUS

Address: *IRBBaseAddress* + 0x4C (IR) *IRBBaseAddress* + 0x8C (UHF)
Type: RO
Reset: 0
Description:

[7:6] **Reserved:** Set to logic 0

[5:4] **RXFIFOSTATUS[1:0]:** Receive FIFO fullness status
00: FIFO empty 01: One word in FIFO
10: Two words in FIFO 11: Three words in FIFO

[3] **Reserved:** Set to logic 0

[2] **RXOVERRUNSTATUS:** Receive overrun status
0: No overrun 1: Overrun occurred

[1] **LASTSYMBOLIRQSTATUS:** Last symbol interrupt status
1: Interrupt active

[0] **RXIRQSTATUS:** Receive interrupt status
1: Interrupt active

IRB_RX_EN RC receive enable

7	6	5	4	3	2	1	0
Reserved							RXENABLE

Address: *IRBBaseAddress* + 0x50 (IR) *IRBBaseAddress* + 0x90 (UHF)
Type: R/W
Reset: 0
Description: When this register is set to 1 the RC receive section is enabled to read incoming data.

IRB_RX_MAX_SYM_PERIOD Maximum RC symbol period

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXMAXSYMBOLTIMEVAL															

Address: *IRBBaseAddress* + 0x54 (IR) *IRBBaseAddress* + 0x94 (UHF)
Type: R/W
Reset: 0
Description: The value in this register 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_OVERRUN Overrun status reset

7	6	5	4	3	2	1	0
Reserved							CLROVERRUN

Address: *IRBBaseAddress* + 0x58 (IR) *IRBBaseAddress* + 0x98 (UHF)
Type: WO
Reset: 0
Description: Set to 1 as part of the procedure for clearing the overrun flag in the register IRB_RX_INT_STATUS. No new data is written into the receive FIFO while this flag is set.

30.3 Noise suppression

IRB_RX_NOISE_SUPPRESS_WIDTH Noise suppression width

7	6	5	4	3	2	1	0
NOISESUPPRESSWIDTH							

Address: *IRBBaseAddress* + 0x5C (IR) *IRBBaseAddress* + 0x9C (UHF)

Type: R/W

Reset: 0

Description: The value, in microseconds, in this register determines the maximum width of noise pulses which the filter suppresses.

30.4 I/O control

RC_IRDA_CONTROL RC or IrDA I/O control

7	6	5	4	3	2	1	0
Reserved							RC_IRDA_CONTROL

Address: *IRBBaseAddress* + 0x60

Type: R/W

Reset: 0

Description: This register controls the data on RC_IRDA_DATA_OUT and RC_IRDA_DATA_IN pins.

1: IrDA data is available on RC_IRDA_DATA_OUT and RC_IRDA_DATA_IN pins.

0: RC data is available on RC_IRDA_DATA_OUT and RC_IRDA_DATA_IN pins.

This register is present only in the receive interface for IR signals.

30.5 Reverse polarity

The two IRB input pins (IRB_IR_IN (PIO5 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_POLINV_REG 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

30.6 Receive status and clock

IRB_RX_STATUS Receive status for IR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RX_FIFO_LEVEL			Reserved		AT_LEAST_1WORD	HALF_FULL	FULL	OVERRUN	LAST_SYM	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 = One block FIFO
010 = Two blocks in FIFO
011 = Three blocks in FIFO
100 = Four blocks in FIFO
101 = Five blocks in FIFO
110 = Six blocks in FIFO
111 = Seven blocks in FIFO (FULL)

[7:6] **Reserved**

[5] **AT_LEAST_1WORD**: At least one word
1: Clears At least one word in FIFO interrupt

[4] **HALF_FULL**
1: Clears FIFO half full Interrupt

[3] **FULL**
1: Clears FIFO full Interrupt

[2] **OVERRUN**
1: Clears FIFO overrun interrupt

[1] **LAST_SYM**: Last symbol
1: Clears Last symbol receive Interrupt

[0] **Reserved**

Clearing the interrupt doesnot clear the status. To clear the status approriate actions (like reading the data from the FIFO) has to be performed.

IRB_SAMPLE_RATE_COMM Sampling frequency division for UHF and IR frequencies

7	6	5	4	3	2	1	0
Reserved				N			

Address: *IRBBaseAddress* + 0x64

Type: R/W

Reset: 0

Description: This register 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 receive portions.

IRB_CLOCK_SELECT Clock select configuration

7	6	5	4	3	2	1	0
Reserved							CLK_SELECT

Address: *IRBBaseAddress* + 0x70

Type: R/W

Reset: 0

Description: This register is used to select if the receive section (both RC and UHF) is clocked by system clock or 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. Clock switching is made by a glitch free mux.

[7:1] **Reserved**: Set to 0

[0] **CLK_SELECT**
0 - System clock
1 - 27 MHz clock

IRB_CLOCK_SELECT_STATUS Clock select status

7	6	5	4	3	2	1	0
Reserved							CLK_STATUS

Address: *IRBBaseAddress* + 0x74

Type: RO

Reset: 0

Description: This register is 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 RX_CLOCK_SELECT_REG, software has to read this register to see if the programmed clock change has happened.

[7:6] **Reserved**

[0] **CLK_STATUS**
0 - System clock
1 - 27Mhz clock

30.7 IrDA Interface

IRB_BAUD_RATE_GEN_IRDA 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 76 below.

Table 76: 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%
34.8k	179.600	180	0x0B4	0.22%
57.6K	108.507	109	0x06D	0.45%
115.2K	54.250	54	0x036	0.46%

IRB_BAUD_GEN_ENABLE_IRDA IBaud rate generation enable for IR

7	6	5	4	3	2	1	0
Reserved							EN_BRG

Address: *IRBBaseAddress* + 0xC4

Type: R/W

Reset: 0

Description: 1: The baud rate generator is enabled.

IRB_TX_ENABLE_IRDA Transmit enable for IR

7	6	5	4	3	2	1	0
Reserved							EN_IRTX

Address: *IRBBaseAddress* + 0xC8

Type: R/W

Reset: 0

Description: 1: The IrDA transmit section is enabled.

IRB_RX_ENABLE_IRDA Receive enable for IR

7	6	5	4	3	2	1	0
Reserved							EN_IRRX

Address: *IRBBaseAddress* + 0xCC
Type: R/W
Reset: 0
Description: 1: The IrDA receive section is enabled.

IRB_IRDA_ASC_CONTROL Asynchronous data control

7	6	5	4	3	2	1	0
Reserved							ASC_CTL

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_RX_PULSE_STATUS_IRDA IReceive pulse status for IR

7	6	5	4	3	2	1	0
Reserved							PSTAT

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_RX_SAMPLING_RATE_IRDA 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.

RX_MAX_SYMB_TIME_IRDA Receive maximum symbol time 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 micro seconds. If an IR pulse grater than this time is detected then the IR pulse is neglected.

31 Synchronous serial controller (SSC)

31.1 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. Four SSCs are provided on the STi7710. 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 31.2.1: *Pin connection and control on page 250*). 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 STi7710, 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) or used separately for SPI full duplex mode using SPI_INPUT_SEL (bits 25 and 26 in the CONFIG_CONTROL_E register).

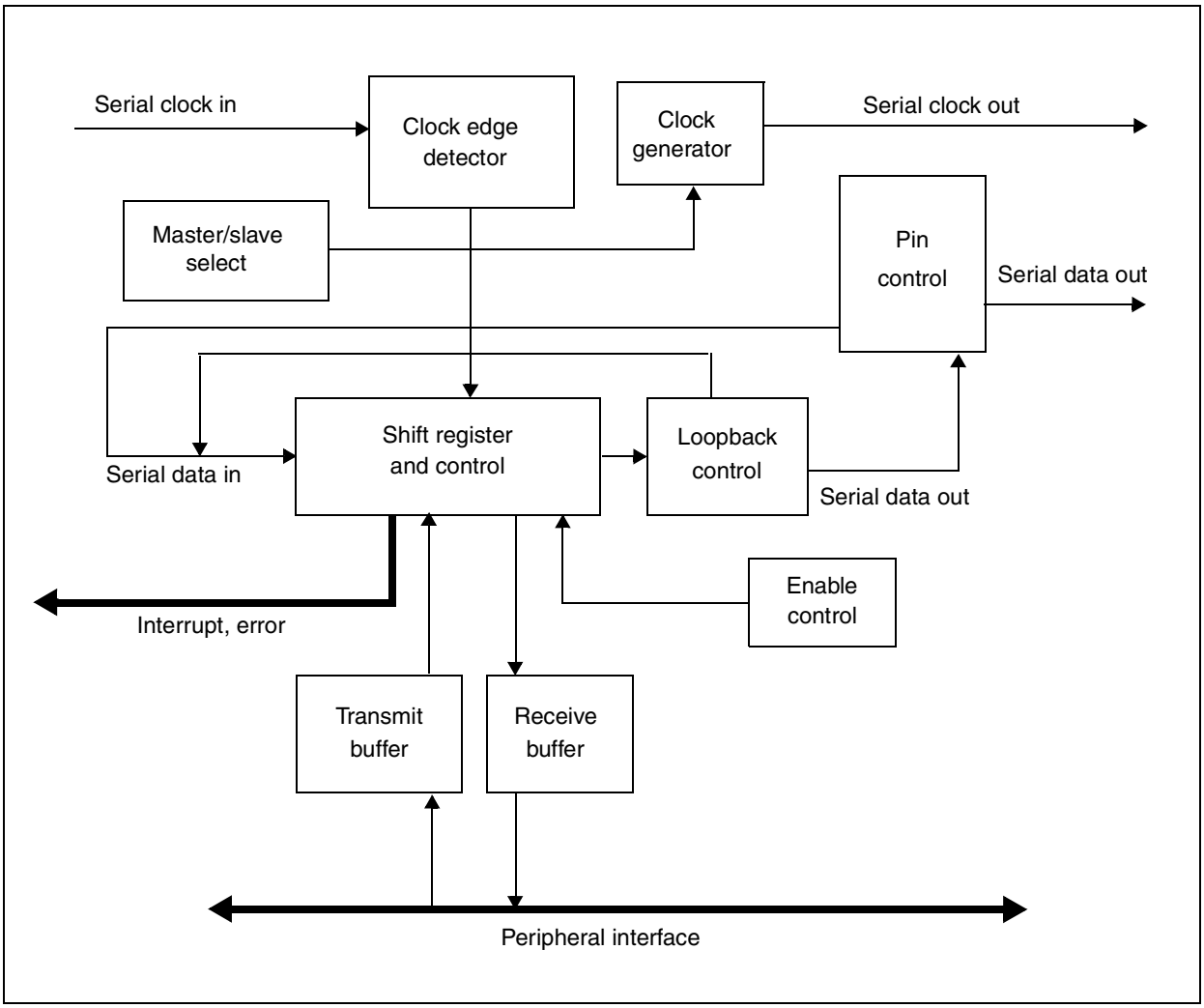
31.2 Basic operation

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 31.2.2: *Clock generation on page 251*.

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 31.2.4: *Shift register on page 253*.

Figure 44: 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 31.2.7: *Transmit and receive buffers on page 255*. 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 STi7710, 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) or used separately for SPI full duplex mode using SPI_INPUT_SEL (bits 25 and 26 in the CONFIG_CONTROL_E register).

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 31.2.8: *Loopback mode* *on page 255*.

The SSC can be turned on and off by setting the enable control. This is described in Section 31.2.9: *Enabling operation* *on page 255*. It can be also be set to operate as a bus master or as a bus slave device. This is described in Section 31.2.10: *Master/slave operation* *on page 255*.

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 31.2.11: *Error detection* *on page 256*.

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 31.3: *I2C operation* *on page 258*.

31.2.1 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].

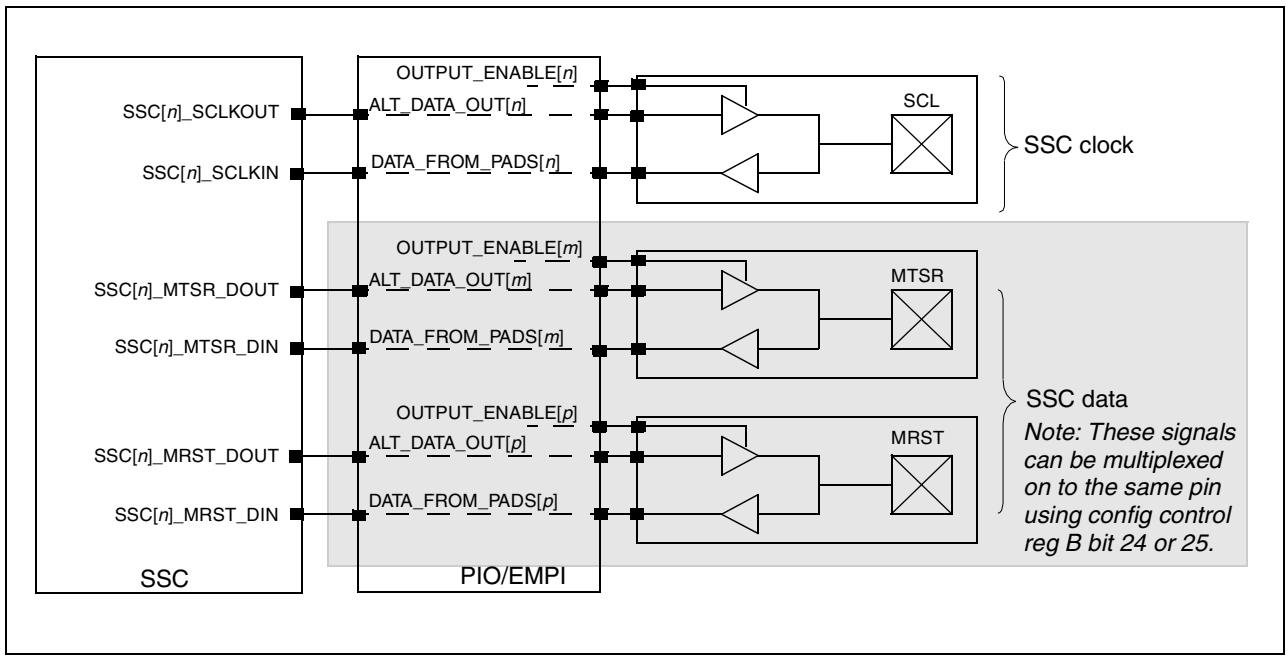
In I^2C mode only, the MTSR pin is used as an input and output. This means only the MTSR pad needs to be used on the I^2C data line. However, for backward compatibility, it is still possible to short MTSR and MRST data pins externally and achieve the same function (the MRST data output is permanently 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 STi7710, 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 or used separately for SPI full duplex mode using SPI_INPUT_SEL (bits 25 and 26 in the CONFIG_CONTROL_E register).

The connections between the SSC ports and the relevant PIO pins are illustrated in Figure 45. 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 45: 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.

31.2.2 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 31.3: *I2C operation on page 258*.

Clock control

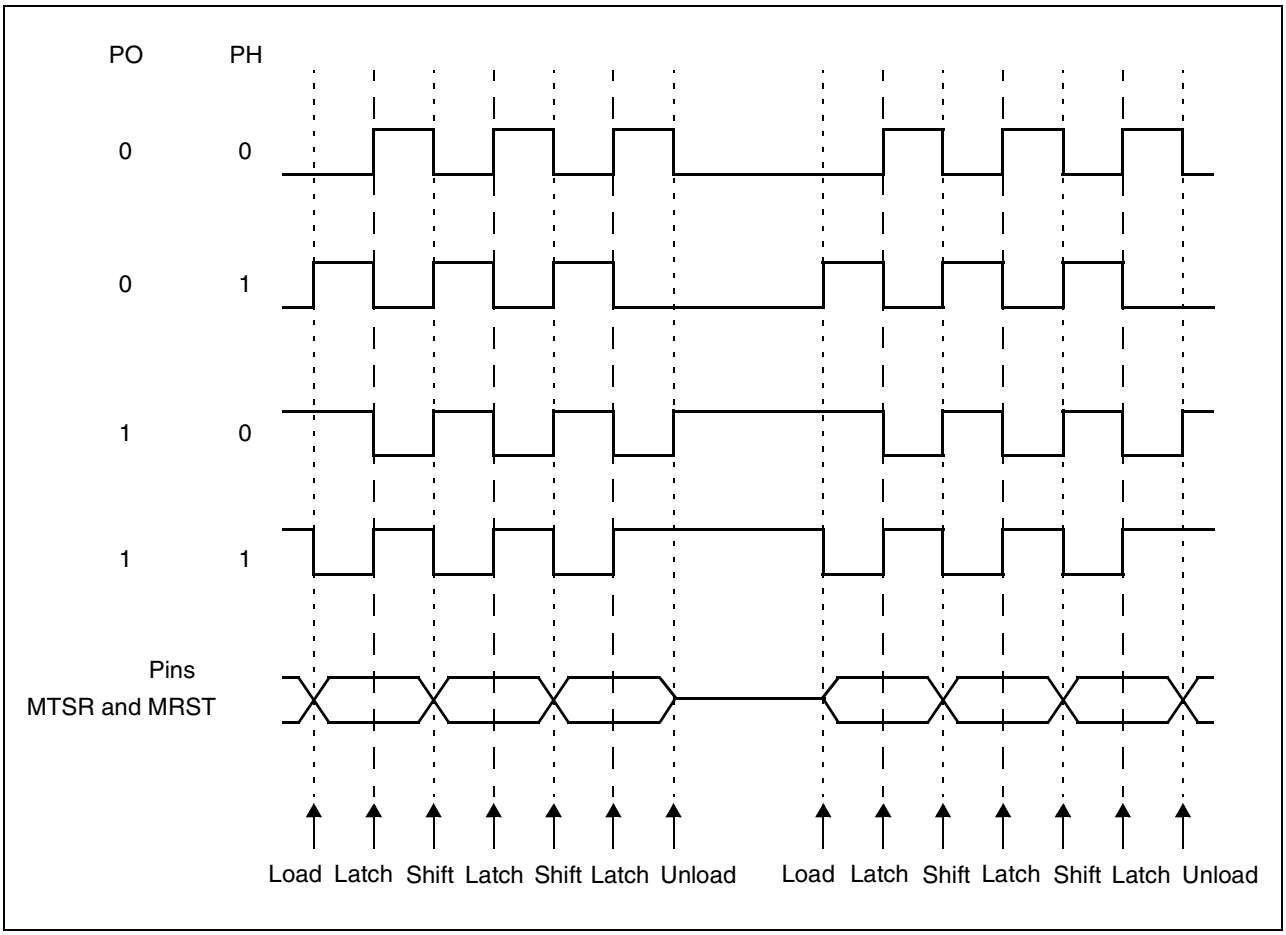
In master mode, the serial clock SCLK, is generated by the SSC according to the setting of the phase bit PH and polarity bit PO in the control register SSC*n*CON.

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 46.

Figure 46: 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.

31.2.3 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 77.

Table 77: 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.

31.2.4 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.

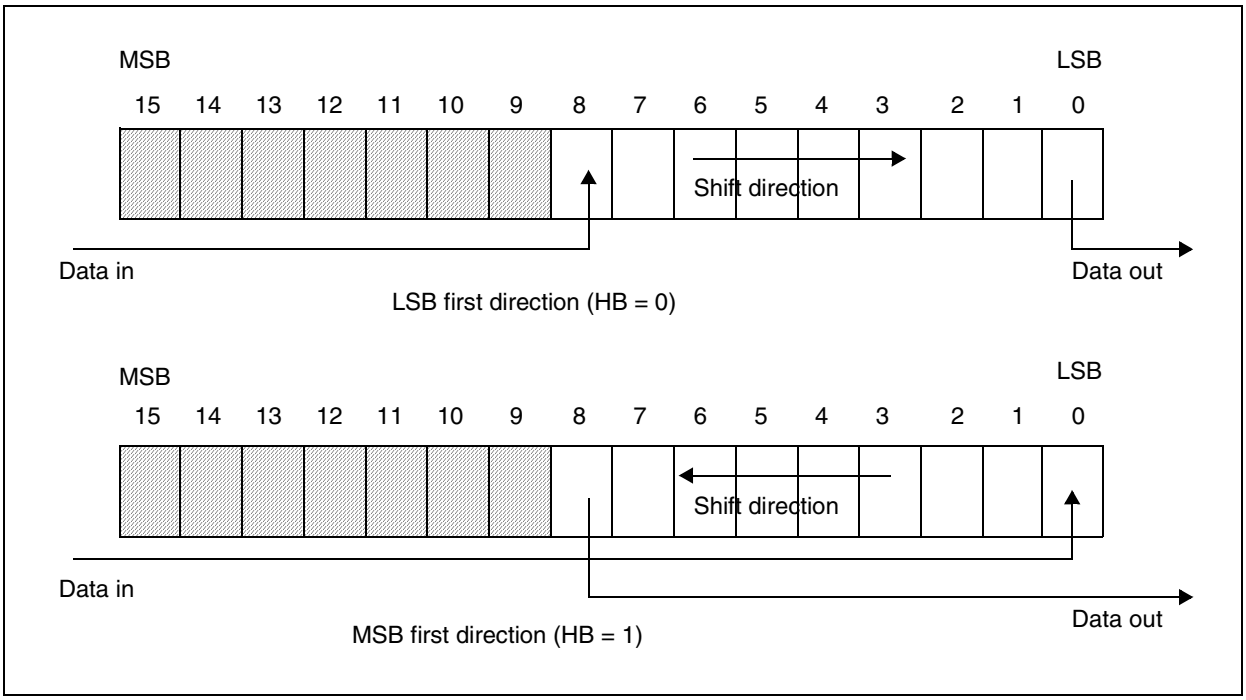
The width of a data frame is also programmable from 2 to 16 bits. This is set by the BM bit field of the control register SSCnCON. A value of 0000 is not allowed. Subsequent values set the bit width to the value plus one; for example 0001 sets the frame width to 2 bits and 1111 sets it to 16 bits.

Note: For I^2C the BM bit in SSCn_CON 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 47.

Figure 47: 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 31.2.2: *Clock generation on page 251*). 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 31.3: *I2C operation on page 258*.

31.2.5 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.

31.2.6 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, NOISE_SUPPRESS_WIDTH_SSC and PRE_SCALER_SSC.

NOISE_SUPPRESS_WIDTH_SSC 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 into this register 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.

31.2.7 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 31.2.11: *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 31.2.11: *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.

31.2.8 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.

31.2.9 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.

31.2.10 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.

31.2.11 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.

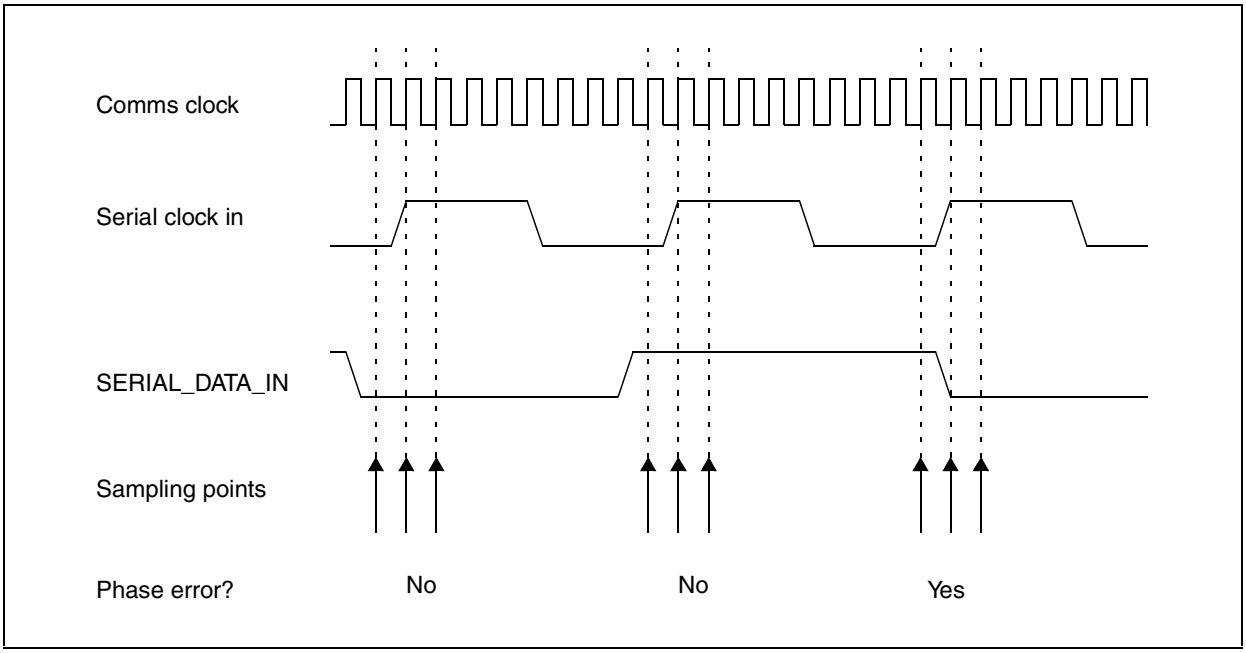
The receive error status bit (and the interrupt, if enabled) is cleared by the next read from the receive buffer.

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 48.

Figure 48: 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.

31.2.12 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.

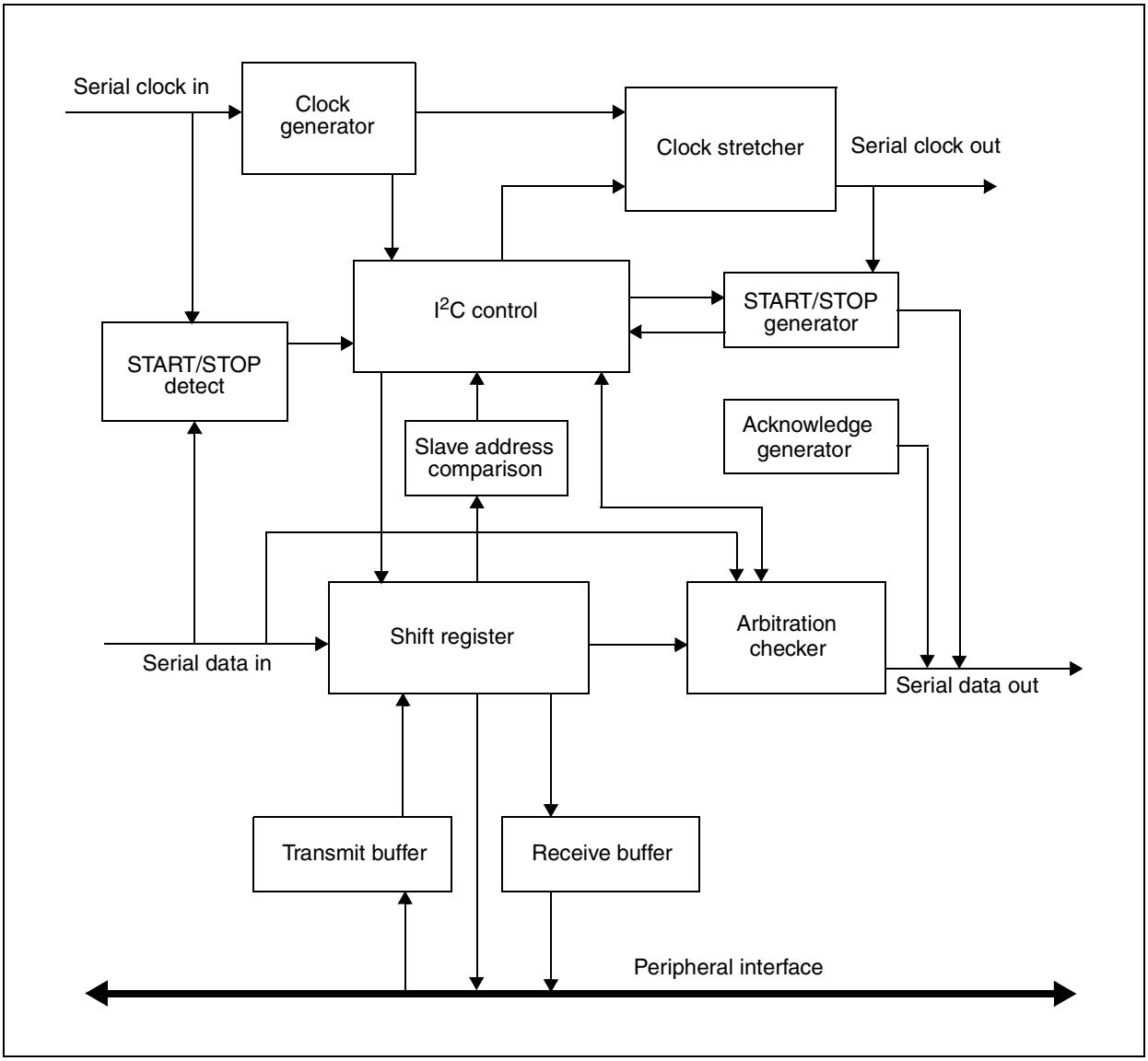
To determine which interrupt condition has occurred, a status register SSCnSTAT is provided which includes a bit for each condition. This is independent of the interrupt enables register SSCnIEN, and determines whether the condition asserts one or more of the interrupt signals.

31.3 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 49.

Figure 49: I²C architecture



31.3.1 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 31.3.2: *Clock synchronization on page 260*). 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 31.3.4: *Slave address comparison on page 262*). 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 31.3.4: *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 31.3.5: *Clock stretching on page 263*).

To generate a START condition, the I²C START condition generate bit STRTG in register SSCnI2C, must be set (see Section 31.3.6: *START/STOP condition generation on page 263*). To generate a STOP condition, the I²C STOP condition generate bit STOPG, must be set (see Section 31.3.6: *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 31.3.7: *Acknowledge bit generation on page 264*).

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 31.3.5: *Clock stretching on page 263*). 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 31.3.5: *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.

31.3.2 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.

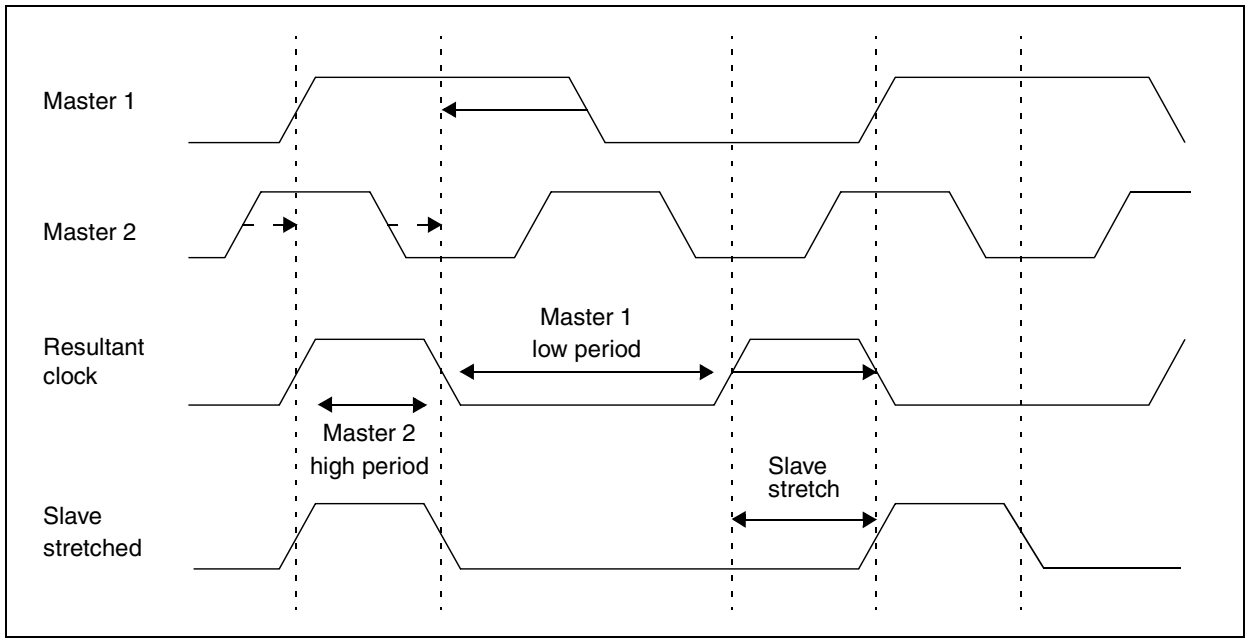
Once the external clock signal goes high, all the master devices begin counting off their high clock pulse. In this case the first master to finish counting attempts to pull the external clock line low and wins (because of the open drain line). The other master devices detect this and abort their high 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 50.

Figure 50: Synchronization and stretching



The SSC implements this clock synchronization mechanism when the I^2C control bit I2CM, is enabled.

31.3.3 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 31.3.8: *Arbitration checking on page 264*).

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.

31.3.4 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.

31.3.5 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 31.3.2: *Clock synchronization on page 260*). 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.

31.3.6 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.

31.3.7 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.

31.3.8 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.

32 Synchronous serial controller (SSC) registers

Addresses are provided as *SSCnBaseAddress* + offset.

The *SSCnBaseAddresses* are:

SSC0: 0x2004 0000,
SSC1: 0x2004 1000,
SSC2: 0x2004 2000,
SSC3: 0x2004 3000.

Table 78: SSC register summary

Register	Description	Offset	Type
SSCnBRG	SSCn baudrate generation	0x000	R/W
SSCnTBUF	SSC n transmit buffer	0x004	WO
SSCnRBUF	SSC n receive buffer	0x008	RO
SSCnCON	SSCn control	0x00C	R/W
SSCnIEN	SSCn interrupt enable	0x010	R/W
SSCnSTAT	SSC n status	0x014	RO
SSCnI2C	SSCn I2C control	0x018	R/W
SSCnSLAD	SSC n slave address	0x01C	WO
CLEAR_STATUS_SSC	SSC clear bit operation	0x080	R/W
NOISE_SUPPRESS_WIDTH_SSC	Noise suppression width	0x100	R/W
PRE_SCALER_SSC	Clock prescaler	0x104	R/W

SSCnBRG SSCn baudrate generation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRG															

Address: *SSCnBaseAddress* + 0x00

Type: R/W

Reset: 1

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.

SSCnTBUF SSC n transmit buffer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															

Address: *SSCnBaseAddress* + 0x04
Type: WO
Reset: 0
Description: Transmit buffer data TD15 to TD0.

SSCnRBUF SSC n receive buffer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD[15:0]															

Address: *SSCnBaseAddress* + 0x08
Type: RO
Reset: 0
Description: Receive buffer data RD15 to RD0.

SSCnCON SSCn control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					LPB	EN	MS	SR	PO	PH	HB	BM			

Address: *SSCnBaseAddress* + 0x0C
Type: R/W
Reset: 0
Description:

[15:11] **Reserved**

[10] **LPB:** SSC loopback bit
0: Disabled, 1: Shift register output is connected to shift register input

[9] **EN:** SSC enable bit
0: Transmission and reception disabled, 1: Transmission and reception enabled

[8] **MS:** SSC master select bit
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 bit
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 bit
0: Pulse in second half cycle 1: Pulse in first half cycle
Must be set in I^2C mode.

[4] **HB:** SSC heading control bit
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

SSCnIEN SSCn interrupt enable

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				REPSTRTEN	NACKEN	Reserved	ARBLEN	STOPEN	AASEN	Reserved	PEEN	REEN	TEEN	TIEN	RIEN

Address: *SSCnBaseAddress* + 0x10

Type: R/W

Reset: 0

Description: This register holds the interrupt enable bits, which can be used to mask the interrupts.

[15:12] **Reserved**

[11] **REPSTRTEN:** I^2C repeated start condition interrupt enable
1: Repeated condition interrupt enabled

[10] **NACKEN:** I^2C NACK condition interrupt enable
1: NACK condition interrupt enabled

[9] **Reserved**

[8] **ARBLEN:** I^2C arbitration lost interrupt enable
1: Arbitration lost interrupt enabled

[7] **STOPEN:** I^2C stop condition interrupt enable
1: Stop condition interrupt enabled

[6] **AASEN:** I^2C addressed as slave interrupt enable
1: Addressed as slave interrupt enabled

[5] **Reserved**

[4] **PEEN:** Phase error interrupt enable
1: Phase error interrupt enabled

[3] **REEN:** Receive error interrupt enable
1: Receive error interrupt enabled

[2] **TEEN:** Transmit error interrupt enable
1: Transmit error interrupt enabled

[1] **TIEN:** Transmitter buffer empty interrupt enable
1: Transmitter buffer empty interrupt enabled

[0] **RIEN:** Receiver buffer full interrupt enable
1: Receiver buffer interrupt enabled

SSCnSTAT SSC n 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* + 0x14

Type: RO

Reset: 2, that is, 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

SSCnI2C SSCn I²C control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			I2CFSMODE	REPSTRTG	Reserved					TXENB	AD10	ACKG	STOPG	STRTG	I2CM

Address: *SSCnBaseAddress* + 0x18

Type: R/W

Reset: 0

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] **TXENB:** 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 bit
0: Disabled 1: Enable I²C features

SSCnSLAD SSC n 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* + 0x1C

Type: WO

Reset: 0

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].

CLEAR_STATUS_SSC SSC clear bit operation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CIR_REPSTRT	CIR_NACK	Reserved	CIR_SSCARBL	CIR_SSCSTOP	CIR_SSCAAS	Reserved					

Address: *SSCnBaseAddress* + 0x80

Type: R/W

Reset: 0

Description:

[15:12] **Reserved**

[11] **CIR_REPSTRT**
1: To clear the SSCREPSTRT in SSCnSTAT

[10] **CIR_NACK**
1: To clear the SCCNACK in SSCnSTAT

[9] **Reserved**

[8] **CIR_SSCARBL**
1: To clear SSCARBL

[7] **CIR_SSCSTOP**
1: To clear SSCSTOP

[6] **CIR_SSCAAS**
1: To clear SSCAAS

[5:0] **Reserved**

NOISE_SUPPRESS_WIDTH_SSC Noise suppression width

7	6	5	4	3	2	1	0
NOISESUPPRESSWIDTH							

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 NOISE_SUPPRESS_WIDTH_SSC are suppressed. Writing 0x00 into this register bypasses the antiglitch filter.

PRE_SCALER_SSC Clock prescaler

7	6	5	4	3	2	1	0
Reserved				PRESCALEVAL			

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.

33 Asynchronous serial controller (ASC)

33.1 Overview

The asynchronous serial controller, also referred to as the UART interface, provides serial communication between the STi7710 and other microcontrollers, microprocessors or external peripherals. The STi7710 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.

33.2 Control

The ASC_n_CONTROL register 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.

33.2.1 Resetting the FIFOs

The 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.

33.2.2 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 the ASC_n_CONTROL register), 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 (RXENABLE) in the ASC_n_CONTROL register. 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 (OVERRUNERROR) in the status register, ASC_n_STATUS, 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_STATUS register is updated to reflect the reception of the new character.

The loop back option (selected by the LOOPBACK bit in the ASC_n_CONTROL register) 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.

33.3 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).

33.3.1 8-bit data frames

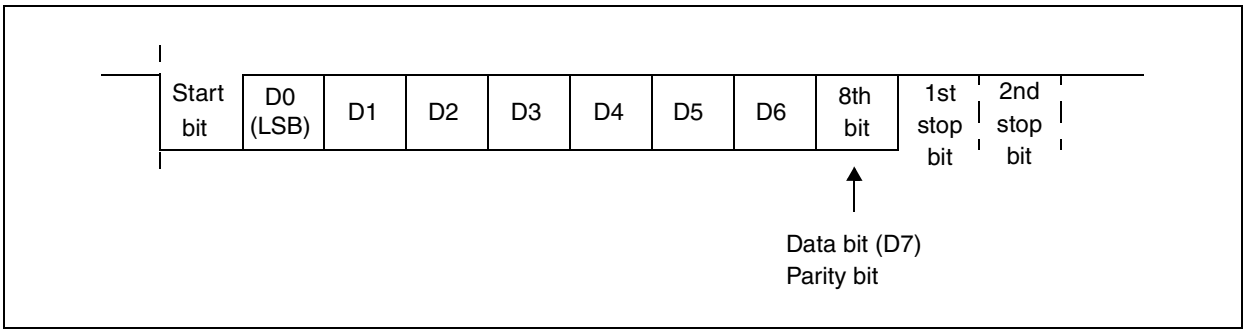
Figure 51 illustrates an 8-bit transmitted data frame. 8-bit frames may use one of the following formats:

- 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 PARITYODD bit in the ASC_n_CONTROL register. 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 51: 8-bit Tx data frame format

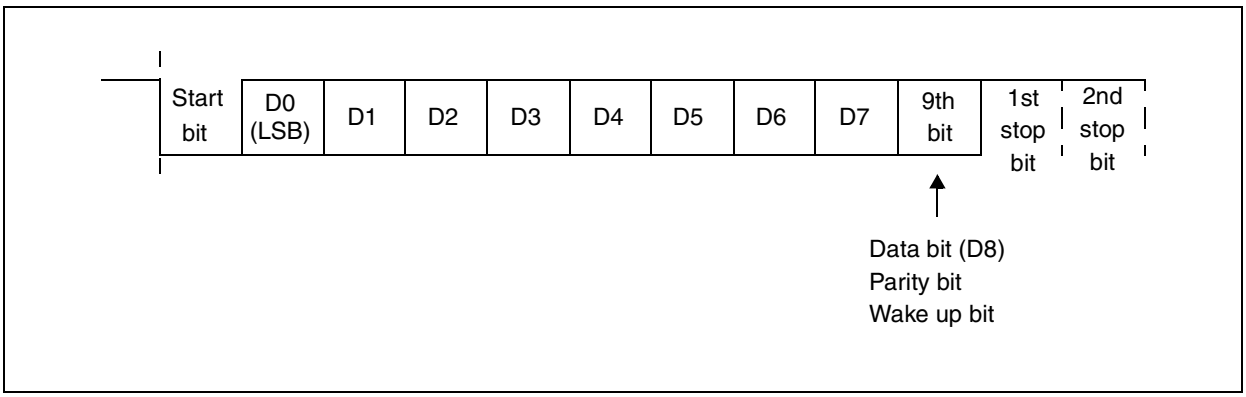


33.3.2 9-bit data frames

Figure 52 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 52: 9-bit Tx data frame format



Parity may be odd or even, depending on the PARITYODD bit in the ASC_n_CONTROL register. 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.

33.4 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_TXBUFFER. If the CTSENABLE bit is set in the ASC_n_CONTROL register 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 the LOOPBACK bit of the ASC_n_CONTROL register) 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 the SCENABLE bit in the ASC_n_CONTROL register is 0, the length of these stop bits is determined by the setting of the STOPBITS field of the ASC_n_CONTROL register. This can either be for 0.5, 1, 1.5 or 2 baud clock periods. In smartcard mode, when the SCENABLE bit in the ASC_n_CONTROL register is 1, the number of stop bits is determined by the value in ASC_n_ GUARDTIME.

33.4.1 Transmission with FIFOs enabled

The FIFOs are enabled by setting the FIFOENABLE bit of the ASC_n_CONTROL register. 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_TXBUFFER.

The TXFULL bit of the ASC_n_STATUS register 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. The TXHALFEMPTY bit of the ASC_n_STATUS register 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 FIFOENABLE bit, it is important to reset the FIFO to empty (by writing to the ASC_n_TXRESET register), or garbage may be transmitted.

33.4.2 Double buffered transmission

Double buffering is enabled and the FIFOs disabled by writing 0 to the FIFOENABLE bit of the ASC_n_CONTROL register. When the transmitter is idle, the transmit data written into the transmit buffer ASC_n_TXBUFFER 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 (TXHALFEMPTY) 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 TXFULL bit is set when the buffer contains 1 character, and a write to ASC_n_TXBUFFER in this situation overwrites the contents. The TXHALFEMPTY bit of the ASC_n_STATUS register is set when the output buffer is empty.

33.5 Reception

Reception is initiated by a falling edge on the data input pin RXD, provided that the RUN and RXENABLE bits of the ASC_n_CONTROL 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 the RXENABLE bit of ASC_n_CONTROL. Any currently received frame is completed including the generation of the receive status flags. Start bits that follow this frame are not recognized.

33.5.1 Hardware error detection

To improve the safety of serial data exchange, the ASC provides three error status flags in the ASC_n_STATUS 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_STATUS 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_STATUS register is set when the RXD pin is not 1 during the programmed number of stop bit times (see Section 33.5).
  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_STATUS 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). The FRAMEERROR bit of the ASC_n_STATUS register 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_STATUS 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 33.3.1 on page 272), the ninth bit of each input entry (bit 8 of 0 to 9) is undefined.

33.5.2 Input buffering modes

FIFO enabled reception

The FIFOs are enabled by setting the FIFOENABLE bit of the ASC_n_CONTROL register. 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, the RXBUFFULL bit of the ASC_n_STATUS register is set to 0. If one or more FIFO entries are present, the RXBUFFULL bit of the ASC_n_STATUS register is set to 1. If the input FIFO is not empty, a read from ASC_n_RXBUFFER gets the oldest entry in the input FIFO.

The RXHALFFULL bit of the ASC_n_STATUS register is set when the input FIFO contains more than eight characters. Writing anything to ASC_n_RXRESET 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.

The OVERRUNERROR bit of the ASC_n_STATUS register 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_RXBUFFER register is read.

After changing the FIFOENABLE 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 the FIFOENABLE bit of the ASC_n_CONTROL register. 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_RXBUFFER). The receive buffer full flag (RXBUFFULL) is set, and the parity error (PARITYERROR) and framing error (FRAMEERROR) 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.

33.5.3 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_STATUS register is 1 when the input FIFO is not empty and the time out counter is zero.

The TIMEOUTIDLE bit of the ASC_n_STATUS 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_STATUS 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_STATUS register is set.

33.6 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 the RUN bit of the ASC_n_CONTROL register is 1, then any value written in the ASC_n_BAUDRATE register 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 the BAUDMODE bit in the ASC_n_CONTROL register. 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.

33.6.1 Baudrates

The baudrate generator provides an internal oversampling clock at 16 times the external baudrate. This clock only ticks if the RUN bit of the ASC_n_CONTROL register is set to 1. Setting this bit to 0 immediately freezes the state of the ASC's transmitter and receiver.

Mode 0

When the BAUDMODE bit in the ASC_n_CONTROL register 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_CLOCK[2], see Chapter 7: *Clocks on page 64*).

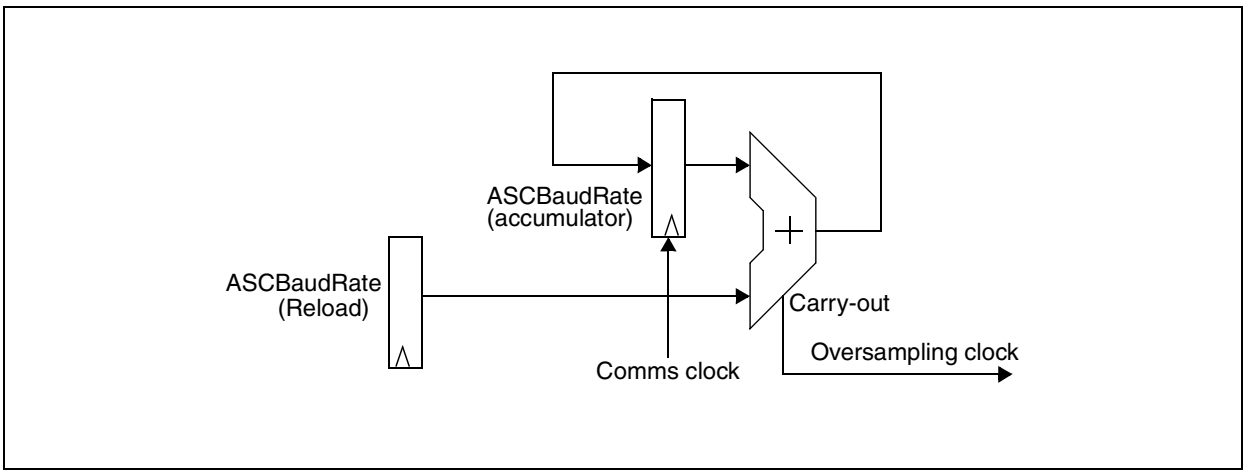
The baudrate counter is clocked by the comms clock. It counts downwards and can be started or stopped by the RUN bit in the ASC_n_CONTROL register. Each underflow of the timer provides one oversampling baudrate clock pulse. The counter is reloaded with the value stored in its 16-bit reload register each time it underflows.

Writes to the ASC_n_BAUDRATE register update the reload register value. Reads from the ASC_n_BAUDRATE register return the current value of the counter.

Mode 1

When the BAUDMODE bit in the ASC_n_CONTROL register is set to 1, the baudrate is controlled by the circuit in Figure 53.

Figure 53: 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_CLOCK[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.

33.7 Interrupt control

Each ASC contains two registers that are used to control interrupts, the status register (ASC_n_STATUS) and the interrupt enable register (ASC_n_INTENABLE). The status bits in the ASC_n_STATUS 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_INTENABLE 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_INTENABLE register, as shown in Figure 54.

The status bits cannot be reset by software because the ASC_n_STATUS register cannot be written to directly. Status bits are reset by operations performed by the interrupt handler:

- transmitter interrupt status bits (TXEMPTY and TXHALFEMPTY) are reset when a character is written to the transmitter buffer,
- receiver interrupt status bit (RXBUFFULL) is reset when a character is read from the receive buffer,
- PARITYERROR and FRAMEERROR status bits are reset when all characters containing errors have been read from the receive input buffer,
- The OVERRUNERROR status bit is reset when a character is read from ASC_n_RXBUFFER.

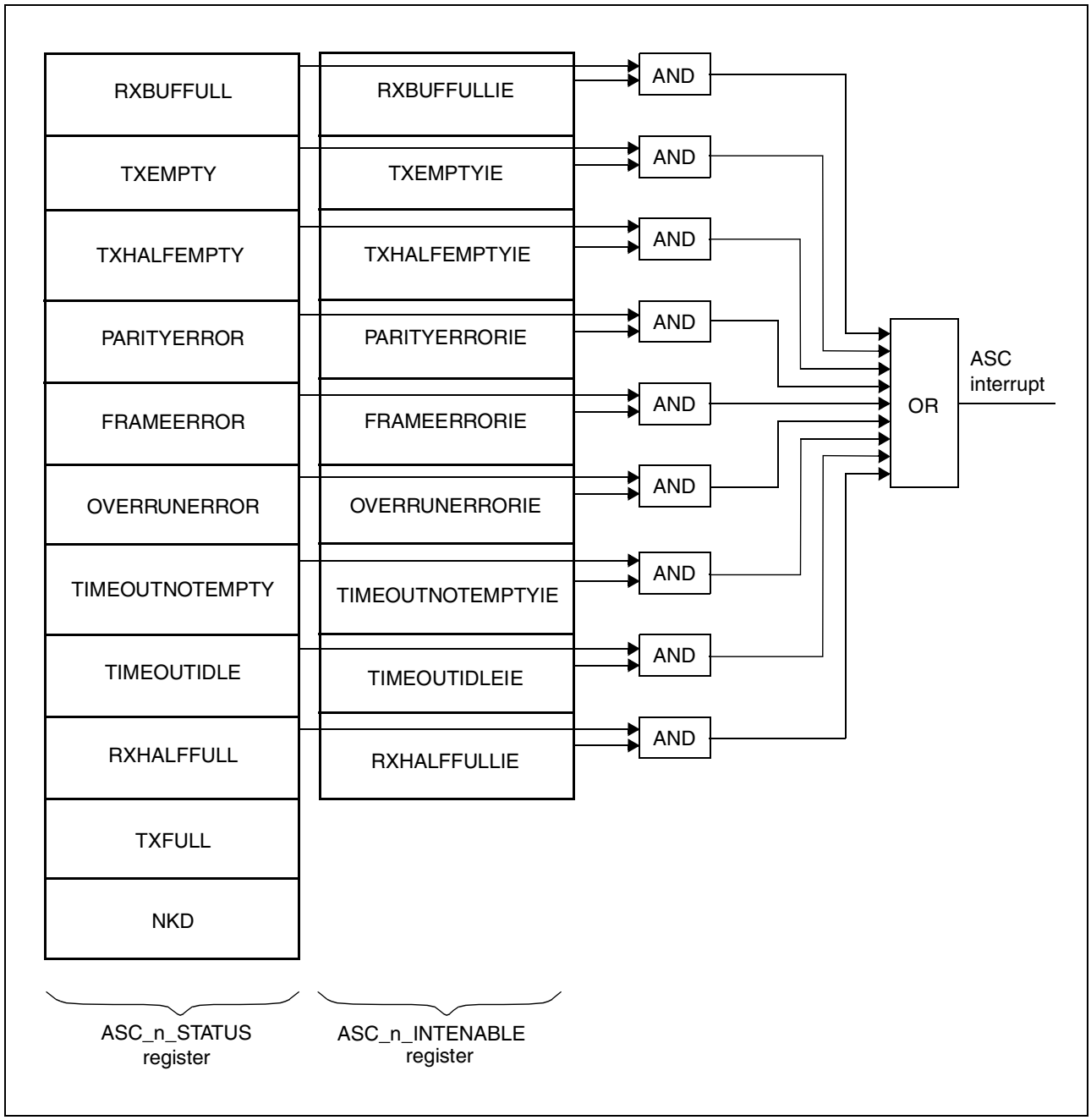
33.7.1 Using the ASC interrupts when FIFOs are disabled (double buffered operation)

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:

- TXHALFEMPTY is activated when data is moved from ASC_n_TXBUFFER to the transmit shift register,
- TXEMPTY is activated before the last bit of a frame is transmitted,

- RXBUFFULL is activated when the received frame is moved to ASC_n_RXBUFFER.

Figure 54: ASC status and interrupt registers



As shown in Figure 54, TXHALFEMPTY is an early trigger for the reload routine, while TXEMPTY indicates the completed transmission of the data field of the frame. Therefore, software using handshake should rely on TXEMPTY at the end of a data block to make sure that all data has really been transmitted.

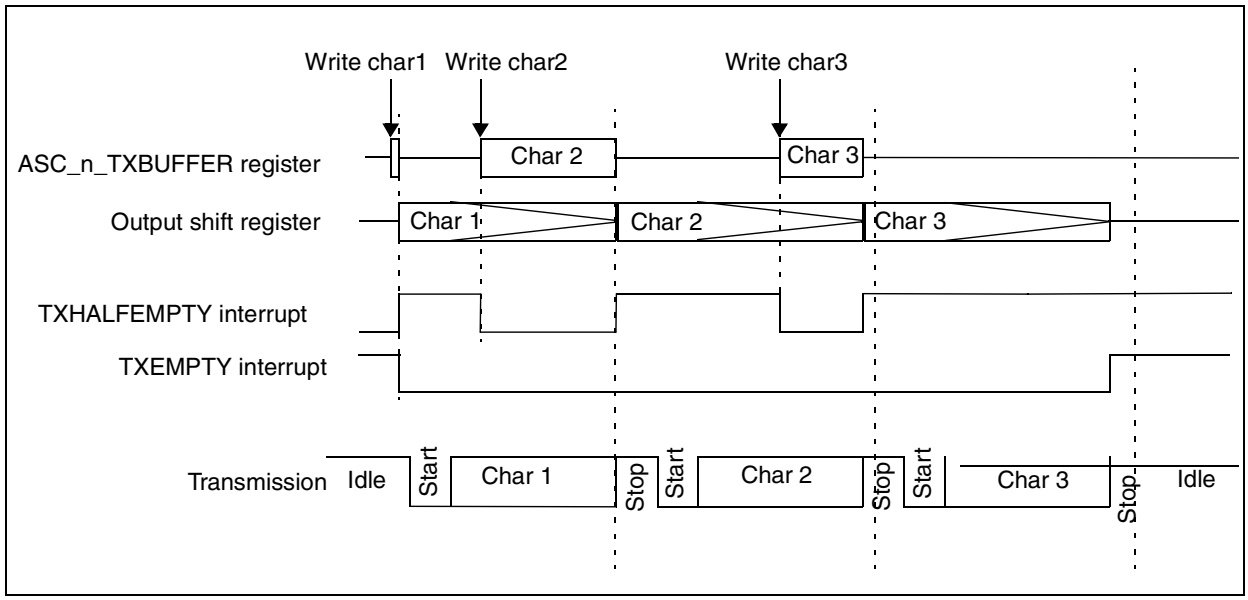
For single transfers it is sufficient to use the transmitter interrupt (TXEMPTY), which indicates that the previously loaded data has been transmitted, except for the last bit of a frame.

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 TXEMPTY 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 (TXHALFEMPTY) to signal for more data allows the service routine to load a complete frame, as ASC_n_TXBUFFER may be reloaded while the previous data is still being transmitted.

33.7.2 Using the ASC interrupts when FIFOs are enabled

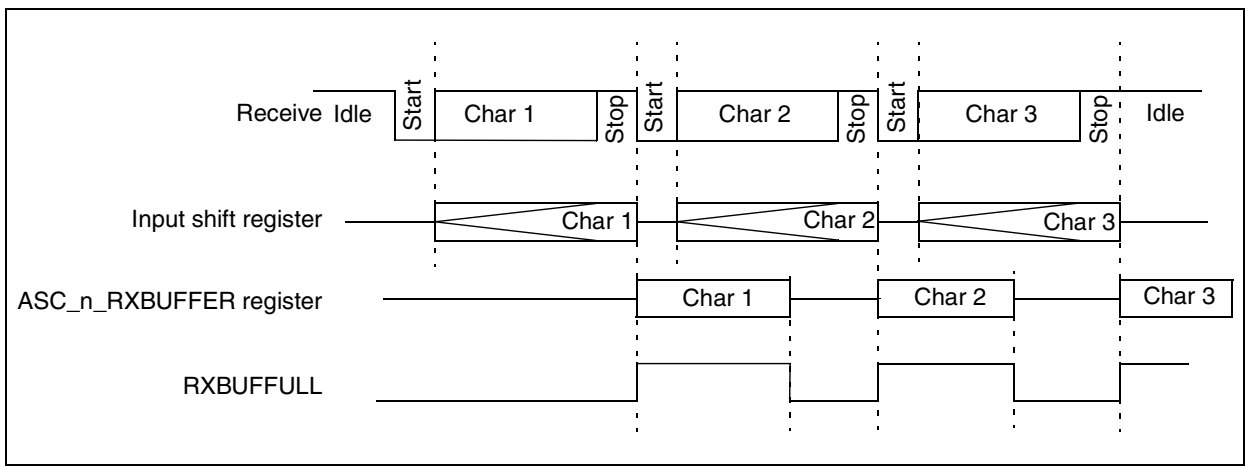
To transmit a large number of characters back to back, the driver routine initially writes 16 characters to ASC_n_TXBUFFER. Then every time a TXHALFEMPTY interrupt fires, it writes eight more. When there is nothing more to send, a TXEMPTY interrupt tells the driver that everything has been transmitted.

Figure 55: 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 56: ASC reception



33.8 Smartcard operation

Smartcard mode is selected by setting the SCENABLE bit in the ASC_n_CONTROL register 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 33.3 on page 272). 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 the SCENABLE bit in the ASC_n_CONTROL register is set to 0, normal UART operation occurs.

Smartcard operation complies with the ISO smartcard specification except where noted (see Section 33.8.4).

33.8.1 Control registers

ASC_n_GUARDTIME

A 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.

33.8.2 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_STATUS register.

Emptying the FIFO causes an interrupt, which can be handled by software. The NKD bit in the ASC_n_STATUS register can be reset by writing to the ASC_n_TXRESET 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 TXEMPTY interrupt can be delayed by programming the ASC_n_GUARDTIME register. In normal operation, TXEMPTY is asserted when the transmit shift register is empty and no further transmit requests are outstanding.
- The receiver enable bit in the ASC_n_CONTROL 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. TXEMPTY is forced low during this time. When the guardtime counter reaches the programmed value TXEMPTY is asserted high.

The de-assertion of TXEMPTY is unaffected by smartcard mode.

33.8.3 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 the NACKDISABLE bit in ASC_n_CONTROL 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 the NACKDISABLE bit in ASC_n_CONTROL is 1 then no automatic NACK generation takes place.

33.8.4 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.

34 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: 0x2003 0000,
UART1: 0x2003 1000,
UART2: 0x2003 2000,
UART3: 0x2003 3000.

There is also one enable register located in the infrared blaster block. This is provided as *IRBBaseAddress* + offset.

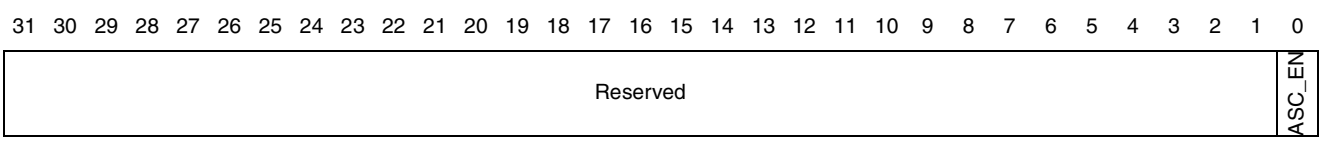
The *IRBBaseAddress* is:

0x2011 5000.

Table 79: ASC register summary

Register	Description	Offset	Type
ASC_ENABLE	ASC I/O enable	0xDO	R/W
ASC_n_BAUDRATE	ASCn baudrate generator	0x00	R/W
ASC_n_CONTROL	ASCn control	0x0C	R/W
ASC_n_GUARDTIME	ASCn guard time	0x18	R/W
ASC_n_INTENABLE	ASCn interrupt enable	0x10	R/W
ASC_n_RETRIES	ASCn number of retries on transmission	0x28	R/W
ASC_n_RXBUFFER	ASCn receive buffer	0x08	RO
ASC_n_RXRESET	ASCn receive FIFO reset	0x24	WO
ASC_n_STATUS	ASCn interrupt status	0x14	RO
ASC_n_TIMEOUT	ASCn time out	0x1C	R/W
ASC_n_TXBUFFER	ASCn transmit buffer	0x04	WO
ASC_n_TXRESET	ASCn transmit FIFO reset	0x20	WO

ASC_ENABLE ASC I/O enable



Address: *IRBBaseAddress* + 0x0D0

Type: R/W

Reset: 0

Description: If bit 0 (ASC_EN) is set to 1, asynchronous data is available on the ASC2_TXD and ASC2_RXD pins.

Note: This register must be set to 1 to enable the ASC block.

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																RELOADVAL															

Address: *ASCnBaseAddress* + 0x00

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 the RUN bit of the ASC_n_CONTROL register 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 the BAUDMODE bit in the ASC_n_CONTROL register.

Mode 0

When the BAUDMODE bit in the ASC_n_CONTROL register 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_CLOCK[2]).

Mode 0 should be used for all baudrates below 19.2 Kbaud.

Table 80 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 80: 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 the BAUDMODE bit in the ASC_n_CONTROL register 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 81: 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_CONTROL 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																		NACKDISABLE	BAUDMODE	CTSENABLE	FIFOENABLE	SCENABLE	RXENABLE	RUN	LOOPBACK	PARITYODD	STOPBITS		MODE		

Address: *ASCnBaseAddress* + 0x0C

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] **NACKDISABLE:** 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] **CTSENABLE:** CTS enable bit

0: CTS ignored 1: CTS enabled

[10] **FIFOENABLE:** FIFO enable bit:

0: FIFO disabled 1: FIFO enabled

[9] **SCENABLE:** Smartcard enable bit

0: Smartcard mode disabled 1: Smartcard mode enabled

[8] **RXENABLE:** 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_GUARDTIME ASCn guard time

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																							GUARDTIME								

Address: *ASCnBaseAddress* + 0x18

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_INTENABLE ASCn 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																							RHF	TOI	TNE	OE	FE	PE	THE	TE	RBE

Address: *ASCnBaseAddress* + 0x10
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

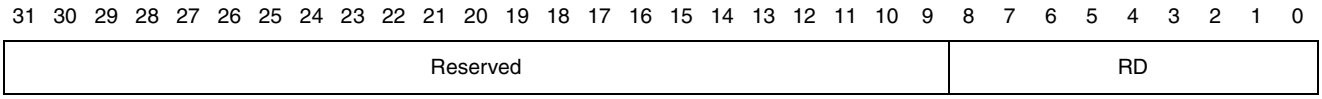
[0] **RBE:** Receiver buffer full interrupt enable
0: Receiver buffer full interrupt disable 1: Receiver buffer full interrupt enable

ASC_n_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																								NUMBER_OF_RETRIES							

Address: *ASCnBaseAddress* + 0x28
Type: R/W
Reset: 1
Description: This register defines the number of transmissions attempted on a piece of data before the UART discards the data. If a transmission still fails after NUMBER_OF_RETRIES the NKD bit is set in the ASC_n_STATUS register where it can be read and acted on by software. This register does not have to be reinitialized after a NACK error.

ASC_n_RXBUFFER ASCn receive buffer



Address: *ASCnBaseAddress* + 0x08

Type: RO

Reset: 0

Description: Serial data reception is only possible when the baudrate generator RUN bit in the ASC_n_CONTROL register 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 the MODE field of the ASC_n_CONTROL register)

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 the MODE bit of the ASC_n_CONTROL register)

[6:0] **RD[6:0]**

Receive buffer data D6 to D0

ASC_n_RXRESET ASCn receive FIFO reset

Address: *ASCnBaseAddress* + 0x24

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_STATUS ASCn 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																					NKD	TF	RHF	TOI	TNE	OE	FE	PE	THE	TE	RBF

Address: *ASCnBaseAddress* + 0x14

Type: RO

Reset: 3 (that is 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] **TOI:** 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] **TNE:** 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, that is, 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_TIMEOUT ASCn time 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																								TIMEOUT							

Address: *ASCnBaseAddress* + 0x1C

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 TIMEOUTNOTEMPTY bit of the ASC_n_STATUS register is 1 when the input FIFO is not empty and the time out counter is zero. The TIMEOUTIDLE bit of the ASC_n_STATUS 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_STATUS register is set.

ASC_n_TXBUFFER 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* + 0x04

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 RUN bit in the ASC_n_CONTROL register 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 the MODE field of the ASC_n_CONTROL register).

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 the MODE field of the ASC_n_CONTROL register).

[6:0] **TD[6:0]:** Transmit buffer data D6 to D0

ASC_n_TXRESET ASCn transmit FIFO reset

Address: *ASCnBaseAddress* + 0x20

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.

35 Smartcard interface

35.1 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 STi7710. 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 33: *Asynchronous serial controller (ASC)* *on page 271* and Chapter 37: *Programmable I/O ports* *on page 297*.

35.2 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 4.

Table 82: Smartcard interface pins

Pins	In/Out[a]	Function
SC0_CLKOUT (PIO0 [3]) SC1_CLKOUT (PIO1 [3])	Out, open drain for 5V cards	Clock for smartcard
SC1_EXTERNAL_CLOCK (PIO1 [2])	In	Optional external clock input to smartcard clock divider #1
SC0_DATAOUT (PIO0 [0]) SC1_DATAOUT (PIO1 [0])	Out, open drain driver	Serial data output. Open drain drive
SC0_DATAIN (PIO0 [1]) SC1_DATAIN (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

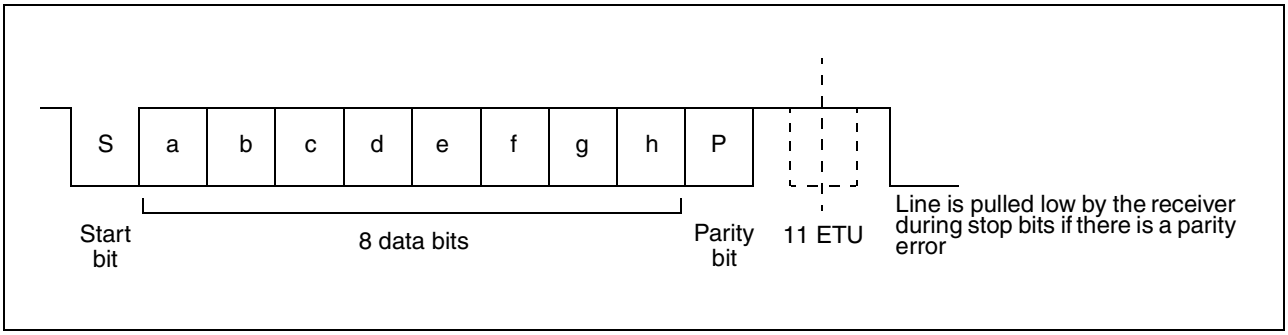
a. *The indicated directions are not set by default. They must be programmed in the PIO configuration registers, since Smart Card interfaces are multiplexed as PIO alternate functions.*

All smart card 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 STi7710 to the smartcard, the ASC must be set up in smartcard mode.

Figure 57: ISO 7816-3 asynchronous protocol



35.3 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 smart card clock generator (that is, glitch free divider) can be set to the 100 MHz system clock or, alternatively, an external reference. For the first smart card interface this alternate source is a dedicated programmable frequency synthesizer. Refer to Chapter 7: *Clocks* *on page 64* for frequency programming details. For the second smart card 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.

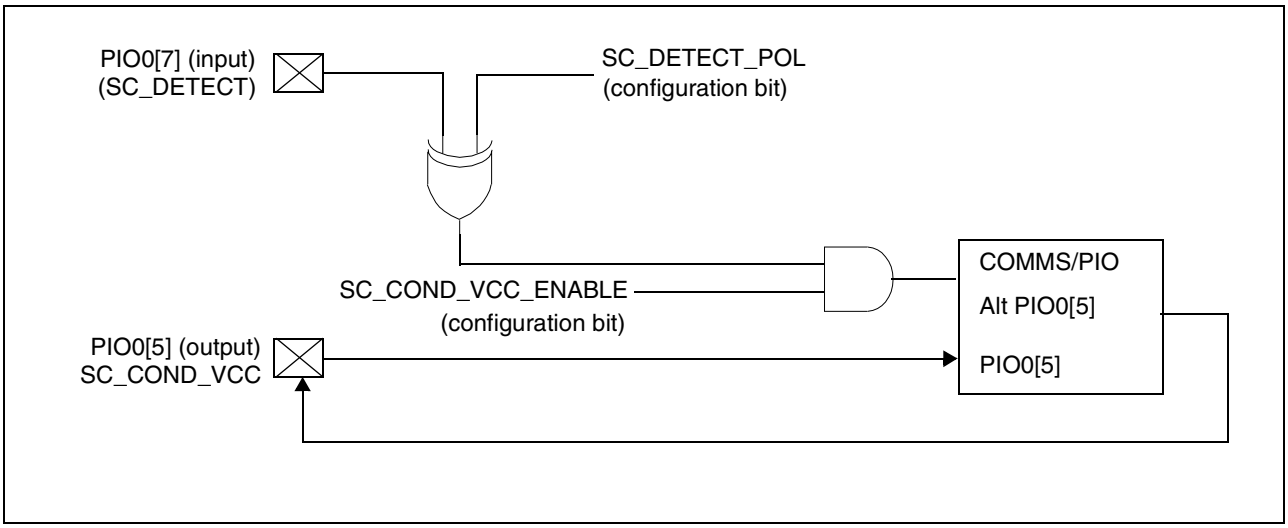
35.4 Smartcard removal and power control

A Smart Card VCC control signal (‘SC_COND_VCC’ alternate out on PIO0) may be directly generated from the smart card 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 6: *Configuration registers on page 60*.

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 smart card 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 smart card interface.

Figure 58: Smartcard VCC control



36 Smartcard interface registers

Addresses are provided as *SmartcardnBaseAddress* + offset.

The *SmartcardnBaseAddresses* are:

SCG0: 0x2004 8000,

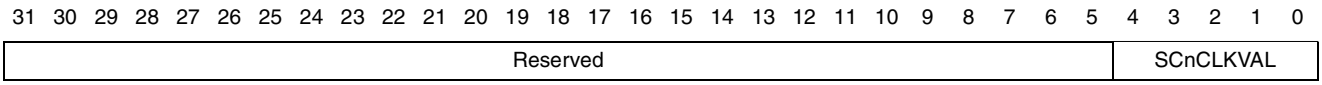
SCG1: 0x2004 9000.

See also Chapter 6: *Configuration registers on page 60*.

Table 83: Smartcard register summary

Register	Description	Offset	Type
SCI_n_CLKVAL	Smartcard n clock	0x00	WO
SCI_n_CLKCON	Smartcard n clock control	0x04	WO

SCI_n_CLKVAL Smartcard n clock



Address: *SmartcardnBaseAddress* + 0x00

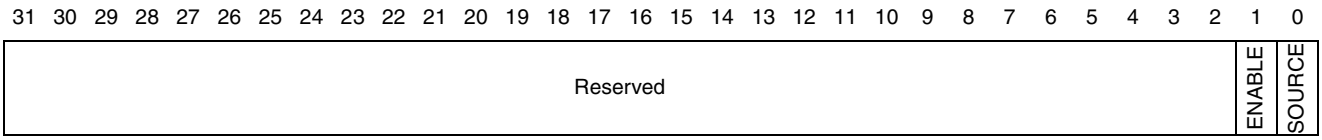
Type: WO

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_CLKCON Smartcard n clock control



Address: *SmartcardnBaseAddress* + 0x04

Type: WO

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] **ENABLE:** 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

37 Programmable I/O 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 *PIOnBaseAddress*. 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_CLEAR_x registers clear bits individually. Writing 1 in these registers resets a corresponding bit in the associated register x; 0 leaves the bit unchanged.

38 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: 0x2002 0000,
PIO1: 0x2002 1000,
PIO2: 0x2002 2000,
PIO3: 0x2002 3000,
PIO4: 0x2002 4000,
PIO5: 0x2002 5000.

Table 84: Programmable I/O ports register summary

Register	Description	Offset	Type
PIO_CLEAR_PnC[2:0]	Clear bits of PnC[2:0]	0x28, 38, 48	WO
PIO_CLEAR_PnCOMP	Clear bits of PnCOMP	0x58	WO
PIO_CLEAR_PnMASK	Clear bits of PnMASK	0x68	WO
PIO_CLEAR_PnOUT	Clear bits of PnOUT	0x08	WO
PIO_PnC[2:0]	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

PIO_CLEAR_PnC[2:0] Clear bits of PnC[2:0]

	7	6	5	4	3	2	1	0
0x28	CLEAR_PC0[7:0]							
0x38	CLEAR_PC1[7:0]							
0x48	CLEAR_PC2[7:0]							

Address: *PIOnBaseAddress* + 0x28 (PIO_CLEAR_PnC0), 0x38 (PIO_CLEAR_PnC1), 0x48 (PIO_CLEAR_PnC2)

Type: WO

Description: PIO_CLEAR_PnC[2:0] allows the bits of registers PIO_PnC[2:0] to be cleared individually. Writing 1 in one of these register clears the corresponding bit in the corresponding PIO_PnC[2:0] register, while 0 leaves the bit unchanged.

PIO_CLEAR_PnCOMP Clear bits of PnCOMP

7	6	5	4	3	2	1	0
CLEAR_PCOMP[7:0]							

Address: *PIOnBaseAddress* + 0x58

Type: WO

Description: PIO_CLEAR_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_CLEAR_PnMASK Clear bits of PnMASK

7	6	5	4	3	2	1	0
CLEAR_PMASK[7:0]							

Address: *PIOnBaseAddress* + 0x68

Type: WO

Description: PIO_CLEAR_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_CLEAR_PnOUT Clear bits of PnOUT

7	6	5	4	3	2	1	0
CLEAR_POUT[7:0]							

Address: *PIOnBaseAddress* + 0x08

Type: WO

Description: PIO_CLEAR_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_PnC[2:0] 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 85.

Table 85: 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_CLEAR_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_CLEAR_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_CLEAR_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: This register 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_CLEAR_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.

39 USB 2.0 host (USBH)

39.1 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 slow, 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 39.1.1)
  - high speed (480 Mbit/s) transmissions
  - microframe and frame 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 39.1.1)
  - low speed (1.5 Mbit/s) and full speed (12 Mbit/s) transmissions
  - power management capabilities

39.1.1 References

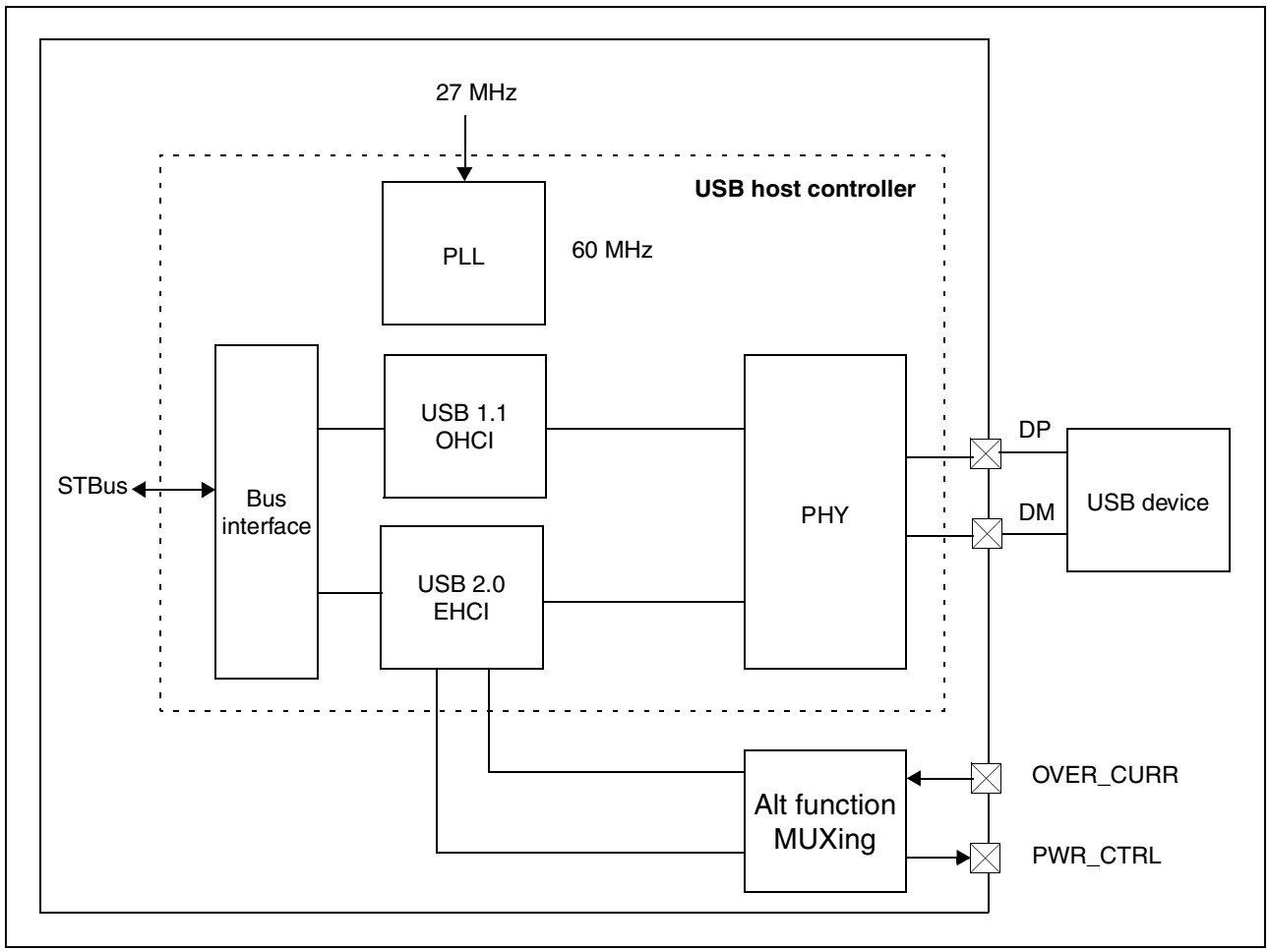
For further details of USB functionality the references below can be downloaded from www.usb.org.

- *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.*

39.2 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 59). 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 59: USB host controller



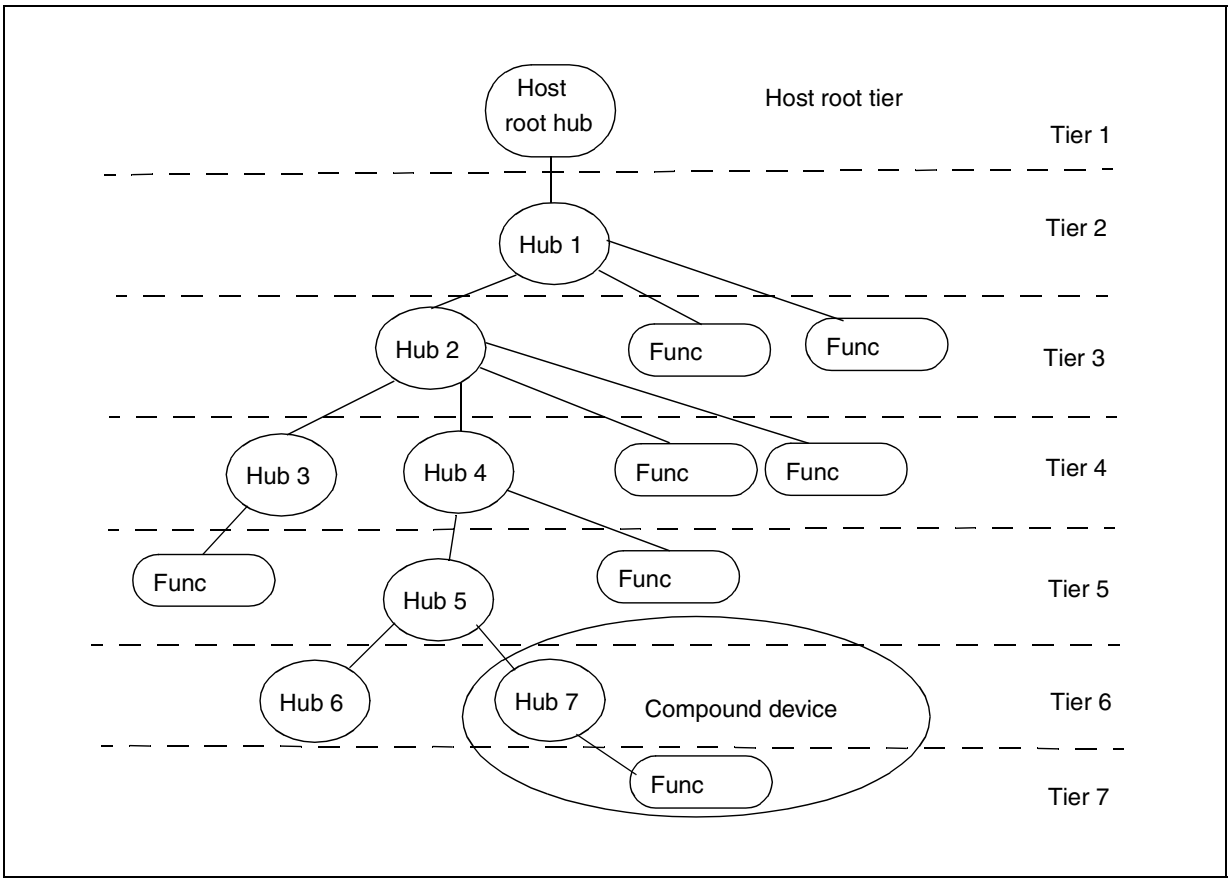
The USB 1.1 host controller has no knowledge of the high-speed mode host controller. High-speed devices are routed to and controlled by the USB 2.0 host controller. When running and configured, the USB 2.0 HC is the default owner of all the root ports. The USB 2.0 HC and its driver initially detect all 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.

39.3 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 60: 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:

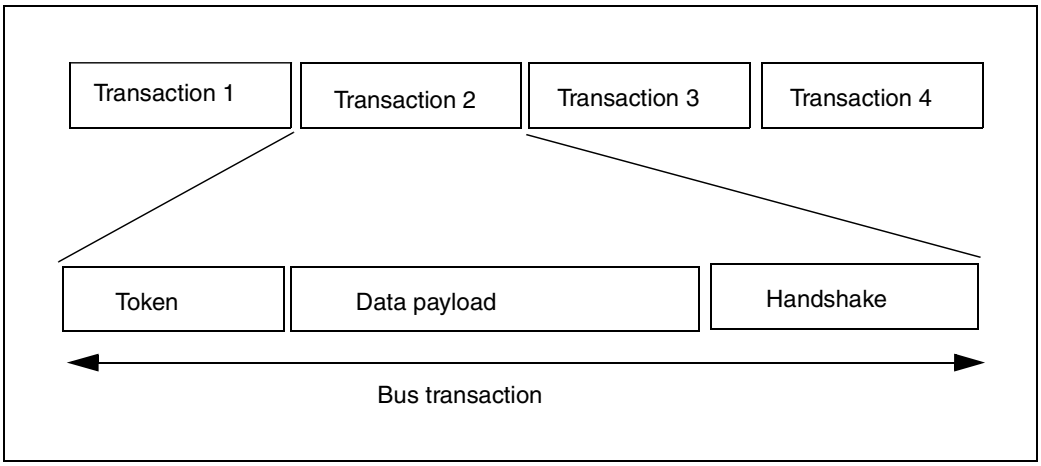
- the type and direction of transaction,
- the USB device address,
- endpoint number.

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 61 describes the process.

Figure 61: 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.

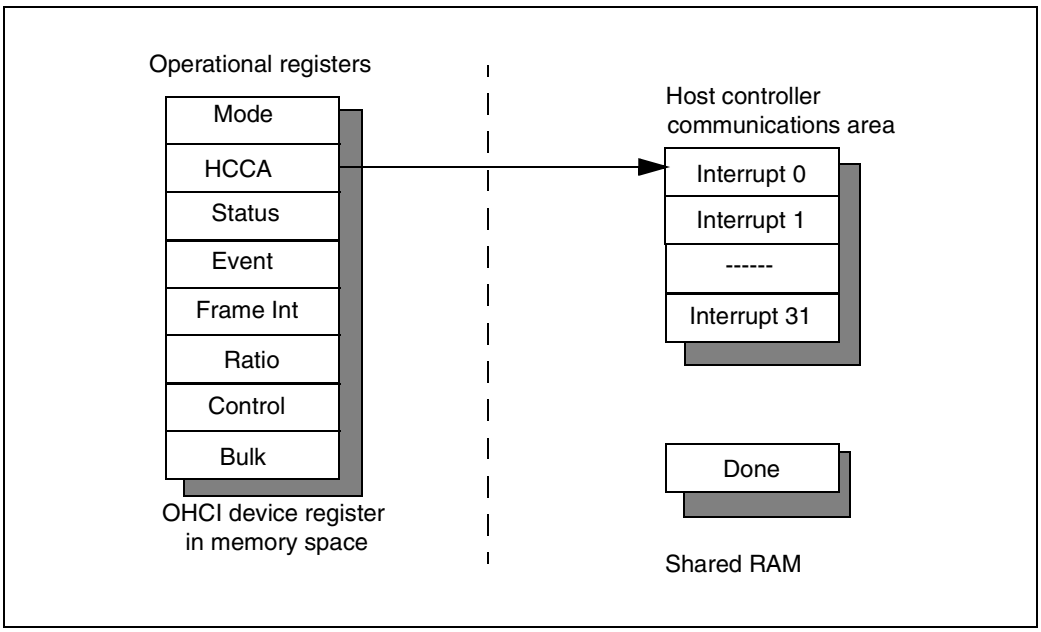
39.4 Data transfers

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 62: USB 1.1 communications channels



40 USB 2.0 host (USBH) registers

There are three sets of registers implemented in the in the USB 2.0 host: OHCI (USB 1.1), EHCI (USB 2) and AHB2 (STBus wrapper).

Addresses are provided as either *OHCIBaseAddress* + offset, *EHCIBaseAddress* + offset, *AHB2BaseAddress* + offset or *SubMicroSystemConfigGlueLogicBaseAddress* + offset.

The *OHCIBufferBaseAddress* is:

0x381F FD00.

The *EHCIBaseAddress* is:

0x381F FE00.

The *AHB2BaseAddress* is:

0x3810 0000.

The *SubMicroSystemConfigGlueLogicBaseAddress* is:

0x3820 1000

Registers in Table 86: *USB 1.1 host register summary* are detailed in the *OpenHCI Open Host Controller Interface Specification for USB*, revision *1.0a*.

Registers in Table 87: *USB 2.0 host register summary* are detailed in the *Enhanced Host Controller Interface Specification for Universal Serial Bus, revision 1.0*.

All registers are 32 bit.

Table 86: USB 1.1 host register summary

Register	Description	Offset	Type (HCD)	Type (HC)
OHCI_HCREVISION	Version number of HC	0x00	RO	RO
OHCI_HCCONTROL	Operating modes for HC	0x04	R/W	R/W
OHCI_HCCOMMANDSTATUS	Shows status and receives commands from HCD	0x08	R/W	R/W
OHCI_HCINTERRUPTSTATUS	Status of events causing interrupts	0x0C	R/W	R/W
OHCI_HCINTERRUPTENABLE	Enables interrupt generation for various events	0x10	R/W	RO
OHCI_HCINTERRUPTDISABLE	Enables interrupt generation for various events	0x14	R/W	RO
OHCI_HCHCCA	HC communication area	0x18	R/W	RO
OHCI_HCPERIODCURRENTED	Current endpoint descriptor	0x1C	RO	R/W
OHCI_HCCONTROLHEADED	First endpoint descriptor of control list	0x20	R/W	RO
OHCI_HCCONTROLCURRENTED	Current endpoint descriptor of control list	0x24	R/W	R/W
OHCI_HCBULKHEADED	First endpoint descriptor of bulk list	0x28	R/W	RO
OHCI_HCBULKCURRENTED	Current endpoint descriptor of bulk list	0x2C	R/W	R/W
OHCI_HCDONEHEAD	Last completed transfer descriptor	0x30	RO	R/W
OHCI_HCFMINTERVAL	Frame time interval	0x34	R/W	RO
OHCI_HCFMREMAINING	Time remaining in the frame	0x38	RO	R/W
OHCI_HCFMNUMBER	Frame number	0x3C	RO	R/W
OHCI_HCPERIODICSTART	Earliest time HC can process periodic list	0x40	R/W	RO
OHCI_HCLSTHRESHOLD	Packet transfer threshold	0x44	R/W	RO

Table 86: USB 1.1 host register summary

Register	Description	Offset	Type (HCD)	Type (HC)
OHCI_HCRHDESCRIPTORA	Description A of root hub	0x48	R/W	RO
OHCI_HCRHDESCRIPTORB	Description B of root hub	0x4C	R/W	RO
OHCI_HCRHSTATUS	Hub status/change	0x50	R/W, RO, WO	R/W, RO
OHCI_HCRHPRTSTATUS_1	Control and report port events	0x54	R/W	R/W

Table 87: 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_USBINTR	USB EHCI interrupt enable	0x18	R/W
EHCI_FRINDEX	USB EHCI frame index	0x1C	R/W
EHCI_CTRLDSSEGMENT	Control data structure segment	0x20	R/W
EHCI_PERIODICLISTBASE	Periodic frame list base address	0x24	R/W
EHCI_ASYNCLISTADDR	Next asynchronous list address	0x28	R/W
EHCI_CONFIGFLAG	Configure flag register	0x50	R/WC, R/W, RO
EHCI_PORTSC_0	Port 0 status and control	0x54	R/W
EHCI_INSNREG00		0x90	
EHCI_INSNREG01		0x94	
EHCI_INSNREG02		0x98	
EHCI_INSNREG03		0x9C	
EHCI_INSNREG04		0xA0	
EHCI_INSNREG05		0xA4	

a. May be written when the WRT_RDONLY bit is set.

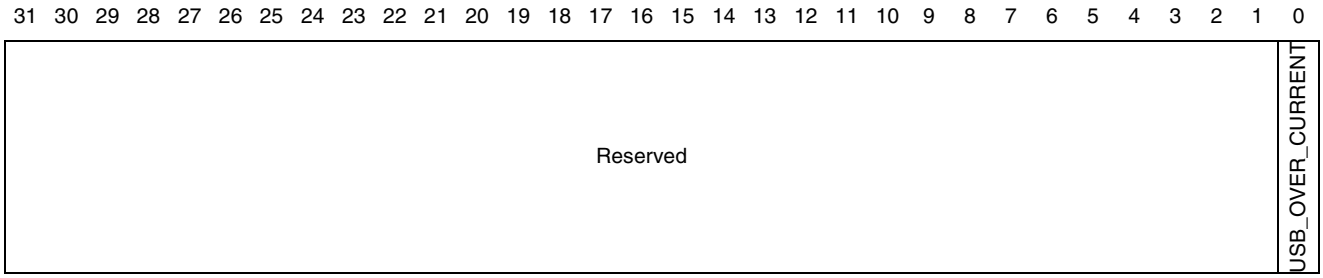
Table 88: STBus wrapper register summary

Register	Description	Offset	Type
AHB2STBUS_STBUS_OPC		0x0F FF00	
AHB2STBUS_MSG_SIZE_CONFIG		0x0F FF04	
AHB2STBUS_CHUNK_SIZE_CONFIG		0x0F FF08	
AHB2STBUS_SW_RESET		0x0F FF0C	
AHB2STBUS_PC_STATUS		0x0F FF10	
AHB2STBUS_FL_ADJ		0x00 0000	
AHB2STBUS_PWAKE_CAP		0x00 0004	
AHB2STBUS_INT_STATUS		0x00 0008	
AHB2STBUS_INT_MASK		0x00 000C	
AHB2STBUS_EHCI_INT_STATUS		0x00 0010	
AHB2STBUS_STRAP		0x00 0014	
AHB2STBUS_OHCI_STATUS		0x00 0018	
AHB2STBUS_POWER_STATE		0x00 001C	
AHB2STBUS_NEXT_POWER_STATE		0x00 0020	
AHB2STBUS_SIMULATION_MODE		0x00 0024	
AHB2STBUS_OHCI_0_APP_IO_HIT		0x00 0028	
AHB2STBUS_OHCI_0_APP_IRQ1		0x00 002C	
AHB2STBUS_OHCI_0_APP_IRQ12		0x00 0030	
AHB2STBUS_SS_PME_ENABLE		0x00 0034	
AHB2STBUS_OHCI_0_LGCY_IRQ		0x00 0038	
AHB2STBUS_EHCI_PME_STATUS_ACK		0x00 0040	

Table 89: USB 2.0 glue register summary

Register	Description	Address	Type
USB_SYSTEM_GLUE	USB system glue (over current)	0x0004	R/W

USB_SYSTEM_GLUE USB system glue (over current)



Address: *SubMicroSystemConfigGlueLogicBaseAddress* + 0x0004

Type: R/W

Reset: 0

Description:

[31:1] **Reserved**

[0] **USB_OVER_CURRENT**

When set, this bit indicates to the USB2 Host Controller that an overcurrent situation has occured on the USB bus. Typically, an external chip would detect the overcurrent on the USB bus, raise an interrupt to the CPU through a PIO, and the interrupt servicing routine would set this bit. As a result the USB2 controller would turn off the power.

41 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.

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
  - memory-to-memory moves or free running transfers
  - paced[1] transfers

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.

41.1 Channel structures

There are two types of channel in the STi7710 FDMA:

- standard 0D, 1D, 2D and paced memory channels,
- an SCD/PES parsing channel,

All the 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.

41.1.1 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 nonword aligned cases.

Paced transfers must be aligned to their opcode size.

41.1.2 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 32byte aligned, OP16 must be 16 byte aligned and so on). The memory side of a paced transfer must be 32-byte aligned.

For SCD/PES parsing the PES buffer must be treated as linear. A circular PES buffer can be described using two linked nodes. The PES buffer must be 128 byte aligned. The ES buffer and Start codes list buffers must also be 32 byte aligned.

41.2 FDMA timing model

41.2.1 Free-running

The FDMA is free-running. Once a channel is started, the 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.

41.2.2 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 63: FDMA request routing

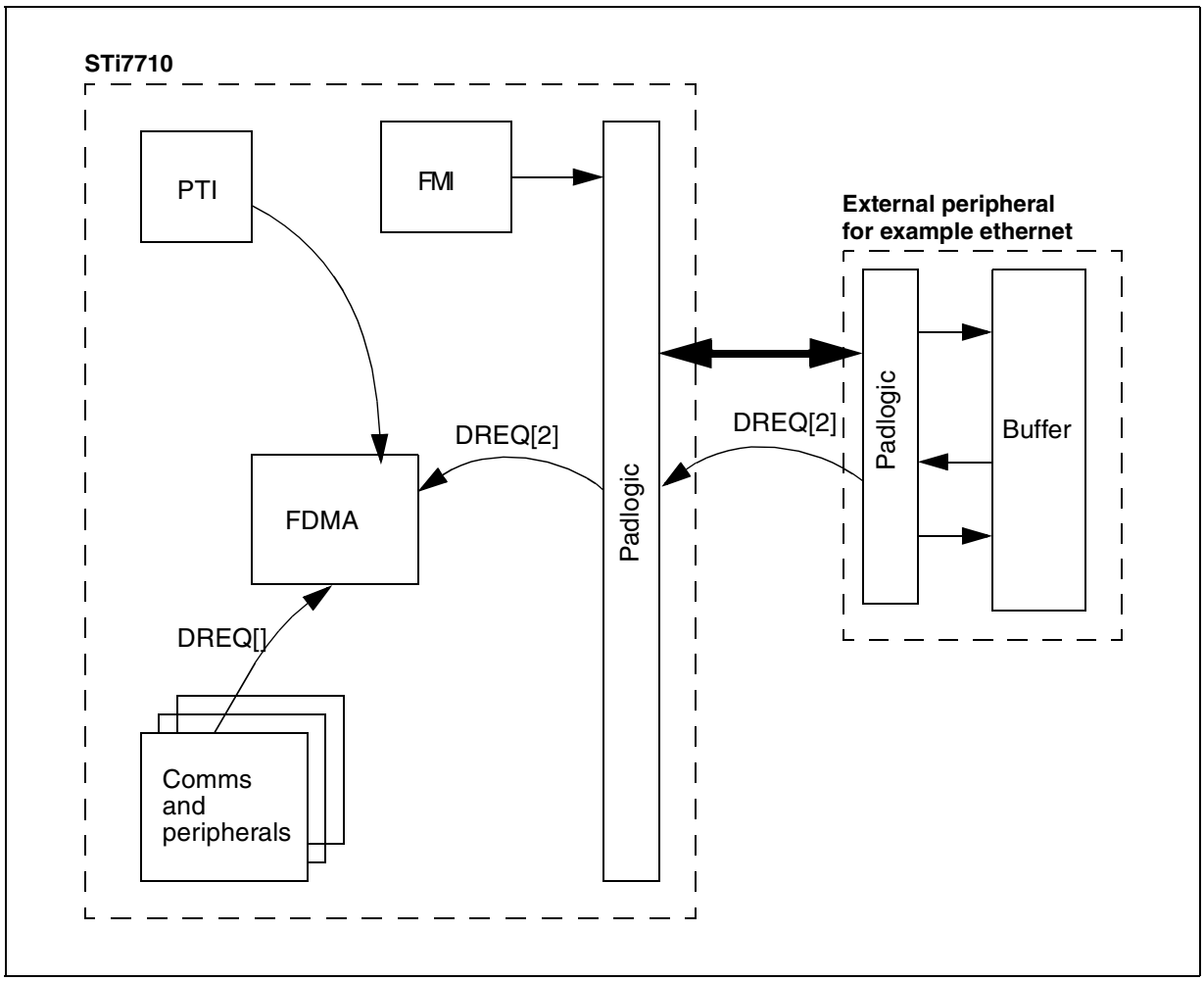


Table 90: FDMA pacing inputs

Block	Block port/signal name	FDMA requests	Description
Reserved		0	
Unused		1 to 7	
COMMs/IRB	IRB_RX_FULL	8	IRB Receive FIFO DREQ
COMMs/UART (x4)	UART[*n*]_RX_FULL	9 to 12	UART Receive FIFO DREQs
	UART[*n*]_TX_EMPTY	13 to 16	UART Transmit FIFO DREQs
COMMs/SSC (x4)	SSC[*n*]_TXBUFF_EMPTY	17 to 20	SSC Transmit FIFO DREQs
	SSC[*n*]_RXBUFF_FULL	21 to 24	SSC Receive FIFO DREQs
Audio	AUDIO_CD_REQ	25	Audio Data Stream Controller DREQ
TSMerger SWTS	SWTS_REQ	26	TSMerger (Software Transport Stream) DREQ
External (x2)	EXT_DMA_REQ_0	27	DREQ from external device (for example peripheral on EMI bus)
	EXT_DMA_REQ_1	28	DREQ from external device (for example peripheral on EMI bus)
Unused		29 to 30	
Reserved		31	

41.2.3 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.

41.2.4 Endianness

The FDMA supports the little endian convention.

41.3 Operating the FDMA

Communication between the CPU and the FDMA is achieved through two 32-bit mailbox registers plus one command and status (FDMA_CMD_STAT) register 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*.

41.3.1 Channel arbitration

Types of transfer are not associated with channel numbers, but are specified in the node's control word (see REQ_MAP field). 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.

41.3.2 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 41.4.3: *SCD/PES parsing on page 322*.

The channel initialization procedure is given below.

1. Create a linked list of nodes describing the transfer in main memory.
2. For SCD/PES parsing write the additional parameters as follows:
   - 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_STAT 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. The FDMA_CMD_STAT register provides the current status of the channel and a pointer to the current node in the linked list.

Note: The FDMA_CMD_STAT 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_STAT 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_STAT.

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.

41.3.3 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_STAT 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_COUNT and FDMA_CMD_STAT registers respectively.

41.3.4 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 a 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

41.3.5 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_STAT.

41.3.6 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.

41.3.7 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_STAT register for the channel (bits 2, 3 and 4). The error bits in the FDMA_CMD_STAT 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_STAT 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.

41.4 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 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.

41.4.1 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 NODE.CONTROL.SRC_INC = CONSTANT SOURCE for a 0-D source, or NODE.CONTROL.DST_INC = CONSTANT SOURCE for a 0-D destination.

Incrementing (1D)

A structure is 1-D if NODE.NBYTES= NODE.LENGTH or if NODE.STRIDE = NODE.LENGTH.

For any 1-D transfer the address is incremented by one byte until NODE.NBYTES bytes are transferred.

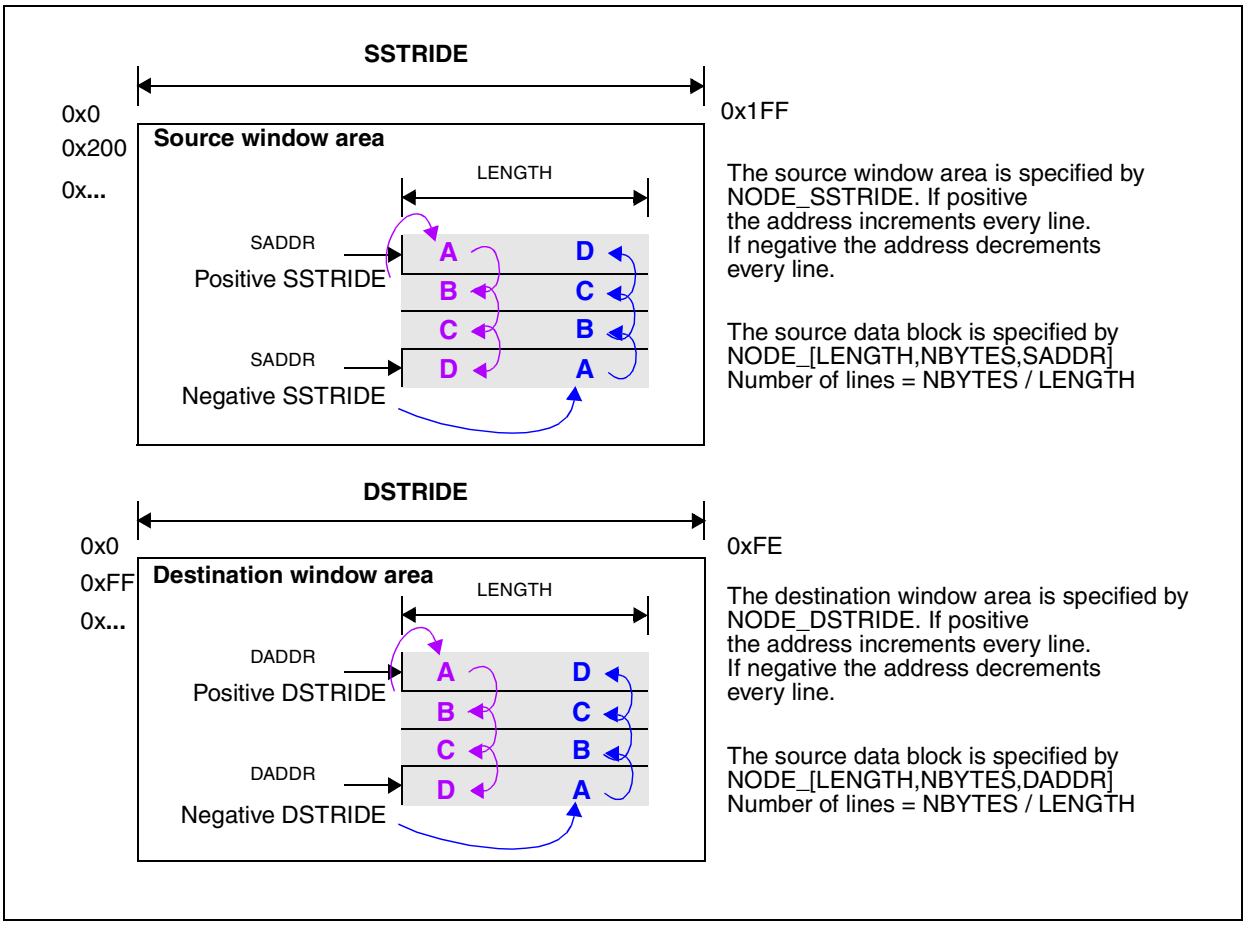
In the case of a incrementing transfer from address NODE.SADDR of NODE.NBYTES bytes, the following bytes at the following addresses are transferred:

NODE.SADDR to NODE.SADDR + (NODE.NBYTES - 1)

Rectangular array (2D)

If NODE.NBYTES ≠ NODE.LENGTH and abs(NODE.[S|D]STRIDE) ≠ NODE.LENGTH then the transfer is 2D.

Figure 64: 2D transfers



The bytes transferred during a 2D transfer are specified by:

```
for (y=0;y<(NODE.NBYTES / NODE.LENGTH);y++)
     for (x=0;x<NODE.LENGTH;x++)
*(char*)(NODE.DADDR+x+NODE.DSTRIDE*y) = *(char*)(NODE.SADDR+x+NODE.SSTRIDE*y)
```

Note: To do a 1D to 2D transfer (or vice-versa) the source or destination stride must be specified (depending on whether the source or destination should be 1-D) to be the same as the length.

Table 91: 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 92: Definitions

Acronym	Description	Variable Name
L	Length	`Length`
SS	Source stride	`SourceStride`
DS	Destination stride	`DestinationStride`
NBytes	Number of bytes to transfer	`NumberBytes`

41.4.2 Paced transfers

The FDMA transfers data from or to 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 (Section 41.1: *Channel structures on page 314*). The paced side is described by the FDMA_REQ_CONTROL[*n*] parameters. The node's control word (NODE_CONTROL) specifies a REQ_MAP number describing which DREQ line and FDMA_REQ_CONTROL 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_CONTROL 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.

Paced peripheral is data source

When the paced peripheral is the data source, data is fetched from the peripheral when it's 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_CONTROL 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.

41.4.3 SCD/PES parsing

A channel configured as a SCD/PES parsing channel fetches data from a linear buffer in main memory (a circular buffer should be handled with a pair of nodes) and detects any start codes within three user-defined ranges: a PES start code range and two elementary stream (ES) start code ranges. The SCD/PES parsing node provides a pointer to the PES buffer and the number of bytes which must be read from the buffer. During SCD/PES parsing, the FDMA processes each node and generates a list of start codes depending on the start code ranges specified in the node. At the end of each node the FDMA can interrupt the host and/or pause the transfer.

The initial write pointer and size of the start code list (SC_WRITE, SC_SIZE) must be provided, as well as the ES buffer parameters to describe the ES buffer (ESBUF_TOP, ESBUF_READ, ESBUF_WRITE, ESBUF_BOT). Also specified are the three start code ranges and whether the range should be detected.

The SCD/PES parsing starts when a START command is issued for the channel. The FDMA can be requested to interrupt the host by setting the INT_ENB and PAUSE_ENB bits in the node's control word.

If PES start code detection is enabled (DETECT_ENABLE in PES_CONTROL = 1), the PES is parsed and the payload (ES) is written to a circular ES buffer. If PES start code detection is disabled, PES headers are not parsed and removed, but ES start code detection is carried out and the output is stored in the circular ES buffer. The start codes list contains the start code and the memory address of the start code (not the offset from the top of the ES buffer). The PTS can also optionally be output to the start codes list.

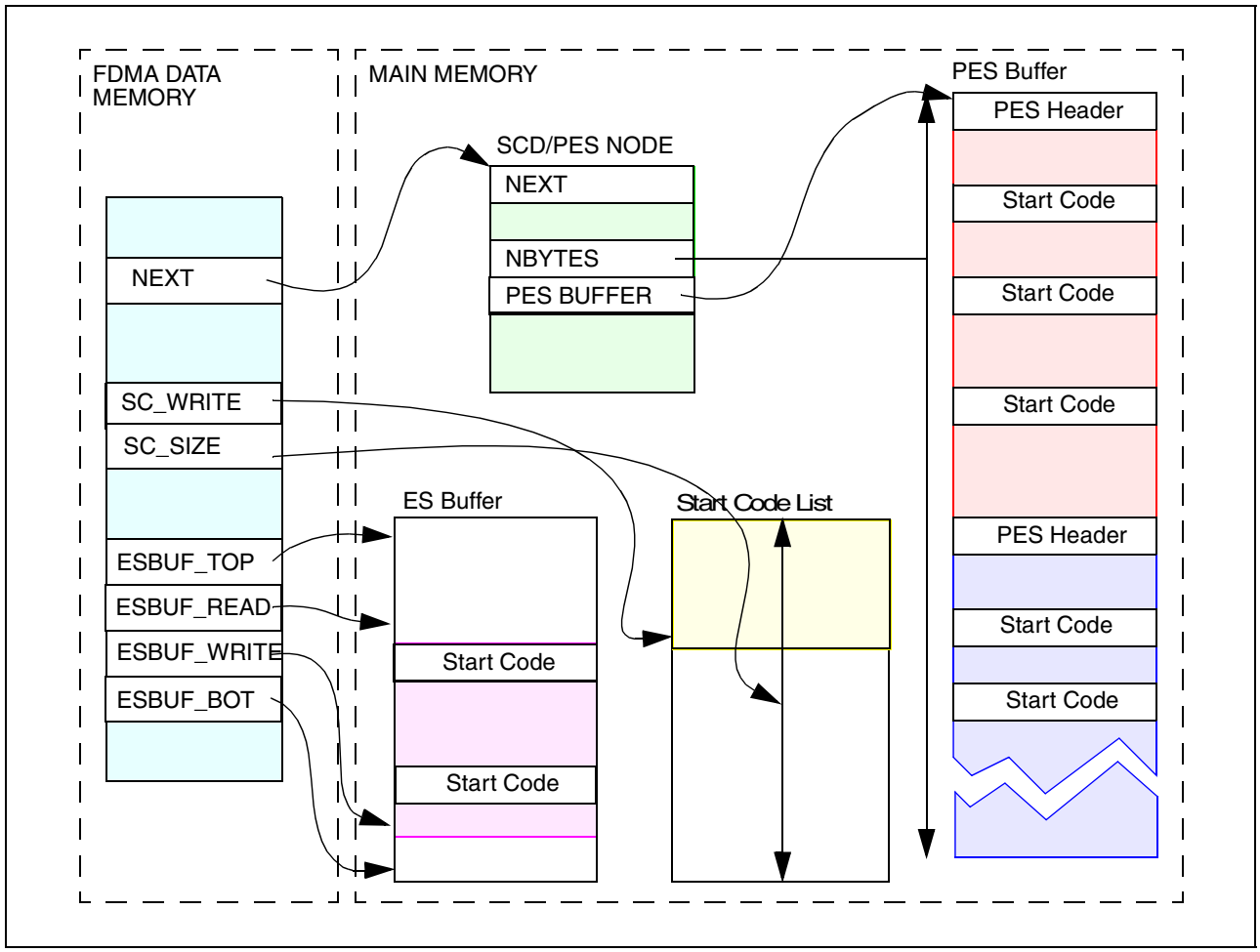
When one-shot mode is selected for a start code range (ONESHOT_MODE in SC*n*_CONTROL = 1), the FDMA detects the first start code in the range. It then ignores the next start codes in the range until the first start code in the other range is found. The FDMA then searches for a start code in the first range once again.

When detection of the PES start code range is enabled (DETECT_ENABLE = 1), the PES stream is parsed, the PES headers removed and the elementary stream is written to the ES buffer. If detection of PES start codes is disabled, start code detection is executed on the incoming ES stream and the stream is copied to the ES buffer.

When PES range detection is enabled using DETECT_ENABLE, the PTS can also be written to the start codes list. For the two ES start code ranges the detection mode must be specified (single-shot or continuous). Three additional data regions specify these parameters. Each node

then references one of these regions. This allows up to three streams to be processed by multiplexing them on the same channel. Each node in the list describes a transfer for one stream.

Figure 65: SCD/PES parsing data structures



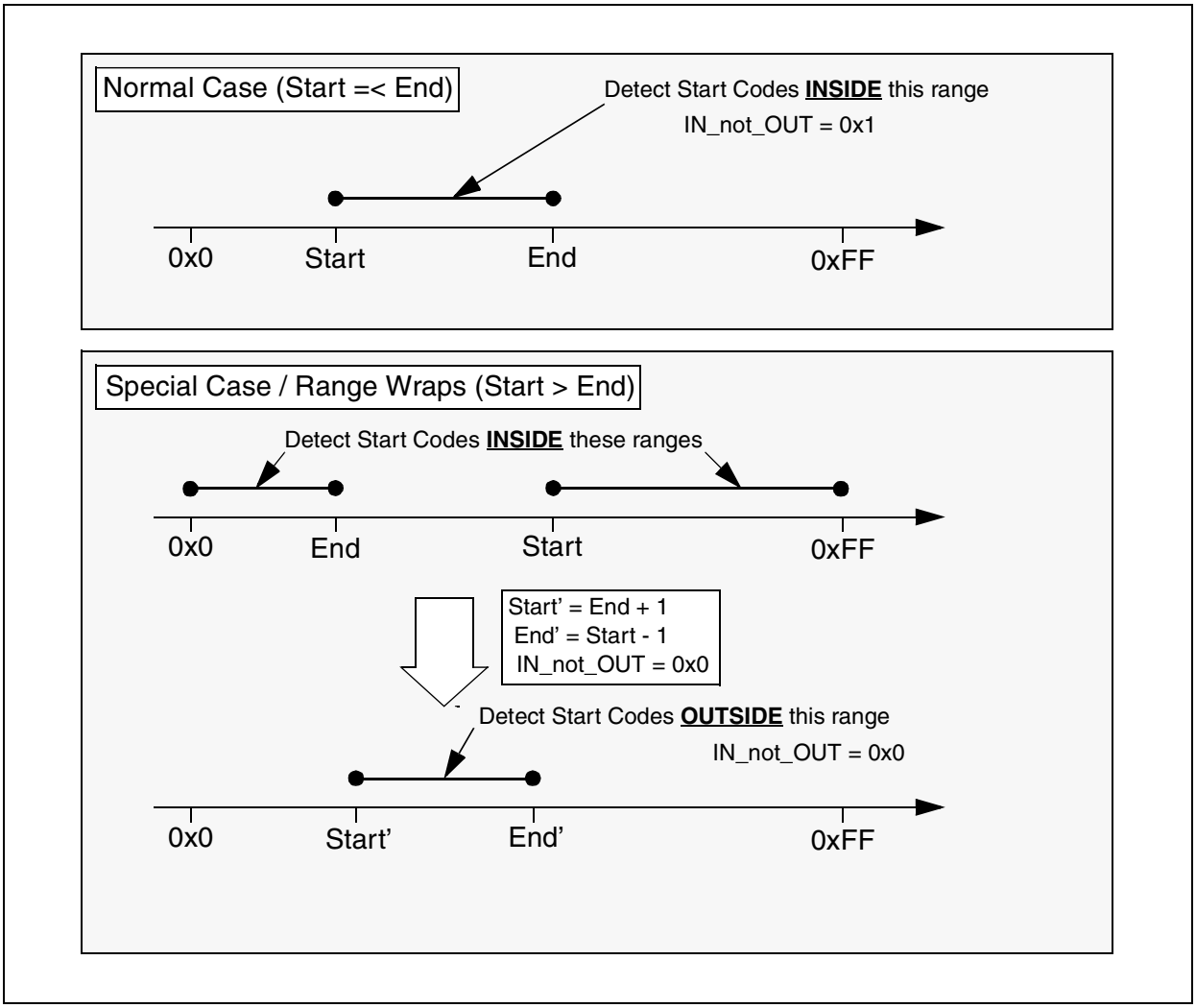
Specifying start code ranges

When specifying the start code ranges the host must ensure that RANGE_END >= RANGE_START (see PES_CONTROL and SCn_CONTROL). 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 66: 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 four, gives the number of entries in the start codes list.

42 Flexible DMA (FDMA) registers

Register addresses are provided as *FDMABaseAddress* + offset or *MemoryOffset* + offset.

The *FDMABaseAddress* is:

0x3805 5000.

In the summary table below, addresses are *FDMABaseAddress* + offset unless shown as '*MO* + ...' (*MemoryOffset* + ...).

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 93: FDMA register summary

Register	Description	Offset	Type
FDMA interface			
FDMA_ID	Hardware ID	0x0000	R/W[a]
FDMA_VERSION	Version number	0x0004	R/W[a]
FDMA_ENABLE	Enable controller	0x0008	R/W[a]
FDMA_CLOCKGATE	Enable controller clock	0x000C	R/W[a]
FDMA_STBUS_SYNC	STBus sync	0x5F88	R/W[a]
Channel interface			
FDMA_REV_ID	Revision number	0x4000	R/W[a]
FDMA_CMD_STAT[n]	Command and status for channel n	0x4030 + *n* x 4 (*n* = 0 to 15)	R/W[b]
FDMA_PTR[n]	Pointer to next node for channel n	0x4070 + *n* x 64 (*n* = 0 to 15)	RO
FDMA_COUNT[n]	Byte count	0x4078 + *n* x 64 (*n* = 0 to 15)	RO
FDMA_REQ_CONTROL[n]	Request control	0x4470 + *n* x 4 (*n* = 0 to 30)	WO
Command mailbox			
FDMA_CMD_STAT	Command status	0x5FC0	RO
FDMA_CMD_SET	Set command	0x5FC4	WO
FDMA_CMD_CLR	Clear command	0x5FC8	WO[a]
FDMA_CMD_MASK	Mask command	0x5FCC	WO
Interrupt mailbox			
FDMA_INT_STAT	Interrupt status	0x5FD0	RO
FDMA_INT_SET	Set interrupt	0x5FD4	WO[a]
FDMA_INT_CLR	Clear interrupt	0x5FD8	WO
FDMA_INT_MASK	Interrupt mask	0x5FDC	WO

Table 93: FDMA register summary

Register	Description	Offset	Type
Memory-to-memory moves and paced registers			
NODE_NEXT	Next register set pointer	*MO* + 0x00	R/W
NODE_CONTROL	Channel control	*MO* + 0x04	R/W
NODE_NBYTES	Transfer count	*MO* + 0x08	R/W
NODE_SADDR	Channel source address	*MO* + 0x0C	R/W
NODE_DADDR	Channel destination address	*MO* + 0x10	R/W
NODE_LENGTH	2D line length	*MO* + 0x14	R/W
NODE_SSTRIDE	2D source stride	*MO* + 0x18	R/W
NODE_DSTRIDE	2D destination stride	*MO* + 0x1C	R/W
SCD/PES parsing registers			
NODE_NEXT	Next node pointer	*MO* + 0x00	R/W
NODE_CONTROL	Node control	*MO* + 0x04	R/W
NODE_NBYTES	PES buffer read size	*MO* + 0x08	R/W
PESBUFFER	PES buffer read pointer	*MO* + 0x0C	R/W
Additional data regions			
SC_WRITE	Start code list write pointer	0x44F0 (Region 0), 0x4530 (Region 1), 0x4570 (Region 2), 0x45B0 (Region 3)	R/W
SC_SIZE	Start code list size	0x44F4 (Region 0), 0x4534 (Region 1), 0x4574 (Region 2), 0x45B4 (Region 3)	R/W
ESBUF_TOP	Top address of elementary stream buffer	0x44F8 (Region 0), 0x4538 (Region 1), 0x4578 (Region 2), 0x45B8 (Region 3)	R/W
ESBUF_READ	Elementary stream buffer read pointer	0x44FC (Region 0), 0x453C (Region 1), 0x457C (Region 2), 0x45BC (Region 3)	R/W
ESBUF_WRITE	Elementary stream buffer write pointer	0x4500 (Region 0), 0x4540 (Region 1), 0x4580 (Region 2), 0x45C0 (Region 3)	R/W
ESBUF_BOT	Elementary stream buffer bottom address	0x4504 (Region 0), 0x4544 (Region 1), 0x4584 (Region 2), 0x45C4 (Region 3)	R/W
PES_CONTROL	PES header start code range control	0x4508 (Region 0), 0x4548 (Region 1), 0x4588 (Region 2)	R/W

Table 93: FDMA register summary

Register	Description	Offset	Type
SCn_CONTROL	Start code range n control	0x450C (Range 1, region 0), 0x4510 (Range 2, region 0), 0x454C (Range 1, region 1), 0x4550 (Range 2, region 1), 0x458C (Range 1, region 2), 0x4590 (Range 2, region 2)	R/W
SCD_STATE	Start code detector state	0x4514 to 0x452F (Region 0), 0x4530 to 0x456F (Region 1), 0x4570 to 0x45AF (Region 2)	R/W
Start code entries			
SC_TYPE	Type of data in this entry	*MO* + 0x0000	R/W
SC_ADDRESS	Memory address of start code	*MO* + 0x0004	R/W
SC_VALUE	Start code value	*MO* + 0x0008	R/W
PTS entries			
PTS_TYPE	Type of data in this entry	*MO* + 0x0000	R/W
PTS_ADDRESS	Memory address of PTS	*MO* + 0x0004	R/W
PTS_UPPER	MSB of PTS value	*MO* + 0x0008	R/W
PTS_LOWER	LSB of PTS value	*MO* + 0x000C	R/W

a. Writable only during initialization.
b. RO (when channel is running), R/W (when channel is idle or paused).

42.1 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_NUMBER																															

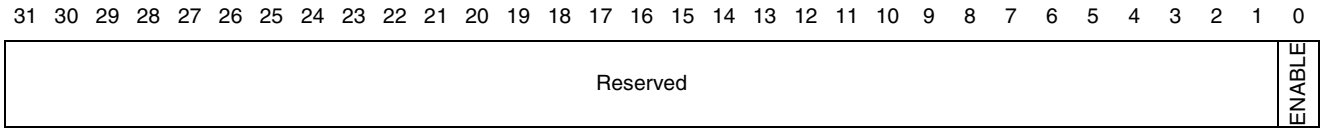
Address: *FDMABaseAddress* + 0x0000
Type: R/W (writable only during initialization)
Reset: 0
Description: Holds hardware ID number.

FDMA_VERSION 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
VERSION_NUMBER																															

Address: *FDMABaseAddress* + 0x0004
Type: R/W (writable only during initialization)
Reset: 0
Description: Holds hardware version number.

FDMA_ENABLE Enable controller

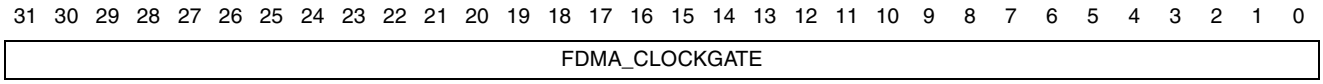


Address: *FDMABaseAddress* + 0x0008
Type: R/W (writable only during initialization)
Reset: 0
Description:

[31:1] **Reserved**

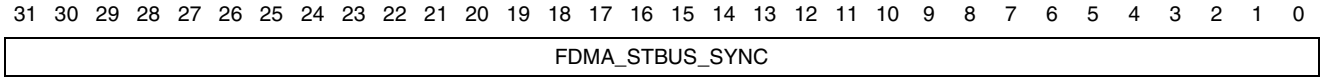
[0] **ENABLE**
1: Block enabled or running
0: Block stopped, CPU can access embedded memory.

FDMA_CLOCKGATE Enable controller clock



Address: *FDMABaseAddress* + 0x000C
Type: R/W (writable only during initialization)
Reset:
Description:

FDMA_STBUS_SYNC STBus sync



Address: *FDMABaseAddress* + 0x5F88
Type: R/W (writable only during initialization)
Reset:
Description:

42.2 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
FDMA_REV_ID																															

Address: *FDMABaseAddress* + 0x4000
Type: R/W (writable only during initialization)
Reset: Undefined
Description: Holds low level revision number.

FDMA_CMD_STAT[n] 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																											ERROR			STATUS	
Channel idle or paused	DATA																											COMMAND				

Address: *FDMABaseAddress* + 0x4030 + *n* x 4 (where *n* = 0 to 15)
Type: RO (when channel is running)
Type: R/W (when channel is idle or paused)
Reset:
Description: The FDMA_CMD_STAT 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 this 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 0 if this channel has never been used
Otherwise: Current node address

[4:2] **ERROR**: error type (only valid if interrupt mailbox error bit is set)
000: interrupt missed
Others: Reserved

[1:0] **STATUS:** channel status
00: channel is idle
10: channel is running
11: channel is paused
01: Reserved

Channel idle or paused

[31:5] **DATA**
0 0001: START and initialize channel *n*
0 0000: RESTART channel *n* (no initialization) from where it was paused
Others: Reserved

[4:0] **COMMAND:** Pointer to node if command is START
Reserved otherwise

FDMA_PTR[n] 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* + 0x4070 + *n* x 64 (where *n* = 0 to 15)
Type: RO
Reset: Undefined
Description:

[31:5] **NODE_PTR**: address of next node (32 byte aligned)

[4:0] **Reserved**

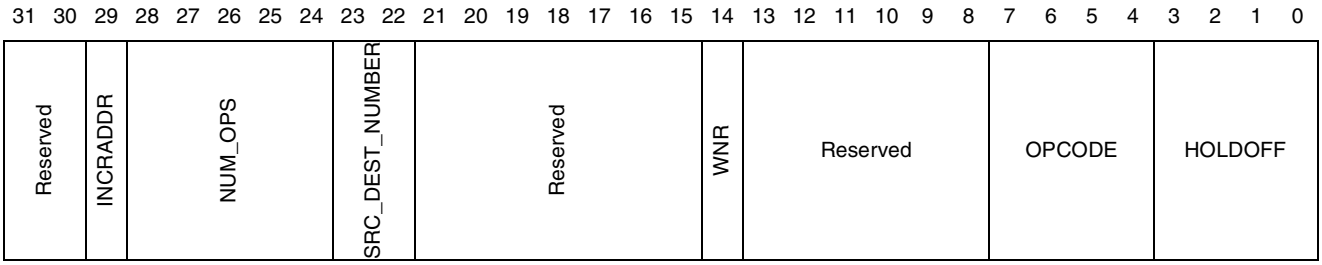
FDMA_COUNT[n] Byte count

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* + 0x4078 + *n* x 64 (where *n* = 0 to 15)
Type: RO
Reset: Undefined
Description:

[31:0] **COUNT**: number of bytes remaining to be transferred for current node

FDMA_REQ_CONTROL[n] Request control



Address: *FDMABaseAddress* + 0x4470 + *n* x 4 (where *n* = 0 to 30)

Type: WO

Reset: Undefined

Description:

[31:30] **Reserved**

[29] **INCRADDR**: increment address
0: no address increment between transfers
1: increment address between transfers

[28:24] **NUM_OPS**: Number of ops
Number of ops per request serviced
0x0: 1 transfer
0x1: 2 transfers
0x2: 3 transfers
0x3: 4 transfers
others: Reserved

[23:22] **SRC_DEST_NUMBER**:
0: Software Transport Stream
1: All other paced channels
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 will be masked for:
0x0: 0us - 0.5us
0x1: 0.5us - 1us
0x2: 1us - 1.5us
others: Reserved

42.3 Command mailbox

FDMA_CMD_STAT 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
CHANNEL15	CHANNEL14	CHANNEL13	CHANNEL12	CHANNEL11	CHANNEL10	CHANNEL9	CHANNEL8	CHANNEL7	CHANNEL6	CHANNEL5	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0

Address: *FDMABaseAddress* + 0x5FC0

Type: RO

Reset: Undefined

Description: Indicates a pending command for channel *n*.

[2 x *n* + 1:2 x *n*] **CHANNEL*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
CHANNEL15	CHANNEL14	CHANNEL13	CHANNEL12	CHANNEL11	CHANNEL10	CHANNEL9	CHANNEL8	CHANNEL7	CHANNEL6	CHANNEL5	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0

Address: *FDMABaseAddress* + 0x5FC4

Type: WO

Reset: Undefined

Description: Notify FDMA there is a command for channel *n*.

[2 x *n* + 1:2 x *n*] **CHANNEL*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
CHANNEL15	CHANNEL14	CHANNEL13	CHANNEL12	CHANNEL11	CHANNEL10	CHANNEL9	CHANNEL8	CHANNEL7	CHANNEL6	CHANNEL5	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0

Address: *FDMABaseAddress* + 0x5FC8

Type: WO at initialization

Reset: Undefined

Description: When set, these bits acknowledge the command for channel *n* by clearing bits in the FDMA_CMD_STAT 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
CHANNEL15		CHANNEL14		CHANNEL13		CHANNEL12		CHANNEL11		CHANNEL10		CHANNEL9		CHANNEL8		CHANNEL7		CHANNEL6		CHANNEL5		CHANNEL4		CHANNEL3		CHANNEL2		CHANNEL1		CHANNEL0	

Address: *FDMABaseAddress* + 0x5FCC

Type: WO

Reset: Undefined

Description: Enable interrupt generation for channel *n*.

[2 x *n* + 1:2 x *n*] **CHANNEL*n***

00: Disable channel *n* messaging

11: Enable channel *n* messaging

Others: Reserved

42.4 Interrupt mailbox

FDMA_INT_STAT 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
ERROR15	CHANNEL15	ERROR14	CHANNEL14	ERROR13	CHANNEL13	ERROR12	CHANNEL12	ERROR11	CHANNEL11	ERROR10	CHANNEL10	ERROR9	CHANNEL9	ERROR8	CHANNEL8	ERROR7	CHANNEL7	ERROR6	CHANNEL6	ERROR5	CHANNEL5	ERROR4	CHANNEL4	ERROR3	CHANNEL3	ERROR2	CHANNEL2	ERROR1	CHANNEL1	ERROR0	CHANNEL0

Address: *FDMABaseAddress* + 0x5FD0

Type: RO

Reset: Undefined

Description: When a channel is running, its FDMA_CMD_STAT 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_STAT 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 error bit.

[*n* x 2] **CHANNELn:** If bit *n* = 1 message pending for channel *n*

[*n* x 2 + 1] **ERRORn:** 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
ERROR15	CHANNEL15	ERROR14	CHANNEL14	ERROR13	CHANNEL13	ERROR12	CHANNEL12	ERROR11	CHANNEL11	ERROR10	CHANNEL10	ERROR9	CHANNEL9	ERROR8	CHANNEL8	ERROR7	CHANNEL7	ERROR6	CHANNEL6	ERROR5	CHANNEL5	ERROR4	CHANNEL4	ERROR3	CHANNEL3	ERROR2	CHANNEL2	ERROR1	CHANNEL1	ERROR0	CHANNEL0

Address: *FDMABaseAddress* + 0x5FD4

Type: WO on initialization

Reset: Undefined

Description: These bits generate an interrupt for channel *n*.

[*n* x 2] **CHANNELn:** generate interrupt for channel *n*

[*n* x 2 + 1] **ERRORn:** 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
ERROR15	CHANNEL15	ERROR14	CHANNEL14	ERROR13	CHANNEL13	ERROR12	CHANNEL12	ERROR11	CHANNEL11	ERROR10	CHANNEL10	ERROR9	CHANNEL9	ERROR8	CHANNEL8	ERROR7	CHANNEL7	ERROR6	CHANNEL6	ERROR5	CHANNEL5	ERROR4	CHANNEL4	ERROR3	CHANNEL3	ERROR2	CHANNEL2	ERROR1	CHANNEL1	ERROR0	CHANNEL0

Address: *FDMABaseAddress* + 0x5FD8

Type: WO

Reset: Undefined

Description: When set these bits acknowledge an interrupt for channel *n* by clearing the relevant status bits.

[*n* x 2] **CHANNEL*n*:** acknowledge message for channel *n*

[*n* x 2 + 1] **ERROR*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
ERROR15	CHANNEL15	ERROR14	CHANNEL14	ERROR13	CHANNEL13	ERROR12	CHANNEL12	ERROR11	CHANNEL11	ERROR10	CHANNEL10	ERROR9	CHANNEL9	ERROR8	CHANNEL8	ERROR7	CHANNEL7	ERROR6	CHANNEL6	ERROR5	CHANNEL5	ERROR4	CHANNEL4	ERROR3	CHANNEL3	ERROR2	CHANNEL2	ERROR1	CHANNEL1	ERROR0	CHANNEL0

Address: *FDMABaseAddress* + 0x5FDC

Type: WO

Reset: Undefined

Description: When set these bits enable interrupt generation for channel *n*.

[*n* x 2] **CHANNEL*n*:** If bit *n* = 1 message pending for channel *n*

[*n* x 2 + 1] **ERROR*n*:** If bit *n* = 1 there is an error for channel *n*

42.5 Memory-to-memory moves and paced transfer registers

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_POINTER																											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.

NODE_CONTROL 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_ENB	PAUSE_ENB	Reserved																							DST_INC		SRC_INC			REQ_MAP		

Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description:

[31]: **INT_ENB**: 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] **PAUSE_ENB:** 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 follows:
1: Select request 1
30: Select request 30
31: Extended node type

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.

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.

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.

NODE_LENGTH 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.

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.

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.

42.6 SCD/PES parsing registers

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_POINTER																											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.

NODE_CONTROL 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_ENB	PAUSE_ENB	Reserved						TAG								Reserved				ADD_DATA				TYPE			REQ_MAP				

Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description:

[31] **INT_ENB**: interrupt at end of node
0: No interrupt at end of node
1: Interrupt host at end of node

[30] **PAUSE_ENB**: 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 - 0x7: Reserved

[4:0] **REQ_MAP**: REQ_MAP/extended node type
0x1F: Extended Node Type
0x0-0x1E: Reserved (DREQ mapping for standard node)

NODE_NBYTES PES buffer read 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
NBYTES

Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: Number of bytes in PES buffer on which SCD/PES parsing should be performed.

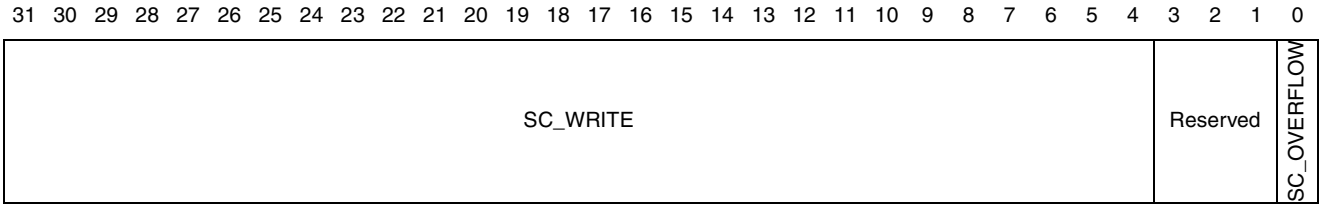
PESBUFFER PES buffer read pointer

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
PESBUFFER_READ

Address: *MemoryOffset* + 0x000C
Type: R/W
Reset: Undefined
Description: Read pointer to PES buffer.

42.6.1 Additional data regions

SC_WRITE Start code list write pointer



Address: *FDMABaseAddress* + 0x44F0 (Region 0), 0x4530 (Region 1), 0x4570 (Region 2), 0x45B0 (Region 3)

Type: R/W

Reset: 0

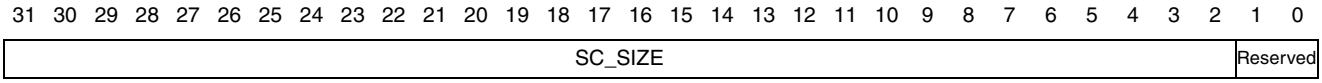
Description:

[31:4] **SC_WRITE**: Write pointer for start codes list (must be 4 word aligned)

[3:1] **Reserved**

[0] **SC_OVERFLOW:** SC List Overflow flag
0: no overflow
1: overflow

SC_SIZE Start code list size



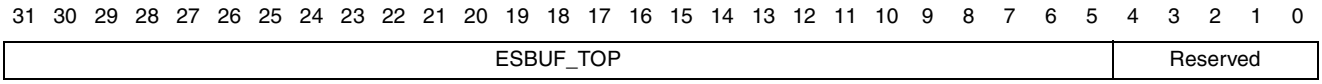
Address: *FDMABaseAddress* + 0x44F4 (Region 0), 0x4534 (Region 1), 0x4574 (Region 2), 0x45B4 (Region 3)

Type: R/W

Reset: 0

Description: Size of start codes list (in number of words).

ESBUF_TOP Top address of elementary stream buffer



Address: *FDMABaseAddress* + 0x44F8 (Region 0), 0x4538 (Region 1), 0x4578 (Region 2), 0x45B8 (Region 3)

Type: R/W

Reset: 0

Description: Address of top of elementary stream buffer(32 byte aligned)

ESBUF_READ Elementary stream buffer read 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
ESBUF_READ																															

Address: *FDMABaseAddress* + 0x44FC (Region 0), 0x453C (Region 1), 0x457C (Region 2), 0x45BC (Region 3)

Type: R/W

Reset: 0

Description:

ESBUF_WRITE Elementary stream buffer write 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
ESBUF_WRITE																															

Address: *FDMABaseAddress* + 0x4500 (Region 0), 0x4540 (Region 1), 0x4580 (Region 2), 0x45C0 (Region 3)

Type: R/W

Reset: 0

Description:

ESBUF_BOT Elementary stream buffer bottom 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
ESBUF_BOT																											Reserved				

Address: *FDMABaseAddress* + 0x4504 (Region 0), 0x4544 (Region 1), 0x4584 (Region 2), 0x45C4 (Region 3)

Type: R/W

Reset: 0

Description: ES buffer bottom pointer (32 byte aligned).

PES_CONTROL PES header start code range 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
IN_NOT_OUT	DETECT_ENABLE	OUTPUT_PTS	Reserved	RANGE_END	RANGE_START

Address: *FDMABaseAddress* + 0x4508 (Region 0), 0x4548 (Region 1), 0x4588 (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_ENABLE:** 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*
* NOTE: RANGE_END >= RANGE_START

[7:0] **RANGE_START:** start of PES start code range*
* NOTE: RANGE_END >= RANGE_START

SCn_CONTROL Start code range n 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
IN_NOT_OUT	DETECT_ENABLE	ONESHOT_MODE	Reserved													RANGE_END								RANGE_START							

Address: *FDMABaseAddress* +
0x450C (Range 1, region 0), 0x4510 (Range 2, region 0),
0x454C (Range 1, region 1), 0x4550 (Range 2, region 1),
0x458C (Range 1, region 2), 0x4590 (Range 2, 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_ENABLE**: 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)*
* NOTE: RANGE_END >= RANGE_START

[7:0] **RANGE_START**: start of start code range (inclusive)*
* NOTE: RANGE_END >= RANGE_START

SCD_STATE Start code detector state

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

Address: *FDMABaseAddress* + 0x4514 to 0x452F (Region 0), 0x4530 to 0x456F (Region 1), 0x4570 to 0x45AF (Region 2)

Type: R/W

Reset: 0

Description: Must be set to 0x0 when a new stream is started on this additional data area.

42.6.2 Start code entries

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

SC_ADDRESS 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
ADDRESS																															

Address: *MemoryOffset*+ 0x0004
Type: R/W
Reset: Undefined
Description: Memory address of start code in ES.

SC_VALUE Start code 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																								START_CODE							

Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: Value of the start code

42.6.3 PTS entries

PTS_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

PTS_ADDRESS Memory address of PTS

31	30	29	28	27	26	25	24	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																															

Address: *MemoryOffset* + 0x0004
Type: R/W
Reset: Undefined
Description: Address where PTS would appear in ES.

PTS_UPPER MSB of PTS 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																															PTS[32]

Address: *MemoryOffset* + 0x0008
Type: R/W
Reset: Undefined
Description: Bit 32 of PTS.

PTS_LOWER LSB of PTS 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
PTS[31:0]																															

Address: *MemoryOffset* + 0x000C
Type: R/W
Reset: Undefined
Description: Bits 0 to 31 of PTS.

43 PWM and counter modules

The STi7710 provides two multi-channel PWM/Timer modules:

- **C1_PWM**, shared with the ST20-C105.
  One of its PWM outputs is reserved for use as a “timer tick” signal by the ST20 CPU for time slicing. Its other I/Os are available for other uses: these are an additional PWM output, C1_PWM_OUT0, and a capture input, PWM_CAPTURE0, both available as PIO alternate functions, plus an interrupt request to the Interrupt Level Controller, C1_PWM_INTERRUPT,.
  Note: different duty cycles can be programmed for the two PWM outputs but this is not true for the period: a single, common period setting has to be chosen for both outputs.
- **PWM_Timer2**, an independent dual PWM-Timer module.
  This includes two identical channels, each with one PWM output routed to an output pin and one interrupt request routed to the ILC. The PWM outputs pins are PWM_OUT_A and PWM_OUT_B (PIO alternate functions) and the interrupt requests are PWMTimer2_Interrupt_A and PWMTimer2_Interrupt_B.

Those modules are described in more detail below.

43.1 C1_PWM

43.1.1 Overview

The C1_PWM provides two PWM outputs (one of them dedicated to the ST20 timer tick function, for time slicing), plus one PWM capture input PWM_CAPTURE0 and four compare functions usable as programmable timers. The capture input can be programmed to detect rising edge, falling edge, both edges or neither edge (disabled). The PWM output is clocked by the system clock (100 MHz nominal); the capture and compare are clocked at twice system clock (200 MHz nominal).

The module is programmed by means of registers described in the next chapter.

The module generates a single interrupt signal. The exact event which caused an interrupt can be determined by reading the status bits in a register, which can then be cleared (acknowledged).

43.1.2 External interface

Table 94: PWM and counter pins

I/O assigned	Function name	Type	Function
N/A	C1_PWM_OUT3	Internal	Timer tick to ST20 CPU for time slicing
PIO4[7]	C1_PWM_OUT0	Out	PWM output
PIO3[7]	PWM_CAPTURE0	In	Capture trigger input
n/a]	C1_PWM_INTERRUPT	Internal	Interrupt Request to ILC

43.1.3 PWM outputs

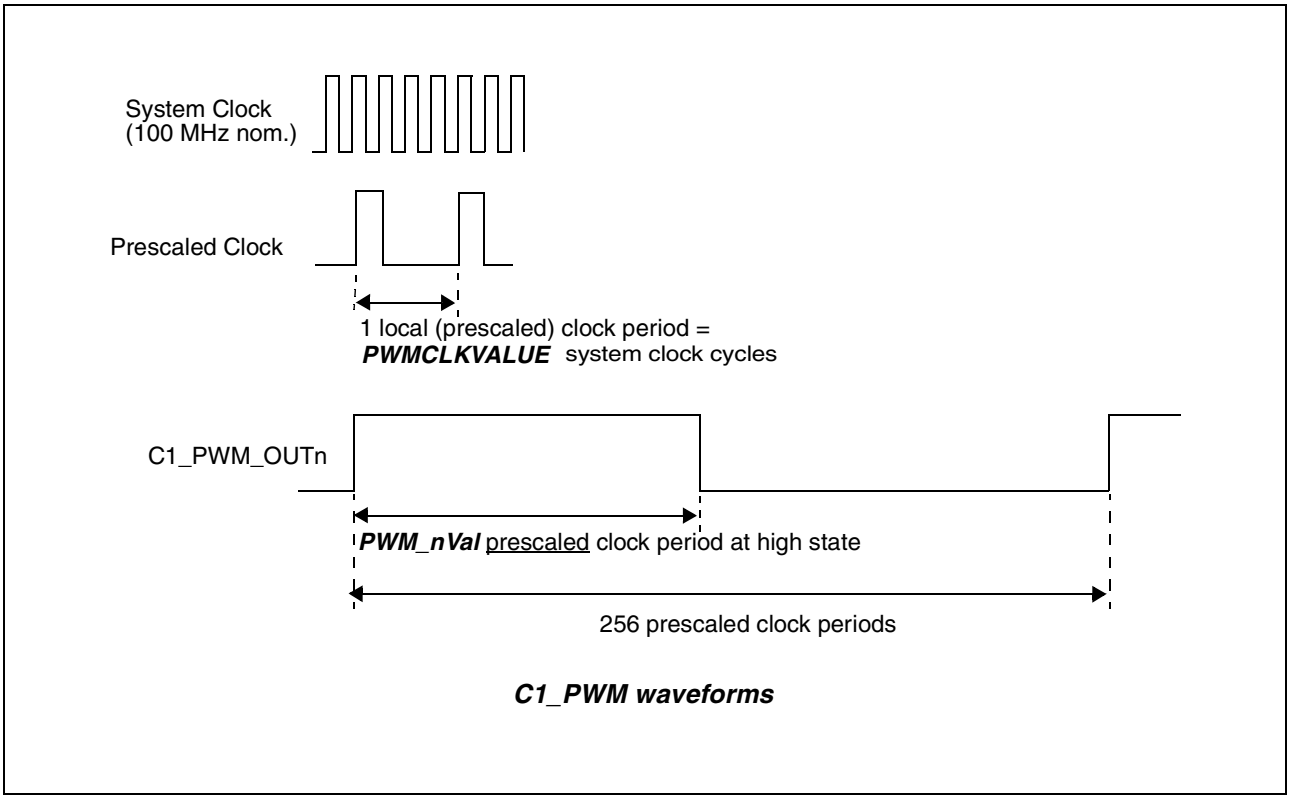
The two PWM outputs C1_PWM_OUT3 and C1_PWM_OUT0 share a common counter that defines the period. The relative width (in counts) of the output pulse on each pin C1_PWM_OUTn is set between 1 and 256 by loading a value from 0 to 255 into the register PWM_nVAL. In other words, PWM_nVAL determines the duty cycle on pin C1_PWM_OUTn, which is (PWM_nVAL+1)/256 (0.4% to 100% in steps of 0.4%). Pulses occur every 256 counts.

The counter is clocked by the system clock (100 MHZ nominal) divided by a prescaler. The prescaling factor, and therefore the period represented by one count, is determined by the value of field PWMCLKVALUE in register PWM_CONTROL. The factor can be from 1 to 16.

The counter (in register PWM_COUNT) is enabled by setting bit PWMEN of register PWM_CONTROL to 1. When it is disabled (PWMEN is 0), C1_PWM_OUTn is forced low. PWM_COUNT is writable at any time but can have a synchronization latency.

When the PWM counter overflows, an interrupt is generated if bit INTEN of register PWM_INTENABLE is set to 1. Bit INT of register PWM_INTSTATUS becomes 1, and can be reset by writing 1 to bit INTACK of register PWM_INTACK.

Figure 67: C1_PWM output waveform



C1_PWM waveforms

43.1.4 Capture input

There is one capture input which shares a common counter with the four compare facilities described below.

What constitutes an event on input PWM_CAPTURE0 is defined by the code in register PWM_CAPTUREEDGE. Possible events are rising edge, falling edge, both or neither (in other words, disabled).

When an input event occurs on input PWM_CAPTURE0, the value of the counter (in register PWM_CAPTURECOUNT) at that time is captured in register PWM_CAPTUREVAL. The value can be 0x0000 0000 to 0xFFFF FFFF.

In addition, when an input event occurs, an interrupt is generated provided the CPTIE bit of the PWM_INTENABLE register is set to 1. Bit CPT of register PWM_INTSTATUS becomes 1, and can be reset by writing 1 to bit CPTIA of register PWM_INTACK.

The counter is not stopped nor reset by any of these events. See Section 43.1.6: Capture/ compare counter, prescaling and clocking for details.

43.1.5 Compare (programmable timer) facilities

There are four programmable timer facilities which share a common counter with the capture input. Each of the four compare registers PWM_nCOMPAREVAL in the module can be set to a value 0x0000 0000 to 0xFFFF FFFF.

When the counter in register PWM_CAPTURECOUNT reaches the value of register PWM_nCOMPAREVAL, the following happens:

- An interrupt is generated provided the CMPIEn bit of the PWM_INTENABLE register is set to 1.
- Bit CMPn of register PWM_INTSTATUS becomes 1, and can be reset by writing 1 to bit CMPIAn of register PWM_INTACK.

The counter is not stopped nor reset by any of these events. See Section 43.1.6 for details of the counter.

43.1.6 Capture/compare counter, prescaling and clocking

The capture/compare counter is clocked from the prescaled system (200 MHz nominal) clock, and is common to all capture and compare functions. The prescaling factor, and therefore the period represented by one count, is determined by the value of field CPTCLKVALUE in register PWM_CONTROL. The factor can be from 1 to 32.

The counter (in register PWM_CAPTURECOUNT) is enabled by setting the CPTEN bit of the PWM_CONTROL register to 1. When it is disabled (CPTEN is 0), none of the capture or compare functions work. PWM_CAPTURECOUNT, like PWM_COUNT, can be read or written at any time.

When the capture/compare counter reaches its maximum count of 0xFFFF FFFF, it wraps round to count up from zero again.

43.2 PWM-Timer2

43.3 Overview

The PWM-Timer2 module serves the following purposes:

- generate - in addition to the same range as that offered by the C1_PWM - very low PWM frequencies (typically 1 to 8 KHz for a 100 MHz master clock).
- enable generation of PWM waveforms completely independent from the ST20-C105 timer tick.
- enable generation of an interrupt on a periodic basis with little software intervention, with a very large range of possible speeds - a few microseconds to over 100 ms.

The PWM-Timer2 is based on two completely independent counters with associated prescalers and duty cycle control, each being usable either to generate an interrupt or generate a PWM waveform (or both if desired interrupt period and PWM period are identical). In the rest of this chapter, the registers, signals and other hardware pertaining to the first PWM/Timer are indexed "_A", those pertaining to the second PWM/Timer are indexed "_B".

43.3.1 Programmable PWM function

The system clock (100 MHz nominal) is first prescaled (divided) by a factor 1 to 65536 according to control bits **PWMClkValue_A/B[15:0]** and triggers an 8 bit counter (from 0 to 255). The counter and prescaler may be stopped by writing a '0' into the **PWMEnable_A/B** bit of the corresponding **Control** register. While disabled, the value in the counter may be read or written as register **PWMCount_A/B[7:0]**. Every 256 counts, the counter triggers the output block to start new pulses.

The prescaler consists of a modulo counter counting from 0 to PWMClkValue[15:0] and then rolling over to zero

For example:
clkvalue = 0x0000 => divide by 1 (that is, generated clock = system clock).
clkvalue = 0x0002 => divide by 3 (modulo-3 count, from 0 to 2)
clkvalue = 0x000F => divide by 16 (modulo-16 count, from 0 to 15)
clkvalue = 0x01FF => divide by 512 (modulo-512 count, from 0 to 511)
clkvalue = 0xFFFF => divide by 65536 (modulo-65536 count, from 0 to 65535)

When the enable bit for that module is low the prescaler is disabled and reset. It may start upon the next rising edge following the setting of the enable bit.

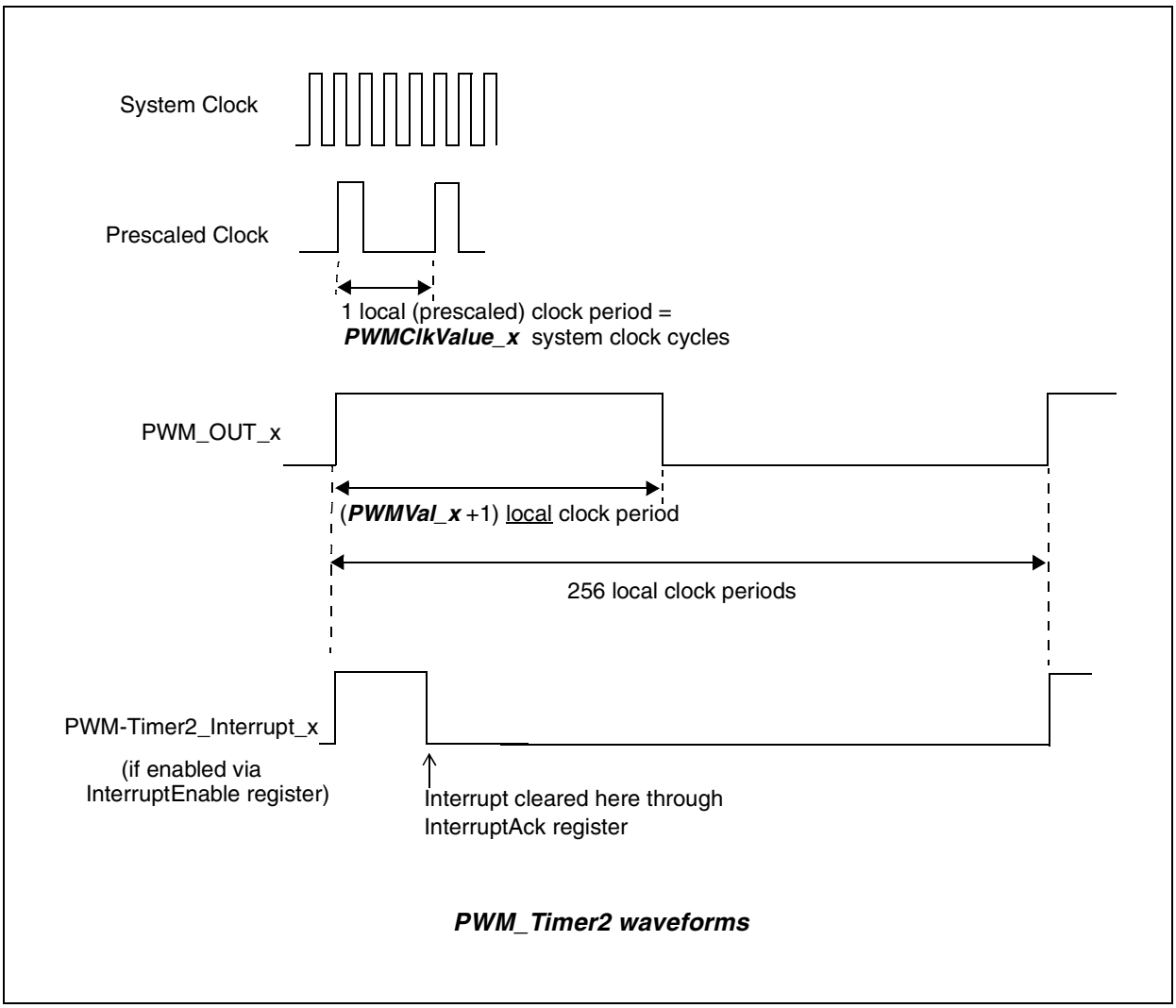
A new PWM pulse is started (pwm_out_A/B rises to high level) every time the 0-to-255 counter rolls over. It goes back to low level after the number of cycles programmed in register **PWMVal_A/B[7:0] + 1**. Therefore PWMVal controls the duty cycle of the PWM signal. For example, if the value programmed is 127 (that is, half the maximum possible) the resulting output will be 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 will be constantly high. The length of the pulse is updated only upon the last count, so that the pulse currently executing will always finish before a pulse of different width is output. Following reset, PWMOut_A/B is low.

The figure below shows the waveforms that are generated.

43.3.2 Periodic interrupt generation

Similarly, an interrupt request is raised (int_out_A/B goes high) when the 0-to-255 rolls over, provided bit PWMIntEn_A/B in the INTERRUPTENABLE register is set. The interrupt request will remain active until cleared by a write access to the corresponding bit in the INTERRUPTACK register (or the interrupt gets disabled). The status of the interrupt is available to software via register INTERRUPTSTATUS.

Figure 68: PWM-Timer2 periodic interrupt generation



44 PWM and counter module registers

The base address for C1_PWM referred to as *C1PWMBaseAddress* is:
0x3000 3000.

The base address for PWM-Timer2 referred to as *PWMTimer2BaseAddress* is:
0x2001 0000.

Note: The PWM-Timer2 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 nor interrupts) should be transposable as is to run on PWM-Timer2.

C1_PWM registers between offset 0x00 and 0x58 that are not described are all reserved.

Table 95: C1_PWM register summary

Register	Description	Offset	Type
PWM_nVAL	PWM n pulse width	0x00 (*n* = 0), 0x0C (*n* = 3)	R/W
PWM_CAPTUREVAL	PWM capture value	0x10	RO
PWM_nCOMPAREVAL	PWM n compare value	0x20 (*n* = 0) to 0x2C (*n* = 3)	R/W
PWM_CAPTUREEDGE	PWM capture event definition	0x30	R/W
PWM_CONTROL	PWM control register	0x50	R/W
PWM_INTENABLE	PWM interrupt enable	0x54	R/W
PWM_INTSTATUS	PWM interrupt status	0x58	RO
PWM_INTACK	PWM interrupt acknowledge	0x5C	WO
PWM_COUNT	PWM output counter	0x60	R/W
PWM_CAPTURECOUNT	PWM capture/compare counter	0x64	R/W

Table 96: C1_PWM register summary

Register	Description	Offset	Type
PWMVAL_A	PWM Value A	0x00	R/W
PWMVAL_B	PWM Value B	0x08	R/W
CONTROL_A	Control A	0x50	R/W
CONTROL_B	Control B	0x58	R/W
PWMCOUNT_A	PWM Count A	0x60	R/W
PWMCOUNT_B	PWM Count B	0x68	R/W
INTERRUPTENABLE	Interrupt enable	0x70	R/W

44.1 C1_PWM REGISTERS

PWM_nVAL PWM n pulse width

Address: *C1PWMBaseAddress* + 0x00 (PWM_0VAL) and 0x0C (PWM_3VAL)

Type: R/W

Reset: Undefined

Description: These registers hold the counter values, which are used to determine the width of the pulse generated on the output pin C1_PWM_OUT0 and on signal C1_PWM_OUT3 used as timer tick by the ST20 CPU (for time slicing).

PWMn pulse width = (PWMnVAL + 1) x prescaled clock period.
If PWMnVAL is 255 then PWMn is always 1 (that is, it does not go low). PWM_1VAL and PWM_2VAL have no associated output pin.

PWM_CAPTUREVAL PWM 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
PWM_CAPTUREVAL																															

Address: *C1PWMBaseAddress* + 0x10

Type: RO

Reset: Undefined

Description: This capture value register holds the 32-bit counter value at the time of the last event occurring at the corresponding PWM_CAPTURE0 pin. When an input event occurs on input pin PWM_CAPTURE0, the value of the counter in register PWM_CAPTURECOUNT at that time is captured in register PWM_CAPTUREVAL. The value can be any 32-bit value.
When an input event occurs, an interrupt is generated if the register bit CPTIEn (register PWM_INTENABLE) is set to 1. Register bit CPT (PWM_INTSTATUS) becomes 1, and can be reset by writing 1 to register CPTIA (PWM_INTACK).
The counter is not stopped or reset by any of these events.

PWM_nCOMPAREVAL PWM n compare value

Address: *C1PWMBaseAddress* + 0x20 (PWM0COMPAREVAL) to 0x2C (PWM3COMPAREVAL)

Type: R/W

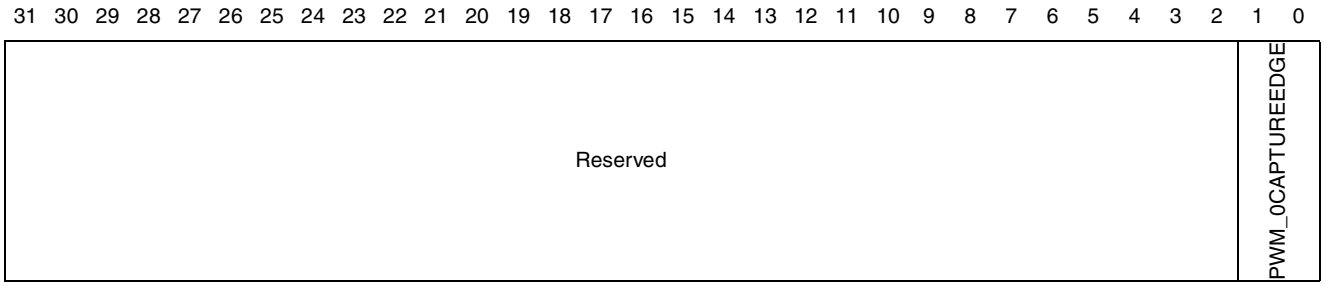
Reset: Undefined

Description: Each of the four compare registers PWM_nCOMPAREVAL in the module can be set to any 32-bit value. When the counter in register PWM_CAPTURECOUNT reaches the value of register PWM_nCOMPAREVAL, the following happens:

- An interrupt is generated if the register bit CMPIEn (register PWM_INTENABLE) is set to 1. Register bit CMPn (PWM_INTSTATUS) becomes 1, and can be reset by writing 1 to register bit CMPIAn (PWM_INTACK).

The counter is not stopped or reset by any of these events.

PWM_CAPTUREEDGE PWM capture event definition



Address: *C1PWMBaseAddress* + 0x30

Type: R/W

Reset: Undefined

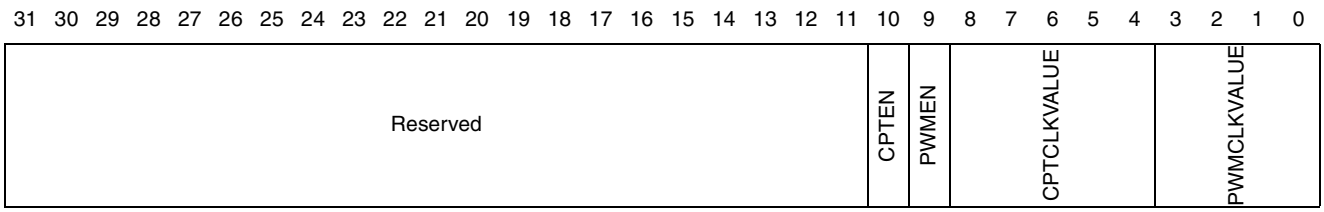
Description: The code in register PWM_CAPTUREEDGE defines what constitutes an event on input pin PWM_CAPTURE0. Possible events are rising edge, falling edge, both or neither (in other words, disabled).

[31:2] **Reserved**

[1:0] **PWM_CAPTUREEDGE**

01: Capture on rising edge
10: Capture on falling edge
11: Capture on rising or falling edge
00: Capture disabled

PWM_CONTROL PWM control register



Address: *C1PWMBaseAddress* + 0x50

Type: R/W

Reset: Undefined

Description:

[31:11] **Reserved**

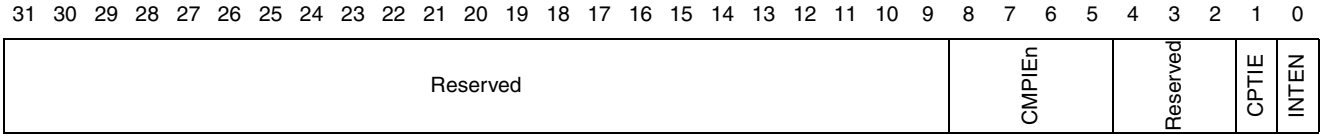
[10] **CPTEN**
Enables capture/compare counter when 1

[9] **PWMEN**
Enables PWM counter when 1

[8:4] **CPTCLKVALUE**
Capture/COmpare clock prescale factor, 0 to 31 (divide clock by value+1)

[3:0] **PWMCLKVALUE**
PWM clock prescale factor 0 to 15 (divide clock by value+1)

PWM_INTENABLE PWM interrupt enable



Address: *C1PWMBaseAddress* + 0x54

Type: R/W

Reset: Undefined

Description:

[31:9] **Reserved**

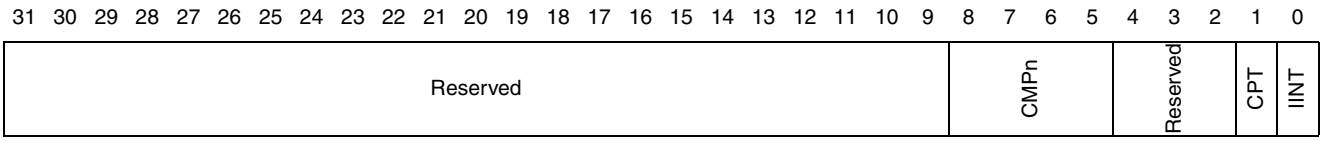
[8:5] **CMPIEn**
Compare 3 to 0 interrupt enable (enabled if 1)

[4:2] **Reserved**

[1] **CPTIE**
Capture interrupt enable (enabled if 1).

[0] **INTEN**
PWM counter overflow interrupt enable (enabled if 1).

PWM_INTSTATUS PWM interrupt status



Address: *C1PWMBaseAddress* + 0x58

Type: RO

Reset: Undefined

Description:

[31:9] **Reserved**

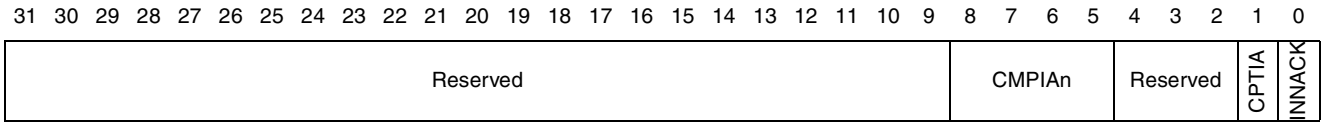
[8:5] **CMPn**
Compare 3 to 0 interrupt status bits (interrupt if set, meaning capture/compare counter has recahed programmed trigger value).

[4:2] **Reserved**

[1] **CPT**
Capture interrupt status bit (interrupt if set)

[0] **INT**
PWM interrutp status bit: If set, PWM counter has overflowed

PWM_INTACK PWM interrupt acknowledge



Address: *C1PWMBaseAddress* + 0x5C
Type: WO
Reset: Undefined
Description:

[31:9] **Reserved**

[8:5] **CMPIAn**
Compare 3 to 0 interrupt acknowledge bits: write 1 to reset associated status bit

[4:2] **Reserved**

[1] **CPTIA**
Capture interrupt acknowledge: write 1 to reset associated status bit

[0] **INTACK**
Interrupt acknowledge: write 1 to reset PWMInt to 0.

PWM_COUNT PWM output counter

Address: *C1PWMBaseAddress* + 0x60
Type: R/W (but see text)
Reset: Undefined
Description: PWM output counter. The counter (in register PWM_COUNT) is enabled by setting register bit PWMENABLE (PWM_CONTROL) to 1. When it is disabled (PWMENABLE = 0), pins PWM[2:0] are forced low. PWM_COUNT is writable at any time but can have a synchronization latency.

PWM_CAPTURECOUNT PWM capture/compare counter

Address: *C1PWMBaseAddress* + 0x64
Type: R/W
Reset: Undefined
Description: This register holds the shared capture/compare counter used by all the capture and compare functions.
The capture/compare counter is clocked from the prescaled comms clock. The prescaling factor, and therefore the period represented by one count, is determined by the value of field CAPTURECLKVALUE in register PWM_CONTROL. The factor can be from 1 to 32.
The counter is enabled by setting register bit CAPTUREENABLE (PWM_CONTROL) to 1. When it is disabled (CAPTUREENABLE = 0), none of the capture or compare functions work. PWM_CAPTURECOUNT can be read or written at any time.
When the capture/compare counter reaches its maximum count of 0xFFFF FFFF, it wraps round to count up from zero again.

44.2 PWM_TIMER-2 REGISTERS

PWMVAL_A PWM Value A

31	30	29	28	27	26	25	24	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																								PWMVAL_A[7:0]							

Address: *PWMTimer2BaseAddress* + 0x00

Type: R/W

Reset:

Description:

[31:8] **Reserved**

[7:0] **PWMVAL_A[7:0]**: PWM_A reload value, defining the duty cycle of output PWM_OUT_A as follows: PWMVal_A + 1 is the number of local (prescaled) clock cycles for which PWM_OUT_A will be high in a period of 256 local (prescaled) clock cycles.

PWMVAL_B PWM Value B

31	30	29	28	27	26	25	24	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																								PWMVAL_B[7:0]							

Address: *PWMTimer2BaseAddress* + 0x08

Type: R/W

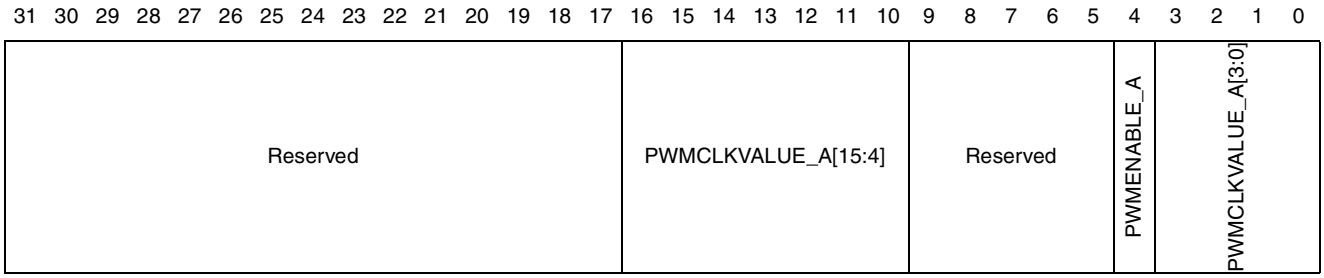
Reset:

Description:

[31:8] **Reserved**

[7:0] **PWMVAL_B[7:0]**: PWM_B reload value, defining the duty cycle of output PWM_OUT_B as follows: PWMVAL_B + 1 is the number of local (prescaled) clock cycles for which PWM_OUT_B will be high in a period of 256 local clock cycles.

CONTROL_A Control A



Address: *PWMTimer2BaseAddress* + 0x50

Type: R/W

Reset:

Description:

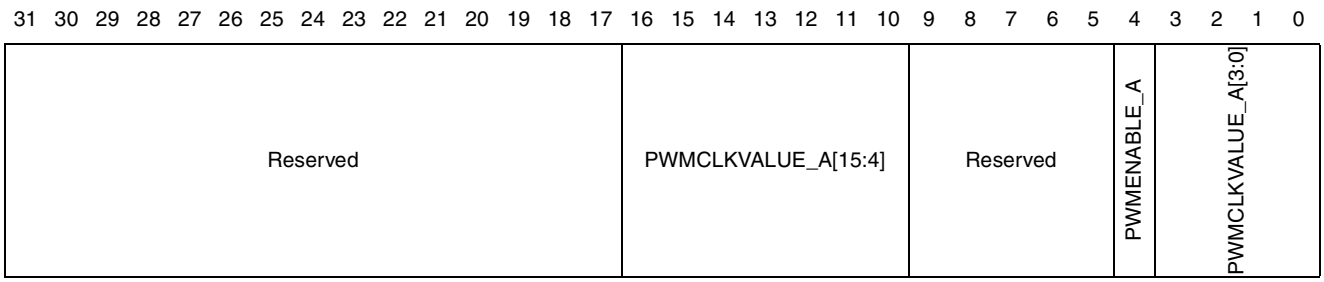
[31:17] **Reserved**

[16:5] **PWMCLKVALUE_A[15:4]**: High order bits of the parameter that defines the period of the local prescaled clock for PWM/Timer A. The local clock enable signal will be generated upon the prescale counter reaching PWMCLKVALUE_A[15:0].

[4] **PWMENABLE_A**:
When '1', prescale counter A and PWM Counter A of the PWM-Timer2 are enabled. When '0', prescale counter A is **cleared** and PWM Counter A is **stopped**.

[3:0] **PWMCLKVALUE_A[3:0]**: Low order bits of the parameter that defines the period of the local prescaled clock: The local clock enable signal will be generated upon the prescale counter reaching PWMCLKVALUE[15:0].

CONTROL_B Control B



Address: *PWMTimer2BaseAddress* + 0x58

Type: R/W

Reset:

Description:

[31:17] **Reserved**

[16:5] **PWMCLKVALUE_B[15:4]**: High order bits of the parameter that defines the period of the local prescaled clock: The local clock enable signal will be generated upon the prescale counter reaching PWMCLKVALUE_B[15:0].

[4] **PWMENABLE_B**:
When '1', prescale counter B and PWM Counter B of the PWM-Timer2 are enabled. When '0', prescale counter B is ***cleared*** and PWM Counter B is ***stopped***.

[3:0] **PWMCLKVALUE_B[3:0]**: Low order bits of the parameter that defines the period of the local prescaled clock fro PWM/Timer B: The local clock enable signal will be generated upon the prescale counter reaching PWMCLKVALUE_B[9:0].

PWMCOUNT_A PWM Count A

31	30	29	28	27	26	25	24	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																								PWMCOUNT_A							

Address: *PWMTimer2BaseAddress* + 0x60

Type: R/W (see text)

Reset:

Description:

[31:8] **Reserved**

[7:0] **PWMCOUNT_A[7:0]**: This offers direct access to counter A of the PWM-Timer2.
Write access (to preset a value for example) is only possible when PWM/Timer A is disabled (PWMENABLE_A = '0').

PWMCOUNT_B PWM Count B

31	30	29	28	27	26	25	24	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																								PWMCOUNT_B							

Address: *PWMTimer2BaseAddress* + 0x68

Type: R/W (see text)

Reset:

Description:

[31:8] **Reserved**

[7:0] **PWMCOUNT_B[7:0]**: This offers direct access to counter B of the PWM-Timer2.
Write access (to preset a value for example) is only possible when PWM/Timer B is disabled (PWMEnable_B = '0').

INTERRUPTENABLE 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																								PWMCOUNT_B							

Address: *PWMTimer2BaseAddress* + 0x70

Type: R/W

Reset:

Description:

[31:2] **Reserved**

[1] **PWMINTEN_B**
0: Interrupts from PWM/Timer B are disabled.
1: PWMT/Timer B generates a high level interrupt as counter B rolls over.

[0] **PWMINTEN_A**
0: Interrupts from PWM/Timer A are disabled.
1:PWMT/Timer A generates a high level interrupt as counter A rolls over.

45 Transport stream merger and router

45.1 Overview

The STi7710 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 a source ID and a 42-bit time stamp.

The transport stream merger 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 / sec 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 Mb/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 69: System overview

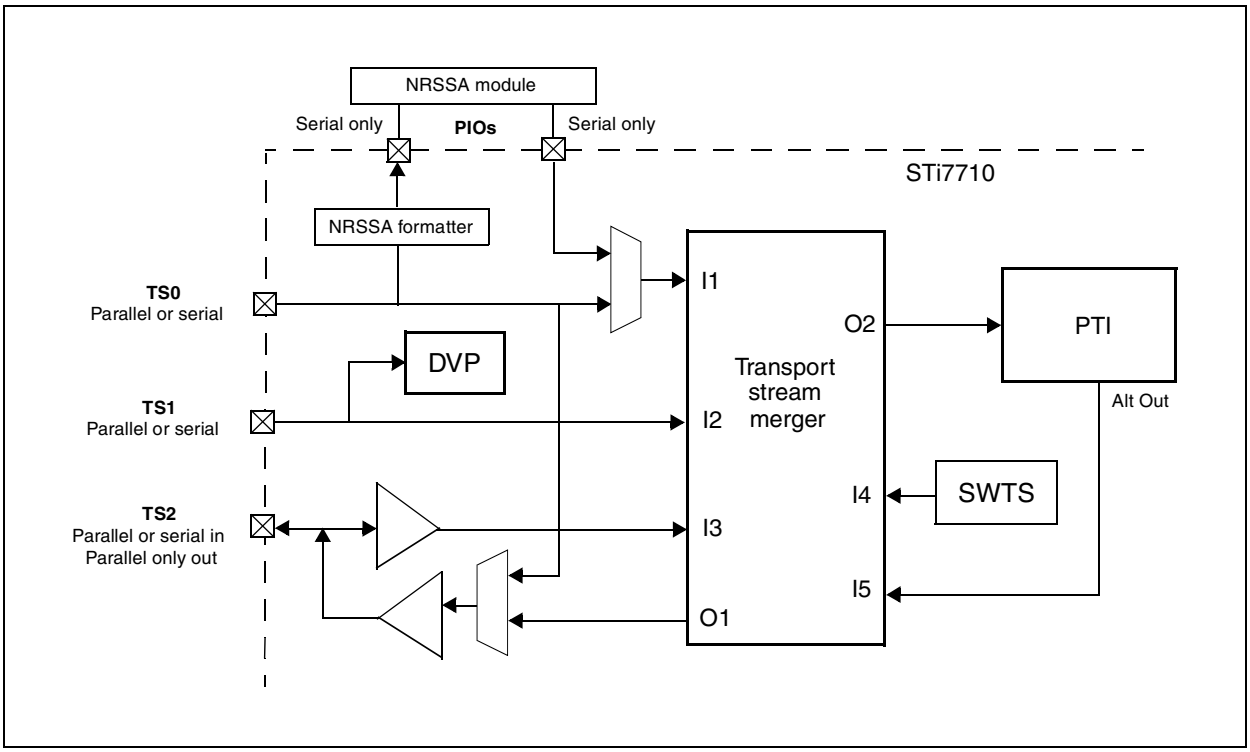


Figure 69 shows the three live input streams delivered using three sets of transport stream interfaces from the pads. Another stream is delivered from the return channel on the PTI (PTI alternative output). The transport stream merger supports parallel or serial streams on the three live stream interfaces.

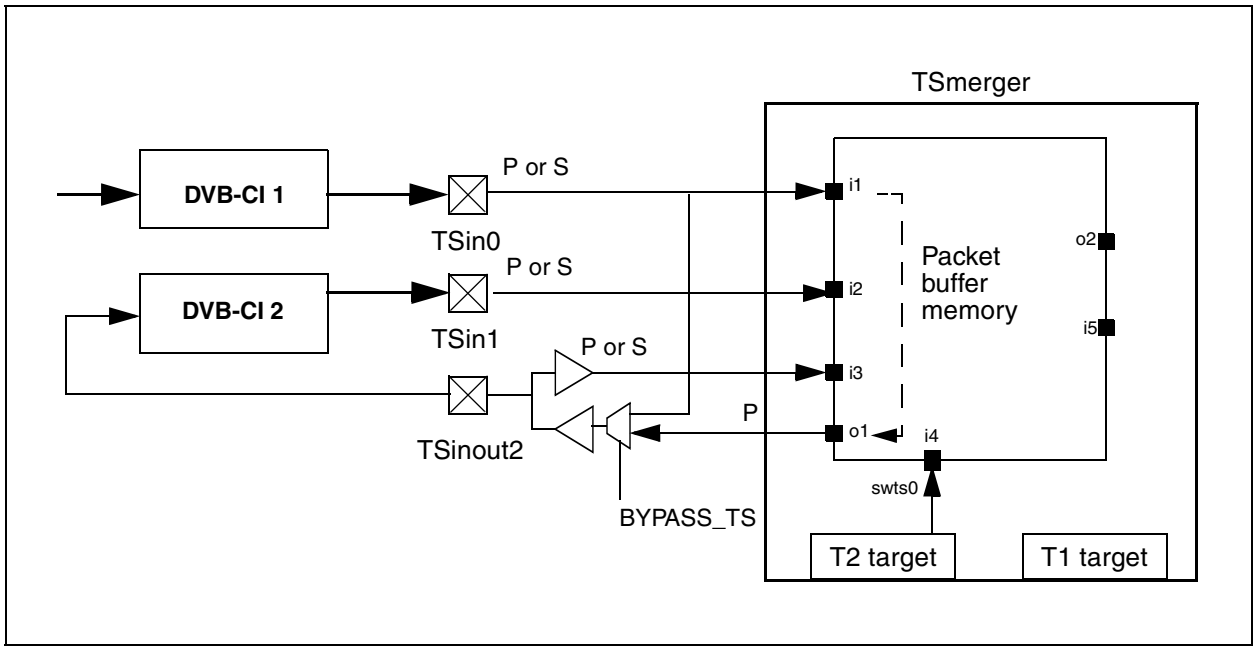
On the STi7710, the input pins TS1 are shared pins. This interface can be used for inputting transport streams for processing by the PTI or for inputting digitized video (D1) conforming to ITU-R BT 656 for processing by the Digital Video Port (DVP).

Multiplexing and formatting performed around the TSMerger itself are controlled by software via System Configuration registers. For details, see Chapter 6: *Configuration registers on page 60*.

45.2 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 STi7710 supports a chain of two DVB-CI modules.

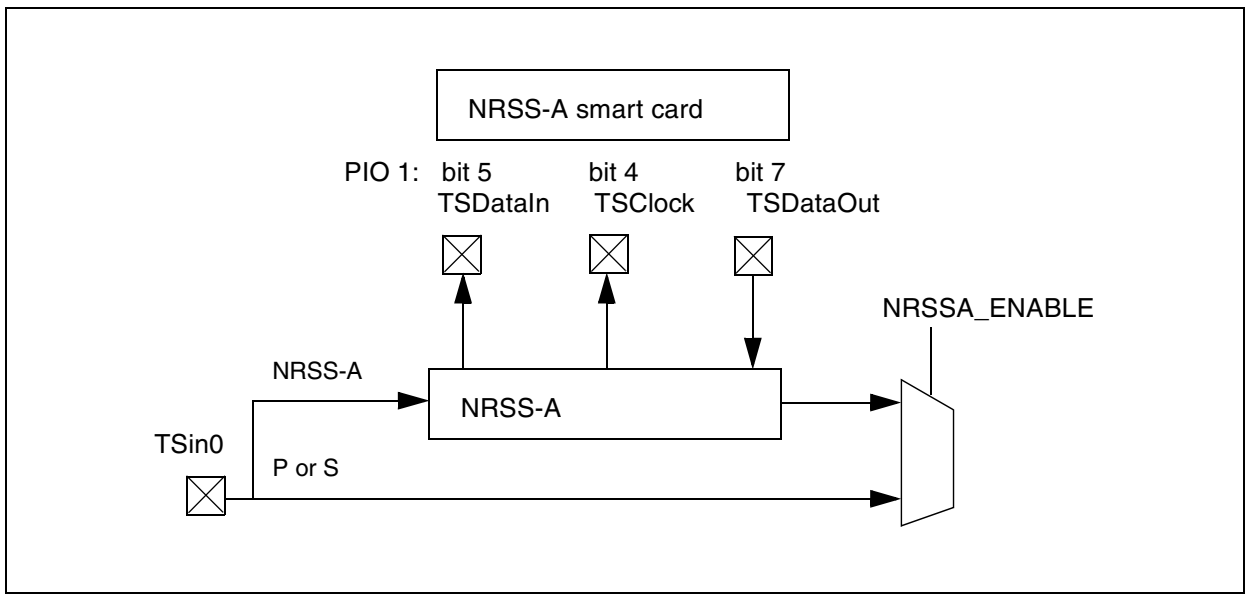
Figure 70: DVB-CI module chain



45.3 NRSS-A interface

When receiving an ATSC stream on TS0, the STi7710 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 STi7710 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 71: NRSS-A interface



45.4 Input examples

Figure 72: Basic ATSC STB with NTSC input

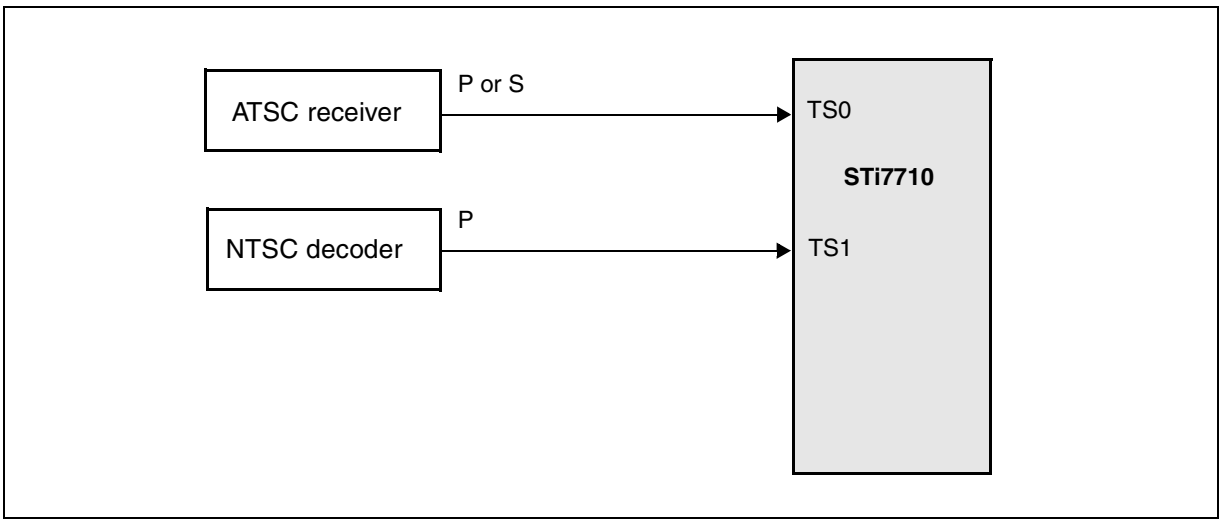


Figure 73: Satellite/cable system

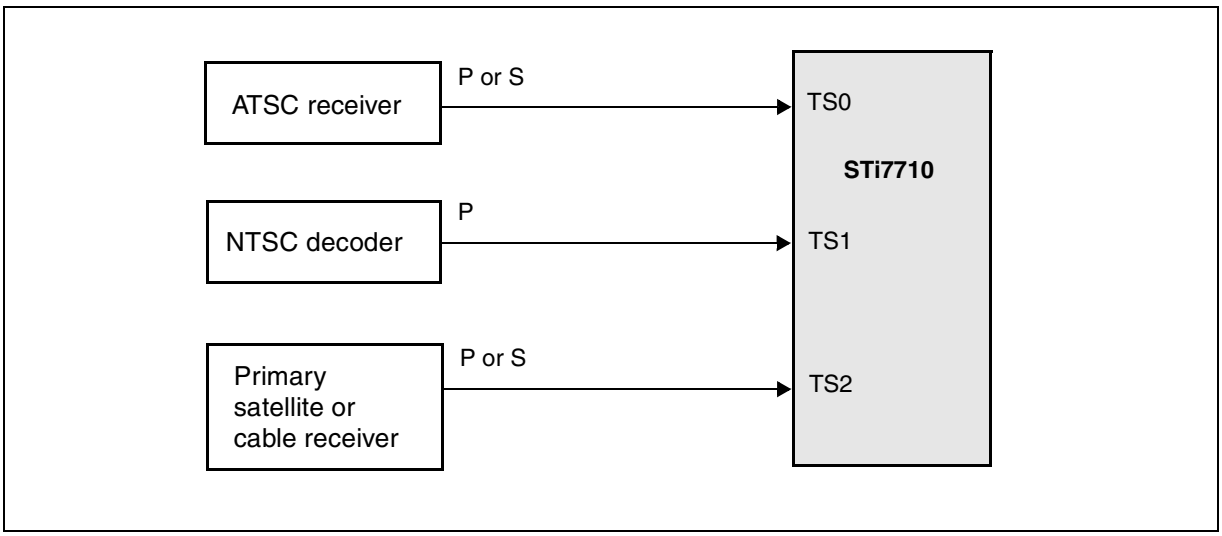


Figure 74: Multi-tuner DVR satellite/cable system with NTSC and ATSC receivers

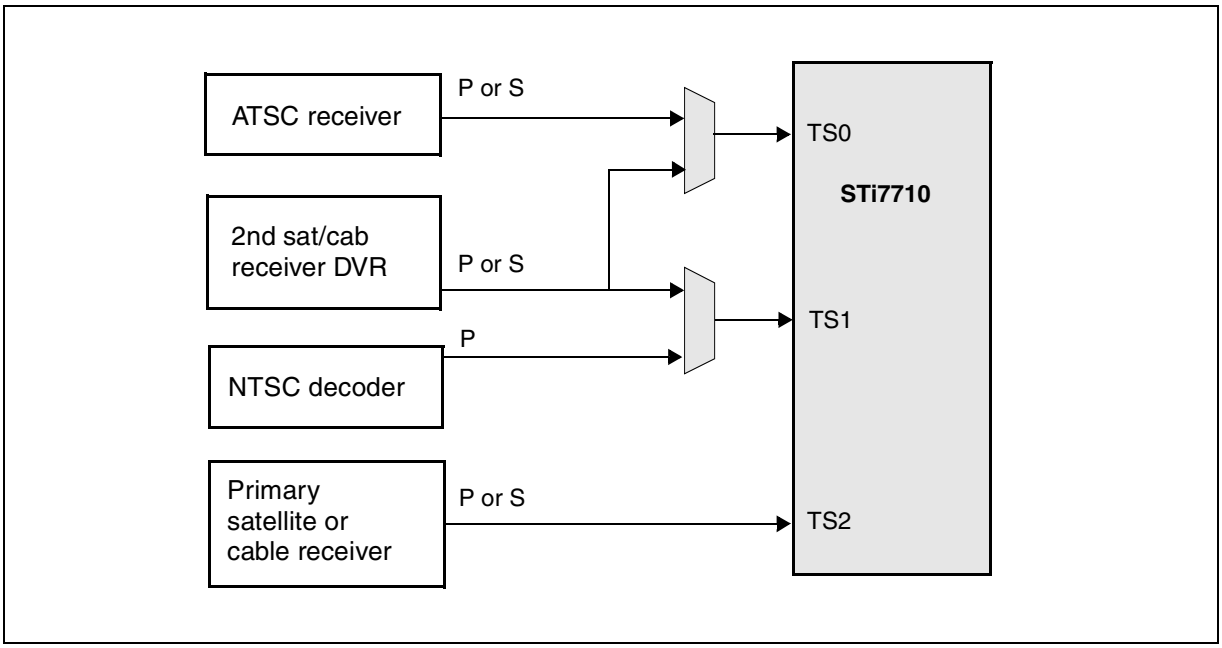


Figure 75: Japan (with 1394 and optional DVR)

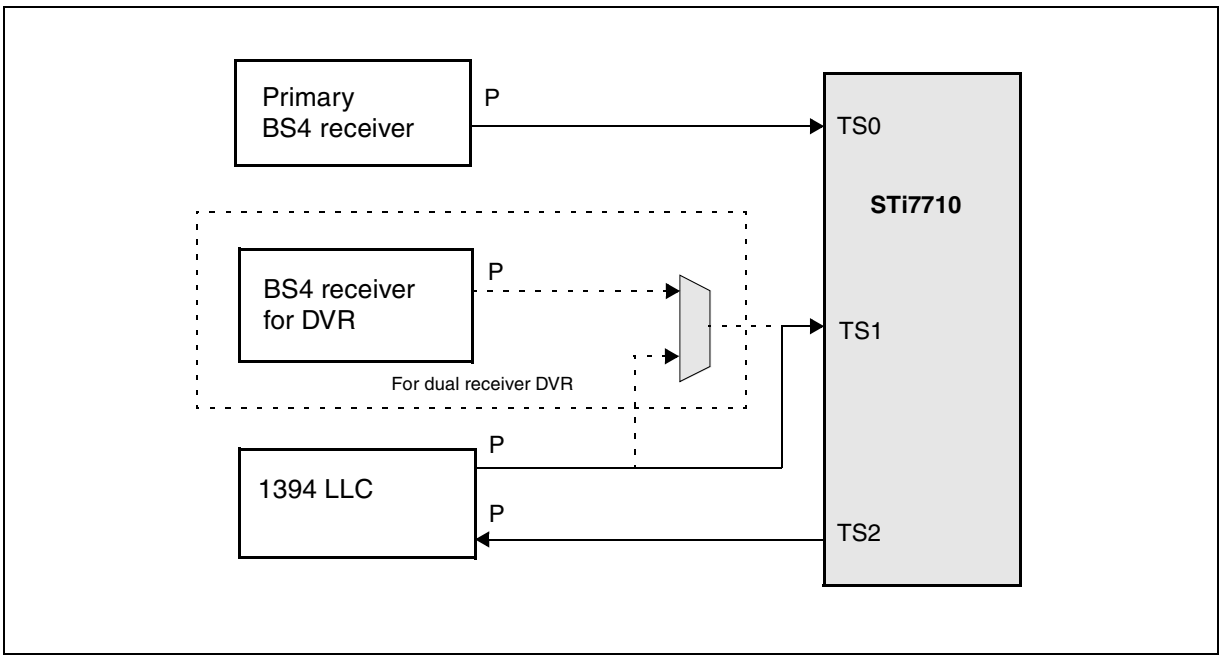


Figure 76: HD Open Cable with D1

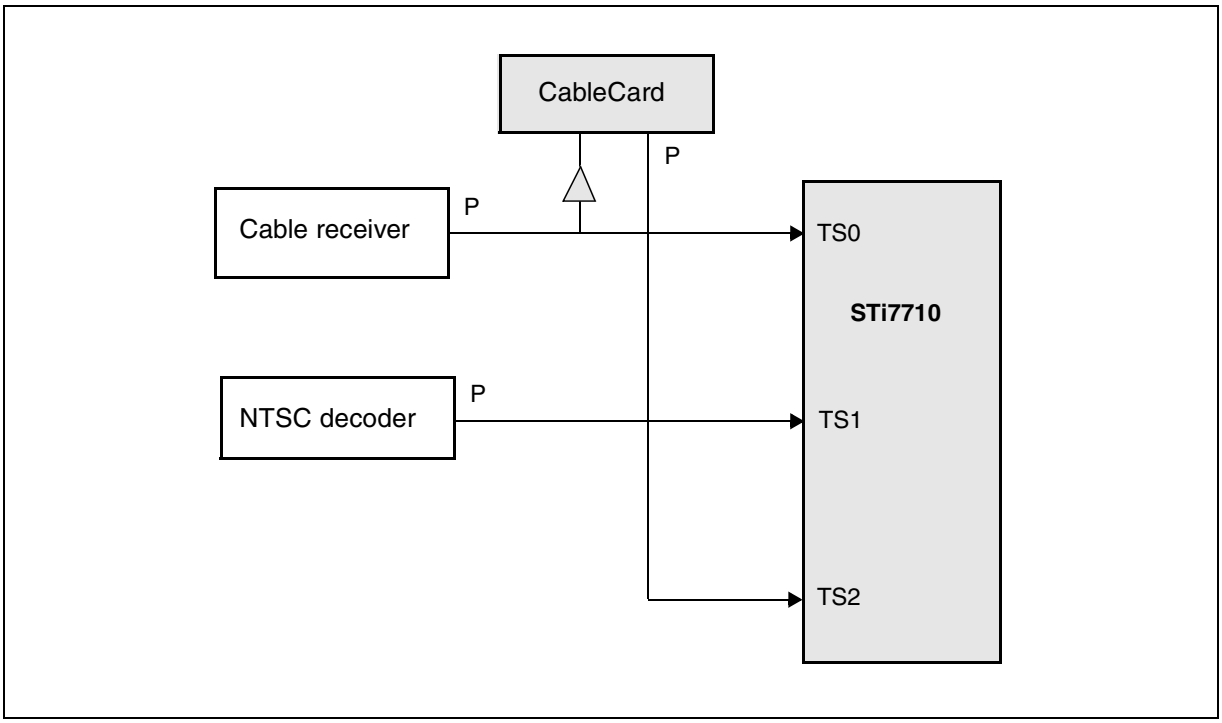
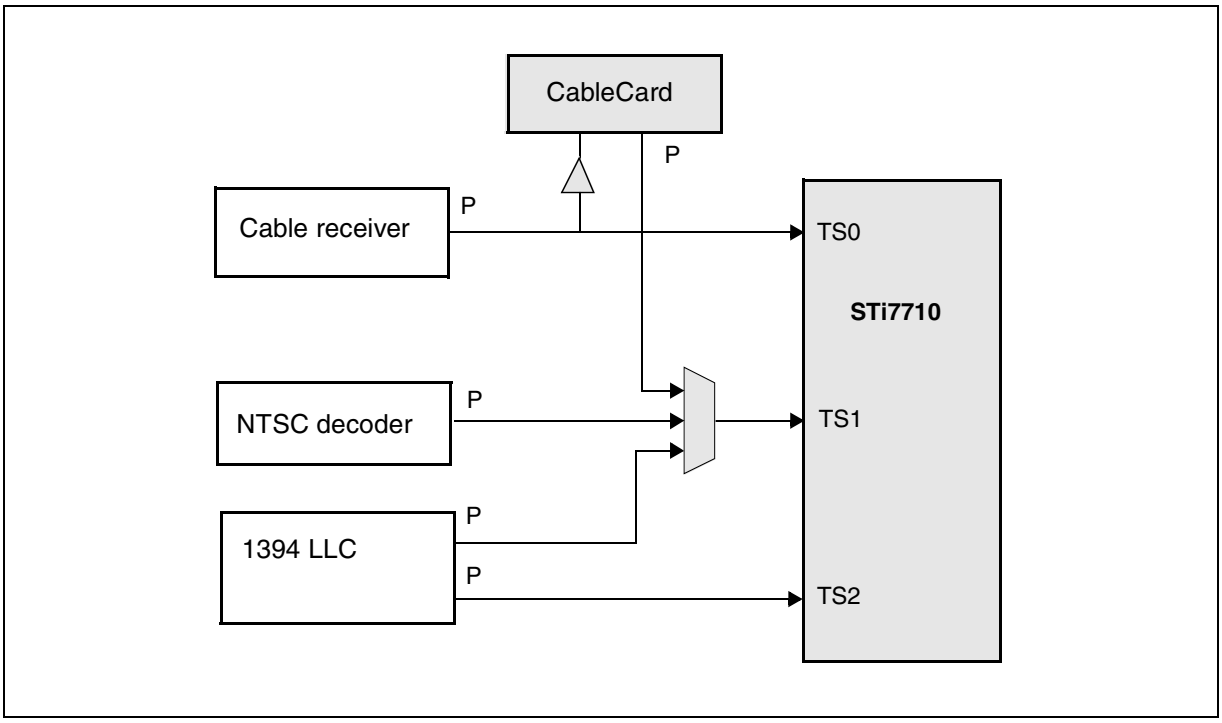
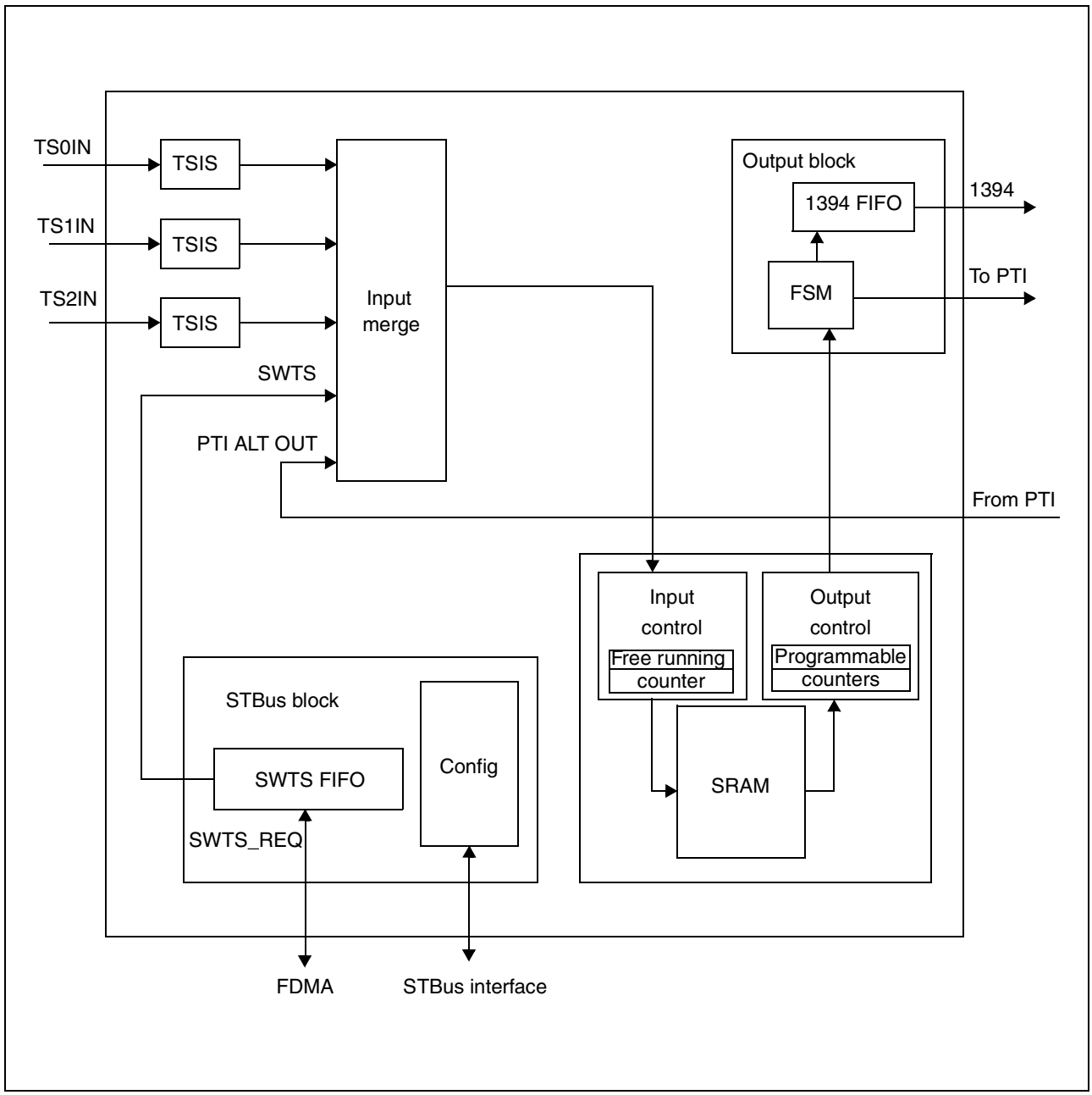


Figure 77: HD Open Cable with D1 and full duplex 1394



45.5 Architecture

Figure 78: Block diagram



45.6 Transport stream formats

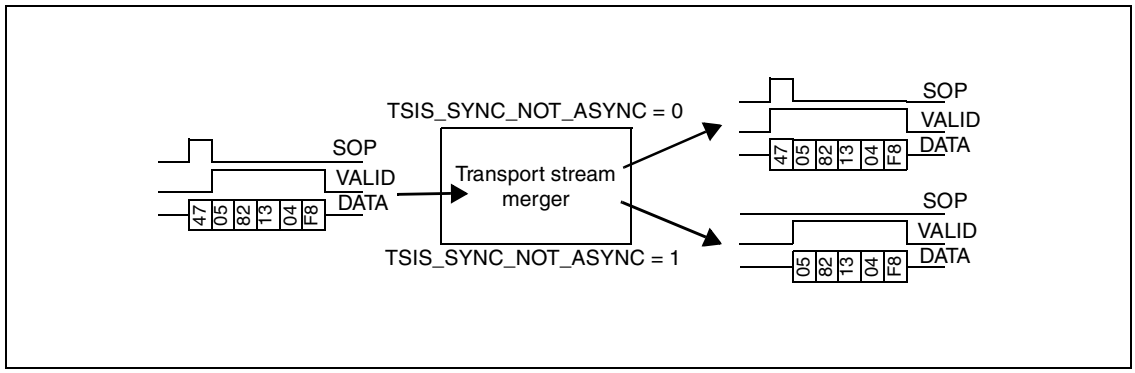
Transport streams arrive in many forms which the transport stream merger must deal with correctly. In order to do this each stream has a number of configuration bits which must be set correctly in order to attain the desired outcome. In the following sections a number of scenarios describe how the transport stream merger stream configuration bits affect transport stream merger functionality.

45.6.1 Off-chip live streams from FECs

Four bits in the TSM_STREAMn_CFG register control the configuration.

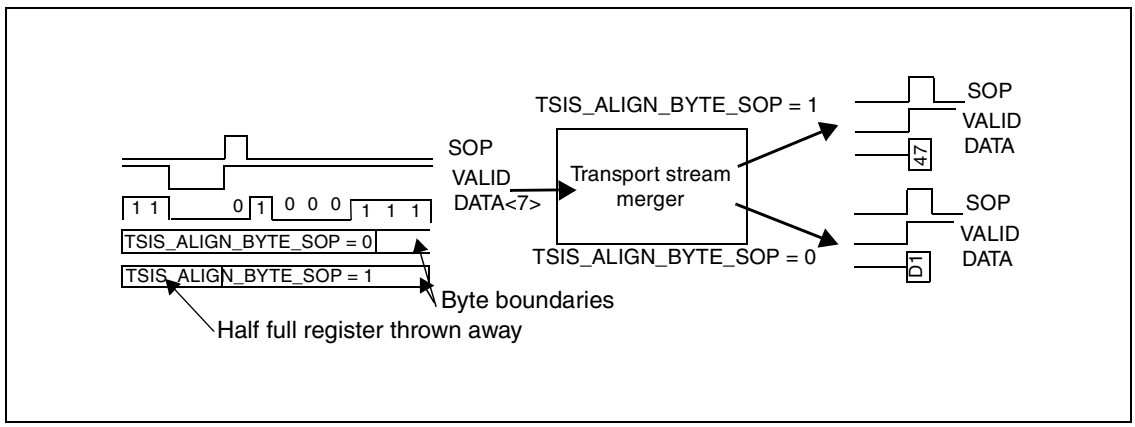
- **Serial or parallel**: The default is serial.
- **Synchronous or asynchronous data**: TSIS_SYNC_NOT_ASYNC is 1 if the data is synchronous, that is valid high for all bytes in the packet (including the start of packet byte). If the transport stream has the first byte of the packet with valid low and start of packet high, use asynchronous mode to allow the start of packet signal to be registered.

Figure 79: 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.

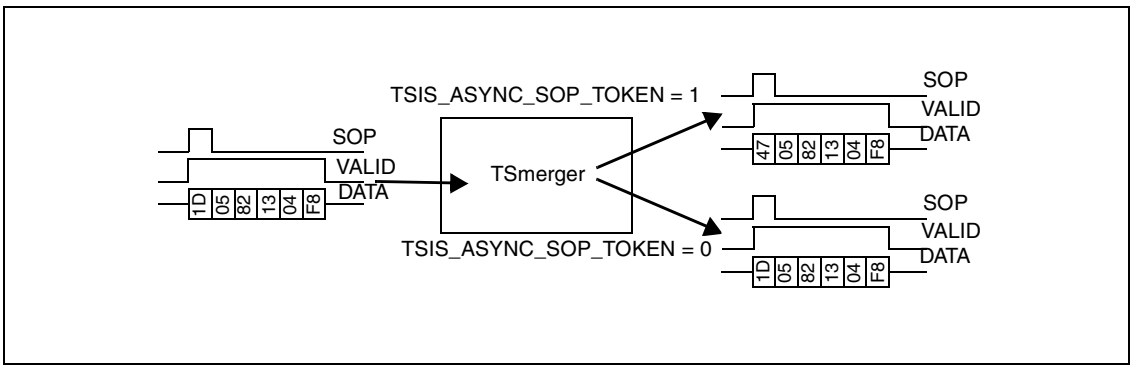
Figure 80: TSIS_ALIGN_BYTE_SOP functionality



- **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 81: TSIS_ASYNC_SOP_TOKEN functionality



- **BYTECLK inversion**: When INVERT_BYTECLOCK is high, the BYTECLK goes through an inverter cell.

45.6.2 SWTS

SWTS playback data is sent across the STBus and into the transport stream merger using the STBus target port. The data is pure data and no other information is given 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 transport stream 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 transport stream 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 SWTSn_AUTO_PACE 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 SWTS_AUTO_PACE 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 transport stream merger for PTI return data. The user must select which PTI to connect this to and this is handled by the PTI_ALT_OUT_AUTO_PACE bit in TSM_PTI_ALT_OUT_CFG.

45.7 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 transport stream 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.

45.8 Transport stream inputs

To use the transport stream 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 45.8.1 describes how this happens.

45.8.1 Packet tags

When a packet is injected into the transport stream 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 84. The structure of the tagging bytes is shown in Figure 82. 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 97. When sending live data through the transport stream 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 97: 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

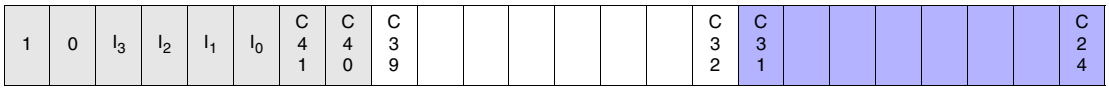
Table 97: ID encoding

Stream number	Stream name	I_3	I_2	I_1	I_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 45.9.2: *Delaying outgoing packets on page 369*.

Figure 82: Tag bytes

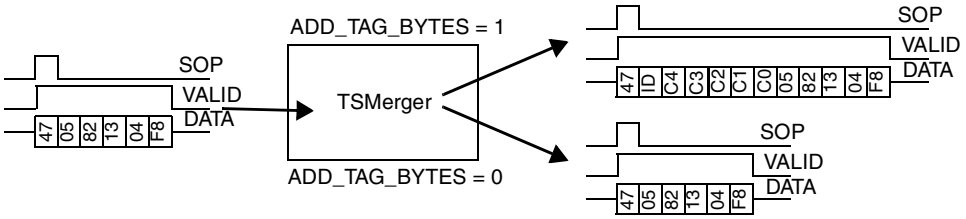
Figure 83:



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 84: 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 transport stream 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 transport stream 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.

45.8.2 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 PACKET_LENGTH (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.

45.9 Transport stream output

The output block of the transport stream merger uses the TSM_1394_DESTINATION and TSM_PTI_DESTINATION registers to decide where source streams are sent.

45.9.1 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 transport stream 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_REMOVE_TAGGING_BYTES to 1.

45.9.2 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_COUNTERn AUTOMATIC_COUNTER should be set to 0. The TSM_STREAMn_STATUS COUNTER_VALUE holds the currently waiting packet header counter value so that the programmable counter can be set to this figure. When AUTOMATIC_COUNTER = 1 and COUNTER_INITIALIZED = 0 then the programmable counter is set to the value of the waiting packet header counter value. Once set, COUNTER_INITIALIZED is set to 1. To reinitialize the counter at any point the COUNTER_INITIALIZED 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 COUNTER_INITALIZED or AUTOMATIC_COUNTER.

If the counters are not required (for example there is no counter value to match in the header) then the COUNTER_INC value should be set to 0. When playing back in x1 mode, the counter is incremented by 1 every 27 MHz. To playback at twice the speed, the counter should be incremented twice every 27 MHz. Set the playback speed in the COUNTER_INC field of the counter register.

45.9.3 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

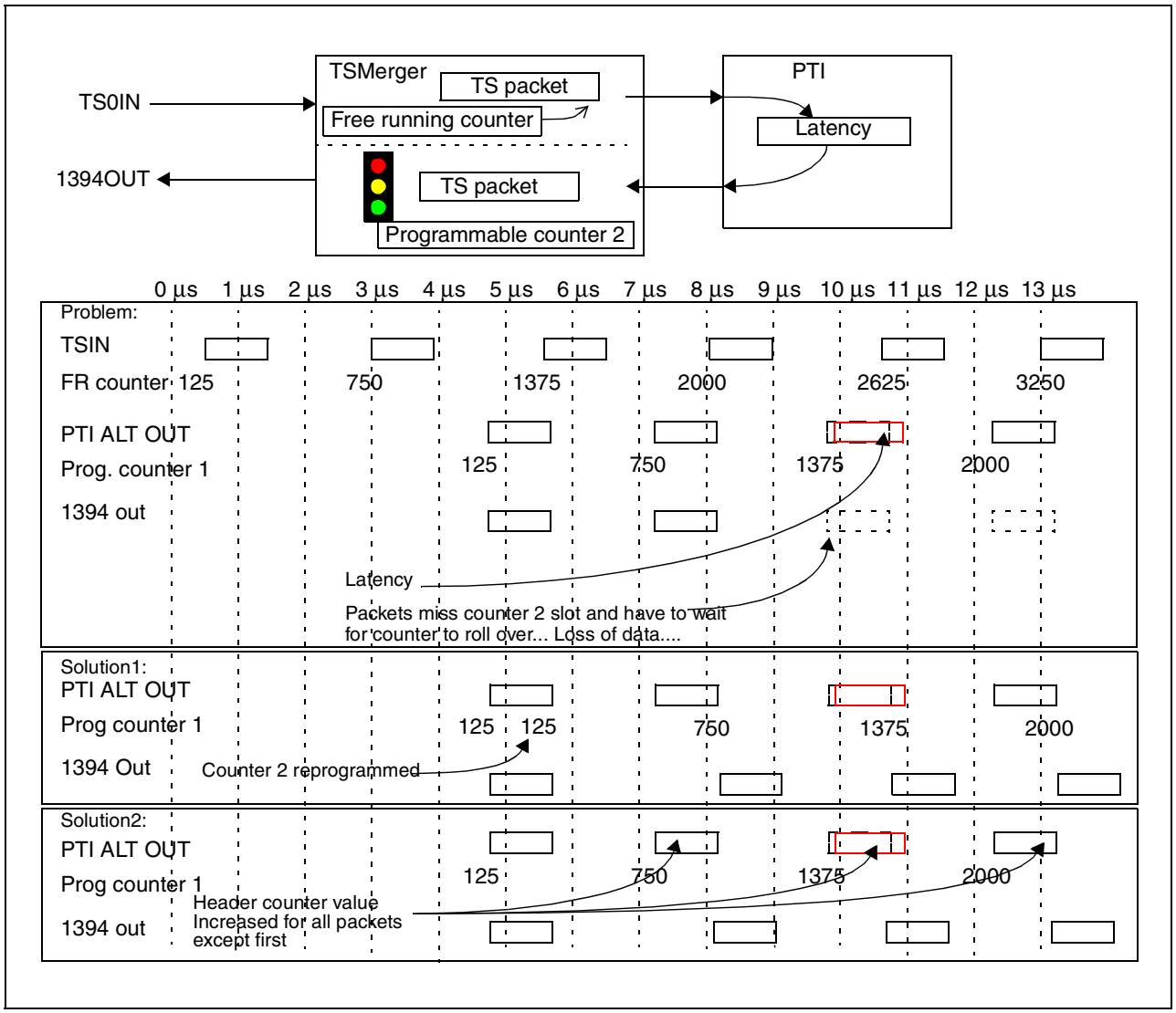
1. A live stream is injected using TS0IN and tagging bytes are added to each packet header. These packets are sent to the PTI where they are processed and returned back to the transport stream merger to be sent dejittered to the 1394 ports.
2. Counter 2 is autoinitialized to the first packet header counter 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 85: Breaking the dejittering mechanism



45.9.4 Stream status

Each stream has a status register (TSM_STREAMn_STATUS) with the following fields:

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.

INPUTFIFO_OVERFLOW

INPUTFIFO_OVERFLOW 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.

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_OVERFLOW scenario is that a packet may be lost. No half packets arrive at the PTI.

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.

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.

46 Transport stream merger registers

Addresses are provided as:

TSMergerBaseAddress + offset (unless stated otherwise) or
TSSystemConfigGlueBaseAddress (SCG) + offset.

The *TSMergerBaseAddress* is:

0x3802 0000.

The *TSSystemConfigGlueBaseAddress* is:

0x3800 1000.

Table 98: Transport stream merger register summary

Register	Description	Offset	Type
TSM_STREAMn_CFG	Stream n configuration	0x0000, 0x0020, 0x0040, 0x0060, 0x0080	R/W
TSM_STREAMn_SYNC	Stream synchronization	0x0008, 0x0028, 0x0048, 0x0068, 0x0088	R/W
TSM_STREAMn_STATUS	Stream status	0x0010, 0x0030, 0x0050, 0x0070, 0x0090	R/W
TSM_PTI_DESTINATION	PTI stream destination	0x0200	R/W
TSM_1394_DESTINATION	1394 interface stream destination	0x0238	R/W
TSM_PROG_COUNTERn	Program counter	0x0240, 0x0248	R/W
TSM_SWTS	SWTS data	0x0280	R/W
TS_SWTS_CFG	SWTS configuration	0x02E0	R/W
TSM_PTI_ALT_OUT_CFG	PTI_ALT_OUT configuration	0x0300	R/W
TS_1394_CFG	1394 port configuration	0x0338	R/W
TS_SYSTEM_CFG	System configuration	0x03F0	R/W
TS_SW_RESET	Software reset	0x03F0	R/W
NRSSA_CONFIG	NRSS-A interface configuration	*SCG* + 0x0008	R/W
TS_CONFIG	Stream configuration	*SCG* + 0x0024	R/W

TSM_STREAMn_CFG Stream n 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												PRIORITY				Reserved			RAM_ALLOC_START					STREAM_ON	REPLACE_ID_TAG	ADD_TAG_BYTES	INVERT_BYTECLK	TSIS_ASYNC_SOP_TOKEN	TSIS_ALIGN_BYTE_SOP	TSIS_SYNC_NOT_ASYNC	TSIS_SERIAL_NOT_PARALLEL

Address: *TSMergerBaseAddress* + 0x0000, 0x0020, 0x0040, 0x0060, 0x0080

Type: R/W

Reset: 0 (TSIS_SERIAL_NOT_PARALLEL: 1)

Description:

[31:20] **Reserved**

[19:16] **PRIORITY**
Defines the stream priority. 0000 = lowest priority and 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 and this field designates the lowest word that is allocated to stream X.

[7] **STREAM_ON**
This register bit gates the transport stream.
The purpose of this register bit is to stop transport data propagating to the SRAM before the circular buffers are setup.
If the stream is from off chip then the byteclks are disabled when this bit is LOW. If the stream is from the PTI alternate outputs or from SWTS then the FIFO enables are disabled when this bit is LOW.

[6] **REPLACE_ID_TAG**
The bit controls whether the memIOcontrol block replaces the second byte of the packet 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**
The bit controls whether the memIOcontrol block incorporates 6 bytes of ID and counter value into the stream header.

[4] **INVERT_BYTECLK**
This bit selects the 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**
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, SWTSs 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 SWTSs and PTI alternate outputs.
0: TSIS is in asynchronous mode.
1: TSIS is in synchronous mode.
Synchronous mode assumes all bytes with valid high are valid.
Asynchronous mode assumes all bytes with either valid high or PACKETCLK high are valid.

[0] **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 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
PACKET_LENGTH																SOP_TOKEN								DROP				SYNC			

Address: *TSMergerBaseAddress* + 0x0008, 0x0028, 0x0048, 0x0068, 0x0088
Type: R/W
Reset: See below
Description:

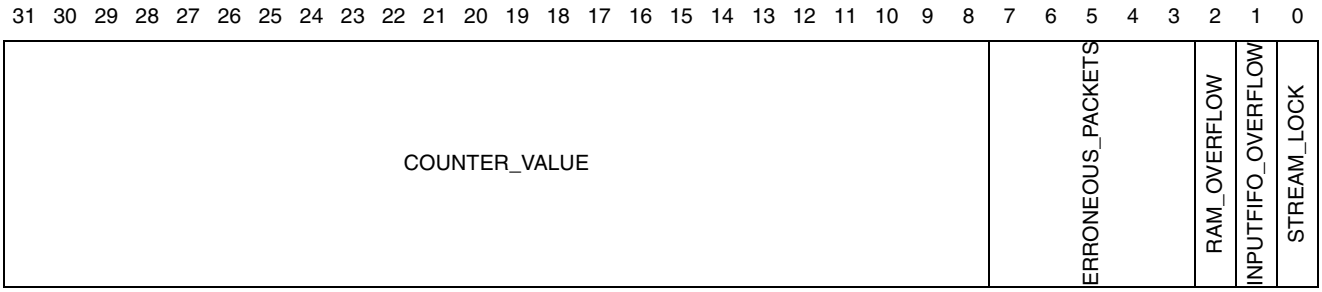
[31:16] **PACKET_LENGTH**
The PACKET_LENGTH bit field states the expected packet size in bytes, that is, a start of packet token should be found every PACKET_LENGTH bytes. Reset: 0000 0000 1011 1100 (0xBC)

[15:8] **SOP_TOKEN**
This field represents the 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**
This field represents the 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**
This field represents the 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 bit field should be set to 0000. Reset: 0000

TSM_STREAMn_STATUS Stream status



Address: *TSMergerBaseAddress* + 0x0010, 0x0030, 0x0050, 0x0070, 0x0090

Type: R/W

Reset: 0

Description:

[31:8] **COUNTER_VALUE**
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] **ERRONEOUS_PACKETS**
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_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] **INPUTFIFO_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_LOCK**
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_DESTINATION PTI stream destination

31	30	29	28	27	26	25	24	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																											STREAM_4	STREAM_3	STREAM_2	STREAM_1	STREAM_0

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_DESTINATION 1394 interface stream destination

31	30	29	28	27	26	25	24	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																											STREAM_4	STREAM_3	STREAM_2	STREAM_1	STREAM_0

Address: *TSMergerBaseAddress* + 0x0238
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.

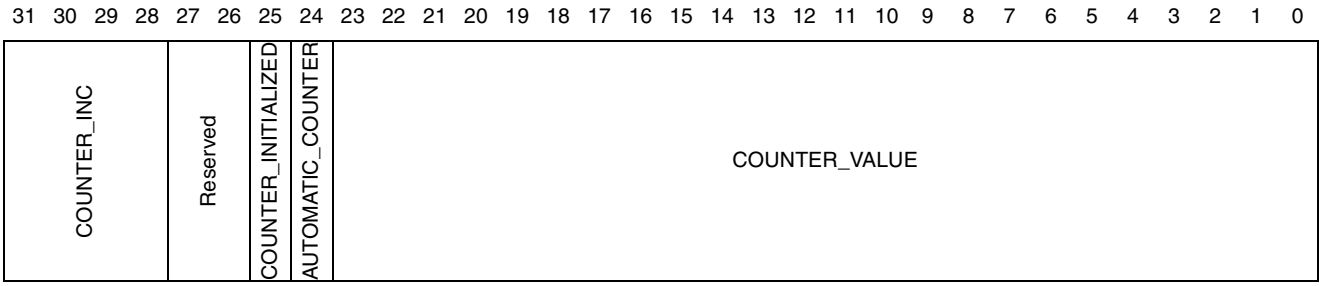
[3] **STREAM_3**
0: Stream 3 (SWTS0) does not go to the 1394 interface.
1: Stream 3 (SWTS0) goes to the 1394 interface.

[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_COUNTERn Program counter



Address: *TSMergerBaseAddress* + 0x0240, 0x0248

Type: R/W

Reset: 0

Description:

[31:28] **COUNTER_INC**
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.

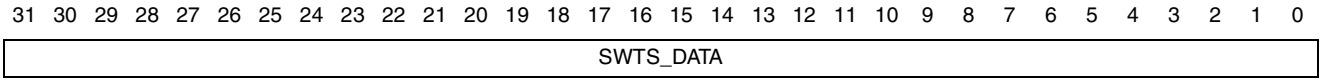
[27:26] **Reserved**

[25] **COUNTER_INITIALIZED**
Only used when AUTOMATIC_COUNTER = 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] **AUTOMATIC_COUNTER**
This bit defines the method of counter initialization.
0: The counter is used what ever is in the field COUNTER_VALUE.
1: The counter is automatically set to the value in the header of the first packet in a stream.

[0:23] **COUNTER_VALUE**
Set the programmable counter value when store operation and read the current counter value with a load operation.

TSM_SWTS SWTS data



Address: *TSMergerBaseAddress* + 0x0280

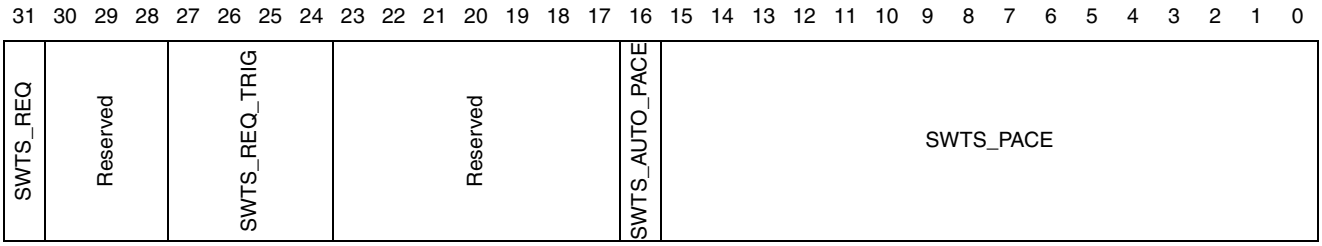
Type: R/W

Reset: 0

Description:

[31:0] **SWTS_DATA**
SWTS data register. Data is expected to be little endian, that is the LS byte is first to arrive at the PTI. Data bits within each byte are organized in a big endian manner.
On a read, this register either sends back the current stored value or if empty the latest stored value.
Access are always granted to this register within a finite number of cycles. The written word gets put into a 256-byte FIFO and the FIFO empties itself at the rate specified in SWTS_CFG.

TS_SWTS_CFG SWTS configuration



Address: *TSMergerBaseAddress* + 0x02E0

Type: R/W

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**

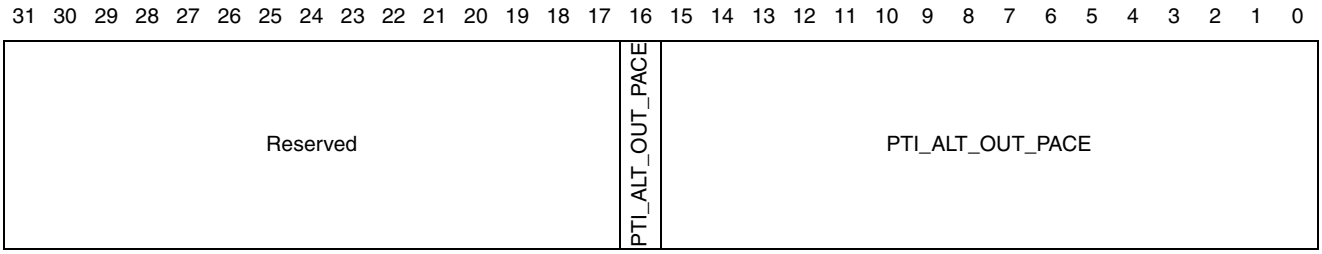
[27:24] **SWTS_REQ_TRIG**
Valid Range 0000 to 1111. The SWTS request signal (O_SWTS_REQ) is high when the fill level falls below (SWTS_REQ_TRIG * 16) bytes. Trigger levels are 0, 16, 32, 64, 96 up to 224, 240, 256 bytes.

[23:17] **Reserved**

[16] **SWTS_AUTO_PACE**
1: The pace of SWTS data is controlled by room in FIFOs and RAM circular buffers.
0: The pace of the SWTS data is controlled by the 16 bits of the SWTS_PACE field.

[15:0] **SWTS_PACE**
This field defines the 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_ALT_OUT configuration



Address: *TSMergerBaseAddress* + 0x0300

Type: R/W

Reset: 0

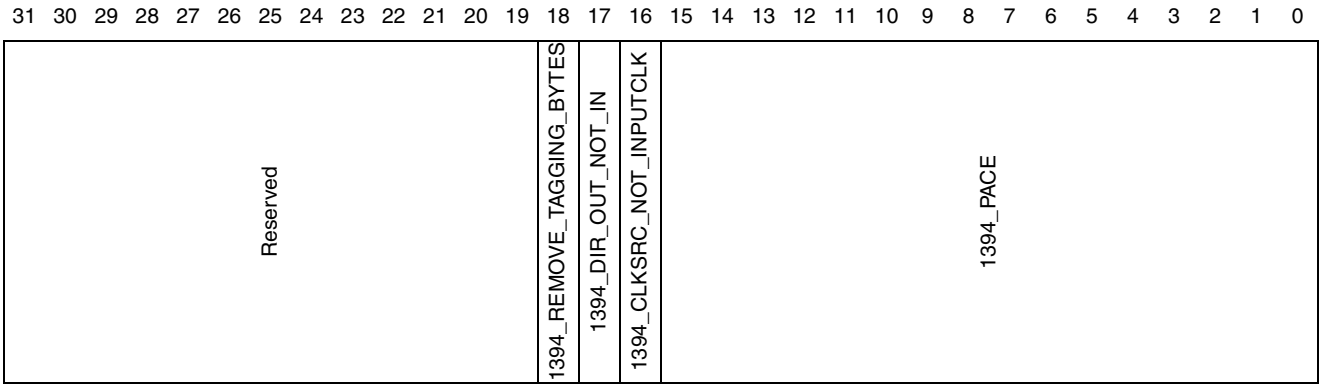
Description:

[31:17] **Reserved**

[16] **PTI_ALT_OUT_AUTO_PACE**
1: The pace of PTI alternative output data requests are controlled by room in FIFOs and RAM circular buffers.
0: The pace of the PTI alternative output data requests are controlled by the 16 bits of the PTI_ALT_OUT_PACE field.

[15:0] **PTI_ALT_OUT_PACE**
This field defines the number of cycles between enable pulses on the PTI alternative output interface.

TS_1394_CFG 1394 port configuration



Address: *TSMergerBaseAddress* + 0x0338

Type: R/W

Reset: 0

Description:

[19:31] **Reserved**

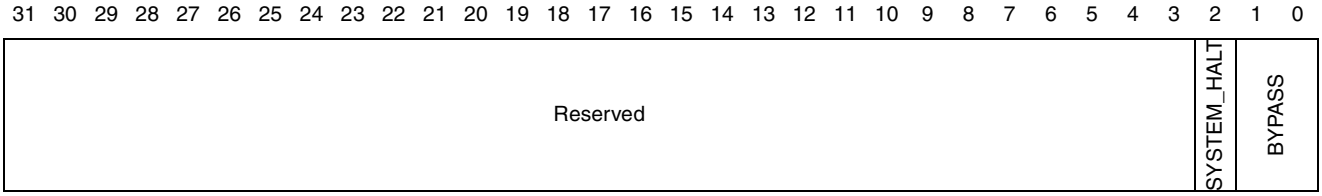
[18] **1394_REMOVE_TAGGING_BYTES**
0: Outgoing stream is unmodified.
1: Outgoing stream has six bytes removed after the start of packet byte.

[17] **1394_DIR_OUT_NOT_IN**
0: 1394 data ports are set to input.
1: 1394 data ports are set to output

[16] **1394_CLKSRC_NOT_INPUTCLK**
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] **1394_PACE**
This field defines the divide ratio of CLK_SYSTEM that the 1394 port produces to clock data out. This register is only used if 1394_CLOCK_SRC = 1

TS_SYSTEM_CFG System configuration



Address: *TSMergerBaseAddress* + 0x03F0

Type: R/W

Reset: 0

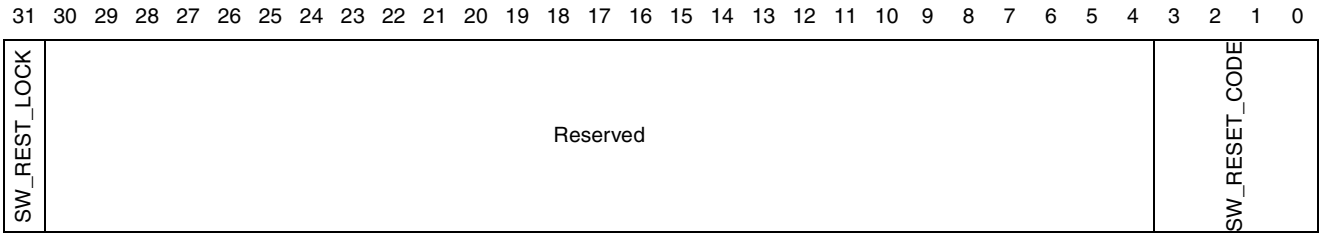
Description:

[31:3] **Reserved**

[2] **SYSTEM_HALT**
Pause system. No read or write accesses to the central SRAM are allowed causing the system to halt.
0: The system is active. Read and Write accesses to the configuration registers are always allowed.
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 transport stream 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_RESET Software reset



Address: *TSMergerBaseAddress* + 0x03F8

Type: R/W

Reset: SW_RESET_CODE: 0, SW_RESET_LOCK: 1

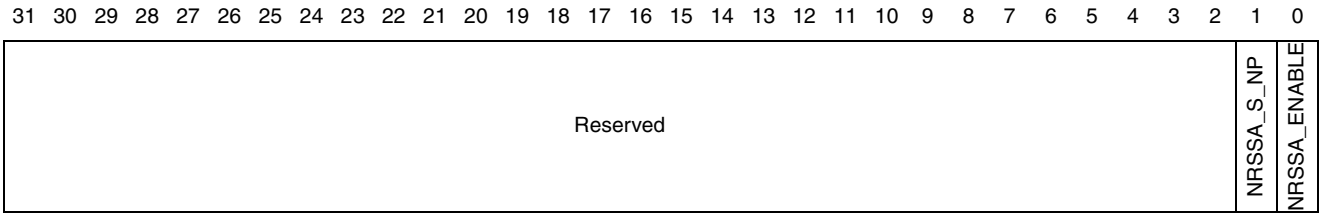
Description:

[31] **SW_RESET_LOCK**
This controls access to the SW_RESET_CODE to minimize an accidental software reset occurring. The lock bit must be reset in a different access to the access that sets SW_RESET_CODE.

[30:4] **Reserved**

[3:0] **SW_RESET_CODE**
If this 4-bit field is set to 0110 (0x6) when SW_RESET_LOCK = 0 then a software reset signal is pulsed for 1 CLK_SYSTEM cycle, after which the SW_RESET_CODE field returns to 0000 and SW_RESET_LOCK returns high.

NRSSA_CONFIG NRSS-A interface configuration



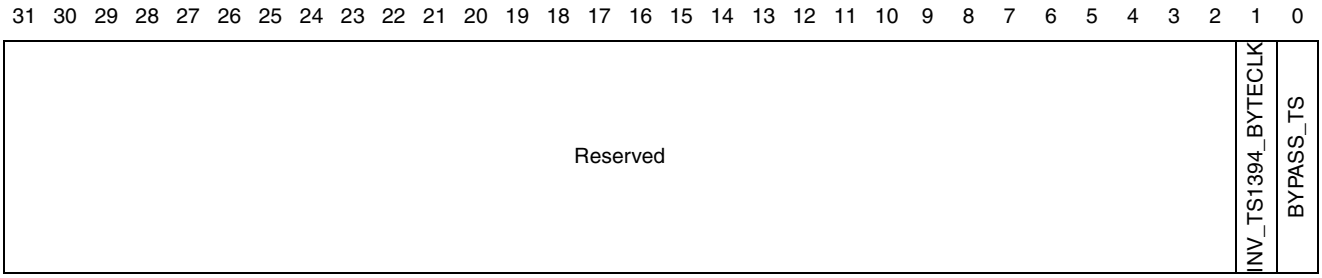
Address: *TSSystemGlueConfigBaseAddress* + 0x0008
Type: R/W
Reset: 0
Description:

[31:2] **Reserved**

[1] **NRSSA_S_NP**: NRSS-A interface serial or parallel input
Selects format of input TS0 to be formatted for NRSS-A module.
0: parallel 1: serial

[0] **NRSSA_ENABLE**: Enable NRSS-A interface
0: TS0 goes directly to TSMerger
1: NRSS-A interface is enabled. Transport stream TS0 is formatted and passed through an external NRSS-A module before being fed to the TSMerger.

TS_CONFIG Stream configuration



Address: *TSSystemGlueConfigBaseAddress* + 0x0024
Type: R/W
Reset: 0
Description:

[31:2] **Reserved**

[1] **INV_TS1394_BYTECLK**
0: 1394 byte clock not inverted
1: 1394 byte clock inverted

[0] **BYPASS_TS**
0: TS interface behaves as indicated by TSMerger configuration bits.
1: signals input on TSIN0 are routed directly to TSOUT2 output.

47 Programmable transport interface (PTI)

47.1 Overview

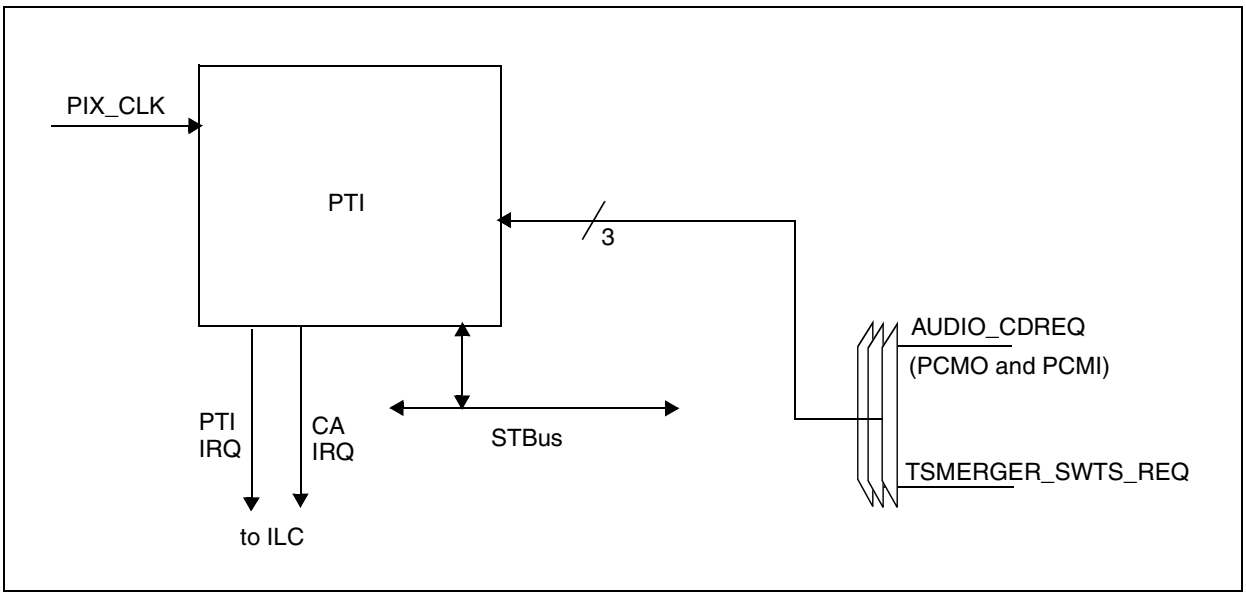
The PTI is a dedicated transport engine. It contains its own CPU and handles the transport deMUX functionality of the set-top box.

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 ST20 processor such that it is locked on to the encoder's clock.

The ST20 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 ST20 memory space through the interface.

The three CDREQ signals are used as pacing signals for the PTI DMAs.

Figure 86: PTI inputs and outputs



There are two sources of CDREQ and DMAREQ that can be routed to the PTI. These are:

- one DMAREQ signal from the audio block,
- one DMAREQ from the transport input of the TSMerger (SWTS).

The targets for these CDREQs are a single NOT_CDREQ for the PTI.

Note: NOT_CDREQ is an active low signal indicating that the target is ready to accept new data.

The MUX control for each target PTI channel is in Table 99.

Table 99: CDREQ_SRC_MUXSEL source select for PTI

Source	PTI CDREQ selection bit[a]	FDMA request number	FDMA signal
AUDIO_CDREQ	0	29	AUDIO_CD_REQ
TSMERGER_SWTS_REQ	1	28	SWTS_REQ

a. From PTI general purpose register configuration bits register [13:15] see *PTI_CFG on page 412*.

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 ST20 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 ST20 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 ST20.

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 ST20. 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 ST20 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.

47.2 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,
- support for incoming MPEG-2 transport streams with a data rate of 138 Mbit/s or greater,
- descrambling using the following algorithms:
  - DVB-CSA,
  - DES: ECB, CBC, DVS042, OFB, CTS,
  - Multi2: ECB, CBC, DVS042, OFB.
- 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 of one data stream directly to decoders,
- 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,
- eight descrambling key pairs per PTI,
- 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 ST20 CPU of buffer state.

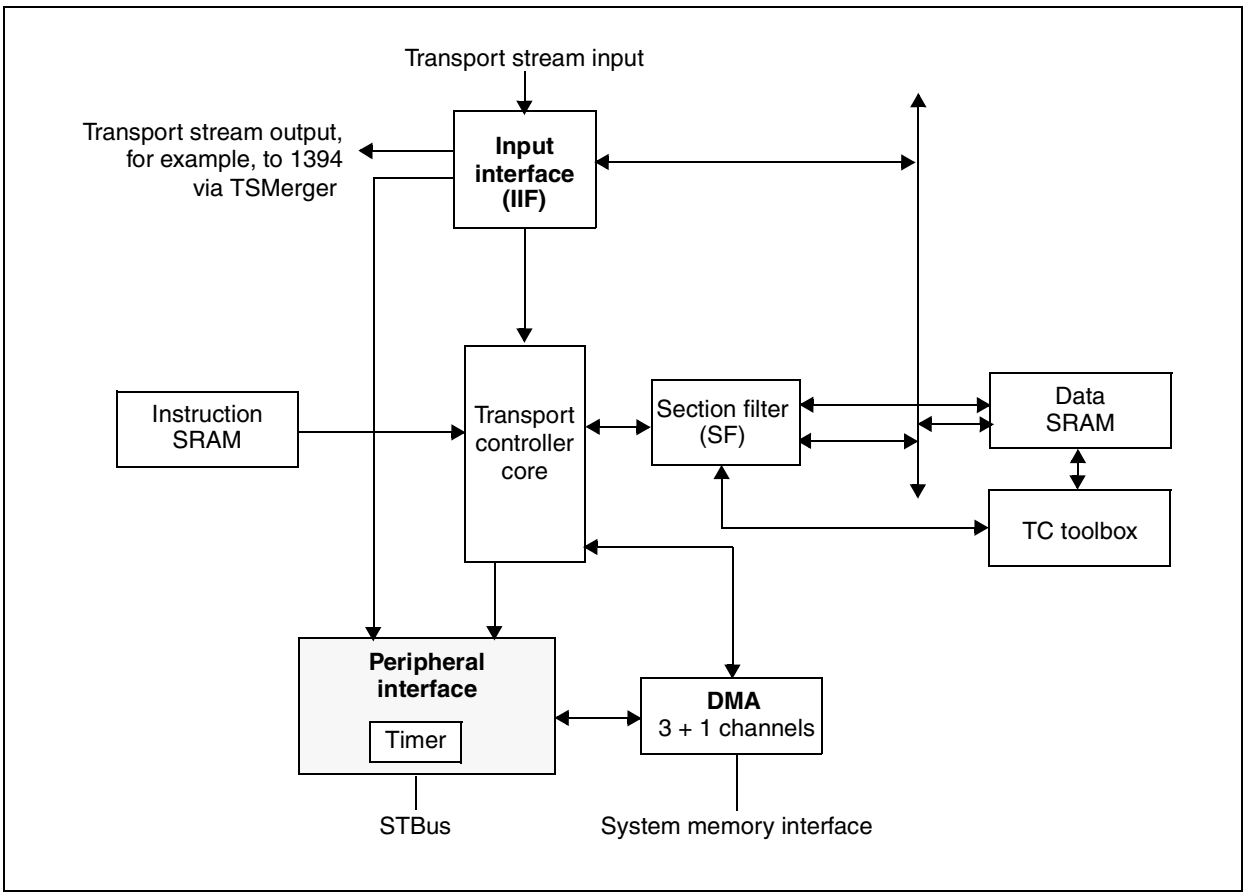
In addition to these transport device functions, the interface copies the entire transport stream or transport packets with selected PIDs from the transport stream through to the PTI output stream interface.

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.

47.3 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 87.

Figure 87: Programmable transport interface architecture



47.3.1 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 ST20 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.

47.3.2 Shared memory

The PTI contains 6.5 Kbytes of data memory, which is accessed as 32-bit words by the ST20 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 ST20 CPU, and is loaded with code by the ST20 before enabling the TC. The ST20 cannot access the instruction memory while the TC is executing.

47.3.3 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.

47.3.4 DMA

The DMA block is responsible for writing processed transport data to ST20 memory.

There are four DMA channels. One channel (channel 0) is used for writing data to ST20 memory under the control of the TC, so that data is sorted into a number of circular buffers. Channels 1 to 3 are dedicated to managing these buffers; data is written directly from these buffers to the video and audio decoders under the control of signals generated by the decoder FIFOs. These three channels are generic DMA engines controlled by the ST20 or TC. A programmable delay (the holdoff time) is built into the data transfer to ensure all data has reached the decoder before the

FIFO 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).

47.3.5 Peripheral interface

The peripheral interface allows the PTI to interact with the ST20 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 47.4.2: Soft reset on page 389*).

47.3.6 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 100.

Table 100: 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.

47.4 PTI operation

The programmable transport interface (PTI) performs the transport parsing and processing functions without intervention of the ST20 CPU. The block is controlled by the ST20 and communicates with it via a shared data SRAM block local to the PTI, registers in the PTI, data structures in the ST20 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 ST20 CPU to initialize and control the block and to provide interrupt status and control.

47.4.1 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 ST20,
- initialization is performed as described below,
- the TC is enabled by setting the TCENABLE bit of the 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 IIFFIFOENABLE register bit must be set high to enable the input FIFO.

47.4.2 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.

47.4.3 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 ST20. 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 ST20.

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 either into a circular buffer or to a decoder compressed data FIFO. 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 ST20 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 ST20 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 ST20 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 ST20 and the interrupt bit itself is used as a handshake for the processing of the PCR by the ST20. 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 ST20 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.

47.4.4 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 ST20 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.

47.5 Interrupt handling

The PTI may interrupt the ST20 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 ST20 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 ST20 resets interrupts by writing to the appropriate bit position in the appropriate interrupt acknowledge register. The ST20 determines which interrupts are currently set by reading the appropriate interrupt status register.

The ST20 enables or disables interrupts by writing 1 or 0 into the appropriate bit position in the appropriate interrupt enable register. Using this mechanism, the ST20 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 ST20 has checked, and setting the interrupt status bit before the ST20 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 ST20 process enables the interrupt by setting the correct interrupt enable bit before making the process wait on a semaphore.

Figure 88 shows the TC and the ST20 processes and mechanism described above.

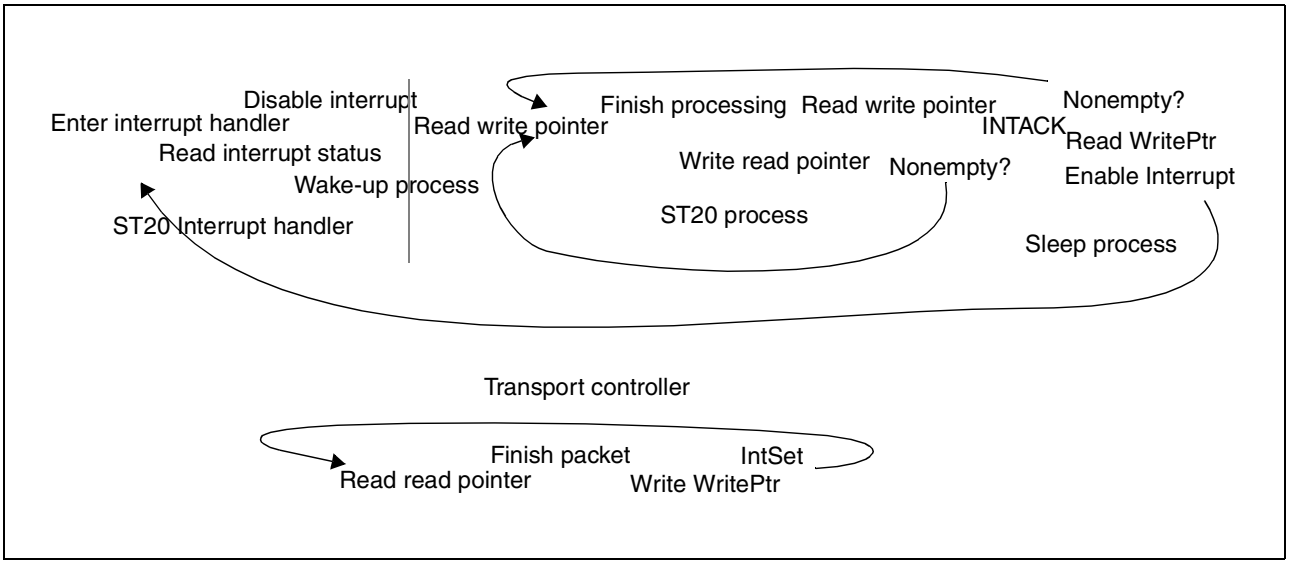
Note: At any given time each process is at any point during the critical regions of code. There is no implied timing for each step of a process only an ordering of steps.

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 ST20 CPU.

Figure 88: PTI buffer interrupt handling



47.6 DMA operation

The PTI contains a four channel DMA controller which is programmed by the TC and the ST20 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.

The TC uses DMA channel 0, a write only DMA, to send the data from a transport packet to a circular buffer in the ST20 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 are 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 ST20 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.

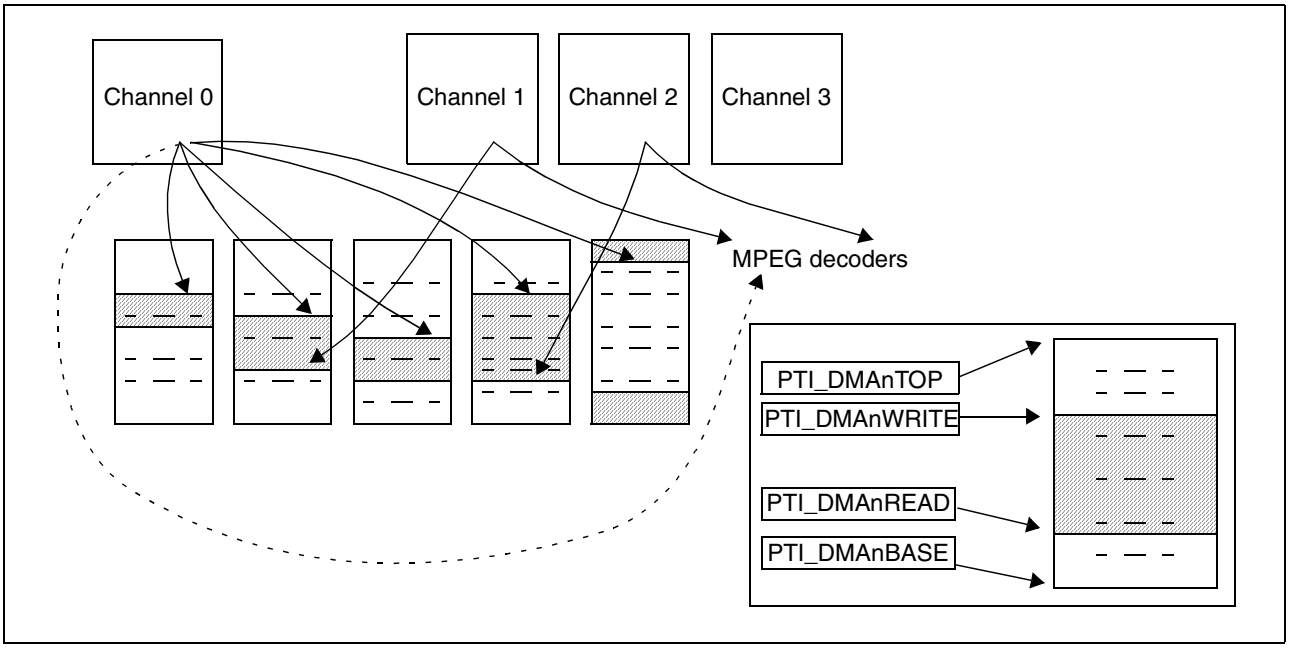
Block moves are supported on channels 1 to 3 using a special DMA mechanism which increments the input address after each write.

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 ST20 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 ST20 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 ST20 process is updated and an interrupt is set to signal to the ST20 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 ST20 CPU if this is writing data into the buffer that DMA channels 1 to 3 are reading from.

Figure 89: DMA channels



47.6.1 Circular buffers

The inset in Figure 89 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.

47.6.2 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.

47.6.3 Flushing the DMA

Before doing a PTI soft reset (see *Section 47.4.2: Soft reset on page 389*), 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.

47.6.4 Performance

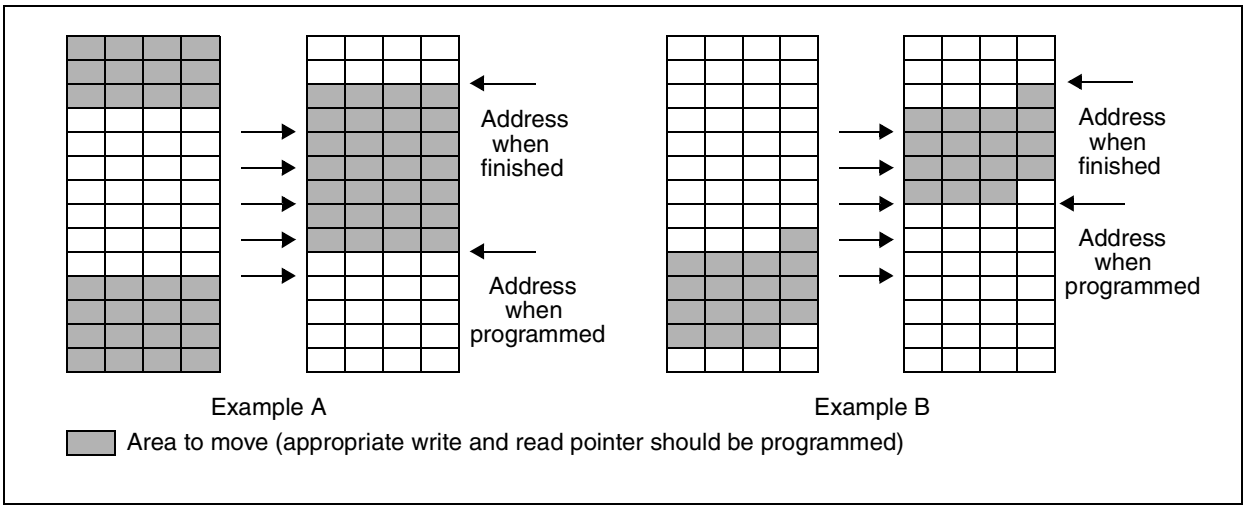
Each DMA channel performs one read, write or burst access when it has been granted by the memory arbiter. When more than one channel is active, one performs an access while another is waiting for valid pulses to come back from the interconnect. So the accesses are interleaved, guaranteeing a high performance.

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.

47.6.5 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 90*.

Figure 90: DMA blockmove



47.7 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 ST20 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.

1. *Generic Coding Of Moving Pictures And Associated Audio: Systems, Recommendation H.222.0*, ISO/IEC 13818-1

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.

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 91 illustrates different examples of CAM configuration.

Figure 91: Memory organization when CAM_CONFIG NOT_MASK bit = 1 (masks disabled).

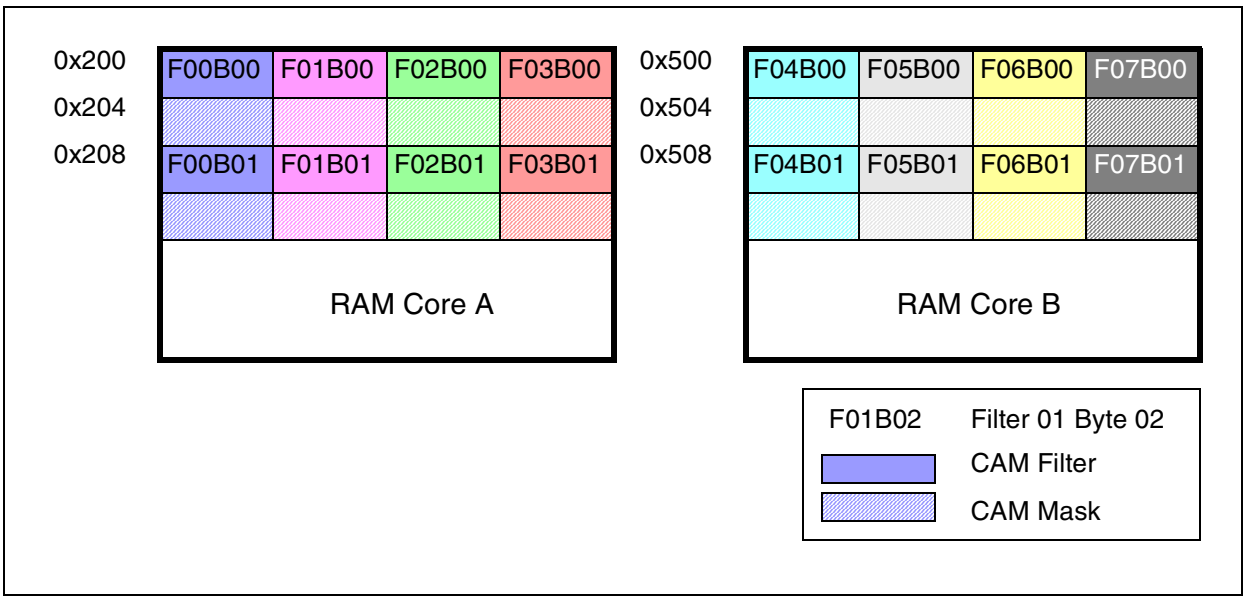
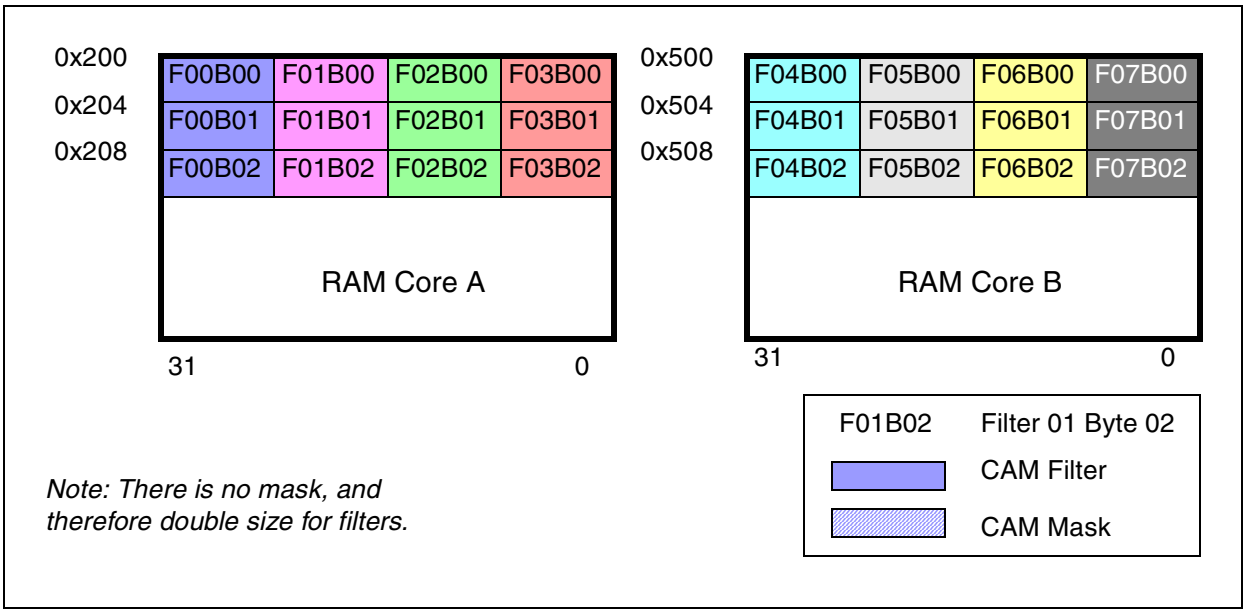
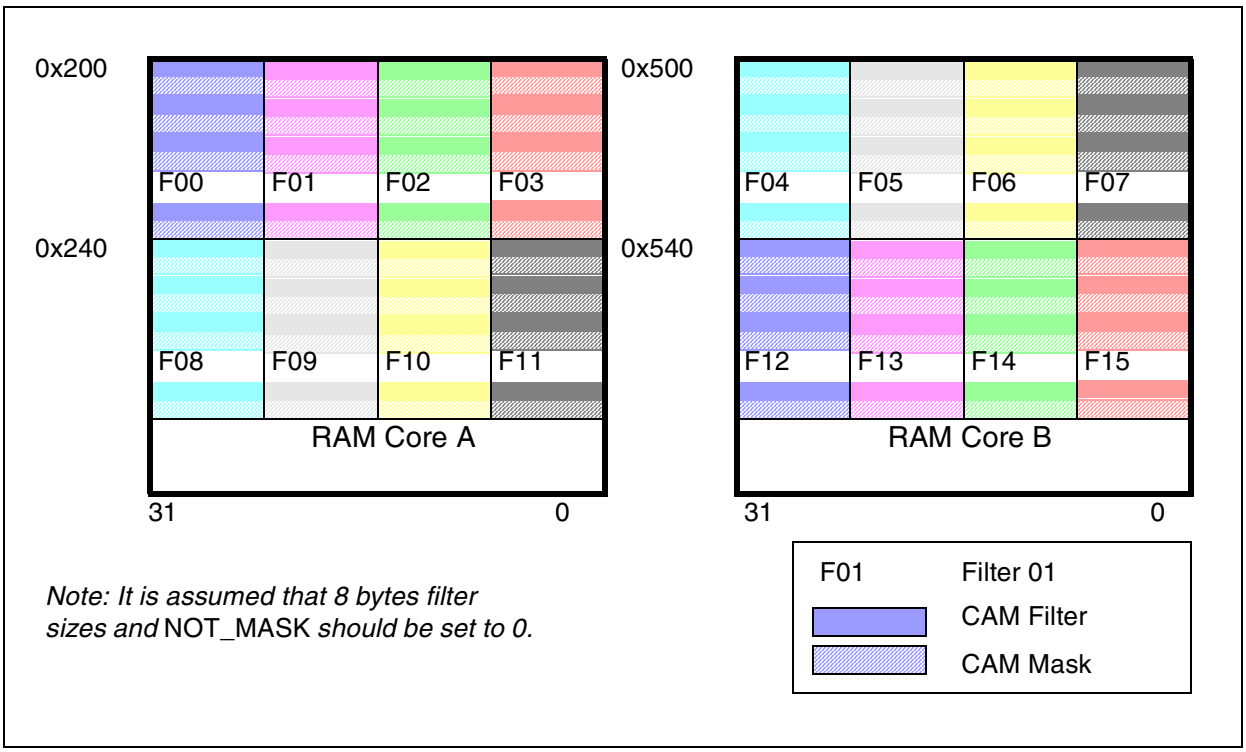


Figure 92: Memory organization with CAM_CONFIG NOT_MASK bit = 0 (masks enabled)



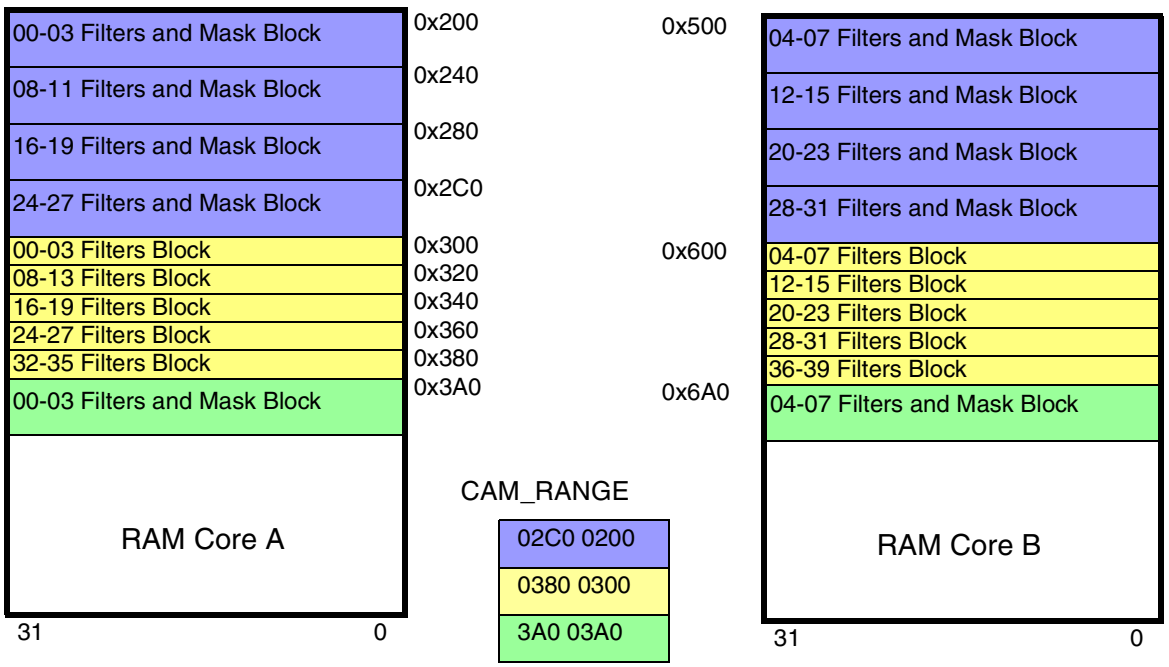
Note: There is no mask, and therefore double size for filters.

Figure 93: Memory organization for 16 filters



Note: It is assumed that 8 bytes filter sizes and NOT_MASK *should be set to 0.*

Figure 94: 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.*

47.7.1 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 ST20. The ST20 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.

47.7.2 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 101 to Table 103.

PTI_SFDATAn has to be arranged inside CAM memory space as in Section 47.7: *Section filter on page 395*

Table 101: 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 102: 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 102: 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 103: 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

47.7.3 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.

47.7.4 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).

48 Programmable transport interface (PTI) registers

Addresses are provided as *PTIBaseAddress* + offset.

The *PTIBaseAddress* is:

0x38010000.

Table 104: PTI register summary

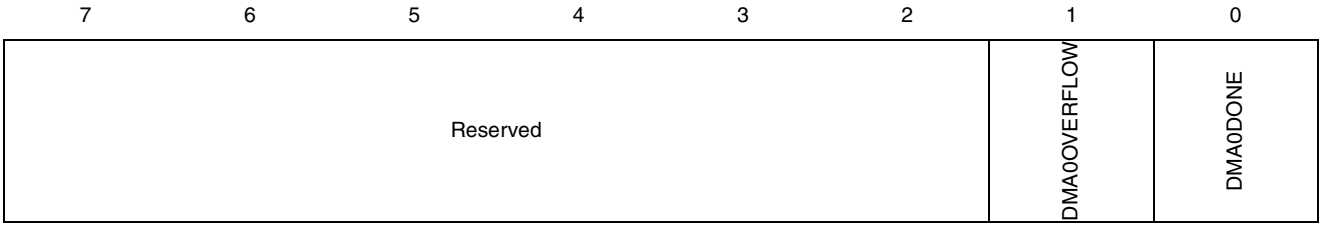
Register	Description	Offset	Type
DMA			
PTI_DMA0STATUS	DMA channel 0 status	0x1018	RW
PTI_DMAnBASE	DMA buffer base address	0x1000, 0x1020, 0x1040, 0x1060	WO
PTI_DMAnHOLDOFF	DMA hold off time	0x1014, 0x1034, 0x1054, 0x1074	R/W
PTI_DMAnREAD	DMA buffer read address	0x100C to 0x106C	WO, R/W
PTI_DMA0SETUP	DMA channel 0 byte not word mode and block move	0x1010	R/W
PTI_DMAnSETUP	DMA channels 1 to 3 byte not word mode and block move	0x1030, 0x1050, 0x1070	R/W
PTI_DMAnTOP	DMA buffer top address	0x1004 to 0x1064	WO
PTI_DMAnWRITE	DMA buffer write address	0x1008, 0x1028, 0x1068	R/W, WO
PTI_DMAnCDADDR	Configuration of CD FIFO address for channels 1 to 3	0x1038, 0x1058, 0x1078	WO
PTI_DMAENABLE	DMA enable	0x101C	WO
PTI_DMAFLUSH	Flush ready for a soft reset	0x105C	R/W
Input interface			
PTI_IIFALTFIFOCOUNT	IIF alternative FIFO count	0x2004	RO
PTI_IIFALTLATENCY	IIF alternative output latency	0x2010	WO
PTI_IIFFIFOCOUNT	IIF count	0x2000	RO
PTI_IIFFIFOENABLE	IIF FIFO enable	0x2008	R/W
PTI_IIFSYNCDROP	IIF sync drop	0x2018	WO
PTI_IIFSYNCLOCK	IIF sync lock	0x2014	WO
PTI_IIFSYNCCONFIG	IIF sync configuration	0x201C	WO
PTI_IIFSYNCPERIOD	IIF sync period	0x2020	WO
PTI configuration			
PTI_CFG	PTI configuration	0x0058	R/W
PTI_AUDPTS	Audio presentation time stamp	0x0040, 0x0044	RO
PTI_INTACKn	PTI interrupt acknowledgment	0x0020 to 0x002C	WO

Table 104: PTI register summary

Register	Description	Offset	Type
PTI_INTENABLEn	PTI interrupt enable	0x0010 to 0x001C	R/W
PTI_INTSTATUSn	PTI interrupt status	0x0000 to 0x000C	RO
PTI_VIDPTS	Video presentation time stamp	0x0048, 0x004C	RO
PTI_STCTIMER	Set STC timer	0x0050 to 0x0054	WO
Transport controller mode			
PTI_TCMODE	Transport controller mode	0x0030	R/W

48.1 DMA registers

PTI_DMA0STATUS DMA channel 0 status



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 ST20 software via one of the interrupt status bits. The interrupt bit is also used as a handshake that the ST20 software has acknowledged the condition. Data is discarded by DMA channel 0 if the buffer it is writing overflows.

[7:2] **Reserved**

[1] **DMA0OVERFLOW**
If 1, the channel 0 circular buffer overflowed. Reset by writing 1 to this bit.

[0] **DMA0DONE**
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_DMAnBASE 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_DMAnHOLDOFF DMA hold off 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
0x1014	Reserved	DMA0WRITESIZE	Reserved	DMA0HOLDOFF
0x1054 0x1034	Reserved	DMA1WRITESIZE	Reserved	DMA1HOLDOFF
0x1058 0x1054	Reserved	DMA2WRITESIZE	Reserved	DMA2HOLDOFF
0x105C 0x1074	Reserved	DMA3WRITESIZE	Reserved	DMA3HOLDOFF

Address: *PTIBaseAddress* + 0x1014, 0x1034, 0x1054 and 0x1074

Type: R/W

Reset: 0 (DMAnHOLDOFF)

4 (DMAnWRITESIZE)

Description: The PTI_DMA[1:3]HOLDOFF registers are used to specify the delay time between the end of a burst of data being transferred and resampling the NOT_CDREQ[1:3] signals before another transfer is started on a DMA channel. The time is in units of byte clock cycles in the range 0 to 31 cycles.

[31:24] **Reserved**

[23:16] **DMAnWRITESIZE**
Number of bytes that can be written by the channel before having to wait for all valid pulses to be returned.

[15:8] **Reserved**

[7:0] **DMAnHOLDOFF**
Number of clock cycles to count between receiving a valid pulse after writing to the CD FIFO, and assuming CD_REQ[n] is valid.

PTI_DMAnREAD 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	DMA0READ
0x1034 0x102C	DMA1READ
0x1038 0x104C	DMA2READ
0x103C 0x106C	DMA3READ

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.

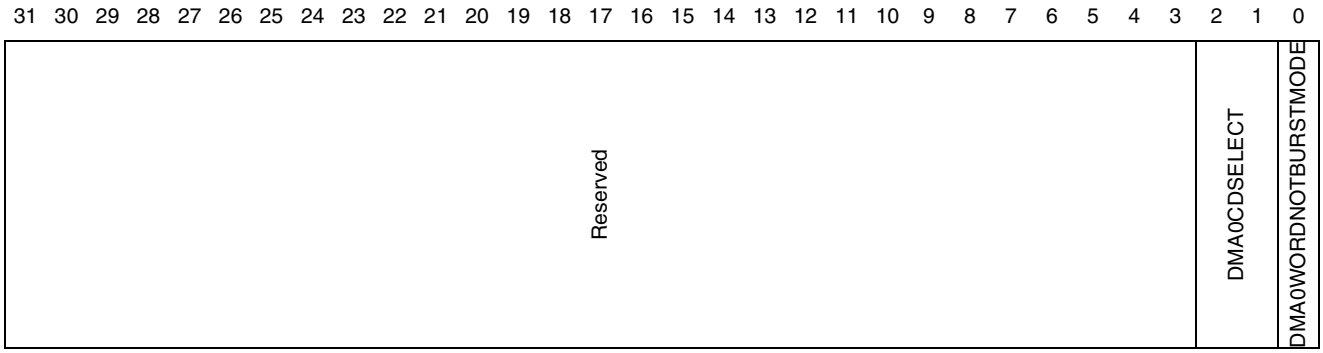
The read pointer is initialized to be equal to the write pointer. As data is read from the buffer, the hardware updates the read pointer.

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 ST20 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_DMA0SETUP DMA channel 0 byte not word mode and block move



Address: *PTIBaseAddress* + 0x1010

Type: R/W

Reset: 0

Description:

[31:3] **Reserved**

[2:1] **DMA0CDSELECT**

Selects which NOT_CDREQ is checked when working in CD FIFO mode.

00: NOT_CDREQ is not checked 01: NOT_CDREQ[1] signal is checked

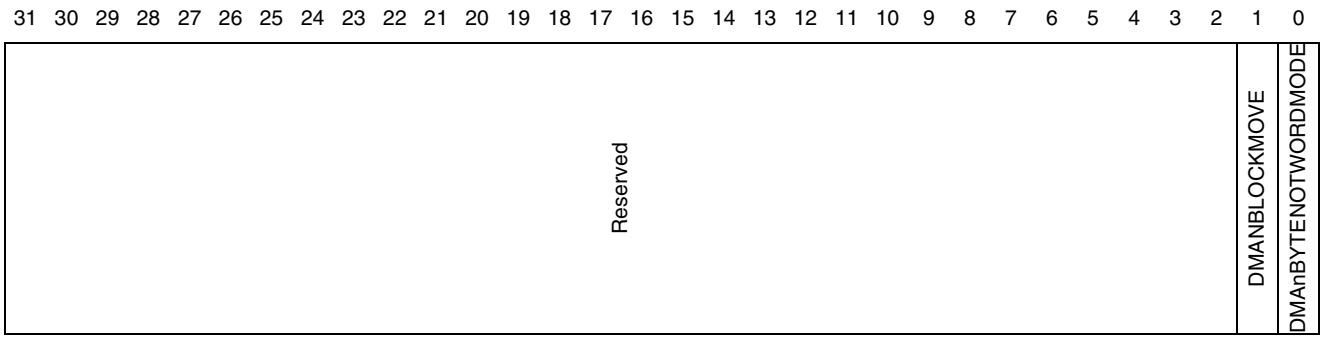
10: NOT_CDREQ[2] signal is checked 11: NOT_CDREQ[3] signal is checked

When 00 is selected channel 0 works under CD_REQ and holdoff mode, but it is assumed that the CDREQ signal is always high when holdoff time is reached.

[0] **DMA0WORNOTBURSTMODE**

0: Enable 4-word-bursts 1: Write word-at-a-time

PTI_DMAnSETUP 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] **DMAnBLOCKMOVE**

0: CDADDR remains fixed.

1: CDADDR is incremented after each write. In this mode the CD FIFO mode is off, and so NOT_CDREQ is ignored and *Holdoff* and *Writelength* are not considered.

[0] **DMAnBYTENOTWORDMODE**

0: Write word-at-a-time 1: Write byte-at-a-time

PTI_DMAnTOP DMA buffer 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
0x1010 0x1004	DMA0TOP
0x1014 0x1024	DMA1TOP
0x1018 0x1044	DMA2TOP
0x101C 0x1064	DMA3TOP

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 DMAnBASE address register.

For channel 0 only, if the buffer size is zero then this address must be equal to the corresponding DMAnBASE address. The DMA channel 0 register would be set up for each PID in the PID data structure in the PTI data SRAM.

PTI_DMAnWRITE 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	DMA0WRITE
0x1024 0x1028	DMA1WRITE
0x1028 0x1048	DMA2WRITE
0x102C 0x1068	DMA3WRITE

Address: *PTIBaseAddress* + 0x1008, + 0x1028, + 0x1048, + 0x1068

Type: WO except DMA0WRITE 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_DMAnCDADDR 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	DMA1CDADDR	Res.
0x1068 0x1058	DMA2CDADDR	Res.
0x106C 0x1078	DMA3CDADDR	Res.

Address: *PTIBaseAddress* + 0x1038, + 0x1058, + 0x1078

Type: WO

Reset: Undefined

Description: PTI_DMAnCDADDR 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. DMAnCDADDR is only defined for MPEG channels (MDC channels).

PTI_DMAENABLE DMA 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	DMA3_ENABLE	DMA2_ENABLE	DMA1_ENABLE	DMA0_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**

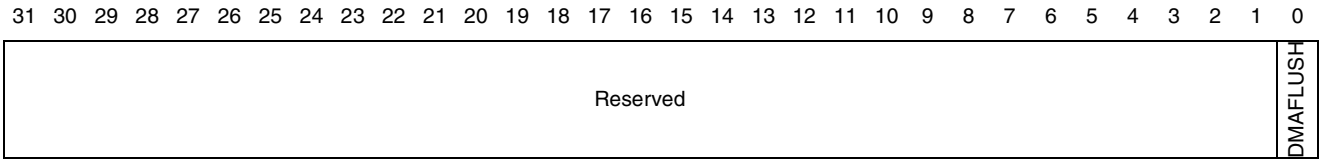
[3] **DMA3ENABLE**
Enable (1) or disable(0) channel 3.

[2] **DMA2ENABLE**
Enable (1) or disable(0) channel 2.

[1] **DMA1ENABLE**
Enable (1) or disable(0) channel 1.

[0] **DMA0ENABLE**
Enable (1) or disable(0) channel 0.

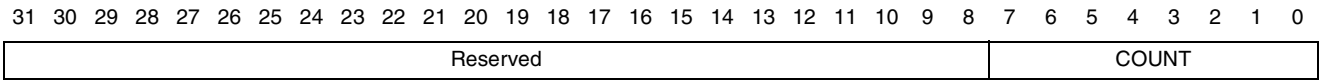
PTI_DMAFLUSH Flush ready for a soft reset



Type: *PTIBaseAddress* + 0x105C
Type: R/W
Reset: 0
Description: 1 is written to PTI_DMAFLUSH when a flush is required. It is reset by writing 0 to the bit.

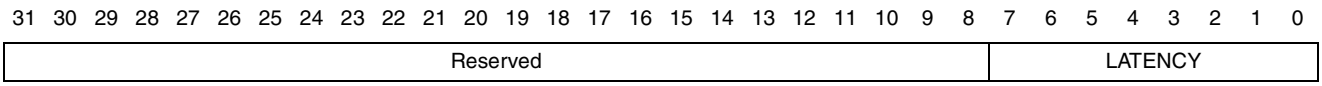
48.2 Input interface registers

PTI_IIFALTFIFOCOUNT IIF alternative FIFO count



Address: *PTIBaseAddress* + 0x2004
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_IIFALTLATENCY IIF alternative output latency



Address: *PTIBaseAddress* + 0x2010
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_IIFFIFOCOUNT 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	COUNT						

Address: *PTIBaseAddress* + 0x2000
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 ST20 reads this register.

[6:0] **COUNT**
The number of bytes in the input FIFO.

PTI_IIFFIFOENABLE 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																								COUNT							

Address: *PTIBaseAddress* + 0x2008
Type: R/W
Reset: Undefined
Description: Allows data into the input FIFO. When this field is zero, the input FIFO is reset. After the ST20 completes the initialization sequence, it should set this field to 1.

PTI_IIFSYNCDROP 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																												DROPPACKETS			

Address: *PTIBaseAddress* + 0x2018
Type: WO
Reset: Undefined
Description: The number of successive erroneous sync bytes found before the lock is treated as lost.

PTI_IIFSYNCLOCK IIF sync 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																												LOCKPACKETS			

Address: *PTIBaseAddress* + 0x2014
Type: WO
Reset: Undefined
Description: The number of successive correct sync bytes found before the sync detection is locked.

PTI_IIFSYNCCONFIG IIF sync 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																														SYNC	

Address: *PTIBaseAddress* + 0x201C
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_PACKETCLK

PTI_IIFSYNCPERIOD IIF sync 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																								SYNCPERIOD							

Address: *PTIBaseAddress* + 0x2020
Type: WO
Reset: Undefined
Description:

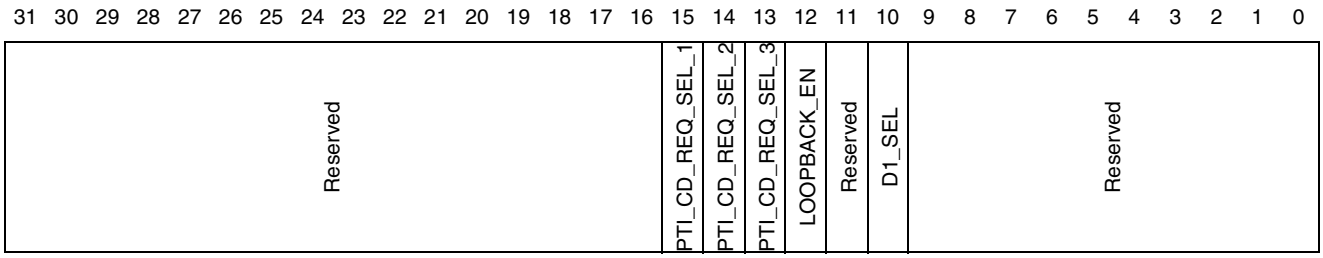
[31:8] **Reserved**

[7:0] **SYNCPERIOD**

Specifies the expected number of TS_IN_BYTECLK_PULSE cycles between SYNC bytes
On reset set to 188.

48.3 PTI configuration registers

PTI_CFG PTI configuration



Address: *PTIBaseAddress* + 0x0058

Type: R/W

Reset: 0

Description:

[31:16] **Reserved**

[15] **PTI_CD_REQ_SEL_1**
0: AUDIO_CDREQ 1: TSMERGER_SWTS_REQ

[14] **PTI_CD_REQ_SEL_2**
0: AUDIO_CDREQ 1: TSMERGER_SWTS_REQ

[13] **PTI_CD_REQ_SEL_3**
0: AUDIO_CDREQ 1: TSMERGER_SWTS_REQ

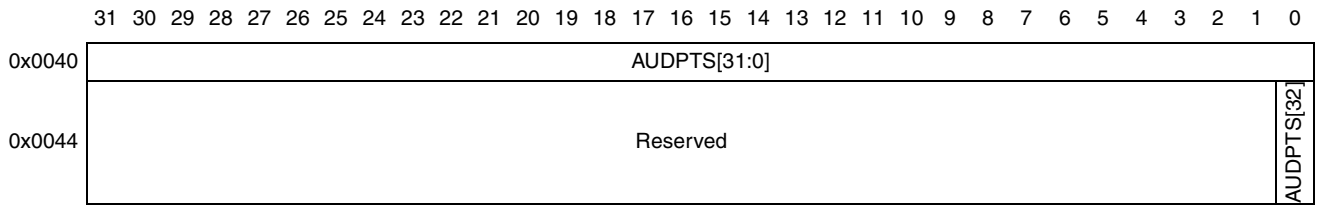
[12] **LOOPBACK_EN:** DVB-CI loopback mode
Reserved, do not modify.

[11] **Reserved**

[10] **D1_SEL**: D1 on TSIN1
Reserved, do not modify.

[9:0] **Reserved**

PTI_AUDPTS 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_INTACKn 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	INTACK0
0x0024	Reserved	INTACK1
0x0028	Reserved	INTACK2
0x002C	Reserved	INTACK3

Address: *PTIBaseAddress* + 0x0020 to 0x002C
Type: WO
Reset: Undefined
Description: Acknowledge the corresponding interrupt bit when a bit is written as 1.

PTI_INTENABLEn 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	INTENABLE0
0x0014	Reserved	INTENABLE1
0x0018	Reserved	INTENABLE2
0x001C	Reserved	INTENABLE3

Address: *PTIBaseAddress* + 0x0010 to 0x001C
Type: R/W
Reset: Undefined
Description: The corresponding interrupt is enabled when a bit is 1.

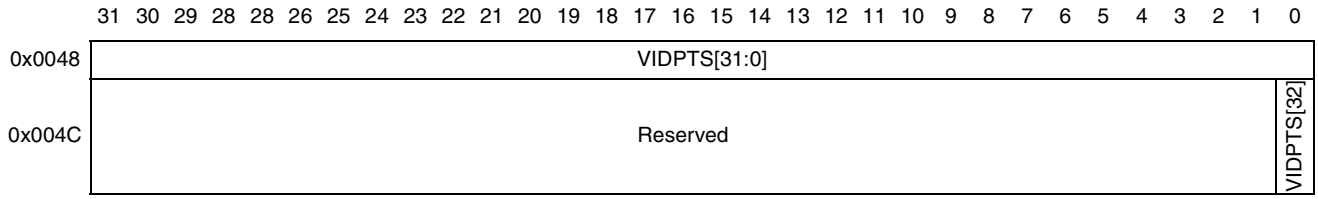
PTI_INTSTATUSn PTI interrupt status

	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	INTSTATUS0
0x0004	Reserved	INTSTATUS1
0x0008	Reserved	INTSTATUS2
0x000C	Reserved	INTSTATUS3

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 PTIINTSTATUS[3:0] registers are used by the TC to raise an interrupt to the ST20 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 INTSTATUS bits to 1, each bit stays set until the interrupt is acknowledged by the ST20 software by writing 1 to the corresponding bit of the corresponding PTIINTACK register. An interrupt is masked by writing 0 to the corresponding enable bit of the corresponding PTIINTENABLE register.

PTI_VIDPTS Video presentation time stamp



Address: *PTIBaseAddress* + 0x0048 and 0x004C

Type: RO

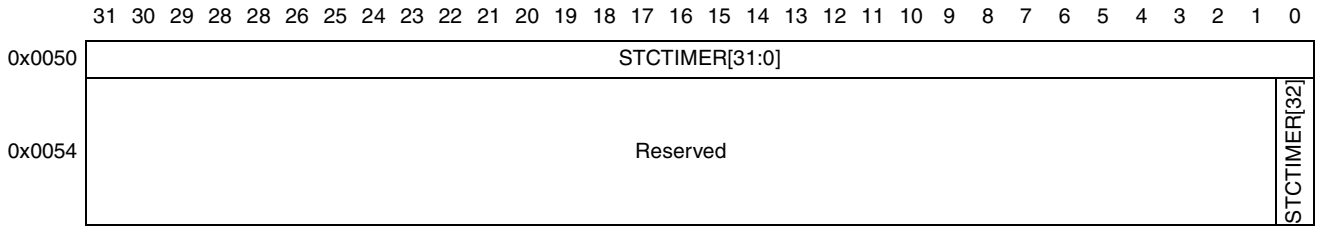
Reset: Undefined

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_STCTIMER Set STC timer



Address: *PTIBaseAddress* + 0x0050 to 0x0054

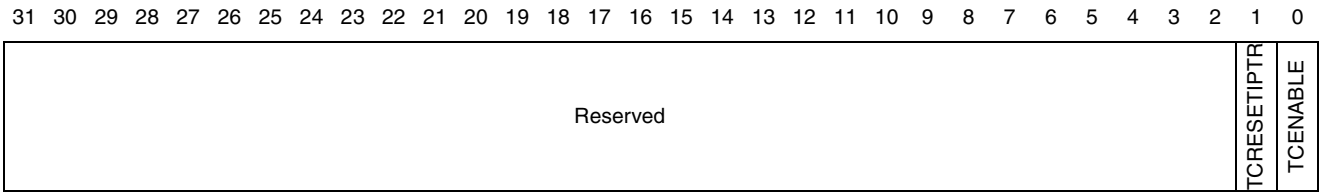
Type: WO

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 STCTIMER[32] before writing to the STCTIMER[31:0] register, as the write to the STCTIMER[31:0] causes the update of the 33-bit counter value.

48.4 Transport controller mode register

PTI_TCMODE **Transport controller mode**



Address: *PTIBaseAddress* + 0x0030

Type: R/W

Reset: Undefined

Description: The PTI_TCMODE register allows the ST20 software to control the transport controller. Under normal operation the software should only set the TCENABLE bit to start the TC executing code.

Resetting the TC instruction pointer

1 Clear the TCENABLE bit.
2 Set the TCRESET bit.
3 Clear the TCRESET bit.
4 Set the TCENABLE bit.

[31:2] **Reserved**

[1] **TCRESETIPTR:** Reset the instruction pointer of the TC (level sensitive).

[0] **TCENABLE**
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

49 DES encryption/decryption system

The DES block encrypts or decrypts blocks of data. The read DMA engine fetches blocks of data from memory and passes these to the PID/KEY search engine and the DES engine. If the data is a packet header, it is PID matched to a key in a lookup table and DES (en)decrypted with that key. The processed data is then written to a buffer FIFO and the write DMA to transfers it back to memory.

49.1 Detailed functional description

49.1.1 Operation overview

This block is a general purpose DES encryption/decryption engine for processing data to and from a hard disk drive. It is designed for memory to memory operations and has read and write DMA channels to support this. Fully programmable header and payload lengths allow the processing of PES packets as well as DVB/DSS transport packets.

Headers bypass the DES engine and are passed in plain text to the block output. They are stored as plain text to allow a header/PID search engine to select the key used by the DES engine. This search engine is capable of matching any of the first 32 bits in a header to a set of eight stored 32-bit values. Each stored value has a pair of keys associated with it, which are also stored locally. The search mechanism is programmable to allow for the automatic PID/key association of DVB and DSS transport streams, and PES packets. Future streams can be matched providing the PID is located in the first 32 bits of the header. A zero length header can be programmed to support encryption of the entire stream.

Odd/even key selection is based on the scrambling mode bits in the header; DVB or DSS mode formats are supported. The keys are written to the key store from an STBus configuration port.

As packets are not necessarily multiples of eight bytes, as required by the DES algorithm, any residue is sent as plain text.

49.1.2 PID/key lookup tables

The PID/key lookup table is a data RAM accessed via a comparison function storing eight PIDs and associated pairs of keys per channel. This equates to 1280 bits of storage per channel, 3840 bits of storage in all. This is logically split into three sections, to allow for the storage of the PID/key pairs of three channels. The same PID with a different key pair may be used in more than one channel at a time.

The table uses the active channel identifier from the DES_CFG register to decide which part of the table is used, and the KEYLUT_MASK bits decide which bits are used for the comparison. When a header word is presented to the block, the PID is compared against each stored PID in turn until a match is found, and the corresponding key data is latched and output to the DES engine. The odd/even scrambling bit contained in the header is used to select one of two possible keys per PID. The format of header selected in the DES_KEYLUT_CFG defines which bits are used for odd or even key selection. It is possible to disable this feature and force even key use as described in the register details, see Chapter 50: *DES encryption/decryption system registers on page 418*.

A search is initialized by a request from the control state machine, once a PID match is obtained this is signalled back to allow DES (en)decryption to start.

If a match is not obtained, then the search engine signals to the control circuitry to send the payload as plain text, that is, bypass the DES engine.

If the mask is set to all zero's for a particular channel then key zero is always used regardless of any embedded PIDs in the header. In this case, the no_match indication used to bypass the engine is disabled.

If a header length of zero is programmed then key zero is always used to encrypt the entire stream. The software must program a nonresidue payload length in this case.

49.1.3 Header bypass

The header data is normally stored as plain text (for PID/KEY retrieval), so it is passed directly to the output FIFO avoiding the DES engine. Bits set in DES_CFG can configure the DES engine such that a variable header and payload size can be handled.

A new transport packet always starts on a fresh DMA burst boundary. Therefore the first piece of data after initialization is always the header, and the DES channel counts through the number of bytes specified in DES_CFG so it can parse the header correctly. A write to the DES_CFG register, bit 4 (SOFT_RESET) resets the control mechanism, so the next data written to the block must be a header. A write to this register when the DES engine is busy returns an error in the STATUS register.

If a header length of zero is programmed and the whole stream is encrypted or decrypted, as would be for PES packets, then DMA start and stop locations are arbitrary controlled by software. In this case, data is treated as a continuous bit stream, and the packet counter just counts multiples of bytes processed, not necessarily packets.

If the payload (de)encryption mode is set to off, the entire packet is passed in plain text via the bypass MUX. If the header is set to zero length, the entire payload is encrypted (with key 0). Encryption mode has priority over header length.

49.1.4 DES encryption/decryption engine

The bytes of data for encryption are passed to this block as pairs of 32-bit words. It implements the Data Encryption Standard (DES) algorithm; this takes two input groups of 32 bits each which form the standard 64 input data word as required by the DES, places it on an accumulator and does an initial bit permutation followed by 16 iterations where the data is combined with the key which is bit rotated by one or two places each time. After a final permutation the value on the accumulator is the (en)decrypted data word, and this is passed to the buffer FIFO for DMA write to the circular buffer. The next pair of 32-bit words is then loaded to form the next input data word.

The decryption works in the same way as the encryption engine but for the direction and the number of places that the keys are rotated each time.

50 DES encryption/decryption system registers

Addresses are provided as *DESBaseAddress* + offset.

The *DESBaseAddress* is:

0x3804 0000.

Table 105: DES register summary

Register	Description	Offset	Type
DES_KEY_xnP_B	Encryption key	0x0000	R/W
DES_KETLUT_PIDxn	PID for channel x key pair n	0x0180	R/W
DES_CFG	DES configuration	0x0200	R/W
DES_INT_PKTCOUNT	Interrupt packet count	0x0204	R/W
DES_STATUS	Status	0x0208	R/W
DES_INT_MASK	Interrupt mask	0x020C	R/W
DES_KEYLUT_CFG	Configuration for key lookup table	0x0210	R/W
DES_KEYLUT_MASK_n	Key lookup table PID mask for channel n	0x0214, 0x0218, 0x021C	R/W
DES_DMA_READ	DMA read address	0x0300	R/W
DES_DMA_WRITE	DMA write address	0x0304	R/W
DES_DMA_READ_COUNT	DMA read word count	0x0308	R/W
DES_DMA_CFG	DMA configuration	0x030C	R/W

DES_KEY_xnP_B Encryption 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
ENCRYPTION_KEY																															

Address: *DESBaseAddress* + 0x0000

Type: Read/Write

Reset: 0000

Description:

[31:0] **ENCRYPTION_KEY**

B = low: The low byte of: P = e: The even key associated with PID no. *n* of channel *x*

P = o: The odd key associated with PID no. *n* of channel *x*

B = high: The high byte of: P = e: The even key associated with PID no. *n* of channel *x*

P = o: The odd key associated with PID no. *n* of channel *x*

DES_KETLUT_PIDxn PID for channel x key pair 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
PID																															

Address: *DESBaseAddress* + 0x0180

Type: Read/Write

Reset: 0000

Description:

[31:0] **PID**

The PID associated with channel x key pair *n.*

This value is masked by DES_KEYLUT_MASK_x and compared with the first header word by the search engine. A match indicates the DES engine should use key pair *n.* (odd or even key usage is decided by the KEYLUT_CFG register)

To reduce the search latency, where possible, software should program the most used PIDs into the lower x locations, as the hardware search engine tests these first.

DES_CFG DES 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
PAYLOAD_SIZE																HEADER_SIZE								Reserved			SOFT_RESET	ACTIVE_CHANNEL		Reserved	DE_NOT_ENCRYPT

Address: *DESBaseAddress* + 0x0200

Type: Read/Write

Reset: 0x0000 0000

Description:

[31:16] **PAYLOAD_SIZE**

Packet payload size (8 to 64 Kbytes)[a]

[15:8] **HEADER_SIZE**

Packet header size (0 to 255 bytes)[b]
For a header size of zero, key 0 is selected

[7:5] **Reserved**

[4] **SOFT_RESET**
Sends a one cycle reset signal to all counters or state machines on the data path only

[3:2] **ACTIVE_CHANNEL**
00: Channel a
01: Channel b
10: Channel c
11: Error
Used to set the active section of the key LUT

[1] **Reserved**

[0] **DE_NOT_ENCRYPT**
0: Encrypt
1: Decrypt

a. The bottom three bits can be thought of as a residue count, to be passed as plain text. Unless processing nonTS packets, these are always zero.

b. Minimum size: payloads of fewer than eight bytes are entirely residue, and can therefore be sent in the blockmove DMA mode.

DES_INT_PKTCOUNT Interrupt packet 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																INT_PKT_COUNT															

Address: *DESBaseAddress* + 0x0204
Type: Read/Write
Reset: 0x0
Description:

[31:16] **Reserved**

[15:0] **INT_PKT_COUNT**
Set interrupt after processing *n* packets
Writing a value to this register clears the interrupt

DES_STATUS 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											ACCESS_ERROR	FIFO_FULL	DMA_WRITE_STATUS	DMA_READ_STATUS	ENGINE_STATUS	PACKET_COUNT																

Address: *DESBaseAddress* + 0x0208
Type: Read/Write
Reset: 0 unless otherwise stated
Description:

[31:21] **Reserved**

[20] **ACCESS_ERROR**
1: CFG register changed when engine still busy

[19] **FIFO_FULL**
0: Output FIFO not full
1: Output FIFO full: latched, cleared on read.

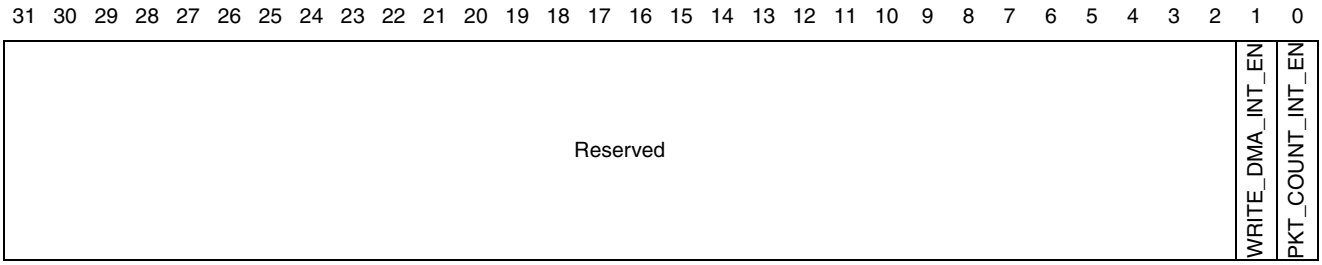
[18] **DMA_WRITE_STATUS**
0: Idle
1: Busy

[17] **DMA_READ_STATUS**
0: Idle
1: Busy

[16] **ENGINE_STATUS**
0: Idle
1: Busy

[15:0] **PACKET_COUNT**
Number of packets processed (wraps at 64 Kbytes)
Reset state: 0000

DES_INT_MASK Interrupt mask



Address: *DESBaseAddress* + 0x020C
Type: Read/Write
Reset: 0x0
Description:

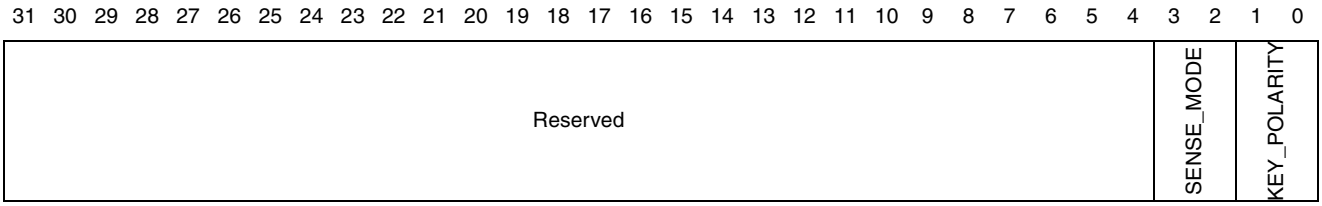
[31:3] **Reserved**

[2] **WRITE_DMA_INT_EN**
Enable interrupt on DMA write channel completing

[1] **PKT_COUNT_INT_EN**
Enable interrupt after processing *n* packets

[0] **Reserved**

DES_KEYLUT_CFG Configuration for key lookup table



Address: *DESBaseAddress* + 0x0210
Type: Read/Write
Reset: 00
Description:

[31:4] **Reserved**

[3:2] **SENSE_MODE**
00: ITU-T rec H.222.0 / ISO/IEC 13818 transport packet mode.
Bits 25:24 of packet header 10: Even, 01: Odd
01: DSS transport packet mode
Bit 3 of the packet header, 0: Even 1: Odd
10, 11: Not defined

[1:0] **KEY_POLARITY**
00: Force even key
01,10: Allow automatic key polarity change.
11: Force odd key

DES_KEYLUT_MASK_n Key lookup table PID mask 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
PID_MASK																															

Address: *DESBaseAddress* + 0x0214, 0x0218, 0x021C

Type: Read/Write

Reset: 0x0000

Description: Key lookup table PID mask for channel *n*.

Any bits that are set are used in the comparison between the first 32 bits of the header and the words stored in the PID lookup RAM. Bits not set are ignored when searching for a match. The active channel bits in the DES_CFG register select which mask and which section of the LUT is used.

Setting this register to zero selects channel *n* key 0.

DES_DMA_READ DMA 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
DMA_READ_ADDRESS																															

Address: *DESBaseAddress* + 0x0300

Type: Read/Write

Reset: 0x0000

Description:

[31:0] **DMA_READ_ADDRESS**
DMA read address

DES_DMA_WRITE DMA 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
DMA_WRITE_ADDRESS																															

Address: *DESBaseAddress* + 0x0304

Type: Read/Write

Reset: 0x0000

Description:

[31:0] **DMA_WRITE_ADDRESS**
DMA write address

DES_DMA_READ_COUNT DMA read word 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
DMA_READ_WORD_COUNT																															

Address: *DESBaseAddress* + 0x0308
Type: Read/Write
Reset: 0x0000
Description:

[31:0] **DMA_READ_WORD_COUNT**
DMA read transfer size. Sets the required transfer size, and reports the number of words left to transfer.

DES_DMA_CFG DMA 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																												DMA_BLOCK_MOVE	Reserved	NOT_INC_READ_POINTER	NOT_INC_WRITE_POINTER

Address: *DESBaseAddress* + 0x030C
Type: Read/Write
Reset: 0
Description:

[31:4] **Reserved**

[3] **DMA_BLOCK_MOVE**
0: Send data via the DES engine and associated control
1: Bypass all the DES circuitry, write data directly to the output FIFO. To allow the DMA engines to operate as a read write DMA channel when the DES engine is not used.

[2] **DMA_REQ_CONTROL**
0: Ignore status of DMA_REQ signal from TSMUX
1: Use DMA_REQ to control DMA requests. This is to prevent the DMA making requests of the TSMUX when it is busy, blocking the interconnect

[1] **NOT_INC_READ_POINTER**
0: Autoincrement DMA read pointer
1: static write pointer

[0] **NOT_INC_WRITE_POINTER**
0: Autoincrement DMA write pointer
1: static write pointer used for DMA transfers to the TSMUX SW port

50.1 PID/key registers

Table 106: PID/key registers

Register	Base address	Size	R/W	Description
DES_KEY_A0E_LOW	0x0000	32	W	Channel A, even key 0, low word
DES_KEY_A0O_LOW	0x0004	32	W	Channel A, odd key 0, low word
DES_KEY_A0E_HI	0x0008	32	W	Channel A, even key 0, high word
DES_KEY_A0O_HI	0x000C	32	W	Channel A, odd key 0, high word
DES_KEY_A1E_LOW	0x0010	32	W	Channel A, even key 1, low word
DES_KEY_A1O_LOW	0x0014	32	W	Channel A, odd key 1, low word
DES_KEY_A1E_HI	0x0018	32	W	Channel A, even key 1, high word
DES_KEY_A1O_HI	0x001C	32	W	Channel A, odd key 1, high word
DES_KEY_A2E_LOW	0x0020	32	W	Channel A, even key 2, low word
DES_KEY_A2O_LOW	0x0024	32	W	Channel A, odd key 2, low word
DES_KEY_A2E_HI	0x0028	32	W	Channel A, even key 2, high word
DES_KEY_A2O_HI	0x002C	32	W	Channel A, odd key 2, high word
DES_KEY_A3E_LOW	0x0030	32	W	Channel A, even key 3, low word
DES_KEY_A3O_LOW	0x0034	32	W	Channel A, odd key 3, low word
DES_KEY_A3E_HI	0x0038	32	W	Channel A, even key 3, high word
DES_KEY_A3O_HI	0x003C	32	W	Channel A, odd key 3, high word
DES_KEY_A4E_LOW	0x0040	32	W	Channel A, even key 4, low word
DES_KEY_A4O_LOW	0x0044	32	W	Channel A, odd key 4, low word
DES_KEY_A4E_HI	0x0048	32	W	Channel A, even key 4, high word
DES_KEY_A4O_HI	0x004C	32	W	Channel A, odd key 4, high word
DES_KEY_A5E_LOW	0x0050	32	W	Channel A, even key 5, low word
DES_KEY_A5O_LOW	0x0054	32	W	Channel A, odd key 5, low word
DES_KEY_A5E_HI	0x0058	32	W	Channel A, even key 5, high word
DES_KEY_A5O_HI	0x005C	32	W	Channel A, odd key 5, high word
DES_KEY_A6E_LOW	0x0060	32	W	Channel A, even key 6, low word
DES_KEY_A6O_LOW	0x0064	32	W	Channel A, odd key 6, low word
DES_KEY_A6E_HI	0x0068	32	W	Channel A, even key 6, high word
DES_KEY_A6O_HI	0x006C	32	W	Channel A, odd key 6, high word
DES_KEY_A7E_LOW	0x0070	32	W	Channel A, even key 7, low word
DES_KEY_A7O_LOW	0x0074	32	W	Channel A, odd key 7, low word
DES_KEY_A7E_HI	0x0078	32	W	Channel A, even key 7, high word
DES_KEY_A7O_HI	0x007C	32	W	Channel A, odd key 7, high word
DES_KEY_B0E_LOW	0x0080	32	W	Channel B, even key 0, low word
DES_KEY_B0O_LOW	0x0084	32	W	Channel B, odd key 0, low word
DES_KEY_B0E_HI	0x0088	32	W	Channel B, even key 0, high word
DES_KEY_B0O_HI	0x008C	32	W	Channel B, odd key 0, high word
DES_KEY_B1E_LOW	0x0090	32	W	Channel B, even key 1, low word
DES_KEY_B1O_LOW	0x0094	32	W	Channel B, odd key 1, low word
DES_KEY_B1E_HI	0x0098	32	W	Channel B, even key 1, high word
DES_KEY_B1O_HI	0x009C	32	W	Channel B, odd key 1, high word

Table 106: PID/key registers

Register	Base address	Size	R/W	Description
DES_KEY_B2E_LOW	0x00A0	32	W	Channel B, even key 2, low word
DES_KEY_B2O_LOW	0x00A4	32	W	Channel B, odd key 2, low word
DES_KEY_B2E_HI	0x00A8	32	W	Channel B, even key 2, high word
DES_KEY_B2O_HI	0x00AC	32	W	Channel B, odd key 2, high word
DES_KEY_B3E_LOW	0x00B0	32	W	Channel B, even key 3, low word
DES_KEY_B3O_LOW	0x00B4	32	W	Channel B, odd key 3, low word
DES_KEY_B3E_HI	0x00B8	32	W	Channel B, even key 3, high word
DES_KEY_B3O_HI	0x00BC	32	W	Channel B, odd key 3, high word
DES_KEY_B4E_LOW	0x00C0	32	W	Channel B, even key 4, low word
DES_KEY_B4O_LOW	0x00C4	32	W	Channel B, odd key 4, low word
DES_KEY_B4E_HI	0x00C8	32	W	Channel B, even key 4, high word
DES_KEY_B4O_HI	0x00CC	32	W	Channel B, odd key 4, high word
DES_KEY_B5E_LOW	0x00D0	32	W	Channel B, even key 5, low word
DES_KEY_B5O_LOW	0x00D4	32	W	Channel B, odd key 5, low word
DES_KEY_B5E_HI	0x00D8	32	W	Channel B, even key 5, high word
DES_KEY_B5O_HI	0x00DC	32	W	Channel B, odd key 5, high word
DES_KEY_B6E_LOW	0x00E0	32	W	Channel B, even key 6, low word
DES_KEY_B6O_LOW	0x00E4	32	W	Channel B, odd key 6, low word
DES_KEY_B6E_HI	0x00E8	32	W	Channel B, even key 6, high word
DES_KEY_B6O_HI	0x00EC	32	W	Channel B, odd key 6, high word
DES_KEY_B7E_LOW	0x00F0	32	W	Channel B, even key 7, low word
DES_KEY_B7O_LOW	0x00F4	32	W	Channel B, odd key 7, low word
DES_KEY_B7E_HI	0x00F8	32	W	Channel B, even key 7, high word
DES_KEY_B7O_HI	0x00FC	32	W	Channel B, odd key 7, high word
DES_KEY_C0E_LOW	0x0100	32	W	Channel C, even key 0, low word
DES_KEY_C0O_LOW	0x0104	32	W	Channel C, odd key 0, low word
DES_KEY_C0E_HI	0x0108	32	W	Channel C, even key 0, high word
DES_KEY_C0O_HI	0x010C	32	W	Channel C, odd key 0, high word
DES_KEY_C1E_LOW	0x0110	32	W	Channel C, even key 1, low word
DES_KEY_C1O_LOW	0x0114	32	W	Channel C, odd key 1, low word
DES_KEY_C1E_HI	0x0118	32	W	Channel C, even key 1, high word
DES_KEY_C1O_HI	0x011C	32	W	Channel C, odd key 1, high word
DES_KEY_C2E_LOW	0x0120	32	W	Channel C, even key 2, low word
DES_KEY_C2O_LOW	0x0124	32	W	Channel C, odd key 2, low word
DES_KEY_C2E_HI	0x0128	32	W	Channel C, even key 2, high word
DES_KEY_C2O_HI	0x012C	32	W	Channel C, odd key 2, high word
DES_KEY_C3E_LOW	0x0130	32	W	Channel C, even key 3, low word
DES_KEY_C3O_LOW	0x0134	32	W	Channel C, odd key 3, low word
DES_KEY_C3E_HI	0x0138	32	W	Channel C, even key 3, high word
DES_KEY_C3O_HI	0x013C	32	W	Channel C, odd key 3, high word
DES_KEY_C4E_LOW	0x0140	32	W	Channel C, even key 4, low word

Table 106: PID/key registers

Register	Base address	Size	R/W	Description
DES_KEY_C4O_LOW	0x0144	32	W	Channel C, odd key 4, low word
DES_KEY_C4E_HI	0x0148	32	W	Channel C, even key 4, high word
DES_KEY_C4O_HI	0x014C	32	W	Channel C, odd key 4, high word
DES_KEY_C5E_LOW	0x0150	32	W	Channel C, even key 5, low word
DES_KEY_C5O_LOW	0x0154	32	W	Channel C, odd key 5, low word
DES_KEY_C5E_HI	0x0158	32	W	Channel C, even key 5, high word
DES_KEY_C5O_HI	0x015C	32	W	Channel C, odd key 5, high word
DES_KEY_C6E_LOW	0x0160	32	W	Channel C, even key 6, low word
DES_KEY_C6O_LOW	0x0164	32	W	Channel C, odd key 6, low word
DES_KEY_C6E_HI	0x0168	32	W	Channel C, even key 6, high word
DES_KEY_C6O_HI	0x016C	32	W	Channel C, odd key 6, high word
DES_KEY_C7E_LOW	0x0170	32	W	Channel C, even key 7, low word
DES_KEY_C7O_LOW	0x0174	32	W	Channel C, odd key 7, low word
DES_KEY_C7E_HI	0x0178	32	W	Channel C, even key 7, high word
DES_KEY_C7O_HI	0x017C	32	W	Channel C, odd key 7, high word
DES_KEYLUT_PIDA0	0x0180	32	R/W	Key look up table PID A0
DES_KEYLUT_PIDA1	0x0184	32	R/W	Key look up table PID A1
DES_KEYLUT_PIDA2	0x0188	32	R/W	Key look up table PID A2
DES_KEYLUT_PIDA3	0x018C	32	R/W	Key look up table PID A3
DES_KEYLUT_PIDA4	0x0190	32	R/W	Key look up table PID A4
DES_KEYLUT_PIDA5	0x0194	32	R/W	Key look up table PID A5
DES_KEYLUT_PIDA6	0x0198	32	R/W	Key look up table PID A6
DES_KEYLUT_PIDA7	0x019C	32	R/W	Key look up table PID A7
DES_KEYLUT_PIDB0	0x01A0	32	R/W	Key look up table PID B0
DES_KEYLUT_PIDB1	0x01A4	32	R/W	Key look up table PID B1
DES_KEYLUT_PIDB2	0x01A8	32	R/W	Key look up table PID B2
DES_KEYLUT_PIDB3	0x01AC	32	R/W	Key look up table PID B3
DES_KEYLUT_PIDB4	0x01B0	32	R/W	Key look up table PID B4
DES_KEYLUT_PIDB5	0x01B4	32	R/W	Key look up table PID B5
DES_KEYLUT_PIDB6	0x01B8	32	R/W	Key look up table PID B6
DES_KEYLUT_PIDB7	0x01BC	32	R/W	Key look up table PID B7
DES_KEYLUT_PIDC0	0x01C0	32	R/W	Key look up table PID C0
DES_KEYLUT_PIDC1	0x01C4	32	R/W	Key look up table PID C1
DES_KEYLUT_PIDC2	0x01C8	32	R/W	Key look up table PID C2
DES_KEYLUT_PIDC3	0x01CC	32	R/W	Key look up table PID C3
DES_KEYLUT_PIDC4	0x01D0	32	R/W	Key look up table PID C4
DES_KEYLUT_PIDC5	0x01D4	32	R/W	Key look up table PID C5
DES_KEYLUT_PIDC6	0x01D8	32	R/W	Key look up table PID C6
DES_KEYLUT_PIDC7	0x01DC	32	R/W	Key look up table PID C7

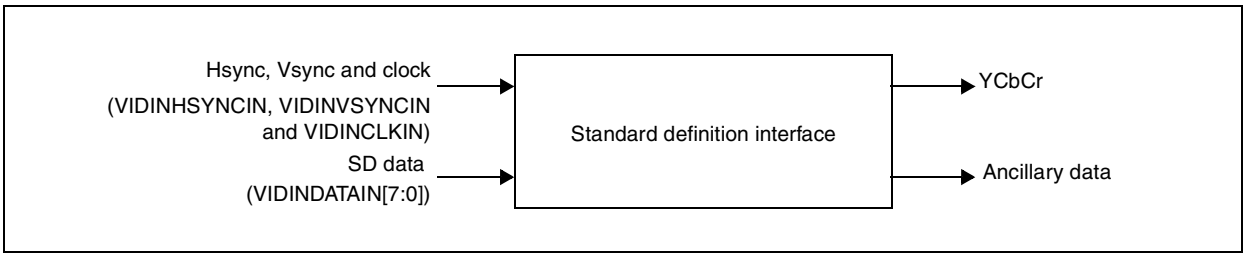
51 Direct access arrangement modem (DAA)

See document *32.4 MHz Differential-Link Interface DAA - EMBEDDED SYSTEM-SIDE DAA MODULE SPECIFICATION*.

52 Digital video port (DVP)

Video data enters the device through one standard-definition interface, the digital video port (DVP), which is described in this chapter.

Figure 95: Video input interface



52.1 Features

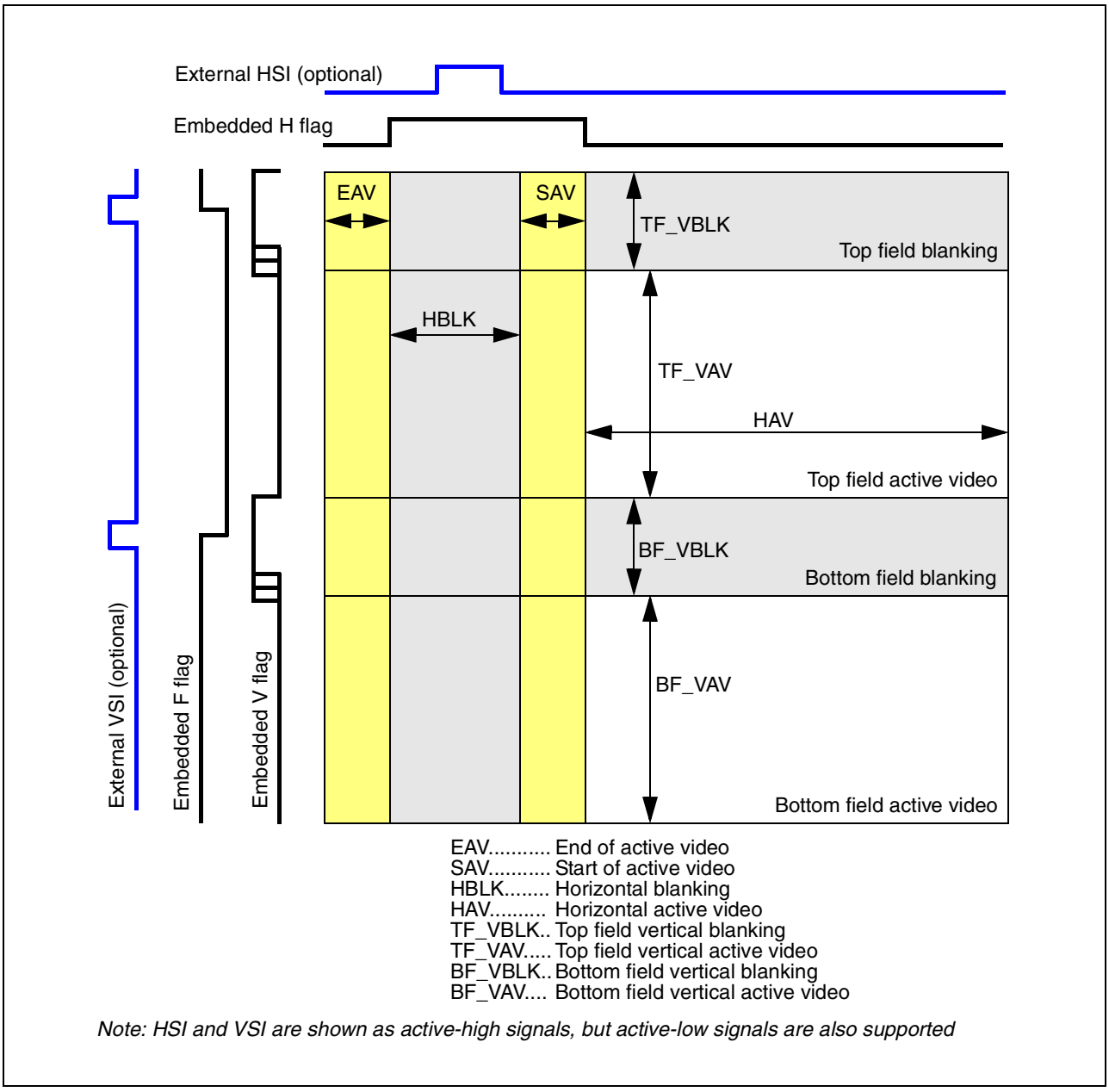
The main DVP features are:

- 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.

52.2 Video decoder

The frame architecture for an ITU-R BT.656 or ANSI/SMPTE 293M-1996 compliant video is shown in Figure 96. Y, Cb, Cr are multiplexed at twice the pixel rate on an 8-bit bus, according to a 4:2:2 sampling pattern

Figure 96: 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 107

Table 107: 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 108 shows typical frame/field parameters for popular standard definition formats.

Table 108: ITU-R BT656 parameters for 525/60i and 625/50i formats

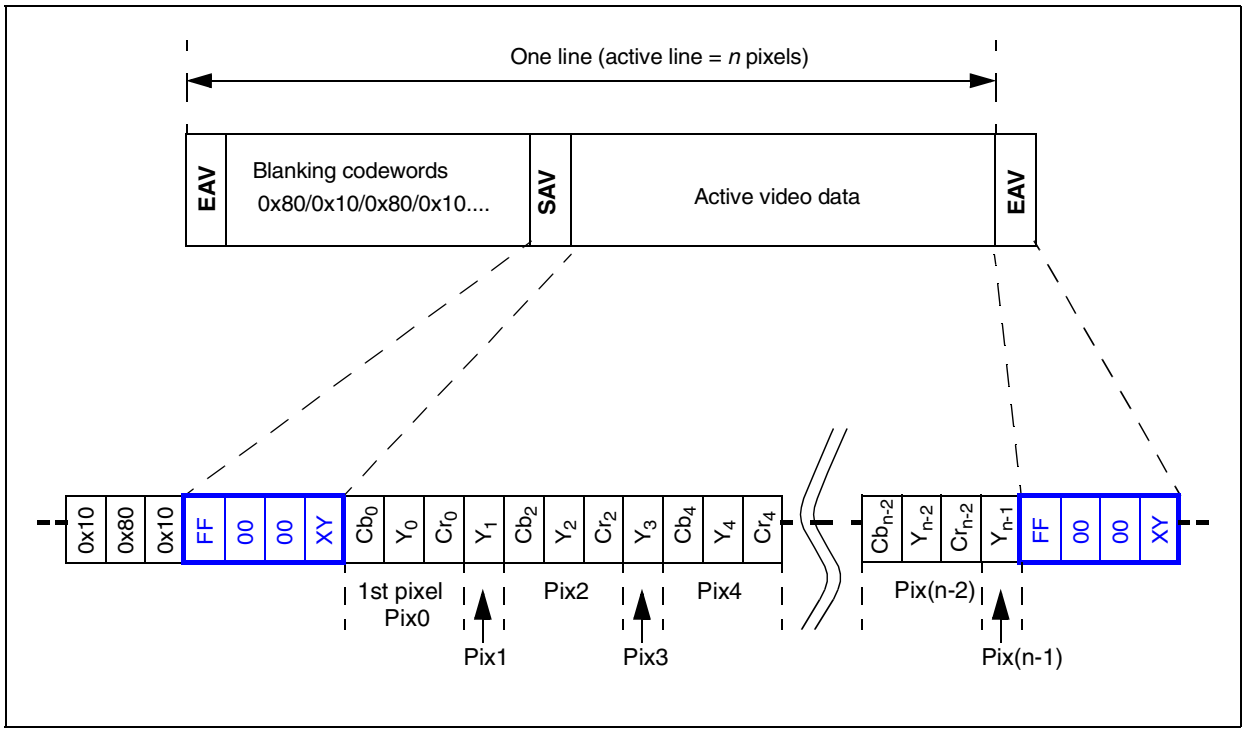
Standard	Parameter	Size	Unit
ITU-R BT.656 525-line interlaced	PixClk frequency	13.5	MHz
	EAV	4	PixClk2X cycles
	SAV	4	
	HBLK	268	
	HAV	1440	
	TF_VBLK	19(23)	lines
	BF_BLK	19(22)	
	TF_VAV	244(240)	
	BF_VAV	243(240)	
ITU-R BT.656 625-line interlaced	PixClk frequency	13.5	MHz
	EAV	4	PixClk2X cycles
	SAV	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 (640x480 active area or 768x576 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 97.

Figure 97: 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.

52.3 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: GAM_DVP_ABA (ancillary data base address), and GAM_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 GAM_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).

53 Digital video port (DVP) registers

Addresses are provided as *DVPBaseAddress* + offset.

The *DVPBaseAddress* is:

0x3820 0000.

The relative register map is as given below:

Figure 98: Register mapping

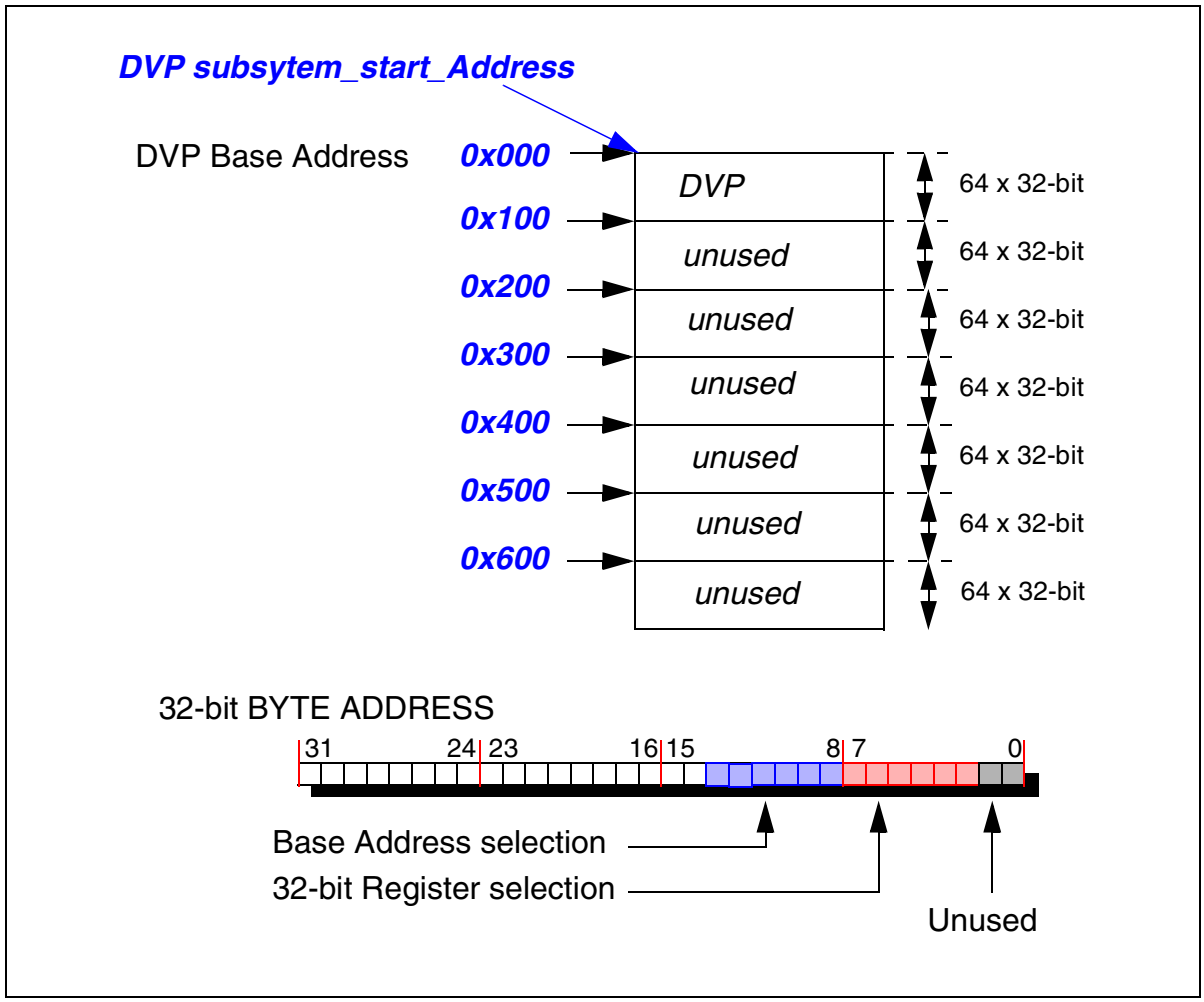


Table 109: DVP register summary

Register	Description	Offset	Type
GAM_DVP_CTL	Control register	0x00	
GAM_DVP_TFO	Top Field Offset (h & v, wrt (0,0))	0x04	
GAM_DVP_TFS	Top Field Stop (h & v, wrt (0,0))	0x08	
GAM_DVP_BFO	Bottom Field Offset (h & v, wrt (0,0))	0x0C	
GAM_DVP_BFS	Bottom Field Stop (h & v, wrt (0,0))	0x10	
GAM_DVP_VTP	Video Top Pointer (memory location for top-left pixel)	0x14	
GAM_DVP_VBP	Video Bottom Pointer (memory location for top-left pixel)	0x18	
GAM_DVP_VMP	Video Memory Pitch	0x1C	R/W
GAM_DVP_CVS	Captured Video Size	0x20	R/W
GAM_DVP_VSD	Vertical synchro delay	0x24	R/W
GAM_DVP_HSD	Horizontal synchro delay	0x28	R/W
GAM_DVP_HLL	Half line length	0x2C	R/W
-	Reserved	0x30 - 0x97	R/W
GAM_DVP_ITM	DVP interrupt mask	0x98	R/W
GAM_DVP_ITS	DVP interrupt status	0x9C	RO
GAM_DVP_STA	DVP interrupt register	0xA0	RO
GAM_DVP_LNSTA	Line Number STAtus	0xA4	RO
GAM_DVP_HLFLN	Half lines per field	0xA8	R/W
GAM_DVP_ABA	Ancillary data Base Address	0xAC	R/W
GAM_DVP_AEA	Ancillary data End Address	0xB0	R/W
GAM_DVP_APS	Ancillary data Page Size	0xB4	R/W
GAM_DVP_PKZ	Packet Size on STBus protocol	0xFC	R/W

53.1 Registers description

GAM_DVP_CTL 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_REQUEST	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_POLARITY	H_REF_POLARITY	EXTERNAL_SYNC	VSYNC_BOT_HALF_LINE_EN	EXT_SYNC_POLARITY	ENA_ANCILLARY_DATA	ENA_VIDEO

Address: *DVPBaseAddress* + 0x00

Type: R/W

Buffer: Double-buffered: update on DVP V_INIT event, except if indicated by ** (immediate)

Reset: 0x8008 0000

Description: The DVP Control register 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 GAM_DVP_CTL, then DVP_RST must be cleared to be able to take into account synchronization inputs (external and embedded).

[31] **DVP_RST**
1: Synchro input disabled (either external or embedded)
0: Synchro inputs enabled

[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_REQUEST**[a]
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 when 8 bits 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_POLARITY**[a]
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_POLARITY**[a]
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] **EXTERNAL_SYNC**[a]
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_POLARITY**[a]
0: negative for both horizontal and vertical (no meaning if EXTERNAL_SYNC= 0)
1: positive for both horizontal and vertical

[1] **ENA_ANCILLARY_DATA**
0: No ancillary data is captured
1: Ancillary data is captured into the device local memory

[0] **ENA_VIDEO**
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.

GAM_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: 00000000

Description: The DVP Top field offset register 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

GAM_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: The DVP Top Field Stop register 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

GAM_DVP_BFO X-Y Bottom 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* + 0x0C

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Bottom Field Offset register 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

GAM_DVP_BFS X-Y Bottom 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* + 0x10

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Bottom Field Stop register 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.

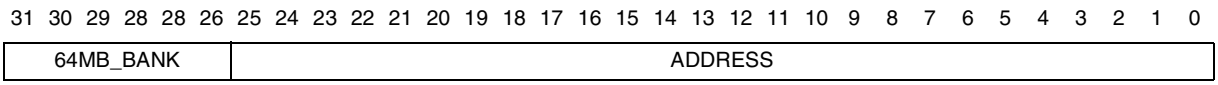
[31:28] **Reserved**

[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**

GAM_DVP_VTP Video top field memory pointer



Address: *DVP BaseAddress* + 0x14

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

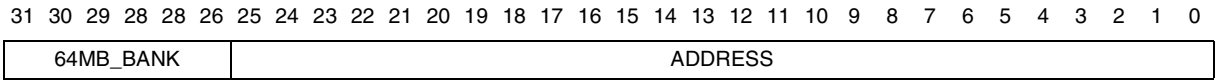
Reset: 0

Description: The DVP Video Top field Pointer register is a 32-bit register containing 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.

GAM_DVP_VBP Video bottom field memory pointer



Address: *DVP BaseAddress* + 0x18

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

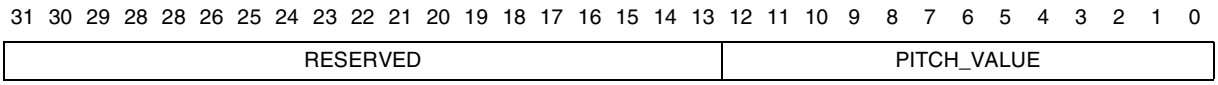
Reset: 0

Description: The DVP Video Bottom field Pointer register is a 32-bit register containing 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.

GAM_DVP_VMP Video memory pitch



Address: *DVP BaseAddress* + 0x1C

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Video Memory Pitch register contains the memory pitch for the captured video, as stored in the memory.

[31:13] **Reserved**

[12:0] **PITCH_VALUE**: Memory pitch for the captured video.
Note: the pitch is the distance inside the memory, in byte unit, between two vertically adjacent pixels within a field

GAM_DVP_CVS Captured video window size

31 30	29 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
BOT_SIZE_RELATION	Reserved	TOP_VIDEO_WINDOW_HEIGHT	Reserved	PIXMAP_WIDTH

Address: *DVP BaseAddress* + 0x20

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Captured Video Size register provides the size of the displayed pixmap 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_VIDEO_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

GAM_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_DELAY												Reserved												VSYNC_V_TOP_DELAY										

Address: *DVP BaseAddress* + 0x24

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Vertical Synchronization delay register 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_DELAY value is given in DVP_HSD register (see next register, GAM_DVP_HSD)):

TOP FIELD:

VSYNC_H_DELAY is the delay of vsout from sample counter clear,

VSYNC_V_TOP_DELAY is the delay from line counter clear;

The total delay from sample counter clear is:

VSYNC_H_DELAY + VSYNC_V_TOP_DELAY * NB_SAMPLES_PER_LINE

BOTTOM FIELD:

External synchros or (internal synchros with VSYNC_BOT_HALF_LINE_EN = 0):

VSYNC_H_DELAY is the delay of VSOUT in samples units from external vertical synchro input.

VSYNC_V_BOT_DELAY is the delay from line counter clear;

the total number of samples delay from vertical synchronization input is:

VSYNC_H_DELAY + VSYNC_V_BOT_DELAY * NB_SAMPLES_PER_LINE

internal synchros with VSYNC_BOT_HALF_LINE_EN = 1 this delay becomes:

VSYNC_H_DELAY + VSYNC_V_BOT_DELAY * NB_SAMPLES_PER_LINE + HALF_LINE_LENGTH

to provide VSOUT on bottom field at the middle of last line of top field

Note: 1 VSYNC_V_TOP_DELAY *and* VSYNC_V_BOT_DELAY *values* ***must*** *be lower than the number of lines per field*

*2 This register is not double buffered (*VSYNC_V_TOP_DELAY *and* VSYNC_V_BOT_DELAY *are taken into account when loaded by the CPU)*

[31:28] **Reserved**

[26:16] **V_BOT_DELAY**: vertical delay for bottom field, in line units

[15:11] **Reserved**

[10:0] **V_TOP_DELAY**: vertical delay for top field, in sample unit

GAM_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_DELAY													Reserved			HSYNC_H_DELAY												

Address: *DVP BaseAddress* + 0x28

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Horizontal Synchronization delay register provides the size of the horizontal delay of HSOUT:

HSYNC_H_DELAY is the delay of hsout in samples units from the sample counter clear signal

VSYNC_H_DELAY is the delay of VSOUT in samples units from the sample counter clear signal (see GAM_DVP_VSD register).

Note: HSYNC_H_DELAY *and* VSYNC_H_DELAY *values* ***must*** *be lower than the number of samples per line.*

Note: *This register is not double buffered (*VSYNC_V_TOP_DELAY *and* VSYNC_V_BOT_DELAY *are taken into account when loaded by the CPU).*

[31:29] **Reserved**

[28:16] **VSYNC_H_DELAY**: horizontal delayof HSOUT, expressed as number of 27 MHz samples.

[15:13] **Reserved**

[12:0] **HSYNC_H_DELAY**: horizontal delayof HSOUT, expressed as number of 27 MHz samples (see DVP_VSD register description).

GAM_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_LENGTH											

Address: *DVP BaseAddress* + 0x2C

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

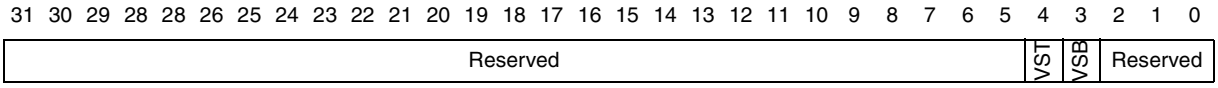
Description: The DVP Half Line Length register provides the number of samples for in a half line. (This is used internally to have vsout shifted one half line after HSOUT, in embedded synchros mode when VSYNC_BOT_HALF_LINE_EN = 1 in the control register).

Description: Note that this is equivalent to the number of pixels per line.

[31:28] **Reserved**

[11:0] **HALF_LINE_LENGTH**: half line length, in sample unit

GAM_DVP_ITM Interrupt mask



Address: *DVPn BaseAddress* + 0x98

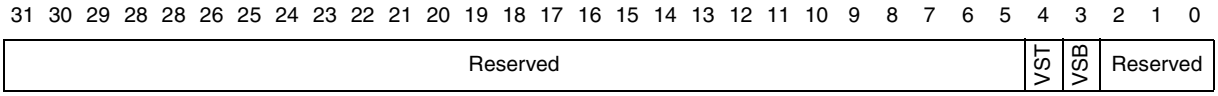
Type: R/W

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.

GAM_DVP_ITS Interrupt status



Address: *DVPn BaseAddress* + 0x9C

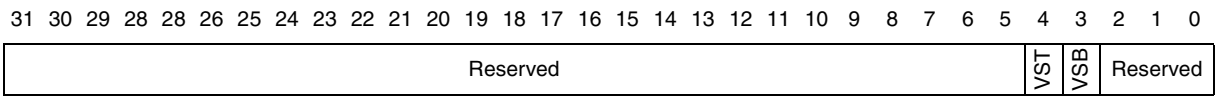
Type: RO

Buffer: -

Reset: 0

Description: When a bit in the DVP_STA register changes from 0 to 1, the corresponding bit in the DVP_ITS register is set, independent of the state of DVP_ITM. If any set DVP_ITS 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 returns deasserted.

GAM_DVP_STA Status



Address: *DVPn BaseAddress* + 0xA0

Type: RO

Buffer: -

Reset: 0

Description: This register contains a set of bits which represents the status of the DVP. Any change from 0 to 1 of any 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**

GAM_DVP_LNSTA Line number 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																					line_number_status[10:0]										

Address: *VTG BaseAddress* + 0xA4
Type: RO
Buffer: -
Reset: 0
Description: This register is loaded on VTG vref output signal (vertical synchronization) with the value of DVP line counter

GAM_DVP_HLFLN Half lines per vertical field

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																								HLFLN[11:0]							

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_NOTOK 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.

GAM_DVP_ABA Ancillary data base 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																														write 0	
ANC_DATA_ADDR																															

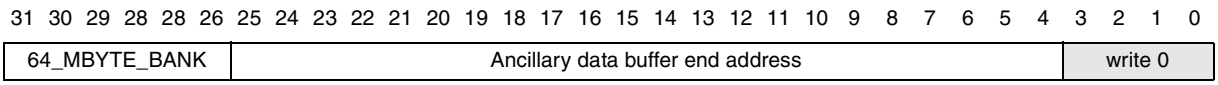
Address: *DVP BaseAddress* + 0xAC
Type: R/W
Buffer: Double-buffered, update on DVP V_INIT event
Reset: 0
Description: The DVP Ancillary data Base Address register is a 32-bit register containing the memory location for the first ancillary data page. The buffer must be aligned on a 128-bit word address 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] write 0

GAM_DVP_AEA Ancillary data end address



Address: *DVP BaseAddress* + 0xB0

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

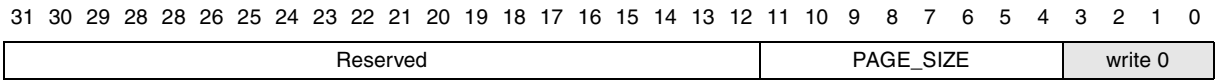
Description: The DVP Ancillary data End Address register is a 32-bit register containing the last memory location for the ancillary data buffer (128-bit word address, then the address counter wraps to the Base Address location).

[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] write 0

GAM_DVP_APS Ancillary data page size



Address: *DVP BaseAddress* + 0xB4

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

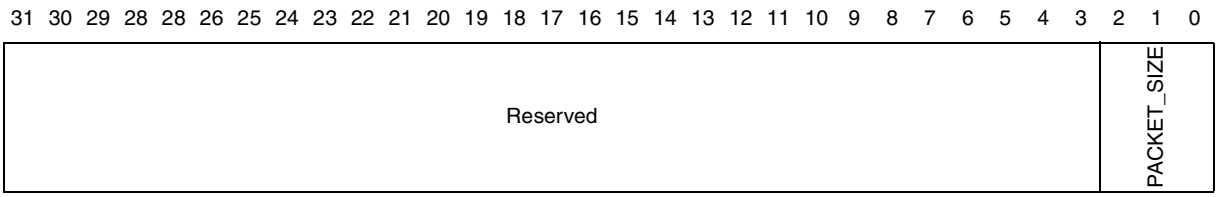
Description: The DVP Ancillary data Page Size register provides the size in bytes, of the memory page used for a single ancillary data packet. The size granularity is 16 bytes, the maximum size is 4096 bytes.

[31:12] **Reserved**

[11:4] **PAGE_SIZE**: Memory size allocated to an ancillary data packet

[3:0] write 0

GAM_DVP_PKZ Maximum packet size



Address: *DVP BaseAddress* + 0xFC

Type: R/W

Buffer: Double-buffered, update on DVP V_INIT event

Reset: 0

Description: The DVP Maximum Packet Size register is a 3-bit register for controlling the maximum size of a data packet during an STBus transaction. This register must be set to 0 in the ST40GFX1.

[31:3] **Reserved**

[2:0] **PACKET_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

54 MPEG video decoder

54.1 Overview

The STi7710 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 MPEG video decoder needs CPU support for each picture.

When a decoding task is launched by the CPU, the MPEG 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 MPEG video decoder supports three extra features which are not part of the MPEG standard:

- overwrite mode, see Section 54.5.3: *Picture decoding configuration* *on page 458*,
- possible simplified B decoding, see Section 54.5.7: *Simplified B decoding* *on page 465*,
- error statistics, see Section 54.5.6: *Error statistics* *on page 464*.

54.2 Main functions

Figure 99 shows the video decoder in a typical environment.

Figure 99: MPEG video decoder in the STi7710

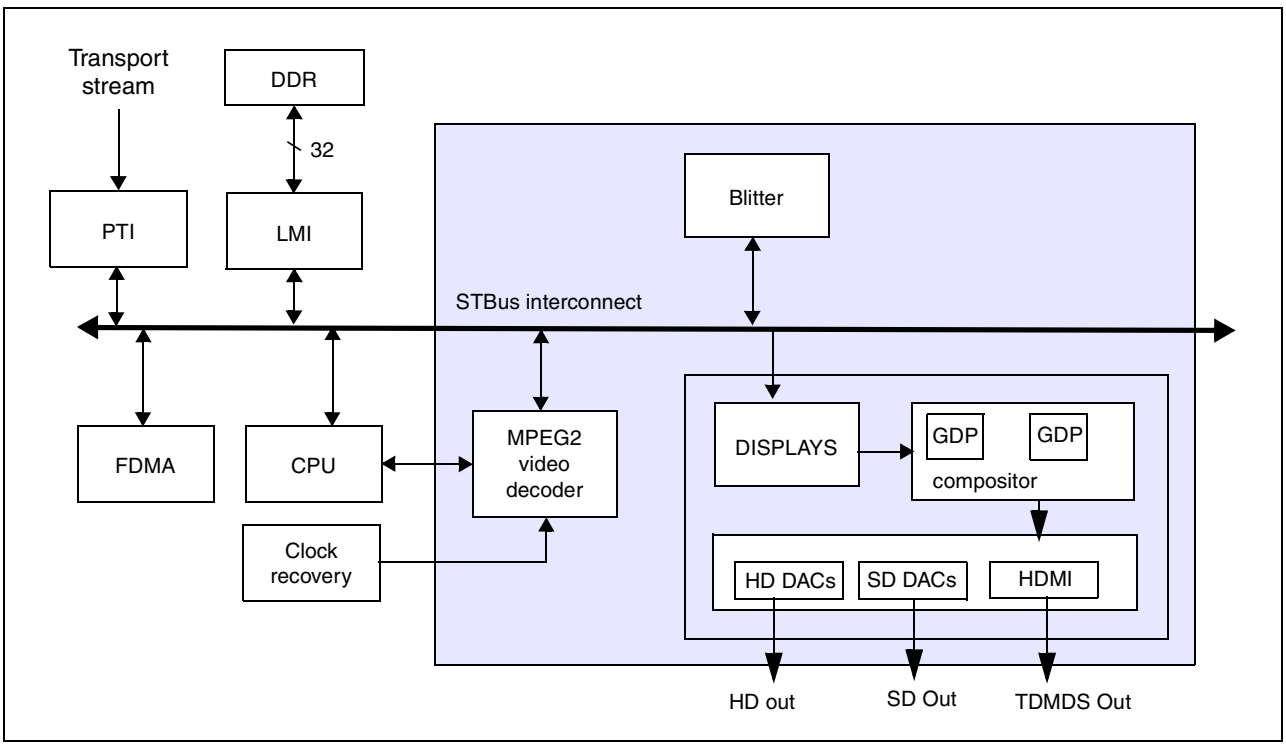
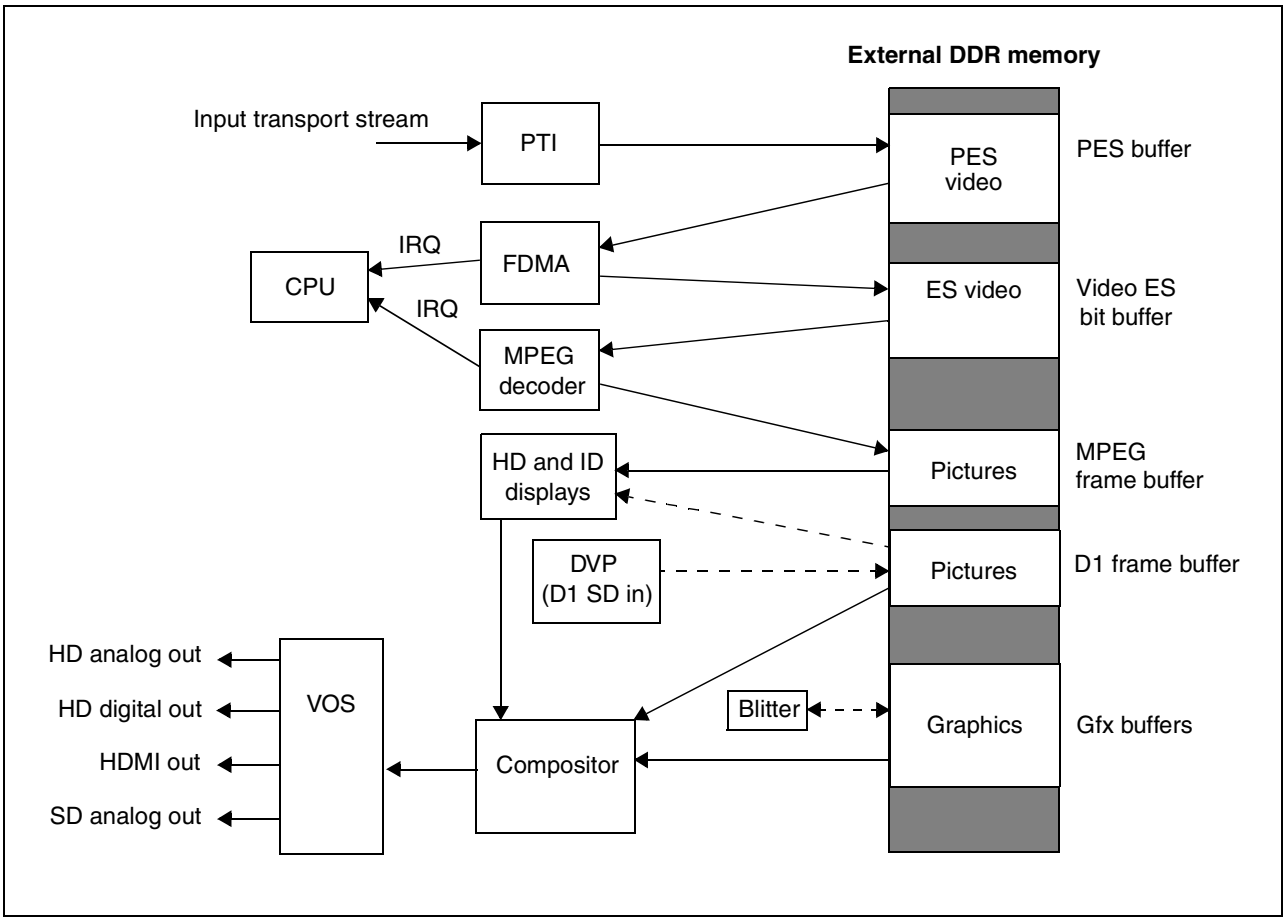


Figure 100 shows an example of dataflow around the MPEG video decoder at the system level.

Figure 100: 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 MPEG video decoder, decoded and reconstructed into frame buffers.
- Frame buffers are accessed by display(s).

54.3 Buffer organization

There are two types of buffer in the MPEG decoder:

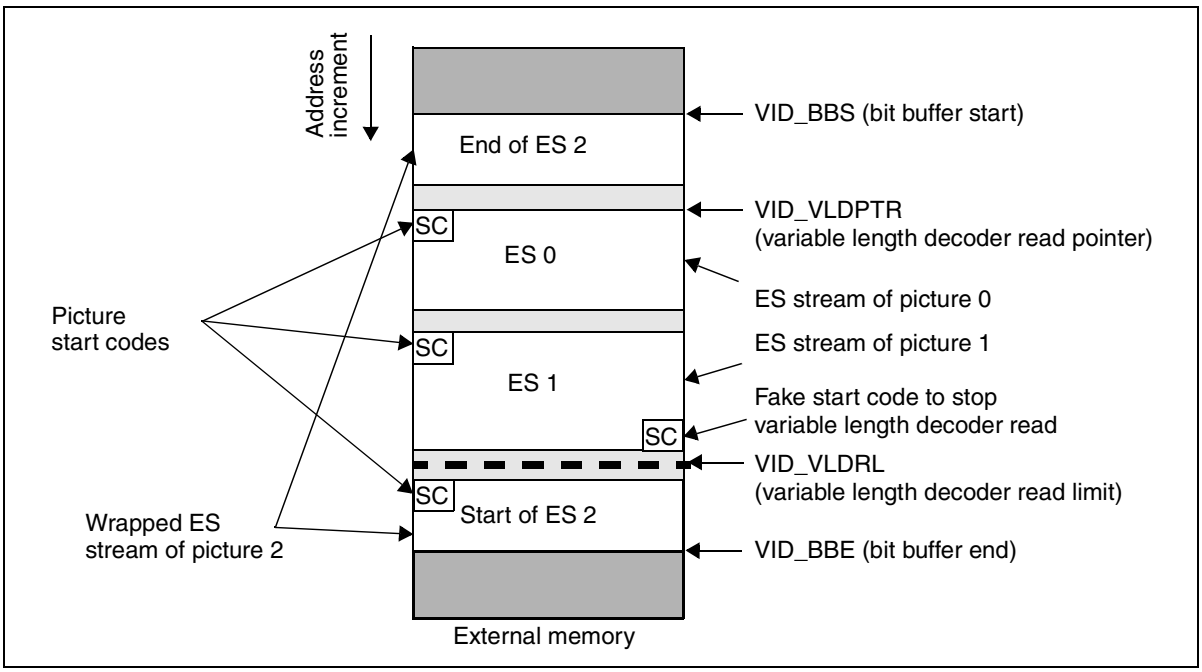
- **bit buffer**: area in external memory where the bitstream is stored,
- **frame buffer**: area in external memory in which decoded pictures are stored.

54.3.1 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 101).

Figure 101: Bit buffer organization



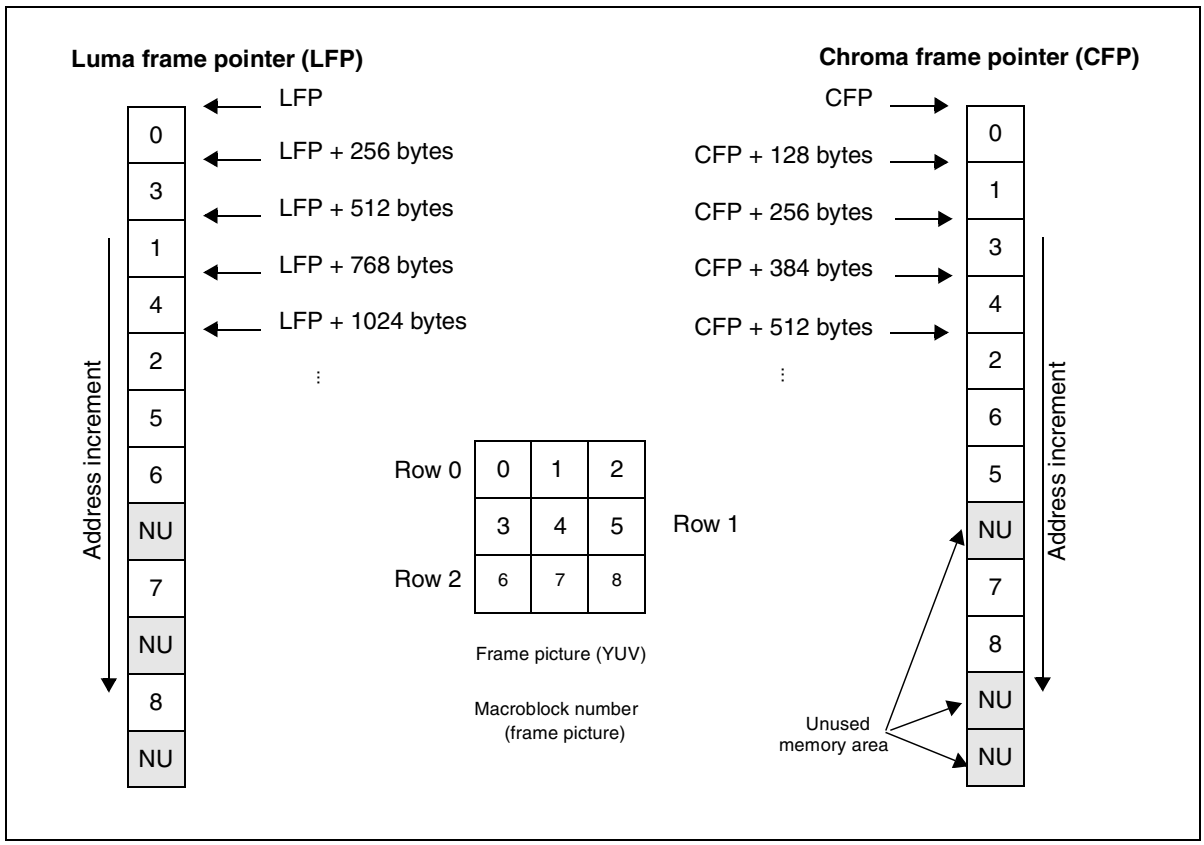
When a decode is launched, the variable length decoder reads data until it reaches a start code different from a slice start code. It can be the picture start code of the subsequent picture, or a fake start code added at the end of the picture to stop the variable length decoder. The variable length decoder read process can also be stopped with VID_VLDRL (see *Stream pointers for variable length decoder read access* *on page 459*).

54.3.2 Frame buffer organization

The video decoder stores both frame and field pictures in frame buffers, the size of which are dependant on picture size/resolution and decimation factors (see). 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 102 shows an example of a frame buffer with an odd width and height (3 x 3 macroblocks).

Figure 102: Frame buffer organization, mapping of 3 x 3 macroblocks



Memory is wasted with an odd number of vertical or horizontal macroblocks, 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* x 2 + 1) / 2 x 256
- *ChromaBufferSize* = (*MBwidth* x 2 + 1) / 2 x (*MBheight* x 2 + 1) / 2 x 128

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 103.

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 104.

Figure 103: Y macroblock storage

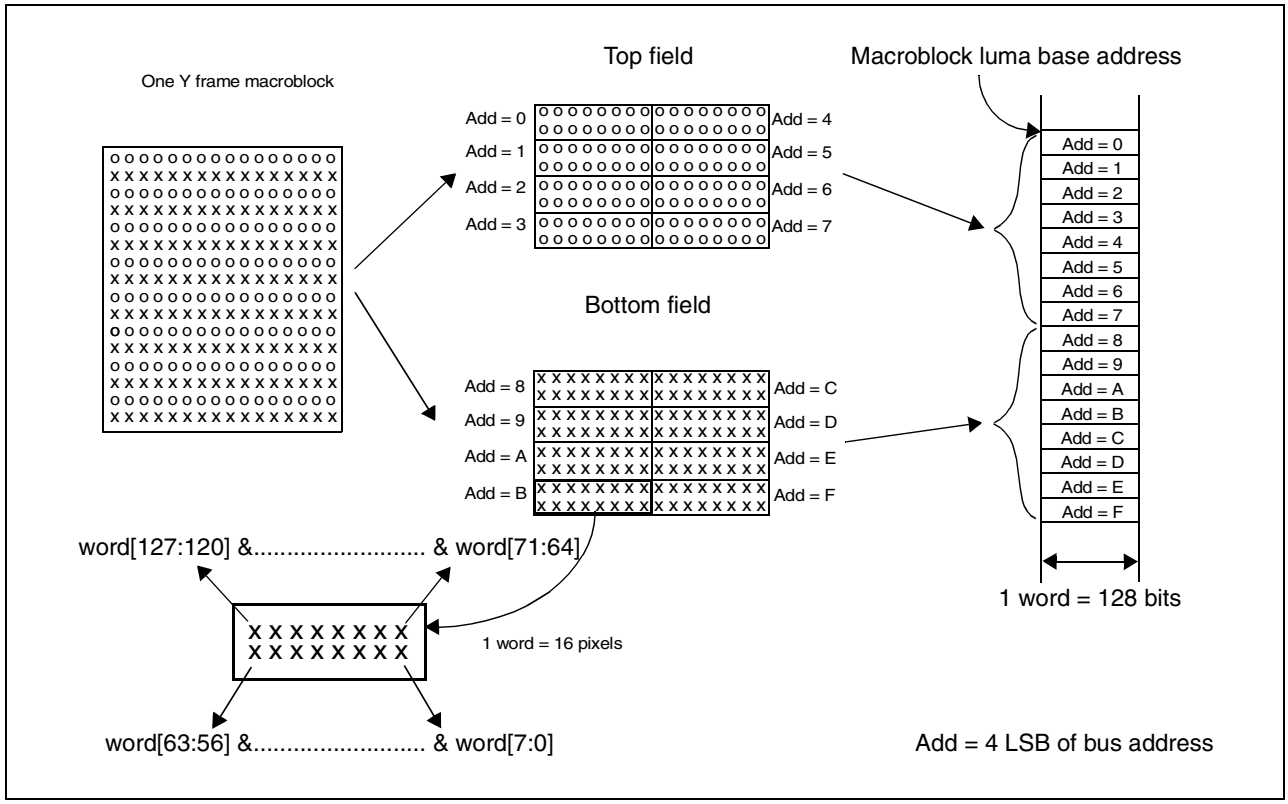
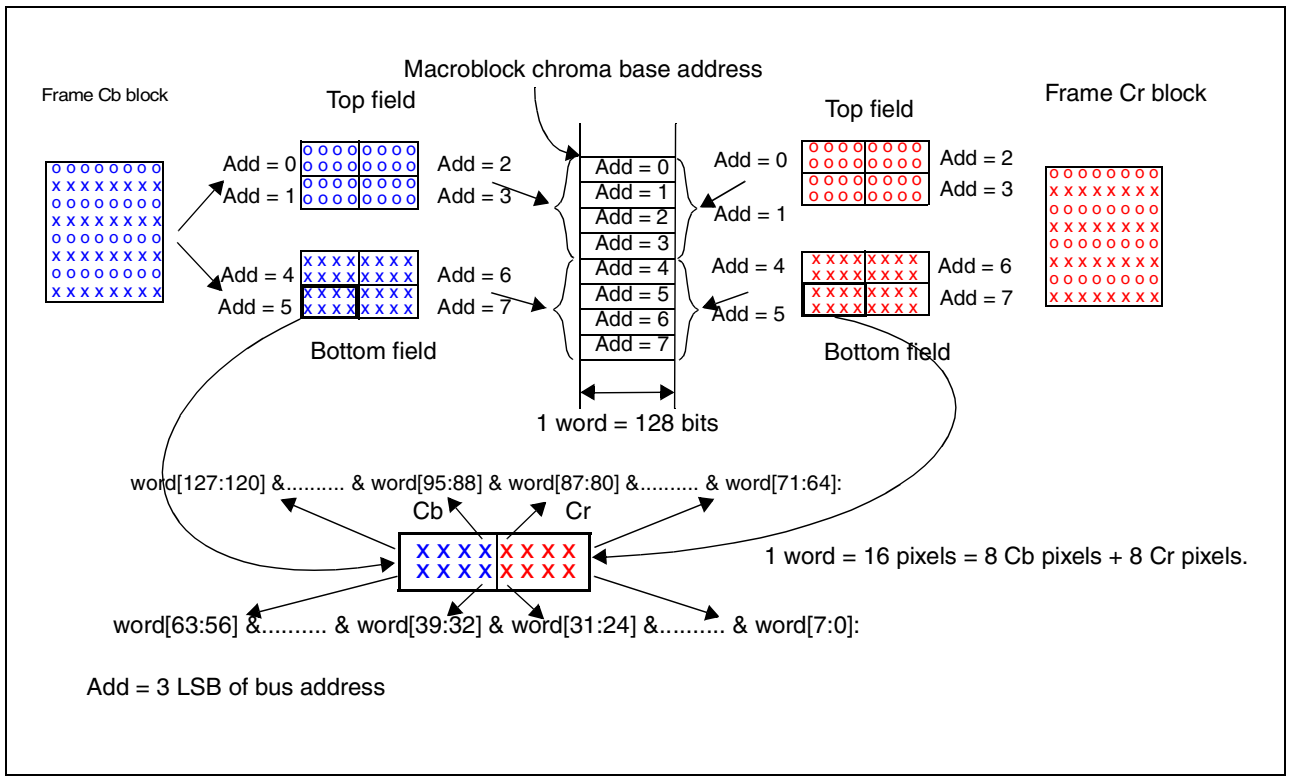


Figure 104: Cr/Cb macroblock storage



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*

MB col = 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*

54.3.3 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 54.3.2 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.

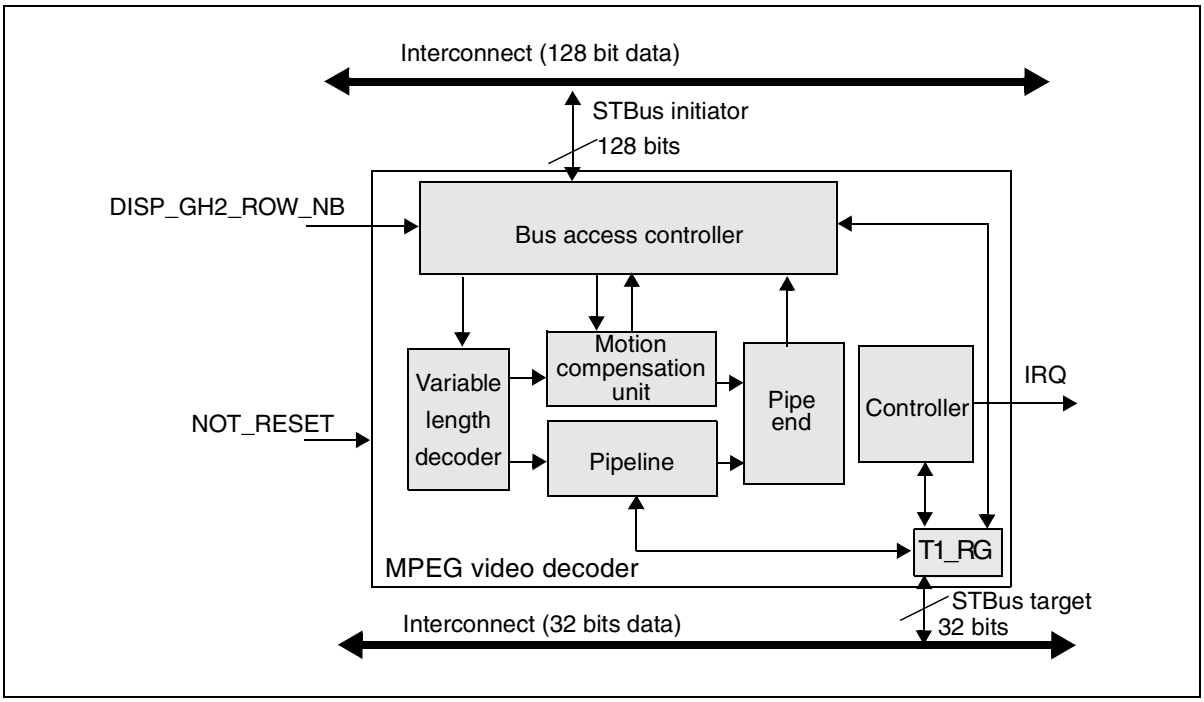
Bit buffer

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.

54.4 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 105: MPEG video decoder block diagram



54.4.1 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.

54.4.2 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 54.5.5: *Error recovery on page 461*),
- provides the number and the location of corrupted macroblocks in the picture (see Section 54.5.6: *Error statistics on page 464*).

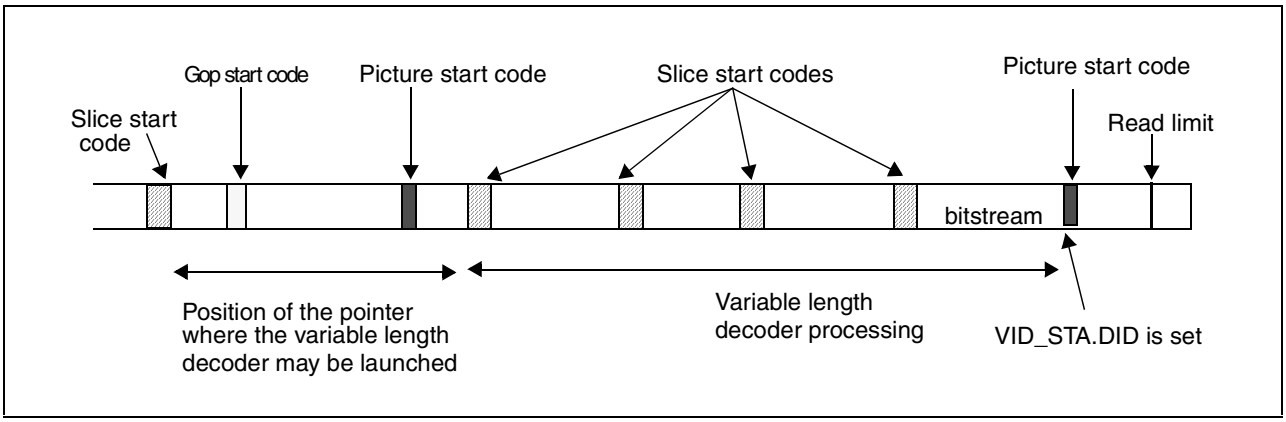
The variable length decoder is launched by the VID_EXE register for each picture to be decoded:

When the variable length decoder is launched with a pointer defined in byte units, 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 106).

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 106: 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).

54.4.3 MPEG pipeline

The variable length decoder sends run length coefficients to the MPEG pipeline. The pipeline processes the six blocks of every macroblock in the following order: Y0, Y1, Y2, Y3, Cb and Cr (refer to MPEG 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).

54.4.4 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.

54.4.5 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.

54.4.6 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 the register CFG_VIDIC. If R_OPC flags an error during a transaction, the CPU is interrupted with bit VID_STA.ROPC.

54.5 Video decoding

54.5.1 Control of decoding

The control is performed by the CPU. The CPU launches the decoder one picture at a time.

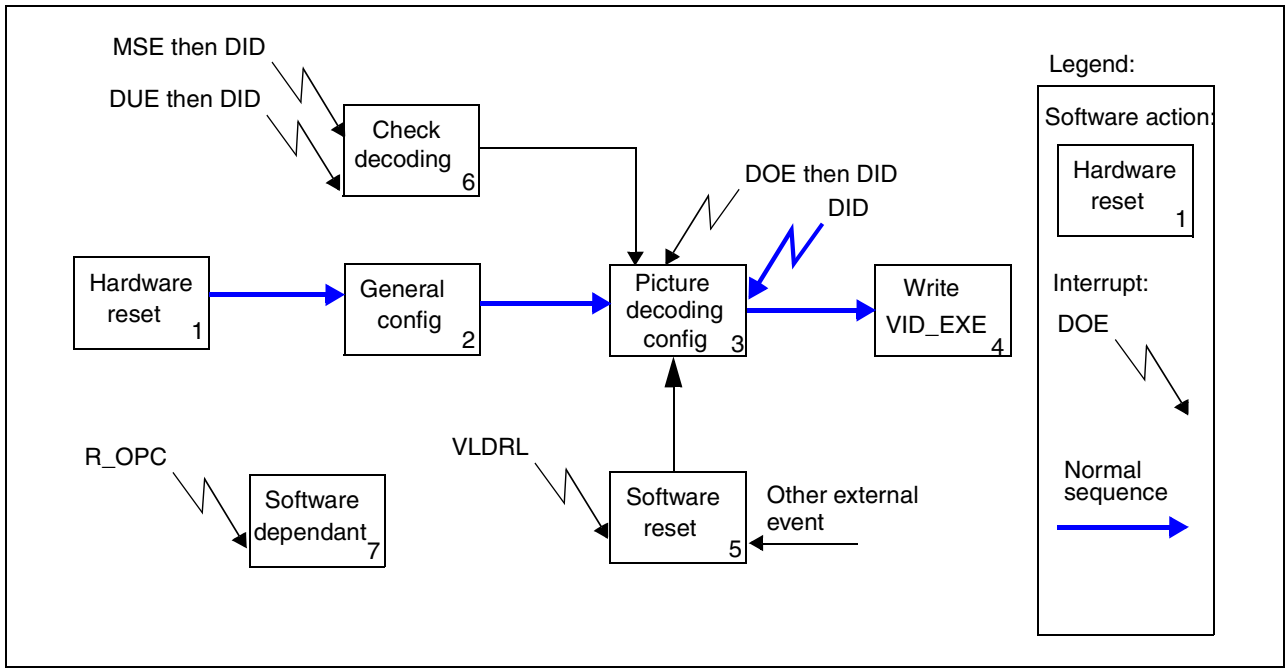
Decoding performed without error

The normal sequence is described below.

1. hardware reset,
2. general configuration of the MPEG video decoder (see *General configuration* *on page 458*),
3. picture decoding configuration (see *Picture decoding configuration* *on page 458*),
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 55: *MPEG video decoder registers on page 466* for more details on interrupts).

Figure 107: 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.

54.5.2 General configuration

The general configuration consists of CFG_VIDIC (video decoder interconnect configuration) and VID_ITM (interrupt mask).

54.5.3 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 MPEG 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 the 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.

54.5.4 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 MPEG 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 usefull when a large zoomout 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 109 and vertical decimation is performed as described in Figure 108. 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 & 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.

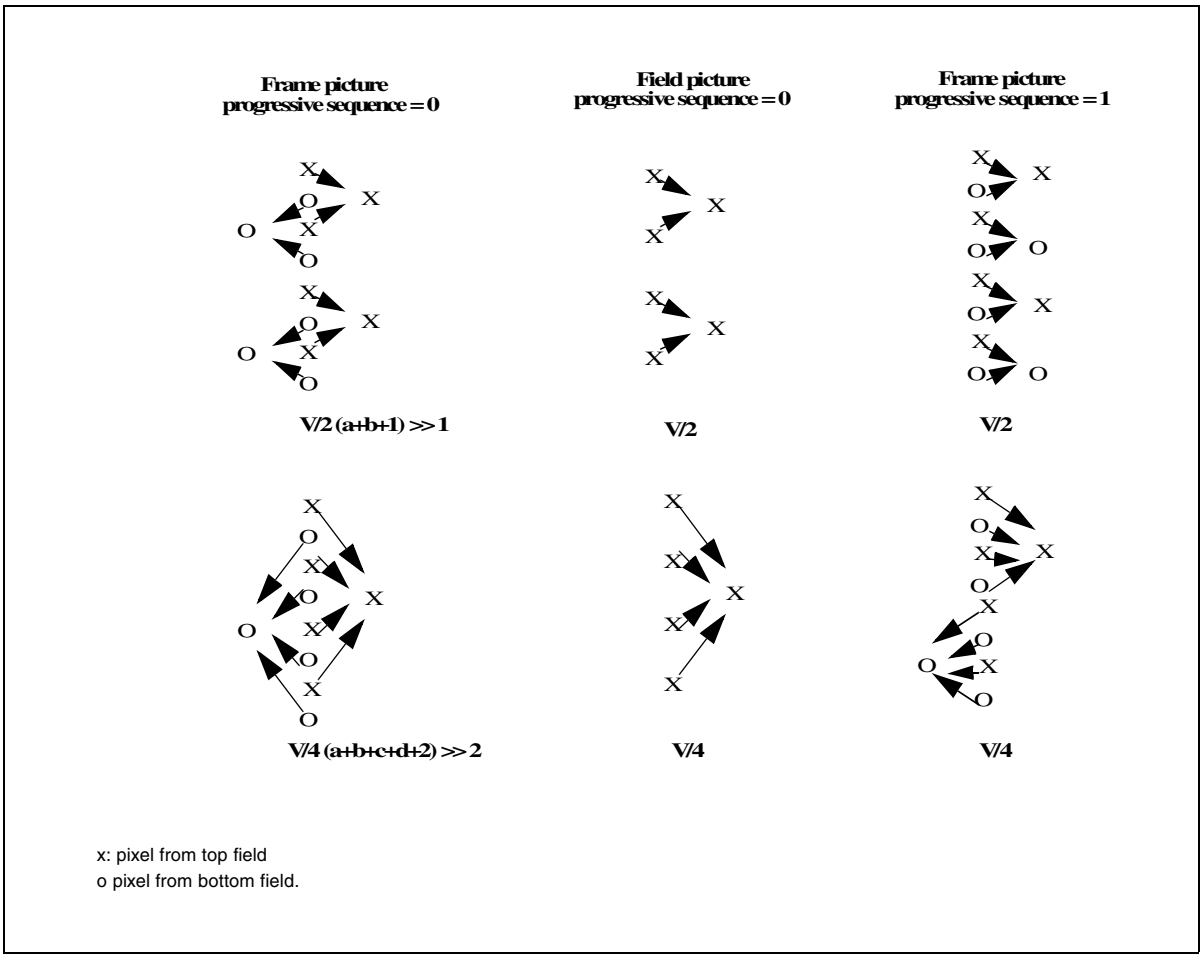


Figure 108: Vertical decimation in secondary reconstruction

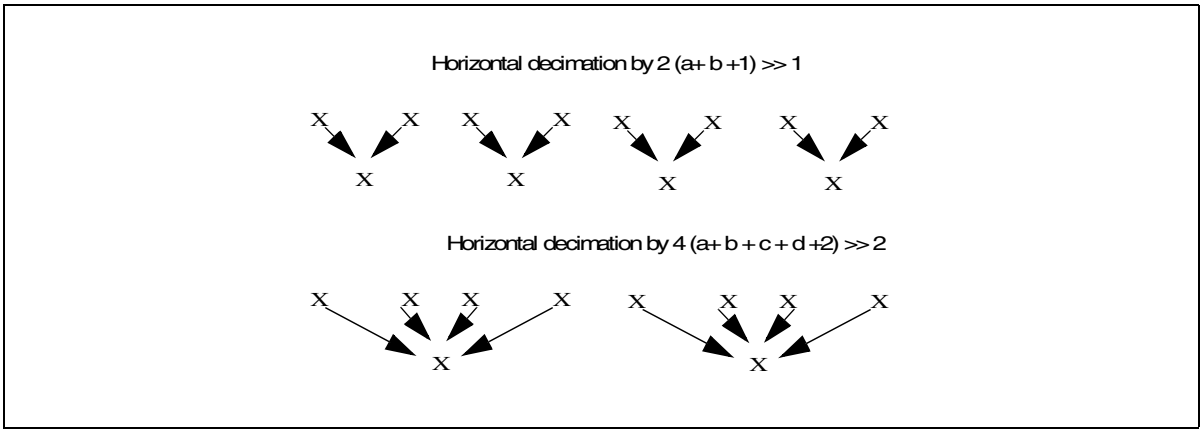


Figure 109: Horizontal decimation in secondary reconstruction

54.5.5 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 110 describes errors that are detected by the MPEG video decoder. Error concealment is described below the table.

Table 110: 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 462*	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 462*	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		

Table 110: Error detection

Layer	Description	Concealment	MBE[a]
Semantic error: The way of encoding slices and macroblocks does not follow MPEG rules.			
Slice	Slice start code combined with macroblock address increment gives a macroblock number already decoded.	See *Procedure one* on page 462	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 463	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 462	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 111 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 111: 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).

54.5.6 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.

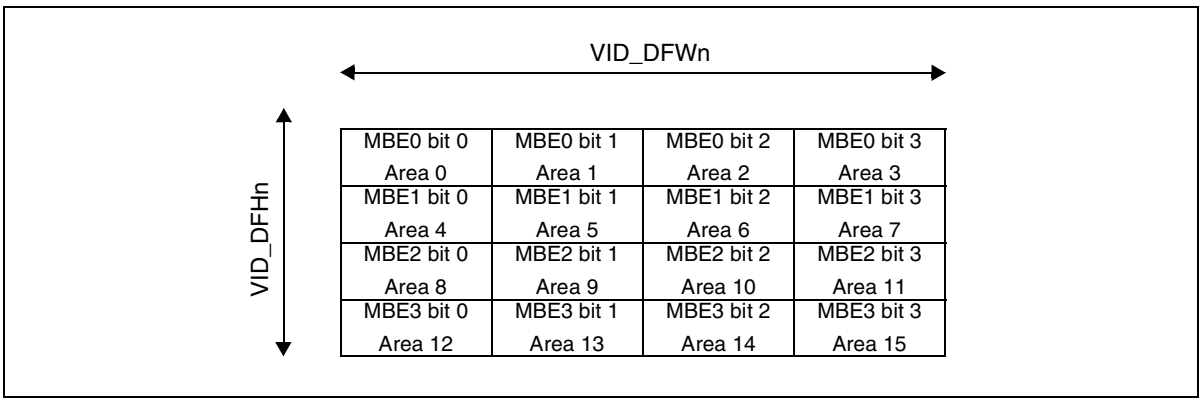
The decoded picture is divided into 16 areas, defined by four horizontal and four vertical slides. The areas are numbered from 0 (top left) to 15 (bottom right). This number is incremented from left to right, top to bottom (see Figure 110). For each area, VID_MBE*n* gives the number of macroblocks reconstructed 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 110: Macroblock error statistics



54.5.7 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).

54.6 Resets

54.6.1 Hardware reset

After a hardware reset, the MPEG video decoder is in an idle state. No processing is done and no request is sent until video decoding is launched.

54.6.2 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 MPEG 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.

55 MPEG video decoder registers

Addresses are provided as *VideoBaseAddress* + offset.

The *VideoBaseAddress* is:

0x3800 2000.

All registers are reset by HW (hardware reset). Read only registers can also be affected by SRS and EXE, see Section 54.6.2: *Software reset on page 465* for details.

When registers are read, the value of the reserved bits are 0 unless specified. No registers are buffered: write action effect is immediate.

Table 112: MPEG video decoder register summary

Register	Description	Offset	Type
VID_EXE	Execute decoding task	0x0008	WO
CFG_VIDIC	Video decoder interconnect configuration	0x0010	R/W
VID_MBEn	Macroblock error statistic (n = 0 to 3)	0x0070, 0x0074, 0x0078, 0x007C	RO
VID_QMWIp	Quantization matrix data, intra-table	0x0100 to 0x013F	R/W
VID_QMWNIp	Quantization matrix data, non–intra-table	0x0180 to 0x01BF	R/W
VID_TIS	Task instruction	0x0300	R/W
VID_PPR	Picture parameters	0x0304	R/W
VID_SRS	Decoding soft reset	0x030C	WO
VID_IT	Interrupt mask	0x03F0	R/W
VID_ITS	Interrupt status	0x03F4	RO
VID_STA	Status	0x03F8	RO
VID_DFH	Decoded frame height	0x0400	R/W
VID_DFS	Decoded frame size	0x0404	R/W
VID_DFW	Decode frame width	0x0408	R/W
VID_BBS	Video elementary stream bit buffer start	0x040C	R/W
VID_BBE	Video elementary stream bit buffer end	0x0414	R/W
VID_VLDRL	Variable length decoder read limit	0x0448	R/W
VID_VLDPTR	Variable length decoder read pointer	0x045C	R/W
VID_MBNM	Macroblock number for the main reconstruction (debug)	0x0480	
VID_MBNS	Macroblock number for secondary reconstruction	0x0484	RO
VID_BCHP	Backward chroma frame buffer	0x0488	R/W
VID_BFP	Backward luma frame pointer	0x048C	R/W
VID_FCHP	Forward chroma frame buffer	0x0490	R/W
VID_FFP	Forward luma frame pointer	0x0494	R/W

Table 112: MPEG video decoder register summary

Register	Description	Offset	Type
VID_RCHP	Main reconstructed chroma frame pointer	0x0498	R/W
VID_RFP	Main reconstructed luma frame pointer	0x049C	R/W
VID_SRCHP	Secondary reconstructed chroma frame buffer	0x04A0	R/W
VID_SRFP	Secondary reconstructed luma frame buffer	0x04A4	R/W
VID_RCM	Reconstruction mode	0x04AC	R/W

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: *VideoBaseAddress* + 0x0008
Type: WO
Reset: Undefined
Description: Writing to the least significant byte of this register starts a decoding task.

CFG_VIDIC Video decoder interconnect 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																									PRDC		PRDL		LP		

Address: *VideoBaseAddress* + 0x0010
Type: R/W
Reset: 0
Description: Description

Maximum packet size for processes on T2 plugs. For some processes, the message size may be smaller that the packet size. In this case, the packet size is equal to the message size.

[31:7] **Reserved**

[6:5] **LP:** maximum packet size for variable length decoder and reconstruction processes. When a packet size is lower than the message size, the packet size is the message size.
000: Message size (default = 16 for variable length decoder, 16 for Rec Y and 8 for Rec C)
010: 8 011: 4
100: 2 Others: Illegal

[4:3] **PRDL:** maximum packet luma prediction.
00: Message size (default = 15) 01: 5
10: 2 Others: Illegal

[2:0] **PRDC**: maximum packet size for chroma prediction.
00: Message size (default = 9) 01: 3
10: 2 Others: Illegal

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: *VideoBaseAddress* + 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: *VideoBaseAddress* + 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: *VideoBaseAddress* +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: *VideoBaseAddress* +0x0300

Type: R/W

Reset: 0

This register contains the decoding task instruction.

[31:4] **Reserved**

[3] **RMM**: reconstruct missing macroblock. If 1, in case of underflow, the missing macroblocks are reconstructed.

[2] **MVC:** motion vector check. When set, this bit ensures that motion vectors used for prediction remain inside the picture (Table 110: *Error detection on page 461*).

[1] **SBD:** simplified B picture decoding. When this bit is set, B picture decoding is simplified to save bandwidth: all bidirectional macro blocks are processed as forward.

[0] **OVW:** enable overwrite mechanism during decoding if set.

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: *VideoBaseAddress* + 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. When this bit is set, the decoder expects an MPEG-2 video bitstream. If it is reset, then an MPEG-1 bitstream is expected.

[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 MPEG 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:** 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. It 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

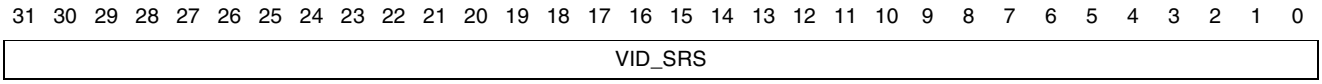
[15:12] **BFV[3:0]:** set equal to BACKWARD_VERTICAL_F_CODE of the picture coding extension.

[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.

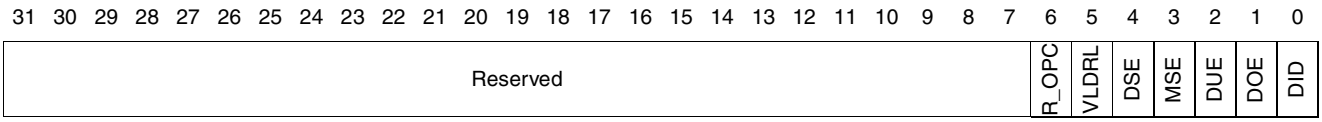
[3:0] **FFH[3:0]:** in MPEG-1 mode FFH[3] is set equal to FULL_PEL_FORWARD_VECTOR of the picture header, and FFH[2:0] is set equal to FORWARD_F_CODE of the picture header. In MPEG-2 mode FFH[3:0] is set equal to FORWARD_HORIZONTAL_F_CODE of the picture coding extension.

VID_SRS Decoding soft reset



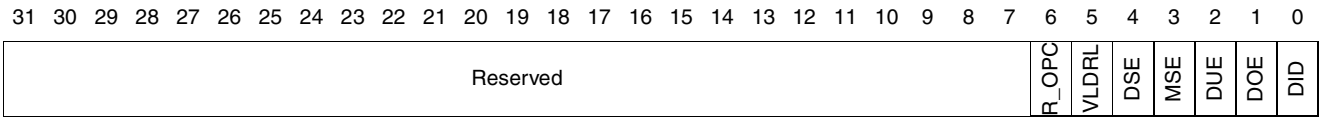
Address: *VideoBaseAddress* + 0x030C
Type: WO
Reset: Undefined
Description: Writing to the least significant byte of this register starts the software reset sequence. The reset is just activated once on writing.

VID_IT Interrupt mask



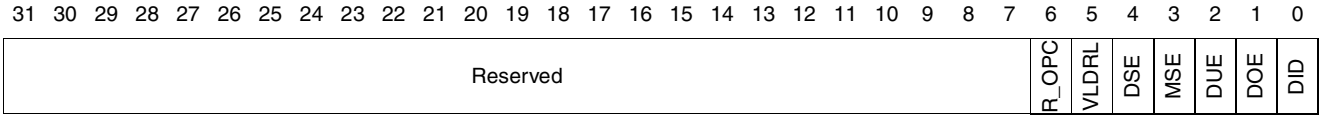
Address: *VideoBaseAddress* + 0x03F0
Type: R/W
Reset: 0
Description: Any bit set in this register enables the corresponding interrupt on the VID_IRQ line. An interrupt is generated when a bit in the VID_STA register changes from 0 to 1 and the corresponding mask bit is set.

VID_ITS Interrupt status



Address: *VideoBaseAddress* + 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 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



Address: *VideoBaseAddress* +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.

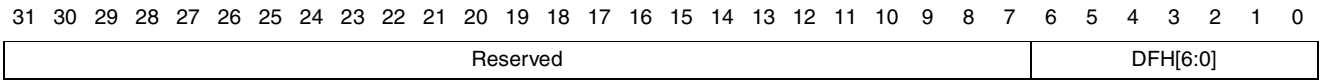
[3] **MSE:** set to 1 at the end of a decoding task if a semantic or a syntax error has been detected by the variable length decoder during the decoding process of the picture. It is reset on a write action on VID_EXE or VID_SRS LS byte.

[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



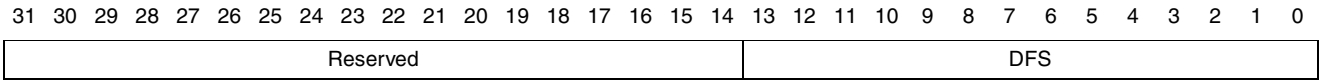
Address: *VideoBaseAddress* + 0x0400

Type: R/W

Reset: 0

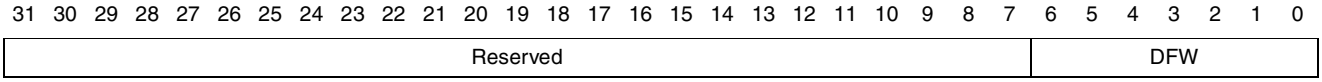
Description: Decoded frame height (in rows of macroblocks). This register is used for error concealment and macroblocks error statistics. This is derived from the vertical size value transmitted in the sequence header. It is divided internally by 2 for field picture decoding.

VID_DFS Decoded frame size



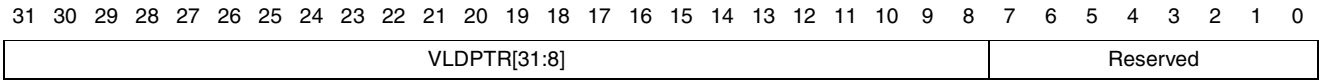
Address: *VideoBaseAddress* + 0x0404
Type: R/W
Reset: 0
Description: This register is set up with a value equal to the number of macroblocks in the decoded picture. This is derived from the horizontal size and vertical size values transmitted in the sequence header. It is divided internally by 2 for field picture decoding.

VID_DFW Decode frame width



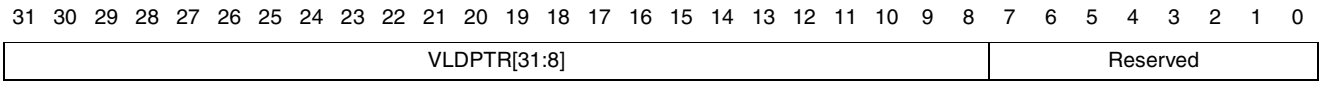
Address: *VideoBaseAddress* + 0x0408
Type: R/W
Reset: 0
Description: This register is set up with a value equal to the width in macroblocks of the decoded picture.This is derived from the horizontal size value transmitted in the sequence header.

VID_BBS Video elementary stream bit buffer start



Address: *VideoBaseAddress* + 0x040C
Type: R/W
Reset: 0
Description: Memory address of the video elementary stream bit buffer start, defined in units of 256 bytes.

VID_BBE Video elementary stream bit buffer end



Address: *VideoBaseAddress* + 0x0414
Type: R/W
Reset: 0
Memory address of the video elementary stream bit buffer end, defined in units of 256 bytes.

VID_VLDRL Variable length decoder read 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
VLDRL[31:7]																								Reserved							

Address: *VideoBaseAddress* + 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 units of 256 bytes.

VID_VLDPTR Variable length decoder read 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
VLDPTR[31:7]																															

Address: *VideoBaseAddress* + 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.

It is taken into account as soon as VID_EXE is accessed. Then it holds the current variable length decoder read pointer address given in 128 bytes unit (7 LSBs are 0).

VID_MBNM Macroblock number for the main reconstruction (debug)

31	30	29	28	27	26	25	24	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									COL							Reserved									ROW							

Address: *VideoBaseAddress* + 0x0480

Type: RO

Reset: 0

Description: This register holds the position of the macroblock to be reconstructed in the frame buffer for the main reconstruction. When no picture is being decoded, it gives the position of the macroblock following the last decoded macroblock for the previous decoded picture.

The value is stable between the end of a decoding task and the next EXE. The first macroblock (macroblock 0) starts at row 0 and column 0.

VID_MBNS Macroblock number for secondary reconstruction

31	30	29	28	27	26	25	24	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									COL							Reserved									ROW							

Address: *VideoBaseAddress* + 0x0484

Type: RO

Reset: 0

Description: This register holds the position of the macroblock to be reconstructed in the frame buffer for secondary reconstruction. When no picture is being decoded, it gives the position of the macroblock following the last decoded macroblock for the previous decoded picture. The value is stable between the end of a decoding task and the next programming of EXE.

The first macroblock (macroblock 0) starts at row 0 and column 0. The column is incremented by 1 every two horizontal decimated macroblocks if horizontal decimation is 2, and every four decimated macroblocks if horizontal decimation is 4. The row is incremented by 1 at the end of each row of decimated MBs.

VID_BCHP Backward 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
BCHP[31:9]																							Reserved								

Address: *VideoBaseAddress* + 0x0488

Type: R/W

Reset: 0

Description: This register holds the start address of the backward frame chroma prediction buffer, defined in units of 512 bytes.

VID_BFP Backward luma frame 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
BFP[31:9]																							Reserved								

Address: *VideoBaseAddress* + 0x048C

Type: R/W

Reset: 0

Description: This register holds the start address of the luma backward prediction frame picture buffer, defined in units of 512 bytes.

VID_FCHP Forward 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
FCHP[31:9]																							Reserved								

Address: *VideoBaseAddress* + 0x0490

Type: R/W

Reset: 0

Description: This register holds the start address of the chroma forward prediction frame picture buffer, defined in units of 512 bytes.

VID_FFP Forward luma frame 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
FFP[31:9]																							Reserved								

Address: *VideoBaseAddress* + 0x0494
Type: R/W
Reset: 0
Description: This register holds the start address of the luma forward prediction frame picture buffer, defined in units of 512 bytes.

VID_RCHP Main reconstructed chroma frame 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
RCHP[31:9]																							Reserved								

Address: *VideoBaseAddress* + 0x0498
Type: R/W
Reset: 0
Description: This register holds the start address of the reconstructed chroma frame picture buffer, defined in units of 512 bytes.

VID_RFP Main reconstructed luma frame 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
RFP[31:9]																							Reserved								

Address: *VideoBaseAddress* + 0x049C
Type: R/W
Reset: 0
Description: This register holds the start address of the reconstructed (decoded) luma picture buffer, defined in units of 512 bytes.

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																							Reserved								

Address: *VideoBaseAddress* + 0x04AC
Type: R/W
Reset: 4
Description: This register holds the start address of the secondary reconstructed chroma frame buffer, defined in units of 512 bytes.

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																									Reserved						

Address: *VideoBaseAddress* + 0x04A4

Type: R/W

Reset: 0

Description: This register holds the start address of the secondary reconstructed luma frame buffer, defined in units of 512 bytes.

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: *VideoBaseAddress* + 0x04AC

Type: R/W

Reset: 4

Description: This register holds the 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.

56 2D graphics processor (blitter)

56.1 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 up to 100 MHz clock rate. For each operation involving source 2, the maximum output rate is 100 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 113: Performance for typical operations, over a range of target pixel sizes (Mpix/s, at output)

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 56.6 on page 502*.

56.2 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 56.7 on page 503.

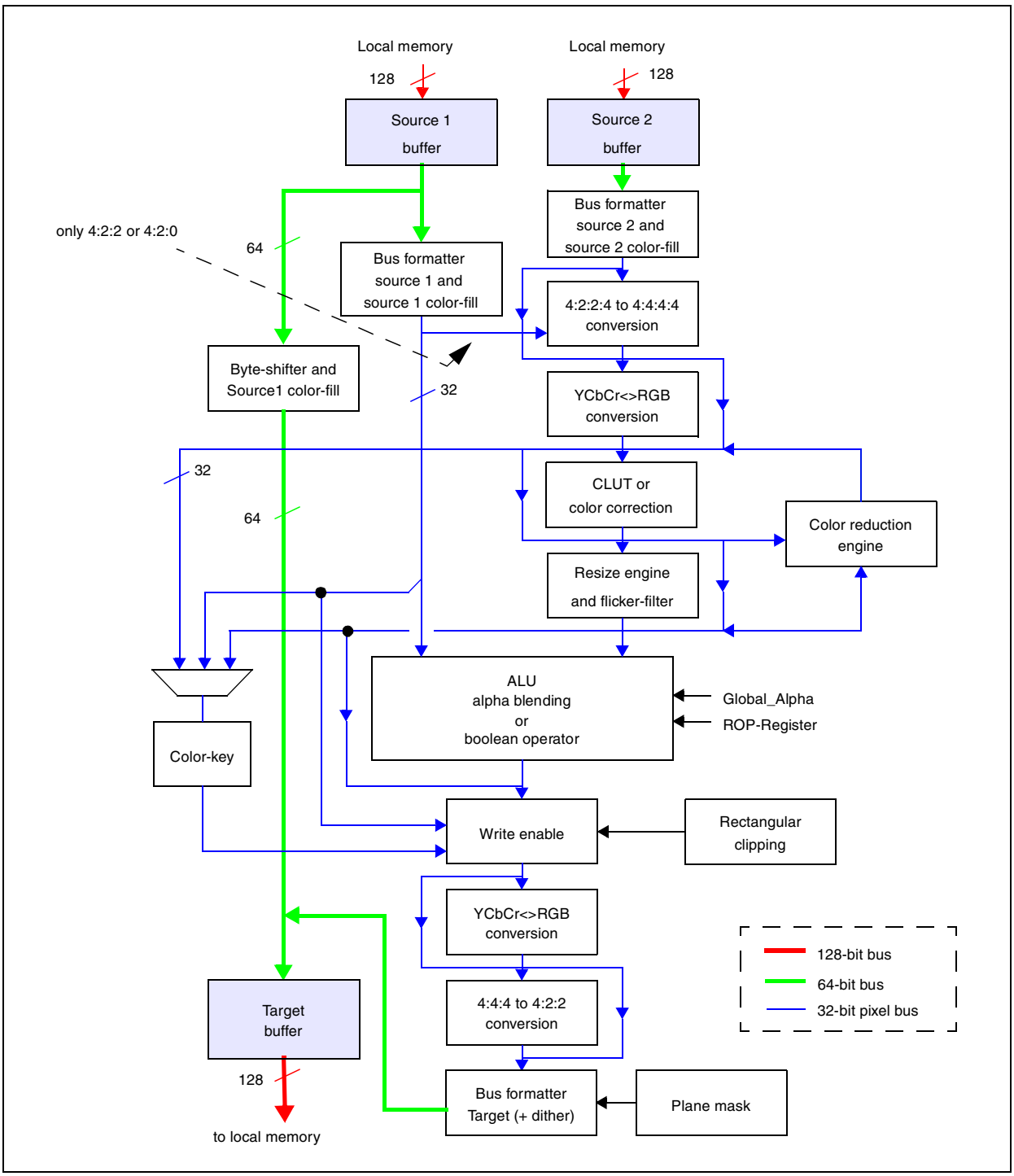
Source and destination windows are all defined using an XY descriptor, with pixel accuracy from 1 bpp to 32 bpp, whatever the format.

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.

56.3 Functional block diagram

The following figure illustrates the blitter functions that can be performed. Each of the blitter functions is briefly described in Section 56.5 on page 483.

Figure 111: 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.

56.3.1 The blitter process

The 2D graphics processor has control and status registers allowing global blitter operation management:

- BLT_CTL 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_CTL) and start the blitter by setting bit START (BLT_CTL), 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.

56.3.2 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.

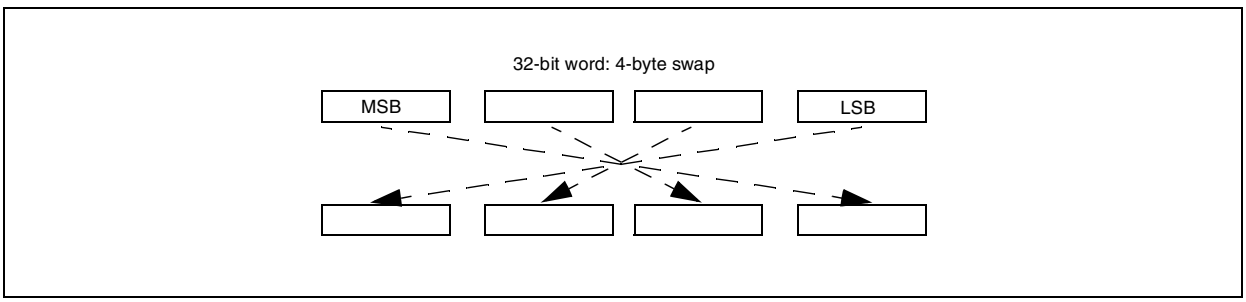
Note: As a result of the number of input formats, output formats, available operators and different configurations, many register combinations have no meaning in terms of the graphics blitter. Registers must be programmed with this in mind.

56.4 Endianness support

The register bit (BLT_PKZ[5] 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.

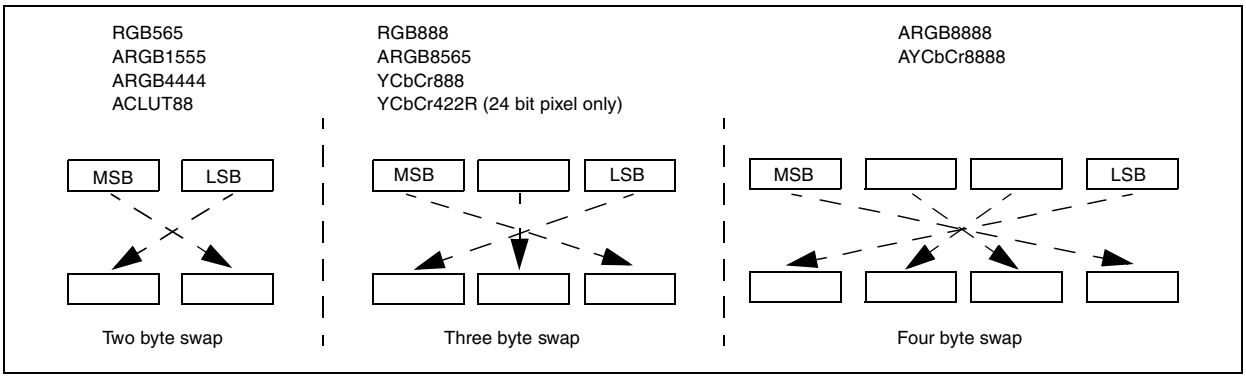
Figure 112: Endianness conversion for nodes, filter coefficients, CLUT, XYL parameters



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 113: 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.

56.5 Blitter functions

56.5.1 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 114A: 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 114B: 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 115: 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 116: 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.

56.5.2 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 (BLT_TSZ) specifies the size of the source 1 window (equals the target size).

56.5.3 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.

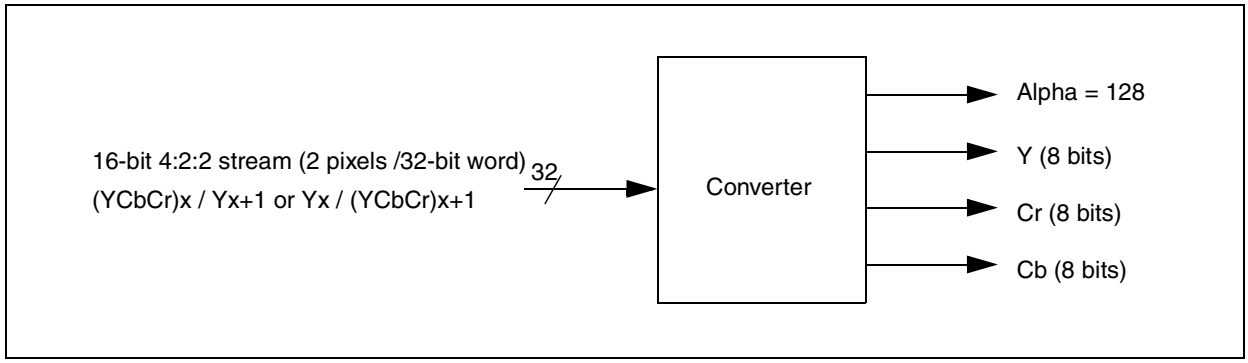
The number of significant bits varies from 1 bpp to 32 bpp according to the bit-depth of the color format.

All color formats are supported, except YCbCr4:2:0MB and YCbCr4:2:2MB.

56.5.4 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 114: 4:2:2 to 4:4:4:4 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 117: 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 118: 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 119: 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 120: 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

56.5.5 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 121: 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 $(Cb_1+2Cb_1+Cb_2) / 4$ $(Cr_1+2Cr_1+Cr_2) / 4$	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 122: 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 123: 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 124: 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 $(Cb_{n-1}+2Cb_n+\mathbf{Cb_n}) / 4$ $(Cr_{n-1}+2Cr_n+\mathbf{Cr_n}) / 4$

56.5.6 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:

$0 \le R, G, B \le 255$

$16 \le Y \le 235$

$16 \le Cb, Cr \le 240$ in offset binary representation, $-112 \le Cb, Cr \le +112$ 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 125: 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 126: 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 127: 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 128: 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:

$0 \le R, G, B \le 255$
$0 \le Y \le 255$
$0 \le Cb, Cr \le 255$ in offset binary representation, $-128 \le Cb, Cr \le +127$ 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 129: 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 130: 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 131: 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 132: 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)

56.5.7 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 133: 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 134: 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 135: 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 136: 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 137: 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 138: 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 139: 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 140: 709 colorimetry / integer matrix / digital range

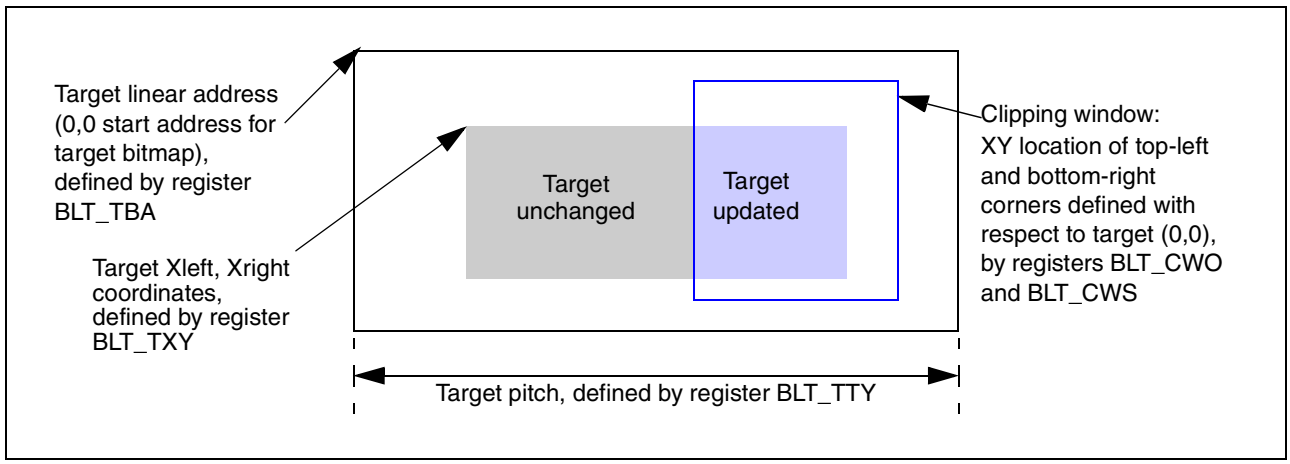
RGB to YCbCr integer matrix as implemented (1.0 <> 1024)												
Y	=		218	xR	+	732	xG	+	74	xB	+	
Cb	=	-	120	xR	-	404	xG	+	524	xB	+	128
Cr	=		524	xR	-	476	xG	-	48	xB	+	128

56.5.8 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 115: 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.

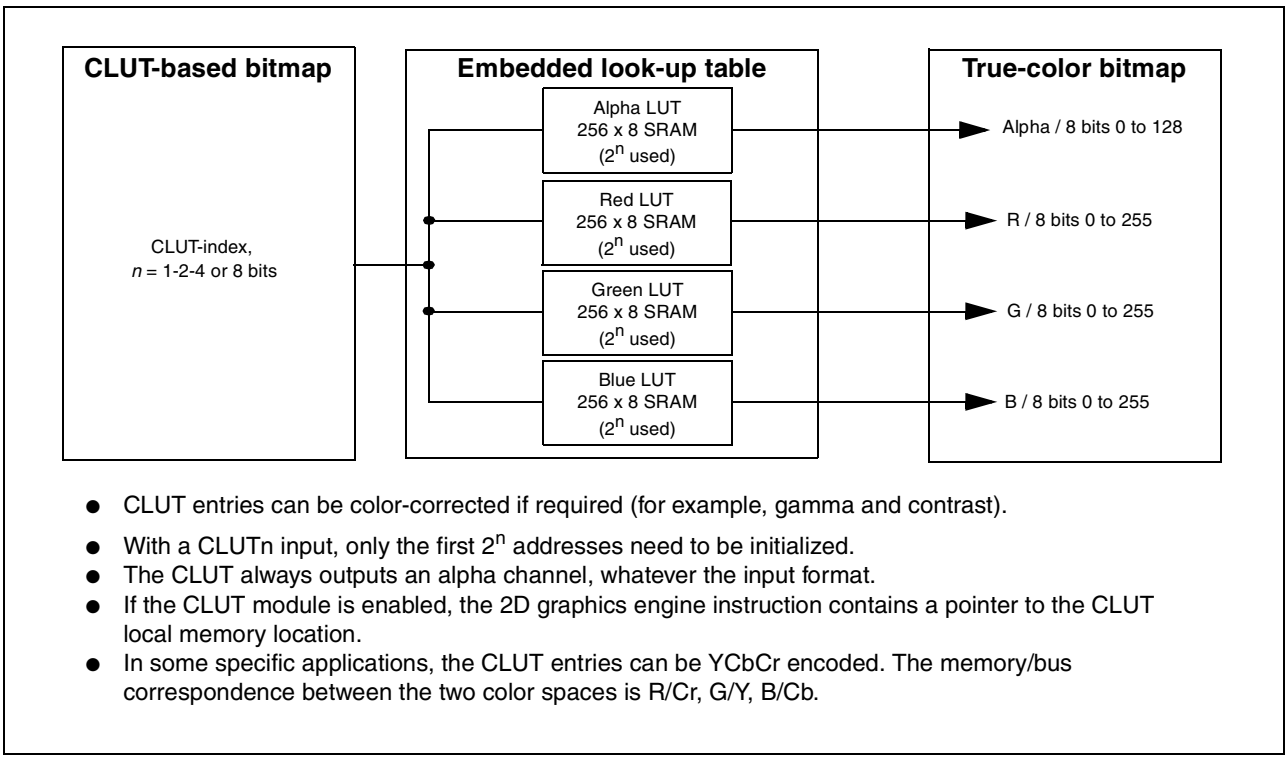
56.5.9 CLUT and color correction on source 2

The internal 256 x 32 LUT can perform color expansion, color reduction and color correction.

Color expansion

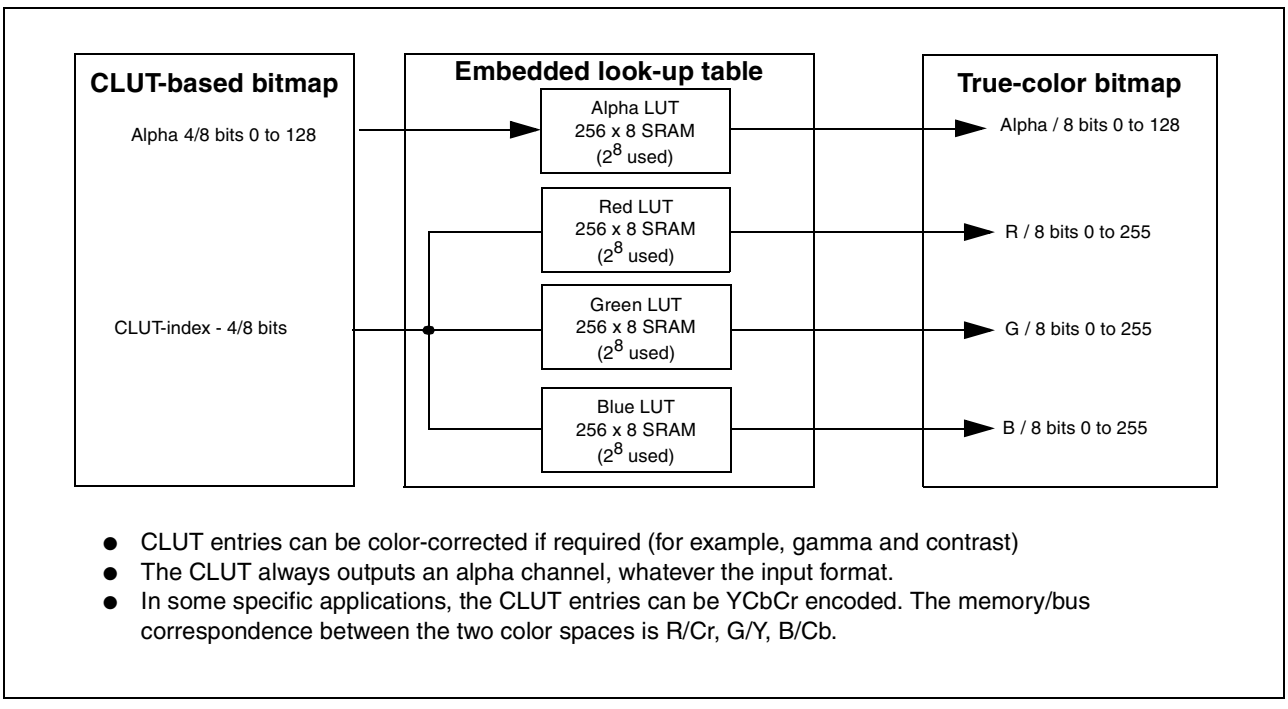
During color expansion a CLUT-based bitmap is transformed into a true-color bitmap, using the embedded look-up table (256x32 SRAM). The following two figures illustrate a CLUT-to-RGB conversion and a CLUT88/44-to-RGB conversion.

Figure 116: 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 117: ACLUT88/44 to ARGB conversion



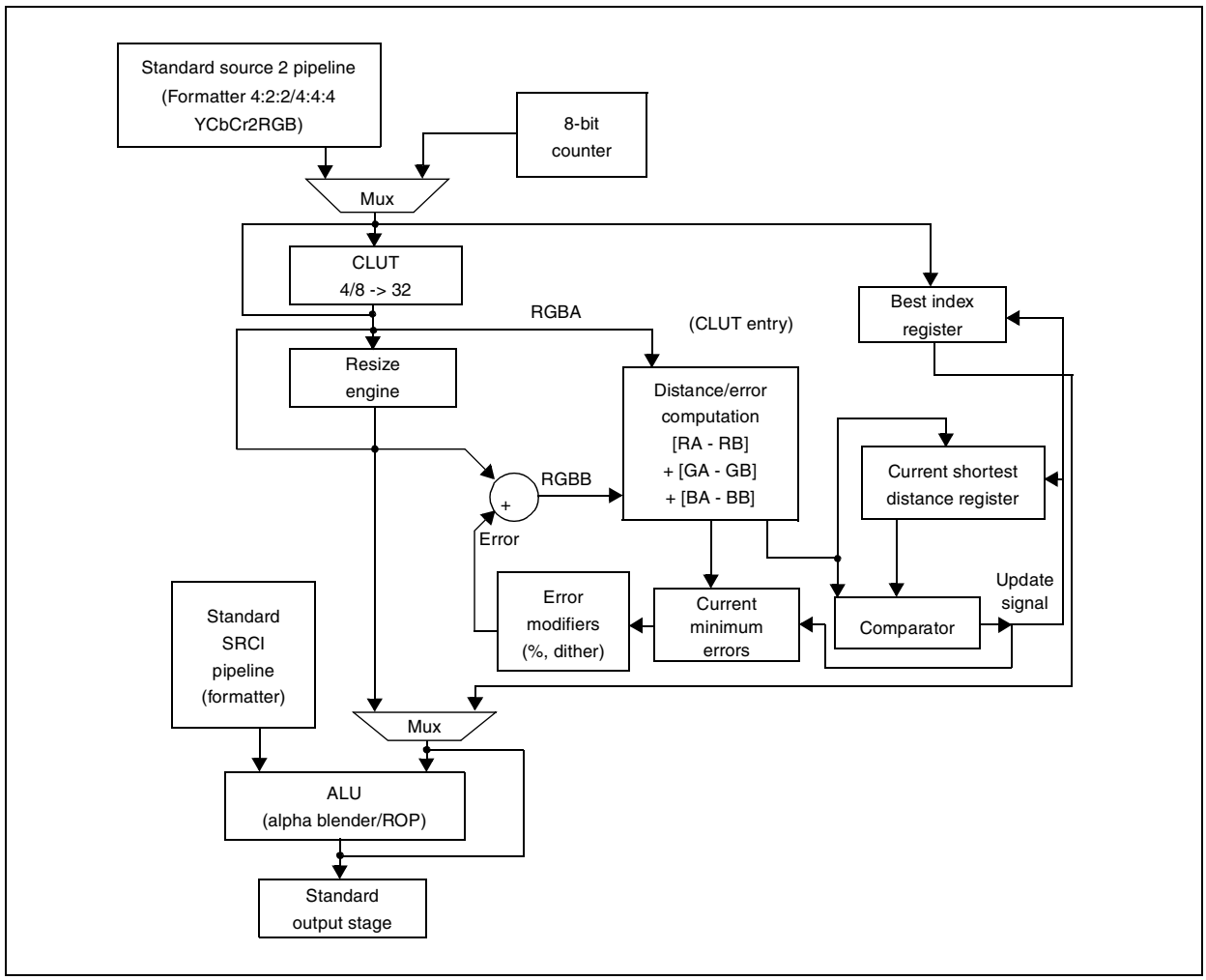
- CLUT entries can be color-corrected if required (for example, gamma and contrast)
- The CLUT always outputs an alpha channel, whatever the input format.
- In some specific applications, the CLUT entries can be YCbCr encoded. The memory/bus correspondence between the two color spaces is R/Cr, G/Y, B/Cb.

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 118: Color reduction engine architecture



The color reduction mechanism can be summarized as follows (see also Figure 118):

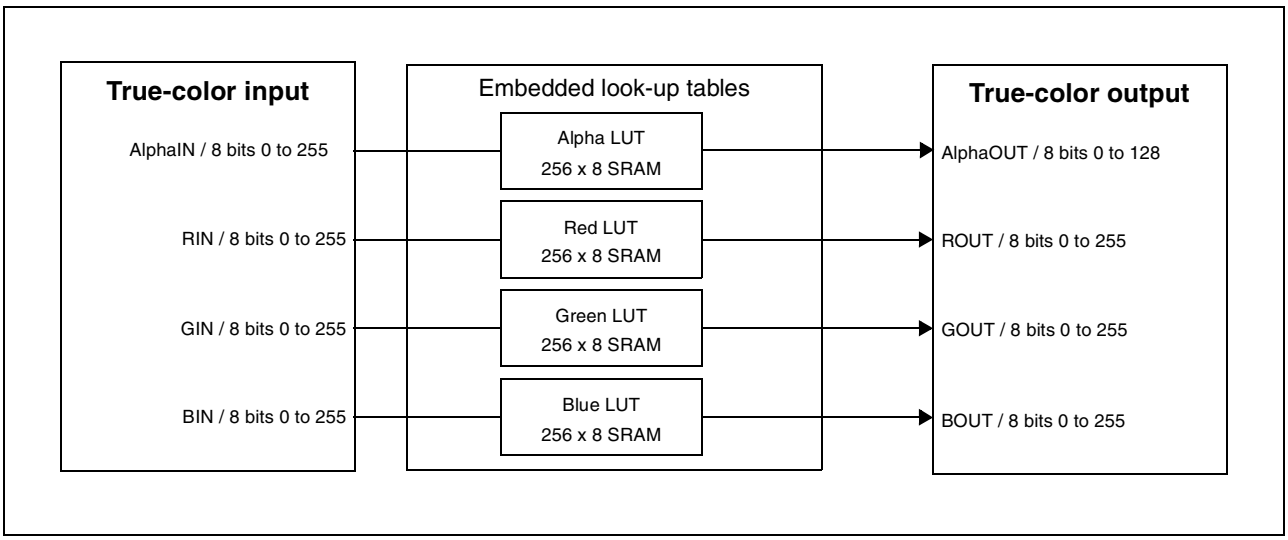
- Best match strategy: the whole CLUT is scanned for each new RGB pixel from source 2. A distance is computed for each CLUT entry and the current source 2 pixel. The entry with the shortest distance is considered as the best possible match, and its index is used to represent the RGB pixel in the target 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 (xLSB, yLSB). 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 119: Color correction for true-color inputs



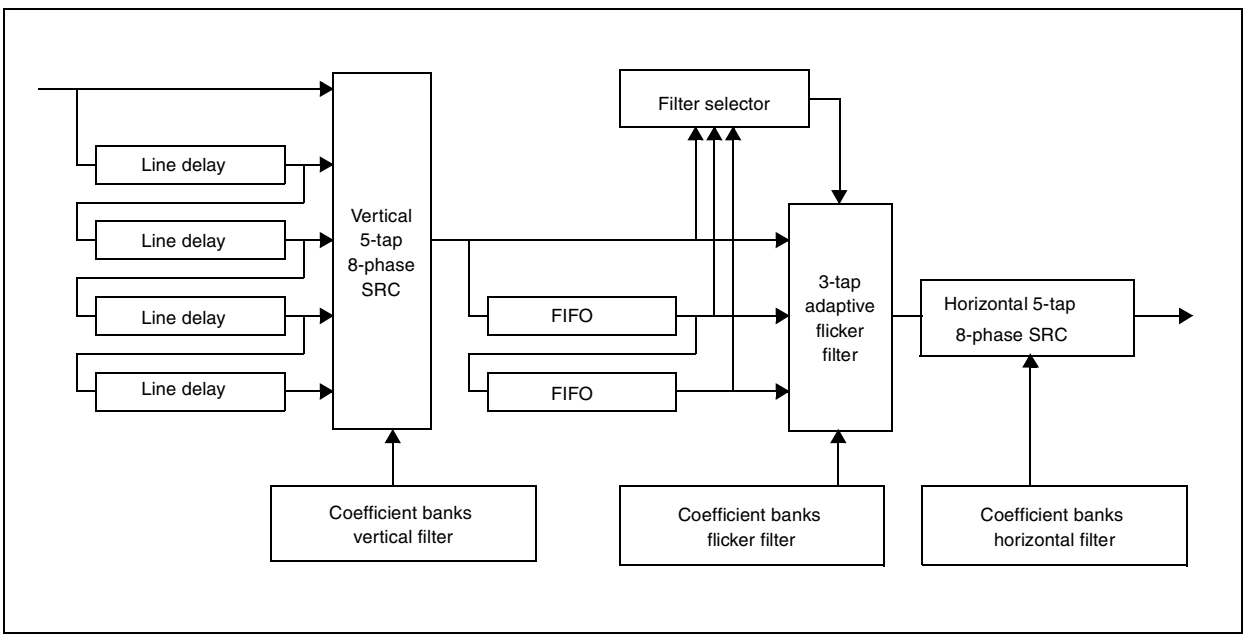
Register BLT_CML is a pointer to the local memory CLUT address.

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).

56.5.10 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 120: 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 494*.
- 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).

56.5.11 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.

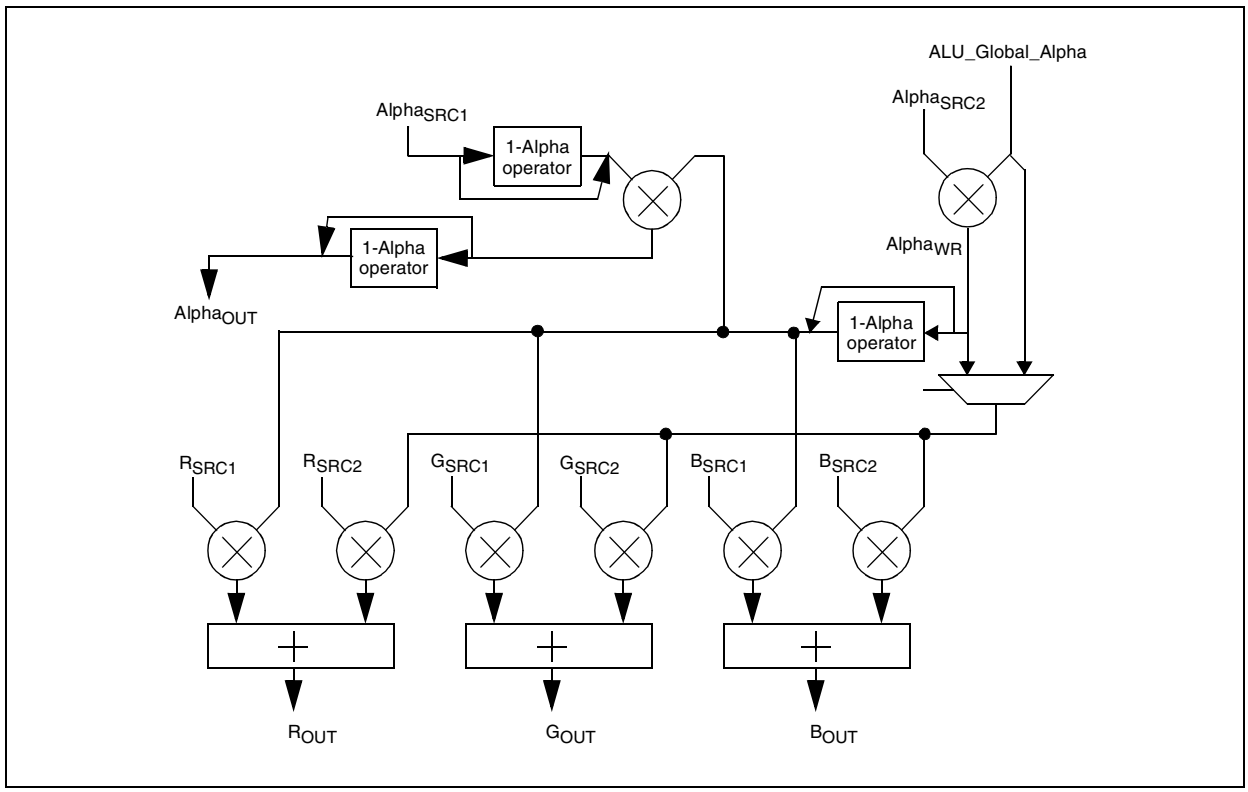
Registers BLT_FF0, BLT_FF1, BLT_FF2, BLT_FF3 define the four filters in terms of their (*n* - 1), *n*, and (*n* + 1) coefficients, and the thresholds of the 1st three filters.

56.5.12 ALU (alpha blending and boolean operator)

Alpha blending

The alpha blender interpolates colors between source 1 and source 2 (see Figure 121).

Figure 121: 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. Source 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 141: 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_REPEAT = 0 and COLOR_FORM = YCbCr420Mb in register BLT_S2TY).

56.5.13 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)
```

56.5.14 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 (BLT_TSZ) is the size of the target window (equals source 1 size).

All formats are available except YCbCR420MB and YCbCR422MB.

56.6 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_REPEAT) 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_REPEAT (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 (BLT_TSZ).WIN_HEIGHT = new line height,
  BLT_S1SZ (BLT_TSZ).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 (BLT_TSZ).WIN_HEIGHT = luma line height,
  BLT_S1SZ (BLT_TSZ).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.

56.7 Blitter XYL mode

56.7.1 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 142: 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 56.5.1 on page 483).

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 143 shows the features available when working in XYL mode.

Table 143: 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).

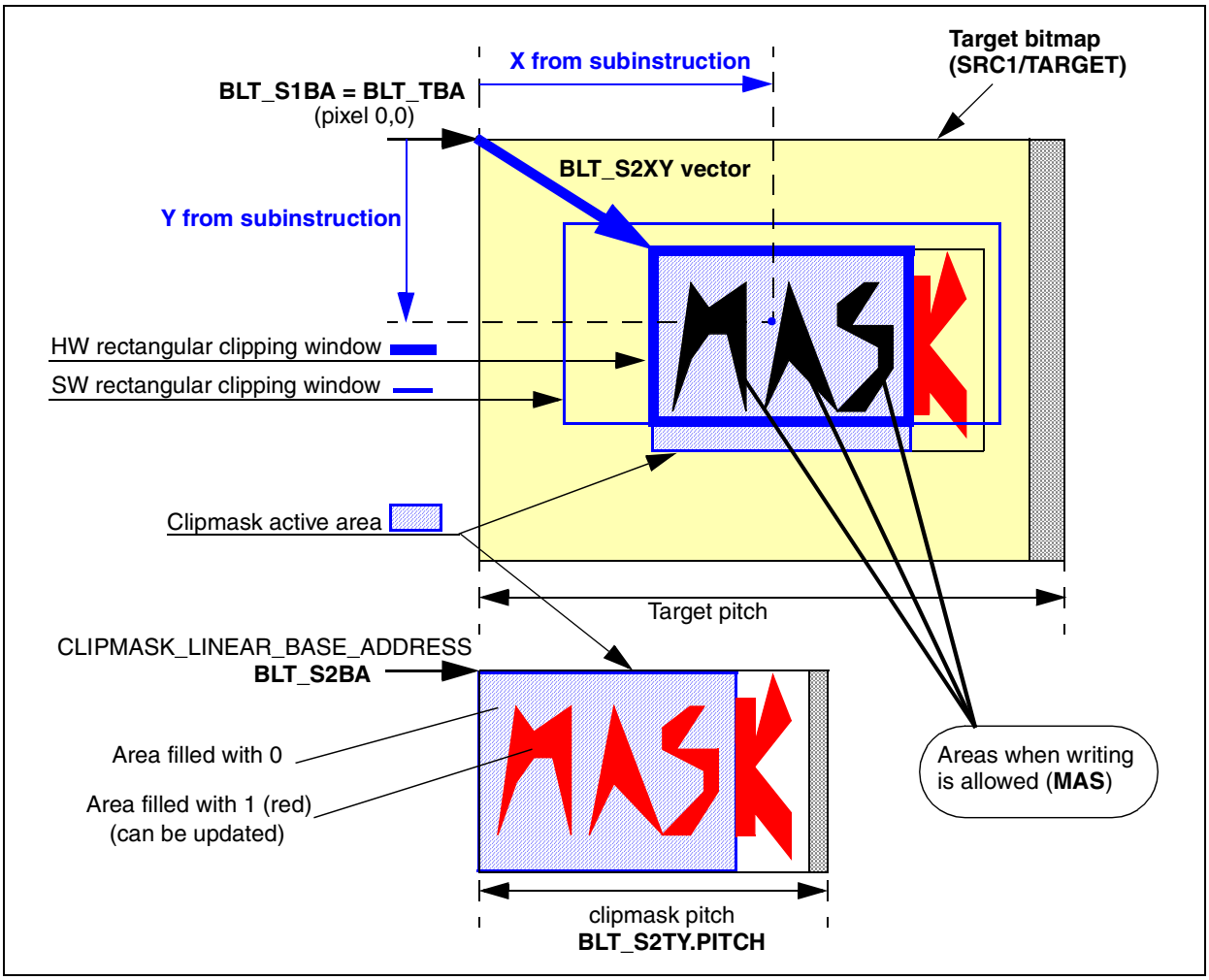
In a MediaHighway drawing context, a clipmask validity window is defined, as well as a standard rectangular clipping window. For the blitter XYL to be compliant, the hardware rectangular clipping window must be programmed (BLT_CWO and BLT_CWS), with the intersection of the two software drawing context 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 56.5.12 on page 498 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 122: Example of an XY access with clipmask (logical ROP)



56.7.2 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 NB_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

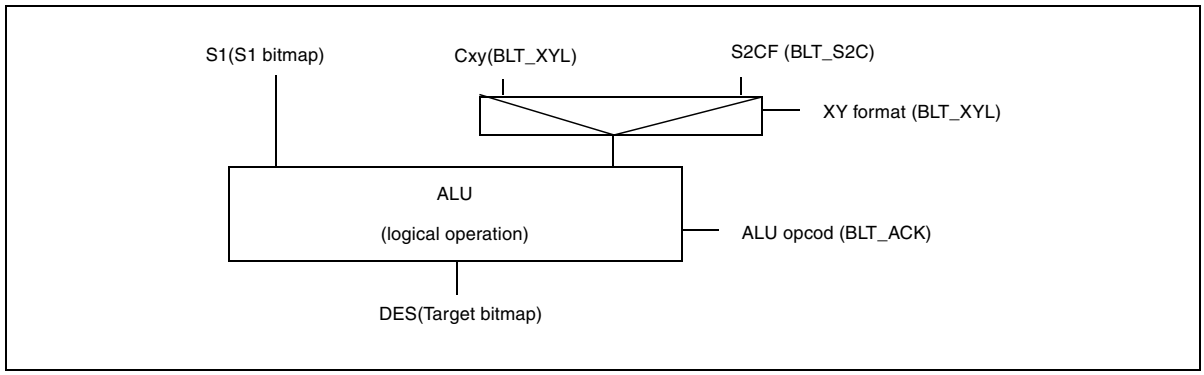
0001, 0010, 0011: logical operation, and blend

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 123: 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)

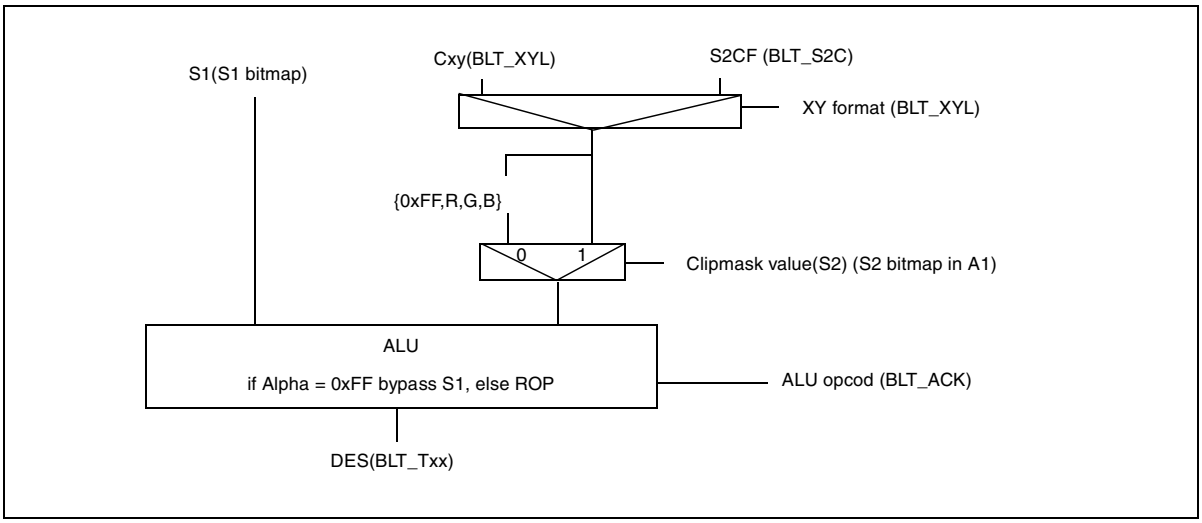
The clipmask is accessed through the source 2 interface. It means source 2 has to be fully set to access the clipmask bitmap (BLT_S2BA, BLT_S2TY and BLT_S2SZ).

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.

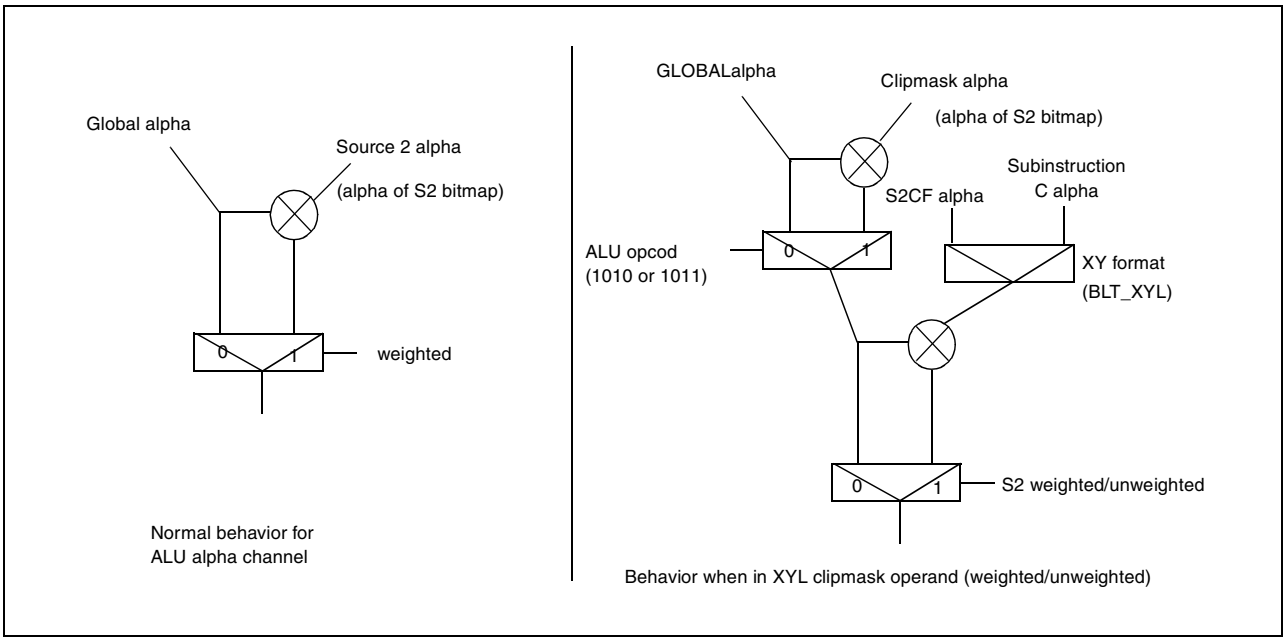
Figure 124: Clipmask ROP in XYL format



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 125.

Figure 125: Destination alpha computation

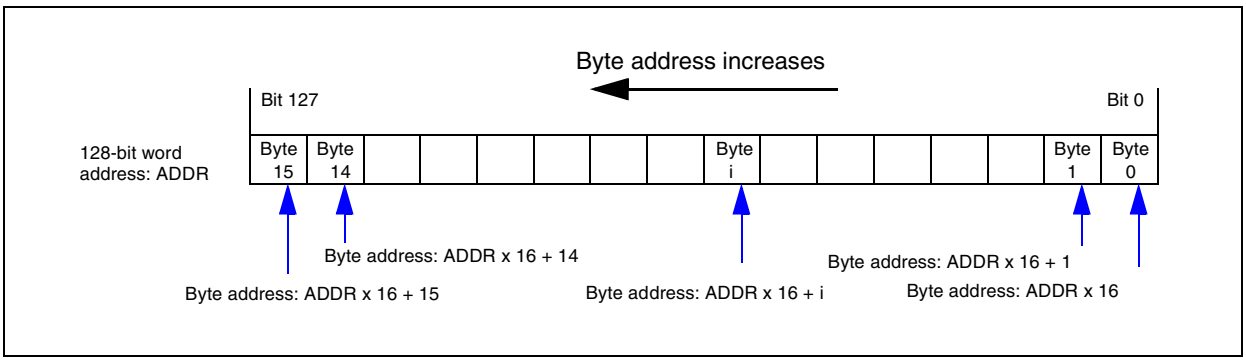


56.8 Local memory storage of supported graphics formats

56.8.1 General aspects

Memory storage (128-bit word) for all graphics data is little endian.

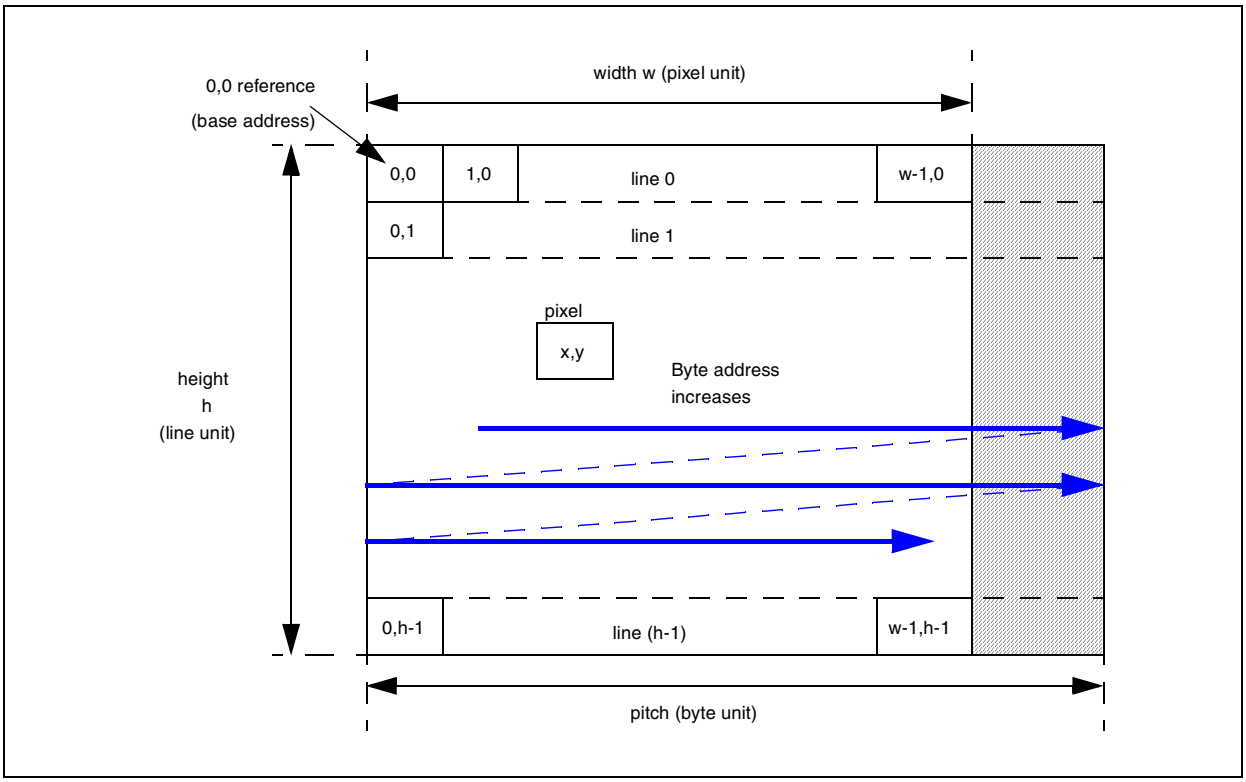
Figure 126: Little endian convention for a 128-bit word



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 127: 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.

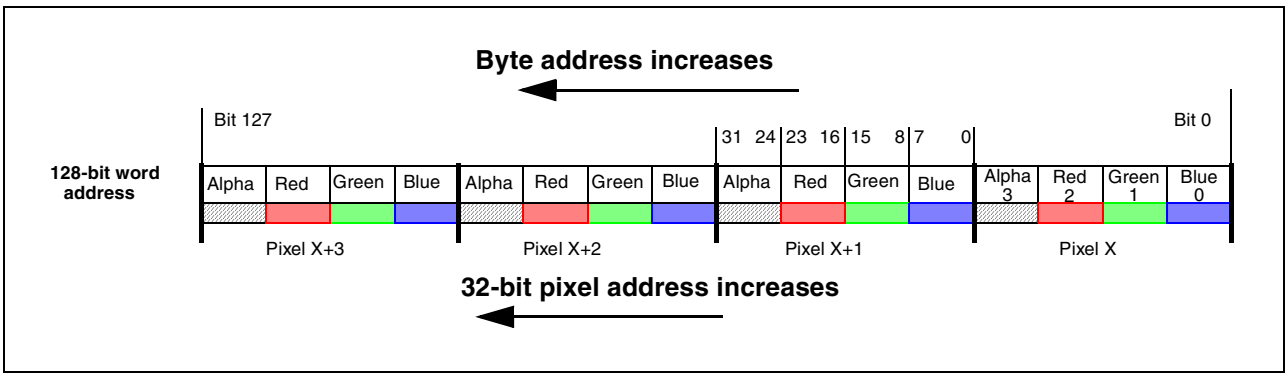
The pitch is the distance in bytes between any pair of vertically adjacent pixels (same x coordinate, y/y+1). The pitch is at least the width multiplied by the number of bytes per pixel, but

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.

56.8.2 32-bit per pixel format - true color

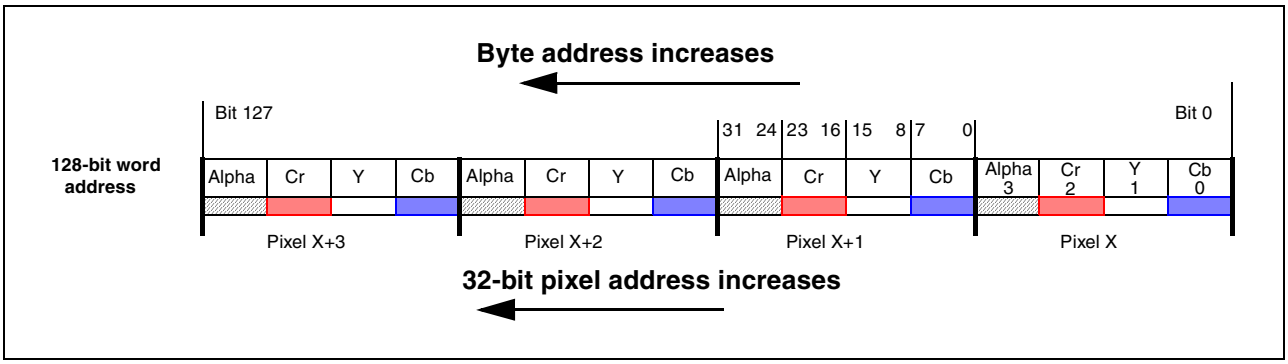
Figure 128: ARGB8888 alignment in a 128-bit word



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 129: AYCbCr8888 alignment in a 128-bit word

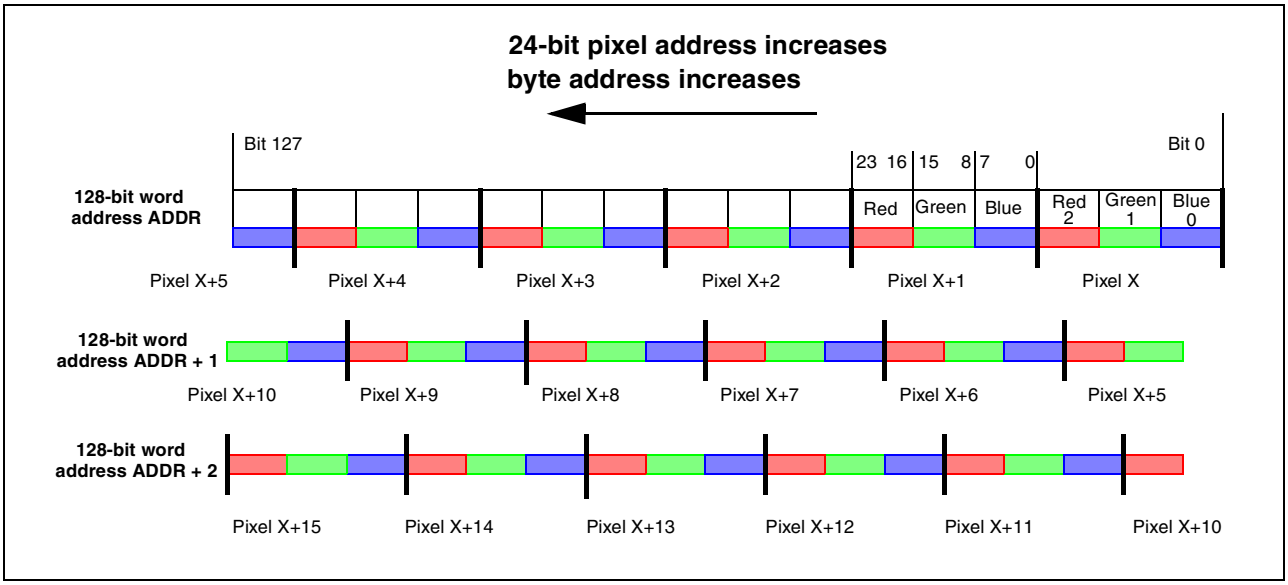


Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

56.8.3 24-bit per pixel format - true color

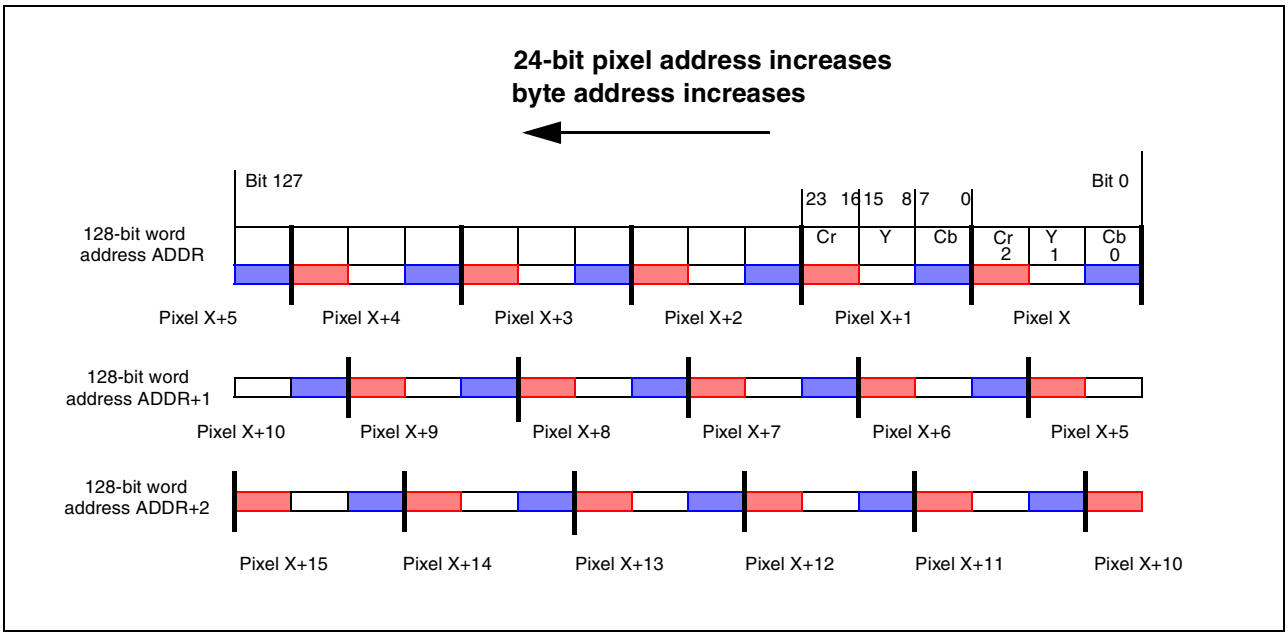
Figure 130: RGB888 alignment in a 128-bit word (3-word period/16-pixel period)



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.

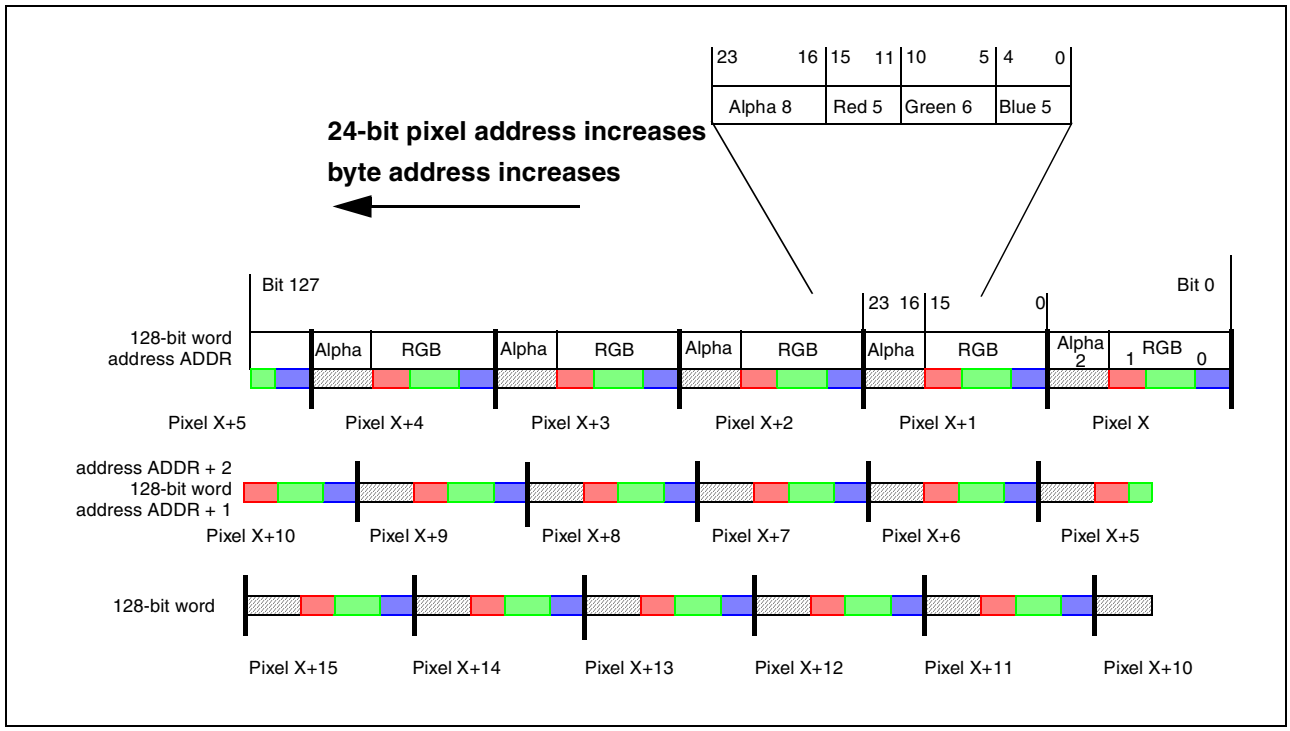
Figure 131: YCbCr888 alignment in a 128-bit word (3-word period/16-pixel period)



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 132: ARGB8565 alignment in a 128-bit word (3-word period/16-pixel period)

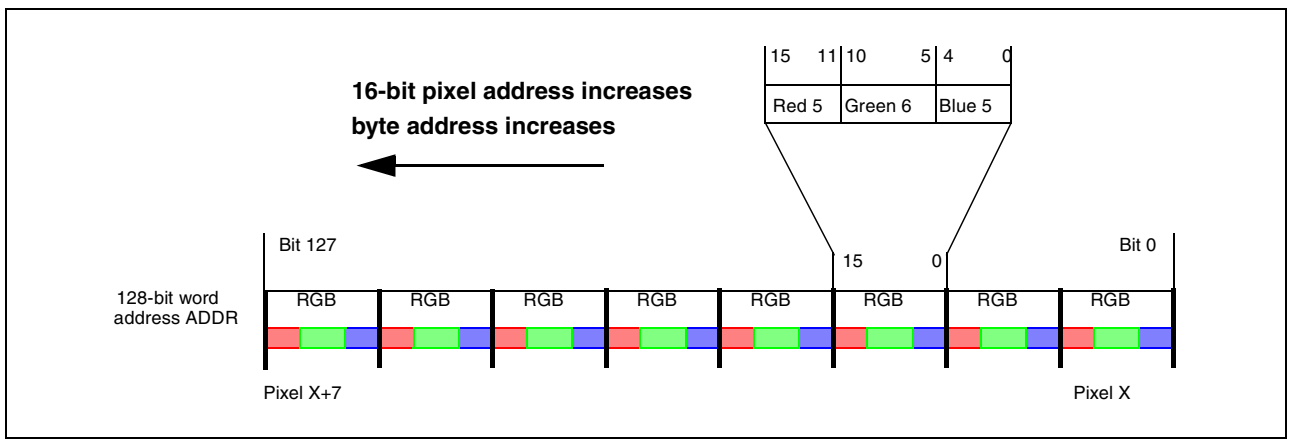


Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

56.8.4 16-bit per pixel format - true color

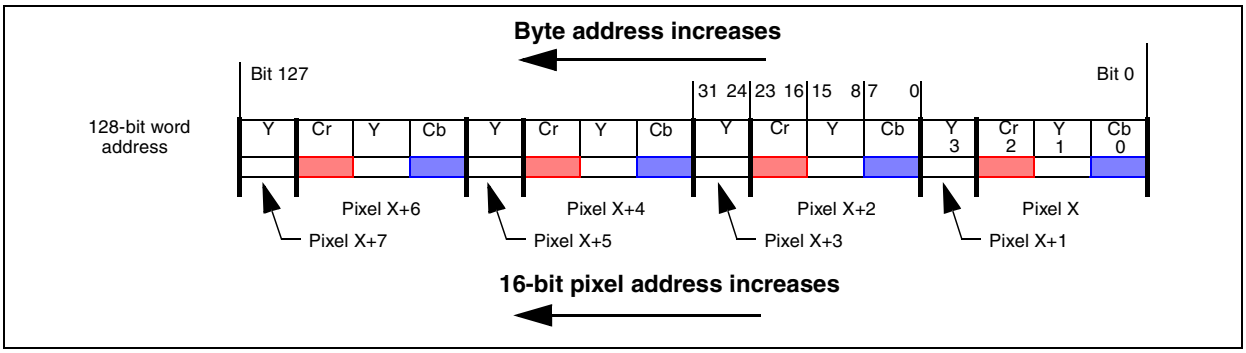
Figure 133: RGB565 alignment in a 128-bit word



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 134: YCbCr 4:2:2 (16 bpp)

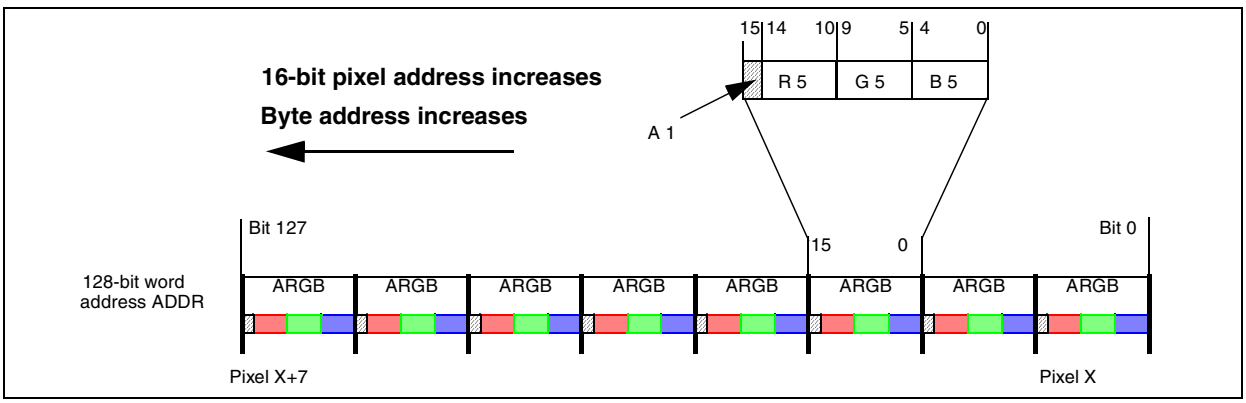


Note: 1 The memory address for pixel (0,0) must be aligned on a 32-bit word address boundary.

2 *The pitch value must be a multiple of four bytes.*

3 *X is even on this diagram.*

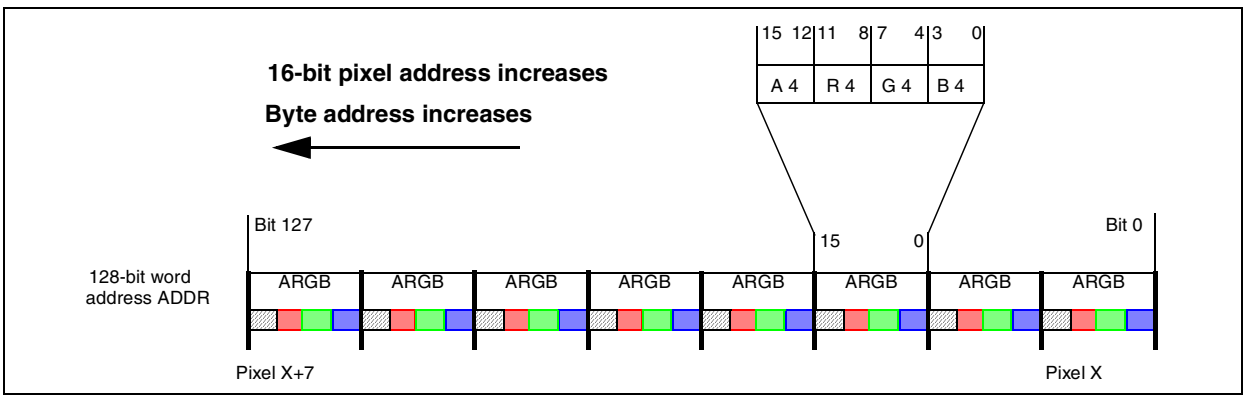
Figure 135: ARGB1555 alignment in a 128-bit word



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 136: ARGB4444 alignment in a 128-bit word

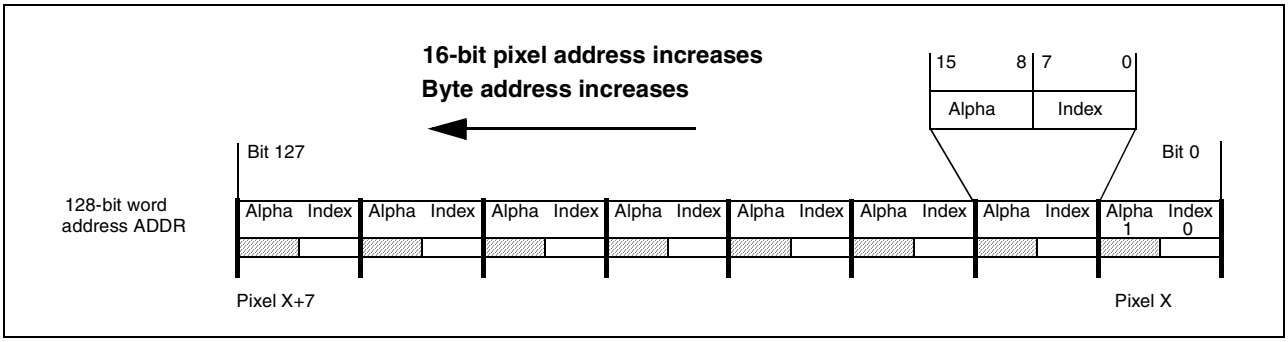


Note: 1 The memory address for pixel (0,0) must be aligned on a 16-bit word address boundary.

2 *The pitch value must be a multiple of two bytes.*

56.8.5 16-bit per pixel format - CLUT based

Figure 137: ACLUT88 alignment in a 128-bit word

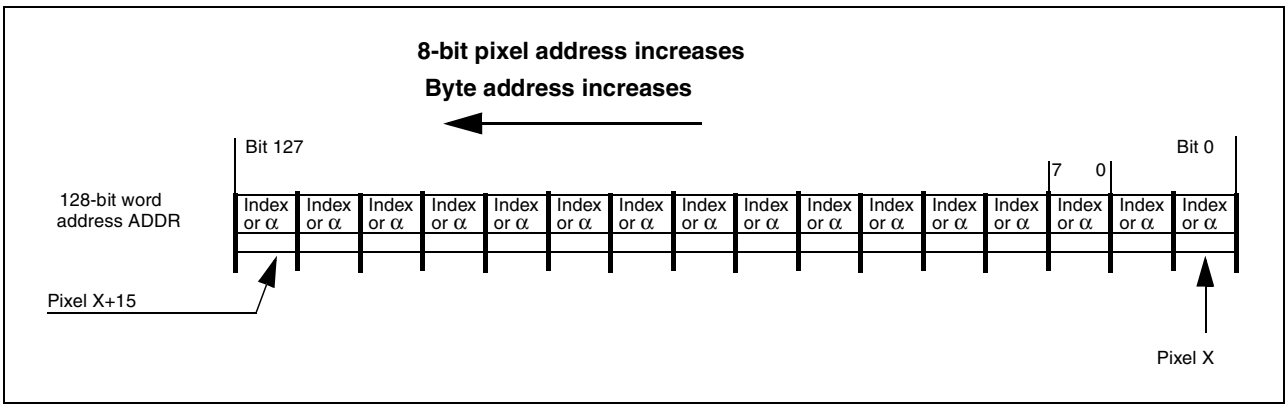


Note: 1 The memory address for pixel (0,0) must be aligned on a 16-bit word address boundary.

2 The pitch value must be a multiple of two bytes.

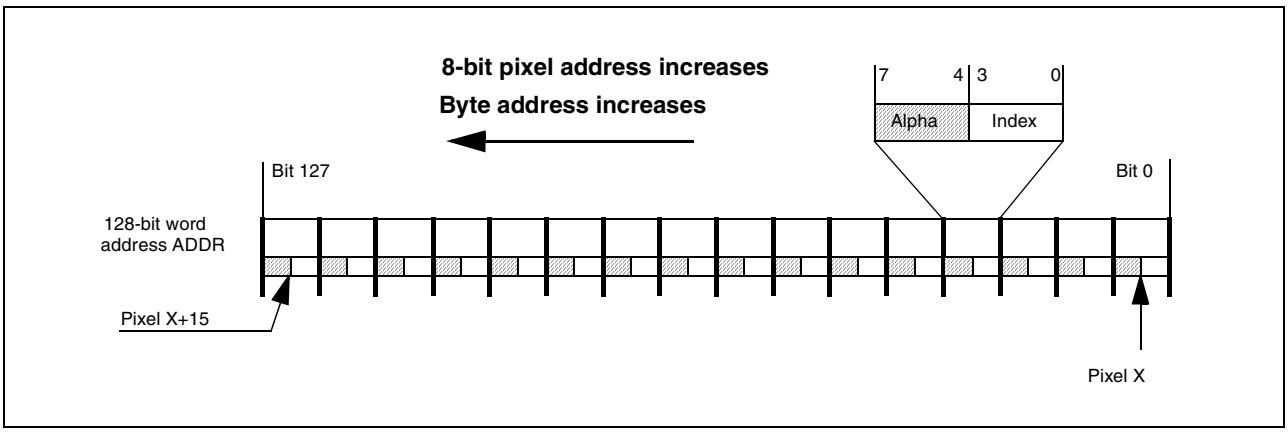
56.8.6 8-bit per pixel format

Figure 138: CLUT8 / A8 alignment in a 128-bit word



Note: The memory address for pixel (0,0) must be byte-aligned.

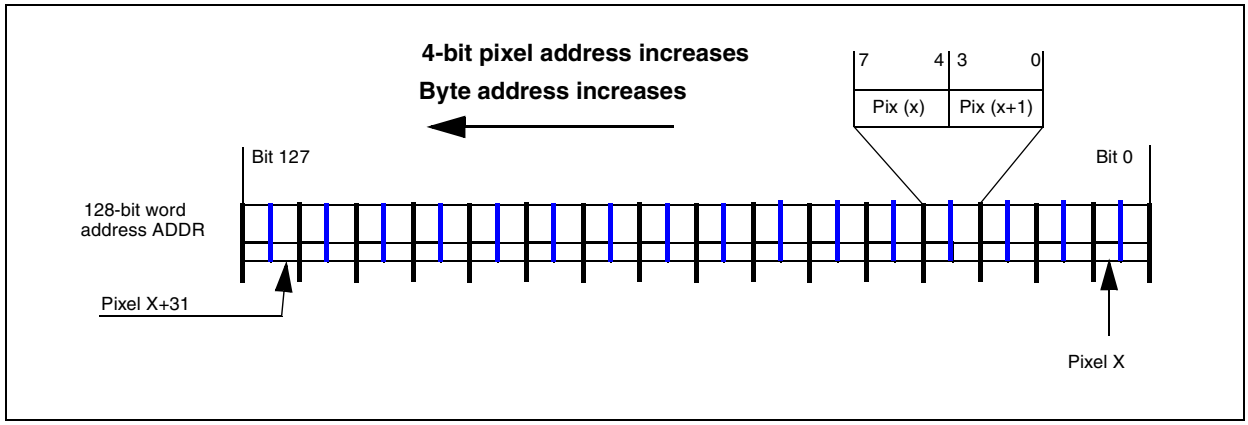
Figure 139: ACLUT44 alignment in a 128-bit word



Note: The memory address for pixel (0,0) must be byte-aligned.

56.8.7 1/2/4 bpp format

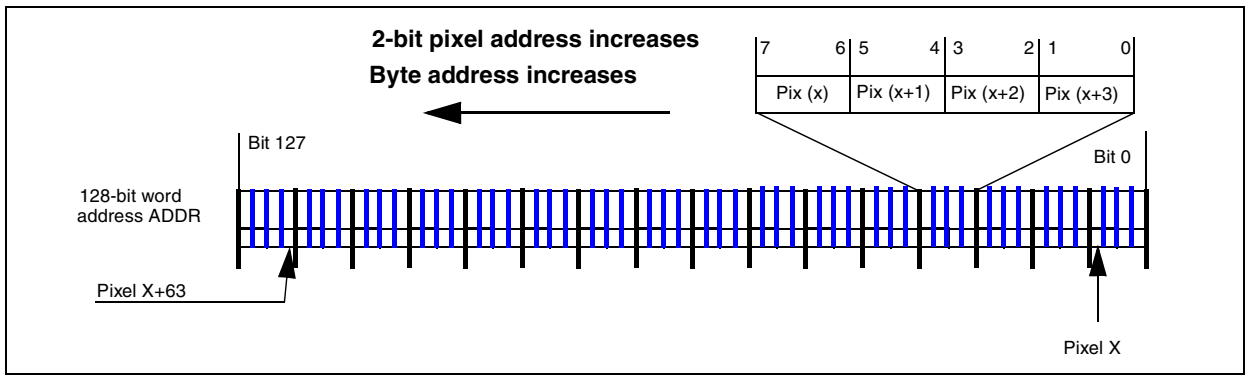
Figure 140: ACLUT4 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be byte-aligned.

2 The pitch remains in byte units.

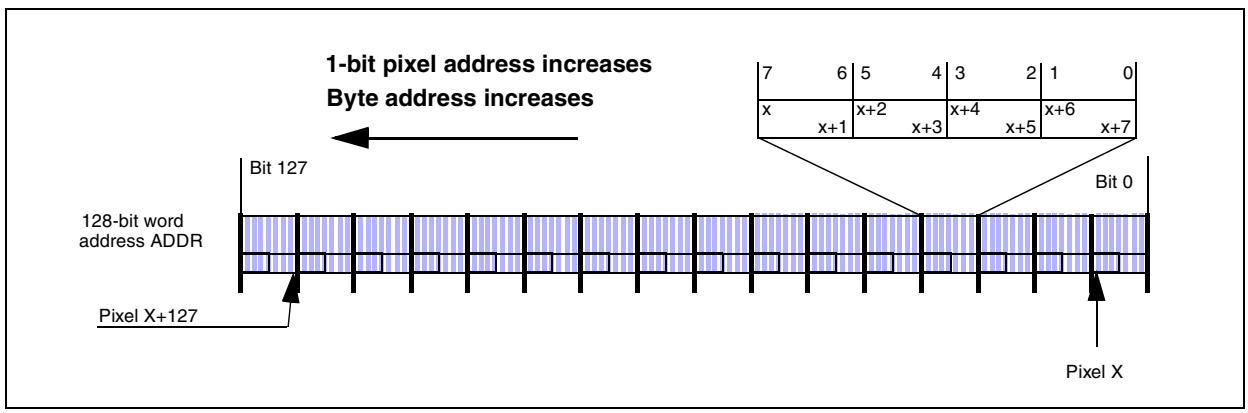
Figure 141: CLUT2 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be byte-aligned.

2 The pitch remains in byte units.

Figure 142: ACLUT1 / A1 alignment in a 128-bit word



Note: 1 The memory address for pixel (0,0) must be byte-aligned.

2 The pitch remains in byte units.

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.

57 2D graphics processor registers

Blitter register addresses are provided as
BlitterBaseAddress + *BlitterRegsBaseAddress* + offset.

Blitter 2D coefficient addresses are provided as
BlitterBaseAddress + *Blitter2DFilterBaseAddress* + offset.

The *BlitterBaseAddress* is:
0x3800 3000.

The *BlitterRegsBaseAddress* is:
0xA00.

The *Blitter2DFilterBaseAddress* is:
0xB00.

57.1 Register map

Table 144: Blitter register map overview

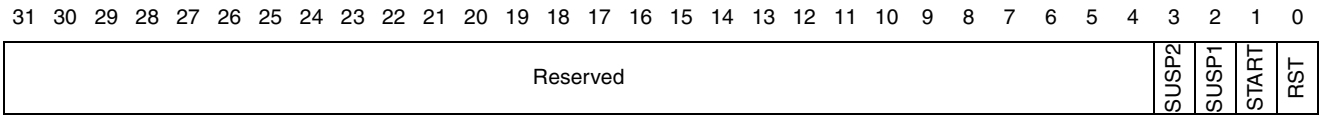
Register function	Register	Offset	Type	128-bit word alignment	
CPU Control / status registers	BLT_CTL	0x00	R/W		
	BLT_STA1	0x04	R/W		
	BLT_STA2	0x08	RO		
	BLT_STA3	0x0C	RO		
General configuration	BLT_NIP	0x10	R/W/LLU	Memory node	Node 128-bit word 1
	BLT_USR	0x14	RO/LLU		
	BLT_INS	0x18	RO/LLU		
Source 1 configuration	BLT_S1BA	0x1C	RO/LLU		
	BLT_S1TY	0x20	RO/LLU		Node 128-bit word 2
	BLT_S1XY	0x24	RO/LLU		
	BLT_S1SZ (BLT_TSZ)	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 144: 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		

57.2 Register descriptions

BLT_CTL Blitter control



Address: *CompositorBaseAddress* + *BlitterOffset* + 0x00
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: This register controls the blitter activity.

[31:4] **Reserved**

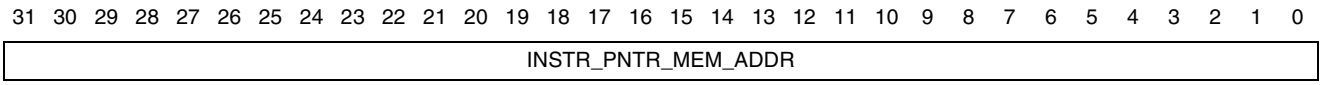
[3] **SUSP2:** when set to 1, this bit suspends the blitter as soon as possible:
- after the current instruction is completed if the 2D-resize engine or the flicker-filter is enabled,
- at the end of the current target line in other cases.

[2] **SUSP1:** when set to 1, this bit suspends the blitter after the current instruction is completed

[1] **START:** write 1 then 0 starts the blitter: The node at the address stored in BLT_NIP is fetched and executed

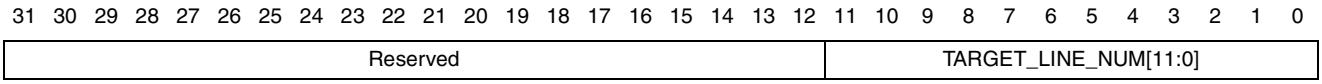
[0] **RST:** soft reset: the blitter goes back to an idle state.

BLT_STA1 Blitter status 1



Address: *CompositorBaseAddress* + *BlitterOffset* + 0x04
Type: RO
Buffer: Immediate
Reset: Undefined
Description: This register provides the memory address of the instruction pointer for the current blitter operation.

BLT_STA2 Blitter status 2



Address: *CompositorBaseAddress* + *BlitterOffset* + 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

31	30	29	28	27	26	25	24	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																												SUSP	IRQr	IRQc	READY

Address: *CompositorBaseAddress* + *BlitterOffset* + 0x0C

Type: R/W (bit 0 is read only)

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_CTL.START, is required to resume; no new node is fetched prior to the execution.)

[1] **IRQc:** Current_Blit_Completed IRQ

[0] **READY:** ready when 1, busy when 0 (read only bit). Bit READY 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

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

Address: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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_CTL							XYL	Reserved		IRQ_MASK			Reserved			PLANE_MASK	Reserved	OCSC	CKEY	RECT_CLIP	FILCK_FILT	2DRESCALE	CLUTOP	ICSC	Reserved	SOURCE2		SOURCE1		

Address: *CompositorBaseAddress* + *BlitterOffset* + 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_CTL**: 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_CTL.
[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] **SOURCE2:** source 2 mode
00: Source 2 disabled
01: Source 2 fetched from memory
10: Reserved
11: Source 2 color fill register

[2:0] **SOURCE1:** 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

31–17	16	15–11	10–0
Reserved	VTG	Reserved	TRIG_LINE_NUM

Address: *CompositorBaseAddress* + *BlitterOffset* + 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

31–26	25–0
S1BA_BANK_NUM	S1BA_MEM_ADDR

Address: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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 byte unit

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 (BLT_TSZ) 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: *CompositorBaseAddress* + *BlitterOffset* + 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**

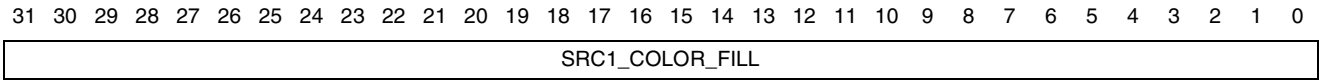
[27:16] **WIN_HEIGHT:** height (in lines) of window to be transferred

[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



Address: *CompositorBaseAddress* + *BlitterOffset* + 0x2C

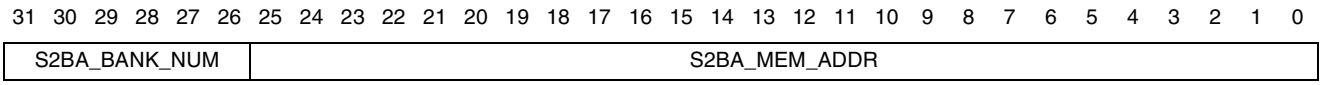
Type: Read/LLU

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



Address: *CompositorBaseAddress* + *BlitterOffset* + 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

31	30	29	28	27	26	25	24	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	LINE_REPEAT	CHROMA_LEFT	VSO	HSO	MB_FIELD	Reserved	ALPHA_RANGE	COLOR_FORM					PIXMAP_PITCH															

Address: *CompositorBaseAddress* + *BlitterOffset* + 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_REPEAT:** 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
00000: RGB565	01000: CLUT1
00001: RGB888	01001: CLUT2
00100: ARGB8565	01010: CLUT4
00101: ARGB8888	01011: CLUT8
00110: ARGB1555	01100: ACLUT44
00111: ARGB4444	01101: ACLUT88
YCbCr types	MISC types
10000: YCbCr888	11000: A1
10010: YCbCr422R	11001: A8
10011: YCbCr422MB	11111: BYTE
10100: YCbCr420MB	
10101: 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 byte unit

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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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. The register content is "don't care" when XYC or XYLC format is used.

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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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 2x2dither 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

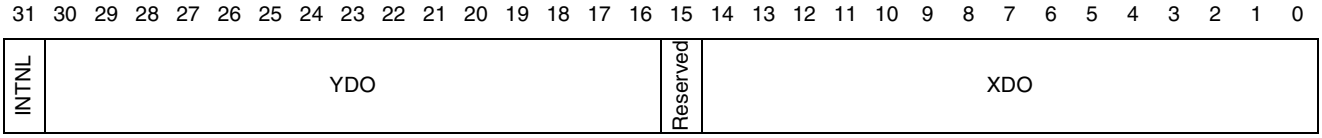
RGB types	CLUT types
00000: RGB565	01000: CLUT1
00001: RGB888	01001: CLUT2
00100: ARGB8565	01010: CLUT4
00101: ARGB8888	01011: CLUT8
00110: ARGB1555	01100: ACLUT44
00111: ARGB4444	01101: ACLUT88
YCbCr types	Misc types
10000: YCbCr888	11000: A1
10010: YCbCr422R	11001: A8
10011: YCbCr422MB	11111: BYTE
10100: YCbCr420MB	
10101: 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 byte unit

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: *CompositorBaseAddress* + *BlitterOffset* + 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)

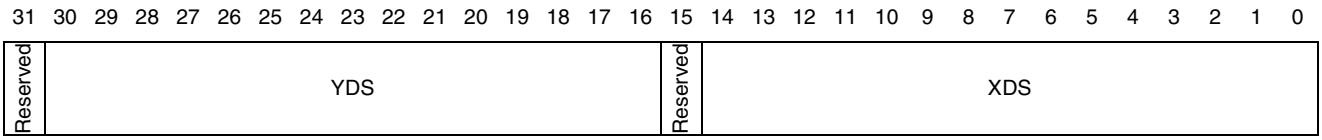
[30:16] **YDO:** Y coordinate (unsigned)

[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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 0x58

Type: Read/LLU

Buffer: Double-bank: automatic hardware toggle

Reset: 0x00

Description: This register controls the 2 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																												Reserved			
																CML_MEM_ADDR															

Address: *CompositorBaseAddress* + *BlitterOffset* + 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 4 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

31	30	29	28	27	26	25	24	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									OFFSET_VSRC			Reserved	OFFSET_HSRC			Reserved		AB_V	AB_H	Reserved		FF_MODE		Reserved		2DVF_MODE			Reserved		2DHF_MODE	

Address: *CompositorBaseAddress* + *BlitterOffset* + 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

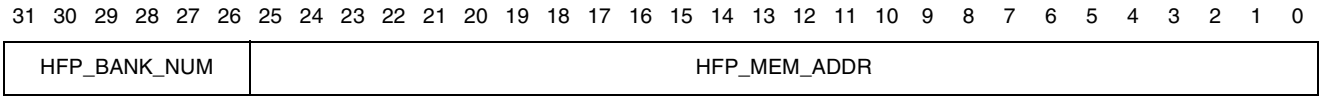
[7] **Reserved**

[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

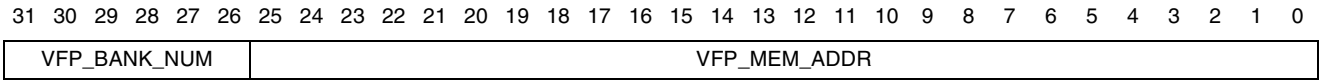


Address: *CompositorBaseAddress* + *BlitterOffset* + 0x64
Type: Read/LLU
Buffer: Double-bank: automatic hardware toggle
Reset: 0x00
Description: This register provides the byte memory location for retrieving the horizontal filter coefficients in the local memory (40 bytes).

[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: *CompositorBaseAddress* + *BlitterOffset* + 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

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

Address: *CompositorBaseAddress* + *BlitterOffset* + 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

31	30	29	28	27	26	25	24	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				T0				COEFF0_N3								COEFF0_N2								COEFF0_N1							

Address: *CompositorBaseAddress* + *BlitterOffset* + 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

31	30	29	28	27	26	25	24	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				T1				COEFF1_N3								COEFF1_N2								COEFF1_N1							

Address: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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] 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] 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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: *CompositorBaseAddress* + *BlitterOffset* + 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
NB_SUBINS																Reserved						SUBINS		Reserved							

Address: *CompositorBaseAddress* + *BlitterOffset* + 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] **NB_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: *CompositorBaseAddress* + *BlitterOffset* + 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

31	30	29	28	27	26	25	24	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		PKT_SIZE		

Address: *CompositorBaseAddress* + *BlitterOffset* + 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

31	30	29	28	27	26	25	24	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																						BANDWIDTH_COUNT									

Address: *CompositorBaseAddress* + *BlitterOffset* + 0xF0
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: This register controls STBus bandwidth limiter.

[31:10] **Reserved**

[9:0] **BANDWIDTH_COUNT**: sets the number of cycle between 2 requests on the STBus

57.3 Blitter 2D-filter coefficients

57.3.1 Register map

Table 145: 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).

57.3.2 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: *CompositorBaseAddress* + *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 8 subfilters for the HSRC, each subfilter has 5 8-bit coefficients. Coefficient j of subfilter i (subposition i) is designated Ki.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: *CompositorBaseAddress* + *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 8 subfilters for the VSRC, each subfilter has 5 8-bit coefficients. Coefficient j of subfilter i (subposition i) is designated Ki.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.

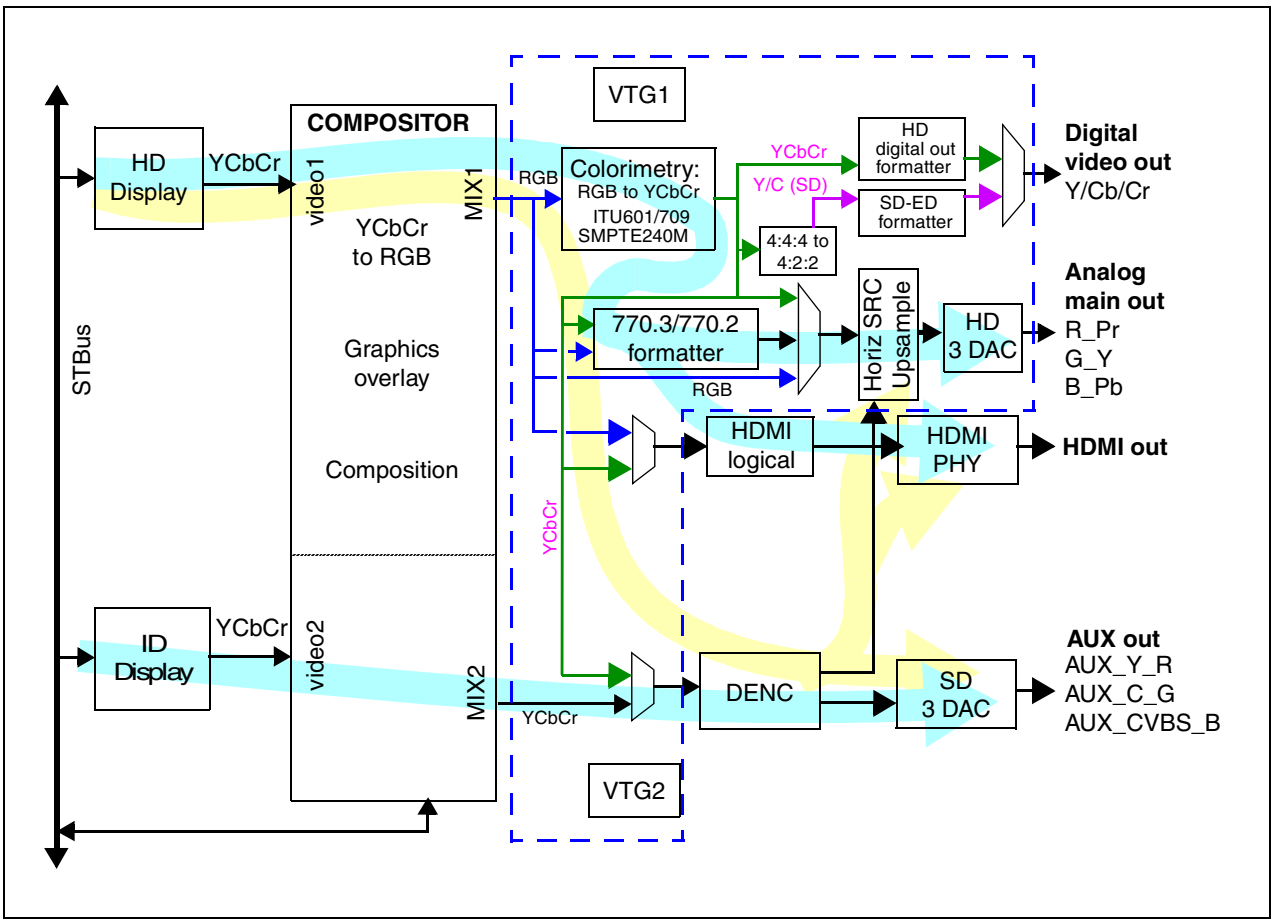
58 Video output stage (VOS)

The VOS receives video data from the Compositor Main and Auxiliary outputs and formats them for use by the DENC, the HD and SD DACs, the HDMI interface and the digital video output interface.

58.1 Display and output subsystem overview

Figure 143 shows the various blocks of the VOS and associated modules.

Figure 143: Output stage glue (dashed box) and related blocks



58.1.1 Typical application: main and auxiliary outputs, with main other than SD

In this configuration the output stage has two main datapaths, corresponding to the two output formats of the chip: HD and SD. These two datapaths are used in typical applications when the chip outputs in both HD (watch) and SD (record - on analog VCR). These datapaths appear in light blue on the above figure. HD in this chapter actually means anything other than 480i (SD interlaced).

Main datapath

(“HD” or rather “multi-standard” datapath)

This datapath is paced by VTG1 and the central node is mixer 1 of the compositor. The HD display reads the video frame buffer from memory, feeds the compositor/mixer 1 that overlay/blend video with required graphics, manages windowing, and outputs RGB components. These components are either directly used or pass through an RGB to YCbCr matrix supporting ITU-R

BT601, ITU-R BT709, or SMPTE240M colorimetry. Data then goes to HDMI and/or analog output. HDMI comprises a digital formatting and crypting block followed by a physical interface. Analog output is preceded by a formatting block so as to allow required output modes as below. Just in front of the DACs is an SRC able to upsample video signals to adapt pixel rate to DAC sampling rate. This SRC has programmable coefficients. With the right set of coefficients it is able to adapt HD pixel rate to DAC sampling rate via adequate interpolation while keeping maximum signal bandwidth. With a different set, “constrained output” is available, whereby the bandwidth of the video is cut down, for example to an equivalent 1/4 of resolution of the incoming video, so that full resolution is available only on encrypted HDMI.

The available outputs are as follows:

Main analog output This deals with HD format up to 3H. Video output can be full-range RGB output, full-range YPbPr output, EIA-770.2 compatible YPbPr with embedded bi-level sync or 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 following display standards are available:
SMPTE274M (1125i), SMPTE293M (525p), SMPTE295M (1250i), SMPTE296M (750p), ARIB-BS4 (525p, 750p, 1125i).

Macrovision encoding is supported on this output when configured for 525p output.

Full scale YPbPr or RGB component with no embedded sync information can be output through this datapath.

Auxiliary analog output This is an SD version of the decoded video, possibly with different graphics overlay and ancillary data. For details, refer to Chapter 60: *Compositor* *on page 599* and Chapter 68: *Digital encoder (DENC)* *on page 749*.

HDMI output supports 480p, 720p and 1080i formats as per HDMI specification.

Additional digital output A digital video output is provided as alternate solution to output video from the main datapath, mainly for debug/diagnosis, or for the optional external HDMI interface. It handles various standards upon the type of video it has to transmit. In the typical application, the following modes exist:

- for high definition content (1080i or 720p): YCbCr 16-bit 4: 2: 2 with embedded sync (digital SMPTE 274M, SMPTE295M)

- for standard (or ‘enhanced’) definition progressive content (480p): 4: 2: 2 with embedded sync (SMPTE 293M)

Auxiliary datapath (“SD” datapath)

This datapath is paced by VTG2; the central node is mixer 2 of the compositor. The ID display reads video frame buffer from the memory. It then feeds the compositor/mixer 2 that overlay/blend video with required graphics, manages windowing, and outputs YCbCr components in CCIR 601 colorimetry. Those components go then through the DENC to be output on the three SD DACs (auxiliary analog output).

Auxiliary analog video output can be either component RGB, YPbPr signals or a standard analog PAL/NTSC composite signal (CVBS) and S-VHS (Y/C) output. The DENC supports insertion of various VBI data on programmed lines, including close caption and teletext.

58.1.2 Alternate application (two SD-only outputs)

In an example alternate application, the main output is also SD and needs to be output on both HD and SD DACs. In this mode another datapath is used, in yellow on Figure 143.

The mixed datapath is paced by VTG1 and central node is mixer 1 of the compositor. The HD display reads video frame buffer from the memory, feeds the compositor/mixer 1 that overlay/blend video with required graphics, manages windowing, and outputs RGB components, all in SD format. In that mode, VTG2, ID Display and mixer 2 are not used. RGB components pass through RGB to YCbCr matrix (601 colorimetry) and go through the DENC, to be output on both main and auxiliary analog output. Between the DENC and the HD DACS is the upsampling SRC with right set of coefficients to perform x4 interpolation.

Main analog output in this mode is SD format, YPbPr with embedded sync.

HDMI output is 480i with duplication of pixel, allowing sampling rate of 27 MHz to fit with HDMI specifications.

Auxiliary analog has the same features as in the typical application described above.

Additional digital output: for standard definition content (480i): YCbCr 8-bit 4: 2: 2 with (ITU-R BT 656) or without embedded sync.

58.2 Display/output subsystem clocking scheme

To accommodate a variety of applications and standards and their associated pixel rate, a variety of clock combinations are possible. To enable usage of a simple analog reconstruction (anti-aliasing) filter on the application board, the HD DAC sampling clock is set to a non variable value; this value is high enough with respect to the useful spectrum to ease external filter design. Adequate upsampling with appropriate sin*x*/*x* compensation is performed internal to the chip by a programmable interpolation filter.

The main pixel master clock, from the clock generator, is a 148.5 MHz clock, 148.5/1.001 MHz clock, 108 MHz clock or 108 x 1.001 MHz clock

- 148.5 MHz is used in a typical application where the Main output is 1080i or 720p at 30 or 60 Hz.
- 148.5/1.001 MHz (approximately 148.352 MHz but exact fraction has to be kept) is used in a typical application where the Main output is 1080i or 720p at 29.97 Hz or 59.94 Hz.
- 108 MHz is used:
  - in a typical application where Main output is 480p at 59.94 Hz,
  - in an alternate application (both outputs are then 480i at 29.97 Hz).
- 108 x 1.001 (108.108 MHz) is used in a typical application where the Main output is 480p at 60 Hz.

Aux pixel master clock, from clock generator is a 27 MHz clock.

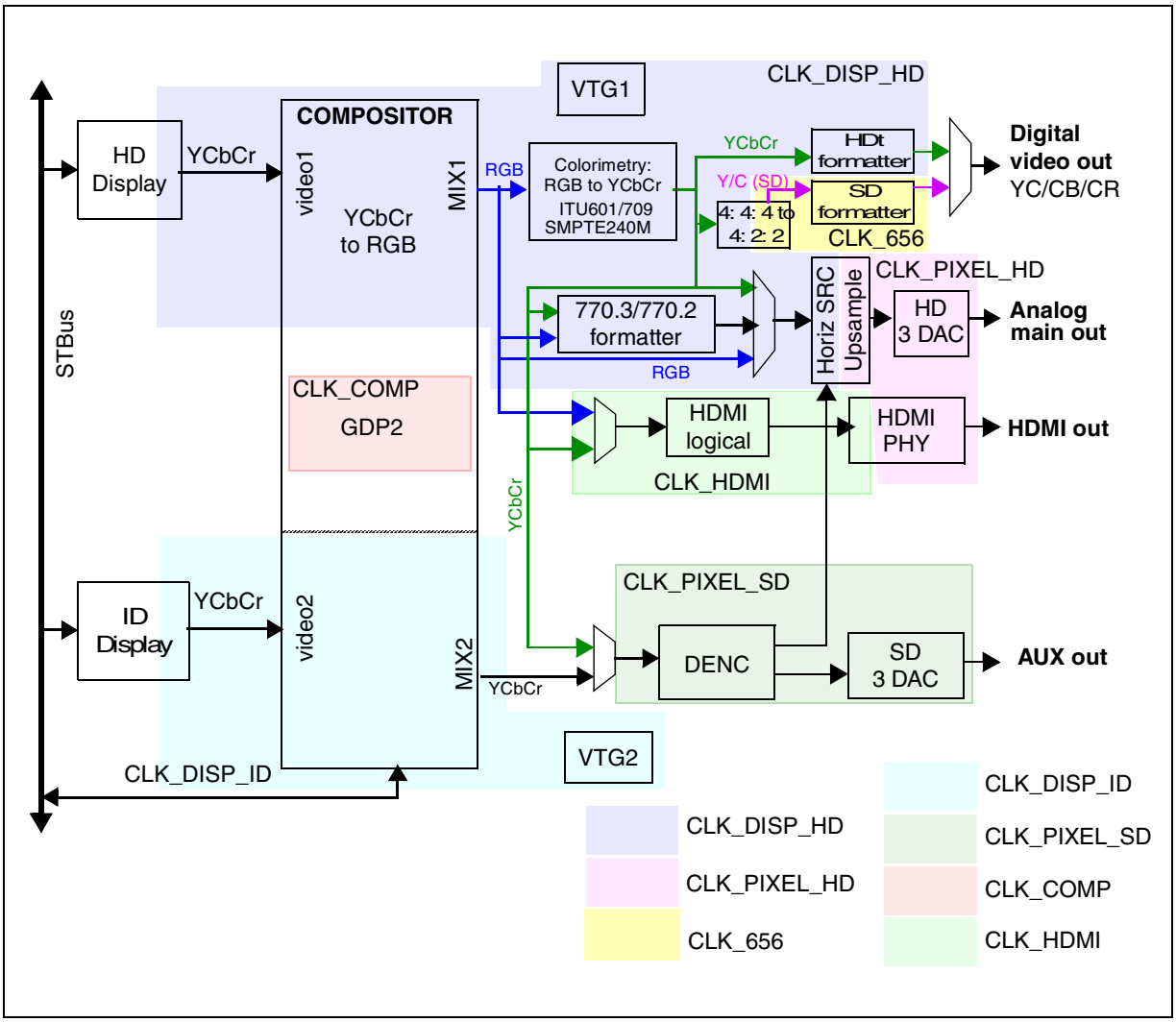
Note: 1080i and 720p are 30/60 Hz display standards; 480i and 480p are 29.97/59.94 Hz display standards. For some applications where it is preferable to keep the input rate for the display, the clock is altered by the 1.001 factor to adapt the output rate by keeping the scanning. This is possible in 1080i, 720p and 480p formats but not in 480i format where the CVBS or C outputs, basically the chroma modulation, are very sensitive to clock rate and can not accept such a divergence.

The VOS implements a large number of clock domains to cope with all applications:

- CLK_PIXEL_HD: main pixel master clock, clocking HD DACs and TMDS block (HDMI phy),
- CLK_DISP_HD: pixel clock of main path (VTG1, Mixer1, HD Display, main path formatters),
- CLK_DISP_ID: pixel clock of auxiliary path (VTG2, Mixer2, ID Display),
- CLK_COMP: pixel clock of GDP2, depending on GDP2 direction selection,
- CLK_PIXEL_SD: double SD pixel rate clock (27 MHz) for DENC and SD DACs,
- CLK_656: double pixel clock for CCIR 656/SMPTE 293M formatter,
- CLK_HDMI: clock used by the HDMI digital formatting block.

For more details see Chapter 7: *Clocks on page 64*.

Figure 144: VOS clock domains



58.2.1 Applications

Table 146 summarizes the values of the clocks, in MHz, with regards to the applications. Related clocks (clocks with phase relationships, obtained by division of the same master) are provided with a background color code: magenta for main pixel clock related, blue for auxiliary related.

Table 146: Clock domains by applications

Application			CLK_PIXEL_HD	CLK_DISP_HD	CLK_PIXEL_SD	CLK_DISP_SD	CLK_656	CLK_HDMI	CLK_COMP
Main & Aux	Main 1080i/30 Hz or 720p/60 Hz	GDP2 on main	148.5	74.25	27	13.5	-	74.25	74.25
		GDP2 on aux	148.5	74.25	27	13.5	-	74.25	13.5
Main & Aux	Main 480p, 59.94 Hz	GDP2 on main	108	27	27	13.5	54	27	27
		GDP2 on aux	108	27	27	13.5	54	27	13.5
Main & Aux	Main 1080i/ 29.97 Hz or 720p/59.94 Hz	GDP2 on main	~148.35	~74.176	27	13.5	-	~74.176	~74.176
		GDP2 on aux	~148.35	~74.176	27	13.5	-	~74.176	13.5
Main & Aux	Main 480p, 60 Hz	GDP2 on main	108.108	27.027	27	13.5	54.054	27.027	27.027
		GDP2 on aux	108.108	27.027	27	13.5	54.054	27.027	13.5
Alter nate	Main 480i, 29.97 Hz	GDP2 on main	108	13.5	27	-	27	27	13.5

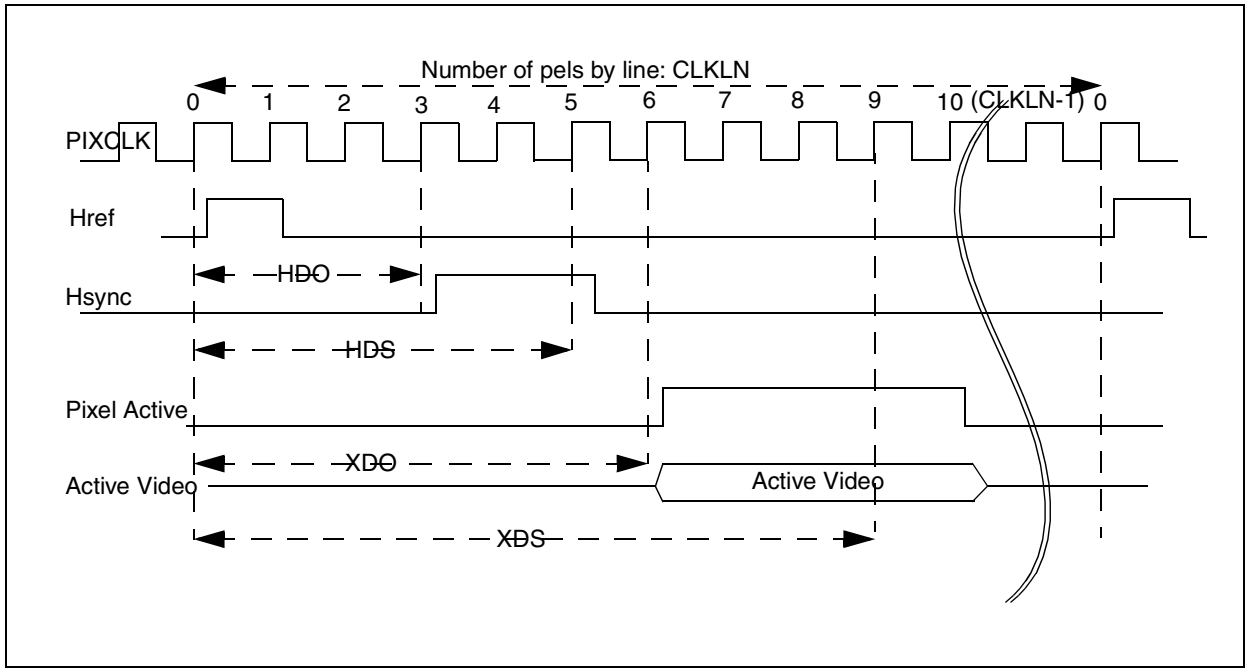
58.3 Video timing generators (VTG)

The two VTGs provide horizontal and vertical video synchronization reference signals to their respective parts of the output stage: VTG1 normally drives the main path and VTG2 drives the auxiliary path (in typical application).

Each VTG provides a vertical temporal reference (Vref) and a horizontal temporal reference (Href) used to generate Vsync and Hsync signals throughout the chip. In addition the VTGs generate signals to be represented at the output of the chip. The shape and polarity of those signals are programmable through HDO/HDS and VDO/VDS registers

58.3.1 Horizontal synchronization generation

Figure 145: Horizontal synchronization generation



All the pixels numbers in a line refer to the rising edge of Href.

- 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 Href and the rising edge of Hsync.
- HDS (register VTGn_HDS) defines the number of PIXCLK cycles between the rising edge of the internal Href and the falling edge of Hsync.
- Registers XDO and XDS, defining the active video line (horizontal dimension of 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 generation of active low HSYNC pulses.

58.3.2 Vertical synchronization generation: (interlaced output)

Figure 146: Vertical synchronization generation: (interlaced output)

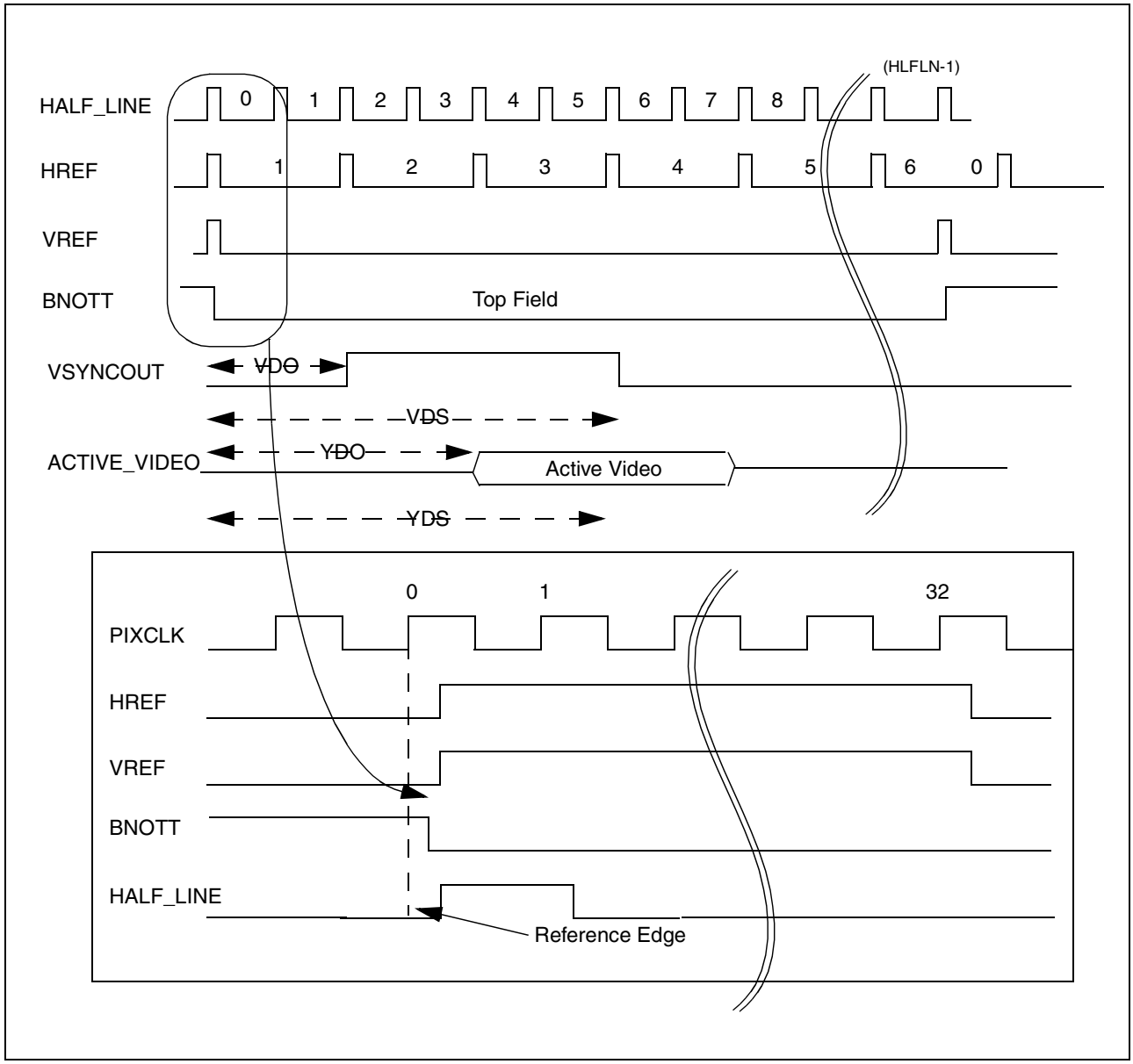


Figure 147: VTG Output (Interlaced picture: top field)

The number of half lines per field (or lines per picture) called HLFLN is specified in the VTGn_HLFLN register. The half line counter is incremented with a half-line resolution. The half-line counter 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 B/NOTT signal is also generated (in progressive output as B/NOTT will be always 0). In case of an interlaced picture with an even number of lines (like in smpte295m standard) a bit will be programmed in a register to force the VTG to be in interlaced mode. In this case the picture will have HLFLN - 1 half lines in one field and HLFLN + 1 in the other field.

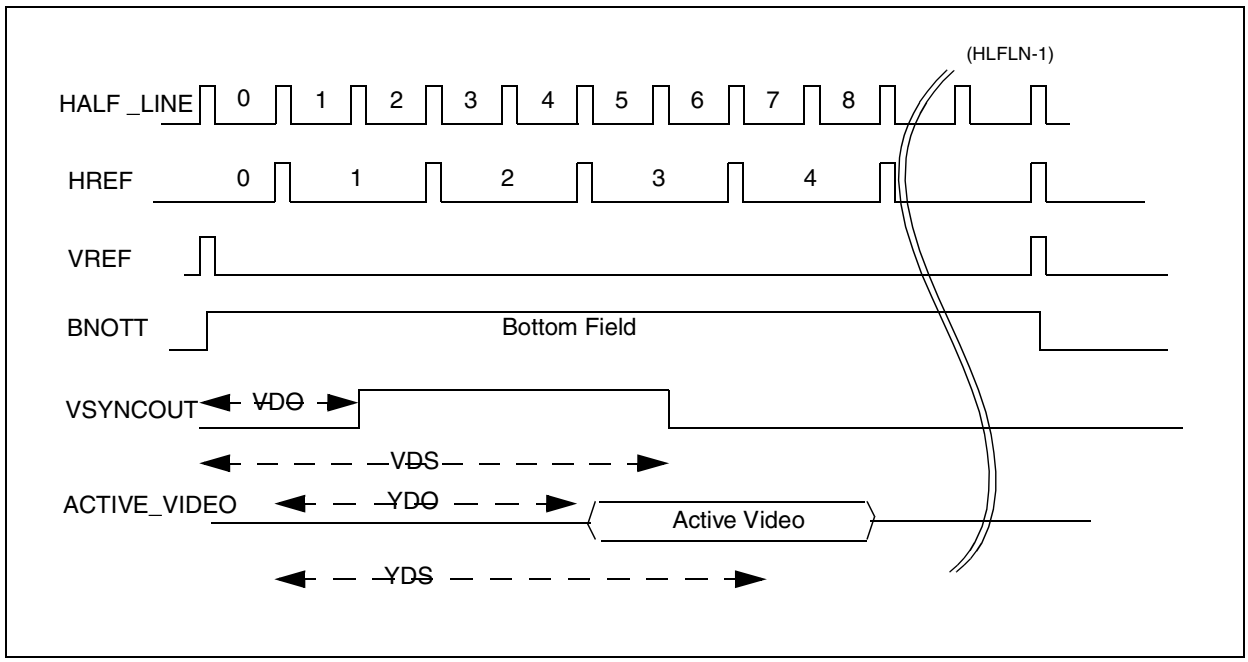
The line counter begins with 1 for vref top and 0 for vref bottom. It is then it is incremented by 1 every href.

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).

VDO (register VTGn_VDO) determines the starting position of the vertical drive output pulse relative to Vref (Half-line increment).

VDS (register VTGn_VDS) determines the ending position of the vertical drive output pulse relative to Vref (Half-line increment).

Figure 148: VTG Output (Interlaced Picture: Bottom Field)



In the above example figure: YDO = 3, YDS = 4, VDO =2, 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.*

58.3.3 Interruptions

The VTG can generate two interruptions VST and VSB. The STA.VST bit is set for a short time at the beginning of the top field corresponding to the falling edge of BnotT. The STA.VSB bit is set for a short time at the beginning of the bottom field corresponding to the rising edge of BnotT. As VST and VSB are pulse they are unlikely to be ever read as a 1 in the STA register. However, they generate interrupts through the ITS register if the corresponding ITM bits are set. In a progressive picture, the VTG continues to generate VST and VSB even if it does not make a lot of sense (as there are no top and bottom fields in a progressive display).

A timer is implemented. This generates an interrupt *n* cycles after the rising edge of vsync (*n* is a programmable value on 24 bits).

Another interrupt is generated each time BNOTT changes, and two other interrupts each time MPEG_BNOTT luma and chroma change. BNOTT concerns the output of the display main and record, MPEG_BNOTTY and MPEG_BNOTTC concern the input of the display main and record.

58.3.4 VTG modes

The following synchronization modes are supported by the STi7710 VTGs.

- For VTG1 (Main - normally HD output): Vref1 and Href1 are always free running; they generate sync signals based on internal counters.
- For VTG2 (Auxiliary - SD output):
  either:
  - Vref2 and Href2 are free running, based on internal counters, or
  - Vref2 and Href2 are based on Vref1 and Href1 from VTG1. This is to slave the auxiliary video onto the main video, or
  - Vref2 is based on Vref1 from VTG1; Href2 is based on internal count. This mode enables slaving VTG2 on a frame basis onto a source with different scanning format.

Figure 149: VTG 1 Href and Vref generation

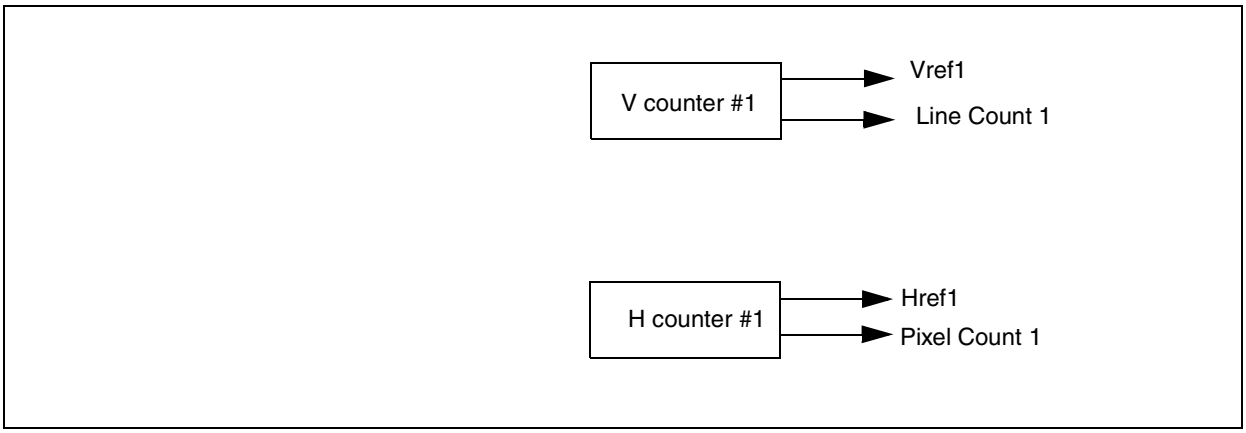
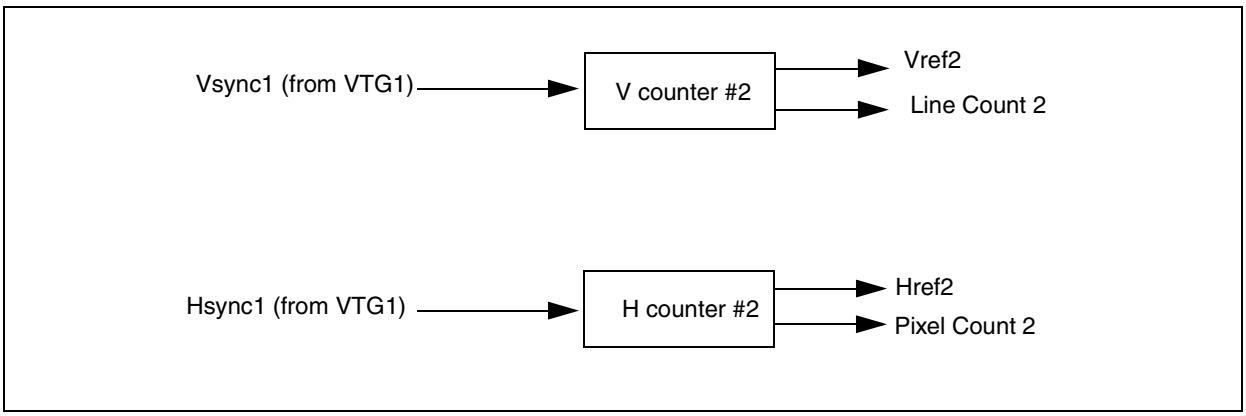


Figure 150: VTG 2 Href and Vref generation



Synchronization onto external video

When the source is MPEG video, the VTGs can be programmed to nominal value, the MPEG clock recovery mechanism normally guarantees that the produced frame rate matches the original frame rate.

In the case of external digitized video (input through the D1/ITU-R656 4: 2: 2 input interface aka DVP), the incoming data is clocked in by the D1 clock supplied externally and not controlled by the chip. Hence some synchronization mechanism is required. The STi7710 has no capability to directly lock the VTG onto the external D1 synchronization signals. Instead two other methods are available:

- Perform in software a “D1 clock recovery” similar to MPEG clock recovery. The number of incoming pixels per generated line and/or incoming lines per generated frame is monitored by software, compared to the nominal value and the internal clocks can be slowed down or accelerated to match the external D1 clock.
- Monitor fullness of the frame buffers and perform frame skip or repeat based on this information.

58.4 RGB to YCrCb color space conversion

RGB values coped within the output stage are actually what is sometimes called R’G’B’ gamma-corrected values. It is assumed that all the video signals coming into the chip are already gamma-corrected. So the following equations do not perform gamma correction. Gamma correction could be necessary in the STi7710 only for non-precorrected graphics (downloaded from the web for example), in this case the gamma correction would be done in the graphic engine (see the graphic specification).

Three equations are implemented:

- RGB to ITU-R BT 601 YCbCr
- RGB to ITU-R BT 709 YCbCr
- RGB to SMPTE 240M YCbCr

Selection is done through configuration bits.

58.5 Digital output formatter

This block 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 SMPTE274M, SMPTE295M, SMPTE293M,
- YCbCr, 4: 2: 2 on 8 bits (8MSBs of interface) with or without embedded EAV/SAV sync: ITU-R 656.

58.6 Analog video output

Video output can be

- full-range RGB output,
- full-range YPbPr output,
- EIA-770.2 compatible YPbPr with embedded bi-level sync, or
- 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 three raster-scanning systems:

- (1920 x 1080i) active picture in a (2200 x 1125i) frame
- (1280 x 720p) active picture in a (1650 x 750p) frame.
- (1920 x 1080i) active picture in a (2200 x 1250i) frame.

In these cases, the maximum PIXCLK frequency is 74.25 MHz.

- (720 x 483p) active picture in a (858 x 525p) frame.

In this case, the maximum PIXCLK frequency is 27 MHz.

With adequate programming of VOS configuration registers, the following display standards can be complied with:
SMPTE274M (1125i), SMPTE293M (525p), SMPTE295M (1250i), SMPTE296M (750p), ARIB-BS4 (525p, 750p, 1125i).

A 770.3 and 770.2 formatter rescales the video data and inserts synchronization pulses before digital to analog conversion, so that the analog waveform conforms to the selected standard:

In 480p, the formatter is able to apply Macrovision protection on the video flow if required.

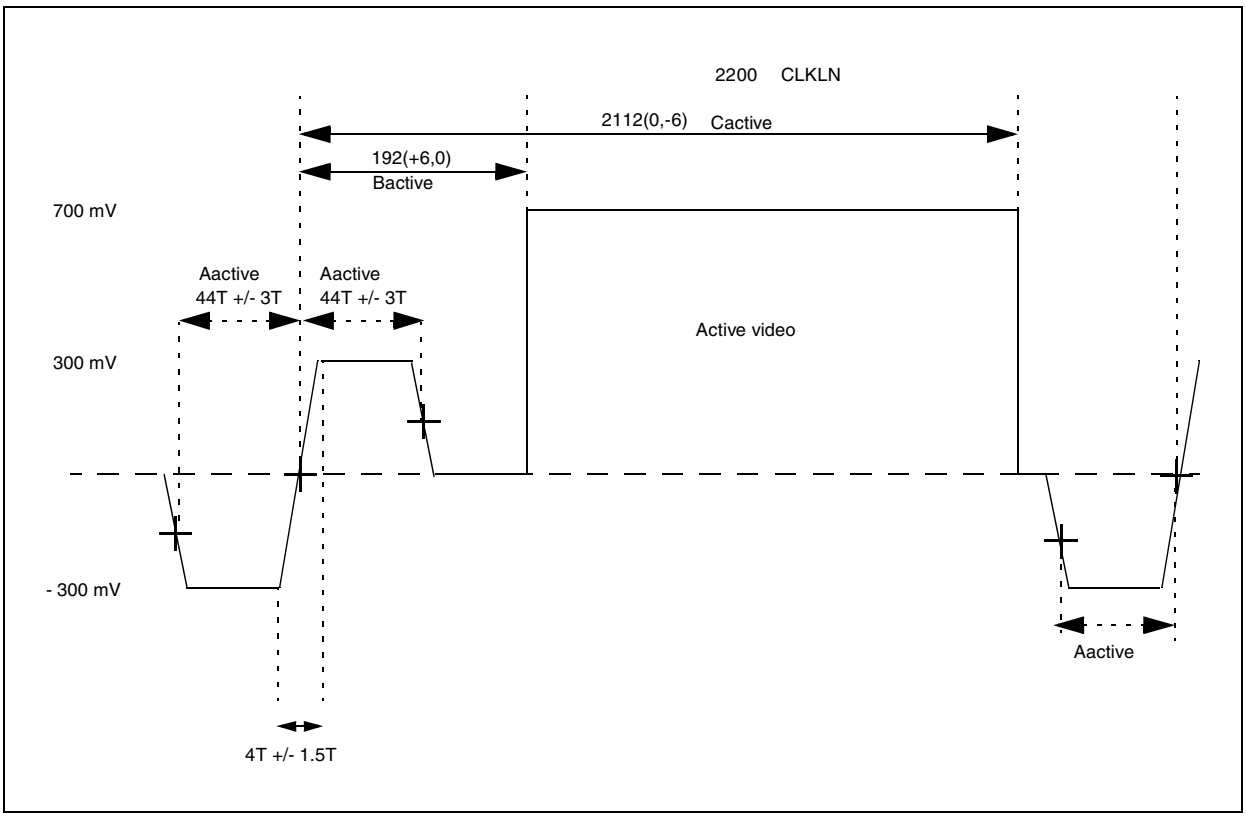
There is flexibility in the programmable configuration parameters to adapt to slight deviations from these standards.

58.6.1 YPbPr compliancy - HD formats

58.6.1.1 Horizontal timing

Analog synchronization (1080i)

Figure 151: Y' analog waveform



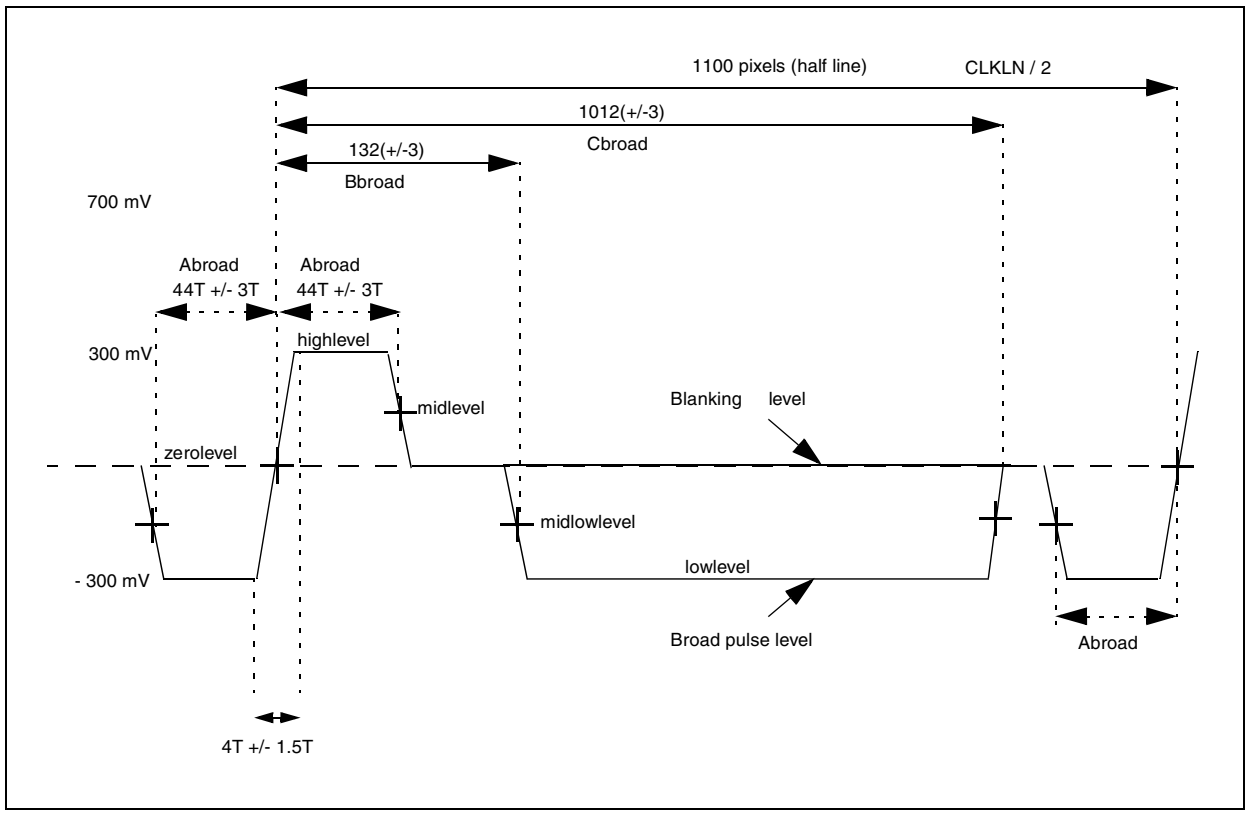
The P'b, P'r analog waveforms look the same, except for the voltage excursion (+/-350 mV).

Horizontally, to handle several raster scanning systems, some values have to be programmed: *Aactive*, *Bactive*, *Cactive* and *CLKLN*, which is already available in the VTG.

Extra-sync pulse

In the interlaced case, 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.

Figure 152: 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).

58.6.1.2 Vertical timing

As indicated above, 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 153: SMPTE 274M AND EIA 770.3 vertical timings

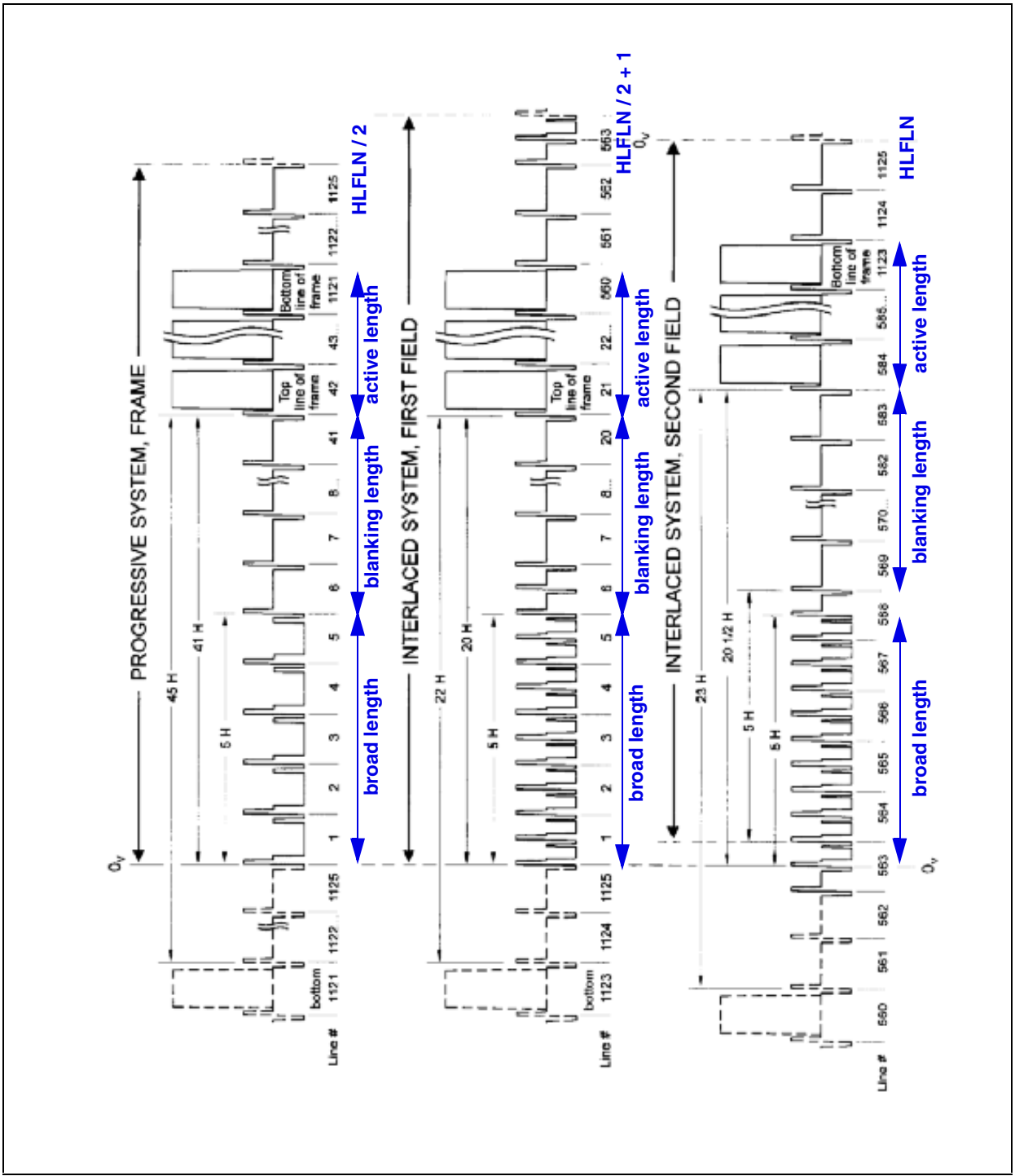
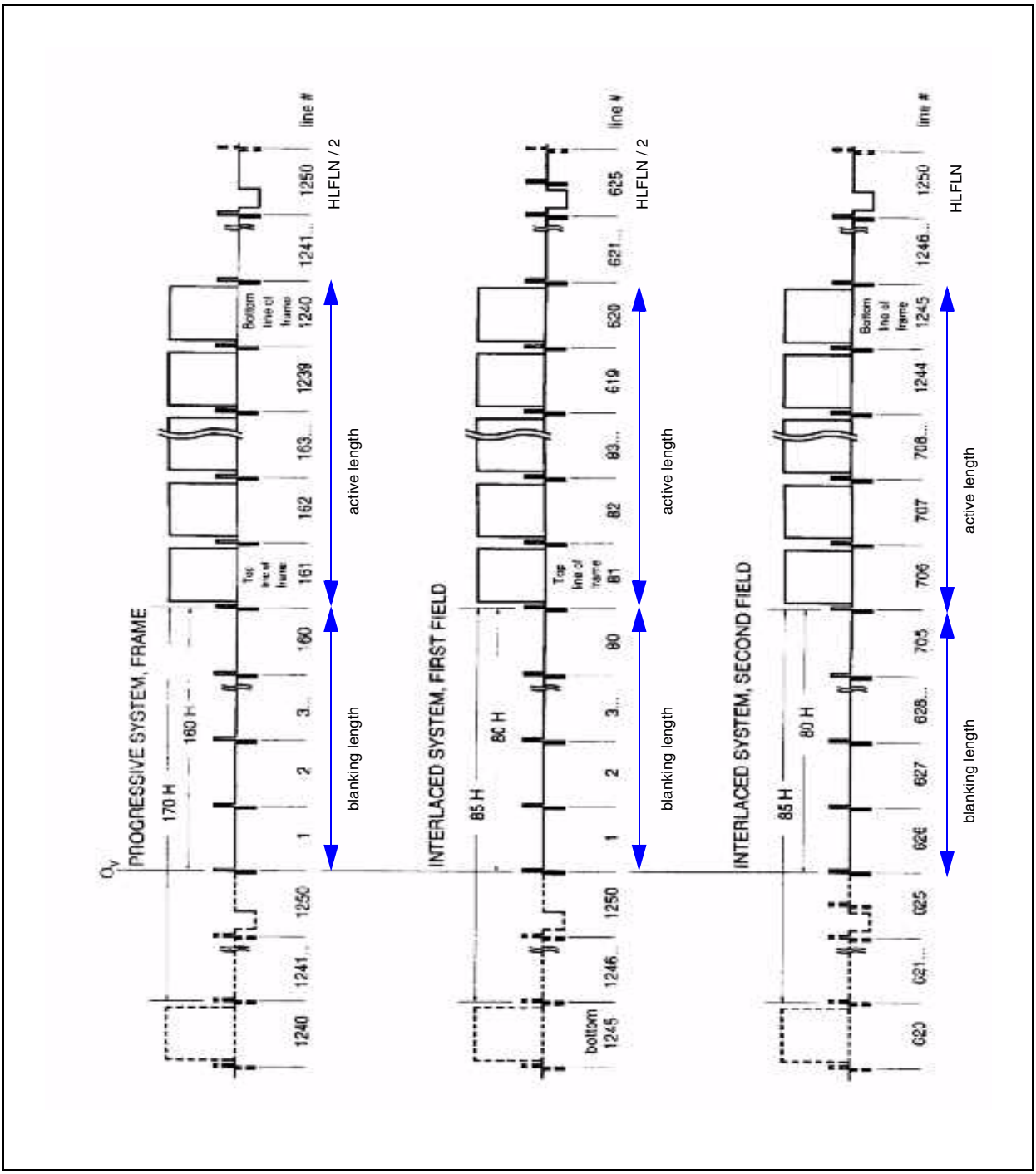


Figure 154: SMPTE295M vertical timings



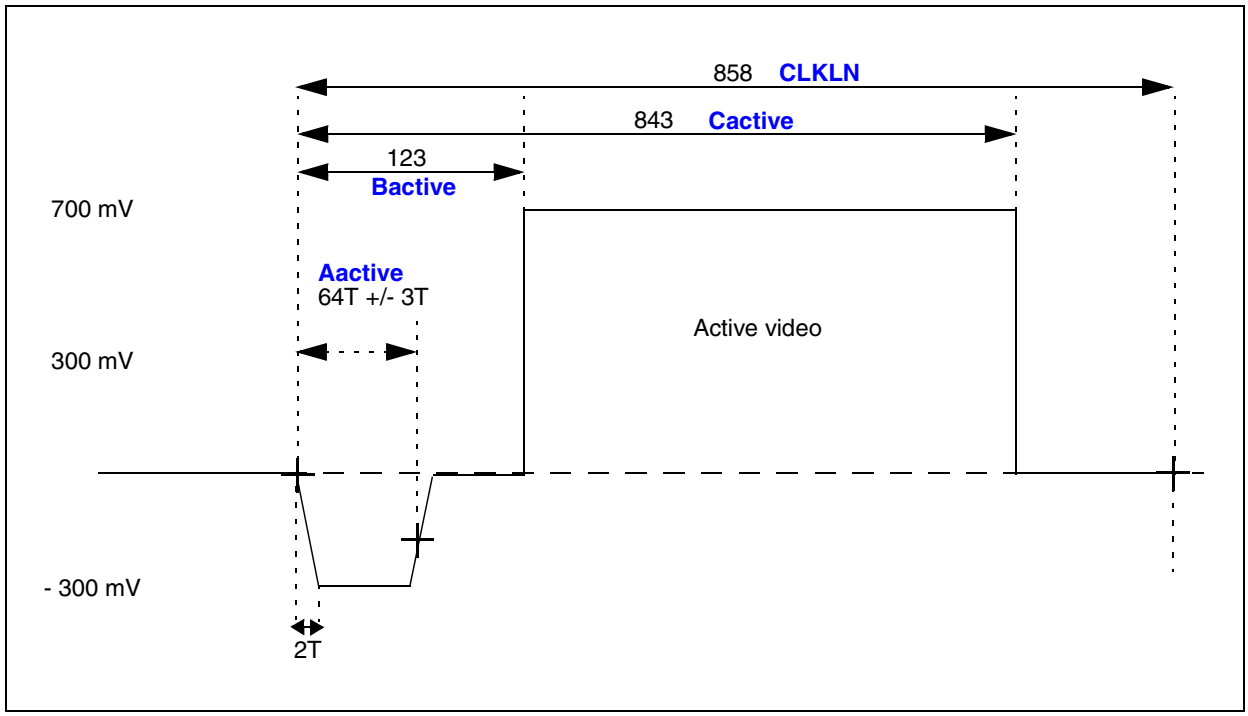
SMPTE295M is a special case because the number of lines is even for both the interlaced and progressive systems. Moreover, there are only two special lines in interlaced and one in progressive. All other lines are either blank lines or active lines. Therefore broadlength must be 0.

58.6.2 YPbPr compliancy - SD/ED formats

58.6.2.1 Horizontal timing

SMPTE293M is a bi-level sync pulse standard. For the control of this raster-scan system no new register is implemented. The registers used for EIA 770.3, SMPTE274M and SMPTE295M are used but not exactly in the same way: take care in their use.

Figure 155: Y' analog waveform



The P'b, P'r analog waveforms look the same, except for the voltage excursion (+/- 350 mV).

Horizontally, some values have to be programmed: Aactive, Bactive, Cactive and CLKLN, which is already available in the VTG.

Figure 156: Vertical timing relating to analog waveform for broad lines

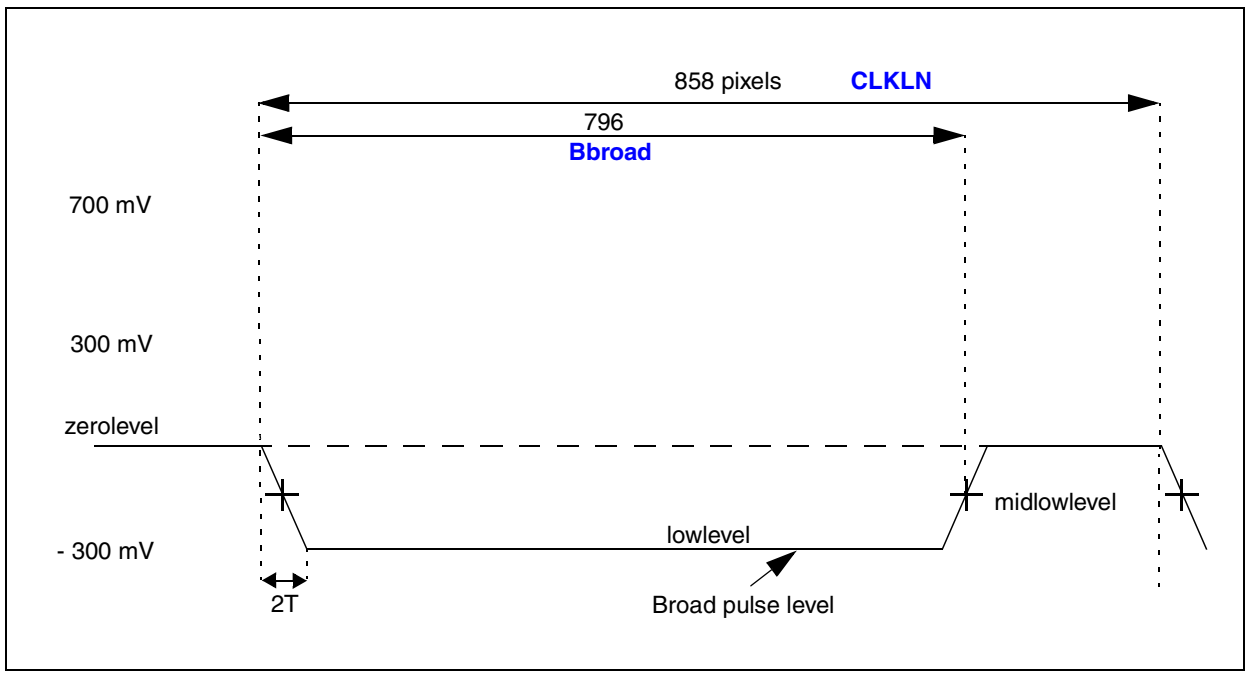
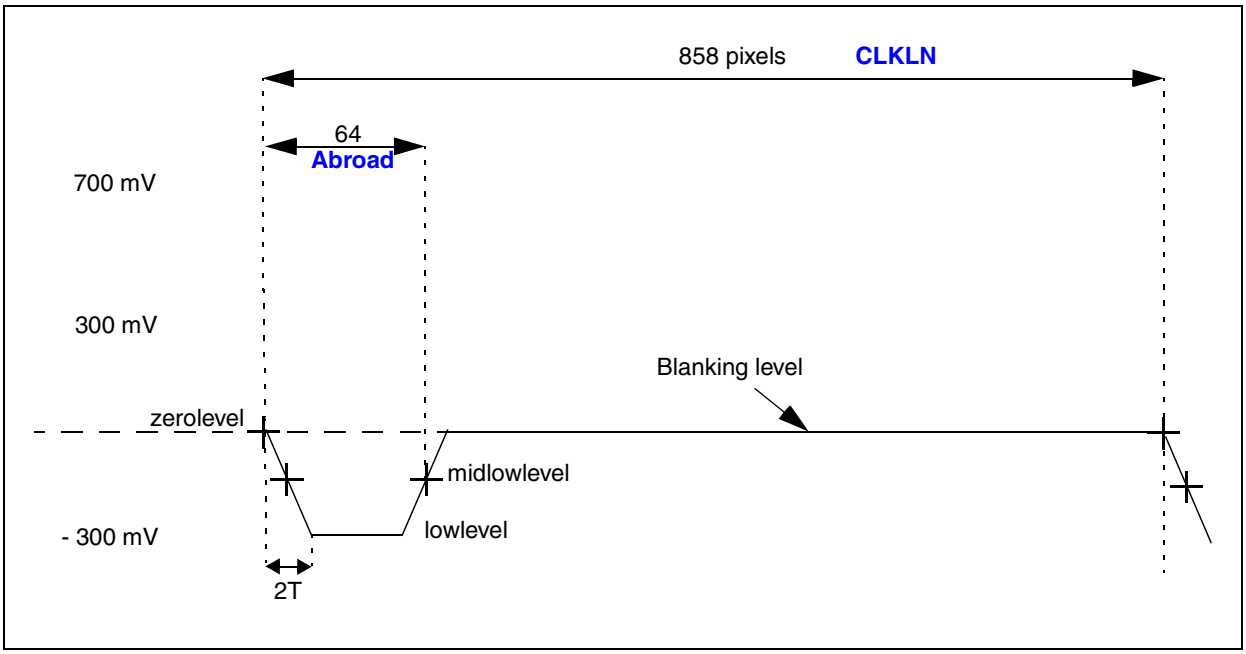


Figure 157: Vertical timing relating to analog waveform for blanking lines



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 only requires the values *Abroad* and *CLKLN* to be programmed. The broad waveform only requires the values *Bbroad* and *CLKN* to be programmed. The value *Cbroad* is not used in the case of SMPTE293M standard.

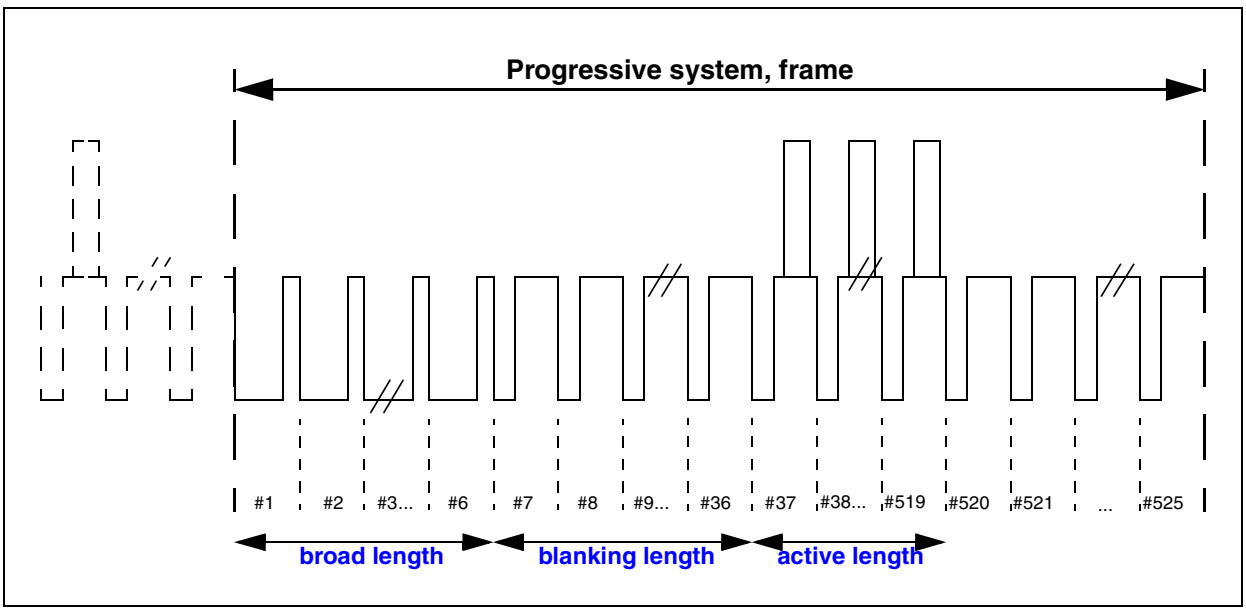
The synchronization level is also programmable (three registers for luma and three for chroma: ZEROLEVEL, MIDLOWLEVEL and LOWLEVEL).

58.6.2.2 Vertical timing

The SMPTE293M standard describes a progressive system.

As indicated above, 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 158: 525p frame waveform



58.7 Upsampler

The function of the upsampler is to adapt the rate of incoming pixels to the HD DAC clock rate. In a typical application (watch and VCR record), pixel rate is defined by the clock used by the main display path (CLK_DISP_HD) and upsampling is done by a factor of either 2 or 4. In an alternate application where all outputs are SD, the upsampler gets its data from the DENC at 27 MHz and upsamples by factor of 4. In the latter application, the DENC itself has already performed x2 upsampling on the original pixels 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.

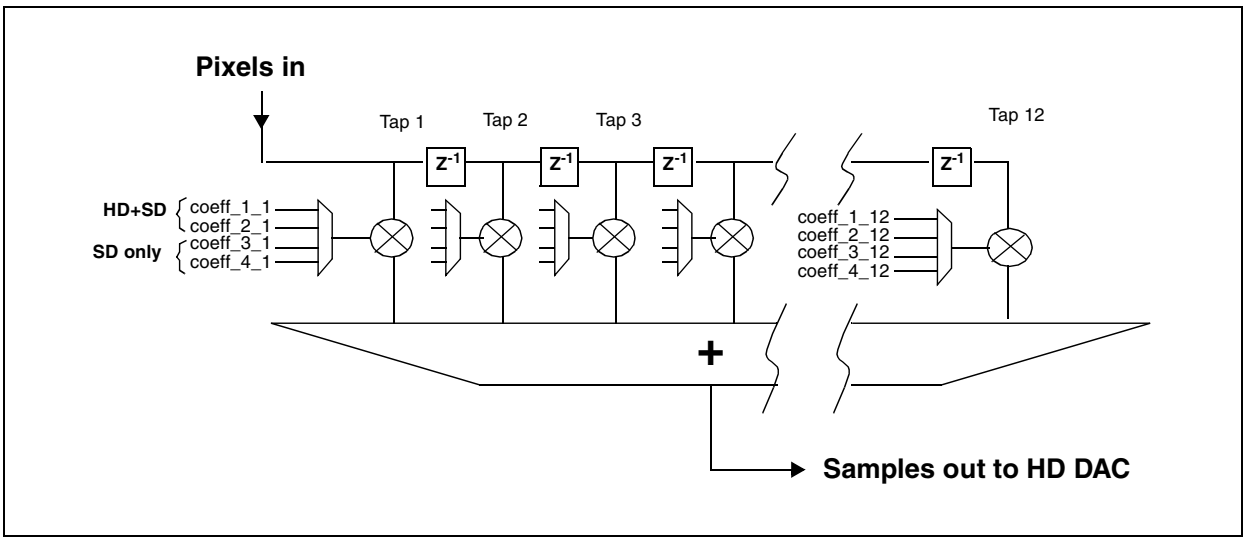
Coefficients will typically be chosen to support one of the 2 following modes:

- high quality mode: the upsampling maintains maximum bandwidth of the incoming video stream; includes adequate sin*x*/*x* compensation to cope with DAC behavior and to fulfill the requirements of supported HD analog standards.
- low pass mode: upsampling includes a 1/4 low pass filter effect on the incoming stream; this supports ‘constrained output’ mode whereby, whilst HDMI produces full resolution encrypted digital video, DAC analog outputs are available in a somewhat 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 configuration registers.

Figure 159: x2 (HD) or x4 (SD) SRC filter structure



59 Video output stage (VOS) registers

Addresses are provided as *VOSBaseAddress* + offset.

The *VOSBaseAddress* is:

0x2010 3000.

Table 147: 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 - Vsync output rising edge	0x0008	R/W
VTG1_HDS	VTG1 - Hsync output falling edge	0x000C	R/W
VTG1_HLFLN	VTG1 - Half line per vertical	0x0010	R/W
VTG1_VDO	VTG1 - Vsync output rising edge	0x0014	R/W
VTG1_VDS	VTG1 - Vsync output falling edge	0x0018	R/W
VTG1_MODE	VTG1 - Mode of operation	0x001C	R/W
VTG1_VTIMER	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_HDRIVE	VTG2 - Hsync, Vsync output enable	0x0034	R/W
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
VTG2_MODE	VTG2 - Mode of operation	0x0050	R/W
VTG2_VTIMER	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

Table 147: VOS register summary

Register	Description	Offset	Type
General VOS configuration			
DSPCFG_CLK	DSPCFG - Display and Global config	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_HIGHL	DHDO - High Level values	0x00A8	R/W
DHDO_MIDLOWL	DHDO - Mid low level values	0x00AC	R/W
DHDO_LOWL	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
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			
Reserved		0x00E0	-

Table 147: VOS register summary

Register	Description	Offset	Type
Upsampler tap coefficients			
COEFF_SET1_1	COEFF - Set 1.1	0x00E4	R/W
COEFF_SET1_2	COEFF - Set 1.2	0x00E8	R/W
COEFF_SET1_3	COEFF - Set 1.3	0x00EC	R/W
COEFF_SET1_4	COEFF - Set 1.4	0x00F0	R/W
COEFF_SET2_1	COEFF - Set 2.1	0x00F4	R/W
COEFF_SET2_2	COEFF - Set 2.2	0x00F8	R/W
COEFF_SET2_3	COEFF - Set 2.3	0x00FC	R/W
COEFF_SET2_4	COEFF - Set 2.4	0x0100	R/W
COEFF_SET3_1	COEFF - Set 3.1	0x0104	R/W
COEFF_SET3_2	COEFF - Set 3.2	0x0108	R/W
COEFF_SET3_3	COEFF - Set 3.3	0x010C	R/W
COEFF_SET3_4	COEFF - Set 3.4	0x0110	R/W
COEFF_SET4_1	COEFF - Set 4.1	0x0114	R/W
COEFF_SET4_2	COEFF - Set 4.2	0x0118	R/W
COEFF_SET4_3	COEFF - Set 4.3	0x011C	R/W
COEFF_SET4_4	COEFF - Set 4.4	0x0120	R/W
HDMI glue logic			
DLL_LOCK	HDMI DLL lock	0x0124	RO

59.1 VTG 1

VTG1_HDRIVE HSync Vsync output 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																														HSD	VSD

Address: 0x0000
Type: R/W
Reset: 0
Description: This register controls the output of HSYNC and Vsync on PAD.

[31:2] **Reserved**

[1] **VSD**: Vsync enable

0: Vsync from VTG1 is not output

1: Vsync from VTG1 is output

[0] **HSD**: Hsync enable

0: Hsync from VTG1 is not output

1: Hsync from VTG1 is output

Note: VOS has 2 VTGs (VTG1 and VTG2), and each VTG has his own set of registers.The two VTGs use the same output signal for Vsync and Hsync. If both VTGs want to output Hsync or Vsync, VTG1 is selected. If only one VTG is involved, it outputs the signal.

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: 0x0004
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **CLKLN**
Total number of pixel clocks per horizontal period (horizontal complete line length).

VTG1_HDO Hsync output 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: 0x0008
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **HDO**
Determines the rising pixel position of the Hsync signal pin output pulse relative to Href, when VTG1_HDRIVE.HSD is set. 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 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: 0x000C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **HDS**
Determines the falling pixel position of the Hsync signal pin output pulse relative to Href, when VTG1_HDRIVE.HSD is set. 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_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: 0x001D
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[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 VTG1_VTGMOD.FI in case of the 1250 line interlaced standard (SMPTE295M).

VTG1_VDO Vsync output 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																					VDO										

Address: 0x0014
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:11] **Reserved**

[10:0] **VDO**
determines the rising pixel position of the Vsync signal pin output pulse relative to Vref, when VTG_1HDRIVE.VSD is set. 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 determine the polarity of the Hsync signal pin.*
2. To get Vsync and Hsync rising on the same clock edge on the line corresponding to VTG1_VDO, VTG1_HDO must be assigned to 0

VTG1_VDS 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																					VDS										

Address: 0x0018
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:11] **Reserved**

[10:0] **VDS**
Determines the falling pixel position of the Hsync signal pin output pulse relative to Href, when VTG_1HDRIVE.VSD is set. The Hsync 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_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	Reserved	

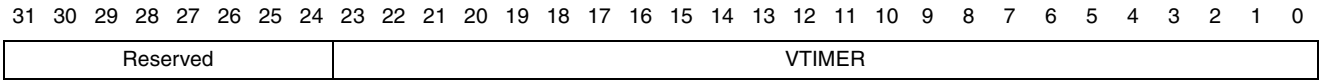
Address: 0x001C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: This register controls the operating mode of the VTG1.

[31:3] **Reserved**

[2] **FI**: Force interleaved
This bit, when set, allows VTG1 to be in interlaced mode when VTG1_HLFLN is even (typically for SMPTE295M when line number is 1250).
0: No force.
1: Force interleaved

[1:0] **Reserved**

VTG1_VTIMER Vsync delayed interrupt timer

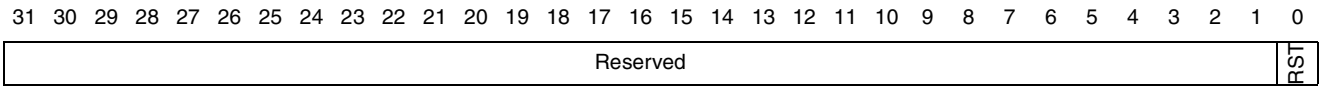


Address: 0x0020
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[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

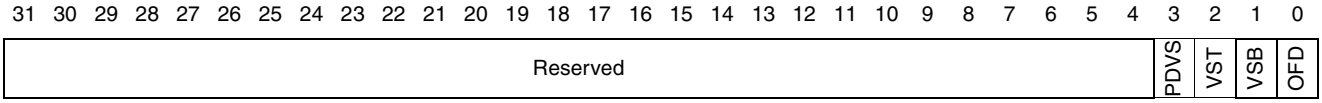


Address: 0x0024
Type: WO
Reset: 000
Description:

[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 generated Vref and Href. In case of interlaced raster, a top field is generated first.

VTG1_ITM Interrupt mask



Address: 0x0028

Type: R/W

Reset: 0

Description: Any bit set in this register enables the corresponding interrupt in 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**

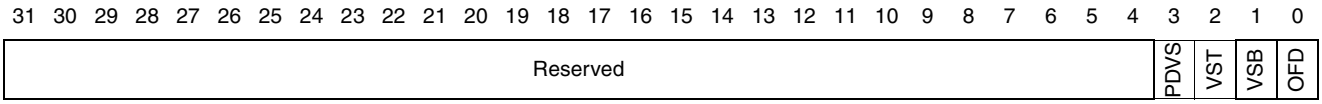
[3] **PDVS**: Programmable delay on VSync
This bit is set for a short time after VSync, with a delay programmed in VTG1_VTIMER register.

[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 ‘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 ‘bottom not top’ signal.

[0] **OFD**: Output field: (only relevant in register VTG1_STA)
0: bottom filed.
1: top filled.

VTG1_ITS Interrupt Status



Address: 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 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 ‘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 ‘bottom not top’ signal.

[0] **OFD**: Output field: (only relevant in register VTG1_STA)
0: bottom filed.
1: top filled.

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: 0x0030

Type: RO

Reset: 0

Description: This register contains the video timing generator status. Any change from 0 to 1 of any of this 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 ever to be 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 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 ‘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 ‘bottom not top’ signal.

[0] **OFD**: Output field: (only relevant in register VTG1_STA)
0: bottom filed.
1: top filled.

59.2 VTG 2

VTG2_HDRIVE HSync Vsync output 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																														HSD	VSD

Address: 0x0034

Type: R/W

Reset: 0

Description: This register controls the output of HSYNC and Vsync on PAD.

Note: VOS has 2 VTGs (VTG1 and VTG2), and each VTG has his own set of registers.The two VTGs use the same output signal for Vsync and Hsync. If both VTGs want to output Hsync or Vsync, VTG1 is selected. If only one VTG is involved, it outputs the signal.

[31:4] **Reserved**

[1] **VSD**: Vsync enable
0: Vsync from VTG2 is not output
1: Vsync from VTG2 is output

[0] **HSD**: Hsync enable
0: Hsync from VTG2 is not output
1: Hsync from VTG2 is output

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: 0x0038
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **CLKLN**
Specifies the total number of pixel clocks per horizontal period (horizontal complete line length).

VTG2_HDO Hsync output 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: 0x003C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **HDO**
Determines the rising pixel position of the Hsync signal pin output pulse relative to Href, when VTG2_HDRIVE.HSD is set. The Hsync output rises from 0 to 1 level when the pixel counter is equal to VTG2_HDO.

Note: The order of VTG2_HDO and VTG2_HDS determine the polarity of the Hsync signal pin.

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: 0x0040
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **HDS**
Determines the following pixel position of the Hsync signal pin output pulse relative to Href, when VTG2_HDRIVE.HSD is set. The Hsync output rises from 0 to 1 level when the pixel counter is equal to VTG2_HDO.

Note: The order of VTG2_HDO and VTG2_HDS determine the polarity of the Hsync signal pin.

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: 0x0044
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[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 (SMPTE295M).

VTG2_VDO Vsync output 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																					VDO										

Address: 0x0048
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:11] **Reserved**

[10:0] **VDO**
Determines the rising pixel position of the Vsync signal pin output pulse relative to Vref, when VTG2_HDRIVE.VSD is set. The Vsync output rises from 0 to 1 level when the half line counter is equal to VTG2_VDO.

Note: 1 The order of VTG2_VDO and VTG2_VDS determine the polarity of the Hsync signal pin.

2 To get Vsync and Hsync rising on the same clock edge on the line corresponding to VTG2_VDO, VTG2_HDO must be assigned to 0

VTG2_VDS 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																					VDS										

Address: 0x004C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:11] **Reserved**

[10:0] **VDS**
Determines the following pixel position of the Hsync signal pin output pulse relative to Href, when VTG2_HDRIVE.VSD is set. The Hsync output 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_VTGMODE 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: 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 SMPTE295M when line number is 1250).
0: No force.
1: Force interleaved

[1:0] **SMD**: Slave mode selection
00: VTG2 is master (internally generate H and V Sync).
01: VTG2 is slave (using H and V Sync from VTG1).
10: VTG2 is slave (using Vsync from VTG1 and generating internal Hsync).

VTG2_VTIMER 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: 0x0054
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:24] **Reserved**

[23:0] **VTIMER**
Holds the 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
																															RST

Address: 0x0058
Type: WO
Reset: 000
Description:

[31:1] **Reserved**

[0] **RST**: Reset
Resets the pixel counters and line counters in the raster generator. This reset is just activated once on writing. When activated, VTG2 reset also generated Vref and Href. In case of interlaced raster, a top field is generated first.

VTG2_R1 Range 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																				RG1											

Address: 0x005C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description:

[31:12] **Reserved**

[11:0] **RG1**
Holds the lower limit of pixel count range of detection of a bottom field. In slave mode, if VTG1 VSync is detected in this range, VTG2 generate a top bottom field. If VTG1 Vsync is detected out of this range, a top field is generated

VTG2_R2 Range 2

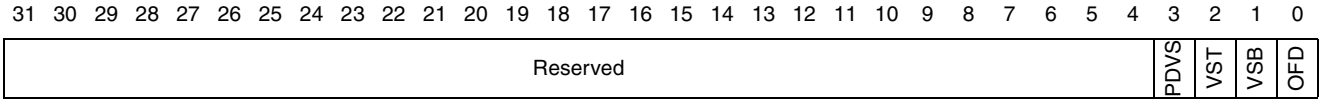
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				RG2											

Address: 0x0060
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 0
Description: The register holds the lower limit of pixel count range of detection of a bottom field. In slave mode, if VTG1 VSync is detected in this range, VTG2 generate a top bottom field. If VTG1 Vsync is detected out of this range, a top field is generated.

[31:12] **Reserved**

[11:0] **RG2**
Holds the lower limit of pixel count range of detection of a bottom field. In slave mode, if VTG1 VSync is detected in this range, VTG2 generate a top bottom field. If VTG1 Vsync is detected out of this range, a top field is generated

VTG2_ITM Interrupt mask



Address: 0x0064

Type: R/W

Reset: 0

Description: Any bit set in this register enables the corresponding interrupt in 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**

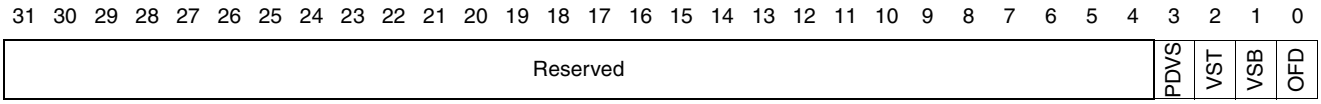
[3] **PDVS**: Programmable delay on VSync.
This bit is set for a short time after VSync, with a delay programmed in VTG1_VTIMER register.

[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 ‘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 ‘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: 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**

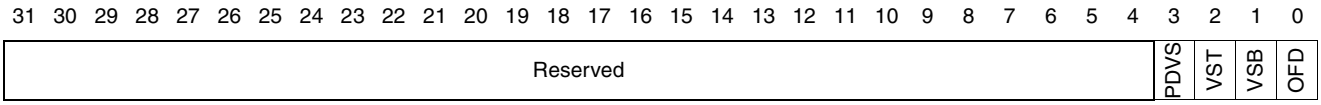
[3] **PDVS**: Programmable delay on VSync
This bit is set for a short time after VSync, with a delay programmed in VTG2_VTIMER register.

[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 ‘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 ‘bottom not top’ signal.

[0] **OFD**: Output field: (only relevant in register VTG2_STA)
0: bottom filed.
1: top filled.

VTG2_STA VTG2 Status



Address: 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 ever to be 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.

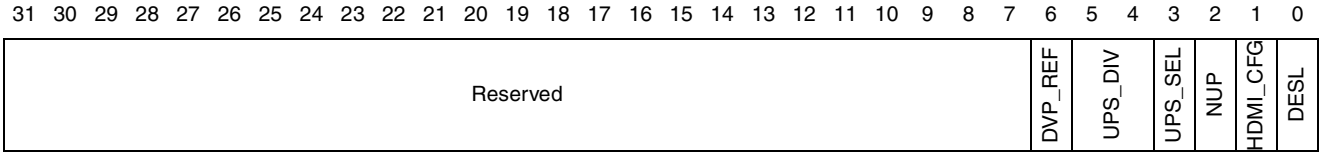
[2] **VST**: Vsync top
This bit is set for a short time at the beginning of the top field, corresponding to the falling edge of the internal ‘bottom not top’ signal.

[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 ‘bottom not top’ signal.

[0] **OFD**: Output field: (only relevant in register VTG2_STA)
0: bottom filed.
1: top filled.

59.3 General VOS configuration

DSPCFG_CLK Display and global configuration



Address: 0x0070

Type: R/W

Reset: 000

Description: This register configures global use of display processors.

[31:7] **Reserved**

[6] **DVP_REF**: Reference to DVP
0: Main Href and Vref (from VTG1) are transmit to DVP
1: Aux Href and Vref (from VTG2) are transmit 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

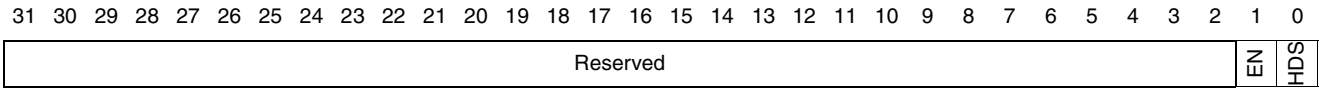
[3] **UPS_SEL**: Upsampler Mode
0: x2 (HD Mode)
1: x4 (SD Mode)

[2] **NUP**: Upsampler enable
0: Upsampler is bypassed.
1: Upsampler is enabled.

[1] **HDMI_CFG**: HDMI config
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

DSPCFG_DIG Display digital output configuration



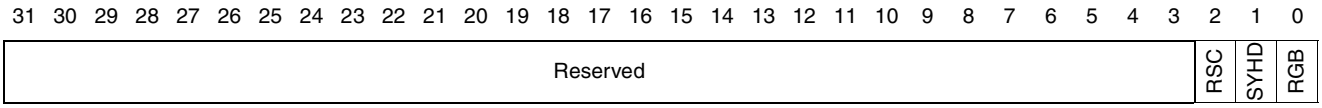
Address: 0x0074
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register configures the digital display output.

[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

DSPCFG_ANA Main analog display output configuration



Address: 0x0078
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register configures the main analog display output.

[31:7] **Reserved**

[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

59.4 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: 0x007C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Note: This register configures the main analog display output

[31:7] **Reserved**

[2] P293: Progressive SMPTE293M
0: Any standard other than progressive SMPTE293M (including interlaced 293M).
1: Progressive SMPTE293M Standard.

[1] P295: Progressive SMPTE295M
0: Any standard other than progressive SMPTE295M (including interlaced 295M).
1: Progressive SMPTE295M 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: 0x0080
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:12] **Reserved**

[11:0] **ABROAD**
Holds the length of high level 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: 0x0084
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:12] **Reserved**

[11:0] **BBROAD**
Holds the length of high level and low level synchronization 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: 0x0088
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Note: This register

[31:12] **Reserved**

[11:0] **CBROAD**
Holds the length of high level 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: 0x008C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:12] **Reserved**

[11:0] **BACTIVE**
Holds the length of high level plus zero level synchronization during active video line, in pixel clock units.

Note: (CACTIVE - BACTIVE) is the number of horizontal active pixels per line

DHDO_CACTIVE CACTIVE 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																				CACTIVE											

Address: 0x0090
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:12] **Reserved**

[11:0] **CACTIVE**
Holds the 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

31	30	29	28	27	26	25	24	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																					BROADL										

Address: 0x0094
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:11] **Reserved**

[10:0] **BROADL**
Holds the number of broad pulse lines to be generated (in interlaced mode, these lines contain the pattern twice)

DHDO_BLANKL Blanking 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																					BLANKL										

Address: 0x0098
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register.

[31:11] **Reserved**

[10:0] **BLANKL**
Holds the 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: 0x009C
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:11] **Reserved**

[10:0] **ACTIVEL**
Holds the 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: 0x00A0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register holds the 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: 0x00A4
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register holds the 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_HIGHL 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: 0x00A8
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register holds the 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_MIDLOWL 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: 0x00AC
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register holds the 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_LOWL 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						LOWLC										Reserved						LOWLL									

Address: 0x00B0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register holds the 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: 0x00B4

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 000

Description: This register 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**: SMPTE240M
0: ITU-R 601 or 709 is selected.
1: SMPTE240M 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																															

Address: 0x00B8

Type: R/W

Buffer: Rebuffered on VSYNC

Reset: 000

Description:

[31:10] **Reserved**

[9:0] **YMLT**
multiplication factor applied to the luma or green signal (Y/G) before the digital to analog converter in order 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: 0x00BC
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:10] **Reserved**

[9:0] **CMLT**
Multiplication factor applied to the chroma or red and blue signal (R/Pr and B/Pb) before the digital to analog converter in order 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 + COFFDHDO_CMLT

DHDO_COFF CHROMA 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																						COFF									

Address: 0x00C0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:10] **Reserved**

[9:0] **COFF**
Offset applied to the chroma or red and blue signal (R/Pr and B/Pb) before the digital to analog converter in order 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

31	30	29	28	27	26	25	24	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																						YOFF									

Address: 0x00C4
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:10] **Reserved**

[9:0] **YOFF**
offset applied to the luma or green signal (Y/G) before the digital to analog converter in order to allow synchronization levels insertion.

Note: G/Y = (YMLT x internal signal G/Y)/1024 + YOFFC656_ACTL

59.5 Digital SD video out

C656_ACTL Active 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																						ACTL									

Address: 0x00C8
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:11] **Reserved**

[10:0] **ACTL**: 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: 0x00CC
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:11] **Reserved**

[10:0] **BACT**
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: 0x00D0
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:12] **Reserved**

[11:0] **BLANKL**
Number of blanking line 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: 0x00D4
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description:

[31:12] **Reserved**

[11:0] **EACT**
Duration in pixel clock between internal HREF and end of active video pixels.

C656_PAL 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																															PAL

Address: 0x00D8
Type: R/W
Buffer: Rebuffered on VSYNC
Reset: 000
Description: This register bit adjusts timing between PAL and NTSC standards.

[31:1] **Reserved**

[0] **PAL**: PAL/NTSC
0: NTSC is selected.
1: PAL is selected.

59.6 HD DAC configuration

DSPCFG_DAC HD DACs 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																									BLRGB	BLANK				SETUP	

Address: 0x00DC
Type: R/W
Reset: 000
Description: This register adjusts timing between PAL and NTSC standards.

[31:7] **Reserved**

[6] **BLRGB**: Blank on RGB configuration
0: Blanking is implemented according to BLANK.
1: Blanking is based on an internal blanking signal.

[5] **BLANK[2]**: Blank: (preventing video output program current)
0: Regular operation mode
1: Forces R/Cr to blank level depending on BLRGB field

[4] **BLANK[1]**: Blank: (preventing video output program current)
0: Regular operation mode
1: Forces B/Cb to blank level depending on BLRGB field

[3] **BLANK[0]**: Blank: (preventing video output program current)
0: Regular operation mode
1: Forces G/Y to blank level.

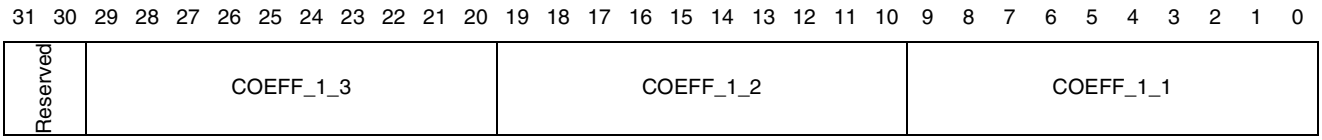
[2] **SETUP[2]**: Setup: Forces a shift on the output level
0: Regular operation mode
1: Forces a shift on the R/Cr output level

[1] **SETUP[1]**: Setup: Forces a shift on the output level
0: Regular operation mode
1: Forces a shift on the B/Cb output level

[0] **SETUP[0]**: Setup: Forces a shift on the output level
0: Correct operation mode
1: Forces a shift on the G/Y output level

59.7 Upsampler tap coefficients

COEFF_SET1_1 Coefficient Set 1 - coeff Tap 1, 2, 3



Address: 0x00E4
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

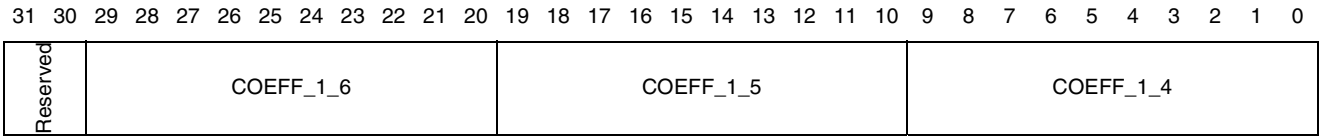
[31:30] **Reserved**

[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)

COEFF_SET1_2 Coefficient Set 1 - coeff Tap 4, 5, 6



Address: 0x00E8
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

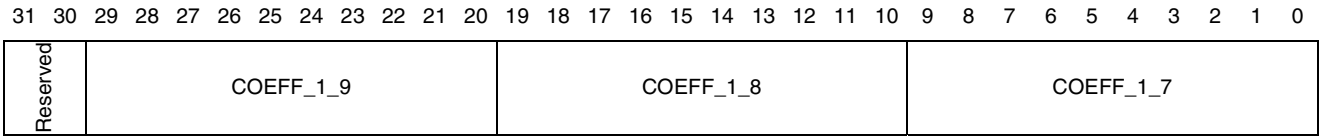
[31:30] **Reserved**

[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)

COEFF_SET1_3 Coefficient Set 1 - coeff Tap 7, 8, 9



Address: 0x00EC
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

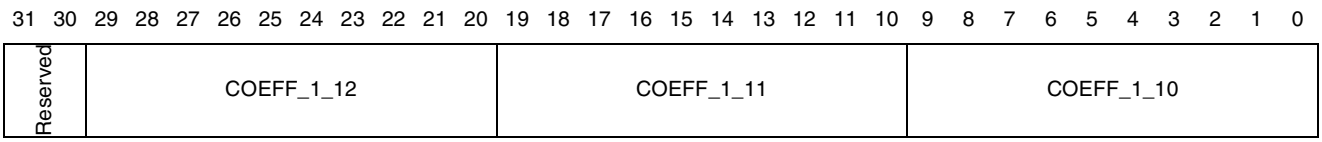
[31:30] **Reserved**

[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)

COEFF_SET1_4 Coefficient Set 1 - coeff Tap 10, 11, 12



Address: 0x00F0
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

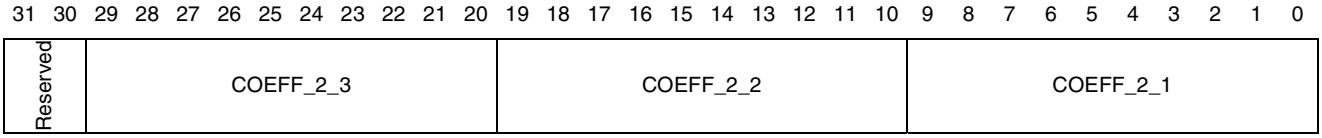
[31:30] **Reserved**

[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)

COEFF_SET2_1 Coefficient Set 2 - coeff Tap 1, 2, 3



Address: 0x00F4
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

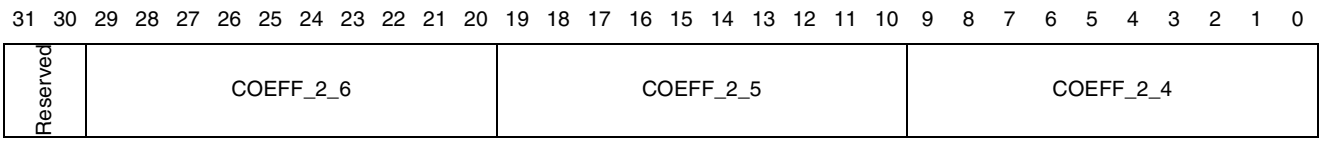
[31:30] **Reserved**

[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)

COEFF_SET2_2 Coefficient Set 2 - coeff Tap 4, 5, 6



Address: 0x00F8
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

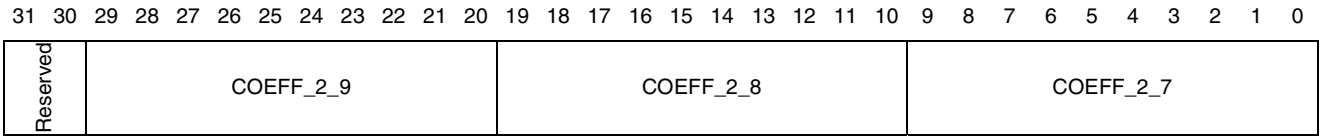
[31:30] **Reserved**

[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)

COEFF_SET2_3 Coefficient Set 2 - coeff Tap 7, 8, 9



Address: 0x00FC
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

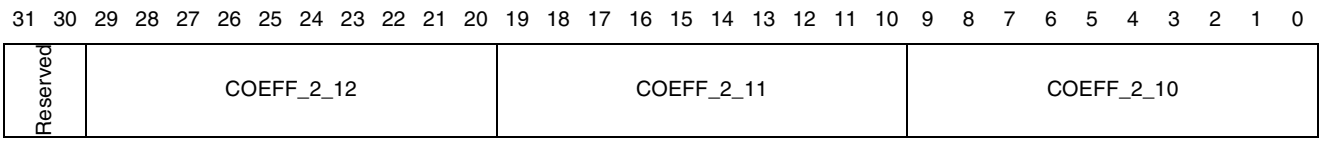
[31:30] **Reserved**

[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)

COEFF_SET2_4 Coefficient Set 2 - coeff Tap 10, 11, 12



Address: 0x0100
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (HD x2 or SD x4).

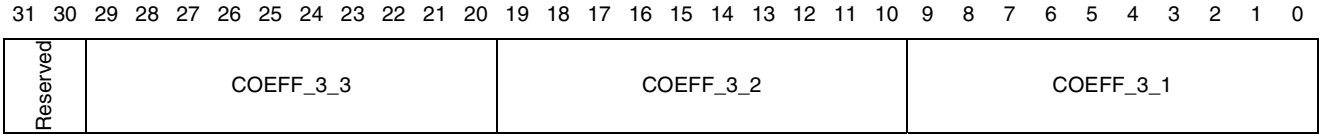
[31:30] **Reserved**

[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)

COEFF_SET3_1 Coefficient Set 3 - coeff Tap 1, 2, 3



Address: 0x0104
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

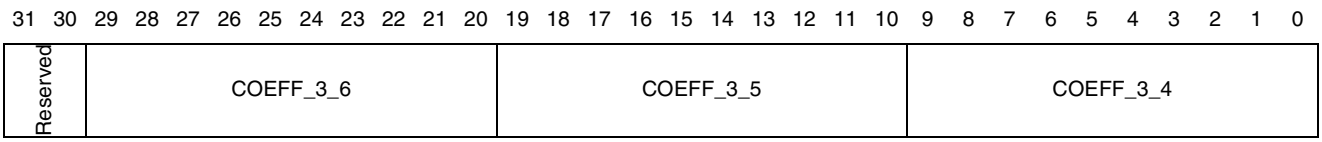
[31:30] **Reserved**

[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)

COEFF_SET3_2 Coefficient Set 3 - coeff Tap 4, 5, 6



Address: 0x0108
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

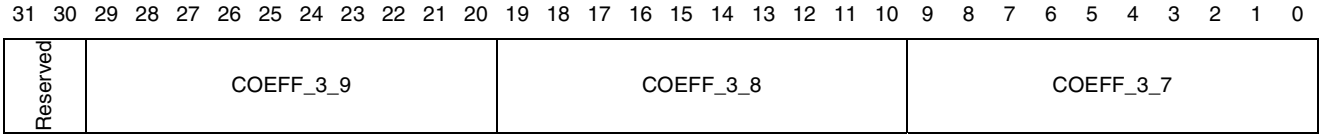
[31:30] **Reserved**

[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)

COEFF_SET3_3 Coefficient Set 3 - coeff Tap 7, 8, 9



Address: 0x010C
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

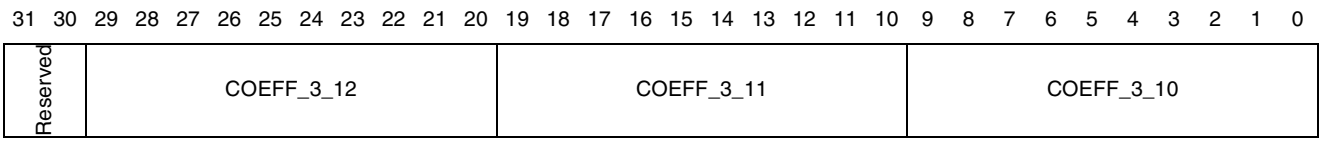
[31:30] **Reserved**

[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)

COEFF_SET3_4 Coefficient Set 3 - coeff Tap 10, 11, 12



Address: 0x0110
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

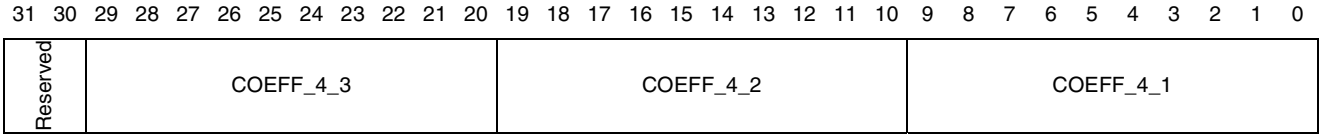
[31:30] **Reserved**

[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)

COEFF_SET4_1 Coefficient Set 4 - coeff Tap 1, 2, 3



Address: 0x0114
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

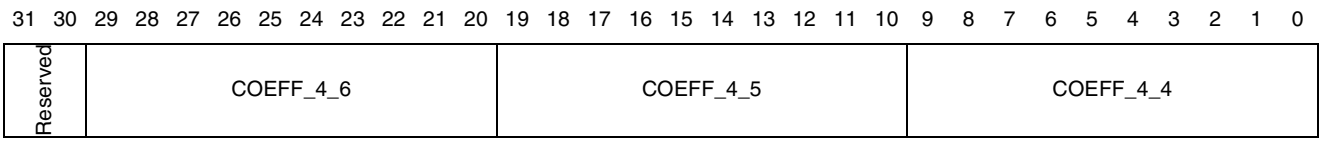
[31:30] **Reserved**

[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)

COEFF_SET4_2 Coefficient Set 1 - coeff Tap 4, 5, 6



Address: 0x0118
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

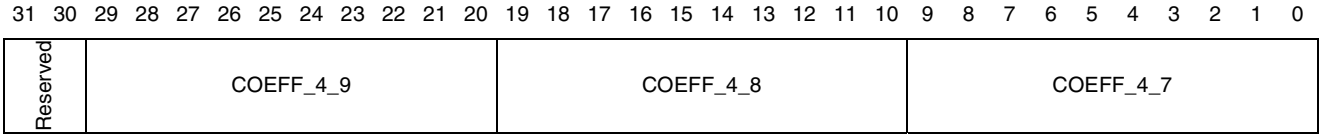
[31:30] **Reserved**

[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)

COEFF_SET4_3 Coefficient Set 4 - coeff Tap 7, 8, 9



Address: 0x011C
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

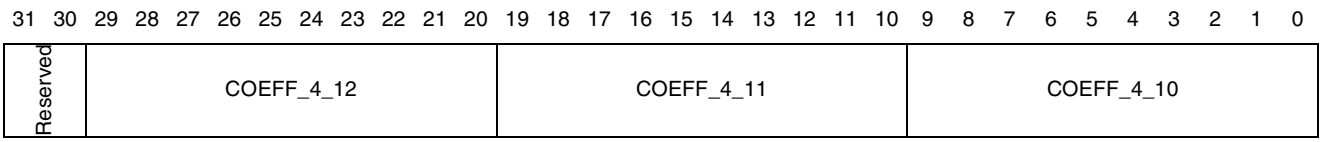
[31:30] **Reserved**

[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)

COEFF_SET4_4 Coefficient Set 4 - coeff Tap 10, 11, 12



Address: 0x0120
Type: R/W
Reset: 000
Description: This register contains coefficients of the SRC used for Upsampling (SD only).

[31:30] **Reserved**

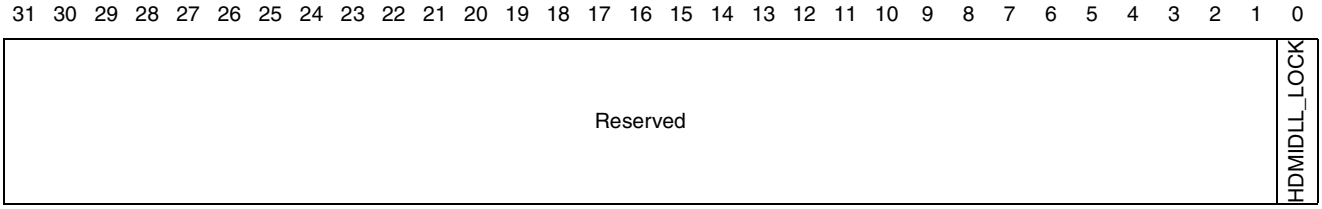
[29:20] **COEFF_4_12**
Coefficient applied to Tap 12 at fourth phase (for x4 upsampling)

[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)

59.8 HDMI glue logic

HDMI_PHY_LOCK HDMI PHY LOCK STATUS



Address: 0x0124
Type: RO
Reset: 000
Description:

[31:1] **Reserved**

[0] **HDMIDLL_LOCK**
Lock status of the HDMI PHY DLL: DLL is locked if reading this bit returns 1, else it is not locked.

60 Compositor

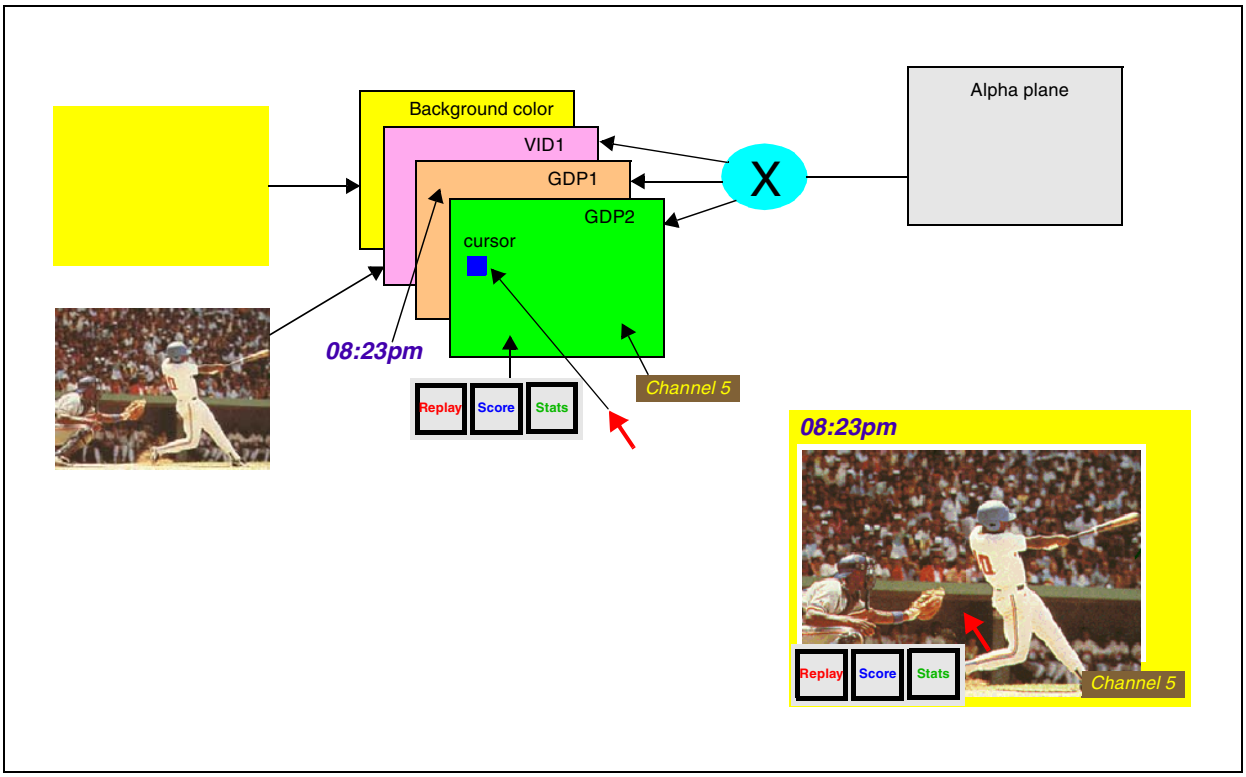
60.1 Overview

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 160). The output of mixer 2 (Figure 161) 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 STi7710's output stage.

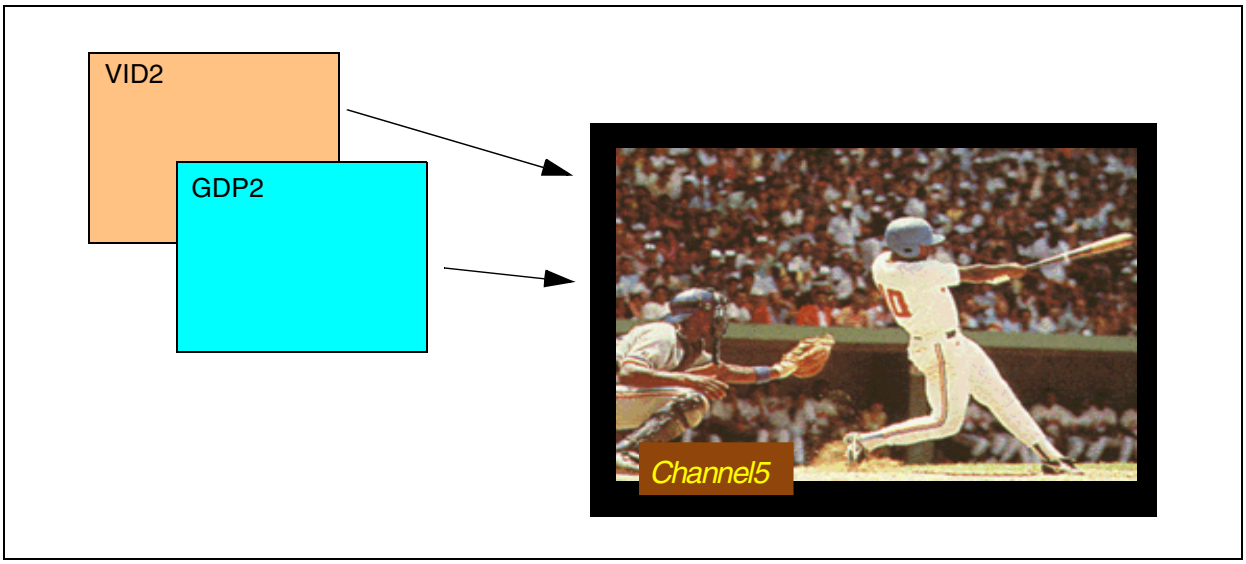
Figure 160: Mixer 1 plane



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 161: Mixer 2 planes



60.2 Compositor layout

Figure 162 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.

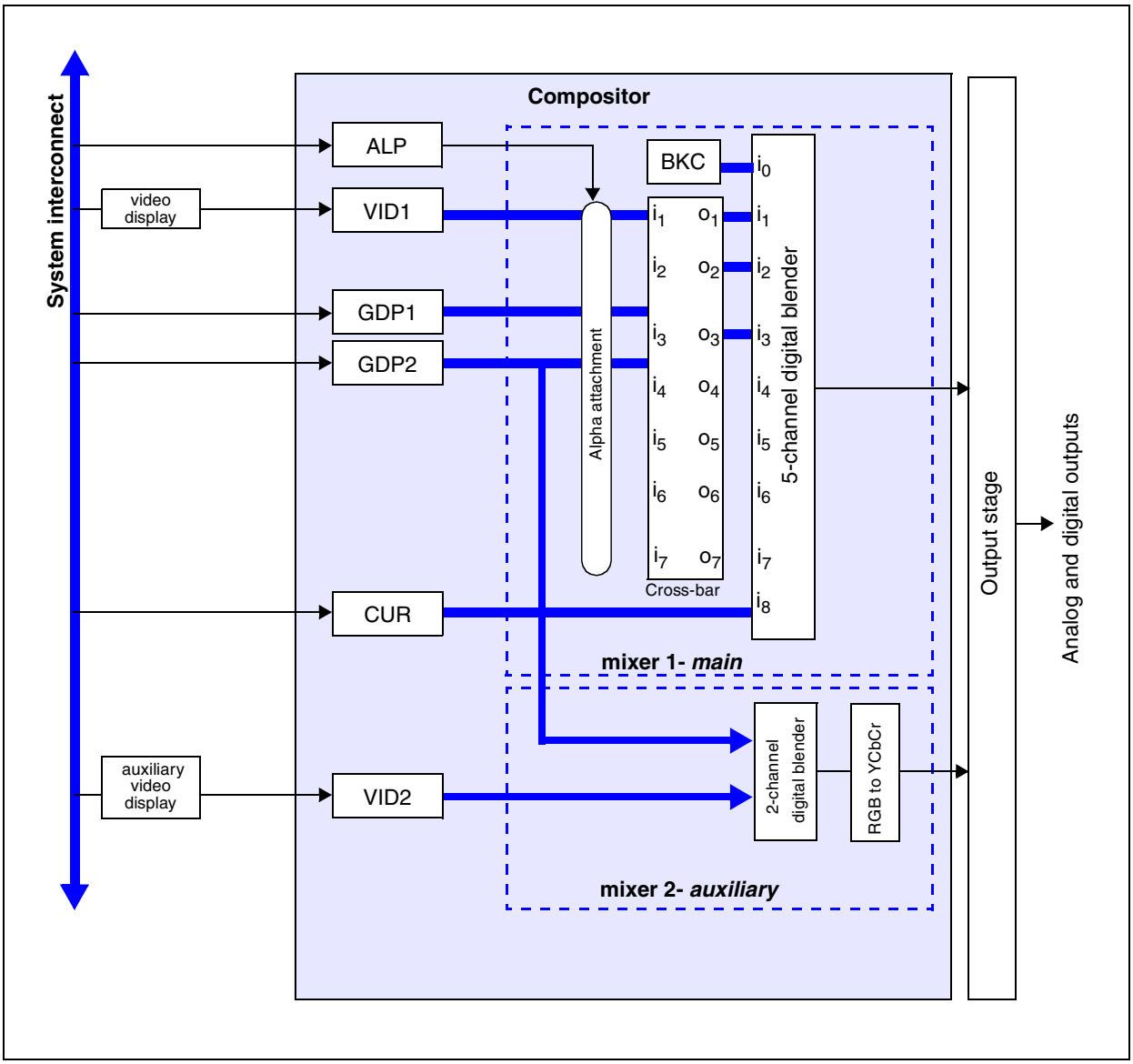
Digital mixer 1 successively blends video layers VID1, both graphics layers (GDP1 and GDP2), the cursor layer and a background color. A cross-bar router enables the hierarchy of the GDP1 and VID1 layers to be programmed. The resulting order is background color, GDP2, (VID1 and GDP1 in programmed order) 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 61 on page 620. Each plane block supports a specific set of bitmap formats. All bitmap formats are described in Section

56.8: *Local memory storage of supported graphics formats on page 508*. Each plane starts reading data from memory when it is enabled in mixer 1 or mixer 2.

Figure 162: Graphics and video block diagram



60.3 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_CTL.

Mixer 1 is composed of three independent sub-blocks: the blender, the cross-bar router, and the alpha plane attachment unit.

60.3.1 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.

60.3.2 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.

60.3.3 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.

60.4 Digital mixer 2 (MIX2) - auxiliary display output

Mixer 2 is a much simpler engine, for the main 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_CTL.

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.

60.5 Generic display pipelines (GDP1 and GDP2)

The generic display pipelines present the following features:

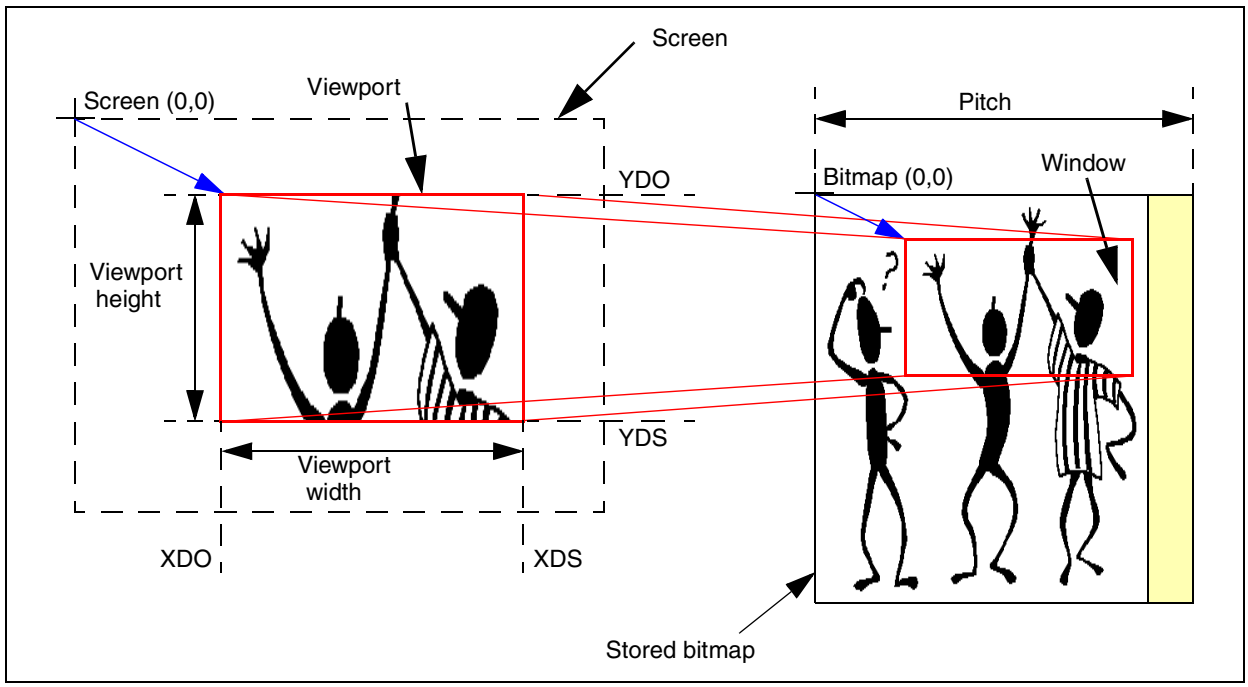
- Link-list-based display engine, for multiple viewport capabilities.
- ARGBargb formats including ARGB1555, ARGB4444, RGB565, RGB888, ARGB8565, ARGB8888.
- YCbCr4:2:2R, YCbCr888 and AYCbCr8888 format support.
- Premultiplied or non-premultiplied RGB component support.
- Little endian and big endian 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).
- 2-tap vertical sample rate converter for vertical resize. This can be used for half-resolution applications, that is rescaling factor 2x or more (rescaling factors of less than 2x are not recommended for interlace displays).
- 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 56.8: *Local memory storage of supported graphics formats* *on page 508*.

60.5.1 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 163: 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, and the vertical resizer parameters).

Figure 164: Typical display link-list configuration for a progressive display

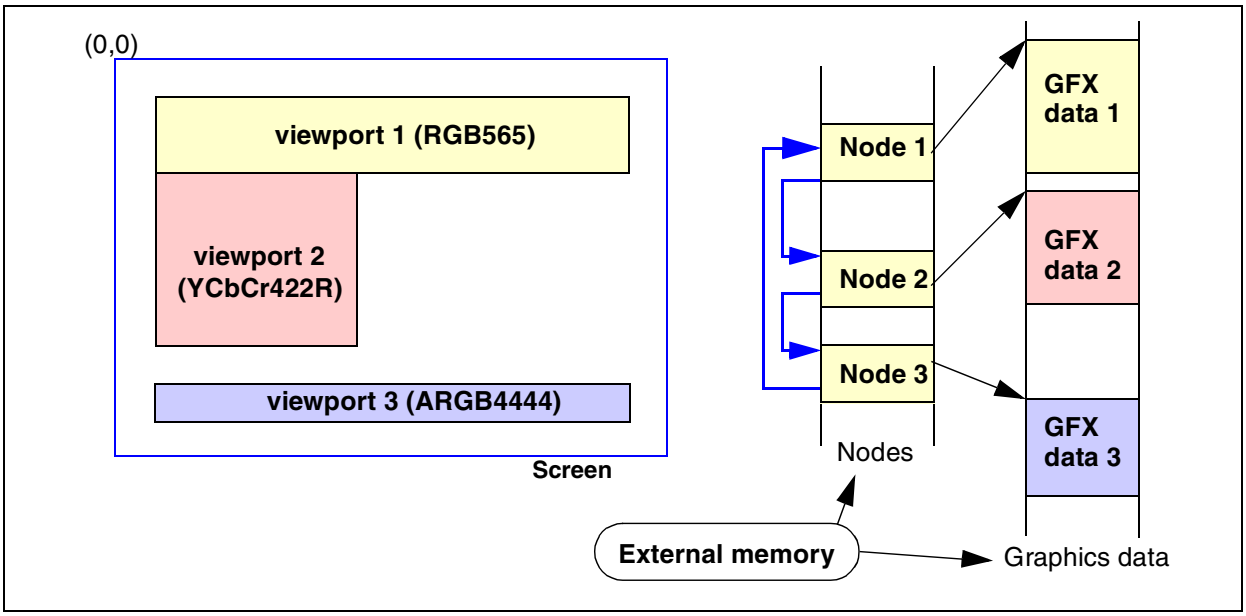
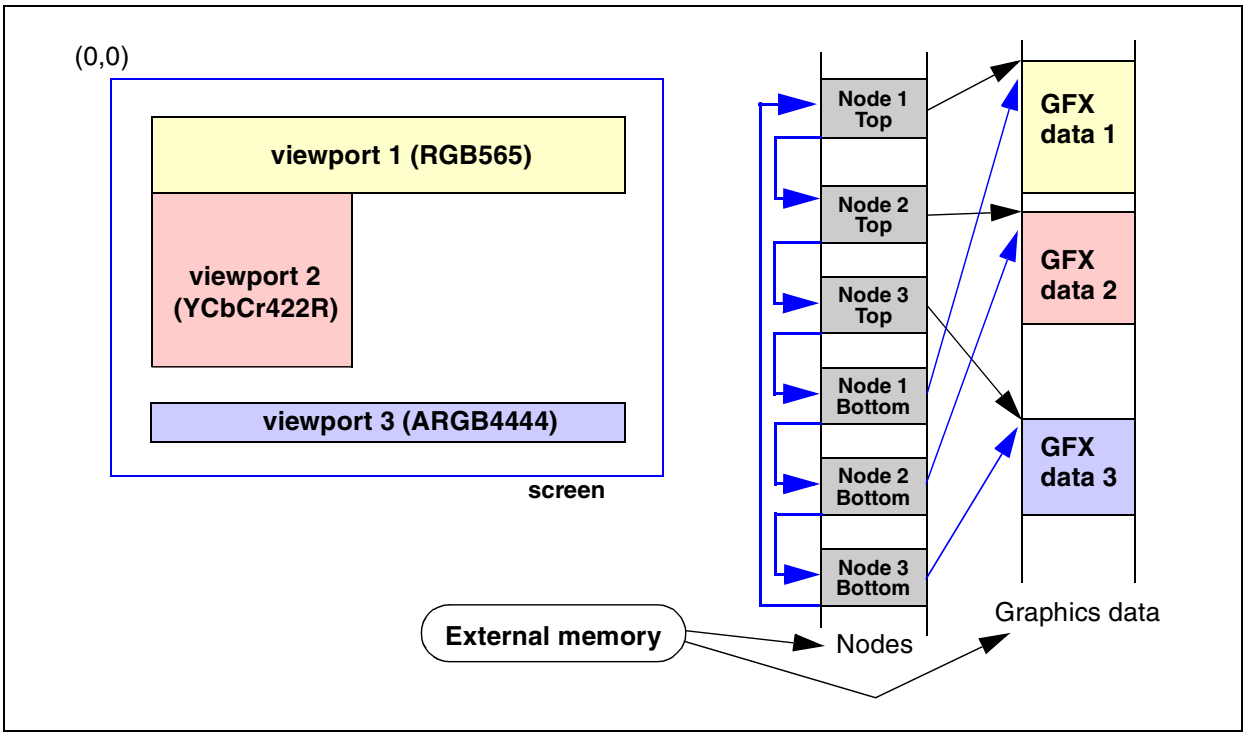


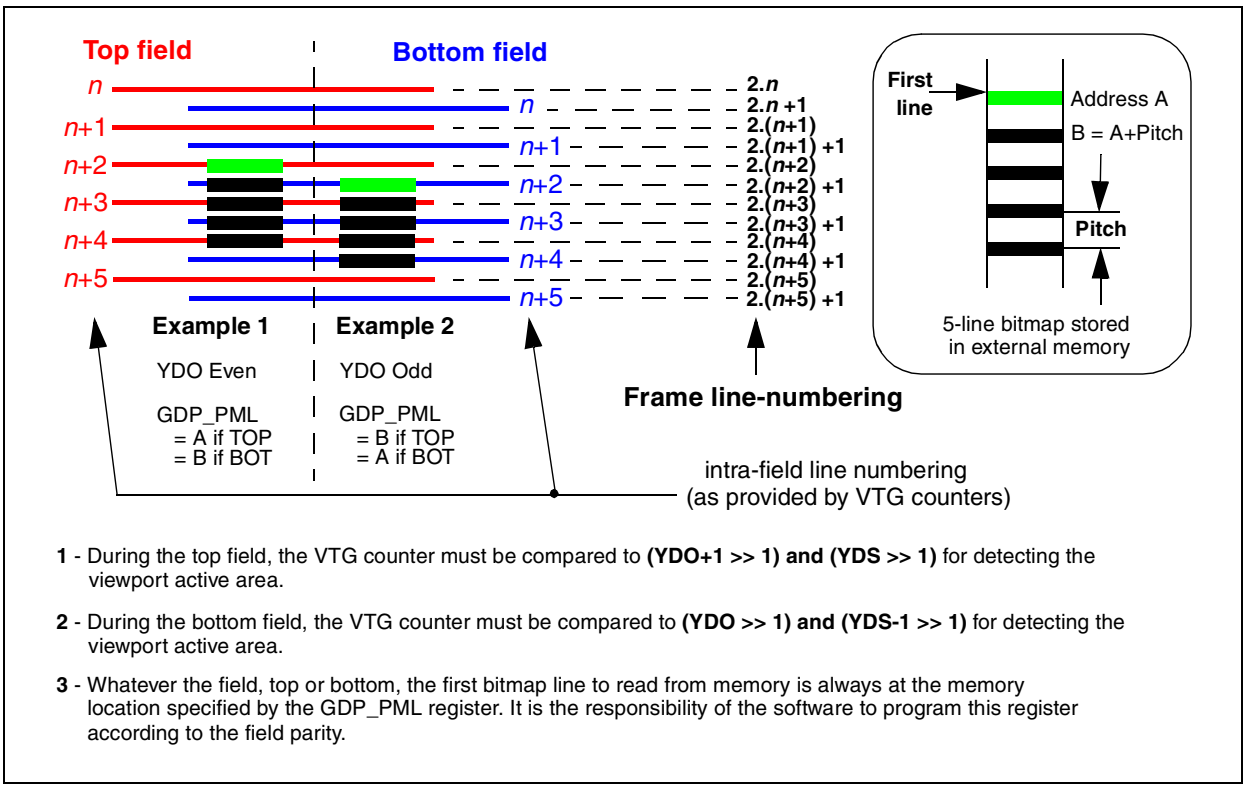
Figure 165: Typical display link-list configuration for an interlaced display



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 166: 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*_CTL 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*_CTL 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_CTL 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

Note: The vertical resizing factor does not impact the bandwidth.

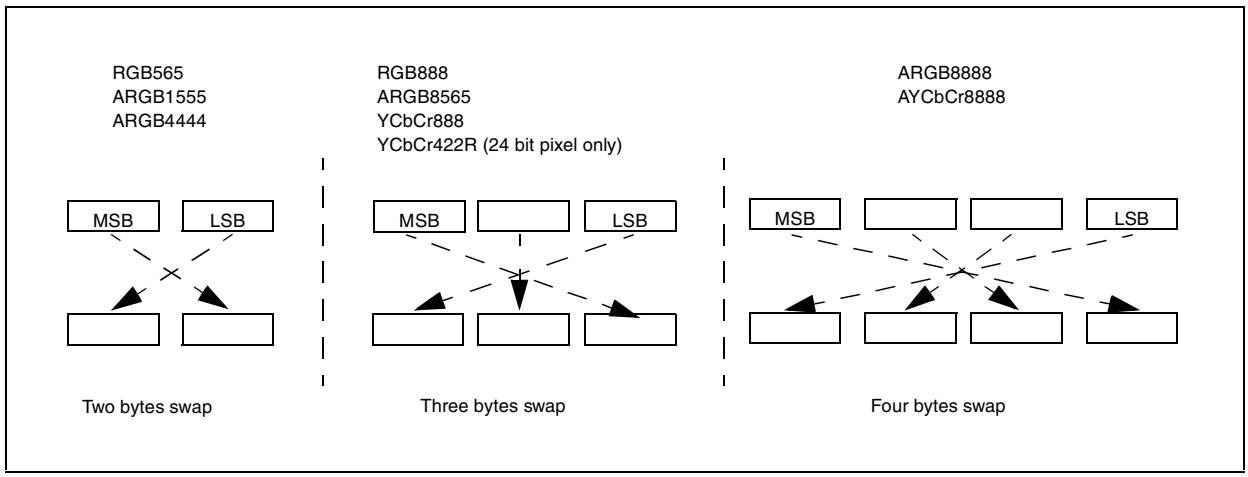
60.5.2 Description of unitary functions

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 167 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_CTL.BIGNOTLITTLE.

Figure 167: 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 148: 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 GLOBAL_ALPHA_0 value. When bit 15 is 1, the alpha output is forced to the GLOBAL_ALPHA_1 value.

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_CTL). 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 the 2D blitter chapter.

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.

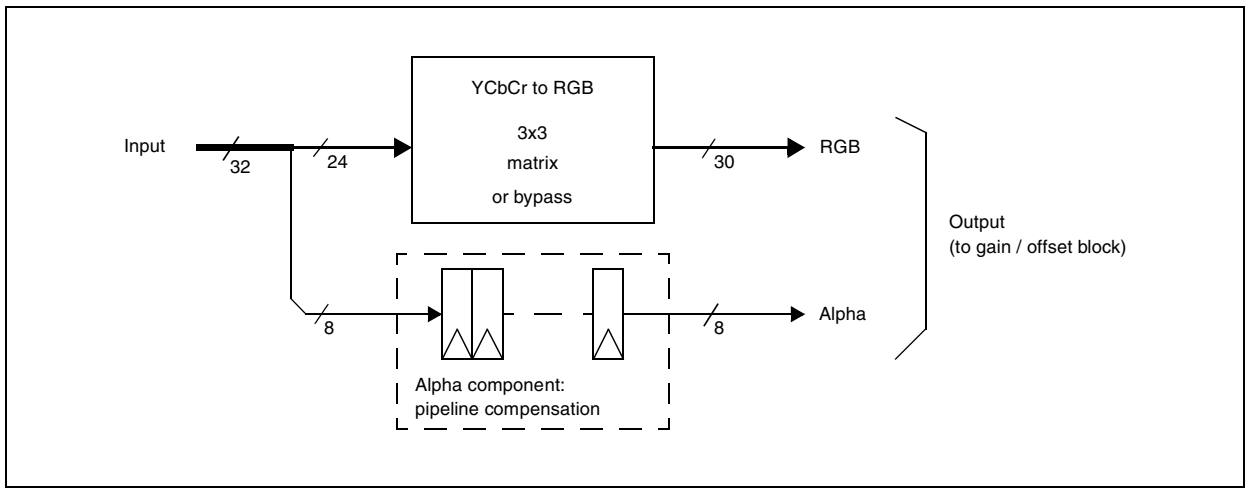
Table 149A: 601 colorimetry / floating-point matrix / digital range

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 149B: 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 168: Color space converter block diagram



2D-resize, vertical filter

The GDP contains a vertical filter that can be used for vertical up/down sampling of graphics displays. The filter is based on a 2-tap sample rate converter with 8 phases and is programmed using GAM_GDPn_VSRC. The SRC increment is programmed using a 2.8 format and the initial phase can be programmed to a resolution of 8 sub-line positions. Resizing can be performed with the filter enabled for interpolation or disabled for line repetition/dropping.

The vertical filter can be used for up or down sampling, but in the case of interlace displays, the best results are achieved when a resizing factor of x 2 or greater is used. The vertical filter is provided to allow storage of HD resolution graphics with a lower resolution, for example x1/4 compared with full resolution storage.

2D-resize, horizontal filter

The GDP contains a horizontal filter that can be used for upsampling of graphics displays. The filter is based on an 5-tap sample rate converter with 8 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 8 sub-pixel 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 150: 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_CTL.ENA_COLOR_KEY to 1. When set, the color key operates as follows:

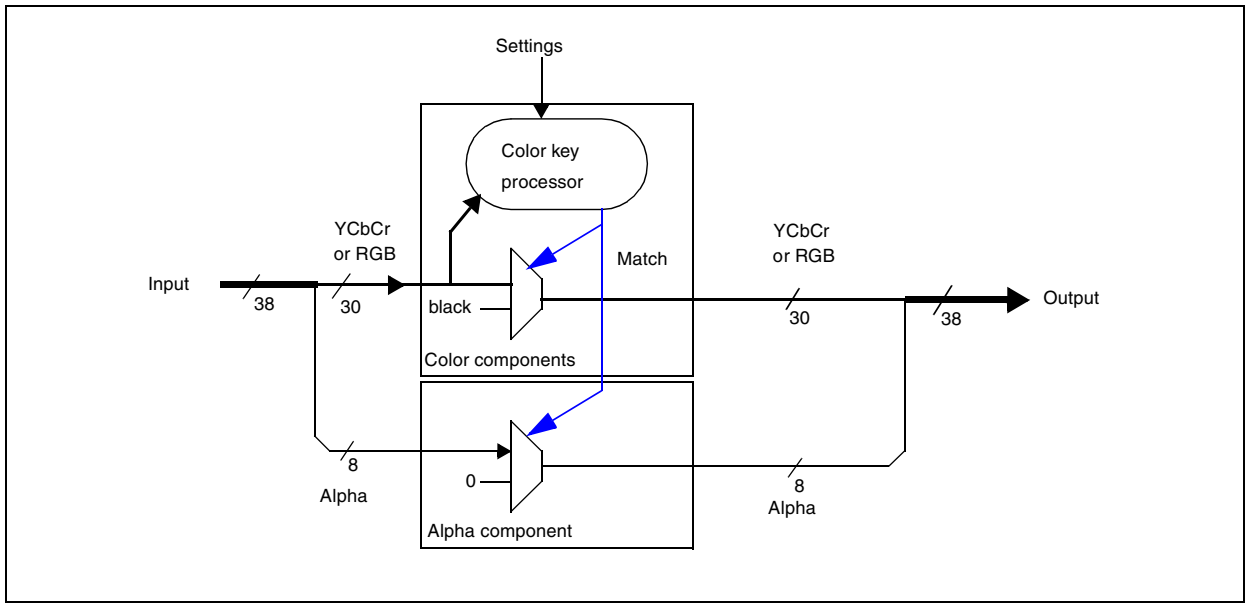
```
COLOR_KEY_MATCH = TRUE
IF
   [(Rin<Rmin) or (Rin>Rmax)] AND GAM_GDPn[21:20] = 11
OR (Rmin<=Rin<=Rmax) AND GAM_GDPn[21:20] = 01
OR GAM_GDPn[21:20] = x0
AND
   [(Gin<Gmin) or (Gin>Gmax)] AND GAM_GDPn[19:18] = 11
OR (Gmin<=Gin<=Gmax) AND GAM_GDPn[19:18] = 01
OR GAM_GDPn[19:18] = x0
AND
   [(Bin<Bmin) 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 Rmin, Gmin, Bmin are programmed in GAM_GDPn_KEY1.

The values *Rmax*, *Gmax* and *Bmax* are programmed in GAM_GDPn_KEY2.

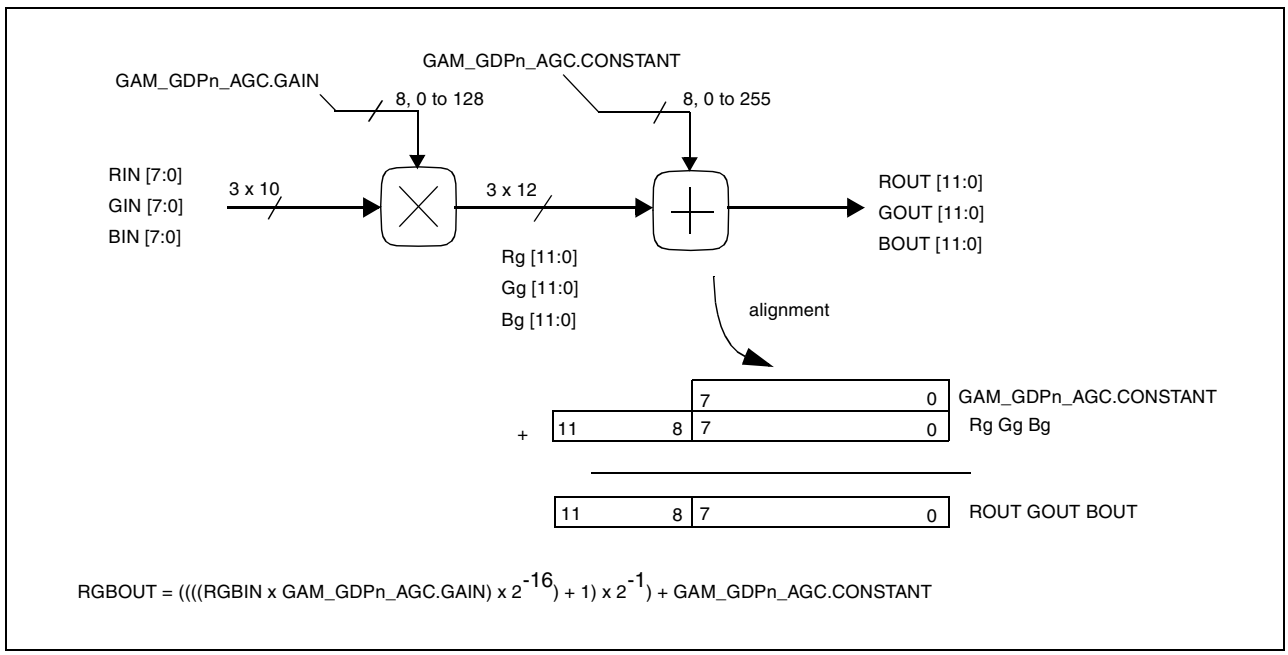
Figure 169: Color key block diagram



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 170: 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 ENA_ALPHA_HBORDER / ENA_ALPHA_VBORDER configuration bits in register GAM_GDPn_CTL.

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 ENA_ALPHA_HBORDER]
OR ((FirstLine OR LastLine)  AND ENA_ALPHA_VBORDER))
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.
ENA_ALPHA_VBORDER, ENA_ALPHA_HBORDER, and PREMULTIPLIED_FORMAT are in GAM_GDPn_CTL.

60.6 Cursor plane (CUR)

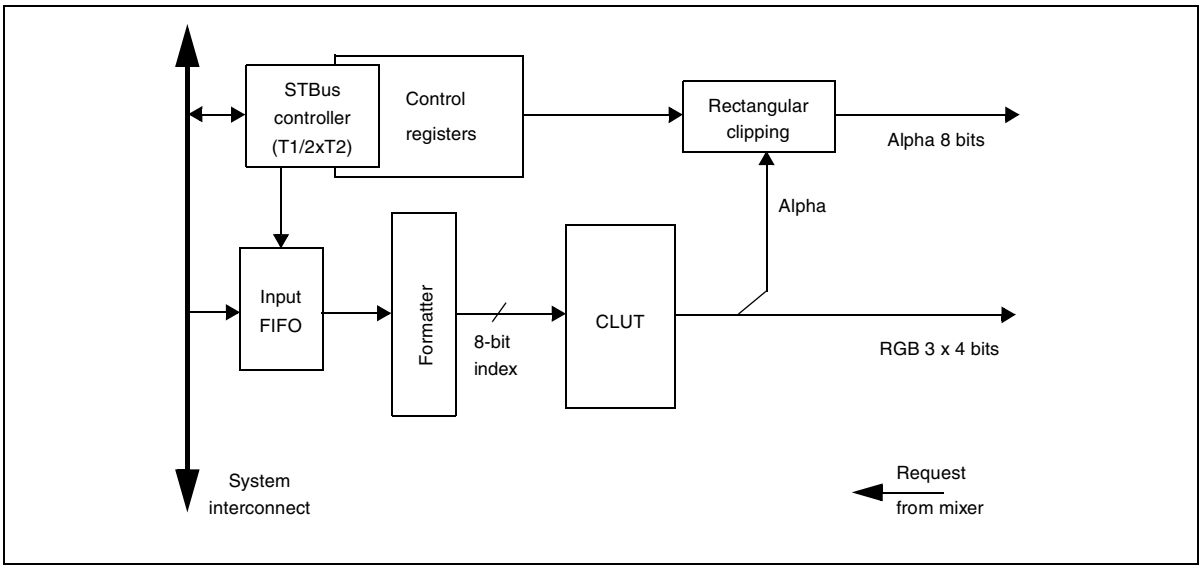
The cursor is defined as a 128x128 pixel area held in off-chip memory, in ACLUT8 format. Each cursor entry is a 16-bit ARGB4444 color + alpha value. The alpha factor of four-bits is for handling an anti-aliased cursor pattern on top of the composed output picture.

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 171: CUR functional block diagram



60.6.1 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. Let us 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:

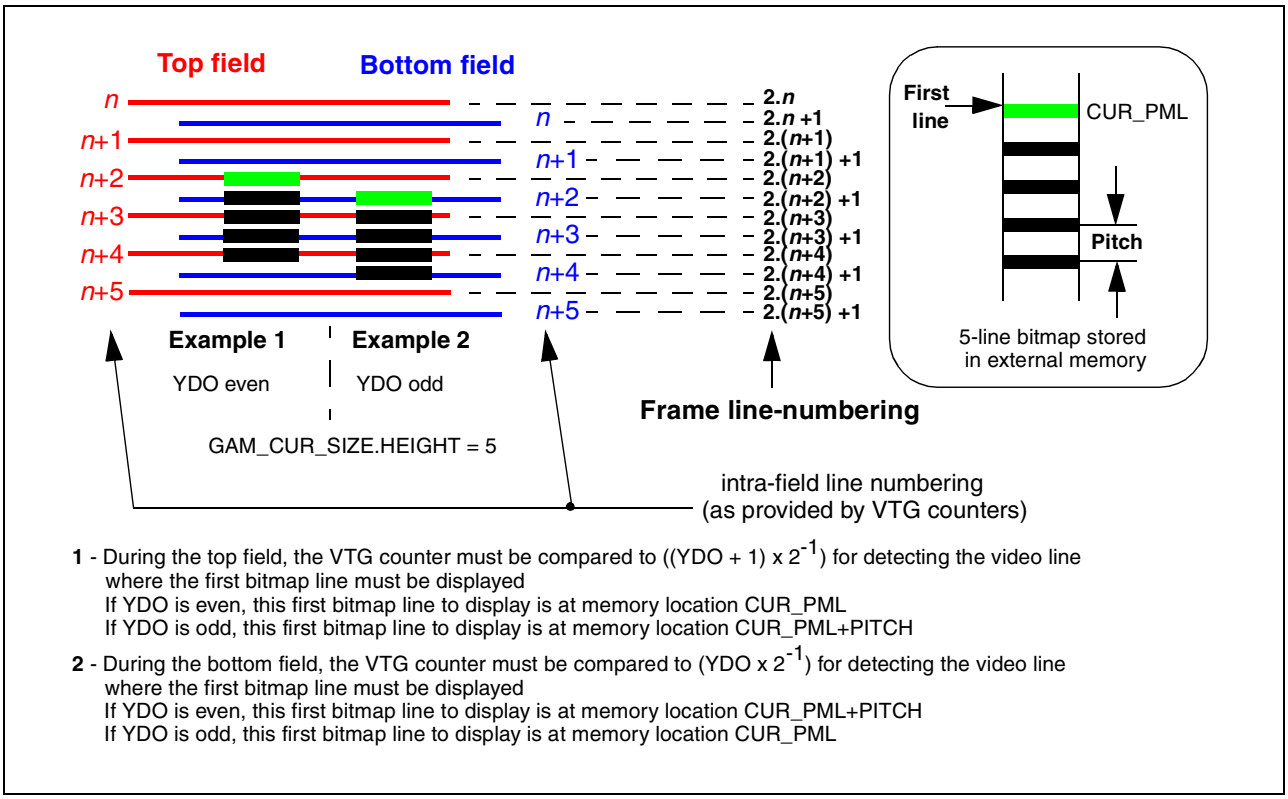
  MAX_CURSOR_WIDTH = 128 - (CUR_MEM_ADDR mod 16)

  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_CTL.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 172: Cursor line numbering assumption in an interlaced display



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_CTL 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_CTL 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).

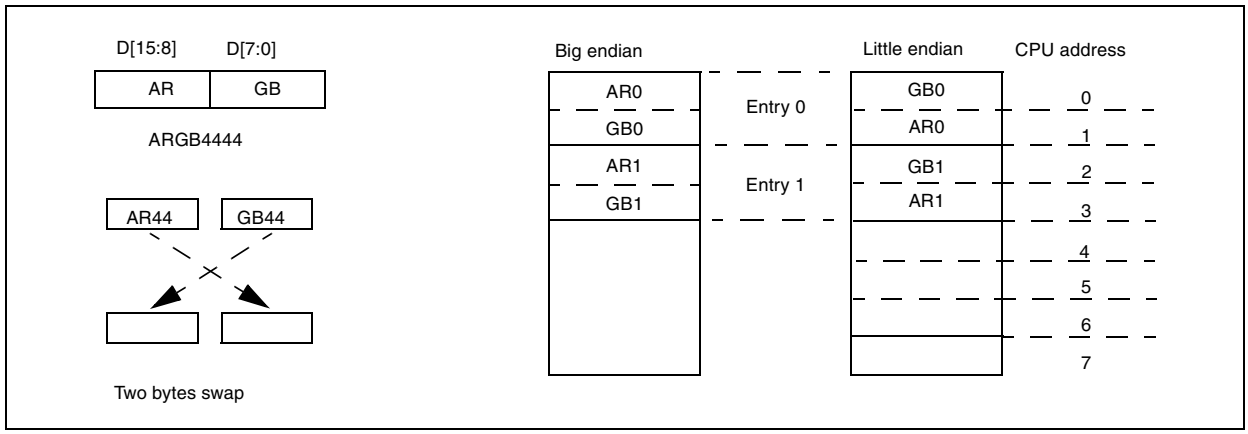
60.6.2 Unitary functions

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 151: 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).

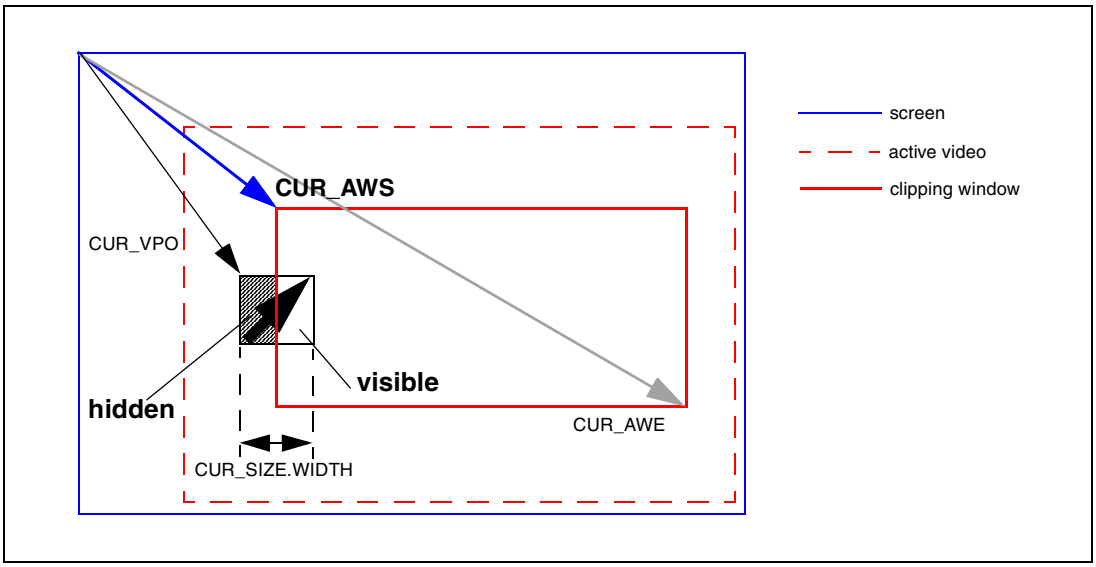
Figure 173 illustrates the use of the hardware cursor clipping window. Bits XDO and YDO are signed (register GAM_CUR_VPO), so that the top-left corner of the cursor pattern can be positioned anywhere in the screen, even in horizontal and vertical blanking.

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 173: Cursor clipping window



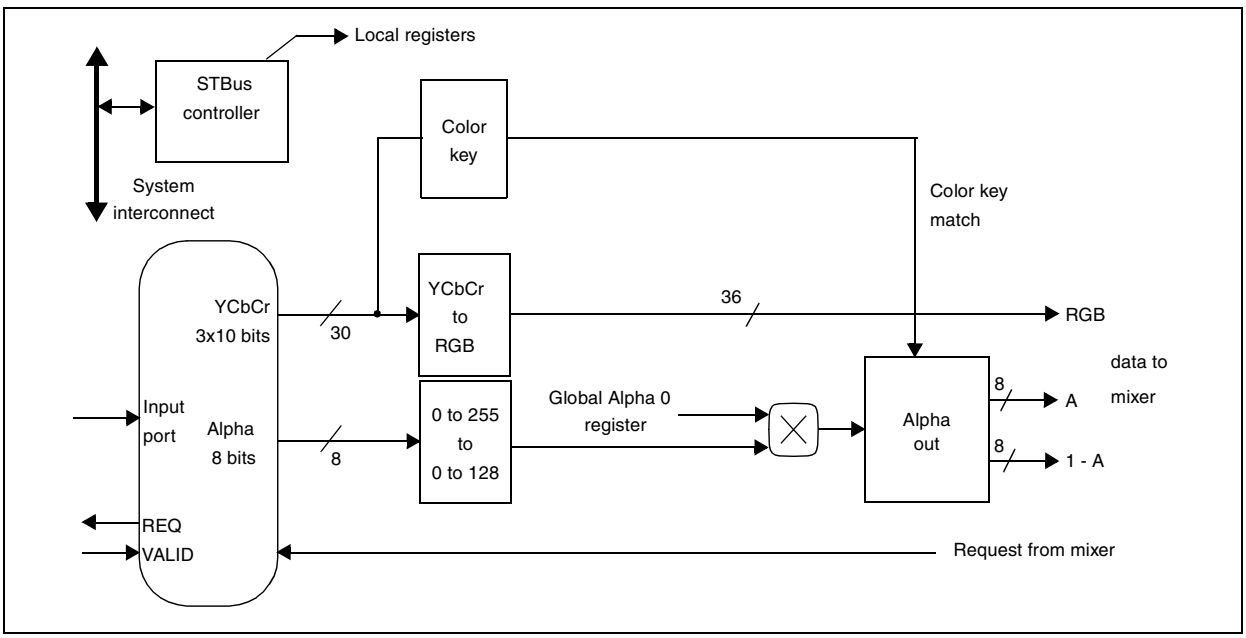
60.7 Video plug (VID)

The STi7710 compositor has two video input plugs, one main: VID1 and one auxiliary: VID2. These inputs accept 3 x 10 bits YCbCr video from the main and auxiliary display processors respectively. Figure 174 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:

- Matrix conversion from YCbCr to RGB (601 or 709), with programmable coefficients.
- Color key capability.
- Global alpha blending (combined with the per pixel alpha channel, if any).
- Vertical and horizontal edge smoothing.

Figure 174: VID functional block diagram



60.7.1 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 152: 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 153: 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.*

60.7.2 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_CTL.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).

60.8 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.

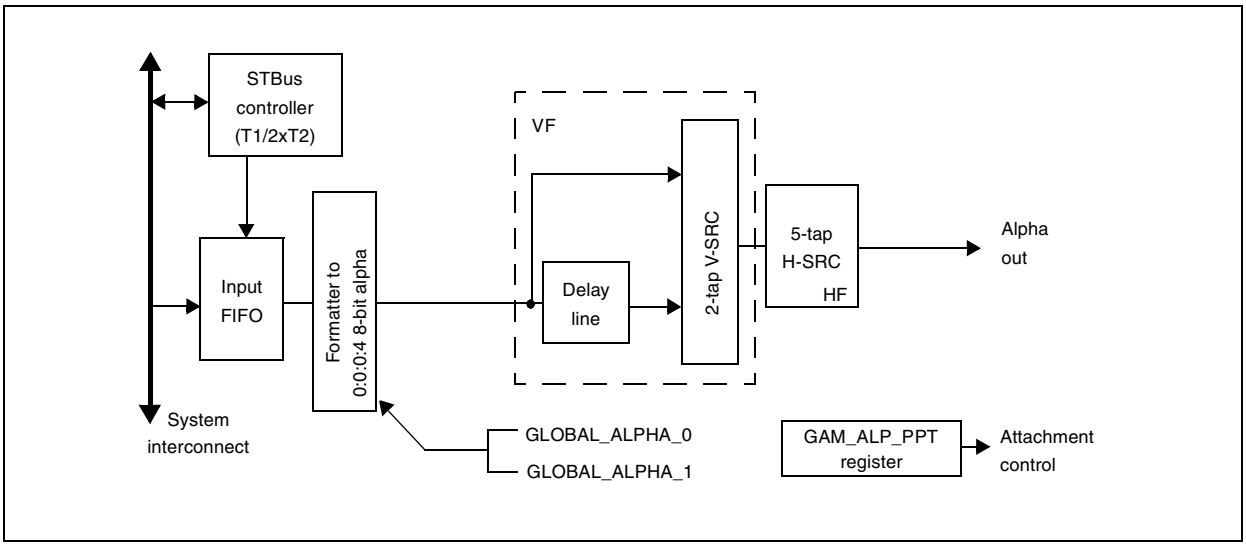
This alpha component can then be attached to a color layer (video or graphics), and is combined with the alpha component of the layer. This occurs in the digital mixer block. The features are:

- 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)
- 2-tap vertical sample rate converter for vertical resize. This can be used for half-resolution applications, that is, rescaling factor 2x or more (rescaling factors of less than 2x are not recommended for interlace displays).

The ALP output format is an 8-bit alpha value, ranging from 0 to 128.

Figure 175: ALP functional block diagram



60.8.1 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

Similar to GDP. Refer to the relevant GDP subsection.

How to modify the display parameters

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

Note: The vertical resizing factor does not impact the bandwidth.

60.8.2 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 154: 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_CTL.ENA_REVERSE_ALPHA is set to 1.

2D-resize, vertical filter and horizontal filter

This is similar to GDP. Refer to Section 60.5.1: *General description of GDPs on page 604*.

61 Compositor registers

61.1 Introduction

All the registers are seen from a start address. The registers of individual submodules of the compositor, have addresses with respect to the corresponding base address (*BaseAddress* + offset).

The relative register map is as given below. Each block occupies a 64x32-bit word address range.

A complete register address is: *CompositorBaseAddress* + *BlockOffset* + register offset.

The *CompositorBaseAddress* is:

0x3821 5000.

Table 155: Compositor/relative register map

Block name	BlockOffset
CURSOR	0x0000 (*CUROffset*)
GDP1	0x0100 (*GDP1Offset*)
GDP2	0x0200 (*GDP2Offset*)
ALP	0x0600 (*ALPOffset*)
VID1	0x0700 (*VID1Offset*)
VID2	0x0800 (*VID2Offset*)
MIX1	0x0C00 (*MIX1Offset*)
MIX2	0x0D00 (*MIX2Offset*)

Table 156: Generic compositor register summary

Offset (0x)	CUR	GDPn	VIDn	ALP	MIX1	MIX2
0000	CTL	CTL	CTL	CTL	CTL	CTL
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	MPR1	HFP		
0034		PPT	MPR2	PPT	CRB	
0038			MPR3		ACT	ACT
003C			MPR4			
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		
00FC	PKZ	PKZ		PKZ		

A complete description of the compositor registers is given in the next sections. The complete register name can be found applying the following rule: *prefix* GAM_ *prefix* 'PIPE'_ register designation.

Example: GAM_CUR_PML

61.2 Register map

Table 157: Mixers 1 and 2, cursor, video-plug and capture register map

Description	Register	Offset	Type
Control	GAM_MIX1_CTL	0x0000	R/W
	GAM_MIX1_CRB	0x0034	R/W
	GAM_MIX1_ACT	0x0038	R/W
Background color	GAM_MIX1_BKC	0x0004	R/W
	GAM_MIX1_BCO	0x000C	R/W
	GAM_MIX1_BCS	0x0010	R/W
Active video area definition	GAM_MIX1_AVO	0x0028	R/W
	GAM_MIX1_AVS	0x002C	R/W
Control	GAM_MIX2_CTL	0x0000	R/W
	GAM_MIX2_ACT	0x0038	R/W
Active video area definition	GAM_MIX2_AVO	0x0028	R/W
	GAM_MIX2_AVS	0x002C	R/W
Control register	GAM_CUR_CTL	0x0000	R/W
Viewport offset	GAM_CUR_VPO	0x000C	R/W
Pixmap-related registers	GAM_CUR_PML	0x0014	R/W
	GAM_CUR_PMP	0x0018	R/W
	GAM_CUR_SIZE	0x001C	R/W
	GAM_CUR_CML	0x0020	R/W
Clipping window	GAM_CUR_AWS	0x0028	R/W
	GAM_CUR_AWE	0x002C	R/W
Miscellaneous	GAM_CUR_PKZ	0x00FC	R/W
Control register	GAM_VIDn_CTL	0x0000	R/W
Alpha blending	GAM_VIDn_ALP	0x0004	R/W
Video viewport definition	GAM_VIDn_VPO	0x000C	R/W
	GAM_VIDn_VPS	0x0010	R/W
Color key control	GAM_VIDn_KEY1	0x0028	R/W
	GAM_VIDn_KEY2	0x002C	R/W
Programmable matrix coefficients	GAM_VIDn_MPRn		
	GAM_VIDn_MPR1	0x0030	R/W
	GAM_VIDn_MPR2	0x0034	R/W
	GAM_VIDn_MPR3	0x0038	R/W
	GAM_VIDn_MPR3	0x003C	R/W

Table 158: Generic display pipeline register map

Description	Register	Offset	Type		128-bit word alignment
Control register	GAM_GDPn_CTL	0x0000	RO/LLU	Memory node	Node 128-bit word 1
Blending and dynamic range control	GAM_GDPn_AGC	0x0004	RO/LLU		
Horizontal sample rate converter control	GAM_GDPn_HSRC	0x0008	RO/LLU		
Viewport definition	GAM_GDPn_VPO	0x000C	RO/LLU		
	GAM_GDPn_VPS	0x0010	RO/LLU		Node 128-bit word 2
Pixmap-related settings	GAM_GDPn_PML	0x0014	RO/LLU		
	GAM_GDPn_PMP	0x0018	RO/LLU		
	GAM_GDPn_SIZE	0x001C	RO/LLU		
Vertical sample rate converter control	GAM_GDPn_VSRC	0x0020	RO/LLU		Node 128-bit word 3
Pointer to the next viewport node	GAM_GDPn_NVN	0x0024	RO/LLU		
Color key	GAM_GDPn_KEY1	0x0028	RO/LLU		
	GAM_GDPn_KEY2	0x002C	RO/LLU		
Pointer the horizontal filter coefficients	GAM_GDPn_HFP	0x0030	RO/LLU		Node 128-bit word 4
Viewport properties	GAM_GDPn_PPT	0x0034	RO/LLU		
Reserved		0x0038	-		
		0x003C	-		
Horizontal filter coefficients	GAM_GDPn_HFCn				
	GAM_GDPn_HFC0	0x0040	RO/LLU	Filter coefficient structure	HF coefficient 128-bit word 1
	GAM_GDPn_HFC1	0x0044	RO/LLU		
	GAM_GDPn_HFC2	0x0048	RO/LLU		
	GAM_GDPn_HFC3	0x004C	RO/LLU		
	GAM_GDPn_HFC4	0x0050	RO/LLU		HF coefficient 128-bit word 2
	GAM_GDPn_HFC5	0x0054	RO/LLU		
	GAM_GDPn_HFC6	0x0058	RO/LLU		
	GAM_GDPn_HFC7	0x005C	RO/LLU		
	GAM_GDPn_HFC8	0x0060	RO/LLU		HF coefficient 128-bit word 3
	GAM_GDPn_HFC9	0x0064	RO/LLU		
Reserved	-	0x0068	-		
		0x006C	-		
STBus protocol / packet maximum size	GAM_GDPn_PKZ	0x00FC	R/W		Not Relevant

Table 159: Alpha plane register map

Description	Register	Offset	Type		128-bit word alignment
Control register	GAM_ALP_CTL	0x0000	RO/LLU	Memory node	Node 128-bit word 1
	GAM_ALP_ALP	0x0004	RO/LLU		
Horizontal sample rate converter control	GAM_ALP_HSRC	0x0008	RO/LLU		
Viewport definition	GAM_ALP_VPO	0x000C	RO/LLU		
	GAM_ALP_VPS	0x0010	RO/LLU		Node 128-bit word 2
Pixmap-related settings	GAM_ALP_PML	0x0014	RO/LLU		
	GAM_ALP_PMP	0x0018	RO/LLU		
	GAM_ALP_SIZE	0x001C	RO/LLU		
Vertical sample rate converter control	GAM_ALP_VSRC	0x0020	RO/LLU		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	RO/LLU		
Reserved		0x0038	-		
		0x003C			
Horizontal filter coefficients	GAM_ALP_HFCn				
	GAM_ALP_HFC0	0x0040	RO/LLU	Filter coefficient structure	HF coefficient. 128-bit word 1
	GAM_ALP_HFC1	0x0044	RO/LLU		
	GAM_ALP_HFC2	0x0048	RO/LLU		
	GAM_ALP_HFC3	0x004C	RO/LLU		
	GAM_ALP_HFC4	0x0050	RO/LLU		HF coefficient 128-bit word 2
	GAM_ALP_HFC5	0x0054	RO/LLU		
	GAM_ALP_HFC6	0x0058	RO/LLU		
	GAM_ALP_HFC7	0x005C	RO/LLU		
	GAM_ALP_HFC8	0x0060	RO/LLU		HF coefficient 128-bit word 3
	GAM_ALP_HFC9	0x0064	RO/LLU		
Reserved		0x0068	-		
		0x006C			
STBus protocol / packet maximum size	GAM_ALP_PKZ	0x00FC	R/W		Not Relevant

61.3 Register descriptions

GAM_ALP_CTL 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	ENA_HFILTER_UPDATE	ENA_REVERSE_ALPHA	Reserved	A1INBYTEORDER	A1BYTESTART			Reserved												ENA_V_RESIZE	ENA_H_RESIZE	Reserved				ALPHARANGE	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] **ENA_HFILTER_UPDATE**: 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] **ENA_REVERSE_ALPHA**: in reverse mode, (128-alpha) operator is applied at the pipeline input.
0: alpha mode 1: reverse alpha mode

[28] **Reserved**

[27] **A1INBYTEORDER**: 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] **A1BYTESTART**: 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

[23:12] **Reserved**

[11] **ENA_V_RESIZE**
0: vertical resize block is disabled 1: vertical resize block is enabled

[10] **ENA_H_RESIZE**
0: horizontal resize block is disabled 1: horizontal resize block is enabled

[9:6] **Reserved**

[5] **ALPHARANGE**: 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

31	30	29	28	27	26	25	24	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_1								GLOBAL_ALPHA_0							

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

31	30	29	28	27	26	25	24	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							HF_FILTER_MODE	Reserved						HSRC_INITIAL_PHASE		Reserved						HSRC_INCREMENT									

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**: if 1, the filter is enabled, otherwise only horizontal resizing

[23:19] **Reserved**

[18:16] **HSRC_INITIAL_PHASE**: the horizontal sample rate converter state-machine initial phase, 0 to 7

[15:10] **Reserved**

[9:0] **HSRC_INCREMENT**: 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_ADDRESS																									

Address: *CompositorBaseAddress* + *ALPOffset* + 0x14

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register is a 32-bit register containing 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_ADDRESS**: 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

31	30	29	28	27	26	25	24	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_VALUE															

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_VALUE**: memory pitch for the displayed pixmap
Note: the pitch is the distance inside the memory, in byte unit, between two vertically adjacent pixels

GAM_ALP_SIZE ALP 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					PIXMAP_HEIGHT											Reserved					PIXMAP_WIDTH										

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 line unit, 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 pixel unit

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_ADDRESS																						Reserved			

Address: *CompositorBaseAddress* + *ALPOffset* + 0x24
Type: R/W/link list update
Buffer: Double-bank, automatic hardware toggle
Reset: 0
Description: This register is a 32-bit register containing 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_ADDRESS**: 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_VSRC ALP vertical sample rate converter

31–25	24	23–19	18–16	15–10	9–0
Reserved	VF_FILTER_MODE	Reserved	VSRC_INITIAL_PHASE	Reserved	VSRC_INCREMENT

Address: *CompositorBaseAddress* + *ALPOffset* + 0x20

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the configuration for the vertical sample rate converter.

[31:25] **Reserved**

[24] **VF_FILTER_MODE**: if 1, filtering is enabled, otherwise vertical resizing

[23:19] **Reserved**

[18:16] **VSRC_INITIAL_PHASE**: the vertical sample rate converter state-machine initial phase, 0 to 7

[15:10] **Reserved**

[9:0] **VSRC_INCREMENT**: the vertical sample rate converter state-machine increment, in 2.8 format

GAM_ALP_HFP ALP horizontal filter pointer

31–26	25–4	3–0
64MB_BANK	H_FILTER_POINTER	Reserved

Address: *CompositorBaseAddress* + *ALPOffset* + 0x30

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register is a 32-bit register containing 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_CTL[30] = 1 (ENA_HFILTER_UPDATE).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:4] **H_FILTER_POINTER**: 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

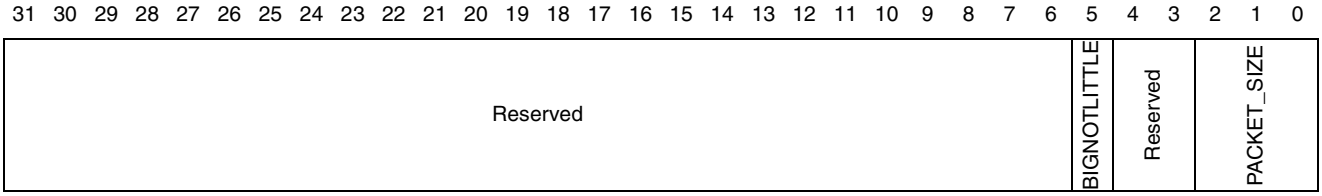
Address: *CompositorBaseAddress* + *ALPOffset* + register offset
HFC0: 0x40, HFC1: 0x44, HFC2: 0x48, HFC3: 0x4C, HFC4: 0x50, HFC5: 0x54, HFC6: 0x58, HFC7: 0x5C, HFC8: 0x60, HFC9: 0x64

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 STi7710 device.

[31:6] **Reserved**

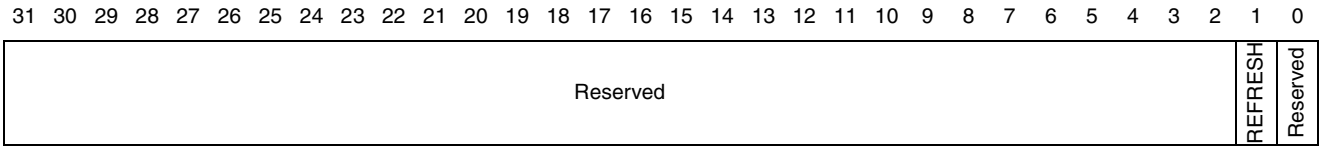
[5] **BIGNOTLITTLE**: CPU endianness
0: little endian CPU 1: big endian CPU

[4:3] **Reserved**: must be set to 0.

[2:0] **PACKET_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_CTL 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 byte

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

31	30	29	28	27	26	25	24	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					AWE_Y											Reserved				AWE_X											

Address: *CompositorBaseAddress* + *CUROffset* + 0x2C

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: This register 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

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																										BIGNOTLITTLE	Reserved		PACKET_SIZE		

Address: *CompositorBaseAddress* + *CUROffset* + 0xFC

Type: R/W

Buffer: Immediate

Reset: 0

Description: This register 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] **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_CTL 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	ENA_HFILTER_UPDATE	LSB_STUFFING_MODE	Reserved	CHROMA_FORMAT	601/709_SELECTION	PREMULTIPLIED_FORMAT	BIGNOTLITTLE	Reserved	R/CR_COLOR_KEY_CONFIGURATION	G/Y_COLOR_KEY_CONFIGURATION	B/CB_COLOR_KEY_CONFIGURATION	Reserved	ENA_COLOR_KEY	ENA_ALPHA_VBORDER	ENA_ALPHA_HBORDER	ENA_V_RESIZE	ENA_H_RESIZE	Reserved		ALPHARANGE		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] **ENA_HFILTER_UPDATE**: 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. Warning: 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_SELECTION**
0: The color space conversion uses the 601 colorimetry
1: The color space conversion uses the 709 colorimetry

[24] **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_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_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_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] **ENA_COLOR_KEY**
0: color key feature is disabled 1: color key feature is enabled

[13] **ENA_ALPHA_VBORDER**: this provides soft horizontal edges for the viewport, and has a 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] **ENA_ALPHA_HBORDER**: This 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] **ENA_V_RESIZE**
0: vertical resize block is disabled 1: vertical resize block is enabled

[10] **ENA_H_RESIZE:**
0: horizontal resize block is disabled 1: horizontal resize block is enabled

[9:6] **Reserved**

[5] **ALPHARANGE**: 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
00000: RGB565 (0x0) 00001: RGB888 (0x1)
00100: ARGB8565 (0x4) 00101: ARGB8888 (0x5)
00110: ARGB1555 (0x6) 00111: ARGB4444 (0x7)
10000: YCbCr888 (0x10) 10010: YCbCr4:2:2R (0x12)
10101: 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

31 30 29 28 27 26 25	24	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	HF_FILTER_MODE	Reserved	HSRC_INITIAL_PHASE	Reserved	HSRC_INCREMENT

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**: if 1, the filter is enabled, otherwise only horizontal resizing.

[23:19] **Reserved**

[18:16] **HSRC_INITIAL_PHASE**: the horizontal sample rate converter state-machine initial phase, 0 to 7.

[15:10] **Reserved**

[9:0] **HSRC_INCREMENT**: the horizontal sample rate converter state-machine increment, in 2.8 format.

GAM_GDPn_VPO GDP 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* + *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_ADDRESS																									

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x14

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register is a 32-bit register containing the memory location for the first pixel to be displayed (top-left corner).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:0] **PIXMAP_ADDRESS**: 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_VALUE															

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_VALUE**: memory pitch for the displayed pixmap.
Note: the pitch is the distance inside the memory, in byte unit, between two vertically adjacent pixels.

GAM_GDPn_SIZE GDP 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: *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.

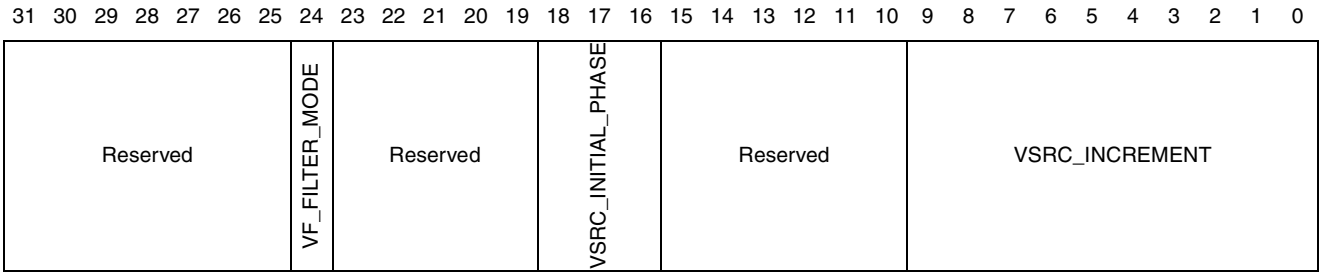
[31:27] **Reserved**

[26:16] **HEIGHT**: pixmap height, in line unit, 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 pixel unit

GAM_GDPn_VSRC GDP vertical sample rate converter



Address: *CompositorBaseAddress* + *GDPnOffset* + 0x20

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register provides the configuration for the vertical sample rate converter.

[31:25] **Reserved**

[24] **VF_FILTER_MODE**: if the bit is 1 then filtering is done, else vertical resizing

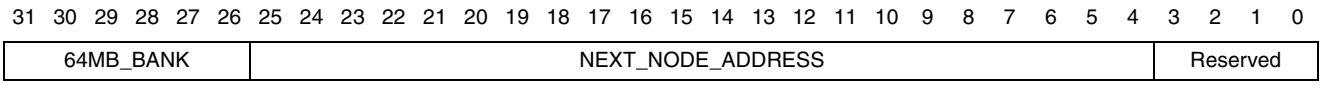
[23:19] **Reserved**

[18:16] **VSRC_INITIAL_PHASE**: the vertical sample rate converter state-machine initial phase, 0 to 7

[15:10] **Reserved**

[9:0] **VSRC_INCREMENT**: the vertical sample rate converter state-machine increment, in 2.8 format

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 is a 32-bit register containing 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_ADDRESS**: 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

31	30	29	28	27	26	25	24	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_POINTER																						Reserved			

Address: *CompositorBaseAddress* + *GDPnOffset* + 0x30

Type: Read/link list update

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register is a 32-bit register containing 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_CTL[30] = 1 (ENA_HFILTER_UPDATE).

[31:26] **64MB_BANK**: 64 Mbyte bank number

[25:4] **H_FILTER_POINTER**: 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 (10 32-bit words that must be fully contained in the specified bank)

[3:0] **Reserved**

GAM_GDPn_PPT GDP 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																												FORCEONMIX2	FORCEONMIX1	IGNOREONMIX2	IGNOREONMIX1

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**

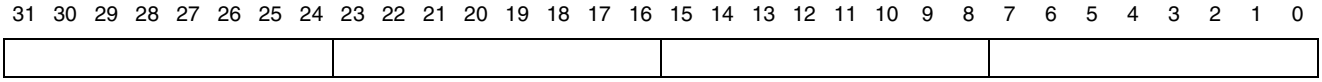
[3] **FORCEONMIX2**: when set, this bit indicates to MIX2 that the current viewport must always be displayed, even out of the MIX2 active video area (defined with MIX2_AVO / MIX2_AVS registers).
Note: The GDP pipeline must be enabled in the MIX2_CTL register.

[2] **FORCEONMIX1**: 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 that the GDP pipeline must be enabled in the MIX1_CTL register.

[1] **IGNOREONMIX2**: when set, this bit indicates to MIX2 that the current viewport must not be displayed, although the GDP pipeline is enabled in the MIX2_CTL register

[0] **IGNOREONMIX1**: when set, this bit indicates to MIX1 that the current viewport must not be displayed, although the GDP pipeline is enabled in the MIX1_CTL register

GAM_GDPn_HFCn GDP HF coefficients registers



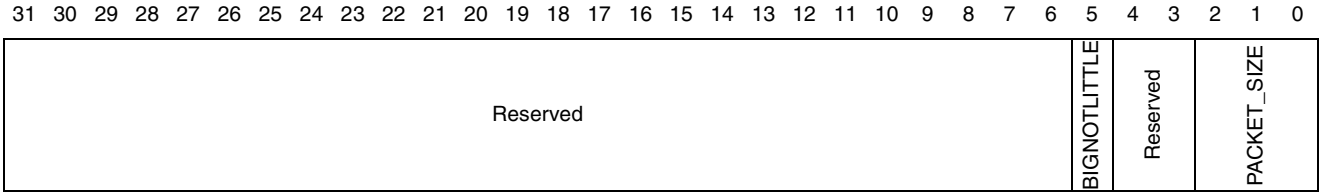
Address: *CompositorBaseAddress* + *GDPnOffset* + register offset
HFC0: 0x40, HFC1: 0x44, HFC2: 0x48, HFC3: 0x4C, HFC4: 0x50, HFC5: 0x54, HFC6: 0x58, HFC7: 0x5C, HFC8: 0x60, HFC9: 0x64

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 STi7710 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] **PACKET_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_CTL MIX 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																AP_DISPLAYENABLE	Reserved					CUR_DISPLAYENABLE	Reserved				GDP2_DISPLAYENABLE	GDP1_DISPLAYENABLE	Reserved	VID1_DISPLAYENABLE	BKC_DISPLAYENABLE

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_DISPLAYENABLE**: Alpha plane is enabled for attachment with a given layer that is enabled on the mixer output

[14:10] **Reserved**

[9] **CUR_DISPLAYENABLE**: CUR display is enabled on the mixer output

[8:5] **Reserved**

[4] **GDP2_DISPLAYENABLE**: GDP 2 display is enabled on the mixer output

[3] **GDP1_DISPLAYENABLE**: GDP 1 display is enabled on the mixer output

[2] **Reserved**

[1] **VID1_DISPLAYENABLE**: Video 1 display is enabled on the mixer output

[0] **BKC_DISPLAYENABLE**: background color display is enabled on the mixer output

GAM_MIX1_BKC Background color

31	30	29	28	27	26	25	24	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_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

31	30	29	28	27	26	25	24	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* + 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

31	30	29	28	27	26	25	24	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* + 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																				DEPTH4			DEPTH3			DEPTH2			Reserved		

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

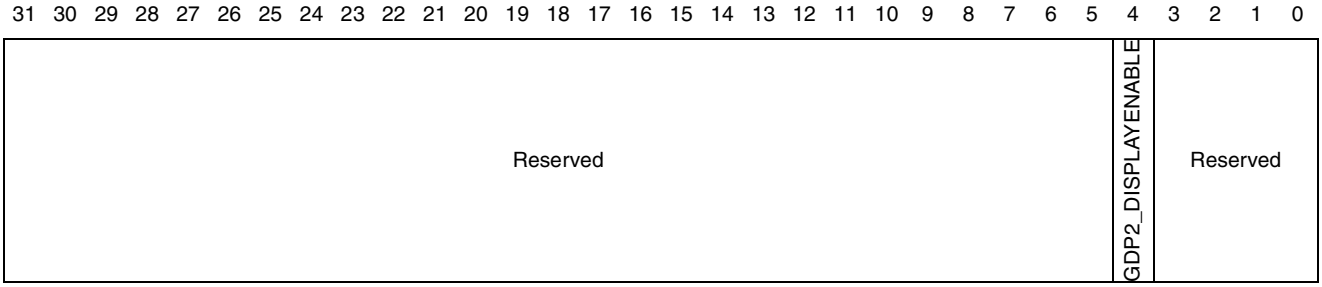
Reset: 0

Description: This register 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_CTL MIX control



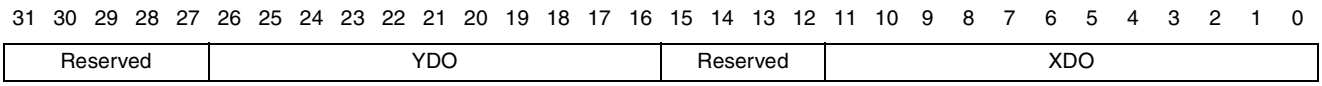
Address: *CompositorBaseAddress* + *MIX2Offset* + 0x00
Type: R/W
Buffer: Double-buffered, update on VTG Vsync
Reset: 0
Description: This register contains the values for controlling the MIX 2 core.

[31:5] **Reserved**

[4] **GDP2DISPLAYENABLE**
1: GDP 2 display is enabled on the mixer output

[3:0] **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

31	30	29	28	27	26	25	24	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* + *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

31	30	29	28	27	26	25	24	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											GDP2_ONVIDACTIVE	Reserved		VID1_ONVIDACTIVE	Reserved												GDP2_ONGFXACTIVE	Reserved			

Address: *CompositorBaseAddress* + *MIX2Offset* + 0x38

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

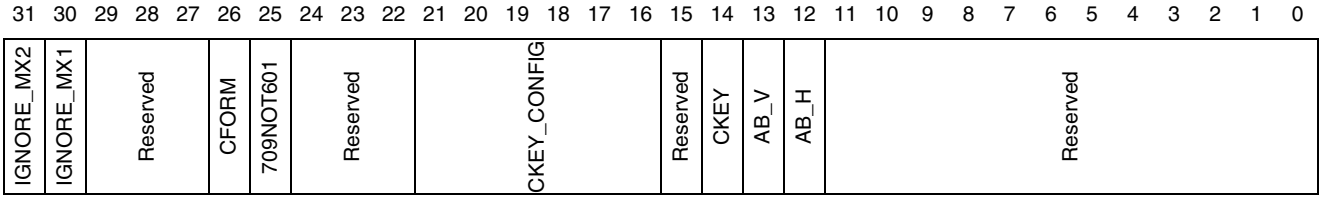
Reset: 0

Description: This register 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_ONVIDACTIVE: if set to 1, the XXX layer is taken into account for the VideoActive flag

XXX_ONGFXACTIVE: if set to 1, the XXX layer is taken into account for the GFXActive flag

GAM_VIDn_CTL VID control



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_CTL. (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_CTL. (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_CONFIG**: configuration of color key

[21:20] VID_CTL[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_CTL[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_CTL[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)

[11:0] **Reserved**

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_VALUE							

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_VALUE**: 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_MPRn Matrix programing 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
0x30	Reserved	COEFF1_RGB	Reserved / Y_OS
0x34	Reserved	COEFF3_R	Reserved / COEFF2_R
0x38	Reserved	COEFF3_G	Reserved / COEFF2_G
0x3C	Reserved	COEFF3_B	Reserved / COEFF2_B

Address: *CompositorBaseAddress* + *VIDnOffset* + register offset
MPR1: 0x30, MPR2: 0x34, MPR3: 0x38, MPR4: 0x3C

Type: R/W

Buffer: Double-buffered, update on VTG Vsync

Reset: 0

Description: These four registers contain the matrix coefficients for the YCbCr to RGB conversion.

GAM_VIDn_MPR1

COEFF1_RGB: luma coefficient in the matrix. Common for R,G,B output components

Y_OS: Y offset applied to the luma component prior to the matrix

GAM_VIDn_MPR2

COEFF3_R: Cr coefficient in the matrix for the R output component

COEFF2_R: Cb coefficient in the matrix for the R output component

GAM_VIDn_MPR3

COEFF3_G: Cr coefficient in the matrix for the G output component

COEFF2_G: Cb coefficient in the matrix for the G output component

GAM_VIDn_MPR4

COEFF3_B: Cr coefficient in the matrix for the B output component

COEFF2_B: Cb coefficient in the matrix for the B output component

The following input component range is assumed (from the video display processor):

- $-448 \leq Y \leq +373$
- $-420 \leq Cb, Cr \leq +420$

The hardwired matrix operators are given by the following expressions (RGB are gamma-corrected):

- R = COEFF1_RGB x (Y + 512 - Y_OS) + COEFF2_R x Cb + COEFF3_R x Cr
- G = COEFF1_RGB x (Y +512 - Y_OS) + COEFF2_G x Cb + COEFF3_G x Cr
- B = COEFF1_RGB x (Y + 512 - Y_OS) + COEFF2_B x Cb + COEFF3_B x Cr

The 12-bit coefficients are split into 2 separate fields:

- [8:0]: 9-bit coefficient (similar to the mantissa in a floating-point representation)
- [11:9]: 3-bit shift value (similar to an exponent in a floating-point representation)

Table 160 shows how to represent a floating-point coefficient (K_{FP}) using this format:

Table 160: Matrix coefficients computation rules

K_{FP} range	Integer equivalent K_I	9-bit programmed value	Shift
$\|K_{FP}\| \geq 2.0$	$\|K_{FP}\|$ x 64	K_I if K_{FP}>=0 else -K_I then keep 9 LSBs	6
$2.0 > \|K_{FP}\| \geq 1.0$	$\|K_{FP}\|$ x 128		5
$1.0 > \|K_{FP}\| \geq 0.5$	$\|K_{FP}\|$ x 256		4
$0.5 > \|K_{FP}\| \geq 0.25$	$\|K_{FP}\|$ x 512		3
$0.25 > \|K_{FP}\| \geq 0.125$	$\|K_{FP}\|$ x 1024		2
$0.125 > \|K_{FP}\| \geq 0.0625$	$\|K_{FP}\|$ x 2048		1
$0.0625 > \|K_{FP}\|$	$\|K_{FP}\|$ x 4096		0

Examples:

- K_{FP} = -0.336
  K_I = (int)(0.336 x 512) = 172 = 0xAC.
  Taking into account the sign, so 0xAC -> 0xFFFF FF54
  Keeping 9 bits: 0x154
  Shift = 3, so the 12-bit coefficient value that represents -0.336 is 0x754.
- K_{FP} = 1.371
  K_I = (int)(1.371 x 128) = 175 = 0xAF.
  Keeping 9 bits: 0xAF
  Shift = 5, so the 12-bit coefficient value that represents 1.371 is 0xAAF.

62 Main and auxiliary-display processors

62.1 Overview

The display processors take the video lines in 4:2:2 raster format (one buffer) or 4:2:0 macroblock format (two buffers) from external memory. 4:2:2 macroblock 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.

62.2 Fully programmable horizontal sample rate converters

- 8 taps interpolation luma/chroma filters.
- Time varying coefficients using polyphase filtering technique.
- 32 phases and 10 bit coefficients (value is -512 to 511) stored in RAM - 16 x 8 x 8 + 3 x 32 bits for luma, same for chroma. See Section : *0x3820 4000. on page 664* for details.
- Horizontal linear resizing of video.
- Linear resizing: continuous resizing of video horizontally, by a programmable factor from 0 and up to x4 with 2^(-13) step.
- Equidistant fractional pixel position down to 1/32 of pixel.
- Horizontal pan down to 1/32 pixel.

62.3 Fully programmable vertical sample rate converters

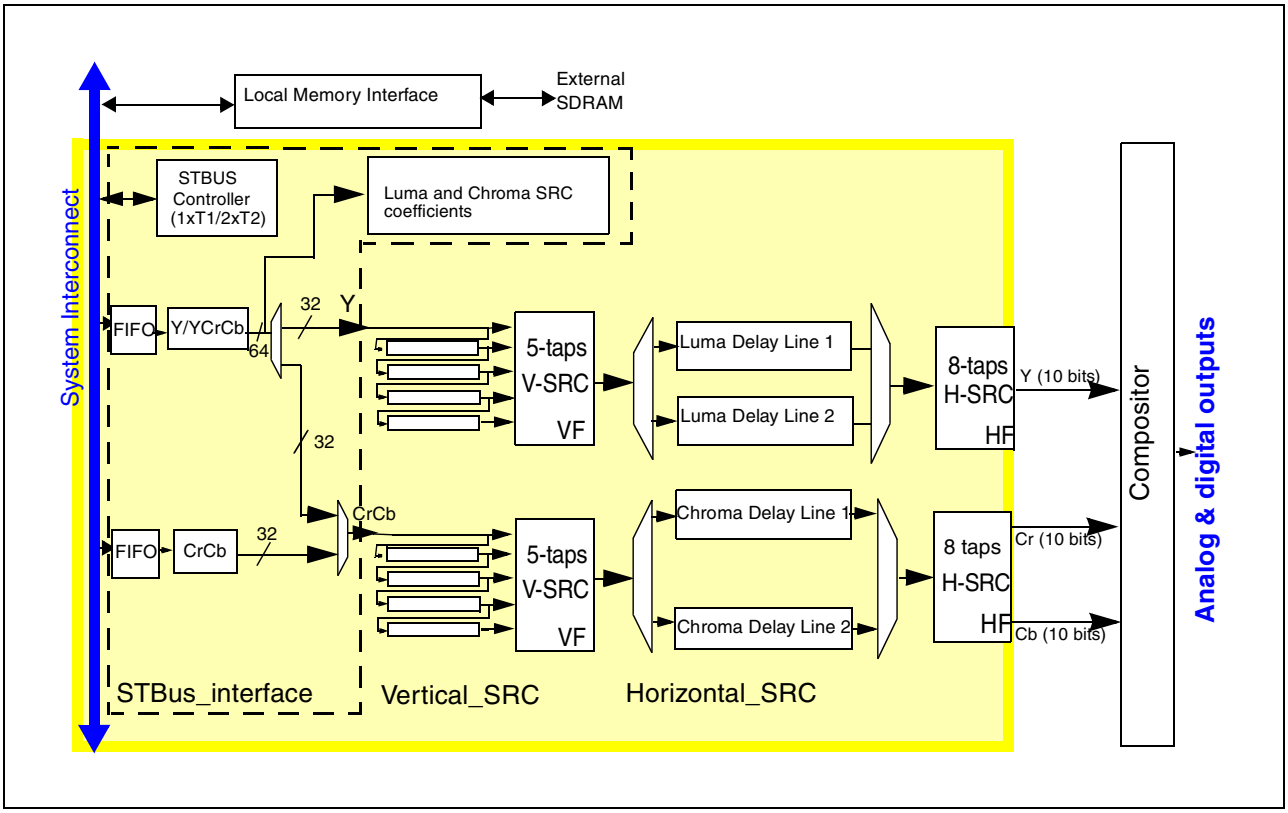
Applications: Zoom in/out, letter box resizing chroma upsampling (4:2:0 to 4:2:2).

- 5 taps interpolation luma/chroma filters.
- Time varying coefficients using polyphase filtering technique.
- 32 phases and 10 bit coefficients stored in RAM, - 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 0 and up to x 4 with 2^(-13) step.

62.4 Block diagram

Figure 176 provides a functional system overview of the display processors, as integrated in an STBus-based interconnect architecture.

Figure 176: Graphics and video generic compositor overview (proposition I)



62.5 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.

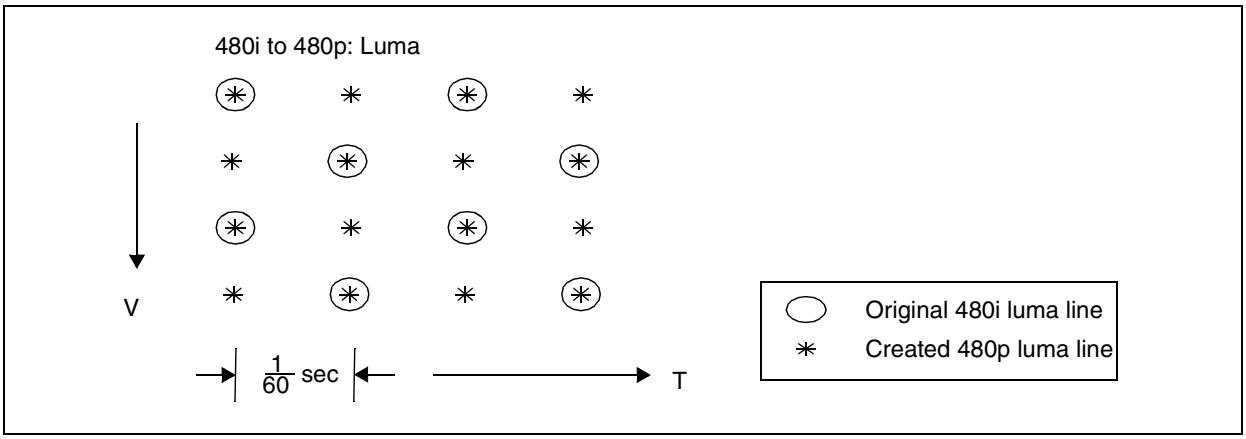
The LMU uses a motion-adaptive, spatial/temporal de-interlacing algorithm controlled by external registers. The luma and chroma blocks are independent but share LMU_BPPL and LMU_VINL register values.

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 177 illustrates the vertical/temporal relationship of lines before and after the luma de-interlacing process.

Figure 177: 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.

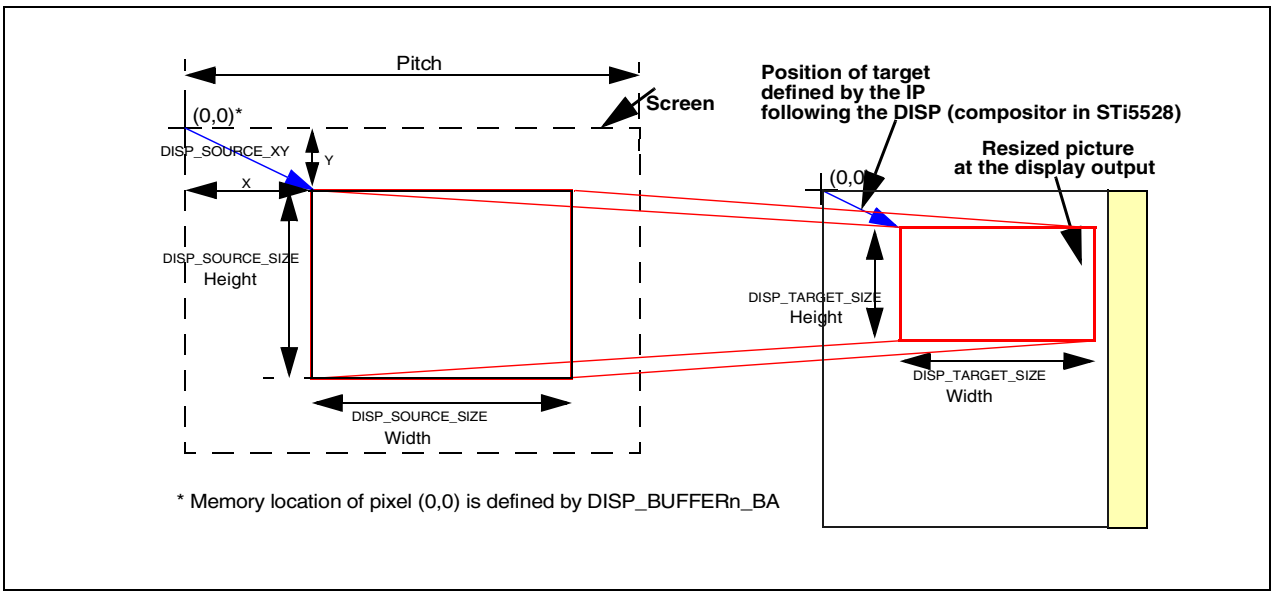
The LMU can be activated independently for luma and chroma.

62.6 Functional description

62.6.1 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 4:2:2R input format is used, and line length in pixel unit when 4:2:0MB or 4:2:2MB input format is used. By changing pitch value, field or frame filtering is done in vertical SRC when 4:2:2R 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 178 and Chapter 62: *Main and auxiliary-display processors on page 655*). This is the only part of picture read from external memory and is considered as 'source' in the following sections.

Figure 178: Source and target definition



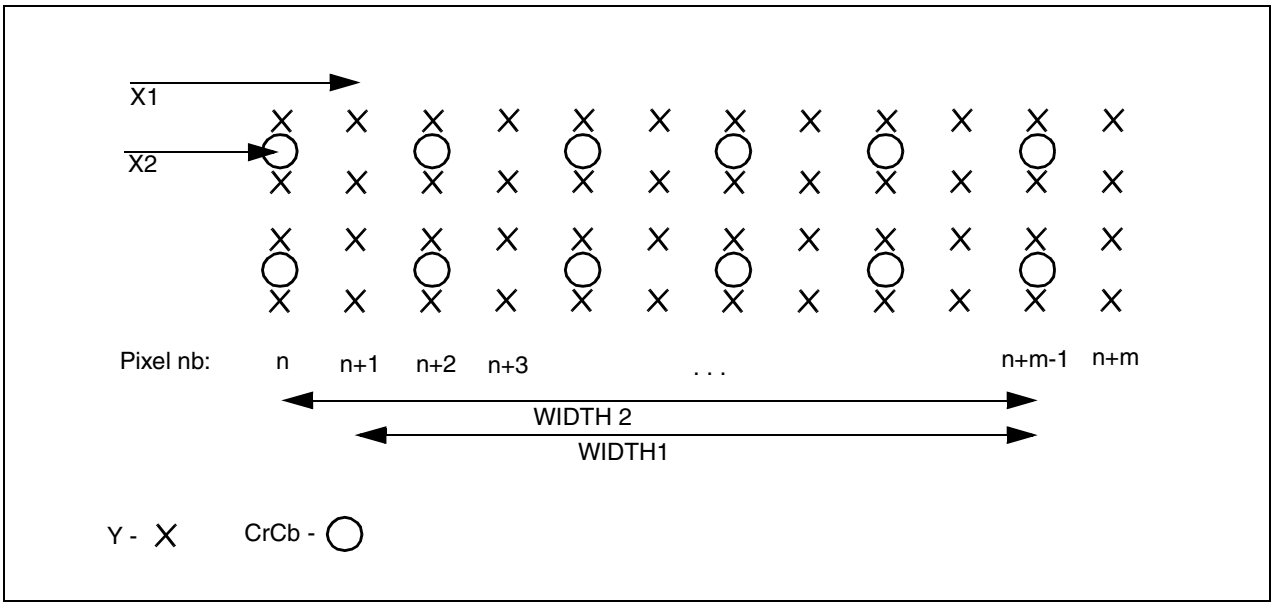
When the 4:2:2 R input format is used, the source picture position is defined by DISP_LUMA_XY.X1 and .Y1, and size by DISP_LUMA_SIZE.WIDTH1 and .HEIGHT1. Both position and size are given with pixel precision. This means that for odd values of X1 and/or (X1+WIDTH1), pixels with only luma information are pointed (see Figure 179). 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 179, 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 179, 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 179: 4:2:2 input format (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 180 shows an example of this.

Figure 180: 4:2:0 input format (example)

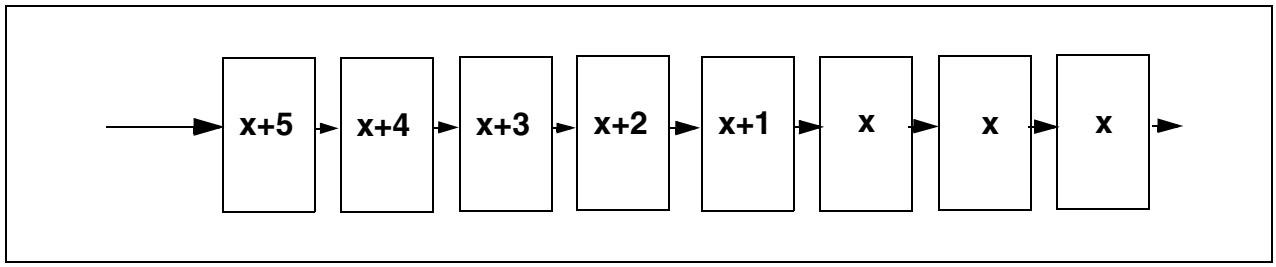


62.6.2 Processing of first and last pixels

In order to generate a clean picture on vertical and horizontal borders, some special configurable processing is necessary. If the first (or last) pixel of source picture to be resized is not the first (or last) pixel of a 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 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 will be loaded in filters (accessed from memory). DISP_LUMA(CHR)_V(H)SRC bits FYPR, FYLR, FCPR and FCLR define how many time first/last pixel/line of that picture is

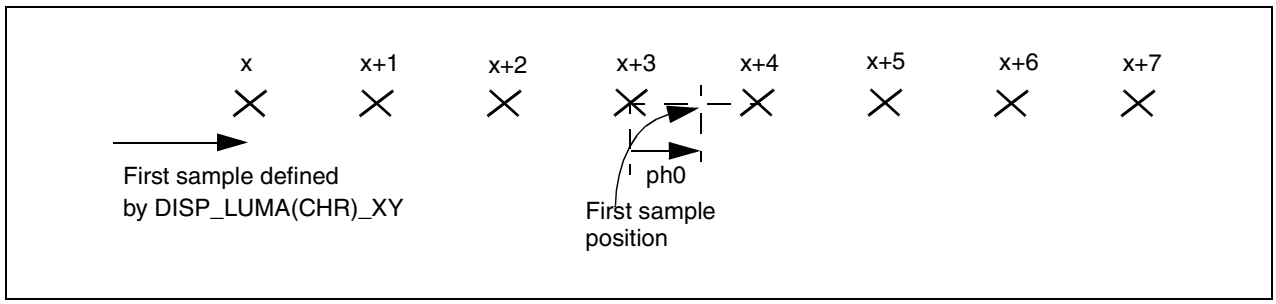
repeated in horizontal/vertical SRC filter. On the next figure an example of initialization of horizontal filter is shown where first sample is repeated two times.

Figure 181: Initiations of horizontal SRC pipeline example (first sample repeated twice)



Horizontal and vertical position of source picture can be defined with accuracy of 1/32 of pixel/sample. It is defined by initial phase of polyphase dto which is 13-bit register (DISP_LUMA/C_HSRC) which gives 1/2^13 accuracy but it is internally rounded to 1/32. On the next figure an example of horizontal pan and scan is shown where FI(c)pr bits defines 0 repeat of the first sample and where initial phase (ph0) is different from 0. In Figure 182, the initial sample is between samples (x+1) and (x+2).

Figure 182: Horizontal pan and scan example

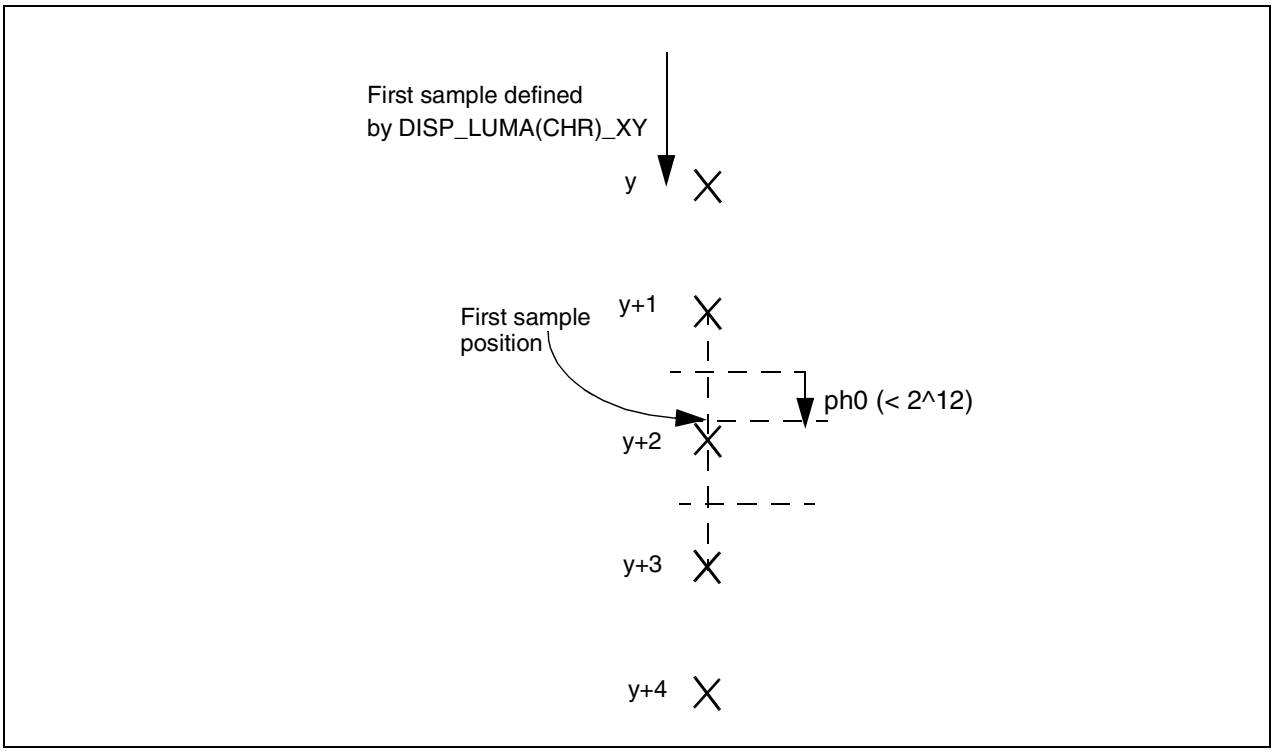


If initial phase is 0 then the first sample is (x+3) in previous example and (x+1) in example of Figure 182.

Vertical pan and scan is defined in the similar manner taking into account that the number of taps is 5 (odd number). In the case where FI(c)Ir 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 183).

Figure 183: Vertical pan and scan example

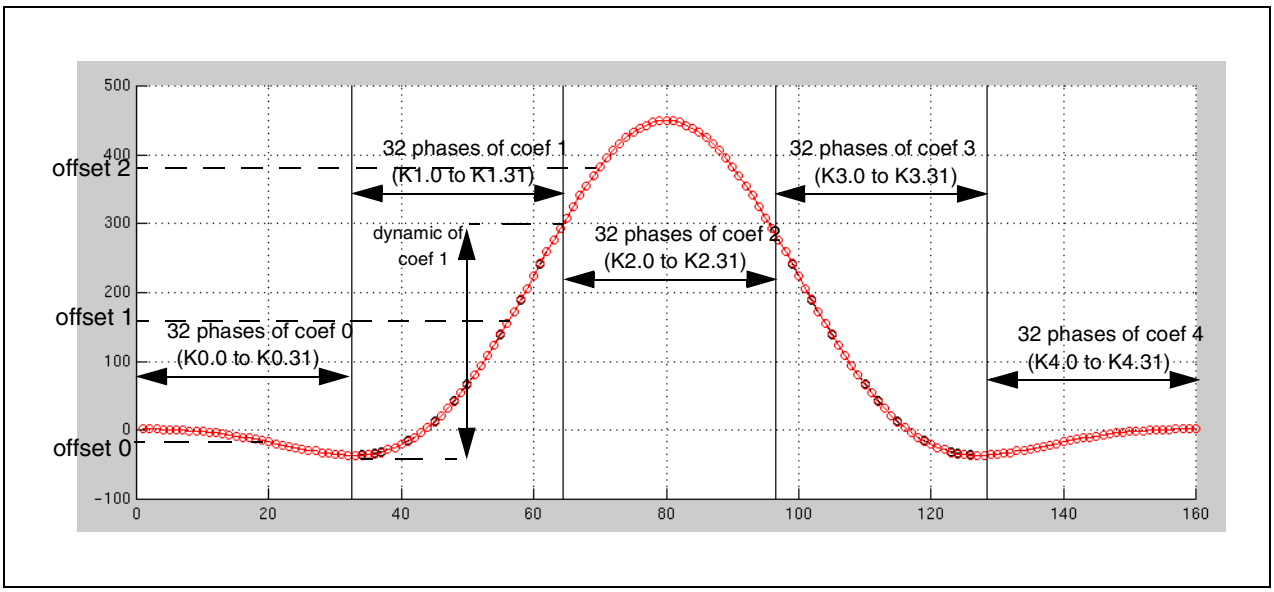


62.6.3 Programming of filter coefficients

An example of five tap polyphase filter (VSRC) set of coefficients is shown on next figure. 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 8-bit value is multiplied by 2 or not to cover the dynamic of 32 phases (case of coef1 set of 32 phases in above example).

Figure 184: Example: five tap polyphase filter (VSRC) set of coefficients



62.6.4 Programming of source/target size and increment

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 horizontal chroma filter increment should be a half of 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 (16 samples of chroma).

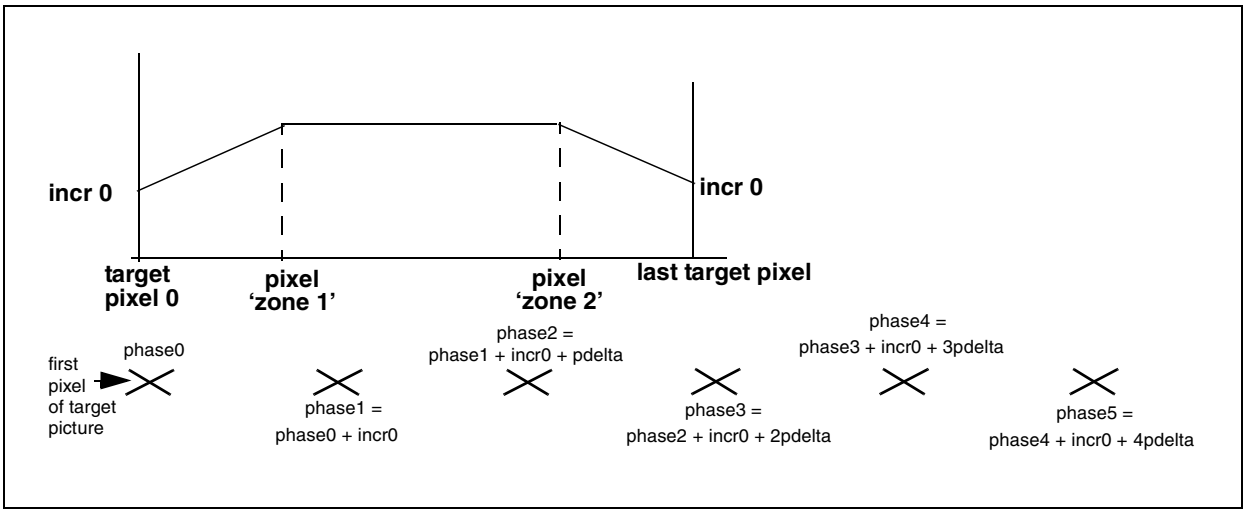
The increment value and initial phase are used to generate pixels at the output. The DISP_TARGET_SIZE register determines the number of pixels/lines generated. This means, if the increment does not respect the ratio between target and source size, the generation of pixels will be stopped after target 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.

62.6.5 Nonlinear zoom

A typical application of nonlinear zoom consists of displaying 4:3 picture on 16:9 screen with minimum deformation in the central part of the screen and greater deformation at 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 185.

The first generated pixel uses *phase0* to find the corresponding coefficients, the second one uses *phase0* + *incr0*, third one uses *phase1* + *incr0* + *pdelta* and so on. *incr0* and *phase0* are the increment and initial phase as defined by DISP_LUMA_HSRC and DISP_C_HSRC registers (defined for non linear zoom) and *pdelta* is 'the increment of the increment' defined by DISP_PDELTA register. Zones where nonlinear zoom is applied are defined by pixel/samples zone1 and zone2 as shown. This means that the increment is incremented from pixel/sample 0 to pixel/sample zone1 and decremented from pixel/sample zone2 to the last pixel/sample of the target picture. zone1 and zone2 are defined by DISP_NLZZD_Y independently for luma and chroma HSRC.

Figure 185: Increment function for nonlinear zoom



63 Main and auxiliary-display processors 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.

Addresses are provided as *MainDisplayBaseAddress* + offset, *AuxDisplayBaseAddress* + offset or *LMUBaseAddress* + offset.

The *MainDisplayBaseAddress* is:

0x3820 2000.

The *AuxDisplayBaseAddress* is:

0x3802 3000.

The *LMUBaseAddress* is:

0x3820 4000.

Table 161: Main and auxiliary display processor register summary

Register	Description	Offset	Type
DISP_CTL	DISP control	0x000	R/W
DISP_LUMA_HSRC	Luma horizontal sample rate converter FSM control	0x004	R/W
DISP_LUMA_VSRC	Luma vertical sample rate converter FSM control	0x008	R/W
DISP_CHR_HSRC	Chroma horizontal sample rate converter FSM control	0x00C	R/W
DISP_CHR_VSRC	Chroma vertical sample rate converter FSM control	0x010	R/W
DISP_TARGET_SIZE	Target size in number of pixels/number of lines	0x014	R/W
DISP_NLZZD_Y	Nonlinear zoom zone definition for luma HSRC	0x018	R/W
DISP_NLZZD_C	Nonlinear zoom zone definition for chroma HSRC	0x01C	R/W
DISP_PDELTA	Nonlinear zoom increment step definition	0x020	R/W
Reserved	-	0x024 - 0x078	-
DISP_STATUS	Status	0x07C	RO
DISP_MA_CTL	Memory access control	0x080	R/W
DISP_LUMA_BA	Luma buffer base address (address of first luma pixel if 4:2:0 MB format or first Cb/Cr address if 4:2:2R format at the input)	0x084	R/W
DISP_CHR_BA	Chroma buffer base address (address of first chroma couple CbCr if 4:2:0MB or 4:2:2MB input format is used)	0x088	R/W
DISP_PMP	Pixmap memory pitch	0x08C	R/W
DISP_LUMA_XY	Luma first pixel source position (buffer 1)	0x090	R/W
DISP_CHR_XY	Chroma first pixel source position (buffer 2)	0x094	R/W
DISP_LUMA_SIZE	Source pixmap size (Luma)	0x098	R/W
DISP_CHR_SIZE	Source pixmap size (Chroma)	0x09C	R/W
DISP_HFP	Horizontal filter coefficients pointer	0x0A0	R/W
DISP_VFP	Vertical filter coefficients pointer	0x0A4	R/W
Reserved	-	0x0A8 - 0x0F8	-

Table 161: Main and auxiliary display processor register summary

Register	Description	Offset	Type
DISP_PKZ	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

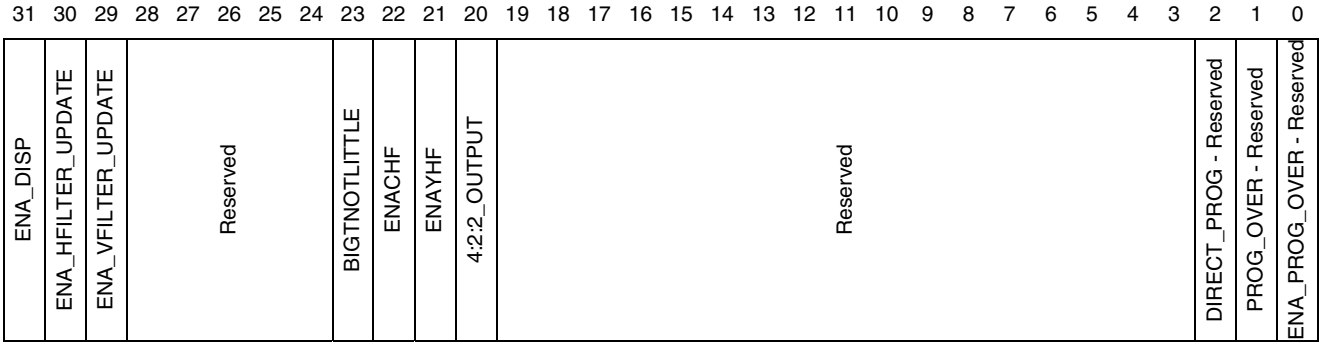
Register	Description	Offset	Type
LMU_CTRL	LMU control	0x00	R/W
LMU_LMP	LMU luma memory pointer	0x04	R/W
LMU_CMP	LMU chroma memory pointer	0x08	R/W
LMU_BPPL	LMU input picture horizontal size	0x0C	R/W
LMU_CFG	LMU configuration	0x10	R/W
LMU_VINL	LMU input picture vertical size	0x14	R/W
LMU_MRS	LMU minimum STBus request	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_ITS	LMU interrupt status	0x28	RO
LMU_AFD	LMU accumulated field difference	0x2C	R/W

63.1 Registers description

63.1.1 Main and auxillary display registers

The following registers are duplicated for both main and auxillary display processors. Their addresses are described as *DisplayBaseAddress* + offset, where *DisplayBaseAddress* is actually either *MainDisplayBaseAddress* or *AuxDisplayBaseAddress.*

DISP_CTL DISP control



Address: *DisplayBaseAddress* + 0x000

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP control register provides the operating mode of the DISP pipe, for the current viewport display.

[31] **ENA_DISP**: DISP bloc enable[a]
0: DISP is disabled (no output is generated)
1: DISP is enabled

[30] **Ena_HFilter_Update:**
0: The coefficients for the H filters are not loaded
1: The coefficients for the H filters are updated from memory on after next vsync
This bit is autocleared after it is taken into account (on next vsync)

[29] **Ena_VFilter_Update**:
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] **EnaCHF**
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] **EnaYHF**
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_Output**
0: 4:4:4 output format (luma and chroma output one cycle out of two of pix2x clock)
1: 4:2:2 output format (chroma output one cycle out of four of pix2x clock)
Not implemented

[19:3] **Reserved**

[2] **DIRECT_PROG - Reserved**
All user registers are double buffered and when this bit is '1', the programmed values in all registers are taken by second register layer as soon as written in first register layer (without waiting for next vsync).

[1] **PROG_OVER - Reserved**
All user registers are double buffered and when this bit is 1 (and ENA_PROG_OVER is 1) then on next vsync all new values of user registers are taken into account and Prog Over is automatically cleared. ENA_PROG_OVER and PROG_OVER bits are not double buffered.

[0] **ENA_PROG_OVER - Reserved**
0: Action of Prog Over bit is disabled
1: Action of Prog Over bit is enabled

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 4:2:2R 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

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

Address: *DisplayBaseAddress* + 0x004
Type: R/W
Buffer: Double-bank, automatic hardware toggle
Reset: 0
Description: The DISP horizontal luma sample rate converter's parameters register provides the configuration for the horizontal sample rate conversion of luma samples.

[31:29] **Fypr**
First pixel is repeated 'Fypr' (First luma pixel repeat) 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_Initial_Phase**
The horizontal sample rate converter state-machine initial phase, 0.13 format.

[15:0] **HSRC_Increment**
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

31	30	29	28	27	26	25	24	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	FYLR															VSRC_INITIAL_PHASE															
																VSRC_INCREMENT															

Address: *DisplayBaseAddress* + 0x008

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP Luma Vertical Sample Rate Converter's parameters register provides the configuration for the vertical sample rate converter.

[31] **Reserved**

[30:29] **FYLR**
First line is repeated 'Fylr' (First luma line repeat) 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_Initial_phase**
The vertical sample rate converter state-machine initial phase, 0.13 format.

[15:0] **VSRC_Increment**
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 will be taken as 100.0000 0000 0000 0

DISP_CHR_HSRC DISP chroma horizontal source FSM 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
FCPR			HSRC_INITIAL_PHASE													HSRC_INCREMENT															

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.[1]

[31:29] **Fcpr**
First chroma pixel is repeated 'Fcpr' (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_Initial_phase**
The horizontal sample rate converter state-machine initial phase, 0.13 format

[15:0] **HSRC_Increment**
The horizontal sample rate converter state-machine increment, in 3.13 format. Maximum value is 010.0000 0000 0000 0. Any greater value will be taken as 010.0000 0000 0000 0

1. If 4:2:2R input format is set, VSRC Increment, VSRC Initial Phase and Fcpr must be programmed as luma (DISP_CHR_VSRC and DISP_LUMA_VSRC must have the same value)

DISP_CHR_VSRC DISP chroma vertical source 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	FCLR	VSRC_INITIAL_PHASE														VSRC_INCREMENT															

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.

[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_INITIAL_PHASE**
The vertical sample rate converter state-machine initial phase, 0.13

[15:0] **VSRC_INCREMENT**
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 will be taken as 100.0 0000 0000 0000

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: The DISP Target Pixmap Size register 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: The DISP nonlinear zoom zone definition register provides the borders of nonlinear zoom in the target picture for luma in pixel units; see Section 62.6.5: *Nonlinear zoom on page 663*.

[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: The DISP nonlinear zoom zone definition register provides the borders of nonlinear zoom in the target picture for chroma in output chroma sample units; see Section 62.6.5: *Nonlinear zoom on page 663*.

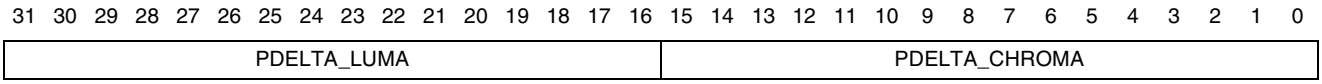
[31:27] **Reserved**

[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



Address: *DisplayBaseAddress* + 0x020

Type: R/W

Buffer: Double-bank, automatic hardware toggle

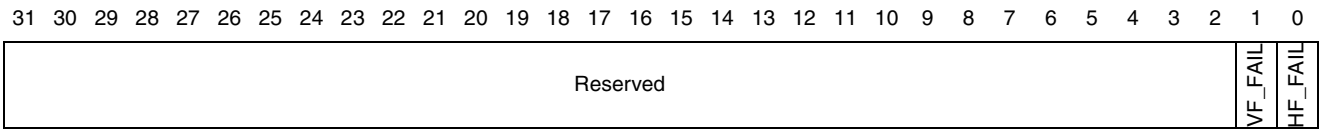
Reset: 0

Description: The DISP nonlinear zoom zone - increment step definition register provides the steps used to increment luma and chroma increments in HSRCs for nonlinear zooms; see Section 62.6.5: *Nonlinear zoom* *on page 663*. 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 chroma_zone1] and decremented within the range [pixel luma_zone2, last pixel]. pixel0, and last pixel corresponds 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 corresponds to first and last chroma sample of target picture.

DISP_STATUS DISP status



Address: *DisplayBaseAddress* + 0x07C

Type: RO

Buffer: Single-bank

Reset: 0

Description: The **DISP Status** provides the information about internal states of the DISP.

[31:2] **Reserved**

[1] **VF_fail**
When the VF requests the data from input FIFO (before delay lines of VF) and it is empty, this bit goes to '1'. This bit is automatically cleared after it is read.

[0] **HF_fail**
When the HF requests the data from delay line (between VF and HF) and it is not able to provide them (VF has not yet finished to fill the requested line into the delay line), this bit goes to '1'. This bit is automatically cleared after it is read.

DISP_MA_CTL 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_SPACE_BETWEEN_REQS										PIX_LOAD_LINE					COEF_LOAD_LINE					MB_FIELDY	INPUT_FORMAT				

Address: *DisplayBaseAddress* + 0x080

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The **DISP MA control** register provides memory access control of the DISP.

[31:27] **Reserved**

[26] **MB_FIELDC**
Chroma access mode in macro-block organized frame buffers (YCbCr420MB and YCbCr422MB format)
0: Access in frame mode
1: Access in field mode (every other line)

[25:16] **MIN_SPACE_BETWEEN_REQS**
Defines the minimum number of STBus cycles between two memory messages (bursts).[a]
The number of cycles between two messages will be max('Min Space Between Reqs' + 1, message duration)

[15:11] **PIX_LOAD_LINE**
Defines how many lines the STBus plug waits after VSYNC before sending the first request in the field to load the display pipe. Number of lines is 2**n* where *n* is the register value. When changing the coefficient set in VSRC and HSRC this request should be sent after new coefficient set is loaded, that is, 2**n* > Coef Load Line (see above). If this is not a 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 Ena V/HFilter update 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 (YCbCr420MB and YCbCr422MB format)
0: Access in frame mode
1: Access in field mode (every other line)

[4:0] **INPUT_FORMAT**
10010: YCbCr4:2:2R 0x12
10011: YCbCr4:2:0MB (or 4:2:2MB) 0x13
10100: YCbCr4:2:0MB (or 4:2:2MB) 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 us) which means 4*45 requests of 128 bit word of luma (12 messages) should be done and there is 64 us/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 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

31	30	29	28	27	26	25	24	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_NUMBER						PIXMAP_MEMORY_POINTER_(LUMA)																									

Address: *DisplayBaseAddress* + 0x084

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP Luma Pixmap Memory Location gives the location of first Y of the FRAME in 4:2:0MB and 4:2:2MB mode or first Cb or Cr of the field in 4:2:2R 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 4:2:0MB and 4:2:2MB input format is used or first Cb or Cr in the field when 4:2:2 R input format is used. In 4:2:2R 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:XMB 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_NUMBER**
64 MByte bank number

[25:0] **PIXMAP_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

31	30	29	28	27	26	25	24	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_NUMBER						PIXMAP_MEMORY_POINTER_(CHROMA)																									

Address: *DisplayBaseAddress* + 0x088

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: This register 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_NUMBER**: 64 MByte bank number

[25:0] **Pixmap_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_VALUE															

Address: *DisplayBaseAddress* + 0x08C

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP pixmap memory pitch register contains the memory pitch for the displayed pixmap, as stored in the memory.

Note: The pitch is the distance inside the memory, in byte unit, between two vertically adjacent samples.

[31:16] **Reserved**

[15:0] **Pitch_value**
Memory pitch of luma source pixmap in 4:2:0MB and 4:2:2MB format or of YCrCb in 4:2:2R format, in byte unit

In 4:2:0MB and 4:2:2MB 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). (4 LSBs are always at '0')

In 4:2:2R format the pitch value is always used as it is (in field access it defines the distance in byte units 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 4 bytes. (2 LSBs are always at '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 4:2:2R input format is used or first luma sample when 4:2:0MB or 4:2:2MB 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 178: *Source and target definition on page 658*).

[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

31	30	29	28	27	26	25	24	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					Y2											Reserved					X2										

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)

31	30	29	28	27	26	25	24	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					HEIGHT1											Reserved					WIDTH1										

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 line unit, 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_NUMBER						H_FILTER_POINTER																						Reserved			

Address: *DisplayBaseAddress* + 0x0A0

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP horizontal filter pointer register is a 32-bit register containing a memory pointer to the set of filter coefficients that must be used for the horizontal sample rate converters (Luma and Chroma).

The coefficients will be loaded only if DISP_CTL[30]= 1 (Ena HFilter Update) after the next vertical synchronization pulse.

[31:26] **64MB_BANK_NUMBER**: 64 MByte bank number

[25:4] **H_Filter_pointer**
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 when 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 35 32-bit words are chroma coefficients for HSRC.

[3:0] **Reserved**

DISP_VFP DISP vertical 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_NUMBER						V_FILTER_POINTER																						Reserved			

Address: *DisplayBaseAddress* + 0x0A4

Type: R/W

Buffer: Double-bank, automatic hardware toggle

Reset: 0

Description: The DISP Vertical Filter Pointer register is a 32-bit register containing a memory pointer to the set of filter coefficients that must be used for the vertical sample rate converters (Luma and chroma).

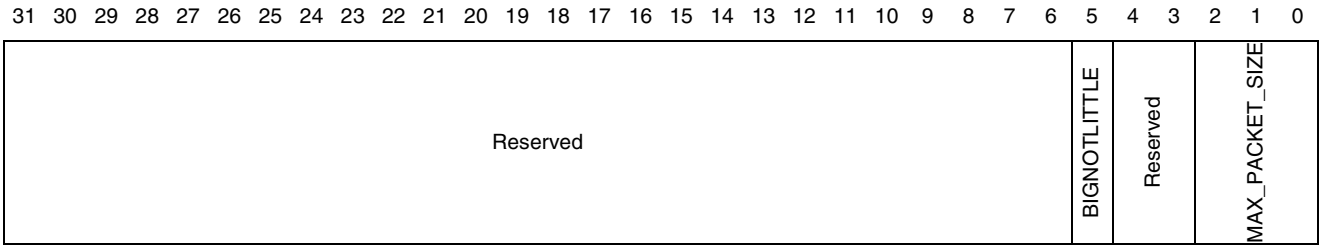
The coefficients will be loaded only if DISP_CTL[29] = 1 (Ena VFilter Update) after next vertical synchronization pulse.

[31:26] **64MB_BANK_NUMBER**: 64 MByte bank number

[25:4] **V_Filter_pointer**
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 (22*2 32-bit words that must be fully contained in the specified bank). First 22 32-bit words are luma coefficients, then next 22 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_PACKET_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 1and 6	Offset K1 and K6	Reserved / Div Factor / Shift 0 and 7	Offset K0 and K7
0x88	Reserved / Shift 3 and 4	Offset K3 and K4	Reserved / Shift 2 and 5	Offset K2 and 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 supposition 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:
Output of the filter is divided by:
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 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 0x84 and 0x88 - '***Offset X and Y***').

At the end, in order to have the sum of coefficients equal to 1 for each supposition (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:

$$\left(K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(8 + Div\overline{factor})}} \right) cd$$

The coefficients are loaded only if DISP_CTL[30] = 1 (Ena HFilter Update).

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 and 3 / Offset K1 and K3	Shift 0 and 4 / Offset K0 and 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 supposition 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

Output of the filter is divided by:

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 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 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 supposition (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:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(8 + Divfactor)}}$$

The coefficients are loaded only if DISP_CTL[29] = 1 (Ena VFilter Update).

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 1and 6	Offset K1 and K6	Reserved / Div Factor / Shift 0 and 7	Offset K0 and K7
0x88	Reserved / Shift 3 and 4	Offset K3 and K4	Reserved / Shift 2 and 5	Offset K2 and 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 supposition 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
Output of the filter is divided by:
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 supposition (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:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(8 + Div\overline{factor})}}$$

The coefficients are loaded only if DISP_CTL[30] = 1 (Ena HFilter Update).

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 and 3 / Offset K1 and K3	Shift 0 and 4 / Offset K0 and 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 supposition 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

Output of the filter is divided by:

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 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 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 supposition (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.

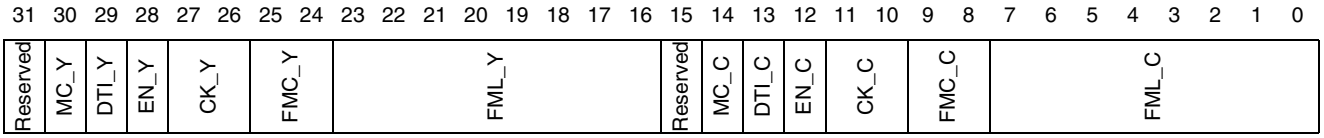
To summarize, the value of coefficients internally used by the filter is given by:

$$K = \frac{K_{reg} \cdot 2^{Shift} + Offset}{2^{(8 + Divfactor)}}$$

The coefficients are loaded only if DISP_CTL[29] = 1 (Ena VFilter Update).

63.1.2 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] **EN_Y**:
When set to 1, the luma LMU is enabled; when set to 0, the luma LMU is disabled

[27:26] **CK_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. The 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] **EN_C**: When set to 1, the chroma LMU is enabled; when set to 0, the chroma LMU is disabled

[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)
01: divide by 2
10: divide by 3
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

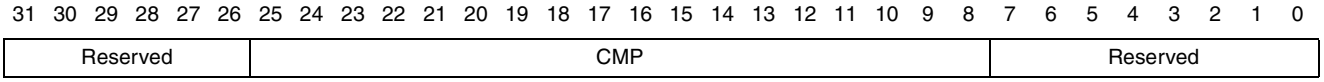
[7:0] **FML_C**: The LMU's film-mode detection algorithm compares the current field to the same field in the previous frame. The comparison starts with line 2 of the chroma field and continues until the line number programmed in FML_C. A separate comparison is made for luma and chroma portions of the 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
Reserved						LMP																		Reserved							

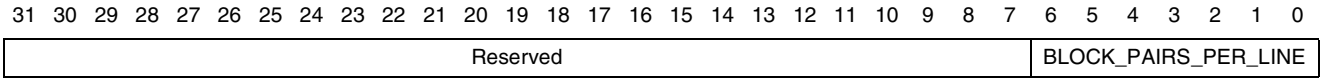
Address: *LMUBaseAddress* + 0x04
Type: R/W
Buffer: Double Buffered: Vsync
Reset: 0
Description: This register points to the start of the memory LMU luma buffer, in units of 256 bytes.

LMU_CMP LMU chroma buffer start pointer



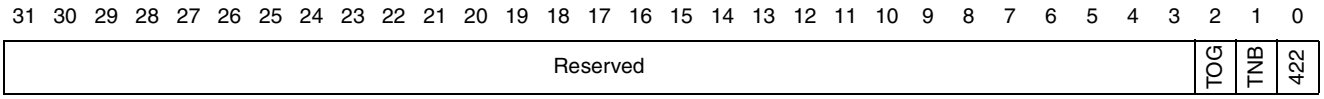
Address: *LMUBaseAddress* + 0x08
Type: R/W
Buffer: Double Buffered: Vsync
Reset: 0
Description: This register points to the start of the memory LMU chroma buffer, in units of 256 bytes.

LMU_BPPL Number of block-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



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 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 multiple of 4 for interlace inputs.

For 4:2:2 inputs, it must be programmed with a value that is multiple of 2 for interlace inputs.

This register is unsigned.

LMU_MRS 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 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 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: 0xXXXXXXXX
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 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_ITS 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: 0xXXXXXXXX
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 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 the received pictures are from 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 DTI_Y of register LMU_CTRL is set to 1, then field DIFF_Y of LMU_AFD is invalid. Similarly, if bit DTI_C of LMU_CTRL is set to 1, then field DIFF_C of LMU_AFD is invalid.

64 High-definition multimedia interface (HDMI)

64.1 Glossary

HDMI	High Definition Multi-media Interface The High-Definition Multimedia Interface 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

64.2 Conventions and Notations

The convention for representing block of *n* bits is bit no 1 to left (MSB) and bit *n* to right (LSB)

HIGH means logic ‘1’ and LOW means logic’0’

Signal active means HIGH unless explicitly specified

Interrupt is asserted when interrupt pin is logic HIGH

The default value read or written into the registers is ‘0’.

64.3 System Description

64.3.1 Brief overview

The HDMI block transmits HDMI-compatible data. It consists primarily of two blocks:

- Analog IP: responsible for serializing the data and transmitting of the differential line
- Digital IP: responsible for frame formatting and encoding.

The Video data is fed to the HDMI IP on three 8 bit buses. The video data is expected to be synchronous to the pix clock. The Hsync and Vsync pulses (synchronous to pix clock) are also fed. The video interface supports **RGB** and **YCbCr 4:4:4** formats only. HDCP IP is expected to be placed external to this IP. The encryption keys generated by the HDCP block are fed to the HDMI on a separate interface. The HDMI just **XORs the data with the keys** generated by the HDCP. If the HDMI is in not authuntication, a default (programmed) value is sent during the video data interval periods. Audio data is **not sent** at all, null packets ands AVI info frames are sent instead.

The HDMI IP has an SPDIF interface, through which the audio data is streamed in. The HDMI IP does not perform compression or de-compression of the audio data. It just inserts the received audio data into the appropriate fields in the HDMI frame. The audio interface performs the Cycle Time Stamp (CTS) calculations and inserts the CTS in the frame appropriately.

STBus type1 interface is provided to configure the HDMI IP. The data for info frame being sent during the data island period of the HDMI frame is also written through this interface.

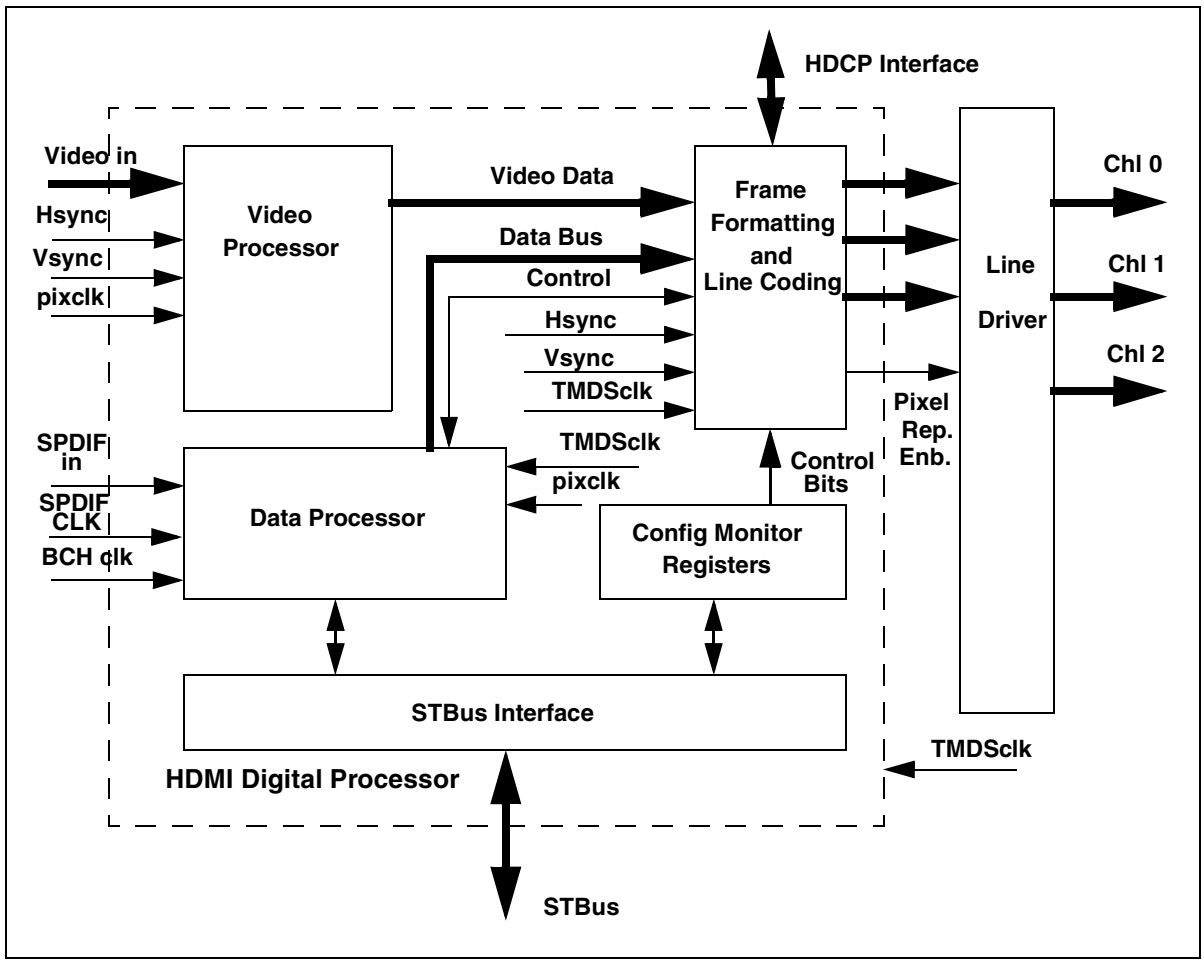
It is expected that the control data is appropriately inserted depending on the configuration of the IP.

The HDMI frame formatter interfaces to the line driver on three 10-bit parallel buses, which are synchronous with the TMDS clock.

64.4 Block Diagram

The block diagram for the HDMI IP is shown in the figure below.

Figure 186: Block diagram of HDMI



The HDMI interface is provided by two blocks: HDMI frame formatter and the line driver.

This document specifies the HDMI frame formatter. It interfaces to line driver on 3 - 10 bit buses. Each bus carries the Transition Minimized Differential signalling (TMDS) data. The TMDS data changes with every rising edge of the TMDS clock. The TMDS clock is n*pixclock where n =1 or 2. The TMDS clock and the pix clock are expected to be synchronous. BCH clock is m* TMDS clock where m = 2 or 4. The BCH clock is used to run BCH encoding blocks of the data packets.

The HDMI frame formatter accepts the video data synchronous to the pix clock (on every rising edge), Audio data is fed through the SPDIF interface. The STBus type1 interface is used to configure and monitor the 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.

The functionality of the different sub-modules housed inside is documented in the following sections.

The following conditions have to be met while generating pix clock, TMDS clock and BCH clock.

- TMDS clock is equal to pix clock or is twice the pix clock

- BCH clock is twice or fourtimes the TMDS clock
- Pix clock, TMDS and BCH clocks are phase aligned.

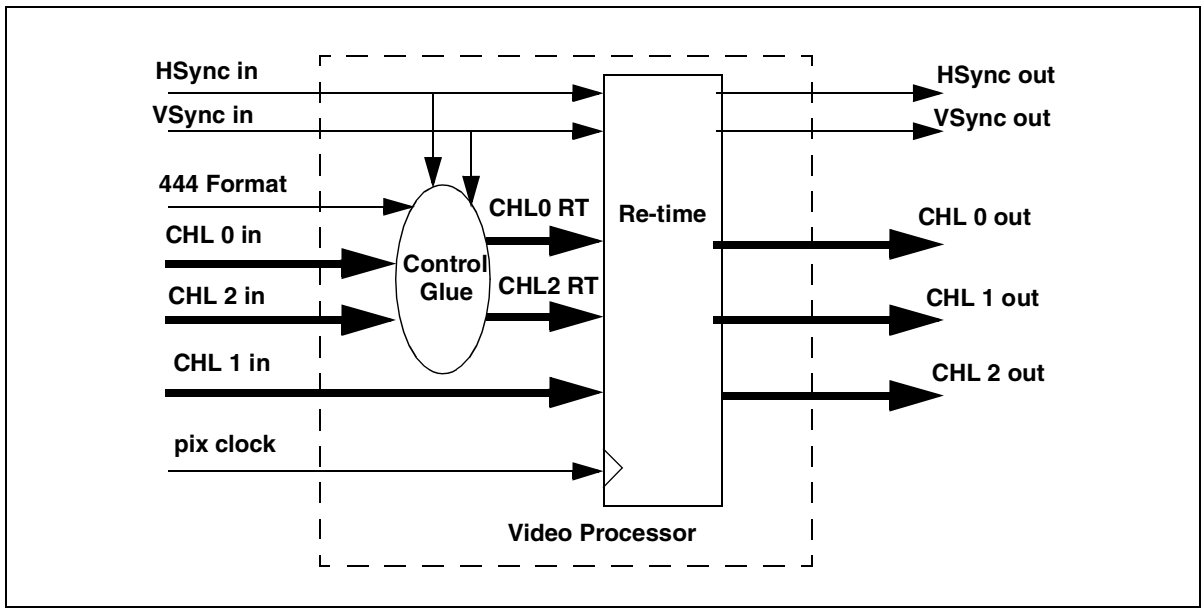
64.4.1 Config Monitor registers

This block houses the different configuration and monitor registers. A detailed description on the registers housed inside is documented in section 6.

64.4.2 Video Processor

The video processor block is responsible to format the video data into the 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 implementation of the video processor is shown in the figure below.

Figure 187: Block diagram of the video processor



It is assumed that the first pixel is the one when Hsync and Vsync are active and a rising clock edge of pix clock. The first pixel of the active video window is inferred from the contents of registers HDMI_ACTIVE_XMIN and HDMI_ACTIVE_YMIN. The maximum number of pixels per line and maximum number of lines per frame are inferred by the values in the registers HDMI_ACTIVE_XMAX and HDMI_ACTIVE_YMAX.

The re-timed Hsync and Vsync (HSync out and VSync out) have to be used by the other 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 the CHL0 in to CHL0 RT and CHL1 in to CHL1 RT.

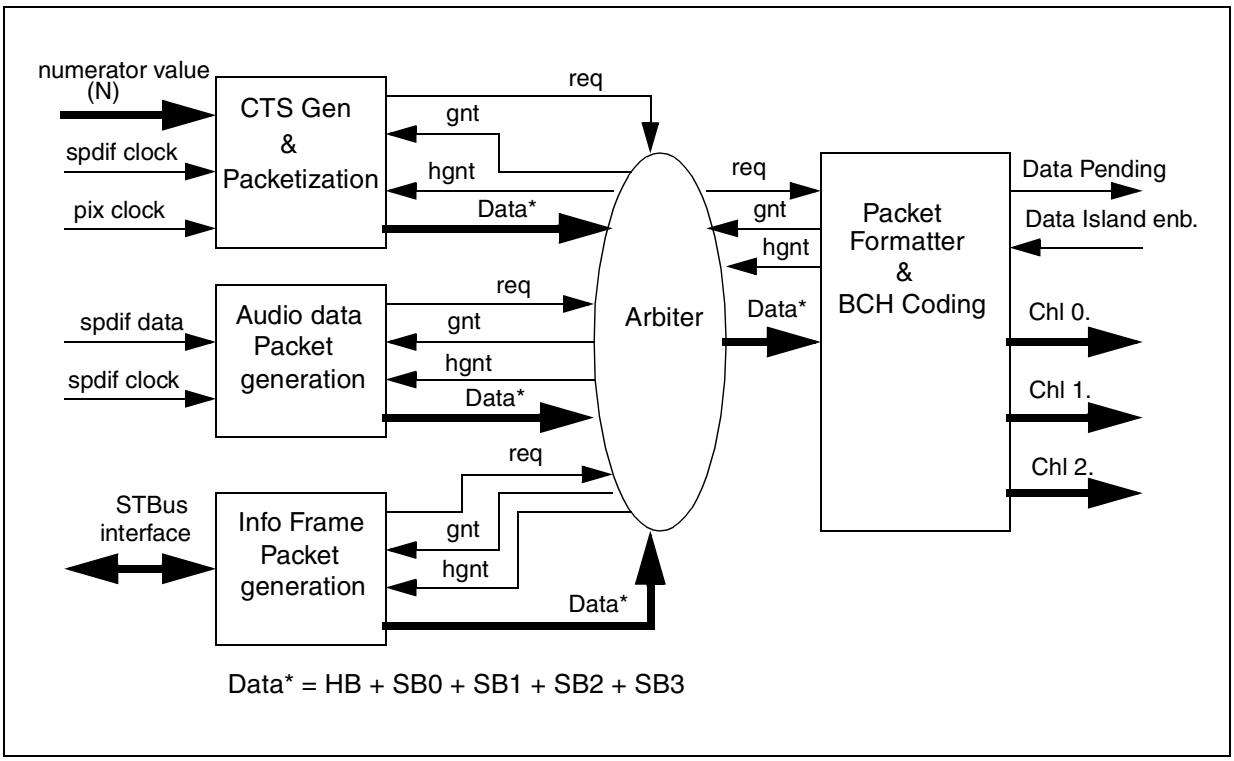
64.4.3 Data Processor

The data processor is responsible to format the data packet and transmit a data packet during the data island period. If there is no valid data, the data processor transmits a null packet. The data processor is expected to transmit the following data in the order specified (subject to the availability of the corresponding data packet) during the data island period.

- Audio Sample clock capture packet
- Audio data packet
- General control Packet
- Info frame packet
- Null packet.

The block diagram of the data processor is shown in the figure below.

Figure 188: Data Processor

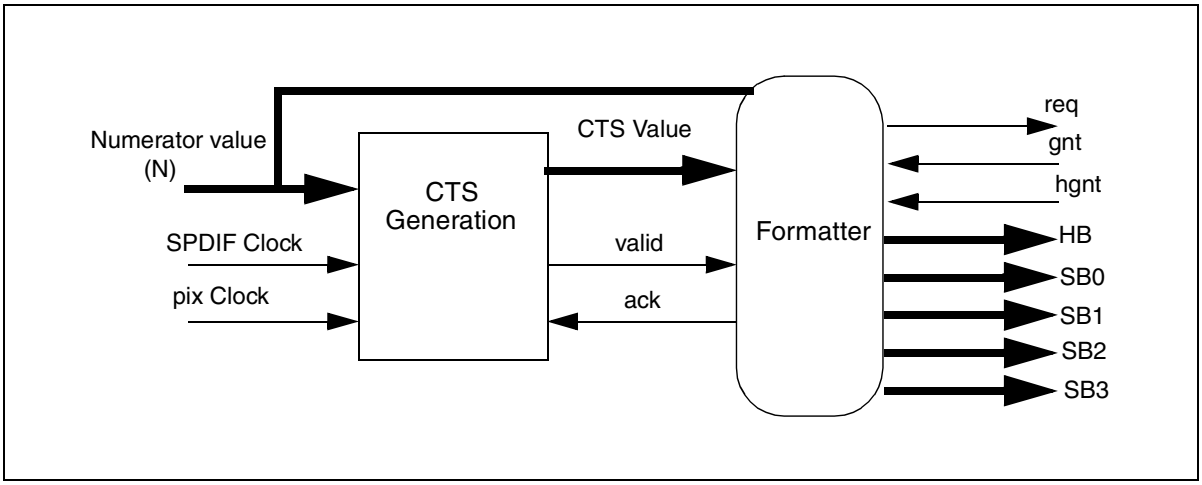


Null Packets are sent in the data field when the encryption is enabled and HDMI is in not authenticated.

64.4.3.1 Audio Sample clock capture

The CTS value generated by the HDMI source is used by the HDMI sync to re-generate the audio sample clock. The CTS is generated according to the equation $128*F_S$ = pixel clock*N/CTS, where F_S is the audio sample clock. In other words the CTS is generated by counting the number of pix clocks in 128*Fs/N duration. This can be implemented using simple counters. The CTS counter has to be a 20bit counter and is initialized (to zero) after every $128*F_S/N$ clocks. **The 128*Fs and pix clock have to be treated as asynchronous with respect to each other** and appropriate care is to be taken during synchronization.

Figure 189: CTS packet generation



64.4.3.2 Audio data packetization

HDMI supports two to eight audio channels that are input on one to four SPDIF inputs. Each of the SPDIF stream is processed by a separate SPDIF processing block. There are four such blocks to process four SPDIF streams. The data from each of the blocks is fed to the formatter which formats the data appropriately.

It is assumed that **each of the four SPDIF streams are synchronous to the SPDIF clock**.

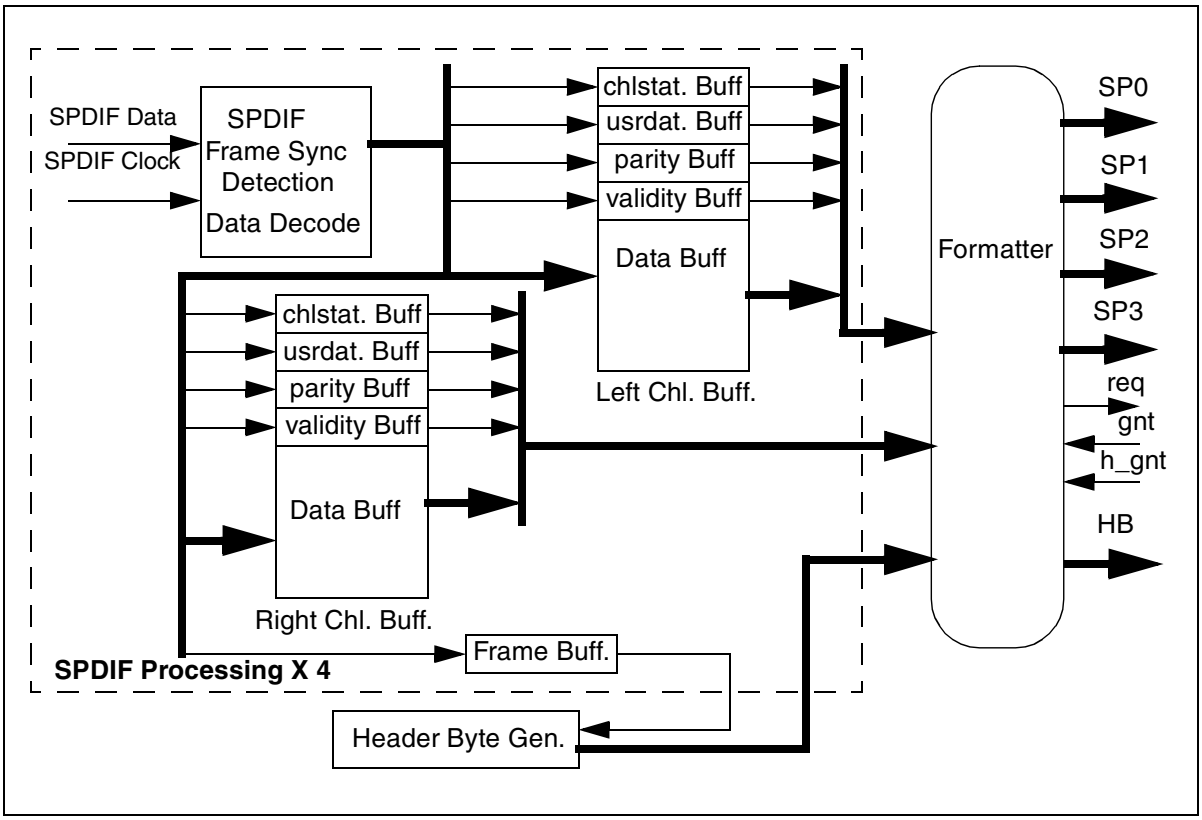
The SPDIF processing block consists of a SPDIF Frame sync detection and data decode block, FIFOs to store the received data, validity, parity, user data and channel status bits for both the left and right channels. It also has a single bit FIFO which is used to store the information regarding the first frame detection. A depth of 16 for each of FIFOs is sufficient. It is expected that the SPDIF sync detection, data decoding and loading of the decoded data is done synchronous to the SPDIF clock. **It is assumed the SDPIF clock is twice the SPDIF bit-rate**. This selection is done to ease the implementation of the sync detection and data decoding. The reading of the data is done by the TMDS clock. The TMDS clock and SPDIF clock are assumed to be asynchronous with each other. To ensure that the write pointer of the FIFO is not corrupted when mapped from SPDIF to TMDS clock domain, the write pointer is grey coded. Grey coding is documented in appendix C. It is also **mandatory** to double re-time the write pointer in the TMDS clock domain.

In addition to the SPDIF processing block, the audio data paketization also has an header byte generation block. This block is responsible for generation of the header bytes for the audio packet.

That the delay introduced by the formatter for inserting the SPDIF data into the HDMI frame should not exceed 1millisecond.

Different blocks housed inside the SPDIF processing block are described in the following sections.

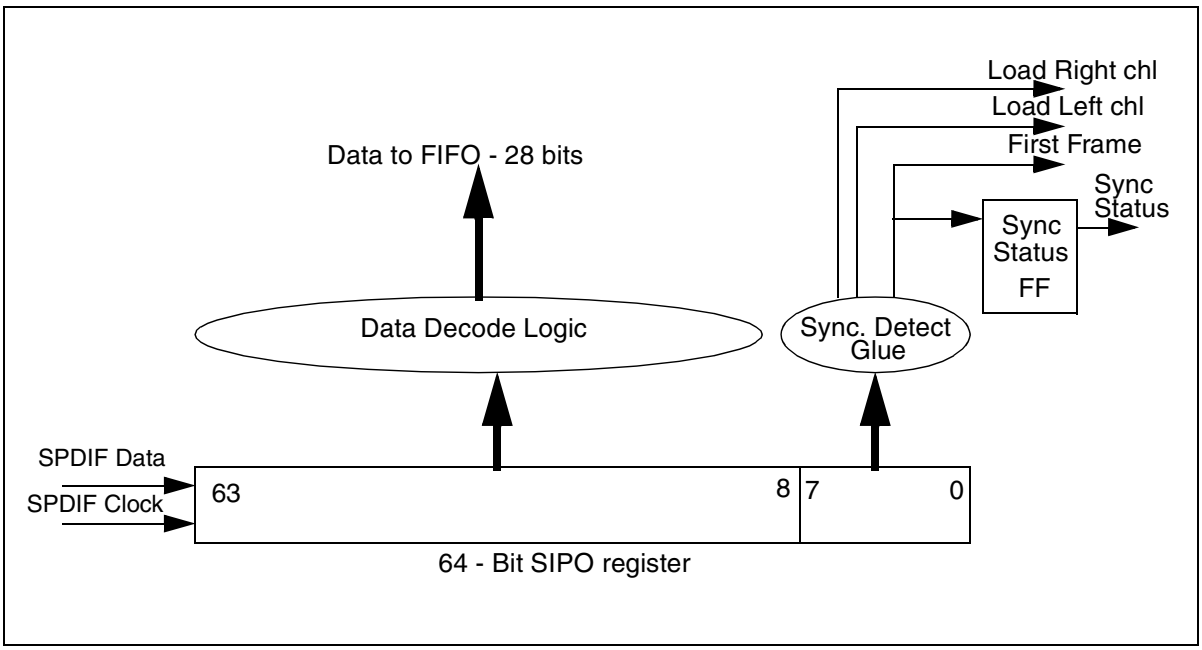
Figure 190: Audio packet generation



SPDIF Frame sync and data decode

The SPDIF frame Sync and data decode block is shown in Figure 191 below.

Figure 191: SPDIF Frame sync and data decode



The serial data from the SPDIF interface is serially shifted into the 64 bit serial in parallel out (SIPO) register. The LSB 8 bits of the shift register are used to detect the sync bits while the MSB 56 bits are used for data decoding. Decoded data is loaded into the left channel FIFO when the

load left channel signal is asserted to ‘1’ and rising edge of the SPDIF clock. The data is loaded into the right channel FIFO when load right channel signal is asserted to ‘1’. The bit in the frame FIFO is set only when load left channel is asserted and first frame signal is asserted to ‘1’, else a ‘0’ is shifted when load left channel is asserted to ‘1’. No data is shifted into the FIFO till the first frame is detected for the first time (sync status is asserted).

The sync detection glue is a simple combinatorial glue. The relation between the input and outputs is tabulated in the table below.

Table 163: SPDIF sync table

Output signal	state	Input data
First Frame	1 0	11101000 or 00010111 All others
Load Left Chl.	1 0	11100010 or 00011101 or ‘1’ of FIrst frame All others
Load Right Chl.	1 0	11100100 or 00011011 All others

The data decode logic consists of an array of two input XOR gates. The two inputs are connected to the successive outputs of the SIPO register.

Header Byte generation

This block is responsible for generating the header bytes for the audio sample packet. The bits in the header packets are modified in accordance to the different conditions of the SPDIF processing block. The layout information is fetched from the register HDMI_AUDIO_LAYOUT.

Formatter

The formatter gets the data from one or four SPDIF processing blocks and the header byte generation block. If the layout selected is ‘0’, then only the data from the first SPDIF processing block is mapped on to the sub-packet buses SP0 to SP3. If layout 1 is selected then the data from SPDIF0 stream is mapped onto sub-packet0 SP0, SPDIF1 to SP1, SPDIF2 to SP2 and SPDIF3 to SP3.

64.4.3.3 General control packet

The general control packet is transmitted if the enable bit in the HDMI_GENCTRL_PACKET_CONFIG register is set. 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.

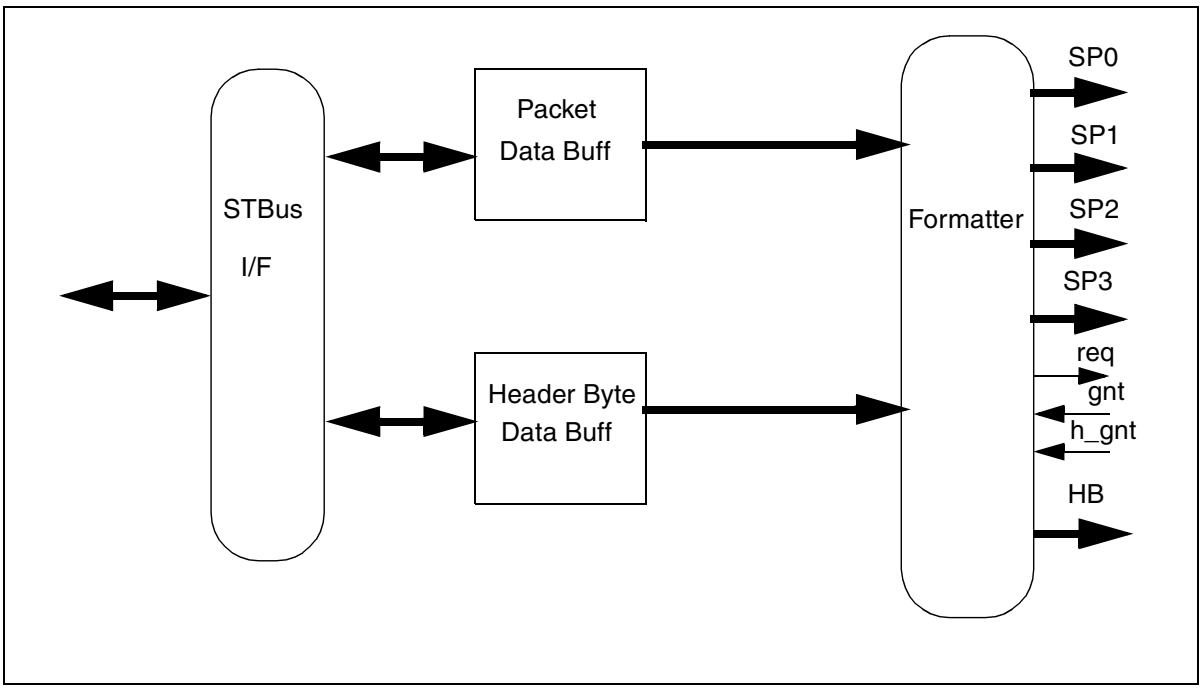
64.4.3.4 Info Frame

This module is responsible for sending the EIA/CEA-861B info frames on the HDMI interface. It is expected that the data corresponding to the header bytes and the packet contents are 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 will be the contents of this location +1 (checksum). It is expected that the **checksum is calculated by the software** and is written into the first byte of the packet word0. The **hardware does not calculate the checksum**. The checksum should be calculated by the software such that byte wise 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 config register is set. An interrupt is generated if enabled.

It is expected that one info packet is transmitted every frame. The info packet can consist of multiple info frames depending on the sync. The maximum bytes that can be contained in an info packet is 255 bytes. This implies a **maximum** of 9 to 10 info frames in one frame. Since interrupt is generated after transmission of each info frame, there can be a maximum of of 10 interrupts every 16 ms.

Figure 192: Info packet generation



To reduce the dependency of the CPU, the HDMI_AVI_BUFFER_EMPTY signal is bristled out, which can be used as pacing signal for DMA. The HDMI_AVI_BUFFER_EMPTY will be set when the data in the buffer is read by the HDMI formatter. It is reset when the enable bit in register HDMI_INFOFRAME_CONFIG is set.

64.4.3.5 Arbiter

The arbiter arbitrates between the requests different sources that intend to transmit data during the data island period of the HDMI frame. The arbiter follows a very simple priority based arbitration scheme with CTS packets being given highest priority, followed by audio data packet , general control packet and Info packet having the least priority. In case there is no packet from the individual sources, the arbiters 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 will be generated if enabled notifying completion of general packet transmission.

64.4.3.6 Packet Formatter + BCH coding

The data packet formatter is shown in the figure below. The module is responsible to fetch the data, perform BCH coding for header packet and individual sub-packets.It gives out the data on the three channels. Each of the three channels is 4 bits wide. The four bits are encrypted with the keys from the HDCP, only if the encryption is enabled and HDMI is authenticated. Then the data is TERC encoded before being fed to the line driver.

The header byte and each of the sub-packet bytes fetched are stored in a 8 bit flip-flop array. Each of these buffers have an BCH encoder associated with it. The BCH encoder associated with header byte is expected to run at the TMDS clock, while the BCH encoders associated with the sub-packet buffers are expected to run at BCH clock frequency. It is expected that the enable pulses are appropriately generated depending on wether the BCH clock is twice or four times the

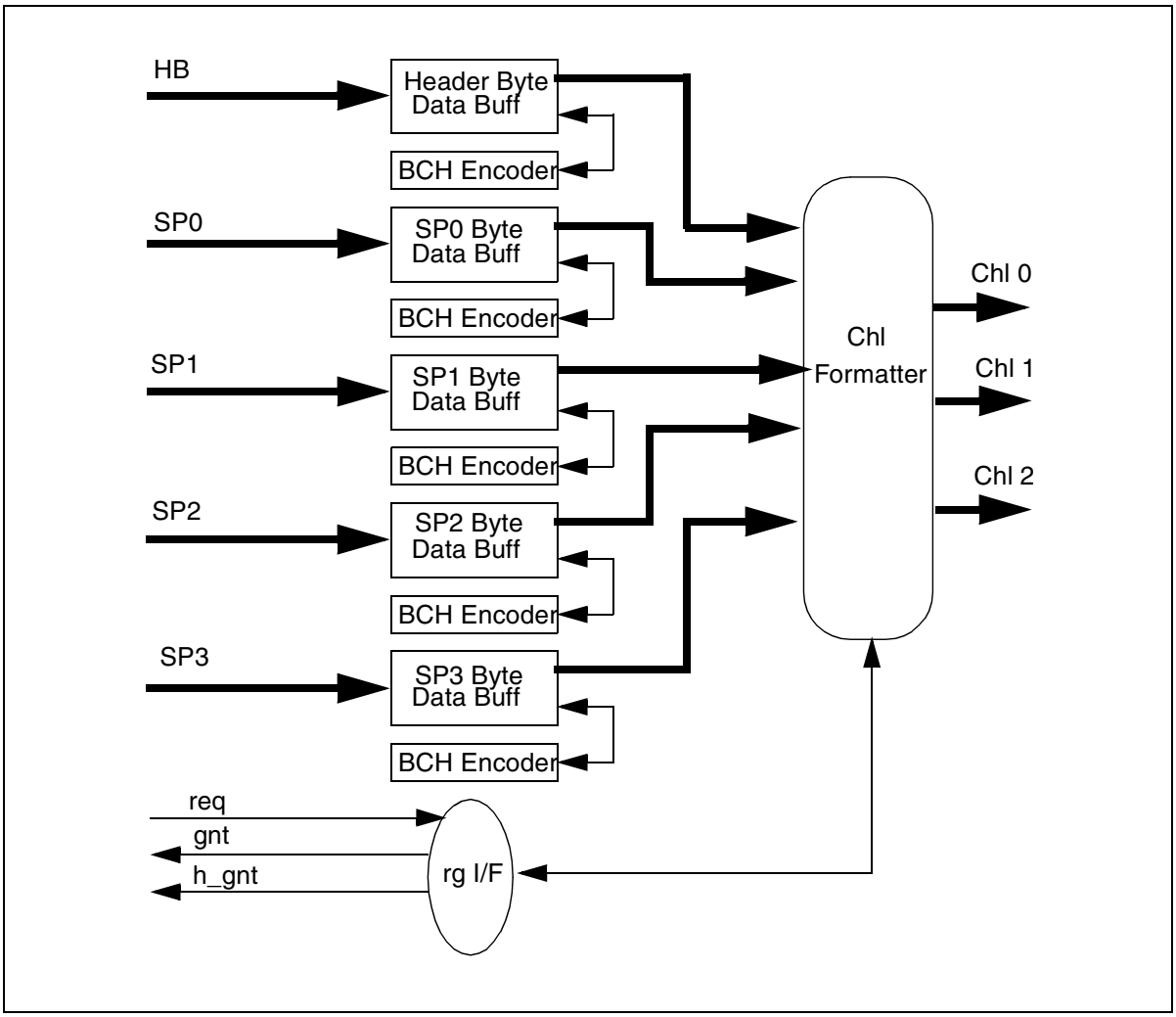
TMDS clock. The contents of the BCH encoder are loaded in the corresponding byte buffer (header byte or sub-packet), the data is then given to the channel formatter.

The channel formatter maps the individual bits of the of the data buffers on to the chl0, chl1 or chl2 bus as shown in the table below. A new byte of the sub-packet is transmitted during the clock periods5,6,7 and 8. Shown by shaded portions in the table.

Table 164: 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 193: Data packet generation



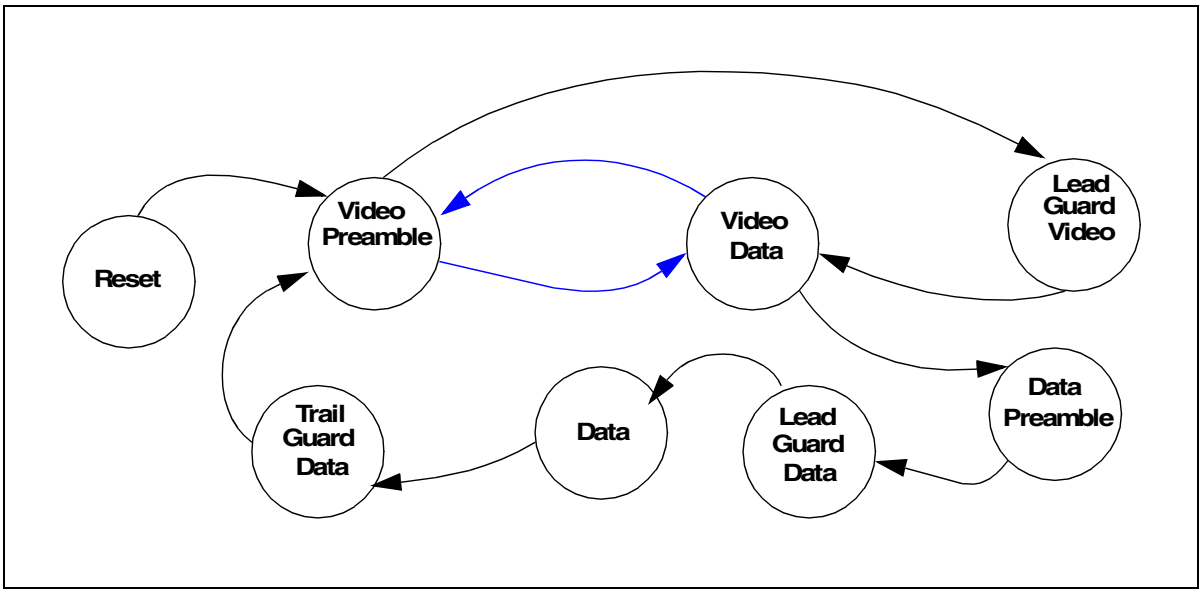
64.4.4 Frame formatting and line coding

This module performs two functions: frame formatting and line coding.

64.4.4.1 Frame formatting

The HDMI frame can be generated using a state machine as shown in the figure below.

Figure 194: HDMI Framer FSM



On reset the HDMI interface **does not** generate any frames. It stays in the reset state till the device enable bit in the HDMI config register is set. The state machine moves out of the reset state only at the beginning of the frame (first pixel).

If the ‘hdmi_not_dvi’ bit in the ‘hdmi_config’ register is not set then the HDMI interface is configured as Digital Visual Interface (DVI) and toggles between video control and video data states only (shown by blue lines).

Table 165 below lists the time the state machine spends in each state and the corresponding data bits on each channel.

Table 165: HDMI states vs. outputs

HDMI State	Time to be spent in clocks	CHL0 Data	CHL1 Data	CHL2 Data
Data Preamble	12(min)	0: Hsync 1: Vsync 7:2 = x	1: CTL0 0: CTL1 7:2 = x	1: CTL2 0: CTL3 7:2 = x
Lead Guard Data	2	0: Hsync 1: Vsync 2 = ‘1’ 3 = ‘1’ 7:4 = x	TMDS[9:0] = 0b0100110011	TMDS[9:0] = 0b0100110011
Data	n*32	0: Hsync 1: Vsync 2 = Header 3 = ‘0’ for first clock, ‘1’ thereafter 7:4 = x	3:0: Data 7:4 = ‘x’	3:0: Data 7:4 = ‘x’

Table 165: HDMI states vs. outputs

HDMI State	Time to be spent in clocks	CHL0 Data	CHL1 Data	CHL2 Data
Trail Guard Data	2	0: Hsync 1: Vsync 2 = '1' 3 = '1' 7:4 = x	TMDS[9:0] = 0b0100110011	TMDS[9:0] = 0b0100110011
Video Preamble	12(min)	0: Hsync 1: Vsync 7:2 = x	1: CTL0 0: CTL1 7:2 = x	0: CTL2 0: CTL3 7:2 = x
Lead Guard Video	2	TMDS[9:0] = 0b1011001100	TMDS[9:0] = 0b0100110011	TMDS[9:0] = 0b1011001100
Video Data	As programmed	TMDS code	TMDS code	TMDS code

It is expected that the extended control periods are generated as programmed in the HDMI mode.

Control data variations for DVI

All the control pins CTL0, CTL1, CTL2 and CTL3 are held to logic '0' internally in the DVI mode. Legacy applications where the CTL0 is toggled at even to odd edges is **not supported**.

However to deal with situations where in the control bits have to be different in DVI mode, the control bits are driven by CTLx bits in 'hdmi_dvi_control_bits' registers. In HDMI mode the contents of this register are neglected. When the encryption is enabled the CTL3 bit status for encryption overrides the value in CTL3 of the register. The default value of these bits is '0'.

Control data during Original Encryption Status Signalling (OESS)

OESS is used **only** in DVI and uses only the CTL3 to signal a fame is being encrypted. This signalling is only used when the HDMI is in authenticated state. The CTL3 must be asserted to '1' for at **least 8 pixel clocks,** starting no closer than 128 pixel clocks from the end of vertical blank. That is CTL3 must to asserted for atleast 8 pixel clocks after 128 pixel clocks after the end of VSYNC.

Control data during Enhanced Encryption Status SIgnalling (EESS)

EESS utilizes all the four control signals. The ESS can be used both in DVI and HDMI modes. The possible patterns for control bits is shown in the table below.

Table 166: EESS

CTL3	CTL2	CTL1	CTL0	Description
1	0	0	1	Encryption Enabled for the Frame
0	0	0	1	Encryption Disabled for the Frame

The CTLx 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 CTLx should be asserted for 16 clocks starting from 512 pixel clocks 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 be 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 tranmitter is in authenticated state.

64.4.4.2 Line Coding

The HDMI uses different encoding techniques during different states of the frame. The encoding schemes employed during each of the frame states is documented in the following sections.

Data Control (preamble)

During the data control period channel0 carries HSYNC and VSYNC on bit0 and bit1 respectively. Channel1 carries CTL0 and CTL1 on bit0 and bit1 respectively, while channel2 carries CTL2 and CTL3 on bit0 and bit1 respectively. The two bits are coded into 10 bits as shown in the table below.

Table 167: Coding of control data

Bit1	Bit0	TMDS output [9:0]
0	0	0b1101010100
0	1	0b0010101011
1	0	0b0101010100
1	1	0b1010101011

Leading Guard band Data

During the leading guard band HSYNC, VSYNC, ‘1’ and ‘1’ are mapped on to bit0, bit1, bit2 and bit3 respectively. These four bits are TERC4 encoded (as described in section 4.5.2.4). Channel1 carries a binary data q_out[9:0] = “0b010011011” and channel2 carries a binary data q_out[9:0] = “0b0100110011”.

Data Island

During the data island period HSYNC, VSYNC and packet header bit0, bit1 and bit2 respectively. Logic ‘0’ is sent on bit3 for the first clock and logic ‘1’ there after. LSB 4 bits of channel1 and channel2 carry packet data. The four bits are TERC4 encoded as shown in the table below.

Table 168: TERC4 Encoding

Input	TERC4 Data [9:0]
0b0000	0b1010011100
0b0001	0b1001100011
0b0010	0b1011100100
0b0011	0b1011100010
0b0100	0b0101110001
0b0101	0b0100011110
0b0110	0b0110001110
0b0111	0b0100111100
0b1000	0b1011001100
0b1001	0b0100111001
0b1010	0b0110011100
0b1011	0b1011000110
0b1100	0b1010001110
0b1101	0b1001110001
0b1110	0b0101100011
0b1111	0b1011000011

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, channel0 carries HSYNC and VSYNC on bit0 and bit1 respectively. Channel1 carries CTL0 and CTL1 on bit0 and bit1 respectively, while channel2 carries CTL2 and CTL3 on bit0 and bit1 respectively. These bits are coded as shown in Table 167. The value of the control bits in video premble are as follows CTL0='1', CTL1='0', CTL2='0' and CTL3='0'

Leading guard band video

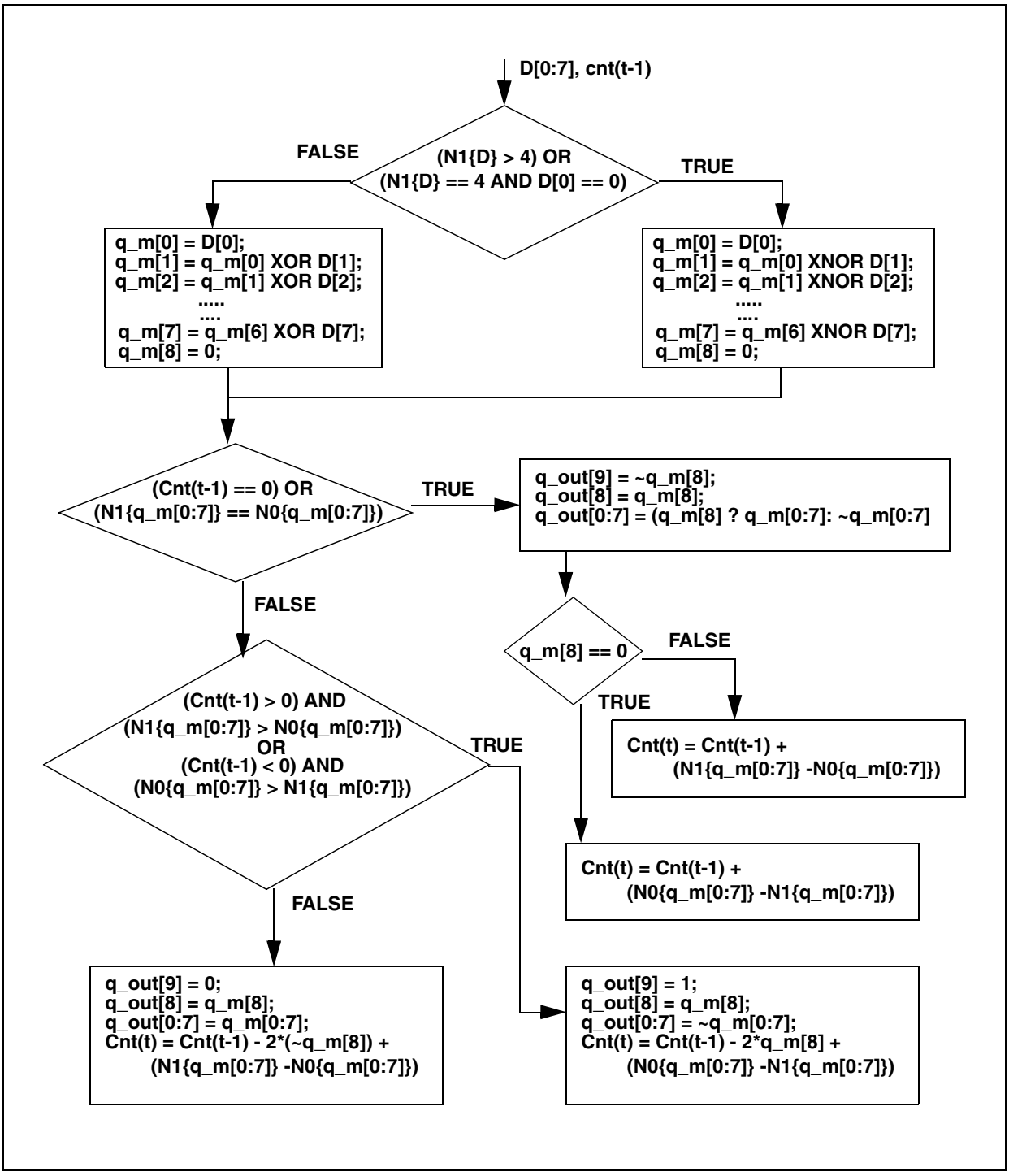
During the video leading guard band, channel0 carries q_out[9:0] = "0b1011001100", channel1 carries q_out[9:0] = "0b0100110011" and channel2 carries q_out[9:0] = "0b1011001100".

Video data

During the video data period the 8 bit input data is encoded as a 10 bit data which has an approximate DC balance as well as reduced transitions. This is done in two stages. In first stage

a 9 bit transitions minimized data is produced. In the second stage a 10th bit which balances the overall DC is added.The process of coding the 8 bits into 10 bits is shown in Figure 195 below.

Figure 195: TMDS Video encode Algorithm



D[0:7]: 8 bit input data

cnt: Register used to keep track of data stream disparity. The count value is considered to be zero at the upon entering video island. That is the disparity counter has to be initialized to zero before the video island starts.

+ve value represents excessive ones, negative value indicates excessive ‘0’s

To be initialized to ‘0’ at the beginning of the Video data period.

q_m: Intermediate value.

q_out: Final 10bit encoded data.

64.4.5 HDCP interface

The HDCP block is expected outputs the keys on a 24 bit data bus. The keys are XORed with the data just before TMDS encoding 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 169: 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 170 below.

Table 170: Encryption Stream mapping for video data

HDCP key in	TERC4 Bits
23:20	Unused
19	Channel2 Bit 3
18	Channel2 Bit 2
17	Channel2 Bit 1
16	Channel2 Bit 0
15:12	Unused
11	Channel1 Bit 3
10	Channel1 Bit 2
9	Channel1 Bit 1
8	Channel1 Bit 0
7:3	Unused
2	Channel0 Bit 2
1:0	Unused

In addition to the 24 bit HDCP key in bus, the HDCP interface is supposed to have HSYNC, VSYNC, Video data active and data island active, pixel repetition enable, TMDS clock signals.

The video data active is at logic ‘1’ when valid video data is sent in the HDMI frame. The data active is at logic ‘1’ when valid data island is present in the HDMI Frame.

The complete list of the signal on the HDCP interface are tabled in section 5.4.

64.4.5.1 Authenticated state

When the HDMI is in authenticated state, HDCP is expected to assert signal ENCRYPTION_ENABLE to logic ‘1’. The HDMI sends the valid video and audio data if the encryption is enabled (HDCP enable bit in register HDMI_CONFIG is set) and the HDMI is in authenticated state. If the HDMI is not in authenticated state, a default value is sent (see register descriptions HDMI_ACTIVE_VIDEO_YMIN, HDMI_ACTIVE_VIDEO_YMAX and HDMI_DEFAULT_CHL0_DATA). If the encryption is not enabled then the valid video and audio data are sent.

64.4.5.2 Enhanced ri Computation mode support

For enhanced ri computation it is required to capture the first pixel of the active window. To facilitate the functionality the HDCP sets the signal RI_PIXEL_CAPTURE to HIGH for one pix clock when VSYNC is asserted. On detection of the signal the HDMI captures values of the pixels. The location of pixel whose value is to be captured has to be programmed in HDMI_ACTIVE_VIDEO_XMIN and HDMI_ACTIVE_VIDEO_YMIB registers. The captured pixels can be read from registers HDMI_CHL0_CAPTURE_DATA, HDMI_CHL1_CAPTURE_DATA and HDMI_CHL2_CAPTURE_DATA. In case the input video format is RGB, the channel0 data corresponds to blue, channel1 corresponds to green and channell2 corresponds to red. An interrupt is generated if enabled.

64.4.6 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 +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 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 logic level of the input pin if enabled.

64.5 Bus Interface and IO Description

The interface signals for the HDMI Digital processing block are listed in the following sections.

64.5.1 System signals

Table 171: System signals

Signal name	I/O	Timing	Description	Logical grouping
STBUS_CLOCK	I		10 ns System Clock	system clock
PIX_CLOCK	I		Pixel clock	
TMDS_CLOCK	I		TMDS clock	
BCH_CLOCK	I		Clock frequency is twice or four times the TMDS clock	
SPDIF_CLOCK	I		SPDIF clock, 128 times Fs	
HOT_PLUG_IN	I	Async	Hot Plug pin	
RST_N *	I		Global reset	Asynchronous Reset

* RST_N is Active LOW and asynchronous. **Synchronization to be performed internally**.

64.5.2 STBus interface signals

Table 172: STBus interface signals

Signal name	I/O	Timing	Description	Logical grouping
HDMI_STBUS1_T_REQ	I	L	Standard STBus signals	STBus interface.
HDMI_STBUS1_T_ADD[31:2]	I	L		
HDMI_STBUS1_T_DATA[31:0]	I	L		
HDMI_STBUS1_T_OPC[3:0]	I	L		
HDMI_STBUS1_T_BE[3:0]	I	L		
HDMI_STBUS1_T_EOP	I	L		
HDMI_STBUS1_T_R_REQ	O	E		
HDMI_STBUS1_T_R_DATA[31:0]	O	E		
HDMI_STBUS1_T_R_OPC	O	E		
HDMI_INTERRUPT	O	E	Interrupt	Interrupt
HDMI_AVI_BUFF_EMPTY	O	E	Pacing signal for DMA	Buffer empty

All Times are measured relative to system clock.

E: early- within 20% of clock cycle

M: mid - within 40% of clock cycle

L: late - within 80% of clock cycle

Note: All STBus signals are synchronous to the STBus clock.

64.5.3 Video Interface

Table 173: Video interface signals

Signal name	I/O	Timing	Description	Logical grouping
HSYNC	I	L	Hsync pulse	Video in interface
VSYNC	I	L	Vsync Pulse	
CHANNEL0_IN[7:0]	I	L	8 bit per pixel video data Refer to section 4.3 for mapping the buses for different video formats	
CHANNEL1_IN[7:0]	I	L		
CHANNEL2_IN[7:0]	I	L		

NOTE: All the inputs are synchronous to pix clock.

64.5.4 Audio Interface

Table 174: Audio interface signals

Signal name	I/O	Timing	Description	Logical grouping
SPDIF_IN[3:0]	I	L	Four streams of SPDIF data	SPDIF interface

Note: SPDIF inputs are synchronous to SPDIF clock, which is 128 times FS.

64.5.5 Line Driver interface Signals

Table 175: Line Driver interface signals

Signal name	I/O	Timing	Description	Logical grouping
HDMI_CHL_0[9:0]	O	E	10 bit TMDA encoded data. Data valid on rising edge of TMDS_CLOCK	Line driver interface
HDMI_CHL_1[9:0]	O	E		
HDMI_CHL_2[9:0]	O	E		
DLL_CONFIG[1:0]	O	static	Defines relation between the TMDS clock and DLL clock 00, 01: CLOCK_DLL= CLOCK_TMDS 10: CLOCK_DLL = 2*CLOCK_TMDS 11: CLOCK_DLL = 4*CLOCK_TMDS	DLL clock config
DLL_LOCK	I	L	1: DLL is locked	DLL lock

Note: The line driver outputs are synchronous to the TMDS clock.

64.5.6 HDCP interface Signals

Table 176: HDCP interface signals

Signal name	I/O	Timing	Description
HSYNC_OUT[A]	O	E	HSYNC out
VSYNC_OUT	O	E	VSYNC out
VIDEO_DATA_ACTIVE	O	E	Video data active (Active HIGH)
DATA_ISLAND_ACTIVE	O	E	Data Island active (Active HIGH)
HDCP_KEY_IN[23:0]	I	M	Encryption Keys from HDCP
ENCRYPTION_ENABLE	I	M	1: Encryption enabled for the frame
RI_PIXEL_CAPTURE	I	L	1: ri pixel capture for the current frame

a. HSYNC_OUT and VSYNC_OUT to HDCP as always active high.

Note: The HDCP interface signals are expected to be synchronous to the TMDS clock.

64.6 Timing Description

All the inputs on the STBus interface to the HDMI cell are assumed late (within 80% of clock) and outputs early (within 20% of clock) with respect to the STBus clock.

On the HDCP interface all the outputs are assumed early and inputs are assumed mid with respect to the TMDS clock.

The TMDS data is assumed to be early with respect to the TMDS clock.

The Video data is assumed to be late with respect to the pix clock.

The SPDIF data is assumed to be late with respect to the SPDIF clock. The SPDIF clock is assumed to be twice the bit rate or 128 times the sampling frequency.

64.7 Clock Domain Specification

The HDMI formatter has five clocks: STBUS_CLOCK, PIX_CLOCK,TMDS_CLOCK, BCH_CLOCK, SPDIF_CLOCK. The clock is expected to be symmetrical (50% +/- 10% duty

cycle). However there is **no restriction** on the mark space ratio of the clock.The nominal STBUS_CLOCK frequency is expected to be 100 Mhz. However it should be possible to synthesize the design at different frequencies.

The nominal, minimum and maximum frequencies of operation expected are documented in the table below.

Table 177: Clock relations

	Minimum	Nominal	Maximum
STBUS_CLOCK		100 MHz	
PIX_ CLOCK	13.5(25) Mhz	74.35 MHz	165 MHz
TMDS_CLOCK	27	74.35 MHz	165 MHz
BCH_CLOCK	54	148.70 MHz	330 MHz
SPDIF_CLOCK	64 KHz	192 KHz	384 KHz

SPDIF_CLOCK is twice the data rate.

The relation between different clocks is tabulated in the table below.

Table 178: Clock relations

	STBUS_CLOCK	PIX_ CLOCK	TMDS_CLOCK	TMDS_CLOCK_2	SPDIF_CLOCK
STBUS_CLOCK		async	async	async	async
PIX_ CLOCK	async		sync	sync	async
TMDS_CLOCK	async	sync		sync	async
BCH_CLOCK	async	sync	sync		async
SPDIF_CLOCK	async	async	async	async	

In the case of two clocks to be synchronous, it is expected that both the clocks are derived from the same source and are integer multiple of each other. It is expected that the rising edges of such clocks match with a pre-defined skew. The maximum allowable skew has to be defined in the implementation specification.

It is expected that there are **no generated clocks** in the design. Any clock divisions are achieved by generating enable pulses.

The BCH clock is either twice the TMDS clock or four times the TMDS clock. It is expected that appropriate enable pulses are generated to clock the BCH encoder. The BCH clock is not divided internally in case the clock is four times the TMDS clock.

64.8 Reset Specification

The global reset is asynchronous and active low. The asynchronous reset is expected to be **synchronized to each of the clock domains internally** before it is being used.

The reset needs to be asserted for a minimum duration equal to the sum of the library removal time and the clock uncertainty.As a safe option, the reset signal should remain asserted for at least one clock period of the slowest clock to ensure proper reset of all state machines and flip-flops.

There is **no** requirement for the clocks to be running or being off during the period the reset remains active. During reset the all the signals are don't care and outputs are at logic '0'.

After de-assertion of the reset the HDMI interface is expected to be inactive (driving all TMDS outputs to logic '0'). The TMDS interface becomes active only after the DEVICE_ENABLE bit is set in the HDMI_CONFIG register.

64.9 Info frame programming

Following sequence of events have to be executed to program an info frame.

- ‘hdmi_status’ register is read and checked if the ‘info_buffer_status’ bit is set to ‘1’.
  The programming has to continue only if this bit is set, else the processor has to wait till this bit is set by the HDMI cell.
- Header bytes have to assembled as a word (with LSB containing HB0) and written into the ‘hdmi_infoframe_headerbyte’ register.
- Packet bytes have to be assembled as words (with Least significant byte containing the Least significant byte of the word) and have to be written into the HDMI_INFOFRAME_PACKETBYTE_FIFO.
  Consecutive writes with different data words to this register fill the FIFO.
- Enable the info frame by setting the ‘enable’ bit in the ‘hdmi_infoframe_config’ register.
  The HDMI cell automatically infers the number of bytes to be transmitted from the lower nibble of the third byte in the ‘hdmi_infoframe_headerbyte’ register and starts transmitting the info frame.
- The Busy bit in the HDMI_INFOFRAME_FIFOSTATUS register is set once the HDMI cell starts transmitting the info frame. It simultaneously resets the enable bit in register HDMI_INFOFRAME_CONFIG. The busy bit is reset only after the complete info frame is transmitted.
- The INFO_BUFFER_STATUS bit in register HDMI_STATUS is set once all the data is read by the HDMI cell, not necessary transmitted.
- An interrupt is generated if enabled once the info frame buffer is empty.
- The interrupt can be cleared by writing ‘1’ into bit CLR_INFOFRAME_INT of the HDMI_INTERRUPT_CLEAR register.

It is expected that one info packet is sent each frame. One info packet may contain multiple inforframes. A maximum of 255 bytes may be carried in one info packet. This implies a maximum of 10 info frames. To facilitate transmission of info packet every new frame, an interrupt can be genered (if enabled) on start of a new frame. Assuming a typical frame rate to be 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 (max).

64.10 Interface Timings

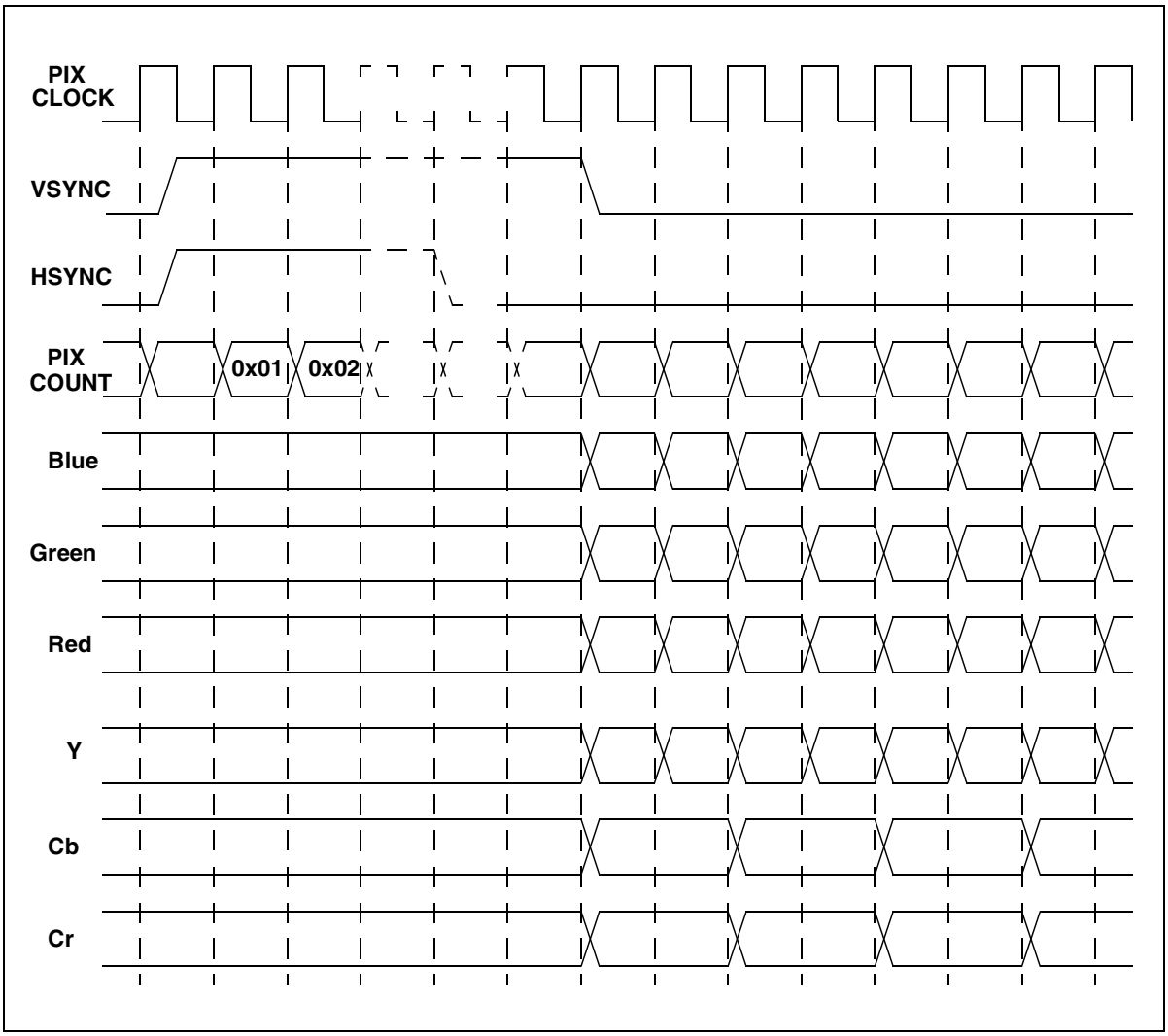
64.10.1 Video Interface

The Video interface timings are shown in the figure below.

The pixel with both HSYNC and VSYNC asserted (at ‘1’ in the figure below) and first rising pix clock edge is assumed to be the first pixel. The assertion of the HSYNC and VSYNC (to ‘1’) for requisite time in HDMI frame is not calculated internally to HDMI. The logic state of the HSYNC

and VSYNC signals is stuffed into the HDMI frame accordingly. The VTG which generates these signals is assumed to take care of these timings.

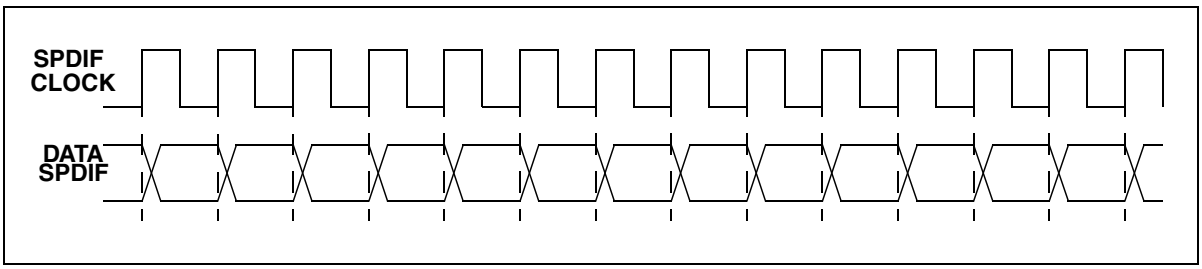
Figure 196: Input video interface timings



64.10.2 SPDIF Interface

The timings on the SPDIF interface are shown in Figure 197 below.

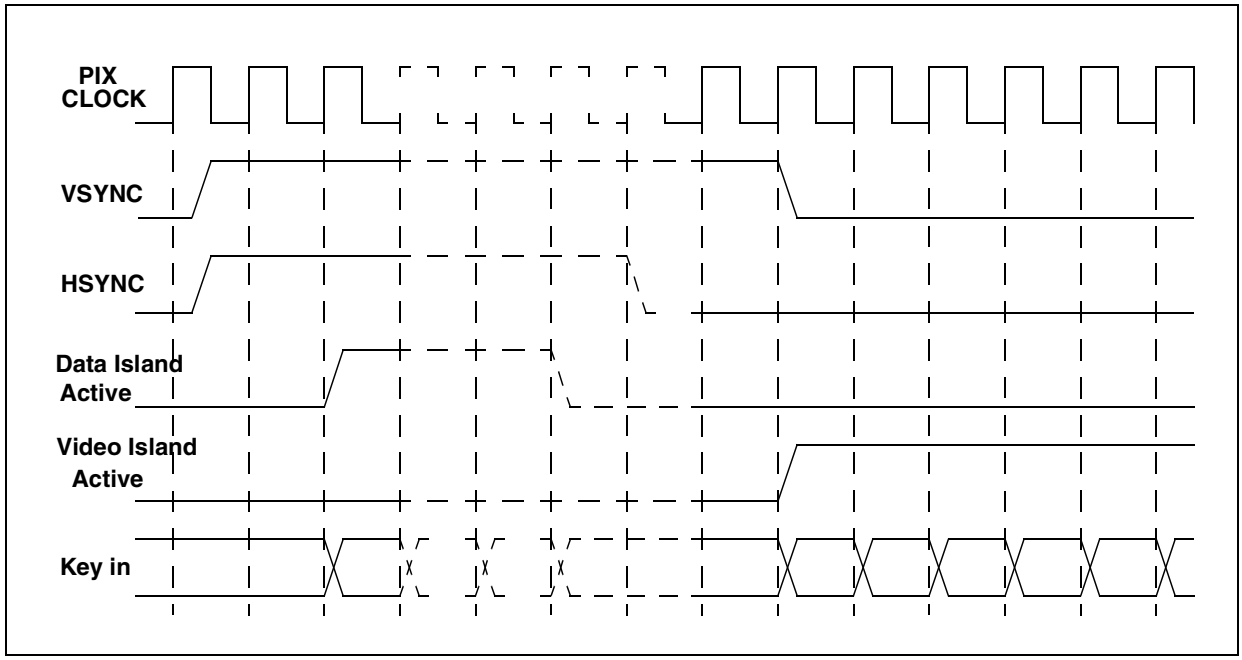
Figure 197: SPDIF interface timings



64.10.3 HDCP interface

The timings on the HDCP interface are shown in the figure below.

Figure 198: HDCP interface timings



64.11 Grey Codes

The grey codes for 16 bit data is shown in the table below.

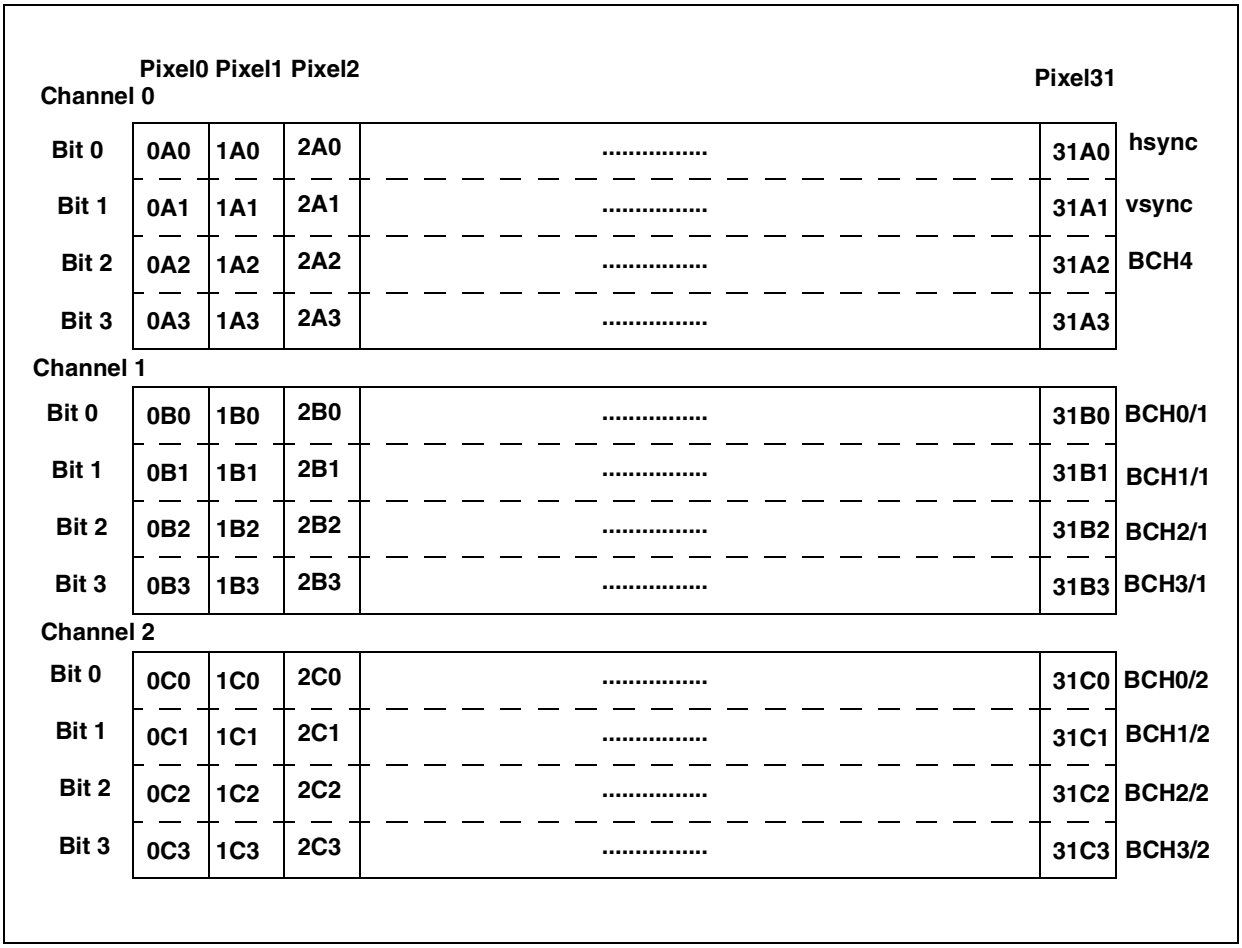
Table 179: Grey codes

Binary value	Grey code
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

64.12 Data Island Definitions

64.12.1 Data Island packet construction

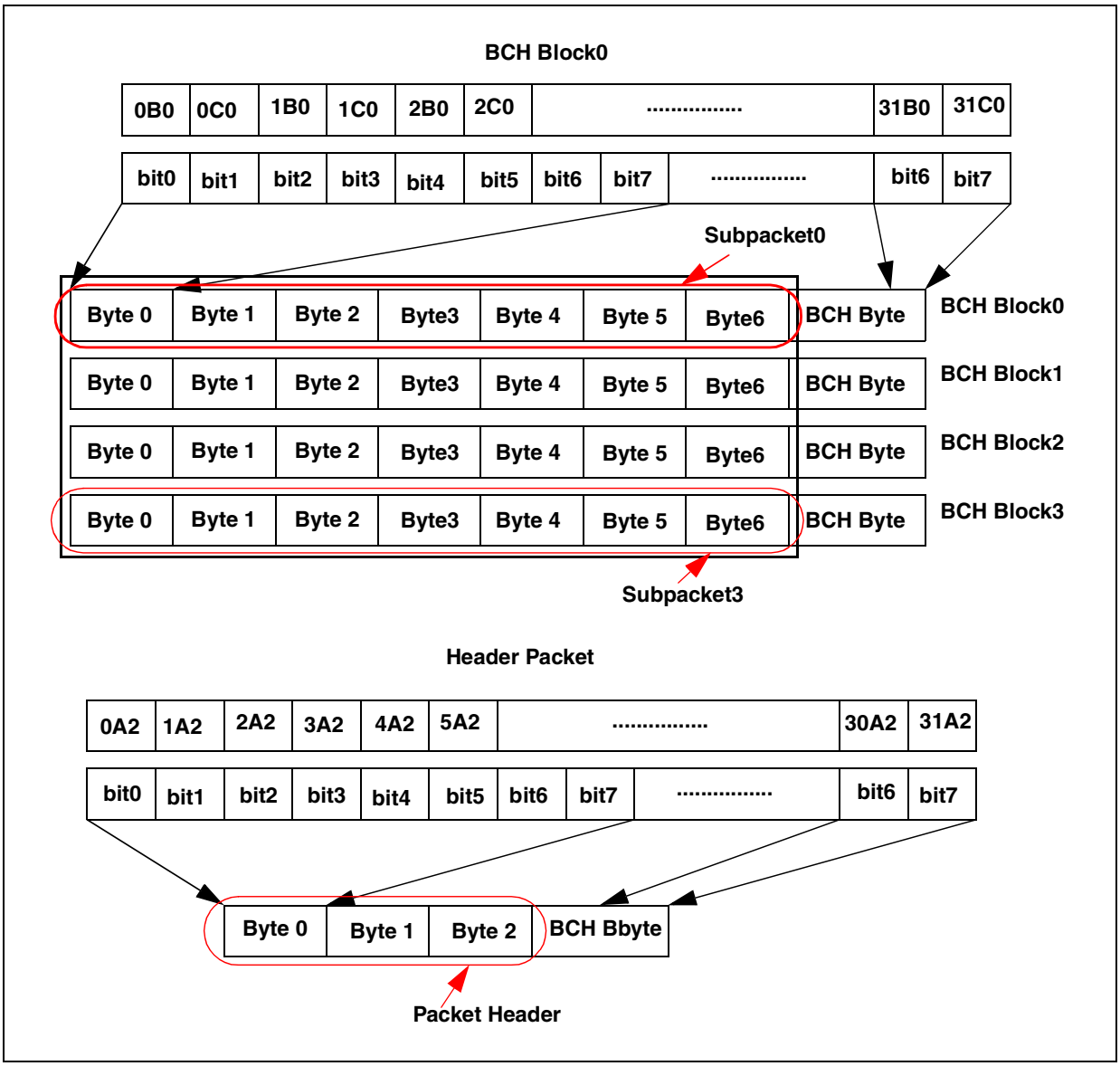
Figure 199: BCH blocks during data Island



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 200 below.

Figure 200: Formation of BCH blocks



64.12.2 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 sub-packets. Each sub-packet 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 is describes in the following sections.

64.12.2.1Null Packet

All the three bytes of the header packet and all the bytes of each of the sub-packets of a null packet contain 0x00.

64.12.2.2Audio Clock Regeneration packet

The contents of the header bytes of the audio clock regeneration packet are shown in Table 180 below.

Table 180: 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 sub-packets contain the same information. The contents of the sub-packet are shown in Table 181 below.

Table 181: Sub-packet 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

64.12.2.3Audio Sample Packet

The contents of the header bytes of the audio sample packet are shown in Table 182 below.

Table 182: 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.spX: indicates if sub-packet X contains an audio sample

sample flat.spX: Bit is set if no useful audio data is available. Valid only if sample present.spX is set.

B.X: is set if sub-packet X contains the first frame of the IEC60958 block.

Table 183: Sub-packet 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 184 below.

Table 184: Layouts for audio sample packets

Layout Value	Max Num Chls	samples	subpkt 0	subpkt 1	subpkt 2	subpkt 3
0	2	4	chl 1,2 sample 0	chl 1,2 sample 1	chl 1,2 sample 2	chl 1,2 sample 3
1	8	1	chl 1,2 sample 0	chl 3,4 sample 0	chl 5,6 sample 0	chl 7,8 sample 0

64.12.2.4Info frame packet

The contents of the header bytes of the info packet are shown in the table below.

Table 185: Header bytes of info packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
HB0	1	Info Frame_type						
HB1	InfoFrame_version							
HB2	0	0	0	InfoFrame_length				

The bytes in the packet body are shown below.

Table 186: Sub-packet 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 sub-packets 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.

64.12.2.5 General Control packet

The general control packet is transmitted only between active edge of VSYNC and 384 pixels following the active edge. All the four sub-packets are identical. The subpacket is shown in the table below.

The contents of the header bytes of the general control packet are shown in the table below.:

Table 187: 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

All the four sub-packets contain the same information. The contents of the sub-packet are shown in the table below.

Table 188: Sub-packet bytes of general control packet

Bytes	Bits							
	7	6	5	4	3	2	1	0
SB0	0	0	0	Clear_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

65 HDMI registers

Addresses are provided as *HDMIBaseAddress* + offset.

The *HDMIBaseAddress* is:

0x2010 3200.

Table 189: Register summary table

Register Name	Description	Offset	Type
HDMI_CONFIG	Configuration	0x000	R/W
HDMI_INTERRUPT_ENABLE	Interrupt enable	0x004	R/W
HDMI_INTERRUPT_STATUS	Interrupt Status	0x008	RO
HDMI_INTERRUPT_CLEAR	Interrupt Clear	0x00C	WO
HDMI_STATUS	Status	0x010	RO
HDMI_EXTS_MAX_DELAY	Max. time between extended control periods	0x014	R/W
HDMI_EXTS_MIN	Duration of extended control period	0x018	R/W
HDMI_DVI_CONTROL_BITS	DVI control bits	0x01C	R/W
Reserved	-	0x020-0x0FC	R/W
HDMI_ACTIVE_VIDEO_XMIN	Active Video window co-ordinates	0x100	R/W
HDMI_ACTIVE_VIDEO_XMAX		0x104	R/W
HDMI_ACTIVE_VIDEO_YMIN		0x108	R/W
HDMI_ACTIVE_VIDEO_YMAX		0x10C	R/W
HDMI_DEFAULT_CHL0_DATA	Channel data that is to be sent when the HDCP is not authenticated	0x110	R/W
HDMI_DEFAULT_CHL1_DATA		0x114	R/W
HDMI_DEFAULT_CHL2_DATA		0x118	R/W
HDMI_CHL0_CAPTURE_DATA	Captured data	0x11C	RO
HDMI_CHL1_CAPTURE_DATA		0x120	RO
HDMI_CHL2_CAPTURE_DATA		0x124	RO
Reserved	-	0x128 - 0x1FC	-
HDMI_AUDIO_LAYOUT	Audio layout 2 or 8 channel audio	0x200	R/W
Reserved	-	0x204 - 0x20C	-
HDMI_INFOFRAME_HEADER_WORD	Header Word of AVI info frame	0x210	R/W
HDMI_INFOFRAME_PACKET_WORD0		0x214	R/W
HDMI_INFOFRAME_PACKET_WORD1		0x218	R/W
HDMI_INFOFRAME_PACKET_WORD2		0x21C	R/W
HDMI_INFOFRAME_PACKET_WORD3		0x220	R/W
HDMI_INFOFRAME_PACKET_WORD4		0x224	R/W
HDMI_INFOFRAME_PACKET_WORD5		0x228	R/W
HDMI_INFOFRAME_PACKET_WORD6		0x22C	R/W
HDMI_INFOFRAME_CONFIG	Config the info frame	0x230	R/W
Reserved	-	0x234 - 0x23C	-
HDMI_INFOFRAME_FIFO_STATUS	Info frame FIFO status	0x240	R/W
HDMI_SAMPLE_FLAT_MASK	Sample flat mask	0x244	R/W
HDMI_GENCTRL_PACKET_CONFIG	Configuration for general control Packet	0x248	R/W
Reserved	-	0x24C - 0x2FC	-

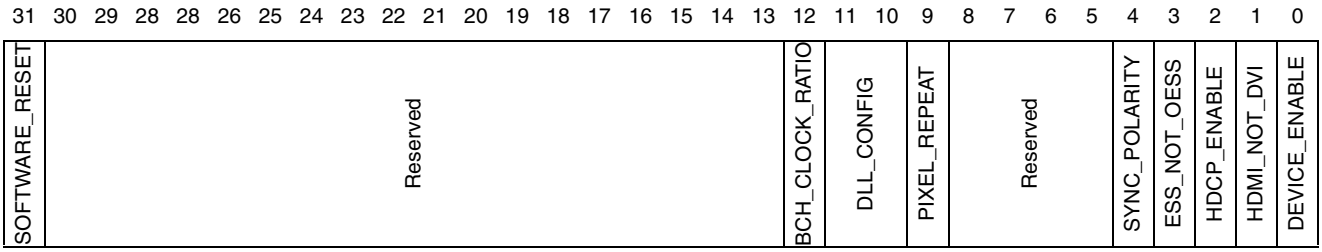
Table 189: Register summary table

Register Name	Description	Offset	Type
HDMI_MISR_REF_SIGNATURE1	MISR1/ channel0 output reference signature	0x300	R/W
HDMI_MISR_REF_SIGNATURE2	MISR2/ channel1 output reference signature	0x304	R/W
HDMI_MISR_REF_SIGNATURE3	MISR3/ channel2 output reference signature	0x308	R/W
HDMI_MISR_TEST_CONTROL	MISR test control	0x30C	R/W
HDMI_MISR_TEST_STATUS	MISR test status	0x310	RO
Reserved	-	0x310 - 0xFFC	-

The register address space is non-contiguous to facilitate addition of any new registers in future. Writes to read only registers have no effect on the contents of these registers and reads from write only registers returns all ‘0’s.

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 1K address map.

HDMI_CONFIG HDMI configuration



Address: *HDMIBaseAddress* + 0x000

Type: R/W

Reset: 0

Description: The register contents like DEVICE_ENABLE, HDMI_NOT_DVI, HDCP_ENABLE, ESS_NOT_OESS, ENABLE_422, and PIXEL_REPEAT are expected to take effect from the new frame onwards.

[31] **SOFTWARE_RESET**
To enable software reset, bit SOFTWARE_RESET has to 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_CLOCK_RATIO**
0: BCH clock = 2 x TMDS clock
1: BCH clock = 4 x TMDS clock

[11:10] **DLL_CONFIG**
Configure relationship between DLL clock and TMDS clock

[9] **PIXEL_REPEAT**
1: Enable pixel repetition by 2

[8:5] **Reserved**
Set to 0

[4] **SYNC_POLARITY**
0: HSYNC and VSYNC active HIGH
1: HSYNC and VSYNC active LOW

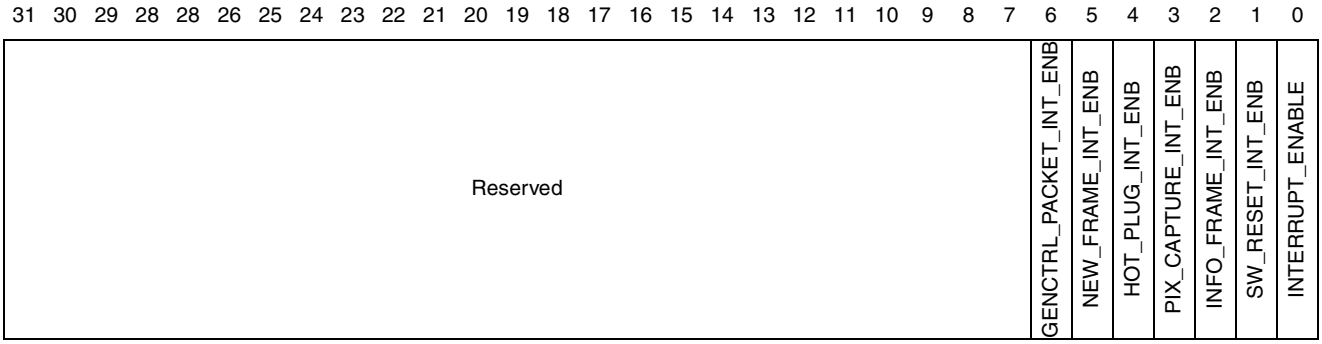
[3] **ESS_NOT_OESS**
This bit is valid only if HDCP_ENABLE is set to '1'
0: OESS Enable
1: ESS Enable

[2] **HDCP_ENABLE**
1: Enable HDCP interface

[1] **HDMI_NOT_DVI**
1: Enable HDMI Compatible frame generation
0: DVI compatible Frame generation

[0] **DEVICE_ENABLE**
1: Enable HDMI cell

HDMI_INTERRUPT_ENABLE HDMI interrupt enable



Address: *HDMIBaseAddress* + 0x004
Type: R/W
Reset: 0
Description:

[30:7] **Reserved**

[6] **GENCTRL_PACKET_INT_ENB**
1: Enable interrupt on completion of general control packet Xmission.

[5] **NEW_FRAME_INT_ENB**
1: Enable interrupt on new (start of) frame

[4] **HOT_PLUG_INT_ENB**
1: Enable interrupt on change in hot_plug_in logic toggle

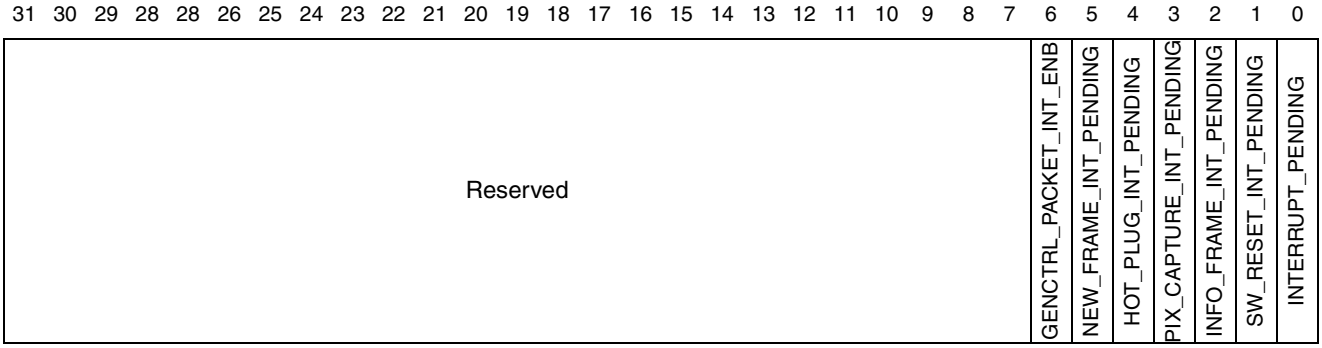
[3] **PIX_CAPTURE_INT_ENB**
1: Enable interrupt after capture of first active pixel

[2] **INFO_FRAME_INT_ENB**
1: Enable interrupt after info frame transmission

[1] **SW_RESET_INT_ENB**
1: Enable interrupt after soft reset completion

[0] **INTERRUPT_ENABLE**
1: Global Interrupt enable

HDMI_INTERRUPT_STATUS 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 bit0 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.

[31:7] **Reserved**

[6] **GENCTRL_PACKET_INT_ENB**
1: General ctrl packet Xmission interrupt pending: set after the general control packet is given to formatter. It is expected that the packet is subsequently transmitted.

[5] **NEW_FRAME_INT_PENDING**
1: New frame interrupt pending - at the first pixel of the frame.

[4] **HOT_PLUG_INT_PENDING:** Hot plug interrupt pending
1: Any change in logic state of hot_plug_in signal is detected.

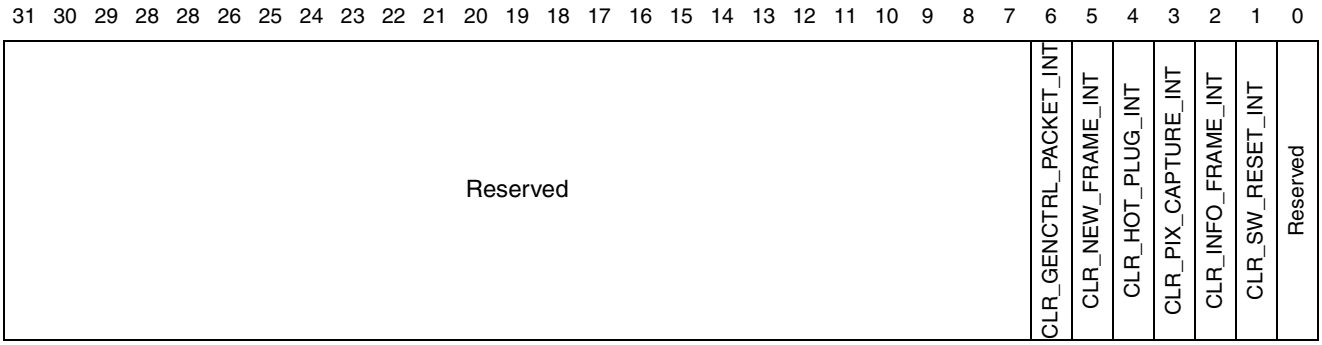
[3] **PIX_CAPTURE_INT_PENDING**
1: Pixel capture interrupt pending - set when the pixel data is captured.

[2] **INFO_FRAME_INT_PENDING**
1: Info frame transmission interrupt pending
Set when the info frame buffer is empty

[1] **SW_RESET_INT_PENDING**
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] **INTERRUPT_PENDING:** Global interrupt pending
1: Global Interrupt Pending - any interrupt status bit is set to '1'. Cannot be cleared by clear bit operation.

HDMI_INTERRUPT_CLEAR Interrupt clear



Address: *HDMIBaseAddress* + 0x00C

Type: WO

Reset: 0

Description:

[31:7] **Reserved**

[6] **CLR_GENCTRL_PACKET_INT**
1: clear general control packet Xmission Interrupt

[5] **CLR_NEW_FRAME_INT**
1: clear new frame interrupt

[4] **CLR_HOT_PLUG_INT**
1: clear hot plug interrupt

[3] **CLR_PIX_CAPTURE_INT**
1: clear pixel capture interrupt

[2] **CLR_INFO_FRAME_INT**
1: clear info frame transmission interrupt

[1] **CLR_SW_RESET_INT**
1: clear soft reset interrupt

[0] **Reserved**

HDMI_STATUS 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																											HOT_PLUG_STS	PIX_CAPTURE_STATUS	INFO_BUFFER_STATUS	SW_RESET_STATUS	Reserved

Address: *HDMIBaseAddress* + 0x010

Type: RO

Reset: 0

Description:

[31:5] **Reserved**

[4] **HOT_PLUG_STS**
Logic level on hot plig in line is stored here

[3] **PIX_CAPTURE_STATUS**
1: after pixel is stored and ri calculation enabled, is reset after reading the pixel contents.

[2] **INFO_BUFFER_STATUS**
1: Info buffer empty.

[1] **SW_RESET_STATUS**
1: Soft reset complete, Bit is reset when software_reset bit in HDMI_CONFIG is reset.

[0] **Reserved**

HDMI_EXTS_MAX_DELAY 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_DELAY			

Address: *HDMIBaseAddress* + 0x014

Type: R/W

Reset: 0

Description: This register holds the maximum time between extented control periods in number of VSYNC occurences. It is expected that the maximum time between the extended control periods does not exceed 50ms. For a frame rate of 50 or 60 Hz, the extended control period has to be sent every two Frames.

[31:4] **Reserved**

[3:0] **MAX_DELAY:** Maximum delay between extended control periods.
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: This register specifies the minimum duration of extended control periods in TMDS clocks.

[31:8] **Reserved**

[7:0] **EXTS_MIN**: Specifies duration of extended control.
0: 256 clocks
Default value to be written: 0x32

HDMI_DVI_CONTROL_BITS Value of control bits in DVI mode

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																												CTL3	CTL2	CTL1	CTL0

Address: *HDMIBaseAddress* + 0x01C

Type: R/W

Reset: 0

Description: This register specifies value of control bits in DVI mode. The value of CTL3 is overridden if encryption is enabled. This register is present as a backup to addrerss any incompatibility issues that may occur post silicon.

[31:4] **Reserved**

[3] **CTL3**: Value of CTL3

[2] **CTL2**: Value of CTL2

[1] **CTL1**: Value of CTL1

[0] **CTL0**: Value of CTL0

HDMI_ACTIVE_VIDEO_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 for only those pixels whose line number is grater than or equal to the contents of HDMI_ACTIVE_VIDEO_YMIN and pixel number of the line is grater than or equal to the value in HDMI_ACTIVE_VIDEO_XMIN register.

[31:13] **Reserved**

[12:0] **XMIN:** xmin value in pix clocks.

HDMI_ACTIVE_VIDEO_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: The contents of this register donate maximum number of pixels per line.

[31:13] **Reserved**

[12:0] **XMAX:** xmax value in pix clocks.

HDMI_ACTIVE_VIDEO_YMIN Ymin 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																			YMIN												

Address: *HDMIBaseAddress* + 0x108
Type: R/W
Reset: 0
Description: The contents of this register specify the line from which an active video starts.

[31:13] **Reserved**

[12:0] **YMIN:** ymin value in pix clocks.

HDMI_ACTIVE_VIDEO_YMAX ymax 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																			YMAX												

Address: *HDMIBaseAddress* + 0x10C
Type: R/W
Reset: 0
Description: Maximum number of lines in a frame are programmed in this register.

[31:13] **Reserved**

[12:0] **YMAX:** Contains the ymax value in pix clocks..

Note: It is expected that xmin, xmax, ymin and ymax are programmed appropriately before device_enable bit is set. If the Values of xmin,ymin,xmax and ymax are at zero then the video frame will not start.

HDMI_DEFAULT_CHL0_DATA Default channel 0 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* + 0x110

Type: R/W

Reset: 0

Description: This register stores a default value of the data that has to be transmitted on channel0 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 channel1 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 channel2 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_CAPTURE_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_CHL1_CAPTURE_DATA Channel 1 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																								CHL1_DATA							

Address: *HDMIBaseAddress* + 0x120
Type: RO
Reset: 0
Description: The captured data corresponding to channel 1 is stored here.

[31:8] **Reserved**

[7:0] **CHL1_DATA**: Contains the channel 1 captured data

HDMI_CHL2_CAPTURE_DATA hannel 2 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																								CHL2_DATA							

Address: *HDMIBaseAddress* + 0x124
Type: RO
Reset: 0
Description: The captured data corresponding to channel 2 is stored here.

[31:8] **Reserved**

[7:0] **CHL2_DATA**: Contains the channel 2 captured data

HDMI_AUDIO_LAYOUT Audio layout

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																														CLOCK_SELECT	LAYOUT

Address: *HDMIBaseAddress* + 0x200
Type: R/W
Reset: 0
Description:

[31:2] **Reserved**

[1] **CLOCK_SELECT**
0: Use pixel clock for CTS calculation
1: Use TMDS clock for CTS calculation

[0] **LAYOUT**
0: Two channel audio
1: 8 channel audio

HDMI_INFOFRAME_HEADER_WORD 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.

[31:2] **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_INFOFRAME_PACKET_WORDN 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
packet_word_n*4+3								packet_word_n*4+2								packet_word_n*4+1								packet_word_n*4+0							

Address: *HDMIBaseAddress* + 0x214 + 4*n*

Type: R/W

Reset: 0

Description: There are seven registers corresponding to 28 bytes in the packet data. **The registers are from n = 0 to 6**.

[31:24] **packet_word_n*4+3**
Contains the packet bytes.

[23:16] **packet_word_n*4+2**
Contains the packet bytes.

[15:8] **packet_word_n*4+1**
Contains the packet bytes.

[7:0] **packet_word_n*4+0**
Contains the packet bytes.

HDMI_INFOFRAME_CONFIG 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																															ENABLE

Address: *HDMIBaseAddress* + 0x234

Type: R/W

Reset: 0

Description:

[31:1] **Reserved**

[0] **ENABLE**
1: Enable info frame with data in the FIFO

HDMI_INFOFRAME_FIFO_STATUS Info frame FIFO 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																			PACKET_BYTES					Reserved						HEADER_BYTES	BUSY

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] **PACKET_BYTES**
00000: FiFO empty
00001: One Byte in FIFO
00010: Two bytes in FIFO
.....
11100: 28 Bytes, FIFO full
11110: Undefined
11111: 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_MASK Sample flat mask

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																												SP3	SP2	SP1	SP0

Address: *HDMIBaseAddress* + 0x244

Type: R/W

Reset: 0

Description: If a bit is set, then the sample flat bit for the corresponding audio sub-packet in the audio sample packet will be set. Setting this bit overrides any internal generated conditions for the sample flat bit for the sub packet.

[31:4] **Reserved**

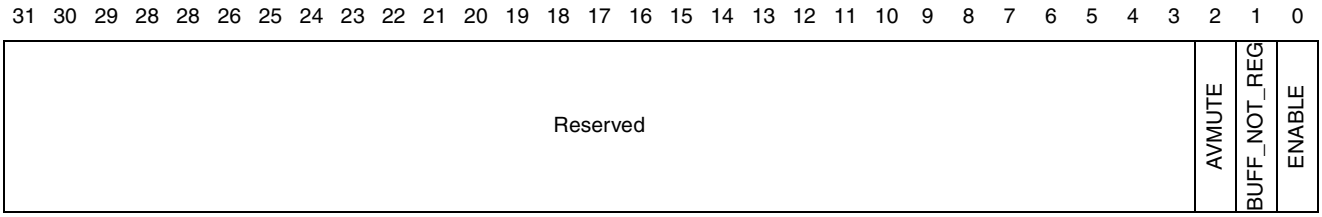
[3] **SP3**: Sample flat bit for sp3

[2] **SP2**: Sample flat bit for sp2

[1] **SP1**: Sample flat bit for sp1

[0] **SP0**: Sample flat bit for sp0

HDMI_GENCTRL_PACKET_CONFIG 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 the Bit0 (Enable bit) is set to '1'.

If bit1 (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 the bit1 (AVMUTE) is at '0' then 'Clear_AVMUTE' bit in the sub-packet is set and 'Set_AVMUTE' bit is reset. If this bit is set then Clear_AVMUTE' bit is reset and Set_AVMUTE' bit is set in the sub-packet. The structure of general control packet is shown in appendix D.2.5.

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 re-set to '0'. An interrupt is generated if enabled after completion of general packet transmission.

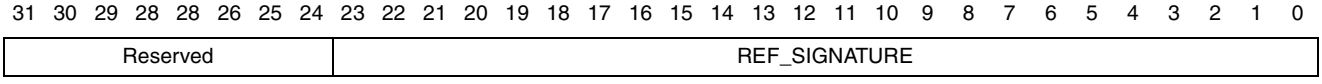
[31:3] **Reserved**

[2] **AVMUTE**
AVMute status

[1] **BUFF_NOT_REG**
0: AVMUTE inferred from next bit
1: Data in the buffer transmitted

[0] **ENABLE**
Enable General control packet Xmission

HDMI_MISR_REF_SIGNATURE1 MISR reference signature 1



Address: *HDMIBaseAddress* + 0x300

Type: R/W

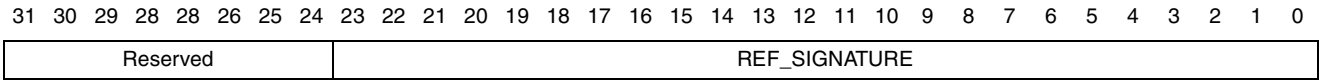
Reset: 0

Description:

[31:24] **Reserved**

[23:0] **REF_SIGNATURE**
MISR1/channel0 outputreference signature

HDMI_MISR_REF_SIGNATURE2 MISR reference signature 2

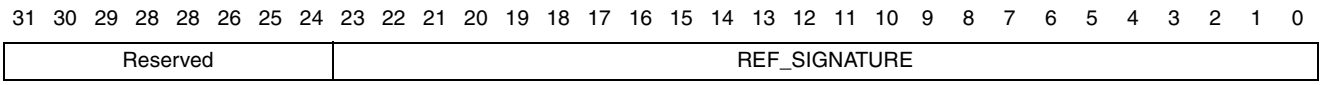


Address: *HDMIBaseAddress* + 0x304
Type: R/W
Reset: 0
Description:

[31:24] **Reserved**

[23:0] **REF_SIGNATURE**
MISR2/channel1 outputreference signature

HDMI_MISR_REF_SIGNATURE3 MISR reference signature 3

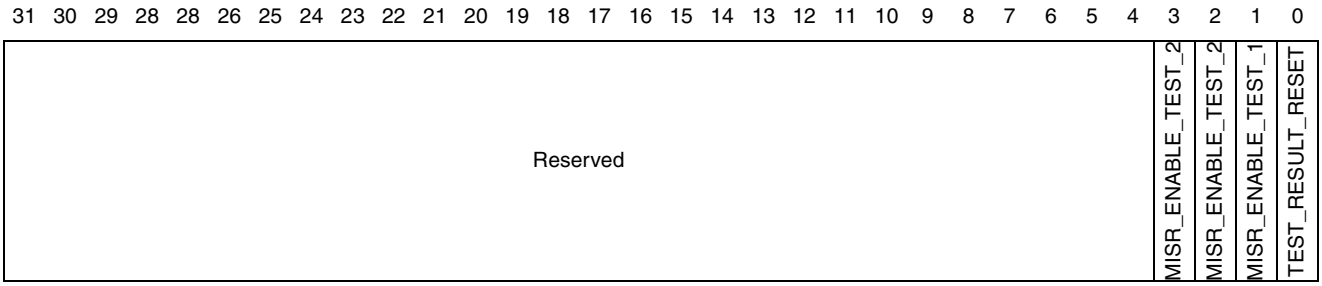


Address: *HDMIBaseAddress* + 0x308
Type: R/W
Reset: 0
Description:

[31:24] **Reserved**

[23:0] **REF_SIGNATURE**
MISR3/channel2 outputreference signature

HDMI_MISR_TEST_CONTROL MISR test control



Address: *HDMIBaseAddress* + 0x30C

Type: R/W

Reset: 0

Description:

[31:4] **Reserved**

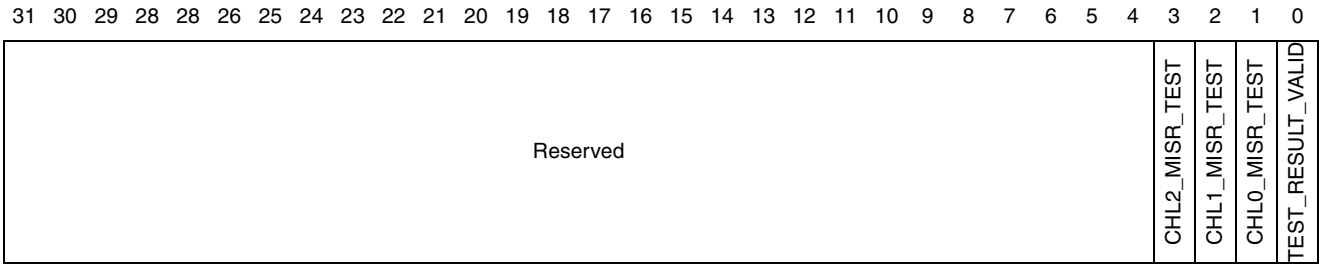
[3] **MISR_ENABLE_TEST_2**
1: signature genration and comparision enabled for MISR3

[2] **MISR_ENABLE_TEST_2**
1: signature genration and comparision enabled for MISR2

[1] **MISR_ENABLE_TEST_1**
1: signature genration and comparision enabled for MISR1

[0] **TEST_RESULT_RESET**
1: Resets the hdmi_misr_test_status register, self-resets to ‘0’

HDMI_MISR_TEST_STATUS MISR test status



Address: *HDMIBaseAddress* + 0x310

Type: RO

Reset: 0

Description:

[31:4] **Reserved**

[3] **CHL2_MISR_TEST**
0: Test pass
1: Test fail

[2] **CHL1_MISR_TEST**
0: Test pass
1: Test fail

[1] **CHL0_MISR_TEST**
0: Test pass
1: Test fail

[0] **TEST_RESULT_VALID**
1: Test complete

66 High-bandwidth digital content protection (HDCP)

66.1 Detailed 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.

66.1.1 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_CNTRL has been deasserted.

Commands are issued from software by writing to the HDCP_AUTH_CMD register.

Command summary

Table 190: Command summary

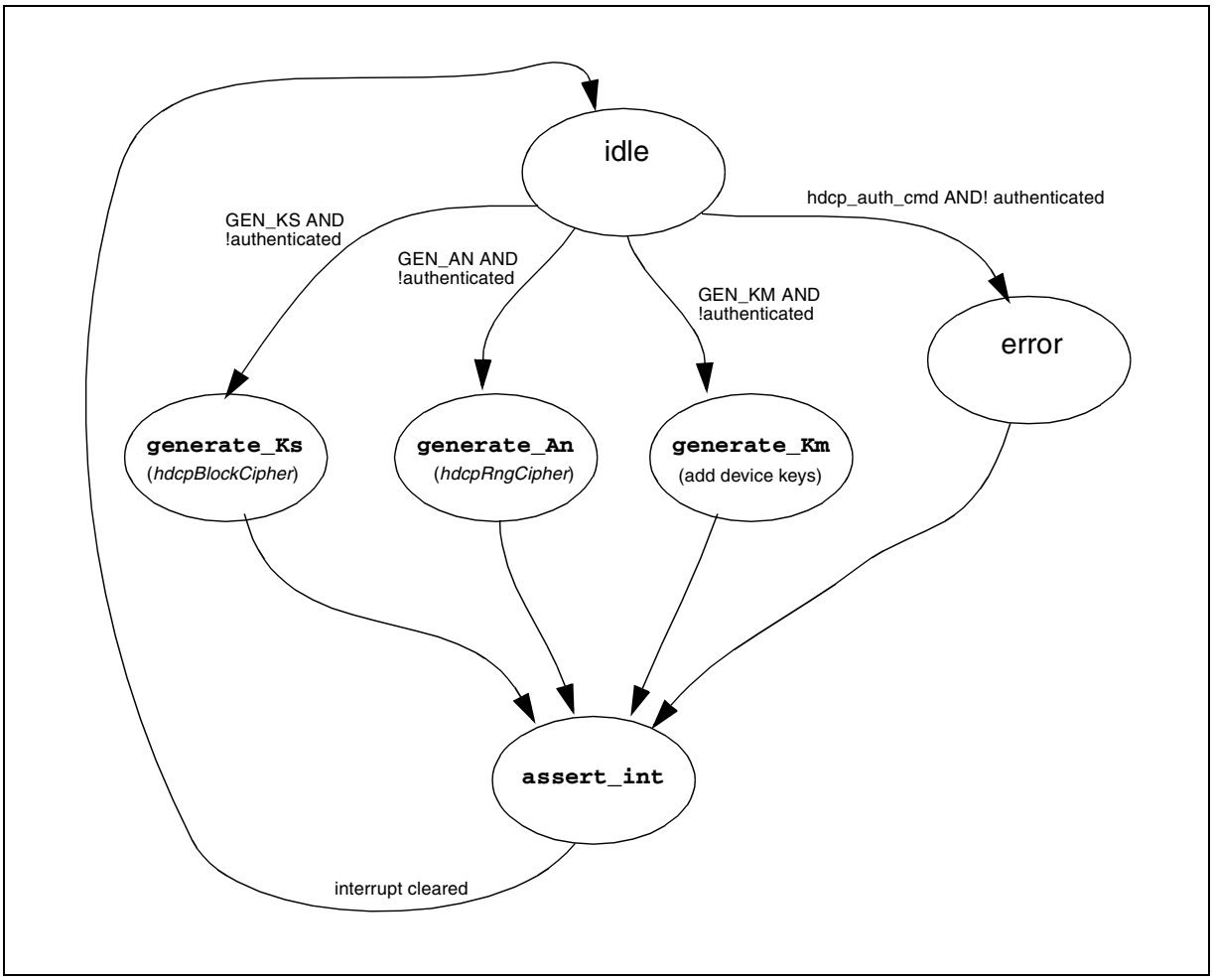
Code	Command	Function
0x01/0x81	**generate_An**	Perform **hdcpRngCipher** function to generate 64-bit Pseudo-random value, *An*, and store in the AN register. *Note: Here, bit 7 of* HDCP_AUTH_CMD *corresponds to Repeater bit (concatenate this bit to An result for storage internally)*
0x02	**generate_Km**	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	**generate_Ks**	perform **hdcpBlockCipher**, using *Km* and *RPT||An* from previous commands to calculate the Session Key, Ks (and also M_0 and R_0, into data register)

Whenever a command has completed successfully, O_INTERRUPT is asserted and HDCP_AUTH_CMD_DONE is set in HDCP_STATUS. 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 AUTHENTICATED bit of HDCP_CNTRL register is set, HDCP_AUTH_CMD_ERROR bit is set in HDCP_STATUS register, and the interrupt is asserted. The status bit and the interrupt are cleared by writing 1 to HDCP_AUTH_CMD_ERROR.

Figure 201: HDCP authentication command flow abstraction shows the modes of operation used by the HDCP Cipher hardware when carrying out authentication commands.

Figure 201: HDCP authentication command flow abstraction



66.1.2 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 204: HDCP data encryption flow abstraction shows the real-time flow taken when encrypting data on the interface. Figure 202: HDCP/HDMI timing diagram shows the physical signal interactions between HDCP and HDMI during a typical frame.

Note: 1 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*).*

2 After I_VIDEO_DATA_ACTIVE *has been deasserted,* I_DATA_ISLAND_ACTIVE *must not be asserted for at least 58 cycles (Line Key being calculated in this period).*

3 I_VSYNC *is assumed to take priority over* I_HSYNC*, if both are asserted simultaneously.*

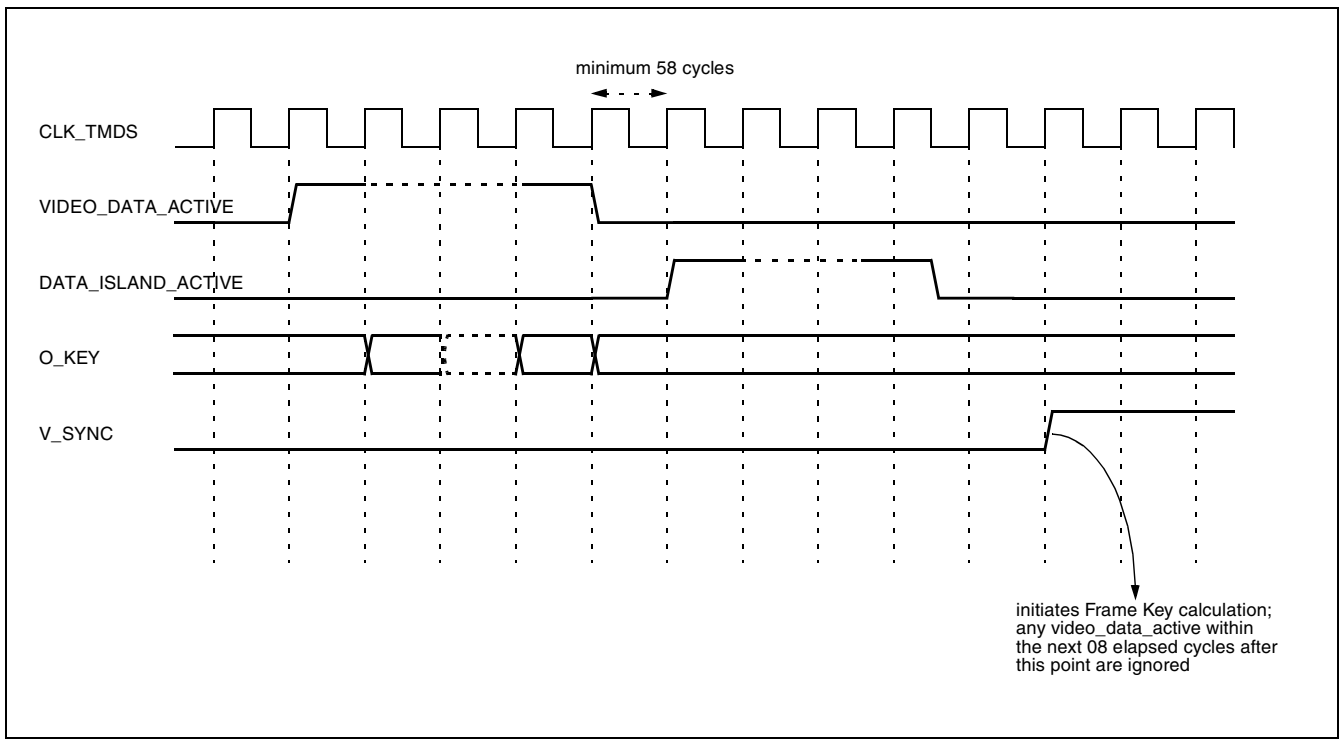
66.1.2.1 O_ENC_ENABLE

This signal is asserted by HDCP when it is authenticated, the ENC_ENABLE register is set, and the AV_MUTE register is not set.

66.1.2.2 O_RX_PIXEL_CAPTURE

Ri is updated every 128th frame (frames are counted as those for which encryption is **enabled**), starting with the 128th frame, unless specified otherwise by I_RATE register. The O_RX_PIXEL_CAPTURE output is asserted for one cycle when the value of *Ri* has been updated (this will occur 108 TMDS cycles after the rising edge of V_SYNC). This allows the HDMI to update its value of *Ri*.

Figure 202: HDCP/HDMI timing diagram



66.2 Software usage

Software drivers should perform the following functions in order to program and use the HDCP Block. See also Figure 203: 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`** 0x01/81 and awaiting the interrupt to indicate completion.
2. **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.

2 For device key encryption, the HDCP Cipher is used in a 'Cipher Feedback Mode' fashion

2.1 For initialization values for B and K registers (MSB on LHS):

b_init <= internal_tied_value(111:56) & iv(55:28)

k_init <= internal_tied_value(55:0) & iv(27:0)

2.2 Clock the block module of the HDCP Cipher for 10 cycles

2.3 *devkey_cipher* <= *devkey_plain* XOR b_out(55:0)

2.4 Repeat steps 2.1 to 2.4 for subsequent keys, but substitute the previous *devkey_cipher* for the *iv* value. '*internal_tied_value*' is communicated separately.

3. **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.

4. **Generate a session key**

   Calculate *Ks* (and *M* and *R*) value by issuing **`hdcp_cmd`** 0x03 and awaiting the interrupt to indicate completion.

At this point the software **must** set the AUTHENTICATED bit in the HDCP_CNTRL register. 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, the value of *R* will be updated every 128 frames. If more often 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_ENABLE 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 the AUTHENTICATED bit in HDCP_CNTRL register, and then re-start the authentication process.

Figure 203: Software flow

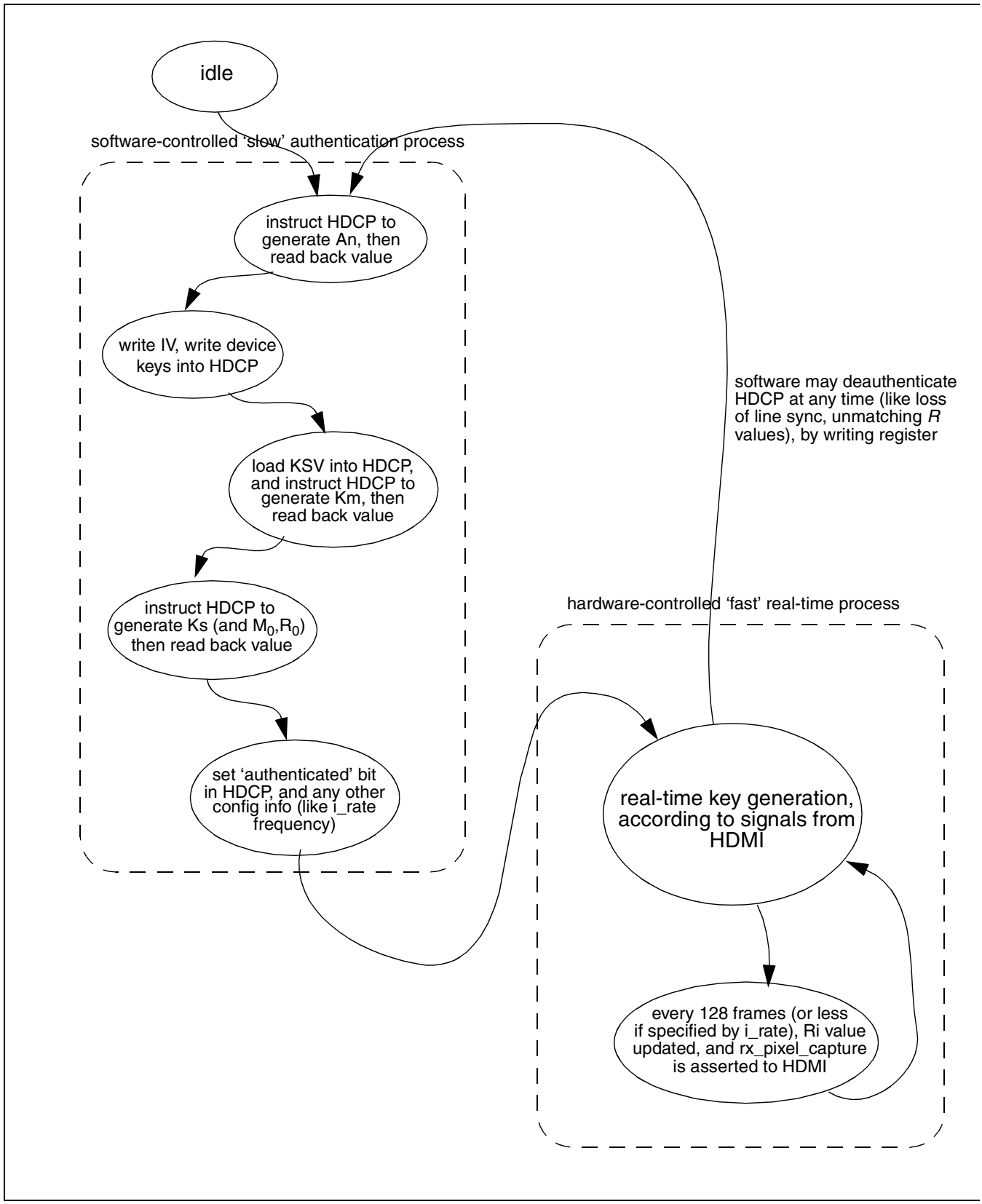
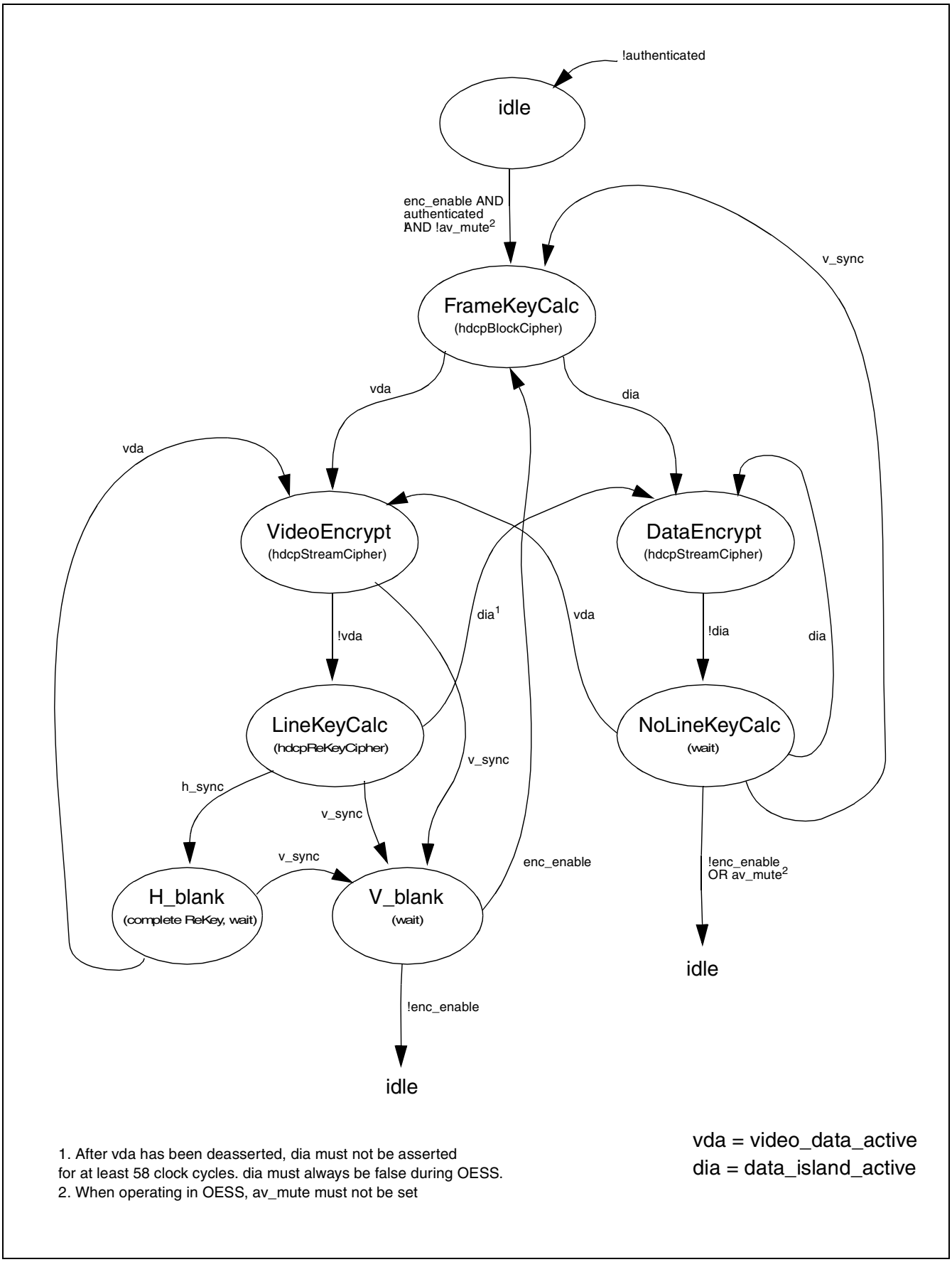


Figure 204: HDCP data encryption flow abstraction



67 HDCP registers

Register addresses are provided as *HDCPBaseAddress* + offset.

The *HDCPBaseAddress* is:

0x2010 3600.

Table 191: HDCP register summary

Register	Description	Offset	Type
HDCP_AUTH_CMD	Authorization command	0x0000	R/W
HDCP_CNTRL	real time control	0x0004	R/W
HDCP_I_RATE	i_rate	0x0008	R/W
HDCP_STATUS	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

31	30	29	28	27	26	25	24	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																												AUTH_CMD			

Address: *HDCPBaseAddress* + 0x0000
Type: R/W
Reset: 0
Description:

[31:4] **Reserved**

[3:0] **AUTH_CMD**
0x01/0x81: **`generate_An`**
0x02: **`generate_Km`**
0x03: **`generate_Ks`**
Other: Reserved

HDCP_CNTRL Real time 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																													AV_MUTE	AUTHORISED	ENC_ENABLE

Address: *HDCPBaseAddress* + 0x0004
Type: R/W
Reset: 0
Description:

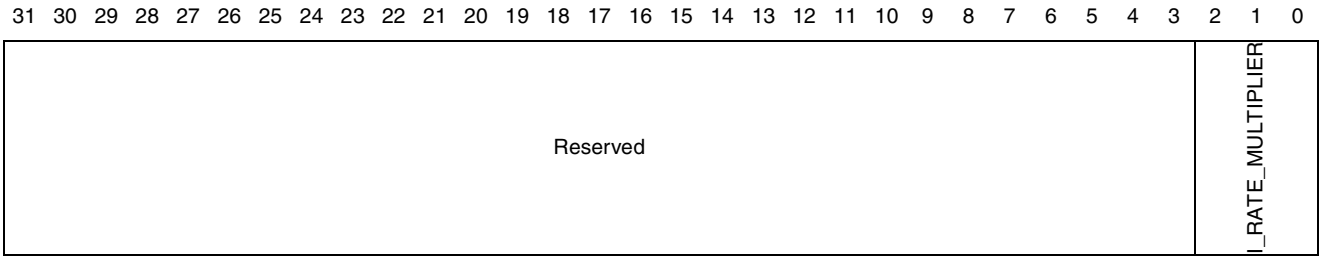
[31:3] **Reserved**

[2] **AV_MUTE:** encryption
1: temporarily disable
0: enable

[1] **AUTHORISED**
1: Yes (enable encryption).
0: No (disable encryption).

[0] **ENC_ENABLE**
1: Enable
0: Disable

HDCP_I_RATE I rate multipler



Address: *HDCPBaseAddress* + 0x0008

Type: R/W

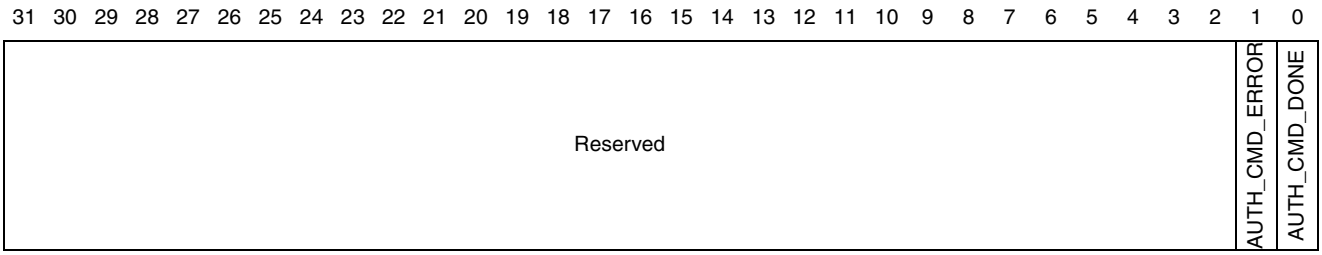
Reset: 0

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_STATUS Status



Address: *HDCPBaseAddress* + 0x000C

Type: R/W

Reset: 0

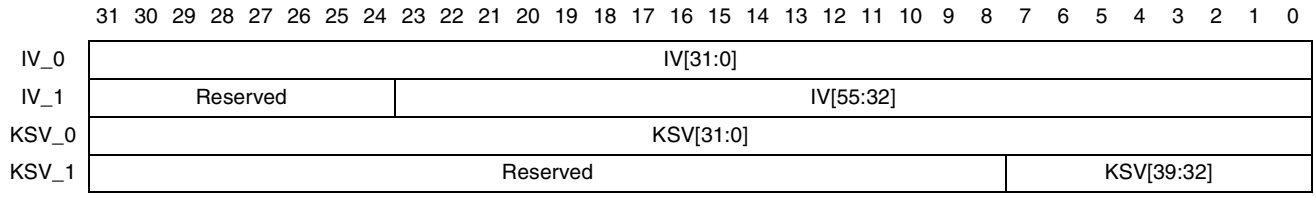
Description:

[31:2] **Reserved**

[1] **AUTH_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**
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 Shared secret/initialization vector



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 CFB decryption of the device keys.

IV_0: [31:1] **IV[31:0]**

IV_1: [31:24] **Reserved**

[23:0] **IV[55:32]**

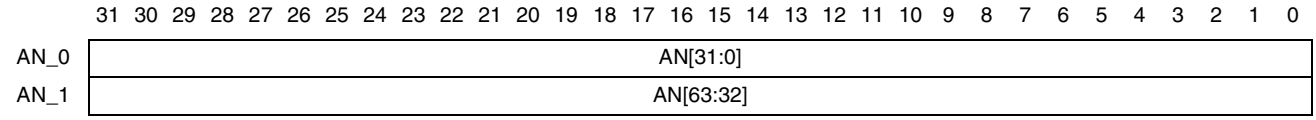
KSV: After device key loading, write the KSV value.

KSV_0: [31:0] **KSV[31:0]**

KSV_1: [31:8] **Reserved**

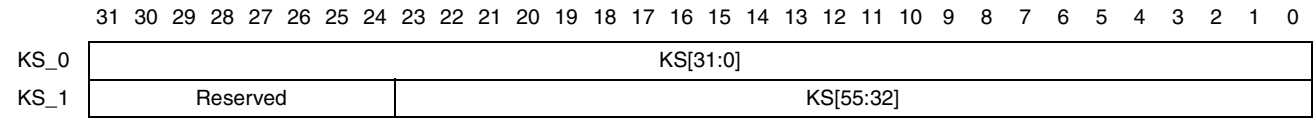
[7:0] **KSV[55:32]**

HDCP_AN_n Pseudo-random value



Address: *HDCPBaseAddress* + 0x0018 (AN_0); + 0x001C (AN_1)
Type: RO
Reset: 0
Description: 64-bit pseudo-random value

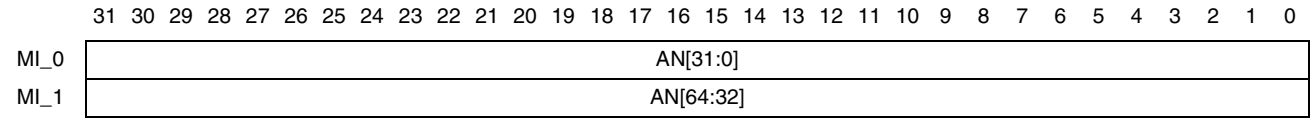
HDCP_KS_n Session key



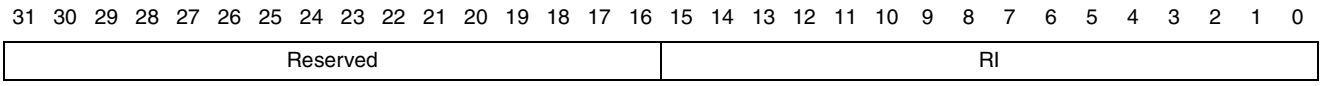
Address: *HDCPBaseAddress* + 0x0020 (KS_0); + 0x0024 (KS_1)
Type: RO
Reset: 0
Description: Session key

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

68 Digital encoder (DENC)

68.1 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 DAC's on 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 STi7710, 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 STi7710. The VTGs are programmed by registers described in Chapter 59: *Video output stage (VOS) registers on page 560*. 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 STi7710. The DENC registers are described in Chapter 69 on page 770.

68.2 Data input format

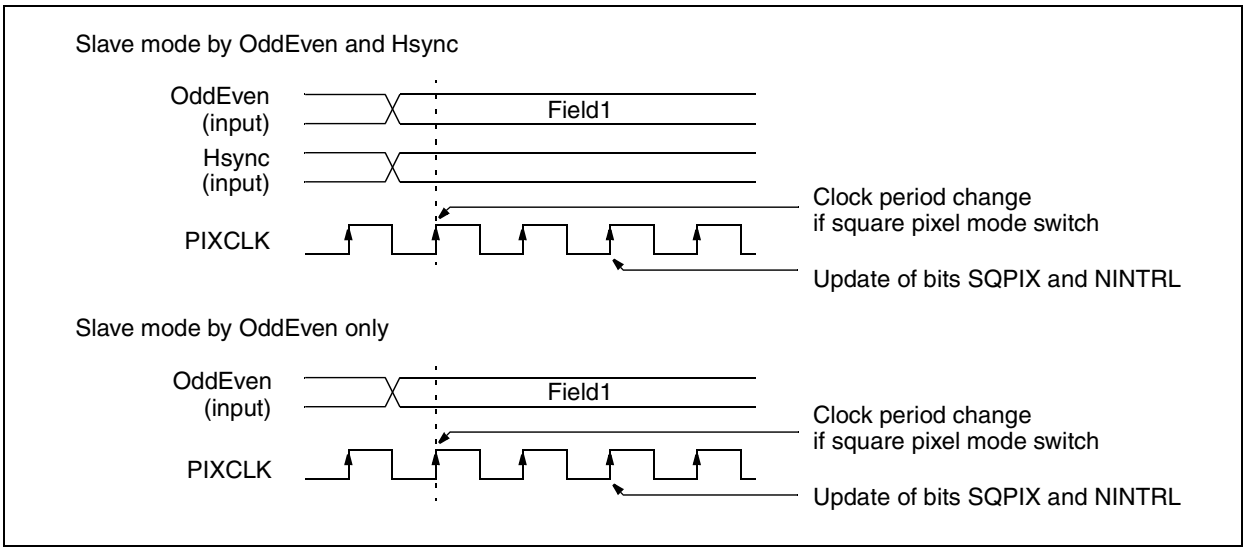
The digital input is a time-multiplexed [[Cb,Y,Cr],Y] 8-bit video stream or 4:4:4 [Y,Cb,Cr] 24-bit video stream. Input samples are latched on the rising of the clock signal PIXCLK.

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 205).

Figure 205: 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.)

68.3 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 68.9).

Figure 206 to Figure 211 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 or StarSight data). 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 206: PAL-BDGHI, PAL-N typical VBI waveform, interlaced mode (ITU-R625 line numbering)

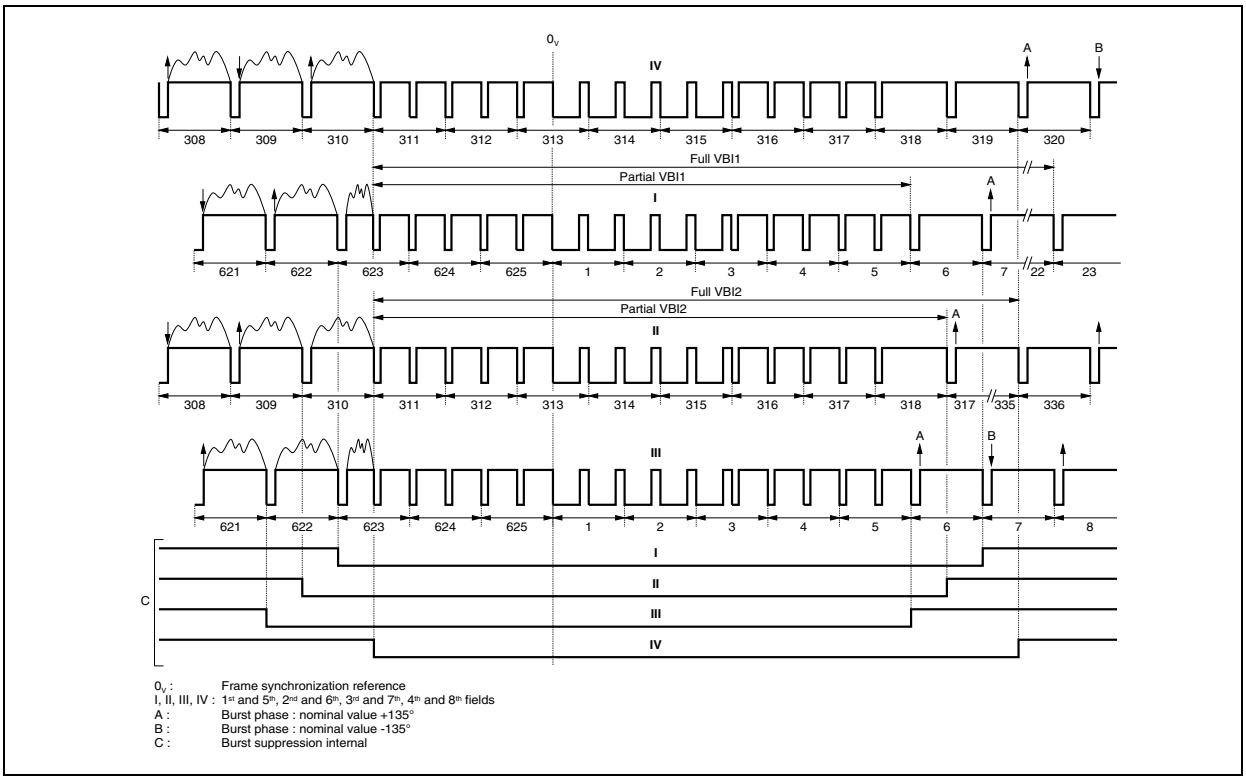


Figure 207: PAL-BDGHI, PAL-N typical VBI waveform, non-interlaced mode (CCIR-like line numbering)

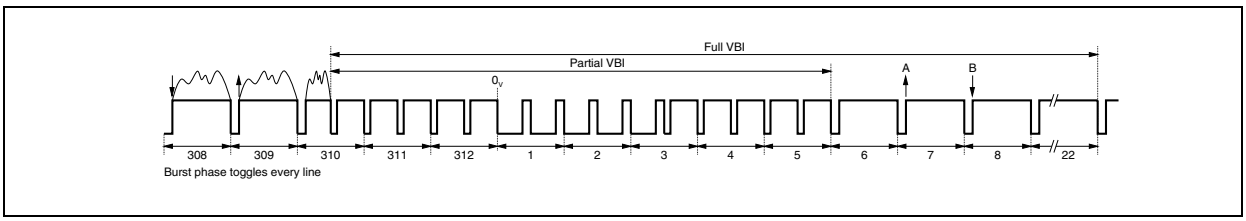


Figure 208: NTSC-M typical VBI waveforms, interlaced mode (SMPTE-525 line numbering)

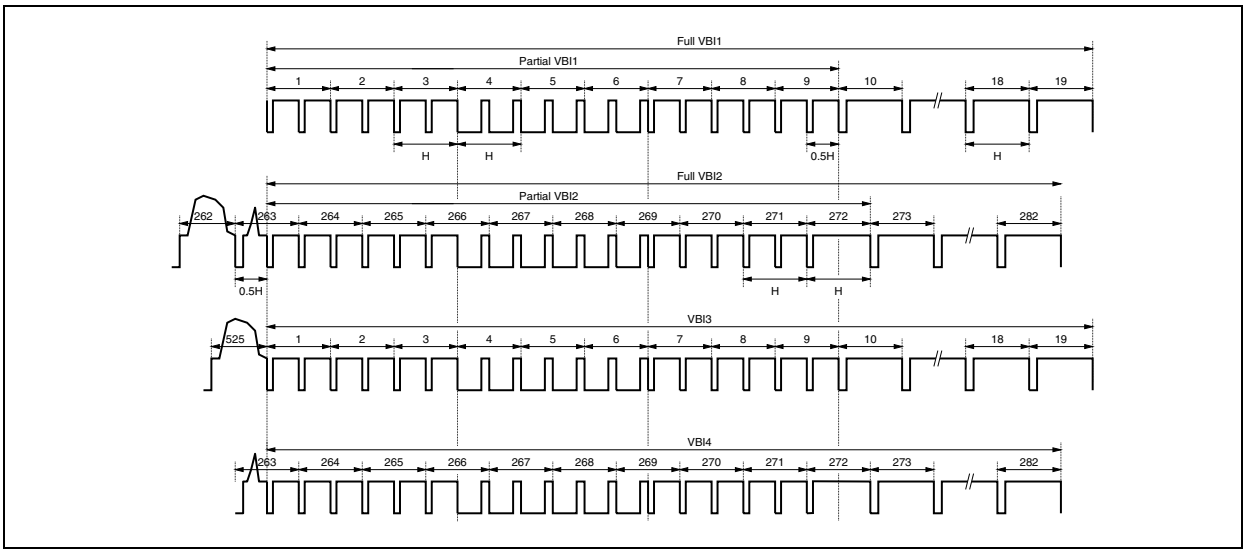


Figure 209: NTSC-M typical VBI waveforms, non-interlaced mode (SMPTE-like line numbering)

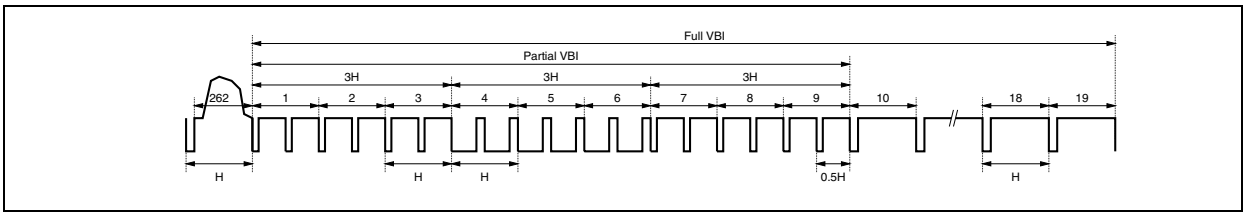


Figure 210: PAL-M typical VBI waveforms, interlaced mode (ITU-R/CCIR-525 line numbering)

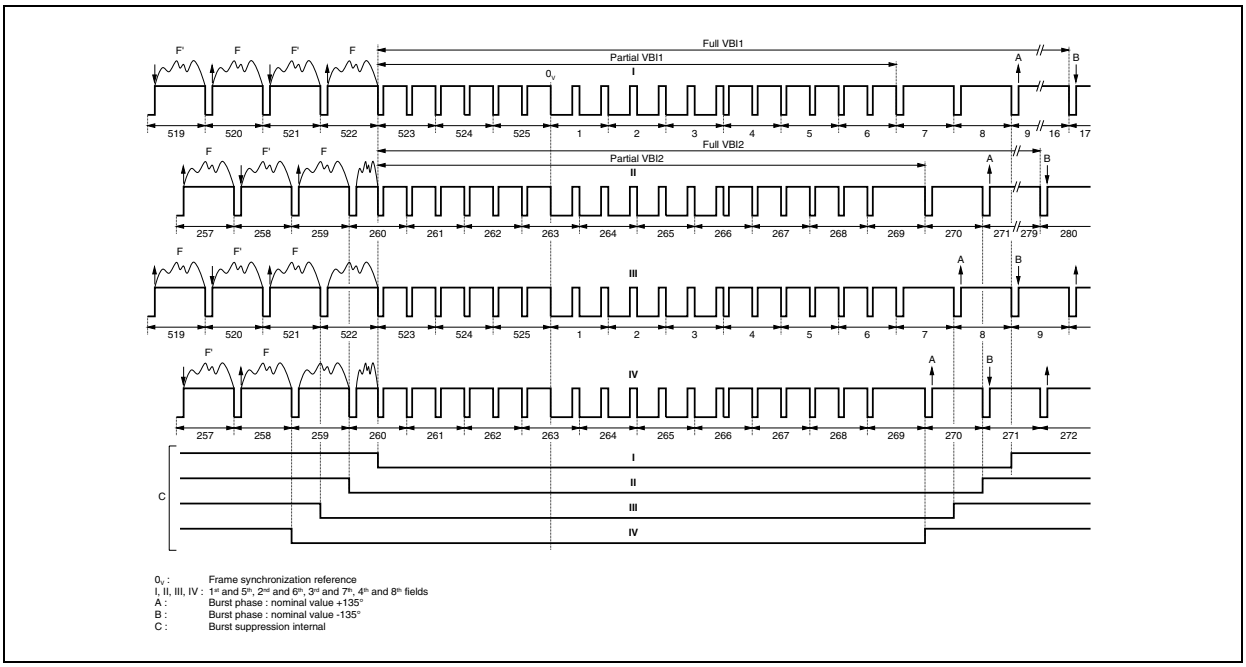
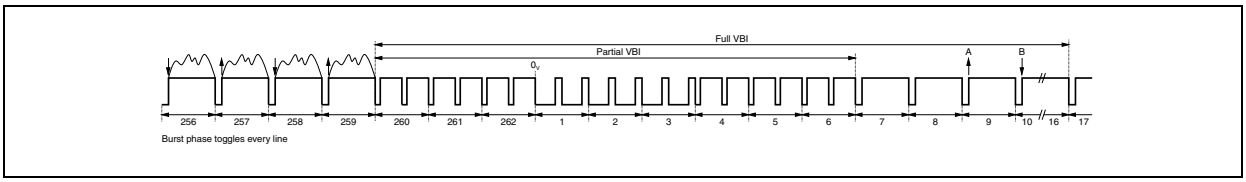


Figure 211: PAL-M typical VBI waveforms, non-interlaced mode (ITU-R/CCIR-like line numbering)



68.4 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 69 on page 770).

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.

68.5 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 STi7710's video timing generators VTGn (see output stage Section 58.3: *Video timing generators (VTG) on page 546*).

68.5.1 Line-based synchronization

OddEven + Hsync based synchronization

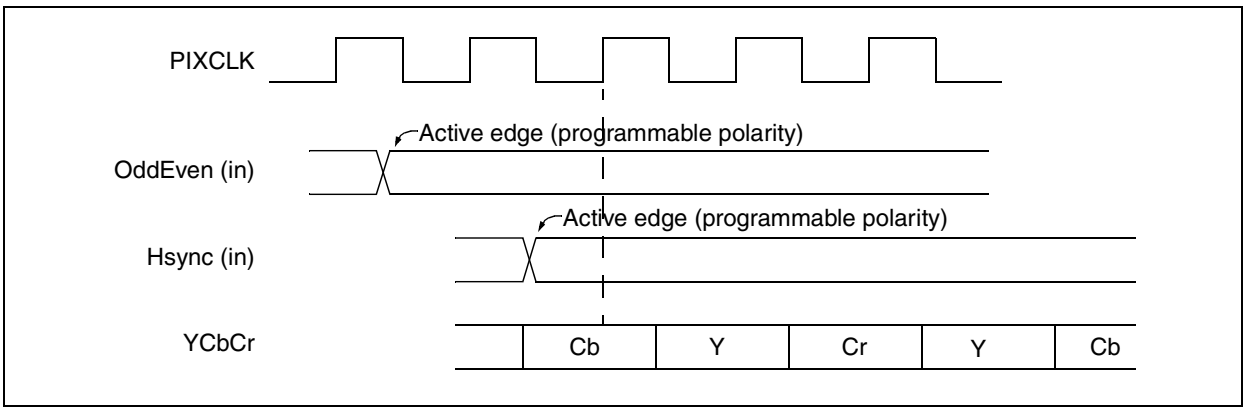
Synchronization is performed on a line-by-line basis by locking onto incoming OddEven and Hsync signals. Refer to Figure 212 for waveforms and timings. The polarities of the active edges of Hsync and OddEven are programmable and independent.

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 212: Hsync + OddEven based slave mode sync signals



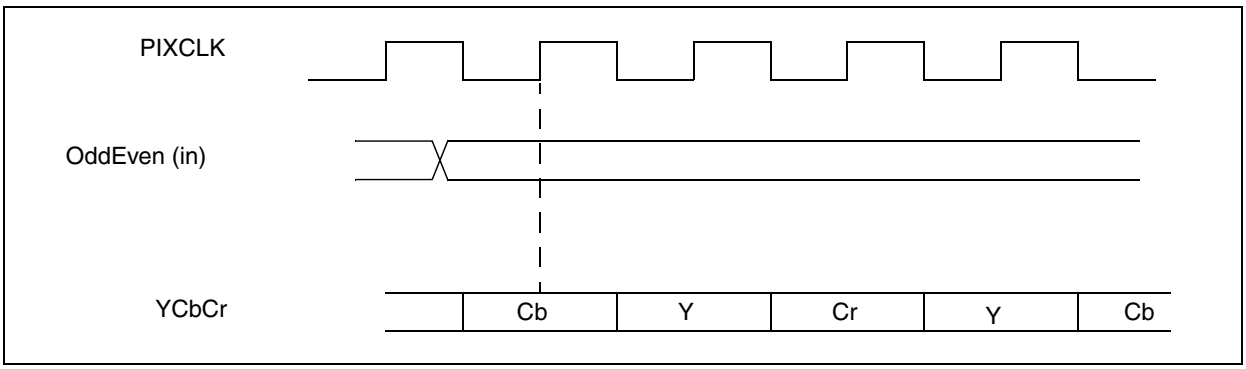
Note: This figure is valid for bits SYNCIN_AD = default.

68.5.2 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 213 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 213: 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.

68.5.3 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 192 shows the decimal values of Y, Cr and Cb corresponding to the autotest color bar.

Table 192: 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 214 and Figure 215. Both figures show the static values corresponding to the input values in Table 192.

Figure 214: Luminance output levels in autotest for NTSC without set-up

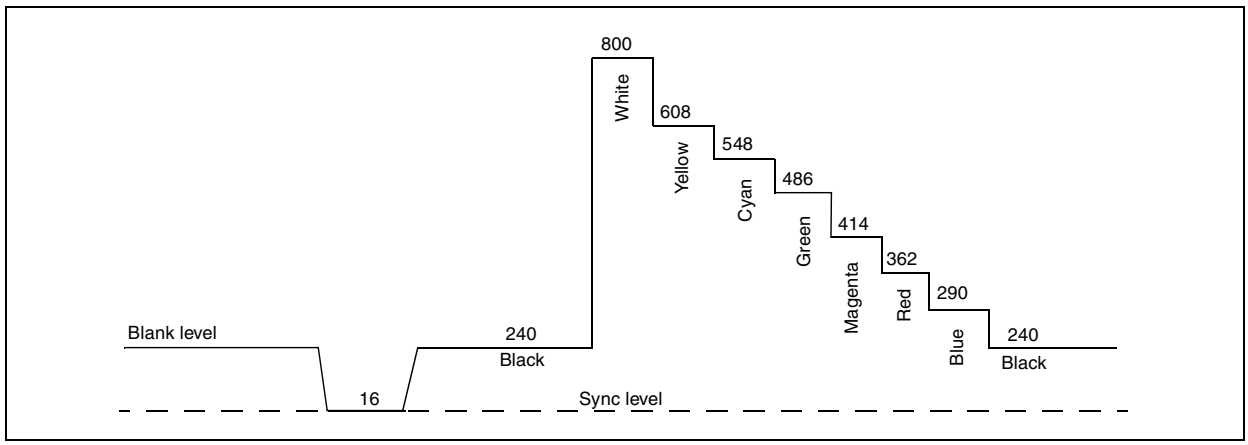
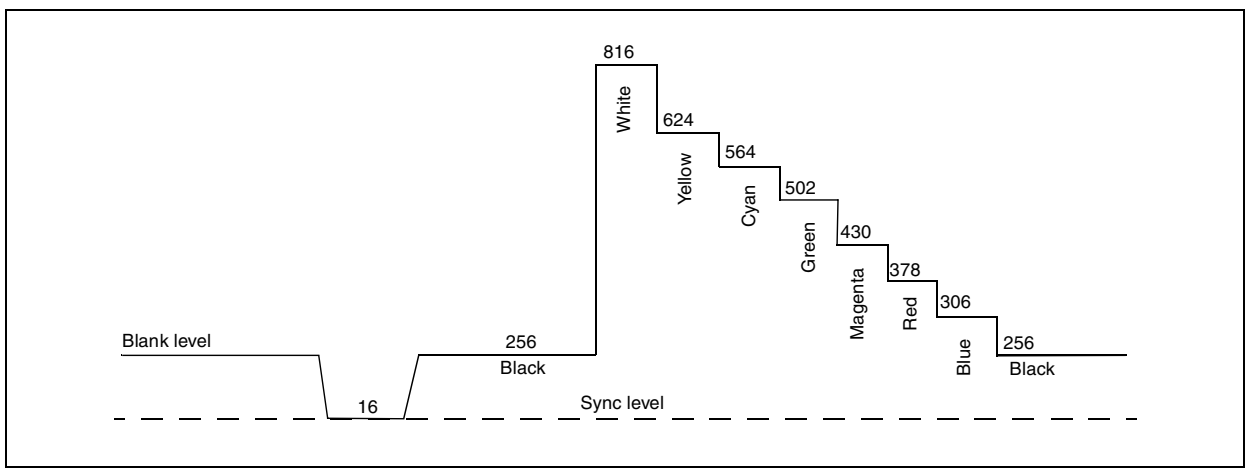


Figure 215: Luminance output levels in autotest for PAL (BGHI) and SECAM



68.6 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.

68.7 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:

$$Fsc = (Increment\_Word / 2^{24}) \times PIXCLK$$

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.

68.8 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.

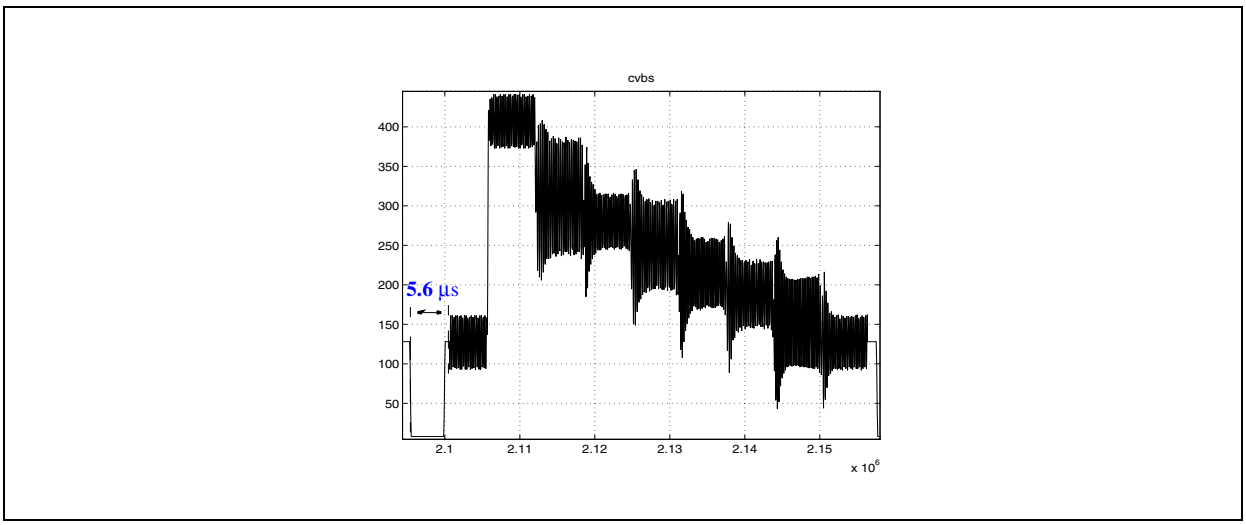
In NTSC, the relation between subcarrier frequency and line length creates a 180° subcarrier phase difference with respect to horizontal sync from one line to the next, according to line parity. So if the burst always starts with the same phase (positive-going zero crossing), the burst is inserted at 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.

68.9 Subcarrier insertion (SECAM)

Figure 216: 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 216.

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.

68.10 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 217, 218 and 219 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_LCOEF0..9. This filter is described in Section 68.11: *Chrominance encoding on page 760*.

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 217: Luma filtering including DAC attenuation

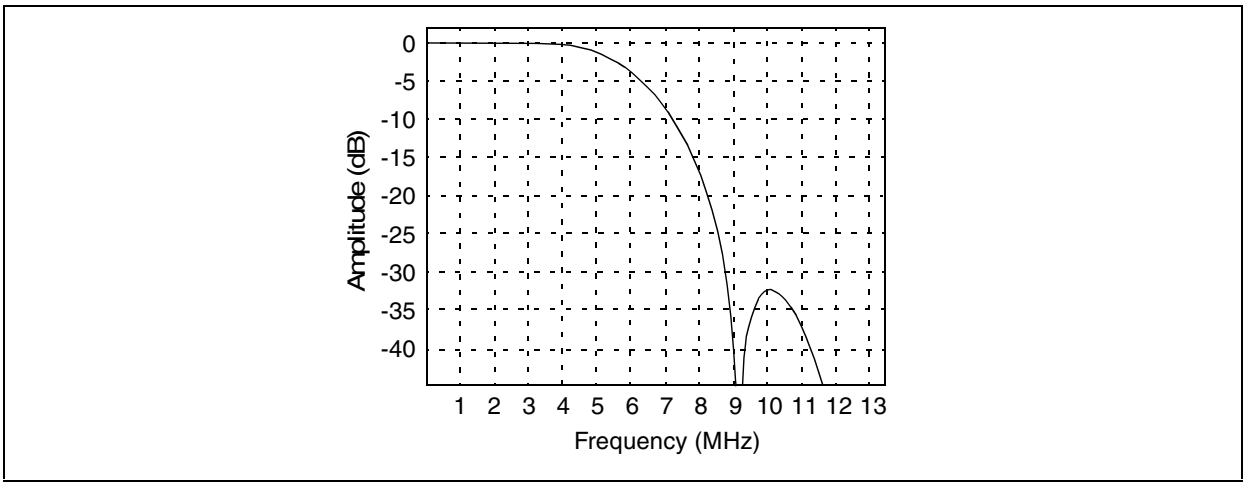


Figure 218: Luma filtering with 3.58 MHz trap, including DAC attenuation

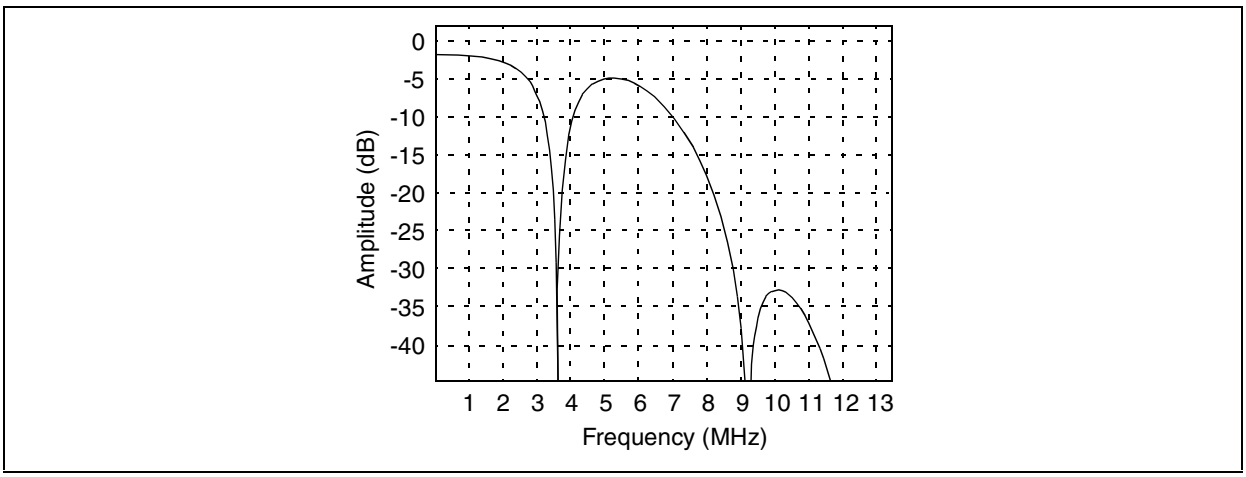
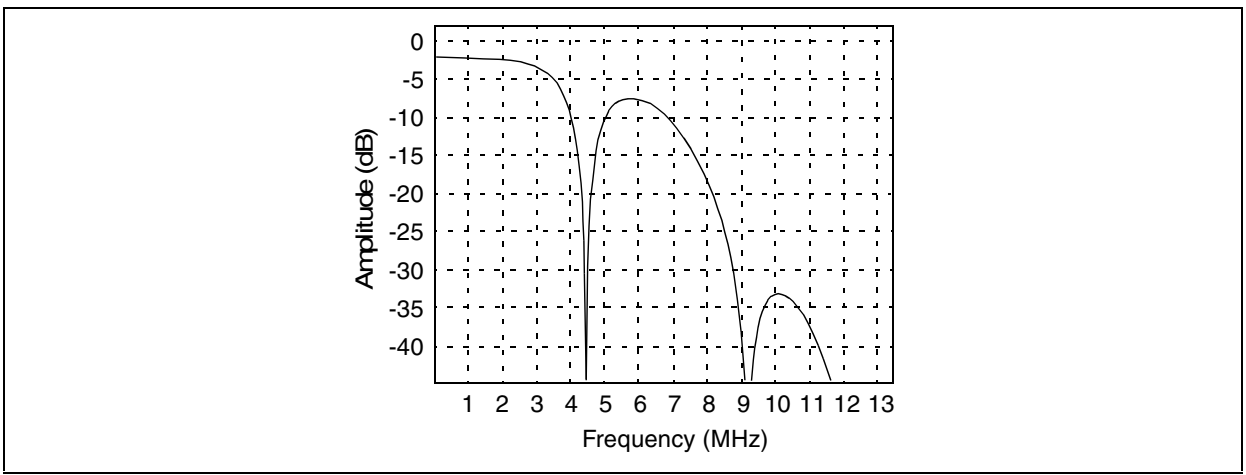


Figure 219: Luma filtering with 4.43 MHz trap, including DAC attenuation



68.11 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 222, 223, 224, 225 and 226 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 $\cos(\pi \times f/(2 \times fck))$, where fck 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 220 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 221 for frequency response of the bell filter with subcarrier frequencies and clipping values.

Figure 220: SECAM chroma filtering (pre-emphasis and 1.3 MHz low pass filtering)

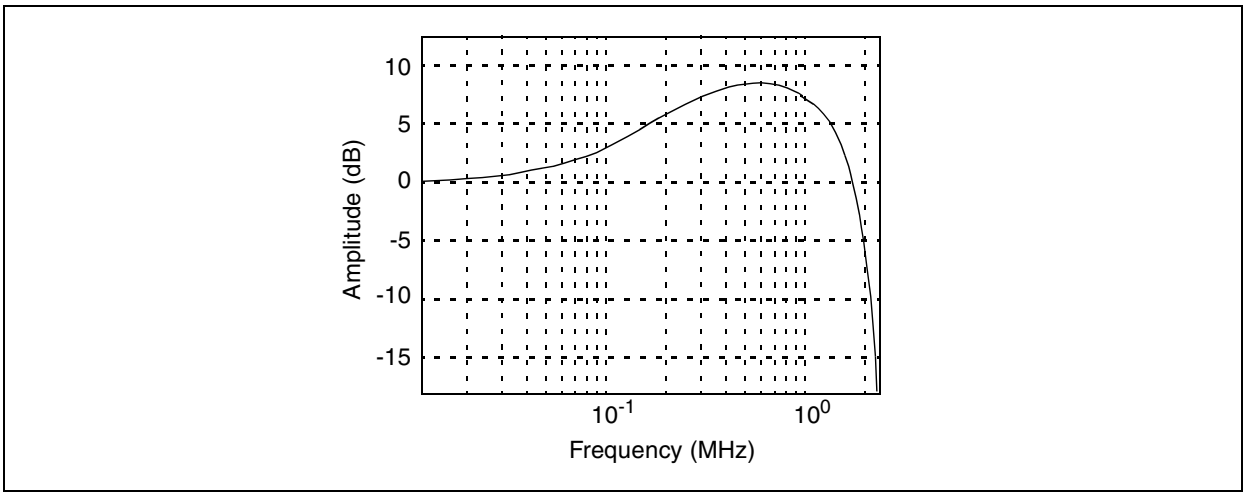


Figure 221: SECAM high-frequency subcarrier pre-emphasis (Bell filtering), including DAC attenuation

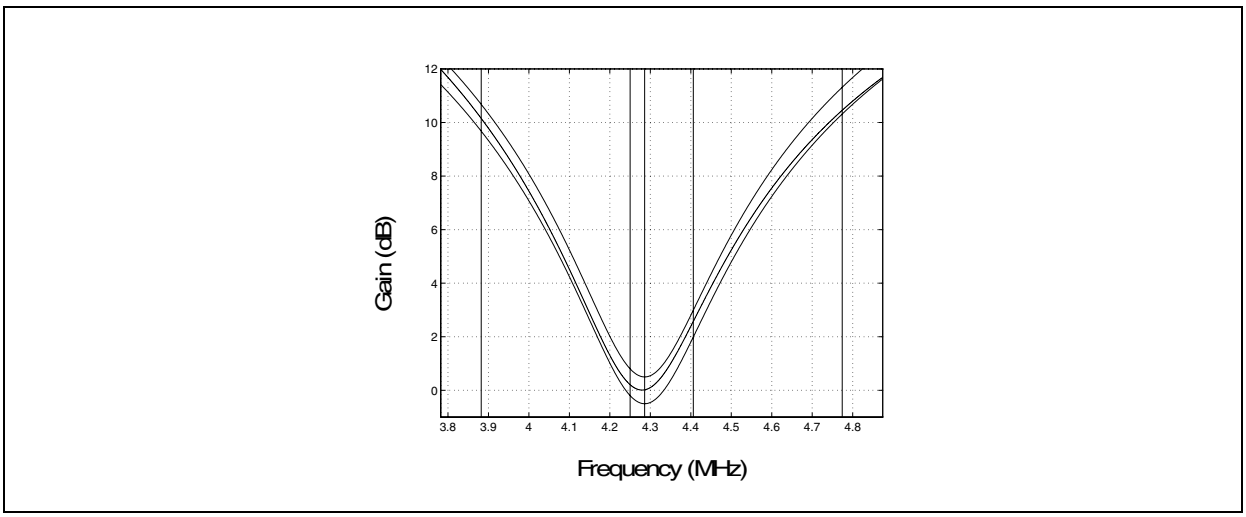
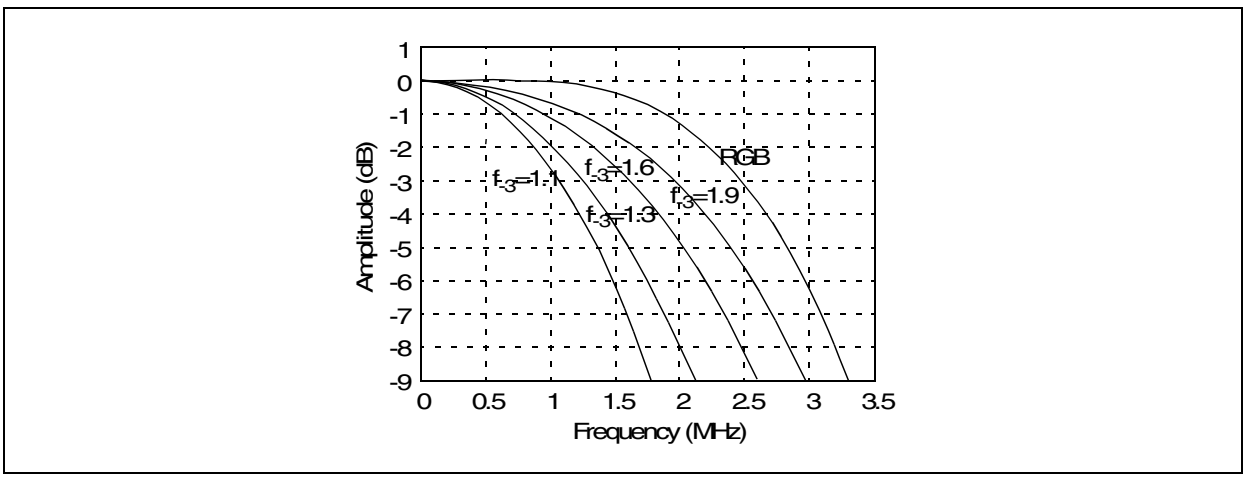


Figure 222: Various chroma filters available and RGB filter



68.12 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 223: 1.1 MHz chroma filter

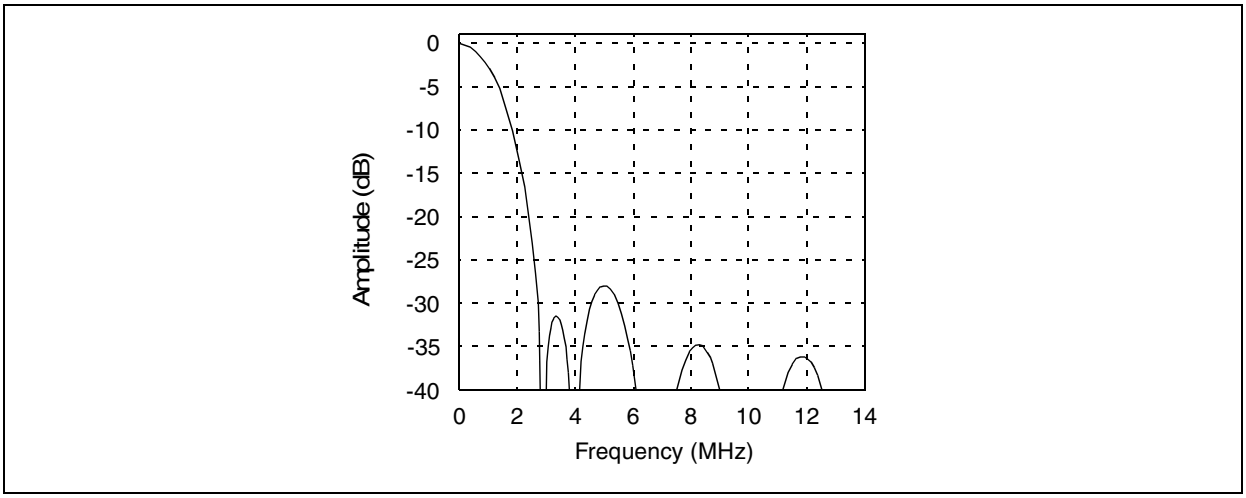


Figure 224: 1.3 MHz chroma filter

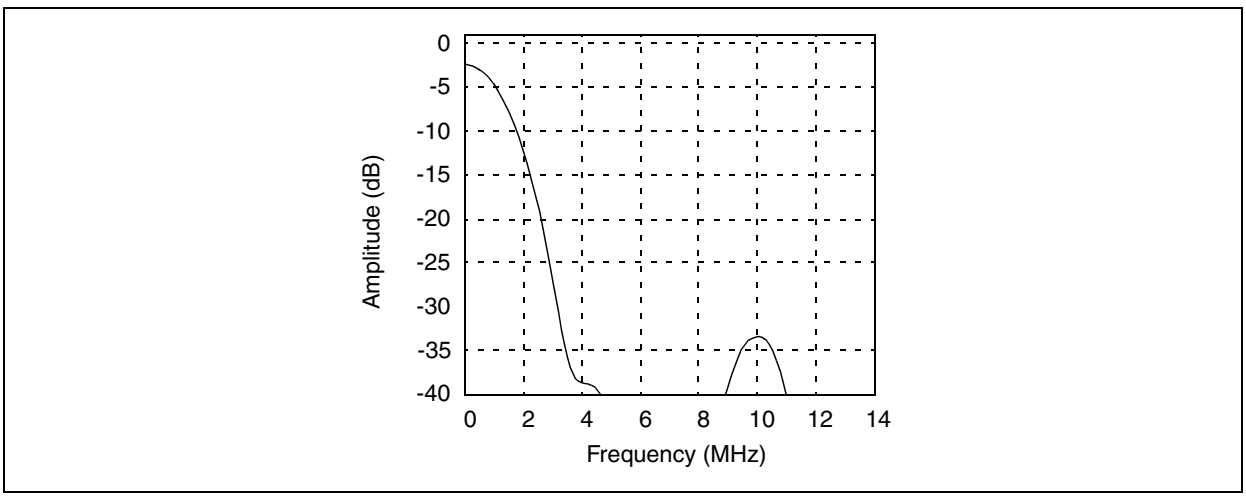


Figure 225: 1.6 MHz chroma filter

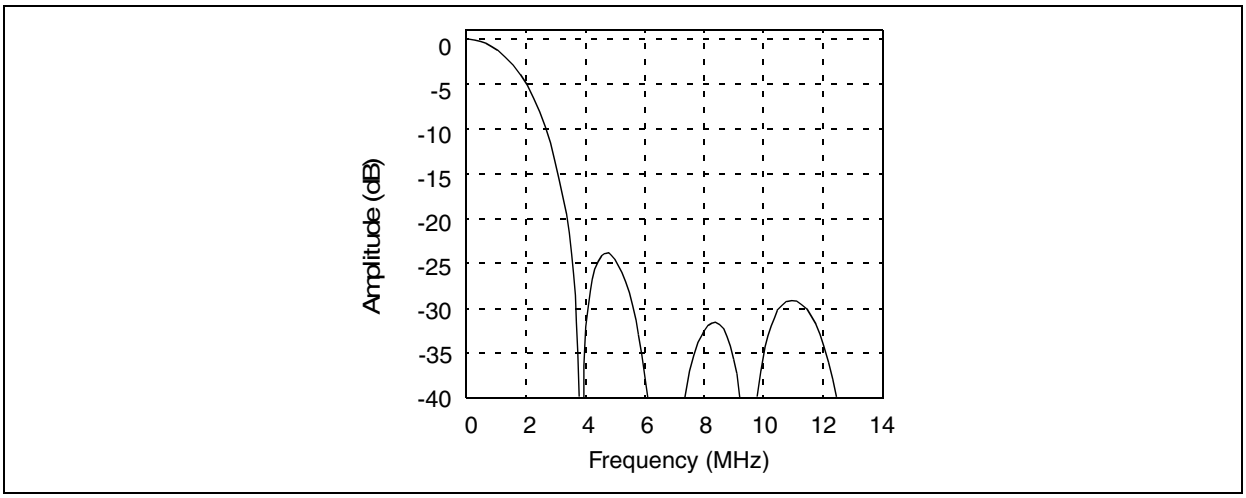
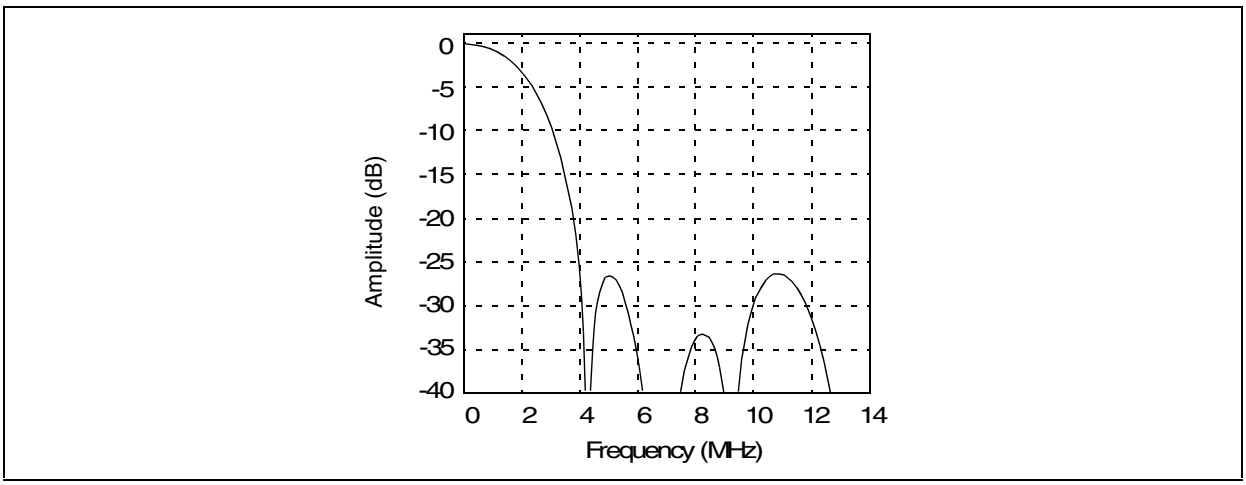


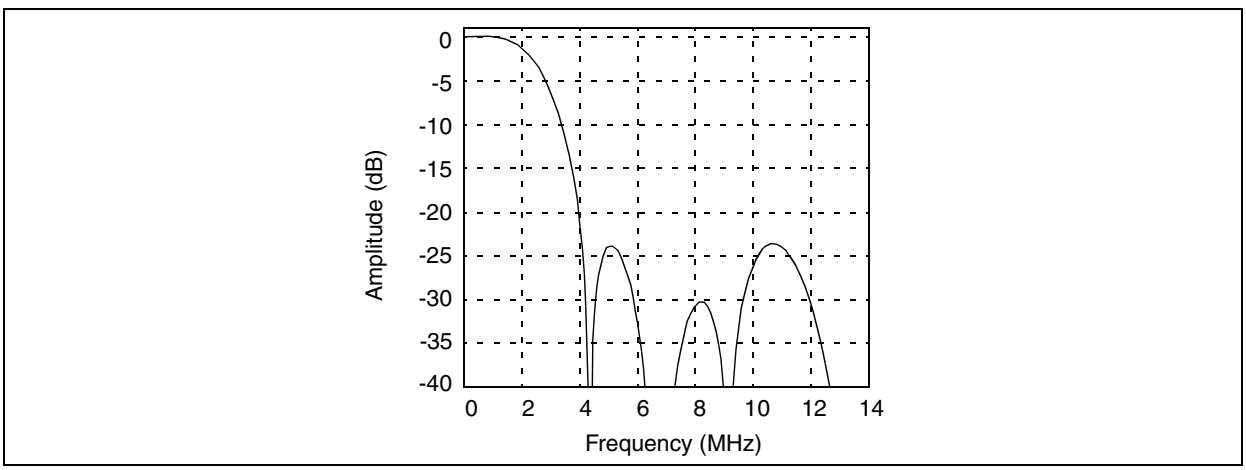
Figure 226: 1.9 MHz chroma filter



68.13 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 227) and is combined with the filtered luma component to generate R,G,B or U,V samples at 27 MHz.

Figure 227: RGB - chroma filtering



68.14 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.

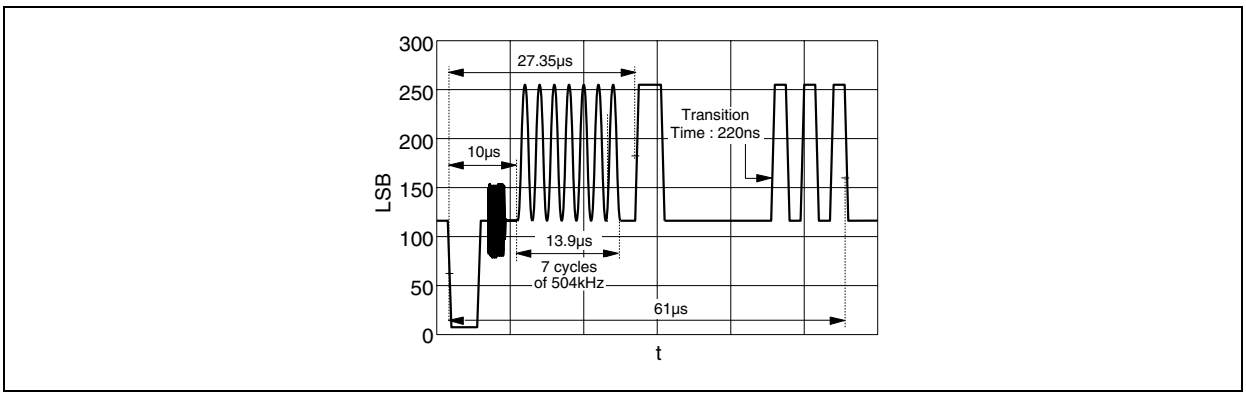
User registers DENC_CCCF1 and DENC_CCCF2 each contain the first and second byte to send (LSB first) after the start bit on the appropriate TV line, where DENC_CCCF1 refers to field 1 and DENC_CCCF2 to field 2. The TV line number where data is to be encoded is programmable using registers DENC_CCLIF1 and DENC_CCLIF2. Selectable lines include those used by the StarSight data broadcast system. 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_STATUS 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 228: Example of closed-caption waveform

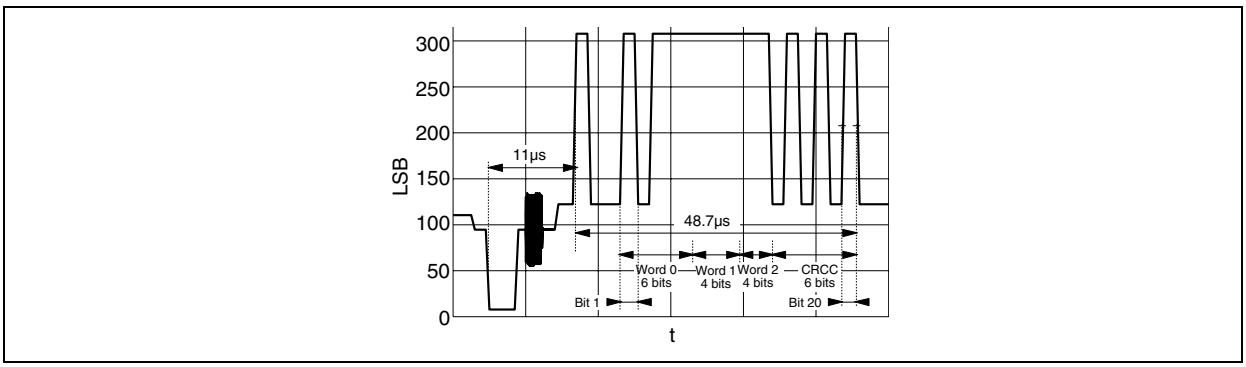


68.15 CGMS encoding

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 229 shows a typical CGMS waveform.

Figure 229: 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.

68.16 WSS encoding

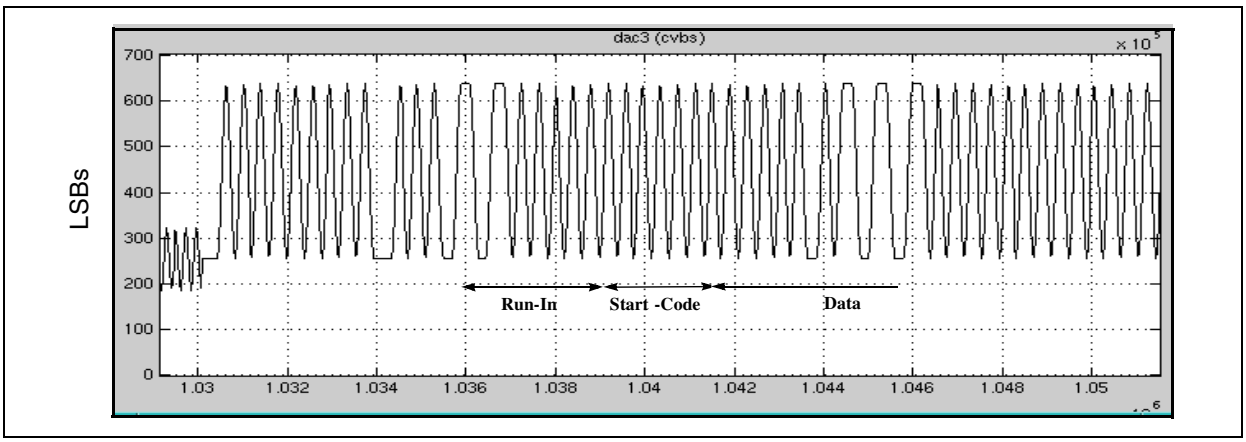
The digital encoder allows WSS (wide screen signalling) in 625-line format, complying with the ETS 300 294 standard. Two bytes are delivered to the circuit through the register interface into two dedicated registers (register DEN_WSSn).

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.

68.17 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 230 shows an example of a VPS waveform.

Figure 230: Example of VPS waveform



68.18 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 (ST20) 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_TTX_CONF_LINES.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

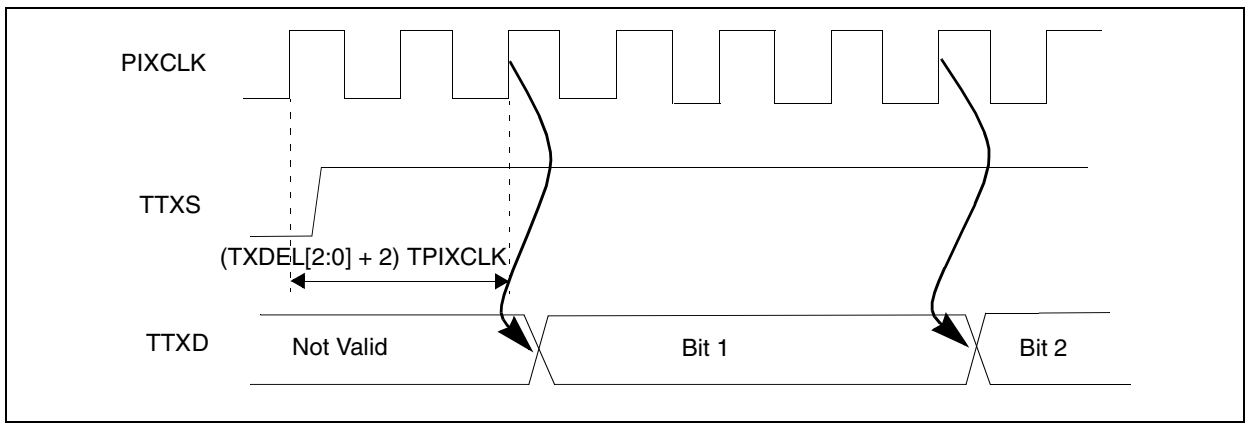
In order to transmit the Teletext data bits at an average rate of 6.9375 Mbit/s, which is about 1/3.89 times the master clock frequency, the following scheme is adopted:

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 231 depicts this graphically for $N = 0$.

Figure 231: TTXS rising to first valid sample delay for TXDEL[2:0] = 0



N is programmable from 0 to 7 by register bits DENC_TTX_CONF_LINES.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_TTX_CONF_LINES). 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 232. Its relative power spectral density is shown in Figure 233 and Figure 234 It is zero at frequencies above 5 MHz, as required by the World System Teletext specification.

Figure 232: Shape and amplitude of a single teletext symbol

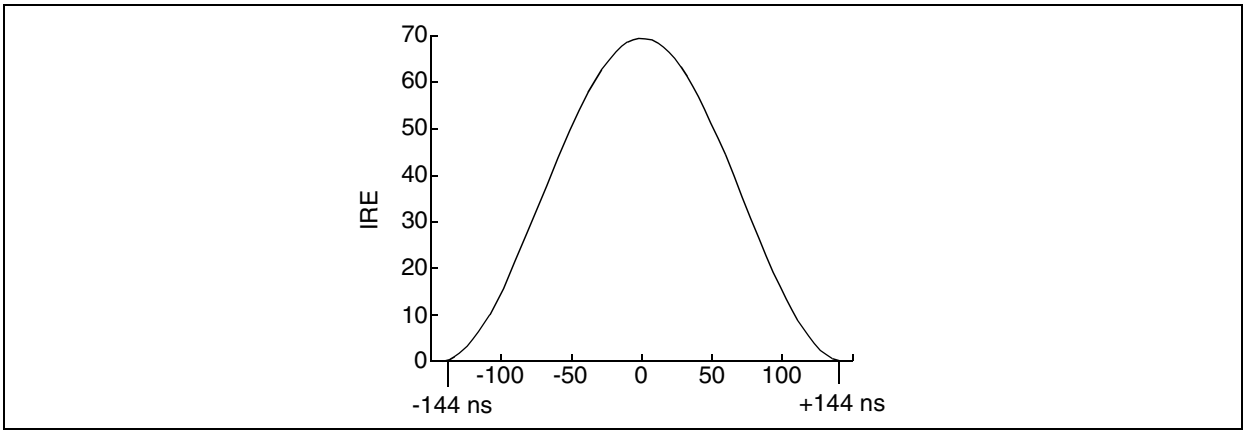


Figure 233: Linear PSD scale

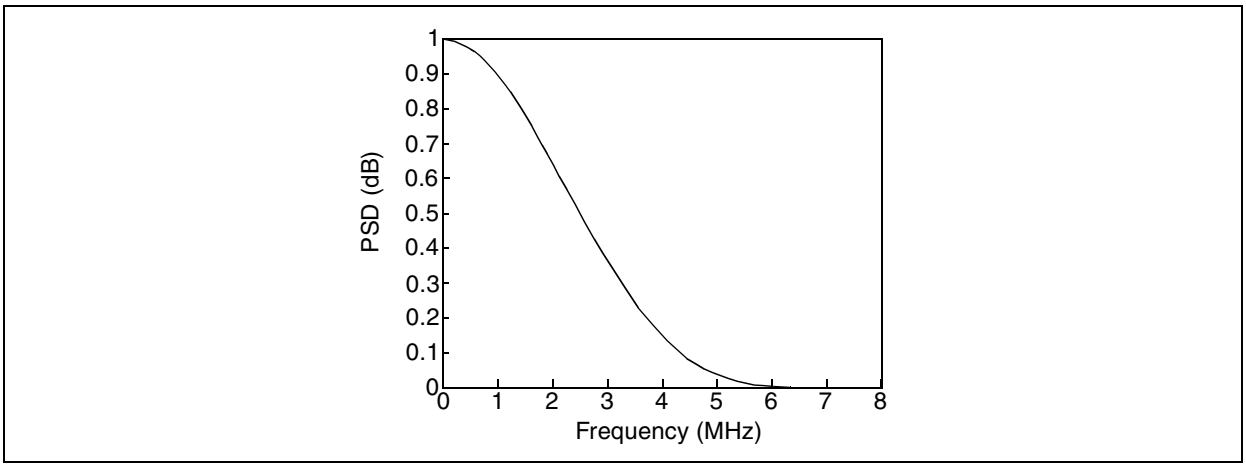
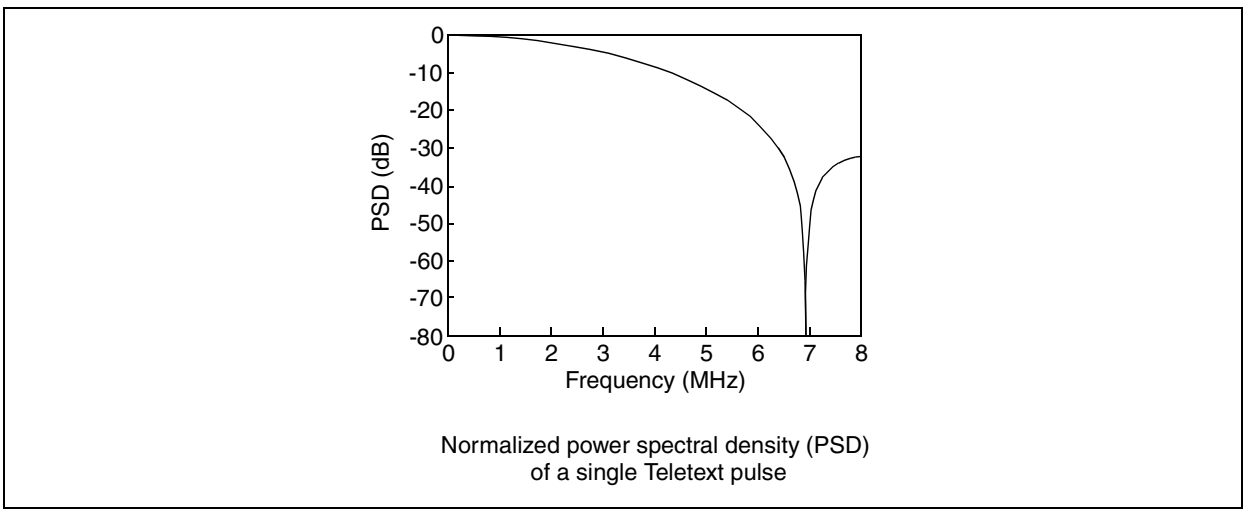


Figure 234: Logarithmic PSD scale



Normalized power spectral density (PSD)
of a single Teletext pulse

68.19 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_CONF[2:0] and DAC456_CONF[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 (Vpp/Upp = 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 Vpp. 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.

69 Digital encoder (DENC) registers

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.

Addresses are provided as *DencBaseAddress* + offset.

The *DencBaseAddress* is:

0x3821 4000.

Each register is mapped into the least significant byte of a 32-bit register space.

Table 193 summarizes the digital encoder registers by function.

Table 193: 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_STATUS	0x024	RO
Digital frequency synthesizer	DENC_DFS_INC2	0x028	R/W
	DENC_DFS_INC1	0x02C	R/W
	DENC_DFS_INC0	0x030	R/W
	DENC_DFS_PHASE1	0x034	R/W
	DENC_DFS_PHASE0	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
	DENC_LINE0	0x050	R/W
	DENC_LINE1	0x054	R/W
	DENC_LINE2	0x058	R/W
Chip ID	DENC_CHIPID	0x060	RO
VPS data	DENC_VPS5	0x064	R/W
	DENC_VPS4	0x068	R/W
	DENC_VPS3	0x06C	R/W
	DENC_VPS2	0x070	R/W
	DENC_VPS1	0x074	R/W
	DENC_VPS0	0x078	R/W

Table 193: 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_TTX_CONF_LINES	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_CONF	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_CCOEF0	0x120	R/W
	DENC_CCOEF1	0x124	R/W
	DENC_CCOEF2	0x128	R/W
	DENC_CCOEF3	0x12C	R/W
	DENC_CCOEF4	0x130	R/W
	DENC_CCOEF5	0x134	R/W
	DENC_CCOEF6	0x138	R/W
	DENC_CCOEF7	0x13C	R/W
	DENC_CCOEF8	0x140	R/W
Luma coefficient	DENC_LCOEF0	0x148	R/W
	DENC_LCOEF1	0x14C	R/W
	DENC_LCOEF2	0x150	R/W
	DENC_LCOEF3	0x154	R/W
	DENC_LCOEF4	0x158	R/W
	DENC_LCOEF5	0x15C	R/W
	DENC_LCOEF6	0x160	R/W
	DENC_LCOEF7	0x164	R/W
	DENC_LCOEF8	0x168	R/W
	DENC_LCOEF9	0x16C	R/W
Hue control	DENC_HUE	0x1A4	R/W

69.1 Configuration and status

DENC_CFG0 Configuration 0

7	6	5	4	3	2	1	0
STD		SYNC			POLH	POLV	FREERUN

Address: *DencBaseAddress* + 0x00

Type: R/W

Reset: 0x92

Description:

[7:6] **STD**[1:0]: 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**[2:0]: 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: reserved
100: reserved
101: reserved
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* + 0x01
Type: R/W
Reset: 0x44
Description:

[31:8] **Reserved**

[7] **BLKLI** [a]: vertical blanking interval selection for active video lines area.
0: [b] (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: [b] f_{-3dB} = 1.6 MHz high definition NTSC filter (ATSC compliant) and PAL M/N (ITU-R 624.4 compliant)
11: f_{-3dB} = 1.9 MHz high definition 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: [b] 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: [b] 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: [b] Blanking level and black level are identical on all lines (ex: 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: [b] 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)

b. Default mode when register bit DENC_CFG.ON is low

DENC_CFG2 Configuration 2

7	6	5	4	3	2	1	0
NINTRL	ENRST	BURSTEN	SET4:4:4	SELRST	RSTOSC_BUF	VALRST	

Address: *DencBaseAddress* + 0x02
Type: R/W
Reset: 0x20
Description:

[31:8] **Reserved**

[7] **NINTRL** [a]: non-interlaced mode select
0:[d] interlaced mode (625/50 or 525/60 system)
1: non-interlaced mode (2 x 312/50 or 2 x 262/60 system)

[6] **ENRST**: cyclic update of DDFS phase
0:[d] no cyclic subcarrier phase reset
1: cyclic subcarrier phase reset depends on VALRST

[5] **BURSTEN**: chrominance burst control
0: burst is turned off on CVBS, chrominance output is not affected
1:[d] burst is enabled

[4] **SET4:4:4**: selects the 4:2:2 or 4:4:4 video input from the mixing unit
0:[d] YCbCr (4:2:2) input: the presence of OSD on the DENC output can be selected or deselected by OSD block header field S.
1: YCbCr (4:4:4) input: in this mode, OSD is always present on the DENC output.
This bit is not used on the STi7710 and should be left at its default reset value.

[3] **SELRST**: selects reset values for direct digital frequency synthesizer
0:[d] 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** [b]: software phase reset of DDFS (direct digital frequency synthesizer) buffer
0:[d] 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 softreset occurs.
RSTOSC_BUF is set back to 0 after the buffer is loaded.

[1:0] **VALRST** [c]
00:[d] Automatic reset of the oscillator every line
01: Automatic reset of the oscillator every 2nd field
10: Automatic reset of the oscillator every 4th field
11:Automatic reset of the oscillator 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. VALRST is taken into account only if bit ENRST 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 SELRST), 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.
d. Default mode when register bit DENC_CFG.ON is low

DENC_CFG3 Configuration 3

7	6	5	4	3	2	1	0
MAIN_ENTRAP	TRAP_4.43	ENCGMS	Reserved	MAIN_DEL_EN	Reserved		ENWSS

Address: *DencBaseAddress* + 0x03

Type: R/W

Reset: 0x00

Description:

[31:8] **Reserved**

[7] **MAIN_ENTRAP** [a]: enable trap filter
0:[b] trap filter disabled 1: trap filter enabled

[6] **TRAP_4.43**: enable trap filter: TRAP_4.43 is taken into account only for main if bit MAIN_ENTRAP is set. No independant selection can be made on main and auxiliary paths.
0:[b] select the trap filter centered around 3.58 MHz 1: select the trap filter centered around 4.43 MHz
See Section 68.10: *Luminance encoding on page 758*

[5] **ENCGMS** [c]: CGMS encoding enable
0:[b] 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] **ENWSS**: WSS encoding enable
0:[b] disabled 1: enabled

a. When SECAM is performed, trap filter must be enabled (set ENTRAP = 1).
b. Default mode when register bit DENC_CFG.ON is low
c. When ENCGMS 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		SYNCOUT_AD		ALINE	DEL4		

Address: *DencBaseAddress* + 0x04
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:[a] nominal 01: +1 PIXCLK
10: +2 PIXCLK 11: +3 PIXCLK

[5:4] **SYNCOUT_AD**: adjustment of outgoing sync signals. Used to ensure correct interpretation of incoming video samples such as Y, Cr or Cb when the encoder is master and supplies sync signals
00:[a] nominal 01: +1 PIXCLK
10: +2 PIXCLK 11: +3 PIXCLK
These bits are not used on the STi7710 and should be left at their default reset values.

[3] **ALINE**: video active line duration control
0:[a] 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

a. Default mode when register bit DENC_CFG.ON is low

DENC_CFG5 Configuration 5

7	6	5	4	3	2	1	0
SELRST_INC	BKDAC1	BKDAC2	BKDAC3	BKDAC4	BKDAC5	BKDAC6	DACINV

Address: *DencBaseAddress* + 0x05
Type: R/W
Reset: 0x00
Description:

[31:8] **Reserved**

[7] **SELRST_INC**: 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**: inverts 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
SOFTRESET	JUMP	DEC_NINC	FREE_JUMP	CFC1	CFC2	TTX_EN	MAXDYN

Address: *DencBaseAddress* + 0x06
Type: R/W
Reset: 0x10
Description:

[31:8] **Reserved**

[7] **SOFTRESET** [a]: software reset
0:[b] no reset 1: software reset

[6:4] **JUMP**, **DEC_NINC**, **FREE_JUMP**: update mode
000: normal mode (no line skip/insert capability)
ITU-R (CCIR): 313/312 or 263/262 non-interlaced: 312/312 or 262/262
0x1: [b] manual mode for line insert (DEC_NINC = 0) or skip (DEC_NINC = 1) capability. Both fields of all the frames following the writing of this value are modified according to bits LREF and LTAR of registers DENC_LINE0/1/2 (by default, LREF = 0 and LTAR = 1 giving the normal mode above).
100: automatic line insert mode. The 2nd field of the frame following the writing of this value is increased. Lines are inserted after line 245 in 525/60 and after line 290 in 625/50. LREF and LTAR are ignored.
110: automatic line skip mode. The 2nd field of the frame following the writing of this value is decreased. Lines are suppressed after line 245 in 525/60 and after line 290 in 625/50. LREF and LTAR are ignored.
1x0: do not use.
These bits are not used on the STi7710 and should be left at their default reset values.

[3:2] **CFC1**, **CFC2**: color frequency control via CFC line
00:[b] update mode disabled (update is carried out by loading registers DENC_DFS_INC0/1/2)
01: update of increment for DDFS just after serial loading via CFC
10: update of increment for DDFS on next active edge of Hsync
11: update of increment for DDFS just before next color burst
These bits are not used on the STi7710 and should be left at their default reset values.

[1] **TTX_EN**: Teletext enable bit
0:[b] disabled 1: enabled

[0] **MAXDYN**: max dynamic magnitude allowed on YCbCr inputs for encoding
0:[b] 0x10 to 0xEB for Y, 0x10 to 0xE0 for chrominance (Cr, Cb)
1: 0x01 to 0xFE for Y, Cr and Cb

a. Bit SOFTRESET is automatically reset after internal reset generation. Software reset is active for 4 PIXCLK periods. When softreset is activated, the DENC is reset as with hardware reset except for the first nine user registers (DENC_CFG0 to DENC_CFG8).
b. Default mode when register bit DENC_CFG.ON is low

DENC_CFG7 Configuration 7

7	6	5	4	3	2	1	0
SECAM	GEN_SECAM	INV_PHI_ SECAM	Reserved			ENVPS	SQPIX

Address: *DencBaseAddress* + 0x07

Type: R/W

Reset: 0x00

Description:

[31:8] **Reserved**

[7:6] **GEN_SECAM**, **SECAM**: where the bit order is SECAM - GEN_SECAM.

00[a]: Standard selected by bits STD1 and STD0 of DENC_CFG0 (PAL or NTSC)

01: Reserved

10: SECAM standard, the subcarrier phase sequence start-point is line1

11: SECAM standard, the subcarrier phase sequence start-point is line23

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:[a] 0, 0, π,... in odd fields and π, π, 0 in even fields 1: π, π, 0 in odd fields and 0, 0, π,... in even fields

[4:2] **Reserved**

[1] **ENVPS**: VPS encoding enable

0:[a] disable 1: enable

[0] **SQPIX**: square pixel mode enable.

0:[a] disable 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* + 0x08
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 68.7: *Subcarrier generation* *on page 756*.

00:[a] disabled
01: enabled - phase is updated with value from phase buffer register (see DENC_CFG2 bit 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:[a] disabled
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
WARNING: 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* + 0x51

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 3 bits to control the luma filter. The following table describes the register's functions.

[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. Bit DEL_EN in register DENC_CFG3 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* + 0x5C
Type: R/W
Buffer: Immediate
Reset: 0x48

Description:

[7:4] **Reserved**

[3] **RGBSAT_EN**: RGB format output not saturated to real colours(241 max and 15 min), but allowed to have more saturated colours(255 max and 0 min).
1: RGB outputs saturated to real colors.
0: RGB outputs allowed to have more saturated colors.

[2] **SECAM_IN_MUX**: SECAM input video select. (Refer to section Section 68.10: *Luminance encoding on page 758*) 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* + 0x5D
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 5 clock cycles in rectangular pixel mode (27 MHz clock) and 4 clock cycles in NTSC square pixel mode (24.5454 MHz clock). Please refer to Section 68.12: *Composite video signal generation on page 762*. Set MAIN_IF_DEL to 1 to enable.

DENC_CFG12 Configuration 12

7	6	5	4	3	2	1	0
Reserved					ENNOTCH	Reserved	

Address: *DencBaseAddress* + 0x5E
Type: R/W
Buffer: Immediate
Reset: 0x00
Description: Notch filtering on the main 4:4:4 luma input (Please refer to Section 68.10: *Luminance encoding on page 758*). Set ENNOTCH to 1 to enable.

DENC_CFG13 Configuration 13

7	6	5	4	3	2	1	0
AUX_NOTMAIN_RGB	RGB_MAXDYN	DAC123_CONF				DAC456_CONF	

Address: *DencBaseAddress* + 0x5F

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 CVBS output chroma bandwidth should always be narrow to reduce cross luma effects.

[7] **AUX_NOTMAINRGB**: select either main video or aux video as RGB.
0: main video is output as RGB.
1: aux video is output as RGB. (default)
Note: No Auxilliary 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 better uses the maximum dynamic range of the video DACs. Blanking level is 64.

[5:3] **DAC123_CONF**

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

Note: No Auxilliary video input support

[2:0] **DAC456_CONF**

000: DAC1 = Y_AUX,	DAC2 = C_AUX,	DAC3 = CVBS_AUX
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
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: No Auxilliary video input support

DENC_STATUS Status

7	6	5	4	3	2	1	0
HOK	ATFR	BUF2_FREE	BUF1_FREE	FIELDCT			JUMP

Address: *DencBaseAddress* + 0x09
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 68.14: *Closed captioning on page 764*)
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 field2 registers access might be temporarily forbidden by resetting bit BUF2_FREE until the next field2 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] **JUMP**
indicates whether a frame length modification has been programmed at '1' from programming of bit JUMP to end of frame(s) concerned
default = 0
Refer to register 6 and registers 21-22-23

69.2 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
0x0C	D							
DFS_INC1	15	14	13	12	11	10	9	8
0x0B	D							
DFS_INC2	7	6	5	4	3	2	1	0
0x0A	D							

Address: *DencBaseAddress* + register offset

Type: R/W

Reset: Undefined

Description: These three registers contain the 24-bit increment used by the DDFS, **only if** bit SELRST_INC (register DENC_CFG5) 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* + 0x0D and 0x0E

Type: R/W

Reset: HUE_EN: Disabled (0), DFS_PHASE: Undefined; never reset and must be explicitly written to.

Description: If bit SELRST = 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
0x0E	Reserved			HUE_EN	Reserved		VALUE	
DFS_PHASE1	o21	o20	o19	o18	o17	o16	o15	o14
0x0D	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’.

1: Enabled

0: Disabled (Default)

DFS_PHASE 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 bits VALRST of DENC_CFG2). 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 SELRST to 1 (DENC_CFG2)
- 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 register DENC_CFG8 bit 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
0x0D	VALUE							
DFS_PHASE1	r21	r20	r19	r18	r17	r16	r15	r14
0x0E	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.

69.3 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* + 0x00F (DEN_WSS1), 0x010 (DEN_WSS2)

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.

69.4 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* + 0x11
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* + 0x12
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* + 0x13
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_CONF = 010)	if not R, G or B (DAC123_CONF != 010)
000000:	81.25	75.00
000001:	81.84	75.78
000010:	82.42	76.56
000011:	83.01	77.34
.. ..		
100000: (default)	100.00	100.00
.. ..		
111111:	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_CONF = 010)	if not R, G or B (DAC123_CONF != 010)
000000:	81.25	75.00
000001:	81.84	75.78
000010:	82.42	76.56
000011:	83.01	77.34
.. ..		
100000: (default)	100.00	100.00
.. ..		
111111:	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* + 0x6A
Type: R/W
Reset: 0x20
Description:

[7:6] **Reserved**

[5:0] **DAC2_MULT[5:0]**

DENC_DAC6_MULT DAC6 multiplying factors

7	6	5	4	3	2	1	0
Reserved		DAC6_MULT[5:0]					

Address: *DencBaseAddress* + 0x6B
Type: R/W
Reset: 0x20
Description:

[7:6] **Reserved**

[5:0] **DAC6_MULT[5:0]**

DENC_LINE0/1/2 Target and reference lines, bytes 0, 1 and 2

	7	6	5	4	3	2	1	0
LINE0								
0x14	LTARG8	LTARG7	LTARG6	LTARG5	LTARG4	LTARG3	LTARG2	LTARG1
LINE1								
0x15	LTARG0	LREF8	LREF7	LREF6	LREF5	LREF4	LREF3	LREF2
LINE2								
0x16	LREF1	LREF0	Reserved					

Address: *DencBaseAddress* + register offset
Type: R/W
Reset: Undefined
Description: These registers are not used on the STi7710 and should be left at their default reset values.

These registers can be used to jump from a reference line (end of that line) to a target line of the same field. However, not all lines can be skipped or repeated without problems. *This functionality should be used with caution*.

69.5 Chip ID

DENC_CHIPID DENC version identification number

7	6	5	4	3	2	1	0
CHIPID							

Address: *DencBaseAddress* + 0x18
Type: RO
Reset: 0xC0
Description:

[31:8] **Reserved**

[7:0] **CHIPID**: 1100 0000

69.6 VPS data

DENC_VPSn VPSn data registers

	7	6	5	4	3	2	1	0
DENC_VPS5: 0x19	S1	S0	Reserved		CNI3	CNI2	CNI1	CNI0
DENC_VPS4: 0x1A	NP7	NP6	D4	D3	D2	D1	D0	M3
DENC_VPS3: 0x1B	M2	M1	M0	H4	H3	H2	H1	H0
DENC_VPS2: 0x1C	MIN4	MIN3	MIN3	MIN2	MIN1	MIN0	C3	C2
DENC_VPS1: 0x1D	C1	C0	NP5	NP4	NP3	NP2	NP1	NP0
DENC_VPS0: 0x1E	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0

Address: *DencBaseAddress* + register offset

Type: Read only

Reset: Undefined

Description: Refer to Section 68.17: *VPS encoding on page 766*.

DENC_VPS5: [7:6] **S[1:0]:** sound
00: don't know 01: mono
10: stereo 11: dual sound

[5:4] **Reserved**

DENC_VPS5: [3:0] **CNI[3:0]**: 4 bits of CNI Reserved for enhancement of VPS

DENC_VPS4: [7:6] **NP[7:6]**: Network or program CNI (see reg68(5:0) for a beginning)

DENC_VPS4: [5:1] **D[4:0]**: Day, binary

DENC_VPS4: [0] DENC_VPS3: [7:5] **M[3:0]:** Month, binary

DENC_VPS3: [4:0] **H[4:0]**: Hour, binary

DENC_VPS2: [7:2] **MIN[5:0]**: Minute, binary

DENC_VPS2: [1:0] DENC_VPS1: [7:6] **C[3:0]:** Country, binary

DENC_VPS1: [5:0] **NP[5:0]**: Network or program CNI (Country and Network Identification)

DENC_VPS0: [7:0] **PT[7:0]**: Program type, binary

69.7 CGMS data

DENC_CGMS0/1/2 CGMS data registers (20 bits only)

CGMS0	7	6	5	4	3	2	1	0
0x1F	Reserved				B1	B2	B3	B4
CGMS1	7	6	5	4	3	2	1	0
0x20	B5	B6	B7	B8	B9	B10	B11	B12
CGMS2	7	6	5	4	3	2	1	0
0x21	B13	B14	B15	B16	B17	B18	B19	B20

Address: *DencBaseAddress* + register offset
Type: R/W
Reset: Undefined
Description:

B1 to **B3**: Word0A

B4 to **B6**: Word0B

B7 to **B10**: Word1

B11 to **B14**: Word2

B15 to **B20**: CRC (not internally computed)

69.8 Teletext

DENC_TTX_CONF_LINES Teletext configuration

7	6	5	4	3	2	1	0
FP_TTXT	TTXDEL2	TTXDEL1	TTXDEL0	TTX_L6	TTX_L7	TTZ_L8	TTX_L9

Address: *DencBaseAddress* + 0x22
Type: R/W
Reset: 0x50
Description:

[7] **FP_TTXT**: full page teletext mode enable (refer to Section 68.18: *Teletext encoding on page 766*)
(0: disabled
1: enabled

[6:4] **TTXDEL[2:0]**: Teletext data latency (* “100” default) (Refer to Section 68.18: *Teletext encoding on page 766*)

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 194: *Teletext line selection for standards other than ITU-R and 625 line systems on page 793*

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* + 0x23, 0x24, 0x25, 0x26

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 194: 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 194: 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_CONF Teletext configuration

7	6	5	4	3	2	1	0
TTXT100IRE	TTXT_ABCD1	TTXT_ABCD0	Reserved				

Address: *DencBaseAddress* + 0x40
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[3:0]				TTX_MASK_O	Reserved		BCS_EN_MAIN

Address: *DencBaseAddress* + 0x41

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[a]
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 Vpp for CVBS on DAC3 for example (and DAC3_MULT = “1000”), when switched to U, DAC6 delivers 0.7 Vpp. 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 Vpp for CVBS on DAC3 for example (and DAC3_MULT = “1000”), when switched to B, DAC6 delivers 0.7 Vpp.

[3] **TTX_MASK_OFF**: masking of 2 bits in Teletext framing code, depending on selected teletext standard

0:[a] enable 1: disable

[2:1] **Reserved**

[0] **BCS_EN_MAIN**: brightness, contrast and saturation control by registers DENC_BRIGHT, DENC_CONTRAST and DENC_SATURATION on 4:4:4 main video input

0: disable 1:[a] enable

a. Default mode when register bit DENC_CFG.ON is low

69.9 Closed caption

DENC_CCCF1 Closed-caption characters/extended data for field 1

CCCF10	7	6	5	4	3	2	1	0
0x27	OPC11	C117	C116	C115	C114	C113	C112	C111
CCCF11	7	6	5	4	3	2	1	0
0x28	OPC12	C127	C126	C125	C124	C123	C122	C121

Address: *DencBaseAddress* + register offset

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 to **C111**: first byte to encode in field 1

OPC12: odd-parity bit of US-ASCII 7-bit character C127 to C121

C127 to **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
0x29	OPC21	C217	C216	C215	C214	C213	C212	C211
CCCF21	7	6	5	4	3	2	1	0
0x2A	OPC22	C227	C226	C225	C224	C223	C222	C221

Address: *DencBaseAddress* + register offset

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** to **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** to **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* + 0x2B

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* + 0x2C

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)

69.10 Input demultiplexor

DENC_BRIGHT Brightness

7	6	5	4	3	2	1	0
B							

Address: *DencBaseAddress* + 0x45

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 Yin is 8-bit input luminance and Yout 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* + 0x46

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, Yin is 8-bit input luminance, Yout 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* + 0x47

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 Crin and Cbin are the 8-bit input chroma, Crout and Cbout 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).

69.11 Chroma coefficient

DENC_CFCOEF0..8 Chroma filter coefficient 0 to 8 (main video)

	7	6	5	4	3	2	1	0
CCOEF0: 0x48	MAIN_FLT_S	MAIN_PLG_DIV[1:0]		CH_COEF0[4:0]				
CCOEF1: 0x49	Reserved	CH_COEF8[8]	CH_COEF1[5:0]					
CCOEF2: 0x4A	Reserved	CH_COEF2[6:0]						
CCOEF3: 0x4B	Reserved	CH_COEF3[6:0]						
CCOEF4: 0x4C	CH_COEF4[7:0]							
CCOEF5: 0x4D	CH_COEF5[7:0]							
CCOEF6: 0x4E	CH_COEF6[7:0]							
CCOEF7: 0x4F	CH_COEF7[7:0]							
CCOEF8: 0x50	CH_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 table 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 what the sum of all the coefficients is.
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_CFCOEF0..8 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] = 10001 means c0 = 1,
- CHROMA_MAIN_COEF1[5:0] = 100111 means c1 = 7,
- CHROMA_MAIN_COEF2[6:0] = 1010100 means c2= 20,
- CHROMA_MAIN_COEF3[6:0] = 1000111 means c3 = 39,
- CHROMA_MAIN_COEF4[7:0] = 01011111 means c4 = 63,
- CHROMA_MAIN_COEF5[7:0] = 01110111 means c5 = 87,
- CHROMA_MAIN_COEF6[7:0] = 01101100 means c6 = 108,
- CHROMA_MAIN_COEF7[7:0] = 01111011 means c7 = 123,
- CHROMA_MAIN_COEF8[8:0] = 010000000 means c8 = 128.

69.12 Luma coefficient

DENC_LCOEF0..9 Luma filter coefficients 0 to 9

	7	6	5	4	3	2	1	0
LCOEF0: 0x52	Reserved		LU_COEF8[8]	LU_COEF0[4:0]				
LCOEF1: 0x53	LU_COEF9[9:8]		LU_COEF1[5:0]					
LCOEF2: 0x54	LU_COEF6[8]	LU_COEF2[6:0]						
LCOEF3: 0x55	LU_COEF7[8]	LU_COEF3[6:0]						
LCOEF4: 0x56	LU_COEF4[7:0]							
LCOEF5: 0x57	LU_COEF5[7:0]							
LCOEF6: 0x58	LU_COEF6[7:0]							
LCOEF7: 0x59	LU_COEF7[7:0]							
LCOEF8: 0x5A	LU_COEF8[7:0]							
LCOEF8: 0x5B	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] = 00001 means +1,

a_1 = LUMA_COEF_1[5:0] = 111111means -1,

a_2 = LUMA_COEF_2[6:0] = 1110111means -9,

a_3 = LUMA_COEF_3[6:0] = 0000011means +3,

a_4 = LUMA_COEF_4[7:0] = 00011111means +31,

a_5 = LUMA_COEF_5[7:0] = 11111011means -5,

a_6 = LUMA_COEF_6[8:0] = 110101100means -84,

a_7 = LUMA_COEF_7[8:0] = 000000111means +7,

a_8 = LUMA_COEF_8[8:0] = 100111101means +317,

a_9 = LUMA_COEF_9[9:0] = 0111111000means +504.

69.13 Hue control

DENC_HUE Hue control

7	6	5	4	3	2	1	0
HUE_PHS_SGN	HUE_PHS_VAL						

Address: *DencBaseAddress* + 0x69

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_CONTROL*, 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
10000000: No phase shift	00000000: No phase shift
11111111: +22.324 degrees phase	01111111: -22.324 degrees phase

[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

70 Teletext DMA

70.1 Teletext DMA overview

The STi7710 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).

70.2 Teletext packet format

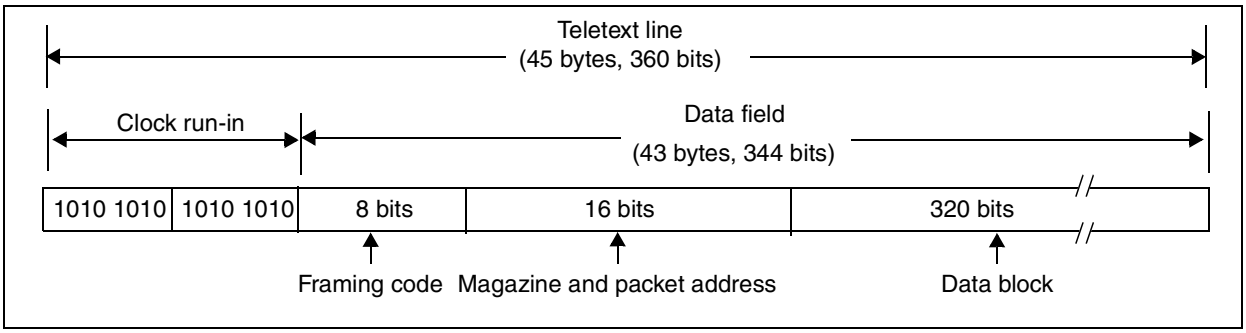
One teletext packet (also called a teletext line) is a stream of 360 bits, transferred at an average frequency of 6.9375 MHz. The data format is the same as the contents of the PES data packet as defined in the ETSI specification. The DMA reads in multiples of 46 bytes and transfers lines of 45 bytes to the digital encoder.

Each teletext packet is composed of the clock run-in and the data-field, as shown in Figure 235.

- 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 235: Teletext packet format



70.3 Data transfer sequence

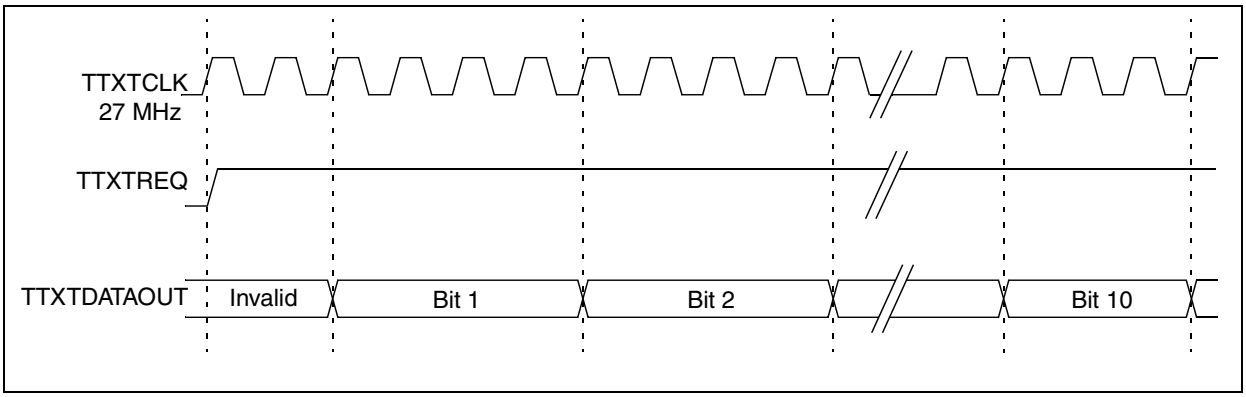
The digital encoder issues a teletext request signal to the teletext DMA, this is shown by the rising edge of signal TTXTREQ (PIO6[6]) in Figure 236. After a delay, programmable from 2 to 9 master-clock periods, the teletext DMA transmits the first valid teletext data bit of the teletext packet.

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,

1. Specification for conveying ITU-R Systems B teletext in digital video broadcasting (DVB) bitstreams.

19, 28 and 37, which are transmitted in three 27 MHz cycles. This is illustrated in Figure 236 for bits 0 to 10.

Figure 236: 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 (see Transmission Protocol below).

The delay between signal TTXTREQ becoming high and the transfer of the first bit of the teletext packet is between 2 and 9, 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.

70.4 Interrupt control

Teletext interrupts can be programmed by register TTXT_INTEN 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_INTSTA and masked by TTXT_INTEN. The interrupt bits are reset when the CPU writes to the acknowledge register, or when a DMA operation is completed.

70.5 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, 2 blocks in each VBI, or 3 blocks in field 1 VBI and one in field 2 VBI). In certain circumstances, up to 4 areas may be defined in each VBI.

The five dedicated digital encoder registers are DEN_TTX1 to DEN_TTX5.

70.6 Teletext external interface

Table 195: Teletext interface pins

Pin	Type	Function name	Function description
PIO6[6]	Input	TTXTREQ	Teletext request
PIO6[7]	Output	TTXTDATAOUT	Teletext output data

71 Teletext DMA registers

Register addresses are provided as *TTXTBaseAddress* + offset.

The TTXTBaseAddress is:

0x0200 6000.

Table 196: Teletext register summary

Register	Function	Offset	Type
TTXT_DMAADD	Teletext DMA address	0x00	R/W
TTXT_DMACNT	Teletext DMA count	0x04	R/W
TTXT_OUTDELAY	Teletext output delay	0x08	R/W
TTXT_INTSTA	Teletext interrupt status	0x18	RO
TTXT_INTEN	Teletext interrupt enable	0x1C	R/W
TTXT_ACKODDEVEN	Teletext acknowledge odd or even	0x20	WO
TTXT_ABORT	Teletext abort	0x24	WO

TTXT_DMAADD 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
DMAADDRESS																															

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_DMACNT 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																					DMACOUNT										

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 bytes x *number_of_teletext_lines_to_send*
- For teletext input operation, the value must be:
  42 bytes x *number_of_teletext_lines_to_receive*

TTXT_OUTDELAY Teletext output 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																							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_INTSTA Teletext interrupt status

7	6	5	4	3	2	1	0
Reserved					EVEN	ODD	INOUT COMPLETE

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_INTEN 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] **INOUTCOMPLETE**: Teletext input or output operation completed.

TTXT_INTEN Teletext Interrupt enable

7	6	5	4	3	2	1	0
Reserved					EVENENABLE	ODDENABLE	INOUT COMPLETEEN

Address: *TTXTBaseAddress* + 0x1C
Type: R/W
Reset: Undefined
Description: This register allows masking of register TTXT_INTSTA.

[31:3] **Reserved**

[2] **EVENENABLE**: enable even field interrupt.

[1] **ODDENABLE**: enable odd field interrupt.

[0] **INOUTCOMPLETEEN**: enable teletext input or output operation completed interrupt.

TTXT_ACKODDEVEN 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_INTSTA.

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.

72 SD triple video DACs

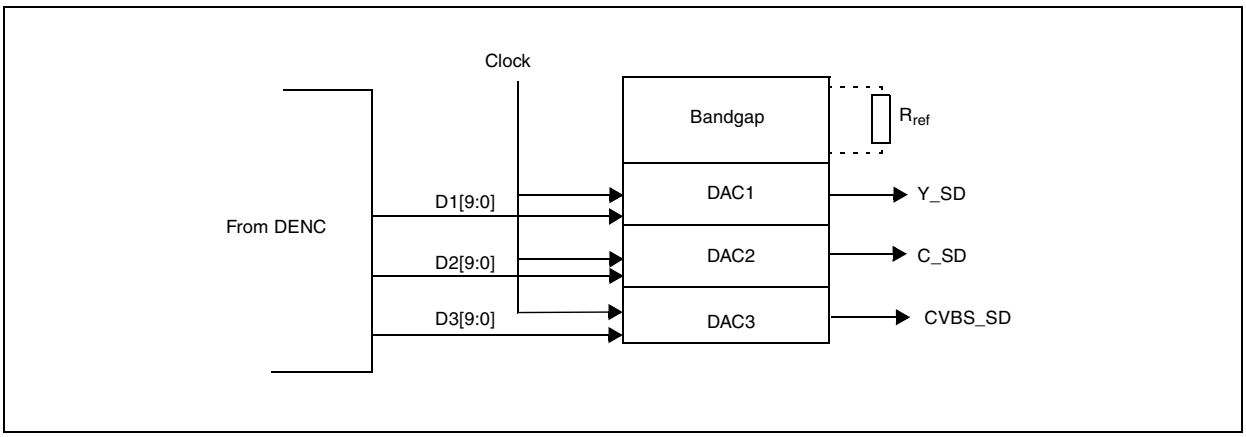
72.1 Description

One on-chip 3 x 10-bit digital-to-analog converter (triple DAC) is used for SD video output. It provides output signals in CVBS, Y, C. Figure 237 below shows the DAC architecture.

An external reference resistor R_{ref} is associated with the bandgap voltage to generate a reference current. This precision resistor, typically 10 KΩ, 1%, is connected between pins SD_REXT and SD_GNDAS_REXT. The associated SD_GNDAS_REXT pin must be connected to the board video analog ground.

Figure 237 shows the global segmented architecture. Current sources provide an output range of 1.4 V with good linearity. Sampled data are available on video outputs after one clock period (on the next rising clock edge).

Figure 237: Triple video DAC schematic



The triple video DAC has its own 3.3 V power and ground supplies for noise reduction. To guarantee good frequency response at high frequencies, these power and ground supplies are not connected to digital power supplies inside the chip. To minimize crosstalk between video outputs, additional analog ground pads have been inserted.

72.2 Input codes for video application

Table 197 lists the reference input codes generated by the digital encoder (DENC) depending on the configuration for the DACs and the standard.

Table 197: Reference input codes

	Y-PAL/SECAM	Y-NTSC	RGB
White (235)	816.00	802.00	602.00
Black (16)	256.00	240.00	41.00
Sync tip	16.00	16.00	-

Note: CVBS = Y + C, so the chrominance component has no effect on the CVBS signal (C is null).

72.3 Video output voltage level

The resistor R_{ref} connected to the bandgap has a direct effect on the output current that flows from the DAC outputs. For the maximum code (1023 in decimal):

I_{out} (max) = 80.539 / R_{ref}

The value of R_{ref} must be carefully chosen: **I_{out} must always be 8 mA ± 17%**, otherwise DAC linearity is not guaranteed. This is achieved with a typical value of R_{ref} = 10 kΩ, giving I_{out} = 8.05 mA for each DAC.

Because of the sensitive relationship between the DAC and R_{ref}, the tolerance on the R_{ref} value must be small. Typically, the R_{ref} resistor must be 1% tolerance.

The output voltage on the RGB output pins depends on the external load resistor R_{load} (connected between the DAC output and ground):

V_{out} (max) = R_{load} x I_{out} (max)

For example, with a typical load value R_{load} = 175 Ω and I_{out} = 8.05 mA, V_{out} (max) = 1.4 V.

Note: V_{out} must always be less than 1.5 V to guarantee linearity.

For any given digital input code, D_{in}, the output voltage of the DAC is given by the following formula:

$V_{out} = D_{in} / 1023 \times V_{out}$ (max) = (D_{in} x R_{load} x 80.539) / (R_{ref} x 1023)

$V_{out} = D_{in} \times R_{load} \times 0.0787 / R_{ref}$

For example, with R_{load} = 175 Ω, R_{ref} = 10 kΩ and D_{in} = 526, V_{out} = 0.72 V.

72.4 Video specifications and DAC setup

In Y-PAL/SECAM, the output video range between code 16 (synchronization level) and code 816 (white level) should be: $V_{out}(816) - V_{out}(16)$ = 1 V. The output video range between code 256 (black level) and code 816 (white level) should be 700 mV. This must be respected for all applications.

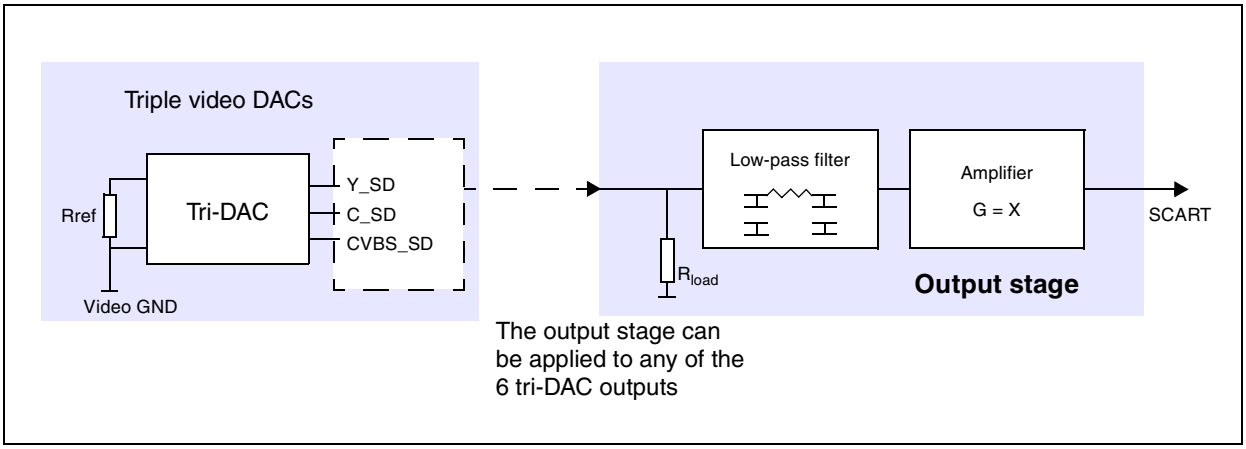
The value of the R_{load} resistor must be chosen according to the value of R_{ref} and the previous formula, to achieve the standard output video range. The nominal value for R_{ref} is 10 kΩ.

According to video specifications ITU-R BT 601, the nominal sampling frequency is 27 MHz. The clock for the DACs is the same as for the DENC, but buffered. This clock is generated by an internal VCXO.

72.5 Output-stage adaptation and amplification

A schematic of the output stage is shown in Figure 238 below. The purpose of the output stage depends on the application and the required price-to-performance ratio. The output stage is connected directly to a Scart connector or to other components (in which case the level and output impedance of the output signal may be different).

Figure 238: Output stage diagram



The amplifier gain must be in accordance with that of the tri-DACs (defined by R_{load} and R_{ref}). If the amplifier gain cannot be set to the standard output video range by R_{ref} and R_{load}, it can be tuned. In common applications (with R_{ref} = 10 kΩ and R_{load} = 175 Ω), a video amplifier should be adequate. The ideal input impedance of the output stage should be greater than R_{load}.

The tri-DACs have no cut-off frequency, therefore, a low-pass filter (around 10 MHz) must be applied to remove harmonics (mainly 27 MHz). If additional attenuation is applied by the filter due to imperfection of the amplifier (generally degrading the C/L ratio), correction must be applied to preserve a good performance. Also, to guarantee good frequency behavior at high frequency, the analog power supply must be separate from the digital power supply. If this is not the case, an additional correction may be required.

73 HD triple video DACs

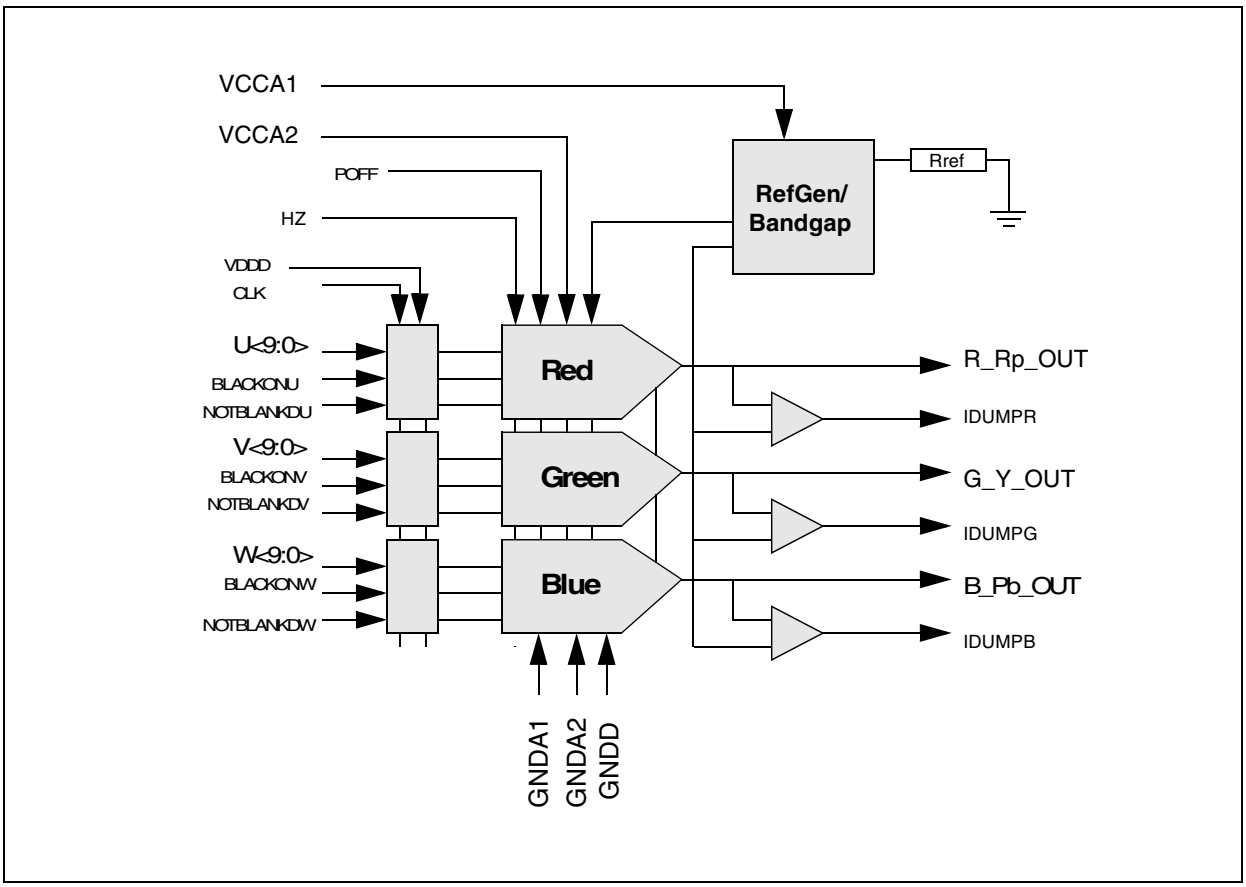
73.1 Description

The high-definition video digital-to-analog converter (DAC) is a triple high performance 10-bit digital to analog converter used to display the main output of the graphics display engine; see Figure 239. It consists of three 10-bit DAC modules joined together: one for each RGB video output. A unique reference circuit controlled by one external resistor sets the full-scale output for all the DACs.

Each module uses a 5/5 segmented architecture to achieve the best performance. The 5 least significant bits control a binary weighted current matrix whereas the 5 most significant bits control a thermometric current matrix. Each DAC is able to output 35 mA current.

The output is high-definition up to 3H, but it can also be used in standard format. The DACs can be fed with three different video formats: full-range RGB, full-range YCbCr or EIA 770.3 with embedded syncs (the pure video signal is reduced by 70% to conform to the EIA 770.3 standard).

Figure 239: Functional block diagram



Current sources have been designed in order to provide an output range of 1.4 V with a good linearity.

The external reference resistor (R_{ref} = 2.41 kΩ, 1%) is associated with a bandgap voltage to generate the voltage reference which is mirrored to provide to the DAC the appropriate current reference.

The macro will have a high immunity to digital noise only if the reference resistor is tied between R_{ext} pin and MASS_QUIET pin.

To Calculate U_{out}:

$U_{out} = [D_{in}] * 0.0625 *[(R_{ext}\text{-MASS\_QUIET})/R_{ref}] * R_{load}$ when blackon is deactivated

$U_{out} = [D_{in}+88] * 0.0625 *[(R_{ext}\text{-MASS\_QUIET})/R_{ref}] * R_{load}$ when blackon is activated

With:

R_{ext} - MASS_QUIET= $V_{bandGap}$ = 1.2197 V

D_{in} = Code value in decimal

R_{ref} = reference resistor

R_{load} = load resistor

73.2 Video output voltage level

When blackon is not asserted,

$V_{out} = D_{in}*.0625*((R_{ext}\text{-}mass\_quiet)/R_{ref})*R_{load}$

The output voltage on the RGB output pins depends on the external load resistor R_{load} (connected between the DAC output and ground):

V_{out} (max) = R_{load} x I_{out} (max)

For example, with a typical load value R_{load} = 37.5 Ω and I_{out} (max) = 30 mA, V_{out} (max) = 1.125 V.

For any given digital input code, the output voltage of the DAC is given by the following formula:

$V_{out} = D_{in} / 1023 \times V_{out}$ (max) = (D_{in} x R_{load} x 1.548) / (R_{ref} x 1023)

$V_{out} = D_{in} \times R_{load} \times 0.00151 / R_{ref}$

For example, with R_{load} = 37.5 Ω, R_{ref} = 2.41 kΩ and D_{in} = 526, V_{out} = 0.165 V

Figure 240: video output levels for a 10-bit DAC with blackon low (“0”)

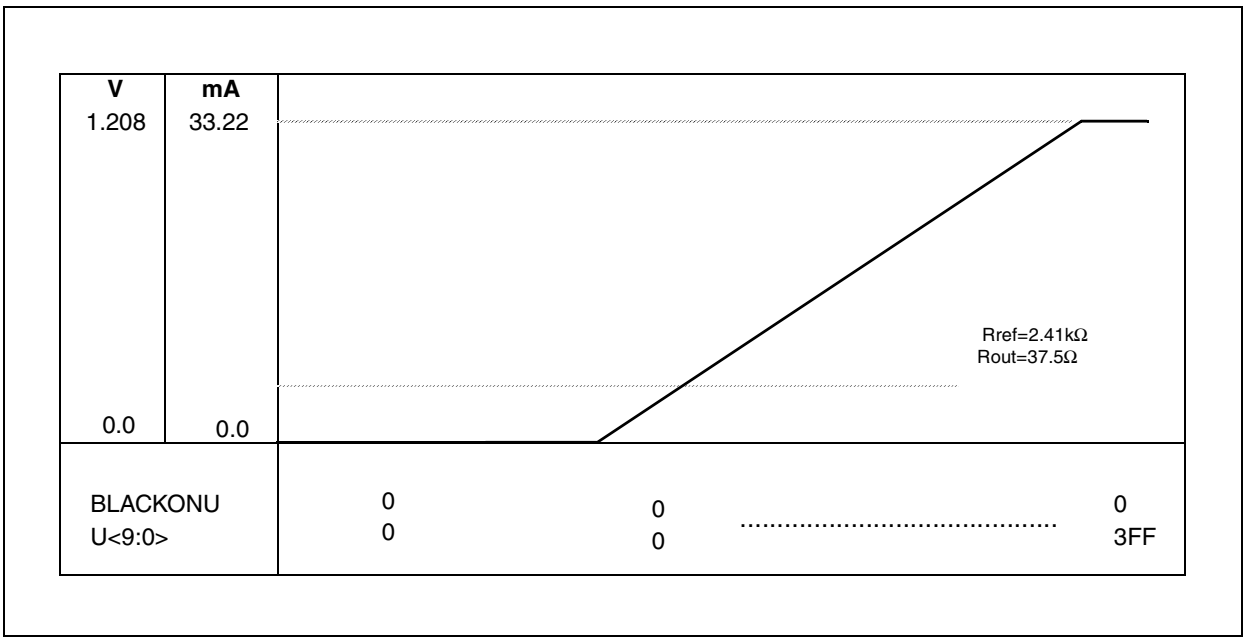
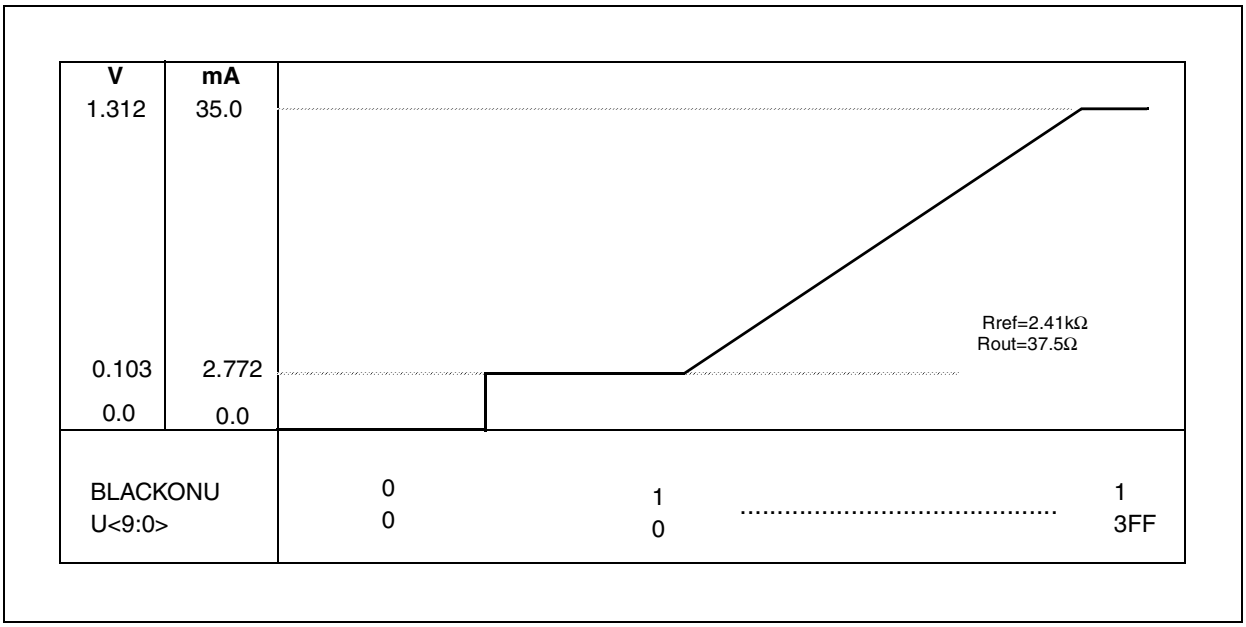


Figure 241: Video output levels for a 10-bit DAC with blackon high (“1”)



73.3 Video specifications and DAC setup

The output video range between code 64 (black level) and code 960 (white level) should be:

V_{out} (960) - V_{out} (64) = 1.128 V.

This gives V_{out}(1023) - V_{out}(0) = 1.29 V.

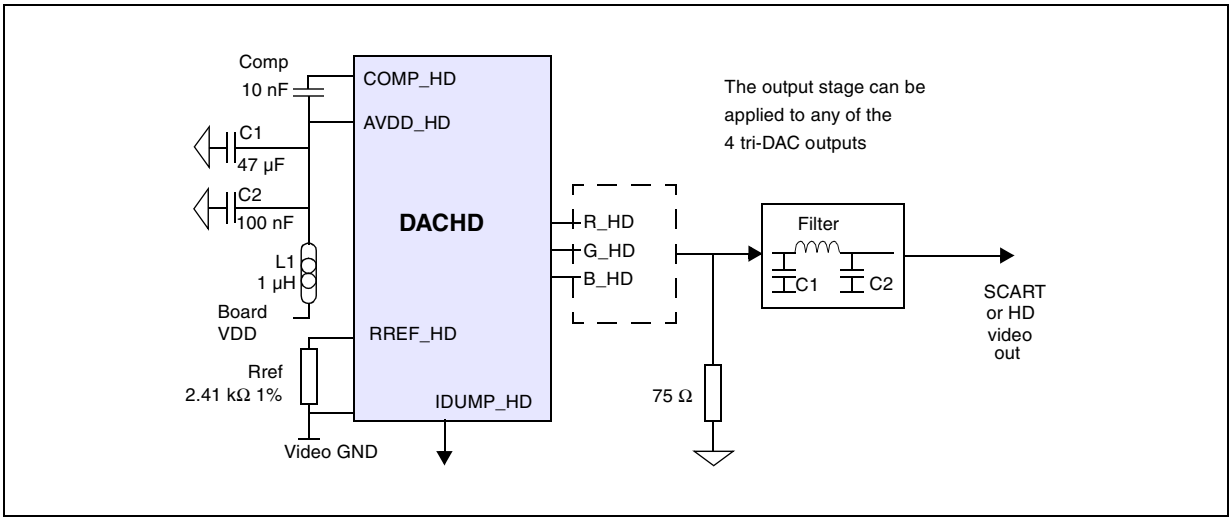
The value of the R_{ref} resistor must be chosen according to the value of R_{load} and the previous formula, to achieve the standard output video range.

According to specifications of ATSC, the nominal sampling frequency is 74.25 MHz. This clock must be as clean as possible to achieve a good signal-to-noise ratio.

73.4 Output-stage adaptation and amplification

A schematic of the output stage is shown below. The purpose of the output stage depends on the application and the required price-to-performance ratio. The output stage is connected directly to a SCART connector or to other components (in which case the level and output impedance of the output signal may be different).

Figure 242: Output stage schematic



The amplifier gain must be in accordance with that of the tri-DACs (defined by R_{load} and R_{ref}). If the gain cannot be set to the standard output video range by R_{ref}, it can be tuned using the amplifier gain. In common applications (with R_{ref} = 2.41 kΩ), a video amplifier gain of 2 should be adequate.

The tri-DACs have no cut-off frequency. A low-pass filter (30 MHz for luma and 15 MHz for chroma for YUV HD output mode) must thus be applied to remove harmonics (mainly 74.25 MHz).

If additional attenuation is applied by the filter due to amplifier imperfection (generally degrading the C/L ratio), a correction must be applied to preserve correct performance.

To guarantee correct frequency behavior at high frequency, the analog and digital power supplies must be separate. If this is not the case, an additional correction may be required.

73.5 Electrical specifications

Table 198: DAC specifications

Symbol	Parameter	Min	Typ	Max	Units
VDD	DC supply voltage	2.97	3.3	3.63	V
Tj	Operating temperature (junction)	0		125	C
PD	Power dissipation		150	500	mW
PDpd	Power dissipation during powerdown mode		0.5	7.2	µW
ILE	Integral non-linearity			± 1	LSB
DLE	Differential linearity			± 0.25	LSB
THD	Total harmonic distortion @ 100 kHz[a]		60		dB
SNR	Signal to noise ratio[a]		62		dB

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

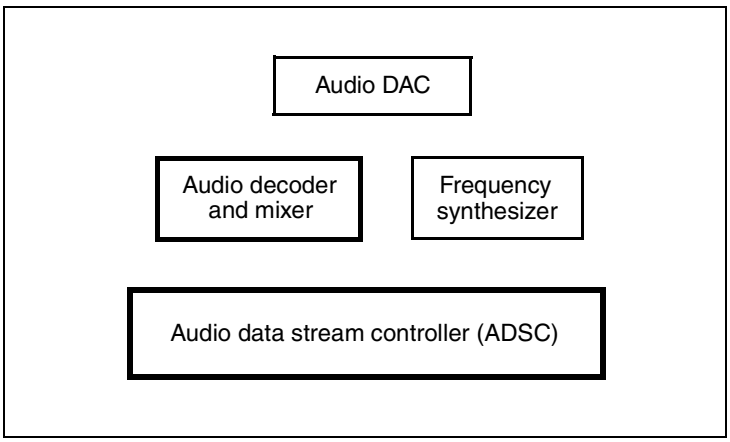
74 Audio data-stream controller (ADSC)

74.1 Overview of the audio subsystem

The audio subsystem includes the following elements:

- **Audio data-stream controller (ADSC)**
  This controls and manages all the audio data streams. It feeds the audio decoder and the S/PDIF formatter with appropriate audio data. It interfaces the external audio input stream.
- **Audio decoder and mixer**
  This audio DSP decodes and mixes the audio stream with PCM audio data and delivers it, coded, decoded or mixed, to the S/PDIF output. The audio stream output is synchronized with the video stream.
- **Frequency synthesizer** associated with the decoder,
- **Audio DAC**.

Figure 243: STi7710 audio subsystem design hierarchy



The audio subsystem supports the following decoding algorithms:

- MPEG 1 (layers I,II,III)
- MPEG 2 layer II multi-channel
- Dolby AC3 (5:1 and 2 channel)
- MP3
- AAC

It also supports a set of post-processing algorithms:

- Dolby Pro Logic
- SRS TruSurround
- SRS TruBass
- SRS Focus

The audio subsystem is able to decode one compressed audio stream and mix it with a PCM “file” stored in local memory in a PCM buffer. The coded or decoded bitstream can be concurrently delivered to an S/PDIF output.

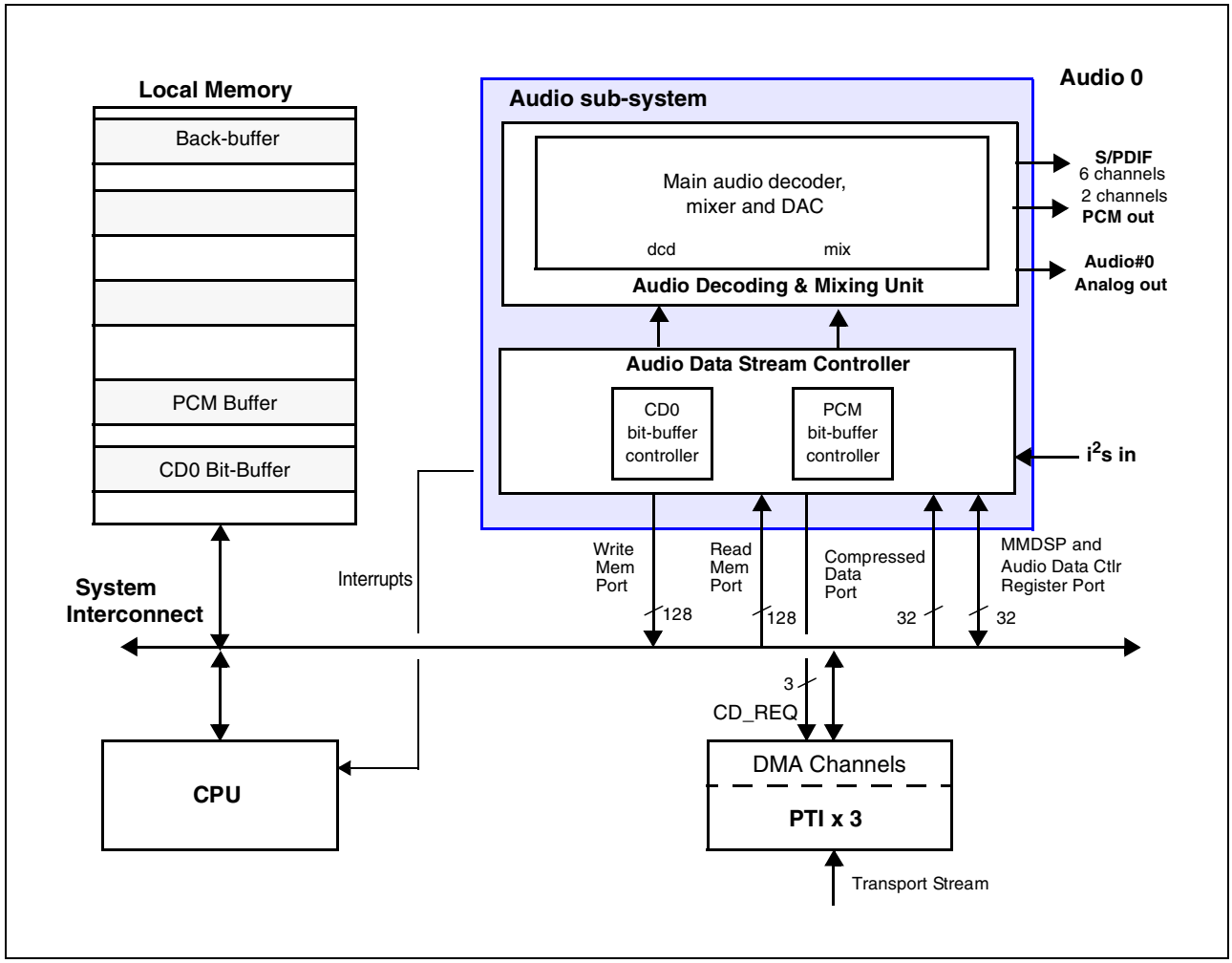
The audio decoder includes a PCM mixer. The data format of the audio streams feeding the audio subsystem can be compressed data (CD) or PCM data.

Both CD and PCM data streams can be internally or externally sourced. When externally sourced, the audio stream is provided to the audio sub-system via an i2s input (serial data).

When internally sourced, the audio stream is read from memory either by FMDA or PTI channel1, and feeds the ADSC either directly from the PTI, or via back-buffering.

Figure 244 shows the audio subsystem within the context of the whole system. The audio subsystem exchanges data or signals with memory, CPU and the PTI as shown in Figure 244.

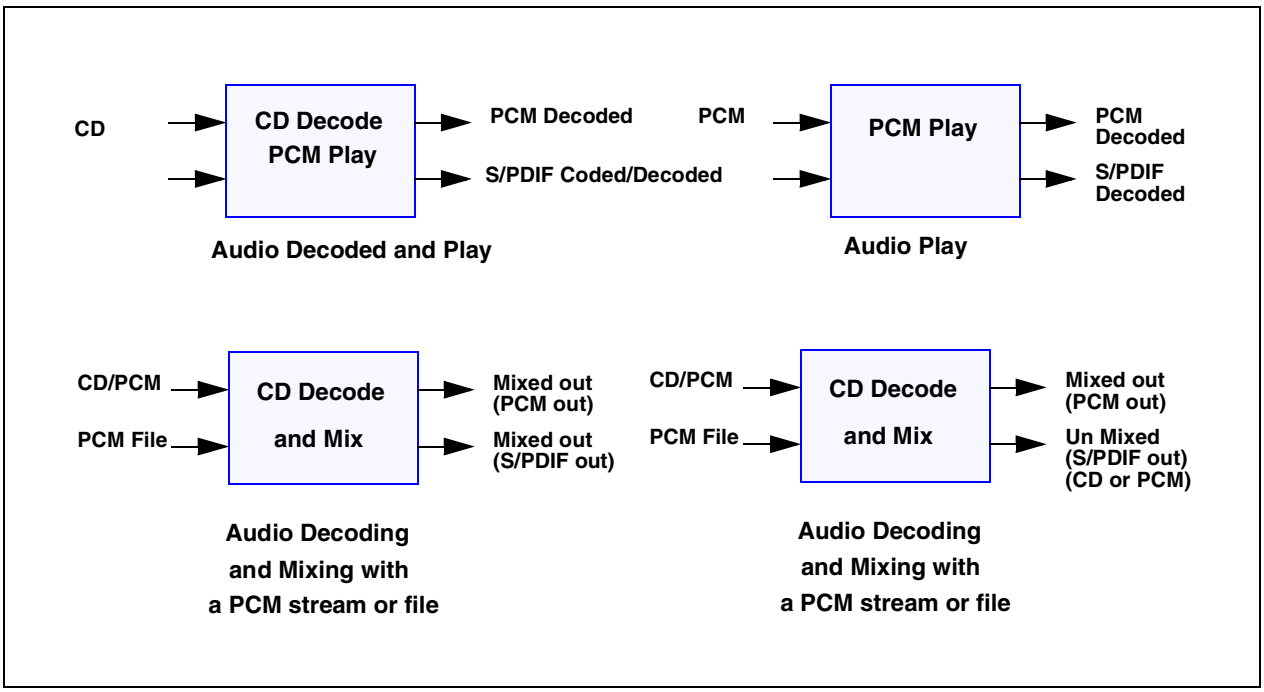
Figure 244: Audio subsystem in context



74.2 Overview of operation

The different scenarios are summarized in Figure 245 below, assuming a dual-audio decode capability.

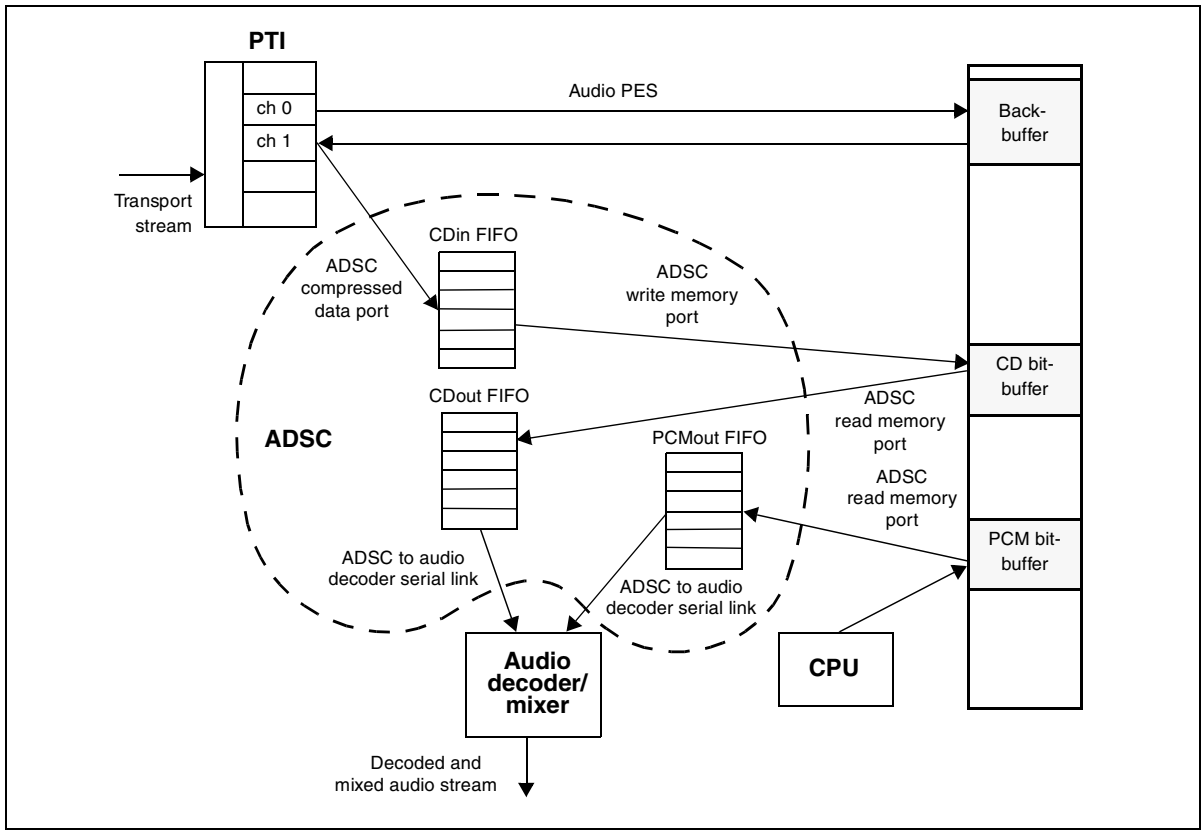
Figure 245: Audio subsystem decode/play scenarios



The data (compressed or PCM) is buffered in memory in a circular bit-buffer (CD0 bit-buffers) or in a static PCM buffer fully managed by software. Data from bit-buffer 0 can be sent to the internal decoder. Data stored in the PCM buffer can only be sent to the PCM mixer input of the decoder.

Figure 246 below shows a standard audio decode flow, where the ADSC mixes a compressed data stream and a PCM data stream.

Figure 246: Standard audio decode data flow



Transfer of compressed data from the PTI to the ADSC

The PTI (Programmable Transport Interface) receives and processes multiple transport streams. After demuxing and descrambling, the PTI sends the audio compressed data (PES) either directly to the ADSC via the compressed data port, or to an intermediate back-buffer located in memory using its DMA engine channel 0. The back-buffer data is then transferred to the compressed data port of the ADSC by the PTI DMA engine (channel 1).

The PTI channel 0 DMA Engine is used to back-buffer the transport stream in a circular buffer.

PTI channel 1 performs single word read and write accesses, and is used to read the channel 0 buffer, and to write the data into the compressed data port of the ADSC. Channel 1 supports flow regulation with the CD0_REQ signal, which is asserted by the ADSC when the ADSC local CDin FIFO has no more room enough to store new data from the PTI. Channel 0 has no flow regulation.

Compressed data bit buffer

After some internal buffering into the local CDin FIFOs of the ADSC, the data is then written in memory in the appropriate bit buffers CD0, CD1 or CD2 via the write memory port of the ADSC. These CD-bit buffers are managed as circular buffers.

When the internal CDout FIFOs of the CD bit-buffer controllers feeding the decoders are not full enough, data is requested by the ADSC and transferred from the bit buffers located in memory into the audio data-stream controller via the read memory port. This data is then serialized to feed the decoders and the mixer of the audio decoding and mixing unit. The audio decoder then parse the PES packets to extract the audio data and to decode the compressed data (ES).

PCM bit buffer

The PCM buffer is written in memory by the CPU or a DMA engine. The PCM data is then transferred to the audio data-stream controller via the read memory port to feed the mixer input of the main audio decoder. The PCM buffer pointers are completely managed by the CPU.

The PCM and CD bit-buffer controllers handle the data transfers, and control the bit-buffer levels and addressing (linear, circular, link-list, one-shot). They also ensure synchronization with the CPU via interrupts. Finally, they feed the DSP with the data to be decoded, played or mixed.

External audio stream

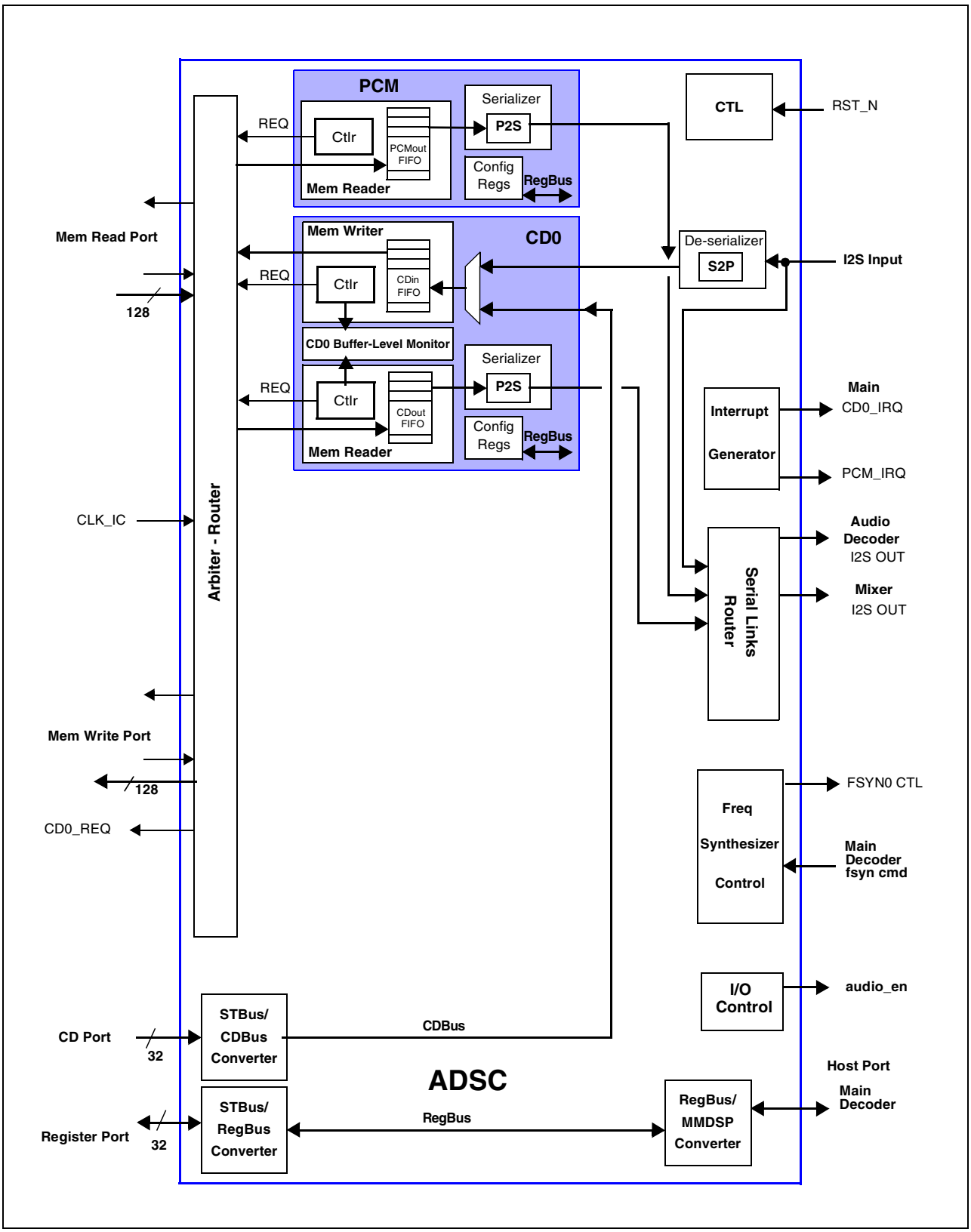
External data arriving via the I²S external input can be captured and buffered by the ADSC, prior to being written in memory in the right bit buffer via the write memory port.

Audio data streams control

The DSP registers and audio data-stream controller configuration registers are read and written by the CPU via the audio decoder and register port of the ADSC. More details can be found in Chapter 75: *Audio data-stream controller (ADSC) registers* *on page 832*.

74.3 ADSC block diagram

Figure 247: ADSC block diagram



74.4 ADSC functionality

74.4.1 CD0 bit-buffer management

The CD0 bit buffer is located in memory and managed as a circular buffer. It contains data that is written and then read by the ADSC itself. The data written into the CD bit buffer can be sourced from the CD port or the external I²S input. The status of the bit buffer and CDin FIFO are available to the CPU via status registers and interrupts.

Bit-buffer definition

The buffer is defined by the following configuration registers:

AUD_BBG: Start of bit buffer. Defines the start address of the buffer in memory.

AUD_BBS: End of bit buffer. Defines the stop address of the buffer in memory.

AUD_BBL: Bit-buffer level. Gives an indication of the buffer's filling level.

AUD_BBT: Bit-buffer threshold. Defines a filling threshold. When this threshold is reached by BBL, status bit AUD_STA.BBF (bit buffer nearly full) is set.

The addresses AUD_BBS and AUD_BBG are aligned on 256-byte boundaries (which correspond to the CDin and CDout FIFOs size). The bottom seven bits of AUD_BBG are assumed to be all zeros and the bottom seven bits of AUD_BBS are assumed to be all ones. The bit buffer can be placed anywhere in the memory space.

Therefore, a bit buffer defined by BBG and BBS contains (BBS - BBG +1) x 256 bytes.

Bit-buffer operation

When the write pointer reaches the value AUD_BBS + 255, the pointer wraps around to AUD_BBG.

When the bit buffer is empty, status bit AUD_STA.BBE (bit buffer empty) is set.

Bit-buffer overflow can be prevented by setting control bit AUD_STP.PBO. This stops the data transfer from the CDin FIFO into the bit buffer when the buffer level reaches AUD_BBS.

Setting bit AUD_PLY.EN enables data transfers between memory and ADSC.

Bit-buffer source selection and capture

The source of the bit buffer can be either the CD port or the external I²S input. This is defined by bit AUD_STP.CP. Setting this bit selects the external I²S input; resetting it selects the CD port.

Setting bit AUD_PLY.CAP causes data from the CD port or the I²S external input to be written to the CDin FIFO.

Bit-buffer status

Register AUD_STA contains the current state of the bit buffers.

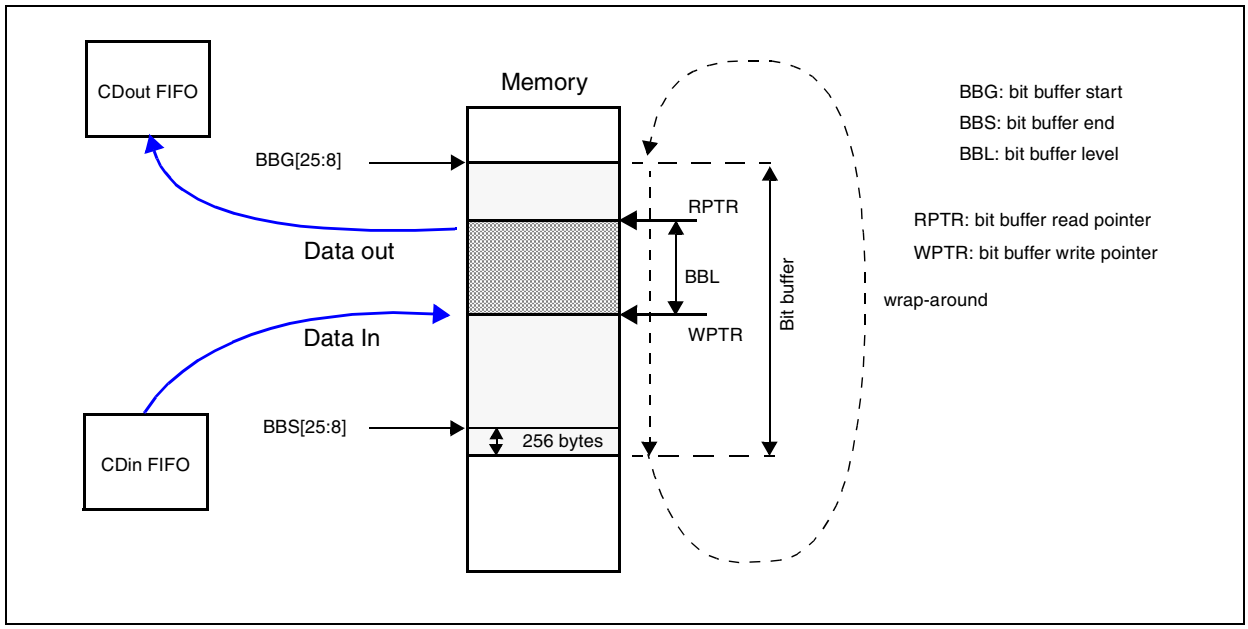
- **Bit buffer empty**: When the bit buffer is empty, bit BBE is set.
- **Bit buffer nearly full**: When the bit buffer level indicated in register AUD_BBL is equal or greater than a predefined threshold indicated in register AUD_BBT, bit BBF is set.
- **CDIn FIFO full**: When the CDin FIFO is full, bit BFF is set.
- **Data transfer error**: When a data transfer error occurs (identified by R_OPC asserting) bit XFE is set.

Bit-buffer interrupts

The interrupts generated by the ADSC are governed by registers AUD_STA (bit-buffer status), AUD_ITS (interrupt status) and AUD_ITM (interrupt mask). As soon as a bit of AUD_STA is set, the corresponding bit of register AUD_ITS is set independently of the corresponding bit in AUD_ITM. If the corresponding bit of AUD_ITM is set, the interrupt is asserted.

Reading register AUD_ITS clears all bits and de-asserts the interrupt signal.

Figure 248: CDn circular bit buffer



74.4.2 PCM buffer management

The PCM buffer located in memory is completely managed by the CPU. PCM files are written by the CPU or the DMA controller directly to the PCM buffer located in memory. These files are played by correctly positioning the PCM buffer pointers.

Note: MIDI and wave files are not supported, and must be converted to PCM by a user application before playing.

Two modes are available to read the PCM buffer from memory: linked-list and one-shot mode.

The status of the PCM buffer is available to the CPU via a status register and an interrupt.

PCM buffer definition

The PCM buffer is defined by the following configuration registers:

- PCM_READ: **PCM buffer read pointer**
  Contains the current read pointer value. This value is set initially by the CPU and then updated by the CD bit-buffer controller when data transfers are executed.
- PCM_REF: **PCM buffer reference pointer**
  When the current read pointer reaches this value, an interrupt is generated and the PCM play stops or continues jumping to the next read pointer depending on bit PCM_MODE.PCLL.
- PCM_NXT: **PCM buffer next read pointer**
  Holds the next read pointer value. When the current read pointer reaches the value of the reference pointer PCM_REF, the value held in PCM_NXT is transferred to PCM_READ if the linked-list mode is enabled.
- PCM_WRITE: **PCM buffer write threshold pointer**
  This pointer is set by the CPU and gives an indication of the write pointer value. When the read pointer reaches this value, an interrupt is generated. This indicates that the buffer is empty or almost empty. It is used to detect error conditions.

The PCM_REF, PCM_READ, PCM_NXT and PCM_WRITE addresses are aligned on 128-byte boundaries. The bottom six bits of these addresses are assumed to be all 0.

PCM buffer modes of operation

Two modes are available and defined by bit PCM_MODE.PCLL. Setting this bit configures the PCM buffer in linked-list mode.

- **Linked-list mode**: as soon as the buffer has been played, the start address of the next buffer is automatically loaded and play continues.
- **One-shot mode**: in this mode, the play operation stops when the read pointer reaches the end of the buffer or the write pointer.

PCM buffer enable

When set, bit PCM_PLAY.EN enables data transfers from memory to the ADSC.

When set, bit PCM_PLAY.PAU stops data transfers from memory to the ADSC and sends zeros to the mixer input.

PCM buffer status

Register PCM_STA gives the state of the PCM buffer at any time.

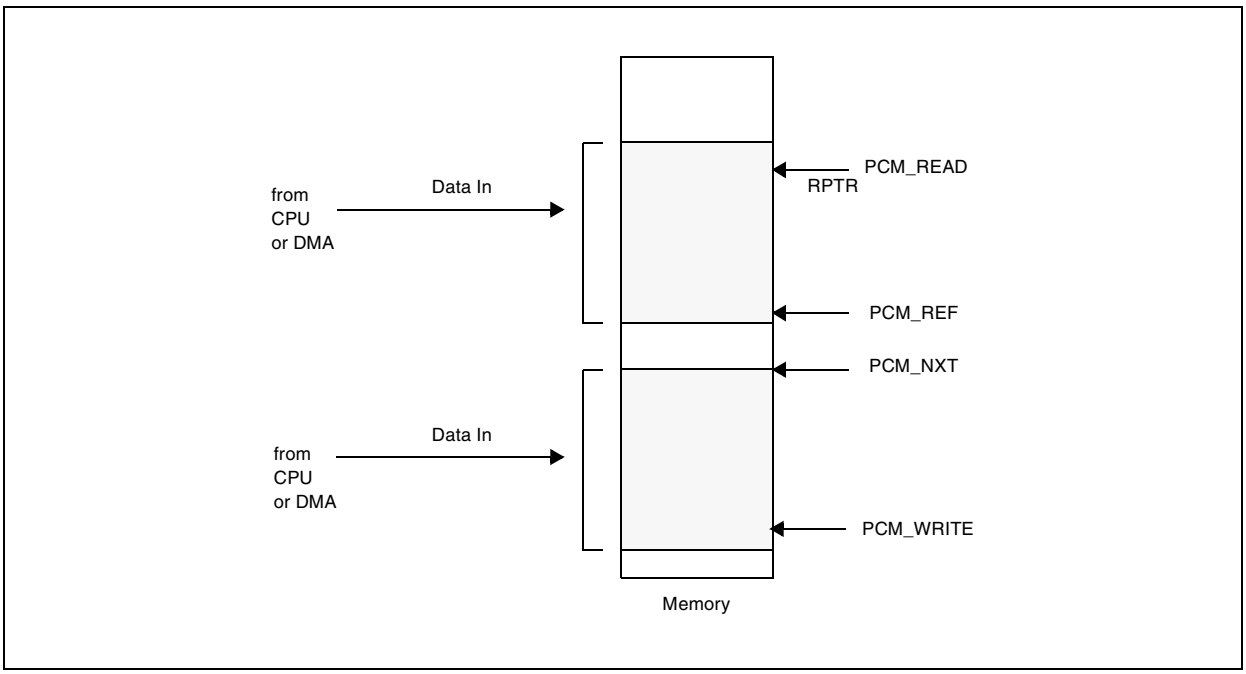
- **PCM buffer empty**: When the PCM buffer is empty, bit PBE is set.
- **PCM buffer reference pointer reached**: When this condition happens, bit ARR is set.
- **Data transfer error**: When a data transfer error occurs (identified by R_OPC asserting) then the bit PCM_STA.XFE is set.

PCM buffer interrupts

The interrupts generated by the ADSC are governed by registers PCM_STA (bit-buffer status), PCM_ITS (interrupt status) and PCM_ITM (interrupt mask).

As soon as a bit of PCM_STA is set, the corresponding bit of PCM_ITS is set, independently of the corresponding bit in PCM_ITM. If the corresponding bit of PCM_ITM is set, the interrupt is asserted. Reading register PCM_ITS clears all bits, de-asserting the interrupt signal.

Figure 249: PCM buffer



Programming sequence

Linked-list mode:

1. Disable the PCM bit buffer.
2. Define the PCM bit buffer: PCM_READ, PCM_REF, PCM_NXT, PCM_MODE, PCM_ITM.
3. Enable the PCM bit buffer.

One-shot mode:

1. Disable the PCM bit buffer.
2. Define the PCM bit buffer: PCM_READ, PCM_REF, PCM_MODE, PCM_ITM.
3. Enable the PCM bit buffer.

When the reference address is reached, the DMA is stopped and an interrupt is generated. To continue, the following sequence can be applied:

1. Disable the PCM bit buffer when receiving the interrupt.
2. Redefine the PCM_READ, PCM_REF, PCM_NXT pointers and the PCM_MODE.
3. Re-enable the PCM bit buffer.

74.4.3 Serial data output operation

The audio data-stream controller gets its data from the CD0 or PCM bit buffers, and delivers this data to the audio decoder and mixer through serial links. Therefore two serializers are included in the ADSC.

The serialization operation of each serializer is defined by the format of the parallel data, the data endianness and the serialization mode.

Memory storage format

Audio samples may be either 8, 16, 24 or 32 bits, and the data may be stored in memory in different formats. The formats (A, B, C and D) are described in Figure 251: *Memory storage formats* *on page 826*. This data can be mapped as big-endian (lowest address to the highest byte of the scalar), or little endian (lowest address to the lowest byte of the scalar).

Data size and data position

For format A, the size of the data to be serialized and the positioning of the data within a 32-bit field is defined in register AUD_SER. Field DSIZE[4:0] defines the size of the data to serialize (usually 16, 18, 20, 24 or 32 bits) and field MSBPOS[4:0] defines the number of left shifts necessary to left-align the data. This information is considered only for the serialization modes 2, 3, 4, 5 and 7.

Serialization order

When serializing a 32-bit word, the first bit to be output is the MSB (WORD[31]).

Byte re-ordering

The byte-ordering of the received data may be little- or big-endian. This feature is supported by a byte swapping capability defined by bits AUD_SER.WSWP[1:0], BSWP2 and BSWP4.

Serialization modes

Up to eight output formats are supported by the serializer. These formats (modes 0 to 7) are described in Figure 252: *Serial output modes* *on page 827*. The format is defined by AUD_SER.SMODE[3:0].

Left-right clock polarity

The first sample produced by the PCM player can be delivered with the LR clock either high or low. This is indicated by AUD_SER.LRCP. When field LRCP is set to 0, the first sample is output with LR clock high; when the bit is set to 1, the first sample is output with LR clock low.

The default value after reset delivers the first sample with LR clock high.

Serializer source selection

The ADSC has two serial outputs, intended to be connected to the serial inputs of the main audio decoder and mixer.

The serial stream can be originated from the CD0 or PCM bit-buffer controller. The source of each serial output is indicated in registers I2S_MDEC_CFG and I2S_MIXR_CFG.

Figure 250: Serializer source selection

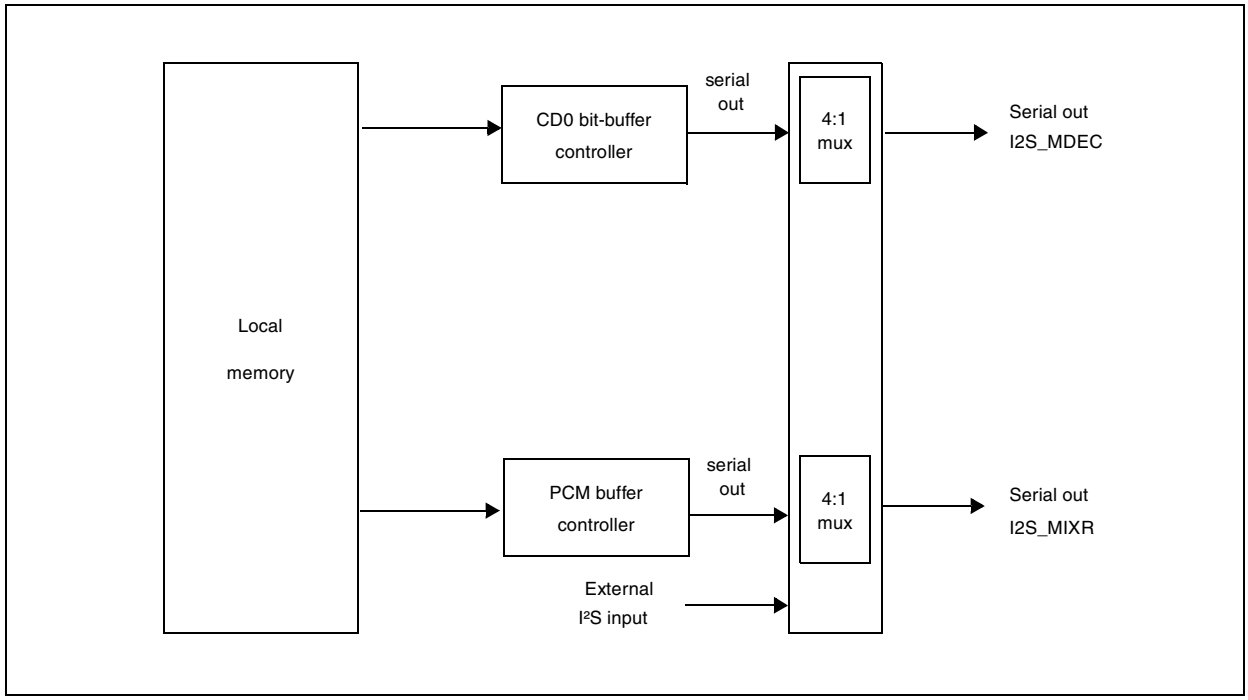


Figure 251: Memory storage formats

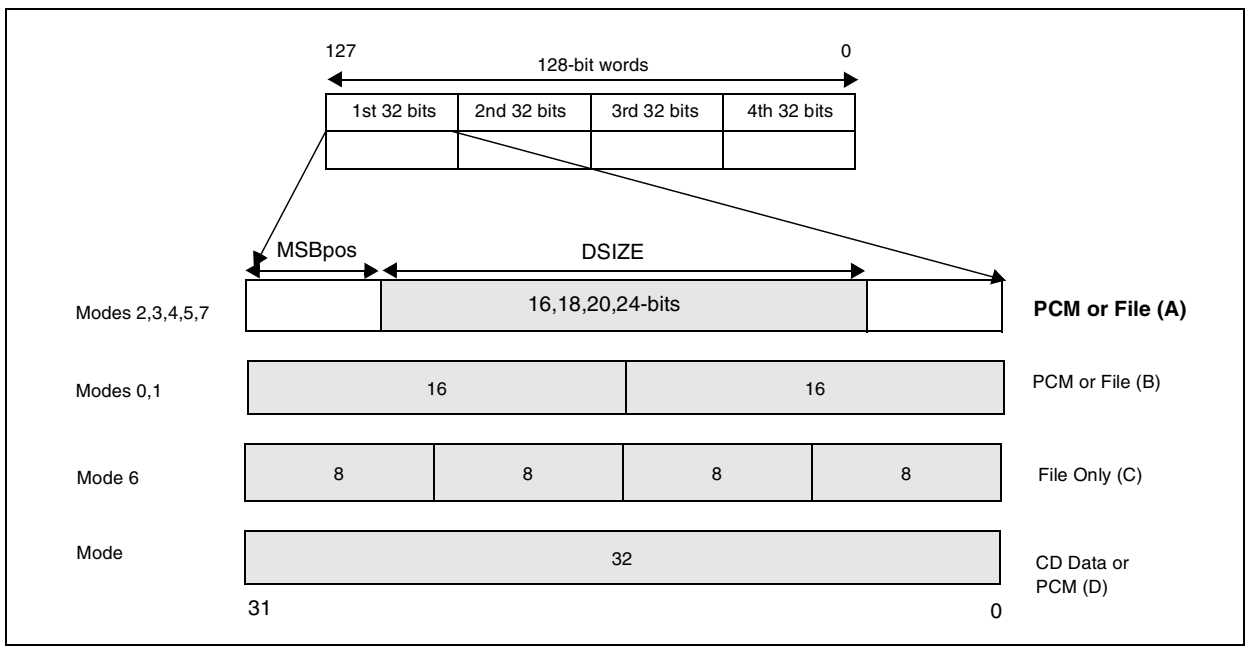


Figure 252: Serial output modes

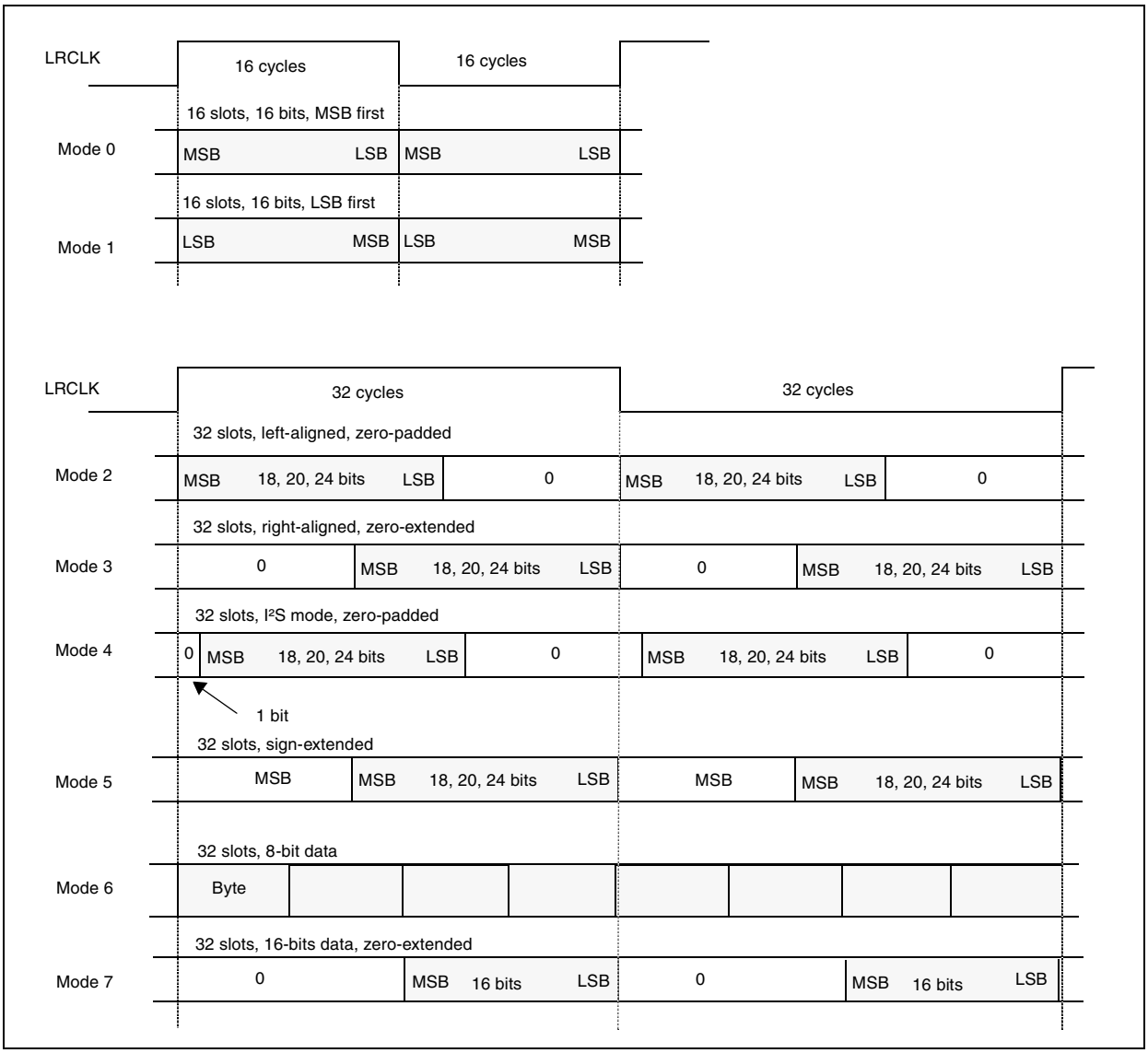


Table 199: Serializer configuration examples

Description	Input Format	Output Format	LR Clock
16, 18, 20, 24 bits PCM samples	A	2, 3, 4, 5, 7	32 serial clock cycles
16 bits PCM	B	0, 1	16 serial clock cycles
Byte File	C	6	32 serial clock cycles
Compressed Data	D	MSB first	N/A
Serial-In same as Serial-Out	D	MSB first	32 or 16 serial clock

74.4.4 Serial input operation

The deserializer block receives its data from the external I²S serial input, builds a 32-bit word, and writes the word into one of the local CDin FIFOs.

Serial input format

In a similar way as the serializer, the deserializer supports the formats Mode 0 to 7 described in Figure 252: *Serial output modes* *on page 827*.

CD bit-buffer selection

The 32-bit word is routed to one or more of the CDin FIFOs as indicated by bit AUD_STP.CP. Setting CP routes the deserializer output to the CDin FIFO instead of the compressed data port.

Memory storage

Data is stored in memory in blocks of 16 bytes by the CD0 bit buffer controller via the write memory port. Data is stored as it is received, in words of 32 bits, representing either one 32-slot word or two 16-slot words.

Left-right clock polarity

The first sample captured by the I²S input can be either in phase with the LR clock high or low. This is indicated by bit PCMI_CFG.LRCP. When LRCP is set to 0, the first sample is captured with LR clock high; when it is set to 1, the first sample is captured with LR clock low.

The default value after reset captures the first sample with LR clock high.

Capture enable

The capture of the serial input stream can be enabled or disabled. This is specified by PCMI_CFG.CAP.

Capture start

The start of the capture can be synchronized either on the first LR clock valid edge (defined by the left-right clock polarity) or on the second valid edge. This is specified by PCMI_CFG.DCK.

Deserializer soft reset

At any time, the deserializer can be reset with bit PCMI_CFG.SRS.

74.4.5 Data storage organization

The bit buffers located in memory contain either bitstreams or PCM samples whose size can be 8, 16, 20 or 24 bits. This data may be read and written by different devices having different endianness properties.

The devices that are able to write to the bit buffers in memory are:

- The PTI via the compressed data port and then via the write memory port,
- The serial I²S input via the write memory port,
- The CPU (ST20),

The devices that are able to read the bit buffers in memory are:

- The read memory port (CD and PCM bit-buffer controllers) to feed the DSP serializers,
- The CPU.

Nevertheless, since the STi7710 contains an ST20 CPU, it is considered a little-endian system.

Endianness versus bslbf128 organization

The compressed data port and the deserializer write data in memory via the write memory port in blocks of 16 bytes (128 bits) as described below. Within this 128-bit string, the left most bit represents the MSB of the earliest data. This organization is referred in this document as bslbf128 (bit-string left bit first 128 bits). The data is written in memory in blocks of 16 bytes incrementing the address by 16 for each new block of 16 bytes.

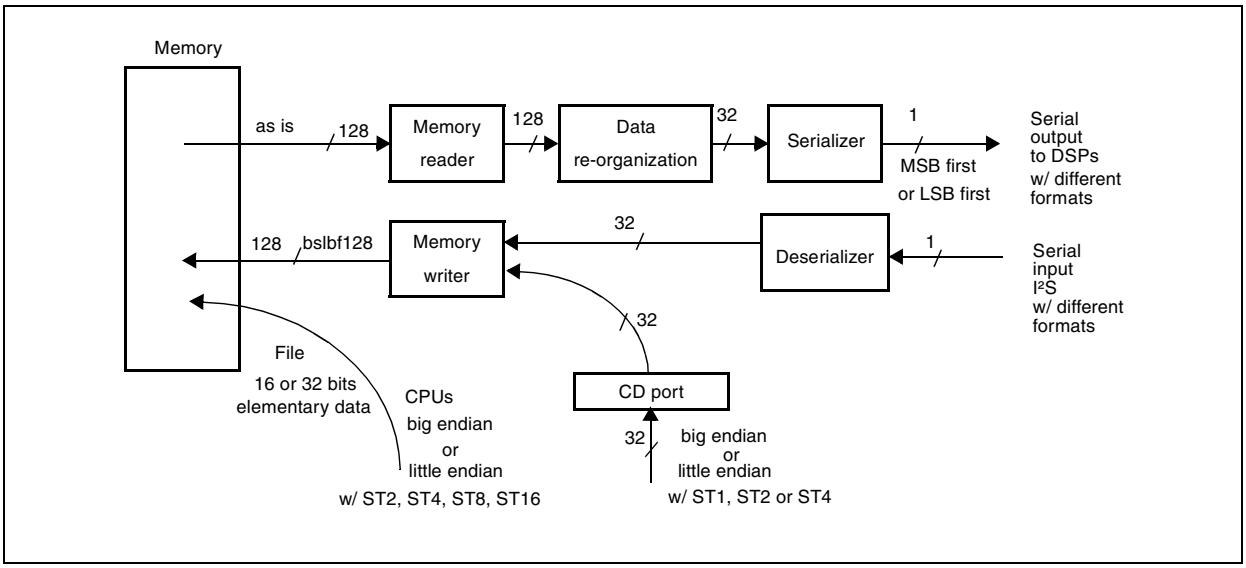
The CPU may write data with big-endian or little-endian byte mapping.

The serializers that get their data via the read memory port expect bslbf128 data organization.

The serial data must be sent to the DSPs MSB first or eventually LSB first. As a consequence, if the data is organized differently in memory (especially in the PCM buffer), after DMA transfers or

CPU write operations, data re-organization (byte reordering) must take place between the memory and the serializers. This is summarized in Figure 253.

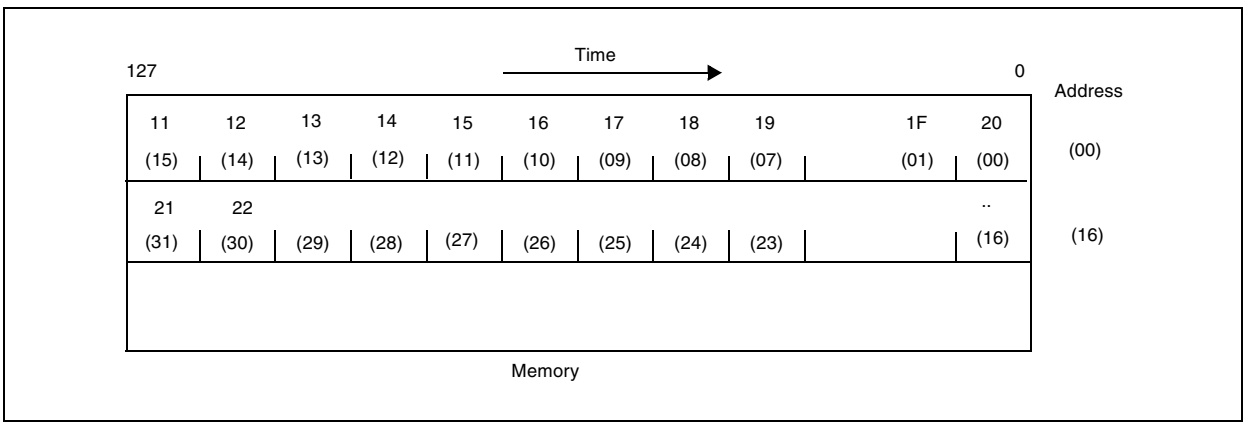
Figure 253: Data streams to/from memory



bslbf128 storage organization

If the incoming bitstream is “11 12 13 14 15 16 17 18 19 1A.... 1F...”, then the data is stored in memory as described in Figure 254. The first byte received is 11.

Figure 254: bslbf128 memory storage

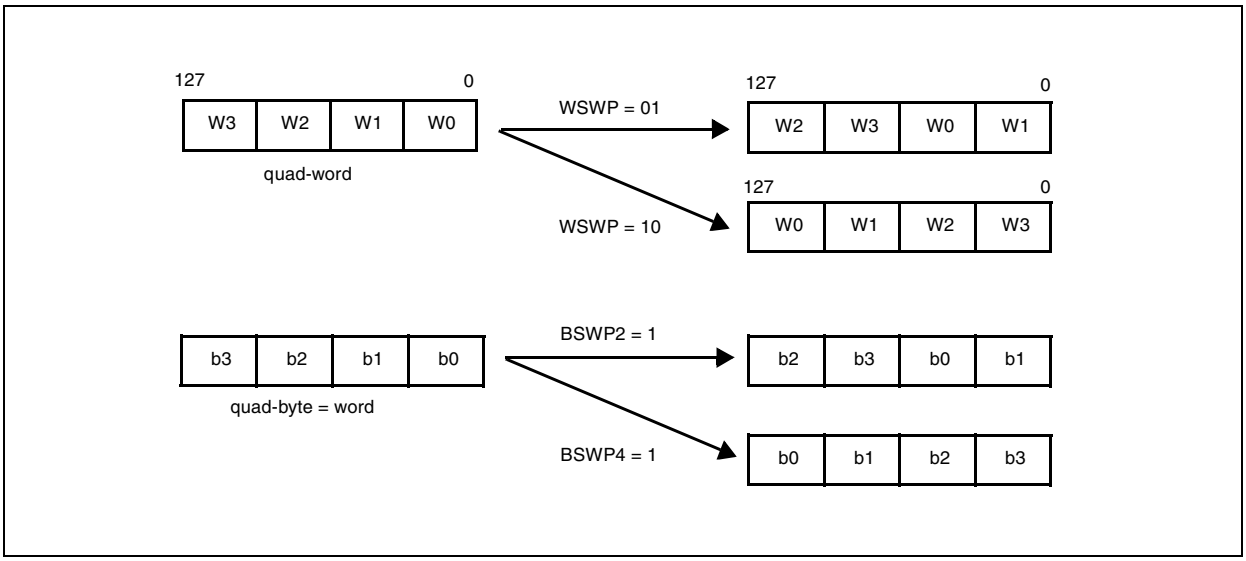


Data re-organization

The ADSC includes control registers to help to solve these data organization discrepancies. These are not intended to solve all potential problems that could occur with software that is not consistent in monitoring the endianness.The ADSC provides the following configuration registers to control data re-organization:

- AUD_SER.WSWP[1:0]: word swap in a quad-word. These two bits define word-swapping within a quad-word (128 bits - W3W2W1W0) read from memory. The left most word (32 bits) is sent first to the serializer.
- AUD_SER.BSWP2: If set, swap two consecutive bytes in a word.
- AUD_SER.BSWP4: If set, swap four consecutive bytes in a word.

Figure 255: Word and byte swap



74.5 Compressed data port

This port is used by the PTI DMA channels or the CPU to store compressed data in the CDin FIFO.

Operation

Since the PTI (programmable transport interface) may use different opcodes, this subblock includes a small four-byte FIFO located between the STBus and the CD FIFO. Data is transferred between this interface and the FIFO in blocks of 32 bits (4 bytes). Therefore, no data is written into the FIFO until a complete 4-byte word is available. The data is routed toward the CD0 local FIFO.

Data organization in memory

The data are stored in memory using the bslbf128 organization described previously.

If the audio stream is: 11 12 13 14 15 16 17 18 19 1A 1B 1C... (11 being the first and earliest byte), the data is stored in memory as follows:

- Memory ADDR[15:0]: 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20
- Memory ADDR[31:16]: 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30

STBus interface

The STBus interface is a write-only interface. Only store operations are supported.

Unmapped addresses

The CD port address space is 4 kbytes. When an access to an unimplemented address is executed, a response opcode fail is returned.

74.6 Interrupts

The ADSC generates two interrupts which are routed to the Interrupt Level Controller (ILC). These are:

- CD0 bit-buffer level interrupt
- PCM bit-buffer level interrupt

The behavior of these interrupts is defined by registers AUD_STA, AUD_ITM, AUD_ITS, PCM_STA, PCM_ITM and PCM_ITS.

Operation

When a change occurs in a Status Register (AUD_STA), the corresponding bit is set in the Interrupt Status Register (AUD_ITS). If the corresponding Interrupt mask bit is set then an interrupt occurs.

Reading the register AUD_ITS clears all the bits of this register. When AUD_ITS is cleared, the corresponding IRQ line returns to 0.

75 Audio data-stream controller (ADSC) registers

Register addresses are provided as *ADSCBaseAddress* + offset.

The *ADSCBaseAddress* is:

0x3821 0000.

Table 200: ADSC register summary

Register	Description	Offset	Type
Audio channel			
AUD_BBG	Audio - start of bit buffer	0x0000	R/W
AUD_BBS	Audio - end of bit buffer	0x0004	R/W
AUD_BBT	Audio - bit buffer threshold	0x0008	R/W
AUD_BBL	Audio - bit buffer level	0x000C	RO
AUD_STP	Audio - setup	0x0020	R/W
AUD_PLY	Audio - play	0x0024	R/W
AUD_SRS	Audio - soft reset status	0x0028	R/W
AUD_CHK	Audio - chunk Size configuration	0x002C	R/W
AUD_STA	Audio - status	0x0040	RO
AUD_ITM	Audio - interrupt mask	0x0044	R/W
AUD_ITS	Audio - interrupt status	0x0048	RO
AUD_SER	Audio - serializer configuration	0x0050	R/W
PCM channel			
PCM_REF	PCM - buffer reference pointer	0x0300	R/W
PCM_NXT	PCM - buffer next read pointer	0x0304	R/W
PCM_READ	PCM - buffer read pointer	0x0308	R/W
PCM_WRITE	PCM - buffer write pointer	0x030C	R/W
PCM_MODE	PCM - playing mode	0x0320	R/W
PCM_PLAY	PCM - playing control	0x0324	R/W
PCM_SRS	PCM - soft reset status	0x0328	R/W
PCM_CHK	PCM - chunk size configuration	0x0330	R/W
PCM_STA	PCM - status	0x0340	R/W
PCM_ITM	PCM - interrupt mask	0x0344	R/W
PCM_ITS	PCM - interrupt status	0x0348	R/W
PCM_SER	PCM - serializer configuration	0x0350	R/W
External serial input (I^2S)			
PCMI_CFG	PCMI - serial input configuration	0x0420	R/W

Table 200: ADSC register summary

Register	Description	Offset	Type
Serial outputs (I²S)			
I2S_MDEC_CFG	I²S main decoder configuration	0x0500	R/W
I2S_MIXR_CFG	I²S mixer configuration	0x0504	R/W
Clockbit configuration			
CKG_BIT_CFG	Clockbit configuration	0x0508	R/W
Clock generation			
CKG_CFG	Clock - generation configuration	0x0600	R/W
CKG_MD	Clock - frequency synthesizer MD parameter	0x0604	R/W
CKG_PE	Clock - frequency synthesizer PE parameter	0x0608	R/W
CKG_SDIV	Clock - frequency synthesizer SDIV parameter	0x060C	R/W
CKG_PSEL	Clock - generation parameter selection	0x0610	R/W
CKG_PROG	Clock - generation parameter validation	0x0614	R/W
I/O control			
IO_CFG	CFG -audio I/O configuration	0x0700	R/W
Audio DAC control			
ADAC_CFG	ADAC - audio DAC configuration	0x0800	R/W
Reserved			
-	-	0x0900	-
		0x0904	

The Audio Compressed Data Input Port is mapped at absolute address 0x2010 0000 and is used as a portal to access the compressed data FIFO through 0D DMAs.

75.1 Audio channel

AUD_BBG Start of bit buffer

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

Address: *ADSCBaseAddress* + 0x0000
Type: R/W
Reset: 0x00
Description: This register holds the starting address of the audio bit buffer in units of 256 bytes.

AUD_BBS End of bit buffer

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

Address: *ADSCBaseAddress* + 0x0004
Type: R/W
Reset: 0x00
Description: This register holds the ending address of the audio bit buffer in units of 256 bytes.

AUD_BBT Bit-buffer threshold

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

Address: *ADSCBaseAddress* + 0x0008
Type: R/W
Reset: 0x00
Description: Stores the threshold level of occupancy of the audio bit buffer, in units of 256 bytes. When this threshold is reached (AUD_BBL is greater than or equal to AUD_BBT), status bit AUD_STA.BBF is set.

AUD_BBL Bit-buffer level

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

Address: *ADSCBaseAddress* + 0x000C

Type: RO

Reset: 0x00

Description: Stores the current level of filling of the audio bit buffer, defined in units of 256 bytes. It can be read at any time.

When AUD_BBL is greater than or equal to AUD_BBT, status bit AUD_STA.BBF is set. When AUD_BBL is zero, status bit AUD_STA.BBE is set.

AUD_STP Audio setup

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

Address: *ADSCBaseAddress* + 0x0020

Type: R/W

Reset: 0x00

Description: Audio channel *n* bitstream setup.

[31:3] **Reserved**

[2] **PFO**: prevent CDin FIFO overflow. When this bit is set, CDin FIFO overflow (and thus the loss of data) is prevented by disabling the transfer of data from the compressed data CD Port to the CDin FIFO when the CDin FIFO is full. The R_REQ is not returned until the CDin FIFO has enough room to accept the new data. Under normal conditions, the CDin FIFO should not overflow.

0: do not prevent CDin FIFO overflow 1: prevent CDin FIFO overflow

[1] **PBO**: prevent bit-buffer overflow. When this bit is set, bit-buffer overflow (and thus the loss of data) is prevented by disabling the transfer of data from the compressed data CDin FIFO to the bit buffer when the bit buffer is full.

0: do not prevent bit-buffer overflow 1: prevent bit-buffer overflow

[0] **CP**: CDin FIFO source. This bit indicates the source of the data written in the CDin FIFO.

0: compressed data port 1: external I²S input

AUD_PLY Audio play control

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

Address: *ADSCBaseAddress* + 0x0024

Type: R/W

Reset: 0x00

Description: Writing the appropriate bit in this register starts the software reset sequence for the corresponding audio channel, disable or enable the corresponding audio channel and disable or enable the capture of data from the CD port or the I²S input.

[31:3] **Reserved**

[2] **EN**: channel enable. When disabled, transfers from or to memory are disabled.
0: channel disabled 1: channel enabled

[1] **CAP**: capture enable. When set, the data from the CD Port or the external I²S input are captured and stored in the CDin FIFOs.
0: capture disabled 1: capture enabled

[0] **SRS**: soft reset. Setting this bit resets the FIFOn DMA channel (read and write and removes all bit-buffer contents). The reset is activated once when SRS is set. SRS must then be reset.
0: soft reset not active 1: soft reset active

AUD_SRS Soft reset status

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

Address: *ADSCBaseAddress* + 0x0028

Type: R/W

Reset: 0x00

Description: This register can be used to check that a soft reset has ended. Reset bit SRS before the soft reset. When SRS is 1, the soft reset is complete.

AUD_CHK Chunk size configuration

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

Address: *ADSCBaseAddress* + 0x002C

Type: R/W

Reset: 0x00

Description: Set the maximum chunk size, in packets.

[31:3] **Reserved**

[2:0] **CHK_SIZE**:
00: Message (8 packets) 01: 1 packet
10: 2 packets 11: 4 packets

AUD_STA Status

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

Address: *ADSCBaseAddress* + 0x0040

Type: RO

Reset: 0x00

Description: This register contains a set of bits which represent the status of the channel at any instant. Any change from 0 to 1 of any of these bits sets the corresponding bit of register AUD_ITS, and can thus potentially cause an interrupt on the AUD*n*_IRQ line. IER is a pulse and is unlikely to be read as a 1.

[31:14] **Reserved**

[13] **BFF**: CDin FIFO full.
This bit indicates when the compressed data CDin FIFO is full; equivalent to the corresponding signal pin CDN_REQ.
0: CDin FIFO not full 1: CDin FIFO full

[12] **Reserved**

[11] **BBE**: bit buffer empty. This bit indicates if the bit buffer has data or not.
0: bit buffer not empty 1: bit buffer empty

[10] **BBF**: bit buffer nearly full. This bit indicates if the bit buffer level (AUD_BBL) is greater or equal to the bit-buffer threshold (AUD_BBT).
0: bit buffer not nearly full 1: bit buffer nearly full

[9] **RXFE**: Read transfer error.
This bit indicates whether a data transfer error over the interconnect has occurred.
0: no data transfer error 1: data transfer error

[8] **WXFE**: Write transfer error.
This bit indicates whether a data transfer error over the interconnect has occurred.
0: no data transfer error 1: data transfer error

[7:0] **Reserved**

AUD_ITM Interrupt mask

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

Address: *ADSCBaseAddress* + 0x0044

Type: R/W

Reset: 0x00

Description: Any bit set in this register enables the corresponding interrupt. An interrupt is generated whenever a bit in the register changes from 0 to 1 and the corresponding mask bit is set.

AUD_ITS Interrupt status

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

Address: *ADSCBaseAddress* + 0x0048

Type: RO

Reset: 0x00

Description: When a bit in AUD_STA changes from 0 to 1, the corresponding bit in AUD_ITS is set independently of the state of the corresponding bit of AUD_ITM. If the corresponding bit in AUD_ITM is set, the interrupt is asserted. Reading AUD_ITS clears all bits in this register. Clearing AUD_ITS leaves the IRQ line in its de-asserted state.

AUD_SER Serializer configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						MSBPOS					DSIZE				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SRS	EN	WSWP		BSWP4	BSWP2	LRCP	SMODE			

Address: *ADSCBaseAddress* + 0x0050

Type: R/W

Reset: 0x00

Description: This register contains a set of bits to configure the serializer (I²S) associated with the audio channel AUD*n*.

It defines the serial mode, the left-right clock polarity, the format and the size of the data read from memory that is serialized.

When read from memory, a word and byte-swap can be optionally done.

[31:26] **Reserved**

[24:21] **MSBPOS**[4:0]: defines the position of the data MSB within a 32-bit field. Gives the number of left-shift necessary to left-align the data.
00000: MSB is in position 31 (left most position) 00001: MSB is in position 30
..... 11111: MSB is in position 0 (right most position)

[20:16] **DSIZE**[4:0]: defines the data size within a 32-bit field.
00000: 1 bit 00001: 2 bits
..... 11111: 32 bits

[15:11] **Reserved**

[10] **SRS**: soft reset.
0: de-asserted 1: asserted

[9] **EN**: serializer enable.
0: disable 1: enable

[8:7] **WSWP**[1:0]: allows the swapping of 4 consecutive 32-bits words (aligned on 16 bytes boundaries).
00: no swap: W3 W2 W1 W0 -> W3 W2 W1 W0
01: swap 2 consecutive words: W3 W2 W1 W0 -> W2 W3 W0 W1
10: swap 4 consecutive words: W3 W2 W1 W0 -> W0 W1 W2 W3
11: Reserved

[6] **BSWP4**: allows the swapping of the 4 consecutive bytes.
0: no swap (reset value) 1: bytes swap: b3 b2 b1 b0 -> b0 b1 b2 b3

[5] **BSWP2**: allows the swapping of the 2 consecutive bytes.
0: no swap (reset value) 1: bytes swap: b3 b2 b1 b0 -> b2 b3 b0 b1

[4] **LRCP**: defines the LR clock polarity.
0: first sample is produced with LR clock high 1: first sample is produced with LR clock low

[3:0] **SMODE**[3:0]: defines the serial output mode.
0000: mode 0: 16 slots, 16-bits data, MSB first
0001: mode 1: 16 slots, 16-bits data, LSB first
0010: mode 2: 32 slots, data left-aligned, padded with zeros
0011: mode 3: 32 slots, data right-aligned, extended with zeros
0100: mode 4: 32 slots, I²S mode (left most bit is zero), padded with zeros
0101: mode 5: 32 slots, data sign-extended
0110: mode 6: 32 slots, 4 x 8-bits data
0111: mode 7: 32 slots, 16-bits data right-aligned, extended with zeros
others

AUD_CDI Compress data input

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

Address: 0x2010 0000
Type: WO
Reset: Undefined
Description: Audio compressed data for channel AUD*n* can be written through this register.

75.2 PCM channel

PCM_REF Audio PCM buffer reference pointer

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

Address: *ADSCBaseAddress* + 0x0300
Type: R/W
Reset: 0x00
Description: This register holds the reference pointer of the audio PCM buffer in units of 128 bytes. When the read pointer reaches the reference pointer, an interrupt is generated and PCM plays stops or continues jumping on next read pointer depending on bit PCM_MODE.PCLL.

PCM_NXT Audio PCM buffer next read pointer

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

Address: *ADSCBaseAddress* + 0x0304
Type: R/W
Reset: 0x00
Description: This register holds the next read pointer address in units of 128 bytes. When the current read pointer reaches REF (PCM_REF), the value programmed in this register is automatically transferred to the read pointer register. This mechanism is used to provide continuous linked-list play of multiple, non-contiguous, PCM buffers.

PCM_READ Audio PCM buffer read pointer

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

Address: *ADSCBaseAddress* + 0x0308
Type: R/W
Reset: 0x00
Description: This register holds the current PCM samples buffer read pointer address in units of 128 bytes. This pointer is updated as the DMA progresses through the buffer.

PCM_WRITE Audio PCM buffer write threshold pointer

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

Address: *ADSCBaseAddress* + 0x030C
Type: R/W
Reset: 0x00
Description: This register holds the write threshold pointer of the audio PCM buffer in units of 128 bytes. When the read pointer reaches the write pointer, an interrupt is generated. This register is used to detect error conditions.

PCM_MODE Audio PCM player mode

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

Address: *ADSCBaseAddress* + 0x0320
Type: R/W
Reset: 0x00
Description: This register selects the linked-list or the single-shot mode.

[31:1] **Reserved**

[0] **PCLL**: Select the PCM player linked-list or single-shot mode.
0: PCM player single shot. The PCM player stops when the read pointer reaches the reference pointer.
1: PCM player linked-list: When the read pointer reaches the reference pointer, the next pointer is automatically transferred to the read pointer and play continues.

PCM_PLAY Audio PCM player control

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

Address: *ADSCBaseAddress* + 0x0324

Type: R/W

Reset: 0x00

Description: This register controls the PCM player.

[31:3] **Reserved**

[2] **PAU**: pause. When set, the PCM channel stops reading data from the PCM buffer and sends zeros to the mixer input. Resetting this bit resumes the playing operation.
0: pause disabled 1: pause enabled

[1] **EN**: channel enable. When disabled, transfers from or to memory are disabled.
0: channel disabled 1: channel enabled

[0] **SRS**: soft reset. Setting this bit resets the FIFOn DMA channel (read and write and removes all bit-buffer contents). The reset is activated once when SRS is set. SRS must then be reset.
0: soft reset not active 1: soft reset active

PCM_SRS Soft reset status

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

Address: *ADSCBaseAddress* + 0x0328

Type: R/W

Reset: 0x00

Description: This register can be used to check that a soft reset has ended. Reset bit SRS before the soft reset. When SRS is 1, the soft reset is complete.

PCM_CHK Chunk size configuration

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

Address: *ADSCBaseAddress* + 0x0330

Type: R/W

Reset: 0x00

Description: Set the maximum chunk size, in packets.

[31:3] **Reserved**

[2:0] **CHK_SIZE**:
00: Message (8 packets)
01: 1 packet
10: 2 packets
11: 4 packets

PCM_STA Audio PCM status

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

Address: *ADSCBaseAddress* + 0x0340

Type: R/W

Reset: 0x00

Description: This register contains a set of bits which represent the status of the PCM channel at any instant. Any change from 0 to 1 of any of these bits sets the corresponding bit in register PCM_ITS, and therefore can potentially cause an interrupt on the PCM_IRQ line.

[31:2] **Reserved**

[1] **PBE**: PCM buffer empty. This bit, when set, indicates that the read pointer is equal to write pointer which means that the buffer is empty. This generates an error condition.

[0] **ARR**: audio reference reached. When set, indicates that the read pointer has reached PCM_REF.

PCM_ITM Audio PCM interrupt mask

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

Address: *ADSCBaseAddress* + 0x0344

Type: R/W

Reset: 0x00 (all interrupts disabled)

Description: Any bit set in this register enables the corresponding interrupt in PCM_IRQ line. An interrupt is generated whenever a bit in register PCM_STA changes from 0 to 1 and the corresponding mask bit is set.

PCM_ITS Audio PCM interrupt status

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

Address: *ADSCBaseAddress* + 0x0348

Type: R/W

Reset: 0x00

Description: When a bit in register PCM_STA changes from 0 to 1, the corresponding bit in PCM_ITS is set, independently of the state of PCM_ITM. If any PCM_ITS bit that is set is unmasked, the line PCM_IRQ is asserted. Reading the PCM_ITS register clears all bits in that register. When PCM_ITS is zero, then the PCM_IRQ line returns to its inactive state.

PCM_SER Serializer configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						MSBPOS					DSIZE				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SRS	EN	WSWP		BSWP4	BSWP2	LRCP	SMODE			

Address: *ADSCBaseAddress* + 0x0350

Type: R/W

Reset: 0x00

Description: This register contains a set of bits to configure the serializer (I²S) associated with the PCM buffer. It defines the serial mode, the left-right clock polarity, the format and the size of the data read from memory that is serialized.

When read from memory, a word and byte-swap can be optionally done.

[31:26] **Reserved**

[25:21] **MSBPOS**[4:0]: Define the position of the data MSB within a 32-bit field. Gives the number of left-shift necessary to left-align the data.
00000: MSB is in position 31 (left most position) 00001: MSB is in position 30
..... 11111: MSB is in position 0 (right most position)

[20:16] **DSIZE**[4:0]: define the data size within a 32-bit field.
00000: 1 bit 00001: 2 bits
..... 11111: 32 bits

[15:11] **Reserved**

[10] **SRS**: soft reset.
0: de-asserted 1: asserted

[9] **EN**: serializer enable.
0: disable 1: enable

[8:7] **WSWP**[1:0]: allows the swapping of 4 consecutive 32-bits words (aligned on 16 bytes boundaries).
00: no swap: W3 W2 W1 W0 -> W3 W2 W1 W0
01: swap 2 consecutive words: W3 W2 W1 W0 -> W2 W3 W0 W1
10: swap 4 consecutive words: W3 W2 W1 W0 -> W0 W1 W2 W3
11: Reserved

[6] **BSWP4**: allows the swapping of the 4 consecutive bytes.
0: no swap (reset value) 1: bytes swap: b3 b2 b1 b0 -> b0 b1 b2 b3

[5] **BSWP2**: allows the swapping of the 2 consecutive bytes.
0: no swap (reset value) 1: bytes swap: b3 b2 b1 b0 -> b2 b3 b0 b1

[4] **LRCP**: defines the LR clock polarity.
0: first sample is produced with LR clock high 1: first sample is produced with LR clock low

[3:0] **SMODE**[3:0]: defines the serial output mode.
0000: mode 0: 16 slots, 16-bits data, MSB first.
0001: mode 1: 16 slots, 16-bits data, LSB first.
0010: mode 2: 32 slots, data left-aligned, padded with zeros.
0011: mode 3: 32 slots, data right-aligned, extended with zeros.
0100: mode 4: 32 slots, I²S mode (left most bit is zero), padded with zeros.
0101: mode 5: 32 slots, data sign-extended.
0110: mode 6: 32 slots, 4 x 8-bits data.
0111: mode 7: 32 slots, 16-bits data right-aligned, extended with zeros.
others

75.3 I²S external serial input

PCMI_CFG PCM input configuration

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

Address: *ADSCBaseAddress* + 0x0420

Type: R/W

Reset: 0x00

Description: This register defines the configuration of the PCM decoder.

[31:4] **Reserved**

[3] **LRCP**: defines the LRCLK level to start the capture.
0: capture starts when LRCLK is high 1: capture starts when LRCLK is low

[2] **DCK**: defines the capture start condition.
0: capture starts when the first LRCLK valid transition occurs.
1: capture starts when the second LRCLK valid transition occurs.

[1] **CAP**: capture enable. Enable the capture.
0: capture disabled 1: capture enabled

[0] **SRS**: soft reset. Resets the I²S deserializer.
0: reset de-asserted 1: reset asserted

75.4 Serial output configuration

I2S_MDEC_CFG Main decoder I²S configuration

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

Address: *ADSCBaseAddress* + 0x0500

Type: R/W

Reset: 0x00

Description: This register defines the source of the serial link connected to the main audio decoder I²S input.

[31:3] **Reserved**

[2:0] **SEL**: main decoder I²S source selection.
000: CD0 buffer
001: PCM buffer
010: I2S External
Others: Nothing sent to decoder

I2S_MIXR_CFG Mixer I²S configuration

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

Address: *ADSCBaseAddress* + 0x0504
Type: R/W
Reset: 0x00
Description: This register defines the source of the serial link connected to the mixer I²S input.

[31:3] **Reserved**

[2:0] **SEL**: mixer I²S source selection.
000: CD0 buffer
001: PCM buffer
010: I2S External
Others: Nothing sent to decoder

75.5 Clock bit configuration

CKG_BIT_CFG Clock bit configuration

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

Address: *ADSCBaseAddress* + 0x0508
Type: R/W
Reset: 0
Description: This register defines the frequency of CLK_DSP_BIT, derived from CLK_IC.

[31:7] **Reserved**

[6:0] **DIV**: Clock factor
00 0000: DSP_CLK_BIT disabled
all others: f(CLK_DSP_BIT) = f(CLK_IC) / DIV

75.6 Clock generation

CKG_CFG Clock generation configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							S27	NDIV		POFF	PRST	PCM	Reserved		SEL

Address: *ADSCBaseAddress* + 0x0600

Type: R/W

Reset: 0x00

Description: These registers define the configuration of the frequency synthesizer associated with the main audio decoder.

[31:9] **Reserved**

[8] **S27**: internal frequency selection. Must be 1

[7:6] **NDIV**[1:0]: frequency synthesizer input divider.

[5] **POFF**: power off.
0: analog section of the synthesizer enabled 1: analog section of the synthesizer disabled

[4] **PRST**: reset frequency synthesizer. When set to 1, stops the digital part of the frequency synthesizer.
0: run 1: reset

[3] **PCM**: frequency synthesizer output selection. When set, the PCM clock is the output of the frequency synthesizer, when reset the PCM clock is the external PCM clock.
0: external PCMCLK 1: frequency synthesizer PCMCLK

[2:1] **Reserved**

[0] **SEL**: synthesizer selected. When set, the clock is synthesized, when reset the frequency synthesizer is internally bypassed and its output presents the external PCMCLK.
0: output of the frequency synthesizer is generated by the synthesizer
1: output of the frequency synthesizer is the external PCM clock input

CKG_MD Frequency synthesizer MD parameter

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

Address: *ADSCBaseAddress* + 0x0604

Type: R/W

Reset: 0x00

Description: MD: frequency synthesizer coarse selection.

CKG_PE Frequency synthesizer PE parameter

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

Address: *ADSCBaseAddress* + 0x0608
Type: R/W
Reset: 0x00
Description: PE: frequency synthesizer fine selection.

CKG_SDIV Frequency synthesizer SDIV parameter

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

Address: *ADSCBaseAddress* + 0x060C
Type: R/W
Reset: 0x00
Description: SDIV: Frequency synthesizer output divider.

CKG_PSEL Frequency synthesizer parameter selection

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

Address: *ADSCBaseAddress* + 0x0610
Type: R/W
Reset: 0x00
Description: PSEL: frequency synthesizer parameter selection. When set, MD, PE and SDIV parameters for the audio frequency synthesizer are those defined in the configuration registers. When reset, these parameters are automatically set from the PCM frequency detection done by the internal audio DSP.

CKG_PROG Frequency synthesizer programming enable

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

Address: *ADSCBaseAddress* + 0x0614
Type: R/W
Reset: 0x00
Description: When set, parameters PE and MD are taken into account by the frequency synthesizer.

75.7 I/O control

IO_CFG Audio I/O configuration

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

Address: *ADSCBaseAddress* + 0x0700
Type: R/W
Reset: 0x00
Description: This register enables the various audio outputs.

[31:3] **Reserved**

[2] **ESP**: Enable SPDIF Output
When set, SPDIF output is enabled. When reset, it is in a high-impedance state.

[1] **ECK**: Enable PCM clock Output
When set, the PCM Clock pad is enabled. When reset, it is in a high-impedance state.

[0] **EMA**: Enable Main Audio Output
When set, the audio output signals pins are enabled (PCM_OUTn, SCLK and LRCLK). When reset, these signals are high-impedance.

75.8 Audio DAC control

ADAC_CFG Audio DAC configuration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SEL	NRST	X2	NSB	SMUTE	PDN

Address: *ADSCBaseAddress* + 0x0800

Type: R/W

Reset: 0x00

Description: This register configures the audio DAC.

[31:6] **Reserved**

[5] **SEL**: DAC input selection. When 0, the input is the main decoder output. When 1, the serial I²S input is directly connected to the DAC.

[4] **NRST**: reset. Active low.

[3] **X2**: double speed input. Active high. Must be 0.

[2] **NSB**: digital power down. Active low.

[1] **SMUTE**: soft mute. Active high.

[0] **PDN**: analog power down. Active low.

76 Audio decoder

76.1 Overview

76.1.1 Input formats

The audio decoder accepts Dolby Digital, MPEG-1 layers I and II, AAC, MP3, DTS bypass, six-channel MPEG-2 layer II, PCM formats, MPEG-2 PES streams for MPEG-2, MPEG-1 and Dolby Digital.

S/PDIF input data (IEC-60958 or IEC-61937 standards) is accepted if external circuitry extracts the PCM clock from the stream and decodes the biphase mark of S/PDIF.

76.1.2 Audio video synchronization

Skip frame, repeat blocks and soft mute frame features can be used to synchronize audio and video data. PTS audio extraction is also supported.

76.1.3 Output formats

The STi7710 generates two analog audio output channels and an S/PDIF digital output.

The digital output is in Sony format. The decoder can format output data according to the IEC-60958 standard (for L/R channels) or the IEC-61937 standard (for compressed data), with a sampling frequency, F_s = 96, 48, 44.1 or 32 kHz.

76.1.4 Sampling frequencies

Sampling frequencies of F_s = 96, 48, 44.1, 32 kHz and half sampling frequencies are supported. A downsampling filter (96/48 kHz) for PCM data is available. MP3 also supports sampling frequencies of 24, 22.05, 16, 11.025 and 8 kHz.

76.1.5 Special modes

The decoder supports dual mode for MPEG and Dolby Digital. It is capable in Dolby Digital and MPEG formats, according to DVD specifications. It includes a Dolby surround compatible downmix, a Pro Logic decoder, and TruSurround XT (virtual 5.1 surround through stereo speakers).

A pink noise generator is also available to help position speakers accurately for optimal surround sound setup.

PCM beep tone is a special mode used for set-top boxes. It generates a triangular wave signal of variable frequency and amplitude on the left and right channels.

In global mute mode, the decoder decodes the incoming bit stream normally, but the PCM and S/PDIF outputs are soft muted. This mode is used to prepare a period of decoding mode that synchronizes audio and video data without sounding the audio.

76.1.6 PCM mixing

The device has PCM mixing capability with associated sample rate conversion.

76.1.7 Virtual surround

TruSurround, SRS Labs virtual technology gives two-speaker playback of multichannel audio. Six channels of both Dolby Digital (AC-3) and MPEG Multichannel audio can be played through only two speakers.

76.1.8 Start procedure

1. Set register AUD_BREAKPOINT (@ 0xAC - also called EMUL_BKCMD) to 8.
2. Set register AUD_CLOCKCMD (@ 0xE8 - also called EMUL_CLKCMD) to 0 to start the dsp clock.
3. Wait until register AUD_INT_RAM (@ 0x3FC) indicates the hard registers are initialized. (AUD_INT_RAM = 1).
4. Load the HOST register's configuration
5. Set register AUD_RUN to 1. It starts the request of data and the decoding
6. Set AUD_PLAY to 1 and AUD_MUTE to 0. The clocks are started as soon as the first data is ready to be output (that is, after one block decoding).

76.2 Architecture overview

76.2.1 Data flow

The audio decoder has a programmable core, optimized for audio decoding algorithms. Dedicated hardware performs bit stream depacking and IEC data formatting.

Figure 256: Architecture and data flow

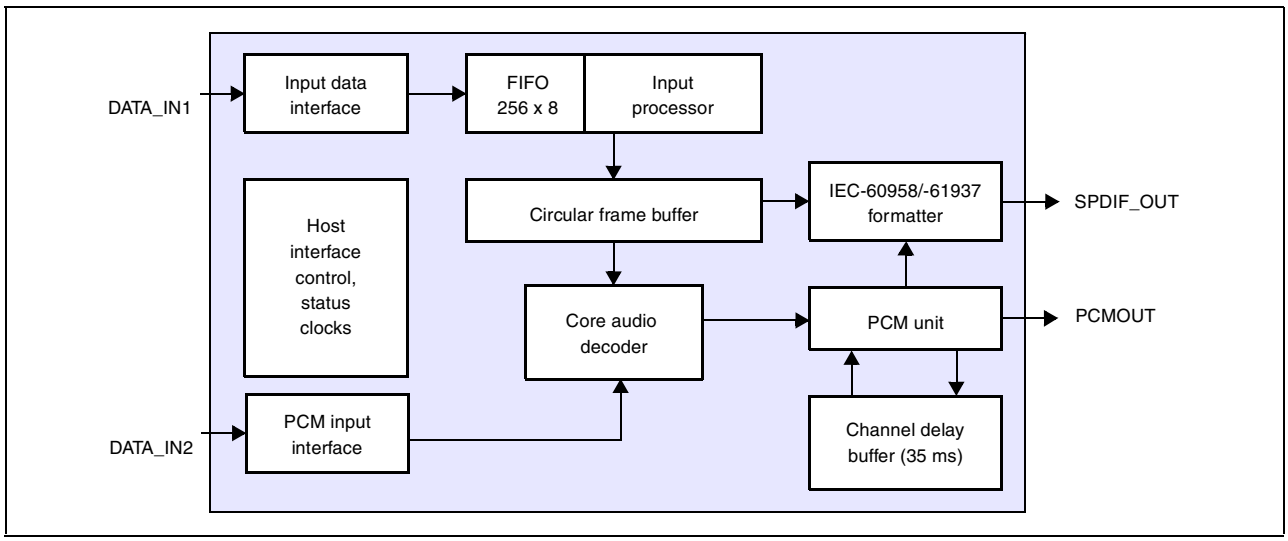
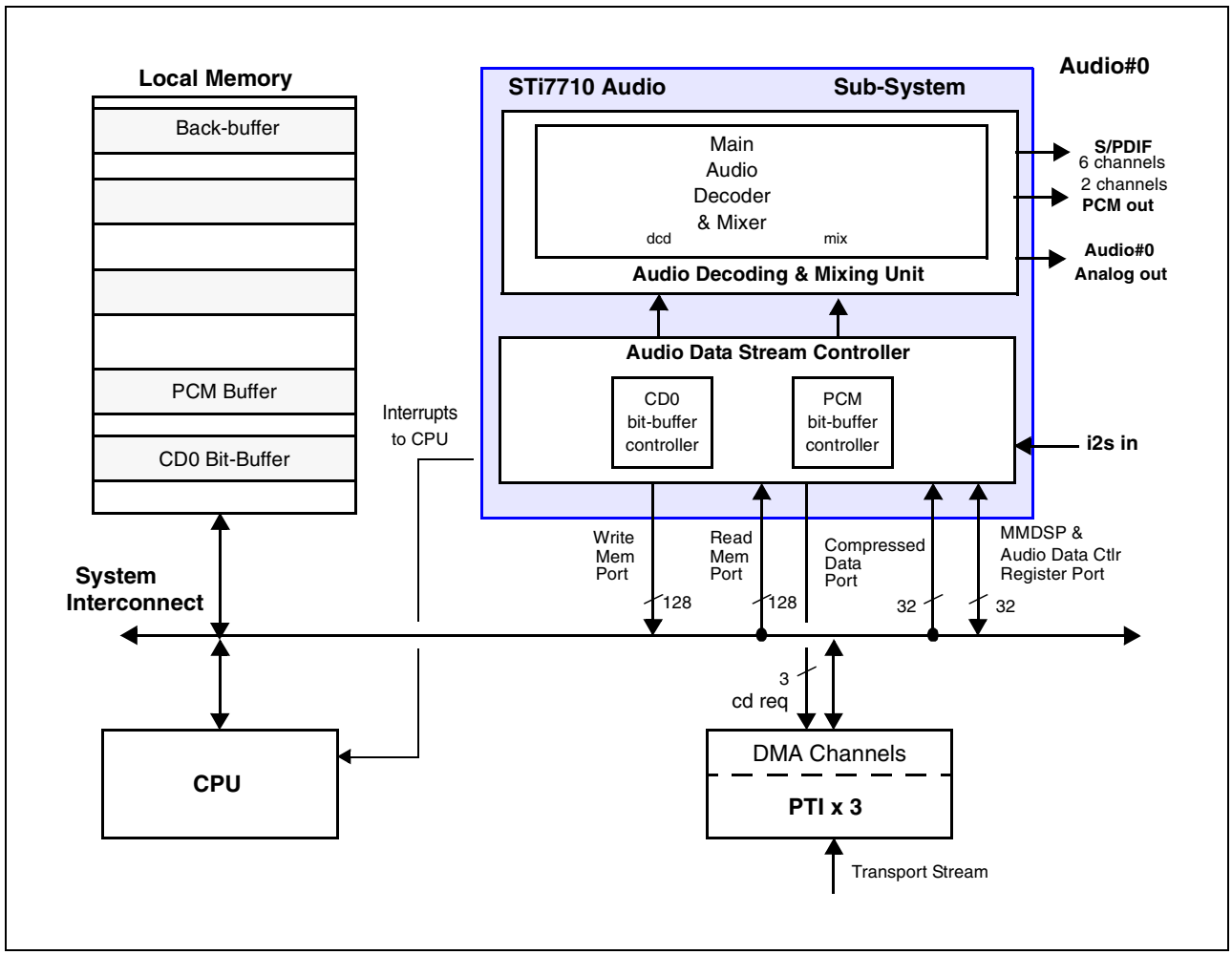


Figure 256 illustrates the audio decoder data flow. The compressed bit stream is transferred from the audio bit buffer (which is mapped into external SDRAM memory) to the audio decoder, via the MPEG DMA, which filters a 256 x 8 -byte FIFO. When the FIFO is filled, data is transmitted from the FIFO to the audio decoder.

1. The input processor (composed of a packet parser and an audio parser) unpacks the bit stream (packet parser) and verifies its syntax (audio parser).
2. The compressed audio frames with their associated information (PTS) are stored in the circular frame buffer.
3. While a second frame is stored in the circular frame buffer, the audio core decoder extracts and decodes the first frame into audio samples.
4. A PCM input captures incoming PCM samples and copies them to the main memory.
   A sample rate converter processes and mixes them to the left/right PCM output.

5. The PCM unit converts the samples to PCM format, and controls the channel delay buffer so that each channel can be delayed independently.
6. Simultaneously, the IEC unit transmits compressed or noncompressed data.
   - In compressed mode, data is extracted directly from the circular buffer and formatted according to the IEC-61937 standard.
   - In noncompressed mode, the IEC unit outputs the left and right PCM channels formatted by the PCM unit, according to the IEC-60958 standard.

Figure 257: Audio decoder block diagram



76.3 Decoding process

Decoding is performed in the following stages. The configuration registers can activate or bypass each stage:

Table 201: Audio decoding stages

Function	Description
Parsing	The input processor parses the bit stream. Parsing discards all of the nonaudio information so that only the elementary audio stream (Dolby Digital, MPEG-1 or MPEG-2, LPCM, PCM, DTS, MP3) is transmitted to the next stage (the circular frame buffer). The parsing stage operates in two phases: the packet parser unpacks the stream, the audio parser checks the syntax of the bit stream.

Table 201: Audio decoding stages

Function	Description
Main decoding	An elementary stream is input and decoded samples are output from this stage. Registers AUD_AC3_DOWNMIX, AUD_MP_DOWNMIX (one to six channels) define the number of output channels. Dolby Digital, MPEG-1 layers I and II, MPEG-2 layer II, AAC, LPCM and MP3 decoding formats are supported. The appropriate stream format must be set by registers AUD_STREAMSEL and AUD_DECODESEL before running the decoder.
Post decoding	Postdecoding includes specific PCM processing: DC filter, de-emphasis filter and downsampling filter. The DC and de-emphasis filters can be independently enabled or disabled by register AUD_PDEC, the downsampling filter by register AUD_DWSMODE. Postdecoding also provides a Pro Logic decoder, described in Section 76.7.5: *Pro Logic® decoding modes on page 861*. The decoder output can also be processed according to SRS Labs TruSurround algorithm. A specific Pro Logic compatible downmix of the main six channels can be applied to VCR outputs and data from second input can be mixed to main left/right output.
Bass redirection	This stage redirects low frequency signals to the subwoofer. The subwoofer is extracted from channels L, R, C, Ls, Rs, LFE. There are six configurations for subwoofer channel extraction, these are set by register AUD_OCFG.
Volume control	The volume is controlled independently for each channel in steps of 0.5 dB, by registers AUD_CHAN_IDX, AUD_VOLUME0 and AUD_VOLUME1.

76.4 Operation

76.4.1 Reset

Software resets the audio decoder by the following procedure.

1. Soft reset: 1 is written to register AUD_SOFTRESET to reset the audio decoder.
2. The interrupt registers (AUD_INTE, AUD_INT, AUD_ERROR) and command registers (AUD_SOFTRESET, AUD_RUN, AUD_PLAY, AUD_MUTE, AUD_SKIP_MUTE_CMD, AUD_SKIP_MUTE_VAL) are reset to zero.
3. The volume registers are reset to 0, no other decoding configurations are changed. The DSP returns to idle mode.

76.4.2 Clocks

The following clocks are used by the audio decoder:

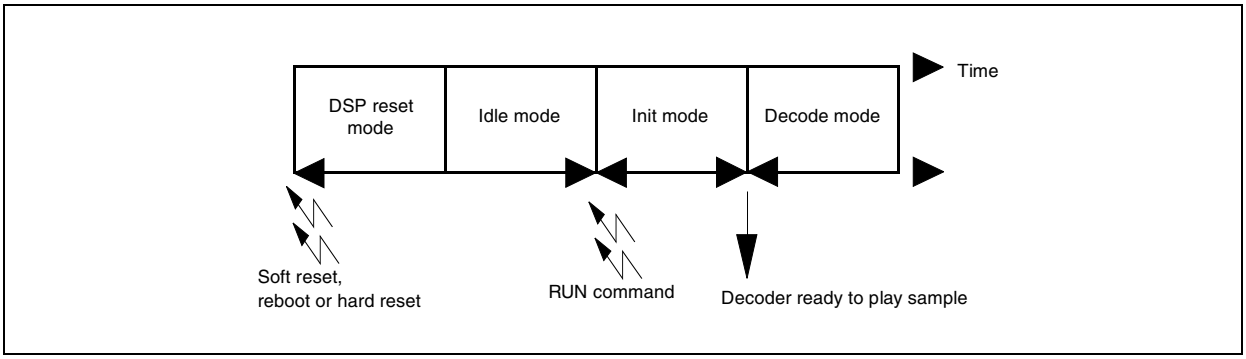
Table 202: Audio decoder clocks

Clock	Description
PCM	(PCMCLK signal) used by the external DACs to convert PCMOUT1, 2, 3. An embedded PLL from the 27 MHz clock input usually generates the signal. If necessary an external PLL can also generate it. The internal frequency synthesizer can generate 256 x F_s or 384 x F_s, where F_s = 12 (MP3 only), 32, 44.1, 48, 96, 16, 22.05 or 24 kHz.
Audio system PLL	The system PLL creates the audio system clock from the 27 MHz input clock.
Bit (SCLK)	The PCM serial clock is the bit clock. It provides clocks for each time slot (16 cycles for each channel in 16-bit mode, 32 cycles for each channel in 18-, 20-, 24-bit modes). The frequency of SCLK is, therefore, fixed to 2 x Nb time slots x F_s, where F_s is the sample frequency. The clock is derived from PCMCLK. Register AUD_PCMDIVIDER must be configured according to the selected output precision and the frequency of PCMCLK, so that the device can construct SCLK: FSCLK = FPCMCLK/{2 x (AUD_PCMDIVIDER + 1)} giving: AUD_PCMDIVIDER = FPCMCLK/(2 X FSCLK) - 1
Word (LRCLK)	The frequency of LRCLK is given by: FLRCLK = FSCLK/32; for 16-bit PCM output, FLRCLK = FSCLK/64; for 18-, 20- or 24-bit PCM output. No special configuration is required. Bit INV in register AUD_PCMCONF changes the polarity.

76.5 Decoding states

There are two decoding modes: idle and decode (see Figure 258). Register AUD_RUN changes the mode.

Figure 258: Decoding states



76.5.1 Reset mode

After a hardware or software reset, the DSP software resets itself then enters the idle mode.

76.5.2 Idle mode

In this mode, the embedded DSP does not decode, that is, no data is processed; the chip is waiting for the **run** command. All configuration registers must be initialized during this mode. In idle mode, even if the chip is not processing data, the DAC clocks can be output, enabling the setup of the external DACs. Once the PCMCLK, SCLK and LRCLK clocks are configured, they can be output by setting register AUD_MUTE.

Table 203: Idle mode, play and mute command effects

PLAY	MUTE	Clock (SCLK, LRCLK) state	PCM output
X	0	Not running if PLAY is set to 0 after reset.[a]	0
X	1	Running	0

a. The **play** command has no effect in this state, as the decoder is not running. It can, however, be issued and may be executed as soon as the decoder enters the decode mode.

76.5.3 Decode mode

This state is entered after the **run** command has been sent, that is, when register AUD_RUN = 1. In this mode, data is processed; the decoder can play sound, or mute the outputs by using registers AUD_PLAY and AUD_MUTE.

To decode and output streams, register AUD_PLAY must be set. If register AUD_MUTE is reset, sound is sent to outputs; if register AUD_MUTE is set, outputs are muted.

Table 204: Decode mode, play and mute command effects[a]

AUD_PLAY	AUD_MUTE	Clock state	PCM output	Decoding
0	0	Not running	0	No
0	1	Running	0	No
1	0	Running	Decoded samples	Yes
1	1	Running	0	Yes

a. Configuration registers cannot be changed in this state, the chip must be soft reset beforehand. Only the following registers can be changed on the fly: AUD_CHAN_IDX, AUD_VOLUME0, AUD_VOLUME1, AUD_OCFG, AUD_AC3_DOWNMIX, AUD_MP_DOWNMIX.

76.6 Stream parsers

The synchronization status of both parsers is provided in register AUD_SYNC_STATUS. Each time the synchronization status of one of the two parsers changes, the interrupt SYN is generated (if enabled) and the status can be read in AUD_SYNC_STATUS.

76.6.1 Packet parser

The packet parser unpacks the stream, sorts packets and transmits data to the audio parser. Before unpacking packets and transmitting data, the packet parser must detect the packet start by recognizing the packet synchronization word.

The parser can be set to search for two packet synchronization words before starting to unpack and transmit, by setting register AUD_PACKET_LOCK to 1. Otherwise, the packet parser starts handling the stream once it has detected information matching the packet synchronization word.

The packet parser is also able to perform selective decoding. It can decode audio packets that match a specified ID. This ID is specified in registers AUD_ID and AUD_ID_EXT; the function is enabled by setting register AUD_ID_EN.

76.6.2 Audio parser

The audio parser verifies the stream syntax, and separates audio from nonaudio data, sending audio data to the frame buffer. The parser must detect the audio sync word corresponding to the type of stream to be decoded.

The parser can be set to detect more than one synchronization word before parsing, by setting register AUD_SYNC_LOCK to a value between 1 and 3. This number represents the number of supplementary sync words to detect before being synchronized.

76.7 Decoding modes

All audio decoder supported configurations are described below.

Table 205: Audio decoder supported configurations

decoder	serial input 1 encoded	2 channel serial output	Post Processing			Max MIPS
MPEG AAC	stereo	stereo copy or TS-XT/mixing	TS-XT or Pcmmixing	*	skip, mute, pause, volume control	80
	6 channels	stereo copy or downmix or TS-XT/ mixing	downmix /TS-XT or Pcmmixing	*	skip, mute, pause, volume control	86
AC3	stereo OR Pro Logic encoded	stereo copy or TS-XT (+mixing)	Dolby Pro Logic +TS-XT	PCM mixing	skip, mute, pause, volume control	86
	6 channels	stereo copy or downmix or TS-XT (+mixing)	downmix /TS-XT	PCM mixing	skip, mute, pause, volume control	81
MPEG1/ MPEG2 2 channels	stereo OR Pro Logic encoded	stereo copy or TS-XT (+mixing)	TS-XT	de-emphasis + PCM mixing	skip, mute, pause, volume control	60
MPEG2	6 channels	stereo copy or downmix or TS-XT (+mixing)	downmix /TS-XT	de-emphasis + PCM mixing	skip, mute, pause, volume control	82
MP3	stereo OR Pro Logic encoded	stereo copy or TS-XT (+mixing)	TS-XT	de-emphasis+ PCM mixing	skip, mute, pause, volume control	80
LPCM video or PCM	stereo OR Pro Logic encoded	stereo copy or TS-XT (+mixing) (+downsampling 96->48)	TS-XT	(de-emphasis OR downsampling 96->48) + PCM mixing	skip, mute, pause, volume control	73
	6 channels	stereo copy or downmix (+downsampling 96->48) (+mixing)	downmix	(de-emphasis OR downsampling 96->48) + PCM mixing	skip, mute, pause, volume control	74
	6 channels	stereo copy or downmix or TS-XT (+mixing)	TS-XT	de-emphasis+ PCM mixing	skip, mute, pause, volume control	46
Beep tone		stereo copy (+mixing)	-	PCM mixing	-	16
Pink Noise	-	-	-	-	-	10
DTS Bypass	-	-	-	-	-	19
Null Decoder		stereo copy (+mixing)	-	PCM mixing	-	~14
pause/mute	-	second input only	-	PCM mixing	-	~14

76.7.1 Dolby Digital decoding modes

The decoder must be programmed to decode a Dolby Digital encoded audio bitstream by setting register AUD_DECODESEL to 0.

The following modes refer to different implementations of the dialog normalization and dynamic range control features. The mode is selected by programming register AUD_AC3_COMP_MOD.

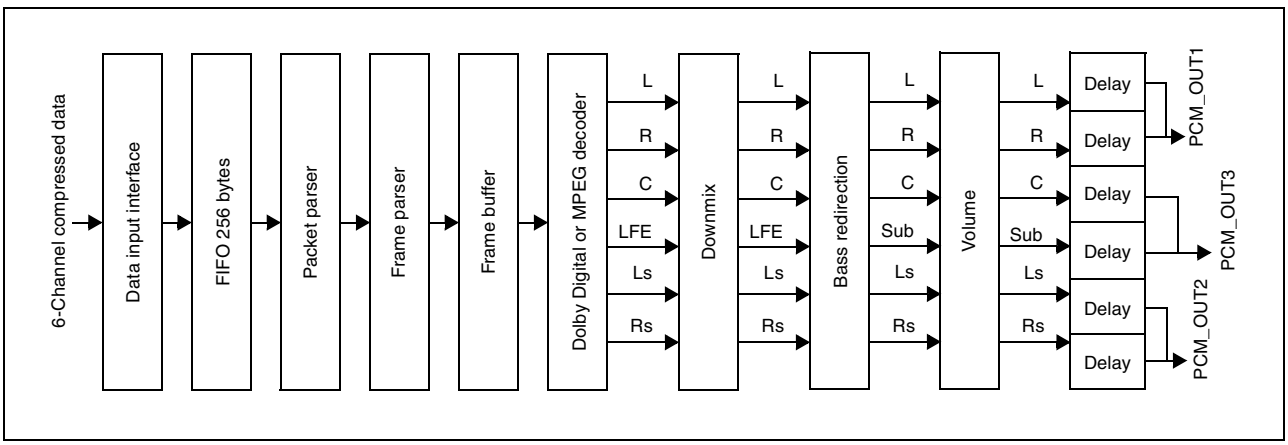
Table 206: Decoding modes

Type	Description
Line	In line mode (AUD_AC3_COMP_MOD = 2), dialog normalization is always enabled. This is done by the decoder itself and dialog is reproduced at a constant level. The two scaling registers AUD_AC3_HDR (for high level cut compression) and AUD_AC3_LDR (for low level boost compression) use and scale the dynamic range control variable encoded in the bit stream. For 2 front/0 rear downmix, the high level cut compression is not scalable.
RF	In RF mode (AUD_AC3_COMP_MOD = 3), the decoder performs dialog normalization. Dialog is reproduced at a constant level. The dynamic range control and heavy compression variables encoded in the bit stream are used, but compression scaling is not allowed. This means that registers AUD_AC3_HDR and AUD_AC3_LDR cannot be used in this mode. An 11 dB gain shift is applied on the output channels.
Custom 0	In custom 0 mode (AUD_AC3_COMP_MOD = 0), the decoder does not perform dialog normalization; this must be done by external circuitry. The two scaling registers AUD_AC3_HDR (for high level cut compression) and AUD_AC3_LDR (for low level boost compression) use and scale the dynamic range control variable encoded in the bit stream.
Custom 1	In custom 1 mode (AUD_AC3_COMP_MOD = 1), the decoder performs dialog normalization. The two scaling registers AUD_AC3_HDR (for high level cut compression) and AUD_AC3_LDR (for low level boost compression) use and scale the dynamic range control variable encoded in the bit stream.

76.7.2 MPEG decoding modes

MPEG-1 layer I and layer II encoded data are decoded, as well as MPEG-2 layer II data with or without extension (that is, 6-channel streams). The MPEG input format must be specified in register AUD_DECODESEL, where AUD_DECODESEL is 1 for MPEG-1 and 2 for MPEG-2. The dataflow is show in Figure 259.

Figure 259: 6-channel compressed data decoding flow



76.7.3 Dual mode decoding modes

In dual mode, two completely independent mono program channels (for example, bilingual) are encoded in the bit stream, referred to as channel 1 and channel 2. Register AUD_MP_DUALMODE in MPEG format, and register AUD_AC3_DUALMODE in Dolby Digital format set the left/right output to the following options:

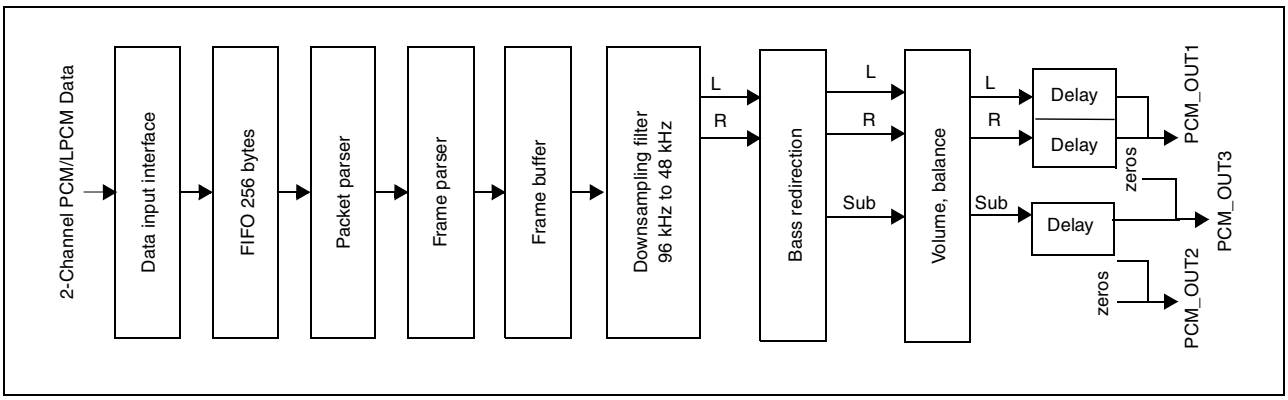
- output channel 1 on both L/R outputs,
- output channel 2 on both L/R outputs,
- mix channels 1 and 2 to monophonic and output on both L/R,
- output channel 1 on left output, and channel 2 on right output.

76.7.4 PCM decoding modes

The decoder supports PCM when register AUD_DECODESEL equals 3.

When decoding PCM streams encoded at 96 kHz, register AUD_DWSMODE configures the filter that downsamples the stream from 96 to 48 kHz.

Figure 260: PCM decoding flow



76.7.5 Pro Logic® decoding modes

Pro Logic compatible downmix

a multichannel bit stream can be decoded and downmixed to provide a 2-channel, Pro Logic-compatible output (Lt, Rt). Registers AUD_AC3_DOWNMIX and AUD_MP_DOWNMIX select this downmix. The two channels can be used as the input of a Pro Logic decoder and player (for example, home theatre).

Pro Logic decoding

a 2-channel Pro Logic bit stream can be decoded. The two channels could come from a Dolby Digital 2-channel bit stream, a LPCM or an MPEG-1 bit stream. The 2-channel bit stream can be converted into a four-channel output (L, R, C, S). The surround (S) is simultaneously sent on Ls and Rs channels. A Pro Logic downmix enables the channels to output on PCM data to be configured. This is done through register AUD_PL_DWNX.

An autobalance feature is available and activated through register AUD_PL_AB. Register AUD_LSDLY (while resetting register AUD_RSDLY) configures the delay on the surround channel.

Bass redirection is performed after the Pro Logic decode. The same bass redirection configuration as those available in non-Pro Logic modes can be used, except that the surround

channels are not added to the bass redirection. In the case of Dolby Digital or MPEG, the Dolby Digital or MPEG stream can be decoded before the Pro Logic decode.

Figure 261: Dolby Digital and Pro Logic decoding flow

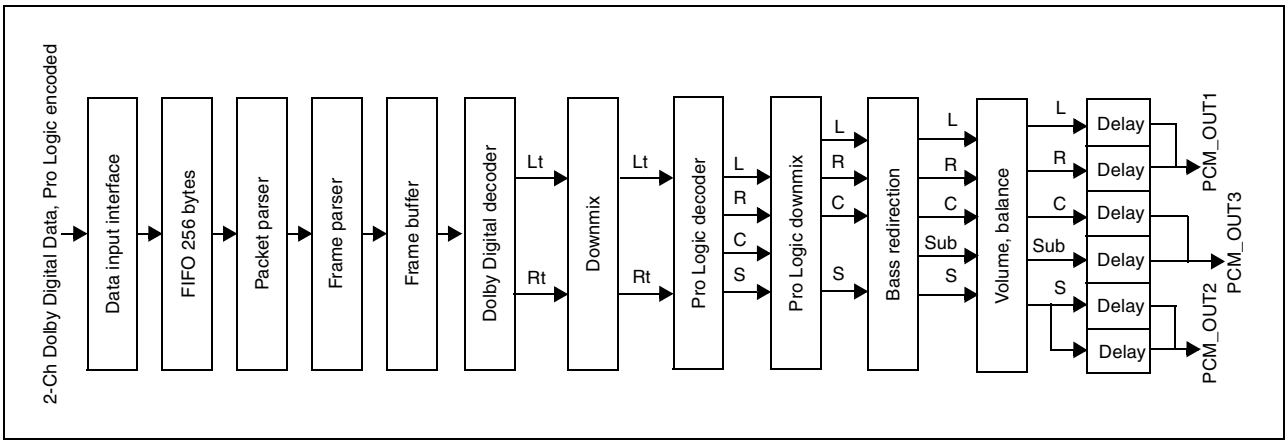


Figure 262: MPEG and Pro Logic decoding flow

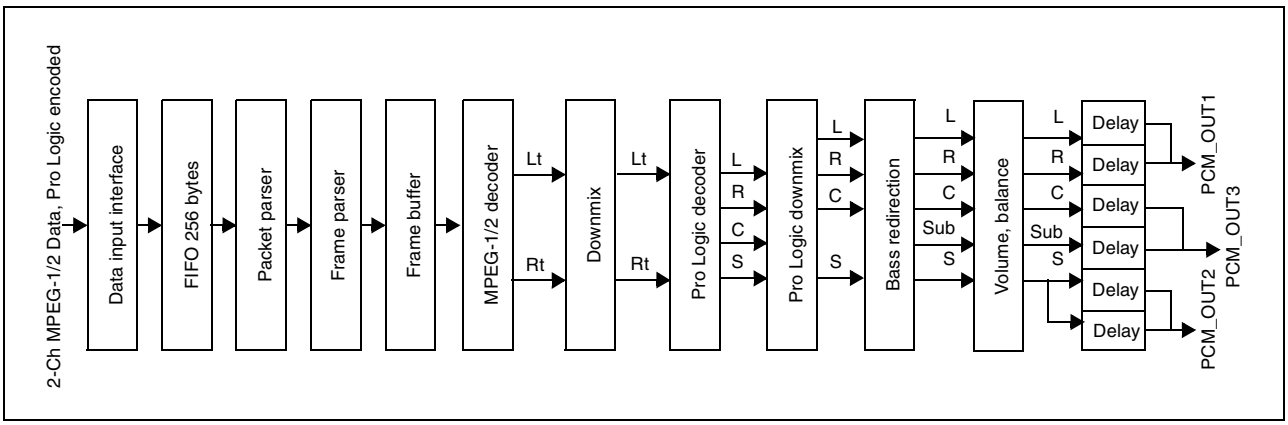
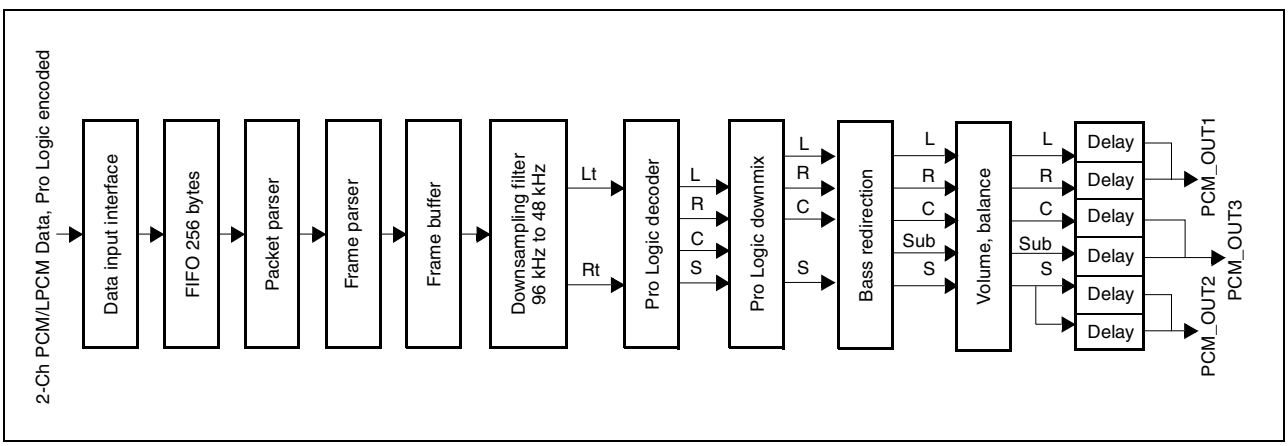


Figure 263: PCM and Pro Logic decoding flow



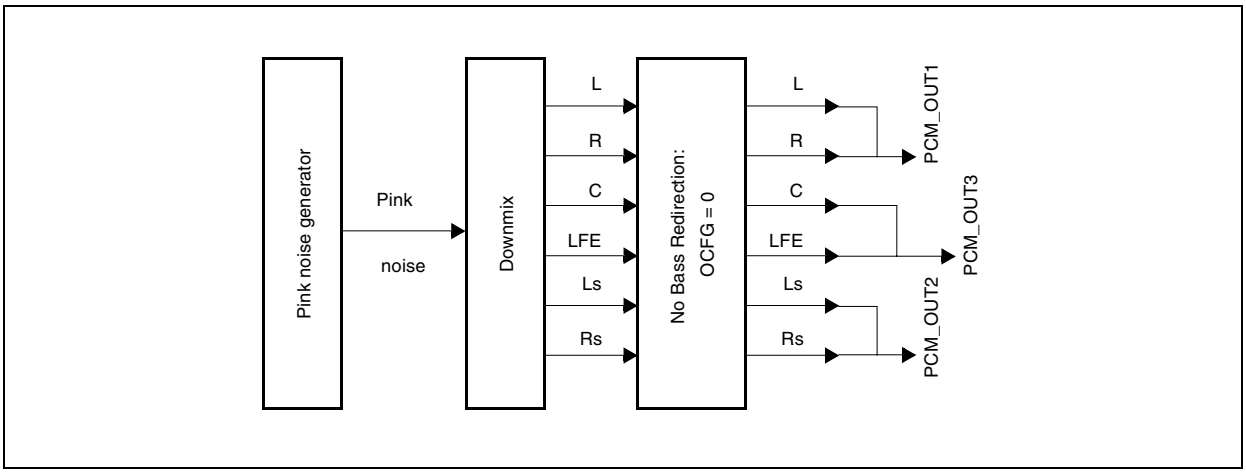
76.7.6 Pink noise decoding modes

The pink noise generator is used to position the speakers in the listening room for optimum sound quality.

The decoder is programmed to generate pink noise by writing the value 4 in register AUD_DECODESEL, and 3 in register AUD_STREAMSEL. Register AUD_PN_CHANNELCONF selects the pink noise output channels.

For pink noise generation, the register configuration should be: AUD_OCFG = 0 and volume control set to 0 dB.

Figure 264: Pink noise decoding flow



Note: The appropriate pink noise level is obtained by attenuating all outputs by 10 dB through volume registers.

76.7.7 MP3 decoding mode

MP3 supports the following frequencies: 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, and 48 kHz.

Volume control is possible in this mode if register AUD_OCFG is set to zero.

76.7.8 AAC decoding mode

The frequency of the second input data with AAC as main decoding should always be less than or equal to the main channel sampling frequency. When PCMmixing with AAC as the main decoding, upsampling of the data is not possible due to the limitations in the available RAM.

76.8 S/PDIF output

76.8.1 Overview of the S/PDIF output

The S/PDIF output pad is a TTL output pad with slew rate control. The DC output capability is 4 mA and the voltage drop is 3 V. This output must be connected to a TTL driver before being connected to a transformer.

The S/PDIF output supports IEC-60958 and IEC-61937 standards. The following registers must be initialized to configure the S/PDIF output.

- The category code must be entered in register AUD_SPDIF_CAT. This code is related to the type of application, and is specified in the digital output interface standard.
- The status bits to be transmitted on the S/PDIF output must be programmed in register AUD_SPDIF_STATUS.
- IEC clock setting must be specified in register AUD_SPDIF_CONF.
- Data type dependent information can be specified in register AUD_SPDIF_DTDI.
- The S/PDIF type is selected through register AUD_SPDIF_CMD. The IEC unit can output decoded data (PCM mode), encoded data, null data or pause bursts.

When configured in IEC-60958 mode, the S/PDIF output is used to transmit the decoded left and right channels. This is selected by choosing the PCM mode in register AUD_SPDIF_CMD and resetting the COM status bit in register AUD_SPDIF_STATUS.

When configured in IEC-61937 mode, the S/PDIF output is used to transmit encoded data taken directly from the frame buffer. This is selected by choosing the encoded mode (bit CMD) in register AUD_SPDIF_CMD and setting the bit COM in register AUD_SPDIF_STATUS. The decompressed data is output simultaneously on the PCM outputs, except in DTS format where only encoded data is transmitted.

When encoded S/PDIF data is output, a latency is inserted between the S/PDIF and PCM outputs. The PCM outputs are delayed with regard to the S/PDIF output. The latency is automatically set when bit LAT is 0 (register AUD_SPDIF_CONF). Standards define the value in autolatency mode. To control the latency manually, set LAT to 1 and define the latency in seconds by use of the register AUD_SPDIF_LATENCY.

When configured in muted mode (register AUD_SPDIF_CMD), the outputs are PCM null data. This can be used to synchronize the external IEC receiver. Register AUD_SKIP_MUTE_CMD bit MUTE is used to transmit bursts of pause frames in IEC-61937 format.

76.9 Interrupts

76.9.1 Interrupt register

The audio decoder contains a 16-bit interrupt register AUD_INT associated with a 16-bit enable register AUD_INTE. A bit set in AUD_INTE enables the corresponding interrupt. The interrupt associated with each bit is given in the AUD_INT description.

According to the type of interrupt, other information such as stream header, type of error detected or PTS value can be obtained by reading associated registers.

76.9.2 Error concealment

The audio core signals errors as interrupts. The core automatically handles most errors, but some require software action. Error categories are defined in register AUD_ERROR.

Dolby Digital decoding errors are signaled in AUD_ERROR but handled directly by the core. Software cannot change these errors. Dolby Digital decoding errors signal that something went wrong during decoding. The core soft mutes the frame and continues to decode.

MPEG decoding errors are signaled in AUD_ERROR but are handled directly by the core. Nothing can be done by the software. They signal that something went wrong during the decoding. The core soft mutes the frame and continues decoding. Only one error in this category indicates a programing error. If triggering MPEG_EXT_CRC_ERROR, bit MC (register AUD_MP_MC_OFF) must be set. This indicates that the decoder tried to decode more than two channels whereas the incoming stream contains only two channels.

Packet and audio synchronization errors are handled internally, and usually indicate that the incoming bit stream is incorrect or that it has been incorrectly input to the chip. In these cases, the decoder resets the corresponding parsing stage (packet or audio parser), then searches for the next correct frame.

The miscellaneous errors LATENCY_TOO_BIG indicates that the latency has been programmed greater than the maximum authorized value. The latency value should be changed or a switch made to autolatency mode. Other miscellaneous errors are handled internally.

76.10 Audio/video synchronization

76.10.1 Presentation time stamp detection

When enabled through register AUD_INTE, the interrupt PTS is generated when a PTS is present in the frame that is being output on PCMOUT (the interrupt is fired when the first decoded samples of the first block of the frame are output).

76.10.2 Pause frames capability

The number of audio blocks for the audio decoder to pause must be programmed in register AUD_SKIP_MUTE_VAL. Then bit BLK of AUD_SKIP_MUTE_CMD must be set. The audio decoder finishes decoding the current frame, soft mutes the next frame, and pauses for the number of blocks specified in AUD_SKIP_MUTE_VAL. When the pause is finished, decoding continues.

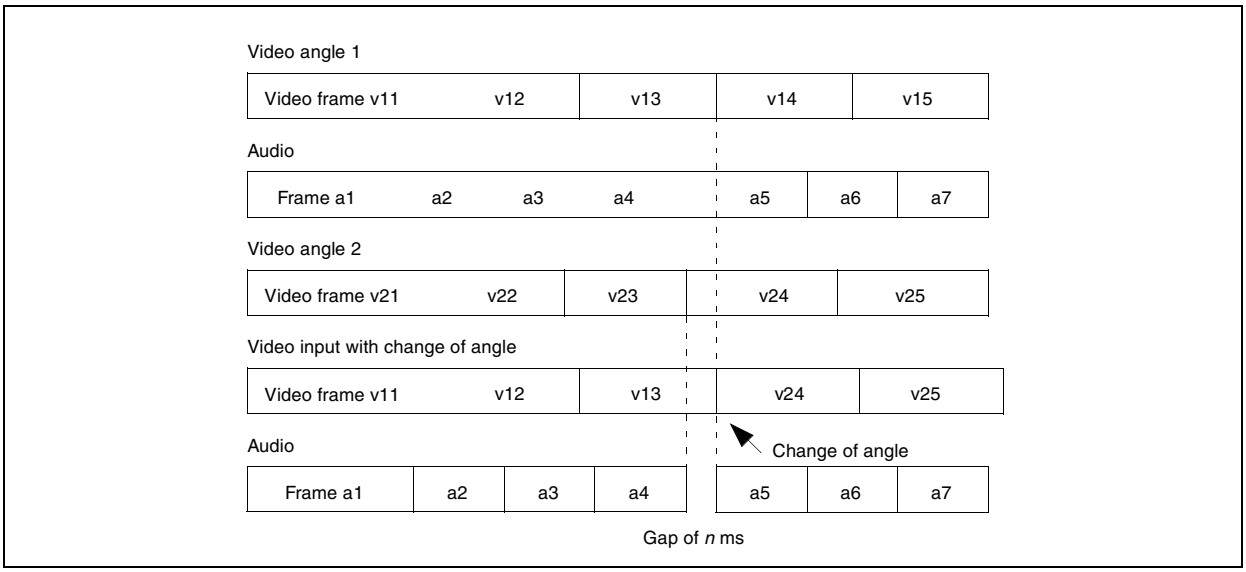
76.10.3 Skip frames capability

The number of frames to skip must be programmed in register AUD_SKIP_MUTE_VAL. Then bit SKP of AUD_SKIP_MUTE_CMD must be set. The audio decoder finishes decoding the current frame, soft mutes the next frame, and skips the number of frames specified in AUD_SKIP_MUTE_VAL. After skipping, it resumes decoding from the next incoming frame.

76.10.4 Pause burst capability

To synchronize video and audio outputs, the audio cell must be able to insert a pause on the output when required. This means that the audio decoder has to stop before decoding a new frame and the output of the audio has to be muted for a period of time as illustrated below.

Figure 265: Pause burst capability example



Register AUD_SKIP_MUTE_CMD initiates a pause.

- If bit PAU is set, a pause is inserted until bit PAU is reset.
- If bit BLK is set, a pause burst is inserted for a duration set by AUD_SKIP_MUTE_VAL.

The granularity of the gap defined by this mechanism is:

- 256 sampling periods for AC-3 (5.3 ms at 48 kHz, 5.8 ms at 44.1 kHz),
- 96 sampling periods for MPEG (2 ms at 48 kHz).

76.11 PCM beep tone

The PCM beep tone is a special mode used for set-top boxes. It generates a triangular signal of variable frequency and amplitude on the left and right channels.

76.11.1 Activating PCM beep tone mode

1. Reset the DSP.
2. Set up the registers AUD_DECODESEL = 7 and AUD_STREAMSEL = 3.
3. Restart the DSP by asserting registers AUD_RUN and AUD_PLAY.

76.11.2 Changing the frequency

Set register AUD_BEEP_FREQ according to the equation below:

Beep_tone frequency = $(F_s/2)/(register\_value + 1)$

76.11.3 Changing the amplitude

The amplitude of the PCM beep tone is 0 dB by default; to change the amplitude, set the registers below:

- AUD_OCFG = 0,
- AUD_CHAN_IDX = 0 (to select the channel pair [left and right]),
- AUD_VOLUME0 = attenuation value (step of -1 dB) on left channel,
- AUD_VOLUME1 = attenuation value (step of -1 dB) on right channel.

The PCM beep tone can be sent to the S/PDIF output when the S/PDIF output is configured in PCM mode.

76.12 PCM mixing

A second serial input (I2S_IN2), which inputs PCM data can be used for PCM mixing with PCMOUT1.

I2S_IN2 accepts the same formats as the main I²S input I2S_IN1 (left/right aligned, delayed or not, 16-, 18-, 20- or 24-bit data word length). Two channel data (synchronized with LRCLK according to the specified format) is stored in a bank of four 16-bit registers. An interrupt is generated each time a stereo sample is received. The DSP moves the received data into a double buffer structure located in the 24-bit RAM. When the buffer is full, the DSP stops requesting new data. AUDIO_REQ_2 is reset when the block sends an interrupt to the core. The software asserts AUDIO_REQ_2 at the end of the interrupt if the frame has not been fully received. When the DSP toggles the buffer, it sets AUDIO_REQ_2 to the next frame to be loaded.

The frame storage buffer is 1536 words wide. Size is defined from the largest frame size used in the decoder (256 for AC-3, 96 for MPEG, 80 or 160 for LPCM) taking into account the possible oversampling of the second input with respect to the main stream and the delta corresponds to a reservation in case of jitter or shift in sampling frequency between the two inputs.

To validate the PCM mixing configuration and **play** and **stop** commands, bit 0 of register AUD_PCMMIX_UPDATE must be set and the host must access (read or write) the generic AUD_PLAY register. This last action triggers an interrupt to the DSP core which then takes into account the new configuration and commands requested for the PCM mixing.

The polarity of AUDIO_REQ_2 is controlled through bit POL_2 of AUD_SIN_SETUP (when it is high, data must be input when AUDIO_REQ_2 = 0).

77 Audio decoder registers

Register addresses are provided as *AudioDecBaseAddress* + offset.

The *AudioDecBaseAddress* is:

0x3821 1000 (offset 0x1000 from *ADSCBaseAddress*).

The audio decoder contains 64 + 192 registers. Two types of registers exist:

- From offsets 0x00 to 0x3F, 64 real registers that can be initialized after reset,
- From offsets 0x40 to 0x13F, 192 memory locations.

  These 'virtual' registers can have different meanings and usage according to the mode in which the device operates.

  For example, in AC-3 mode, the register at address 0x69 is AUD_AC3_COMP_MOD, whereas in MPEG mode, it is AUD_MP_PROG_NUMBER.

  CAUTION: These 'virtual' registers cannot be hardware reset: they contain undefined values at reset and require initialization after each hardware reset.

In this document, only user registers are described. Undocumented registers are reserved and must never be accessed (neither in Read nor in Write mode). Read-only registers must never be written to.

The following table lists the register map by address and function; each audio decoder register is then described individually.

Table 207: Audio decoder register summary

Register function	Name	Offset	Type	Notes
Start up	AUD_BREAKPOINT	0xAC	WO	
	AUD_CLOCKCMD	0xE8	WO	
	AUD_INT_RAM	0x3FC	RO	
Version	AUD_VERSION	0x00	RO	
	AUD_IDENT	0x01	RO	
	AUD_SOFTVER	0x71	R/W	
RS232 activation	AUD_RS232_INTERF_ECHO	0xEF	R/W	
Setup and input	AUD_SIN_SETUP	0x0C	R/W	a, b
	AUD_CAN_SETUP	0x0D	R/W	a
PCM configuration	AUD_PCMDIVIDER	0x54	R/W	c
	AUD_PCMCONF	0x55	R/W	
	AUD_PCMCROSS	0x56	R/W	
	AUD_SFREQ	0x05	R/W	d

Table 207: Audio decoder register summary

Register function	Name	Offset	Type	Notes
ADC input/second input	AUD_ADCIN_PLAY	0xAD	R/W	
	AUD_SFREQ2	0x94	R/W	b
	AUD_ADCIN_MODE	0x95	R/W	
	AUD_ADCIN_CFG	0xB1	R/W	
	AUD_ADCIN_USERSETUP	0xAB	R/W	
	AUD_ADCIN_USERSETUP2	0xAC	R/W	
	AUD_ADCIN_L_VOL	0x88	R/W	
	AUD_ADCIN_R_VOL	0x89	R/W	
	AUD_ADCIN_SHIFT	0xFC	R/W	
PCM mixing	AUD_PCMMIX_UPDATE	0xB2	R/W	
	AUD_PCMMIX_FIRSTINPUT_VOLUME	0xB3	R/W	
	AUD_PCMMIX_MIX_COEFFICIENT	0xBA	R/W	
	AUD_PCMMIX_SRC_MSB	0xB8	R/W	
	AUD_PCMMIX_SRC_LSB	0xB9	R/W	
	AUD_PCMMIX_SRC_HANDSHAKE	0xB7	R/W	
	AUD_PCMMIX_ACK	0x16	R/W	
VCR configuration (PCMOUT0)	AUD_VCR_OUTPUT	0xAE	R/W	
	AUD_LVCR_DLY	0xAF	R/W	b
	AUD_RVCR_DLY	0xB0	R/W	
Channel delay setup	AUD_LDLY	0x57	R/W	e
	AUD_RDLY	0x58	R/W	
	AUD_CDLY	0x59	R/W	
	AUD_SUBDLY	0x5A	R/W	
	AUD_LSDLY	0x5B	R/W	
	AUD_RSDLY	0x5C	R/W	
	AUD_UPDATE	0x5D	R/W	d
S/PDIF output set-up	AUD_SPDIF_CMD	0x5E	R/W	f
	AUD_SPDIF_CAT	0x5F	R/W	d
	AUD_SPDIF_CONF	0x60	R/W	c
	AUD_SPDIF_STATUS	0x61	R/W	
	AUD_SPDIF_REP_TIME	0x75	R/W	
	AUD_SPDIF_LATENCY	0x7E	R/W	d
	AUD_SPDIF_DTDI	0x7F	R/W	

Table 207: Audio decoder register summary

Register function	Name	Offset	Type	Notes
Audio command	AUD_SOFTRESET	0x10	WO	a
	AUD_PLAY	0x13	R/W	
	AUD_MUTE	0x14	R/W	
	AUD_RUN	0x72	R/W	
	AUD_SKIP_MUTE_CMD	0x73	R/W	d
	AUD_SKIP_MUTE_VAL	0x74	R/W	
Interrupt	AUD_INTE	0x07, 08	R/W	a
	AUD_INT	0x09, 0A	RO	
Interrupt status	AUD_SYNC_STATUS	0x40	RO	
	AUD_ANCCOUNT	0x41	RO	
	AUD_HEAD4	0x42	RO	d
	AUD_HEAD3	0x43	RO	
	AUD_HEADLEN	0x44, 45	RO	
	AUD_PTS	0x46 - 0x4A	R/W	
	AUD_ERROR	0x0F	RO	
Decoding algorithm	AUD_DECODESEL	0x4D	R/W	f
	AUD_STREAMSEL	0x4C	R/W	
	AUD_DEC_SWITCH	0xFF	R/W	
System synchronization	AUD_PACKET_LOCK	0x4F	R/W	
	AUD_ID_EN	0x50	R/W	a
	AUD_ID	0x51	R/W	
	AUD_ID_EXT	0x52	R/W	
	AUD_SYNC_LOCK	0x53	R/W	f
Post decoding and Pro Logic	AUD_PDEC	0x62	R/W	c
	AUD_PL_AB	0x64	R/W	
	AUD_PL_DWNX	0x65	R/W	
	AUD_DWSMODE	0x70	R/W	

Table 207: Audio decoder register summary

Register function	Name	Offset	Type	Notes
Bass redirection	AUD_VOLUME0	0x4E	RWS	g
	AUD_VOLUME1	0x63	RWS	
	AUD_OCFG	0x66	R/W	c
	AUD_CHAN_IDX	0x67	R/W	
Dolby Digital configuration	AUD_AC3_DECODE_LFE	0x68	R/W	
	AUD_AC3_COMP_MOD	0x69	R/W	
	AUD_AC3_HDR	0x6A	R/W	
	AUD_AC3_LDR	0x6B	R/W	
	AUD_AC3_RPC	0x6C	R/W	
	AUD_AC3_KARAMODE	0x6D	R/W	
	AUD_AC3_DUALMODE	0x6E	R/W	
	AUD_AC3_DOWNMIX	0x6F	R/W	
	AUD_AC3_STATUS0	0x76	RO	d
	AUD_AC3_STATUS1	0x77	RO	
	AUD_AC3_STATUS2	0x78	RO	
	AUD_AC3_STATUS3	0x79	RO	
	AUD_AC3_STATUS4	0x7A	RO	
	AUD_AC3_STATUS5	0x7B	RO	
	AUD_AC3_STATUS6	0x7C	RO	
	AUD_AC3_STATUS7	0x7D	RO	
MPEG configuration	AUD_MP_SKIP_LFE	0x68	R/W	c
	AUD_MP_PROG_NUMBER	0x69	R/W	
	AUD_MP_DUALMODE	0x6E	R/W	
	AUD_MP_DRC	0x6A	R/W	
	AUD_MP_CRC_OFF	0x6C	R/W	
	AUD_MP_MC_OFF	0x6D	R/W	
	AUD_MP_DOWNMIX	0x6F	R/W	
	AUD_MP_STATUS0	0x76	RO	d
	AUD_MP_STATUS1	0x77	RO	
	AUD_MP_STATUS2	0x78	RO	
	AUD_MP_STATUS3	0x79	RO	
	AUD_MP_STATUS4	0x7A	RO	
	AUD_MP_STATUS5	0x7B	RO	

Table 207: Audio decoder register summary

Register function	Name	Offset	Type	Notes
AAC configuration	AUD_AAC_FORMAT	0x69	R/W	c
	AUD_AAC_MIX	0x6D	R/W	
	AUD_AAC_CRC_OFF	0x6C	R/W	
	AUD_AAC_STATUS0	0x76	RO	d
	AUD_AAC_STATUS1	0x77	RO	
	AUD_AAC_STATUS2	0x78	RO	
	AUD_AAC_STATUS3	0x79	RO	
	AUD_AAC_STATUS4	0x7A	RO	
	AUD_AAC_STATUS5	0x7B	RO	
MP3 configuration	AUD_CRC_OFF	0x6C	RO	d
	AUD_HEADROOM	0x6A	R/W	
LPCM configuration	AUD_DOWNSAMPLING	0x70	R/W	
	AUD_CHANNEL_ASGN	0xA8	R/W	
	AUD_MULTI_CHANNEL	0xA9	R/W	
	AUD_LPCM_DMIX	0x6F	R/W	
DTS status	AUD_DTS_1416BITS_MODE	0x69	R/W	
	AUD_DTS_STATUS0	0x76	RO	
	AUD_DTS_STATUS1	0x77	RO	
	AUD_DTS_STATUS2	0x78	RO	
	AUD_DTS_STATUS3	0x79	RO	
PCM beep-tone	AUD_BEEP_FREQ	0x68	R/W	c
	AUD_CHANNELCONF	0x69	R/W	
Pink noise	AUD_PN_CHANNELCONF	0x69	R/W	
De-emphasis	AUD_DEEMPH	0xB5	R/WS	d

Table 207: Audio decoder register summary

Register function	Name	Offset	Type	Notes
Downmix	AUD_DOWNMIX	0x6F	R/W	c
Dual mode	AUD_DUALMODE	0x6E	R/W	
TruSurround XT™ configuration	AUD_SRS_CALL - see AUD_PDEC	0x62	R/W	
	AUD_PL_AB	0x64	R/W	
	AUD_SRS_PFSPACE_GAIN	0xF4	R/W	
	AUD_SRS_PFCENTER_GAIN	0xF5	R/W	
	AUD_SRS_PRSPACE_GAIN	0xF6	R/W	
	AUD_SRS_PRCENTER_GAIN	0xF7	R/W	
	AUD_SELECT_CENTER	0xF8	R/W	
	AUD_SRS_INPUT_GAIN	0xF9	R/W	
	AUD_SRS_SHIFT	0xFB	R/W	
	AUD_TSXT_6DB_GAIN	0xF8	R/W	
SRS® focus configuration	AUD_POSTPROC - see AUD_ADCIN_PLAY	0xAD	R/W	
	AUD_FOCUS_ELEVATION	0xFA	R/W	
	AUD_FOCUS_BASS_COMP	0xFD	R/W	
SRS TruBass® configuration	AUD_POSTPROC - see AUD_ADCIN_PLAY	0xAD	R/W	
	AUD_TBASS_MODE	0x8A	R/W	
	AUD_TBASS_CONTROL	0x8B	R/W	

a. Register modification is **always** taken into account by the audio decoder. Any change to these registers is taken into account immediately.

b. Register created or modified for PCM-mixing functionality.

c. Register modification is taken into account **after every decoded data block** or **just after reset** (soft or hard). The decoded block is related to the granularity of the computation in the audio decoder software. A block is 256 samples in Dolby Digital, 96 samples in MPEG and 80 samples in LPCM/PCM.

d. Register modification is taken into account **after every data frame**. A frame is: 1152 samples in MPEG I/II, 1536 samples in Dolby Digital, 384 samples in MPEG-1 layer 1 and 80 samples in LPCM/PCM.

e. The delay registers are updated when bit 0 of the AUD_UPDATE register is set to 1.

f. Register modification is taken into account **only when the dsp is run after reset** (soft or hard).

g. The volume is updated when CHAN_IDX is set to the appropriate value.

77.1 Start up

AUD_BREAKPOINT Set breakpoints

7	6	5	4	3	2	1	0
AUD_BREAKPOINT							

Address: *AudioDecBaseAddress* + 0x00AC
Type: WO
Description: Audio breakpoint: set to 0x08 on start up. Also known as EMUL_BKCMD.

AUD_CLOCKCMD Start decoder clock

7	6	5	4	3	2	1	0
AUD_CLOCKCMD							

Address: *AudioDecBaseAddress* + 0x00E8
Type: WO
Description: Set to 0x00 to start the decoder clock. This allows the CPU clocks to be used by the audio cell. Also known as EMUL_CLKCMD.

AUD_INT_RAM Status of hardware registers

7	6	5	4	3	2	1	0
AUD_INT_RAM							

Address: *AudioDecBaseAddress* + 0x03FC
Type: RO
Description: Hardware registers status: when set to 1, indicates that the hardware registers are initialized and the audio decoder is ready.

77.2 Version

AUD_VERSION Audio hardware version

7	6	5	4	3	2	1	0
AUD_VERSION							

Address: *AudioDecBaseAddress* + 0x00
Type: RO
Description: This register gives the audio hardware version (binary decimal encoded).

AUD_IDENT Chip identity

7	6	5	4	3	2	1	0
AUD_IDENT							

Address: *AudioDecBaseAddress* + 0x01

Type: RO

Description: IDENT is a read-only register and is used to identify the IC on an application board. IDENT always has the value 0x72 for chips without AAC and 0x73 for chips with AAC.

AUD_SOFTVER Software version

7	6	5	4	3	2	1	0
AUD_SOFTVER							

Address: *AudioDecBaseAddress* + 0x71

Type: R/W

Reset:

Description: This register gives the version of micro-code which is running on the device. The actual version number is 0.4 (register value = 4 (0x04)).

77.3 RS232 activation

AUD_RS232_INTERF_ECHO Enable RS232 input

7	6	5	4	3	2	1	0
Reserved						RS232_ON	Reserved

Address: *AudioDecBaseAddress* + 0xEF

Type: R/W

Reset: 0

Description:

[7:2] **Reserved**

[1] **RS232_ON**
1: enable RS232 input.

[0] **Reserved**

77.4 Setup and input

AUD_SIN_SETUP Input data setup

7	6	5	4	3	2	1	0
Reserved				ADC_POL_REQ	I2S_POL_REQ	INMODE	

Address: *AudioDecBaseAddress* + 0x0C
Type: R/W
Reset: 0
Description:

[7:4] **Reserved**

[3] **ADC_POL_REQ:** polarity of the second serial input (ADCIN) request signal
1: data must be input when REQ is low.

[2] **I2S_POL_REQ:** polarity of the main input (I2Sin) request signal
1: data must be input when REQ is low.

[1:0] **INMODE**: serial data input interface mode
00: parallel input
01: serial input
10: reserved
11: serial input with LR clock

AUD_CAN_SETUP A/D converter setup

7	6	5	4	3	2	1	0
Reserved				16/32	SCLKPOL	LRPOL	I2SMODE

Address: *AudioDecBaseAddress* + 0x0D
Type: R/W
Reset: 0
Description: If HOST_sinsetup Imode is set to 11 (serial input with LR clock) these bit are used to configure the serial format:

[7:4] **Reserved**

[3] **16/32**
0: slot count is 32 but only 16 first are used

[2] **:SCLKPOL**
1: data sampled on falling edge of bit clock

[1] **LRPOL**
1: first channel (Left) when LRCLK = 1

[0] **I2Smode**
1: LRCLK is delayed by one cycle

77.5 PCM configuration

AUD_PCMDIVIDER Divider for PCM clock

7	6	5	4	3	2	1	0
PCMDIVIDER							

Address: *AudioDecBaseAddress* + 0x54

Type: R/W

Reset: Undefined

Description: The PCM divider must be set according to the formula below, where DAC_SCLK is the bit clock for the DAC. When Div is set to 0, DAC_SCLK is equal to DAC_PCMCLK:

Div = (DAC_PCMCLK/ (2 x DAC_SCLK)) -1

When the internal PLL is used, DAC_PCMCLK=384 x Fs or 256 x Fs.

If DAC_PCMCLK = 384 x fs, the formula becomes:

Div = (192 x Fs/DAC_SCLK) -1

If DAC_SCLK is 32 x Fs (common case with the 16 bit DAC), Div must be set to 5.

Table 208:

PCM divider value	Mode description	
	DAC_PCMCLK	DAC mode
5	384 Fs	16-bit
3	256 Fs	16-bit
2	384 Fs	32-bit
1	256 Fs	32-bit

AUD_PCMCONF PCM configuration

7	6	5	4	3	2	1	0
Reserved	ORD	DIF	INV	FOR	SCL	PREC[1:0]	

Address: *AudioDecBaseAddress* + 0x55
Type: R/W
Reset: Undefined
Description:

[7] **Reserved**

[6] **ORD**
PCM Order: only significant in 16-bit mode. When set, LSB is sent first. When reset, MSB is sent first.

[5] **DIF**
0: Left padded.

[4] **INV**
0: Left channel is output when DAC_LRCLK is low (I^2S format).
1: Left channel is output when DAC_LRCLK is high (Sony format).

[3] **FOR**:
0: I^2S format
1: Sony format

[2] **SCL**:
When set, the polarity of DAC_ SCLK is inverted, the PCM outputs and DAC_ LRCLK are stable for the DACs on the falling edge of DAC_ SCLK.
When reset, PCM outputs and DAC_ LRCLK are stable on the rising edge of DAC_ SCLK.

[1:0] **PREC[1:0]**: PCM Precision
00: 16 bit mode (16 slots)
01: 18 bit mode (32 slots)
10: 20 bit mode (32 slots)
11: 24 bit mode (32 slots)

AUD_PCMCROSS Cross PCM channels

7	6	5	4	3	2	1	0
CVCR[1;0]		CLR[1;0]		CSW[1:0]		LRS[1:0]	

Address: *AudioDecBaseAddress* + 0x56
Type: R/W
Reset: Undefined
Description: The PCMCROSS register only acts if bit PFC of register AUD_SPDIF_DTDI is set.

[7:6] **CVCR[1:0]**: Cross Left and Right VCR channels. (PCMOUT0)
00: Left channel is mapped on the Left output, Right channel is mapped on the Right output.
01: Left channel is duplicated on both outputs.
10: Right channel is duplicated on both outputs.
11: Right channel and Left channel are toggled.

[5:4] **CLR[1:0]**: Cross Left and Right channels (PCMOUT1)

[3:2] **CSW[1:0]**: Cross Center and Subwoofer (PCMOUT2).

[1:0] **LRS[1:0]**: Cross Left and Right Surround (PCMOUT3).

AUD_SFREQ Sampling frequency

7	6	5	4	3	2	1	0
AUD_SFREQ							

Address: *AudioDecBaseAddress* + 0x05

Type: R/W

Reset: 0

Description: This status register holds the code of the current output sampling frequency. If the audio stream is encoded (Dolby Digital, MPEG) or packetized (DVD_LPCM), the sampling frequency is automatically read in the audio stream and written into this register by the audio DSP.

For PCM stream, this register is written to by the application. The value in AUD_SFREQ corresponds to the following frequencies:

Table 209: Sampling frequency

Value	Frequency (kHz)
0	48
1	44.1
2	32
3	-
4	96
5	88.2
6	64
7	-

Value	Frequency (kHz)
8	24
9	22.05
10	16
11	-
12	12
13	11.025
14	8
15	-

77.6 ADC input/second input

AUD_ADCIN_PLAY Second input processing switch

7	6	5	4	3	2	1	0
FOCUS	PLAY	Reserved				TRUBASS	Reserved

Address: *AudioDecBaseAddress* + 0xAD
Type: R/W
Reset: Undefined
Description: (also known as AUD_POSTPROC)

[7] **FOCUS: Enable focus**

[6] **PLAY**
0: Switch off input 2 processing (even if data is coming in).
1: Switch on input 2 processing

[5:2] **Reserved**

[1] **TRUBASS**: Enable TruBass

[0] **Reserved**

AUD_SFREQ2 Sampling frequency of second input

7	6	5	4	3	2	1	0
SFREQ2[7:0]							

Address: *AudioDecBaseAddress* + 0x94
Type: R/W
Reset: Undefined
Description:

Table 210:

Value	Frequency (kHz)
0	48
1	44.1
2	32
3	-
4	96
5	88.2
6	64
7	-

Value	Frequency (kHz)
8	24
9	22.05
10	16
11	-
12	12
13	11.025
14	8
15	-

Note: SFREQ2 is supported from 8 to 48 kHz.

AUD_ADCIN_MODE Configure the input mode of PCM data

7	6	5	4	3	2	1	0
CHAN_SWAP	Reserved	SKIP		PCM_INPUT_MODE[3:0]			

Address: *AudioDecBaseAddress* + 0x95
Type: R/W
Reset: Undefined
Description:

[7] **CHAN_SWAP**:
0: do not swap channels
1: swap Left/Right channels
Restrictions apply.

[6] **Reserved**

[5:4] **SKIP**
0: all the incoming samples are captured
1: skip one sample out of 2 on second input. AUD_SFREQ2 must be set to half the actual sampling frequency of the input line.
2: skip one sample out of 4 on second input. AUD_SFREQ2 must be set to a 1/4th of the actual sampling frequency of the input line.
This aims to test low Sampling Frequency second input using an S/PDIF link for instance.

[3:0] **PCM_INPUT_MODE[3:0]**
In decimal:
0: 16 slots
1: 16 slots, LSB first
2: 32 slots, Left aligned
3: 32 slots, Right aligned
4: 32 slots I^2S mode
5: 32 slots, sign extended
6: 32 slots, 8-bits data
7: 32 slots, 16-bits data
8: 32 slots, 8-bits data, mono
9: user setup mode. see AUD_ADCIN_USERSETUP and AUD_ADCIN_USERSETUP2 registers

AUD_ADCIN_CFG Configure second input hardware processing

7	6	5	4	3	2	1	0
PREC2[1:0]		CHK_SYNC2	LRCLK2_VAL	INV_LRCLK2	SCLK2_EDGE	MSBFIRST	I2SNOTSONY

Address: *AudioDecBaseAddress* + 0xB1

Type: R/W

Reset: Undefined

Description: This register is used for configuring the second input hardware processing.

[7:6] **PREC2[1:0]**:
00: 16 bits
01: 18 bits
10: 20 bits
11: 24 bits

[5] **CHK_SYNC2**
1: the capture of incoming data begin on second LRCLK2 cycle. This ensures good synchronization
0: the capture begin on first LRCLK2 cycle

[4] **LRCLK2_VAL**: LRCLK2 START VALUE:
if (LRCLK xor INV_LRCLK2) = LRCLK2_Start_value then sample is output on LEFT channel
else (LRCLK xor INV_LRCLK2) = not(LRCLK2_Start_value) then sample is output on RIGHT channel.

[3] **INV_LRCLK2**:
1: LR CLK input signal is inverted

[2] **SCLK2_EDGE:**
0: SCLKIN2 strobes DATA2 on falling edge; 1: on rising edge

[1] **MSBFIRST**:
1: MSB arrives first; 0: LSB arrives first

[0] **I2SNOTSONY**: Input Standard

AUD_ADCIN_USERSETUP Size of second input data

7	6	5	4	3	2	1	0
Reserved			DATA_SIZE				

Address: *AudioDecBaseAddress* + 0xAB

Type: R/W

Reset: Undefined

Description: This register contains the size of the second input data minus one. For example, 0x17 means that one LRCLK input edge contains 24 bits of data.

AUD_ADCIN_USERSETUP2 Second input data position

7	6	5	4	3	2	1	0
Reserved			DATA_POSITION				

Address: *AudioDecBaseAddress* + 0xAC

Type: R/W

Reset: Undefined

Description: This register contains the position of the second input data that is to say the number of bits to shift to find the LSB of each data.

AUD_ADCIN_L_VOL Second input left attenuation

7	6	5	4	3	2	1	0
AUD_ADCIN_L_VOL							

Address: *AudioDecBaseAddress* + 0x88
Type: R/W
Reset: Undefined
Description: This register contains the scaling factor applied to the Left channel of the second input. This value varies from 0 to 1.0.

AUD_ADCIN_R_VOL Second input right attenuation

7	6	5	4	3	2	1	0
AUD_ADCIN_R_VOL							

Address: *AudioDecBaseAddress* + 0x89
Type: R/W
Reset: Undefined
Description: This register contains the scaling factor applied to the Right channel of the second input. This value varies from 0 to 1.0.

AUD_ADCIN_SHIFT Second input level shift

7	6	5	4	3	2	1	0
AUD_ADCIN_SHIFT							

Address: *AudioDecBaseAddress* + 0xFC
Type: R/W
Reset: Undefined
Description: This register contains the shift value applied on the 2 channels of the second input.
The final attenuation on the second input is:
If AUD_ADCIN_SHIFT > 0,
Left level = (original Left level * AUD_ADCIN_LEFT_VOL/256) << AUD_ADCIN_SHIFT
Right level = (original Right level * AUD_ADCIN_RIGHT_VOL/256) << AUD_ADCIN_SHIFT
If AUD_ADCIN_SHIFT < 0,
Left level = (original Left level * AUD_ADCIN_LEFT_VOL/256) >> abs(AUD_ADCIN_SHIFT)
Right level = (original Right level * AUD_ADCIN_RIGHT_VOL/256) >> abs(AUD_ADCIN_SHIFT)

77.7 PCM mixing

AUD_PCMMIX_UPDATE PCM mixing configuration

7	6	5	4	3	2	1	0
PRE_VOLUME	ADCIN_PRE_VOLUME	L/R_COPY_ON_VCR	MIX_2ND_INPUT		USE_SRCT	LOAD_SRCT	ADC_CPY

Address: *AudioDecBaseAddress* + 0xB2

Type: Type: R/W

Reset: HW: Undefined / SW: N/A

Description:

[7] **PRE_VOLUME**
1: a pre_volume is applied on first input with the coefficient set in AUD_PCMMIX_FIRSTINPUT_VOLUME.
0: no pre_volume
Note: If bits 3 and 5 of AUD_MIX_UPDATE are simultaneously set, the "MIX copy on VCR" action (bit 5) has priority over the "2nd input on VCR" (bit 3).

[6] **ADCIN_PRE_VOLUME**
1: a pre_volume is applied on second input with the coefficients set in AUD_ADCIN_L_VOL, AUD_ADCIN_R_VOL and AUD_ADCIN_SHIFT.
0: no pre_volume on second input

[5] **L/R_COPY_ON_VCR**
1: copy L/R channels (PCMOUT1) on VCR channels (PCMOUT0). This is eventually done after mixing has been done between second input and PCMOUT1. (see M IX bit)
0: no copy on VCR

[4:3] **MIX_2ND_INPUT**
if AUD_PCMMIX_MIX_COEFFICIENT is not 0:
00: no mix
01: mix 2nd input with LR; L/R = alpha In2 + (1-alpha) L/R
10: mix 2nd input with VCR; VCR = alpha In2 + (1-alpha) VCR
11: mix 2nd input with LR and VCR; L/R = alpha In2 + (1-alpha) L/R
VCR = alpha In2 + (1-alpha) VCR

[2] **USE_SRCT**
1: force DSP to USE downloaded table for SRC, instead of ROMed table.

[1] **LOAD_SRCT**
1: initiate the download of a new USER coefficient table for SRC

[0] **ADC_CPY**
1: copy the 2nd input after SRC on VCR output

AUD_PCMMIX_FIRSTINPUT_VOLUME A volume can be applied on main channels

7	6	5	4	3	2	1	0
NB_CHANS	VOLUME						

Address: *AudioDecBaseAddress* + 0xB3
Type: R/W
Reset: Undefined
Description:

[7] **NB_CHANS**
0: volume is applied on L/R only.
1: volume is applied on the 6 main channels

[6:0] **VOLUME**: volume attenuation by 1 dB step from +12 to - 64 dB.
0: +12 dB
12: 0 dB
76: -64 dB

AUD_PCMMIX_MIX_COEFFICIENT Mixing Level of 2nd input

7	6	5	4	3	2	1	0
MIX_LEV[7:0]							

Address: *AudioDecBaseAddress* + 0xBA
Type: R/W
Reset: Undefined
Description: This register stores the level of the second input relative to L/R outputs (PCMOUT1) when activating AUD_PCMMIX_UPDATE.MIX.

The mixer outputs:

PCMOUT1 = (1-mix_level) * PCMOUT1 + mix_level * second_input

with mix_level = MIX_LEV / 255

AUD_PCMMIX_SRC_MSB Input of the MSByte of a new SRC coefficient

7	6	5	4	3	2	1	0
SRC_COEFF_MSB[7:0]							

Address: *AudioDecBaseAddress* + 0xB8
Type: R/W
Reset: Undefined
Description: See AUD_PCMMIX_ACK and AUD_PCMMIX_SRC_HANDSHAKE registers

AUD_PCMMIX_SRC_LSB Input of the LSByte of a new SRC coefficient

7	6	5	4	3	2	1	0
SRC_COEFF_LSB[7:0]							

Address: *AudioDecBaseAddress* + 0xB9
Type: R/W
Reset: Undefined
Description: See AUD_PCMMIX_ACK and AUD_PCMMIX_SRC_HANDSHAKE registers.

AUD_PCMMIX_SRC_HANDSHAKE Allows the input of a new SRC coefficient

7	6	5	4	3	2	1	0
HSHAKE[7:0]							

Address: *AudioDecBaseAddress* + 0xB7
Type: R/W
Reset: Undefined
Description: See AUD_PCMMIX_ACK, AUD_PCMMIX_SRC_MSB and AUD_PCMMIX_SRC_LSB registers.

USER must Write 0x01 when coefficient in AUD_PCMMIX_SRC_MSB and AUD_PCMMIX_SRC_LSB are ready to be received by DSP

DSP Write 0x00 when coefficient in AUD_PCMMIX_SRC_MSB and AUD_PCMMIX_SRC_LSB have been read

AUD_PCMMIX_ACK Activates the dsp acquisition of a new SRC coefficient

7	6	5	4	3	2	1	0
Reserved							AUD_ACK

Address: *AudioDecBaseAddress* + 0x16
Type: R/W
Reset: 0
Description:

[7:1] **Reserved**

[0] **AUD_ACK**
USER must write 0x01 when AUD_PCMMIX_SRC_MSB and AUD_PCMMIX_SRC_LSB are loaded with a new SRC coefficient and when AUD_PCMMIX_SRC_HANDSHAKE is set to 1 in order to initiate the capture of this coefficient by the DSP. The DSP then resets AUD_PCMMIX_ACK and AUD_PCMMIX_SRC_HANDSHAKE registers. This bit generates an interrupt to the DSP core.

77.8 VCR configuration

AUD_VCR_OUTPUT Possible configurations for the VCR output

7	6	5	4	3	2	1	0
Reserved			STEREO	PRL	Reserved	COPY	Reserved

Address: *AudioDecBaseAddress* + 0xAE
Type: R/W
Reset: Undefined
Description:

[7:5] **Reserved**

[4] **STEREO**
1: a stereo 2/0 downmix is done and output on the VCR.

[3] **PRL**
1: a prologic downmix is done and output on the VCR.

[2] **Reserved**

[1] **COPY**
1: enables a hardware copy of 'Left/Right' channels to the 'VCR' channels.

[0] **Reserved**

AUD_LVCR_DLY Specifies delay on Left VCR output (PCMOUT0)

7	6	5	4	3	2	1	0
DELAY	Reserved						

Address: *AudioDecBaseAddress* + 0xAF
Type: R/W
Reset: Undefined
Description: *Delay on Left VCR channel, expressed in number of groups of 16 samples.*
LVCRDLY = delay (ms) * Fs (kHz) / 16

AUD_RVCR_DLY Specifies delay on Right VCR output (PCMOUT0)

7	6	5	4	3	2	1	0
DELAY	Reserved						

Address: *AudioDecBaseAddress* + 0xB0
Type: R/W
Reset: Undefined
Description: *Delay on Right VCR channel, expressed in number of group of 16 samples.*
RVCRDLY = delay (ms) * Fs (kHz) / 16

77.9 Channel delay setup

The unit for the register delay contents is a group of 16 samples. The maximum delay is 35 ms (so the sum of the delays on the six channels is limited to 35 ms).
For example, when the sampling frequency is 48 kHz,
the sum of all the delays must be less than 35 ms * 48 kHz /16 samples = 105 units (with one unit = 1 group of 16 samples).

Note: When only one surround channel is present (in Pro Logic or other mode), the Right Surround delay must be cleared, and the Left delay channel is used for both surround channels.

AUD_LDLY Left channel delay

7	6	5	4	3	2	1	0
AUD_LDLY							

Address: *AudioDecBaseAddress* + 0x57
Type: R/W
Reset: Undefined
Description: Delay on Left channel, expressed in number of group of 16 samples.
LDLY = delay (ms) * Fs (kHz) / 16

AUD_RDLY Right channel delay

7	6	5	4	3	2	1	0
AUD_RDLY							

Address: *AudioDecBaseAddress* + 0x58
Type: R/W
Reset: Undefined
Description: Delay on Right channel, expressed in number of group of 16 samples.
RDLY = delay (ms) * Fs (kHz) / 16

AUD_CDLY Centre channel delay

7	6	5	4	3	2	1	0
AUD_CDLY							

Address: *AudioDecBaseAddress* + 0x59
Type: R/W
Reset: Undefined
Description: Delay on Center channel, expressed in number of groups of 16 samples.
CDLY = delay (ms) * Fs (kHz) / 16

AUD_SUBDLY Subwoofer channel delay

7	6	5	4	3	2	1	0
AUD_SUBDLY							

Address: *AudioDecBaseAddress* + 0x5A
Type: R/W
Reset: Undefined
Description: Delay on Subwoofer channel, expressed in number of groups of 16 samples.
SUBDLY = delay (ms) * Fs (kHz) / 16

AUD_LSDLY Left Surround channel delay

7	6	5	4	3	2	1	0
AUD_LSDLY							

Address: *AudioDecBaseAddress* + 0x5B
Type: R/W
Reset: Undefined
Description: Delay on Left Surround channel, expressed in number of groups of 16 samples.
LSDLY = delay (ms) * Fs (kHz) / 16

AUD_RSDLY Right surround channel delay

7	6	5	4	3	2	1	0
AUD_RSDLY							

Address: *AudioDecBaseAddress* + 0x5C
Type: R/W
Reset: Undefined
Description: *Delay on Right Surround channel, expressed in number of group of 16 samples.*
RSDLY = delay (ms) * Fs (kHz) / 16.
Note: When only one surround channel is used, this register must be reset at initialization.

AUD_UPDATE PCM delay update

7	6	5	4	3	2	1	0
Reserved							DLY

Address: *AudioDecBaseAddress*+ 0x5D
Type: R/W
Reset: 0
Description:

[7:1] **Reserved**

[0] **DLY**
This bit must be set to 1 to force the DSP to update its delays values (read from the audio delay registers).
0: Delay values held in the audio delay registers are **not** updated in the DSP.
(the DSP will keep the delay values set previously).
1: The delay values held in the audio delay registers are updated in the DSP (the DSP uses the new values). This bit is automatically reset to zero after it the update has been carried out.

77.10 S/PDIF output set-up

AUD_SPDIF_CMD S/PDIF control

7	6	5	4	3	2	1	0
AUX	Reserved					CMD	

Address: *AudioDecBaseAddress* + 0x5E

Type: R/W

Reset: Undefined

Description: This register is the S/PDIF control register. Several modes are available, selected by value:

[7] **AUX**
0: PCMOUT1 (L/R) is transmitted to the S/PDIF out when S/PDIF is in PCM mode (see CMD bit)
1: PCMOUT0 (VCR) is transmitted to the S/PDIF out when S/PDIF is in PCM mode (see CMD bit)

[6:2] **Reserved**

[1:0] **CMD**
00: Off mode: The S/PDIF is not working, the output line is idle.
01: Muted mode: The outputs are PCM null data.
10: PCM mode: The outputs are PCM data. Only the PCMOUT0 or PCMOUT1 decoded channels are transmitted (see AUX bit)
11: Encoded: The compressed bitstream is transmitted (See IEC61937 standard)

AUD_SPDIF_CAT Category code

7	6	5	4	3	2	1	0
CATCODE							

Address: *AudioDecBaseAddress* + 0x5F
Type: R/W
Reset: Undefined
Description: The table below defines the category codes; values not listed are reserved.

Table 211: Category codes

Code	Description
0 0 0 0 0 0 0	General
1 0 0 0 0 0 0	Experimental
X X X 0 0 0 0	Reserved
X X X 1 0 0 0	Solid state memory
0 0 0 0 1 0 0	Broadcast reception of digital audio - Japan
1 1 0 0 1 0 0	Broadcast reception of digital audio - United states
0 0 0 1 1 0 0	Broadcast reception of digital audio - Europe
1 0 0 0 1 0 0	Broadcast reception of digital audio - electronic software delivery
X X X X 1 0 0	Broadcast reception of digital audio - All other states are reserved
0 0 0 0 0 1 0	Digital/digital converters and signal processing PCM encoder/decoder
0 1 0 0 0 1 0	Digital/digital converters and signal processing Digital sound sampler
0 0 1 0 0 1 0	Digital/digital converters and signal processing Digital signal mixer
0 0 1 1 0 1 0	Digital/digital converters and signal processing Sample rate converter
X X X X 0 1 0	Digital/digital converters and signal processing All other states are reserved
X X 0 0 1 1 0	A/D converter without copyright
X X 1 0 1 1 0	A/D converter with copyright (using copy and L bits)
X X X 1 1 1 0	Broadcast reception of dig. audio
0 0 0 0 0 0 1	Laser optical CD - Compatible with IEC 908
0 0 0 1 0 0 1	Laser optical CD - Not compatible with IEC 908 (Magneto optical)
X X X X 0 0 1	Laser optical All other states are reserved
0 0 0 0 1 0 1	Musical instruments, microphones... Synthesizer
0 0 0 1 1 0 1	Musical instruments, microphones... Microphone
X X X X 1 0 1	Musical instruments, microphones, etc. All other states are reserved
0 0 0 0 0 1 1	Magnetic tape or disks DAT
0 0 0 1 0 1 1	Magnetic tape or disks, digital audio sound, VCR
X X X X 0 1 1	Magnetic tape or disks; all other states are reserved
X X X X 1 1 1	Reserved
7	Only category codes XXXX100, XXX1110, XXXX001 -> L bit
0	Original, commercially pre-recorded data
1	No indication of first generation or higher
7	All other categories
0	No indication of first generation or higher
1	Original, commercially pre-recorded data

AUD_SPDIF_CONF S/PDIF PCMCLK divider

7	6	5	4	3	2	1	0
LAT	SM	RND	DIV[4:0]				

Address: *AudioDecBaseAddress* + 0x60

Type: R/W

Reset: Undefined

Description:

[7] **LAT**: Configure the latency mode between the S/PDIF output (in mode compressed) and the Audio output.
0: Auto-Latency:
For AC3, the auto-latency is the transmission time for 2/3 of the payload, plus the time to decode an audio block.
For MPEG auto-latency, the latency is the following time depending of the sampling frequency in the incoming bitstream: MPEG 48 kHz: 20.90ms, MPEG 44.1 kHz: 22.95 ms, MPEG 32 kHz: 32.53 ms.
1: User-programmable latency - the AUD_SPDIF_LATENCY register is used.

[6] **SM**: Sync mute mode, must be set to zero.

[5] **RND**: This bit is used to have “16-bit rounding” on the S/PDIF (when in PCM mode)
0: no rounding
1: rounding
This bit has no effect on the precision of PCMOUT0, 1, 2, 3 data

[4:0] **DIV[4:0]**: DAC_PCMCLK divider. Must be set according to the formula:
in 16 bit mode: IECDIV=(1+PCMDIV)/2-1;
in 32 bit mode: IECDIV=PCMDIV

The table below shows the relationship between the value of the IEC divider and the value of the PCM divider.

Table 212: IEC divider and PCM divider values

PCM divider value	Mode description	IEC divider value
5	DAC_PCMCLK = 384Fs, DAC is 16-bit mode	2
3	DAC_PCMLK = 256 Fs, DAC is 16-bit mode	1
2	DAC_PCMLK = 384 Fs, DAC is 32-bit mode	2
1	DAC_PCMLK = 256 Fs, DAC is 32-bit mode	1

AUD_SPDIF_STATUS S/PDIF status bit

7	6	5	4	3	2	1	0
Reserved	SFR				PRE	COPY	COM

Address: *AudioDecBaseAddress* + 0x61

Type: R/W

Reset: Undefined

Description: This register is used to set the value of the status bit in the IEC958 data stream.

[7] **Reserved**

[6:3] **SFR**
0000: if sampling frequency = 44.1 kHz
0010: if sampling frequency = 48 kHz
0011: if sampling frequency = 32 kHz
1010: if sampling frequency = 96 kHz

[2] **PRE**
1: output has pre-emphasis
0: output does not have pre-emphasis

[1] **COPY**
1: copy allowed
0: copy not allowed

[0] **COM: Co**mpress data bit.
1: compressed mode
0: noncompressed mode.

AUD_SPDIF_REP_TIME S/PDIF repetition time of a pause frame

7	6	5	4	3	2	1	0
AUD_SPDIF_REP_TIME							

Address: *AudioDecBaseAddress* + 0x75

Type: R/W

Reset: Undefined

Description: In compressed mode, a burst of pause frames is sent when there are no more data to transmit (due to an error or a gap in the incoming bitstream, for example). This register sets the size of a pause frame in IEC frames: Dolby Digital = 4, MPEG = 32.

AUD_SPDIF_LATENCY Latency value

7	6	5	4	3	2	1	0
LATENCY							

Address: *AudioDecBaseAddress* + 0x7E

Type: R/W

Reset: Undefined

Description: If bit LAT of register AUD_SPDIF_CONF is set, a delay can be configured between the output of IEC61937 in compressed mode and the output of the audio decoder. To configure a latency (in unit of seconds) this register has to be set according the following formula:

Latency = *L* x *Fs*/8 where, *L* = Latency in s and *Fs* = Sampling frequency in Hz

The minimum latency delay is 0; the maximum latency delay is the time to decode a frame:

Latency:

Dolby Digital: *L* = 1536 samples / *Fs*
MPEG: *L* = 1152 samples / *Fs*.

AUD_SPDIF_DTDI S/PDIF data-type information

7	6	5	4	3	2	1	0
PFC	DTD	Reserved	DTDI				

Address: *AudioDecBaseAddress* + 0x7F

Type: R/W

Reset: Undefined

Description:

[7] **PFC**
1: PCMCROSS function enabled

[6] **DTD**
1: Data-type dependent information used for the S/PDIF in compressed mode, can be set by the user. Refer to IEC958 standard for more information.
0: Transmitted DTDI are extracted from the stream.

[5] **Reserved**

[4:0] **DTDI[4:0]**
In Dolby digital mode: DTDI[4..3] = {0,0}; DTDI[2..0] = BSMOD[2..0]
In MPEG mode: DTDI[4..2] = {0,0,0}; DTDI[1] = DR; DTDI[0] = K

77.11 Audio command

AUD_SOFTRESET Soft reset

7	6	5	4	3	2	1	0
Reserved							SOFTRESET

Address: *AudioDecBaseAddress* + 0x10

Type: W0

Description: When bit 0 of this register is set, a soft reset occurs. The command registers and the interrupt registers listed below are cleared.The decoder goes into idle mode and the volumes are cleared.

[7:1] **Reserved**

[0] **SOFTRESET**

Command registers:
See AUD_MUTE, AUD_RUN, AUD_PLAY, AUD_SKIP_MUTE_CMD and AUD_SKIP_MUTE_VAL.

Interrupt registers:
See AUD_INTE,AUD_INT and AUD_ERROR.

AUD_PLAY Play

7	6	5	4	3	2	1	0
Reserved							PLAY

Address: *AudioDecBaseAddress* + 0x13

Type: R/W

Reset: 0

Description:

[7:1] **Reserved**

[0] **PLAY**
The PLAY command is treated according to the state of the decoder:
- When in idle mode, the PLAY value is not taken into account by the decoder.
- When in init mode, the PLAY value is not taken into account by the decoder.
- When in decode mode, PLAY enables the decoding as below:

Table 213: Play register values

PLAY value	MUTE value	DAC_SCLK, DAC_LRCLK state	DAC_PCMOUT	Decoding
0	0	Not running	0	No
0	1	Running	0	No
1	0	Running	Decoded samples	Yes
1	1	Running	0	Yes

AUD_MUTE Mute

7	6	5	4	3	2	1	0
Reserved							MUTE

Address: *AudioDecBaseAddress* + 0x14
Type: R/W
Reset: 0
Description:

[7:1] **Reserved**

[0] **PLAY**
The MUTE command is handled differently according to the state of the decoder:
- When in idle mode after hardware reset, setting MUTE to 1 automatically runs the DAC_SCLK and DAC_LRCLK clocks and outputs them to the DACs.
- When playing, setting MUTE to 1 mutes the PCM outputs.
- The AUD_MUTE register has no effect on the S/PDIF output.

AUD_RUN RUN decoding

7	6	5	4	3	2	1	0
Reserved							RUN

Address: *AudioDecBaseAddress* + 0x72
Type: R/W
Reset: 0
Description:

[7:1] **Reserved**

[0] **RUN**
his register enables to exit from idle mode. After a soft or hard reset the decoder is in idle mode. It stays in this mode until the RUN is set.
In run mode the decoder takes into account the state of all the configuration registers and begins to decode.
The AUD_RUN register can only be reset by the SOFTRESET or REBOOT commands.

AUD_SKIP_MUTE_CMD Skip or mute commands

7	6	5	4	3	2	1	0
Reserved				PAU	BLK	SKP	MU

Address: *AudioDecBaseAddress* + 0x73

Type: R/W

Reset: 0

Description: The register is taken in account at a beginning of decoding a frame.

[7:4] **Reserved**

[3] **PAU:** Pause command.
If set, the decoder does a pause during the time of decoding one block, then and forever it tests if the bit has been reset by host. If the bit is still set, it does a pause during the time of decoding a block again and so on. If reset, it plays the incoming stream. For IEC60958/61937:
- In compressed mode, the IEC61937 transmits a Pause burst, whose duration is the time of decoding a frame with gap_length_parameter_1. It then checks if the bit is still set. If the bit is still set, it transmits a Pause frame whose duration is the time of decoding a block frame with gap-length parameter-2, and so on. If reset, it transmits the next frame.
- In noncompressed mode, the IEC60958 transmits the same data as the decoder.

[2] **BLK**: Pause blocks of frames.
If set, the decoder does a pause during the number of blocks set in the AUD_SKIP_MUTE_VAL register. This command is useful for audio-video synchronization when the video decoder is in late. Once pause of blocks is finished, the decoder clears this bit. For IEC60958/61937:
- In compressed mode, the IEC61937 transmits pause bursts. The values of repetition time and Gap length parameters are given in the following table.

- In noncompressed mode, the IEC60958 is paused as the decoder. (0s are transmitted)

Table 214: Values of repetition time and gap length parameters

Samples		Dolby digital format
by frames	by blocks	
1536	256	MPEG layer II
1152	96	MPEG layer I
384	96	-

[1] **SKP**: Skip Frame
If set, the decoder skips the number of frames set in the AUD_SKIP_MUTE_VAL register. This command is useful for Audio-Video synchronization when the audio decoder is in late. Once the frame skipped, the decoder clears this bit. This command is available for all decoders. For IEC60958/61937:
- In compressed mode, the IEC61937 do not transmit the skipped frames.
- In noncompressed mode, the IEC60958 do not transmit the skipped frame as the decoder.

[0] **MU**: Global mute command.
1: PCM and S/PDIF outputs are muted. In compressed mode, the S/PDIF transmits bursts of pause-frames
In noncompressed mode, the IEC60958 transmits exactly the same data as the decoder.
0: Normal operation.

AUD_SKIP_MUTE_VAL Skip frames or mutes blocks of frame

7	6	5	4	3	2	1	0
AUD_SKIP_MUTE_VAL							

Address: *AudioDecBaseAddress* + 0x74

Type: R/W

Reset: 0

Description: This register works according to pause block of frames (bit BLK) and skip frame (bit SKP) of the AUD_SKIP_MUTE_CMD register.

If the command SKIP is selected, the decoder skips *n* frames, and then clears the register.

If the command PAUSE_OF_BLOCK is selected, the decoder pauses for *n* blocks, and then clears the register. *n* is the value of the AUD_SKIP_MUTE_VAL register. The following table gives the number of samples in a block or in a frame.

Table 215: Block and frame samples

Standard	Samples	
	By frames	By blocks
Dolby Digital	1536	256
MPEG layer II	1152	96
MPEG layer I	384	96

77.12 Interrupt

AUD_INTE Interrupt enable

	7	6	5	4	3	2	1	0
0x08	INTE[15:8]							
0x07	INTE[7:0]							

Address: *AudioDecBaseAddress* + 0x07 - 0x08

Type: R/W

Reset: 0

Description: The audio decoder contains a 16-bit interrupt register associated with a 16 bit “enable” register. A bit set in this register will enable the corresponding interrupt. The interrupt associated with each bit is given in the register INT description.

AUD_INT Interrupt

	7	6	5	4	3	2	1	0
0x0A	Reserved				FIO	FBE	FBF	PCM
0x09	ANC	PTS	BOF	DEM	SFR	ERR	HDR	SYN

Address: *AudioDecBaseAddress* + 0x09 - 0x0A

Type: RO

Reset: 0

Description: An interrupt is signalled whenever one of the bits of INT become set. This can only occur if the corresponding bit is set in the INTE register. The Table below shows the condition indicated by each bit.

0x0A:

[7:4] **Reserved**

[3] **FIO**: FIFO Input has Overflowed.[a]

[2] **FBE**: The frame buffer memory contains 1 frame which has begun to be decoded. The next frame begins to be parsed.

[1] **FBF**: Frame Buffer Full: The frame buffer memory contains 2 frames: one decoded, and one parsed for next decoding.

[0] **PCM**: PCM Output Underflow.[a]

0x09:

[7] **ANC**: Ancillary Data Registered.[b]

[6] **PTS**: First Bit of New Frame with PTS at Output Stage.[b]

[5] **BOF**: First Bit of New Frame at Output Stage.[a]

[4] **DEM**: De-emphasis Changed.[a]

[3] **SFR**: Sampling frequency changed.[a]

[2] **ERR**: Error Detected.[b]

[1] **HDR**: Valid Header Registered.[b]

[0] **SYN**: Change in Synchronization Status.[b]

a. Cleared when the MSB of the interrupt register is read or when a reset occurs.

b. Cleared when the MSB of the corresponding register is read, or when a reset occurs. Related registers are listed in the following table.

Table 216: Registers related to bits of AUD_INT

Address	Name
0x0F	AUD_ERROR
0x40	AUD_SYNC_STATUS
0x41	AUD_ANCCOUNT
0x42	AUD_HEAD4
0x46	AUD_PTS

77.13 Interrupt status

AUD_SYNC_STATUS Synchronization status

7	6	5	4	3	2	1	0
Reserved				PAC		FRA	

Address: *AudioDecBaseAddress* + 0x40

Type: RO

Reset: Undefined

Description: This register indicates the status of the audio parser for synchronization. It is used in conjunction with PACKET_LOCK and SYNCK_LOCK registers. On read the synchronization status interrupt bit is cleared (INT.SYN is cleared).

[7:4] **Reserved**

[3:2] **PAC**: Packet Status
0 0: Research packet synchronization word
0 1: Wait for confirmation - - a synchro word has been detected but the parser has not yet detected PACKET_LOCK+1 synchro words.
1 0: Synchronized - PACKET_LOCK + 1 synchro words have been detected
1 1: Not used

[1:0] **FRA**: Frame Status
0 0: Research audio synchronization
0 1: Wait for confirmation - a synchro word has been detected but the parser has not yet detected SYNC-LOCK+1 synchro words.
1 0: Synchronized - SYNC_LOCK + 1 synchro words have been detected
1 1: Not used

AUD_ANCCOUNT Ancillary data

7	6	5	4	3	2	1	0
AUD_ANCCOUNT							

Address: *AudioDecBaseAddress* + 0x41

Type: RO

Reset: Undefined

Description: This value gives the number of ancillary data in the stream. The ancillary data interrupt bit ANC of the AUD_INT register is cleared by a read.

AUD_HEAD4 HEADER 4

	7	6	5	4	3	2	1	0
AC-3	0	0	0	0	0	BSMOD		
MPEG-2	0	0	0	0	0	0	DR	K
OTHER	0	0	0	0	0	0	0	0

Address: *AudioDecBaseAddress* + 0x42

Type: RO

Reset: Undefined

Description: This register contains header data HEAD[31:24]. The contents depend on the type of the frame. HEAD4[7:3] = 00000 in all cases.

When the host reads this register, the corresponding interrupt bit (HDR) is cleared.

Dolby Digital (AC-3)

[2:0] **HEAD4[2:0]**
BSMOD if a Dolby Digital frame

MPEG-2

[2] 0

[1] **DR**
1: Dynamic range exists

[0] 0: Normal mode; 1: Karaoke mode.

Other

In all other types of frame HEAD4[2:0] = 000

AUD_HEAD3 HEADER 3

7	6	5	4	3	2	1	0
0	0	0	0	DTYPE			

Address: *AudioDecBaseAddress* + 0x43

Type: RO

Reset: Undefined

Description: This register contains header data HEAD[23:16].HEAD3[7:5] = 000, in all cases HEAD3[4:0] = DTYPE

DTYPE is the data type and is defined as follows:

[7:4] 0

[4:0] **DTYPE**

0000: Null data or Linear PCM

0001: Dolby Digital

0100: MPEG-1 Layer I

0101: MPEG-1 Layer II or MPEG-2 word extension

0110: MPEG-2 Layer II with extension

1001: MPEG-2 Layer II low sample rate

This register can not detect the data-type of data in a stream.

AUD_HEADLEN Frame length

	7	6	5	4	3	2	1	0
0x44	HEADLEN[15:8]							
0x45	HEADLEN[7:0]							

Address: *AudioDecBaseAddress* + 0x44 - 0x45

Type: RO

Reset: Undefined

Description: The HEADLEN register contains the bit length of the compressed data frame HEAD[15:0]. HEADER registers are all updated as soon as the decoder begins to decode a frame.

AUD_PTS PTS

	7	6	5	4	3	2	1	0
0x46	Reserved							PTS[32]
0x47	PTS[31:24]							
0x48	PTS[23:16]							
0x49	PTS[15:8]							
0x4A	PTS[7:0]							

Address: *AudioDecBaseAddress* + 0x46 to 0x4A

Type: R/W

Reset: Undefined

Description: When the PTS interrupt is activated, a new PTS value is stored in this register. Once the PTS[32] value is read bit PTS of the AUD_PTS register is cleared.

AUD_ERROR ERROR code

7	6	5	4	3	2	1	0
AUD_ERROR							

Address: *AudioDecBaseAddress* + 0x0F

Type: RO

Reset: 0

Description: This is a status register, when read by the ST20, this register and the corresponding interrupt register are cleared. This 7-bit register is ANDed with 0x7F to get the correct value. The value in the ERROR register indicates the type of error that has occurred. These errors are defined in the table below.

The warnings/errors for AC3 and MPEG1, 2, 3 decoder overlap from value 1 to value 21. There are no warnings in AC3 mode, only critical errors.

Table 217: Decoder warnings

Value	Warnings		
	MPEG1, 2	**MP3**	**AC3**
1	MPEG2_MC_CRC_ERROR	MP3_CRC_ERROR	EXPAND_DELTA_PAST_END_ARRAY
2	MPEG2_EXT_CRC_ERROR	MP3_CUTOFF_ERROR	XDCALL_TRY_TO_REUSE_REMAT_FLG
3	MPEG1_CRC_ERROR	MP3_BIG_VALUE_ERROR	XDCALL_TRY_TO_REUSE_COUPLING_STRA
4	MPEG_MC_LAYER1_NOT_SUPPORTED	MP3_HUFFTABLE_ERROR	XDCALL_CANT_COUPLE_IN_DUAL_MODE
5	MPEG_MC_EXT_BAD_SYNCWORD_ERROR	-	XDCALL_TRY_TO_REUSE_CPL_LEAK
6	-	-	XDCALL_TRY_TO_REUSE_SNR
7	-	-	XDCALL_TRY_TO_REUSE_BIT_ALLOC
8	-	-	XDCALL_TRY_TO_REUSE_COUPLING_EXPONENT_STRA
9	MPEG2_ILLEGAL_HEADER	MP3_MOD_BUF_SIZE_ERROR	XDCALL_TRY_TO_REUSE_EXPONENT_STRA
10	MPEG_BITRATE_ERROR	MP3_HUFFMAN_DECODE_ERROR	XDCALL_TRY_TO_REUSE_LFE_EXPONENT_STRA
11	MPEG_LAYER1_NOT_SUPPORTED	MP3_DYNPART_EXCHANGE_ERROR	XDCALL_CHBWCOD_IS_TOO_HIGH
12	MPEG_LAYER2_NOT_SUPPORTED	MP3_GR_LENGTH_ERROR	BSI_ERR_REV
13	MPEG_ILLEGAL_LAYER	MP3_CH_LENGTH_ERROR	BSI_ERR_CHANS
14	MPEG2_ILLEGAL_CHN_CONFIG	MP3_INPUT_BIT_AVAILABLE_ERROR	CRC_NOT_VALID
15	MPEG_MC_NOT_SUPPORTED	MP3_HEAD_FRAMELENGTH_ERROR	-
16	-	MP3_DYNPART_LENGTH_ERROR	-
17	-	MP3_BLOCK_TYPE_ERROR	-
18	-	MP3_HEAD_EMPHASIS_ERROR	-
19	-	MP3_HEAD_SAMP_FREQ_ERROR	-
20	-	MP3_HEAD_LAYER_ERROR	-
21	-	MP3_ILLEGAL_MODE	-

Table 218: Decoder errors

Error Name	Value
Dolby Digital Decoding	
NO-ERROR	0
Packet Synchronization	
BAD_SUB_STREAM_ID	22
PACK_HEADER_FIELD_PRESENCE_NOT_SUPPORTED	23
BAD_INFO_IN_DVD_PACKET_HEADER	24
BAD_INFO_IN_PES_PACKET_HEADER	25
BAD_INFO_IN_DVDA_PACKET_HEADER	26
BAD_INFO_IN_PACKMP1_PACKET_HEADER	27
BAD_INFO_IN_SPDIF_Pc	28
BIT_ERROR_IN_BURST_PAYLOAD	29
BAD_NULL_DATA_BURST	30
BAD_STREAM_ID	31
SYNCHRO_PACKET_NOT_FOUND	32
Audio Synchronization	
BAD_CRC_AC3	33
BAD_LPCM_QUANTIZATION_WORDLENGTH	34
BAD_AUDIO_SAMPLING_FREQUENCY	35
BAD_MPEG_LAYER	36
MPEG_BITRATE_FREE_FORMAT	37
NOT_SUPPORTED_AC-3_FRMSIZECOD	38
BAD_CRC_MPEG_FRONT_END	39
BAD_MPEG_EXTENDED_RESERVED_BIT	40
MPEG_EXTENDED_SYNC_NOT_FOUND	41
MPEG_EXTENDED_LENGTH_TOO_SMALL	42
SYNCHRO_AUDIO_NOT_FOUND	64
Other errors	
LATENCY_TOO_BIG	67
REPEAT_BLOCK_ERROR	68
UNKNOW_SFREQ_FOR_LATENCY	69
LATENCY_TOO_SMALL	70
BAD_SFREQ	75
BAD_VMAX_MODE	76
SRC_ERROR	86
ADCIN_MODE_INCORRECT	87
ADCIN_OVERFLOW_ERROR	88
BAD_HOST_CONFIG	128

77.14 Decoding algorithm

The table below shows how the AUD_STREAMSEL and AUD_DECODESEL registers are programmed for different types of bitstream.

Table 219: AUD_STREAMSEL and AUD_DECODESEL programming

AUD_STREAMSEL (0x4C)	AUD_DECODESEL (0x4D)	Mode
0	0	MPEG2 PES carrying Dolby Digital (ATSC)
0	1	MPEG2 PES carrying MPEG1 frames
0	2	MPEG2 PES carrying MPEG2 frames
0	10	MPEG2 PES carrying MPEG2-AAC frames
1	0	MPEG2 PES carrying Dolby Digital frames for DVD
1	2	MPEG2 PES carrying MPEG2 frames for DVD
1	3	MPEG2 PES carrying linear PCM for DVD
1	6	MPEG2 PES carrying DTS frames for DVD
2	1	MPEG1 packet carrying MPEG1 audio
3	0	Dolby Digital frames elementary streams
3	1	MPEG1 frame elementary streams
3	2	MPEG2 frame elementary stream
3	3	Stereo PCM (16-bit samples)
3	4	Pink noise generator
3	7	Activate PCM beep tone
3	9	MP3 elementary streams
3	10	MPEG2-AAC elementary streams
5	0	IEC61937 Input with Dolby Digital frames
5	1	IEC61937 Input with MPEG1 frames
5	2	IEC61937 Input with MPEG2 frames
5	6	IEC61937 Input with DTS frames

AUD_DECODESEL Decoding algorithm

7	6	5	4	3	2	1	0
Reserved				DEC			

Address: *AudioDecBaseAddress* + 0x4D
Type: R/W
Reset: Undefined
Description: This register identifies the audio data-type.

[7:4] **Reserved**

[3:0] **DEC**
0x00: Dolby Digital decoding
0x01: MPEG1
0x02: MPEG2
0x03: PCM/LPCM
0x04: Pink noise generator
0x06: DTS bypass
0x07: PCM beep tone generator
0x09: MP3
0x0A: AAC

AUD_STREAMSEL STREAM selection

7	6	5	4	3	2	1	0
Reserved						STRSEL	

Address: *AudioDecBaseAddress* + 0x4C
Type: R/W
Reset: Undefined
Description:

[7:3] **Reserved**

[2:0] **STRSEL**
0x00: PES
0x01: PES DVD
0x02: Packet MPEG1
0x03: Elementary Stream/IEC60958
0x04: Reserved
0x05: S/PDIF IEC61937

AUD_DEC_SWITCH Main decoding run-time switching

7	6	5	4	3	2	1	0
Reserved							DEC_SWITCH

Address: *AudioDecBaseAddress* + 0xFF
Type: R/W
Reset: Undefined
Description: This register is used for run-time switching of decoder. It is to be used as a control register from ST20 to the audio decoder to communicate a change in main decoding.

[7:1] **Reserved**

[0] **DEC_SWITCH**

77.15 System synchronization

AUD_PACKET_LOCK Packet lock

7	6	5	4	3	2	1	0
AUD_PACKET_LOCK							

Address: *AudioDecBaseAddress* + 0x4F

Type: R/W

Reset: Undefined

Description: This register specifies the number of supplementary packet synchro words that the packet parser must detect before it is considered as synchronized, and can send data to the audio parser (max=1, min=0). In this way, stream data can not be sent to the audio parser instead of packet sync words.

PACKET_LOCK = 0: the packet parser is synchronized when it has detected one packet synchro word.

PACKET_LOCK = 1: the packet parser is synchronized when it has detected two packet synchro words.

AUD_ID_EN Enable audio ID

7	6	5	4	3	2	1	0
AUD_ID_EN							

Address: *AudioDecBaseAddress* + 0x50

Type: R/W

Reset: Undefined

Description: If set to 1, the audio decoder decodes only the stream corresponding to the stream-id or sub-stream-id of the packet layer. This selection is done through AUDIO_ID or AUDIO_ID_EXT registers. If set to 0, the decoder decodes all the audio packets.

AUD_ID Audio ID

7	6	5	4	3	2	1	0
AUD_ID							

Address: *AudioDecBaseAddress* + 0x51

Type: R/W

Reset: Undefined

Description: When decoding packets, it is possible to specify an identifier for a selected program. AUDIO_ID must be written with the packet ID. This feature is enabled when the register AUDIO_ID_EN is set, and only packets with matching ID are decoded.

For MPEG1 packets or PES MPEG2 packets carrying MPEG streams, the 5 LSB bits are significant.

For PES MPEG2 packets carrying AC3 streams, the AUDIO_ID choice is not available in the standard. In case of DVD packets, the 3 LSB bits of this register are significant.

These bits correspond to the stream number defined in the STREAM_ID field of the audio packet header.

AUD_ID_EXT Audio extension

7	6	5	4	3	2	1	0
AUD_ID_EXT							

Address: *AudioDecBaseAddress* + 0x52

Type: R/W

Reset: Undefined

Description: The 3 LSB bits of this register are significant. In case of DVD MPEG2 audio with extension bitstream (see DVD specifications), this register is used to select the stream defined in the STREAM_ID of the packets containing MPEG2 extension bit stream data.

AUD_SYNC_LOCK SYNC lock

7	6	5	4	3	2	1	0
AUD_SYNCH_LOCK							

Address: *AudioDecBaseAddress* + 0x53

Type: R/W

Reset: Undefined

Description: This register specifies the number of supplementary audio synchro words that the audio parser must detect before it is considered as synchronized, and can send data to the decoder. In this way, stream data can not be sent to the decoder instead of audio sync words. Max value = 3; min value = 0.

SYNC_LOCK = 0, the audio parser is synchronized when it has detected one audio synchro word.

SYNC_LOCK = *n* > 0, when the audio parser has detected one synchro word, it waits until it detects *n* supplementary audio sync words. When it has detected (SYNC_LOCK+1) sync words, it sends the data to the decoder.

77.16 Post decoding and Pro Logic

AUD_PDEC Post decoder control

7	6	5	4	3	2	1	0
Reserved	SRS	DEM	DCF	DB	Reserved		PL

Address: *AudioDecBaseAddress* + 0x62

Type: R/W

Reset: Undefined

Description: This register controls the post decoder operations. It is also known as AUD_SRS_CALL.

[7] **Reserved**

[6] **SRS**:
1: the SRS procedure is called

[5] **DEM**: When high the de-emphasis filter is activated.

[4] **DCF**: When high the DC filter is activated.

[3] **DB**:
1: the double stereo procedure is called. It consists in a copy of L/R onto Ls/Rs channels in order to have a pseudo 5-channels decoder effect

[2:1] **Reserved**

[0] **PL**
1: Pro Logic decoding is activated.
0: the PL decoder is activated only if the output of the previous decoding stage is Pro Logic encoded.

AUD_PL_AB Pro logic auto balance

7	6	5	4	3	2	1	0
Reserved						PL_WS	PL_AB

Address: *AudioDecBaseAddress* + 0x64

Type: R/W

Reset: Undefined

Description:

[7:2] **Reserved**

[1] **PL_WS**: Select wide surround mode:
0: disabled
1: enabled

[0] **PL_AB**: Select the auto-balance function (used to track out gain between Lt and Rt)
0: disabled
1: enabled

AUD_PL_DWNX Pro logic decoder downmix

7	6	5	4	3	2	1	0
AUD_PL_DWNX							

Address: *AudioDecBaseAddress* + 0x65
Type: R/W
Reset: Undefined
Description: This value in this register controls the function of the Pro Logic downmix.

Table 220: Pro Logic downmix control

Value	Mode
0, 1, 2	Pro Logic is disabled
3	3/0 (L, C, R) three stereo
4	2/1 (L, R, S) phantom
5	3/1 (L, C, R, S)
6	2/2 (L, R, S, S) phantom
7	3/2 (L, C, R, S, S)

Phantom mode means that the Center channel is not used.

AUD_DWSMODE Downsampling filter

7	6	5	4	3	2	1	0
AUD_DWSMODE							

Address: *AudioDecBaseAddress* + 0x70
Type: R/W
Reset: Undefined
Description: This register controls the downsampling filter. When decoding a 96 kHz DVD-LPCM stream, it may be necessary to downsample the stream to 48 kHz.

Table 221: Downsampling filter control

Value	Mode
0 or 1	Automatic (according to input bitstream frequency)
2	Suppress downsampling

77.17 Bass redirection

AUD_VOLUME0 Volume of first channel

7	6	5	4	3	2	1	0
AUD_VOLUME0							

Address: *AudioDecBaseAddress* + 0x4E

Type: R/W Specific mode

Reset: 0

Reset: Undefined

Description: This register reads or writes the attenuation that is applied to the channel selected by CHAN_IDX. The attenuation is *k* dB where *k* is the contents of the register.

0: VOLUME0 can be written with the attenuation that will be applied to Left channel.

1: VOLUME0 can be written with the attenuation that will be applied to Center channel.

2: VOLUME0 can be written with the attenuation that will be applied to Left Surround channel.

5: reading VOLUME0 provides the attenuation that is applied to Left channel.

6: reading VOLUME0 provides the attenuation that is applied to Center channel.

7: reading VOLUME0 provides the attenuation that is applied to Left Surround channel.

other: meaningless.

AUD_VOLUME1 Volume of second channel

7	6	5	4	3	2	1	0
AUD_VOLUME1							

Address: *AudioDecBaseAddress* + 0x63

Type: R/W Specific mode

Reset: 0

Reset: Undefined

Description: This register reads or writes the attenuation that is applied to the channel selected by CHAN_IDX. The attenuation is -*k* dB where *k* is the contents of the register.

If CHAN_IDX = 0, then VOLUME1 can be written with the attenuation that will be applied to Right channel.

If CHAN_IDX = 1, then VOLUME1 can be written with the attenuation that will be applied to Subwoofer channel.

If CHAN_IDX = 2, then VOLUME1 can be written with the attenuation that will be applied to Right Surround channel.

If CHAN_IDX = 5, then reading VOLUME1 provides the attenuation that is applied to Right channel.

If CHAN_IDX = 6, then reading VOLUME1 provides the attenuation that is applied to Subwoofer channel.

If CHAN_IDX = 7, then reading VOLUME1 provides the attenuation that is applied to Right Surround channel.

For other values of CHAN_IDX, then the contents of this register are meaningless.

AUD_OCFG Output configuration

7	6	5	4	3	2	1	0
LFE	LVL	Reserved			OCFG_NUM		

Address: *AudioDecBaseAddress* + 0x66

Type: R/W

Reset: Undefined

Description: This register specifies the bass redirection scheme (see Dolby specifications for further information on this scheme). In this table, LP means low-pass filter, HP means high-pass filter. For OCFG_NUM equal to 2 or 3, the Subwoofer can be output if bit LFE = 1. For all other values of OCFG_NUM, bit LFE has no effect.

[7] **LFE**
0: Subwoofer is not output
1: Subwoofer is output

[6] **LVL**
Channel level
0: Default level
1: In case of OCFG2: Boost Left and Right outputs by 12 dB, remove 12 dB attenuation on C, Ls and Rs outputs.
in case of OCFG3: Boost all channels by 8 dB when Subwoofer output is disabled and 4 dB when Subwoofer output is enabled

[5:3] **Reserved**

[2:0] **OCFG_NUM**
0: ALL, All channels are rounded according to the selected output precision, (24b -> 16b, 24 -> 18b.) and scaled (volume control) only.

1: LSW, Low frequencies are extracted from the six input channels and redirected to the Subwoofer.
SUB = LP (L + R + LS + Rs + C + LFE)
Low frequencies are removed from all channels
L = HP (L) R = HP (R) C = HP (C)
LS = HP (LS) RS = HP (RS)

2: LLR, Low frequencies are extracted from C, LFE, LS and RS channels and redirected to Left and Right channels if Subwoofer is not output or to the Subwoofer.
C =HP (C), Rs = HP (Rs), Ls = HP (Ls)
If Subwoofer is output, SUB = LP(LFE + C + Ls+ Rs), L = L, R = R
If Subwoofer is not output, L = L + LP (C + LFE + LS + RS), R = R + LP (C + LFE + LS + RS)

3: SLP, Low frequencies are redirected to the Left, Right and Surround channels or can be output on the Subwoofer.
If Subwoofer is output, SUB = LFE, L = L + LP(C), R = R + LP(C), Ls = Ls, Rs = Rs
If Subwoofer is not output, L = L + LP(C) + LFE, R = R + LP(C) + LFE, Ls = Ls + LFE, Rs = Rs + LFE.

4: simplified configuration

5: BYP, All channels are directly routed to PCM outputs.

6: as per configuration 0 plus compute all channels onto sub-woofer

7: reserved

AUD_CHAN_IDX Channel Index

7	6	5	4	3	2	1	0
Reserved					CHAN_IDX		

Address: *AudioDecBaseAddress* + 0x67
Type: R/W
Reset: 4
Reset: Undefined
Description: This register identifies the pair of channels and the type of access:

[7:3] **Reserved**

[2:0] **CHAN_IDX**

Table 222: Channel index

value	Channel pair	Access	comment
value	Channel pair	Access	comment
0	Left and Right	write	
1	Center and Subwoofer	write	
2	Left Surround and Right Surround	write	
3	not used	not used	
4	no pair selected	none	Indicates that volume can be read or written
5	Left and Right	read	
6	Center and Subwoofer	read	
7	Left Surround and Right Surround	read	

To read a volume, the register AUD_CHAN_IDX must be set to the appropriate value. The DSP indicates that the attenuation is readable through registers AUD_VOLUME0 and AUD_VOLUME1 by changing automatically the AUD_CHAN_IDX to value 4.

To write a volume, the attenuation of the pair of channel should be written in AUD_VOLUME0 and AUD_VOLUME1 registers. Then the AUD_CHAN_IDX register is written to the appropriate value. The attenuation is updated on the next audio block and AUD_CHAN_IDX value is automatically changed to 4.

77.18 Dolby Digital configuration

AUD_AC3_DECODE_LFE Decode LFE

7	6	5	4	3	2	1	0
AUD_AC3_DECODE_LFE							

Address: *AudioDecBaseAddress* + 0x68

Type: R/W

Reset: Undefined

Description: When this register is set to 1, the device decodes LFE channel (if present). otherwise the register must be set to 0.

AUD_AC3_COMP_MOD Compression mode

7	6	5	4	3	2	1	0
AUD_AC3_COMP_MOD							

Address: *AudioDecBaseAddress* + 0x69

Type: R/W

Reset: Undefined

Description: The value of this register defines the compression mode. In custom A mode, the dialog normalization function is not done by the audio decoder, it has to be done by an external analog part. In all other modes the normalization is done by audio decoder.

Table 223: Compression mode

Value	Meaning
0	Custom A (Analog)
1	Custom D (Digital)
2	Line out
3	RF mode

AUD_AC3_HDR High dynamic range

7	6	5	4	3	2	1	0
AUD_AC3_HDR							

Address: *AudioDecBaseAddress* + 0x6A

Type: R/W

Reset: Undefined

Description: This register corresponds to the Dynamic range scale factor for high level signals, also called cut factor in the Dolby specifications.

HDR = 255 * Cut Factor (in decimal), where the cut factor is a fractional number between 0 and 1. It is used to scale the dynamic range control word for high-level signals that would otherwise tend to be reduced.

When HDR = 0xff (cut factor = 1.0), the high level signals reduction is the one given in the stream. A value of zero disables the high-level compression. This word is ignored if the compression mode is set to RF mode.

AUD_AC3_LDR Low dynamic range

7	6	5	4	3	2	1	0
AUD_AC3_LDR							

Address: *AudioDecBaseAddress* + 0x6B

Type: R/W

Reset: Undefined

Description: This register corresponds to the Dynamic range scale factor for low level signals, also called boost factor in the Dolby specifications.

LDR = 255 * *BoostFactor* (in decimal), where *BoostFactor* is a fractional number between 0 and 1.0.

The boost factor scales the dynamic range control-word for low-level signals that would otherwise tend to be amplified. When LDR = 0xFF (boost factor = 1.0), and the low level signals amplification is maximum. A value of zero disables the low-level amplification. This word is ignored if the compression mode is set to RF mode.

AUD_AC3_RPC Repeat count

7	6	5	4	3	2	1	0
AUD_AC3_RPC							

Address: *AudioDecBaseAddress* + 0x6C

Type: R/W

Reset: Undefined

Description: When a CRC error is detected, previous blocks can be repeated or muted. This register specifies the number of audio blocks to repeat before muting. If this is zero, then blocks are muted until the next frame is decoded.

AUD_AC3_KARAMODE Karaoke downmix

7	6	5	4	3	2	1	0
AUD_AC3_KARAMODE							

Address: *AudioDecBaseAddress* + 0x6D

Type: R/W

Reset: Undefined

Downmix mode when a karaoke bit stream is received. A Karaoke bitstream can be composed of 5 channels, which are: L (Left), R (Right), M (Music), V1 (Vocal 1) and V2 (Vocal 2). There are two major modes when receiving a Karaoke bitstream: aware and capable.

- When in 'aware' mode (AUD_KARAMODE = 0), a predefined downmix is applied on all incoming channels.
- When in 'capable' mode (AUD_KARAMODE = 4, 5, 6, 7), the user can choose to reproduce or not the two incoming vocal channels, V1 and V2.

An additional mode is added (AUD_KARAMODE = 3) to allow multi-channel reproduction. In this case, the downmix specified by the AUD_DOWNMIX and AUD_DUALMODE registers is applied. The following table summaries the different modes:

Table 224: Karaoke downmix

Value	Mode	Comment
0	Aware	Left = L + Clev*M + Slev*V1, Right = R + Clev*M + Slev*V2
1		Not used
2		Not used
3	Multichannel	Consider bitstream as multi-channel: Perform downmix according to DOWNMIX and DUALMODE registers
4	Capable	Do not reproduce V1, V2: Left = L + clev*M, Right = R + clev*M
5		Reproduction V1 only: Left = L + clev*M + 0.707*V1, Right = R + clev*M + 0.707V1
6		Reproduction V2 only: Left = L + clev*M + 0.707*V2, Right = R + clev*M + 0.707V2
7		Reproduction V1, V2: Left = L + clev*M + V1, Right = R + clev*M + V2

Left = Output Channel,
Right = Output Channel, L, R, M, V1, V2 = Input Channels (coded in Dolby Digital karaoke bitstream),
Clev = Center Mix Level (value provided in the bitstream),
Slev = Surround Mix Level (value provided in the bitstream). For further information ref. to annex C of ATSC standard "Digital Audio Compression (AC-3)".

AUD_AC3_DUALMODE Dual mode

7	6	5	4	3	2	1	0
AUD_AC3_DUALMODE							

Address: *AudioDecBaseAddress* + 0x6E
Type: R/W
Reset: Undefined
Description: See AUD_DUALMODE register

AUD_AC3_DOWNMIX Downmix

7	6	5	4	3	2	1	0
AUD_AC3_DOWNMIX							

Address: *AudioDecBaseAddress* + 0x6F
Type: R/W
Reset: Undefined
Description: See AUD_DOWNMIX register

AUD_AC3_STATUS0 Dolby Digital status

7	6	5	4	3	2	1	0
Reserved	FS_CODE		BITRATE_CODE				

Address: *AudioDecBaseAddress* + 0x76
Type: RO
Reset: Undefined
Description: This register contains bit stream information extracted from the stream.

[7] **Reserved**

[6:5] **FS_CODE**
Code identifying the sampling frequency

[4:0] **BITRATE_CODE**
Code identifying the bitrate. Bitrate[4..0] = frmsizecod[5..1]

AUD_AC3_STATUS1 Dolby Digital status register 1

7	6	5	4	3	2	1	0
Reserved				LFE	ACMOD		

Address: *AudioDecBaseAddress* + 0x77
Type: RO
Reset: Undefined
Description: This register contains bit stream information extracted from the stream.

[7:4] **Reserved**

[3] **LFE**
Indicates if LFe channel is present in the stream

[2:0] **ACMOD**
Audio coding mode. Indicates which channels are in use.

AUD_AC3_STATUS2 Dolby Digital status register 2

7	6	5	4	3	2	1	0
BSMOD			BSID				

Address: *AudioDecBaseAddress* + 0x78
Type: RO
Reset: Undefined
Description: This register contains bit stream information extracted from the stream.

[7:5] **BSMOD**
Bit stream mode, indicates the type of service

[4:0] **BSID**
Bit stream identification, indicates the version of the standard

AUD_AC3_STATUS3 Dolby Digital status register 3

7	6	5	4	3	2	1	0
Reserved				CMIXLEVEL		SURMIXLEVEL	

Address: *AudioDecBaseAddress* + 0x79
Type: RO
Reset: Undefined
Description: This register contains bit stream information extracted from the stream.

[7:4] **Reserved**

[3:2] **CMIXLEVEL**
Downmix level of Center channel

[1:0] **SURMIXLEVEL**
Downmix level of Surround channel

AUD_AC3_STATUS4 Dolby Digital status register 4

7	6	5	4	3	2	1	0
Reserved			DSURMOD		COPYRIGHT	ORIGBS	LANCODE

Address: *AudioDecBaseAddress* + 0x7A

Type: RO

Reset: Undefined

Description: This register contains bit stream information extracted from the stream.

[7:5] **Reserved**

[4:3] **DSURMOD**
In 2/0 mode, indicates if the stream is Dolby Surround encoded

[2] **COPYRIGHT**
When at 1, indicates that the stream is protected by copyright

[1] **ORIGBS**
When at 1, indicates that the stream is an original

[0] **LANCODE**
When at 1, indicates that a language code is provided in the stream

AUD_AC3_STATUS5 Dolby Digital status register 5

7	6	5	4	3	2	1	0
LANCODE							

Address: *AudioDecBaseAddress* + 0x7B

Type: RO

Reset: Undefined

Description: This register contains the code of the language of the audio service, extracted from the stream.

AUD_AC3_STATUS6 Dolby Digital status register 6

7	6	5	4	3	2	1	0
Reserved			DIALOG_NORMALIZATION				

Address: *AudioDecBaseAddress* + 0x7C

Type: RO

Reset: Undefined

Description: This register contains the code indicating the dialog normalization level extracted from the stream. For more information, see Dolby specifications.

[7:5] **Reserved**

[4:0] **DIALOG_NORMALIZATION**
See Dolby specifications

AUD_AC3_STATUS7 Dolby Digital status register 7

7	6	5	4	3	2	1	0
ROOM_TYPE		MIX_LEVEL					AUDPRODIE

Address: *AudioDecBaseAddress* + 0x7D
Type: RO
Reset: Undefined
Description: This register contains bit stream information extracted from the stream.

[7:6] **ROOM_TYPE**
If AUDPRODIE is set, mix level indicates the sound level.

[5:1] **MIX_LEVEL**
If AUDPRODIE is set, mix level indicates the sound level.

[0] **AUDPRODIE**
if set, indicates that room type and mix level are provided.

77.19 MPEG configuration

AUD_MP_SKIP_LFE Channel skip

7	6	5	4	3	2	1	0
Reserved							SKIP

Address: *AudioDecBaseAddress* + 0x68
Type: R/W
Reset: 0
Reset: Undefined
Description:

[7:1] **Reserved**

[0] **SKIP**
1: the LFE channel is skipped.
0: the LFE channel is decoded (if present).

AUD_MP_PROG_NUMBER Program number

7	6	5	4	3	2	1	0
Reserved							PROG

Address: *AudioDecBaseAddress* + 0x69
Type: R/W
Reset: 0
Reset: Undefined
Description:

[7:1] **Reserved**

[0] **PROG**
When the stream is in Second Stereo mode, this register specifies which program is played. Select program #0 or #1 where 0: L0,R0 in front channels, 1: L2,R2 in front channels

AUD_MP_DUALMODE MPEG setup dual mode

7	6	5	4	3	2	1	0
AUD_MP_DUALMODE							

Address: *AudioDecBaseAddress* + 0x6E
Type: R/W
Reset: 0
Reset: Undefined
Description: See AUD_DUALMODE register.

AUD_MP_DRC Dynamic range control

7	6	5	4	3	2	1	0
Reserved							DRC

Address: *AudioDecBaseAddress*+ 0x6A
Type: R/W
Reset: 0
Reset: Undefined
Description:

[7:1] **Reserved**

[0] **DRC**
When bit DRC=1, dynamic range control is enabled.
DRC is applied as soon as the decoder finds in the ancillary data an information which has the same syntax as the DRC. However since ancillary data may convey any information the HOST must know if the stream conveys DRC or something else.
For DVD discs, a specific field of the DVD system layer tells the host if DRC is encoded or not in the ancillary data. The HOST must use this info to switch the DRC.

AUD_MP_CRC_OFF CRC check off

7	6	5	4	3	2	1	0
AUD_MP_CRC_OFF							

Address: *AudioDecBaseAddress* + 0x6C
Type: R/W
Reset: Undefined
Description: When register is set to 1, the CRC in MPEG frame is not checked. When register is set to 0, the CRC in MPEG frame is checked if exists. If a CRC error occurs, the decoder soft mutes the frame (but does note stop).

AUD_MP_MC_OFF Multichannel

7	6	5	4	3	2	1	0
Reserved							MC

Address: *AudioDecBaseAddress* + 0x6D

Type: R/W

Reset: Undefined

Description:

[7:1] **Reserved**

[0] **MC**

When MC=1, the multi-channel part of the bitstream is not decoded, only the MPEG-1 compatible bitstream is decoded. Bit MC must be set to 1 for an MPEG-1 bitstream.

AUD_MP_DOWNMIX MPEG downmix

7	6	5	4	3	2	1	0
AUD_MP_DOWNMIX							

Address: *AudioDecBaseAddress* + 0x6F

Type: R/W

Reset: 0x08

Reset: Undefined

Description: In the table below, LO, RO, CO, LsO, RsO represent the output channels after downmix, and L, R, C, LS, RS are the audio channels.

The coefficients Kj, KC, Kr, KS, depend on the number of input channels. In the above table, the equations are given for a 5 channels input bitstream. If the input bitstream does not contain five channels (L, C, R, LS, RS), the coefficient *Kj* corresponding to the channel not present is equal to 0. If the MPEG bitstream contains only one surround channel (S), replace (KS x (LS + RS)), (KS x LS and (KS x RS) by (KS x S) in the above equations. Generic downmix equations are normalized by the max sum of coefficients. For karaoke modes, equations are not normalized.

Table 225: Downmix

Value	Output mode	Comment
0x00	2/0 (Dolby Surround LT, RT)	LT = (L + 0.707C - 0.707 x 0.5 (LS + RS)) /2.414, RT = (R + 0.707C + 0.707 x 0.5 (LS + RS)) /2.414
0x01	1/0 (C) = Mono	CO = Kj x L + C + Kr x R + KS (LS + RS)
0x02	2/0 (L, R) = Stereo	LO = (L + KC x C + KS x LS)/(1 + KC + KS), RO = (R + KC x C + KS x RS)/(1 + KC + KS)
0x03	3/0 (L, C, R)	LO = L + KS x LS, RO = R + KS x RS, CO = C
0x04	2/1 (L, R, S)	LO = L + KC x C, RO = R + KC x C, LsO = RsO = KS x (LS + RS)
0x05	3/1 (L, C, R, S)	LO = L, RO = R, CO = C, LsO = RsO = KS x (LS + RS)
0x06	2/2 (L, R, LS, RS)	LO = L + KC x C, RO = R + KC x C, LsO = LS, RsO = RS
0x07	3/2 (L, C, R, LS, RS)	LO = L, RO = R, CO = C, LsO = LS, RsO = RS
0x10	2/0 Karaoke capable: V1 OFF, V2 OFF	Lk = L + 0.707 G, Rk = R + 0.707 G
0x11	2/0 Karaoke Capable: V1 ON, V2 OFF (Dolby Digital like)	Lk = L + 0.707 A1 + 0.707 G, Rk = R + 0.707 A1 + 0.707 G
0x12	2/0 Karaoke Capable: V1 OFF, V2 ON (Dolby Digital like)	Lk = L + 0.707 A2 + 0.707 G, Rk = R + 0.707 A2 + 0.707 G
0x13	2/0 Karaoke Capable: V1 ON, V2 ON	Lk = L + 0.707 A1 + 0.707 G, Rk = R + 0.707 A2 + 0.707 G
0x14	2/0 Karaoke Capable: V1 ON, V2 OFF	Lk = L + 0.707 A1 + 0.707 G, Rk = R + 0.707 G
0x15	2/0 Karaoke Capable: V1 OFF, V2 ON	Lk = L + 0.707 G, Rk = R + 0.707 A2 + 0.707 G
0x20	3/0 Karaoke Capable: V1 OFF, V2 OFF	Lk = L, Ck = G, Rk = R
0x21	3/0 Karaoke Capable: V1 ON, V2 OFF (Dolby Digital like)	Lk = L, Ck = G + A1, Rk = R
0x22	3/0 Karaoke Capable: V1 OFF, V2 ON (Dolby Digital like)	Lk = L, Ck = G + A2, Rk = R
0x23	3/0 Karaoke Capable: V1 ON, V2 ON	Lk = L + 0.707 A1, Ck = G, Rk = R + 0.707 A2
0x24	3/0 Karaoke Capable: V1 ON, V2 OFF	Lk = L + 0.707 A1, Ck = G, Rk = R
0x25	3/0 Karaoke Capable: V1 OFF, V2 ON	Lk = L, Ck = G, Rk = R + 0.707 A2

AUD_MP_STATUS0 MPEG status register 0

7	6	5	4	3	2	1	0
ID	LAY[1:0]		P	BRI[3:0]			

Address: *AudioDecBaseAddress* + 0x76
Type: RO
Reset: Undefined
Description:

[7] **ID**: Identifier

[6:5] **LAY[1:0]**: Layer

[4] **P**: Protection Bit

[3:0] **BRI[3:0]**: Bit rate index

AUD_MP_STATUS1 MPEG status register 1

7	6	5	4	3	2	1	0
SFR[1:0]		PAD	PRI	MOD[1:0]		MEX[1:0]	

Address: *AudioDecBaseAddress* + 0x77
Type: RO
Reset: Undefined
Description:

[7:6] **SFR[1:0]**: Sampling Frequency

[5] **PAD**: Padding Bit

[4] **PRI**: Private Bit

[3:2] **MOD[1:0]**: Mode

[1:0] **MEX[1:0]**: Mode Extension

AUD_MP_STATUS2 MPEG status register 2

7	6	5	4	3	2	1	0
Reserved				C	OCB	EMP[1:0]	

Address: *AudioDecBaseAddress* + 0x78
Type: RO
Reset: Undefined
Description:

[7:4] **Reserved**

[3] **C**: Copyright

[2] **OCB:** Original/Copy Bit

[1:0] **EMP[1:0]**: Emphasis rate index

AUD_MP_STATUS3 MPEG status register 3

7	6	5	4	3	2	1	0
CEN[1:0]		SUR[1:0]		LFE	AMX	DEM[1:0]	

Address: *AudioDecBaseAddress* + 0x79
Type: RO
Reset: Undefined
Description:

[7:6] **CEN[1:0]**: Center

[5:4] **SUR[1:0]**: Surround

[3] **LFE**

[2] **AMX**: Audio mix

[1:0] **DEM[1:0]**: Dematrix procedure

AUD_MP_STATUS4 MPEG status register 4

7	6	5	4	3	2	1	0
EXT	NML[2:0]			MFS	MLY	CIB	CIS

Address: *AudioDecBaseAddress* + 0x7A
Type: RO
Reset: Undefined
Description:

[7] **EXT**: Extension bitstream present

[6:4] **NML[2:0]**: Number of Multi-lingual Channels

[3] **MFS**: Multi-lingual FS

[2] **MLY**: Multi-lingual Layer

[1] **CIB**: Copyright ID Bit

[0] **CIS**: Copyright ID Start

AUD_MP_STATUS5 MPEG status register 5

7	6	5	4	3	2	1	0
AUD_MP_STATUS							

Address: *AudioDecBaseAddress* + 0x7B
Type: RO
Reset: Undefined
Description: The number of extended ancillary data bytes is contained in this register.

77.20 AAC configuration

AUD_AAC_FORMAT AAC stream format

7	6	5	4	3	2	1	0
Reserved			IEC1937_CODE				FORMAT

Address: *AudioDecBaseAddress* + 0x69

Type: R/W

Reset: Undefined

Description:

[7:5] **Reserved**

[4:1] **IEC1937_CODE**
Gives the compress data type that must be inserted in the compressed IEC frame. If this field is set to 0 then the compress data type defaults to 7.

[0] **FORMAT**
0: ADIF format; 1: ADTS format.
Format should be set to 1 for any broadcast application.

AUD_AAC_MIX AAC mixing options

7	6	5	4	3	2	1	0
MX_IDX				Reserved	MX_2	MX_LR	MX_NORM

Address: *AudioDecBaseAddress* + 0x6D
Type: R/W
Reset: Undefined
Description:

[7:4] **MX_IDX**: Normalization index

[3] **Reserved**

[2] **MX_2**: Output second Stereo

[1] **MX_LR**: Up mix L/R to Center

[0] **MX_NORM:** Normalize downmix equations using MX_IDX
0: Mixing is normalized by the sum of coefficients

Table 226: Mix normalization

MX_IDX	normalize_mix
0	Default: 1.0/Sqrt(2)
1	1.0
2	1.0/(2*0.7)
3	1.0/(3*0.7)
4	1.0
5	1.0/(1 + 0.7)
6	1.0/(1 + 2*0.7)
7	1.0/(1 + 3*0.7)
8	1/2
9	1.0/(2 + 0.7)
10	1.0/(2 + 2*0.7)
11	1.0/(3 + 3*0.7)
higher values are reserved	

AUD_AAC_CRC_OFF AAC CRC disable

7	6	5	4	3	2	1	0
AUD_AAC_CRC_OFF							

Address: *AudioDecBaseAddress* + 0x6C
Type: R/W
Reset: Undefined
Description: Must be set to one to disable the CRC.

AUD_AAC_STATUS0 AAC status register 0

7	6	5	4	3	2	1	0
ID	LAY[1:0]		P	SFRI[3:0]			

Address: *AudioDecBaseAddress* + 0x76
Type: RO
Reset: Undefined
Description:

[7] **ID**: Identifier

[6:5] **LAY[1:0]**: Layer

[4] **P**: Protection Bit

[3:0] **SFRI[3:0]**: Sampling Frequency index

AUD_AAC_STATUS1 AAC status register 1

7	6	5	4	3	2	1	0
CHCFG[2:0]			PRI	NBRDB		CIB	CIS

Address: *AudioDecBaseAddress* + 0x77
Type: RO
Reset: Undefined
Description:

[7:5] **CHCFG[2:0]**: Channel Configuration

[4] **PRI**: Private Bit

[3:2] **NBRDB[1:0]**: Number of Raw data blocks

[1] **CIB**: Copyright_id_bit

[0] **CIS**: copyright_id_start

AUD_AAC_STATUS2 AAC status register 2

7	6	5	4	3	2	1	0
Reserved							

Address: *AudioDecBaseAddress* + 0x78
Type: RO
Reset: Undefined

AUD_AAC_STATUS3 AAC status register 3

7	6	5	4	3	2	1	0
NCH			NF		NC	NB	

Address: *AudioDecBaseAddress* + 0x79
Type: RO
Reset: Undefined
Description:

[7:5] **NCH[2:0]**: Number of channel

[4:3] **NF[1:0]**: Number of front channel

[2] **NC**: Number of Center channel

[1:0] **NB[1:0]**: Number of back channel

AUD_AAC_STATUS4 AAC status register 4

7	6	5	4	3	2	1	0
NL	NFSCE[1:0]		PSRE	RESERVED	FLGHT[10:8]		

Address: *AudioDecBaseAddress* + 0x7A
Type: RO
Reset: Undefined
Description:

[7] **NL**: LFE

[6:5] **NFSCE[1:0]**: Number of single channel element

[4] **PSE**: Pseudo Surround Enable

[3] **Reserved**

[2:0] **FLGTH[10:8]**: Frame length in byte (bitfield [10:8] out of [10:0]) Channels

AUD_AAC_STATUS5 AAC status register 5

7	6	5	4	3	2	1	0
FLGTH[7:0]							

Address: *AudioDecBaseAddress* + 0x7B
Type: RO
Reset: Undefined
Description: Frame Length in bytes (bitfield [7:0] out of [10:0])

77.21 MP3 configuration

In MP3 mode, only the downmix value 1 (mono) is available. The other values have no effect because the decoder output can only be 1 or 2 channels.

If the MP3 bitstream is mono encoded, a copy of the mono output is automatically done on L and R channels.

AUD_CRC_OFF MP3 CRC Off

7	6	5	4	3	2	1	0
Reserved							CRC_OFF

Address: *AudioDecBaseAddress* + 0x6C
Type: RO
Reset: Undefined
Description:

[7:1] **Reserved**

[0] **CRC_OFF**
1: The CRC in MP3 frame is not checked.
0: The CRC is checked and if a CRC error occurs, the corresponding frame is muted.

AUD_HEADROOM Headroom

7	6	5	4	3	2	1	0

Address: *AudioDecBaseAddress* + 0x6A
Type: R/W
Reset:
Description: Not used - set to 0x00

77.22 LPCM configuration

AUD_DOWNSAMPLING Downsampling

7	6	5	4	3	2	1	0
Reserved					DISABLE	AUTO	

Address: *AudioDecBaseAddress* + 0x70

Type: R/W

Reset: Undefined

Description:

[7:3] **Reserved**

[2] **DISABLE**

[1:0] **Auto**:
0 or 1: Downsampling is automatically applied if fs = 96 kHz.
2: no downsampling.
The SFREQ register is automatically set to the output frequency value and not to the original input frequency value.

AUD_CHANNEL_ASGN Channel assignment

7	6	5	4	3	2	1	0
AUD_CHANNEL_ASGN							

Address: *AudioDecBaseAddress* + 0xA8

Type: R/W

Reset: Undefined

Description: This register describes the output configuration of the audio channels. The value must be comprised between 0 (mono) and 23 (8 channels). See specification for Read-only Disc, Part4. Version 1.0 table C1.2.

AUD_MULTI_CHANNEL Multi channel

7	6	5	4	3	2	1	0
AUD_MULTI_CHANNEL							

Address: *AudioDecBaseAddress* + 0xA9

Type: R/W

Reset: Undefined

Description: This register describes the multi-channel structure for output channels.
0: Stereo
1: Multi-channel

AUD_LPCM_DMIX LPCM downmix

7	6	5	4	3	2	1	0
AUD_LPCM_DMIX							

Address: *AudioDecBaseAddress* + 0x6F
Type: R/W
Reset: Undefined
Description: 2: stereo downmix (using coefficients from host 0x96 to 0xA7)
8: downmix not applied

77.23 DTS status

AUD_DTS_1416BITS_MODE DTS 14/16 bits stream mode

7	6	5	4	3	2	1	0
Reserved							MODE

Address: *AudioDecBaseAddress* + 0x69
Type: R/W
Reset: Undefined
Description:

[7:1] **Reserved**

[0] **MODE**: This bit must be set with the DTS encoding mode.
0: Stream is in 14-bit mode
1: Stream is in 16-bit mode
Auto detection is performed; this means that if the software cannot find the good synchronization word in the stream, it will automatically change the value of this register (from 0 to 1 or from 1 to 0) and try to synchronize on the other 14/16 bits mode.

AUD_DTS_STATUS0 DTS status register 0

7	6	5	4	3	2	1	0
NBLKS							

Address: *AudioDecBaseAddress* + 0x76
Type: RO
Reset: Undefined
Description: This register contains the number of PCM samples per frame in the stream. The value is extracted from the stream and can be read after each BOF interrupt. The number of PCM samples/frame is: (NBLKS+1)* 32

AUD_DTS_STATUS1 DTS status 1

7	6	5	4	3	2	1	0
Reserved		AMODE					

Address: *AudioDecBaseAddress* + 0x77
Type: RO
Reset: Undefined
Description:

[7:6] **Reserved**

[5:0] **AMODE**:
This register contains the arrangement mode of the audio channels. It defines the number and type of conveyed channels.The value AMODE is extracted from the incoming bitstream and can be read after each BOF interrupt.

Table 227: DTS status 1

AMODE	No of channels	Arrangement
0	1	A
1	2	A+B (dual mono)
2	2	L+R (stereo)
3	2	(L+R) + (L-R)
4	2	Lt+Rt
5	3	C+L+R
6	3	L+R+S
7	4	C+L+R+S
8	4	L+R+SL+SR
9	5	C+R+L+SL+SR
10	6	CL+CR+LF+RF+LR+RR
11	6	C+L+R+LR+RR+OV
12	6	CF+CR+LF+RF+LR+RR
13	7	CL+C+CR+L+R+SL+SR
14	8	CL+CR+L+R+LS1+SL2+SR1+SR2
15	8	CL+C+CR+L+R+SL+S+SR
16/63	-	User defined

Key:

L:	Left	OV:	Overhead
R:	Right	A:	Channel 1
C:	Center	B:	Channel 2
S:	Surround	CR:	Center Right
F:	Front	RR:	Rear Right
R:	Rear	LS1:	Left Surround 1
T:	Total	CF:	Center Front

AUD_DTS_STATUS2 DTS status 2

7	6	5	4	3	2	1	0
Reserved				SFREQ			

Address: *AudioDecBaseAddress* + 0x78
Type: RO
Reset: Undefined
Description:

[7:4] **Reserved**

[5:0] **SFREQ**
This register contains a code corresponding to the sampling frequency of the stream. The value SFREQ is extracted from the stream and can be read after each BOF interrupt

Table 228: DTS status 2

SFREQ	Sampling frequency (kHz)
0	Invalid
1	8
2	16
3	32
4	64
5	128
6	11.025
7	22.05

SFREQ	Sampling frequency (kHz)
8	44.1
9	88.2
10	176.4
11	12
12	24
13	48
14	96
15	192

AUD_DTS_STATUS3 DTS status 3

7	6	5	4	3	2	1	0
Reserved			BITRATE				

Address: *AudioDecBaseAddress* + 0x79
Type: RO
Reset: Undefined
Description:

[7:5] **Reserved**

[4:0] **Bitrate**
This code corresponds to the bitrate of the stream. The value RATE is extracted from the stream and can be read after each BOF interrupt

Table 229: Bitrate

Value	Transmission bitrate (kbps)
0	32
1	56
2	64
3	96
4	112
5	128
6	192
7	224
8	256
9	320
10	384
11	448
12	512
13	576
14	640
15	768

Value	Transmission bitrate (kbps)
16	960
17	1024
18	1152
19	1280
20	1344
21	1408
22	1411.2
23	1472
24	1536
25	1920
26	2048
27	3072
28	3840
29	open
30	variable
31	lossless

77.24 PCM beep-tone

In PCM beep tone mode, AUD_DOWNMIX register must be set to 8.

AUD_BEEP_FREQ PCM beep tone frequency

7	6	5	4	3	2	1	0
BEEP_FREQ							

Address: *AudioDecBaseAddress* + 0x68
Type: R/W
Reset: HW: undefined / SW: 0
Description: The value in this register sets the PCM beep tone frequency according to the formula:
beep tone frequency = (Fs/2)/(BEEP_FREQ)
with BEEP_FREQ in the range 8 - 254.

AUD_CHANNELCONF PCM beep tone channel configuration

7	6	5	4	3	2	1	0
Reserved		RS	LS	LFE	C	R	L

Address: *AudioDecBaseAddress* + 0x69
Type: R/W
Reset: HW: undefined / SW: 0
Description:

[7:6] **Reserved**

[5] **RS**
1: Right Surround channel contains PCM beep tone
0: Right Surround channel is forced to 0

[4] **LS**
1: Left Surround channel contains PCM beep tone
0: Left Surround channel is forced to 0

[3] **LFE**
1: LFE channel contains PCM beep tone
0: LFE channel is forced to 0

[2] **C**
1: Center channel contains PCM beep tone
0: Center channel is forced to 0

[1] **R**
1: Right channel contains PCM beep tone
0: Right channel is forced to 0

[0] **L**
1: Left channel contains PCM beep tone
0: Left channel is forced to 0

77.25 Pink noise

In Pink noise mode, the AUD_DOWNMIX register must be set to 8; AUD_OCFG must be set to 0, and attenuation on all channels must be set to -10 dB.

AUD_PN_CHANNELCONF PCM pink noise channel configuration

7	6	5	4	3	2	1	0
Reserved		RS	LS	LFE	C	R	L

Address: *AudioDecBaseAddress* + 0x69

Type: R/W

Reset: HW: undefined / SW: 0

Description:

[7:6] **Reserved**

[5] **RS**
1: Right Surround channel contains pink noise
0: Right Surround channel is forced to 0

[4] **LS**
1: Left Surround channel contains pink noise
0: Left Surround channel is forced to 0

[3] **LFE**
1: LFE channel contains pink noise
0: LFE channel is forced to 0

[2] **C**
1: Center channel contains pink noise
0: Center channel is forced to 0

[1] **R**
1: Right channel contains pink noise
0: Right channel is forced to 0

[0] **L**
1: Left channel contains pink noise
0: Left channel is forced to 0

77.26 De-emphasis

AUD_DEEMPH De-emphasis

7	6	5	4	3	2	1	0
FORCE	Reserved					E[1:0]	

Address: *AudioDecBaseAddress* + 0xB5

Type: R/W Specific mode

Reset: 0

Description:

[7] **FORCE**
The de-emphasis filter specified here is applied only if bit DEM of the AUD_PDEC register is set.
Whenever the de-emphasis status changes an interrupt is generated.
This register can be used in each decoder, but only MPEG and DVD_LPCM standards have an emphasis information in the bitstream header.
In MPEG and DVD_LPCM modes:
if FORCE = 0, the emphasis information is extracted from the bitstream and the register AUD_DEEMPH is written with the right corresponding value.
If FORCE = 1 the value extracted from the bitstream is ignored. However if this value is different from the one of AUD_DEEMPH register an interrupt is generated at each frame.
In all other modes the bit FORCE must be set to 1 to force the de-emphasis

[6:2] **Reserved**

[1:0] E[1:0]
00: none, 01: 50/15s, 10: reserved, 11: CCITT J.17

77.27 Downmix

AUD_DOWNMIX Downmix

7	6	5	4	3	2	1	0
DOWNMIX_VALUE							

Address: *AudioDecBaseAddress* + 0x6F

Type: R/W

Reset: Undefined

Description: The downmix table is identical for all decoders but the downmix coefficients may differ. (mix coefficient is different between AC3 and MPEG).

Table 230: Downmix

Value	Output mode
0x00	2/0 (Dolby Surround LT, RT)
0x01	1/0 (C) = Mono
0x02	2/0 (L, R) = Stereo
0x03	3/0 (L, C, R)
0x04	2/1 (L, R, S)
0x05	3/1 (L, C, R, S)
0x06	2/2 (L, R, LS, RS)
0x07	3/2 (L, C, R, LS, RS)
0x08	No downmix

77.28 Dual mode

AUD_DUALMODE Dual mode

7	6	5	4	3	2	1	0
AUD_DUALMODE							

Address: *AudioDecBaseAddress* + 0x6E

Type: R/W

Reset: Undefined

Description: This register allows additional downmix to be set when in 2/0 output mode or when receiving a “Dual mode” incoming bitstream (example: a disk with 2 different languages on channel 1 and channel 2). In the following table, channel 1 and 2 represent the output channels after downmix performed with AUD_DOWNMIX.

This register enables Mono downmix when AUD_DOWNMIX = 2 and AUD_DUALMODE = 3.

Table 231: Dual mode

Value	Comment
0	Output as Stereo
1	Output Channel 1 on both output L/R
2	Output Channel 2 on both output L/R
3	Mix Channel 1 and 2 to monophonic and output on both L/R

77.29 TruSurround XT™ configuration

CAUTION: TruSurround/TruSurround XT, Focus and TruBass currently support only 32, 44.1 and 48 kHz sampling frequencies. If the audio decoder is configured to have another sample frequency, the value used is always 48 kHz.

AUD_SRS_CALL See AUD_PDEC

AUD_PL_AB SRS/TruSurround mode

7	6	5	4	3	2	1	0
TS-XT_MODE				HEADPHONE			

Address: *AudioDecBaseAddress* + 0x64

Type: R/W

Reset: HW: Undefined / SW: N/A

Description: SRS/TruSurround is activated by setting the bit SRS of the AUD_PDEC register.

[7:4] **TS-XT_MODE**

4 bits to describe the SRS decoder mode

Table 232: SRS/TruSurround XT and headphone modes

Value TS-XT_Mode	TruSurround XT Modes	HeadPhone mode
0	K1_0	No
1	K2_0	Yes
2	K2_1	No
3	K2_2	No
4	K3_0	No
5	K3_1	No
6	K3_2	No
7	K3_3 (Not Supported)	No
8	kPL2	No
Others	Disable post-process	No

Sampling frequency: 3 frequencies are supported: 32, 44.1 and 48 kHz.

Default Gain of TruSurround XT will be selected if all gain registers will be Set to 0, As given below:

Table 233:

Control	Gain, dB			
	Normal Modes	SRS Modes	3/1 (Pro-Logic Decoded)	3/2 (Pro-Logic II Decoded)
Front Center	-20	-9	-20	-20
Front Space	-4	-4	-4	-4
Rear Center	-11	-11	-11	-11
Rear Space	+9	+9	+5	+3

[3:0] **HEADPHONE**

AUD_SRS_PFSPACE_GAIN TruSurround front space gain

7	6	5	4	3	2	1	0
AUD_SRS_PFSPACE_GAIN							

Address: *AudioDecBaseAddress* + 0xF4

Type: R/W

Reset: HW: Undefined / SW: 0

Description: Perspective front space gain. The value in this register must be set to 2 * *X* to achieve a gain of -*X* dB except for a value of 10 which corresponds to -9 dB.

AUD_SRS_PFCENTER_GAIN TruSurround front center gain

7	6	5	4	3	2	1	0
AUD_SRS_PFCENTER_GAIN							

Address: *AudioDecBaseAddress* + 0xF5

Type: R/W

Reset: HW: Undefined / SW: 0

Description: Perspective front Center gain. The value in this register must be set to 2**X* to achieve a gain of -*X* dB except for a value of 10 which corresponds to -9 dB.

AUD_SRS_PRSPACE_GAIN TruSurround rear space gain

7	6	5	4	3	2	1	0
AUD_SRS_PRSPACE_GAIN							

Address: *AudioDecBaseAddress* + 0xF6

Type: R/W

Reset: HW: Undefined / SW: 0

Description: Perspective Rear space gain. The value in this register must be set to 2**X* to achieve a gain of -*X* dB except for a value of 10 which corresponds to -9 dB.

AUD_SRS_PRCENTER_GAIN TruSurround rear center gain

7	6	5	4	3	2	1	0
AUD_SRS_PRCENTER_GAIN							

Address: *AudioDecBaseAddress* + 0xF7

Type: R/W

Reset: HW: Undefined / SW: 0

Description: Perspective Rear Center gain. The value in this register must be set to 2**X* to achieve a gain of -*X* dB except for a value of 10 which corresponds to -9 dB.

AUD_SELECT_CENTER TruSurround center surround channel selection

7	6	5	4	3	2	1	0
Reserved							CS

Address: *AudioDecBaseAddress* + 0xF8
Type: R/W
Reset: HW: Undefined / SW: 0
Description:

[7:1] **Reserved**

[0] **CS**
Must be set to zero.

AUD_SRS_INPUT_GAIN TruSurround input gain

7	6	5	4	3	2	1	0
AUD_SRS_INPUT_GAIN							

Address: *AudioDecBaseAddress* + 0xF9
Type: R/W
Reset: HW: Undefined / SW: 0
Description: This register is used to provide attenuation at input stage. Set value in this register to 2**X* to achieve a gain of -*X* dB.

AUD_SRS_SHIFT TruSurround split block size

7	6	5	4	3	2	1	0
AUD_SRS_SHIFT							

Address: *AudioDecBaseAddress* + 0xFB
Type: R/W
Reset: HW: Undefined / SW: 0
Description: The audio decoder SRS implementation cannot accept a block size more than **288** samples. Future decoders may accept larger blocks, in which case, block size can be divided using this register by 2, 4, 8... For division of block size by two, set this bit to 1.

AUD_TSXT_6DB_GAIN TruSurround output gain

7	6	5	4	3	2	1	0
Reserved						GAIN	Reserved for TSXT

Address: *AudioDecBaseAddress* + 0xF8
Type: R/W
Reset: HW: Undefined / SW: 0
Description:

[6:2] **Reserved**

[1:0] **GAIN**
Must be set to 1 to provide an additional gain of 6 dB.

[1:0] **Reserved** for TSXT.

77.30 SRS® focus configuration

AUD_POSTPROC See AUD_ADCIN_PLAY

AUD_FOCUS_ELEVATION SRS focus percentage of elevation

7	6	5	4	3	2	1	0
AUD_FOCUS_ELEVATION							

Address: *AudioDecBaseAddress* + 0xFA
Type: R/W
Reset: HW: Undefined / SW: 0
Description: The value set in this register corresponds to the percentage of elevation required. Maximum permitted value is 100, if set above 100 then automatically sets the value to 100.

AUD_FOCUS_BASS_COMP SRS focus bass compensation gain

7	6	5	4	3	2	1	0
AUD_FOCUS_BASS_COMP							

Address: *AudioDecBaseAddress* + 0xFD
Type: R/W
Reset: HW: Undefined / SW: 0
Description: Bass compensation gain, to select -*X* dB, set value in register to 2**X*.
Note: This register works only while focus is selected in the stand-alone mode (without TS-XT).

77.31 SRS TruBass® configuration

AUD_POSTPROC See AUD_ADCIN_PLAY

AUD_TBASS_MODE SRS TruBass mode

7	6	5	4	3	2	1	0
AUD_TBASS_MODE							

Address: *AudioDecBaseAddress* + 0x8A
Type: R/W
Reset: HW: Undefined / SW: 0
Description: This register selects the TruBass mode (Speaker cutoff frequency).

Table 234: TruBass mode

Value	Speaker cutoff frequency (Hz)
0	40
1	60
2	100
3	150
4	200
5	250
6	300
7	400

AUD_TBASS_CONTROL SRS TruBass elevation percentage

7	6	5	4	3	2	1	0
AUD_TBASS_CONTROL							

Address: *AudioDecBaseAddress* + 0x8B
Type: R/W
Reset: HW: Undefined / SW: 0
Description: This register is used for TruBass elevation. The value assigned to this register correspond directly the percentage of elevation required. If the register exceeds 100, it is automatically set to 100.

78 Audio DAC

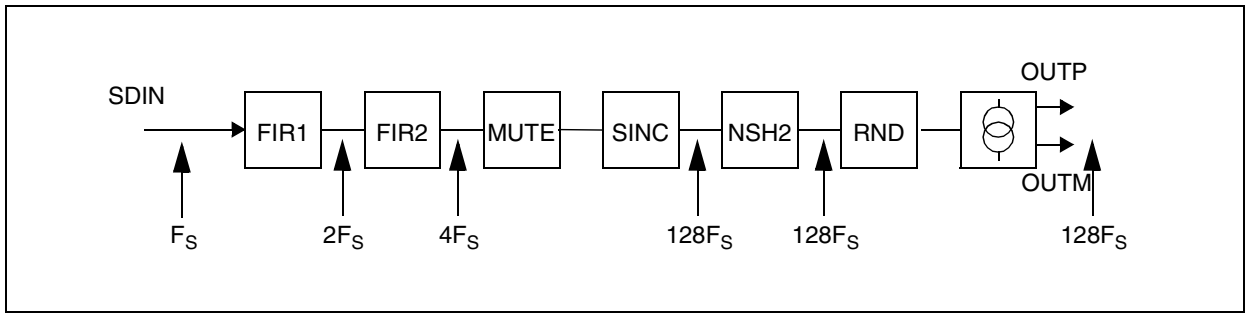
78.1 Description

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 as well as any other audio frequencies below 48 kHz.

Figure 266: 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.

78.2 Operating modes

78.2.1 Reset

A low level on NOTRESETIN or a reset of bit ADAC_CFG.NRST (see page 851) puts the system in reset mode by initializing internal counters and control registers.

At reset, the audio DAC is disabled.

78.2.2 Supplies

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

- AUDDACGNDA, AUDDACGNDAS, AUDDACVDDA: ground and 3.3 V analog supplies for the switches (control of the current sources) are supplied to the chip externally.
- AUDDACGND33, AUDDACVDD33: ground and 3.3 V analog supplies for the polarization block and the current sources are supplied to the chip externally.

78.2.3 Input/output signals

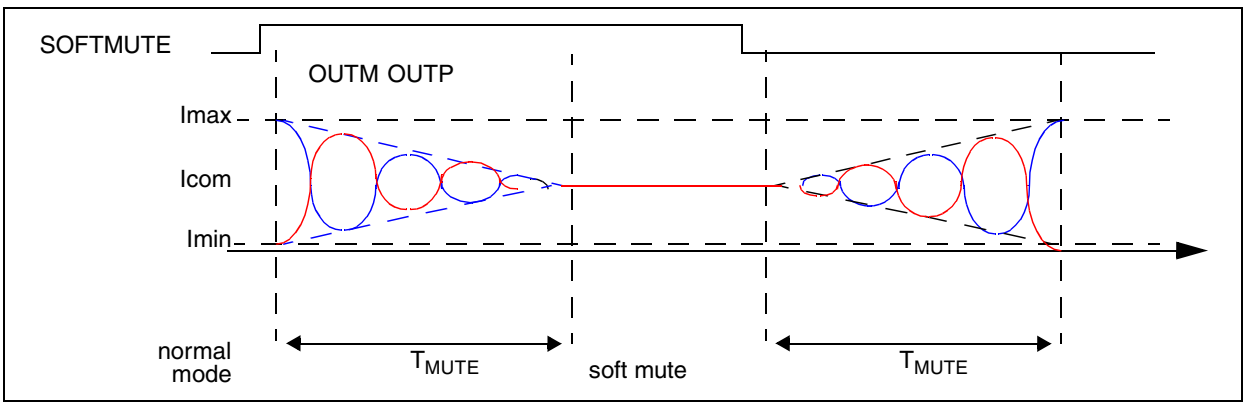
Table 235: Audio DAC output signals

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.
AUDANAIREFOUT	DAC reference current output. This pin should be connected to an external resistor.
AUDANAVBGFILIN	DAC filtered reference voltage input. This pin should be connected to an external 10 µF capacitor (ground connection AUDDACGNDA).

78.2.4 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 Icom, 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 267: Soft mute and digital power down



78.2.5 Digital and analog power down

The digital part of the audio DAC can be disabled by setting ADAC_CFG.NSB to 0. An automatic softmute 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 not be avoidable.

78.3 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/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 AUDANAIREFOUT of the DAC. A typical value for R_{REF} is 200 Ω. R_{REF} should always be higher than 175 Ω to get proper band gap functionality.

79 Electrical specifications

79.1 Power supplies

The various voltages required for power supplies are:

- **1.2V** nominal for the **digital core and analog cells**
- **3.3V** nominal for the **I/O ring** (3.3 V capable)
- **2.5V** nominal for **SSTL pads** (DDR interface)

Power consumption is around 2.5 W maximum for the digital core and IOs, plus analog power consumption (including one 35 mA DAC) of about 1 W, making a total of **3.5 Watt overall**.

- With a 27x27 BGA, four layer substrate, four layer PCB: R_{th} is around 18 °C/W.

The package ambient temperature must be kept below **60 °C**.

Table 236: Power supply requirements

Supply	Pin name	Comments
3.3V I/O supply	VDDE3V3	
1.2V core supply	VDD	
2.5V LMI supply	VDE2V5	SSTL levels
3.3V video DAC I/O	VDDE_ANA	
1.2V video DAC and freq synth	VDD_ANA	
1.2V HDMI power supply	VDD_HDMI	
3.3V HDMI I/O supply	VDDE_HDMI	
1.2V USB PHY and USB PLL power supply	VDD_USB	

Power down mode

In power down mode, most internal clocks are disabled or dramatically slowed down, reducing power consumption to less than **200 mW** at 25 °C ambient temperature (for example, when the box has cooled down after use); however at 125 °C (that is, just after powering down the box), up to **400 mW** is possible

79.2 Audio DACs

Table 237: Audio DAC parameters

Parameter	Baseband analog, dB
SNR	-90 @ -3 dBfs -111 @ -60 dBfs
Signal to (THD+N)	90
Dynamic range	110
THD	-98
Inter channel isolation	-90
Out of band noise	-70

79.3 SD Video DAC

Table 238: SD VIdeo DAC parameters

Parameter	Min	Typ	Max	Units	Notes
Differential nonlinearity (DNL)	-0.5		0.5	LSB	
Integral nonlinearity (INL)	-1		1	LSB	
Dynamic current	7	8	9	mA	1
Output Voltage			1.6	V DC	1
Analog Bandwidth	14			Mhz	
DAC to DAC matching		+/- 1	+/- 2.5	%	
Conversion rate			80	Mhz	
SNR, BW = 10 Mhz, F_{clk} = 80Mhz		50		dB	
THD, F_{out} = 5Mhz, F_{clk} = 80Mhz		50			
REXT - GNDAS_REXT (Vbangap)	-7%	1.214	+7%	V	

1. The output current of 8mA is with typical bandgap voltage (1.214V) and R_{ref} = 9.7kohm

79.4 HD Video DAC

Table 239: HD VIdeo DAC parameters

Parameter	Min	Typ	Max	Units
Differential nonlinearity (DNL)	-0.5		0.5	LSB
Integral nonlinearity (INL)	-2		2	LSB
DAC to DAC matching		+/- 1.5	+/- 2.5	%
THD, F_{in} = 4Mhz, F_{clk} = 216Mhz	46	52	60	dB

79.5 DAA

See DAA document.

79.6 USB2.0

The interface on the STi7710 is compatible with USB2.0 and USB1.1 - refer to USB1.1 and USB2.0 specifications.

80 Timing specification

80.1 System

80.1.1 Reset timing

The $t_{RSTHRSTL}$ value in the following figure and table is for power-on reset when the device is cold.

Figure 268: Reset timings

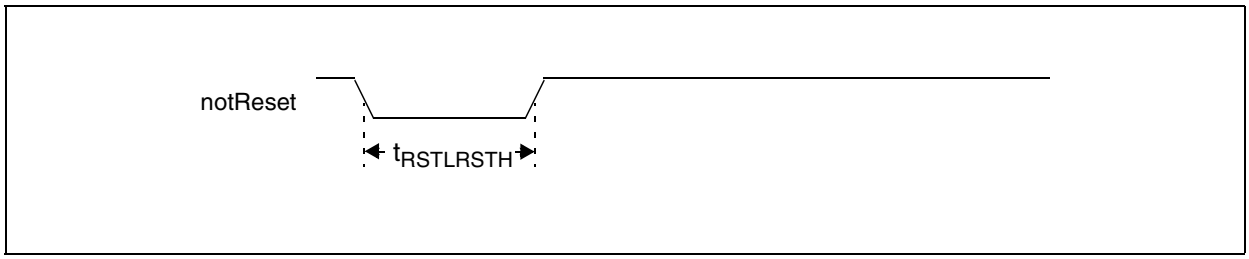


Table 240: Reset timings figures

Symbol	Parameter	Min	Nom	Max	Units
$t_{RSTLRSTH}$	notReset pulse width low[a]	10	-	-	ms

a. Crystal oscillator start-up times can be long and may dominate the time from power-up to reset.

80.1.2 Watchdog reset out

Figure 269: WatchDog output timings

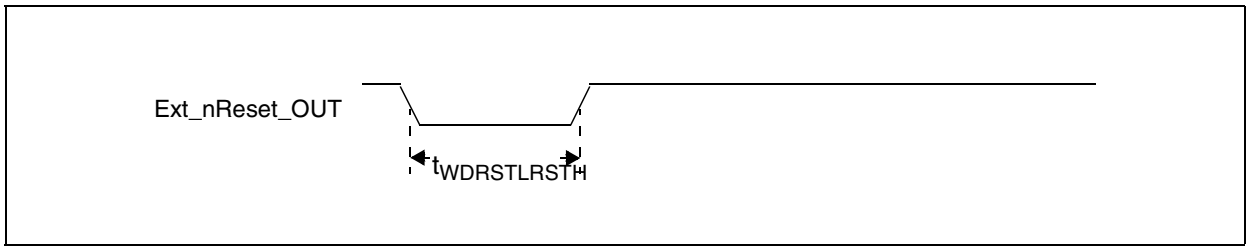


Table 241: WatchDog output figures

Symbol	Parameter	Min	Nom	Max	Units
$t_{WDRSTLRSTH}$	Ext_nReset_OUT pulse width low	10	-	200	ms

80.1.3 Low power mode timings

Table 242: LPC requirements

	Min	Max
Clock frequency	FS1.4 / 1024 (fs_clockgen)	46.875 kHz

80.1.4 System clock

External 27 MHz clock

Either a 27 MHz clock can be fed into CLK27IN, or a crystal pi network may be connected between CLK27IN and CLK27OSC. The crystal option and internal DCO is the option recommended by STMicroelectronics.

Crystal requirements

The external 27 MHz crystal has the following requirements:

- 27.000 MHz fundamental frequency,
- parallel resonance,
- 40 ppm accuracy.

The capacitance of the clk_xtal net has been checked and is below the oscillator maximum capacitance (1.13 pF / 1.28 pF).

80.2 LMI DDR-SDRAM timings

For write sequences, the timing reference is the LMI_CLK edge.

For reads, the timing reference is the LMI_notDQS 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 + 0.5 pF wireload,
- output threshold = 0.5 * VDD25.

Figure 270: LMI DDR-SDRAM timings

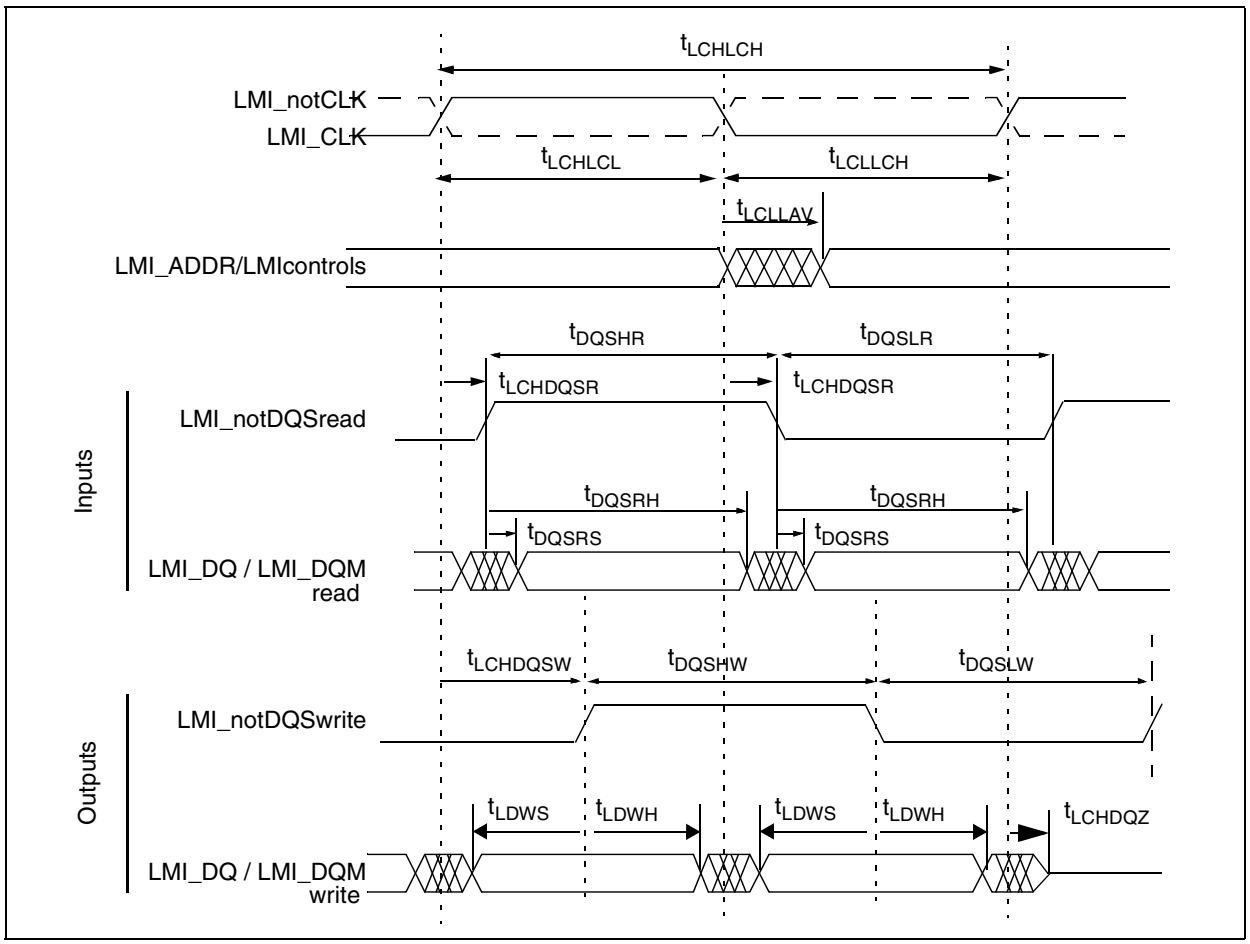


Table 243: LMI DDR-SDRAM AC timings

Symbol	Parameter	Min	Max	Units	Note
Outputs					
t_{LCHLCH}	LMI_CLK period	5.26		ns	
t_{LCHLCL}	LMI_CLK high time	0.45		t_{LCHLCH}	
t_{LCLLCH}	LMI_CLK low period	0.45		t_{LCHLCH}	
t_{DQSHW}	Write LMI_notDQS high	0.45		t_{LCHLCH}	
t_{DQSLW}	Write LMI_notDQS low	0.45		t_{LCHLCH}	
t_{LCLLAV}	LMI_CLK falling edge to LMI_ADDR/LMI controls valid	-0.08	+0.29	ns	a
$t_{LCHDQSW}$	LMI_CLK edge to write LMI_notDQS edge	-0.12	-0.05	ns	
t_{LDWS}	Write data valid to write LMI_notDQS edge (setup)	1.13		ns	
t_{LDWH}	LMI_notDQS edge to write data valid (hold)	1.13		ns	
t_{LCHDQZ}	LMI_CLK high to write data Z		7.05	ns	
Inputs					
$t_{LCHDQSR}$	Allowable arrival time of LMI_notDQS edge with respect to LMI_CLK edge	-0.75	+0.75	ns	
t_{DQSHR}	Read LMI_notDQS high time	0.45		t_{LCHLCH}	
t_{DQSLR}	Read LMI_notDQS low time	0.45		t_{LCHLCH}	
t_{DQSRS}	Latest allowable invalid read data after LMI_notDQS edge		0.5	ns	
t_{DQSRH}	Earliest allowable invalid data before following LMI_notDQS edge	t_{LCHLCL} - 0.55		ns	

a. Applies to LMI_notCS[1:0], LMI_notCAS, LMI_notRAS, LMI_RDNOTWR, LMI_ADDR[12:0], LMI_notBANK[1:0], LMI_CLKEN.

80.3 Audio input AC specification

The Audio input is characterized with respect to the ASTRB input clock (rising edge). They are clocked on the rising edge of this clock.

Figure 271:

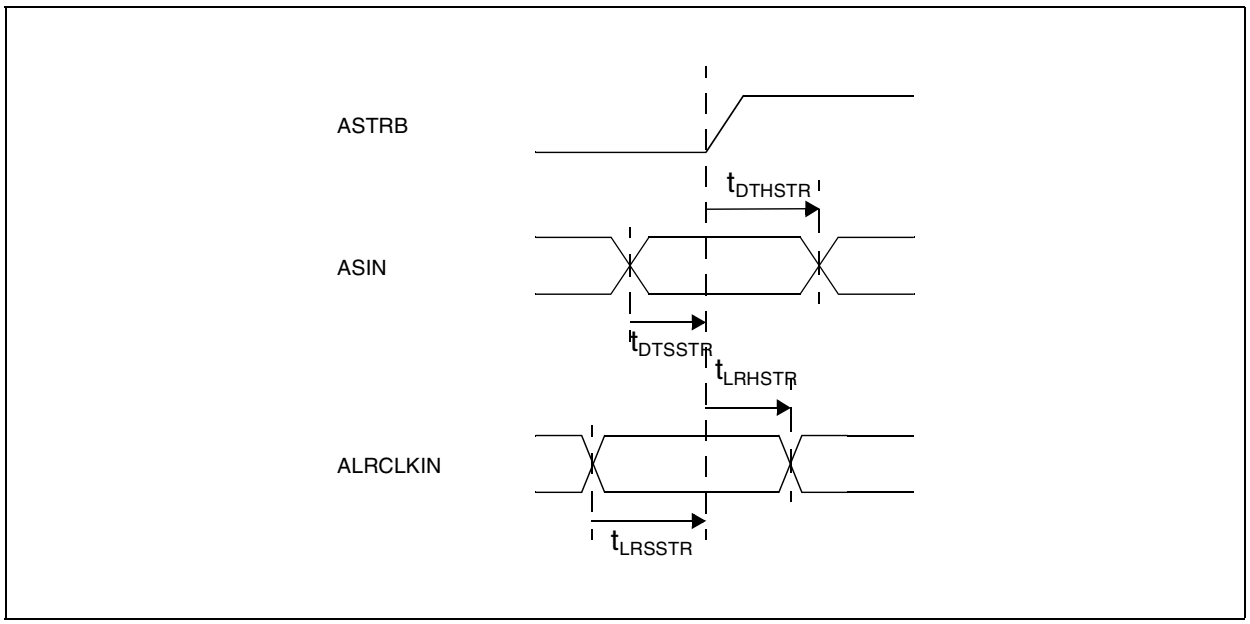


Table 244:

Symbol	Parameter	min	max	Units
	ASTRB maximum frequency		15	MHz
tDTSSTR	ASIN setup with respect to ASTRB rising edge	2		ns
tDTHSTR	ASIN hold with respect to ASTRB rising sdge	0		ns
tLRSSTR	ALRCLKIN setup with respect to ASTRB rising edge	2		ns
tLRHSTR	ALRCLKIN hold with respect to ASTRB rising edge	0		ns

80.4 Audio output AC specification

Ouput clock: SCLK

Outputs: LRCLK, AUDIO_PROD_TEST

Figure 272:

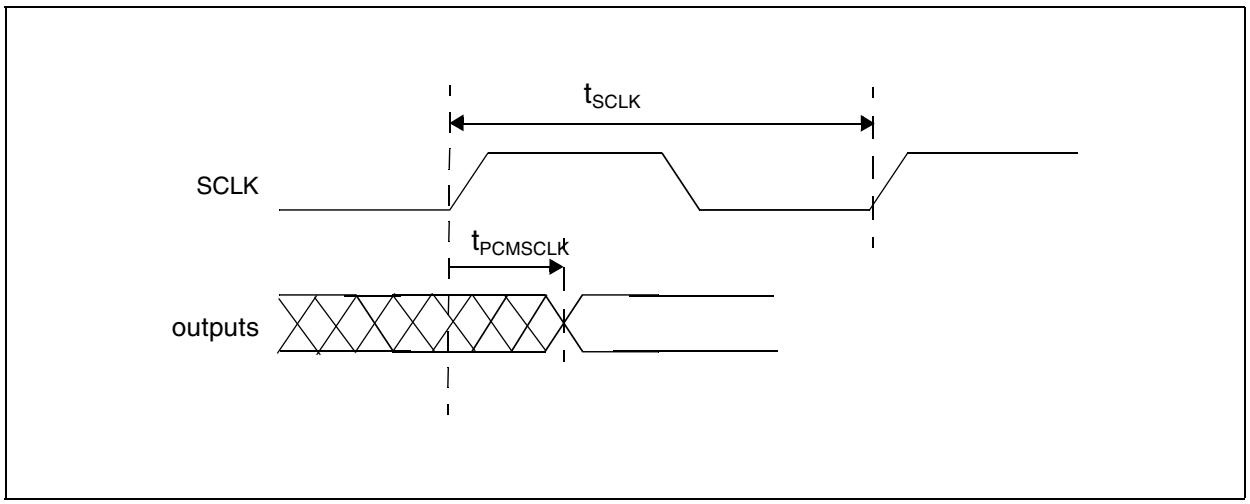


Table 245:

Symbol	Parameter	min	max	Units
tSCLK	SCLK clock period	133.2		ns
tPCMSCLK	Output delay with respect to SCLK		0.5	ns

80.5 SPDIF

The S/PDIF output is Manchester encoded and hence self-clocking. There are no requirements on this signal relative to other chip signals. However, as the signal carries its own clock there is a jitter requirement. This is a derived clock that is measured on an application board.

80.6 General PIO

Figure 273: General PIO timings

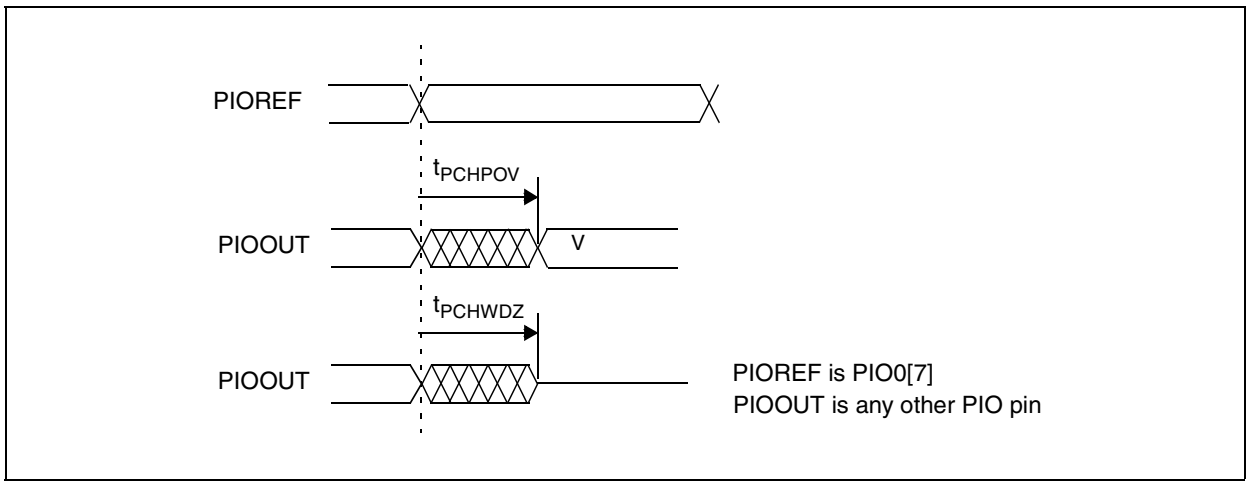


Table 246: General PIO timing parameters

Symbol	Parameter	Min	Max	Units
t_{PCHPOV}	PIOREF transition to PIO output valid	0.1	9.3	ns
t_{PCHWDZ}	PIO tri-state after PIOREF transition	0.2	7.4	ns

80.7 DMA

A pair of request signals for DMA (EXT_DMA_REQ1 and EXT_DMA_REQ2). These are used mostly by the FDMA and are asynchronous.

80.8 Digital video input

Input clock: DCKI (rising edge)

SYNC Inputs: DHSI, DVSI

Figure 274:

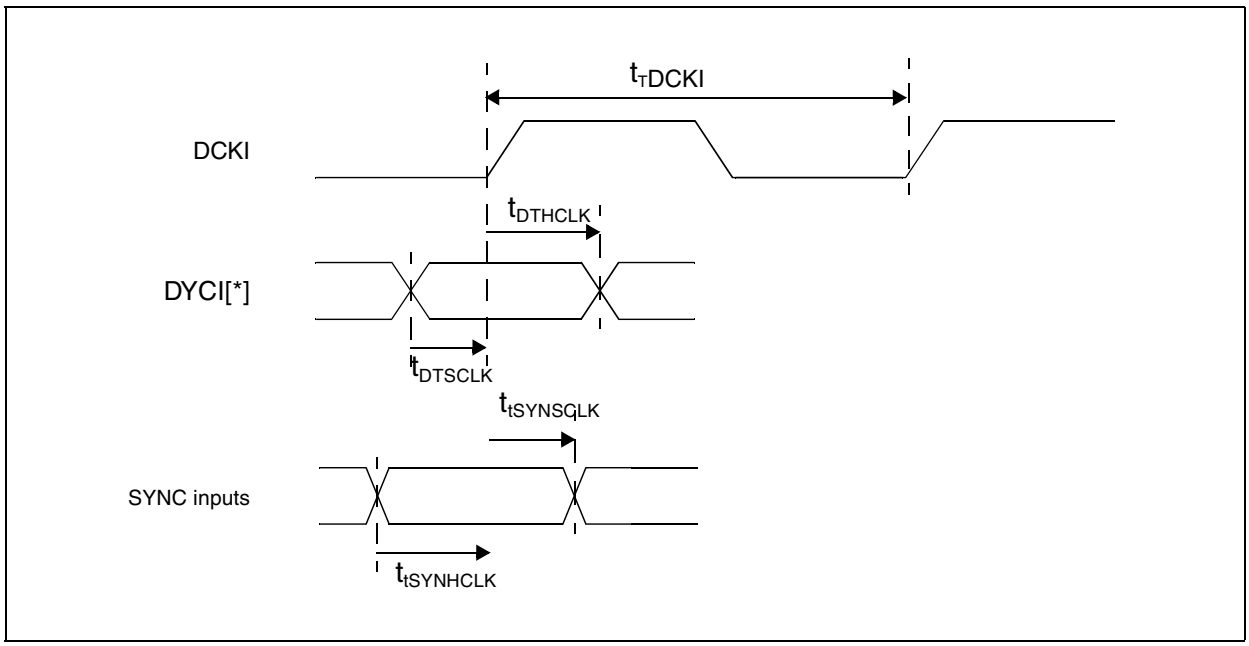


Table 247:

Symbol	Parameter	min	max	Units
tDCKI	DCKI period	37.04		ns
tDTSCLK	Data inputs setup with respect to DCKI rising edge	0		ns
tDTHCLK	Data inputs hold with respect to DCKI rising sdge	4		ns
tSYNSCLK	HSYNC, VSYNC inputs setup respect to DCKI rising edge	0		ns
tSYNHCLK	HSYNC, VSYNC inputs hold respect to DCKI rising edge	4		ns

80.9 HD Digital Video output (measures done on FUNC1)

Output clock: PIXCLK (rising edge)
SYNC : DHSO, DVSO
DATA_OUT : DYCIO[15:0], CLAMP

Figure 275:

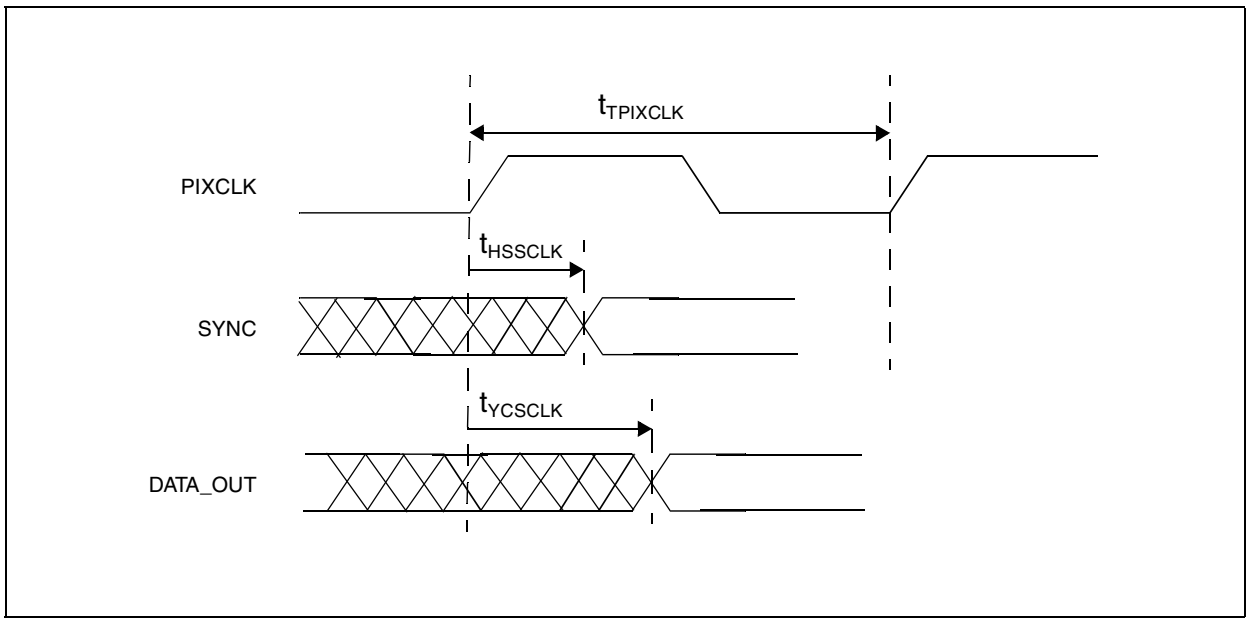


Table 248:

Symbol	Parameter	min	max	Units
tPIXCLK	PIXCLK period	13.46		ns
tHSSCLK	SYNC output delay with respect to PIXCLK		4.1	ns
tVCSCLK	DATA_OUT output delay with respect to PIXCLK		5.1	ns

Note: PIXCLK has to be inverted. Launch data on falling edge, capture on rising.

80.10 SD Digital Video output (measures done on FUNC4)

Output clock: PIXCLK (rising edge)

SYNC : embedded syncs in DYCIO

DATA_OUT : DYCIO[15:0], CLAMP

Figure 276:

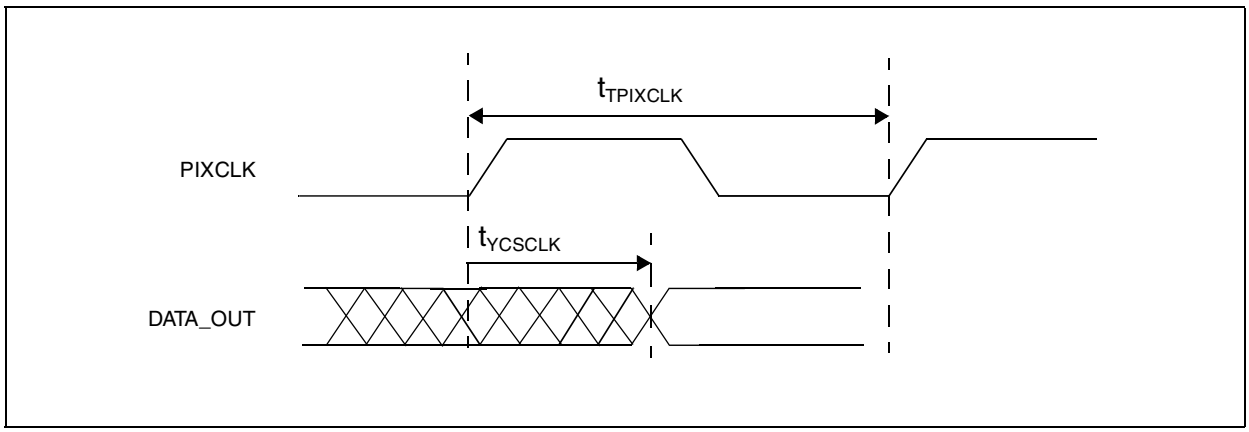


Table 249:

Symbol	Parameter	min	max	Units
tPIXCLK	PIXCLK period	18.52		ns
tVCSCLK	DATA_OUT output delay with respect to PIXCLK		4.7	ns

Note: PIXCLK has to be inverted. Launch data on falling edge, capture on rising.

80.11 Transport stream input AC specification

Input clocks: TSBYTECLK0, TSBYTECLK1, TSBYTECLK2 (rising edge)

Inputs: TSVALID0, TSPKTERR0, TSPKTCLK0, TSDATA0, TSVALID1, TSPKTERR1, TSPKTCLK1, TSDATA1, TSVALID2, TSPKTERR2, TSPKTCLK2, TSDATA2

Figure 277:

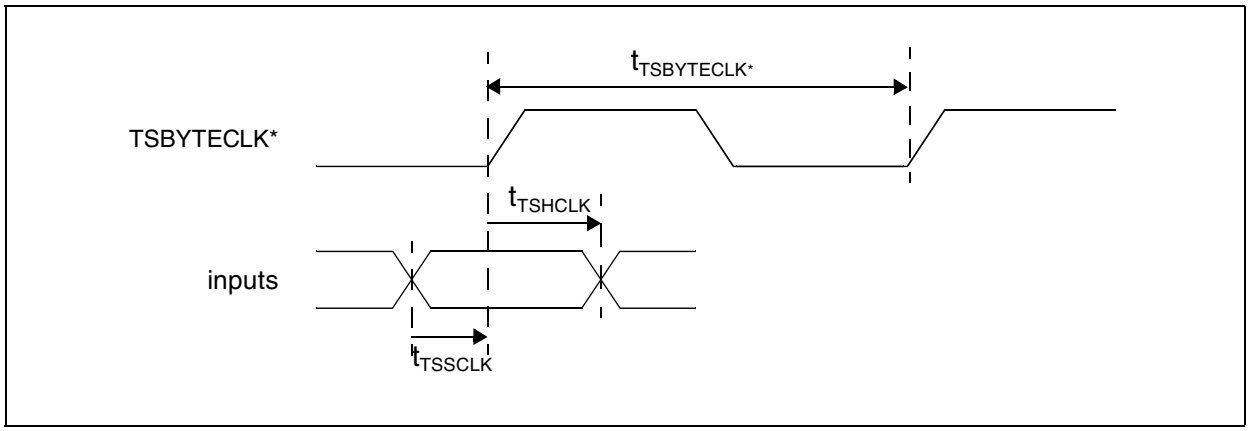


Table 250:

Symbol	Parameter	min	max	Units
Input clock	TSBYTECLK* period	37.04		ns
tDTSCLK	inputs setup with respect to TSBYTECLK rising edge	2.5		ns
tDTHCLK	inputs hold with respect to TSBYTECLK rising sdge	2.5		ns

80.12 Transport stream output AC specification

Output clock: TSBYTECLK2 (falling edge)

Outputs: TSVALID2, TSPKTERR2, TSPKTCLK2, TSDATA2

Figure 278:

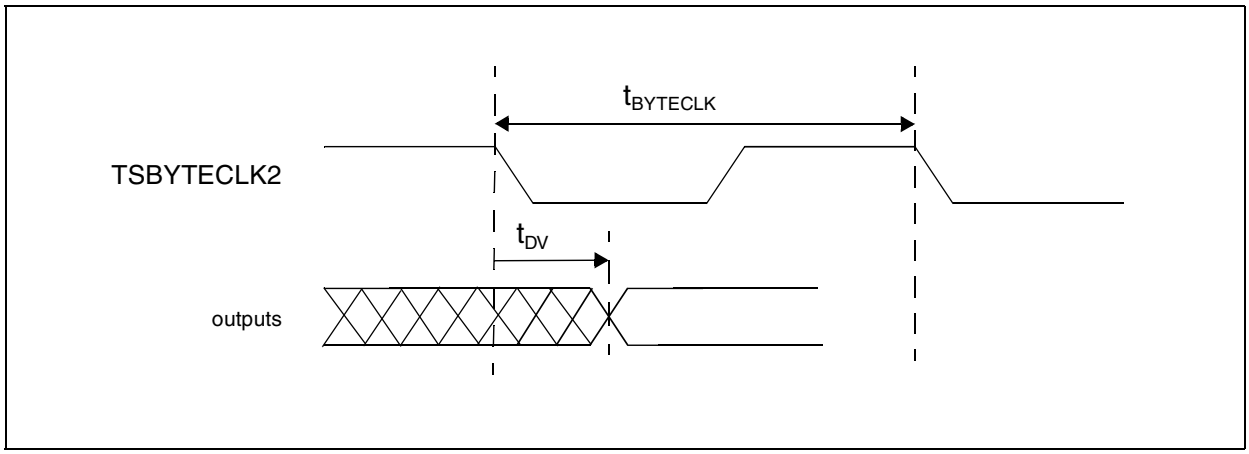


Table 251:

Symbol	Parameter	min	max	Units
tBYTECLK	TSBYTECLK2 period	20		ns
tDV	Output delay with respect to TSBYTECLK2	-3.8	3.0	ns

80.13 EMI

The EMI supports synchronous accesses to SFlash™ and also access to asynchronous peripherals.

80.13.1 Synchronous devices

Outputs are generated and inputs sampled with respect to the rising edge of the bus clock. The static timing characteristics for all EMI pins are shown in Figure 279 and Table 252. EMI all outputs refers to all the EMI_ADDR[*], EMI_DATA[*], EMI_NCS[*], EMI_NBE[*], EMI_NOE, EMI_NLBA, EMI_NBAA, EMI_RDNWR outputs.

EMI all inputs refers to all EMI_DATA[*], EMI_WAIT, EMI_BUS_REQ inputs.

These values are static offsets within a bus clock cycle.

Figure 279: EMI interface timings for synchronous transactions

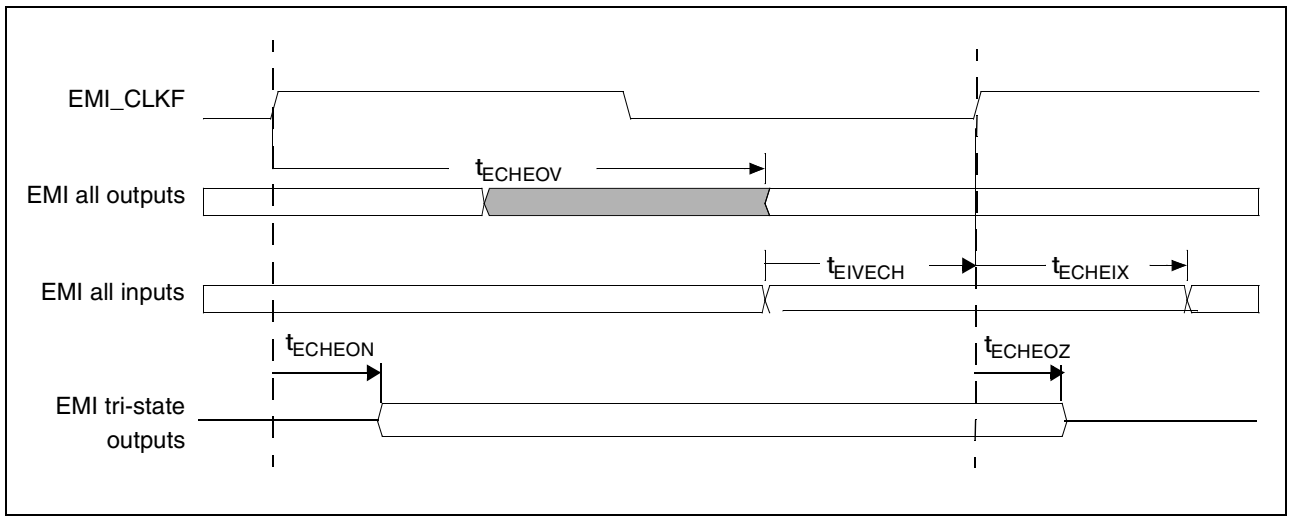


Table 252: EMI synchronous interface parameters

Output values assume a 25 pF external load			
Symbol	**Description**	**Min (ns)**	**Max (ns)**
t_{ECHEOV}	Bus clock rising edge to valid output data	-0.85	2.5
t_{EIVECH}	Input valid to rising clock edge (input set-up time)	6.9	-
t_{ECHEIX}	Rising clock edge to input invalid (input hold time)	-3.7	-
t_{ECHEON}	Rising clock edge to tri-state outputs	-1.7	-
t_{ECHEOZ}	Rising clock edge to outputs from tri-state to valid	-	6.55

Table 252 assumes an external load of 15 pF. If larger loads are used t_{ECHEOV} must be derated accordingly, as in Figure 280.

Bus cycle time is programmed as a division (1, 2 or 3) of the EMI subsystem clock frequency (clk_sys).

80.13.2 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 and strobes) are matched with a tolerance of ±2.25 ns using equal loads (nominal 15 pF). Any additional skew is dependent on the external load spread. Increasing the load delays the signal. The signals are delayed as in Figure 280.

The input latchpoint for a read access is determined by the number of programmed EMI subsystem clock cycles for the latchpoint.

If all signals are equally loaded with 15 pF, the time between the address bus switching and a strobe switching is only *n* programmed phases ±2.25 ns.

Figure 280: Asynchronous access: read

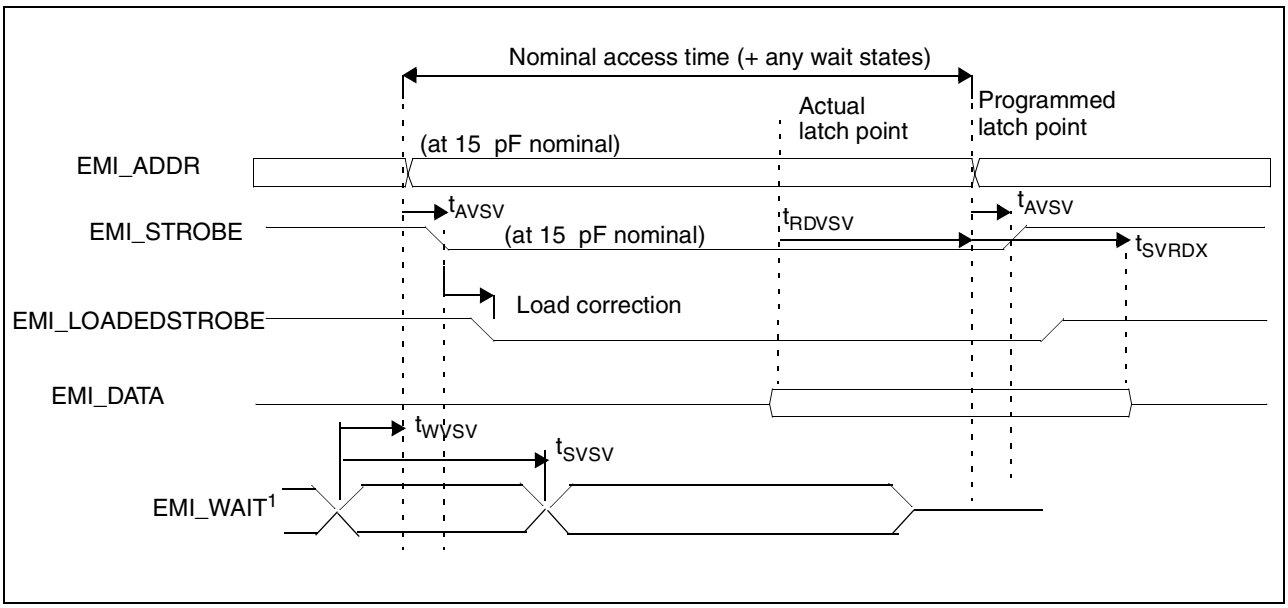


Figure 281: Asynchronous access: write

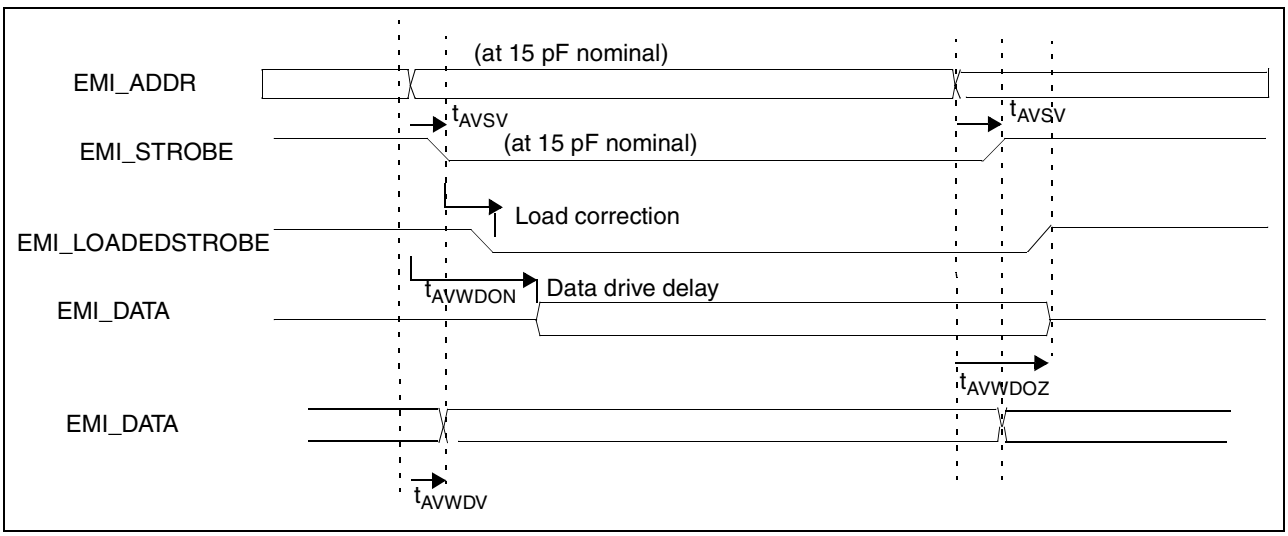


Table 253: EMI asynchronous access mode timings

Values assume a 15 pF external load				
Symbol	**Description**	**Min (ns)**	**Max (ns)**	**Notes**
t_{AVSV}	Address valid to output strobe valid	-2.15	2.25	a,b
t_{RDVSV}	Read data valid to programmed latch point (read set-up time)	8.15	-	c
t_{SVRDX}	Read data hold after programmed latch point (read hold time)	0	-	d
t_{WVSV}	Wait valid to programmed latch point (wait set-up time)	6.25	-	e,f
t_{WVWV}	Minimum wait valid time (before state change)	>1T	-	f
t_{AVWDON}	Address valid to write data valid from tri-state	-	1.35	g
t_{AVWDOZ}	Address valid to write data valid to tri-state	-	0.05	
t_{AVWDV}	Address valid to write data valid from on state	-2.15	2.3	h

a. Skew plus nominal N programmed EMI subsystem clock cycles of strobe delay.

b. Example: if load on address is 15 pF and load on strobe is 30 pF, tAVSV is -2.15 + 0.8 ns minimum, 2.25 + 0.8 ns maximum = -1.35 ns minimum, 3.05 ns maximum.

If the load on the address is 30 pF and the load on the strobe is 15 pF, tAVSV is -2.15 ns - 0.8 ns minimum, 2.25 ns -0.8 ns maximum = -2.95 ns minimum, 1.45 ns maximum.

c. Skew from nominal programmed read latch point.

d. Minimum values are guaranteed by design.

e. Use an output strobe programmed to fall on the rising edge of the EMI subsystem clock, as a reference. EMI_WAIT is sampled at the end of the first and subsequent clock cycles of the access, except for the penultimate cycle.

f. EMI_WAIT is synchronized by the EMI pad logic using the EMI subsystem rising clock edge, so violating set-up time just means that if the signal was sampled low, no wait state is inserted. If sampled high a wait state is inserted. The signal is resampled on the next EMI subsystem clock cycle, where T is the EMI subsystem clock period.

g. Skew from nominal programmed phases of data drive delay.

h. No data drive delay.

80.14 TAP timings

Figure 282: TAP timings

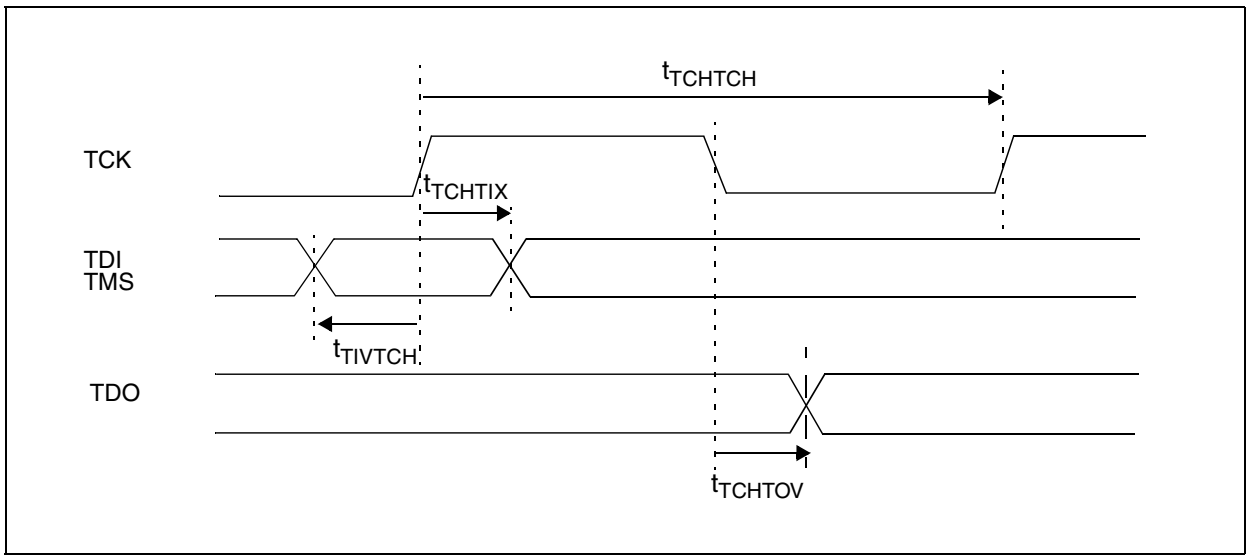


Table 254: TAP timings

Symbol	Parameter	Min	Nom	Max	Units
t_{TCHTCH}	TCK period[a]	20	-	-	ns
t_{TIVTCH}	TAP inputs set-up time[b]	2.2	-	-	ns
t_{TCHTIX}	TAP input hold time[b]	4	-	-	ns
t_{TCHTOV}	TCK low to TAP output valid	-	-	9.4	ns

a. Timing analysis should guarantee that TCK runs at 50 MHz because the boundary scan chain is to be used as a scan chain so speed is very important.

b. Set-up and hold time should be equivalent to 30% of the given TCK period.

There are three other DCU signals, TRST, DCUTRIGGERIN and DCUTRIGGEROUT. They ave dedicated pins on the STi7710. These signals can be considered asynchronous and therefore have no timing constraints. DCUTRIGGERIN needs only to be asserted long enough for the CPU to sample it.

80.15 USB 2.0

Most of the system peripherals are grouped into the communications block and have relatively low bandwidth requirements, but the USB block is separate and has its own STBus initiator. The interface on the STi7710 is compatible with USB 2.0 and USB 1.1. See the USB 1.1 and USB 2.0 specifications.

81 Register index

82 Revision history

Version B	
Throughout	Thoroughly revised according to engineering review, new and updated source material. Data missing in Rev A (“TBA”) inserted. Register summaries completed for all register chapters and register type columns added. Many minor corrections and open issues resolved. Some register lists reordered by offset where possible. Several base addresses corrected.
Base addresses	Chapter deleted (virtual duplicate of Chapter 16: Memory map)
2 Architecture	2.5.2 Programmable transport interface (PTI) MAC match modes deleted.
3 Pin list and alternative functions	Pins L1 and M1 swapped - DAA_C1A and DAA_C2A. Corrected E14 - VDD_RING (1.2V)
4 Connections	Reset information added to tables. CLAMP signal corrected to NOT_CLAMP and description clarified. Table 13:Transport stream: DYC_ signal descriptions added. Table 20:External memory interface (EMI): EMI_PRTSZ signal renamed as EMI_PRTSZ_8NOT16 and decription clarified.
7 Clocks	Table 27: Clock target frequencies and usages completely revised.
12 Resets registers	RESET_CTRL_0 description revised.
14 Interrupt system and 15 Interrupt system registers	Much revised and corrected.
19 External memory interface (EMI)	Several changes and corrections.
20 EMI registers	Reset values corrected. Bank 5 described correctly as virtual. Buffer registers contain top rather than base addresses.
21 Local memory interface (LMI)	SDRAM technology supported missing: 512 Mbits. LMICLK output can be also 200 Mz.
22 LMI registers	LMI_STR register description updated.
29 Infrared transmitter/receiver and 30 Infrared transmitter/receiver registers	Updated - FIFO depth increased from 4 to 8 words Some registers added.
45 Transport stream merger and router	
40 USB 2.0 host (USBH) registers	Register offsets changed, registers added.
37 Programmable I/O ports	Parallel IO changed to Programmable IO
43 PWM and counter modules and 44 PWM and counter module registers	Completely revised
45 Transport stream merger and router	45.4 Input examples Figure removed - DVR satellite/cable system - nonshared D1
54 MPEG video decoder	Section 54.5.4: *Main and secondary reconstruction* added, explaining decimation.
58 Video output stage (VOS)	Diagrams clarified.
59 Video output stage (VOS) registers	Registers updated.

60 Compositor	Capture pipeline sections removed.
61 Compositor registers	Section 61.1: *Introduction on page 620*: References to CAP removed. All capture pipeline registers (GAM_CAP_*) removed.
64 High-definition multimedia interface (HDMI) and 65 HDMI registers	Chapters added
69 Digital encoder (DENC) registers	Many registers updated; all offsets multiplied by 4.
70 Teletext DMA and 71 Teletext DMA registers	Chapters added
73 HD triple video DACs	SVM_HD removed
77 Audio decoder registers	Added descriptions for several registers. Added registers: HOST_DM_COEFT_n, AUD_POSTPROC, AUD_SRS_CALL, AUD_HEADROOM.
79 Electrical specifications	Chapter added
80 Timing specification	Chapter added
Version A	
	First draft

Confidential