STx5119

Low-cost interactive set-top box decoder

ADVANCE DATASHEET

Confidential

Enhanced ST20 32-bit VL-RISC CPU
- 200 MHz, single cycle cache, 4 Kbyte instruction cache, 4 Kbyte data cache, 2 Kbyte SRAM

Unified memory interface
- up to166 MHz,16-bit wide SDR SDRAM interface

Programmable flash memory interface
- four separately configurable banks, 8/16-bits wide
- SRAM, peripheral, Flash, SFlash™ support
- support for low-cost DVB-CI

Programmable transport interface (PTI)
- single transport stream input
- support for DVB transport streams
- integrated DVB, descrambler

MPEG-2 MP@ML video decoder
- fully programmable horizontal and vertical SRCs

Graphics/display
- blitter based display compositor
- 8 bpp CLUT graphics, 256 x 30 bits (AYCbCr) CLUT entries. 16 bpp true color graphics, RGB565, ARGB1555 formats. Link-list control
- 2D paced blitter engine with fill function

PAL/NTSC/SECAM encoder
- RGB, CVBS, Y/C and YUV outputs with four 10-bit DAC outputs. RGB/CVBS or YUV/CVBS or YC/CVBS.
- encoding of CGMS, Teletext, WSS, VPS, close caption

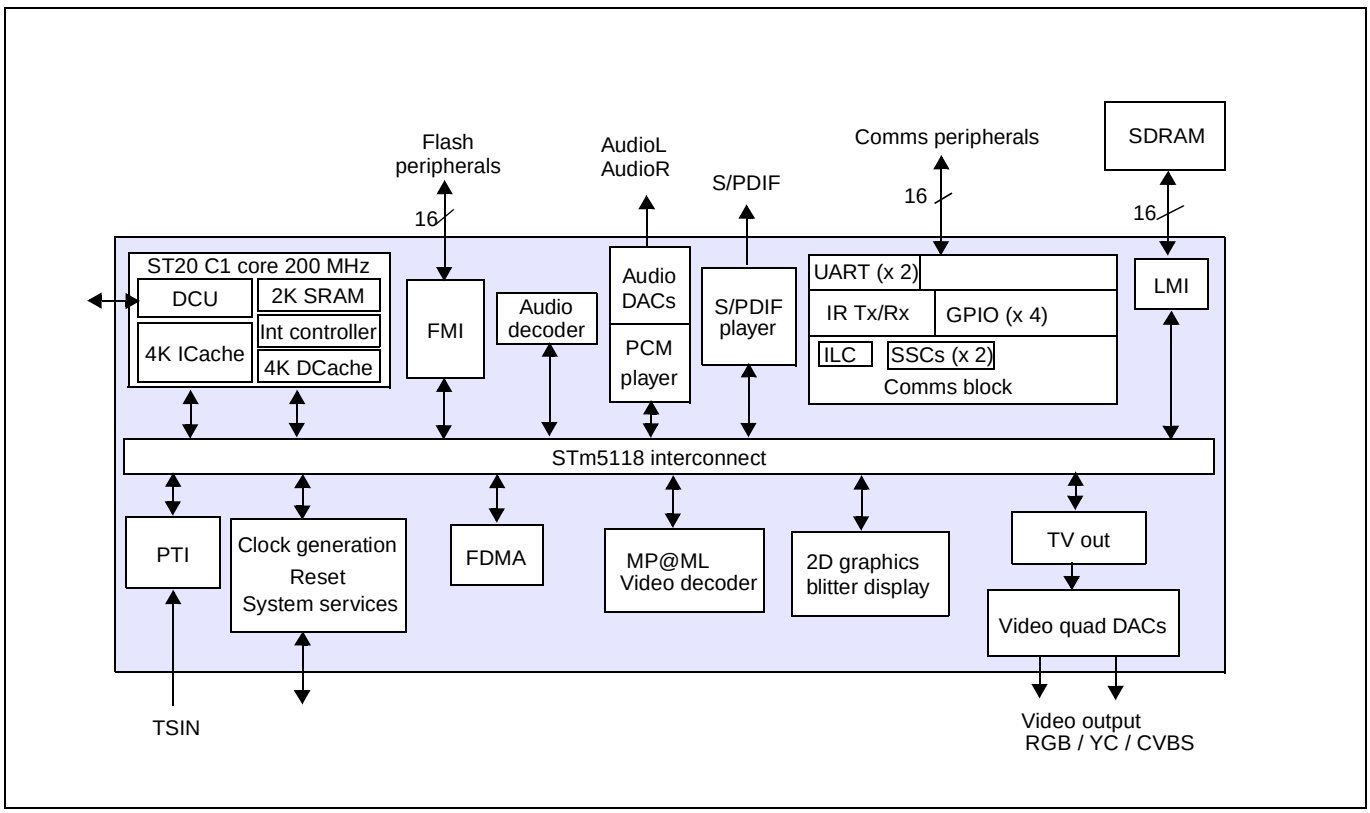
Audio subsystem
- MPEG-1 layers I/II
- simultaneous MPEG audio decode on audio DACs and output of Dolby streams on S/PDIF
- IEC958/IEC1937 digital audio output interface
- integrated stereo audio DAC system

Central DMA controller

On-chip peripherals
- two ASCs (UARTs) with Tx and Rx FIFOs
- three 8-bit banks of parallel I/O and one 7-bit bank
- one smartcard interface and clock generator
- two SSCs for I^2C/SPI master/slave interfaces
- infrared transmitter/receiver
- integrated VCXO
- low-power / RTC / watchdog controller

JTAG/TAP interface

Package 24 mm x 24 mm LQFP216



Contents

1 Applications

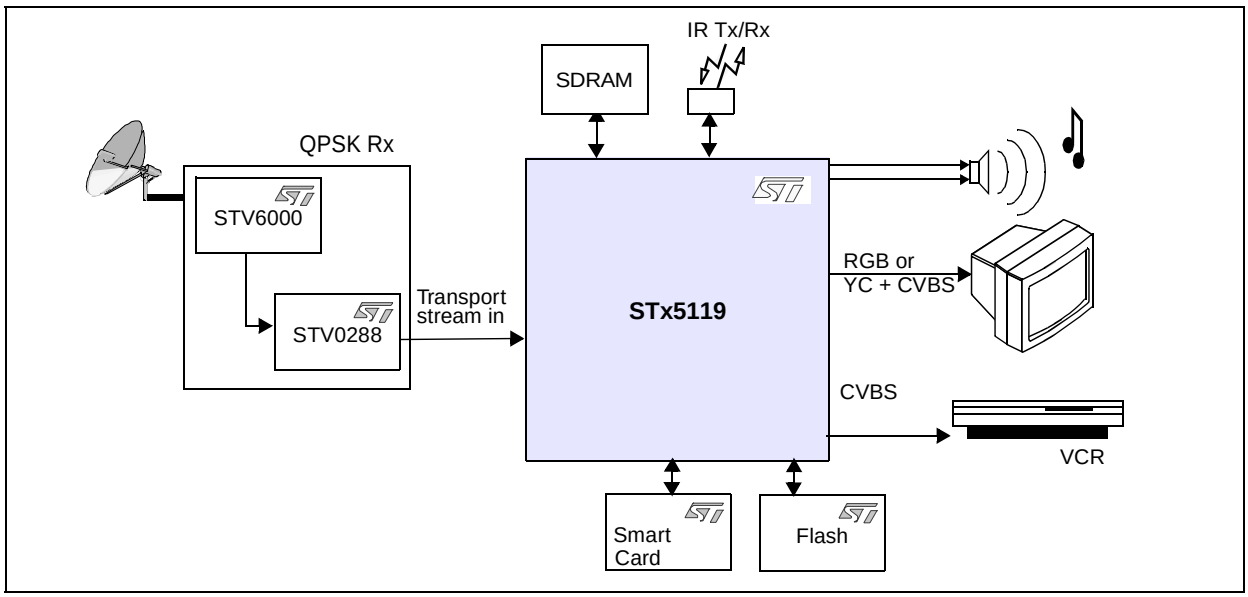
The STx5119 is the latest in the family of Omega2 set-top box ICs providing a high performance, low cost system on a chip for MPEG processing in cable, satellite or digital terrestrial STBs.

The STx5119 delivers enhanced performance with respect to previous generation devices. Main memory is based upon a single 16-bit external SDRAM.

The display architecture of the device is based on a high performance blitter engine which supports CLUT8 and RGB16 formats for video and OSD/graphics displays.

The STx5119 can be used for basic set-top boxes. A typical application is shown in Figure 1.

Figure 1: Free to air satellite



2 Audio and video summary specification

Table 1: Summary specification

Video decoder	
Feature	**Description**
Video decode display	Based on the Omega2, provides a memory to memory decode into YC 4:2:0 macroblock format, provides simple resizing based on x2 and x0.5.
Bit streams accepted	MPEG-1 video (ISO/IEC 11172-2), MPEG-2 video (ISO/IEC 13818-2) MPEG-2 Packetized elementary stream (PES) format as defined by ISO/IEC 13818-1 MPEG-1 ISO/IEC 11172-1 packets
MPEG-2 profiles/levels supported	Main profile at main level (MP@ML), main profile at low level (MP@LL) Simple profile at main level (SP@ML)
Maximum picture size	Width: 720 x 576 pixels = 45 x 36 Number of macroblocks: 45 x 36 = 1620
Motion vector range	MPEG-1: -1024 to 1023 (full pel), -512 to 511.5 (half pel) horizontal and vertical MPEG-2: -1024 to 1023.5 horizontal and -128 to 127.5 vertical
PTI compressed data input	Serial peak input rate: 100 Mbit/s
SDRAM LMI interface	16-bit wide SDRAM interface. Peak bandwidth 532 Mbyte/s (133 MHz). Supports 64-Mbit, 64-Mbit, 64-Mbit, 128-Mbit, 256-Mbit, or 512-Mbit SDRAM. Fully cacheable address space for data and instructions, with cacheability controlled in 512 Kbyte blocks for up to 8 Mbytes.
Start code detection	Automatic detection of start codes (of picture layer and above) to enable the microcontroller to access header data Counters provided for time-stamp tracking
Decoding pipeline	Instruction register sets up each picture and defines pipeline operation Double-buffered quantization matrices enable loading of new tables concurrently with decoding
Error concealment	Automatic concealment of errors detected by VLD and decoding pipeline by macroblock copy

Table 1: Summary specification

Display	
Feature	**Description**
Video clock	27 MHz nominal
Video output	External pel clock Horizontal/vertical synchronization provided by internal digital encoder or external source Interlaced output 3:2 pull-down operation supported On-chip up-/down-sampling with anti-aliasing filter Vertical chroma reconstruction or luma filtering up to 5-tap filter
Pan and scan vectors	Horizontal: maximum vector size: 2047 pels, resolution: 1/8 pel Vertical: maximum vector size: 1022 lines, resolution: 1 line
On-screen display (OSD)	Blitter-based composition The OSD plane is managed as a set of horizontal bands with a specification comprising configuration, bitmap and, for CLUT formats, palette information for each region. The OSD operates in one of two modes, palette mode or true color mode. **Palette mode**: Each region can be independently specified with a resolution of 8 bpp. Regions are frame based. Each region palette can support up to 256 colors with up to 24 bits resolution per color entry. **True color mode**: Each region can be independently specified with a 16 bpp resolution in one of the following direct color formats: RGB565 and ARGB1555. A vertical interfield, antiflicker filter is provided to reduce flicker on interlace displays. It is available for both palette and true color modes.
Audio decoder	
Bit streams accepted	MPEG-1, layers I and II elementary streams
Performance	ISO/IEC 11172-3 Layers I and II All MPEG input bit rates supported with sampling rates of 32 kHz, 44.1 kHz and 48 kHz, free format at 32 kHz and 48 kHz sampling rates Decodes in single channel, dual channel, stereo, or joint stereo modes
Error concealment	Automatic error concealment on CRC or synchronization error detection
General	
Support for A/V sync	PTS/DTS extraction from MPEG packet layers with automatic association

3 Architecture features

3.1 Introduction

The STx5119 is a low-cost Omega2 MPEG device that delivers highly integrated features that provide an overall system cost reduction. The device implements a fully unified SDRAM based memory architecture and integrates the Omega2 video decoder cell together with a blitter engine and a multichannel DMA controller to provide enhanced performance for graphics and real-time stream transfers.

Transfer of data such as pixmaps, audio streams, stills and PES can be performed efficiently using the STx5119's DMA.

A true-color mode provides OSD graphics allowing the display of RGB16 formats: RGB565 and ARGB1555. This directly supports up to 65,536 colors in a region.

3.2 Omega2 (STBus) interconnect

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

3.3 Processor core

The ST20-C106 processor core is composed of the ST20C1+ CPU running at 200 MHz, a diagnostic controller unit (for low intrusion, real-time debugging), memory (4 Kbyte instruction cache, 4 Kbyte data cache and 2 Kbyte SRAM) and a 16 priority-level interrupt controller.

3.4 Memory subsystem

The STx5119 has a memory interface and a peripheral/flash interface.

The STx5119's local memory interface is used for all data requirements in unified memory applications including graphics, video and audio buffers. It provides 16-bit wide SDRAM support at up to 166 MHz.

The FMI provides support for 16-bit wide peripherals, flash and synchronous flash.

Instructions can execute in place from flash/SFlash™ on the FMI or can be copied to SDRAM on the LMI. The following sections provide an overview of the different memory interfaces.

Local memory interface (LMI)

The LMI is a 16-bit wide SDRAM interface operating at up to 166 MHz. It supports 64-Mbit or 128-Mbit SDRAM. The LMI provides a fully cacheable address space for data and instructions, with data cacheability controlled in 512 Kbyte blocks for up to 8 Mbytes.

Flash memory interface (FMI)

The FMI provides a glueless interface to SRAM, flash, SFlash and peripherals, in up to four configurable banks over a 16-bit wide interface. Bus cycle strobe timings can be programmed from 0 to 15 phases for slower peripherals. Support is provided for control of DVB-CI.

3.5 Transport stream processing

The STx5119 supports single transport stream input.

It is possible to support DVB-CI configurations as shown in Figure 2 and Figure 3.

Figure 2: Dual DVB-CI support

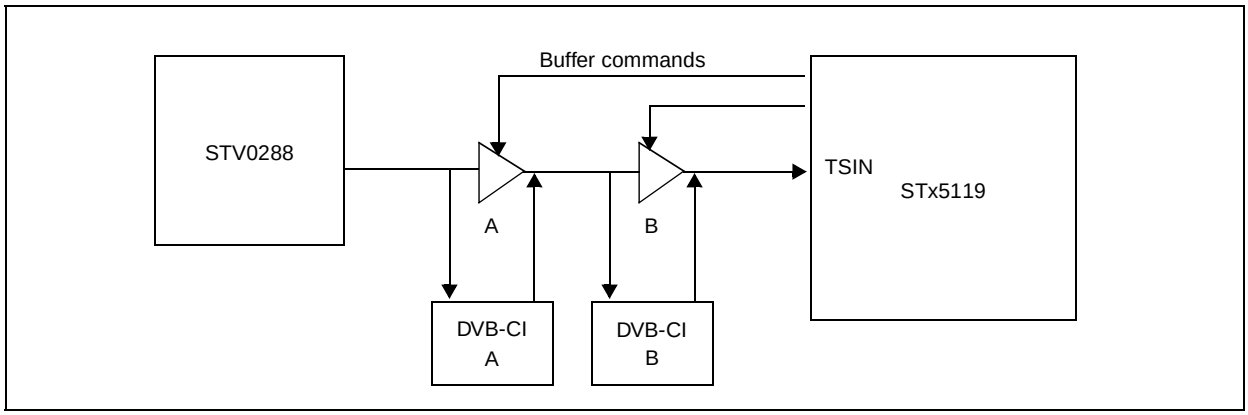
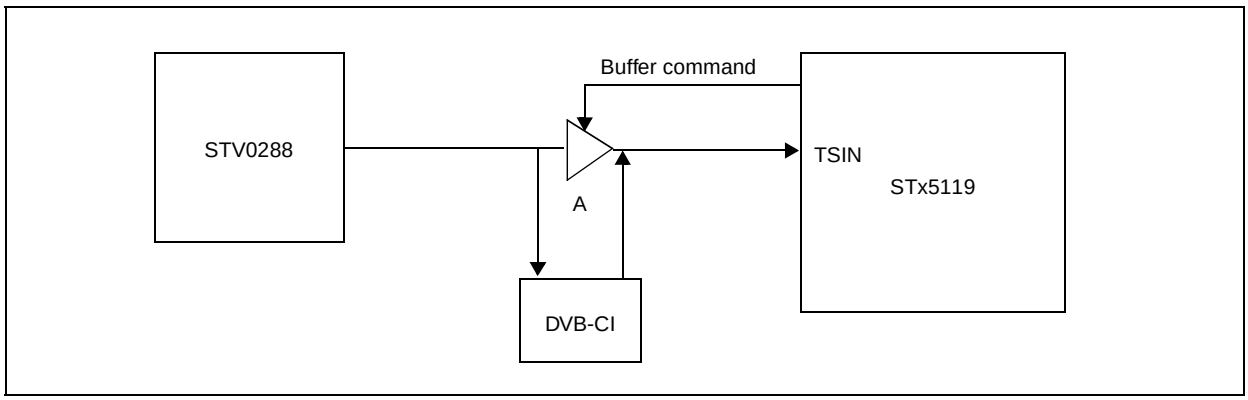


Figure 3: Single DVB-CI support



Programmable transport interface (PTI)

The PTI performs transport-stream descrambling, demultiplexing and data filtering. PES data is transferred by DMA to audio and video decoders via circular buffers. Section data is transferred by DMA to separate buffers for further processing by the CPU.

DVB transport streams can be handled by the PTI with data rates up to 100 Mbit/s.

The PTI performs PID filtering to select audio, video and data packets to be processed. 48 PID slots can be supported by the PTI.

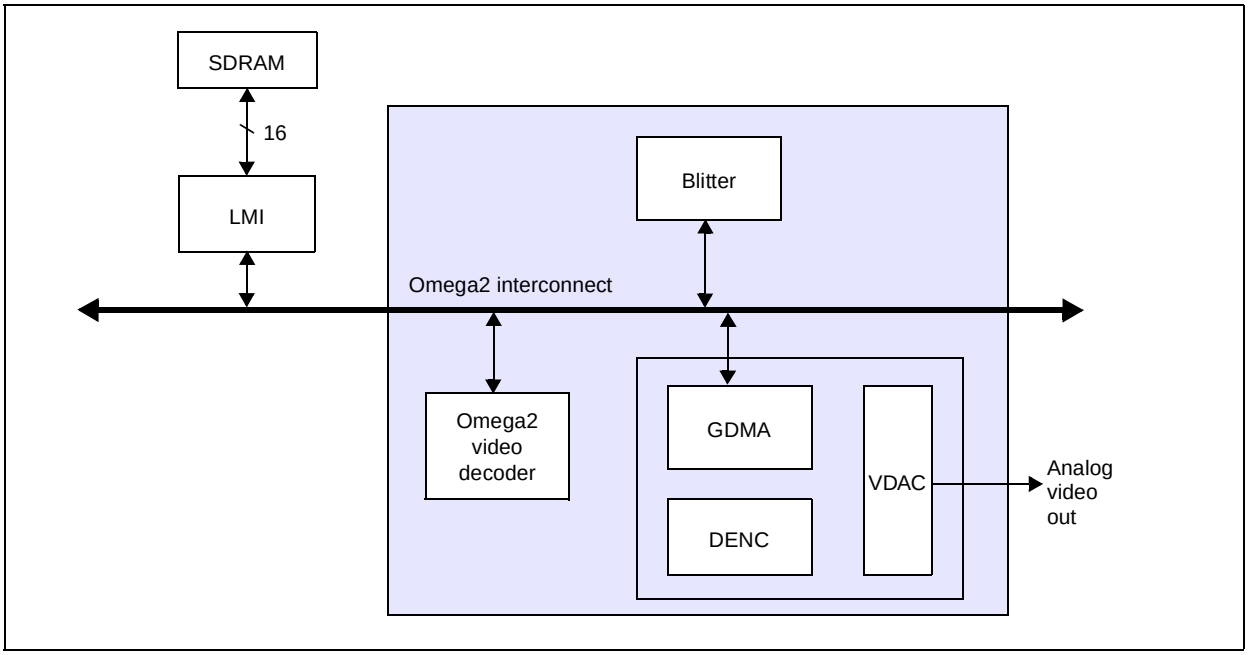
The PTI cipher descrambles DVB-CSA streams.

The PTI has a section filter core that filters DVB standard sections using 32 x 8-byte filters.

3.6 MPEG graphics and display processing

The MPEG graphics and display architecture shown in Figure 4 provides the graphics, video-stream processing and display capabilities of the STx5119.

Figure 4: Graphics and display subsystem



Video decode

The video decoder is based on the Omega2 cell and provides a memory to memory decode into YC 4:2:0 macroblock format. It is also able to provide simple resizing based on x2 and x0.5.

3.7 Graphics and display

Display

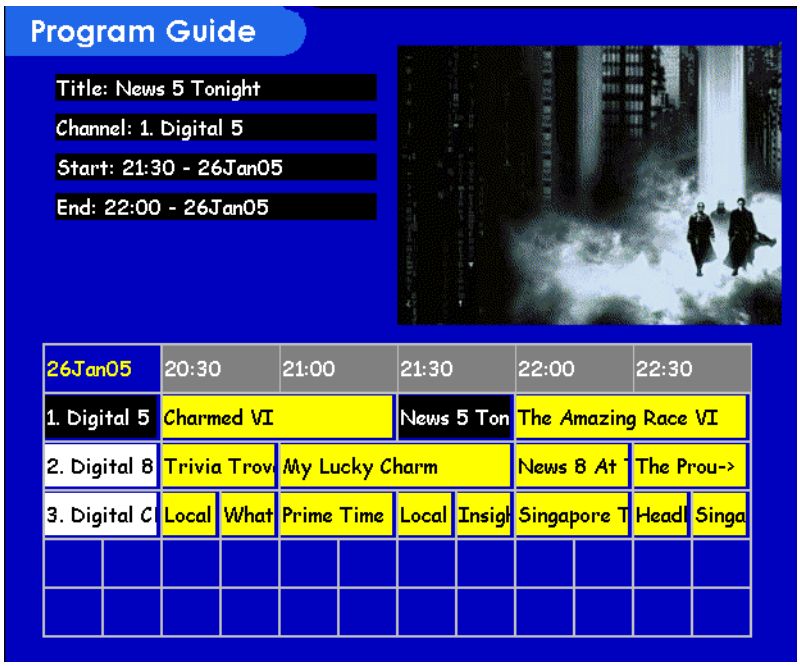
The STx5119 uses a blitter based display architecture, providing improved graphics and increased flexibility for building applications as illustrated in Figure 5

The following features are supported:

- SMART GUI,
- Picture in Graphic,
- mosaic channels presentation.

The composition is assisted by the 2D graphic hardware accelerator.

Figure 5: Supported applications



Graphics

The graphics operates in one of two modes:

Palette mode Each region can be independently specified with a resolution of 8 bpp. Regions are frame based. Each region palette can support up to 256 colors with up to 24 bits resolution per color entry.

True color mode Each region can be independently specified with a 16 bpp resolution in either RGB565 or ARGB1555 direct color formats.

A vertical inter-field, anti-flicker filter is provided to reduce flicker on interlace displays. It is available for both palette and true color modes.

3.8 Digital encoder

The digital converter processes YCbCr 4:4:4/4:2:2 from main memory, which the blitter has generated, and produces a standard analog baseband PAL/SECAM/NTSC signal, and component (YUV/RGB).

The digital encoder can handle interlaced mode in all standards. ITU-T 601 aspect ratio displays can be supported in all standards. The digital encoder performs closed-caption, CGMS, WSS, Teletext and VPS encoding and allows Macrovision™ 7.01/ 6.1 copy protection.

One integrated quad-DAC provides four analog TV outputs, on which it is possible to output either of the following:

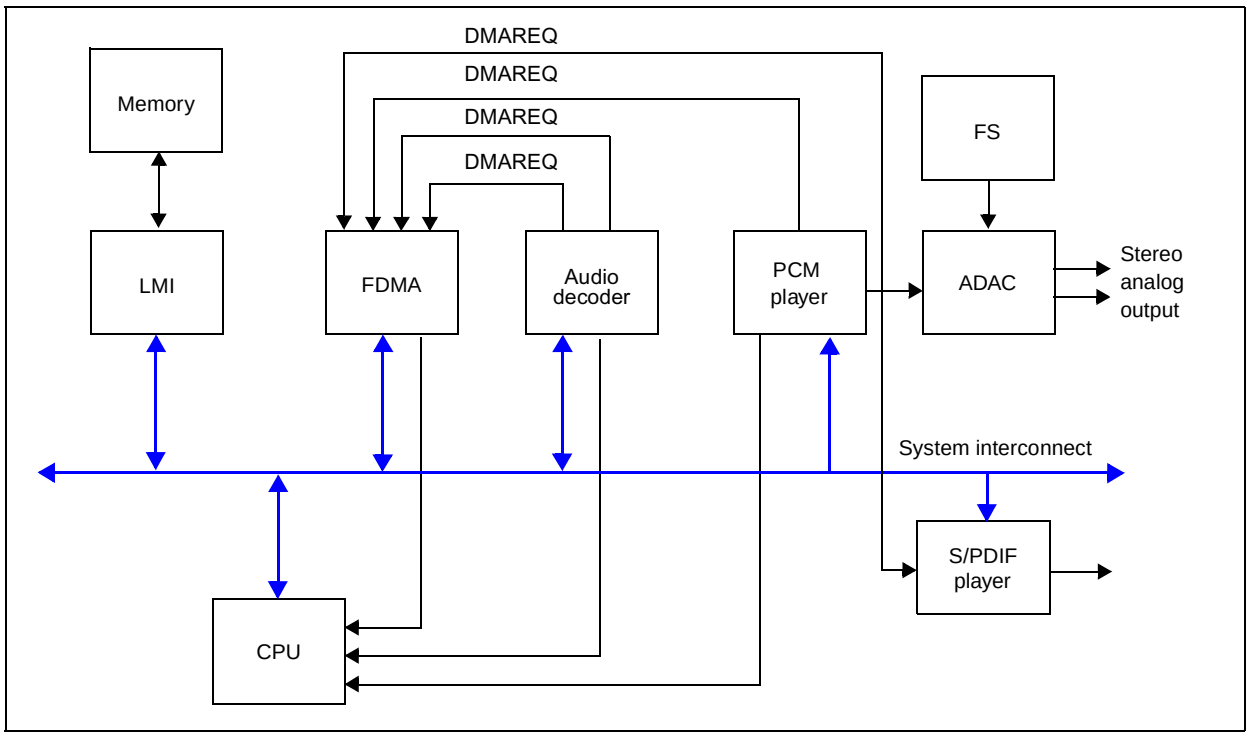
- CVBS + RGB,
- CVBS + YUV,
- S-VHS (Y/C) + CVBS1 + CVBS2.

3.9 Audio subsystem

The audio subsystem illustrated in Figure 6 supports the following features:

- audio decoding of MPEG-1 layers I and II,
- simultaneous MPEG-1 audio decoding and the compressed output of Dolby streams on the S/PDIF,
- audio sample rates of 32 kHz, 44.1 kHz and 48 kHz.

Figure 6: Audio subsystem



The audio subsystem consists of the following units:

- audio decoder,
- S/PDIF player,
- PCM player,
- integrated 24-bit stereo audio DAC system,
- audio/digital frequency synthesizer (generates the PCM and sample rate clocks).

3.10 Central DMA controller

The STx5119 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 external peripherals and for audio and video stream transfers within the STx5119.

3.11 Internal peripherals

The STx5119 has many dedicated internal peripherals for digital TV receiver applications, including:

- two ASCs (UARTs), that are generally used as smartcard controllers or for modem application,
- two SSCs for I²C master/slave interfaces, with SPI support,
- four GPIO ports,
- Infrared blaster/decoder interface module,
- DVB-CI support,
- fully integrated digital VCXO,
- interrupt level controller,
- low-power RTC watchdog controller,
- DCU toolset support,
- JTAG/TAP interface.

3.12 Clock generation

All system clocks are generated using the clock generator block. This contains two high-frequency PLLs that are divided down to produce a series of phase-related programmable clock channels.

The STx5119 has a clock master. The Flash clock output may be phase aligned to optimize the external bus performance of the FMI.

VCXO functionality has been integrated using a special purpose frequency synthesizer, thus removing the need for an external varactor diode or VCXO module.

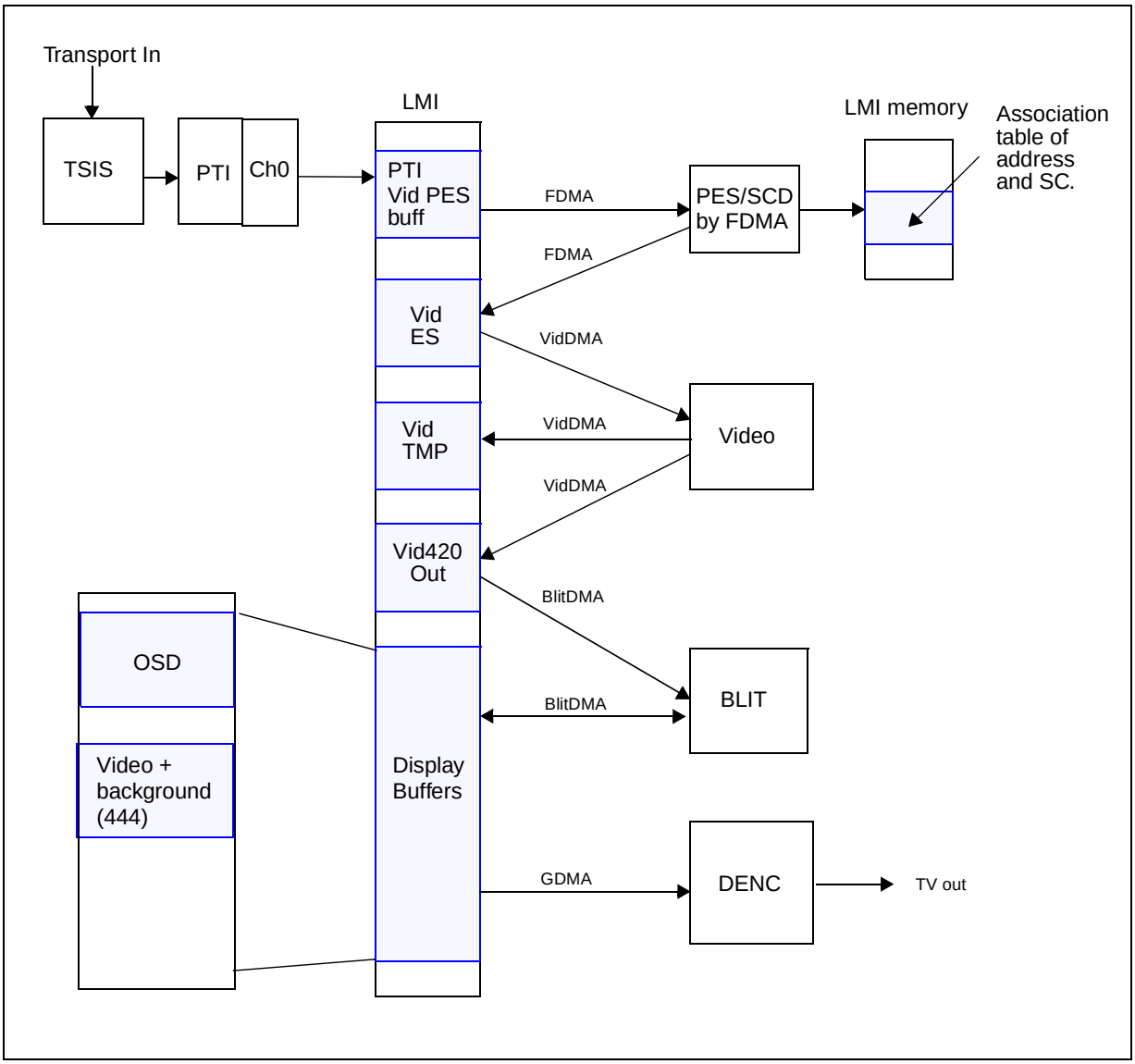
4 Dataflow

This chapter describes the system dataflow for AV reproduction from the front-end compressed stream, through the AV decoders and various buffers to the final AV presentation.

In the following dataflow, the programmable transport interface (PTI) performs the transport stream deMUX. The channel 0 DMA generates audio and video circular buffers in main memory.

4.1 Video decode flow

Figure 7: Video decode flow



The PTI channel 0 circular video buffer is accessed by the FDMA which is a linear linked list machine. The synchronization is described in the following paragraph.

The host CPU is interrupted by a programmed timer event from system services, for example every 10 ms. The interrupt routine reads the ch0 write pointer from the PTI; using this, and the previous write pointer, a single pass linear access is activated using a dedicated process (PPSCD) running on the FDMA (with hardware assist) to perform PES parsing and start code detect.

The FDMA PPSCD process generates an elementary stream (ES) buffer in main memory and an association table which can be placed in external memory or local SRAM.

Each entry of the PPSCD table contains the following information:

- start code value,
- PTS if present,
- start code offset in PES buffer,
- start code address in the ES buffer.

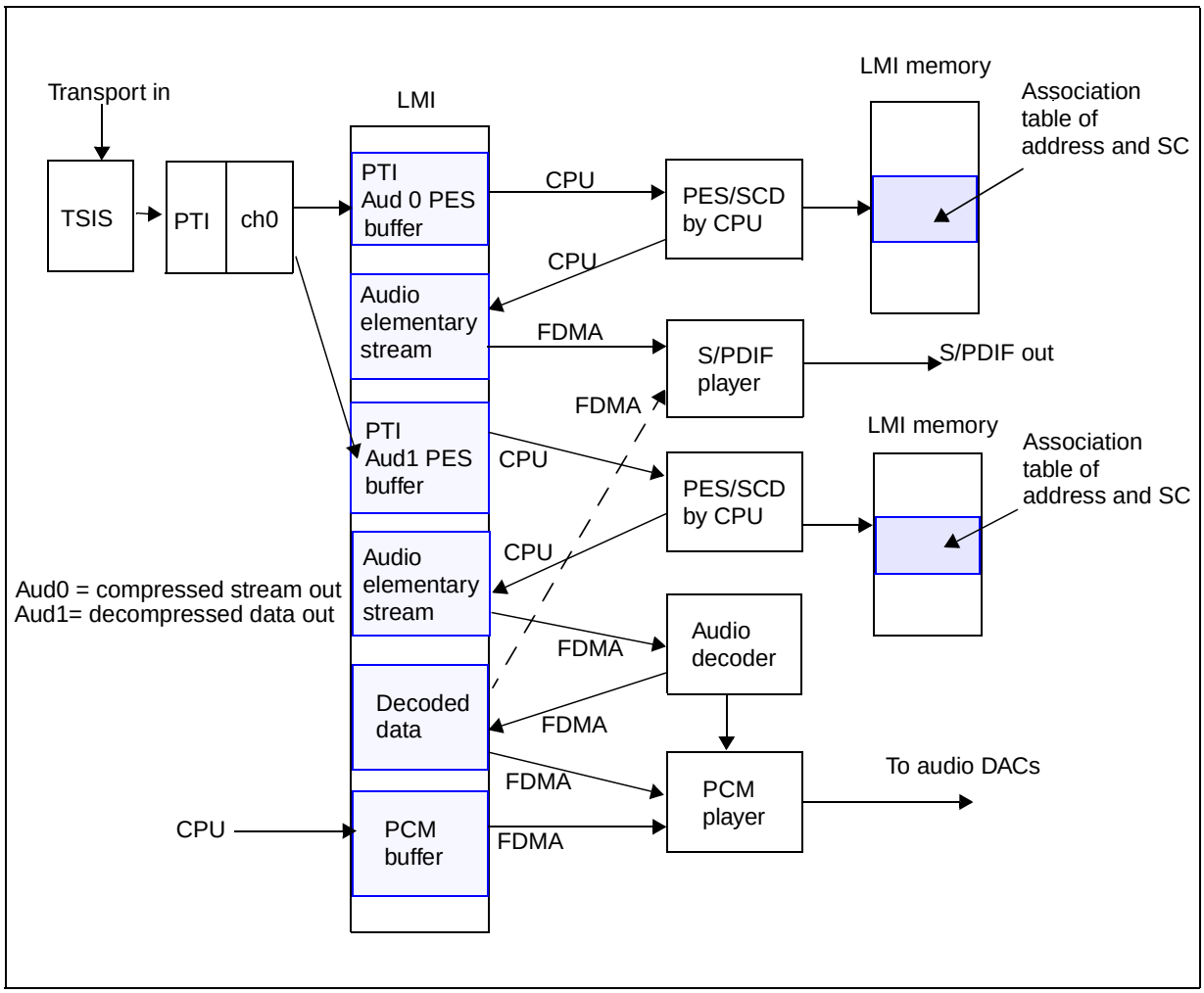
This table is accessed by the video driver running on the host CPU. The ES video buffer is accessed directly using the linear DMA engine inside the video decoder. This block generates several intermediate buffers during the prediction and reconstruction phases of decode before generating a final video buffer in YCbCr 4:2:0 format.

The video buffer is read directly by the blitter which performs multiple passes to format convert and blend the video buffer with a background and OSD overlay, before rendering a final display buffer in YCbCr 4:2:2 format.

The picture composed by the blitter is always frame based. The final presentation buffer is read by the GDMA which drives the digital encoder to produce composite video output. The GDMA should be programmed to access the presentation buffer field based.

4.2 Standard audio decode flow

Figure 8: Standard audio decode flow



The PTI fills the circular audio PES buffers in the external memory. The PES parsing and start code detection is performed by the CPU or by FDMA. The PES parsing and start code detection process is the same as the process described in the video decode data flow.

The elementary stream, corresponding to the bypass (Dolby stream), is moved to the S/PDIF player by the FDMA. The FDMA performs appropriate formatting for the S/PDIF player. The S/PDIF player outputs the data on the S/PDIF output.

The audio elementary stream to be decoded is moved to the audio decoder. The decoded data is then read by the FDMA and appropriately buffered in the external memory. The decoded PCM stream is then fed to the PCM player. The output of the PCM player drives the audio DACs.

The FDMA can also move the decoded audio, which is buffered external to the audio decoder, to the S/PDIF player after appropriate formatting. The formatting consists of the addition of channel status, user data and validity flag.

The CPU generates a PCM buffer which can be used to generate the beep tones or sync noise. The contents of the buffer are moved to the PCM player by the FDMA. The quad frequency synthesizer provides the PCM and the S/PDIF clocks.

4.3 Real-time traffic

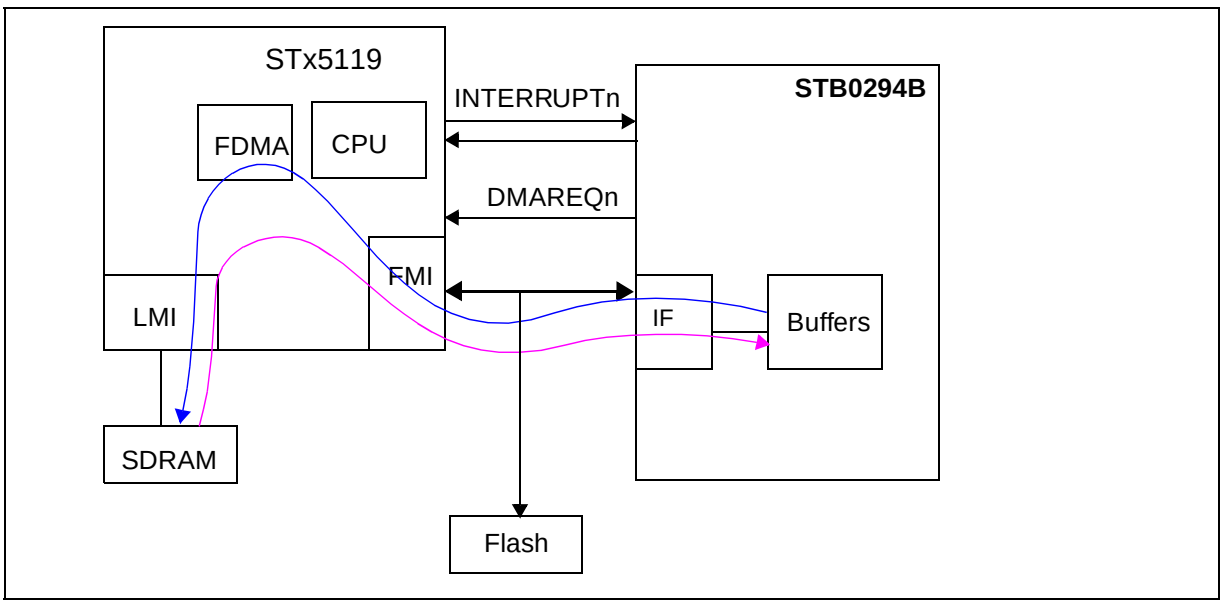
4.3.1 Paced external

Cable front-end using the STB0294B device

The STx5119 accesses the STB0294B via a target interface connected to the FMI. Datagrams may be moved to and from internal buffering within the STB0294B using the FDMA. Flow control is achieved by using standard DMAREQ pacing signals that are routed to the FDMA within the STx5119.

Note: The STx5119 and FMI external bus is shared with a flash memory.

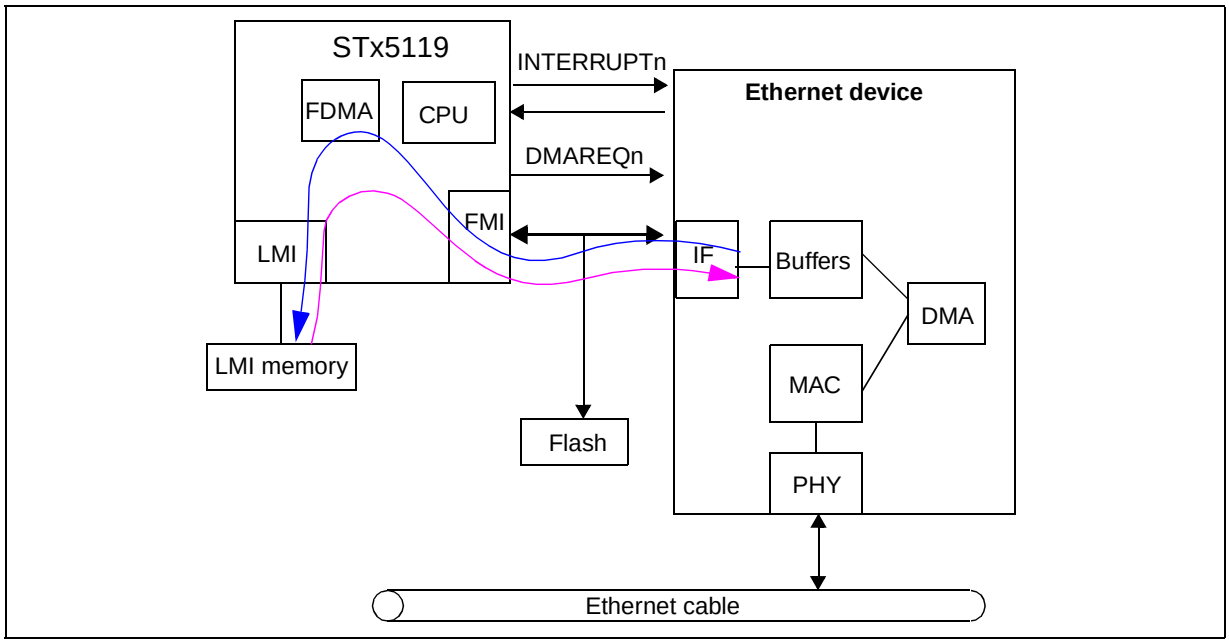
Figure 9: Cable front-end device



4.3.2 External ethernet

External ethernet devices are mapped to the FMI in the same way as the STB0294B. The STx5119 FDMA fetches from and writes to buffers in the external ethernet device. A service request is made by asserting an interrupt to the STx5119.

Figure 10: External ethernet device



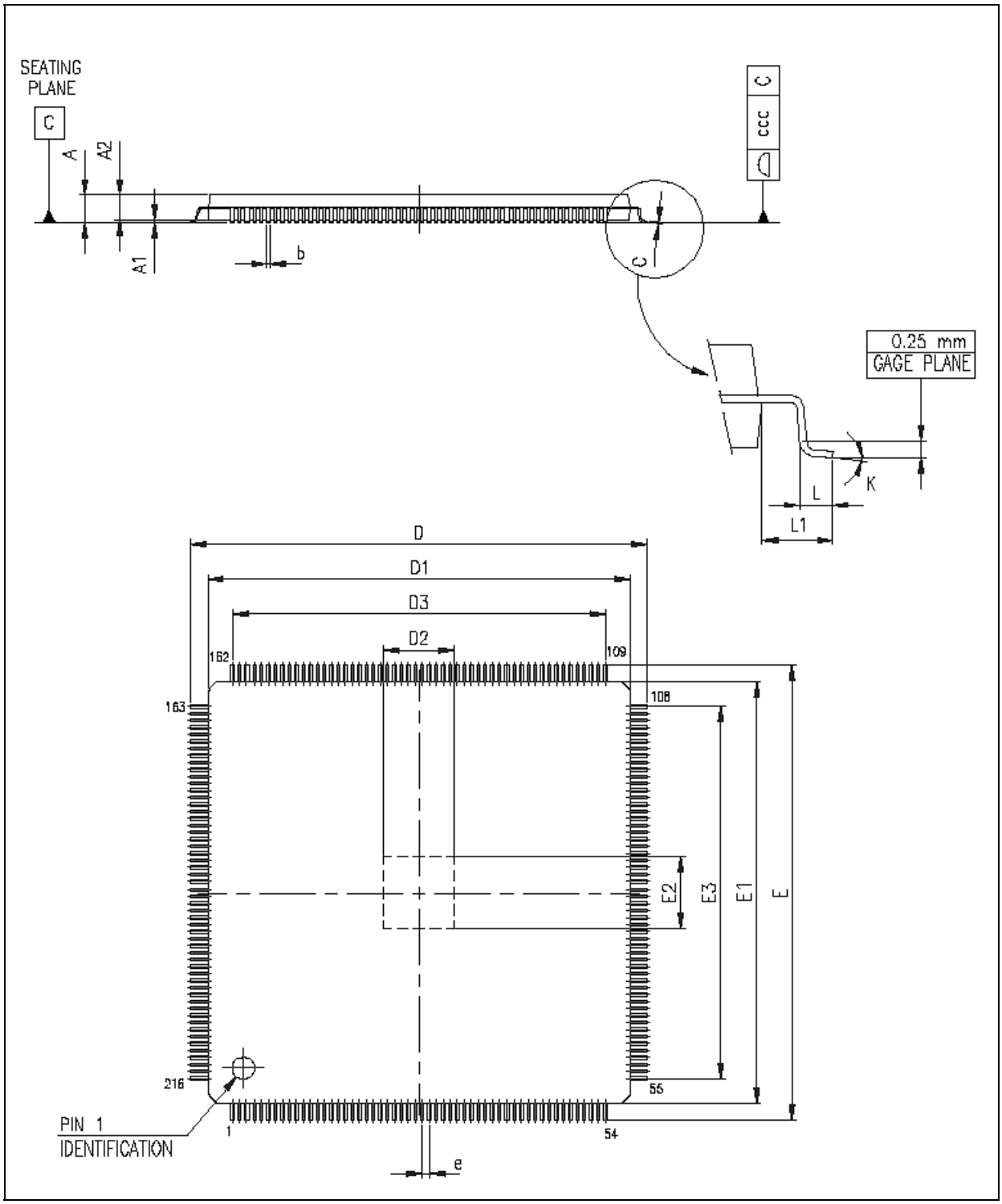
5 Package specifications

The STx5119 is packaged in a 216-pin LQFP-EP (exposed pad) package. Table 2 gives the values of the dimensions marked in Figure 11.

Table 2: Package dimensions

Dimension	Databook (mm)			Drawing (mm)		
	Minimum	Typical	Maximum	Minimum	Typical	Maximum
A	-	-	1.600	-	-	1.600
A1	0.050	-	0.150	0.050	0.100	0.150
A2	1.350	1.400	1.450	1.350	1.400	1.450
b	0.130	0.180	0.230	0.130	0.180	0.230
C	0.090	-	0.200	0.090	0.127	0.200
D	25.800	26.000	26.200	25.900	26.000	26.100
D1	23.800	24.000	24.000	23.800	24.000	24.000
D2	2.000	-	-		8.000	
D3	-	21.200	-	-	21.200	-
E	25.800	26.000	26.200	25.900	26.000	26.100
E1	23.800	24.000	24.200	23.800	24.000	24.200
E2	2.000	-	-		8.000	
E3	-	21.200	-	-	21.200	-
e	-	0.400	-	-	0.400	-
L	0.450	0.600	0.750	0.450	0.600	0.750
L1	-	1.000	-	-	1.000	-
k	0d	3.5d	7d	0d	3.5d	7.d
ccc	-	-	0.080	-	-	0.080

Figure 11: STx5119 LQFP-EP package



5.1 Exposed pad description

The TQFP/LQFP package with exposed pad is made with a deep down set leadframe so that the bottom side of the die paddle is flush with the bottom side of the package and is exposed. This allows the exposed pad to be soldered directly to the PCB and facilitates high heat dissipation. Although direct soldering of the exposed pad to the PCB is the best way to get effective power dissipation, thermal glue can be used as well.

6 Pin information

6.1 Pin data

Figure 12: Pin-out

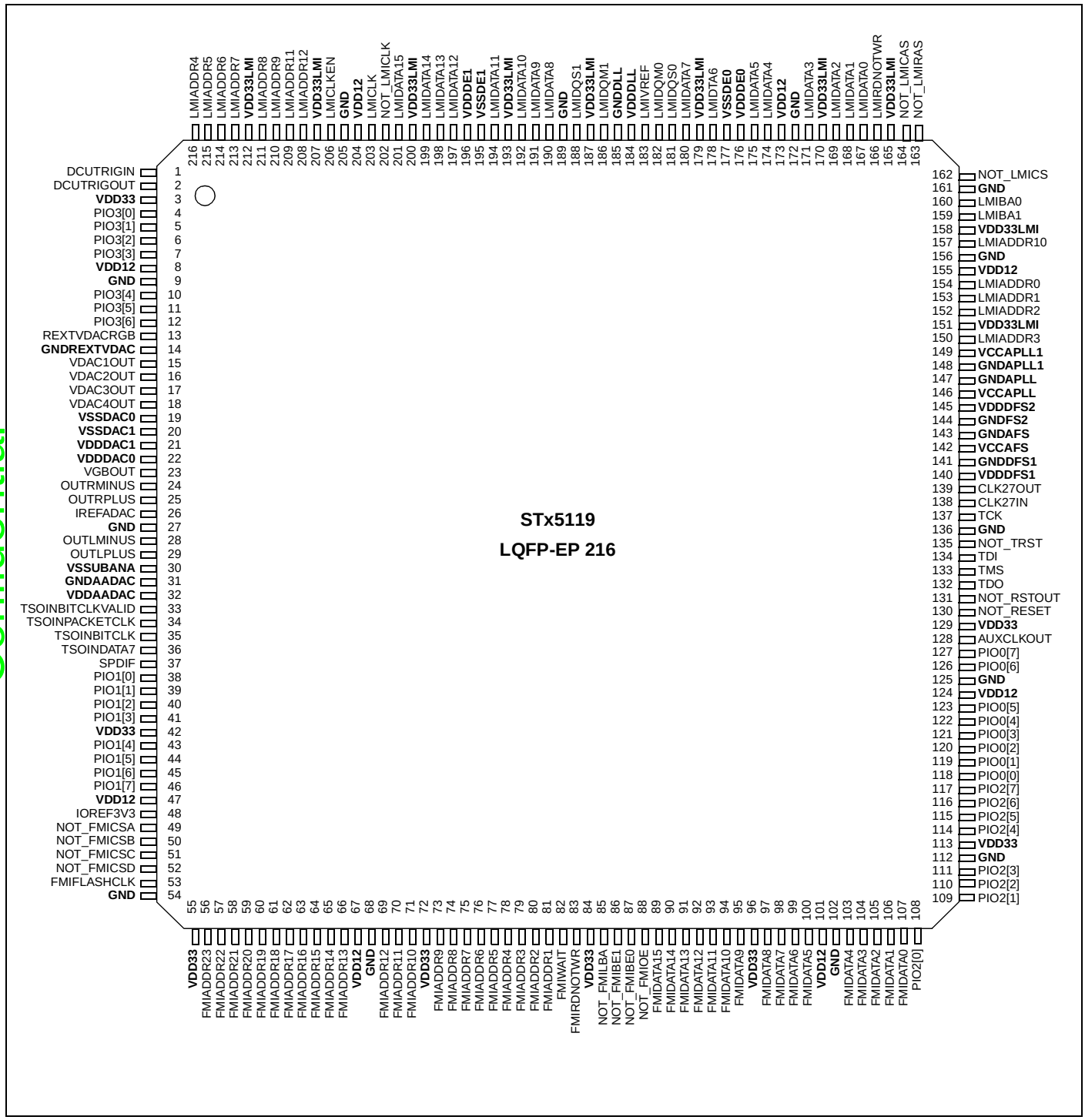


Table 3 shows the STx5119 pin list.

Table 3: Pin list sorted by number

Pin number	Type	Main function
1	I	DCUTRIGGERIN
2	O	DCUTRIGGEROUT
3	POWER	VDD3.3V
4	I/O	PIO3[0]
5	I/O	PIO3[1]
6	I/O	PIO3[2]
7	I/O	PIO3[3]
8	POWER	VDD1.2V
9	GND	GND
10	I/O	PIO3[4]
11	I/O	PIO3[5]
12	I/O	PIO3[6]
13	I/O	REXTVDACRGB
14	GND	GNDTEXTVDAC
15	O	VDAC1OUT
16	O	VDAC2OUT
17	O	VDAC3OUT
18	O	VDAC4OUT
19	GND	VSSVDAC0
20	GND	VSSVDAC1
21	POWER	3.3V VDAC1
22	POWER	3.3V VDAC0
23	I/O	VBGOUT
24	O	OUTRMINUS
25	O	OUTRPLUS
26	I/O	IREFADAC
27	GND	GND
28	O	OUTLMINUS
29	O	OUTLPLUS
30	GND	VSSUBANA
31	GND	GNDAADAC
32	POWER	3.3V AADAC
33	I	TS0INBITCLKVALID
34	I	TS0INPACKETCLK
35	I	TS0INBITCLK
36	I	TS0INDATA7
37	I/O	S/PDIF
38	I/O	PIO1[0]
39	I/O	PIO1[1]
40	I/O	PIO1[2]
41	I/O	PIO1[3]

Table 3: Pin list sorted by number

Pin number	Type	Main function
42	POWER	VDD3.3V
43	I/O	PIO1[4]
44	I/O	PIO1[5]
45	I/O	PIO1[6]
46	I/O	PIO1[7]
47	POWER	VDD1.2V
48	GND	IOREF3V3 (connect to ground)
49	O	NOT_FMICSA
50	O	NOT_FMICSB
51	O	NOT_FMICSC
52	O	NOT_FMICSD
53	O	FMIFLASHCLK
54	GND	GND
55	POWER	VDD3.3V
56	O	FMIADDR23
57	O	FMIADDR22
58	O	FMIADDR21
59	O	FMIADDR20
60	O	FMIADDR19
61	O	FMIADDR18
62	O	FMIADDR17
63	O	FMIADDR16
64	O	FMIADDR15
65	O	FMIADDR14
66	O	FMIADDR13
67	POWER	VDD1.2V
68	GND	GND
69	O	FMIADDR12
70	O	FMIADDR11
71	O	FMIADDR10
72	POWER	VDD3.3V
73	O	FMIADDR9
74	O	FMIADDR8
75	O	FMIADDR7
76	O	FMIADDR6
77	O	FMIADDR5
78	O	FMIADDR4
79	O	FMIADDR3
80	O	FMIADDR2
81	O	FMIADDR1
82	I	FMIWAIT
83	O	FMIRDNOTWR

Table 3: Pin list sorted by number

Pin number	Type	Main function
84	POWER	VDD3.3V
85	O	NOT_FMILBA
86	O	NOT_FMIBE1
87	O	NOT_FMIBE0
88	O	NOT_FMIOE
89	I/O	FMIDATA15
90	I/O	FMIDATA14
91	I/O	FMIDATA13
92	I/O	FMIDATA12
93	I/O	FMIDATA11
94	I/O	FMIDATA10
95	I/O	FMIDATA9
96	POWER	VDD3.3V
97	I/O	FMIDATA8
98	I/O	FMIDATA7
99	I/O	FMIDATA6
100	I/O	FMIDATA5
101	POWER	VDD1.2V
102	GND	GND
103	I/O	FMIDATA4
104	I/O	FMIDATA3
105	I/O	FMIDATA2
106	I/O	FMIDATA1
107	I/O	FMIDATA0
108	I/O	PIO2[0]
109	I/O	PIO2[1]
110	I/O	PIO2[2]
111	I/O	PIO2[3]
112	GND	GND
113	POWER	VDD3.3V
114	I/O	PIO2[4]
115	I/O	PIO2[5]
116	I/O	PIO2[6]
117	I/O	PIO2[7]
118	I/O	PIO0[0]
119	I/O	PIO0[1]
120	I/O	PIO0[2]
121	I/O	PIO0[3]
122	I/O	PIO0[4]
123	I/O	PIO0[5]
124	POWER	VDD1.2V
125	GND	GND

Table 3: Pin list sorted by number

Pin number	Type	Main function
126	I/O	PIO0[6]
127	I/O	PIO0[7]
128	O	AUXCLKOUT
129	POWER	VDD3.3V
130	I	NOT_RESET
131	O	NOT_RSTOUT
132	O	TDO
133	I	TMS
134	I	TDI
135	I	NOT_TRST
136	GND	GND
137	I	TCK
138	I	CLK27IN
139	ZO[1]	CLK27OUT
140	POWER	VDDDFS1
141	GND	GNDDFS1
142	POWER	VCCAFS
143	GND	GNDAFS
144	GND	GNDDFS2
145	POWER	VDDDFS2
146	POWER	VCCAPLL
147	GND	GNDAPLL
148	GND	GNDAPLL1
149	POWER	VCCAPLL1
150	O	LMIADDR3
151	POWER	VDD33LMI
152	O	LMIADDR2
153	O	LMIADDR1
154	O	LMIADDR0
155	POWER	VDD1.2V
156	GND	GND
157	O	LMIADDR10
158	POWER	VDD33LMI
159	O	LMIBA1
160	O	LMIBA0
161	GND_COMP33LMI	GND (connect to ground)
162	O	NOT_LMICS
163	O	NOT_LMIRAS
164	O	NOT_LMICAS
165	POWER	VDD33LMI
166	O	LMIRDNOTWR
167	I/O	LMIDATA0

Table 3: Pin list sorted by number

Pin number	Type	Main function
168	I/O	LMIDATA1
169	I/O	LMIDATA2
170	POWER	VDD33LMI
171	I/O	LMIDATA3
172	GND	GND
173	POWER	VDD1.2V
174	I/O	LMIDATA4
175	I/O	LMIDATA5
176	POWER	VDDDE0
177	GND	VSSDE0
178	I/O	LMIDATA6
179	POWER	VDD33LMI
180	I/O	LMIDATA7
181	O	LMIDQS0 (Not used, do not connect)
182	O	LMIDQM0
183	I	LMIVREF (Connect to GND)
184	POWER	VDDDLL
185	GND	GNDDLL
186	O	LMIDQM1
187	POWER	VDD33LMI
188	O	LMIDQS1 (Not used, do not connect)
189	GND	GND
190	I/O	LMIDATA8
191	I/O	LMIDATA9
192	I/O	LMIDATA10
193	POWER	VDD33LMI
194	I/O	LMIDATA11
195	GND	VSSDE1
196	POWER	VDDDE1
197	I/O	LMIDATA12
198	I/O	LMIDATA13
199	I/O	LMIDATA14
200	POWER	VDD33LMI
201	I/O	LMIDATA15
202	O	NOT_LMICLK (Not used, do not connect)
203	O	LMICLK
204	POWER	VDD1.2V
205	GND	GND
206	O	LMICLKEN
207	POWER	VDD33LMI
208	O	LMIADDR12
209	O	LMIADDR11

Table 3: Pin list sorted by number

Pin number	Type	Main function
210	O	LMIADDR9
211	O	LMIADDR8
212	POWER	VDD33LMI
213	O	LMIADDR7
214	O	LMIADDR6
215	O	LMIADDR5
216	O	LMIADDR4

1. ZO is the output pin of oscillator.

All digital pads are 3.3 V capable, selected pads are 5 V tolerant.

Table 4: Electrical summary: digital pads

Interface	Drive (mA)	5 V tolerant	Pull up/down	Tstate control
SDRAM		No	No	Yes
FMI	4	No	Yes[1]	Yes
Transport	4	Yes (data only)	Yes (data only)	Yes (data only)
Test/Debug	4	Yes (except on TCK)	Yes (except on TCK, TDO)	Yes (except on TCK)
Digital I/O[2]	4	Yes	Yes (except PIO3[3:0])[3]	Yes
System	4	Yes (except clock and reset input pins)	Yes (except clock and reset input pins)	No

1. Except FMIADDR22 and FMIDDR23 that have no pull-up/down.
2. PIO3 bit 6 is 3.3 V capable but not 5 V tolerant.
3. Pins PIO3[3:0] (I^2C interface) are open-drain and all other digital pads are push pull.

6.2 Alternative functions (mapped to GPIO pins)

To improve flexibility and to allow the STx5119 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. 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 connected when the PIO bit is configured to push-pull function mode.

Figure 13: I/O port pins

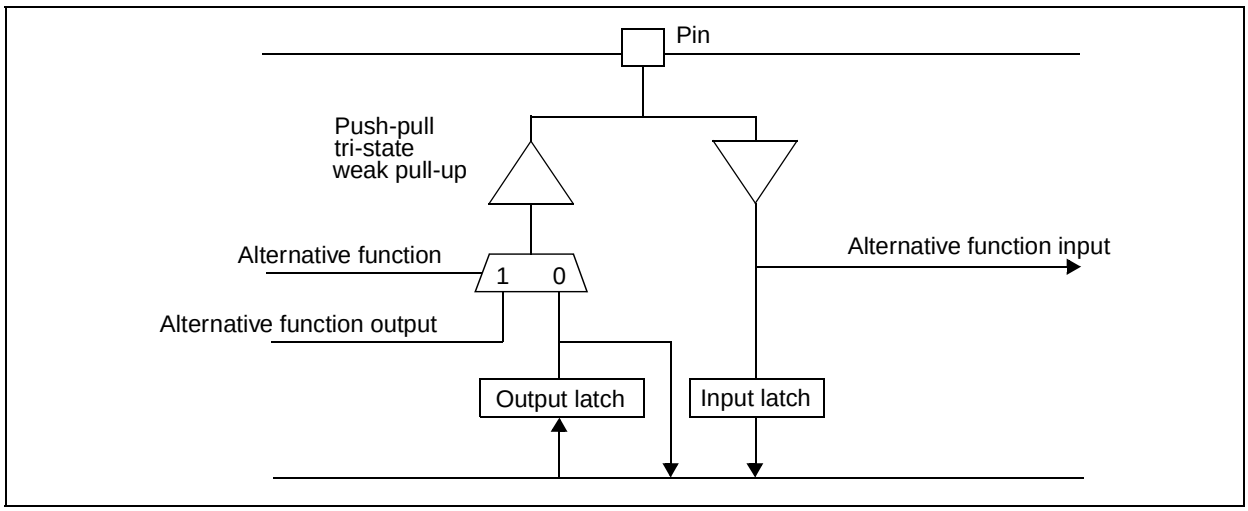


Table 5 to Table 8 show the mapping of the alternative functions. Unless otherwise stated, the default after reset for alternative functions is the PIO port or pin function.

The alternative configurations are set using the PIOn_ALTFOP_MUX_SEL_BUS [7:0] bits in the CONFIG_CTRL_F, and CONFIG_CTRL_G top-level configuration registers.

Table 5: PIO0 alternative functions

Pins	Bit	Alt 0	Dir	Alt1	Dir
118	0	SC0_DATAOUT/ ASC0_TXD	O	SCO_DATAINOUT	B
119	1	SC0_DATAIN/ ASC0_RXD	I	SSC1_MRST_DINOUT	B
120	2	SC0_CG_EXTCLK	I	SSC1_SCLKINOUT	B
121	3	SC0_CG_CLK	O	SCO_FSCLK0	O
122	4	SYSRV_SC_RESET_0	O	ASC0_RTS	O
123	5	SYSRV_SC_POWER_0	O	ASC0_CTS	I
126	6			SSC1_MTSR_DINOUT	B
127	7	SYSRV_SC_DETECT_0	I		

Note: Alt 0, Alt1 configured by PIO0_ALTFOP_MUX_SEL_BUS[7:0]:
Alt 0: PIO0_ALTFOP_MUX_SEL_BUS(n) = 0
Alt 1: PIO0_ALTFOP_MUX_SEL_BUS(n) = 1
where n = the bit number.

Table 6: PIO1 alternative functions

Pins	Bit	Alt 0	Dir	Alt 1	Dir	Alt 2	Dir
38	0	TS0INDATA0	I	Reserved		Reserved	I
39	1	TS0INDATA1	I				I
40	2	TS0INDATA2	I				I
41	3	TS0INDATA3	I				
43	4	TS0INDATA4	I				
44	5	TS0INDATA5	I				
45	6	TS0INDATA6	I				
46	7	TS0INERROR	I				

Note: Alt0, Alt1 configured by PIO1_ALTFOP_MUX_SEL_BUS[7:0]:
Alt 0: PIO1_ALTFOP_MUX_SEL_BUS(n) = 0
Alt 1: PIO1_ALTFOP_MUX_SEL_BUS(n) = 1
where n = the bit number.

Table 7: PIO2 alternative functions

Pins	Bit	Alt 0	Dir	Alt1	Dir	Alt 2	Dir
108	0	ASC1_TXD	O	PCM_DATAIN	I	Reserved	
109	1	ASC1_RXD	I	PCM_LRCLKIN	I		
110	2	ASC1_RTS	O	PCM_SCLKIN	I		
111	3	ASC1_CTS	I	PCM_MCLK	I		
114	4	IRB_UHF_IN	I	IRB_PPM_IN	I		
115	5	IRB_PPM_OUT	O	PCM_DATAOUT	O		
116	6	FDMA_REQ_O	I	PCM_LRCLKOUT	O		
117	7	FPRESET (Longtime Reset)	I	PCM_SCLKOUT	O		

Note: Alt0, Alt1 configured by PIO2_ALTFOP_MUX_SEL_BUS[7:0]:
ALT 0: PIO2_ALTFOP_MUX_SEL_BUS(n) = 0
ALT 1: PIO2_ALTFOP_MUX_SEL_BUS(n) = 1
where n = the bit number.

Table 8: PIO3 alternative functions

Pins	Bit	Alt 0	Dir	Alt1	Dir	Alt 2	Dir
4	0	SSC0_MTSR_DINOUT	B	Reserved	O	DMAREQ(0)	O
5	1	SSC0_SCLKINOUT	B		O	DMAREQ(1)	O
6	2	SSC1_MTSR_DINOUT	B		O	EXT_INT_OUT(0)	O
7	3	SSC1_SCLKINOUT	B		O	EXT_INT_OUT(1)	O
10	4	EXTINT2	I		O	FDMA_REQ1	I
11	5	EXTINT1	I	DVBCI_IOWR	B		
12	6	EXTINT0	I	DVBCI_IORD	O		

Note: 1 Alt0, Alt1 configured by PIO3_ALTFOP_MUX_SEL_BUS0[7:0]:
Alt 0: PIO3_ALTFOP_MUX_SEL_BUS0(n) = 0
Alt 1: PIO3_ALTFOP_MUX_SEL_BUS0(n) = 1

2 Alt 2, Alt3 configured by PIO3_ALTFOP_MUX_SEL_BUS1[7:0]:
Alt 2: PIO3_ALTFOP_MUX_SEL_BUS1(n) = 0
Alt 3: PIO3_ALTFOP_MUX_SEL_BUS1(n) = 1
where n = the bit number.

7 Connections

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

- power supplies (analog and digital) on page 37,
- system (reset, XTAL_27) on page 38,
- transport (PTI) input on page 39,
- 16-bit FMI on page 39,
- 16-bit LMI on page 41,
- audio interface on page 42,
- video interface on page 42,
- peripherals:
  - asynchronous serial controller on page 43,
  - synchronous serial controller on page 44,
  - programmable I/O on page 44,
  - miscellaneous on page 46,
- debug link (TAP, DCU) on page 47.

7.1 Power supplies

Table 9: Analog power supply pins

Pin	Assignment	Number	Description
	Audio DAC		
32	VDDAADAC	1	3.3 V power for audio DAC
31	GNDAADAC	1	Ground for audio DAC
30	VSSUBANA	1	Substrate ground for all DACs
	Video DAC		
22	VDDDAC0	1	3.3 V power supply for video DAC 1, 2 and 3
21	VDDDAC1	1	3.3 V power supply for CVBS video DAC 4
19	VSSVDAC0	1	Ground for video DAC 1, 2 and 3
20	VSSVDAC1	1	Ground for video DAC 4
	Frequency synthesizers		
140	VDDDFS1	1	Frequency synthesizer 1.2 V digital power dedicated to frequency synthesizer 1
141	GNDDFS1	1	Frequency synthesizer 1.2 V digital ground dedicated to frequency synthesizer 1
142	VCCAFS	1	Frequency synthesizer 1.2 V analog power shared between frequency synthesizer 1 and 2
143	GNDAFS	1	Frequency synthesizer 1.2 V analog ground shared between frequency synthesizer 1 and 2
144	GNDDFS2	1	Frequency synthesizer 1.2 V digital ground dedicated to frequency synthesizer 2
145	VDDDFS2	1	Frequency synthesizer 1.2 V digital power dedicated to frequency synthesizer 2
	PLLs		
146	VCCAPLL	1	1.2 V PLL1 analog power
147	GNDAPLL	1	1.2V PLL1 analog ground
148	GNDAPLL1	1	1.2 V PLL2 analog ground
149	VCCAPLL1	1	1.2V PLL2 analog power
	SDRAM		
184	VDDDLL	1	1.2 V for DLL
185	GNDDLL	1	Ground for DLL
176	VDDDE0	1	1.2 V for delay element 1
177	VSSDE0	1	Ground for delay element 1
196	VDDDE1	1	1.2 V for delay element 2
195	VSSDE1	1	Ground for delay element 2

Table 10: Digital power supply pins

Pin	Assignment	Number	Function
8, 47, 67, 101, 124, 155, 173, 204	VDD12	8	1.2 V digital power supply
151, 158, 165, 170, 179, 187, 193, 200, 207, 212	VDD33LMI	10	3.3 V power supply for LMI
3, 42, 55, 72, 84, 96, 113,129	VDD33	8	3.3 V power supply
EP[1], 9, 27, 54, 68, 102, 112, 125, 136, 156, 161, 172, 189, 205	GND	14	Ground
48	IOREF3V3	1	Ground

1. Expose pad

7.2 System

Table 11: System pins

Pin	Assignment	I/O	Description
138	CLK27IN	I	27 MHz crystal circuit
139	CLK27OUT	O	27 MHz crystal circuit
128	AUXCLKOUT	O	Auxiliary clock for general use
130	NOT_RESET	I	System reset
131	NOT_RSTOUT	O	Reset out

7.3 Transport stream

Table 12: Transport stream 0 pins

Pin	Assignment	I/O	Description
35	TS0INBITCLK	I	TSIN0 Parallel//Serial in
33	TS0INBITCLKVALID	I	
34	TS0INPACKETCLK	I	
36	TS0INDATA[7]	I	
45	TS0INDATA[6]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[6]
44	TS0INDATA[5]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[5]
43	TS0INDATA[4]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[4]
41	TS0INDATA[3]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[3]
40	TS0INDATA[2]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[2]
39	TS0INDATA[1]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[1]
38	TS0INDATA[0]	I	Parallel interface data bits for TSIN0 multiplexed to PIO1[0]
46	TS0INERROR	I	TS0IN Error multiplexed to PIO1[7]

7.4 Flash memory interface (FMI)

Table 13: FMI pins

Pin	Assignment	I/O	Description
89	FMIDATA15	I/O	Bidirectional data
90	FMIDATA14		
91	FMIDATA13		
92	FMIDATA12		
93	FMIDATA11		
94	FMIDATA10		
95	FMIDATA9		
97	FMIDATA8		
98	FMIDATA7		
99	FMIDATA6		
100	FMIDATA5		
103	FMIDATA4		
104	FMIDATA3		
105	FMIDATA2		
106	FMIDATA1		
107	FMIDATA0		

Table 13: FMI pins

Pin	Assignment	I/O	Description
56	FMIADDR23	O	Address (8 Mbyte)
57	FMIADDR22		
58	FMIADDR21		
59	FMIADDR20		
60	FMIADDR19		
61	FMIADDR18		
62	FMIADDR17		
63	FMIADDR16		
64	FMIADDR15		
65	FMIADDR14		
66	FMIADDR13		
69	FMIADDR12		
70	FMIADDR11		
71	FMIADDR10		
73	FMIADDR9		
74	FMIADDR8		
75	FMIADDR7		
76	FMIADDR6		
77	FMIADDR5		
78	FMIADDR4		
79	FMIADDR3		
80	FMIADDR2		
81	FMIADDR1		
52	NOT_FMICSD	O	Chip select
51	NOT_FMICSC/ADDR24		
50	NOT_FMICSB		
49	NOT_FMICSA		
88	NOT_FMIOE	O	Output enable
86	NOT_FMIBE1	O	Byte enable
87	NOT_FMIBE0		
85	NOT_FMILBA	O	Flash device load burst address
83	FMIRDNOTWR	O	Read not write
82	FMIWAIT	I	Wait input (extend read/write cycle)
53	FMIFLASHCLK	O	Peripheral clock

Note: Pins for FMIADDR22 and FMIADDR23 should be pulled up with an external resistor for correct operation of the device.

7.5 SDRAM interface for LMI

The LMI supports 64-,128-, 256- or 512-Mbit memory configurations, organized as 2 Mbytes x 4 banks x 16 bits. The 64-Mbit configuration uses all signals on the interface, with the LMI's LMIADDR[10:0] pins being directly connected to SDRAM's A[10:0]. For other configurations LMIADDR[12:11] are also connected to the SDRAM's A12 and A11 bits, as appropriate. Only one device can be connected to a board, operating in LVTTL weak mode.

Table 14: LMI pins

Pin	Assignment	I/O	Description
201	LMIDATA15	I/O	Bidirectional data
199	LMIDATA14		
198	LMIDATA13		
197	LMIDATA12		
194	LMIDATA11		
192	LMIDATA10		
191	LMIDATA9		
190	LMIDATA8		
180	LMIDATA7		
178	LMIDATA6		
175	LMIDATA5		
174	LMIDATA4		
171	LMIDATA3		Bidirectional data
169	LMIDATA2		
168	LMIDATA1		
167	LMIDATA0		
208	LMIADDR12	O	Address
209	LMIADDR11		
157	LMIADDR10		
210	LMIADDR9		
211	LMIADDR8		
213	LMIADDR7		
214	LMIADDR6		
215	LMIADDR5		
216	LMIADDR4		
150	LMIADDR3		
152	LMIADDR2		
153	LMIADDR1		
154	LMIADDR0		
159	LMIBA1	O	Bank select
160	LMIBA0		
182	LMIDQM0	O	Data I/O mask lower
186	LMIDQM1		Data I/O mask upper
181	LMIDQS0	O	Data strobe lower (Not used, do not connect)
188	LMIDQS1		Data strobe upper (Not used, do not connect)
162	NOT_LMICS	O	Chip select

Table 14: LMI pins

Pin	Assignment	I/O	Description
163	NOT_LMIRAS	O	Latch row address strobe
164	NOT_LMICAS	O	Latch column address strobe
166	LMIRDNOTWR	O	Write enable signal
206	LMICLKEN	O	Clock enable
203	LMICLK	O	Clock
202	NOT_LMICLK	O	Inverted clock (Not used, do not connect)
183	LMI_VREF	I	Input reference voltage (Connect to GND)

7.6 Audio interface

Table 15: Analog audio DAC pins

Pin	Assignment	I/O	Description
28	OUTLMINUS	O	Audio DAC left differential current output
24	OUTRMINUS	O	Audio DAC right differential current output
29	OUTLPLUS	O	Audio DAC left differential current output
25	OUTRPLUS	O	Audio DAC right differential current output
23	VBGOUT	O	Audio DAC filtered output reference voltage
26	IREFADAC	I	Current reference for Audio DAC

Table 16: Digital audio pins

Pin	Assignment	I/O	Description
37	S/PDIF	O	Digital audio

7.7 Video interface

Table 17: Analog video DAC pins

Pin	Assignment	I/O	Description
15	VDAC1OUT	O	Video DAC outputs
16	VDAC2OUT	O	
17	VDAC3OUT	O	
18	VDAC4OUT	O	
14	GNDREXTVDAC	I	Video DAC external resistor ground
13	REXTVDACRGB	I	VDAC external resistor input

7.8 Peripherals

7.8.1 Asynchronous serial controller (ASC)

There are two ASCs available. The ASC0 is smartcard capable, see Table 19: *Smartcards pin mapping* on page 43 for mapping details.

Table 18: Asynchronous serial controllers pin mapping

Pin	Alternative function	I/O	Description	Usual assignment
Asynchronous serial controller 0				
126	ASC0_NOTOE	O		PIO0[6]
118	ASC0_TXD	O	ASC0 transmit data signal	PIO0[0]
119	ASC0_RXD	I	ASC0 receive data signal	PIO0[1]
122	ASC0_RTS	O	ASC0 request to send signal	PIO0[4]
123	ASC0_CTS	I	ASC0 clear to send signal	PIO0[5]
108	ASC1_TXD	O	ASC1 transmit data signal	PIO2[0]
109	ASC1_RXD	I	ASC1 receive data signal	PIO2[1]
110	ASC1_RTS	O	ASC1 request to send signal	PIO2[2]
111	ASC1_CTS	I	ASC1 clear to send signal	PIO2[3]

7.8.2 Smartcard

Table 19: Smartcards pin mapping

Pin	Alternative function	I/O	Description	Usual assign-ment
	Smartcard 0			
122	SC0_RESET			PIO0[4]
123	SC0_COMD_VCC			PIO0[5]
126	SC0_DIR			PIO0[6]
127	SC0_DETECT			PIO0[7]
118	SC0_DATAOUT		Serial data output	PIO0[0]
119	SC0_DATAIN		Serial data in	PIO0[1]
120	SC0_CG_EXTCLK		External clock	PIO0[2]
121	SC0_CG_CLK		Clock for smartcard from system clock	PIO0[3]
122	SYSRV_SC_RESET_0		Reset to card	PIO0[4]
123	SYSRV_SC_POWER_0		Smartcard power	PIO0[5]
127	SYSRV_SC_DETECT_0		Smartcard detection	PIO0[7]
118	SC0_DATAINOUT			PIO0[0]
121	SC0_FSCLK0		Clock for smartcard from smartcard FS	PIO0[3]

7.8.3 Synchronous serial controllers (SSC0 and SSC1)

There are two synchronous serial controllers: SSC0 and SSC1.

For I^2C operation, only two wires are required for half duplex operation. The SSC1 can be an I^2C master or slave because the pin MTSR_DINOUT is connected internally to the MTSR or MRST data in and data out ports of the SSC under software configuration control.

For SPI operation, three wires are required for full duplex operation. Hence MTSR data and MRST data signals are on separate pins.

The SSC0 is available for use in I^2C mode only, as its MRST and MTSR pins are not available separately.

Table 20: Synchronous serial controller pin mapping

Pin	Alternative function	I/O	Description	Usual assignment
	Synchronous serial controller 0			
5	SSC0_SCLKINOUT	O	Serial clock in	PIO3[1]
4	SSC0_MTSR_DINOUT	I	Serial data input for SSC0: master transmit, slave receive	PIO3[0]
	Synchronous serial controller 1			
119	SSC1_MRST_DINOUT	O	Serial data for SSC1: master receive, slave transmit	PIO0[1]
120, 7	SSC1_SCLKINOUT	O	Serial clock in	PIO0[2], PIO3[3]
126, 6	SSC1_MTSR_DINOUT	I	Serial data input for SSC1: master transmit, slave receive	PIO0[6], PIO3[2]

7.8.4 Programmable I/O

Table 21: Programmable I/O pins

Pin	Assignment	I/O	Description
127	POI0[7]	I/O	PIO port 0
126	PIO0[6]		
123	PIO0[5]		
122	PIO0[4]		
121	PIO0[3]		
120	PIO0[2]		
119	PIO0[1]		
118	PIO0[0]		
46	PIO1[7]	I/O	PIO port 1
45	PIO1[6]		
44	PIO1[5]		
43	PIO1[4]		
41	PIO1[3]		
40	PIO1[2]		
39	PIO1[1]		
38	PIO1[0]		

Table 21: Programmable I/O pins

Pin	Assignment	I/O	Description
117	PIO2[7]	I/O	PIO port 2
116	PIO2[6]		
115	PIO2[5]		
114	PIO2[4]		
111	PIO2[3]		
110	PIO2[2]		
109	PIO2[1]		
108	PIO2[0]		
12	PIO3[6]	I/O	PIO port 3
11	PIO3[5]		
10	PIO3[4]		
7	PIO3[3]		
6	PIO3[2]		
5	PIO3[1]		
4	PIO3[0]		

7.8.5 Miscellaneous

Other pins available are the infrared blaster interface, external DMA request, front panel reset, external interrupts and digital audio outputs.

Table 22: Miscellaneous

Pin	Alternative function	I/O	Description	Usual assignment
114	IRB_UHF_IN	I	UHF channel input for IRB	PIO2[4]
115	IRB_PPM_OUT	O	PPM output of IRB	PIO2[5]
114	IRB_PPM_IN	I	PPM input of IRB	PIO2[4]
115	PCM_DATAOUT	O		PIO2[5], alt1
116	PCM_LRCLKOUT	O		PIO2[6], alt1
117	PCM_SCLKOUT	O		PIO2[7], alt1
116	FDMA_REQ_0	O		PIO2[6], alt0
117	FPRESET	I		PIO2[7], alt0
10	EXTINT2	I		PIO3[4], alt0
11	EXTINT1	I		PIO3[5], alt0
12	EXTINT0	I		PIO3[6], alt0
4	DMAREQ(0)	O		PIO3[0], alt2
5	DMAREQ(1)	O		PIO3[1], alt2
6	EXT_INT_OUT(0)	O		PIO3[2], alt2
7	EXT_INT_OUT(1)	O		PIO3[3], alt2

7.9 Debug link

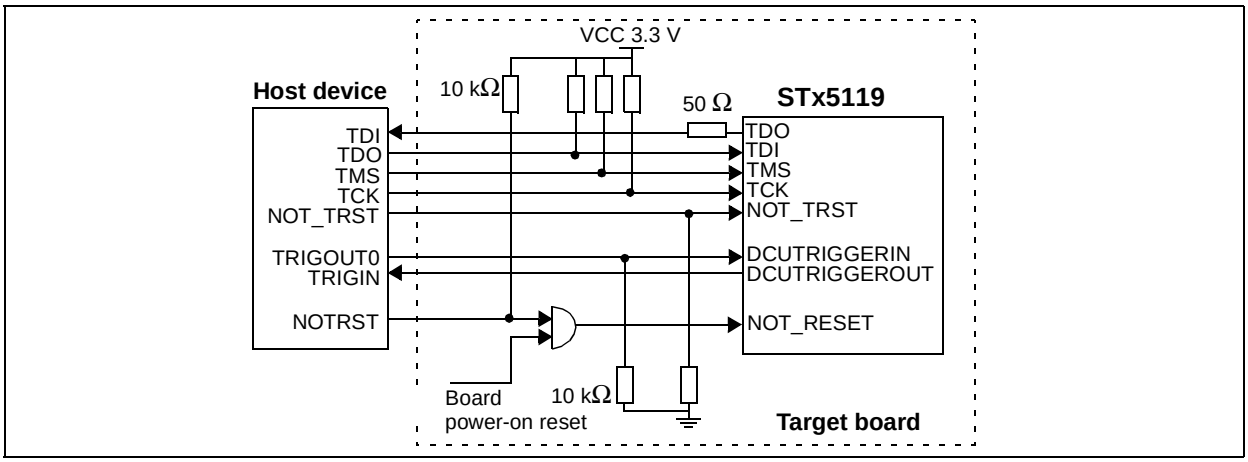
Table 23: JTAG test access port (TAP) pins

Pin	Assignment	I/O	Description
134	TDI	I	TAP boundary scan test data input
133	TMS	I	TAP boundary scan test mode select
137	TCK	I	TAP boundary scan test clock
135	NOT_TRST	I	TAP boundary scan test logic reset
132	TDO	O	TAP boundary scan test data output

Table 24: DCU pins

Pin	Assignment	I/O	Description
1	DCUTRIGGERIN	I	External trigger input to DCU
2	DCUTRIGGEROUT	O	Signal to trigger external debug circuitry

Figure 14: Recommended connections



Note: If there is a lot of noise on the clock line a capacitor in the range of 10 to 100 pF can be fitted between TCK and ground near the target STx5119.

8 Configuration registers

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

- alternative assignments on pins,
- switching of S/PDIF output between the audio decoder and the S/PDIF player,
- video DAC control,
- audio DAC control,
- interrupt direction,
- EMI pull-ups,
- smartcard clock selection,
- DMAREQ direction and polarity.

Addresses are provided as either the *ICMonitorBaseAddress* or the *ICControlBaseAddress* + offset.

The *ICMonitorBaseAddress* is:

0x2040 2000.

The *ICControlBaseAddress* is:

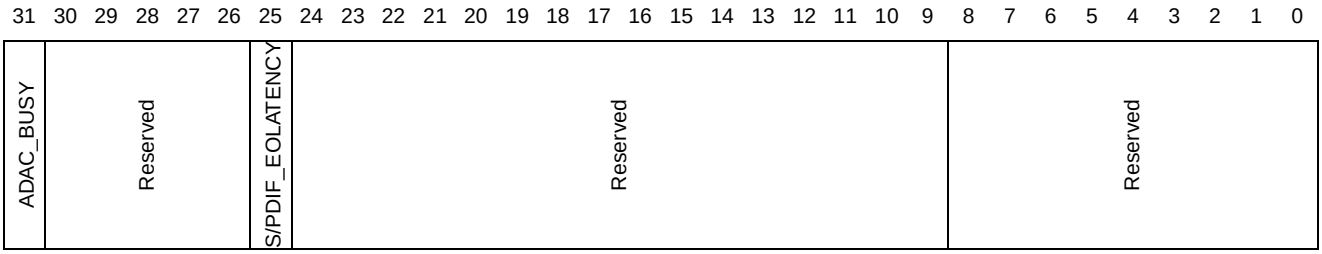
0x2030 3000.

Table 25: Configuration registers

Register name	Description	Address offset	Type
CONFIG_MONITOR_E	Configuration monitoring register E, see page 49	0x0028[1]	RO
CONFIG_MONITOR_G	Configuration monitoring register G, see page 49	0x0030[1]	RO
CONFIG_CTRL_C	Configuration control C, see page 50	0x0000[2]	R/W
CONFIG_CTRL_D	Configuration control D, see page 51	0x0004[2]	R/W
CONFIG_CTRL_E	Configuration control D, see page 51	0x0008[2]	R/W
CONFIG_CTRL_F	Configuration control F, see page 52	0x000C[2]	R/W
CONFIG_CTRL_G	Configuration control G, see page 53	0x0010[2]	R/W
CONFIG_CTRL_H	Configuration control H, see page 53	0x0014[2]	R/W

1. Uses *ICMonitorBaseAddress*
2. Uses *ICControlBaseAddress*

CONFIG_MONITOR_E Configuration monitoring register E



Address: *ICMonitorBaseAddress* + 0x0028

Type: Read only

Reset: 0

Description:

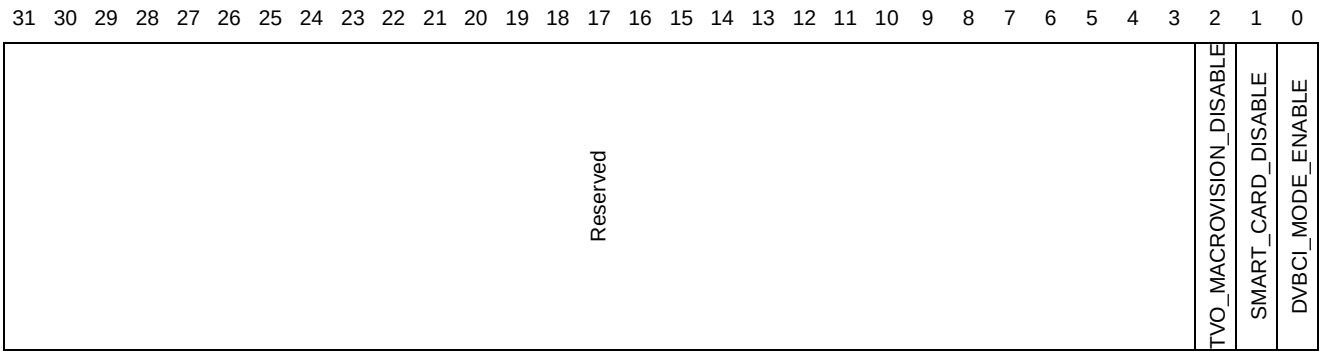
[31] **ADAC_BUSY:** Audio DAC busy signal low in standby mode, from ADAC

[30:26] **Reserved**

[25] **S/PDIF_EOLATENCY:** sync bit for S/PDIF out, presentation time for S/PDIF stream

[24:9] **Reserved**

CONFIG_MONITOR_G Configuration monitoring register G



Address: *ICMonitorBaseAddress* + 0x0030

Type: Read only

Reset: 0

Description:

[2] **TVO Macrovision disable**
0: Enable
1: Disable

[1] **Smartcard disable**
0: Enable
1: Disable

[0] **Enable DVBCI mode**
0: Disable
1: Enable

CONFIG_CTRL_C Configuration control C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAREQ_POLARITY		Reserved				FMI_PULLUP_DISABLE		Reserved	IRB_DISABLE	DVBCI_BANK1_ENABLE	DVBCI_BANK0_ENABLE	Reserved					TSIS_SOP_TOKEN								TSIS_SYNC_NOT_ASYNC	TSIS_ALIGN_BYTE_SOP	TSIS_ASYNC_SOP_TOKEN	TSIS_SERIAL_NOT_PARALLEL	Reserved		

Address: *ICControlBaseAddress* + 0x0000

Type: Read/write

Reset: 0000

Description:

[31:30] **DMAREQ_POLARITY[1:0]:** External PIO request selective inversion.
FDMA request mapping
00: no invert
01: invert DMAREQ[0]
10: invert DMAREQ[1]
11: invert DMAREQ[1:0]

[29:26] **Reserved**

[25:24] **FMI_PULLUP_DISABLE[1:0]**
00: no disable
01:Reserved
11: disable for FMI pullups
10:Reserved

[23] **Reserved**

[22] **IRB_DISABLE**
0: enable
1: disable

[21] **DVBCI_BANK1_ENABLE**
0: disable
1: enable

[20] **DVBCI_BANK0_ENABLE**
0: disable
1: enable

[19:15] **Reserved**

[14:7] **TSIS_SOP_TOKEN[7:0]**
The data value to be placed into the first byte of each packet when TSIS_ASYNC_SOP_TOKEN = 1

[6] **TSIS_SYNC_NOT_ASYNC**
0: the input stream is assumed asynchronous (packetclk is high for non validated byte)
1: the input stream is synchronous, (only validated bytes are ever considered)

[5] **TSIS_ALIGN_BYTE_SOP**
0: no alignment will occur on serial to parallel conversion
1: the start of packet signal is used to indicate the bit start of a packet and therefore byte boundary

[4] **TSIS_ASYNC_SOP_TOKEN**
0: no data substitution takes place
1: the start token of each packet is replaced by TSIS_SOP_TOKEN [7:0]

[3] **TSIS_SERIAL_NOT_PARALLEL**
0: parallel
1: serial

[2:0] **Reserved**

CONFIG_CTRL_D Configuration control D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						Reserved		Reserved	PCM_IRLOCK_INVERT	Reserved		FDMA_CLK-SELECT	ADAC_SOFT_ MUTE_DISABLE	ADAC_DIGITAL-POWER_DOWN	ADAC_ANALOG-POWER_DOWN	ADAC_BG-POWER_DOWN	ADAC_MODE_SELECT	ADAC_RST_N	VIDEO_DAC_POFF_YCC_N	VIDEO_DAC_CMDR_YCC	VIDEO_DAC_CMDS_YCC	VDAC_HZ_YCC	Reserved								

Address: *ICControlBaseAddress* + 0x0004

Type: Read/write

Reset: 0000

Address:

[31:26] **Reserved**

[25:24] **Reserved**

[23] **Reserved**

[22] **PCM_IRLOCK_INVERT**
o: no inversion
1: invert IR lock

[21:20] **Reserved**

[19] **FDMA_CLK_SELECT:**
0: from PLL
1: from frequency synthesizer

[18] **ADAC_SOFT_ MUTE_DISABLE**
0: soft mute on
1: soft mute disabled

[17] **ADAC_DIGITAL-POWER_DOWN**
0: default during chip operation
1: reserved

[16] **ADAC_ANALOG-POWER_DOWN**
0: default during chip operation
1: reserved

[15] **ADAC_BG-POWER_DOWN**
0:
1: analog bandgap power down

[14] **ADAC_MODE_SELECT**
0: MCLK(FS) 256, SCLK(FS) 64, FS(KHz) 32-48, Mode normal
1: MCLK(FS) 128, SCLK(FS) 64, FS(KHz) 96, Mode double

[13] **ADAC_SOFT_RST_N**
0: soft reset applied
1: soft reset inactive

[12] **VIDEO_DAC_POFF_YCC_N**
1: disable all three DACs. This bit is inverted so on power up it is low.

[11] **VIDEO_DAC_CMDR_YCC:** level control for UOUT, WOUT, VOUT

[10] **VIDEO_DAC_CMDS_YCC:** UOUT, WOUT, VOUT

[9] **VDAC_HZ_YCC:** YCC outputs are high impedance when = 1, see Section 34.4: *Video specifications and DAC set up* on page 367

[8:0] **Reserved**

CONFIG_CTRL_E Configuration control E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					SMATCARD_CLK0_SEL	Reserved									

Address: *ICControlBaseAddress* + 0x0008
Type: Read/write
Reset: 0000
Address:

[10] **SMARTCARD_CLK0_SEL**
0: Smartcard clock is provided from external source through PIO0 [2] (default)
1: Smartcard clock is provided from system services.

CONFIG_CTRL_F Configuration control F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PIO2_ALTFOP_MUX_SEL								PIO1_ALTFOP_MUX_SEL								PIO0_ALTFOP_MUX_SEL							

Address: *ICControlBaseAddress* + 0x000C
Type: Read/write
Reset: 0000
Address:

[23:16] **PIO2_ALTFOP_MUX_SEL[7:0]**
Selection of alternative functions on PIO port 2

[15:8] **PIO1_ALTFOP_MUX_SEL[7:0]**
Selection of alternative functions on PIO port 1

[7:0] **PIO0_ALTFOP_MUX_SEL[7:0]**
Selection of alternative functions on PIO port 0

CONFIG_CTRL_G Configuration control G

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO3_ALTFOP_MUX_SELECT_BUS1								PIO3_ALTFOP_MUX_SELECT_BUS0								Reserved															

Address: *ICControlBaseAddress* + 0x0010

Type: Read/write

Reset: 0000

Address:

[31:24] **PIO3_ALTFOP_MUX_SEL_BUS1[7:0]**
Selection of alternative functions on PIO port 3

[23:16] **PIO3_ALTFOP_MUX_SEL_BUS0[7:0]**
Selection of alternative functions on PIO port 3

CONFIG_CTRL_H Configuration control H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SSC1_MTSR_DIN_SEL	SSC1_SCLKIN_SEL	SSC0_RXD_SEL	Reserved																								

Address: *ICControlBaseAddress* + 0x0014

Type: Read/write

Reset: 0000

Address:

[31:28] **Reserved**

[27] **SSC1_MTSR_DIN_SEL**
0: PIO0 [6]
1: PIO3 [2]

[26] **SSC1_SCLKIN_SEL**
0: PIO0 [2]
1: PIO3 [3]

[25] **SSC0_RXD_SEL**
0: PIO0 [0]
1: PIO0 [1]

[24:0] **Reserved**

9 Register base addresses

9.1 Base addresses

Table 26 describes the base addresses for each block in the STx5119, listed in alphabetical order.

Table 26: Register base addresses

Module	Base address	Register summary
2D blitter display engine	0x2100 0000	Table 79: *2D blitter display engine registers* on page 246
Asynchronous serial controller: UART0	0x2083 0000	Table 107: *Asynchronous serial controller (ASC) registers* on page 474
Asynchronous serial controller: UART1	0x2083 1000	
Audio decoder	0x2070 0000	Table 92: *Audio decoder registers* on page 373
Digital encoder	0x2090 0000	Table 86: *Digital encoder (DENC) registers* on page 320
FMI config	0x2020 0000	Table 52: *Flash and peripheral memory interface (FMI) registers* on page 114
FMI Bank 0 FMI Bank 1 FMI Bank 2 FMI Bank 3	0x4000 0000 0x7F40 0000 0x7F80 0000 0x7FC0 0000	
Flexible DMA	0x20D0 0000	Table 105: *Flexible DMA (FDMA) registers* on page 420
Graphics DMA: GDMA 1	0x2090 0C00	Table 81: *Graphics DMA (GDMA) registers* on page 288
Infrared transmitter/receiver	0x2081 8000	Table 106: *Infrared transmitter/receiver registers* on page 449
Interconnect: monitor	0x2040 2000	Table 25: *Configuration registers* on page 48
Interconnect: control	0x2030 3000	
Interrupt controller	0x3000 4000	Table 28: *Interrupt system registers* on page 63
Interrupt level controller	0x2080 0000	
Local memory interface	0xE000 0000	Table 37: *LMI registers* on page 93
Memory: DCache	0x3000 1000	Table 30: *DCache and ICache registers* on page 79
Memory: ICache	0x3000 2000	
MPEG video decoder	0x2050 0000	Table 69: *MPEG video decoder registers* on page 203
PCM player	0x2140 0000	Table 95: *PCM player registers* on page 384
PIO: PIO0	0x2082 0000	Table 112: *Programmable I/O port registers* on page 513
PIO: PIO1	0x2082 1000	
PIO: PIO2	0x2082 2000	
PIO: PIO3	0x2082 3000	
PTI	0x20E0 0000	Table 64: *Programmable transport interface (PTI) registers: DMA* on page 173 Table 65: *Programmable transport interface (PTI) registers: others* on page 173
S/PDIF player	0x2060 0000	Table 99: *S/PDIF player and GPFIFO registers* on page 394

Table 26: Register base addresses

Module	Base address	Register summary
Synchronous serial controller: SSC0	0x2084 0000	Table 111: *Synchronous serial controller (SSC) registers* on page 501
Synchronous serial controller: SSC1	0x2084 1000	
System services	0x20F0 0000	Table 58: *System services registers* on page 137
TVOut Teletext	0x2090 0700	Table 80: *TVOut Teletext registers* on page 281
Video timing generator	0x2090 0400	Table 88: *Video timing generator (VTG) registers* on page 355

10 Interrupt system

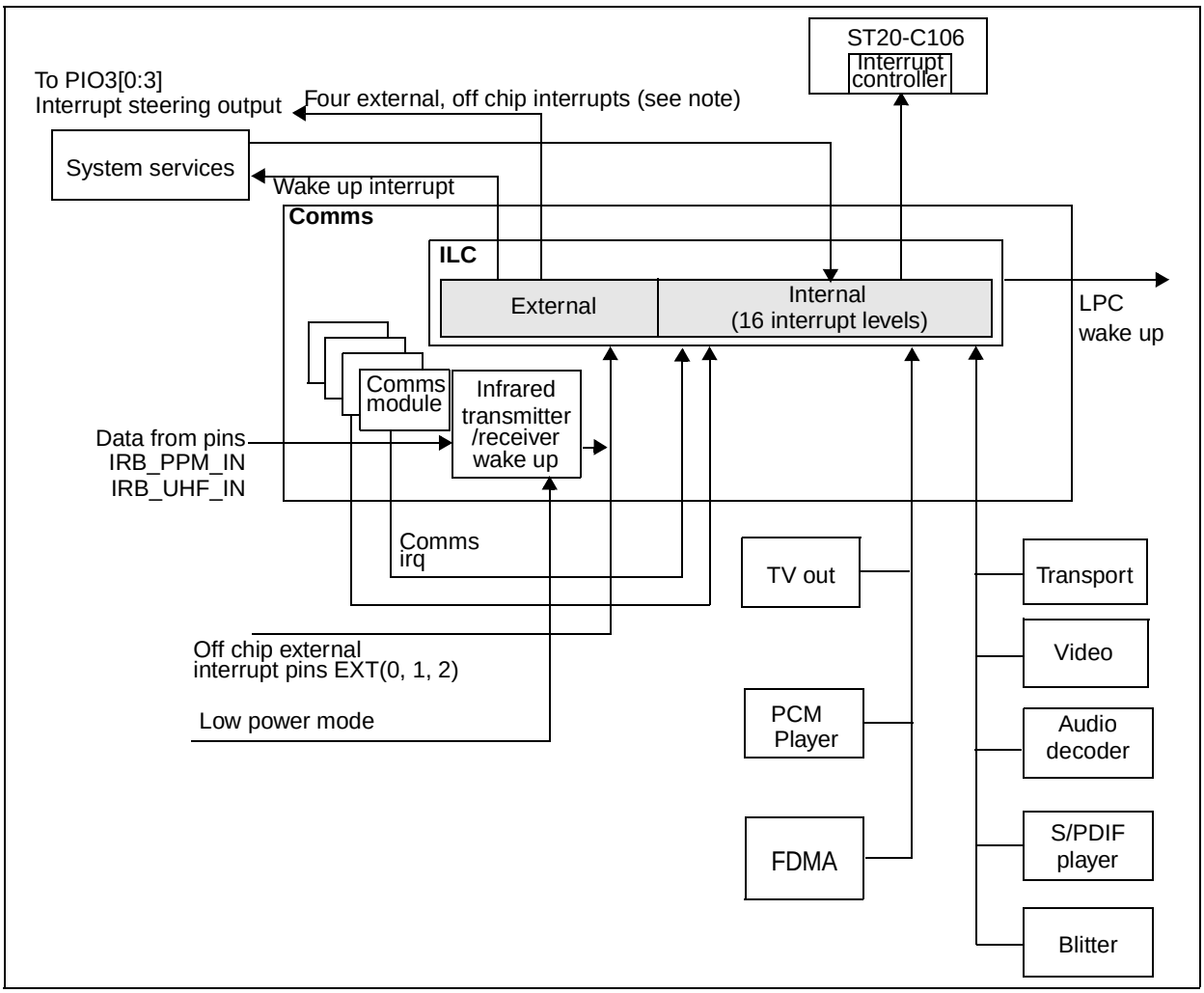
10.1 Overview

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 an on-chip interrupt controller (see Section 10.2: *Interrupt controller* on page 57) and an on-chip interrupt level controller (ILC) (see Section 10.3: *Interrupt level controller* on page 60). The ILC multiplexes the 44 incoming interrupt sources onto the 20 programmable ILC interrupt level inputs (16 to the CPU and 4 available externally). Multiplexing is controlled by software. An overview of the interrupt system is shown in Figure 15.

Figure 15: Interrupt network



Pins INTERRUPT[3:0] can be connected to both the interrupt steering outputs or the four external interrupt inputs.

Note: The four remote OFF chip interrupts are routed from the ILC to PIO3 pads, two of which are routed are as DMAREQ, PIO3[0,1] and two are routed as external interrupts EXT_INT_OUT, PIO3[2:3].

10.2 Interrupt controller

The interrupt controller 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.

All interrupts have a higher priority than the low priority process queue. Each interrupt can be programmed to be at a lower or higher priority than the high priority process queue by writing to the priority bit in the INC_HANDLERWPTR registers. Interrupts which are specified as higher priority must be contiguous from the highest numbered interrupt downwards. For example, if eight interrupts are programmed as high priority and eight as low, then the higher priority interrupts must be set to interrupt [15:8] and the lower priority interrupts to interrupt [7:0].

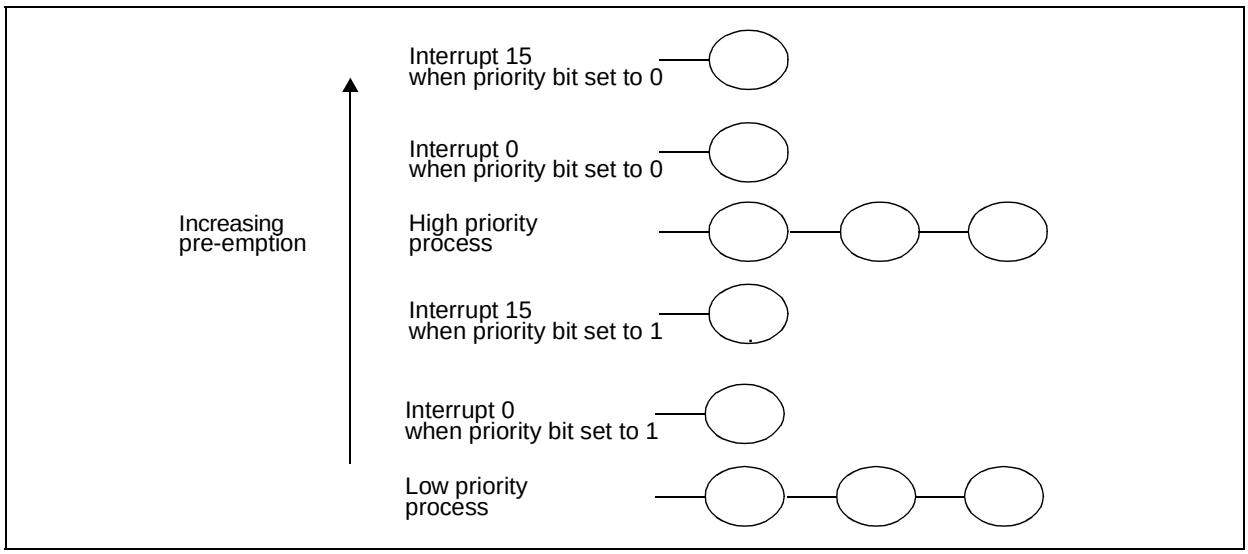
Each of the 16 interrupt levels of the interrupt controller can be programmed with an interrupt trigger mode, using the INC_TRIGGERMODE register. The trigger mode can be set to be high or low level, rising edge, falling edge or any edge sensitive. All on-chip module interrupt sources produce active high level interrupt signals. Therefore the interrupt level for these interrupt sources must be programmed with a high level trigger mode.

Each of the 16 interrupt levels can be programmed to be enabled or disabled by the INC_MASK register. The default state of INC_MASK is that all interrupt levels are disabled. A corresponding level bit is set in the INC_PENDING register if the interrupt signal from the interrupt level controller matches the level trigger condition. If this is the highest priority bit set in the INC_PENDING register, the CPU then executes the interrupt handler associated with that level by the INC_HANDLERWPTR register and the INC_PENDING bit is then reset. If the level bit set in the INC_PENDING register is not the highest priority bit set, then the bit remains set until it is the highest priority level bit, then the CPU executes the associated interrupt handler for that level. The CPU only executes the interrupt handler and then clear the INC_PENDING register bit if it is enabled in the INC_MASK register. Software can write to the INC_PENDING register to generate a software interrupt on any of the 16 interrupt levels.

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

The interrupt controller also contains an INC_EXEC register. This is used by the interrupt controller logic to keep a record of which interrupt level handler is currently executing on the CPU (or was previously executing before being preempted by a high priority process, for low priority interrupts) and which levels have been preempted 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.

Figure 16: Interrupt priority



10.2.1 Interrupt vector table

The interrupt controller contains a table of pointers to interrupt handlers. There are 16 interrupt handlers, each controlled by a work space register INC_HANDLERWPTR 0 to 15. The table of pointer values contains a work space pointer for each interrupt level.

The INC_HANDLERWPTR registers access the code, data and interrupt save area of the interrupt handler. The position of the INC_HANDLERWPTR register in the interrupt table sets the priority of the interrupt.

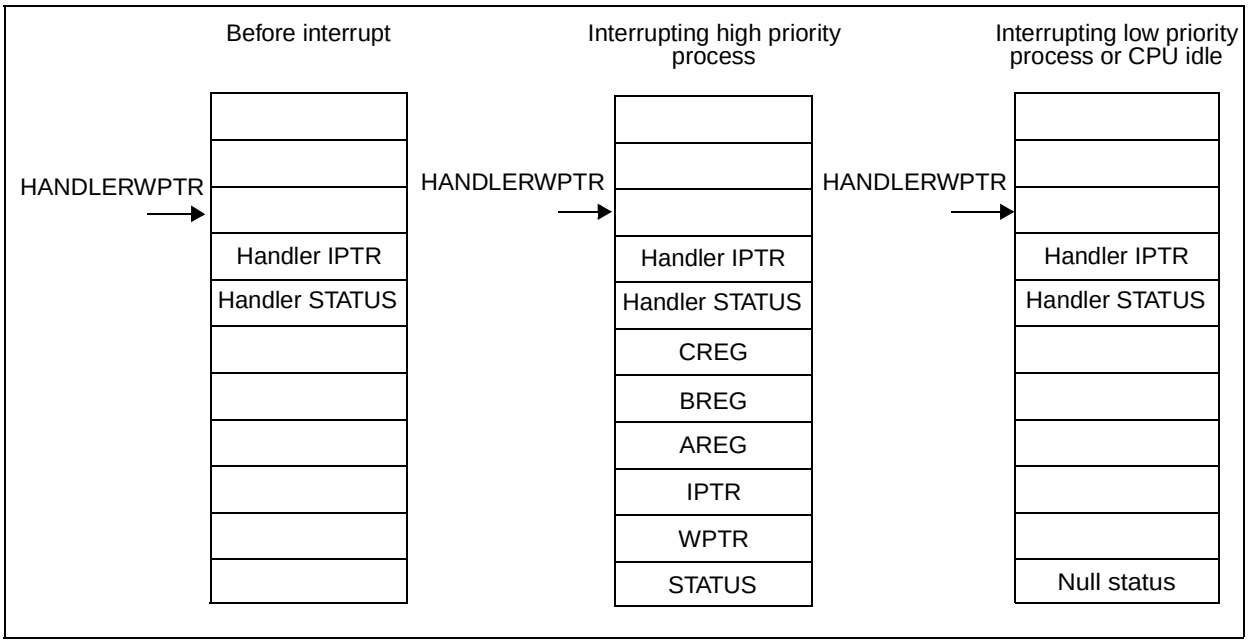
The operating system run time library (OS20) supports the setting and programming of the vector table.

10.2.2 Interrupt handlers

The CPU can receive an interrupt request from the interrupt controller at any interruptible point in its execution and 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 as shown in Figure 17. Each interrupt level has its own work space.

Figure 17: State of interrupted process



The interrupt routine is initialized with space below HANDLERWPTR. The IPTR and STATUS word for the routine are stored there permanently. This should be programmed before the HANDLERWPTR is written into the vector table.

The behavior of the interrupt differs depending on the priority of the CPU and the interrupt, when the interrupt occurs. If an interrupt occurs when the CPU is running at high priority, and the interrupt is set at a higher priority than the high priority process queue, the CPU saves the current process state (AREG, BREG, CREG, WPTR, IPTR and STATUS) into the work space of the interrupt handler. The value of HANDLERWPTR, which is stored in the interrupt controller, points to the top of this work space. The values of IPTR and STATUS to be used by the interrupt handler are loaded from this work space and starts executing the handler. The value of WPTR is then set to the bottom of this save area.

If an interrupt occurs when the CPU is running at high priority, and the interrupt is set at a lower priority than the high priority process queue, no action is taken and the interrupt waits in a queue until the high priority process queue is empty (see Section 10.2.4: *Pre-emption and interrupt priority* on page 59).

Interrupts always take priority over low priority processes. If a high priority interrupt occurs when the CPU is idle or running at low priority, the STATUS is saved. This indicates that no valid process is running (null status). The state of the interrupted low priority processes is stored in shadow registers. This state can be accessed via the **ldshadow** (load shadow registers) and **stshadow** (store shadow registers) instructions. The interrupt handler is then run at high priority.

When the CPU runs at low priority and the interrupt has lower priority than a high priority process, the CPU saves the current process state (AREG, BREG, CREG, WPTR, IPTR and STATUS) into the work space of the interrupt handler. The value HANDLERWPTR, which is stored in the interrupt controller, points to the top of this work space. The values of IPTR and STATUS to be used by the interrupt handler are loaded from this work space and starts executing the handler. The value of WPTR is then set to the bottom of this save area.

When the interrupt routine has completed it must adjust WPTR to the value at the start of the handler code and then execute the **iret** (interrupt return) instruction. This restores the interrupted state from the interrupt handler structure and signals to the interrupt controller that the interrupt has completed. The processor then continues from where it was before being interrupted.

10.2.3 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 between fast internal SRAM memory and interrupt latency.

10.2.4 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 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 the INC_EXEC register. 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 interrupt controller 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 preempts 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.

10.2.5 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 perform CPU 2D block move instructions.
  A workaround is to store the state of the shadow registers used by the 2D block move instruction into memory, with the **stshadow** instruction. The state must be restored after the 2D block move instruction by using the **ldshadow** instruction.
- Interrupt handlers must not cause program traps.
  (However, they may be trapped by a scheduler trap.)

10.3 Interrupt level controller

The interrupt level controller multiplexes 64 internal and 8 external interrupt source signals on to the 16 interrupt level inputs of the interrupt controller. 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 is provided to allow routing of a further four interrupts off chip, any interrupt source may be mapped on to any of the four outputs.

The incoming interrupt signals can be generated by on-chip subsystems or received from external pins. Table 27 on page 61 assigns each of the interrupt sources to a number n from 0 to 63. Software assigns a signal n to one of the 16 interrupt levels by writing the priority of the required input in the register ILC_PRIORITY. Each of the 44 interrupt sources in the interrupt level controller can be selectively enabled or disabled at source, by writing to the ILC_ENABLE register. This is in addition to the individual masking of the 16 levels in the interrupt controller. This disables the interrupt source from generating an interrupt, without disabling all other interrupt sources mapped on to that interrupt level. This would be the case if the INC_MASK register in the interrupt controller 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) edges, or on the high or low logic level of the incoming interrupt source signal. This is controlled by writing to the ILC_MODE registers.

The default state of the interrupt level controller trigger mode registers is no trigger, therefore, these registers need to be programmed if external interrupts are to be enabled. The default state of the enable registers is low, therefore these registers need to be programmed before internal interrupts can be serviced.

When setting trigger modes in the interrupt level controller, the corresponding trigger mode for the interrupt level in the interrupt controller that the source(s) are mapped to, must be programmed to high level. Otherwise, the trigger mode in the interrupt controller and the interrupt level controller may conflict. The ILC_INPUT_INTERRUPT register has the same function as in the STi5500, STi5505, STi5508, STi5518 and can be used to indicate the current logic state of all the interrupt sources. This register is just a buffered version of the interrupt source signals before the trigger mode detection stage. ILC_INPUT_INTERRUPT does not latch the signal, as the ILC_STATUS register does, for interrupt sources defined with an edge sensitive trigger mode. The ILC_STATUS register 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 interrupt level controller would have the interrupt source trigger mode set to be rising edge. On a rising edge the corresponding bit in the ILC_STATUS register would be set high and would remain set until explicitly cleared by the interrupt handler routine writing to the corresponding bit in the ILC_CLEAR_STATUS register. However if the pulse was short, by the time the interrupt handler was executed and had read the ILC_INPUT_INTERRUPT register, the pulse may have returned to a logic low and the bit would be read as zero. Thus the cause of interrupt could not be determined if more than one interrupt source had been multiplexed on to the interrupt level.

So now, using the ILC_STATUS register, it is possible to multiplex interrupt sources of different types, including edge sensitive, on to the same interrupt level in the interrupt controller.

The STx5119 interrupt level controller also has two registers mapped into its register address space, that have no connection with normal interrupt operation. These registers control the 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. The register ILC_WAKEUP_ACTIVE_LEVEL controls whether the four external interrupt pins are active high or low, to wake up the CPU from low power mode. The setting of this register has no effect on the triggering of the external interrupt pins in the interrupt level controller. The register ILC_WAKEUP_ENABLE is a mask register to enable or disable the external interrupt pins from waking up the CPU from low power mode. Again, this has no effect on the masking of these interrupts in the interrupt level controller.

10.4 Interrupt assignments

All interrupts are active high. Interrupts from the internal peripherals and external pins are assigned as in Table 27.

Table 27: STx5119 interrupt assignments

INT n	Peripheral	Description of the function
0	PIO0	Compare function
1	PIO1	Compare function
2	PIO2	Compare function
3	PIO3	Compare function
4	Reserved	
5	SSC0	SSC0TIR, SSC0RIR, SSC0EIR I^2C master
6	SSC1	SSC1TIR, SSC1RIR, SSC1EIR I^2C master
7	ASC0	ASC3TIR, ASC3TBIR, ASC3RIR, ASC3EIR
8	ASC1	ASC2TIR, ASC2TBIR, ASC2RIR, ASC2EIR
9-13	Reserved	
14	PTI	Programmable transport interface
15	PDES	Descrambler interface
16-18	Reserved	
19	IR blaster	Tx, Rx interrupts
20-21	Reserved	
22	Video decoder	Video decoder
23	Audio decoder	Audio decoder
24-25	Reserved	
26	S/PDIF	Audio S/PDIF player
27	PCM	Audio PCM player
28 to 31	Reserved	
32	VTG	TV out
33	Reserved	
34	FDMA	FDMA interrupts
35	BLIT CQ1	Blitter
36	Reserved	
37	BLIT AQ1	Blitter
38-40	Reserved	
41	DCO_CORRECTION	System services
42	CPU Tick	Timer interrupt from system services
43	Pwm timer int	Timer interrupt from CPU
44 to 63	Reserved	
External inputs		
64 (EXT0)	ExtIntIn0	From external devices
65(EXT1)	ExtIntIn1	
66(EXT2)	ExtIntIn2	
67(EXT3)	Reserved	
68 (EXT4)	IR wake up	From pins via pulse stretcher

Table 27: STx5119 interrupt assignments

INT n	Peripheral	Description of the function
69-70	Reserved	
71 EXT7	FDMA/PIO[0]	FDMA
External outputs		
0(EXT0)	ExtIntOUT0	To external devices
1 (EXT1)	ExtIntOUT1	
2-5	Reserved	
6 (EXT6)	FDMA REQ0	
7 (EXT7)	FDMA REQ1	
Wake up	Wake up system services	
C1 interrupts	ILC_INTERRUPT_ LOCAL_OUT	

11 Interrupt system registers

Addresses are provided as either the *IntControllerBaseAddress* or the *ILCBaseAddress* + offset.

The *IntControllerBaseAddress* is:

0x3000 4000.

The *ILCBase*Address is:

0x2080 0000.

Table 28: Interrupt system registers

Register name	Description	Address offset	Type
Interrupt controller registers[1]			
INC_CLEAR_EXEC	Clear a bit of the EXEC register, see page 65	0x048	WO
INC_CLEAR_MASK	Clear a bit of the interrupt enable mask, see page 65	0x028	WO
INC_CLEAR_PENDING	Clear a bit of the pending register, see page 66	0x018	WO
INC_EXEC	Interrupts executing, see page 66	0x040	R/W
INC_GLOBAL_MASK	Enable all interrupts, see page 66	0x030	R/W
INC_HANDLERWPTR0	Interrupt handler 0 work space pointer, see page 67	0x100	R/W
INC_HANDLERWPTR1	Interrupt handler 1 work space pointer, see page 67	0x104	R/W
INC_HANDLERWPTR2	Interrupt handler 2 work space pointer, see page 67	0x108	R/W
INC_HANDLERWPTR3	Interrupt handler 3 work space pointer, see page 67	0x10C	R/W
INC_HANDLERWPTR4	Interrupt handler 4 work space pointer, see page 67	0x110	R/W
INC_HANDLERWPTR5	Interrupt handler 5 work space pointer, see page 67	0x114	R/W
INC_HANDLERWPTR6	Interrupt handler 6 work space pointer, see page 67	0x118	R/W
INC_HANDLERWPTR7	Interrupt handler 7 work space pointer, see page 67	0x11C	R/W
INC_HANDLERWPTR8	Interrupt handler 8 work space pointer, see page 67	0x120	R/W
INC_HANDLERWPTR9	Interrupt handler 9 work space pointer, see page 67	0x124	R/W
INC_HANDLERWPTR10	Interrupt handler 10 work space pointer, see page 67	0x128	R/W
INC_HANDLERWPTR11	Interrupt handler 11 work space pointer, see page 67	0x12C	R/W
INC_HANDLERWPTR12	Interrupt handler 12 work space pointer, see page 67	0x130	R/W
INC_HANDLERWPTR13	Interrupt handler 13 work space pointer, see page 67	0x134	R/W
INC_HANDLERWPTR14	Interrupt handler 14 work space pointer, see page 67	0x138	R/W
INC_HANDLERWPTR15	Interrupt handler 15 work space pointer, see page 67	0x13C	R/W
INC_MASK	Interrupt enable mask, see page 68	0x020	R/W
INC_NUmINTS	Number of interrupt levels supported, see page 68	0x004	RO
INC_PENDING	Interrupt pending, see page 69	0x010	R/W
INC_REVID	ID code and revision number, see page 68	0x000	RO
INC_SET_EXEC	Set a bit of the EXEC register, see page 69	0x044	WO
INC_SET_MASK	Set an interrupt enable mask, see page 69	0x024	WO
INC_SET_PENDING	Set a bit of the pending register, see page 70	0x014	WO
INC_TRIGGERMODE0	Interrupt 0 trigger mode, see page 70	0x180	R/W
INC_TRIGGERMODE1	Interrupt 1 trigger mode, see page 70	0x184	R/W
INC_TRIGGERMODE2	Interrupt 2 trigger mode, see page 70	0x188	R/W
INC_TRIGGERMODE3	Interrupt 3 trigger mode, see page 70	0x18C	R/W

Table 28: Interrupt system registers

Register name	Description	Address offset	Type
INC_TRIGGERMODE4	Interrupt 4 trigger mode, see page 70	0x190	R/W
INC_TRIGGERMODE5	Interrupt 5 trigger mode, see page 70	0x194	R/W
INC_TRIGGERMODE6	Interrupt 6 trigger mode, see page 70	0x198	R/W
INC_TRIGGERMODE7	Interrupt 7 trigger mode, see page 70	0x19C	R/W
INC_TRIGGERMODE8	Interrupt 8 trigger mode, see page 70	0x1A0	R/W
INC_TRIGGERMODE9	Interrupt 9 trigger mode, see page 70	0x1A4	R/W
INC_TRIGGERMODE10	Interrupt 10 trigger mode, see page 70	0x1A8	R/W
INC_TRIGGERMODE11	Interrupt 11 trigger mode, see page 70	0x1AC	R/W
INC_TRIGGERMODE12	Interrupt 12 trigger mode, see page 70	0x1B0	R/W
INC_TRIGGERMODE13	Interrupt 13 trigger mode, see page 70	0x1B4	R/W
INC_TRIGGERMODE14	Interrupt 14 trigger mode, see page 70	0x1B8	R/W
INC_TRIGGERMODE15	Interrupt 15 trigger mode, see page 70	0x1BC	R/W
Interrupt level controller registers[2]			
ILC_INPUT_INTERRUPT	Input interrupt register, see page 71	0x080, 0x084, 0x08C	RO
ILC_STATUS	Status register, see page 71	0x200, 0x204, 0X20C	RO
ILC_CLEAR_STATUS	Clear status locations, see page 72	0x280, 0x284, 0X28C	WO
ILC_ENABLE	Enable registers, see page 72	0x400, 0x404, 0X40C	R/W
ILC_CLEAR_ENABLE	Clear enable locations, see page 73	0x480, 0x484, 0X48C	WO
ILC_SET_ENABLE	Set enable locations, see page 73	0x500, 0x504, 0X50C	WO
ILC_WAKEUP_ENABLE	Wake up enable registers, see page 74	0x608	R/W
ILC_WAKEUP_ACTIVE_ LEVEL	Wake up level registers, see page 74	0x688	R/W
ILC_PRIORITY0	Priority register 0, see page 75	0x800	R/W
ILC_PRIORITY1	Priority register 1, see page 75	0x808	R/W
ILC_PRIORITY2	Priority register 2, see page 75	0x810	R/W
ILC_PRIORITY3	Priority register 3, see page 75	0x818	R/W
ILC_PRIORITY4	Priority register 4, see page 75	0x820	R/W
ILC_PRIORITY5	Priority register 5, see page 75	0x828	R/W
ILC_PRIORITY6	Priority register 6, see page 75	0x830	R/W
ILC_PRIORITY7	Priority register 7, see page 75	0x838	R/W
ILC_PRIORITY8	Priority register 8, see page 75	0x840	R/W
ILC_PRIORITY9	Priority register 9, see page 75	0x848	R/W
ILC_PRIORITY10	Priority register 10, see page 75	0x850	R/W

Table 28: Interrupt system registers

Register name	Description	Address offset	Type
ILC_PRIORITY11	Priority register 11, see page 75	0x858	R/W
ILC_PRIORITY12	Priority register 12, see page 75	0x860	R/W
ILC_PRIORITY13	Priority register 13, see page 75	0x868	R/W
ILC_PRIORITY14	Priority register 14, see page 75	0x870	R/W
ILC_PRIORITY15	Priority register 15, see page 75	0x878	R/W
ILC_MODE0	Mode register 0, see page 75	0xA04	R/W
ILC_MODE1	Mode register 1, see page 75	0xA0C	R/W
ILC_MODE2	Mode register 2, see page 75	0xA14	R/W

1. Uses *IntControllerBaseAddress* as the base address.
2. Uses *ILCBaseAddress* as the base address.

11.1 Interrupt control registers

INC_CLEAR_EXEC Clear a bit of the exec register

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																CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR

Address: *IntControllerBaseAddress* + 0x048

Type: Write only

Description: When set, bits 0 to 15 of this register clear the corresponding interrupt exec bit (INTEX) in register INC_EXEC. There is one bit for each of the 16 interrupt levels, where bit 0 corresponds to bit INTEX0. Bit 16 of this register clears all of the bits of the INC_EXEC register.

Note: This register must not be used as its operation is undefined.

INC_CLEAR_MASK Clear a bit of the interrupt enable mask

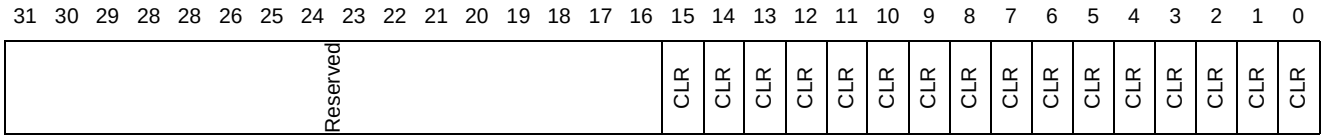
31	30	29	28	28	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR

Address: *IntControllerBaseAddress* + 0x028

Type: Write only

Description: When set, bits 0 to 15 of this register clear the corresponding interrupt enable bit (INTEN) in register INC_MASK. There is one bit for each of the 16 interrupt levels, where bit 0 corresponds to bit INTEN0. Bit 16 of this register clears all of the bits of the INC_MASK register.

INC_CLEAR_PENDING Clear a bit of the pending register

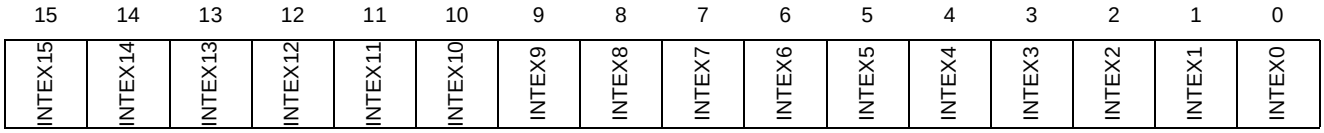


Address: *IntControllerBaseAddress* + 0x018

Type: Write only

Description: When set, bits 0 to15 of this register clear the corresponding interrupt pending bit (PENDINT) in register INC_PENDING. There is one bit for each of the 16 interrupt levels, where bit 0 corresponds to bit PENDINT0. Bit 16 of this register clears all of the bits of the INC_PENDING register.

INC_EXEC Interrupts executing



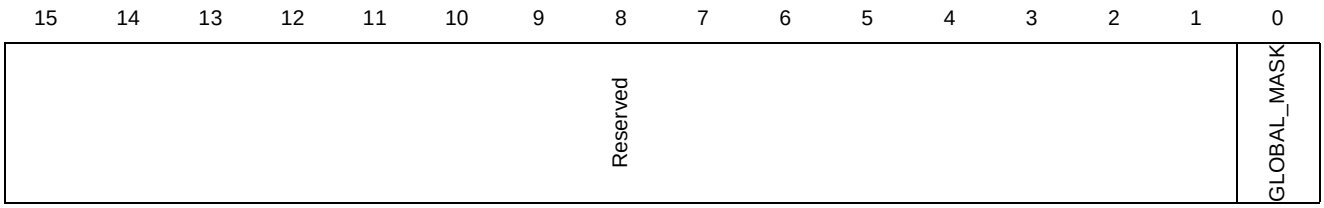
Address: *IntControllerBaseAddress* + 0x040

Type: Read/write

Reset: 0

Description: This register records whether interrupt levels 0 to15 are currently executing and preempting interrupts. The bit is set when the CPU starts running code for that interrupt, where INTEX0 corresponds to interrupt level 0. The highest priority interrupt bit is reset once the interrupt handler executes a return from interrupt instruction (**iret**).
This register is specified as being read/write, but users should not attempt to write to this register, as the device operation is undefined.

INC_GLOBAL_MASK Enable all interrupts



Address: *IntControllerBaseAddress* + 0x030

Type: Read/write

Reset: 0

Description: This register has similar functionality to the INC_MASK register, except that it has a single bit which masks all of the interrupts. Writing to GLOBAL_MASK has no effect on the contents of the INC_MASK register.

[15:1] **Reserved**

[0] **GLOBAL_MASK**
1: global mask enabled
0: global mask disabled.

INC_HANDLERWPTRn Interrupt handler work space 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
0x100	HANDLERWPTR0[31:2]	Res	P
0x104	HANDLERWPTR1[31:2]	Res	P
0x108	HANDLERWPTR2[31:2]	Res	P
0x10C	HANDLERWPTR3[31:2]	Res	P
0x110	HANDLERWPTR4[31:2]	Res	P
0x114	HANDLERWPTR5[31:2]	Res	P
0x118	HANDLERWPTR6[31:2]	Res	P
0x11C	HANDLERWPTR7[31:2]	Res	P
0x120	HANDLERWPTR8[31:2]	Res	P
0x124	HANDLERWPTR9[31:2]	Res	P
0x128	HANDLERWPTR10[31:2]	Res	P
0x12C	HANDLERWPTR11[31:2]	Res	P
0x130	HANDLERWPTR12[31:2]	Res	P
0x134	HANDLERWPTR13[31:2]	Res	P
0x138	HANDLERWPTR14[31:2]	Res	P
0x13C	HANDLERWPTR15[31:2]	Res	P

Address: *IntControllerBaseAddress* + 0x100 to 0x13C

Type: Read/write

Reset: Undefined

Description: This register points to the work space of the corresponding interrupt handler (HANDLERWPTR0 corresponds to interrupt level 0). The base of the work space is 32-bit word aligned, so the two least significant bits of the 32-bit address are always zero, and are not held. Each register contains a priority bit P which determines whether the interrupt is at a higher or lower priority than the high priority process queue. Before the interrupt is enabled by writing 1 in the INC_MASK register, the software must ensure that there is a valid HANDLERWPTRn in the register.

[31:2] **HANDLERWPTRn[31:2**]
The 30 most significant bits of the address of the work space of the interrupt handler.

[1] **Reserved**

[0] **P**
Sets the priority of the interrupt. If this bit is set to 0, the interrupt is a higher priority than the high priority process queue; if this bit is 1, the interrupt is a lower priority than the high priority process queue.
0: High priority
1: Low priority

INC_MASK Interrupt enable 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																INTEN15	INTEN14	INTEN13	INTEN12	INTEN11	INTEN10	INTEN9	INTEN8	INTEN7	INTEN6	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0

Address: *IntControllerBaseAddress* + 0x020

Type: Read/write

Reset: 0

Description: This register selectively enables or disables interrupts from the interrupt level controller, connected to each of the 16 interrupt levels. The global interrupt disable bit, disables all interrupt levels, whatever the state of the individual interrupt mask bits.
The INC_PENDING register contains a pending flag for each interrupt level. The INC_MASK register masks the INC_PENDING register to control what interrupts the CPU, while continually monitoring interrupts.
On start up, INC_MASK is initialized to zeros so all interrupts are disabled both globally and individually. When 1 is written to the GLOEN bit, the individual interrupt channels are still disabled. To enable an interrupt channel, 1 must also be written to the corresponding INTEN bit.

[31:16] **Reserved**

[15:0] **INTEN[15:0]**
1: Interrupt enabled
0: Interrupt disabled.

INC_MASK is mapped on to registers INC_SET_MASK and INC_CLEAR_MASK so that bits can be set or cleared individually.

INC_NUMINTS Number of interrupt levels

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
NUMBER_OF_INTERRUPTS																															

Address: *IntControllerBaseAddress* + 0x004

Type: Read only

Description: This register contains the number of interrupt levels supported.

INC_REVID ID code and revision number

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
REVID																															

Address: *IntControllerBaseAddress* + 0x000

Type: Read only

Description: This register contains unique ID code and revision number.

INC_PENDING Interrupt pending

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PENDINT15	PENDINT14	PENDINT13	PENDINT12	PENDINT11	PENDINT10	PENDINT9	PENDINT8	PENDINT7	PENDINT6	PENDINT5	PENDINT4	PENDINT3	PENDINT2	PENDINT1	PENDINT0

Address: *IntControllerBaseAddress* + 0x010

Type: Read/write

Reset: 0

Description: This register contains one bit per interrupt level. A read of this register examines the state of the interrupt controller; a write can explicitly trigger an interrupt.
A bit is set when the triggering condition for an interrupt level is met. All bits are independent so that several bits can be set in the same cycle. Once a bit is set, a further triggering condition has no effect. The triggering condition is independent of INC_MASK.
The highest priority interrupt bit is reset, once the interrupt controller has made an interrupt request to the CPU.
The interrupt controller receives interrupt requests and makes an interrupt request to the CPU when it has a pending interrupt request of higher priority than the currently executing interrupt handler.
If the software needs to write or clear some bits of the INC_PENDING register, the interrupts should be masked (by writing or clearing the INC_MASK register) before writing or clearing the INC_PENDING register. The interrupts can then be unmasked.
The INC_PENDING register is mapped on to registers INC_SET_PENDING and INC_CLEAR_PENDING so that bits can be set or cleared individually.

INC_SET_EXEC Set a bit of the exec register

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																SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET

Address: *IntControllerBaseAddress* + 0x044

Type: Write only

Description: This register sets bits of the INC_EXEC register individually. Writing 1 in this register sets the corresponding bit in the INC_EXEC register, 0 leaves the bit unchanged.

Note: Do not write to this register as device operation is undefined.

INC_SET_MASK Set an interrupt enable 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																SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET

Address: *IntControllerBaseAddress* + 0x024

Type: Write only

Description: This register sets bits of the INC_MASK register individually. Writing 1 in this register sets the corresponding bit in the INC_MASK register, 0 leaves the bit unchanged.

INC_SET_PENDING Set a bit of the pending register

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																SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	SET

Address: *IntControllerBaseAddress* + 0x014

Type: Write only

Description: This register sets bits of the INC_PENDING register individually. Writing 1 in this register sets the corresponding bit in the INC_PENDING register, 0 leaves the bit unchanged.

INC_TRIGGERMODEn Interrupt trigger 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
0x180	Reserved																															TRIGGER0
0x184	Reserved																															TRIGGER1
0x188	Reserved																															TRIGGER2
0x18C	Reserved																															TRIGGER3
0x190	Reserved																															TRIGGER4
0x194	Reserved																															TRIGGER5
0x198	Reserved																															TRIGGER6
0x19C	Reserved																															TRIGGER7
0x1A0	Reserved																															TRIGGER8
0x1A4	Reserved																															TRIGGER9
0x1A8	Reserved																															TRIGGER10
0x1AC	Reserved																															TRIGGER11
0x1B0	Reserved																															TRIGGER12
0x1B4	Reserved																															TRIGGER13
0x1B8	Reserved																															TRIGGER14
0x1BC	Reserved																															TRIGGER15

Address: *IntControllerBaseAddress* + 0x180 to 0x1BC

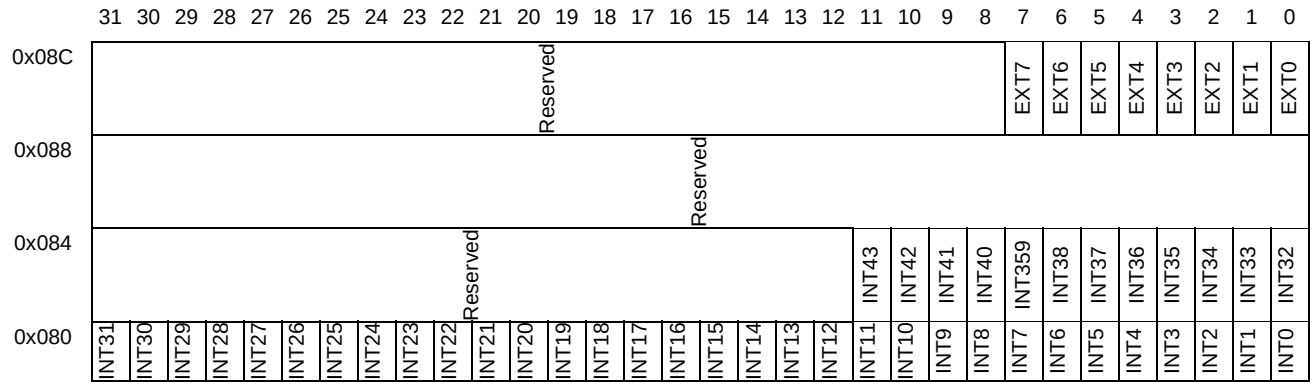
Type: Read/write

Reset: Undefined

Description: These registers control the triggering conditions of the interrupts. Each interrupt channel can be programmed to trigger on rising or falling edges or high or low levels on the incoming interrupt signal.
Level triggering is different from edge triggering in that if the input is held at the triggering level, a continuous stream of interrupts is generated.

11.2 Interrupt level controller registers

ILC_INPUT_INTERRUPT Input interrupt register



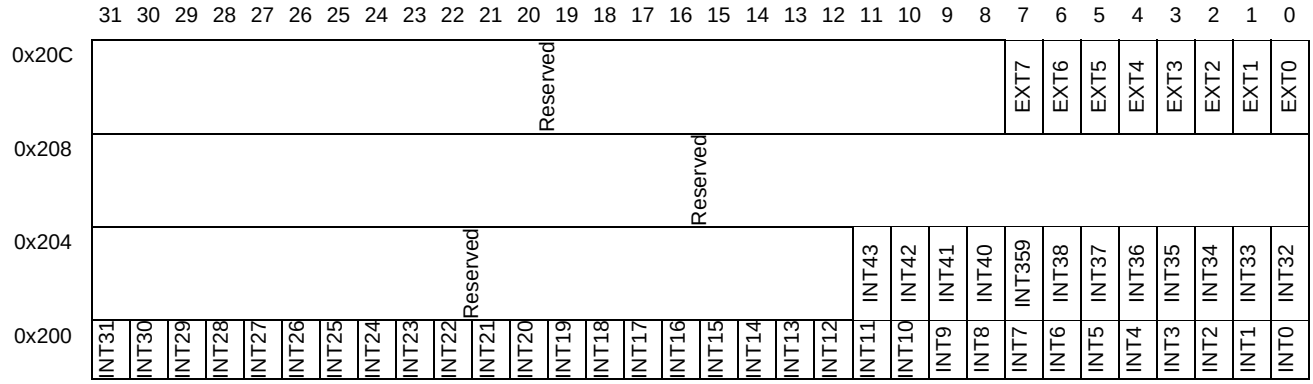
Address: *ILCBaseAddress* + 0x080, 0x084, 0x088 and 0x08C

Type: Read

Reset: 0

Description: The synchronized version of all the interrupt numbers can be read from this register. There are two input interrupt registers from 0x080 to 0x084. The contents of this register are set to logic 0 on reset.

ILC_STATUS Status registers



Address: *ILCBaseAddress* + 0x200, 0x204, 0x208 and 0x20C

Type: Read

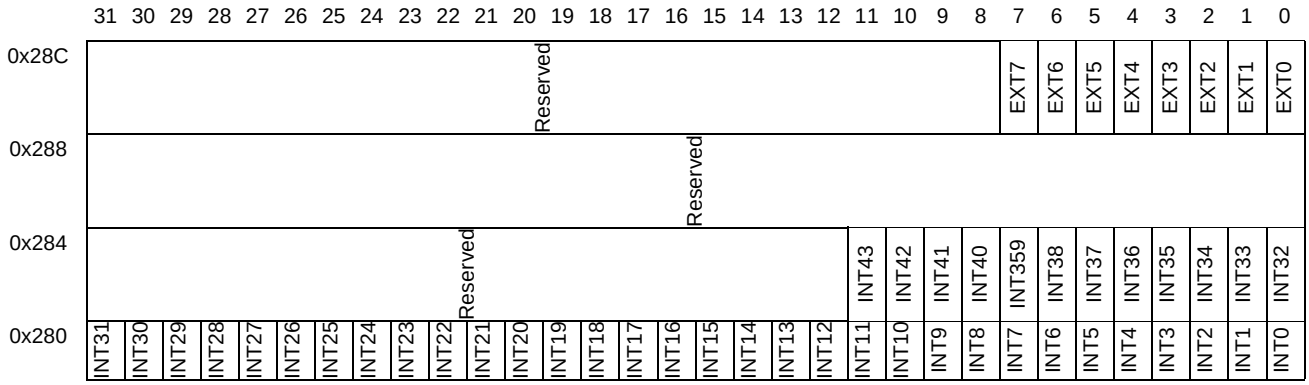
Reset: 0

Description: The status of the interrupt numbers is inferred by reading this register. The bit in the status register is at logic 1 on the following conditions.

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

There are two status registers from 0x200 to 0x27C. The contents of this register are at logic 0 on reset.

ILC_CLEAR_STATUS Clear status locations



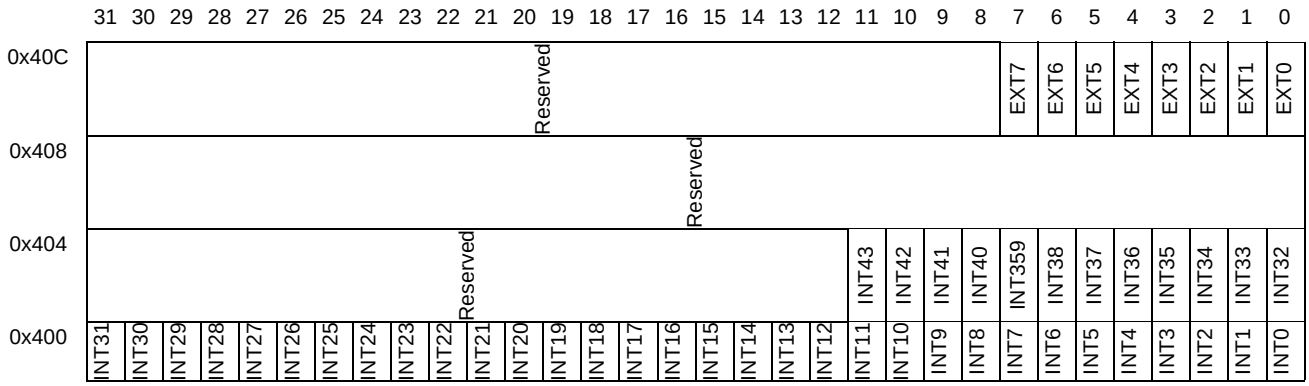
Address: *ILCBaseAddress* + 0x280, 0x284, 0x288 and 0x28C

Type: Write only

Reset: Undefined

Description: This location is used to clear bits in the status register. The locations that correspond to external interrupts are valid. The status is cleared only if the interrupt trigger mode is edge sensitive that is the rising edge, falling edge or any edge. To clear the status bit, a clear bit operation is to be performed on this location. Clear bit operation is performing a write operation with logic 1 on data bus corresponding to the locations which have to be cleared.
There are two clear status locations at 0x280 and 0x284.

ILC_ENABLE Enable registers



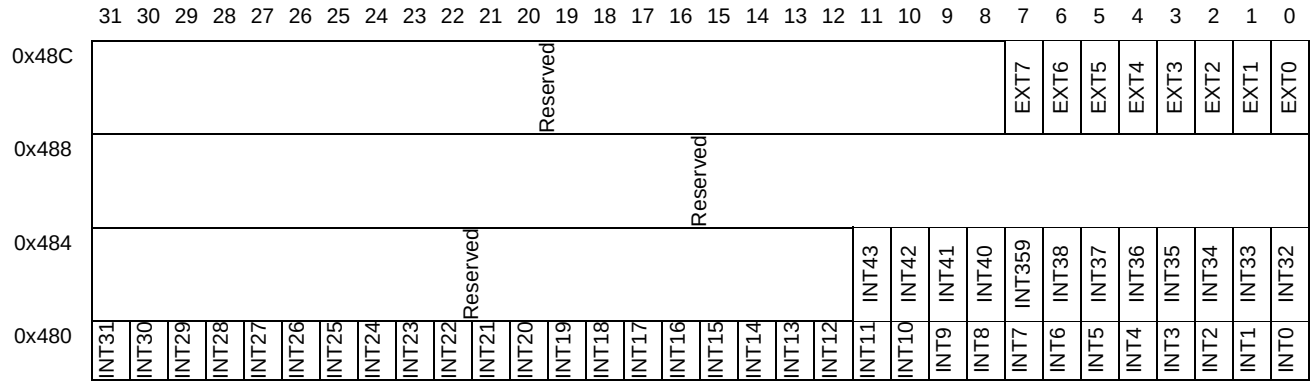
Address: *ILCBaseAddress* + 0x400, 0x404, 0x408 and 0x40C

Type: Read/write

Reset: 0

Description: Interrupt generation from an interrupt number is enabled only if the corresponding bit in the enable register is set to logic 1. The contents of this register are at logic 0 on reset. There are two enable registers at 0x400 and 0x404.

ILC_CLEAR_ENABLE Clear enable locations



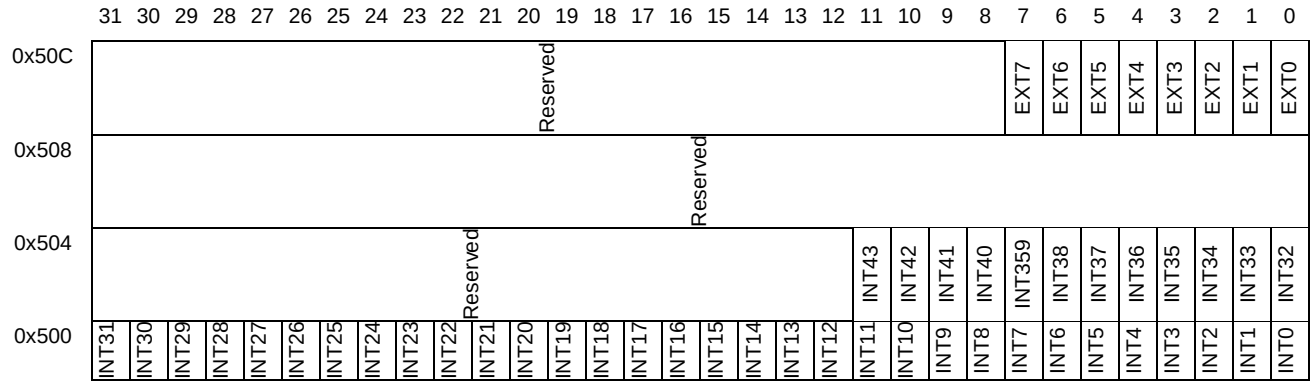
Address: *ILCBaseAddress* + 0x480, 0x484, 0x488 and 0x48C

Type: Write only

Reset: Undefined

Description: Any bit in the enable register can be cleared by performing a clear bit operation on the appropriate location.
There are two locations at 0x480 and 0x484 corresponding to respective enable registers.

ILC_SET_ENABLE Set enable locations



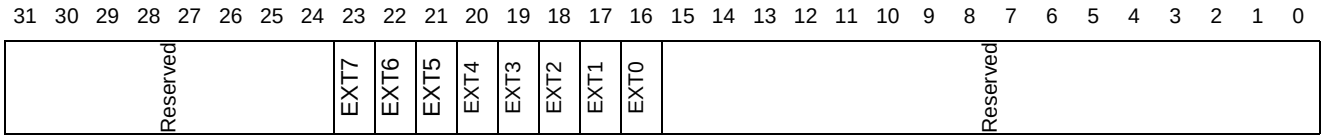
Address: *ILCBaseAddress* + 0x500, 0x504, 0x508 and 0x50C

Type: Write only

Reset: Undefined

Description: Any bit in the enable register can be set by performing a set bit operation on the appropriate location (set bit operation is performing a write operation with logic 1 on the data bus corresponding to the location which has to be set).
There are two locations at 0x500 and 0x504 corresponding to respective enable registers.

ILC_WAKEUP_ENABLE Wake up enable registers



Address: *ILCBaseAddress* + 0x608

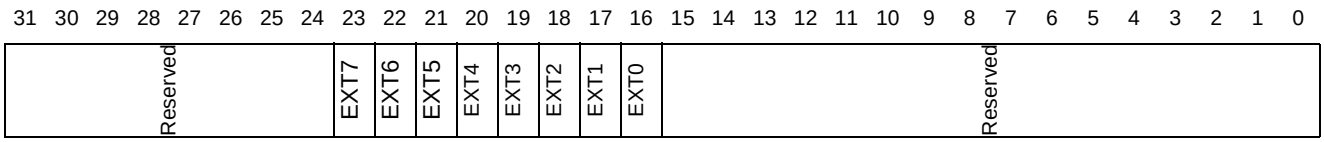
Type: Read/write

Reset: 0

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 logic 1. The contents of this register are set to logic 0 on reset.
The location at 0x604 corresponds to respective interrupt numbers.

Note: Interrupts EXT4 to EXT7 are not valid outputs, see Table 27: *STx5119 interrupt assignments* on page 61 *for a list of interrupt numbers.*

ILC_WAKEUP_ACTIVE_LEVEL Wake up level registers



Address: *ILCBaseAddress* + 0x688

Type: Read/write

Reset: 1

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 then a wake up by interrupt is generated, only if the corresponding external interrupt is at logic 1. Writing logic 0 generates the wake up by interrupt when corresponding external interrupt pin is at logic 0. The contents of this register are set to logic 1 on reset.
The location at 0x688 corresponds to respective interrupt numbers.

Note: Interrupts EXT4 to EXT7 are not valid outputs, see Table 27: *STx5119 interrupt assignments* on page 61 *for a list of interrupt numbers.*

ILC_PRIORITYn Priority 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																Priority levels 0 to 15 Bits 14:4 must be set to 0 at all times															

Address: *ILCBaseAddress* + 0x800 + (n x 0x008)

Type: Read/write

Reset: 0

Description: The priority register assigns the interrupt number to one of the available interrupt levels. The assignment is done by writing 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. ILC can only map on to four interrupt levels of an external processor, therefore the values between 0x8000 to 0x8003 are valid. Other values from 0x8004 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.
The register is set to 0x0000 on reset. The address of the priority register corresponding to interrupt number n is at 0x800 + (n x 0x008).

ILC_MODEn Mode 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																													MODE2	MODE1	MODE0

Address: *ILCBaseAddress* + 0xA04 (Mode0), 0xA0C (Mode1), 0xA14 (Mode2)

Type: Read/write

Reset: 0

Description: The mode register is used to program the trigger mode for a given interrupt number. This is a 3-bit register. The mode register is present only for external interrupts. No mode register is present for internal interrupts. All the internal interrupt numbers are assumed active high.

- The trigger modes can be programmed to appropriate trigger levels by writing the values as shown in the table below. Use the syntax 0xnnnn nnnn for the address.
- Include the values outside the address range.

[31:3] Reserved

[2:0] **MODE[2:0]**

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

The mode register is set to 0x00 on reset.

12 Memory system

The STx5119 has a fully unified memory system.

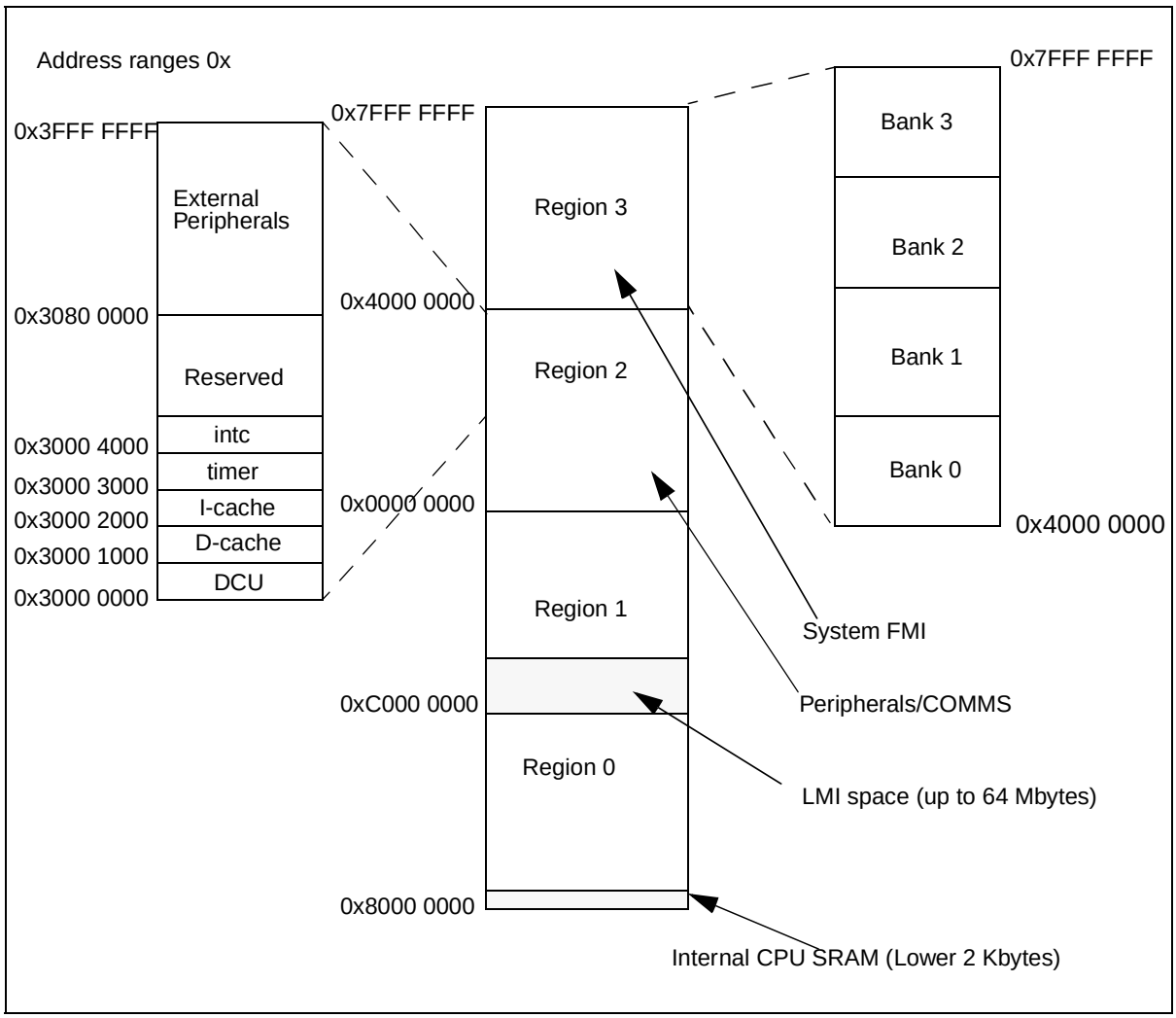
12.1 Memory map

The memory space is divided into four regions:

- region 0: addresses which map onto the C106 embedded SRAM,
- region 1: addresses which map onto the off-chip SDRAM connected to the LMI,
- region 2: addresses which map onto the on-chip peripheral configuration registers,
- region 3: addresses which map onto the flash and peripheral memory interface (FMI).

Figure 18 gives the top level address map breakdown.

Figure 18: Memory map



Region 0 is decoded inside the C106 so CPU accesses to that region are not passed out to the STBus. For that reason region 0 contains only the CPU SRAM. See Table 29.

Table 29: Address map

Address range		Size	Sector	Region type	Subregion type
From	To				
0xC000 0000	0xFFFF FFFF	1 Gbyte	1	LMI buffer	LMI
0x8000 0000	0xBFFF FFFF	1 Gbyte	0	Internal SRAM	Internal SRAM (bottom 2 Kbytes) Address is wrapped around[1]
0x7FC0 0000	0x7FFF FFFF	3.9 Mbytes	3	FMI	FMI bank 3
0x7F80 0000	0x7FBF FFFF	3.9 Mbytes			FMI bank 2
0x7F40 0000	0x7F7F FFFF	3.9 Mbytes			FMI bank 1
0x4000 0000	0x7F3F FFFF	1012 Mbytes			FMI bank 0
0x3080 0000	0x3FFF FFFF	247 Mbytes	2	Peripherals	External peripheral (reserved)[1]
0x3000 5000	0x307F FFFF	247Mbytes			Reserved[1]
0x3000 4000	0x3000 4FFF	4 Kbytes			INTC[1]
0x3000 3000	0x3000 3FFF	4 Kbytes			Timer[1]
0x3000 2000	0x3000 2FFF	4 Kbytes			I-Cache[1]
0x3000 1000	0x3000 1FFF	4 Kbytes			D-Cache[1]
0x3000 0000	0x3000 0FFF	4 Kbytes			DCU[1]
0x2110 0000	0x2FFF FFFF	238 Mbytes			Reserved
0x2100 0000	0x210F FFFF	1 Mbyte			Blitter configuration
0x20F0 0000	0x20FF FFFF	1 Mbyte			System services
0x20E0 0000	0x20EF FFFF	1 Mbyte			PTI configuration
0x20A0 0000	0x20DF FFFF	4 Mbytes			Reserved
0x2090 0000	0x209F FFFF	1 Mbyte			TV OUT configuration
0x2080 0000	0x208F FFFF	1 Mbyte			COMMS configuration
0x2070 0000	0x207F FFFF	1 Mbyte			Audio configuration
0x2060 0000	0x2068 FFFF	576 Kbytes			S/PDIF configuration
0x2050 0000	0x205F FFFF	1 Mbyte			Video configuration
0x2040 3000	0x204F FFFF	1 Mbyte			Reserved
0x2030 0000	0x203F FFFF	1 Mbyte			FDMA configuration
0x2020 0000[2]	0x202F FFFF	1 Mbyte			System FMI configuration
0x0000 0000	0x201F FFFF	514 Mbytes			Reserved

1. Not accessible by external peripherals, Visible only to CPU. Reserved for external DMA
2. To be decoded on to the same port as FMI

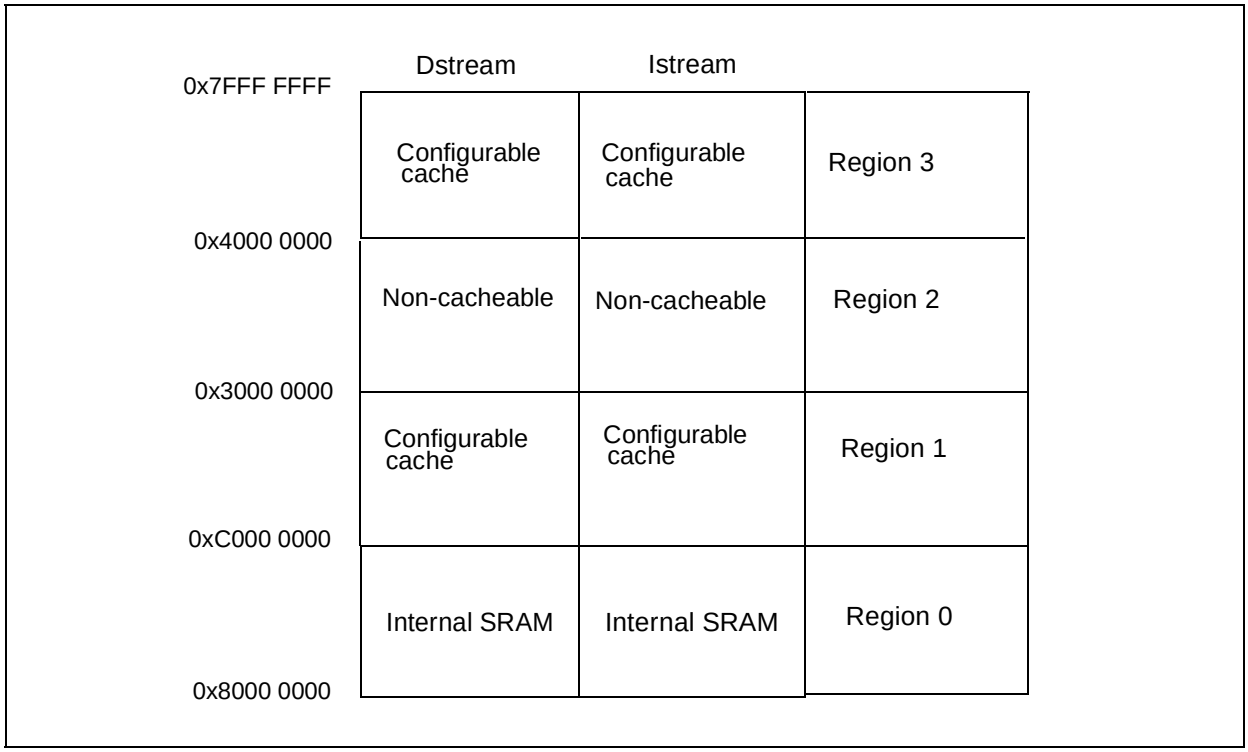
Note: Spaces that are reserved, allow accesses but return unknown data. Reserved spaces must not be programmed as this causes unpredictable behavior.

12.2 CPU cache

There are 4 Kbytes of instruction cache and 4 Kbytes data cache in the STx5119 core. Instruction cache can be accessed only by the CPU fetch unit, and data cache can be accessed only by the CPU execution unit and diagnostic controller unit (DCU).

The caches are direct mapped with write back and are arranged as 256 lines of 16 bytes each. Refill/write back operations always utilise a 16-byte LD/ST access across the interconnect. Selective parts of memory may be configured as cacheable as shown in Figure 19, and all non cacheable requests are arbitrated with the refill/write back stream and compete for the interconnect.

Figure 19: Cacheability



Instruction code may be cached in regions 1 and 3 allowing code to be placed in either the shared memory (LMI space) or FMI space.

The STx5119 has 2 Kbytes of SRAM, this may be accessed by the CPU/DCU. The SRAM cannot be accessed from other STx5119 subsystems outside the core.

13 Cache registers

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

The *DCacheBaseAddress* is:

0x3000 1000

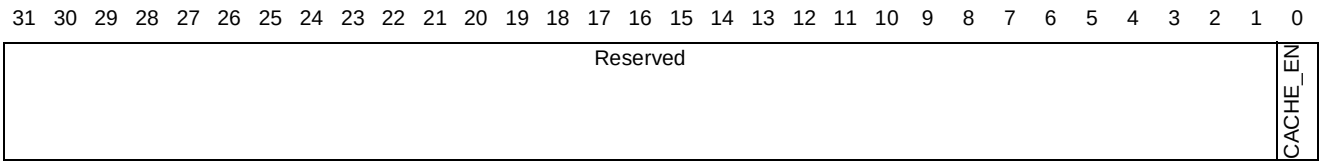
The *ICacheBaseAddress* is:

0x3000 2000

Table 30: DCache and ICache registers

Register name	Description	Address Offset	Type
CACHE_EN	Global cacheing enable, page 79	0x00	RW
INVALIDATE	Start invalidate, page 80	0x10	WO
FLUSH	Start flush, page 80	0x14	WO
STATUS	Monitor status, page 80	0x18	RO
REGION_0_EN	Configure region 0 cacheability, page 81	0x20	RW
REGION_1_BLOCK_EN	Configure region 1 cacheability, page 81	0x28	RW
REGION 1_TOP_EN		0x2C	RW
REGION_2_EN	Configure region 2 cacheability, page 82	0x30	RW
REGION_3_BLOCK_EN	Configure region 3 cacheability, page 83	0x38	RW
REGION_3_BANK_EN		0x3C	RW

CACHE_EN Global cacheing enable



Address: *CacheBaseAddress* + 0x00

Type: R/W

Reset: 0

Description:

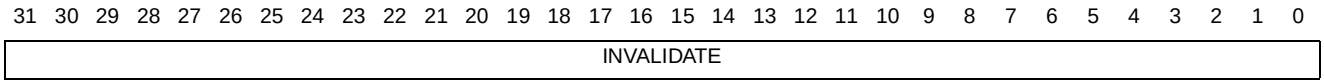
[31:16] **Reserved**

[0] **CACHE_EN**

1: Enabled regions are cacheable according to the other configuration registers.

0: No access is cacheable.

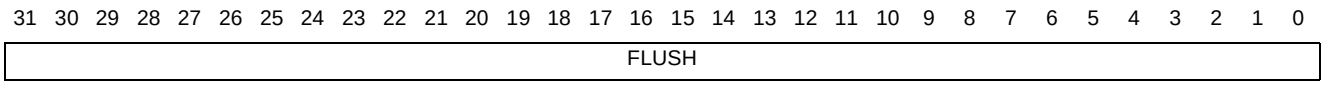
INVALIDATE Start invalidate



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

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

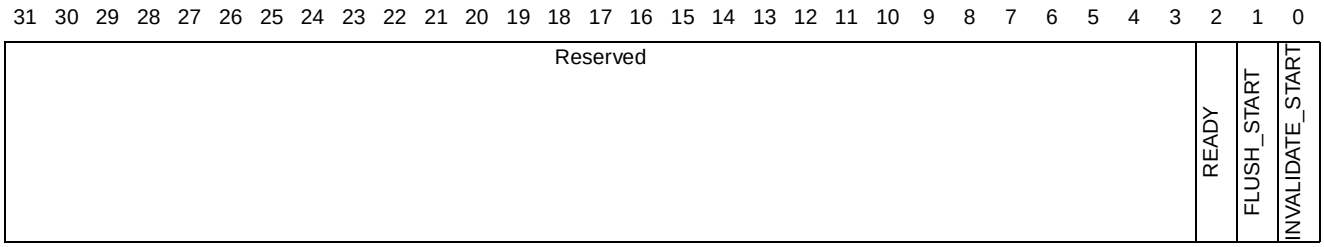
FLUSH Start flush



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:

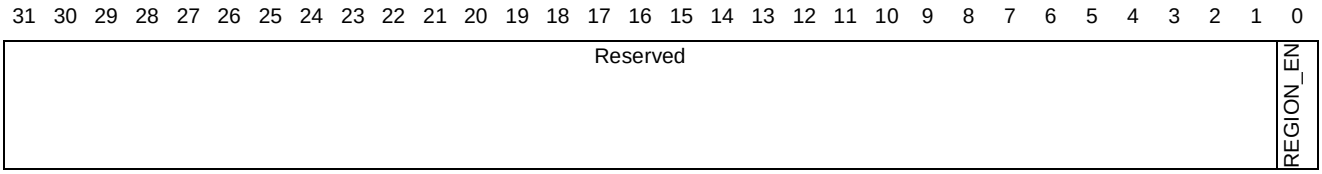
[31:3] **Reserved**

[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_EN Configure region 0 cacheability

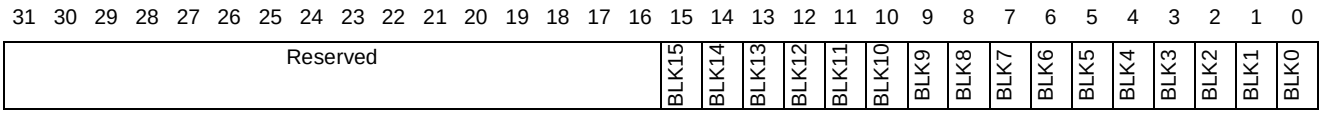


Address: *CacheBaseAddress* + 0x20
Type: R/W
Reset: 0
Description:

[31:1] **Reserved**

[0] **REGION_EN**
1: All of Region 0 (0x8000 0000 - 0xBFFF FFFF) is cacheable.
0: this area is not cached.

REGION_1_BLK_EN Configure region 1 cacheability (block)



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

[31:16] **Reserved**

[15] **BLK15**
0: Not cacheable
1: Block 15 (0xC078 0000 - 0xC07F FFFF) uses cache.

[14] **BLK14**
0: Not cacheable
1: Block 14 (0xC070 0000 - 0xC077 FFFF) uses cache.

[13] **BLK13**
0: Not cacheable.
1: Block 13 (0xC068 0000 - 0xC06F FFFF) uses cache.

[12] **BLK12**
0: Not cacheable.
1: Block 12 (0xC060 0000 - 0xC067 FFFF) uses cache.

[11] **BLK11**
0: Not cacheable.
1: Block 11 (0xC058 0000 - 0xC05F FFFF) uses cache.

[10] **BLK10**
0: Not cacheable.
1: Block 10 (0xC050 0000 - 0xC057 FFFF) uses cache.

[9] **BLK9**
0: Not cacheable.
1: Block 9 (0xC048 0000 - 0xC04F FFFF) uses cache.

[8] **BLK8**
0: Not cacheable.
1: Block 8 (0xC040 0000 - 0xC047 FFFF) uses cache.

[7] **BLK7**
0: Not cacheable.
1: Block 7 (0xC038 0000 - 0xC03F FFFF) uses cache.

[6] **BLK6**
0: Not cacheable.
1: Block 6 (0xC030 0000 - 0xC037 FFFF) uses cache.

[5] **BLK5**
0: Not cacheable.
1: Block 5 (0xC028 0000 - 0xC02F FFFF) uses cache.

[4] **BLK4**
0: Not cacheable. 1: Block 4 (0xC020 0000 - 0xC027 FFFF) uses cache.

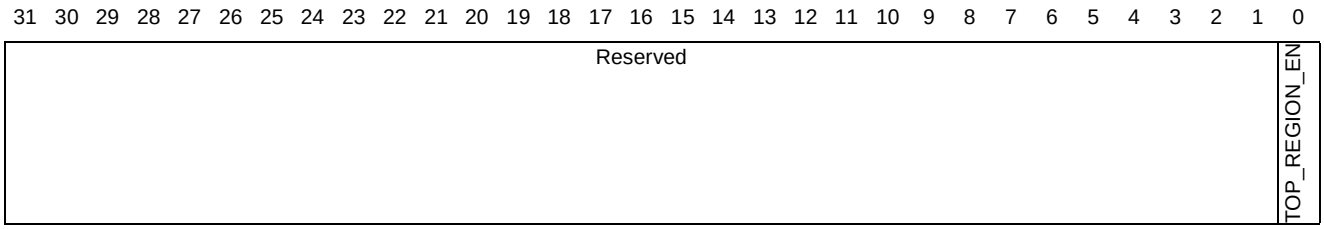
[3] **BLK3**
0: Not cacheable. 1: Block 3 (0xC018 0000 - 0xC01F FFFF) uses cache.

[2] **BLK2**
0: Not cacheable. 1: Block 2 (0xC010 0000 - 0xC017 FFFF) uses cache.

[1] **BLK1**
0: Not cacheable. 1: Block 1 (0xC008 0000 - 0xC00F FFFF) uses cache.

[0] **BLK0**
0: Not cacheable. 1: Block 0 (0xC000 0000 - 0xC007 FFFF) uses cache.

REGION_1_TOP_EN Configure region 1 cacheability (top)

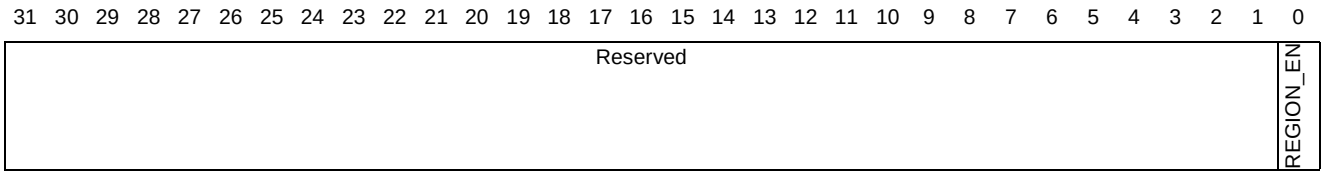


Address: *CacheBaseAddress* + 0x2C
Type: R/W
Reset: 0
Description:

[31:16] **Reserved**

[0] **TOP_REGION_EN**
1: All of top Region 1 (0xC080 0000 - 0xFFFF FFFF) is cacheable.
0: This area is not cached.

REGION_2_EN Configure region 2 cacheability



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

[31:1] **Reserved**

[0] **REGION_EN**
1: All of Region 2 (0x0000 0000 - 0x3FFF FFFF) is cacheable.
0: This area is not cached.

REGION_3_BLK_EN 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																BLK15	BLK14	BLK13	BLK12	BLK11	BLK10	BLK9	BLK8	BLK7	BLK6	BLK5	BLK4	BLK3	BLK2	BLK1	BLK0

Address: *CacheBaseAddress* + 0x38

Type: R/W

Reset: 0

Description:

[31:16] **Reserved**

[15] **BLK15**
0: Not cacheable. 1: Block 15 (0x4078 0000 - 0x407F FFFF) uses cache.

[14] **BLK14**
0: Not cacheable. 1: Block 14 (0x4070 0000 - 0x4077 FFFF) uses cache.

[13] **BLK13**
0: Not cacheable. 1: Block 13 (0x4068 0000 - 0x406F FFFF) uses cache.

[12] **BLK12**
0: Not cacheable. 1: Block 12 (0x4060 0000 - 0x4067 FFFF) uses cache.

[11] **BLK11**
0: Not cacheable. 1: Block 11 (0x4058 0000 - 0x405F FFFF) uses cache.

[10] **BLK10**
0: Not cacheable. 1: Block 10 (0x4050 0000 - 0x4057 FFFF) uses cache.

[9] **BLK9**
0: Not cacheable. 1: Block 9 (0x4048 0000 - 0x404F FFFF) uses cache.

[8] **BLK8**
0: Not cacheable. 1: Block 8 (0x4040 0000 - 0x4047 FFFF) uses cache.

[7] **BLK7**
0: Not cacheable. 1: Block 7 (0x4038 0000 - 0x403F FFFF) uses cache.

[6] **BLK6**
0: Not cacheable. 1: Block 6 (0x4030 0000 - 0x4037 FFFF) uses cache.

[5] **BLK5**
0: Not cacheable. 1: Block 5 (0x4028 0000 - 0x402F FFFF) uses cache.

[4] **BLK4**
0: Not cacheable. 1: Block 4 (0x4020 0000 - 0x4027 FFFF) uses cache.

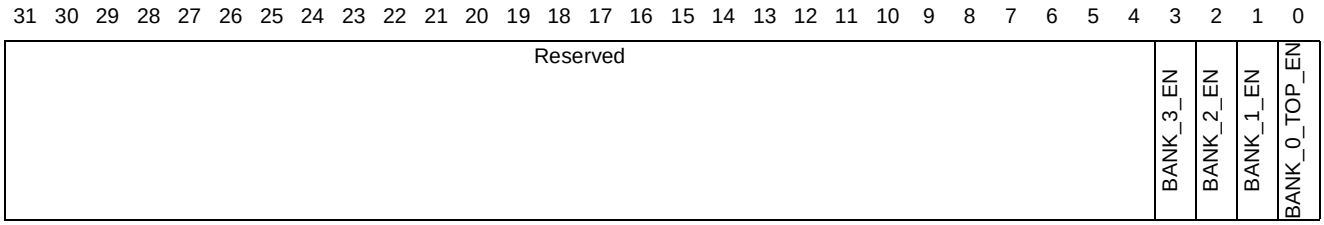
[3] **BLK3**
0: Not cacheable. 1: Block 3 (0x4018 0000 - 0x401F FFFF) uses cache.

[2] **BLK2**
0: Not cacheable. 1: Block 2 (0x4010 0000 - 0x4017 FFFF) uses cache.

[1] **BLK1**
0: Not cacheable. 1: Block 1 (0x4008 0000 - 0x400F FFFF) uses cache.

[0] **BLK0**
0: Not cacheable. 1: Block 0 (0x4000 0000 - 0x4007 FFFF) uses cache.

REGION_3_BANK_EN Configure region 3 cacheability (Bank)



Address: *CacheBaseAddress* + 0x3C

Type: R/W

Reset: 0

Description:

[31:4] **Reserved**

[3] **BANK_3_EN**
1: All of bank 3 (0x7000 0000 - 0x7FFF FFFF) is cacheable.
0: This area is not cached.

[2] **BANK_2_EN**
1: All of bank 2 (0x6000 0000 - 0x6FFF FFFF) is cacheable.
0: This area is not cached.

[1] **BANK_1_EN**
1: All of bank 1 (0x5000 0000 - 0x5FFF FFFF) is cacheable.
0: This area is not cached.

[0] **BANK_0_TOP_EN**
1: Top of bank 0 (0x4080 0000 - 0x4FFF FFFF) is cacheable.
0: This area is not cached.

14 Local memory interface (LMI)

14.1 Introduction

The local memory interface (LMI) provides the interface between the STBus and the external main memory SDRAM.

The SDRAM controller comprises:

- 16-bit external bus width,
- a single quad bank SRD SDRAM device,
- SDRAM technology: 64-Mbit, 128-Mbit, 256-Mbit, 512-Mbit (main memory size from 8-Mbytes to 64-Mbytes).

14.2 STBus interface

The functionality of the STBus interface is described in the following subsections.

14.2.1 STBus port

The STBus port is divided into one register block and *n* number of data blocks. Each data block is 16 Mbytes in size. The address range of the LMI is defined between the memory base address (0xC000 0000) and the register base address (0xE000 0000).

Data blocks populate starting from 0xC000 0000 inclusive. The upper boundary address of data blocks is defined by the UBA field of the LMI_SDRA[0:1] register (the SDRAM array attribute register's upper bound address). See Section 14.3.1: *Main memory configuration* on page 86)

The control populate starts from 0xE000 0000 inclusive to 0xE000 0050. Table 31 shows the LMI address space organization in the STx5119.

Table 31: LMI address space for the STx5119

	LMI registers	LMI data blocks
Lowest address	0xE000 0000	0xC000 0000
Highest address	0xE000 0050	UBA

When the LMI is active, memory accesses from the STBus to the data blocks cause accesses to be made on the external memory bus. Each external memory access consists of a number of phases, each representing an action performed on the external memory bus. The behavior of the external pins of the LMI can be programmed to allow the LMI to drive the external memory bus in an appropriate way for different SDRAM parts as shown in Table 32 and Table 33.

Table 32: SDRAM output impedance setting

LMI_CLK_OFFSET_CTRL[15:14]	Output buffer impedance range
00	25 Ω
01	40 Ω
10	55 Ω
11	70 Ω (default)

Table 33: SDRAM pad drive strength control

LMI_CLK_OFFSET_CTRL[17:16]	MODE =1 2.5 V series terminated configuration
00	5 pF
01	15 pF (default)
10	25 pF
11	35 pF

14.2.2 Coherency

The memory of the LMI is coherent as viewed from the STBus. All requests are processed sequentially on the local memory interface in the order of the receipt of those requests by LMI. However, the system can decouple the generation of response packets from the actual external memory bus accesses. For store transactions, the corresponding store response packets may not be returned to the initiator on the STBus until the write operations are actually completed on the SDRAM interface. Since the local system is the sole owner of the external main memory and all the requests to the same address are processed in order (as they are received from the STBus interface) on the SDRAM interface, memory coherency is achieved.

A swap packet comes with store data. When processing a swap request, the LMI initiates a read transaction on the SDRAM interface. Once the data is received by the SDRAM controller, a write command is issued.

14.3 SDRAM interface

The LMI's SDRAM controller supports one SDRAM device. The functionality of the SDRAM interface is described in the following subsections.

14.3.1 Main memory configuration

The LMI supports a single memory device which is 16 bits wide. The population on the array ranges from 16 Mbytes to 64 Mbytes.

The memory location is placed in the UBA field in the LMI_SDRA0 register. NOT_LMICS is asserted if the STBus request access to the LMI's data block and the request address [31:21] is less than UBA in LMI_SDRA0.

Note: The LMI_SDRA0 and LMI_SDRA1 registers must contain the same values.

14.3.2 SDRAM interface pins

The external pins are described in Table 34.

Table 34: SDRAM interface pins

Name	I/O	Size	Description
LMIMCLKOUT	Output	1	SDRAM clock output
NOT_LMICLKOUT	Output	1	NOT_LMICLKOUT are differential clock outputs to 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.
NOT_LMICS	Output	1	Chip select Perform the function of selecting the particular SDRAM components during the active state

Table 34: SDRAM interface pins

Name	I/O	Size	Description
LMIRDNOTWR	Output	1	Write enable signal WE asserted during writes to SDRAM
LMIADDR[12:0]	Output	13	Row and column address
LMIBA[1:0]	Output	2	Bank address
LMIDATA[15:0]	I/O	16	Memory data
LMIDQS[1:0]	I/O	2	Input/output data strobe These pins provide the read and write data strobe signal to/from the receiver circuit of DRAM controller. LMI drives LMIDQS pins in write (store) cycles, while SDRAM drives it in read (load) cycles.
LMIDQM[1:0]	Output	2	Input/output data mask These pins act as byte enables during write cycles.
NOT_LMIRAS	Output	1	Row address strobe The NOT_LMIRAS signal generates encoded SDRAM commands.
NOT_LMICAS	Output	1	Column address strobe The NOT_LMICAS signal generates encoded SDRAM commands.
LMIVREF	Input	1	Input reference voltage

14.3.3 SDRAM devices

The LMI splits the physical memory address into banks, row and column addresses. The LMI contains 13 external address pins. LMIBA[1:0] specifies which bank, while LMIADDR[12:0] indicates row and column addresses in each bank. The row address selects a page in an SDRAM. The column address selects a datum in a row. The LMI supports memories where row addresses are up to 13 bits.

Table 35 summarizes the SDRAM devices which are used to construct the memory subsystem. They also illustrate LMIADDR pin MUXing against the SDRAM address split. The LMI's LMIADDR[12:0] pins are directly connected to SDRAM's A[12:0]. The address split column in the table specifies the row and column address split within a given bank.

Note: The mapping of BA0 and BA1 (bank select address bits) described in Table 35 on page 87 *can be bypassed enabling the bank remapping feature (see the* ENABLE_BA0 *and* ENABLE_BA1 *fields of* LMI_SDRA[0:1] *registers.*

Table 35: Row and column addressing for memory size and number of banks

SDRAM type	Address split	Page size	Array size	RAS CAS	BA 1	BA 0	MA12	MA11	MA10/ AP	MA9	MA8	MA[7:0]
128 Mbit 4 bank												
8 Mbit x 16	12 x 9	1 Kbyte	16 Mbytes	RAS CAS	12 12	11 11		10	23 AP	22	21 9	[20:13] [8:1]
256 Mbit 4 bank												
16 Mbit x 16	13 x 9	1 Kbyte	32 Mbytes	RAS CAS	12 12	11 11	10	24	23 AP	22	21 9	[20:13] [8:1]
512 Mbit 4 bank												
16 Mbit x 16	13 x 10	2 Kbytes	64 Mbytes	RAS CAS	12 12	11 11	25	24	23 AP	22 10	21 9	[20:13] [8:1]

14.3.4 Initializing SDRAM devices

An initialization sequence to an SDRAM device must be done after power-on reset by the driver software. The operating system bootup code or driver software to initialize SDRAM should not read or write SDRAM before completion of the initialization.

1. The system provides four power in certain sequence. VDD first, VDDQ next then VREF and VTT. VTT is not provided to the LMI. It is externally connected to DQ, DQS and other pins through a series of termination registers. This is required to prevent latch-up in SDRAM devices. LMI should be able to support this power-up sequence. During and after power-on reset CKE 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. CKE should be brought high with the **deselect** command. After this point, unless the LMI sends some command, the LMI has to send the **deselect** command.
4. A precharge all (**pall**) is issued to SDRAM.
5. A mode register set (**mrs**) command is issued to program the extended mode register to enable the DLL.
6. The **mrs** command is issued to program the mode register (LMI_SDMR), reset the DLL in SDRAM and program the operating parameters, for example burst length and CAS latency.
7. A precharge all (**pall**) is issued to SDRAM.
8. Wait ten cycles after the DLL reset and send two **cbr** commands to SDRAM.
9. An **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 autocleaning of the DLL initialization bit.
10. After 200 cycles from DLL reset, external memory becomes accessible.

The LMI's SDRAM controller provides two mechanisms for accomplishing the initialization sequence.

1. **nop, pall, ckeh** and **cbr**
   The LMI_SCR register's SMS (SDRAM mode select) field is written with appropriate values to prompt the SDRAM controller to start issuing to start issuing one of these commands. For instance, when SMS = 3'b100, it results in a single CBR cycle on the SDRAM interface. When SMS = 3'b011, it results in the CKE signals going high.
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 the LMI_MIM register, and then a write to the LMI_SDMR register in the control block. The SDRAM controller then issues an **mrs** command to all SDRAM devices on array (array [n]).

Example: issuing an mrs command

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

- STBus address = 0xC000 0048,
- STBus data bus [15:0] contains the value to be written to the SDRAM's mode register,
- data[9:0] is copied to MA[9:0], data[15:12] to MA[13:10] and data [11:10] to BA[1:0] when an **mrs** command is issued to the SDRAM devices.

Software needs to ensure that the SDRAM timing specification (between the **mrs** command and the first operational command) is met. One way to ensure this is to perform several SDRAM control register reads.

Subsequently, SMS in the SCR register is written with 3'b000, the normal SDRAM operation can then be started.

Note: Software must program the LMI_MIM *register before writing to* LMI_SDMR*.*

14.3.5 Operations

The SDRAM controller supports most SDRAM commands. Table 36 lists all commands supported.

Table 36: SDRAM command truth table

Function	Symbol	CKE [n - 1]	CKE [n]	NOT_ CS	NOT_ RAS	NOT_ CAS	NOT_ WE	MA11	AP[1] (MA10 /MA8)	BA [1:0]	MA [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	V	X
Precharge all banks	**pall**	H	X	L	L	H	L	X	H	X	X
Autorefresh	**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

1. AP pin: LMI uses the BY32AP bit in the MIM register to determine if the MA8 pin is used to indicate **pre** and **pall** commands.

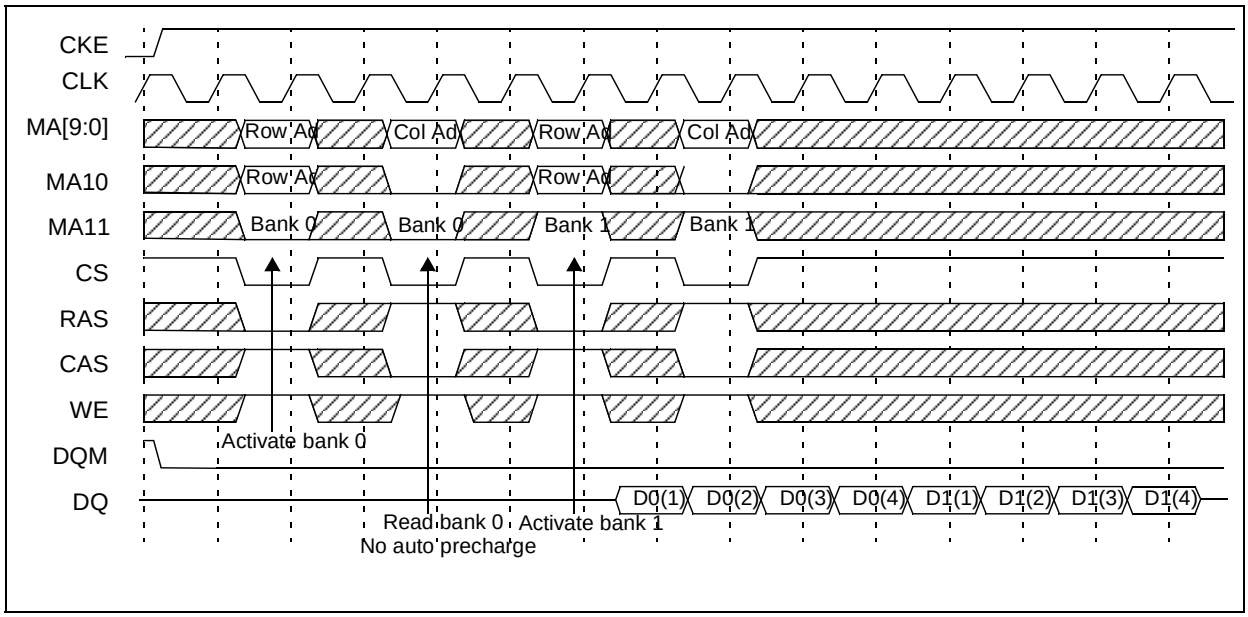
Note: The LMI does not support full-page burst operation. The LMI issues a **bst** *command to terminate the burst read-only in SDRAM mode.*

The timing for issuing these commands is governed by the SDRAM timing register LMI_STR. The LMI's SDRAM controller can open up to four pages for each SDRAM array and fully exploit the multibank 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.

Multibank ping-pong transaction

Two bank ping-pong access is illustrated in Figure 20. The peak bandwidth is obtained in this scenario.

Figure 20: SDRAM two bank ping pong read



14.3.6 Refresh

When DRAM refresh enable is 1 (DRE in LMI_MIM = 1), The LMI can automatically generate refresh cycles. A 12-bit quantity (DRI, DRAM refresh interval in LMI_MIM) 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 DRI's 12-bit value into an internal counter. This counter is decremented by 1 in each memory clock cycle. When the counter reaches 0, DRI's value is reloaded into the counter and the next refresh interval is started.

All banks must be closed before refresh operation can be performed. The SDRAM controller issues a precharge all (**pall**) command if there are any open pages. The SDRAM controller then issues an auto refresh command (**cbr**) after the TRP parameter is satisfied. The next row **act** command can be issued TRFC clock (SRFC in LMI_STR) later.

The SDRAM controller performs 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 1,562. For a 100 MHz memory clock, then this allows 1,024 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.

14.3.7 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 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. At first, no initiators should be issuing transaction requests to the LMI.The LMI does not de-assert the grant signal during the standby so it is mandatory not to send any requests during this phase.
2. The standby management program should issue a **cbr** command as the last command to LMI.
3. The standby management program asserts STBY_REQ to LMI.
4. All outstanding transaction requests are serviced.
5. The SDRAM controller issues a self-refresh command and lowers LMICLKEN TO SDRAM. The SDRAM autonomously refreshes itself for the duration of the power-down mode.
6. LMI asserts STBY_ACK to PMU. The memory clock (MCLKO) can now be stopped. The clock controlling the padlogic FFs can be switched off when STBY_REQ (provided by PMU) and STBY_ACK (provided by the LMI core) are both asserted (STBY_REQ/STBY_ACK has to be combined to act as enable signal of gated clock cell outside the LMI core).

*Note: The LMI core switches off both its clocks (memory and bus clocks) under the same conditions (*STBY_REQ *and* STBY_ACK *both high).*

Leaving standby causes other than power-on reset

1. PMU de-asserts STBY_REQ. As an effect of this, the 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) is implemented outside the LMI core.
2. The LMI de-asserts STBY_ACK, and starts to count down from (256 x CST (SCR)) to zero every memory clock cycle.
3. When countdown reaches zero, the SDRAM controller asserts the LMICLKEN pin and sends **deselect** commands continuously. SDRAM exits from self-refresh mode.
4. The LMI starts to count down from (16 x XSR(STR)) to zero every memory clock cycle. In this period no SDRAM command is sent to the memory.
5. For SDRAM, the LMI issues numbers of **cbr** commands defined by the BRFSH field in CSR.

15 Local memory interface (LMI) registers

If the LMI is active, the transactions to the registers are processed only when there are no outstanding data block transactions. While the LMI is processing control block transactions, the SDRAMs are in idle state. After processing control block transactions, the LMI's DRAM controller then continues with its normal behavior which reflects the state of the control registers. The LMI ensures that the change from the original behavior to the subsequent behavior is achieved instantaneously at a boundary between SDRAM commands during the processing of that transaction.

Addresses are provided as the *LMIBaseAddress* + offset.

The *LMIBaseAddress* is:

0xE000 0000.

Table 37: LMI registers

Register name	Description	Address offset	Type
LMI_MIM	Memory interface mode, see page 93	0x0008	R/W
LMI_SCR	SDRAM control, see page 94	0x0010	R/W
LMI_STR	SDRAM timing, see page 95	0x0018	R/W
LMI_CLK_OFFSET_CTRL	LMI clock offset control register, see page 96	0x0028	R/W
LMI_SDRA0	SDRAM array attribute, see page 97	0x0030	R/W
LMI_SDRA1	SDRAM array attribute, see page 97	0x0038	R/W
LMI_SDMR0	SDRAM array mode, see page 98	0x0048	WO
LMI_SDMR1	SDRAM array mode, see page 98	0x0050	WO

LMI_MIM Memory interface 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				DRI												Reserved				B732AP	DIMM	DRE	Reserved				COMB8	COMB16	Reserved		DCE

Address: *LMIBaseAddress* + 0x0008

Type: Read/write

Reset: 0 (except for DRI: 0x061A

Description: The LMI_MIM register specifies the configuration of the DRAM interface.

[31:28] **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. Precharges all bank command's indicator for x32 SDRAM devices.
0: MA8 pin is not used when LMI issues **pre** or **pall** commands.
1: MA8 pin is used when LMI issues **pre** or **pall** commands.

[10] **DIMM:** Registered DIMM module. Constructs the external array.
1: Data output delayed by 1 memory clock cycle and 1 memory clock cycle added to CAS latency

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

[8] **Reserved**

[7:5] **Reserved**
Must be set to 00 (reset value is 10)

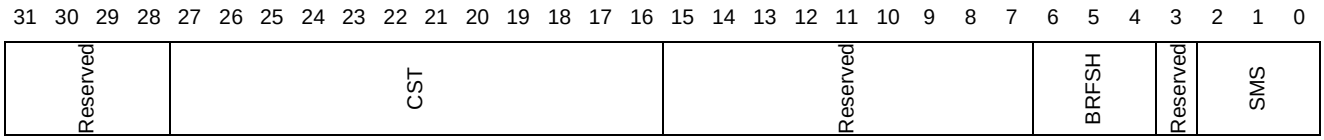
[4] **COMB8:** group two load/store 8 transactions in one load/store 16 when addresses are suitable.
0: Grouping disabled 1: Grouping enabled

[3] **COMB16:** group two load/store 16 transactions in one load/store 32, when addresses are suitable.
0: Grouping disabled 1: Grouping enabled

[2:1] Must be set to 00 for SDR 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



Address: *LMIBaseAddress* + 0x0010
Type: Read/write
Reset: 0
Description:

[31:28] Reserved

[16:27] **CST:** Clock stabilization time. These bits specify clock stabilization time on return from module standby mode.
0: There is no stabilization time before LMI start operation with SDRAM.
<>0: Count down this number with 1/256 memory clock. When it reaches 0, the 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] **Reserved**

[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 DCE in LMI_MIM register = 1
001: **nop** command enable
When SMS is written with this value and DCE: 1, the LMI issues one **nop** command to the SDRAM interface. To have [n] number of **nop** commands, SMS must be written with 001 *n* times.
010: Precharge all banks
When SMS is written with this value and DCE: 1, the LMI issues one **pall** command to the SDRAM interface. To have [n] number of **pall** commands, SMS must be written with 010 *n* times.
011: Clock enable signal LMICLKEN is active. At reset the clocks are disabled.
100: **cbr** enable
When SMS is written with this value and DCE: 1, the LMI issues one **cbr** command to the SDRAM interface. To have *n* number of **cbr** commands, SMS must be written with 011 *n* times.
101, 110, 111: Reserved

LMI_STR SDRAM timing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SXSR				SRCDW		SWTR	DISABLE_TWTR	SDPL1	SRFC				SCL				SDPL0	SRRD	SRAS			SRC				SRCDR		SRP	

Address: *LMIBaseAddress* + 0x0018

Type: Read/write

Reset: 0

Description:

[31:30] **Reserved**

[29:26] **SXSR**: Exit self-refresh to next command. Controls the number of memory clock cycles from self-refresh exit to next command. These bits specify the number of cycles to wait after the self-refresh exit command before starting to send again commands to the SDRAM.
0: There is no time to wait.
<>0: Count down this number with 1/16 memory clock. When it reaches 0, LMI starts operations

[25:24] **SRCDW**: TRCDW, RAS to write CAS delay. Controls the number of memory clock cycles from a row activate command to a write column command (for the same bank) for write operations.
00: 2 clocks of RAS to CAS delay 01: 3 clocks of RAS to CAS delay
10: 4 clocks of RAS to CAS delay 11: 5 clocks of RAS to CAS delay

[23] **SWTR**: TWTR, write-to-read interruption. TWTR parameter: Last data in (during write) to read command parameter.
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] **SDPL1:** See SDPL0

[20:17] **SRFC**: TRFC, autorefresh RAS cycle time. Minimum delay between autorefresh and ACT (to the same bank) autorefresh and autorefresh (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 memory clock cycles between when a read command is sampled by the SDRAMs and when the processor samples read data from SDRAMs
0010: 2 clocks
All others: Reserved

[12] **SDPL0**: SDRAM's TWR.
From the end of postamble to **pre** or **pall** command SDPL0, SDPL1
00: 1 clock 01: 2 clocks
10: 3 clocks 11: 4 clocks

[11] **SRRD**: TRRD, RAS to RAS active delay. Specifies delay from ACT bank *n* to ACT bank *m* command (different bank).
0: 2 clocks
1: 3 clocks

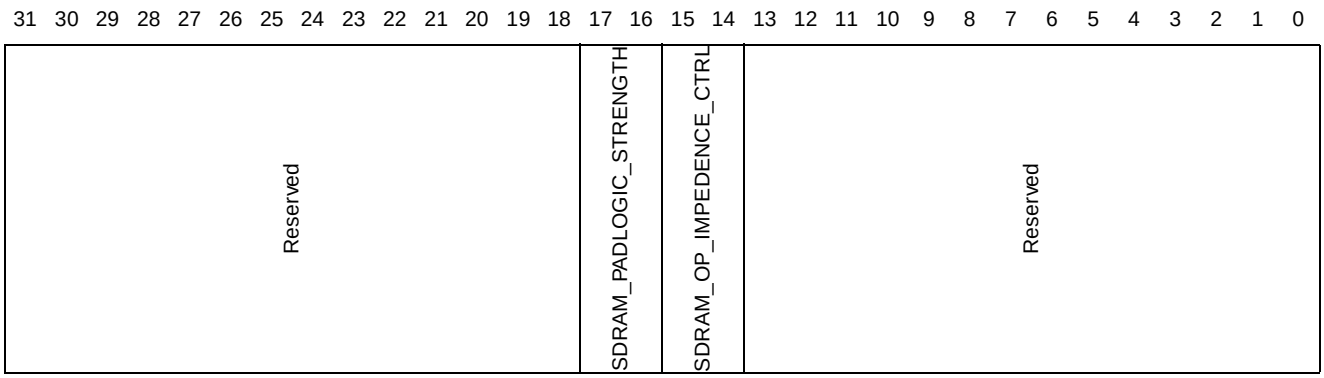
[10:8] **SRAS**: TRAS, RAS active time. ACT to PRE command (for the same bank).
001: 5 clocks
010: 6 clocks
011: 7 clocks
100: 8 clocks
101: 9 clocks
110: 10 clocks
All others: Reserved

[7:4] **SRC**: TRC, RAS cycle time. Minimum delay between ACT and autorefresh (to the same bank), ACT and ACT (to the same bank).
0000: 6 clocks
0001: 7 clocks
0010: 8 clocks
0011: 9 clocks
0100: 10 clocks
0101: 11 clocks
0110: 12 clocks
0111: 13 clocks
1000: 14 clocks
1001: 15 clocks
All others: Reserved

[3:2] **SRCDR**: TRCDR, RAS to read CAS delay. Controls the number of memory clock cycles from a row activate command to a column command (for the same bank) for read operations
00: 2 clocks of RAS to CAS delay.
01: 3 clocks of RAS to CAS delay
10: 4 clocks of RAS to CAS delay
11: 5 clocks of RAS to CAS delay

[1:0] **SRP**: TRP, RAS precharge to ACT command. Controls the number of memory clock cycles 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_CLK_OFFSET_CTRL LMI clock offset control register



Address: *LMIBaseAddress* + 0x0028
Type: Read/write
Reset: 0x1C000
Description:

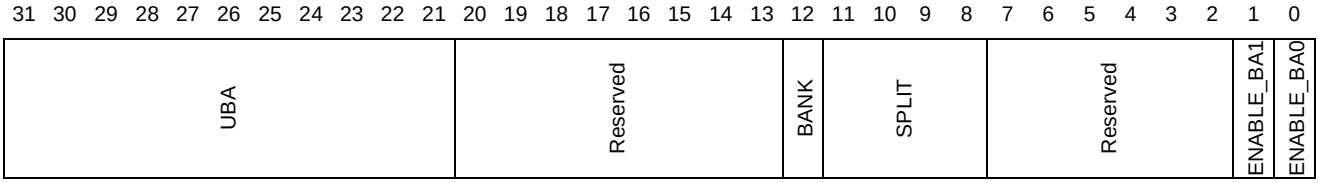
[63:18] **Reserved**

[17:16] **SDRAM_PAD_STRENGTH**
00: 5 pF
01: 15 pF
10: 25 pF
11: 35 pF

[15:14] **SDRAM_OP_IMPEDENCE_CTRL**
00: 25 Ω
01: 40 Ω
10: 55 Ω
11: 70 Ω

[13:0] **Reserved**

LMI_SDRA0 and LMI_SDRA1 SDRAM array attribute



Address: *LMIBaseAddress* + 0x0030 (SDRA0) and 0x0038 (SDRA1)

Type: Read/write

Reset: 0

Description: LMI_SDRA0 and LMI_SDRA1 must be set to the same values.

[31:21] **UBA**: array upper boundary address. Defines the upper boundary address of the external SDRAM array in 2 Mbytes granularity.
UBA specifies the external array's upper boundary address [31:21].

[20:13] **Reserved**

[12] **Bank.** Always set to 1.

[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: Reserved
0001: Reserved
0010: Reserved
0011: Reserved
0100: 12 x 8
0101: 12 x 9
0110: Reserved
0111: Reserved
1000: Reserved
1001: 13 x 9
1010: 13 x 10
1011: Reserved
11nn: Reserved

[7:2] **Reserved**

[1] **ENABLE_BA1**: enable/disable the bank remapping feature for BA1.
0: Disable bank remapping feature for BA1. The bank select bit is calculated as in Table 35 on page 87.
1: Enable bank remapping feature for BA1. Bit 9 of STBus address is used as bank select 1.
If this feature is enabled bit 9 of the incoming STBus address is swapped with the STBus address bit that the LMI uses as BA1 according to its configuration as in Table 35 on page 87. This swap is done before the STBus address is processed by the LMI and only for external memory accesses. In other words the swap is not done during a modeset operation or internal register accesses.

[0] **ENABLE_BA0**: enable/disable the bank remapping feature for BA0.
0: Disable bank remapping feature for BA0. The bank select bit is calculated as in Table 35 on page 87.
1: Enable bank remapping feature for BA0. Bit 8 of STBus address is used as bank select 0
If this feature is enabled bit 8 of the incoming STBus address is swapped with the STBus address bit that the LMI use as BA0 according to its configuration as in Table 35 on page 87.
This swap is done before the STBus address is processed by the LMI and only for external memory accesses. In other words the swap is not done during a modeset operation or internal register accesses.

LMI_SDMR[1:0] SDRAM array mode

Address: *LMIBaseAddress* + 0x0048, + 0x0050

Type: Write only

Reset: Undefined

Description: This register is a write-only virtual register, since physically it is not contained in the processor chip. A write to this virtual register triggers an SDRAM mode register set command to be issued to an SDRAM device.
The value on STBus data bus TARGET_1_DATA[9:0] is copied to the MA[9:0] pins, TARGET_1_DATA[11:10] is output to BA[1:0] and TARGET_1_DATA[15:12] is driven to MA[13:10].
In response to the mode register set command, the SDRAM device then latches MA[11:0] and BA[1:0] into its mode register. A read to this register returns an undefined value.
Please refer to the SDRAM manufacturer's datasheet 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.

Caution when programming SDRAM's mode register

To effectively support STBus load 32 and store 32 packets, the LMI's SDRAM controller uses DT (SDRAM type) and BW (external data bus width) in the LMI_MIM register to determine the burst length.

Table 38: Determining burst length using the LMI_MIM register

Bus width (BW bit)	Burst length
00: 16-bit	8
01: 32-bit	8
10: 64-bit	4

For a 16-bit external data bus width, the LMI splits an STBus load 32 or store 32 packet into multiple SDRAM transactions, with a burst of 8 for each transaction. Therefore the BL field of the SDRAM device's mode register must be programmed to match the LMI's burst length behavior in the third column.

16 Flash and peripheral memory interface (FMI)

The FMI is a highly flexible memory device that 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 four possible external memory banks.

The position, size, clock frequency and memory type supported is dependent on how the associated control registers, FMI_BANK[0:3], 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.

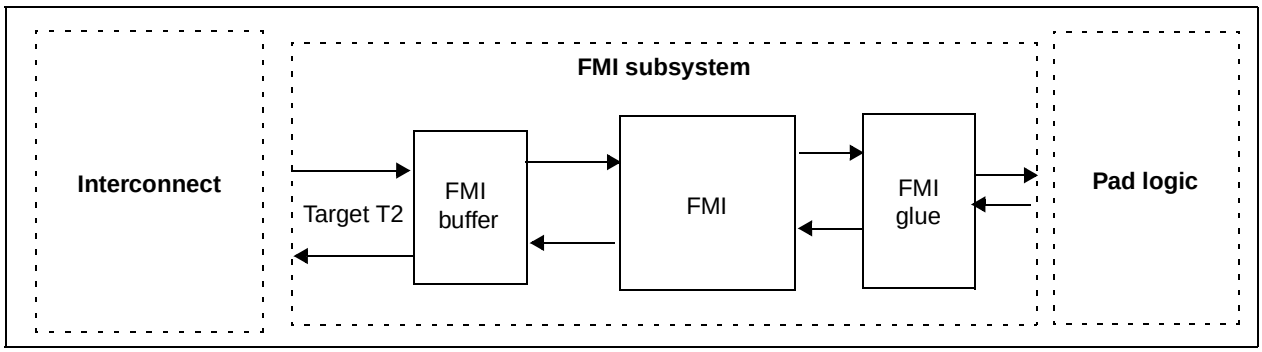
As part of the boot process, the FMI configuration registers should be programmed to match the memory supported in that system, defining the memory size, the location in the address and the device type connected. Each bank can only accommodate one type of device, but different device types can be placed in different banks to provide glueless support for mixed memory systems. Only 16-bit flash memory can be used to boot.

The STx5119 FMI features:

- FMI as bus master,
- support for four banks of 64 Mbytes, each of which may be independently configured,
- support for synchronous or asynchronous flash,
- asynchronous support for peripherals and SRAM,
- burst flash support (AMD AM29BL162C, STMicroelectronics M58LW064, Intel 28F160F3 compatible),
- SRAM support,
- 16-bit width data bus (boot is always from a 16-bit device),
- 2 Mbyte contiguous addressing, implemented as address subdecoding on bank 2 and 3,
- strobe/address phases generation,
- support for DVB-CI.

The FMI is allocated a 1-Gbyte region of the available memory map which is separated into two spaces. One is configuration space used to control the behavior of the FMI, the other a memory region which is mapped on to four configurable FMI banks. Boot bank is bank 0.

Figure 21: FMI subsystem main architecture



FMI endianness is fixed at system reset to little endian 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 left-most bit, bit 31, is the most significant bit and the right-most bit, bit 0, is the least significant.

16.1 Supported burst flash memory devices

The FMI supports the following flash memory devices in banks 0, 2 and 3:

- AMD AM29BL162C,(bank 2 and 3)
- STMicroelectronics M58LW064A/B,
- Intel 28F800F3/ 28F160F3.

The FMI implements a superset of operational modes so that it is compatible with most of the main functions listed for the three flash families. See Table 39 for a comparison of the supported features.

Table 39: STMicroelectronics, AMD, Intel flash features comparison

	AMD	STMicroelectronics	Intel
Size	16 Mbits	64 Mbits	8/16 Mbits
Maximum[1] operating frequency	40 MHz	60 MHz	60 MHz
Data bus	16 bits fixed	16 bits	16 bits fixed
Main operations	Asynchronous single access write Sync burst read Asynchronous single access read	Asynchronous single access write Sync burst read Asynchronous single access read	Asynchronous single access write Sync burst read Asynchronous single access read
Burst size	Maximum burst supported is 8 words	1-2-4-8 words Set by burst configuration register	4 to 8 words Set by read confide register
Burst style[2]	Linear burst -32 words	Sequential burst	Linear burst
X-latency[3]	70-90-120 ns	7-8-9-10-12[4] cycles	2-3-4-5-6 cycles
Y-latency[5]	1 cycle	1-2 cycles	1 cycle
Burst suspend/ resume	Yes, using burst address advance (BAA) input by MUXing the BAA signal on to FMIADDR25[6]	Yes via burst address advance (B) input	No automatic advance
Ready/busy pin[7]	Yes (RD/BY)	Yes (RD/BY)	No
Ready for burst[8]	No	Yes (R)	Yes (W)

1. The flash operating frequency, clock divide ratios and system frequency should be consistent with the maximum operating frequency.
2. 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 asynchronous write operations. On STMicroelectronics and Intel devices the burst is enabled synchronously using the burst configuration register.
3. 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 STMicroelectronics, it is the time elapsed from the sample valid of starting address to the data being output from memory for Intel and AMD.
4. 10 to 12 only for F = 50 MHz
5. Y-latency is the time elapsed from the current valid data that is output to the next data valid in output during a burst.
6. See Section 16.4.5: *AMD flash support* on page 109.
7. 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.
8. 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.

16.2 Signal and strobe generation

The FMI generates strobe transitions on either edge of the reference clock. The relationship of the strobes is programmed inside the FMI, to allow strobe edges to be controlled with a resolution of half a clock cycle. This is most used for improving timing resolution (for example, extending hold time) for slow asynchronous peripherals (1/2 or 1/3 of the FMI subsystem clock).

At 108 MHz the extra resolution is not generally needed, and for synchronous interfaces, the FMI does not support negative edge transition.

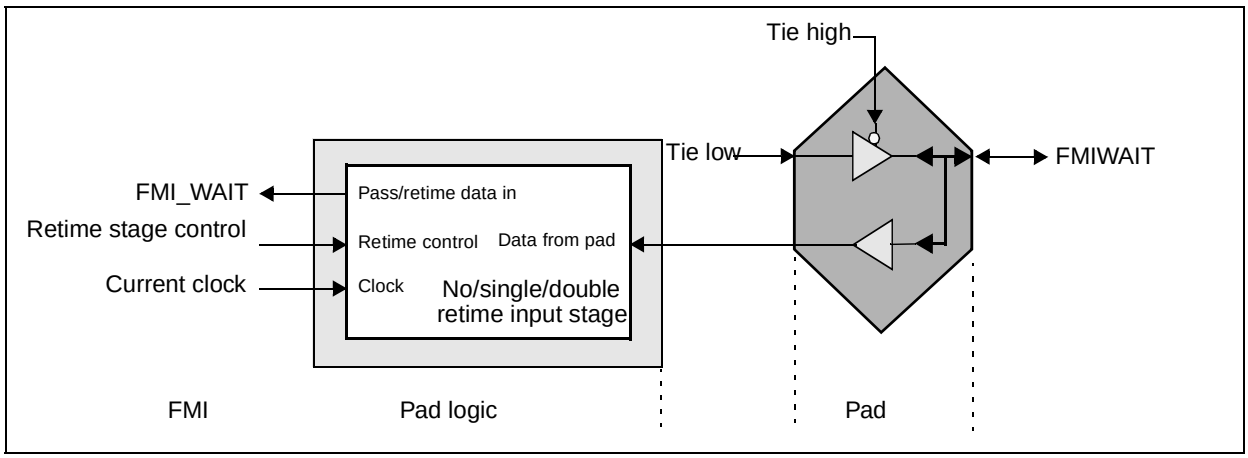
Note: For synchronous modes, the signals are synchronous to the rising edge of the current memory clock being used.

16.2.1 Input strobe generation

FMIWAIT

FMIWAIT may be passed to the generic FMI as a retimed input. The retimed input can be dynamically changed between a single retime input stage or a double retime input stage, selectable by the retime stage control input. Both retime stages are synchronized to the current clock. FMIWAIT uses the internal FMI control signal FMI_WAIT.

Figure 22: Synchronous input strobe pad logic connections



When accessing a device, the number of retime stages is programmable through FMI_GENCFG[3:2] as shown in Table 40.

Table 40: FMIWAIT retime stage control mapping

Number of retime stages	Retime stage control[1:0]
0 (signal passed directly to the FMI)	1x
1 (single retime stage)	01
2 (double retime stage)	00 (default)

The boot indicator (FMI_GENCFG[20]), an FMI register bit, is set by the software to logic 1 when the boot sequence is complete. Any subsequent reboot (hard or soft) forces this bit to be reset to logic 0.

16.2.2 Output strobe generation

External peripherals, such as memory devices and microprocessors, require control signals for their correct operation. These signals are generated internally by the FMI and are then passed to the pads. If necessary they are also inverted.

Figure 23: Output strobe pad connections

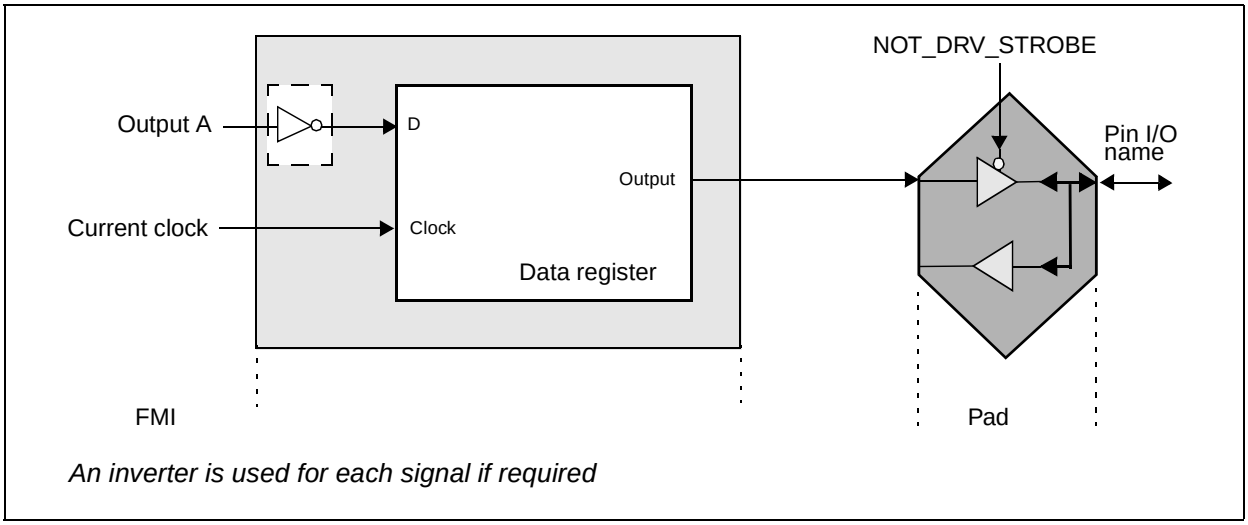


Table 41 details the signal connection for the internal FMI control signals to the external device pins.

Table 41: Control signal-to-pin connections

FMI signal name	Description	Pin I/O name
FMI_BE[0]	Byte enable 0	NOT_FMIBE0
FMI_BE[1]	Byte enable 1	NOT_FMIBE1
FMI_OE	Output enable	NOT_FMIOE
FMI_LBA	Flash device load burst address	NOT_FMILBA
FMI_RDNOTWR	Read not write	FMIRDNOTWR

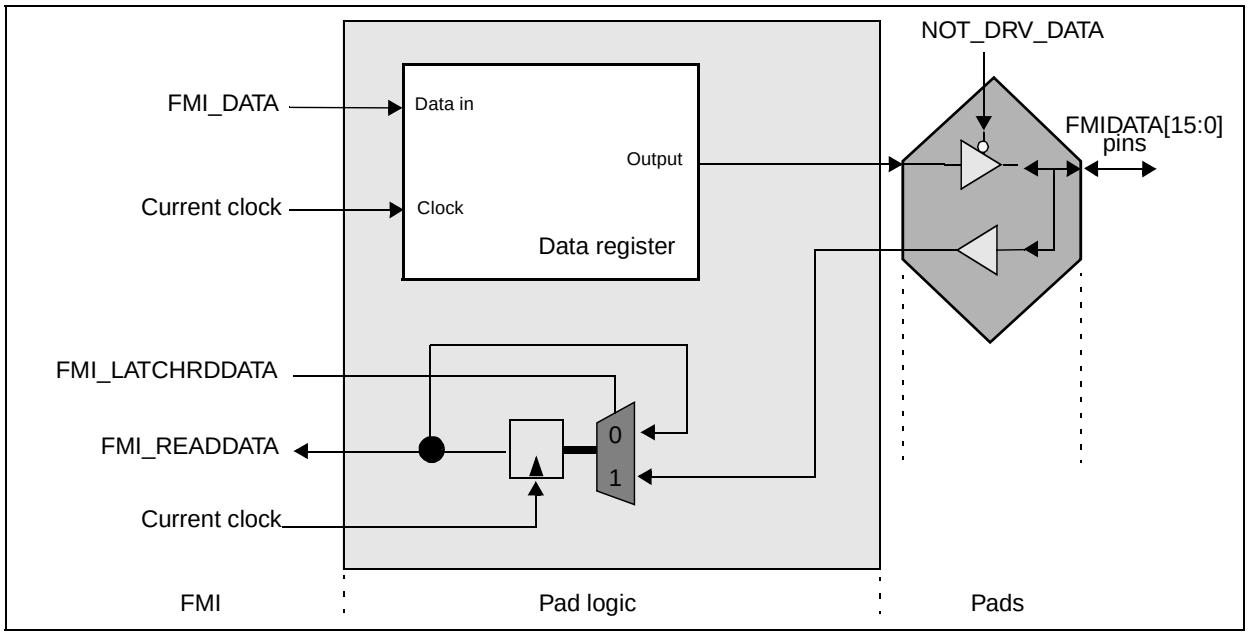
The FMI provides four chip selects for external devices dedicated to banks 0 to 3. These correspond to the pads as detailed in Table 42.

Table 42: Chip select signal-to-pin connections

FMI signal name	Description	Pin I/O name
FMI_CSA	Chip select on bank 0	NOT_FMICSA
FMI_CSB	Chip select on bank 1	NOT_FMICSB
FMI_CSC	Chip select on bank 2	NOT_FMICSC
FMI_CSD	Chip select on bank 3	NOT_FMICSD

16.2.3 Data bus control

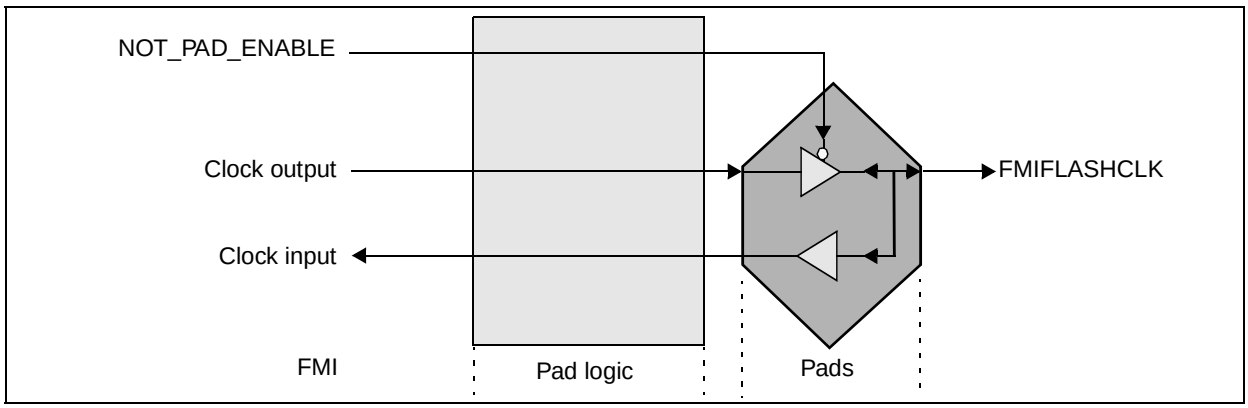
Figure 24: Data bus and output control



16.2.4 External clock control

FMI pad logic passes the necessary FMI clock to the external devices. The clock operates as output only.

Figure 25: External clock



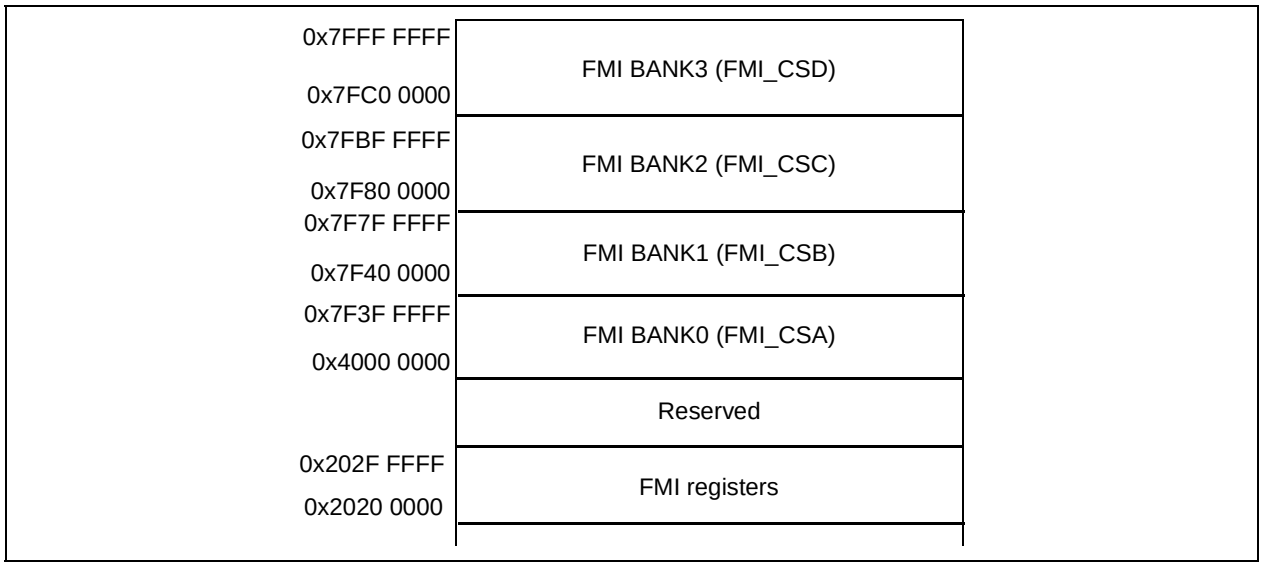
16.3 Bank position and size

Region 3 of memory is organized into four banks of different size. Each bank is enabled and its size is set using the FMI bank sizing registers. The position and size of each bank is specified by programming the top address of each bank into the registers. This is done for address bits 29 to 22 giving a minimum bank size of 4 Mbytes.

Note: The boot memory is always located at the top of bank 0 as the C106 expects to boot from the bottom of region 0.

On the STx5119 the FMI can only boot from a 16-bit device.

Figure 26: FMI subsystem memory map



After reset the memory space area is subdivided in four regions as shown in Table 43.

Table 43: ST20 FMI bank sizing register reset value

Register name	Reset value
BANK_0_TOP_ADDRESS	Top address (bits 29 to 22) = 1111 1100 (0xFC)
BANK_1_TOP_ADDRESS	Top address (bits 29 to 22) = 1111 1101 (0xFD)
BANK_2_TOP_ADDRESS	Top address (bits 29 to 22) = 1111 1110 (0xFE)
BANK_3_TOP_ADDRESS	Top address (bits 29 to 22) = 1111 1111 (0xFF)
BANKS_ENABLED	100 (bank 0 only)

16.4 Bank options

The FMI can interface to a wide variety of SRAM, ROM, Flash, SFlash™ and other peripheral devices. Each of the FMI banks can be set up separately to support different types of devices. There are restrictions on certain banks as can be seen in Table 44.

Table 44: Allowable bank options

	Bank 0	Bank 1	Bank 2	Bank 3
AMD flash	N	N	Y	Y
Intel/ST Flash	Y	N	Y	Y
Sub-bank decoding	N	N	Y	Y
DVB-CI	N	Y	N	N

A typical configuration is shown in Table 45.

Table 45: Typical FMI configuration

FMI bank	Capability	FMI address	Size (Mbytes)
3	Asynchronous peripheral	0x7FC0 0000	8
2	Asynchronous peripheral	0x7F00 0000	8
1	DVB-CI	0x7F40 0000	496
0	Flash	0x4000 0000	512

The configuration is held in memory-mapped registers within the FMI. Three types of configuration registers are present: global, bank control (FMI_CONFIGn) and bank options (FMI_GENCFG and FMI_CONFIG_CTRLn).

Each FMI_BANK[n] (n = [0:3]) contains a set of four 32-bit registers that are used to control each bank depending on the type of device that is connected.

Table 46 describes how the configuration region of each bank is divided.

Table 46: FMI bank[n] control register formats

Register name	Description
FMI_CONFIGDATA0	General parameters, see page 118
FMI_CONFIGDATA1	Read parameters, see page 119
FMI_CONFIGDATA2	Write parameters, see page 120
FMI_CONFIGDATA3	General parameters, see page 121
Reserved	

The type and organization of each set of bank registers is dependent on the value in DEVICETYPE (FMI_CONFIGDATA0) which defines the type of memory or device attached to that bank.

The interface can be configured either for an asynchronous nonmultiplexed address and data bus (SRAM/peripheral/flash interface) or for SFlash™. To configure an asynchronous nonmultiplexed device use FMI_CONFIGDATA0, FMI_CONFIGDATA1 and FMI_CONFIGDATA2. To configure for SFlash™, set bit 26 (WE_USE_OE_CONFIG) in FMI_CONFIGDATA0 to and set FMI_CONFIGDATA3.

Both memory types and their associated control registers are described in Chapter 17: *Flash and peripheral memory interface (FMI) registers* on page 114.

16.4.1 Default configuration

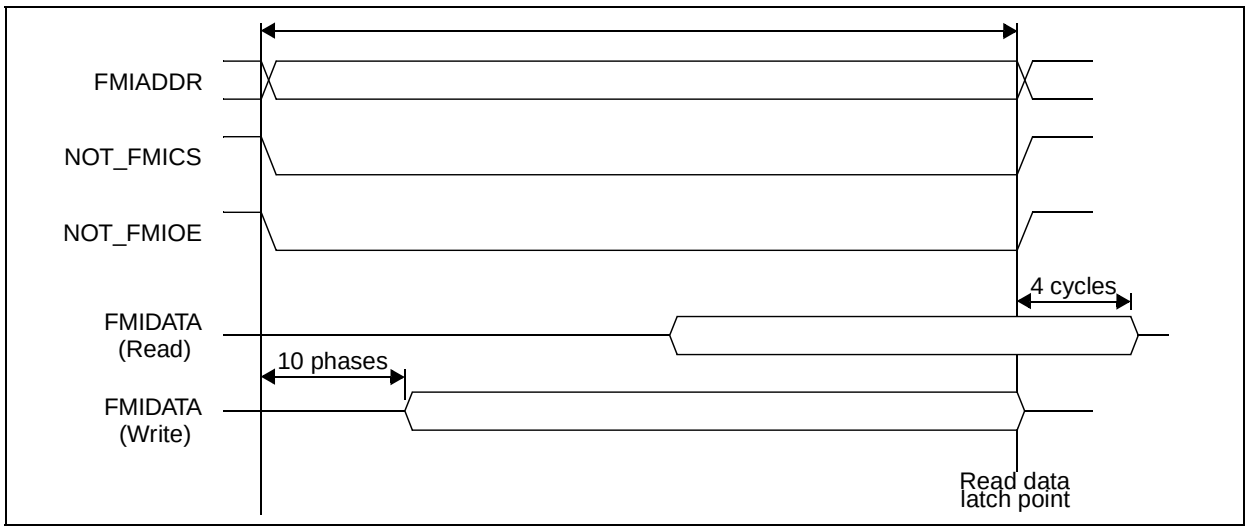
Following reset, a default setting (in registers FMI_CONFIGDATA0 and FMI_CONFIGDATA1) is loaded into all four banks. This allows the FMI to access data from a slow asynchronous ROM memory.

Table 47: Default configuration after an asynchronous boot

Parameter	Bits	Default value
FMI_CONFIGDATA0		
WAITPOLARITY	[25]	Active high
LATCHPOINT	[24:20]	End of access cycle
DATADRIVEDELAY	[19:15]	10 phases
BUSRELEASETIME	[14:11]	4 cycles
CSACTIVE	[10:9]	Active during read-only
OEACTIVE	[8:7]	Active during read-only
BEACTIVE	[6:5]	Inactive
PORTSIZE	[4:3]	Value of the signal FMI_PRTSZ_INIT (8- or 16-bit)
DEVICETYPE	[2:0]	Peripheral
FMI_CONFIGDATA1		
ACCESSTIMEREAD	[30:24]	(18 + 2 = 20 cycles)
CSE1TIMEREAD	[23:20]	0 phases
CSE2TIMEREAD	[19:16]	0 phases
OEE1TIMEREAD	[15:12]	0 phases
OEE2TIMEREAD	[11:8]	0 phases
CYCLENOTPHASE	[31]	Phase
BEE1TIMEREAD	[7:4]	3 phases
BEE2TIMEREAD	[3:0]	3 phases

Table 47 gives the default state after an asynchronous boot. The remaining configuration parameters are not relevant for an asynchronous boot.

Figure 27: Default asynchronous configuration



As peripheral interfaces are used immediately after reset to boot the device 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 FMI 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 FMI's configuration is inconsistent with the memory device's.

16.4.2 Clock reconfiguration for synchronous devices

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, set up the configuration state associated with that bank and program the required clock ratio associated with the memory type in the register FMI_FLASHCLKSEL.

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

16.4.3 Sub-bank decoding on bank 3

To support 2-Mbyte contiguous memory address space in the boot bank, bank 3 supports subdecoding. The bank 3 subdecoding is controlled by the logic block referred to as the CS control logic in Figure 28.

Figure 28: Bank 3 subdecoding chip select connections

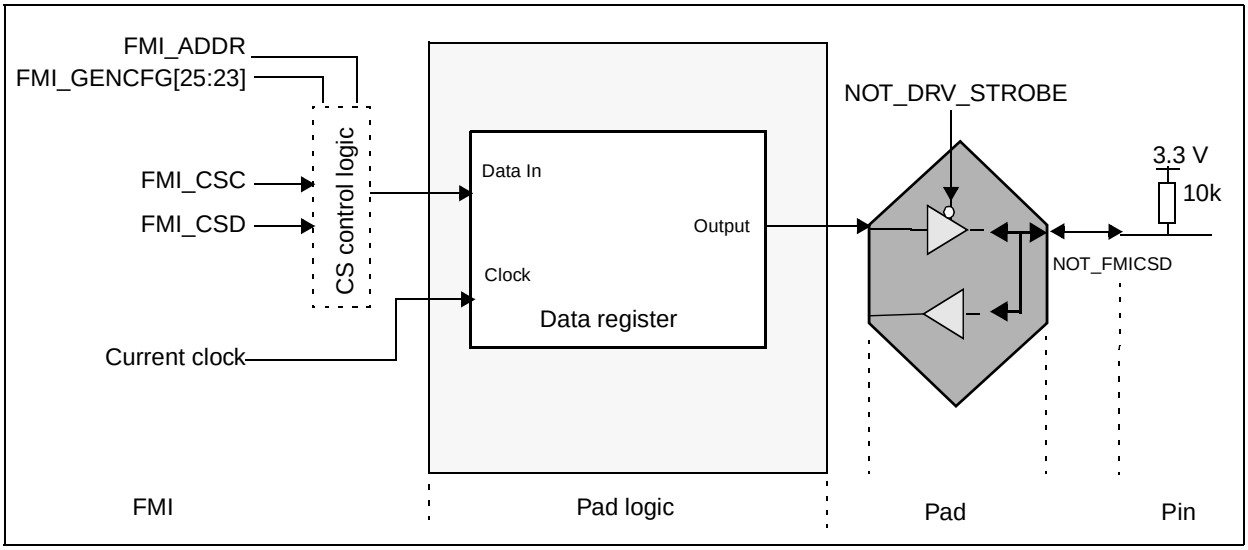


Table 48: Bank 3 sub-bank size control logic

Sub-bank size control (FMI_GENCFG[25:23])	Sub-bank size	Sub-bank selection control
000	4 Mbit (512 Kbyte)	Unshifted address[19]
001	8 Mbit (1 Mbyte)	Unshifted address[20]
010	16 Mbit (2 Mbyte)	Unshifted address[21]
011	32 Mbit (4 Mbyte)	Unshifted address[22]
100	64 Mbit (8 Mbyte)	Unshifted address[23]
101	1 Gbit (128 Mbyte)	Unshifted address[27]
Others	1 Gbit (128 Mbyte)	Unshifted address[27]

Table 49: Bank 3 control logic parameters

Bank 3 subdecoding control (FMI_GENCFG[22])	Sub-bank selection control	Outputs	Function	Inverter needed	Pin I/O name
0: Disabled	-	FMI_CSC	Bank 2 chip select	Yes	NOT_FMICSC
1: Enabled	0: Lower bank	FMI_CSD	Bank 3 lower subdecoded chip select		
	1: Upper bank	Inactive			
0: Disabled	-	FMI_CSD	Bank 3 chip select	Yes	NOT_FMICSD
1: Enabled	0: Lower bank	Inactive	Bank 3 upper subdecoded chip select		
	1: Upper bank	FMI_CSD			

16.4.4 DVB-CI mode

Support for DVB-CI mode is documented in a separate application note.

16.4.5 AMD flash support

This allows AMD burst flash BAA signal support on banks 2 and 3 through MUXing with FMIADDR[25]. Bank 2 is set using FMI_GENCFG[27] and bank 3 using FMI_GENCFG[26].

16.5 Generic access cycle

Figure 29 shows a generic access cycle and the allowable values for each timing field are shown.

Figure 29: Generic access cycle

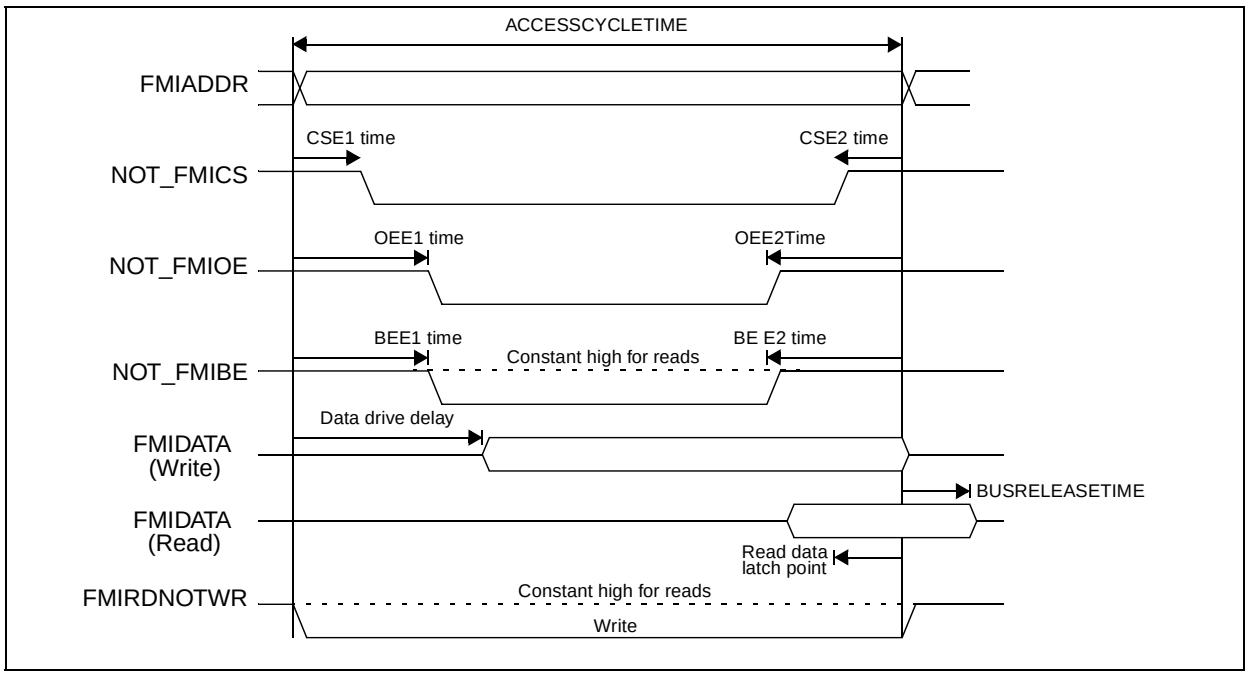


Table 50: Strobe timing parameters for peripherals

Name	Register	Bit	Programmable value
ACCESSTIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[30:24] [30:24]	2 cycles + 0 to 125 cycles
BUSRELEASETIME	FMI_CONFIGDATA0	[14:11]	0 to 15 cycles
DATADRIVEDELAY	FMI_CONFIGDATA0	[19:15]	0 to 31 phases after start of access cycle
CSE1TIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[23:20] [23:20]	Falling edge of CS. 0 to 15 phases or cycles after start of access cycle
CSE2TIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[19:16] [19:16]	Rising edge of CS. 0 to 15 phases or cycles before end of access cycle
OEE1TIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[15:12] [15:12]	Falling edge of OE. 0 to 15 phases or cycles after start of access cycle
OEE2TIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[11:8] [11:8]	Rising edge of OE. 0 to 15 phases or cycles before end of access cycle.
BEE1TIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[7:4] [7:4]	Falling edge of BE. 0 to 15 phases or cycles after start of access cycle

Table 50: Strobe timing parameters for peripherals

Name	Register	Bit	Programmable value
BEE2TIME	FMI_CONFIGDATA1 FMI_CONFIGDATA2	[3:0] [3:0]	Rising edge of BE. 0 to 15 phases or cycles before end of access cycle
LATCHPOINT	FMI_CONFIGDATA0	[24:20]	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, write, neither or both.

Table 51: 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

16.6 FMI flash interface

16.6.1 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 the FMI_FLASHCLKSEL register.

Burst mode flash accesses consist of multiple read accesses which must be made in a sequential order. The FMI maps system memory operations on to one or more burst flash accesses depending on the burst size configuration, operation size and the starting address of the memory access.

Note: 1 Continuous burst is not supported by the FMI.

2 32-word burst size is partially supported by the FMI. The burst is interrupted when the required data has been read.

3 Asynchronous page mode read is not supported by the FMI.

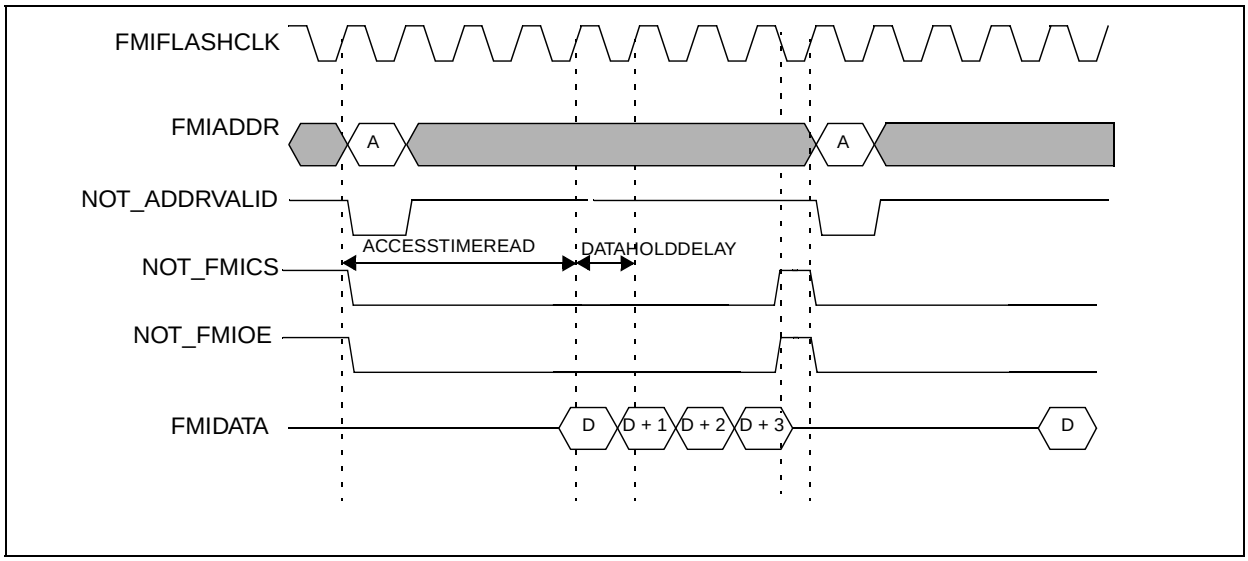
4 Interleaved burst mode is not supported by FMI because of the implementation of multiple reads only using synchronous burst mode (feature provided by all the three families of flash chips adopted).

FMI supports a asynchronous single write.

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

In Figure 30 a typical burst access with burst length of four words is shown.

Figure 30: 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.

16.6.2 Burst interrupt and burst reiteration

The FMI 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 the three families of flash devices (burst read interrupt for an STMicroelectronics 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 (FMI_LBA) 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 FMI on the STBus interface, there are three possible outcomes.

- **n = m**

  The FMI performs one burst access during which it gets the exact number of words as requested (see example A In Figure 31 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.***n > m***

  If the starting address is aligned on an *m*-byte word boundary, the FMI 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.

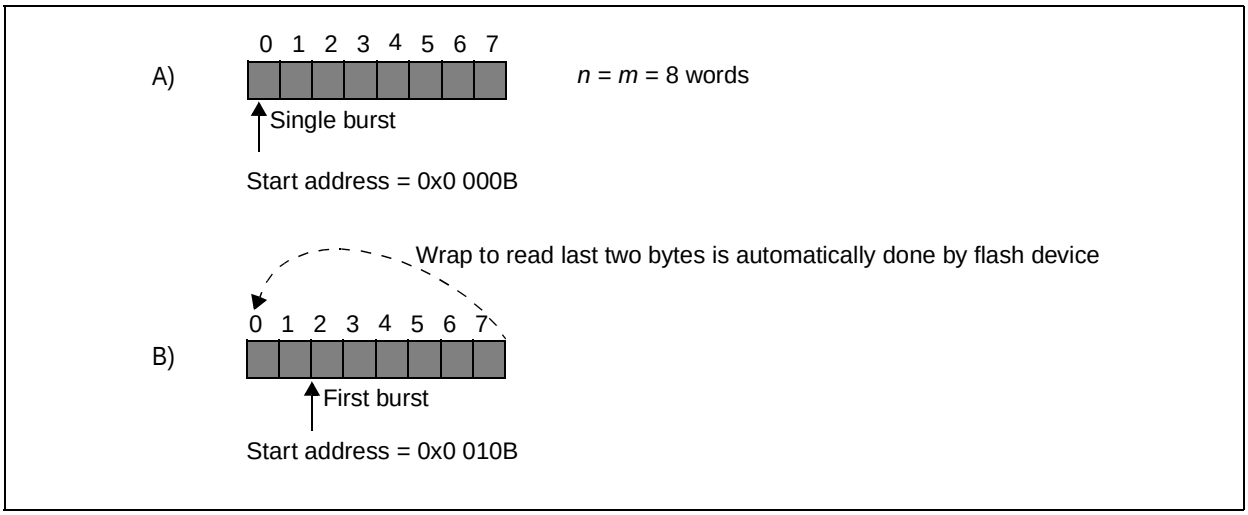
- ***n < m***

  The FMI 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.

Figure 31: Burst on a flash with a single access



16.6.3 Synchronous burst enable

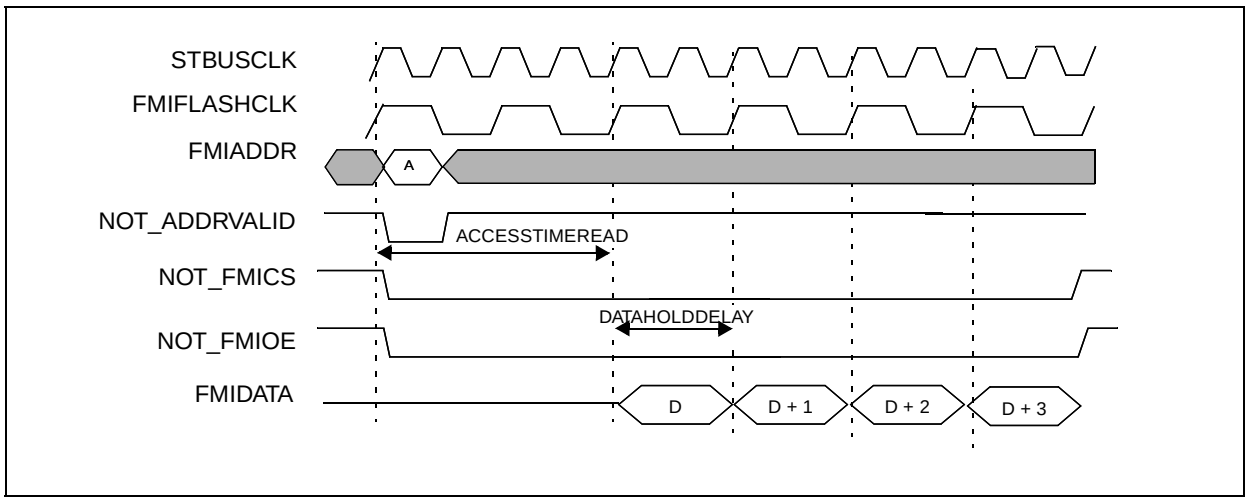
This operation is controlled by software and must only be performed when all other configuration registers in the FMI have been programmed.

Table 39: *STMicroelectronics, AMD, Intel flash features comparison* on page 100, shows that for STMicroelectronics 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.

16.6.4 Lower clock rates

Many SFlash™ devices operate in the 30 to 50 MHz clock range (Table 39: *STMicroelectronics, AMD, Intel flash features comparison* on page 100) whereas the FMI operates up to a clock frequency of 108 MHz. To deal with this difference, the FMI runs in a lower speed mode. The hardware in the FMI 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 FMI timings, to set up and latch, on the appropriate edge of this slower clock.

Figure 32: Half speed FMI SFlash™ clock



16.6.5 Pull up resistors

Pins for FMI ADDR22 and FMI ADDR23 should be pulled up with an external resistor for correct operation of the device.

17 Flash and peripheral memory interface (FMI) registers

Addresses are provided as either the *FMIBaseAddress* or the *FMIBufferBaseAddress* + offset. The *FMIBaseAddress* is: 0x2020 0000. The *FMIBufferBaseAddress* is: 0x202F F800.
All FMI buffer registers are nonvolatile.

Table 52: Flash and peripheral memory interface (FMI) registers

Register name	Description	Address offset	Type
FMI bank sizing[1]			
BANK_0_TOP_ADDRESS	Top address (29 to 22) of bank0, see page 115	0x0020	R/W
BANK_1_TOP_ADDRESS	Top address (29 to 22) of bank1, see page 115	0x0030	R/W
BANK_2_TOP_ADDRESS	Top address (29 to 22) of bank2, see page 115	0x0040	R/W
BANK_3_TOP_ADDRESS	Top address (29 to 22) of bank3, see page 115	0x0050	R/W
BANKS_ENABLED	Number of enable banks, see page 116	0x0060	R/W
FMI global settings[2]			
FMI_STATUSCFG	Status register (flags bank configuration update), see page 116	0x0010	RO
FMI_STATUSLOCK	Status register (flags bank configuration lock), see page 117	0x0018	RO
FMI_LOCK	Lock register, see page 117	0x0020	R/W
FMI_FLASHCLKSEL	Select clock speed for all SFlash, see page 117	0x0050	WO
FMI_CLKENABLE	Enable clock generation for different device	0x0068	WO
FMI bank control[2]			
FMI_CONFIGDATA0	FMI general parameters, see page 118	0x0100 (Bank0) 0x0140 (Bank1) 0x0180 (Bank2) 0x01C0 (Bank3)	R/W
FMI_CONFIGDATA1	FMI read parameters, see page 119	0x0108 (Bank0) 0x0148 (Bank1) 0x0188 (Bank2) 0x01C8 (Bank3)	R/W
FMI_CONFIGDATA2	FMI write parameters, see page 120	0x0110 (Bank0) 0x0150 (Bank1) 0x0190 (Bank2) 0x01D0 (Bank3)	R/W
FMI_CONFIGDATA3	FMI general parameters, see page 121	0x0118 (Bank0) 0x0158 (Bank1) 0x0198 (Bank2) 0x01D8 (Bank3)	R/W
FMI bank options[2]			
FMI_GENCFG	FMI general configuration, see page 122	0x0028	R/W

1. Uses *FMIBufferBaseAddress*
2. Uses *FMIBaseAddress*

17.1 FMI bank sizing

BANK_0_TOP_ADDRESS Memory bank 0 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																								BANK_0_TOP_ADDRESS							

Address: *FMIBufferBaseAddress* + 0x0020

Type: Read/write

Reset: 0x00FC

Description: Contains bits 29 to 22 of the top address of memory bank 0.
Accesses to this address space cause transfer on FMI bank0 (4000 0000 to 0x7F3F FFFF).

BANK_1_TOP_ADDRESS 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																								BANK_1_TOP_ADDRESS							

Address: *FMIBufferBaseAddress* + 0x0030

Type: Read/write

Reset: 0x00FD

Description: Contains bits 29 to 22 of the top address of memory bank 1.
Accesses to this address space cause transfer on FMI bank 1(0x7F40 0000 to 0x7F7F FFFF).

BANK_2_TOP_ADDRESS 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																								BANK_2_TOP_ADDRESS							

Address: *FMIBufferBaseAddress* + 0x0040

Type: Read/write

Reset: 0x00FE

Description: Contains bits 29 to 22 of the top address of memory bank 2.
Accesses to this address space cause transfer on FMI bank 2 (0x7F80 0000 to 0x7FBF FFFF).

BANK_3_TOP_ADDRESS 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																								BANK_3_TOP_ADDRESS							

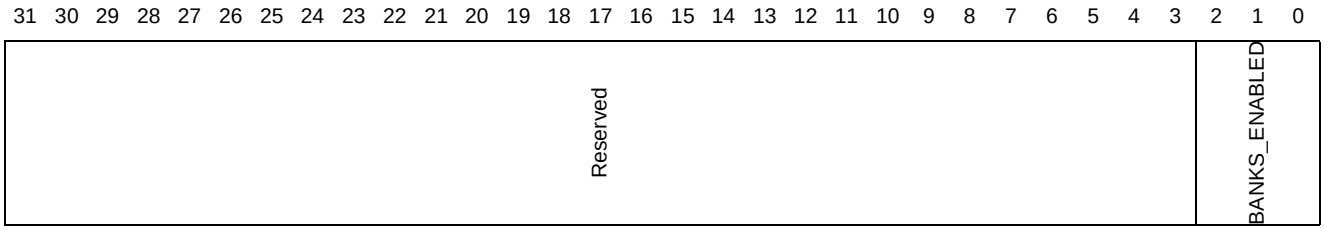
Address: *FMIBufferBaseAddress* + 0x0050

Type: Read/write

Reset: 0x00FF

Description: Contains bits 29 to 22 of the top address of memory bank 3.
Accesses to this address space cause transfer on FMI bank 3 (0x7FC0 0000 to 0x7FFF FFFF).

BANKS_ENABLED Enabled bank register



Address: *FMIBufferBaseAddress* + 0x0060

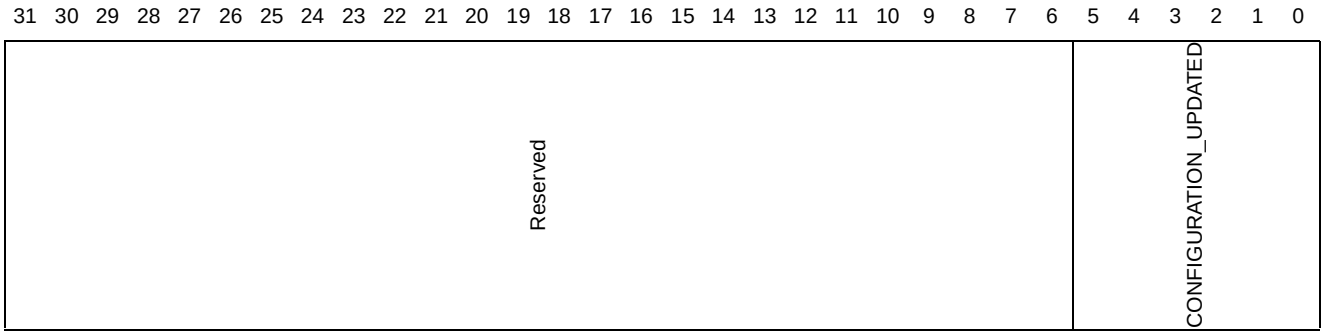
Type: Read/write

Reset: 0x06

Description: Contains the total number of bank registers enabled.
At reset all the banks are enabled.
1 (001) = Bank 3 enabled
2 (010) = Banks 3 and 2 enabled
3 (011) = Banks 3 down to 1 enabled
4 (100) = All banks enabled
When the number of banks is reduced by the BANKS_ENABLED register, the last bank (that is the bottom bank) takes its own area plus the remaining area of the banks disabled. For example if only three banks are enabled, BANK 0 is disabled, then the BANK 1 region contains its own area plus the BANK 0 area.

17.1.1 FMI global setting

FMI_STATUSCFG FMI status configuration register



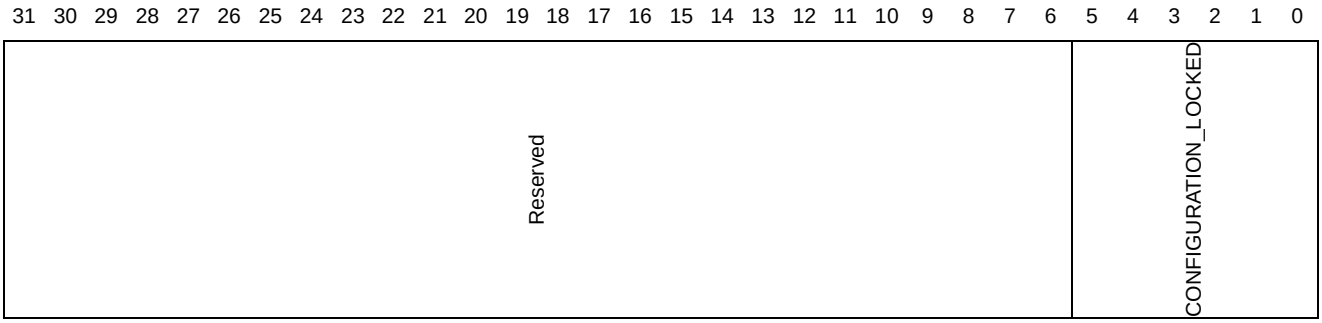
Address: *FMIBaseAddress* + 0x0010

Type: Read only

Reset: Undefined

Description: If bit [n] is set, then all configuration registers associated with bank [n] have been written to at least once.

FMI_STATUSLOCK FMI status configuration lock register



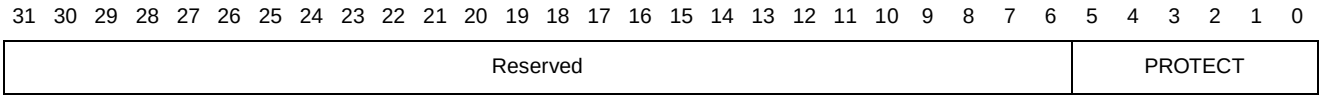
Address: *FMIBaseAddress* + 0x0018

Type: Read only

Reset: Undefined

Description: If bit [n] is set, then all configuration registers associated with bank [n] are locked and further write accesses is ignored.

FMI_LOCK FMI lock register



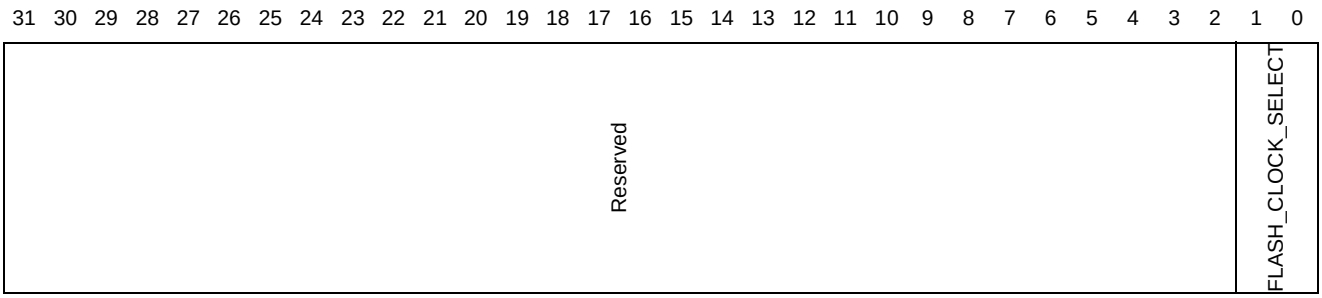
Address: *FMIBaseAddress* + 0x0020

Type: Read/write

Reset: Undefined

Description: If bit [n] is set, then the registers FMI_BANK[n].FMI_CONFIGDATA[0:3] may only be read. Subsequent writes to these registers are ignored.

FMI_FLASHCLKSEL FMI flash burst clock select register



Address: *FMIBaseAddress* + 0x0050

Type: Write only

Reset: Undefined

Description:

[31:2] **Reserved**

[1:0] **FLASH_CLOCK_SELECT:** Set clock ratio for burst flash clock

00: 1:1 flash operates at STBus clock
01: 1:2 flash operates at 1/2 of STBus clock
10: 1:3 flash operates at 1/3 of STBus clock
11: Reserved

17.2 FMI bank control

Any bit in the control registers which is defined as reserved should be set to 0.

FMI_CONFIGDATA0 FMI general parameters

31–27	26	25	24–23	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_CONFIG	WAITPOLARITY	LATCHPOINT	DATADRIVEDELAY	BUSRELEASETIME	CSACTIVE	OEACTIVE	BEACTIVE	PORTSIZE	DEVICETYPE					

Address: *FMIBaseAddress* + 0x100 (Bank 0), 0x140 (Bank 1), 0x180 (Bank 2), 0x1C0 (Bank 3)

Type: Read/write

Reset: 0

Description:

[31:27] **Reserved**

[26] **WE_USE_OE_CONFIG**

This bit must be set to 1 for the SFlash™ bank (like STM58LW064A/B). It requires a configurable READNOTWRITE signal for asynchronous 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 READNOTWRITE 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**

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

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

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

[10:5] **CSACTIVE, OEACTIVE, BEACTIVE:** See Table 51: *Active code settings* on page 110

[4:3] **PORTSIZE**

00: Reserved
01: Reserved
10: 16-bit
11: 8-bit

[2:0] **DEVICETYPE**

001: Normal peripheral
100: Burst flash
Other: Reserved

FMI_CONFIGDATA1 FMI read 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
CYCLENOTPHASEREAD	ACCESSTIMEREAD	CSE1TIMEREAD	CSE2TIMEREAD	OEE1TIMEREAD	OEE2TIMEREAD	BEE1TIMEREAD	BEE2TIMEREAD

Address: *FMIBaseAddress* + 0x108 (Bank 0), 0x148 (Bank 1), 0x188 (Bank 2), 0x1C8 (Bank 3)

Type: Read/write

Reset: 0

Description:

[31] **CYCLENOTPHASEREAD**
Change measure unit for e1/e2 time accesses from phases to cycles
0: The e1(e2) timewrite for CS, BE, OE are expressed in STBus clock phases
1: They are expressed in cycles

[30:24] **ACCESSTIMEREAD**
2 to 127 STBus 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

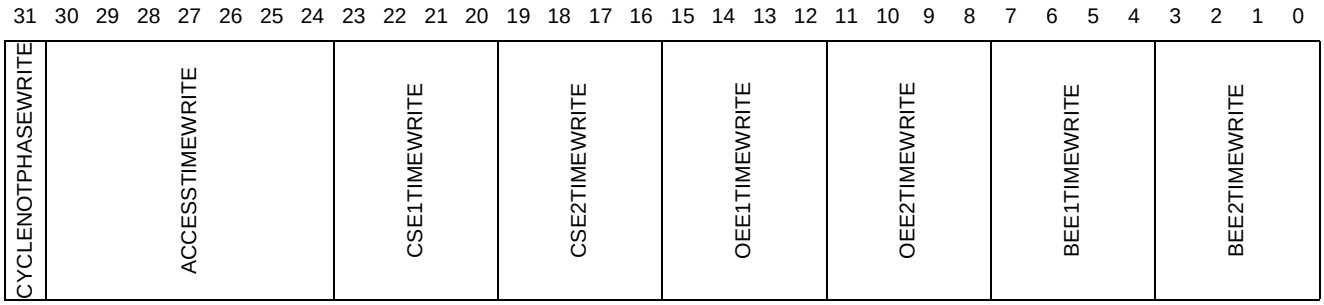
[15:12] **OEE1TIMEREAD**
Falling edge of OE: 0 to 15 phases/cycles after start of access cycle

[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

FMI_CONFIGDATA2 FMI write parameters



Address: *FMIBaseAddress* + 0x110 (Bank 0), 0x150 (Bank 1), 0x190 (Bank 2), 0x1D0 (Bank 3)

Type: Read/write

Reset: 0

Description:

[31] **CYCLENOTPHASEWRITE**
Change measure unit for e1/e2 time accesses from phases to cycles:
0: The e1(e2) timewrite for CS, BE, OE are expressed in STBus 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
The value is used for falling edge of WE as well if the bit WE_USE_OE_CONFIG (FMI_CONFIGDATA0, bit 26) is set to one.

[11:8] **OEE2TIMEWRITE**
(WEE2TIMEWRITE)
Rising edge of OE. 0 to 15 phases/cycles before end of access cycle
The value is used for rising edge of OE as well if the bit WE_USE_OE_CONFIG (FMI_CONFIGDATA0, bit 26) is set to one.

[7:4] **BEE1TIMEWRITE**
Falling edge of BE. 0 to 15 phases/cycles after start of access cycle

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

FMI_CONFIGDATA3 FMI general parameters

31–27	26	25–10	9–8	7–6	5–2	1	0
Reserved	STROBEONFALLING	Reserved	BURST_SIZE	DATALATENCY		DATAHOLDDELAY	BURSTMODE

Address: *FMIBaseAddress* + 0x118 (Bank 0), 0x158 (Bank 1), 0x198 (Bank 2), 0x1D8 (Bank 3)

Type: Read/write

Reset: 0

Description:

[31:27] **Reserved**

[26] **STROBEONFALLING**[1]

0: Strobes, data and address for burst generated on rising edge of the flash clock

1: Strobes, data and address for burst generated on falling edge of the flash clock

[25:10] **Reserved**

[9:7] **BURST_SIZE**

The number of bytes which map on to the device's burst mode.

000: 2 001: 4

010: 8 011: 16

100: 32 101: 64[2]

110: 128 111: Reserved

Only valid in burst mode

[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

1. Configuration of FMI_CONFIGDATA0, FMI_CONFIGDATA1, FMI_CONFIGDATA2 relates only to the asynchronous behavior (normal peripheral and normal asynchronous behavior of flash). These registers must be programmed in terms of STBus clock cycle. FMI_CONFIGDATA3 must be configured only if there is burst flash and refers to the synchronous behavior of flash. The parameters in this register must be programmed in terms of flash clock cycles.
2. 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 maximum transfer is 32 bytes on FMI, so in these cases the burst on flash is always interrupted.

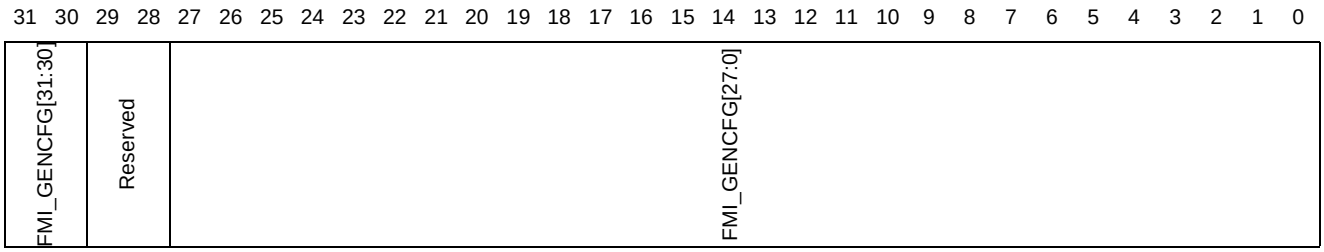
Note: Any bit in the configuration register, which is defined as reserved, should be set to 0.

FMI_CONFIGDATA3 does not need to be configured for a normal asynchronous peripheral.

The strobe on falling feature of FMI 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 FMI assumes memory always samples on the rising edge anyway. The strobes on falling feature has been implemented only to possibly extend the hold time of half a cycle to help pad logic implementation.

17.3 FMI bank options

FMI_GENCFG General purpose configuration output



Address: *FMIBaseAddress* + 0x0028

Type: Read/write

Reset: 0

Description:

[31:30] DVB-CI enable

[29] Reserved

[28] Reserved

[27] AMD flash for bank 3

[26] AMD flash for bank 2

[25:23] Sub-bank 3 size

[22] Sub-bank 3 enable

[21:0] Reserved

[3:2] NOT_FMIBUSREQ retime options

[1:0] FMIWAIT retime options

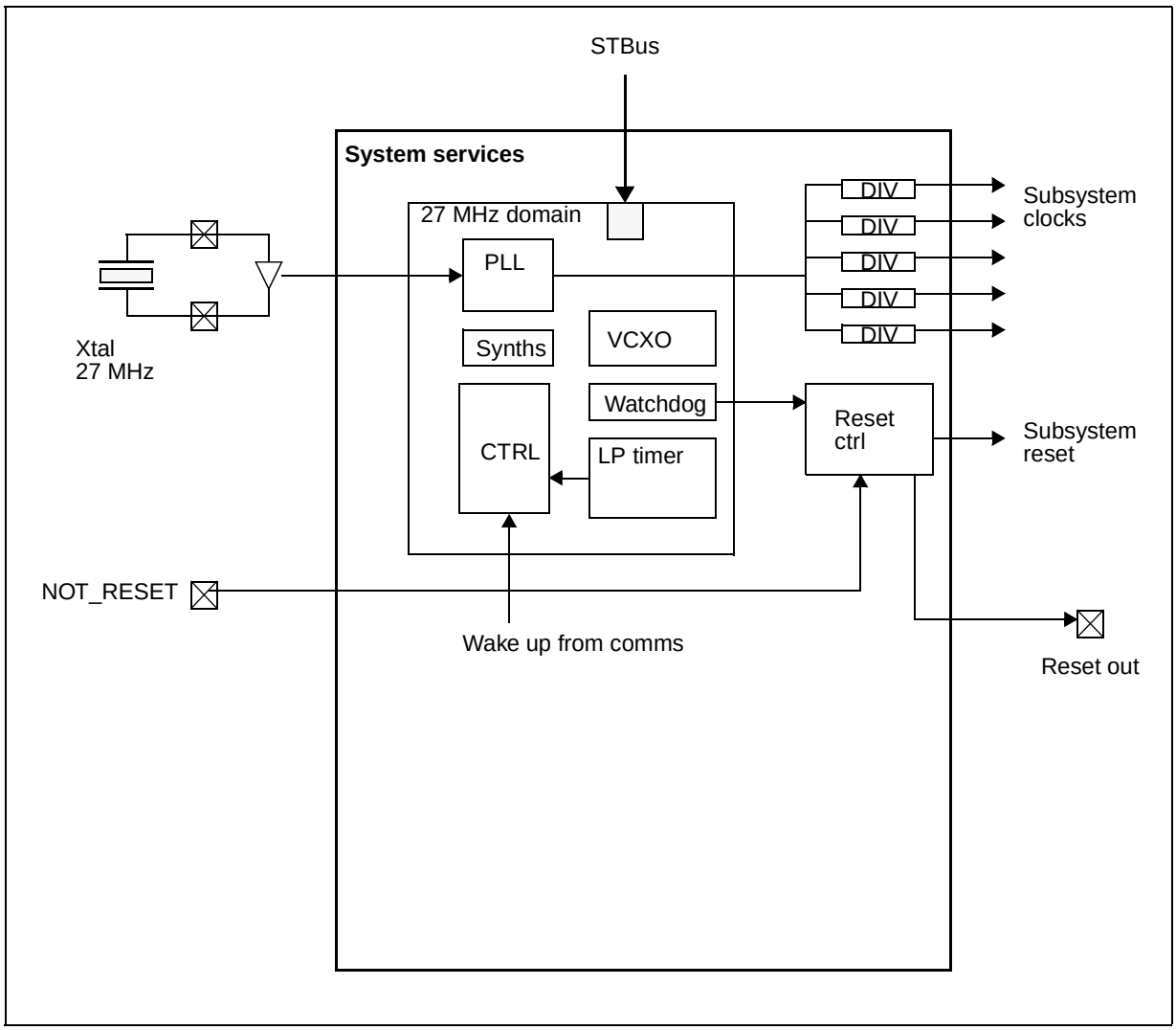
18 System services

18.1 Brief overview

The system services block groups together a number of functions:

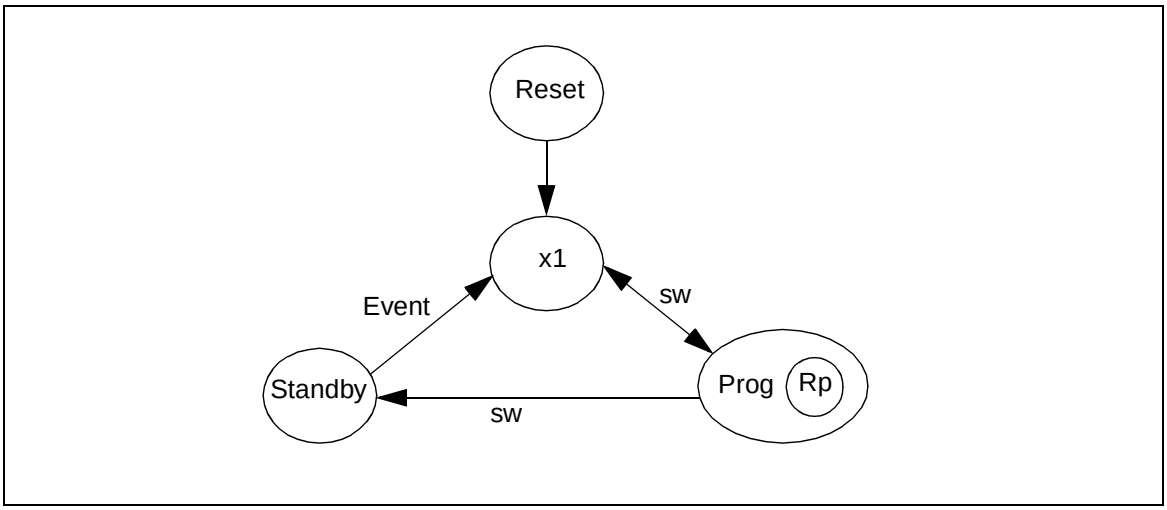
- clock generation and distribution,
- low speed reduced power monitoring and control,
- real-time counter,
- integrated clock recovery using a digitally-controlled oscillator (DCO),
- smartcard power control,
- reset control,
- CPU programmable interrupt.

Figure 33: STx5119 system services architecture



18.2 States of operation

Figure 34: States of operation diagram



There is one mode of operation which is clock master. Within this mode there are three states of operation:

- x1,
- programmable (reduced power mode exists within the programmable mode of operation),
- standby.

Following reset the system services block transitions into the x1 state causing all clocks to be sourced from 27 MHz. All others transitions into the x1 state after reset are through software by programming the MODE_CONTROL register or event-based from standby mode.

To program the divider channels the software must cause a transition to the x1 state, the subsystem clocks are shut down (zeroed) in a clean way and then re-started synchronously using the external 27 MHz clock. The divider channels can then be programmed. When programming is complete a further software transition is made into the programmable state, this transition causes a second clean switch to the newly programmed clock divider outputs.

It is possible to transition from programmable mode to standby mode with software. The return path transition to x1 mode only occurs if the low power alarm is activated. Also if there is an external interrupt (wake up) from the comms subsystem.

The reason standby transitions to x1 mode is because the PLL could be disabled within standby. Therefore no PLL clocks are running.

18.2.1 Programming the registers

After reset all registers are locked and cannot be written to. A write can only occur when the lock register is unlocked. The registers can be unlocked as in the procedure below.

1. Unlock the registers by writing the keyword followed by the bit wise inverse.
2. Configure the registers as required.
3. Lock the registers by writing a 1 to the KEY_LOCK bit (bit 8 of REGISTER_LOCK_CFG).

18.3 Clock generation and distribution

STx5119 clock generation provides the following features:

- multiple subsystem clocks allowing decoupling of the CPU from the interconnect for maximum performance,
- dynamic clock change for performance tuning or reduced power,
- FMI master-only clock generation and deskew,
- LMI master-only clock generation deskew is performed in the padlogic,
- digital VCXO clock recovery support for MPEG.

STMicroelectronics provides a basic initialization procedure for clock generation set up, which is available for standard modes.

For flexibility the clocking scheme is based on two PLLs and two quadruple synthesizers.

18.3.1 Subsystem clocks

The subsystem clocks are generated using the frequency synthesizer PLLs as shown in Table 53. The resulting ranges are used to divide the blocks among a number of clock domains.

Table 53: Subsystem clock frequencies

Subsystem	Name	Clock frequency (MHz)			Source
		Min	Max	Default	
CPU	CPU_CLK	200	243	200	PLL1
Interconnect, PTI, Comms, TV out	SYS_CLK	90	110	100	
Blitter	BLIT_CLK	100	160	100	
FMI	FLASH_CLK	50	110	100	
Audio decode, LCMPEG	AV_CLK	16	50	50	
FDMA	FDMA_SLIM	150	250	150	
LMI	LMI_CLK from LMI	133	166	133	PLL2
LMI padlogic	4 x LMI_CLK	532	620	532	
VDAC, TV out	PIX_CLK	27	60	27	QFS1
Comms	DSS_CLK	1	27	13.5	
Auxiliary clock	AUX_CLK	32 kHz	-	27	
ADAC, PCM player	PCM_CLK	8	60	12.3	QFS2
S/PDIF player	SPDIF_CLK	8	30	12.3	

18.3.2 PLL clocks

The STx5119 uses two PLLs in a frequency synthesizer configuration, primarily driven from an external 27 MHz crystal.

The PLLs provide system clocks up to 620 MHz. These two clocks are individually selectable by the programmable dividers that source the clocks shown in Table 53, resulting in greater system flexibility.

PLL frequency calculation

The PLLs have three dividers which are referred to as M (pre-divider), N (feedback-divider) and P (post-divider). These dividers are located in the PLLx_CONFIGn registers. The output clock of the frequency of the PLL is controlled by binary values applied to the programmable registers as defined by the following formula:

$$F_{(clockout)} = \frac{2 \times N}{M \times 2^P} \times F_{(refclock)}$$

where the values of M, N and P must satisfy the following constraints:

$$1 \le M \le 256,\ 1 \le N \le 256,\ 0 \le P \le 5$$

$$1MHz \le \frac{F_{(refclock)}}{M} \le 200MHz$$

$$200MHz \le \left(\frac{2 \times N}{M}\right) \times F_{(refclock)} \le 650MHz$$

$$1MHz \le F_{(refclock)} \le 400MHz$$

$$6.25MHz \le \left(\frac{2 \times N}{M \times 2^P}\right) \times F_{(refclock)} \le 650MHz$$

Table 54 gives subsystem frequencies for functional, reset and reduced power modes.

Table 54: Clock frequency requirement for different modes

	Functional mode		Reset		Standby mode	Reduced power[1,2]
	MHz	Divide ratio	MHz[3]	Divide ratio	MHz	MHz
PLL						
PLLA	399	NA	399	N/A	0	0
PLLB	533	NA	533	N/A	0	0
Programmable dividers						
CPU	200	3(A)	200	2(A)	0	2
SDRAM	533	1(B)	533	1(B)	0	27
LMI	133	4(B)	133	4(B)	0	6.75
Blitter	100	5(B)	100	3(B)	0	13.5
SYS_CLOCK	100	6(A)	100	5(B)	0	4.5
AV_CLOCK	50	8(A)	50	7(B)	0	3.375
FDMA	288	2(A)	266	2(B)	0	13.5
Flash	108	6(A)	106	5(B)	0	4.5
Frequency synthesizers						
PIX_CLK	27	N/A	27	N/A	0	27
PCM_CLK	var		24.576		0	var
SPDIF_CLK	var		27		0	var
SC_CLK	var		27		0	var
DAA_CLK	32.4		32.4		0	32.4
AUX_CLK	var		27		0	var
AUDIO_CLK	80		80		0	80

1. During reduced power mode, the programmable dividers are sourced with 27 MHz and the functional divide ratio is applied. For this table the Func1 divide ratios are applied.

2. Reduced power cannot be used for semi-synchronous operation.
3. These frequencies are a fallback setting in case x1 to programmable mode is not functioning.

Note: The default reset values in the register PLLx_CONFIG0 will not result in the required PLL frequencies (399 MHz and 533 MHz). To achieve the correct frequencies, PLLA and PLLB must be programmed with the values 0x2404 and 0x4406 respectively following a reset. See Section 18.2.1: *Programming the registers* on page 124*.*

Programmable clock divider

To program the dividers with a particular divide ratio the following values should be used from Table 55.

Table 55: Configuration values for programmable divide ratios

Divider divide ratio	CLKDIVn_DEPTH[1]	CLKDIVn_SEQ[19:0][2]	CLKDIVn_HNO[1]	CLKDIVn_EVEN[1]
2	0x01	0x00AAA	0x1	0x1
3	0x01	0x00DB6	0x0	0x0
4	0x05	0x0CCCC	0x1	0x1
4.5	0x07	0x3399C	0x1	0x0
5	0x04	0x0739C	0x0	0x0
5.5	0x00	0x0071C	0x1	0x0
6	0x01	0x00E38	0x1	0x1
6.5	0x02	0x01C78	0x1	0x0
7	0x03	0x03C78	0x0	0x0
7.5	0x04	0x07878	0x1	0x0
8	0x05	0xF0F0	0x1	0x1
8.5	0x06	0x1E1F0	0x1	0x0
9	0x07	0x3E1FO	0x0	0x0
9.5	0x08	0x7C1F0	0x1	0x0
10	0x09	0xF83E0	0x1	0x1
11	0x00	0x7E0	0x0	0x0
12	0x01	0x00FC0	0x1	0x1
13	0x02	0x01FC0	0x0	0x0
14	0x03	0x03F80	0x1	0x1
15	0x04	0x7F80	0x0	0x0
16	0x05	0x0FF00	0x1	0x1
17	0x06	0x1FF00	0x0	0x0
18	0x07	0x3FE00	0x1	0x1
19	0x08	0x7FE00	0x0	0x0
20	0x09	0xFFC00	0x1	0x1
Semi-synchronous operation				
2	0x01	0x00555	0x1	0x1
4	0x05	0x03333	0x1	0x1
6	0x01	0x001C7	0x1	0x1

1. CLKDIVn_CONFIG2
2. CLKDIVn_CONFIG0, CLKDIVn_CONFIG1

18.3.3 Frequency synthesizer clocks

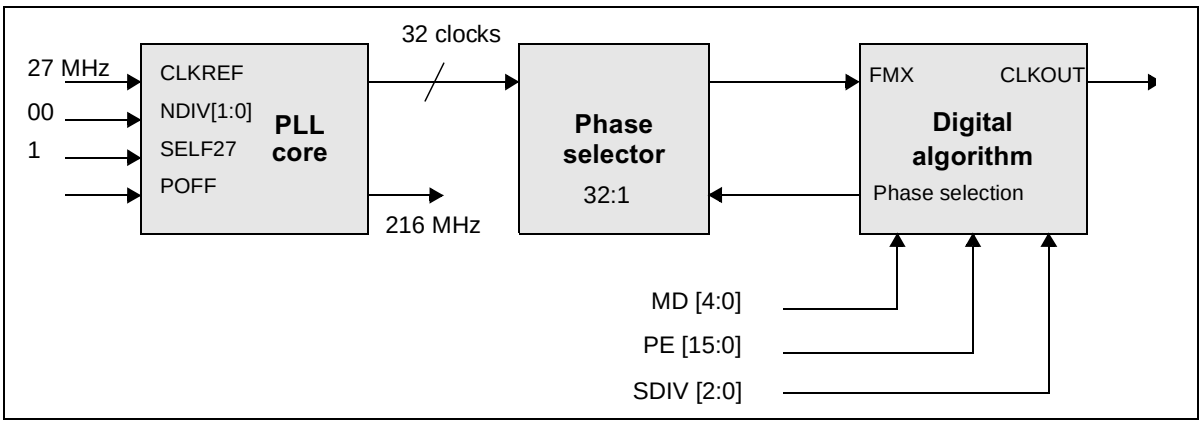
There are two quadruple frequency synthesizers used within the STx5119 with four programmable output clocks per frequency synthesizer. The clocks are generated by each frequency synthesizer using a fixed and stable 27 MHz reference clock. Internally a 216 MHz frequency is generated as well as thirty two 216 MHz clocks with 32 different phase offsets. Currently two outputs from frequency synthesizer B are spare.

The output clock is obtained by a digital algorithm which selects 1 out of 32 phases. Each output clock is configurable between 1 and 216 MHz.

Registers control the output frequency and set up using the following parameters:

- MD[4:0] in the [clock]_SETUP0 register: coarse selector for the phase taps selection (-1 to -16)
- PE[15:0] in the [clock]_SETUP1 register: fine selector for the phase taps selection (1 to 2**15)
- SDIV[2:0] in the [clock]_SETUP0 register: output divider (2 to 256) - SDIV:
  - SDIV = 000 divides by 2,
  - SDIV = 001 divides by 4,
  - SDIV = 010 divides by 8,
  - SDIV = 011 divides by 16,
  - SDIV = 100 divides by 32,
  - SDIV = 101 divides by 64,
  - SDIV = 110 divides by 128,
  - SDIV = 111 divides by 256.

Figure 35: Frequency synthesis



The frequency of the output clock is given by the formula:

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

where F_{PLL} = 216 MHz.

To avoid glitches at the frequency synthesizer output, only the MD, PE and EN_PRG parameters can be changed.

Table 56: PCM audio frequency synthesizer (FS1) programming

Fs (kHz)	256 * Fs (MHz)	MD [4:0]	PE [15:0]	SDIV [2:0]
8	2.048	0x1A = 26	0x5100 = 20736	6
11.025	2.8224	0x13 = 19	0x6F05 = 28421	6
12	3.072	0x11 = 17	0x3600 =13824	6
16	4.096	0x1A = 26	0x5100 = 20736	5
22.05	5.6448	0x13 = 19	0x6F05 = 28421	5
24	6.144	0x11 = 17	0x3600 =13824	5
32[1]	8.192	0x1A = 26	0x5100 = 20736	4
44.1[1]	11.2896	0x13 = 19	0x6F05 = 28421	4
48[1]	12.288	0x11 = 17	0x3600 =13824	4
64	16.384	0x1A = 26	0x5100 = 20736	3
96	24.576	0x11 = 17	0x3600 =13824	3
88.2	22.5792	0x13 = 19	0x6F05 = 28421	3
128	32.768	0x1A = 26	0x5100 = 20736	2
176.4	45.1584	0x13 = 19	0x6F05 = 28421	2
192	49.152	0x11 = 17	0x3600 =13824	2

1. Audio decoder sampling frequency.

Table 57: S/PDIF frequency synthesizer (FS2) programming

S/PDIF_CLK (MHz)	MD [4:0]	PE [15:0]	SDIV [2:0]
12.288 (48 kHz x 256)	0x11 = 17	0x3600 =13824	4
11.2896 (44.1 kHz x 256)	0x13 = 19	0x6F05 = 28421	4
8.192 MHz (32 kHz x 256)	0x1A = 26	0x5100 = 20736	4

Multiplexing after the frequency synthesizer outputs allows clock switching when in standby mode (LP_MODE_SYNTH_DIS[n]). A glitch free multiplexer (GFM) switches cleanly between frequency synthesizer output when transitioning between the x1 and programmable states. See Section 18.5: *Standby mode* on page 130

For improved performance FS_CLK[6] can be used as an additional source clock for CLK_AUX. The switch between the divider clock output and FS_CLK[6] is glitch free and controlled by the ALT_TRANSPORT_SELECT register. The advantage of using a frequency synthesizer clock is the maximum frequency of 216 MHz with fine adjust capabilities.

18.4 Reduced power mode

Reduced power mode gives power saving on individual subsystems and is programmed through the REDUCED_POWER_CONTROL register. The reduced power is achieved by sourcing the divider channel with 27 MHz and using the existing programmed divide ratio to provide a divided 27 MHz subsystem clock output. Bit 9 of the register configures all dividers to reduced power mode. Bits [8:0] configures all dividers to reduced power mode for individual subsystems.

18.5 Standby mode

Low power control (standby mode) is the shut down of subsystems in aid of power consumption. The systems services block is kept operational with 27 MHz and restarts the subsystem clocks in the event of:

- low power alarm,
- external wake up.

For additional power saving the PLLs and frequency synthesizers can also be turned off.

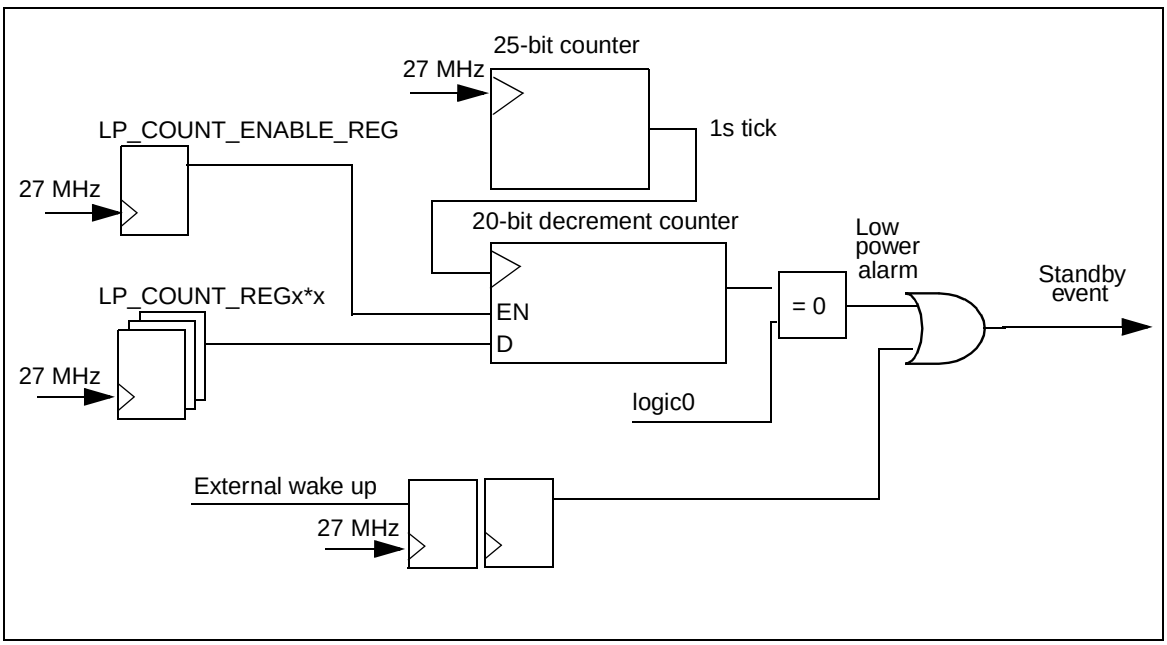
When transitioning back to x1, the subsystems are first clocked off 27 MHz. If registers need to be programmed this can also be done during the transition.

Standby mode, like reduced power mode, is software configurable. After programming the MODE_CONTROL register to the correct value the system services block transitions from F_{max} subsystem clock to logic 0. Software can also override standby mode so that instead of transitioning to logic 0 the resultant subsystem clock is 27 MHz.

18.5.1 Low power control

The low power alarm (LPA) is enabled by LP_COUNT_ENABLE. On assertion of LP_COUNT_ENABLE the LP_COUNT figure is decremented from the preloaded value on a 1 second timer tick. When the decremented count reaches 0 the low power alarm is asserted (the maximum count is 12 days before the low power alarm is asserted).This causes the control FSM to transition to x1 mode. The other event is the active high external wake up. As Figure 36 shows this is synchronized into the 27 MHz domain.

Figure 36: Low power control architecture



18.6 Real-time counter (RTC)

There are two real-time counters (RTCs) in system services, both run continuously within separate clock domains. RTC_LP runs from the 32 kHz LPCLKIN (LP clock domain) and RTC_27 runs from the 27 MHz CLK27IN (27 MHz clock domain). Both RTCs keep track of real time and run in parallel. System requirements determine which RTC is used. The 32 kHz LPCLKIN clock is generated inside the system services using the CLK_27 (27 MHz) input.

18.6.1 RTC_LP

RTC_LP is a 64-bit counter. It is shadowed by two 32-bit registers (RTCS_LSB_LP and RTCS_MSB_LP) in the 27 MHz domain. RTC_LP is never reset and is only programmed through the shadow registers.

Note: Data is always available for reading and the returned value always agrees with what was last written, even if not yet accepted by the RTC_LP.

When a previous write has not yet been accepted a second write cannot take place.

18.6.2 RTC_27

RTC_27 runs continuously from a divided CLK27IN tick. The control of the counter is identical to the control of RTC_LP. The counter tick is a divide of 27 MHz to make it as close to 32 kHz (32.768 kHz) as possible. The tick generated is 27 MHz / 824. Therefore the generated LPCLKIN is 32.767 kHz.

18.7 Integrated clock recovery (VCXO)

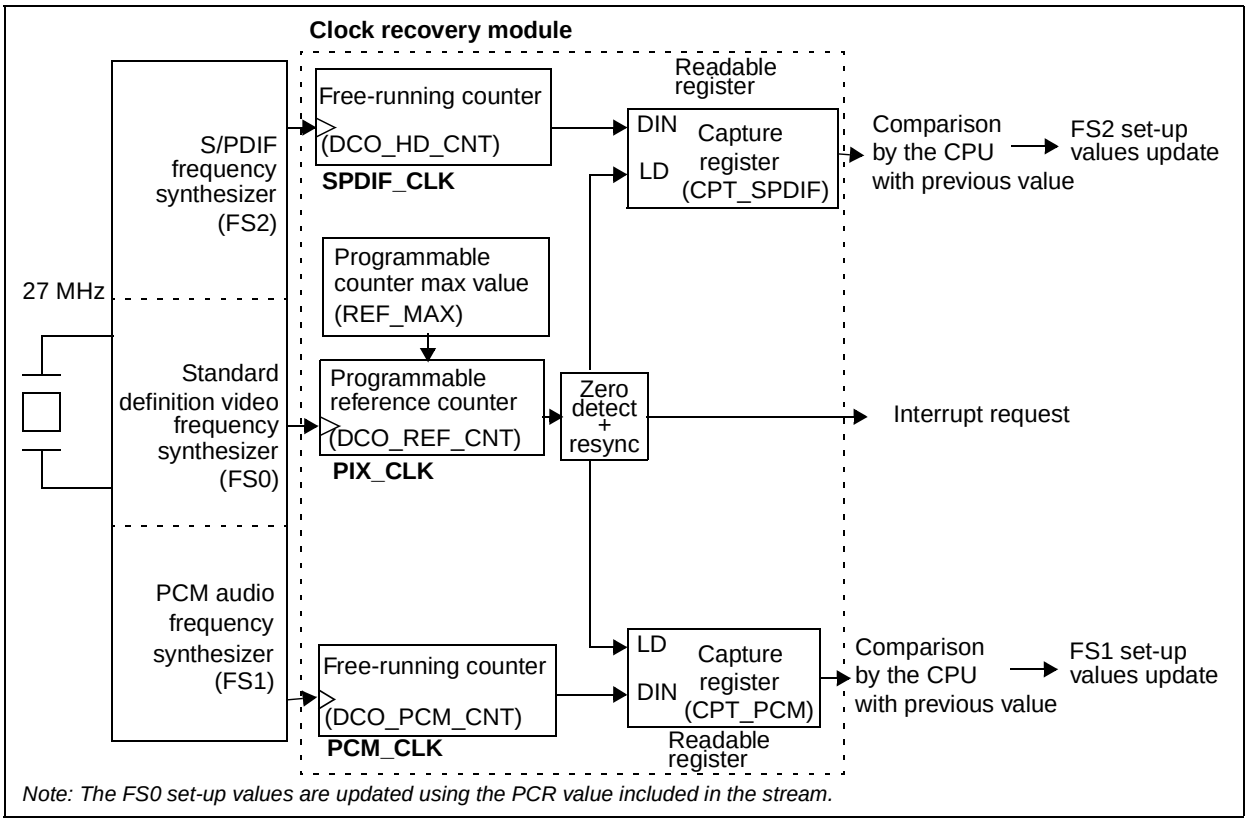
The integrated clock recovery system adjusts three local clocks generated from a triple frequency synthesizer which uses a fixed and stable 27 MHz clock produced by a crystal as a reference.

These three clocks are:

- PIX_CLK (27 MHz) used for standard definition display,
- SPDIF_CLK (74.25 or 148.5 MHz) used for S/PDIF output,
- PCM_CLK (256 * Fs where Fs can take several possible values, for example 32 kHz, 44.1 kHz, 48 kHz) used for audio output.

The VCXO (voltage-controlled crystal oscillator) clock recovery scheme is heavily reliant on interaction between hardware and software. It periodically adjusts the fine adjustment inputs on frequency synthesizer A and B for two of the outputs. The fine adjustment can be carried out independently, while the frequency synthesizers give improved jitter performance.

Figure 37: Block diagram of the clock recovery module



The recovery is done as usual for PIX_CLK by comparing the 42-bit PCR value located in the adaptation field of the stream, with the local STC value when a packet arrived. This generates a potential correction that is applied to the PIX_CLK frequency synthesizer. The frequency synthesizer is programmed with new set-up values to slow or accelerate the clock.

For the audio PCM clock recovery two counters are used:

- a PCM free-running counter clocked by the PCM audio frequency synthesizer FS1.
- a reference counter clocked by the standard definition video frequency synthesizer FS0. The maximum value of this counter is programmable defining the time interval between two consecutive resets. This counter is used as a time-base.

When this reference counter resets, then the values of the free-running counter clocked at PCM_CLK is captured into a readable register. This event generates an interrupt to the CPU. The CPU reads the value and compares it with the previously captured value. The difference between two adjacent values gives an indication of the correction to apply to the PCM audio frequency synthesizer FS1.

The decision to correct the frequency synthesizer set-up is under the control of the software.

The same principle applies for the recovery of the SPDIF_CLK. A free-running counter is clocked with the S/PDIF frequency synthesizer FS2. The same reference counter is used. When this counter resets then the output of the free-running counter clocked at SPDIF_CLK is captured into a readable register.

The counters are 32-bit counters.

18.7.1 Clock recovery commands

The clock recovery module accepts the following command:

`Clock Recovery Reset`: resets the reference counter, PCM counter and HD video counter. See also *DCO_CMD* on page 151.

18.7.2 Clock recovery interrupts

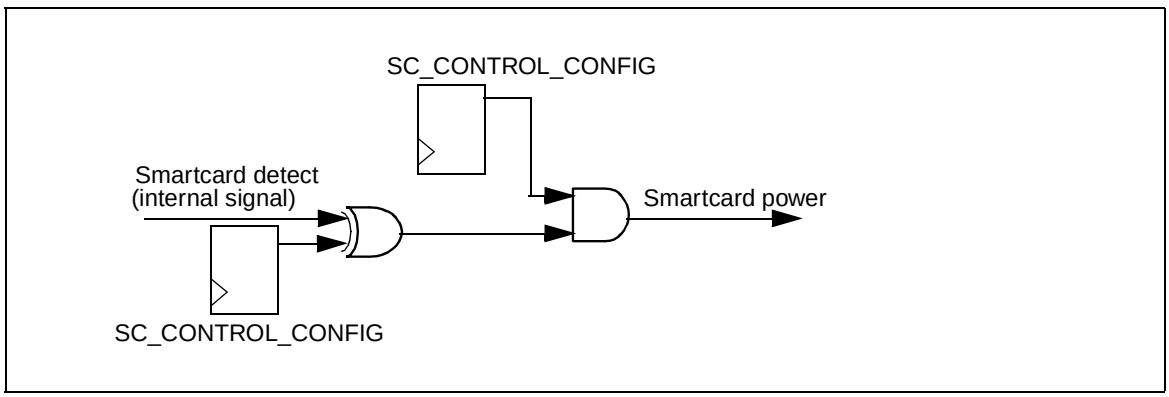
An interrupt line is available, generated from the internal clock recovery mechanism. The interrupt is asserted when the reference counter clocked at PIX_CLK resets.

This interrupt status is captured in the CMD register and only cleared when written.

18.8 Smartcard power control

Smartcard power control detects when the smartcard has been removed from the system. It is a hardware detect as power to the smartcard needs to be removed before software can respond. As can be seen from Figure 38 smartcard power can be disabled or inverted using SC_CONTROL_CONFIG[0].

Figure 38: Smartcard power control architecture

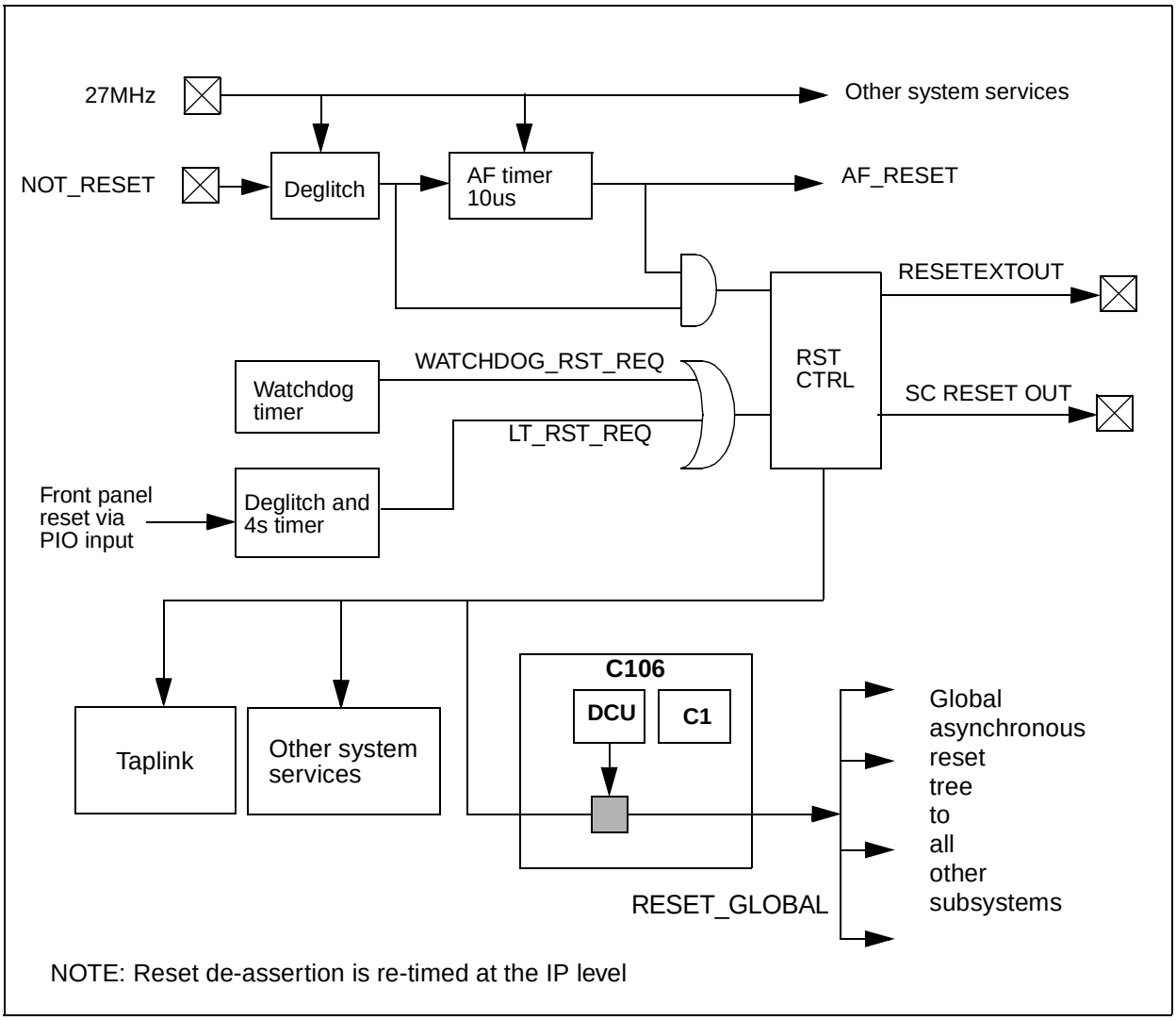


18.9 Reset control

Figure 39 shows the number of different conditions that generate a subsystem or external reset. These are:

- NOT_RESET,
- watchdog reset,
- long time-out reset,
- smartcard insertion reset,
- programmable CPU interrupt.

Figure 39: Reset architecture



18.9.1 Combining resets

When RESET_SELECT_ALT in the RESET_STATUS register is selected resets are combined into one pulse of a minimum 200 ms (stretched reset). The stretched reset is triggered internally by watchdog, long time-out, NOT_RESET and smartcard insertion signals as well as by an external reset. Internal resets are maskable. Therefore the smartcard insertion reset can be disabled using maskable bits.

18.9.2 Power on reset

The power on reset is either the 200 ms stretched reset or the normal reset. When the device powers up, if FMI address 23 is pulled high, the RESET_SELECT_ALT bit is written and the reset stretched.

18.9.3 Watchdog

The watchdog timer is enabled by the WD_COUNT_ENABLE in the WATCHDOG_COUNTER_CFG1 register. On assertion of WD_COUNT_ENABLE the WD_COUNT figure is decremented from the preloaded value on a 1 s timer tick. When the decremented count reaches 1 the watchdog reset is asserted. This causes a subsystem and global asynchronous reset.

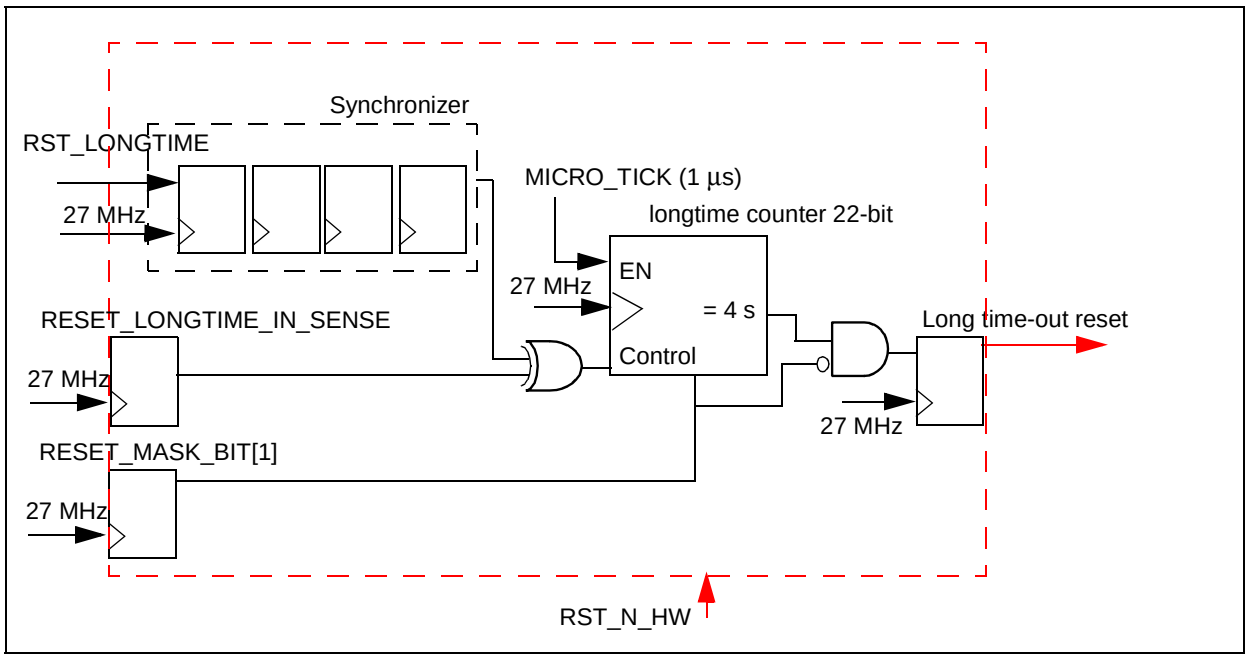
A masking bit in the RESET_STATUS register masks the watchdog reset.

18.9.4 Long time-out reset timer

The long time-out reset timer generates a reset signal if the RST_LONGTIME pin is active for approximately 4 s. The RST_LONGTIME pin is an enable of a counter that generates a 1 s timer tick. This timer tick clocks a second counter which generates a reset pulse when the count reaches 4 s, see Figure 40. A deglitcher provides a clean enable signal to the synchronous counter. The value of the count is zero when the enable is de-asserted.

A masking bit in the RESET_STATUS register masks the long time-out reset. See Figure 40.

Figure 40: Long time-out reset timer architecture



18.9.5 Smartcard reset detect

When the STx5119 detects that a smartcard has been inserted it asserts a reset pulse. A deglitcher removes any bounce in the signal from the smartcard. Smartcard reset is enabled by the SC_INS_RST_EN bit in the SC_INSERTION_RST_CFG register. The sense of the detect signal can also be inverted using the SC_DETECT_SENSE bit in the same register.

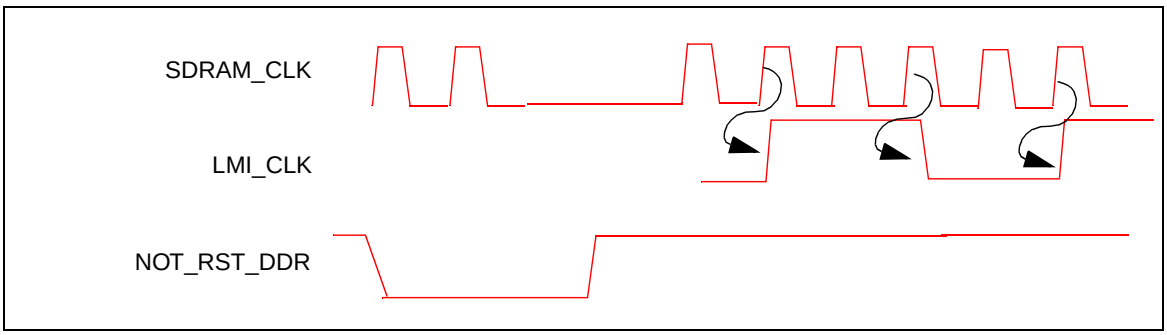
18.9.6 SDRAM interface

There are two resets to the SDRAM interface.

- **Asynchronous reset to the SDRAM**
  The SDRAM clock is disabled a number of cycles before and after the asynchronous reset.
- **Soft reset to the DLL**
  When system services is switched to x1 mode the DLL is reset so that the lower frequency of SDRAM and LMI clocks does not affect the DLL. When switching to programmable mode the DLL comes out of soft reset. Figure 41 shows the timing relationship between NOT_RST_SDRAM the SDRAM clock and the LMI clock.

Note: LMI_CLK is a divide by 4 of DDR_CLK.

Figure 41: SDRAM interface timing



18.10 CPU programmable interrupt

The CPU programmable interrupt is asserted when a count ranging from 1 to 100 ms reaches zero. The counter (CPU_INT_COUNT_CONFIG[6:0] in CPU_INT_CFG) is decremented on a microsecond tick. The reset condition is 10 ms but it can be programmed over a range of 1 to 100 ms. The generated interrupt is synchronized into the CPU clock domain. It's status is captured by a status bit in the CPU_INT_CFG register which is cleared when written to.

Note: Registers (except addresses 0x160 to 0x16F) are read only until they have been unlocked (write access). This is carried out with the REGISTER_LOCK_CFG register (address 0x300). See Section 18.2.1: *Programming the registers* on page 124.

19 System services registers

Addresses are provided as the *SysServBaseAddress* + offset.

The *SysServBaseAddress* is:

0x20F0 0000.

Table 58: System services registers

Register name	Description	Address offset	Type
PLLA_CONFIG0	PLLA configuration 0, see page 141	0x000	R/W
PLLA_CONFIG1	PLLA configuration 1, see page 141	0x004	R/W
PLLB_CONFIG0	PLLB configuration 0, see page 141	0x008	R/W
PLLB_CONFIG1	PLLB configuration 1, see page 141	0x00c	R/W
FSA_SETUP	FSA setup configuration register, see page 144	0x010	R/W
SPARE1_FS_CLK_SETUP0	Spare 1 clock from FSsetup configuration registers, see page 144, page 145	0x014	R/W
SPARE1_FS_CLK_SETUP1		0x018	R/W
PCM_CLK_SETUP0	PCM clock setup configuration register, see page 144, page 145	0x020	R/W
PCM_CLK_SETUP1		0x024	R/W
SPDIF_CLK_SETUP0	S/PDIF clock setup configuration register, see page 144, page 145	0x030	R/W
SPDIF_CLK_SETUP1		0x034	R/W
SC_CLK_SETUP0	Smartcard clock setup configuration register, see page 144, page 145	0x040	R/W
SC_CLK_SETUP1		0x044	R/W
PIX_CLK_SETUP0	PIX Clock from FS setup configuration register see page 144, page 145	0x054	R/W
PIX_CLK_SETUP1		0x058	R/W
FDMA_FS_CLK_SETUP0	FDMA Clock from FS setup configuration register see page 144, page 145	0x060	R/W
FDMA_FS_CLK_SETUP1		0x064	R/W
AUX_CLK_SETUP0	Auxiliary clock setup configuration register, page 144, page 145	0x070	R/W
AUX_CLK_SETUP1		0x074	R/W
SPARE2_FS_CLK_SETUP0	Spare2 clock from FS setup configuration register, page 144, page 145	0x080	R/W
SPARE2_FS_CLK_SETUP1		0x084	R/W
CPU_CLKDIV_CONFIG0	CPU clock divider configuration registers, see page 142, page 143,	0x090	R/W
CPU_CLKDIV_CONFIG1		0x094	R/W
CPU_CLKDIV_CONFIG2		0x098	R/W
LMI_CLKDIV_CONFIG0	LMI clock divider configuration registers, see page 142, page 143	0x0A0	R/W
LMI_CLKDIV_CONFIG1		0x0A4	R/W
LMI_CLKDIV_CONFIG2		0x0A8	R/W
BLT_CLKDIV_CONFIG0	BLIT clock divider configuration registers, see page 142, page 143	0x0B0	R/W
BLT_CLKDIV_CONFIG1		0x0B4	R/W
BLT_CLKDIV_CONFIG2		0x0B8	R/W
SYS_CLKDIV_CONFIG0	SYS clock divider configuration registers, see page 142, page 143	0x0C0	R/W
SYS_CLKDIV_CONFIG1		0x0C4	R/W
SYS_CLKDIV_CONFIG2		0x0C8	R/W

Table 58: System services registers

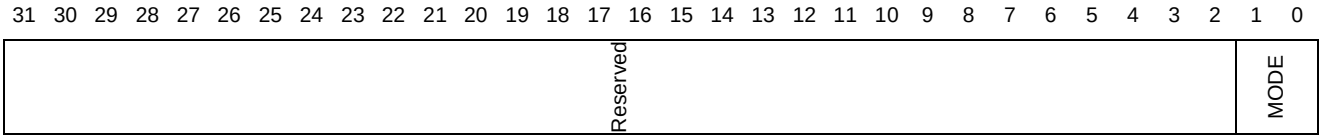
Register name	Description	Address offset	Type
FDMA_CLKDIV_CONFIG0	FDMA clock divider configuration registers, see page 142, page 143	0x0D0	R/W
FDMA_CLKDIV_CONFIG1		0x0D4	R/W
FDMA_CLKDIV_CONFIG2		0x0D8	R/W
VID_CLKDIV_CONFIG0	Video clock divider configuration registers, see page 142, page 143	0x0E0	R/W
VID_CLKDIV_CONFIG1		0x0E4	R/W
VID_CLKDIV_CONFIG2		0x0E8	R/W
SPARE_PLL_CLKDIV_CONFIG0	SPARE PLL clock divider configuration registers, see page 142, page 143	0x0F0	R/W
SPARE_PLL_CLKDIV_CONFIG1		0x0F4	R/W
SPARE_PLL_CLKDIV_CONFIG2		0x0F8	R/W
FLASH_CLKDIV_CONFIG0	Flash clock divider configuration registers, see page 142, page 143	0x100	R/W
FLASH_CLKDIV_CONFIG1		0x104	R/W
FLASH_CLKDIV_CONFIG2		0x108	R/W
MODE_CONTROL	Mode FSM control, see page 139	0x110	R/W
REDUCED_POWER_ CONTROL	Clock divider reduced power selects, see page 146	0x114	R/W
LP_MODE_DIS0	Clock divider low power control register, see page 146	0x118	R/W
LP_MODE_DIS1	Frequency synthesizer low power control register, see page 147	0x11C	R/W
LP_MODE_COUNTER_CFG0	Low power count value, see page 147	0x120	R/W
LP_MODE_COUNTER_CFG1	Low power count value and low power enable, see page 147	0x124	R/W
WATCHDOG_COUNTER_ CFG0	Watchdog counter value, see page 153	0x130	R/W
WATCHDOG_COUNTER_ CFG1	Watchdog counter value and watchdog enable, see page 154	0x134	R/W
RESET_STATUS	Reset status register, see page 152	0x140	R/W
SC_POWER_DETECT_CFG	Smartcard Power control register, see page 152	0x144	R/W
SC_INSERTION_RST_CFG	Smartcard Insertion reset control register, see page 154	0x148	R/W
CPU_INT_CFG	CPU interrupt control register, see page 155	0x150	R/W
DCO_SD_COUNT	DCO reference counter value, see page 150	0x160	R/W
DCO_CMD	DCO command register, see page 151	0x164	R/W
DCO_PCM_COUNT	DCO PCM count value, see page 151	0x168	RO
DCO_HD_COUNT	DCO S/PDIF counter value, see page 151	0x16C	RO
DCO_MODE_CFG	DCO mode control for pixel/PCM clocks, see page 150	0x170	R/W
CLOCK_SELECT_CFG	Multiplexer selects for PLL/dividers, see page 141	0x180	R/W
DIVIDER_FORCE_CFG	Force of control FSM, see page 140	0x184	R/W
CLOCK_OBSERVATION_ CFG	Multiplexer for AUX_CLK, see page 140	0x188	R/W
RTCS_LSB_LP	Low power timer LSB register, see page 148	0x200	R/W
RTCS_MSB_LP	Low power timer MSB register, see page 148	0x204	R/W

Table 58: System services registers

Register name	Description	Address offset	Type
RTCS_CONTROL_LP	Low power timer control register, see page 149	0x208	R/W
RTCS_LSB_27	Low power timer LSB register, see page 149	0x210	R/W
RTCS_MSB_27	Low power timer MSB register, see page 149	0x214	R/W
RTCS_CONTROL_27	Low power timer control register, see page 149	0x218	R/W
REGISTER_LOCK_CFG	Locking/unlocking register control, see page 139	0x300	R/W

19.1 System services control registers

MODE_CONTROL System services mode control



Address: *SysServBaseAddress* + 0x110

Type: Read/write

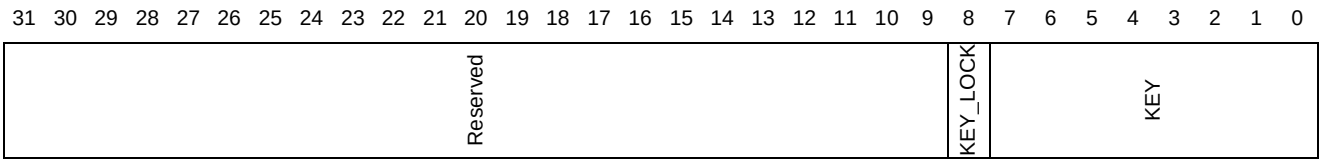
Reset: 0x01

Description:

[31:2] **Reserved**

[1:0] **MODE**[1:0]: mode control:
00: Stay in current state
01: Transition into x1 state
10: Transition into programmable mode
11: Transition into standby mode
Note that this register does not show the current mode

REGISTER_LOCK_CFG Lock and unlock register control



Address: *SysServBaseAddress* + 0x300

Type: Read/write

Reset: 0x0000: KEY[7:0], 0x1: KEY_LOCK

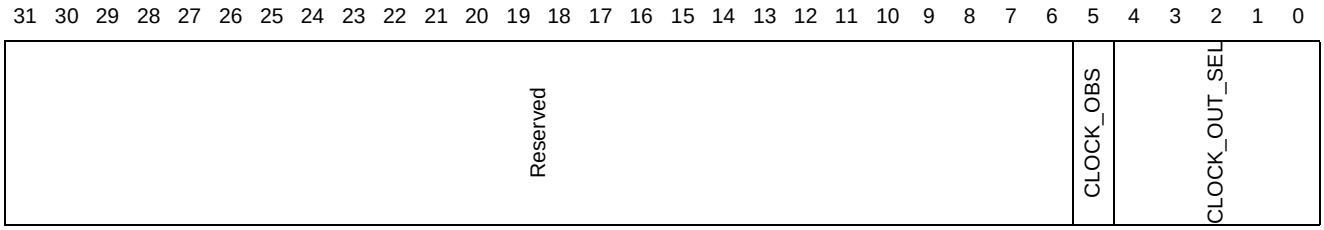
Description: All registers (except the DCO registers at addresses at *SysServBaseAddress* + 0x160 to 0x16F and at *DCOBaseAddress*) are read only until they have been unlocked (given write access).

[31:9] **Reserved**

[8] **KEY_LOCK**: signifies lock state of registers:
0: Registers unlocked for access.
1: All registers locked. Read access only.

[7:0] **KEY**[7:0]: Unlocking keyword, write keyword followed by bit wise inverse to unlock. The keyword can be anything, for example:
key[7:0] = 11110000 followed in the next access by
key[7:0] = 00001111

CLOCK_OBSERVATION_CFG AUX_CLK alternative sources



Address: *SysServBaseAddress* + 0x188

Type: Read/write

Reset: 0x0000

Description:

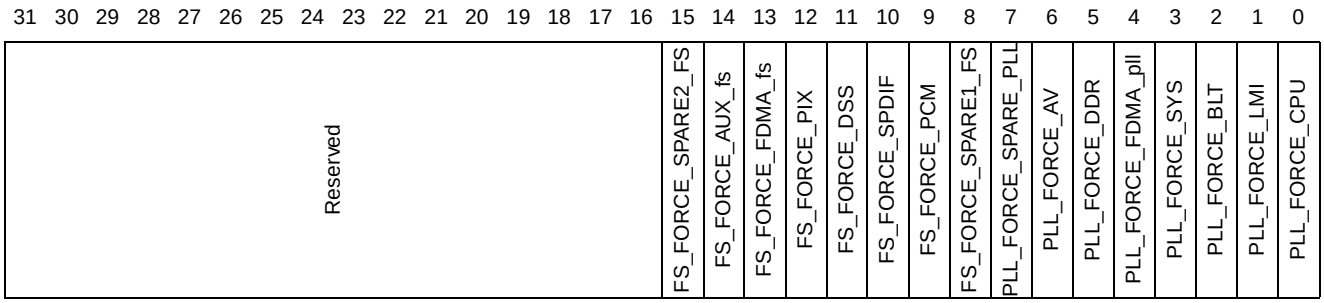
[31:6] **Reserved**

[5] **CLOCK_OBS:** observe alternate clocks on AUX_CLK
0: AUX_CLK (default)
1: Alternate clocks selected by CLOCK_OUT_SEL[4:0]

[4:0] **CLOCK_OUT_SEL**[4:0]: select to source AUX_CLK with other system clocks:

00000: Selects CLK_CPU div by 2	00001: Selects CLK_LMI
00010: Selects CLK_BLT div by 2	00011: Selects CLK_SYS
00100: Selects CLK_FDMA_pll div by 2	00101: Selects CLK_DDR div by 4
00110: Selects CLK_AV	00111: Selects SPARE_PLL_CLK
01000: Selects CLK_FLASH	01001: Selects SPARE1_FS_CLK
01010: Selects CLK_PCM	01011: Selects SPDIF_CLK
01100: Selects CLK_DSS	01101: Selects PIX_CLK
	01111: Select SPARE2_FS_CLK
10000: Selects CLK_27	10001: Selects CLK_LP

DIVIDER_FORCE_CFG Force to programmed output in x1 mode



Address: *SysServBaseAddress* + 0x184

Type: Read/write

Reset: 0x0000

Description:

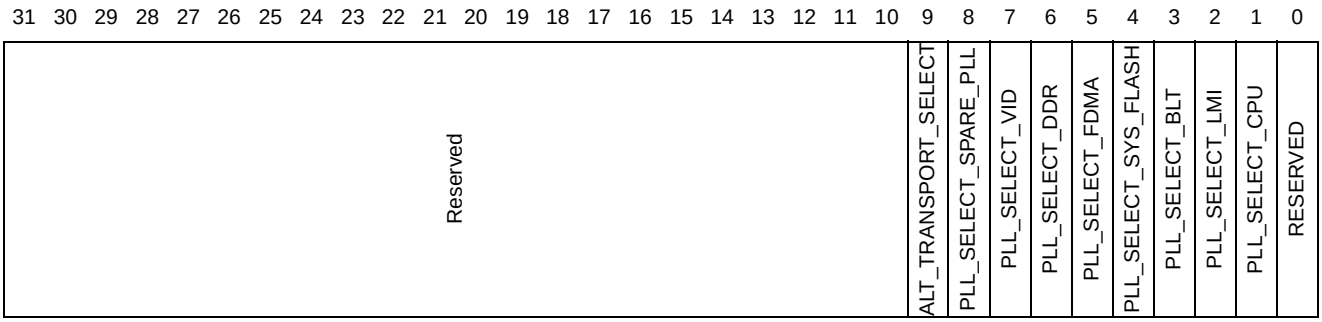
[31:16] **Reserved**

[15:8] **FS_FORCE_x**[7:0]: force frequency synthesizer clocks to programmed output when in x1 mode
0: x 1 mode output clock
1: FS output clock

[7:0] **PLL_FORCE_x**: force divider clocks to programmed output when in x1 mode
0: x 1 mode output clock
1: Divided PLL output clock

19.2 PLL clock generation registers

CLOCK_SELECT_CFG PLL and divider selection



Address: *SysServBaseAddress* + 0x180

Type: Read/write

Reset: 0x00FC

Description:

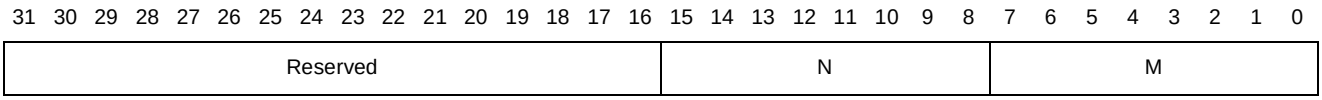
[31:10] **Reserved**

[9] **ALT_TRANSPORT_SELECT**: selects source for AUX_CLK
0: Select PLLA or PLLB
1: Select AUX_CLK generated from FS.
Only glitch clean if programmed within x1 mode

[8:1] **PLL_SELECT_x**: selects PLLA or PLLB for dividers
0: Select PLLA
1: Select PLLB
1bit/divider for system flexibility
Only glitch clean if programmed within x 1 mode
PLL_SELECT_SYS_FLASH selects PLL for CLK_SYS and CLK_FLASH.
The DDR_CLK is sourced from PLLB if either PLL_SELECT_LMI or PLL_SELECT_DDR = 1. The source for DDR_CLK is PLLA when both PLL_SELECT_LMI and PLL_SELECT_DDR are set to 0.

[0] Set to '0'

PLLx_CONFIG0 PLL configuration 0



Address: *SysServBaseAddress* + 0x000 (PLLA), 0x008 (PLLB)

Type: Read/write

Reset: 0x8512 (PLLA: 380.8 MHz), 0xB212 (PLLB: 508.7MHz) (important, see note)

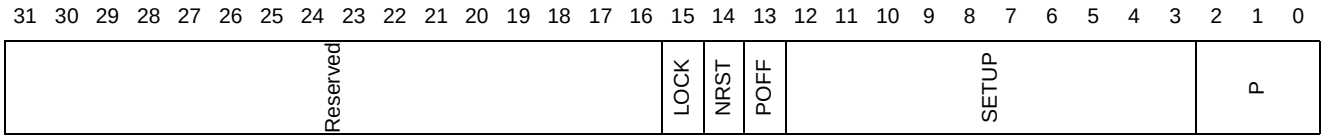
[31:16] **Reserved**

[15:8] **N**[7:0]: N-1 Pre-divider ratio setup from 1 to 126

[7:0] **M**[7:0]: M-1 Pre-divider ratio setup from 1 to 7

Note: The default reset values will not result in the required PLL frequencies (400 MHz and 533 MHz). To achieve the correct frequencies, PLLA and PLLB must be programmed with the values 0x2404 and 0x4406 respectively following a reset. See Section 18.2.1: *Programming the registers* on page 124.

PLLx_CONFIG1 PLL configuration 1



Address: *SysServBaseAddress* + 0x004 (PLLA), 0x00C (PLLB)

Type: Read/write

Reset: 0x4838

Description:

[31:16] **Reserved**

[15] **LOCK:** Read only

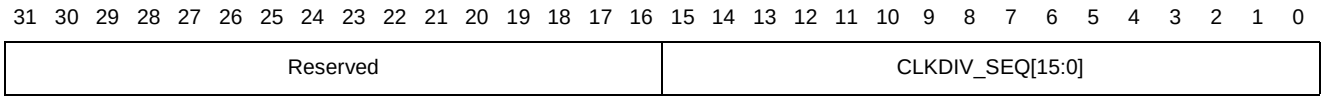
[14] **NRST**: NRST 0 bypass mode, clockout = refclock
1 PLL is on

[13] **POFF:** Power down control
1: power off
0: power on

[12:3] **SETUP**[9:0]: setup[9] not connected
SETUP[8:6] lock detector threshold
SETUP[5] allows lock detector to be reset
SETUP[4] set to 0
SETUP[3:0] charge pump control

[2:0] **P**[2:0]: post-divider ratio setup from 1 to 32

[clock]_CLKDIV_CONFIG0 Clock divider sequence bit pattern



Address: *SysServBaseAddress* + 0x090 (CPU_CLKDIV_CONFIG0),
0x0A0 (LMI_CLKDIV_CONFIG0), 0x0B0 (BLT_CLKDIV_CONFIG0),
0x0C0 (SYS_CLKDIV_CONFIG0), 0x0D0 (FDMA_PLL_CLKDIV_CONFIG0),
0x0E0 (AV_CLKDIV_CONFIG0), 0x0F0 (SPARE_PLL_CLKDIV_CONFIG0),
0x100 (FLASH_CLKDIV_CONFIG0)

Type: Read/write

Reset: 0x0AAA (CLK_CPU), 0xCCCC (CLK_LMI),
0x0DB6 (CLK_BLT), 0x739C (CLK_SYS),
0x0AAA (CLK_FDMA), 0x3C78 (CLK_AV),
0x0DB6 (CLK_SPARE_PLL), 0x0739C (CLK_FLASH)

Description:

[31:16] **Reserved**

[15:0] **CLKDIV_SEQ**[15:0]: Clock divider divide sequence bit pattern

[clock]_CLKDIV_CONFIG1 Clock divider sequence bit pattern

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

Address: *SysServBaseAddress* + 0x094 (CPU_CLKDIV_CONFIG1),
0x0A4 (LMI_CLKDIV_CONFIG1), 0x0B4 (BLT_CLKDIV_CONFIG1),
0x0C4 (SYS_CLKDIV_CONFIG1), 0x0D4 (FDMA_PLL_CLKDIV_CONFIG1),
0x0E4 (AV_CLKDIV_CONFIG1), 0x0F4 (SPARE_PLL_CLKDIV_CONFIG1),
0x104 (FLASH_CLKDIV_CONFIG1)

Type: Read/write

Reset: 0x0000

Description:

[31:5] **Reserved**

[4:0] **CLKDIV_SEQ**[19:16]: Clock divider divide sequence bit pattern

[clock]_CLKDIV_CONFIG2 Clock divider 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																									HNO	EVEN	RESERVED	CLKDIV_DEPTH			

Address: *SysServBaseAddress* + 0x098 (CPU_CLKDIV_CONFIG2),
0x0A8 (LMI_CLKDIV_CONFIG2), 0x0B8 (BLT_CLKDIV_CONFIG2),
0x0C8 (SYS_CLKDIV_CONFIG2), 0x0D8 (FDMA_PLL_CLKDIV_CONFIG2),
0x0E8 (AV_CLKDIV_CONFIG2), 0x0F8 (SPARE_PLL_CLKDIV_CONFIG2),
0x108 (FLASH_CLKDIV_CONFIG2)

Type: Read/write

Reset: 0x0071 (CLK_CPU), 0x0075 (CLK_LMI), 0x0011 (CLK_BLT),0x0014 (CLK_SYS),
0x0071 (CLK_FDMA_PLL), 0x0013 (CLK_AV),
0x0011 (CLK_SPARE_PLL), 0x0014 (CLK_FLASH)

Description:

[31:7] **Reserved**

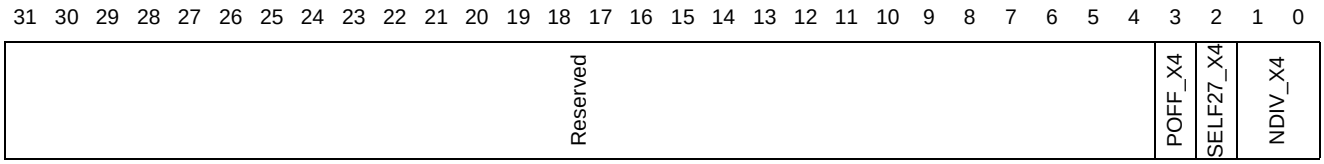
[5:6] **EVEN**, **HNO:**
EVEN 0 HNO 0: Divide by odd whole number, for example 5
EVEN 0 HNO 1: Divide by half ratio, for example 4.5
EVEN 1 HNO 1: Divide by even whole number, for example 6
EVEN 1 HNO 0: Reserved

[4] **RESERVED:**
This bit is always set to ‘1’.

[3:0] **CLKDIV_DEPTH**[3:0]: Number of sequence bits, binary coded.

19.3 Frequency synthesizer clock generation registers

FSx_SETUP Set up frequency synthesizer



Address: *SysServBaseAddress* + 0x010 (FSA)

Type: Read/write

Reset: 0x04

Description:

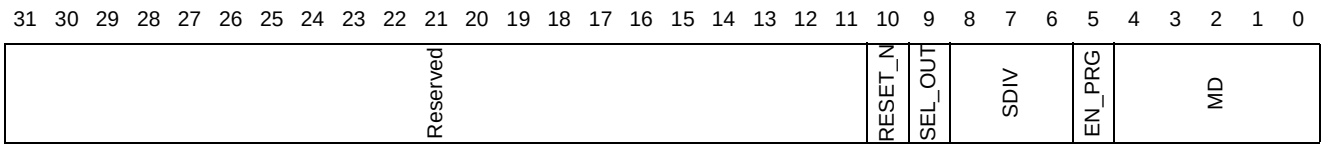
[31:4] **Reserved**

[3] **POFF_X4:** analog part
0: Analog part switched on
1: Analog part switched off

[2] **SELF27_X4**:
0: Internal reference frequency of PLL is 13.5MHz
1: Internal reference frequency of PLL is 27 MHz

[1:0] **NDIV_X4:** coding of the input divider.
00: Input frequency divided by 1
01: Input frequency divided by 2
10: Input frequency divided by 4
11: Divider output set to 0

[clock]_CLK_SETUP0 Clock set up 0



Address: *SysServBaseAddress* + 0x014 (SPARE1_FS_SETUP0), 0x020 (PCM_CLK_SETUP0), 0x030 (SPDIF_CLK_SETUP0), 0x040 (DSS_CLK), 0x054 (PIX_CLK_SETUP0), 0x060 (FDMA_FS_CLK_SETUP0), 0x070 (AUX_FS_CLK_SETUP0), 0x080 (SPARE2_FS_SETUP0).

Type: Read/write

Reset: 0x0B9F: SPARE1_FS_CLK, 0x0AC3: PCM_CLK, 0x0B9F: SPDIF_CLK, 0x0B9F: DSSS_CLK, 0x0B9A: PIX_CLK, 0x0B91 FDMA_FS_CLK,0x0B9F: AUX_CLK, 0x0B9F: SPARE2_FS_CLK

Description:

[31:11] **Reserved**

[10] **RESET_N**
0: Digital CLK algorithm is stopped, PE and MD keep the same values, an output clock is still generated and only internal registers are set to 0 (robust use).
1: Digital CLK algorithm works normally

[9] **SEL_OUT**
0: CLK is the signal given by EXT_CLK
1: CLK is the clock generated by the frequency synthesizer (normal use)

[8:6] **SDIV:** programming of the output divider from 2 to 256 for CLK generation

[5] **EN_PRG:** new incoming data (ME and PE) are taken into account when set to 1.
EN_PRG HAS no effect for PIX_CLK, PCM_CLK and SPDIF_CLK. The EN_PRG pin for these channels is controlled by DCO_MODE_CFG.

[4:0] **MD:** Coarse selector bus for digital algorithm of the phase taps choice

Note: *Before transitioning from X1 to PROG mode these registers must be set to the values in Table 59.*

Table 59: Register values to set before entering PROG mode

Register name	Programmed value	Meaning
PCM_CLK_SETUP0	0x0EF1	24.576 MHz
SPDIF_CLK_SETUP0	0x0EBF	27 MHz
DSS_CLK_SETUP0	0x0EBF	27 MHz
PIX_CLK_SETUP0	0x0EBF	27 MHz
AUX_CLK_SETUP0	0x0EBF	27 MHz

[clock]_CLK_SETUP1 Clock set up 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																PE															

Address: *SysServBaseAddress* + 0x018 (SPARE1_FS_SETUP1),0x024 (PCM_CLK_SETUP1), 0x034 (SPDIF_CLK_SETUP1), 0x044 (DSS_CLK_SETUP1), 0x058 (PIX_CLK_SETUP1), 0x064 (CLK_FDMA), 0x074 (AUX_CLK_SETUP1),0x084 (SPARE2_FS_SETUP1)

Type: Read write

Reset: 0x0000: SPARE1_FS_CLK, 0x3600: PCM_CLK, 0x0000: SPDIF_CLK, 0x000: DSS_CLK, 0x2AAA: PIX_CLK, 0x000: FDMA_FS_CLK, 0x0000: AUX_CLK, 0x3333: SPARE2_FS_SETUP1

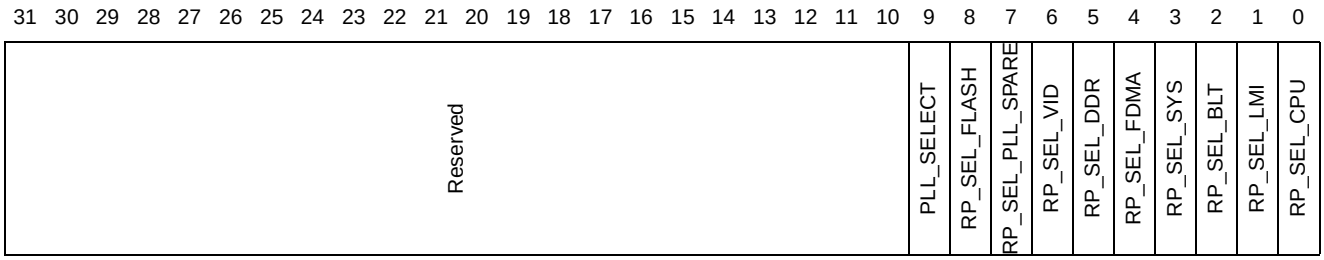
Description:

[31:16] **Reserved**

[15:0] **PE:** Fine selector bus for digital algorithm of the phase taps choice

19.4 Reduced power control

REDUCED_POWER_CONTROL Select reduced power mode.



Address: *SysServBaseAddress* + 0x114

Type: Read/write

Reset: 0x0000

Description: The reduced power is achieved by sourcing the divider channel with 27 MHz and using the existing programmed divide ratio to provide a divided 27 MHz subsystem clock output. Bit 9 of the register configures all dividers to reduced power mode. Bits [8:0] configures all dividers to reduced power mode on an individual basis.

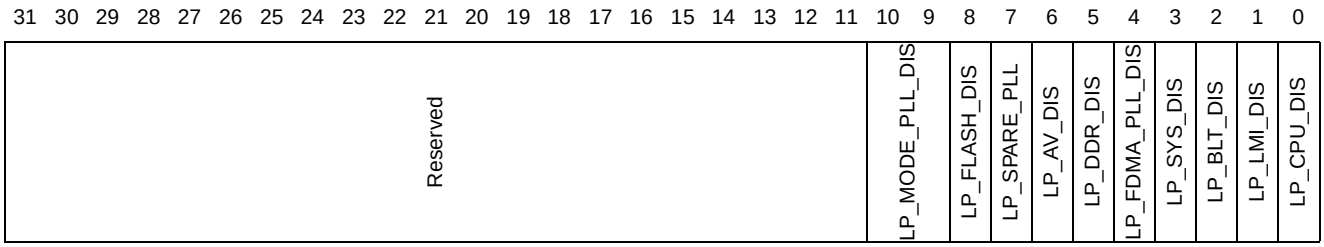
[31:10] **Reserved**

[9] **PLL_SELECT:** selects reduced power for all programmable dividers.

[8:0] **RP_SEL_x**: selects reduced power mode for each programmable divider:

19.5 Low power control registers

LP_MODE_DIS0 Subsystem clock disable



Address: *SysServBaseAddress* + 0x118

Type: Read/write

Reset: 0x01FF

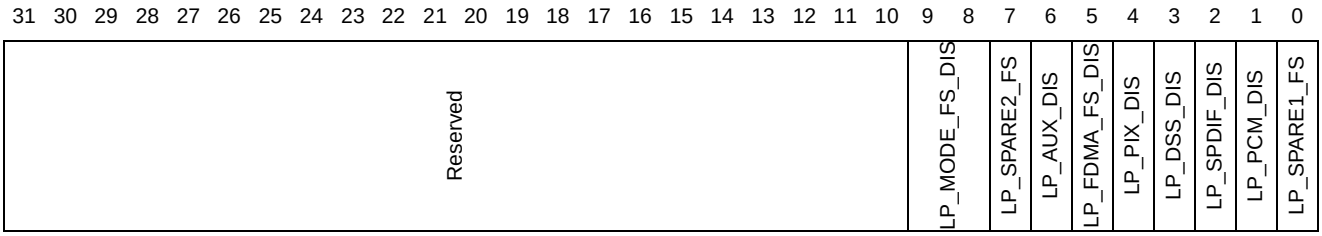
Description:

[31:11] **Reserved**

[10:9] **LP_MODE_PLL_DIS**[1:0]: disables PLL clocks when in standby mode.
LP_MODE_PLL_DIS[0] = 1: PLLA is at logic 0 when in standby mode.
LP_MODE_PLL_DIS[1] = 1: PLLB is at logic 0 when in standby mode.
This is to give lower power requirements in standby mode.

[8:0] **LP_x_DIS**: disables clock when in standby mode.
1: CLK_x is at logic 0 when in standby mode.
If not asserted then CLK_x is 27 MHz.

LP_MODE_DIS1 Subsystem clock disable



Address: *SysServBaseAddress* + 0x11C

Type: Read/write

Reset: 0x00FF

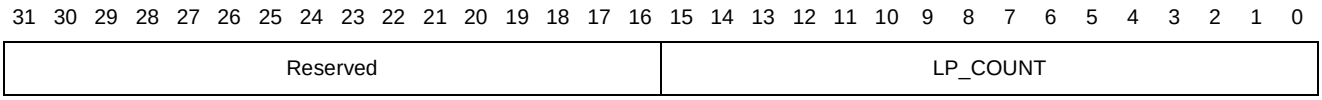
Description:

[31:10] **Reserved**

[9:8] **LP_MODE_FS_DIS**[1:0]: disables frequency synthesizer clocks when in standby mode.
This is to give lower power requirements in standby mode: 10 mW max (analog), 8 mW max (digital)
LP_MODE_FS_DIS[1] = 1: FSB analog component is switched off.
LP_MODE_FS_DIS[0] = 1: FSA analog component is switched off.

[7:0] **LP_x_DIS**: disables subsystem clock when in standby mode.
If not asserted the clock is 27MHz.

LP_MODE_COUNTER_CFG0 Low power counter configuration



Address: *SysServBaseAddress* + 0x120

Type: Read/write

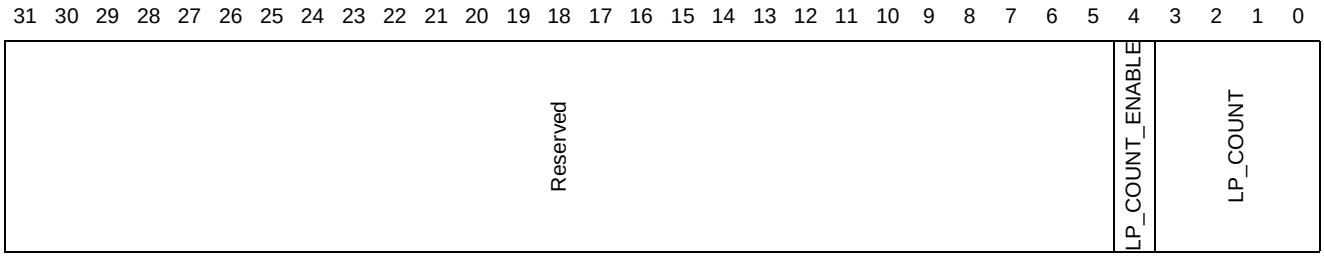
Reset: 0xFFFF

Description:

[31:16] **Reserved**

[15:0] **LP_COUNT**[15:0]: low power count. Binary coded.
LP_COUNT in LP_MODE_COUNTER_CFG1 and LP_MODE_COUNTER_CFG0 is combined to give a maximum count of 1048576

LP_MODE_COUNTER_CFG1 Low power counter configuration



Address: *SysServBaseAddress* + 0x124

Type: Read/write

Reset: 0x000F

Description:

[31:5] **Reserved**

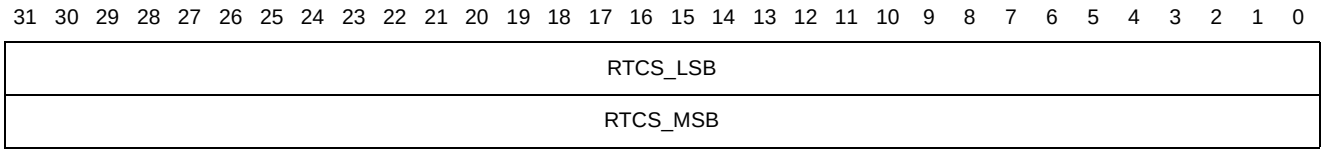
[4] **LP_COUNT_ENABLE**: low power counter enable

[3:0] **LP_COUNT**[19:16]: low power count value programmed by comms subsystem through interconnect. Binary coded.
LP_COUNT in LP_MODE_COUNTER_CFG1 and LP_MODE_COUNTER_CFG0 is combined to give a maximum count of 1048576

19.6 Real-time counter registers

19.6.1 CLK_LP

RTCS_LSB/MSB_LP LSW and MSW of RTC_LP



Address: *SysServBaseAddress* + 0x200 (RTCS_LSB_LP), 0x204 (RTCS_MSB_LP),

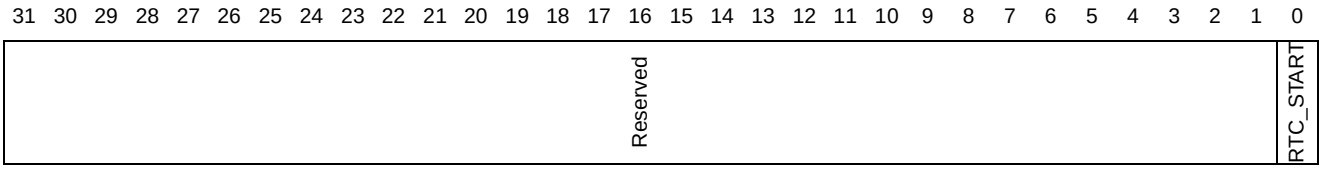
Type: Read/write

Reset: 0x00000000

Description:

[31:0] **RTCS_LSB/MSB**: LSW and MSW of RTC_LP register except during low power mode and when writing a new value to the RTC.

RTCS_CONTROL_LP RTC_LP counter start



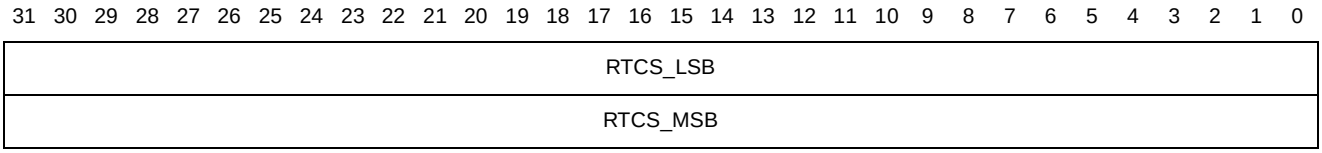
Address: *SysServBaseAddress* + 0x208
Type: Read/write
Reset: 0x0000
Description:

[31:1] **Reserved**

[0] **RTC_START**: 1 starts RTC_LP counting
0 does not stop counting

19.6.2 CLK_27

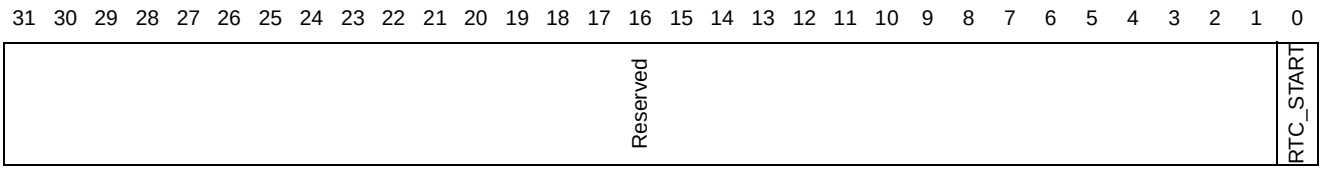
RTCS_LSB/MSB_27 LSW and MSW of RTC_27



Address: *SysServBaseAddress* + 0x210 (RTCS_LSB_27), 0x214 (RTCS_MSB_27)
Type: Read/write
Reset: 0x00000000
Description:

[31:0] **RTCS_LSB/MSB**: LSW and MSW of RTC_27 register except during low power mode and when writing a new value to the RTC.

RTCS_CONTROL_27 RTC_27 counter start



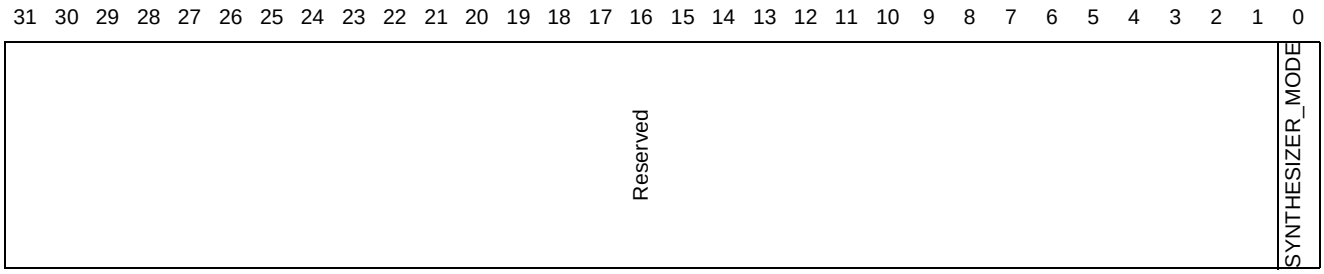
Address: *SysServBaseAddress* + 0x218
Type: Read/write
Reset: 0x0000
Description:

[31:1] **Reserved**

[0] **RTC_START**: 1 starts RTC_27 counting
0: Does not stop counting
1: Starts RTC_27 counting

19.7 Integrated clock recovery (DCO) registers

DCO_MODE_CFG DCO mode

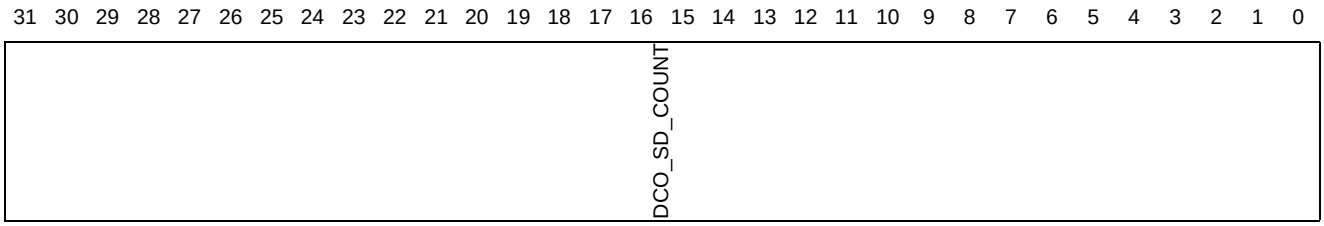


Address: *SysServBaseAddress* + 0x170
Type: Read/write
Reset: 0x0
Description:

[31:1] **Reserved**

[0] **SYNTHESIZER_MODE**: DCO FSM control bit.
0: Programming complete/switch to glitch free clock.
1: Program EN/PE/MD bits.

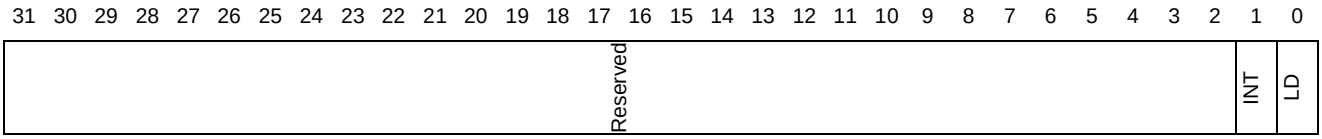
DCO_SD_COUNT DCO reference counter



Address: *SysServBaseAddress* + 0x160
Type: Read/write
Reset: 0x0
Description:

[31:0] **DCO_SD_COUNT**: DCO SD count value. is read when DCP_CMD/LD is 1 or SD_COUNT is 0.

DCO_CMD DCO command



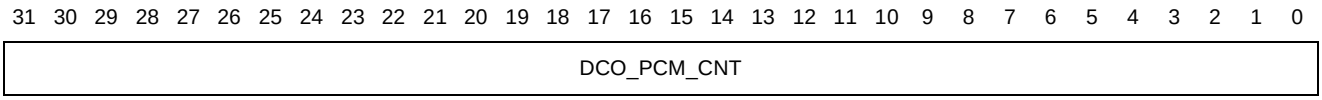
Address: *SysServBaseAddress* + 0x164
Type: Read/write
Reset: 0x0: LD, 0x0: INT
Description:

[31:2] **Reserved**

[1] **INT:** interrupt
When read:
0: No interrupt
1: Interrupt
When 1 is written to this bit the interrupt is cleared and the bit must return to 0 to allow correct operation.

[0] **LD:** reference counter load.
1: Loads DCO_SD_COUNT with value of register DCO_HD_COUNT and DCO_PCM_COUNT with zero.
0: Allows counting.

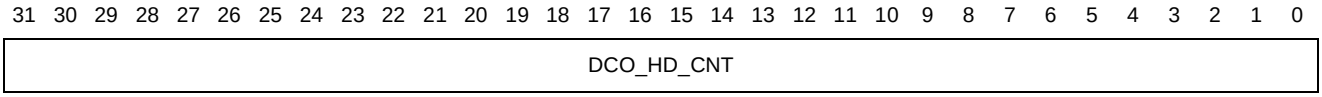
DCO_PCM_COUNT PCM_CLK counter



Address: *SysServBaseAddress* + 0x168
Type: Read only
Reset: 0x0
Description:

[31:0] **DCO_PCM_CNT:** DCO PCM counter value

DCO_HD_COUNT SPDIF_CLK counter

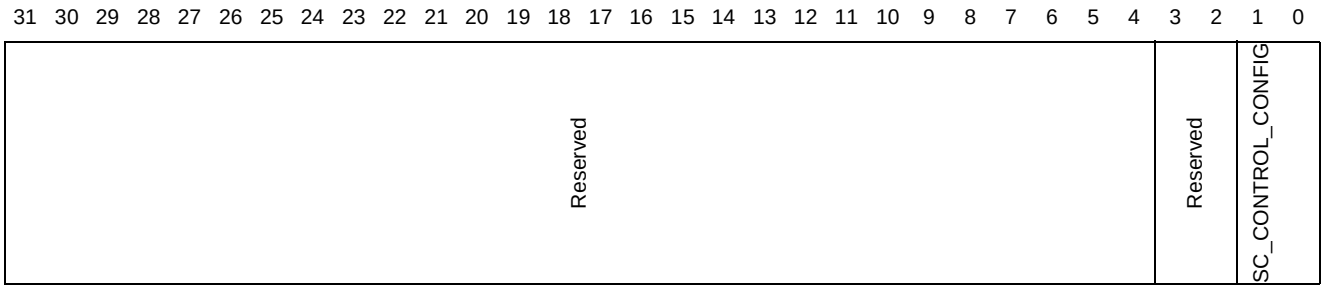


Address: *SysServBaseAddress* + 0x16C
Type: Read only
Reset: 0x0
Description:

[31:0] **DCO_HD_CNT:** DCO S/PDIF counter value

19.8 Smartcard power control registers

SC_POWER_DETECT_CFG Smartcard power control



Address: *SysServBaseAddress* + 0x144

Type: Read/write

Reset: 0x0

Description:

[31:4] **Reserved**

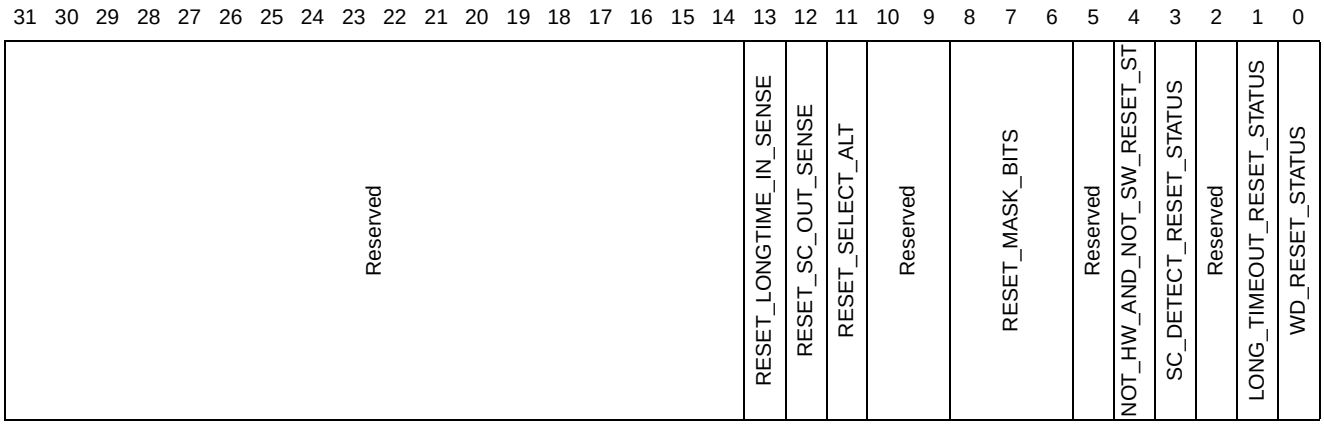
[3:2] **Reserved**

[1] **SC_CONTROL_CONFIG** [1]: Disables smartcard power control for smartcard0
0: Disabled
1: Enabled

[0] **SC_CONTROL_CONFIG** [0]: enables non-inverted/inverted sense power control for smartcard0
0: Enables non-inverted sense
1: Enables inverted sense

19.9 Reset control registers

RESET_STATUS Reset status



Address: *SysServBaseAddress* + 0x140

Type: Read/write

Reset: 0x0: RESET_LONGTIME_IN_SENSE, 0x0: RESET_SC_OUT_SENSE, 0x0: RESET_SELECT_ALT, 0x7: RESET_MASK_BITS[2:0], 0x1: PAD_RESET_STATUS,0x0: SC_DETECT_RESET_STATUS

0x0:LONG_TIMEOUT_RESET_STATUS, 0x0: WD_RESET_STATUS,
0x0: TST_SCANMODE_RESET_STATUS, 0x0: CRYPTO_RESET_STATUS

Description:

[31:14] **Reserved**

[13] **RESET_LONGTIME_IN_SENSE**
0: Active high 1: Active low

[12] **RESET_SC_OUT_SENSE**
0: Active high 1: Active low

[11] **RESET_SELECT_ALT:** select 200ms reset for all reset sources. Alternate reset capabilities.
0 selects 200ms reset pulse 1 selects other resets

[10:9] **Reserved**

[8:6] **RESET_MASK_BITS**[2:0]
0: No mask on reset 1: Mask on reset
[0] Mask for watchdog reset [1] Mask for long time reset
[2] Mask for smartcard detect reset

[5] **TST_SCANMODE_RESET_STATUS**
O: No scanmode reset asserted

[4] **PAD_RESET_STATUS**
0: No pad reset 1: Reset state

[3] **SC_DETECT_RESET_STATUS**
0: No smartcard detect reset 1: Smartcard detect reset asserted

[2] **CRYPTO_RESET_STATUS**
0: No crypto reset asserted

[1] **LONG_TIMEOUT_RESET_STATUS**
0: No long time out reset 1: Long time out reset asserted

[0] **WD_RESET_STATUS**
0: No watchdog reset 1: Watchdog reset asserted

WATCHDOG_COUNTER_CFG0 Watchdog counter 0

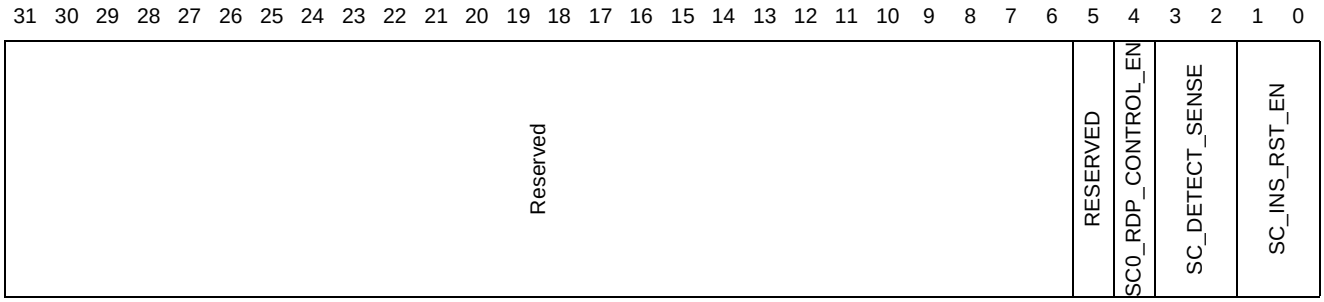
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WD_COUNT															

Address: *SysServBaseAddress* + 0x130
Type: Read/write
Reset: 0xFFFF
Description:

[31:16] **Reserved**

[15:0] **WD_COUNT**[15:0]: watchdog count. Binary coded.
Maximum count of 1048576 seconds. This does not record the current watchdog count.

SC_INSERTION_RST_CFG Smartcard insertion reset



Address: *SysServBaseAddress* + 0x148

Type: Read/write

Reset: 0x0

Description:

[31:6] **Reserved**

[5:4] **SC_RDP_CONTROL_EN**
1: override all other functional controls on SC set of pads.

[3:2] **SC_DETECT_SENSE:** Enables noninverted/inverted sense of smartcard insertion
0: Enables noninverted sense
1: Enables inverted sense

bit[2] is for smartcard 0

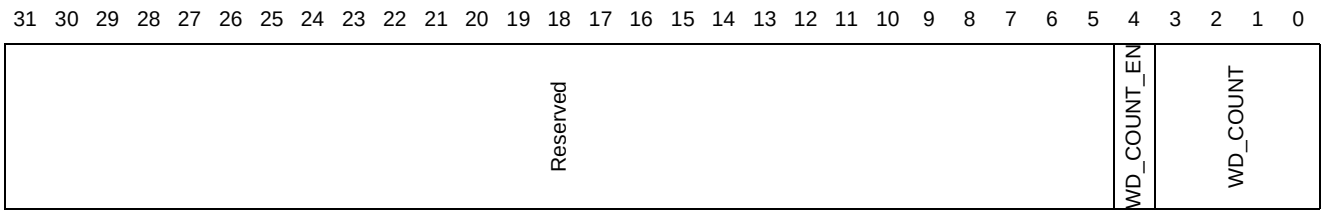
bit[3] is **RESERVED** and should be always set to '0'

[1:0] **SC_INS_RST_EN**
0: No hardware insertion reset
1: Hardware insertion reset

bit[0] is for smartcard 0

bit[1] is **RESERVED** and should be always set to '0'

WATCHDOG_COUNTER_CFG1 Watchdog counter 1



Address: *SysServBaseAddress* + 0x134

Type: Read/write

Reset: 0x0: WD_COUNT_EN, 0xF: WD_COUNT

Description:

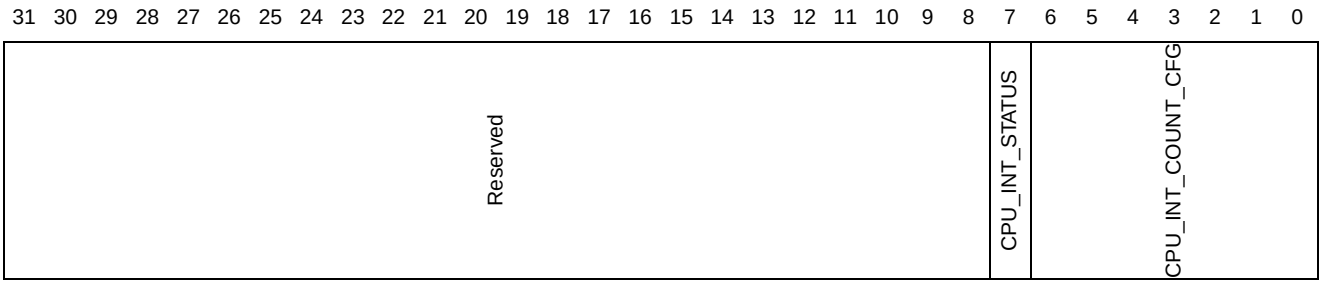
[31:5] **Reserved**

[4] **WD_COUNT_EN:** watchdog counter enable

[3:0] **WD_COUNT**[19:16]: watchdog count. Binary coded.
The values in WATCHDOG_COUNTER_CFG0 and WATCHDOG_COUNTER_CFG1 combine to form a maximum count of 1048576 seconds.

19.10 CPU programmable interrupt register

CPU_INT_CFG CPU interrupt configuration



Address: *SysServBaseAddress* + 0x150

Type: Read/write

Reset: 0x0: CPU_INT_STATUS. 0x0A: CPU_INT_COUNT_CFG

Description:

[31:8] **Reserved**

[7] **CPU_INT_STATUS:** When read shows status of interrupt:
1: interrupt reached 0: Interrupt not reached.
Writing 0 clears interrupt and status.
Writing from 0 to 1, loads a new count value and the count is started. Leave at 1 to continue counting.

[6:0] **CPU_INT_COUNT_CFG:** Count value programmable between 1.024 ms to ~130ms in 1.024ms increments.

20 Programmable transport interface (PTI)

20.1 Overview

The PTI is a dedicated transport engine. It contains its own CPU and handles the transport deMUX functionality of the set-top box.

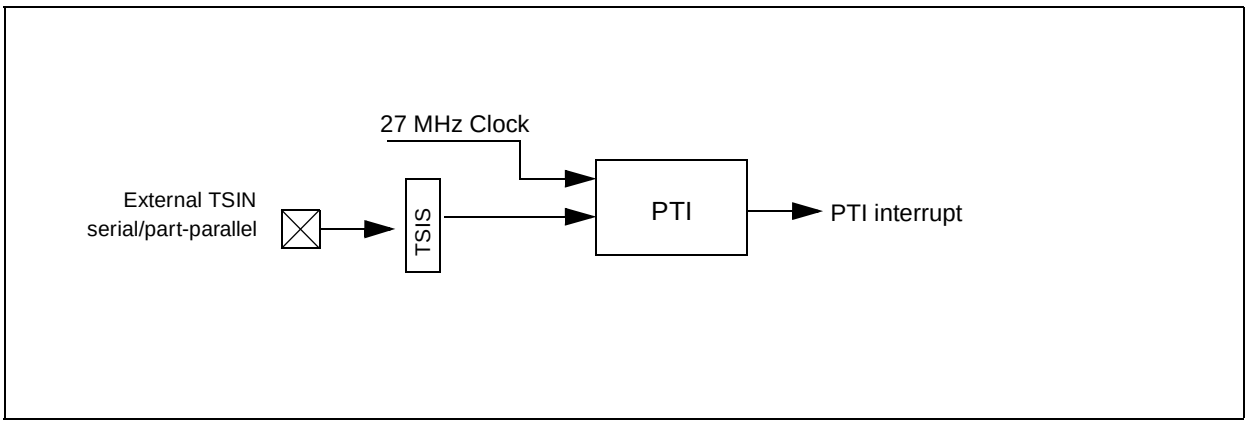
The transport stream sub system is a wrapper level that instantiates the TSIS, PTI and glue logic. The schemes that allow dual POD and DVBCI support are described in the Section 3.5: Transport stream processing on page 15.

The TSIN input can be configured as a serial or parallel input. The interface between the transport stream clock (TSBYTECLOCK) and the SYSCLK domain is confined to the TSIS block. The TSIS communicates with the PTI on system clock.

The PTI maintains an internal system time clock timer (STC) which keeps track of the encoder clock. The STC is clocked off the recovered 27 MHz clock which is derived from the clockgen integrated VCXO. The set-top box clock recovery software drives the VCXO in such a way that its output clock is locked onto the encoder's clock.

The ST20 sets up the PTI and loads its program through the T1 interface. On reception of a new transport packet the PTI writes its content to the ST20 memory space through the T2 interface.

Figure 42: PTI inputs and outputs



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, 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.

20.2 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 up to 138 Mbit/s,
- descrambling using the following algorithms:
  - DVB-CSA,
  - CBC, DVS042, OFB, CTS,
  - ECB, CBC, DVS042, OFB.
- framing of transport packets (sync byte detection),
- section filtering up to 32 x 8 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 data stream to audio and video MPEG decoders via buffers in memory,
- 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 up to 96 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.

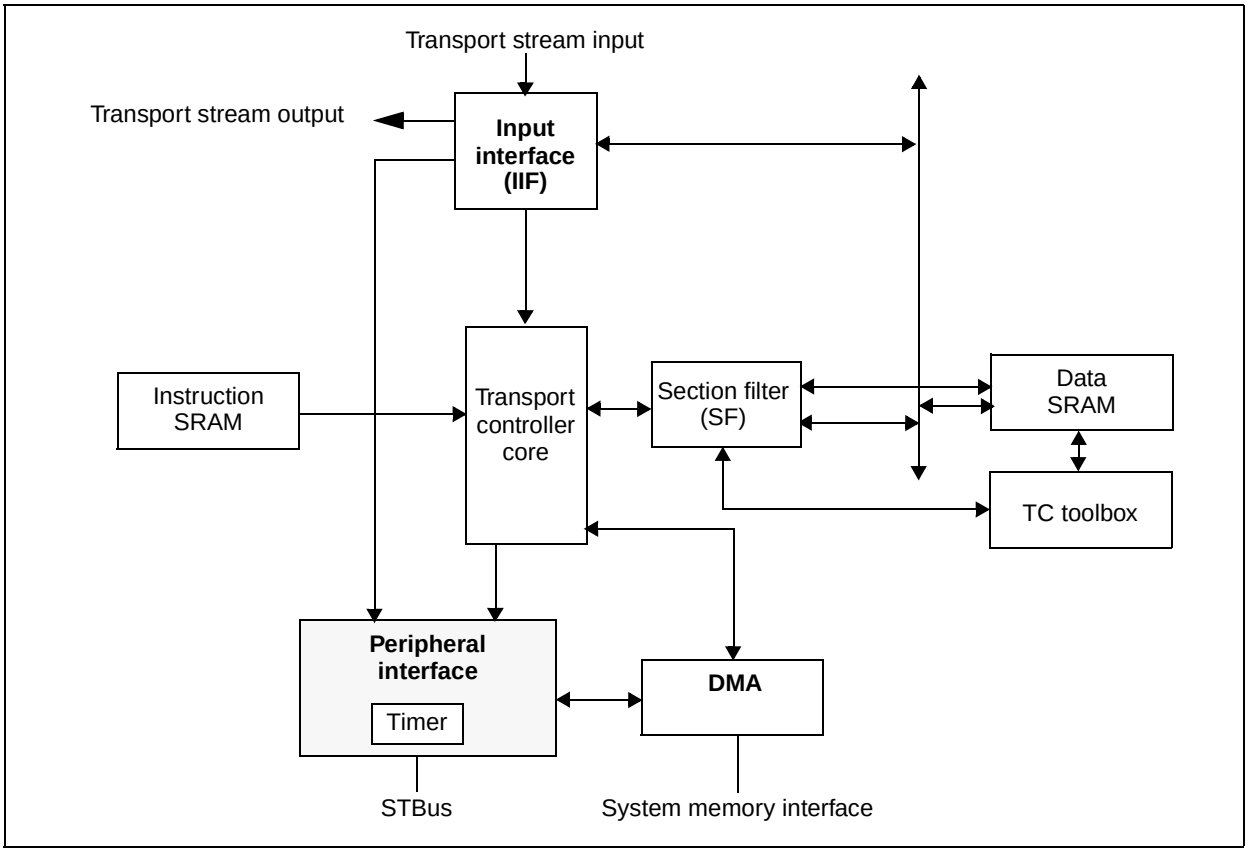
In addition to these transport device functions, the interface copies the entire transport stream or transport packets with selected PIDs from the transport stream through to the PTI output stream interface.

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.

20.3 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 43.

Figure 43: Programmable transport interface architecture



20.3.1 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 96 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. 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.

20.3.2 Shared memory

The PTI contains 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 6-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.

20.3.3 DMA

The DMA block is responsible for writing processed transport data to ST20 memory.

There is one DMA channel. DMA 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. Data can also be written directly to the video and audio decoders under the control of signals generated by the decoder FIFOs. 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-, or 8-word bursts. It may also be configured to output data directly to the decoders.

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,
- channel 0 status,
- memory map selection (programming mode).

20.3.4 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 20.4.2: *Soft reset* on page 161).

20.3.5 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 60.

Table 60: 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.

20.4 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.

20.4.1 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.

20.4.2 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.

20.4.3 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, 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): 32 filters of 8 bytes each,
- long match mode (LMM): 16 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.

20.4.4 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.

20.5 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.

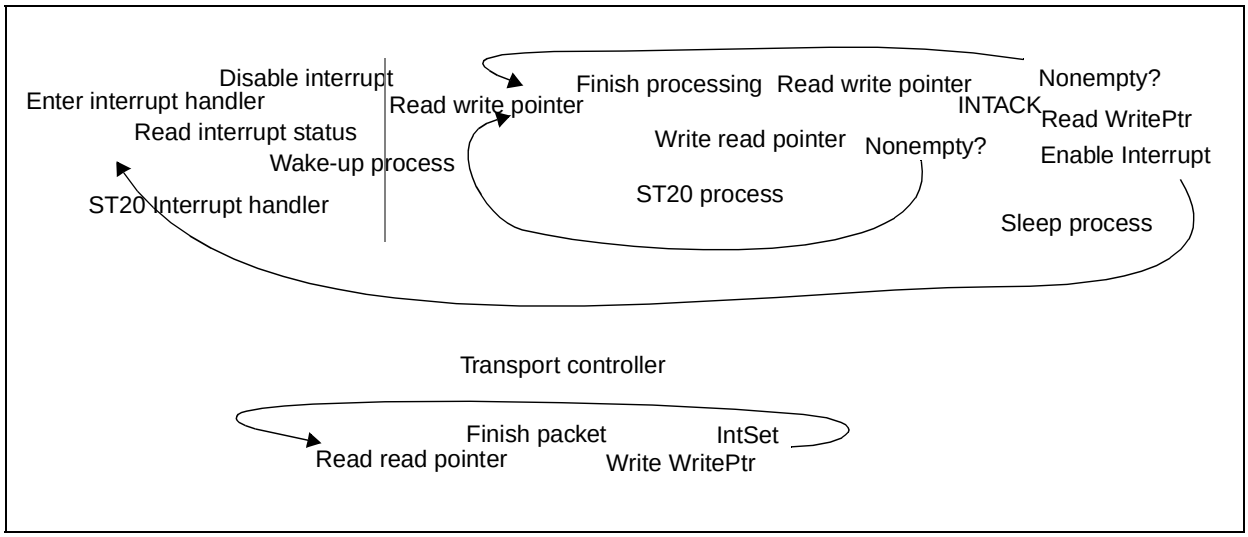
The TC sets a given interrupt by writing to the appropriate bit position in the interrupt status register and the ST20 resets interrupts by writing to the appropriate bit position in the appropriate interrupt acknowledge register. The ST20 determines which interrupts are currently set by reading the 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 44 shows the TC and the ST20 processes and mechanism described above.

Figure 44: PTI buffer interrupt handling



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.

20.6 DMA operation

The PTI contains a DMA controller which is programmed by the TC and the ST20 CPU. The channel 0 is used to write or read data to or from circular buffers defined by a base and a top pointer. There is also a read and a write byte pointer.

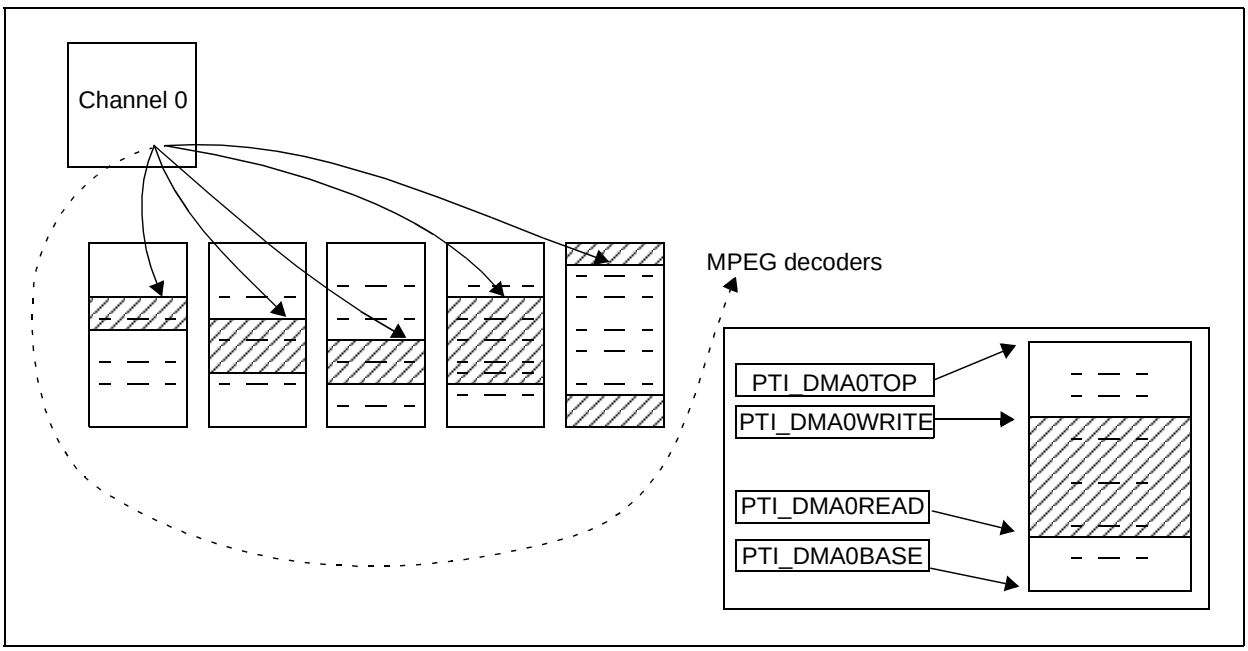
The TC uses DMA channel 0, a write only DMA, to send the data from a transport packet to a circular buffer in the ST20 memory space, or when required, directly to a decoder mapped on to a fixed memory location. Since the TC reloads the registers of channel 0 at the start of processing each transport packet, this allows payloads from transport packets with different PIDs to be output to different buffers. 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.

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.

Figure 45: DMA channel



20.6.1 Circular buffers

The inset in Figure 45 shows how the buffer pointer registers are used to implement circular buffers for the DMA channel. 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 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.

20.6.2 Flushing the DMA

Before doing a PTI soft reset (see Section 20.4.2: *Soft reset* on page 161), 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.

20.7 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.

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 32 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 4-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

1. *Generic Coding Of Moving Pictures And Associated Audio: Systems, Recommendation H.222.0*, ISO/IEC 13818-1

is 16 x 16-byte or 32 x 8-byte filter plus mask sets. If masks are not required each CAM has 32 x 16-byte or 64 x 8-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 46 illustrates different examples of CAM configuration.

Figure 46: Memory organization when CAM_CONFIG NOT_MASK bit = 1 (masks disabled)

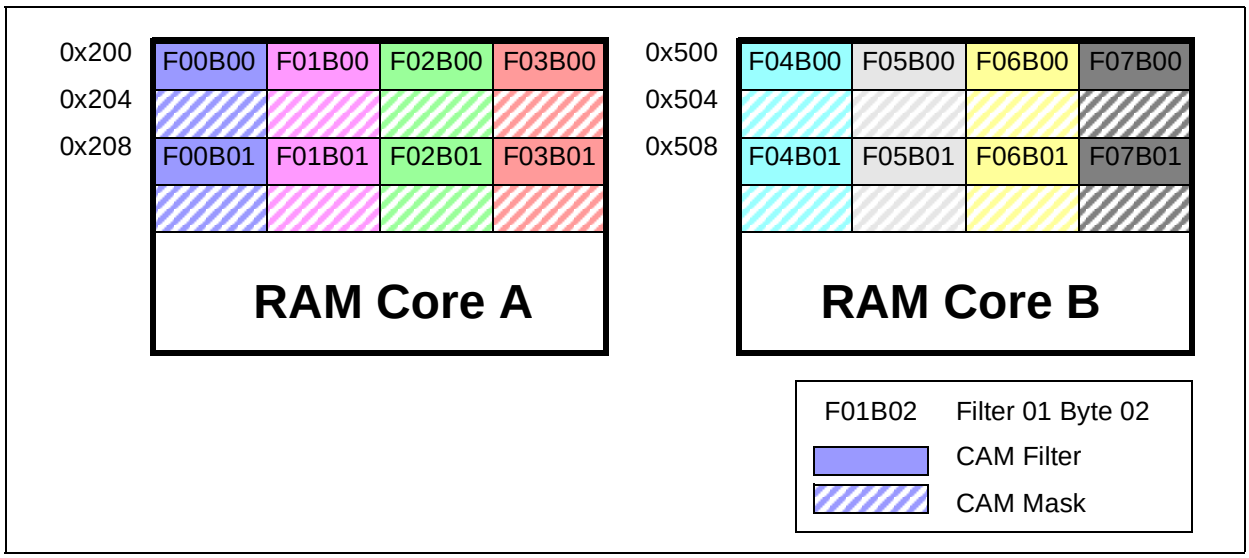


Figure 47: Memory organization with CAM_CONFIG NOT_MASK bit = 0 (masks enabled)

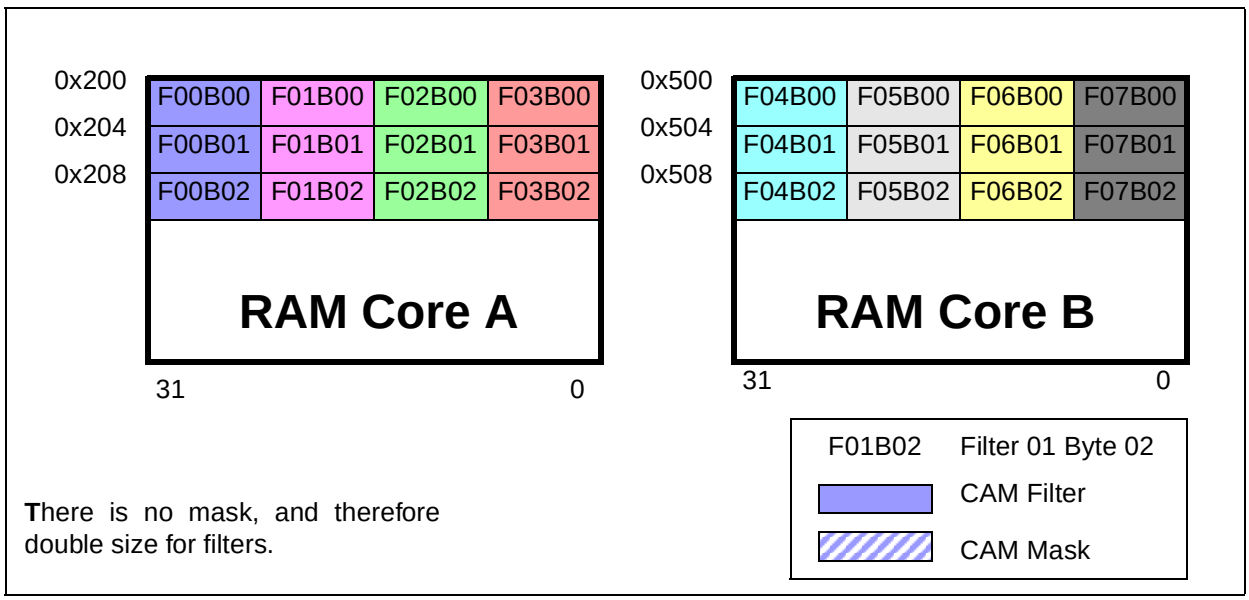


Figure 48: Memory organization for 16 filters

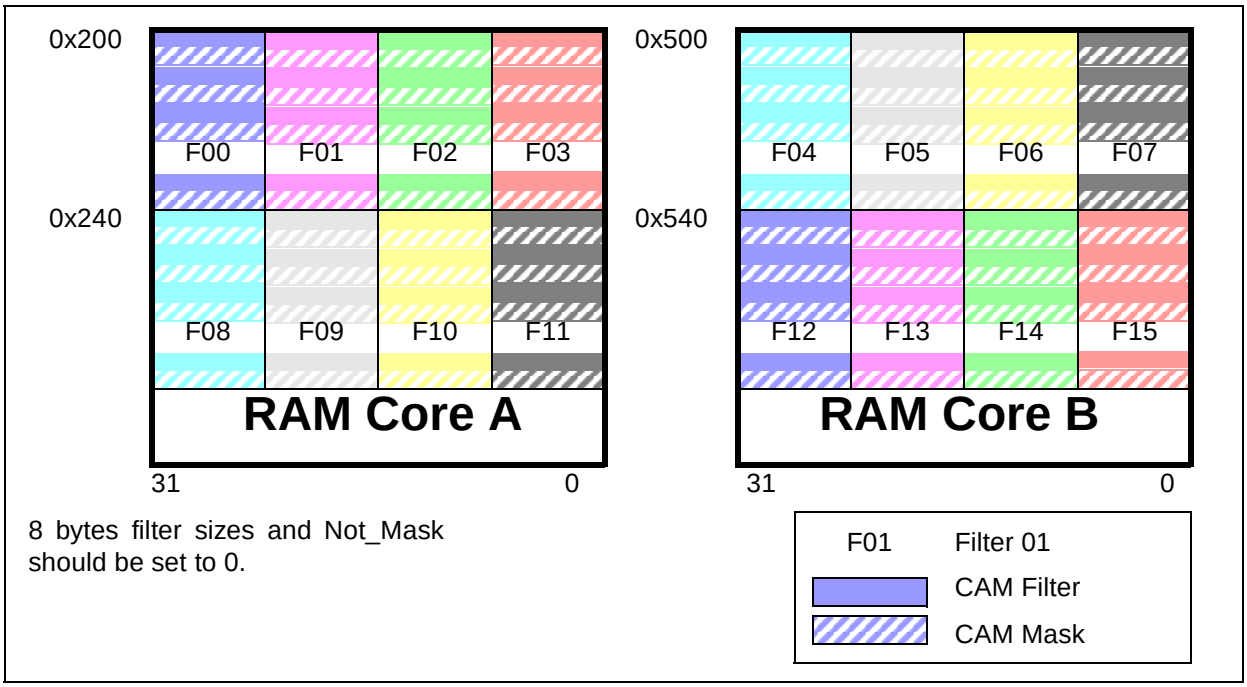
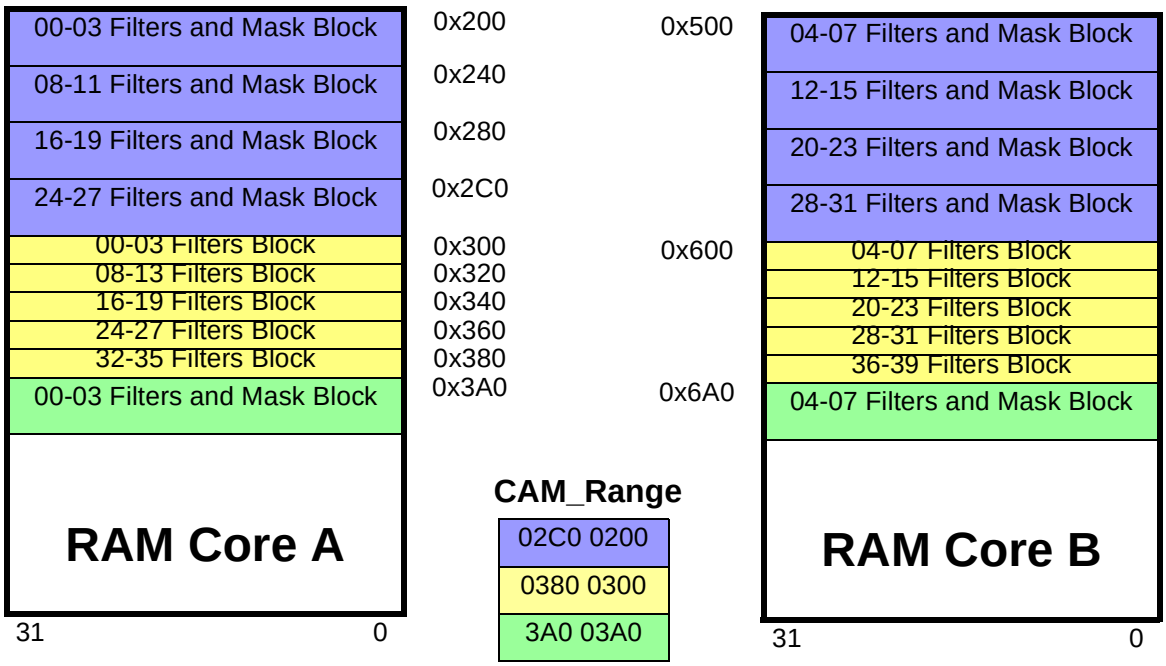


Figure 49: Memory organization when different and mixed cases



This example has filters for three different types of streams. The second (40 filters in yellow) are without mask. It has been assumed (see the memory address) that the filters for all 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 for each of the sets in different contexts.

20.7.1 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.

20.7.2 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 filter n and CAM B filter n,
- positive/negative matching mode: matches on CAM A filter n and not CAM B filter 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 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 61 to Table 63.

PTI_SFDATAn has to be arranged inside CAM memory space as in Section 20.7: *Section filter* on page 165

Table 61: SMM mapping

Section bytes	CAM A	CAM B
5[5:1][1]	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		

1. Used when bit ENn of register PTI_SFNOTMATCHn = 1

Table 62: LMM mapping

Section bytes	CAM A	CAM B
5[5:1][1]	PTI_SFNOTMATCHn[4:0]	Not used
0	PTI_SFDATAn[63:56]	Not used
1	Not used	
2		
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]

1. Used when bit ENn of register PTI_SFNOTMATCHn = 1

Table 63: Positive/negative matching mode mapping

Section bytes	CAM A	CAM B
5[5:1][1]	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		

1. Used when bit ENn of register PTI_SFNOTMATCHn = 1

20.7.3 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.

20.7.4 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).

21 Programmable transport interface (PTI) registers

Addresses are provided as the *PTIBaseAddress* + offset.

The *PTIBaseAddress* is:

0x20E0 0000.

Table 64: Programmable transport interface (PTI) registers: DMA

Register name	Description	Address offset	Type
PTI_DMA0STATUS	DMA channel 0 status, see page 174	0x1018	R/W
PTI_DMA0BASE	DMA buffer base address for channel 0, see page 175	0x1000	WO
PTI_DMA0HOLDOFF	DMA hold off time for channel 0 (for PTI3 only), see page 175	0x1014	R/W
PTI_DMA0READ	DMA buffer read address for channel 0, see page 176	0x100C	WO
PTI_DMA0SETUP	DMA channel 0 byte not word mode and block move (for PTI3 only), see page 176	0x1010	R/W
PTI_DMA0TOP	DMA channel 0 buffer top address, see page 177	0x1004	WO
PTI_DMA0WRITE	DMA channel 0 buffer write address, see page 177	0x1008	R/W
PTI_DMAENABLE	DMA enable, see page 178	0x101C	WO
PTI_DMASEC_START	Start address of the current section, see page 178	0x103C	R/W
PTI_DMAFLUSH	Flush ready for a soft reset, see page 178	0x105C	R/W
PTI_DMAPTI3PROG	Switch PTI1/PTI3 memory maps, see page 179	0x107C	R/W

Table 65: Programmable transport interface (PTI) registers: others

Register name	Description	Address offset	Type
Input interface registers			
PTI_IIFFIFOCOUNT	IIF count, see page 180	0x6000	RO
PTI_IIFALTFIFOCOUNT	IIF alternative FIFO count, see page 179	0x6004	RO
PTI_IIFFIFOENABLE	IIF FIFO enable, see page 180	0x6008	R/W
PTI_IIFALTLATENCY	IIF alternative output latency, see page 179	0x6010	R/W
PTI_IIFSYNCLOCK	IIF sync lock, page 180	0x6014	R/W
PTI_IIFSYNCDROP	IIF sync drop, see page 180	0x6018	R/W
PTI_IIFSYNCCONFIG	IIF sync configuration, see page 181	0x601C	R/W
PTI_IIFSYNCPERIOD	IIF sync period, see page 181	0x6020	R/W
PTI_IIF_HFIFO_COUNT	HFIFO number of bytes, see page 181	0x6024	R/W
PTI configuration registers			
PTI_INTSTATUS(N)	PTI interrupt 0 status, see page 183	0x0000 to 0x000C	RO
PTI_INTENABLE(N)	PTI interrupt 0 enable, see page 183	0x0010 to 0x01C	R/W
PTI_INTACK(N)	PTI interrupt (n) acknowledgment, see page 183	0x0020 to 0x002C	WO
PTI_AUDPTS	Audio presentation time stamp, see page 182	0x0040 to 0x0044	RO

Table 65: Programmable transport interface (PTI) registers: others

Register name	Description	Address offset	Type
PTI_VIDPTS	Video presentation time stamp, see page 184	0x0048 and 0x004C	RO
PTI_STCTIMER	Set STC timer, see page 184	0x0050 and 0x0054	WO
PTI_CFG	PTI configuration, see page 182	0x0058	R/W
Transport controller mode register			
PTI_TCMODE	Transport controller mode, see page 185	0x0030	R/W

21.1 DMA registers

PTI_DMA0STATUS DMA channel 0 status

7	6	5	4	3	2	1	0
Reserved					DMAFIFOFULL	DMA0OVERFLOW	DMA0DONE

Address: *PTIBaseAddress* + 0x1018

Type: Read/write

Reset: 0

Description: The PTI_DMA0STATUS register shows whether DMA channel 0 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.

[7:3] **Reserved**

[2] **DMAFIFOFULL**
If set to 1, the input fifo has reached a full condition and stalled the interface to the TC, preventing any data loss. Writing 1 clears the flag.

[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_DMA0BASE 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
DMA0BASE																															

Address: *PTIBaseAddress* + 0x1000

Type: Write only

Description: Holds the base address of the DMA buffer for the 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_DMA0HOLDOFF 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
Reserved								DMA0WRITESIZE								Reserved								DMA0HOLDOFF							

Address: *PTIBaseAddress* + 0x1014

Type: Read/write

Reset: 0 (DMAnHOLDOFF)
0x10(DMAnWRITESIZE)

Description: The PTI_DMAHOLDOFF registers are used to specify the delay time between the end of a burst of data being transferred and resampling the NOT_CDREQ signal 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.

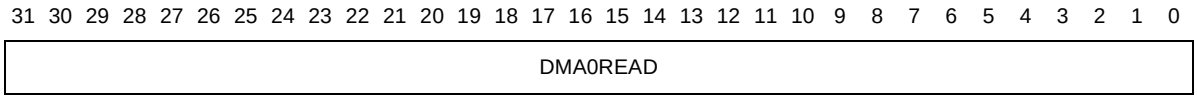
[31:24] **Reserved**

[23:16] **DMAWRITESIZE**
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_DMA0READ DMA buffer read address



Address: *PTIBaseAddress* + 0x100C

Type: Read/Write

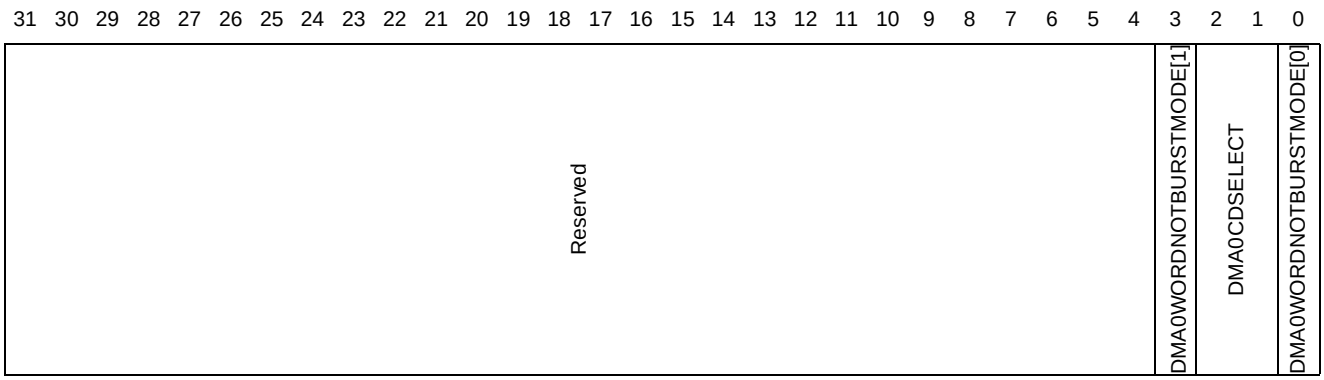
Reset: Undefined

Description: The register holds the read address within the DMA buffer for the DMA channel. The address in the register is always a pointer to the next byte of data 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 is 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. The ST20 software must perform the following checks:

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: Read/write

Reset: 0

Description:

[31:4] **Reserved**

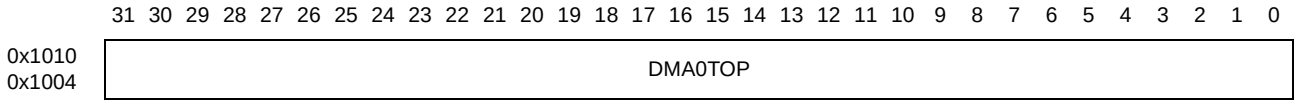
[3] **DMA0WORNOTBURSTMODE[1]**

0xx0: enable 4-word-bursts
0xx1: write word-at-a-time
1xx0: enable 8 word-bursts
1xx1: 16 word-bursts

[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]**
See DMA0WORNOTBURSTMODE[1]

PTI_DMA0TOP DMA buffer top address

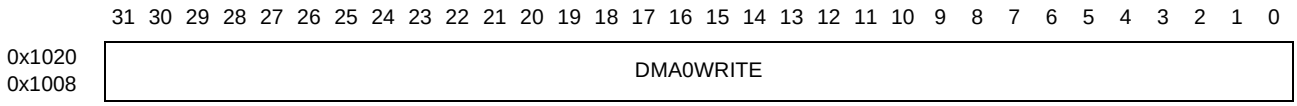


Address: *PTIBaseAddress* + 0x1004

Type: Write only

Description: This register 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_DMA0WRITE DMA buffer write address



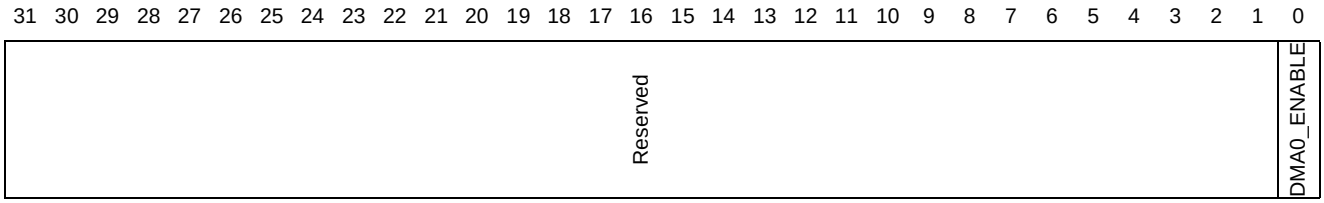
Address: *PTIBaseAddress* + 0x1008

Type: Read/write

Reset: 0x00

Description: This register holds the write address within the DMA buffer for the 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 is set for each PID in the PID data structure in the PTI data SRAM and an updated write pointer is 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.

PTI_DMAENABLE DMA enable



Address: *PTIBaseAddress* + 0x101C

Type: Write only

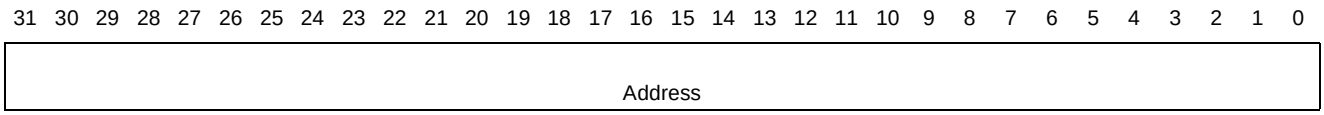
Reset: Undefined

Description: This register controls the enabling of the DMA channel. 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.

[31:1] **Reserved**

[0] **DMA0ENABLE**
Enable (1) or disable(0) channel 0.

PTI_DMASEC_START Start address of the current section



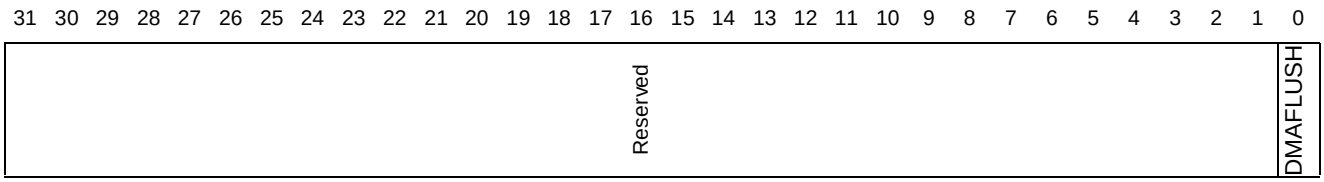
Type: *PTIBaseAddress* + 0x103C

Type: Read/write

Reset: 0x0000

Description: Address of the start of the section that is currently being read by channel 0.

PTI_DMAFLUSH Flush ready for a soft reset



Type: *PTIBaseAddress* + 0x105C

Type: Read/write

Reset: 0

Description: 1 is written to PTI_DMAFLUSH when a flush is required. It self resets to 0 on completion.

PTI_DMAPTI3PROG Switch PTI1/PTI3 memory maps

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															DMAPTI13PROG

Type: *PTIBaseAddress* + 0x107C

Type: Read/write

Reset: 0B0

Description: Switch between PTI1 and PTI3 memory maps.

[31:1] **Reserved**

[0] **DMAPTI3PROG**

o: forces DMA to use PTI1 memory map (default)

1: enable DMA to access PTI3 registers through PTI3 memory map.

21.2 Input interface registers

PTI_IIFALTFIFOCOUNT IIF alternative FIFO 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																								COUNT							

Address: *PTIBaseAddress* + 0x6004

Type: Read only

Reset: Undefined

Description: The number of bytes in the alternative FIFO. This FIFO may overflow since there is no flow control on its input.

PTI_IIFALTLATENCY IIF alternative output latency

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

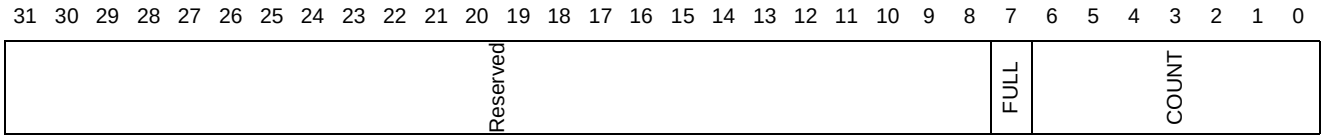
Address: *PTIBaseAddress* + 0x6010

Type: Read/ write

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



Address: *PTIBaseAddress* + 0x6000

Type: Read only

Reset: Undefined

Description:

[31:8] **Reserved**

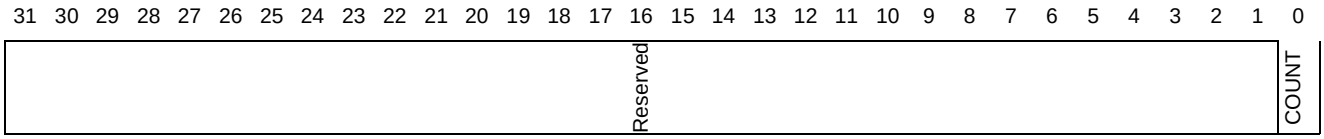
[7] **FULL**

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



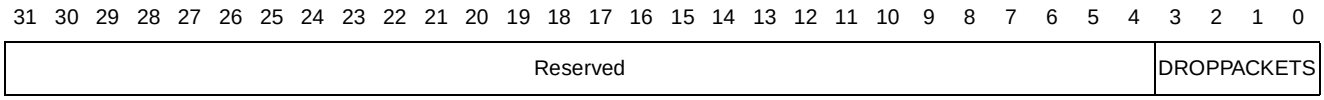
Address: *PTIBaseAddress* + 0x6008

Type: Read/write

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



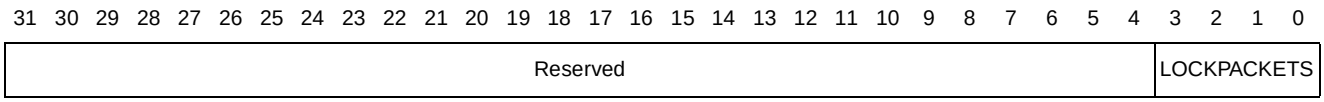
Address: *PTIBaseAddress* + 0x6018

Type: Read/ write

Reset: Undefined

Description: The number of successive erroneous sync bytes found before the lock is treated as lost. Should not be 0 to activate sync drop.

PTI_IIFSYNCLOCK IIF sync lock



Address: *PTIBaseAddress* + 0x6014

Type: Read/ write

Reset: Undefined

Description: The number of successive correct sync bytes found before the sync detection is locked. Should not be 0 to activate sync lock.

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* + 0x601C
Type: Read/ write
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
SYNCPERIOD																															

Address: *PTIBaseAddress* + 0x6020
Type: Read/ write
Reset: 0x00BC
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.

PTI_IFF_HFIFO_COUNT HFIFO number of bytes

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

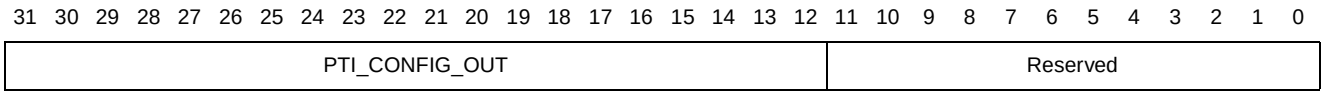
Address: *PTIBaseAddress* + 0x6024
Type: Read/ write
Reset: 0x00
Description: Number of bytes in header FIFO.

[32:8] **Reserved**

[7:0] **HFIFO_LEVEL**

21.3 PTI configuration registers

PTI_CFG PTI configuration



Address: *PTIBaseAddress* + 0x0058

Type: Read/write

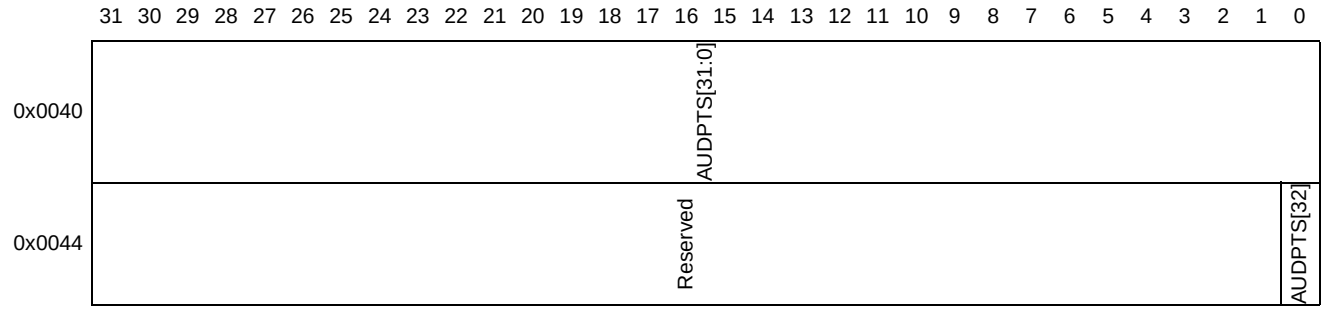
Reset: 0x0000000

Description: This register is present in the PTI and the output is available as PTI_CONFIG_OUT. The TSIS configuration is done using the CONFIG_CRTL_C. See *Configuration control C* on page 50.

[31:12] **PTI_CONFIG_OUT**

[11:0] **Reserved**

PTI_AUDPTS Audio presentation time stamp



Address: *PTIBaseAddress* + 0x0040 and 0x0044

Type: Read only

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: Write only
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

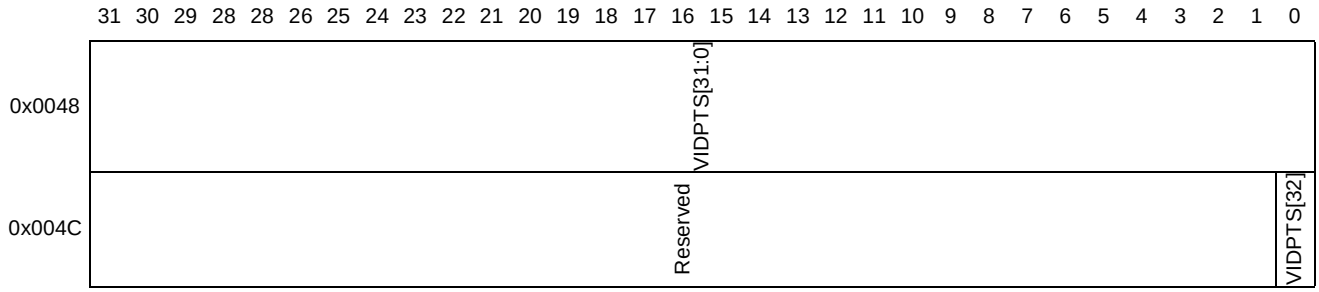
Address: *PTIBaseAddress* + 0x0010 to 0x001C
Type: Read/write
Reset: Undefined
Description: The corresponding interrupt is enabled when a bit is 1.

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: Read only
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

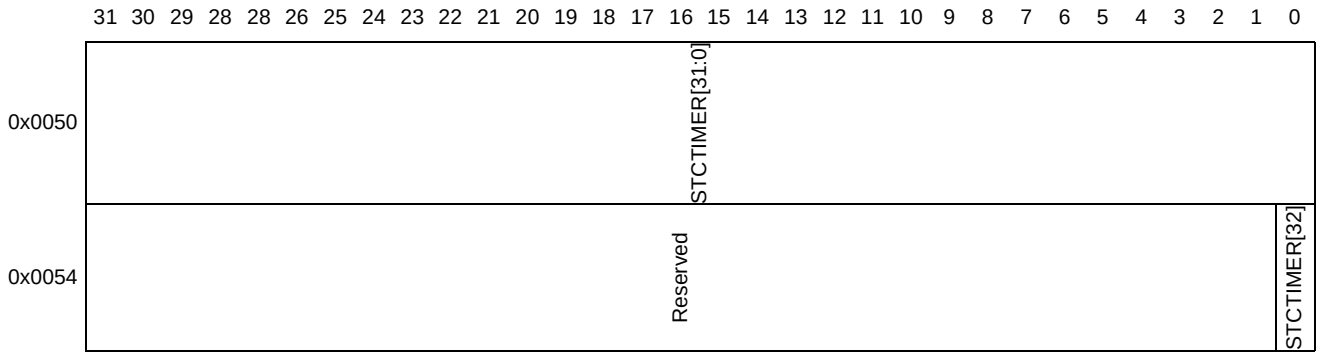
Type: Read only

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



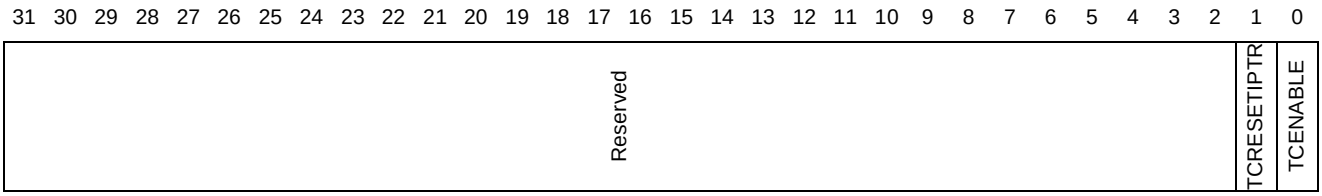
Address: *PTIBaseAddress* + 0x0050 to 0x0054

Type: Read/write

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.

21.4 Transport controller mode register

PTI_TCMODE **Transport controller mode**



Address: *PTIBaseAddress* + 0x0030

Type: Read/write

Reset: 0

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

22 MPEG video decoder

22.1 Overview

The STx5119 MPEG video decoder is a real-time MPEG1 and MPEG2 decoder. It is compliant with the main profile and main 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). In this document, reconstruction may be also called “main reconstruction”. 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 two extra features which are not part of the MPEG standard:

- possible simplified B decoding, see Section 22.5.6: *Simplified B decoding* on page 202,
- error statistics, see Section 22.5.5: *Error statistics* on page 201.

22.2 Main functions

Figure 50 shows the video decoder in the STx5119 environment.

Figure 50: MPEG video decoder in the STx5119

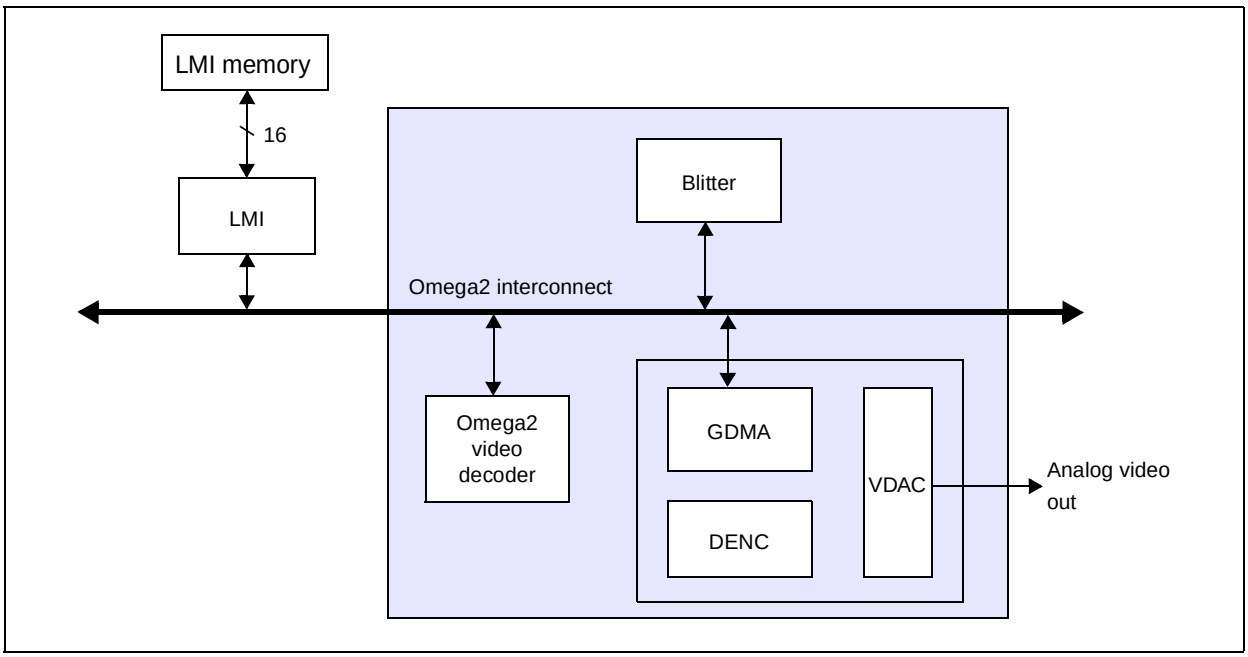
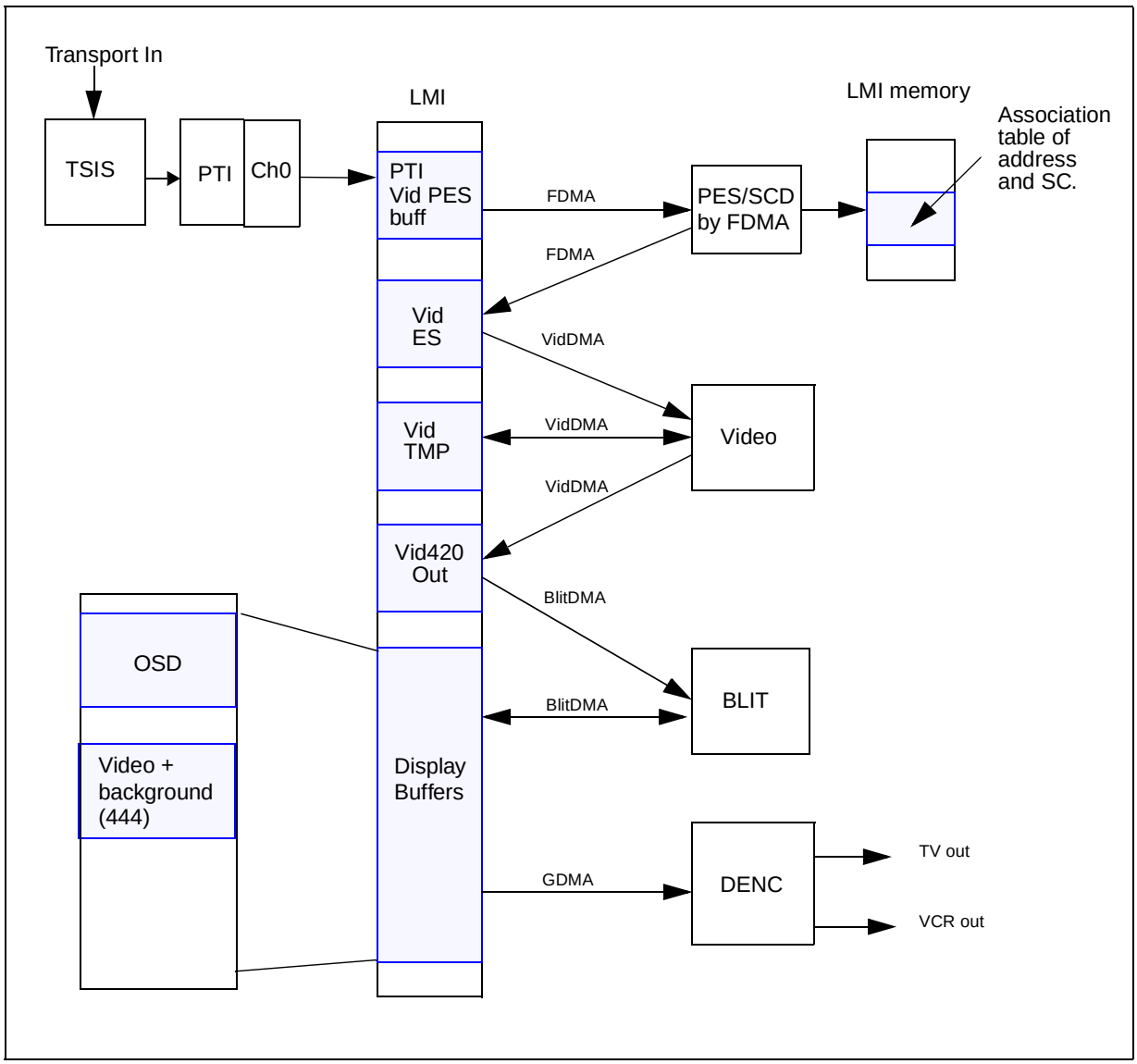


Figure 51 shows the video decode data flow.

Figure 51: Video decode flow



The PTI channel 0 circular video buffer is accessed by the FDMA which is a linear linked list machine. The synchronization is described in the following paragraph.

The host CPU is interrupted by a programmed timer event from system services, for example every 10 ms. The interrupt routine reads the ch0 write pointer from the PTI; using this, and the previous write pointer, a single pass linear access is activated using a dedicated process (PPSCD) running on the FDMA (with hardware assist) to perform PES parsing and start code detect.

The FDMA PPSCD process generates an elementary stream (ES) buffer in main memory and an association table which can be placed in external memory or local SRAM.

Each entry of the PPSCD table contains the following information:

- start code value,
- PTS if present,
- start code offset in PES buffer,
- start code address in the ES buffer.

This table is accessed by the video driver running on the host CPU. The ES video buffer is accessed directly using the linear DMA engine inside the video decoder. This block generates

several intermediate buffers during the prediction and reconstruction phases of decode before generating a final 420-Mbyte video buffer.

The video buffer is read directly by the blitter which performs multiple passes to format convert and blend the video buffer with a background and OSD overlay before rendering a final 4:2:2 display buffer.

The picture composed by the blitter is always frame based. The final presentation buffer is read by the GDMA which drives the digital encoder to produce component and composite video output. The GDMA should be programmed to access the presentation buffer field based.

22.3 Buffer organization

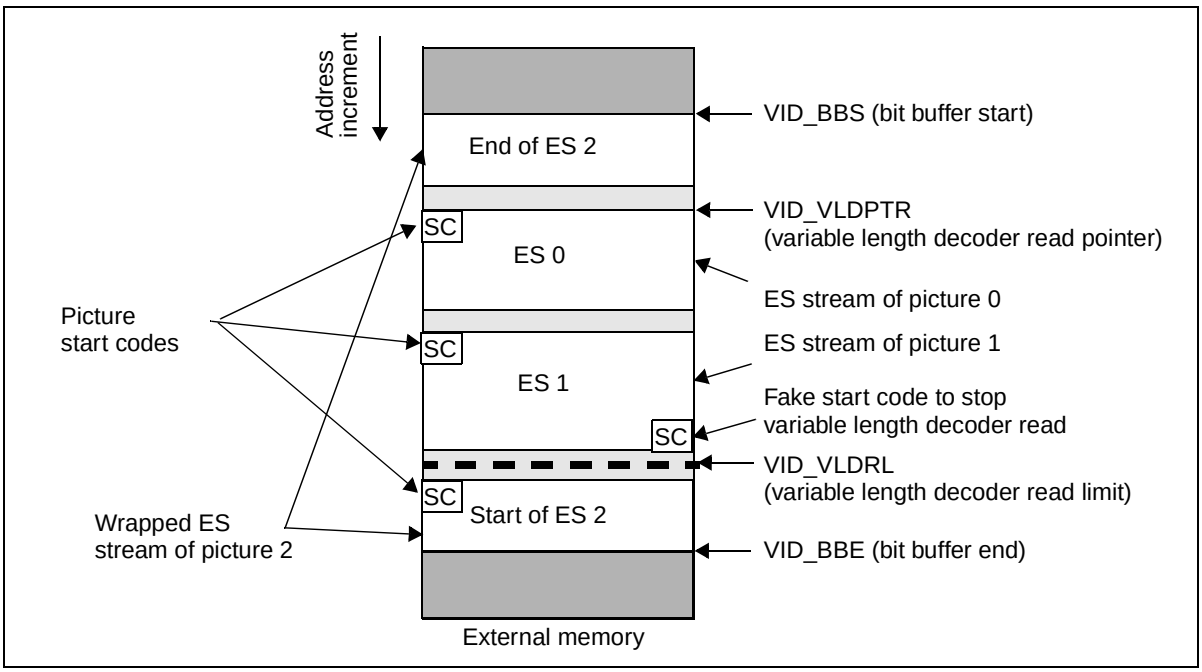
There are two types of buffer in the MPEG decoder:

- Bit buffer: area in external memory where the bitstream is stored,
- Frame buffer**:** area in external memory in which decoded pictures are stored.

22.3.1 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.

Figure 52: Bit buffer organization



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 52).

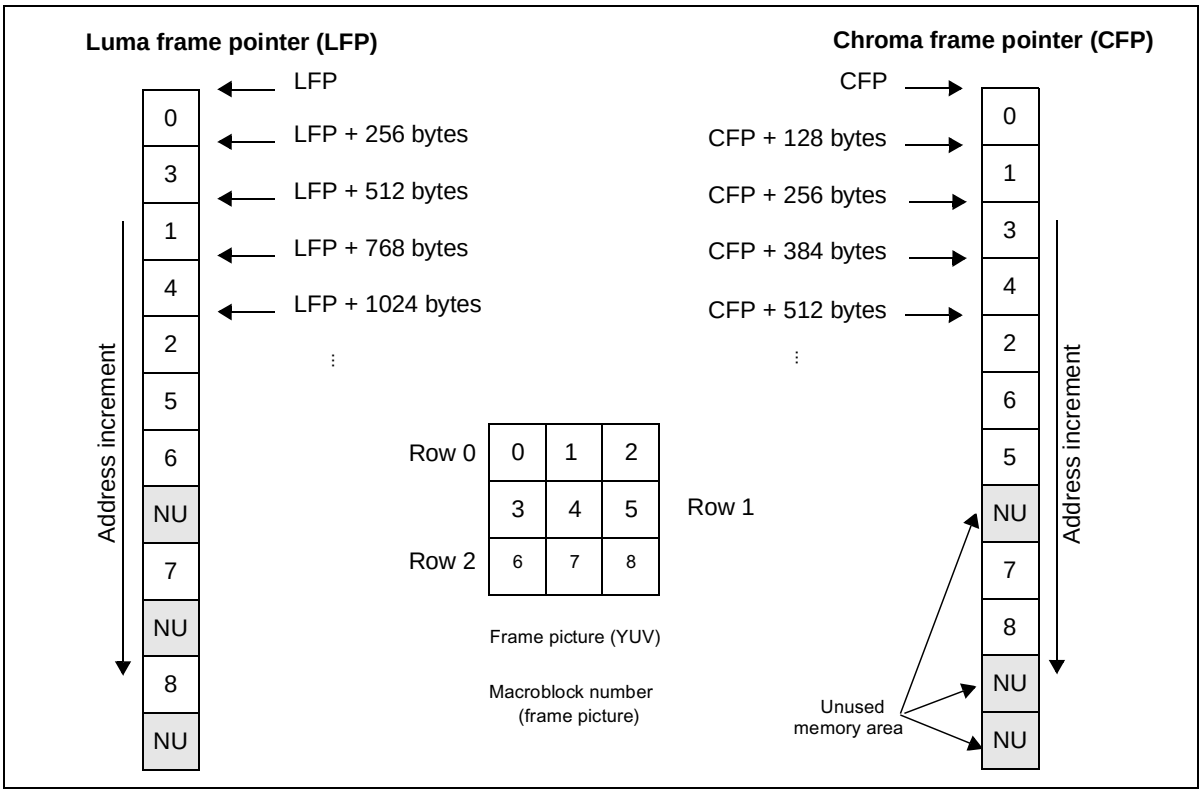
When a decode is launched, the variable length decoder reads data until it reaches a start code different from a slice start code. It can be the picture start code of the subsequent picture, or a fake start code added at the end of the picture to stop the variable length decoder. The variable length decoder read process can also be stopped with VID_VLDRL (see *Stream pointers for variable length decoder read access* on page 197).

22.3.2 Frame buffer organization

The video decoder stores both frame and field pictures in a buffer which corresponds to a frame buffer. 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 53 is an example of a frame buffer with an odd width and height (3 macroblocks x 3 macroblocks).

Figure 53: 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 * ((MBheight+1) DIV 2)*2 * 256
- ChromaBufferSize = ((MBwidth+1) DIV 2)*2 * ((MBheight+1) DIV 2)*2 * 128

Where DIV correspond to the integer part of a division:

x DIV y = integer part of (x/y)

A luminance macroblock contains16 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 54.

Chroma macroblock (Cb + Cr) data contains 8 words of 128 bits. The chroma macroblocks are stored in the same order as luma data (left part of top field, right part of top field, left part of bottom field, right part of bottom field) but a word of 128 bits contains 8 bytes of Cr data and 8 bytes of Cb data. A description is given in Figure 55.

Figure 54: Y macroblock storage

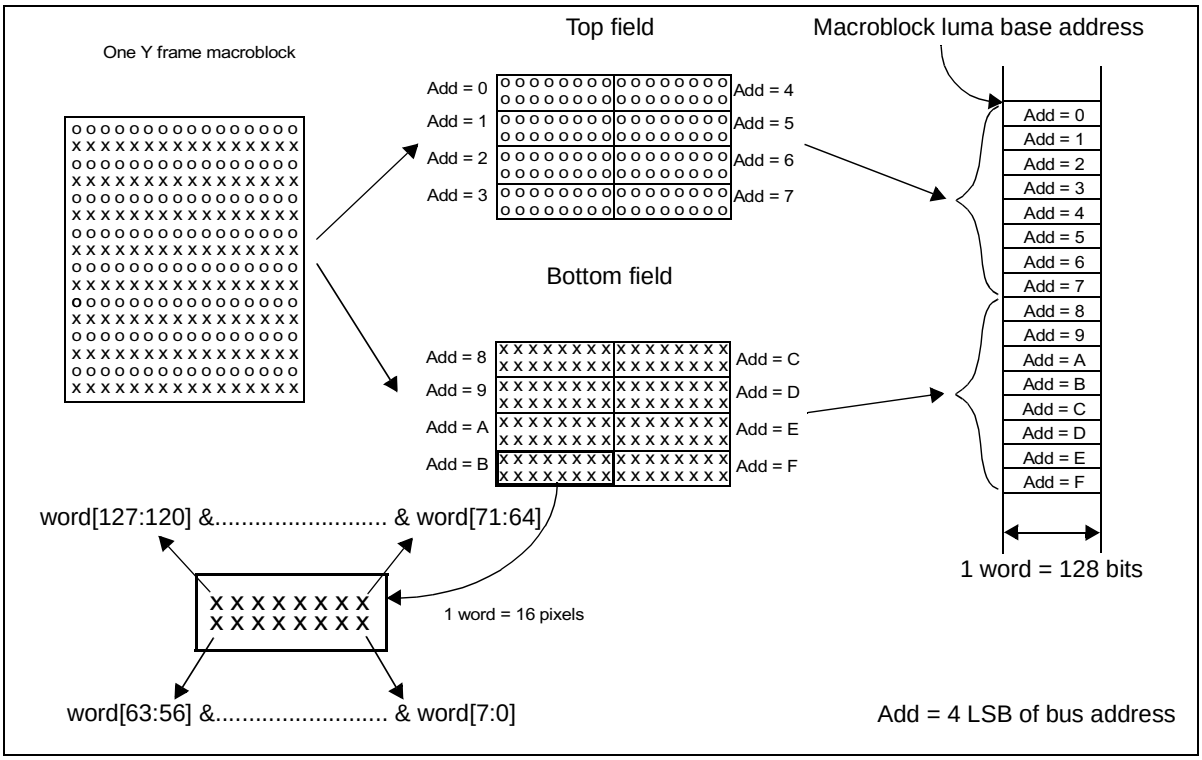
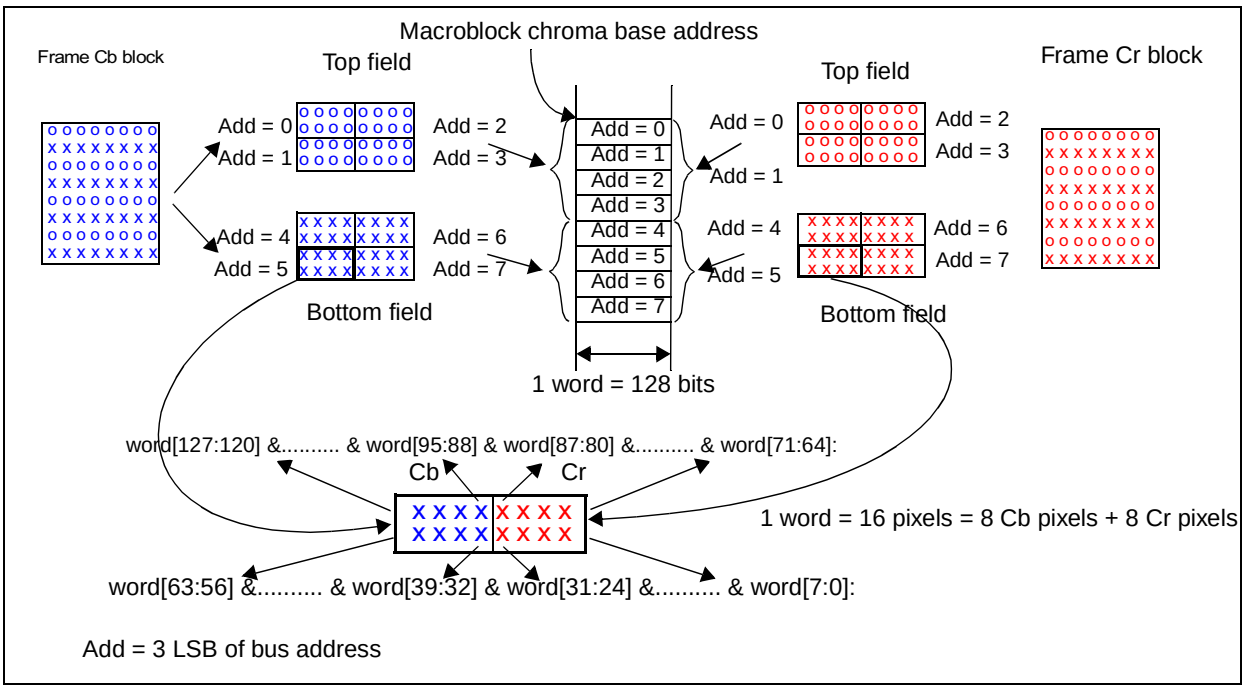


Figure 55: Cr/Cb macroblock storage



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*

22.3.3 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 PAL pictures (720 x 576 pixels), is 5.014 Mbits (3.318 Mbits for luma frame buffer and 1.696 Mbits for chroma frame buffer, see Section 22.3.2 for details).

A minimum of three frame buffers is required to decode field pictures. A minimum of four is needed for frame pictures.

For trick modes (slow, normal or fast, backward or forward play), the number of frame buffers is much higher, depends on the speed of the decoder and the required fluidity quality.

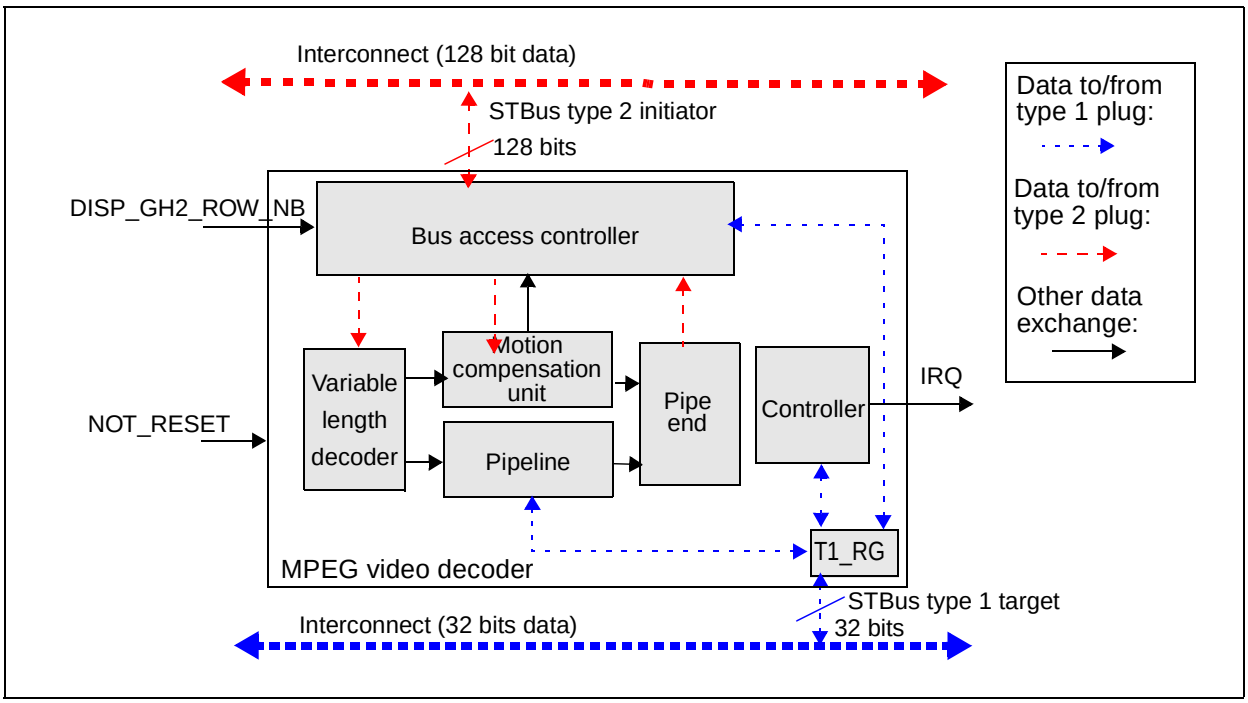
Bit buffer

The memory need in term 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.

22.4 Video decoding tasks

A task is the decoding of a single picture. It is specified by the task instruction set up before the decoding of each picture in register VIDn_TIS.

Figure 56: MPEG video decoder block diagram



22.4.1 Controller

The control of decoding tasks is managed by the controller block. The controller block:

- holds some debug registers: VID_DBG2, VID_VLDS, VID_VMBN, VID_MBE0, VID_MBE1, VID_MBE2 and VID_MBE3,
- 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.

22.4.2 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:

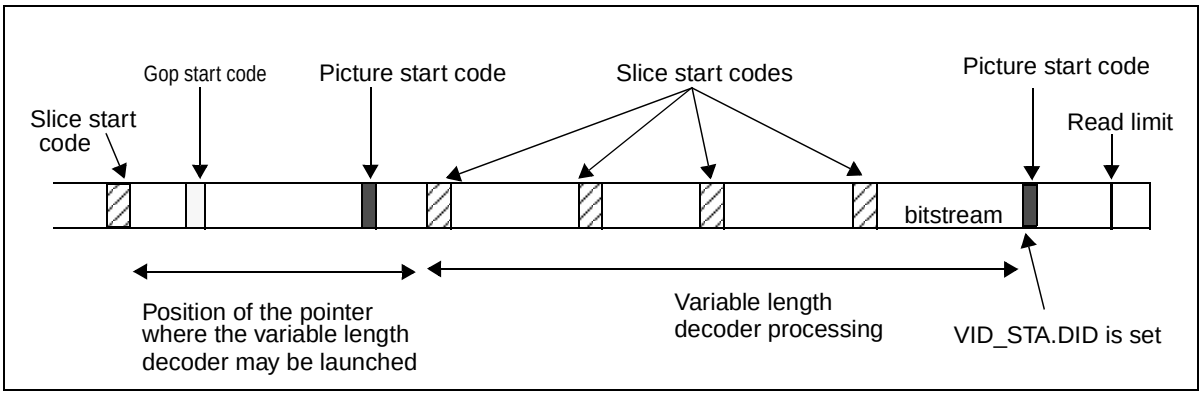
- variable length decodes the slice layer, macroblock layer and block layer,
- decodes the macroblock address increment and process skipped macroblock,
- decodes the motion vectors and delivers them to the motion compensation unit,
- detects bitstream errors and conceals corrupted and missing data (see Section 22.5.4: *Error recovery* on page 198),
- provides the number and the location of corrupted macroblocks in the picture (see Section 22.5.5: *Error statistics* on page 201).

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 LSB 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 57).

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 57: 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).

22.4.3 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 Macroblock structure in MPEG 2 standard, ISO 13818-2). 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).

22.4.4 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.

22.4.5 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.

22.4.6 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. The CPU is interrupted with bit ROPC in VID_STA if R_OPC flags an error during a transaction.

22.4.7 Register accesses

The T1RG interface is an STBus target and implements a 32-bit data interface for registers. Registers are also implemented on 32-bit width data. This interface uses little endian mode.

If a nonsupported opcode or a nonexistent address is presented to this interface, then a response packet is generated containing one response cell for each request cell received. The R_OPC is asserted to flag the error and the R_DATA returned by the interface is “dummy”. The CPU is interrupted with bit R_OPC (VID_STA). The MPEG video decoder does not check inconsistent operations between the opcode and the byte enable. It is the byte enable which is taken into account prior to the opcode.

Table 66: Supported opcode for T1_RG

Plug	Load	Store
T1_RG	LD4	ST4

22.5 Video decoding

22.5.1 Control of decoding

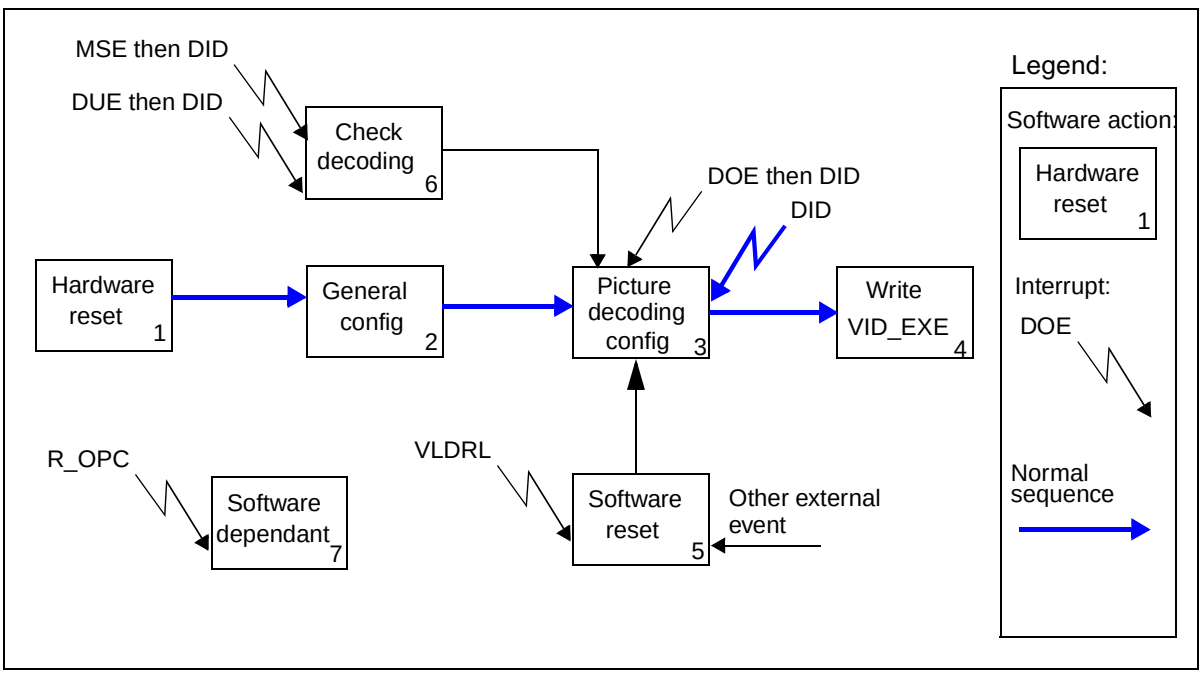
The control is performed by the CPU. The CPU launches the decoder one picture at a time.

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 196),
3. Picture decoding configuration (see *Picture decoding configuration* on page 196),
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 23: *MPEG video decoder registers* on page 203 for more details on interrupts).

Figure 58: 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_MBEx) 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.

22.5.2 General configuration

The general configuration consists of CFG_VIDIC (video decoder interconnect configuration) and VID_ITM (interrupt mask).

22.5.3 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. The whole number of macroblocks in the frame picture is held in VID_DFS.

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. 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 decoding).

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: 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.

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.

22.5.4 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. The Table 67 describes errors that are detected by the MPEG video decoder. The way to conceal is described after the table.

Table 67: Error detection

Layer	Description	Concealment	MBE[1]
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 1* on page 199	Yes
	`macroblock_type`	See *Procedure 1* on page 199	Yes
	`frame_motion_type` and `field_motion_type`: the value 00 is reserved	See *Procedure 1* on page 199	Yes
	`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 1* on page 199	Yes
	`coded_block_pattern_420`	See *Procedure 1* on page 199	Yes
	`end_of_macroblock` for D pictures: it shall be equal to 1.	See *Procedure 1* on page 199	Yes
	`motion_code[r][s][t]`	See *Procedure 1* on page 199	Yes
Block	`first_dct_coef`	See *Procedure 1* on page 199	Yes
	`next_dct_coef`	See *Procedure 1* on page 199	Yes

Table 67: Error detection

Layer	Description	Concealment	MBE[1]
Semantic errors: 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 1* on page 199	Yes
Slice	Slice start code combined with macroblock address increment gives a macroblock number higher than the macroblock number to be decoded.	See *Procedure 2* on page 200	Yes
Macroblock	**mb_addr_increment** may result in a macroblock address greater than the picture size. The macroblock is outside the picture.	See *Procedure 2* on page 200	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 1* on page 199	Yes

1. When MBE is yes, the error is taken into account in VID_MBEn registers. When no, the error is not taken into account.

Procedure 1

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 68 shows the rules used to fetch concealment macroblocks. Skipped macroblocks are not mentioned since they always refer to one of the prediction types listed.

Table 68: 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 2

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. Underflow errors the underflow flag in the status register is always set.

Overflow or underflow error

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 DOE in (VID_STA).

An underflow error occurs when the variable length decoder has found in the bitstream less 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) or to VID_VMBN if RMM = 0 (missing macroblocks are not concealed).

22.5.5 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_MBEn (macroblock errors) registers. The VID_MBEn 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 MSE in VID_STA is set. If no error occurred, this bit is 0.

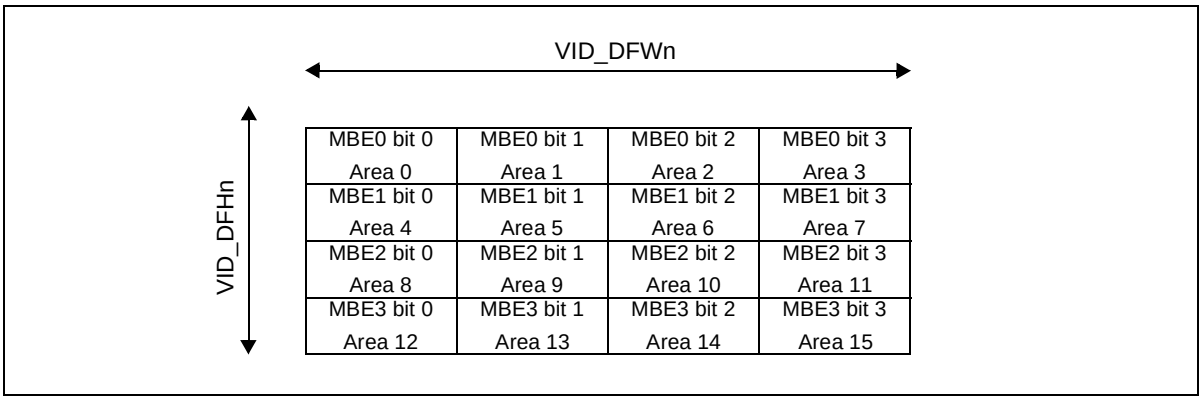
The decoded picture is divided into 16 areas defined by four horizontal and four vertical slides. The areas are numbered from 0 (top left area) to 15 (bottom right). This number is incremented from left to right, top to bottom (Figure 59). For each area, VID_MBEn gives the number of macroblocks reconstructed with error concealment.

If the picture size in macroblock units is a multiple of 4 in both directions, each of the 16 areas contains the same number of macroblocks DFS>>4. If it is not a multiple of 4 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 2.

Figure 59: Macroblock error statistics



22.5.6 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 629.5 Mbit/s, but for simplified B this is equivalent to a P-picture at 384.5 Mbits/s.

This feature has an impact on picture quality, but has no effect on the P dual prime picture.

22.6 Resets

22.6.1 Hardware reset

After a hardware reset, the MPEG video decoder is in an idle state. No processing is done and no request is sent on Type 2 initiator until video decoding is launched.

22.6.2 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,
  - debug registers content: VID_MBNM, VID_VMBN and VID_VLDS (VID_VLDS set to 0x0000 00FF),

  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.

23 MPEG video decoder registers

All registers are reset by HW. HW means hardware reset. Read only registers can also be affected by SRS and EXE, see Section 22.6.2: *Software reset* on page 202 for details.

When registers are read, the value of the reserved bits are 0 unless specified. No register is buffered: write action effect is immediate.

Addresses are provided as the *MPEGBaseAddress* + offset.

The *MPEGBaseAddress* is:

0x2050 0000.

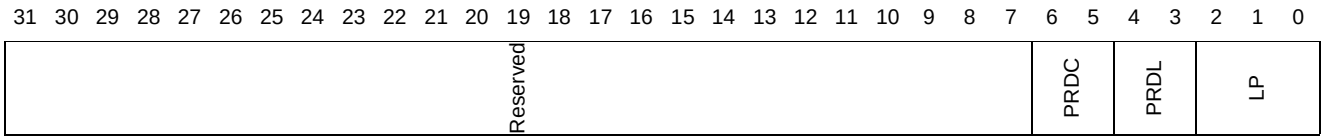
Table 69: MPEG video decoder registers

Register name	Description	Address offset	Type
VID_EXE	Execute decoding task, see page 206	0x0008	WO
CFG_VIDIC	Video decoder interconnect configuration, see page 204	0x0010	R/W
VID_MBE0	Macroblock error statistic 0, see page 207	0x0070	RO
VID_MBE1	Macroblock error statistic 1, see page 207	0x0074	RO
VID_MBE2	Macroblock error statistic 2, see page 207	0x0078	RO
VID_MBE3	Macroblock error statistic 3, see page 207	0x007C	RO
VID_QMWIp	Quantization matrix data, intra table, see page 209	0x0100 to 0x013F	R/W
VID_QMWNIp	Quantization matrix data, nonintra table, see page 209	0x0180 to 0x01BF	R/W
VID_TIS	Task instruction, see page 211	0x0300	R/W
VID_PPR	Picture parameters, see page 208	0x0304	R/W
VID_SRS	Decoding soft reset, see page 210	0x030C	WO
VID_ITM	Interrupt mask, see page 207	0x03F0	R/W
VID_ITS	Interrupt status, see page 208	0x03F4	RO
VID_STA	Status register, see page 211	0x03F8	RO
VID_DFH	Decoded frame height, see page 205	0x0400	R/W
VID_DFS	Decoded frame size, see page 206	0x0404	R/W
VID_DFW	Decode frame width, see page 206	0x0408	R/W
VID_BBS	Video elementary stream bit buffer start, see page 205	0x040C	R/W
VID_BBE	Video elementary stream bit buffer end, see page 204	0x0414	R/W
VID_VLDRL	Variable length decoder read limit, see page 212	0x0448	R/W
VID_VLDPTR	Variable length decoder read pointer, see page 212	0x045C	R/W
VID_BCHP	Backward chroma frame buffer, see page 205	0x0488	R/W
VID_BFP	Backward luma frame pointer, see page 205	0x048C	R/W
VID_FCHP	Forward chroma frame buffer, see page 206	0x0490	R/W

Table 69: MPEG video decoder registers

Register name	Description	Address offset	Type
VID_FFP	Forward luma frame pointer, see page 207	0x0494	R/W
VID_RCHP	Main reconstructed chroma frame pointer, see page 210	0x0498	R/W
VID_RFP	Main reconstructed luma frame pointer, see page 210	0x049C	R/W

CFG_VIDIC Video decoder interconnect configuration



Address: *MPEGBaseAddress* + 0x0010

Type: Read/write

Reset: 0

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.

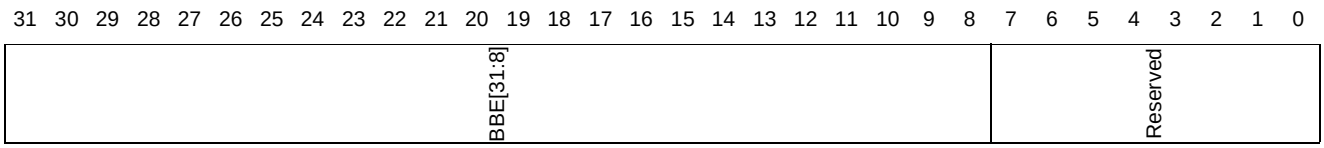
[31:0] **Reserved**

[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 = 8 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_BBE Video elementary stream bit buffer end



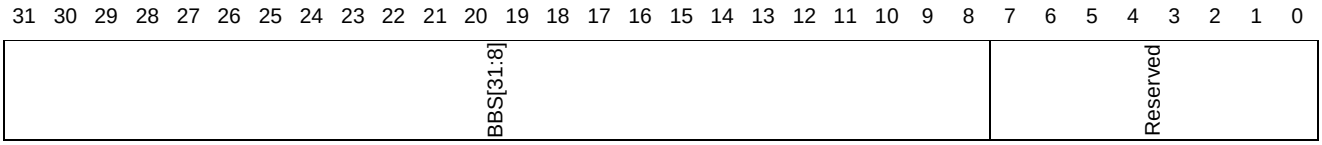
Address: *MPEGBaseAddress* + 0x0414

Type: Read/write

Reset: 0

Description: Memory address of the video elementary stream bit buffer end, defined in units of 256 bytes.

VID_BBS Video elementary stream bit buffer start



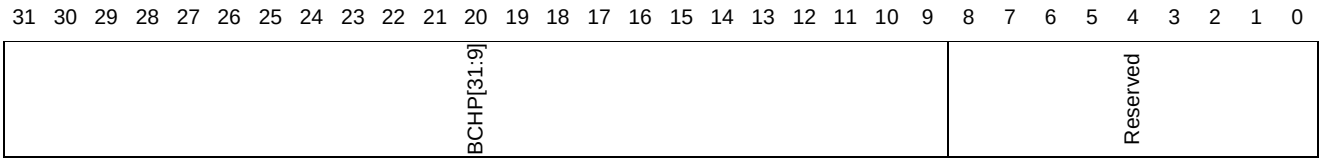
Address: *MPEGBaseAddress* + 0x040C

Type: Read/write

Reset: 0

Description: Memory address of the video elementary stream bit buffer start, defined in units of 256 bytes.

VID_BCHP Backward chroma frame buffer



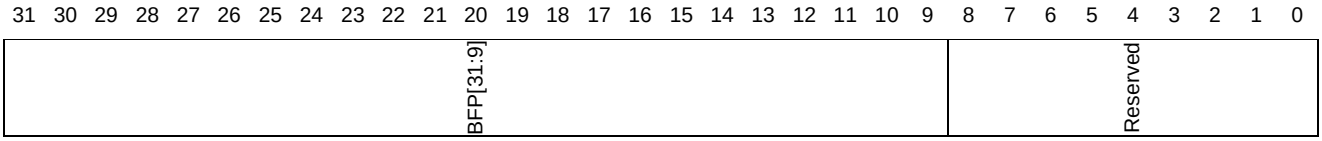
Address: *MPEGBaseAddress* + 0x0488

Type: Read/write

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



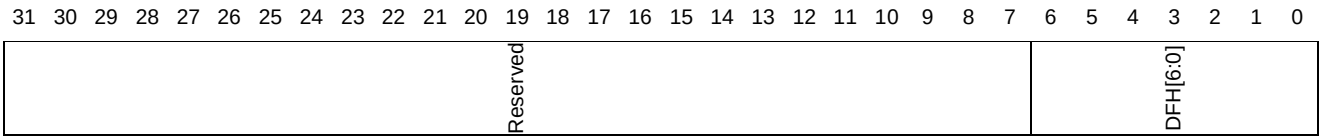
Address: *MPEGBaseAddress* + 0x048C

Type: Read/write

Reset: 0

Description: This register holds the start address of the luma backward prediction frame picture buffer, defined in units of 512 bytes.

VID_DFH Decoded frame height



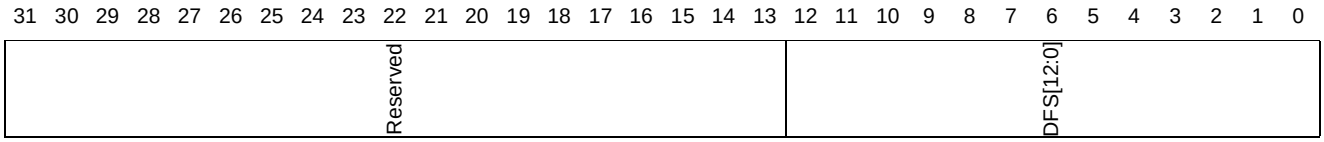
Address: *MPEGBaseAddress* + 0x0400

Type: Read/write

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



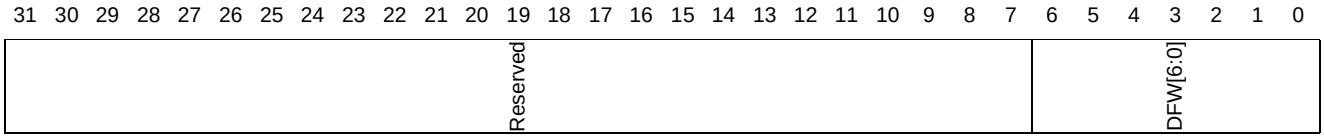
Address: *MPEGBaseAddress* + 0x0404

Type: Read/write

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



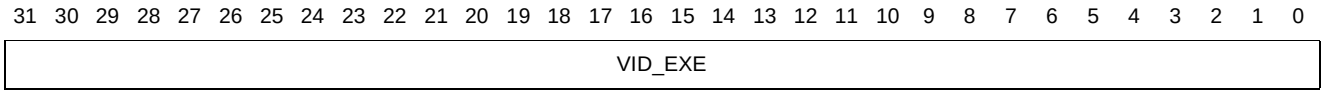
Address: *MPEGBaseAddress* + 0x0408

Type: Read/write

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_EXE Execute decoding task



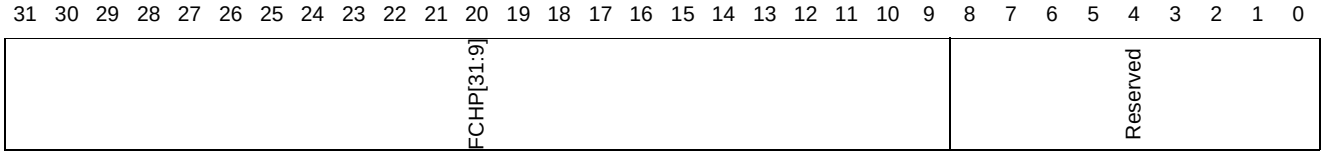
Address: *MPEGBaseAddress* + 0x0008

Type: Write only

Reset: Undefined

Description: Writing to the least significant byte of this register starts a decoding task.

VID_FCHP Forward chroma frame buffer



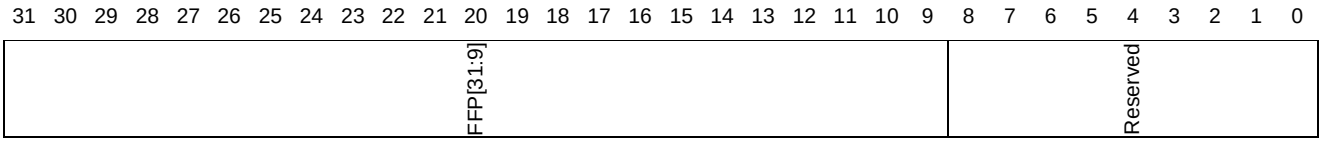
Address: *MPEGBaseAddress* + 0x0490

Type: Read/write

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



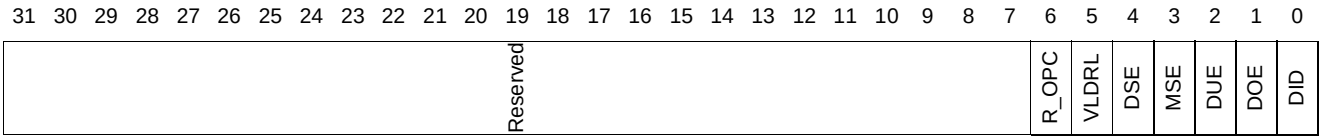
Address: *MPEGBaseAddress* + 0x0494

Type: Read/write

Reset: 0

Description: This register holds the start address of the luma forward prediction frame picture buffer, defined in units of 512 bytes.

VID_ITM Interrupt mask



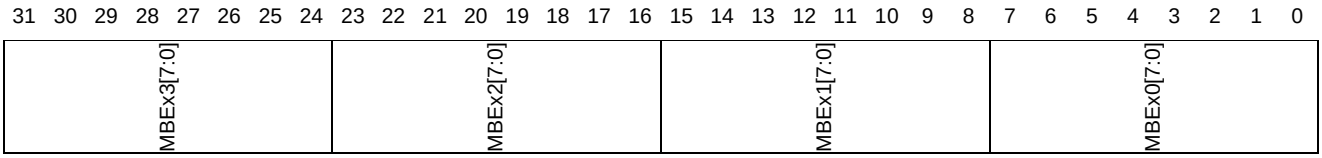
Address: *MPEGBaseAddress* + 0x03F0

Type: Read/write

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_MBEn Macroblock error statistic (n = 0 to 3)



Address: *MPEGBaseAddress* + 0x0070 (VID_MBE0), + 0x0074 (VID_MBE1), 0x0078 (VID_MBE2), + 0x007C (VID_MBE3)

Type: Read only

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_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																									R_OPC	VLDRL	DSE	MSE	DUE	DOE	DID

Address: *MPEGBaseAddress* + 0x03F4

Type: Read only

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_PPR Picture parameters

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MP2	FFN		TFF	FRM	CMV	QST	IVF	AZZ	PCT		DCP		PST		BFV				FFV				BFH				FFH			

Address: *MPEGBaseAddress* + 0x0304

Type: Read/write

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 that 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_QMWIp Quantization matrix data, intra table

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

Address: *MPEGBaseAddress* + 0x0100 to 0x013F

Type: Read/write

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, nonintra table

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

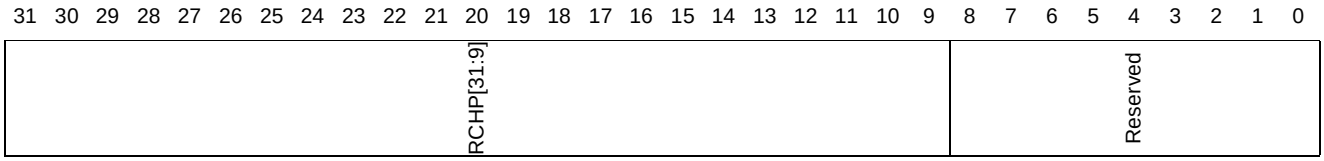
Address: *MPEGBaseAddress* +0x0180 to 0x01BF

Type: Read/write

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_RCHP Main reconstructed chroma frame pointer



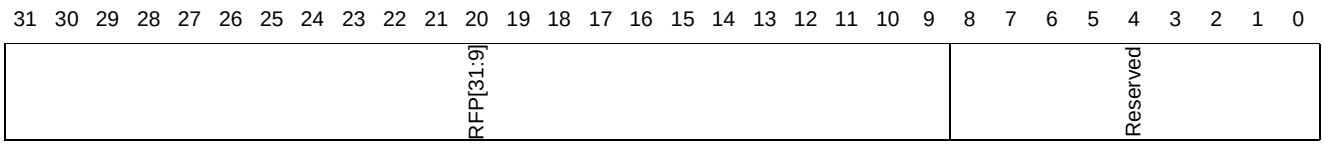
Address: *MPEGBaseAddress* + 0x0498

Type: Read/write

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



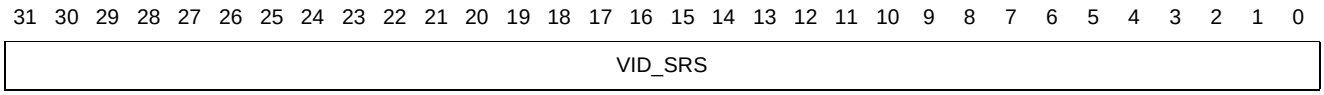
Address: *MPEGBaseAddress* + 0x049C

Type: Read/write

Reset: 0

Description: This register holds the start address of the reconstructed (decoded) luma picture buffer, defined in units of 512 bytes.

VID_SRS Decoding soft reset



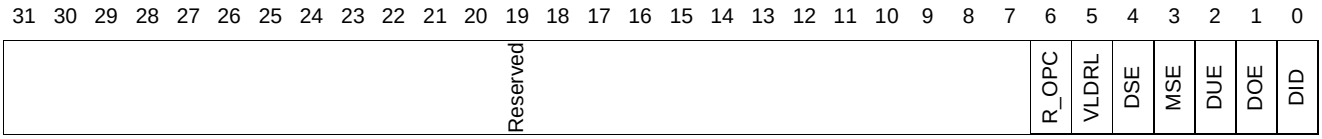
Address: *MPEGBaseAddress* + 0x030C

Type: Write only

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_STA Status register



Address: *MPEGBaseAddress* +0x03F8

Type: Read only

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.

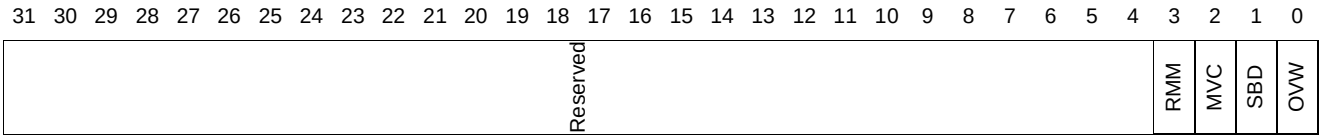
[3] **MSE:** set to 1 at the end of a decoding task if a semantic or a syntax error has been detected by the variable length decoder during the decoding process of the picture. It is reset on a write action on VID_EXE or VID_SRS LS byte.

[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_TIS Task instruction



Address: *MPEGBaseAddress* +0x0300

Type: Read/write

Reset: 0

Description: 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 67: *Error detection* on page 198).

[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_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: *MPEGBaseAddress* + 0x045C

Type: Read/write

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_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: *MPEGBaseAddress* + 0x0448

Type: Read/write

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 128 bytes unit.

24 2D blitter display engine

The 2D blitter display engine is a software controlled output display generator that can also be used as a CPU accelerator for graphics picture handling. It is a triple-source 2D DMA, with a set of powerful operators.

The 2D blitter display engine retrieves data from the local memory through three input sources, Source 1, Source 2 and Source 3. Sources 1, 2 and 3 are used simultaneously for read/modify/write operations.

- Source 1 is used for frequent operations such as color-fill or simple source copy. It performs according to the pixel format.
- Source 2 is a general-purpose input to which all operators apply.
- Source 3 is an additional input working concurrently with source 2 to retrieve macroblock formats from memory in a single node.

The processing pipeline is always in ARGB8888 or AYCbCr8888 format whatever the format of the source inputs.

The 2D blitter display engine is software controlled using multiple queues which have a link-list structure. Each node of the link list is an instruction that contains all the necessary information to proceed. The 2D blitter display engine, handles both high and low priority queues, known as composition and application queues respectively. The composition queue is real-time constrained (that is it must be finished during a frame), while application queues are executed in the remaining time.

The 2D graphics processor operates at up to ***F*** = 200 MHz clock rate. For each operation involving source 2, the maximum output rate is ***F*** Mpixel/s, whatever the pixel depth. This rate is constant except for 2D resizing for downsampling, where it is ***F*** 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 8*__F__ Mbytes/s.

These values are the maximum performances that the 2D blitter display engine can reach. They suppose that no read/write memory word request has been postponed by the memory arbiter mechanism.

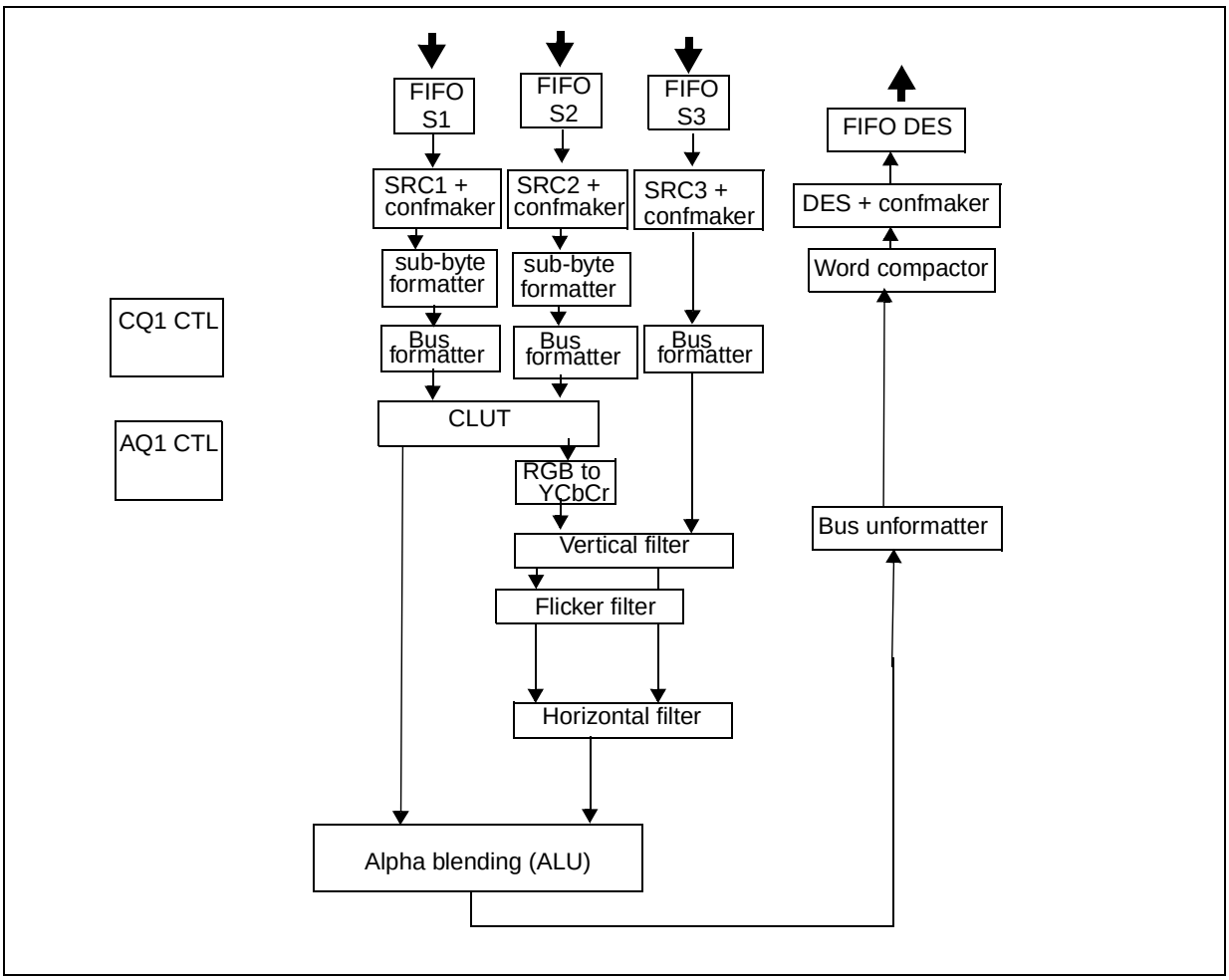
The macroblock format can be used as a source for the 2D graphics engine. Because these formats uses both the source 2 and source 3 buffers, special modes are required. These modes are described in Section 24.7.5: *Using the 4:2:x macroblock-based plane as a source* on page 239.

24.1 2D blitter display engine functions

- 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.
- 4:2:2 / 4:20 capabilities, as source format.
- Color space conversion RGB to YCbCr (only 601 matrix).
- Color expansion (CLUT to true color).
- Color correction (gamma, contrast, gain).
- 2D resize engine with high quality filtering.
- Fixed coefficient flicker filter from memory to memory.
- Rectangular clipping.
- Programmable source/target scanning direction, both horizontally and vertically, in order to cope correctly with overlapping source and destination area.

The blitter display configuration for the STx5119 is given in Figure 60.

Figure 60: Blitter display



The 2D blitter display engine works from memory to memory with a triple, dual or single source and one target. The target can be one of the sources. A set of overlaps between sources and target are supported, on condition 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. One register (BLT_INS) is used to control the flow of data with operator enables. Each time an operator is enabled, its behavior must be specified with operator specific registers. If the blitter display is not in direct mode (64-bit internal bus), the ALU operator is always enabled and must be programmed.

24.2 Recommended scheme

For best results and to perform clean resizing and filtering on graphics objects, STMicroelectronics recommends processing graphics planes in progressive mode, and processing video input according to whether it is encoded as field- or frame-based data.

This means that a frame buffer (a buffer containing a progressive picture or top and bottom fields interlaced, even if in an interlaced display) is always generated. The GDMA then accesses this buffer with a pitch of 1 or 2 lines depending on whether the display scan style is progressive or interlaced.

Figure 61: Recommended scheme: field-based video data

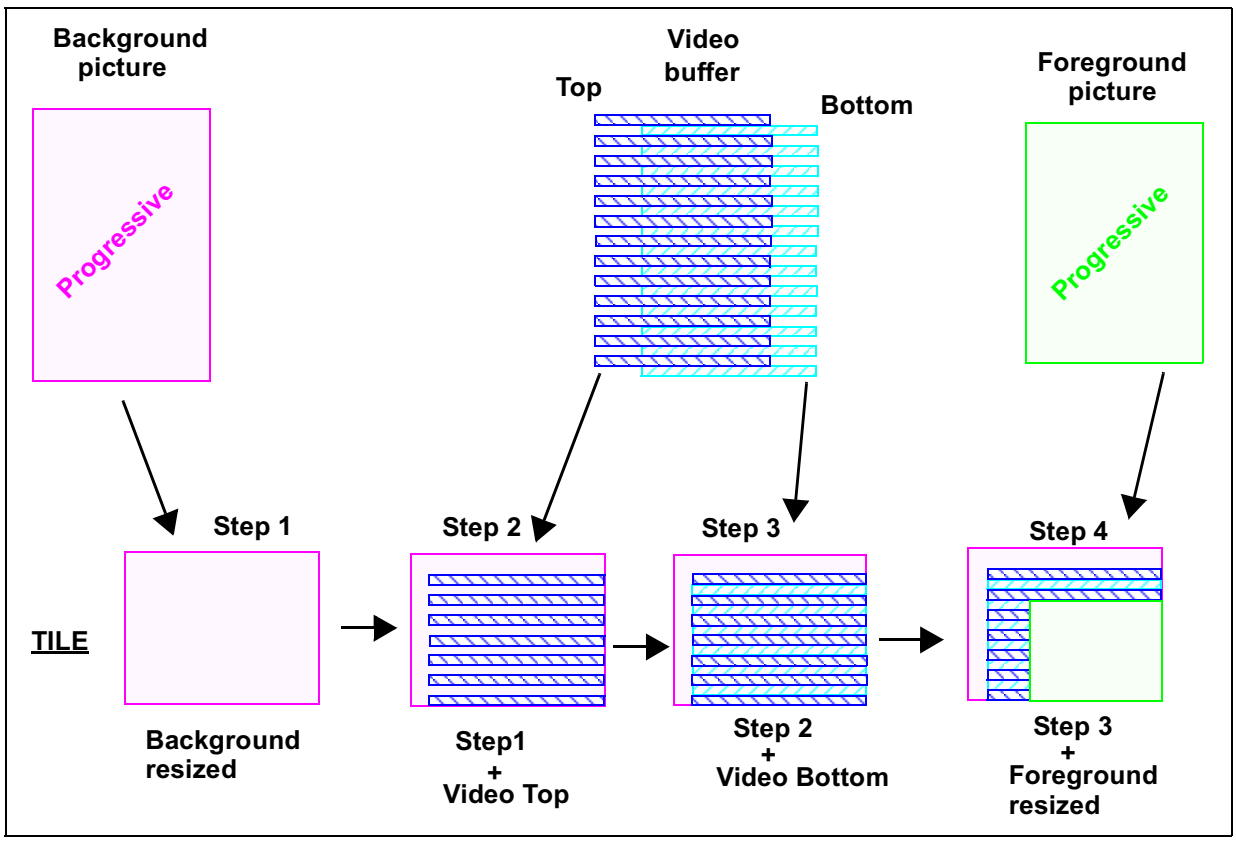
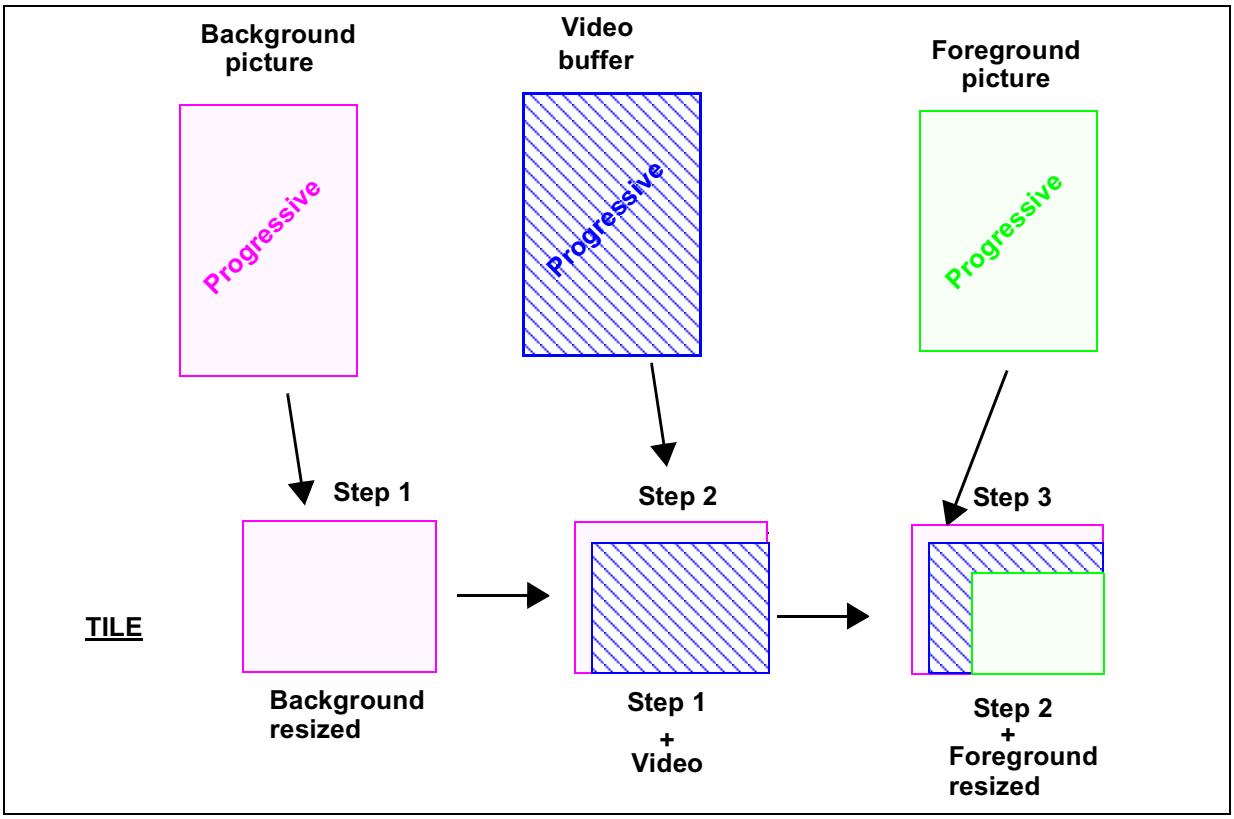


Figure 62: Recommended scheme: frame-based video data



24.3 Types of source queue

24.3.1 Composition queue (CQ)

The composition queue generates the output display buffer. This queue is high priority. As it is mandatory to render one new frame or field within one frame or field time and to avoid errors on the display, no delay is tolerated. The queue needs to be started on an event (VSYNC, line number, CPU go).

Triggering

Usually, the composition is launched at a specific event. There are three ways of launching a composition queue.

- **VSYNC**
  When the 2D blitter display engine is triggered on a VSYNC, it is assumed that the composition linked list was generated during the last frame or field. This provides the maximum amount of time (a full frame or field) to generate (render) the output display buffer
  VSYNC triggering has a latency between decoding and displaying a picture. In Figure 63 the system has a two frame latency between decode and display.
- **Line triggering**
  Line triggering is the same as VSYNC triggering, except it happens when the VTG line counter reaches the value described in BLT_CQ_TRIG_CTL.TRIG_LINE_NUM. This avoids having another subscriber to the VSYNC event.
  In this mode, the system also has a two field or frame latency between decode and display.
  A line trigger on the first line is not allowed, as it is the same as a VSYNC trigger.

- **Soft triggering**

  In soft triggering the 2D blitter display engine is launched at the most appropriate time by writing to BLT_CQ_TRIG_CTL.TRIG_COND. This is useful where bandwidth is sufficient and a reduced latency is required.

  In this configuration, the system has a latency of one field or frame between decode and display, but it introduces more constraints on the composition queue execution time.

Figure 63: VSYNC triggering

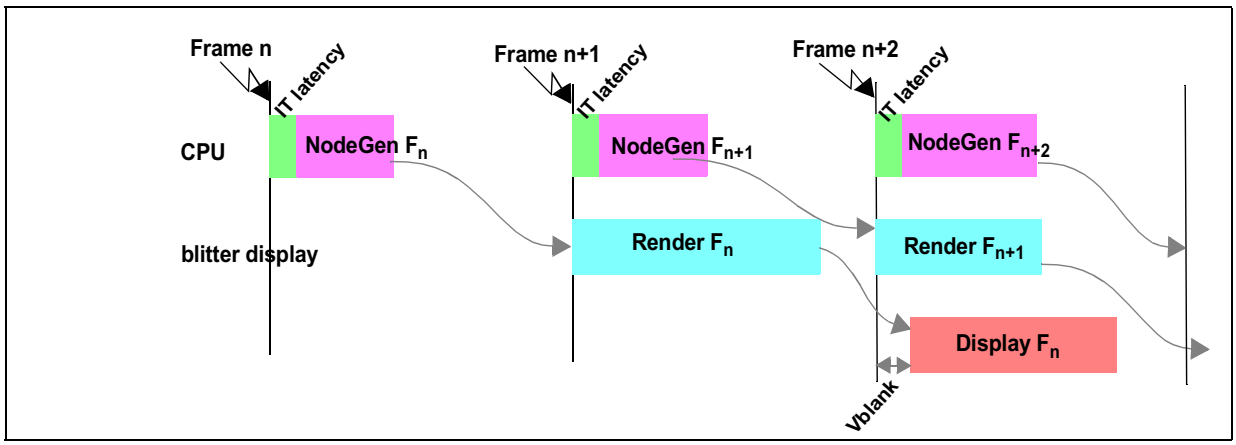


Figure 64: Line number triggering

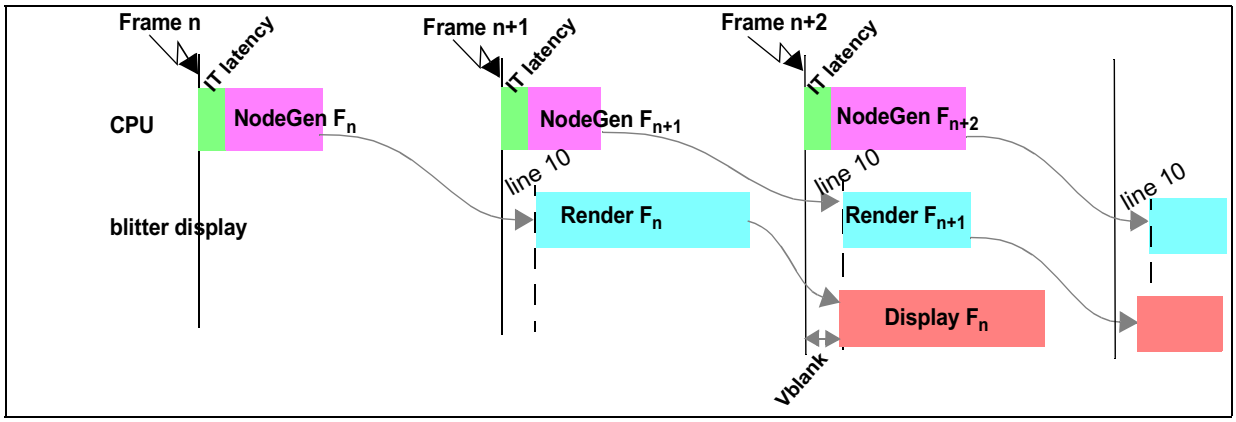
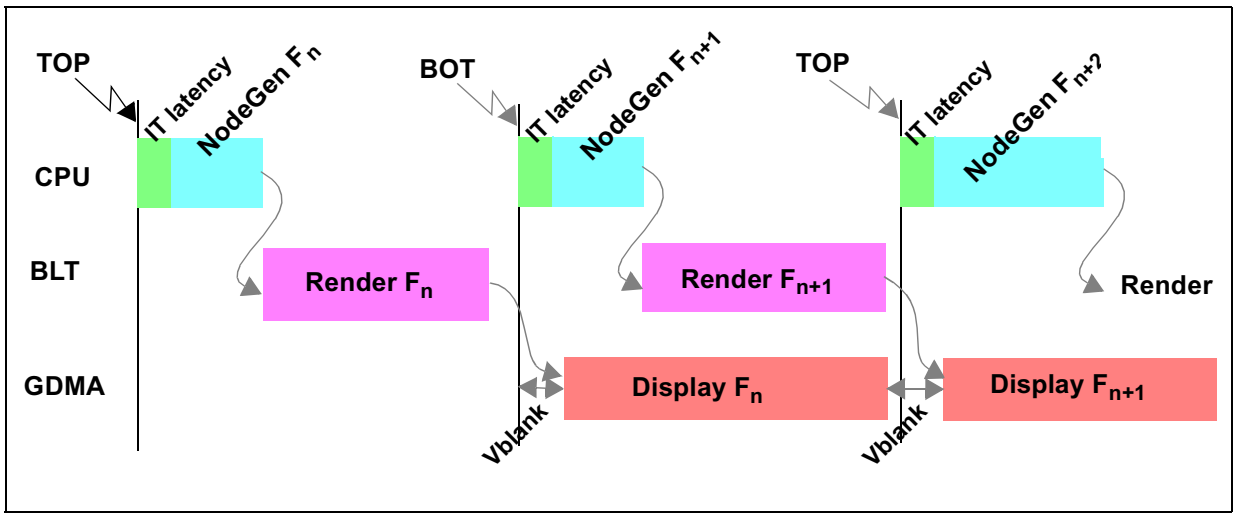


Figure 65: Soft triggering

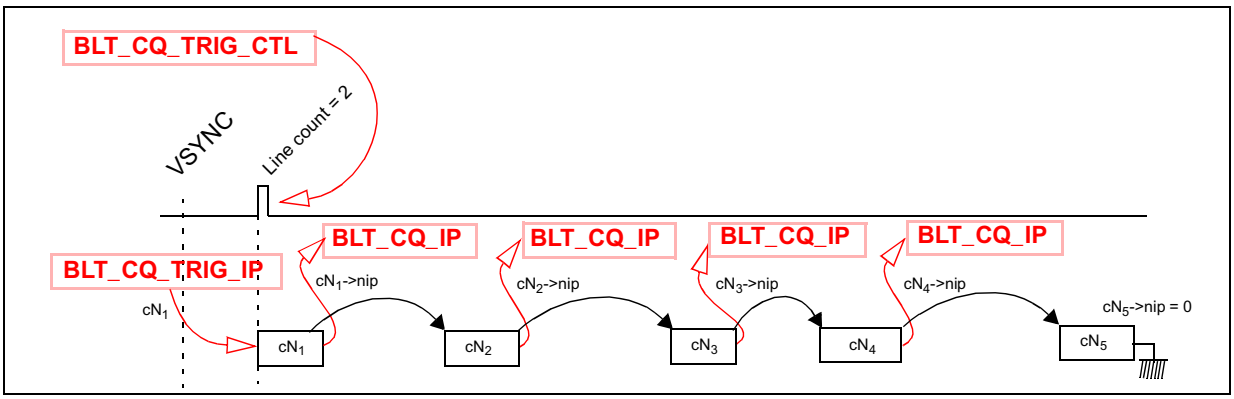


Control

The composition queue is launched when reaching the trigger event (as described in previous section, using the BLT_CQ_TRIG_CTL.TRIG_COND field).

Once this event is met, the blitter fetches the node stored at the BLT_CQ_TRIG_IP address, and starts executing this node and the linked ones. Each time a node is executed, the BLT_NIP register (provided in the node registers) is stored in the BLT_CQ_IP register.

Figure 66: Standard composition queue flow



The BLT_CQ_TRIG_IP register provides the absolute start of the queue. It is used only once in a field or frame when the trigger condition is reached. After the trigger, it can be modified any time the composition queue is executing.

Once the blitter reaches NULL BLT_NIP, the composition queue is completed. This queue is then idle until the next triggering event.

Interrupts

There are four interrupt conditions for each composition queue and interrupt status bits are provided in the BLT_ITS register. The CQ associated interrupt line is asserted until the corresponding ITS status bit is cleared by writing 1. See *BLT_ITS* on page 248. All interrupts are maskable through the BLT_CQ_TRIG_CTL.IRQ_MASK field.

- **Queue completed**

  The queue completed interrupt notifies that the display has finished generating and the blitter is now available to process other queues.

- **Node notification**

  The node notification interrupt is node based and notifies the level or location of processing within a composition queue. The blitter issues the IRQ when the node has finished executing, but continues processing the queue. The last node that issued the IRQ is shown in the BLT_CQ_IP register.

  To issue this IRQ, the blitter must have the BLT_CQ_TRIG_CTL.IRQ_MASK field and the node register (BLT_INS.BLITCOMPIRQ) enabled.

- **Queue retriggered**

  The queue retriggered interrupt occurs when the blitter did not generate the output display (the full composition queue) within the requested time.

  If this occurs the actions detailed in the Exception handling section should be taken.

Exception handling

- **Finish current queue**
  The blitter executes all the nodes of the queue sequentially, until it reaches the last node, and then starts executing the next composition queue (pointed by BLT_CQ_TRIG_IP).
  This assumes that during the time needed for a single composition, the blitter is able to finish both the previous and the current composition. If this recovery succeeds, no errors are seen on the display, but it could also lead to a significant amount of unprocessed data for each field or frame.
- **Suspend current queue**
  The blitter executes the current node until it completes, and then jumps to the start of the next composition queue by loading BLT_CQ_TRIG_IP. This allows for a clean drop and restart. It assumes that the retrigger condition will not be reproduced in the next field or frame.
- **Abort current queue**
  The current node is executed until the end of the current line, and then jumps to the start of the next composition queue by loading the BLT_CQ_TRIG_IP. This behavior is the same as the previous one, except, it stops as soon as possible to jump to the next queue.
- **Stop current queue**
  The blitter executes the current node until the end of the current line, and then disables the queue (that is it returns to idle if no other queue is pending). An action is needed to relaunch the composition queue.

24.3.2 Application queues (AQ)

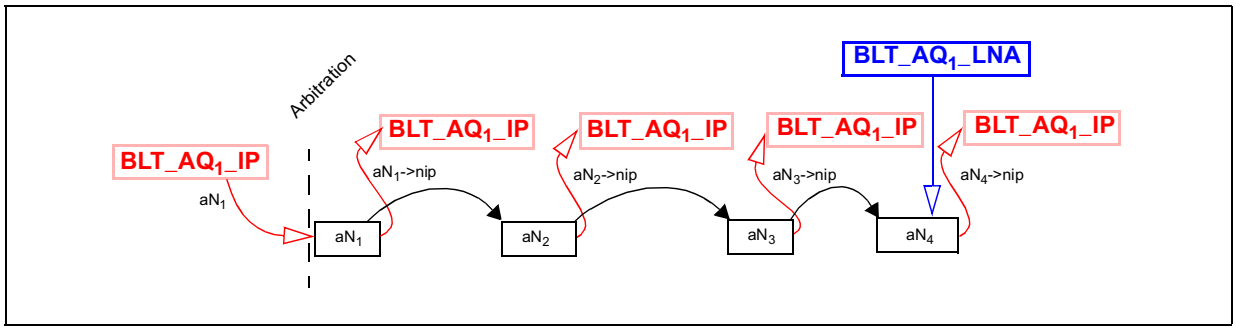
Application queues allow the module to be used as a blitter. These queues are of low priority when compared to the composition queue as the nodes are not real-time. Application queues are slave of composition queue execution and are arbitrated together, with programmable priorities.

Control

An application queue is launched whenever the composition queue is idle. This means that in any time slot not used for composition, an application queue may be launched between VSYNC and line trigger in pace down time and between composition completion and the next trigger event.

The application queue is controlled using two pointers: the first node (BLT_AQ_IP register) and last node pointer (BLT_AQ_LNA register).

Figure 67: Two pointers mechanism for application queues

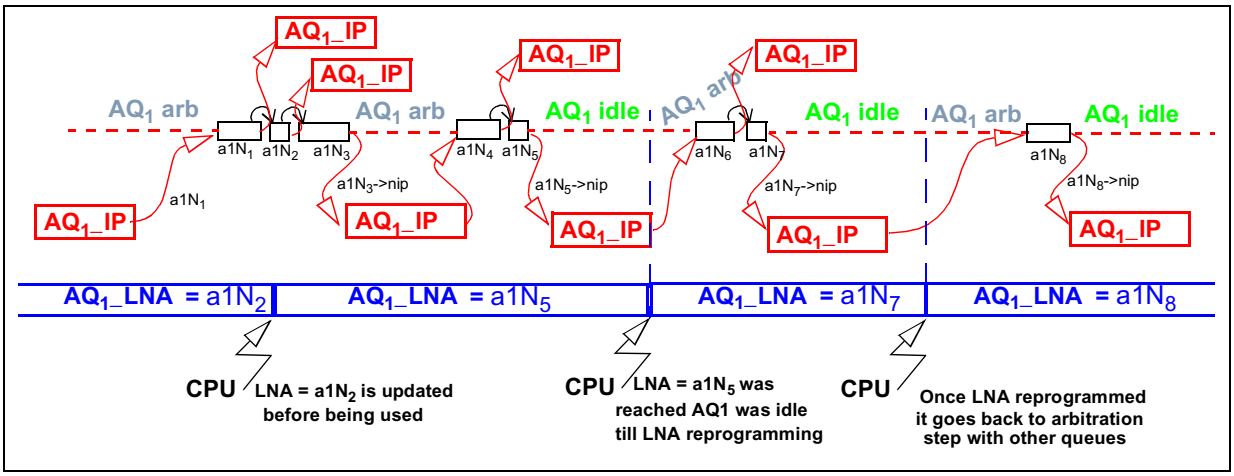


Using two pointers to handle a single queue allows nodes to be added at the end of the queue without having to interrupt the blitter.

For the new nodes to be taken into account, a new value is written in the last node address register. This implies that the nodes for each application queue are created in a dedicated

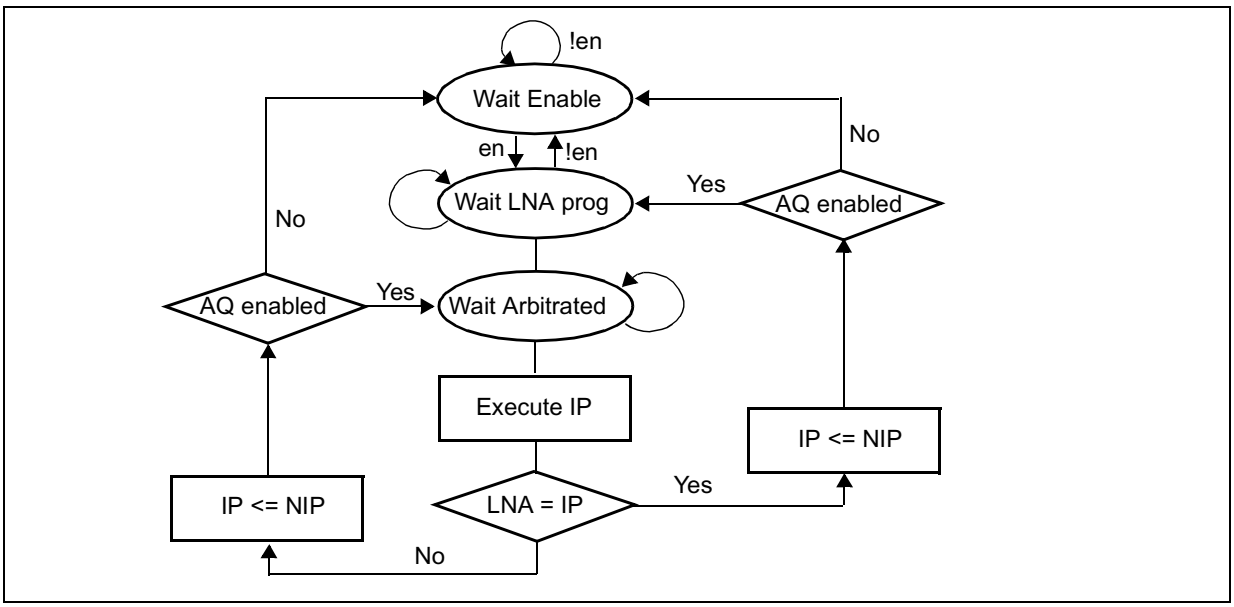
memory space, so that the BLT_NIP in the last node is also a valid field (even if the content of next node is not described.)

Figure 68: End-of-queue management



The application queue standard behavior is described in Figure 69.

Figure 69: Application queue flow chart



Interrupts

Application queues can provide several types of interrupt, depending upon the condition reached. All these interrupt requests are maskable through the BLT_AQ_CTL.IRQ_MASK field. For each application queue, status bits are provided in the BLT_ITS register, and a register (BLT_AQ_STA) indicating the address of the node which last generated an interrupt request. The application queue associated interrupt line is asserted until the corresponding interrupt status bit is cleared by writing 1 in BLT_ITS. See *BLT_ITS* on page 248.

- **LNA reached**
  LNA reached notifies that the application queue has been processed and is now idle. The last node was executed, (BLT_AQ_IP has reached BLT_AQ_LNA) and the queue is now waiting for new nodes to be executed).
- **Node notification**
  Node notification notifies the level and location of the processing within a queue. After issuing this interrupt the next node continues to be processed in the queue. The BLT_AQ_STA register shows the address of the last node which generated an interrupt.
  To issue this interrupt, the BLT_AQ_CTL.IRQ_MASK and the node register BLT_INS.BLITCOMPIRQ must be enabled.

Exception handling

As for the composition queue, application queues are able to support nonstandard behavior.

This allows for efficient, fast queue switching by directing how the application queue hands back to the composition queue.

- **Suspend current queue**
  The node is executed until its end, and then the application queue is released in favor of the requesting composition queue. This behavior does not generate an interrupt, as nothing is stopped or aborted. This is the default behavior, but could lead to some execution time problems, if the application node took too long to finish.
- **Abort current queue**
  This is the fastest solution. The current line is finished and the queue is immediately released in favor of the requesting composition queue. This behavior generates an interrupt (if enabled) as it means that the stopped node is the new starting point for the application queue.
  It might be useful to poll the BLT_AQ_STA register to verify that it is not always the same node which is interrupted for each new pace up. This means that the execution time of this application node is too long to be interleaved within a pace interval.

24.3.3 Composition queue and application queue interleaving flow

Figure 70: Queue interleaving without pace

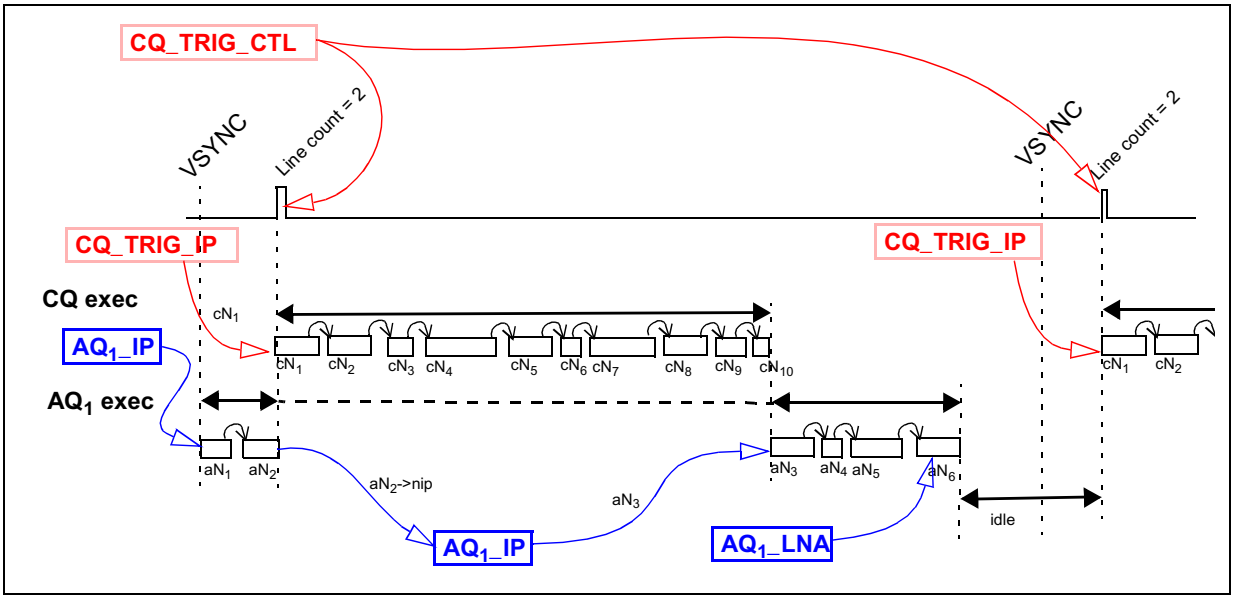
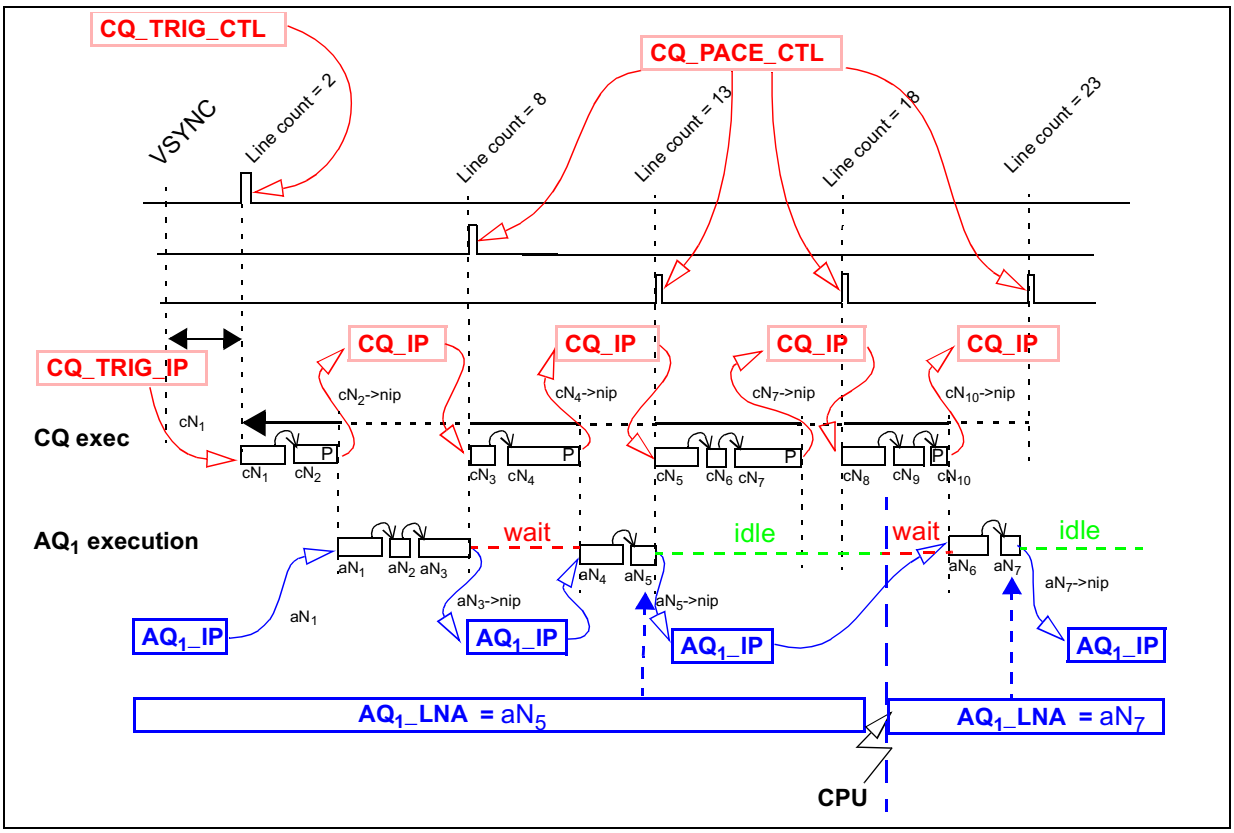


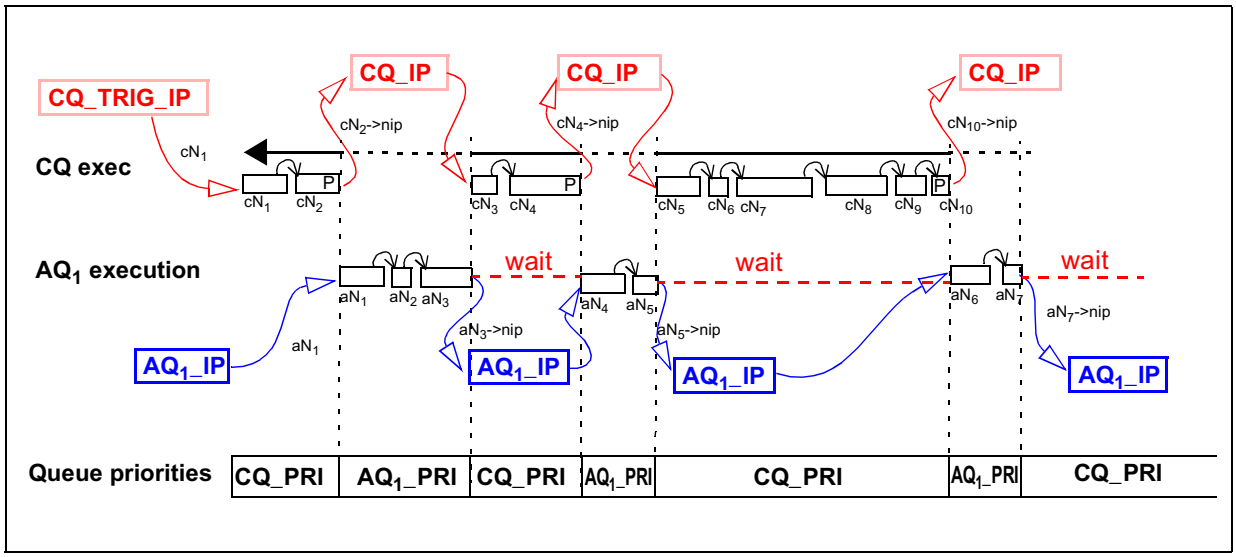
Figure 71: Queue interleaving



Queue priority

A unique priority is set for each queue, by using the fields in the BLT_PRI.CQ_PRI and BLT_PRI.AQ_PRI registers. Generally, the composition queue has a high priority in the system to ensure the composition finishes in the time required, while the application queues have the lower priorities as they are executed in the system idle time.

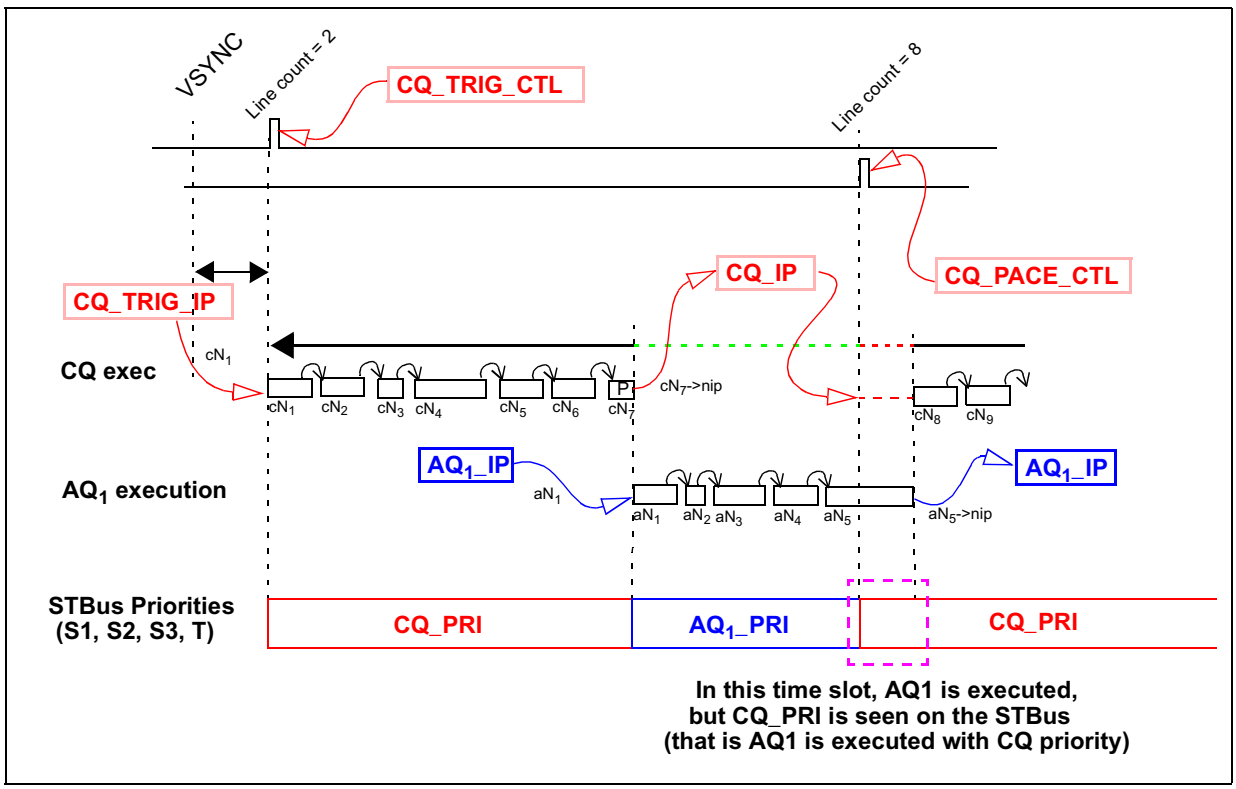
Figure 72: Queue priority



Priority forwarding

When an application queue is executing, and a composition queue is awakened (it is triggered or receives a pace up event), the AQ_PRI is replaced by CQ_PRI, so that the last node(s) to execute in the application queue are executed with the composition queue priority. This mechanism is called priority forwarding.

Figure 73: STBus priority forwarding

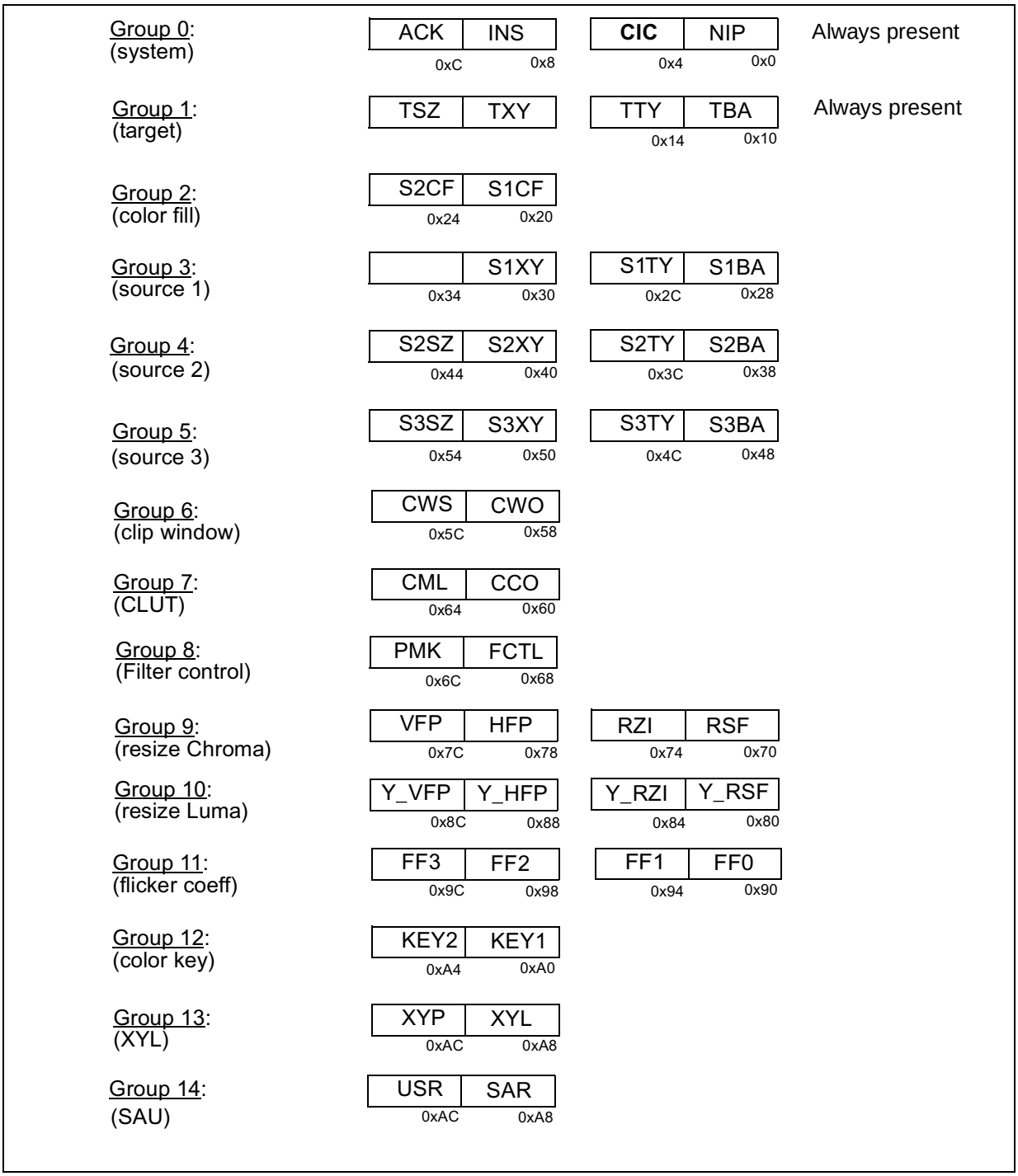


24.3.4 Short linked list

As the output display buffer is being composed hundreds of nodes for each field or frame are generated. To save bandwidth short linked-list mode writes a subset of registers for each node. This subset describes the functionality used, and does not write the registers of unused functions. Nodes can therefore vary in length, according to the functions used. This allows short nodes for simple operations, and longer ones for more complex operations.

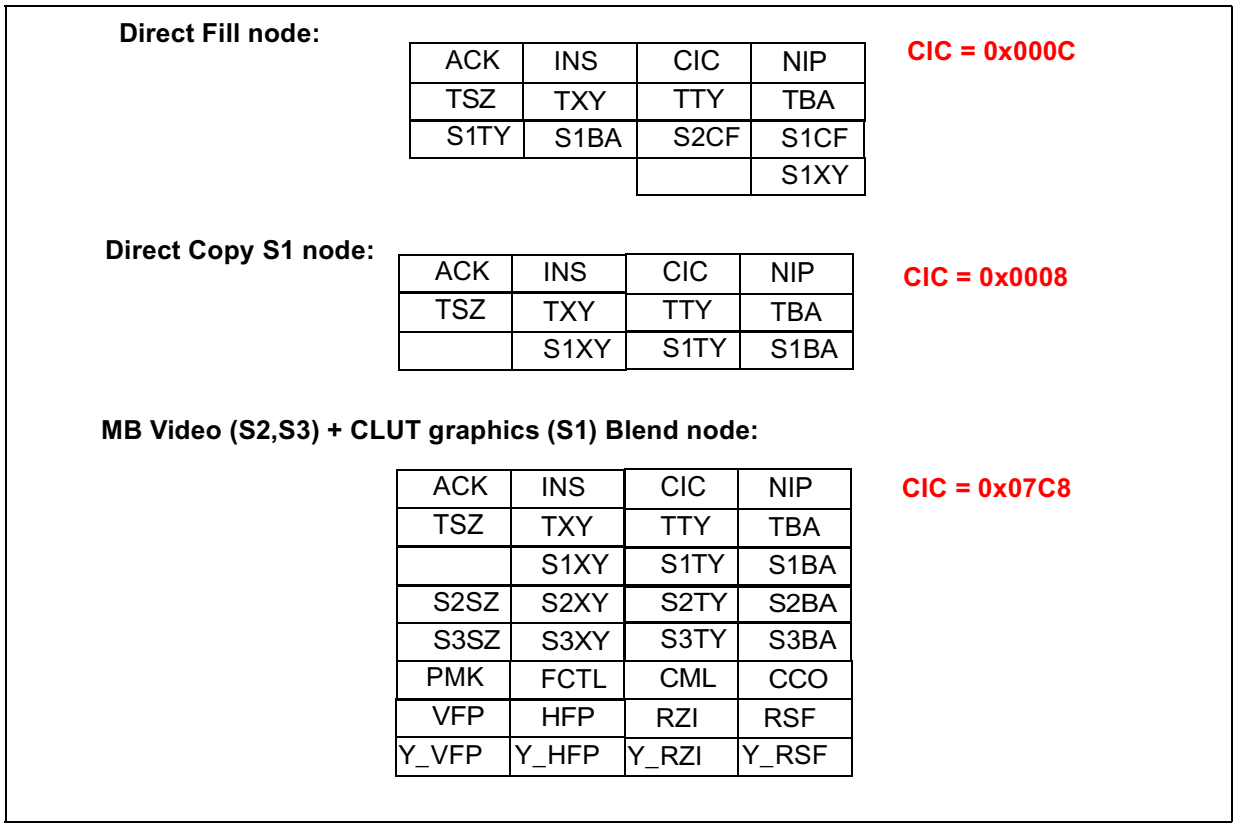
To enable the required functions for short linked-list mode, set the corresponding bits in the BLT_CIC (current instruction content or node resolution key) register

Figure 74: Functional groupings of registers



Functional register groupings are of variable length either 64- or 128-bit width. Only groups 0 and 1 are always present. The groupings depend upon which operation is performed. Unused registers are set to their reset value when the node is loaded.

Figure 75: Examples of short linked lists



24.4 Decoding the source

24.4.1 Source address generation

Source 2

The source 2 address generator scans a rectangular window, according to a 2D 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 * (*NumberOfBytesPerPixel*)

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 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 macroblock formats).

Register BLT_S2XY is the co-ordinate that specifies the start of the input source 2 window relative to the base address.

Register BLT_S2SZ specifies the size of the source 2 window.

Source 1

Source 1 is used for frequent operations such as color-fill or simple source-copy.

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 description registers.

Register BLT_S1BA is the memory base address of the bitmap selected for source 2 (absolute [0,0] location, top-left pixel).

Register 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.

Register BLT_S1XY is the coordinate that specifies the start of the source 1 input window with respect to the base address.

Register BLT_S1SZ does not exist, as S1 can not be resized, S1 width and height are always equal to target width and height (described in BLT_TSZ).

Source 3

Source 3 is used for macroblock luminance part access only. It is always used in conjunction with source 2 which accesses the chrominance part while source 1 accesses luminance.

The source 3 address generator scans a rectangular window, according to a XY addressing scheme. The XY pixel address is converted into a physical memory address, using the source 3 and source 2 description registers.

Register BLT_S3BA is the memory base address of the luminance block of the bitmap selected for source 3 and source 2 (absolute [0,0] location, top-left pixel).

Register BLT_S3TY provides the pitch of source 3. All other values are retrieved from BLT_S2TY register (COLOR_FORMAT, VSO, HSO) as source 2 and source 3 are always used together when the macroblock format is used.

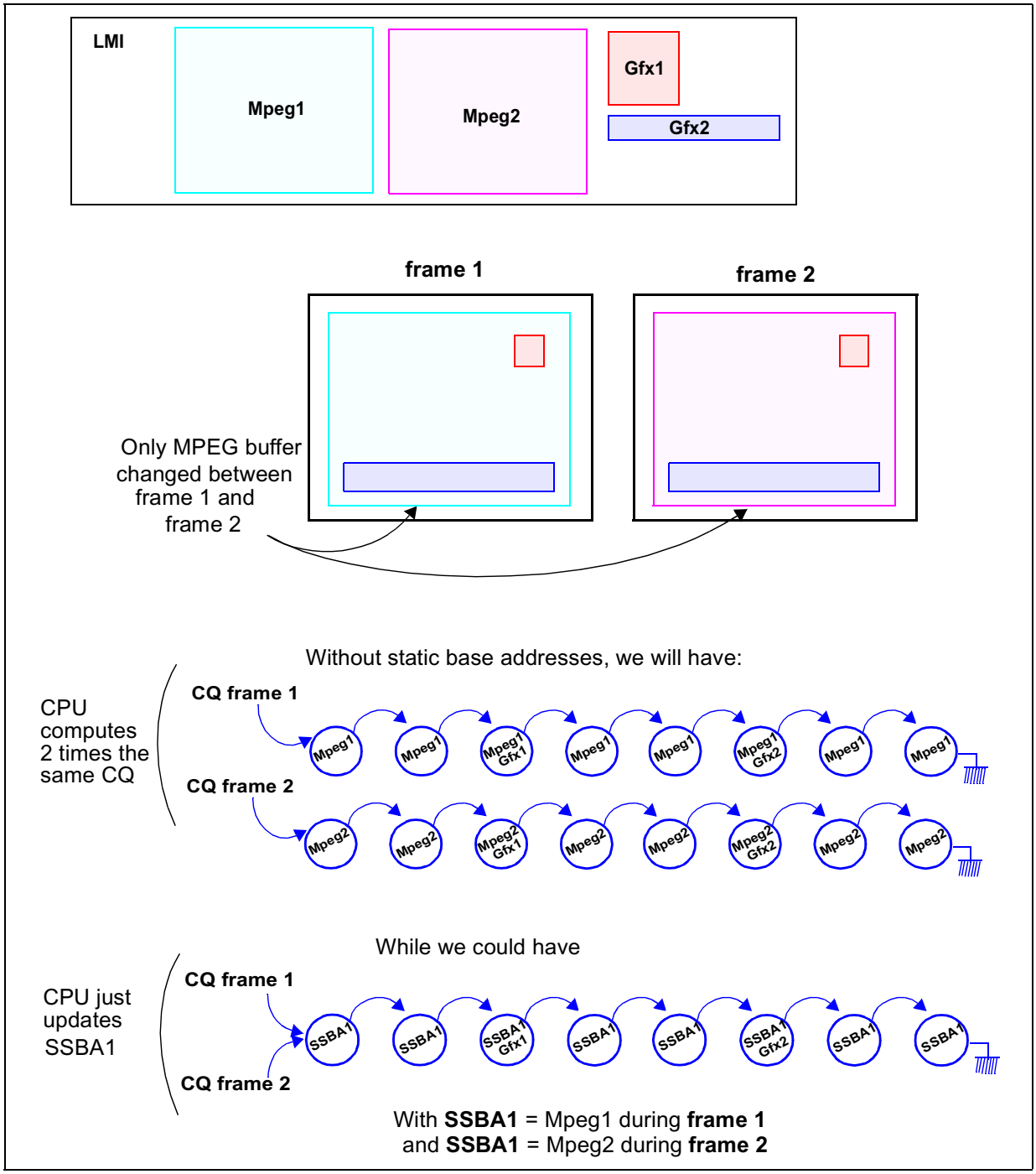
Register BLT_S3XY is the coordinate that specifies the start of the source 3 input window with respect to the base address.

Register BLT_S3SZ specifies the size of the source 3 window.

Static base addresses

When there is no change to display plane position or size the same linked list can be used on several fields or frames to avoid recomputing the full composition queue. Only the base address registers (S1BA, S2BA, S3BA, and TBA) need to change**.**

Figure 76: Static base addresses example



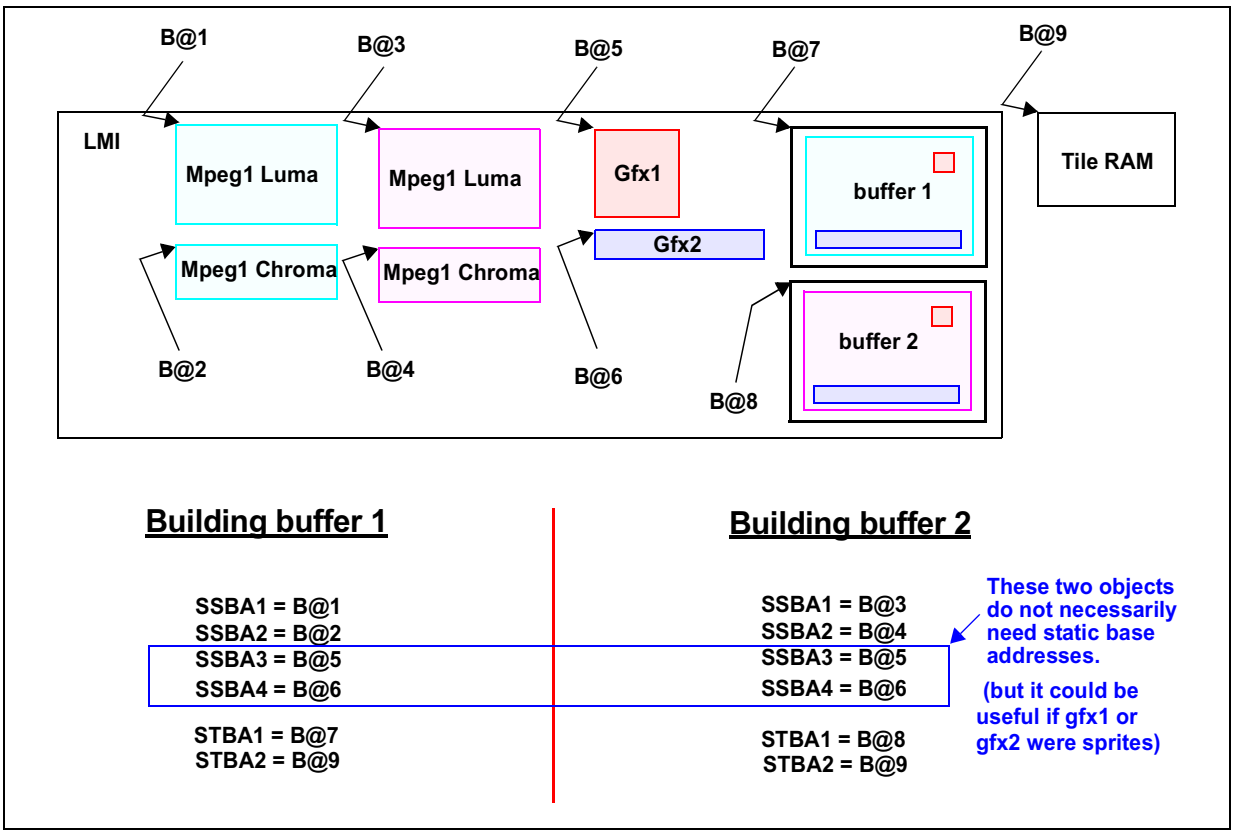
For the static base address, eight registers are shared between the three input sources, and two registers are available for the output target.

Use of the static source base address (SSBA) and the STBA (T for target) is determined in the SAR register.

If the value contained in BLT_SAR.Sn_SSBA equals zero, then the Sn base address is taken from the linked list in the BLT_SnBA register. If the value contained in BLT_SAR.Sn_SSBA is not equal to zero, its value points to the register to be used by Sn as its base address.

The same mechanism is used for the target address, through the STBAn registers.

Figure 77: SSBA/ STBA mechanism



Note: Static base addresses are available for the composition queue only.

The Number of SSBA and STBA register is scalable depending upon the target device. As this static base addresses register mechanism is intended to save CPU bandwidth, the number of static registers is set according to the target application. For a full list of the available registers see *BLT_SSBAn* on page 250.

Source 2

The source 2 formatter converts a 64-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 shown in Table 70.

Table 70: 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	
ARGB8565	A8		R5/3MSB or 000		G6/2MSB or 00		B5/3MSB or 000	
ARGB8888	A8		R8		G8		B8	
ARGB4444	0/A4/100 (keep 0 and 128)		R4/4MSB or 0000		G4/4MSB or 0000		B4/4MSB or 0000	
YCbCr888	128		Cr8		Y8		Cb8	

Table 70: 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
ARGB1555	A1/0000000 or KEY1.A0 or KEY2.A1		R5/3MSB or 000		G5/3MSB or 000		B5/3MSB or 000	

Table 71: 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	128		0 (don't care)				0000000I_0	
ACLUT2	128		0 (don't care)				000000I_1.I_0	
ACLUT4	128		0 (don't care)				0000I_3...I_0	
ACLUT8	128		0 (don't care)				I_7...I_0	
ACLUT44	0/A4/100 (keep 0 and 128)		0 (don't care)				0000I_3...I_0	
ACLUT88	A8		0 (don't care)				I_7...I_0	

Table 72: 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 73: 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 (first pixel YCbCr)	Cb 1rst pixel		Y 1rst pixel		Cr 1rst pixel		Y 2nd pixel	
YCbCr 4:2:2 (first pixel Y-only)	Y 1rst 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 2 pixels simultaneously. The alpha channel (128 = opaque) is inserted by the 4:2:2 to 4:4:4 converter (next block in the pipeline). This converter is automatically enabled for this configuration.

When the input picture has an 8-bit per-pixel alpha component, this component can have a 0:128 or 0:255 range. The internal 8-bit alpha format being 0:128, a 0:255 alpha component is converted using the following formula: $A_{0..128} = (A_{0..255} + 1) >> 1$

Source 1

In fast direct-copy mode, the source 1 bus formatter is unused. Rectangular graphics areas are transferred faster, but sub-byte color formats are not supported. The rectangular clipping feature cannot be used.

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 64-bit input bus into a pixel bus, that feeds the ALU operator.

In normal mode, the bus formatter is similar to the one described for source 2, except for YCbCr4:2:2R which is not supported as a source 1 operand. Source 1 data can be combined with source 2 data using the ALU operator.

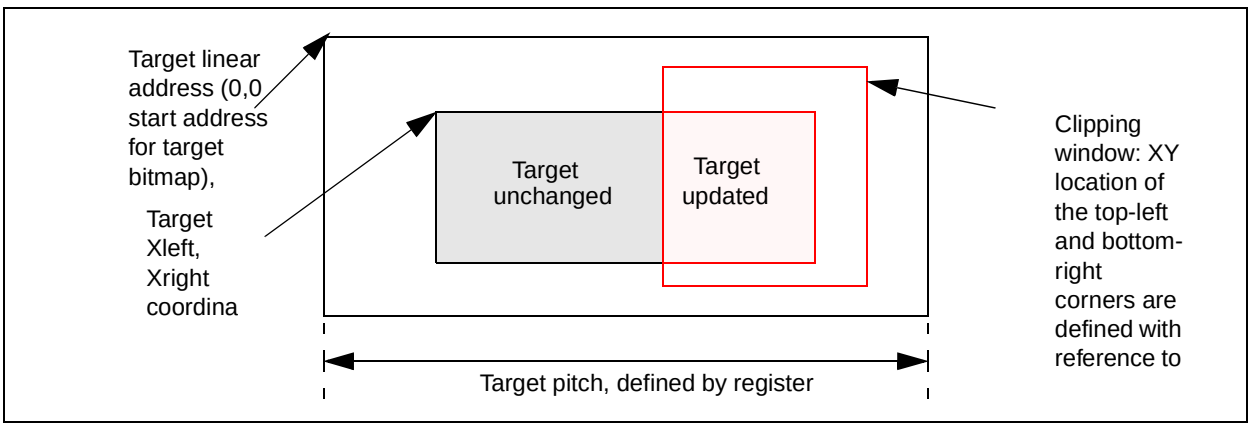
24.5 Clipping

24.5.1 Rectangular clipping

For a rectangular window each write access to the target plane can be enabled on a pixel-per-pixel basis.

The base address of the 2D graphics engine is in x-y coordinates. Hardware clipping uses four registers to define the rectangular clipping area. Figure 78 shows a block transfer operation inside a buffer, using a hardware clipping window. The buffer is defined using a linear start address. The display area and the clipping window are defined by x-y coordinates. The display function is only valid inside the clipping window.

Figure 78: 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.

Register bits BLT_CWO.XDO and BLT_CWO.YDO define the target base address for the start of the window.

Register bits BLT_CWS.XDS and BLT_CWS.YDS define the target base address for the window end.

24.6 Color operations and conversion

24.6.1 RGB to YCbCr conversion

Graphic matrix

Converting from RGB to YCbCr the Y component is clipped between 16 and 235, after applying the matrix. The chroma range is -112/+112, and can be optionally encoded in offset-binary format (+128, 16/240 range).

- BLT_CCO.CCO_INCOL sets the input color converter colorimetry (601).
- BLT_CCO.CCO_INSIGN sets the input color converter chroma color sign.
- BLT_CCO.CCO_INDIR sets the input color converter direction (RGB to YCbCr).

Table 74 to Table 75 show the hard wired matrices used for the conversion (assuming offset binary chroma)

Table 74: 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 75: 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

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/+126, and can be optionally encoded in offset-binary format (+128, 1/254 range).

- BLT_CCO.CCO_INCOL sets the input color converter colorimetry (601).
- BLT_CCO.CCO_INSIGN sets the input color converter chroma color sign.
- BLT_CCO.CCO_INDIR sets the input color converter direction (RGB to YCbCr), and the output color converter is automatically programmed with the other direction.
- BLT_CCO.CCO_INGFXnVID sets the input color converter to graphic or video matrix.

Table 76 to Table 77 shows the hard-wired matrices used for the conversion (assuming offset binary chroma).

Table 76: 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 77: 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

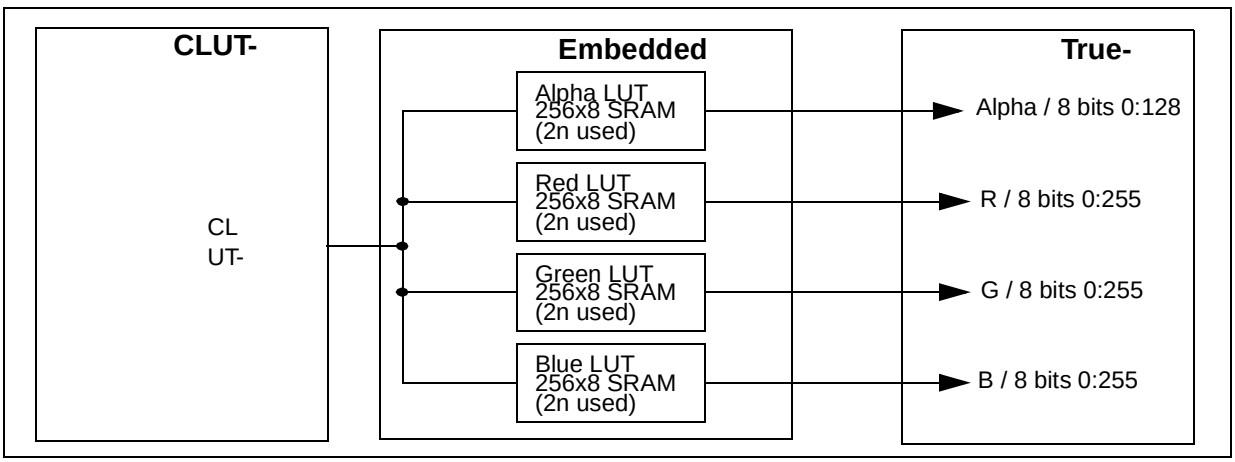
24.6.2 Color expansion

During color expansion a CLUT-based bitmap is transformed into a true-color bitmap, using the embedded look-up-table (256 x 32 SRAM). Figure 79 illustrates a CLUT-to-RGB conversion and Figure 80 a CLUT88/44-to-RGB conversion.

In a CLUT-to-RGB conversion:

- CLUT entries can be color-corrected if required (for example gamma, 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 79: ACLUTn to (A)RGB conversion

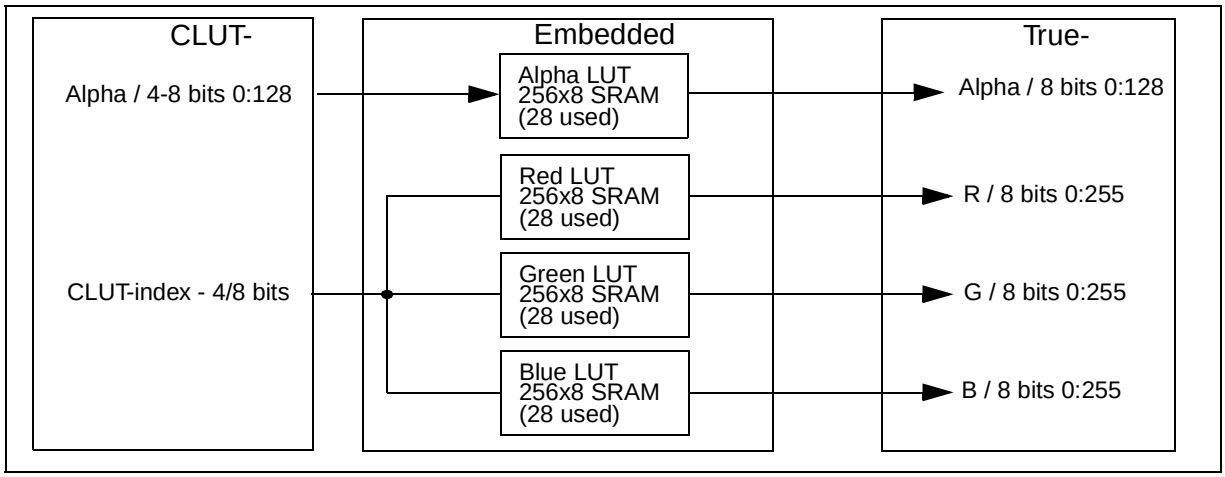


In a CLUT88/44-to-RGB conversion:

- CLUT entries can be color-corrected if required (for example gamma, 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.

Figure 80: ACLUT88/44 to ARGB conversion

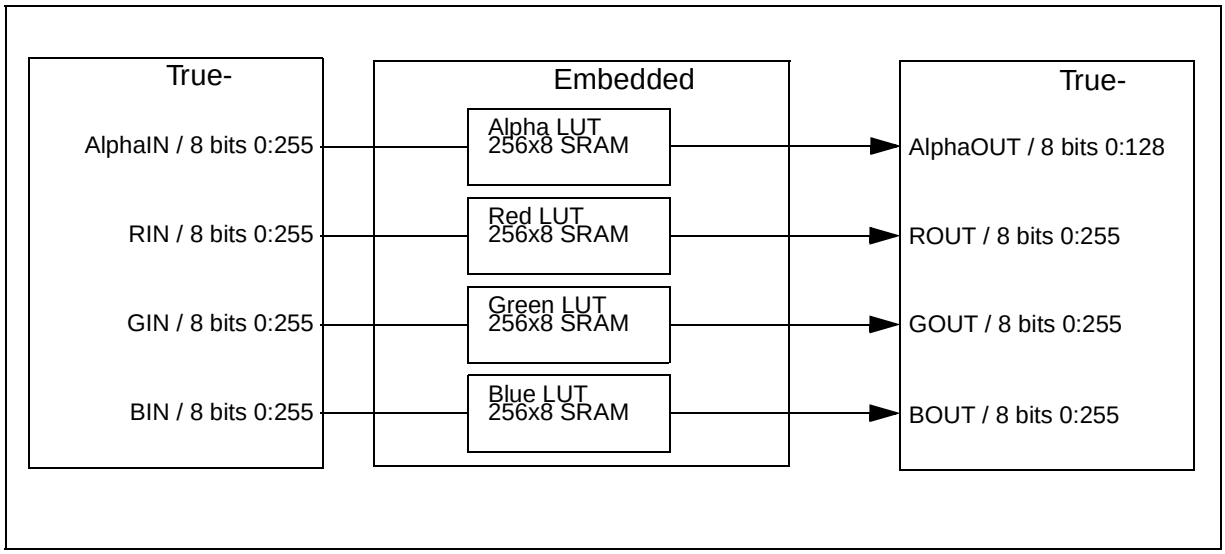


Note: Color expansion is performed exclusively on either source 1 or source 2.

24.6.3 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 (color effect, conversion to a gray-scale bitmap).

Figure 81: Color correction for true-color inputs



Register BLT_CML is a pointer to the local memory CLUT address.

Register bit BLT_CCO.CLUT_UPDATE enables CLUT refresh from memory.

Register bit BLT_CCO.CLUT_MODE sets the CLUT mode: color expansion and color correction.

Note: Color correction is performed exclusively on either source 1 or source 2.

24.6.4 Color fill

Registers BLT_S1CF and BLT_S2CF provide the solid color value for filling during an operation. BLT_S1CF allows the direct fill mode to be used for faster performance.

The number of significant bits varies from 1 bpp to 32 bpp according to the bit depth of the color format.

Color fill is supported for all color formats except for macroblocks.

Direct fill and direct copy modes

These modes are used on source 1 only. They allow the output bit rate to be doubled compared to simple copy or fill operations such as rectangle fill and rectangle move. The data is not interpreted when copying.

Note: Only raster formats are supported.

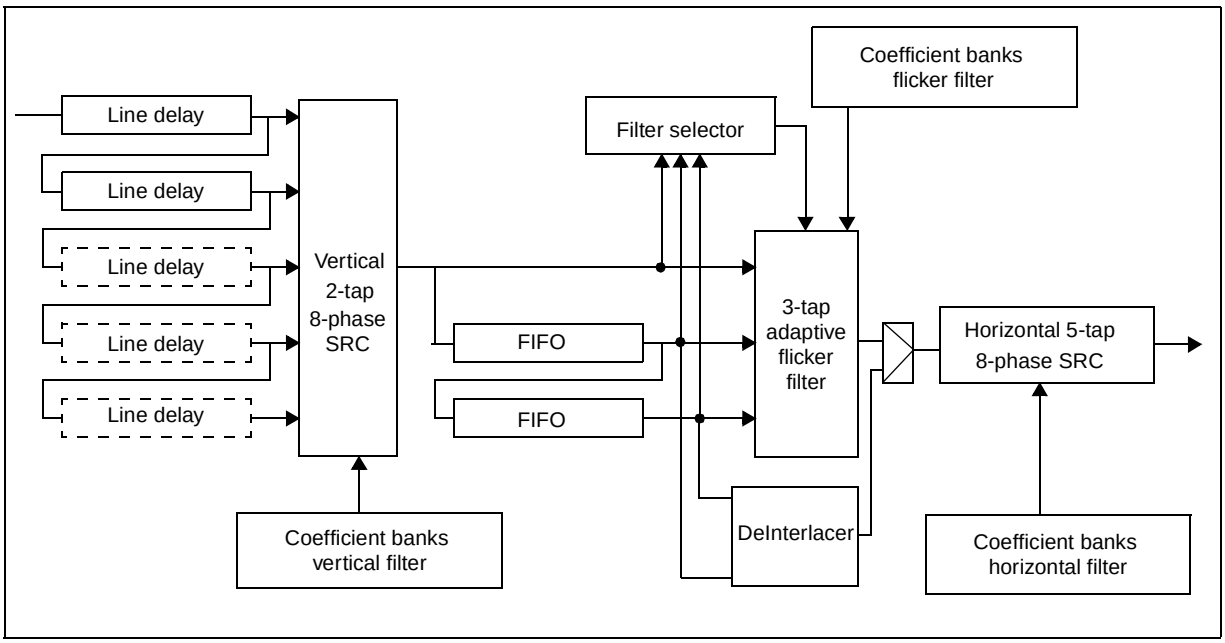
To perform in direct fill or direct copy mode source 1 should be enabled and sources 2 and 3 (and their associated operators) disabled. The target and source 1 should have the same data format and the same vertical and horizontal pixel scan order. In direct fill mode the scan order should be top-to-bottom and left-to-right (that is S1TY_HSO and S1TY_VSO are equal to 0.)

24.7 2D resizing and filtering

24.7.1 Overview

The 2D resize engine is composed of four sub-blocks: a vertical source, a horizontal source, a context-sensitive flicker filter and a deinterlacer. It is illustrated in Figure 82.

Figure 82: 2D resize engine overview



24.7.2 Vertical filter

The vertical filter is a 2 taps polyphase filter, based on a 6.10 bit sample rate converter. It uses the upper 3-bits of the decimal part to give eight subpositions. Thus it has eight sets of two coefficients subposition based. The vertical filter can be used in two ways depending upon the input format. For all raster formats, it operates as a single device applying the same filter on all the components (A, R, G an B) provided by source 2. For all macroblock formats, it becomes two

separate filters, applying different coefficients and resize factors on two separate paths: the luminance and chrominance path (source 3 and source 2).

Figure 83: Vertical filter in macroblock mode

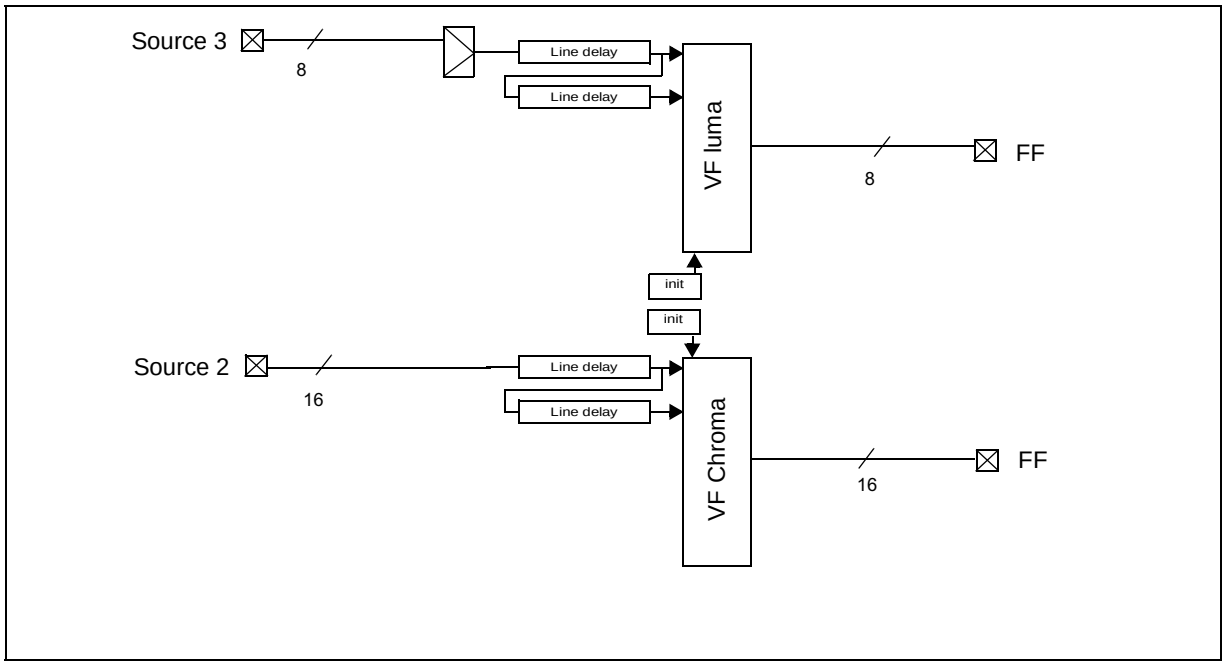
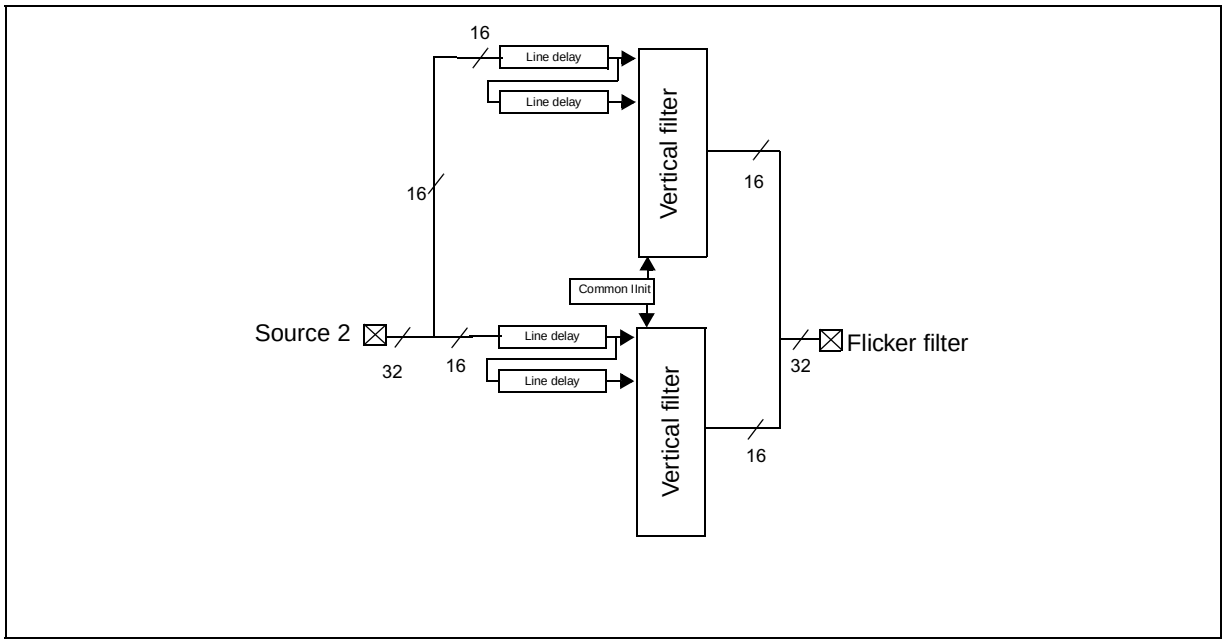


Figure 84: Vertical filter in raster mode



To use the vertical filter in raster mode and initialize the filter, set:

- the initial subposition in BLT_RZI.VSRC_INIT,
- the resizing factor in the BLT_FCTL.VSRC_INC field.

These input values are used by both paths if the blitter display deals with raster formats.

To use the vertical filter in 4:2:0 macroblock mode and initialize the filter, set:

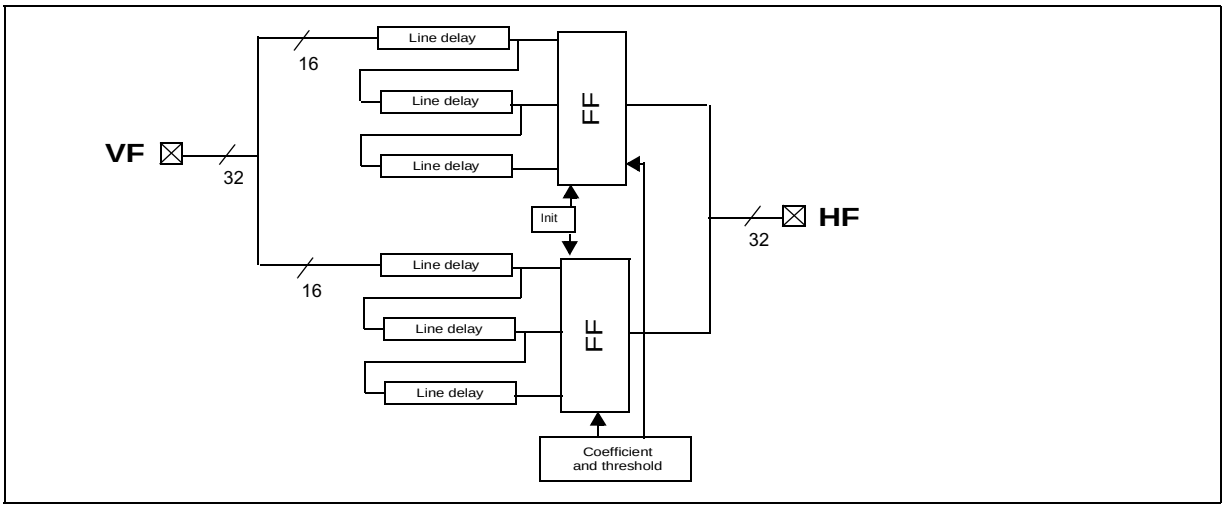
- two initial subpositions in BLT_RZI.VSRC_INIT and BLT_Y_RZI.VSRC_INIT,
- the resizing factor needs to be set in the BLT_RSF.VSRC_INC and BLT_Y_RSF.VSRC_INC fields.

Vertical filter programing in macroblock format is detailed in Section 24.7.5: *Using the 4:2:x macroblock-based plane as a source* on page 239.

Note: When the input format is 4:2:0 macroblock, the two luma and chroma paths will not provide the same amount of data. The chroma path therefore needs to be upsampled by two.

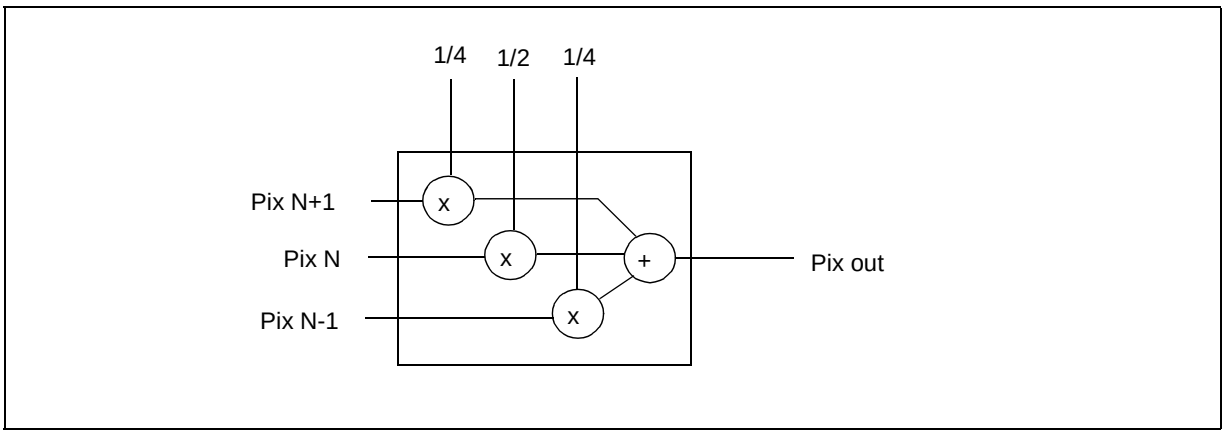
24.7.3 Fixed coefficient flicker filter

Figure 85: Flicker filter block diagram



The flicker filter is a 3-tap filter, and applies a simple flicker algorithm: 1- 2 - 1.

Figure 86: Flicker filter 1-2-1 data path



24.7.4 Horizontal filter

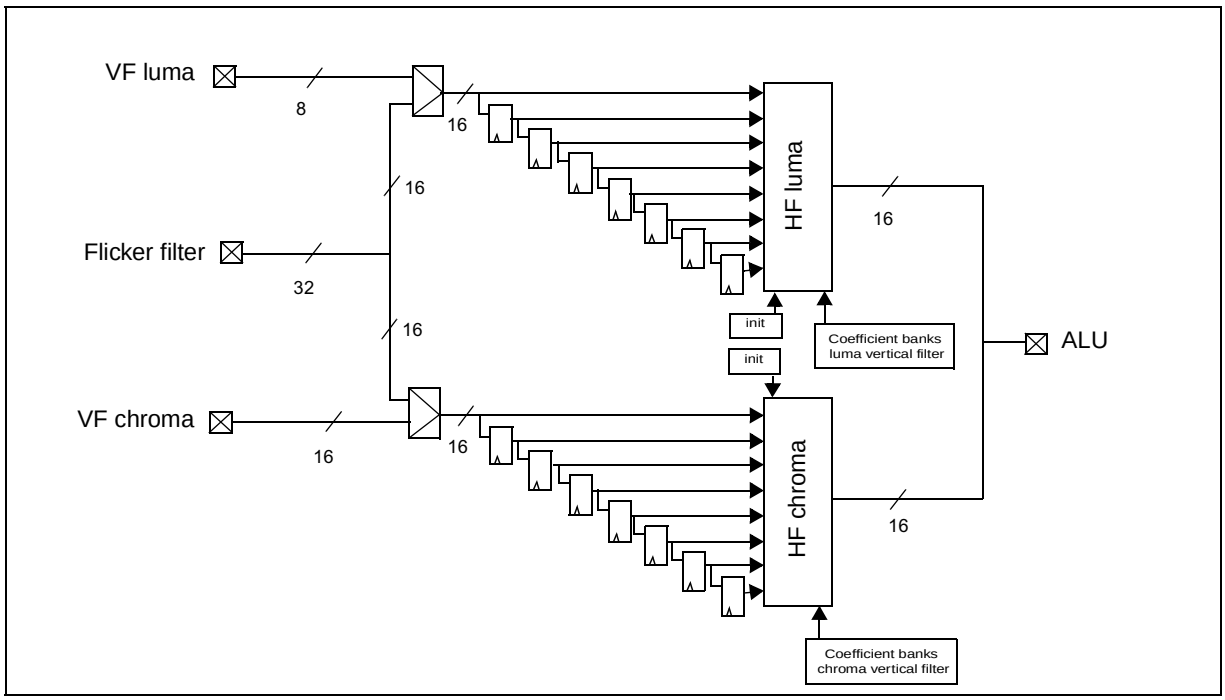
The horizontal filter is an 8-taps polyphase filter, with a 3-bit subposition (eight values), with eight sets of eight coefficients. The horizontal filter data path is based on a 6.10 sample rate converter, which first upsamples the incoming pixels, and applies the subposition selected filtering coefficients on the samples (if enabled) to generate output pixels.

Like the vertical filter, the horizontal filter can be used in two distinct forms depending upon input format:

- a single device for all raster input formats,
- as two separate filters, applying different coefficients and resize factors on two separate paths for the 4:2:0 macroblocks format.

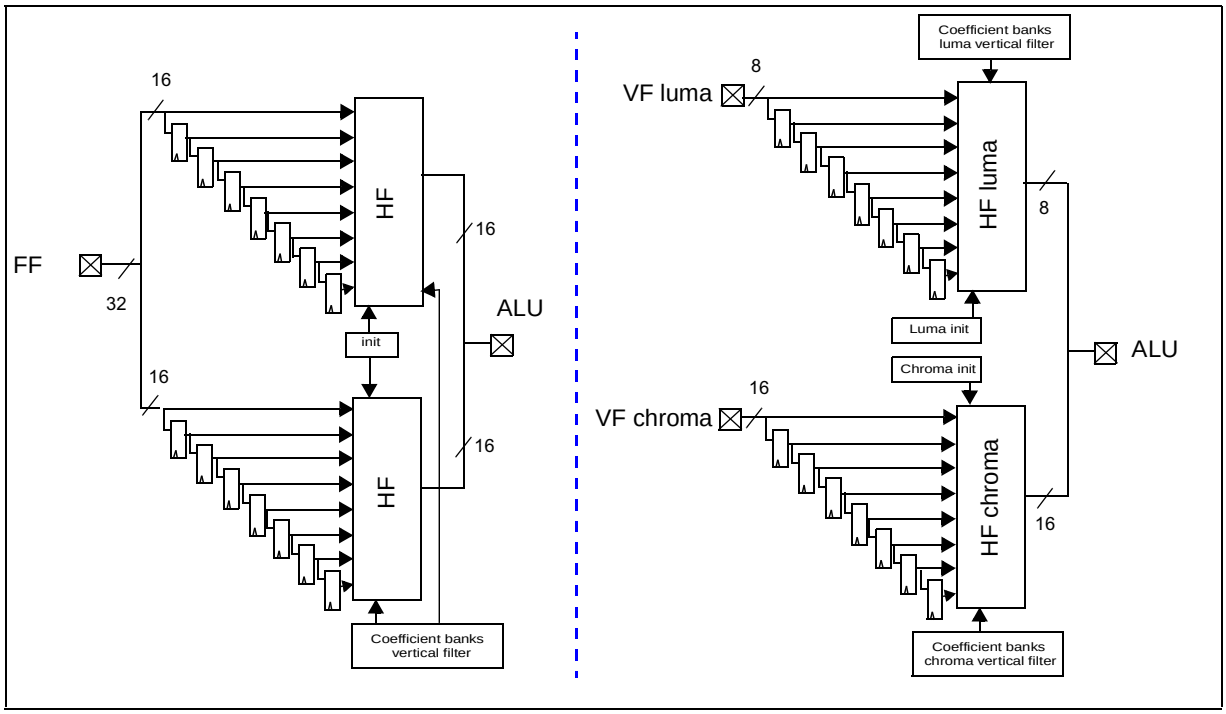
When in 4:2:0 macroblock formats, chrominance also needs to be horizontally upsampled so fewer chroma samples than luma samples are received. The vertical filter data path needs to be split in two separate and independent paths, with different rescale factors, initialization, and coefficients.

Figure 87: Horizontal filter block diagram



Depending on the data processed by the horizontal filter, there could be two different views as shown in Figure 88.

Figure 88: Horizontal filter raster and macroblock modes



To use the horizontal filter in raster mode

To initialize the filter, set:

- the initial subposition in BLT_RZI.HSRC_INIT,
- the number of times the first pixel needs to be repeated to deal with filter startup in BLT_RZI.HNB_REPEAT,
- the resizing factor in the BLT_RSF.HSRC_INC.

These input values are used by both paths if the blitter display deals with raster formats.

To use the vertical filter in macroblock mode.

To initialize the filter set:

- two initial subpositions in BLT_RZI.HSRC_INIT and BLT_Y_RZI.HSRC_INIT,
- the number of time the first line needs to be repeated to deal with filter startup in BLT_RZI.HNB_REPEAT and BLT_Y_ RZI.HNB_REPEAT.
- the resizing factor needs to be set in the BLT_RSF.HSRC_INC and BLT_Y_RSF.HSRC_INC.

Horizontal filter programing in macroblock format is detailed in Section 24.7.5.

24.7.5 Using the 4:2:x macroblock-based plane as a source

Because chrominance data in macroblock format is compressed, one luma sample does not necessarily have a corresponding chroma sample. As the ALU processes in 4:4:4 format, chroma samples must be upsampled prior to use.

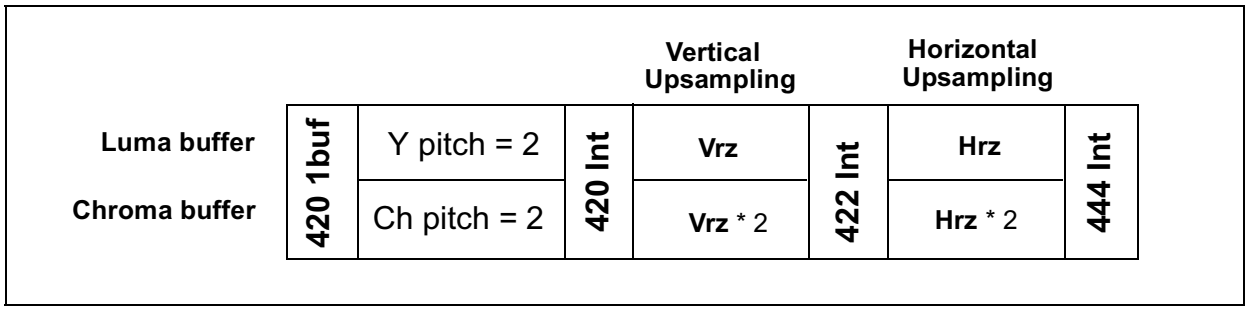
The simplest and cleanest way to upsample chrominance is to use the horizontal and vertical filters. The programming of the filters varies according to the content encoding style and the display style.

The vertical and horizontal filters and resizers are enabled as follows.

- BLT_FCTL.2DHF_MODE = horizontal filter active on both color and alpha,
- BLT_FCTL.2DVF_MODE = vertical filter active on both color and alpha,
- BLT_FCTL.LUMA_2DHF_MODE = both luma horizontal resizer and horizontal filter active,
- BLT_FCTL.LUMA_2DVF_MODE = both luma vertical resizer and vertical filter active.

Note: In the next sections, Vrz and Hrz stand for the vertical and horizontal resize factors. If no resize other than a chroma upscale is applied, these are equal to 1. The increment value programmed in the XXX_INC registers is 1/Vrz and 1/Hrz.

Figure 89: Content interlaced, display interlaced



For content interlaced with display interlaced modes the 2D blitter display engine is programmed as below.

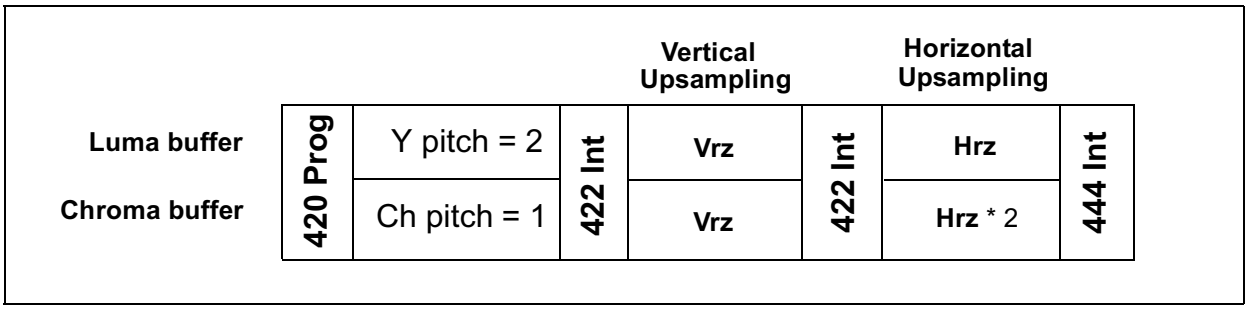
BLT_S2BA = chroma base address (or B@ + pitch depending field)
BLT_S2TY.PIXMAP_PITCH = chroma pitch
BLT_S2TY.COLOR_FORM = 4:2:0 MB
BLT_S2TY.MB_FIELD = field
BLT_S3BA = luma base address (or B@ + pitch)
BLT_S3TY.PIXMAP_PITCH = luma pitch
BLT_S3TY.MB_FIELD = field
BLT_RSF.VSRC_INC = 1 / (Vrz * 2)
BLT_RSF.HSRC_INC = 1 / (Hrz * 2)
BLT_RZI = filters initialization (picture settings dependant)
BLT_Y_RSF.VSRC_INC = 1 / Vrz
BLT_Y_RSF.HSRC_INC = 1 / Hrz
BLT_Y_RZI = filters initialization (picture settings dependant)

For 4:2:2 MB format replace

BLT_S2TY.COLOR_FORM = 4:2:2 MB

BLT_RSF.VSRC_INC = Vrz

Figure 90: Content progressive, display interlaced

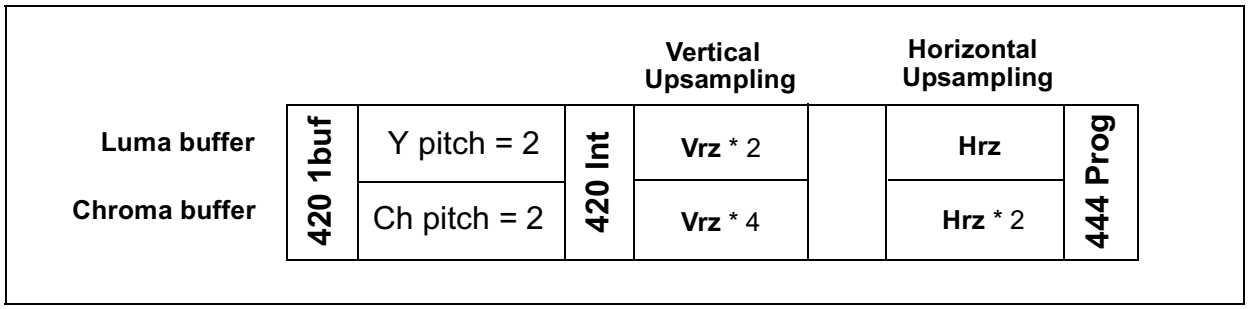


BLT_S2BA = Chroma base address
BLT_S2TY.PIXMAP_PITCH = Chroma pitch
BLT_S2TY.COLOR_FORM = 4:2:0 MB
BLT_S2TY.MB_FIELD = frame
BLT_S3BA = Luma base address (or B@ + pitch)
BLT_S3TY.PIXMAP_PITCH = Luma pitch
BLT_S3TY.MB_FIELD = field
BLT_RSF.VSRC_INC = 1 / Vrz (in 6.10 format)
BLT_RSF.HSRC_INC = 1 / Hrz * 2 (in 6.10 format)
BLT_RZI = Filters initialization (picture settings dependant)
BLT_Y_RSF.VSRC_INC = 1 / Vrz (in 6.10 format)
BLT_Y_RSF.HSRC_INC = 1 / Hrz (in 6.10 format)
BLT_Y_RZI = Filters initialization (picture settings dependant)

For 4:2:2 MB format replace:

BLT_S2TY.COLOR_FORM = 4:2:2 MB
BLT_S2TY.MB_FIELD = field

Figure 91: Content interlaced, display progressive



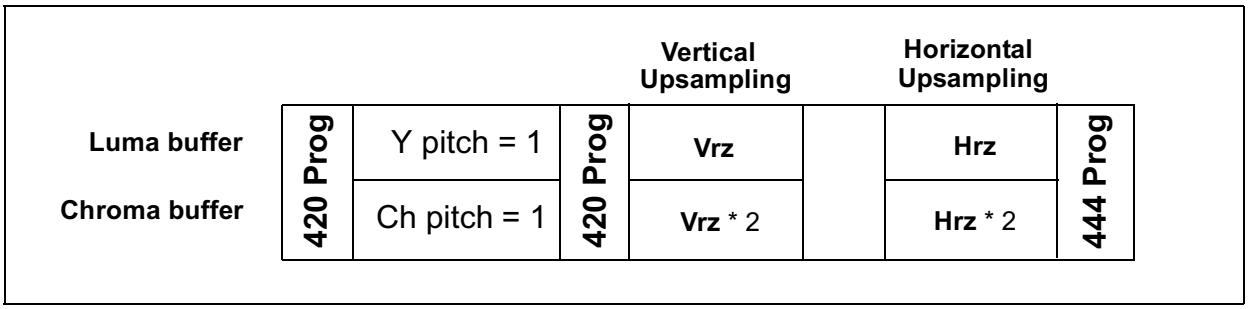
BLT_S2BA = Chroma base address (or B@ +pitch)
BLT_S2TY.PIXMAP_PITCH = Chroma pitch
BLT_S2TY.COLOR_FORM = 4:2:0 MB
BLT_S2TY.MB_FIELD = field
BLT_S3BA = Luma base address (or B@ + pitch)
BLT_S3TY.PIXMAP_PITCH = Luma pitch
BLT_S3TY.MB_FIELD = field
BLT_RSF.VSRC_INC = 1 / (Vrz * 4) (in 6.10 format)
BLT_RSF.HSRC_INC = 1 / (Hrz * 2) (in 6.10 format)
BLT_RZI = Filters initialization (picture settings dependant)
BLT_Y_RSF.VSRC_INC = 1 / (Vrz * 2) (in 6.10 format)
BLT_Y_RSF.HSRC_INC = 1 / Hrz (in 6.10 format)
BLT_RZI = Filters initialization (picture settings dependant)

For 4:2:2 MB format, simply replace:

BLT_S2TY.COLOR_FORM = 4:2:2 MB

BLT_RSF.VSRC_INC = 1 / (Vrz * 2)

Figure 92: Content progressive, display progressive



BLT_S2BA = Chroma base address

BLT_S2TY.PIXMAP_PITCH = Chroma pitch

BLT_S2TY.COLOR_FORM = 4:2:0 MB

BLT_S2TY.MB_FIELD = frame

BLT_S3BA = Luma base address (or B@ + pitch)

BLT_S3TY.PIXMAP_PITCH = Luma pitch

BLT_S3TY.MB_FIELD = frame

BLT_RSF.VSRC_INC = 1 / (Vrz * 2) (in 6.10 format)

BLT_RSF.HSRC_INC = 1 / (Hrz * 2) (in 6.10 format)

BLT_RZI = Filters initialization (picture settings dependant)

BLT_Y_RSF.VSRC_INC = 1 / Vrz (in 6.10 format)

BLT_Y_RSF.HSRC_INC = 1 / Hrz (in 6.10 format)

BLT_Y_RZI = Filters initialization (picture settings dependant)

For 4:2:2 MB format replace

BLT_S2TY.COLOR_FORM = 4:2:2 MB

BLT_RSF.VSRC_INC = 1 / Vrz

24.8 Blending operations

24.8.1 Output unformatting

The output bus unformatter transforms the internal ARGB8888 or AYCbCr8888 32-bit bus in the output pixel format bus.

Table 78: Target unformatter output

Output format	Alpha	Color data		
	31:24	23:16	15:8	7:0
ARGB8888	A[7:0] or expand	R[7:0]	G[7:0]	B[7:0]
RGB888	N/A	R[7:0]	G[7:0]	B[7:0]
ARGB8565	A[7:0] or expand	R[7:3]+0.5 or dither	G[7:2]+0.5 or dither	B[7:3]+0.5 or dither
RGB565	N/A	R[7:3]+0.5 or dither	G[7:2]+0.5 or dither	B[7:3]+0.5 or dither
ARGB1555	A[8]	R[7:3]+0.5 or dither	G[7:3]+0.5 or dither	B[7:3]+0.5 or dither
ARGB4444	A[6:3] 0xF if128, 0x0 kept	R[7:4]+0.5 or dither	G[7:4]+0.5 or dither	B[7:4]+0.5 or dither
A1	A[7]	N/A	N/A	N/A
A8	A[7:0] or expand	N/A	N/A	N/A
AYCbCr8888	A[7:0] or expand	Cr[7:0]	Y[7:0]	Cb[7:0]
YCbCr888	N/A	Cr[7:0]	Y[7:0]	Cb[7:0]

2 x 2 dithering avoids leveling the output picture when reducing the output dynamic (reducing from 8 bits). It consists of introducing pseudorandom noise on the lower levels of decimated color.

The dither algorithm is:

DitherPattern = (Ypos and 0x1) + (((Xpos and 0x1)^(Ypos and 0x1)) << 1)

Then (example for a RGB565 output)

r = r[7:0] + *DitherPattern*<<1; R =r >>3
g = g[7:0] + *DitherPattern*; G = g>>2
b = b[7:0] + *DitherPattern*<<1; B =b >>3
This dither mechanism is enabled by setting the BLT_TTY.RGB_ROUND bit.

The alpha can be stored in the 128 to 0 or 255 to 0 range by setting the BLT_TTY.ALPHA_RANGE bit. Internal processing is done in 128 to 0 mode, so when this bit is set, the alpha channel is expanded as follows:

$Alpha_{255} = ((Alpha_{128} - 1) << 1) + 1$

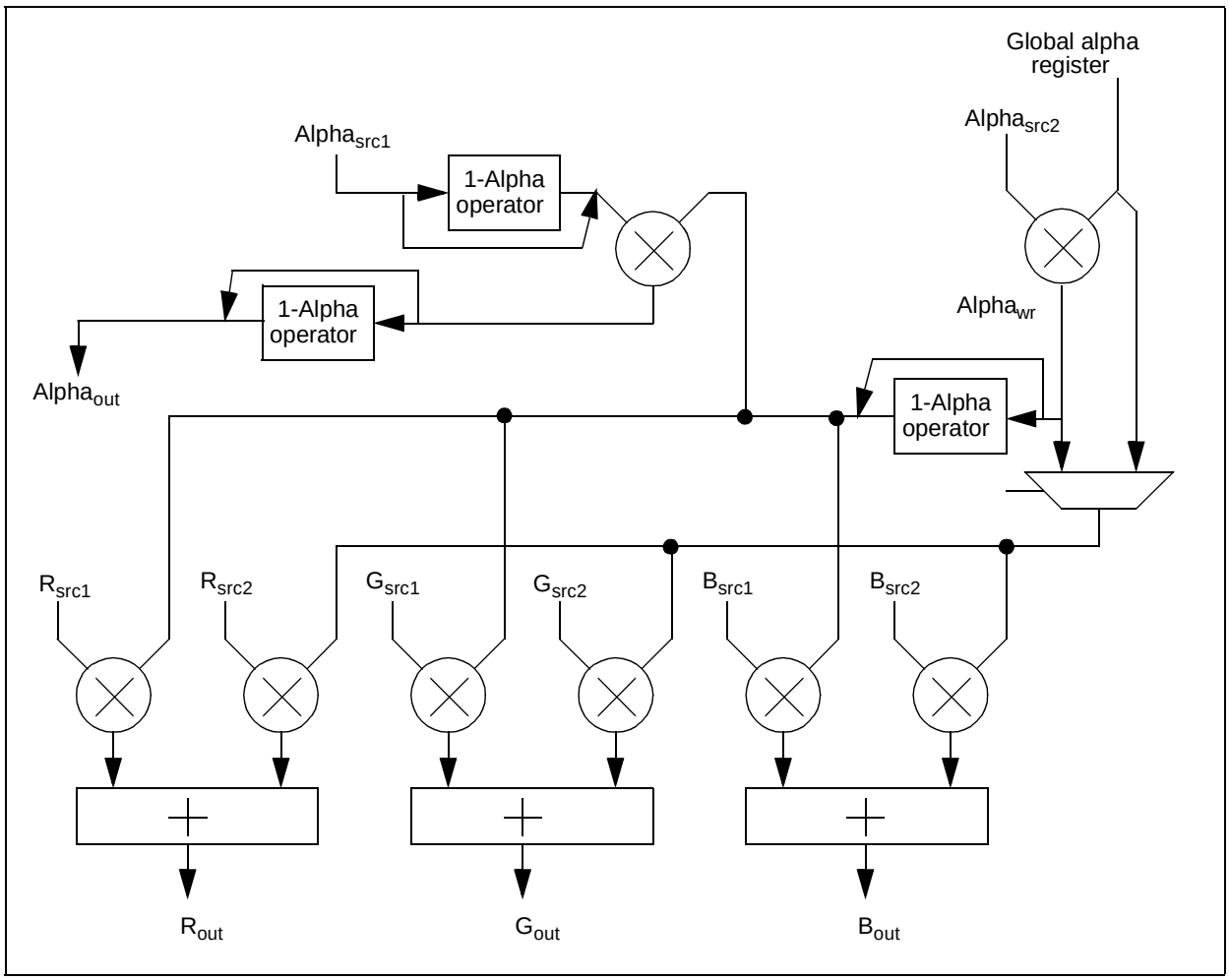
24.8.2 Alpha blending

The alpha blender interpolates colors between source 1 and source 2 (see Figure 93.)

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 with register bit BLT_ACK.GALPHA (global alpha register).

Source 2 supports alpha premultiplied or nonalpha premultiplied color formats (premultiplied ARGB formats are AR, AG and AB). The multiplexor is necessary to select between $Alpha_{wr}$ and the global alpha register.

Figure 93: Alpha blender architecture



Blending equations are given in Figure 94.

Figure 94: Blender reference equations

If Src2 is not pre-multiplied: $\mathbf{RGB_{out}} = Asrc2 \times Global\_Alpha\_Register \times PRGBsrc2 + (1 - Asrc2 \times Global\_Alpha\_Register) \times PRGBsrc1$
If Src2 is pre-multiplied: $\mathbf{RGB_{out}} = Global\_Alpha\_Register \times PRGBsrc2 + (1 - Asrc2 \times Global\_Alpha\_Register) \times PRGBsrc1$
In any case, the resulting translucency is: $(1 - Alphaout) = (1 - Asrc2 \times Global\_Alpha\_Register) \times (1 - Asrc1)$ *This is equivalent to:* $Alphaout = Asrc2 \times Global\_Alpha\_Register + Asrc1 \times (1 - Asrc2 \times Global\_Alpha\_Register)$ *or* $Alphaout = Asrc1 + Asrc2 \times Global\_Alpha\_Register \times (1 - Asrc1)$

The $Alpha_{out}$ component is written into the target bitmap, only if the target format includes a perpixel alpha component.

Note: When a target format has a perpixel alpha component the color components RGB_{out} are computed as premultiplied by this alpha value. The display pipeline is aware of this when blending 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:

1. 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, for example: ARGB8888).
2. The intermediate bitmap uses the source 2 pipeline and the background picture uses the source 1 pipeline. They are blended together.

25 2D blitter display engine registers

Addresses are provided as the *BDispBaseAddress* + offset.

The *BDispBaseAddress* is:

0x2100 0000.

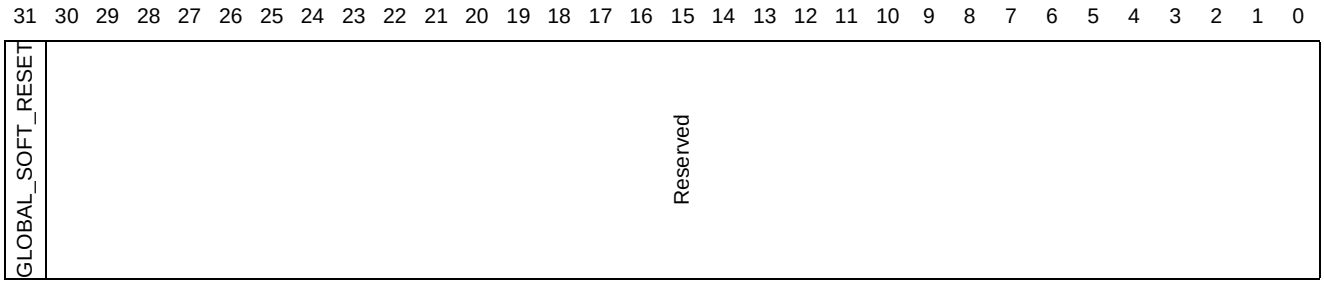
Table 79: 2D blitter display engine registers

Register name	Description	Address offset	Type
BLT_CTL	Control register, see page 248	0x0A00	R/W
BLT_ITS	Interrupt register, see page 248	0x0A04	R/W
BLT_STA	Status register, see page 249	0x0A08	R
BLT_SSBA1	Static sources base address, see page 250	0x0A10	R/W
BLT_SSBA2		0x0A14	R/W
BLT_SSBA3		0x0A18	R/W
BLT_SSBA4		0x0A1C	R/W
BLT_SSBA5		0x0A20	R/W
BLT_SSBA6		0x0A24	R/W
BLT_SSBA7		0x0A28	R/W
BLT_SSBA8		0x0A2C	R/W
BLT_SSBA9		0x0AA0	R/W
BLT_SSBA10		0x0AA4	R/W
BLT_SSBA11		0x0AA8	R/W
BLT_SSBA12		0x0AAC	R/W
BLT_SSBA13		0x0AB0	R/W
BLT_SSBA14		0x0AB4	R/W
BLT_SSBA15		0x0AB8	R/W
BLT_SSBA16		0x0ABC	R/W
BLT_STBA1	Static Target Base address, see page 250	0x0A30	R/W
BLT_STBA2		0x0A34	R/W
BLT_CQ_TRIG_IP	Composition queue trigger instruction pointer, see page 251	0x0A40	R/W
BLT_CQ_TRIG_CTL	Composition queue trigger control register, see page 251	0x0A44	R/W
BLT_CQ_IP	Composition queue instruction pointer, see page 252	0x0A4C	R
BLT_AQ_CTL	Application queue 1 control register, see page 253	0x0A60	R/W
BLT_AQ_IP	Application queue 1 instruction pointer, see page 253	0x0A64	R/W
BLT_AQ_LNA	Application queue 1 last node address, see page 254	0x0A68	R/W
BLT_AQ_STA	Application queue 1 status register, see page 254	0x0A6C	R
BLT_PRI	STBus priorities, see page 254	0x0AF8	R/W
Linked list (node) registers			
BLT_NIP	Next instruction pointer, see page 255	0x0C00	R/LLU
BLT_CIC	Current instruction content, see page 256	0x0C04	R/LLU
BLT_INS	Instruction register, see page 257	0x0C08	R/LLU
BLT_ACK	ALU and color key control, see page 258	0x0C80	R/LLU

Table 79: 2D blitter display engine registers

Register name	Description	Address offset	Type
BLT_TBA	Target base address, see page 259	0x0C10	R/LLU
BLT_TTY	Target type, see page 259	0x0C14	R/LLU
BLT_TXY	Target XY location, see page 260	0x0C18	R/LLU
BLT_T_S1_SZ	Target and source 1 window size, see page 261	0x0C1C	R/LLU
BLT_S1CF	Source 1 color fill, see page 261	0x0C20	R/LLU
BLT_S2CF	Source 2 color fill, see page 262	0x0C24	R/LLU
BLT_S1BA	Source 1 base address, see page 262	0x0C28	R/LLU
BLT_S1TY	Source 1 type, see page 263	0x0C2C	R/LLU
BLT_S1XY	Source 1 XY location, see page 264	0x0C30	R/LLU
BLT_S2BA	Source 2 base address, see page 264	0x0C38	R/LLU
BLT_S2TY	Source 2 type, see page 264	0x0C3C	R/LLU
BLT_S2XY	Source 2 XY location, see page 266	0x0C40	R/LLU
BLT_S2SZ	Source 2 window size, see page 266	0x0C44	R/LLU
BLT_S3BA	Source 3 base address, see page 267	0x0C48	R/LLU
BLT_S3TY	Source 3 type, see page 267	0x0C4C	R/LLU
BLT_S3XY	Source 3 XY location, see page 268	0x0C50	R/LLU
BLT_S3SZ	Source 3 window size, see page 268	0x0C54	R/LLU
BLT_CWO	Clipping window offset, see page 268	0x0C58	R/LLU
BLT_CWS	Clipping window stop, see page 269	0x0C5C	R/LLU
BLT_CCO	CLUT and color space conversion, see page 270	0x0C60	R/LLU
BLT_CML	CLUT memory location, see page 271	0x0C64	R/LLU
BLT_F_RZC_CTL	Filter and 2D resize control, see page 271	0x0C68	R/LLU
BLT_RSF	Resize scaling factor, see page 272	0x0C70	R/LLU
BLT_RZI	Resizer initialization, see page 272	0x0C74	R/LLU
BLT_Y_RSF	Luma resize scaling factor, see page 273	0x0C80	R/LLU
BLT_Y_RZI	Luma resizer initialization, see page 273	0x0C84	R/LLU
BLT_KEY1	Color key 1, see page 274	0x0CA0	R/LLU
BLT_KEY2	Color key 2, see page 274	0x0CA4	R/LLU

BLT_CTL Control register



Address: *BDispBaseAddress* + 0x0A00

Type: R/W

Buffer: Immediate

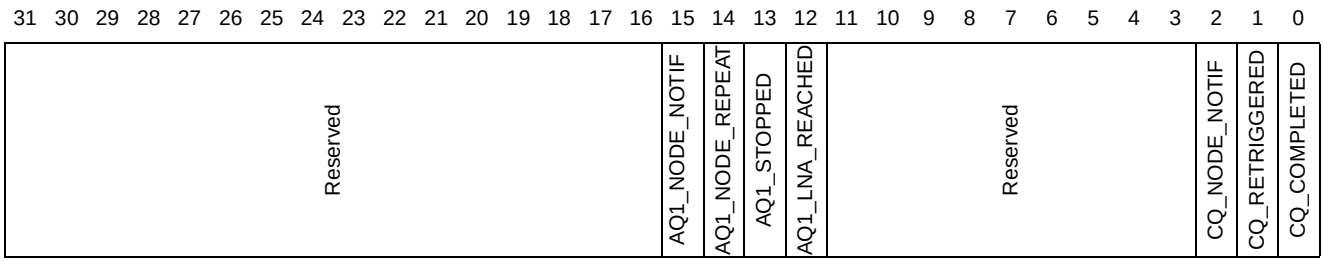
Reset: 0

Description: This register controls the 2D blitter display engine activity.

[31] **GLOBAL_SOFT_RESET**
Close all DMA, reset all state machine to idle, reset all the registers.

[30:0] **Reserved**

BLT_ITS Interrupt status register



Address: *BDispBaseAddress* + 0x0A04

Type: R/W

Buffer: Immediate

Reset: 0

Description: This register shows the activity of the 2D blitter display engine (for identifying the interrupt source, if any).

Note: *Setting the appropriate bit in this register resets the interrupt request (for example writing 0x2 in this register clears the CQ_RETRIGGERED interrupt status bit, once processed).*
Status Bits have no meaning if the corresponding interrupt is disabled (see BLT_CQ_TRIG_CTL[22:20] and BLT_AQ_CTL[22:20]).

[31:16] **Reserved**

[15] **AQ_NODE_NOTIF**
When set, indicates that a “tagged” node has been executed in AQ

[14] **AQ_NODE_REPEAT**
When set, indicates that AQ node was stopped and will be reposted

[13] **AQ_STOPPED**
When set, indicates that AQ was stopped (following EVENT_BEHAV directive)

[12] **AQ_LNA_REACHED**
When set, indicates that Application Queue is finished (Last Node was executed).

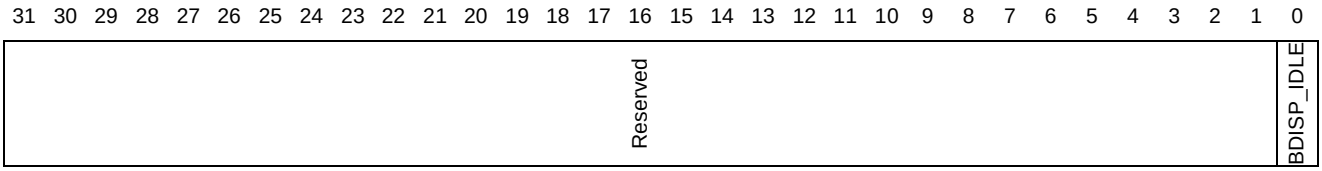
[11:3] **Reserved**

[2] **CQ_NODE_NOTIF**
When set, indicates that a “tagged” node has been executed in CQ.

[1] **CQ_RETRIGGERED**
When set, indicates that CQ was retriggered before completion

[0] **CQ_COMPLETED**
When set, indicates that composition queue is finished

BLT_STA Status register 1



Address: *BDispBaseAddress* + 0x0A08
Type: R
Buffer: Immediate
Reset: 1
Description: This register provides the status of the 2_ blitter display engine.
Note: This value is useful when a node has been suspended to enable a higher priority operation.

[31:1] **Reserved**

[0] **BDISP_IDLE**
When set, this status indicates that the 2D blitter display engine is idle (no node to execute on any queue)

BLT_SSBAn Static source base address 1 to 16

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
SSBA1_BANK_NUM	SSBA1_MEMORY_ADDRESS
SSBA2_BANK_NUM	SSBA2_MEMORY_ADDRESS
SSBA3_BANK_NUM	SSBA3_MEMORY_ADDRESS
SSBA4_BANK_NUM	SSBA4_MEMORY_ADDRESS
SSBA5_BANK_NUM	SSBA5_MEMORY_ADDRESS
SSBA6_BANK_NUM	SSBA6_MEMORY_ADDRESS
SSBA7_BANK_NUM	SSBA7_MEMORY_ADDRESS
SSBA8_BANK_NUM	SSBA8_MEMORY_ADDRESS
SSBA9_BANK_NUM	SSBA9_MEMORY_ADDRESS
SSBA10_BANK_NUM	SSBA10_MEMORY_ADDRESS
SSBA11_BANK_NUM	SSBA11_MEMORY_ADDRESS
SSBA12_BANK_NUM	SSBA12_MEMORY_ADDRESS
SSBA13_BANK_NUM	SSBA13_MEMORY_ADDRESS
SSBA14_BANK_NUM	SSBA14_MEMORY_ADDRESS
SSBA15_BANK_NUM	SSBA15_MEMORY_ADDRESS
SSBA16_BANK_NUM	SSBA16_MEMORY_ADDRESS

Address: *BDispBaseAddress* + 0x10 (SSBA1), 0x14 (SSBA2), 0x18 (SSBA3), 0x1C (SSBA4), 0x20 (SSBA5), 0x24 (SSBA6), 0x28 (SSBA7), 0x2C (SSBA8), 0xA0 (SSBA9), 0xA4 (SSBA10), 0xA8 (SSBA11), 0xAC (SSBA12), 0xB0 (SSBA13), 0xB4 (SSBA14), 0xB8 (SSBA15), 0xBC (SSBA16)

Type: R/W

Buffer: Double buffered on trig event.

Reset: 0

Description: These registers provides static base addresses selectable by any source to be substituted to the node register BLT_SxBA.

[31:26] **SSBAn_BANK_NUM**
64 Mbyte bank number

[25:0] **SSBAn_MEM_ADDR**
Static memory address

BLT_STBAn Static target base address 1 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
STBA0_BANK_NUM	STBA0_MEMORY_ADDRESS
STBA1_BANK_NUM	STBA1_MEMORY_ADDRESS

Address: *BDispBaseAddress* + 0x0A30 (STBA1), 0x0A34 (STBA2)

Type: R/W

Buffer: Double buffered on trig event

Reset: 0

Description: These registers provides static base addresses selectable by the target to be substituted to the node register BLT_TBA.

[31:26] **STBAn_BANK_NUM**
64 Mbyte bank number

[25:0] **STBAn_MEM_ADDR**
Static memory address

BLT_CQ_TRIG_IP CQ trigger 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
TRIG_BANK_NUM						CQ_TRIG_MEMORY_ADDRESS																										

Address: *BDispBaseAddress* + 0x0A40

Type: R/W

Buffer: Immediate

Reset: 0

Description: This register provides the memory location of the next instruction to execute when composition queue n is first triggered.

[31:26] **TRIG_BANK_NUM**
64 Mbyte bank number

[25:0] **CQ_TRIG_MEM_ADDR**
Static memory address

BLT_CQ_TRIG_CTL CQ trigger control register

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

Address: *BDispBaseAddress* + 0x0A44

Type: R/W

Buffer: Immediate

Reset: 0

Description: This register controls composition queue n activity.

[31] **QUEUE_EN**
When set, composition queue n is enabled

[30:28] **Reserved**

[27:24] **RETRIG_BEHAV**
Queue behavior on retrig (exclusive behaviors)
0000: Finish current queue
0010: Suspend and start next queue (that is complete current node and start next queue)
0100: Abort and start next queue (that is Complete current line, and start next queue)
1000: Stop this queue (that is Complete cur line, and wait for CPU to relaunch

[23:20] **IRQ_MASK**
IRQ enable mask
0000: Queue interrupt disabled
0001: Enable queue completed interrupt
0010: Enable queue retriggered interrupt
0100: Enable node completed interrupt (node based, only tagged nodes generate this IRQ)
Others: Reserved

[19:16] **Reserved**

[15:12] **TRIG_COND**
Trigger condition control
0000: No trig
0001: Wait for line count event in top field
0010: Wait for VSYNC top field
0011: Reserved
0100: Reserved
0101: Wait for line count event in bottom field
0110: Wait for Vsync bottom field
0011: Reserved
1xxx: Soft trigger. Bit [15] is automatically reset to 0.

[11] **Reserved**

[10:0] **TRIG_LINE_NUM**
Expected line to trigger (when trig cond is 001 and 101)

BLT_CQ_IP CQ instruction pointer

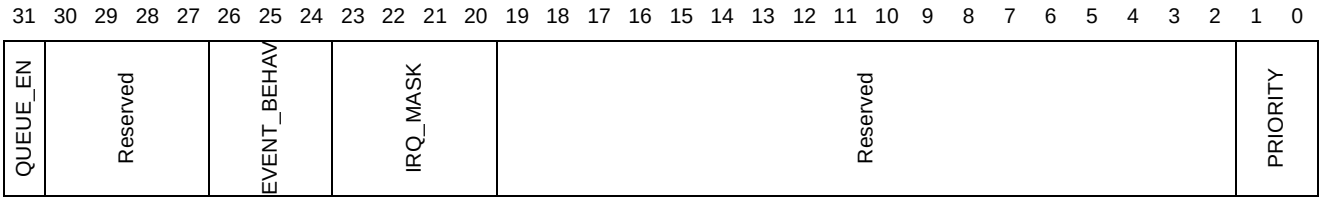
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INST_BANK_NUM						INST_MEMORY_ADDRESS																									

Address: *BDispBaseAddress* + 0x0A4C
Type: R
Buffer: Immediate
Reset: 0
Description: This register provides the memory address of the current operation in composition queue n.

[31:26] **INST_BANK_NUM**
64 Mbyte bank number

[25:0] **INST_MEM_ADDR**
Static memory address

BLT_AQ_CTL AQ control register



Address: *BDispBaseAddress* + 0x0A60

Type: R/W

Buffer: Immediate

Reset: 0

Description: This register controls the Application Queue.

[31] **QUEUE_EN**
When set, Application queue is enabled

[30:27] **Reserved**

[26:24] **EVENT_BEHAV**

Behavior on application queue event (Trig /soft trigger)

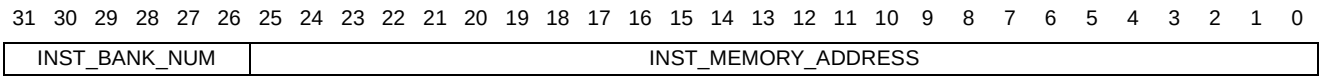
000: Suspend current node (default): finish node execution before releasing core to caller.
010: Abort current node (finish current line, and report current node as first node to execute)
100: Stop this queue (disable queue, CPU will need to be relaunched)

[23:20] **IRQ_MASK**
Trigger condition control
0000: Interrupt disabled
0001: Node repeat interrupt (that is when abort node behavior is selected)
0010: Queue stopped interrupt (that is when stop queue behavior is selected)
0100: LNA reached interrupt
1000: Enable node completed interrupt (Node based, only tagged nodes generate this IRQ

[19:2] **Reserved**

[1:0] **PRIORITY**
Queue priority for AQ arbitration
00: Application queue priority is 0
01: Application queue priority is 1
10: Application queue priority is 2
11: Application queue priority is 3
Note: The highest priority value cannot exceed the number of application queues.

BLT_AQ_IP AQ instruction pointer



Address: *BDispBaseAddress* + 0x0A64

Type: R/W

Buffer: Immediate

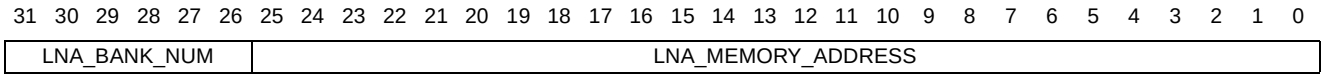
Reset: 0

Description: This register provides the memory address of the current operation in the application queue.

[31:26] **INST_BANK_NUM**
64 Mbyte bank number

[25:0] **INST_MEM_ADDR**
Static memory address

BLT_AQ_LNA AQ last node address



Address: *BDispBaseAddress* + 0x0A68

Type: R/W

Buffer: Immediate

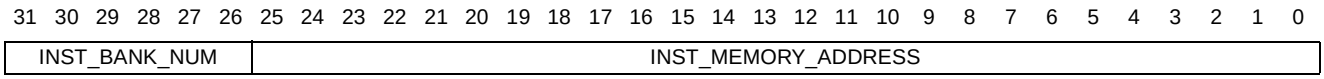
Reset: 0

Description: This register provides the memory address of the last operation to execute in the application queue.

[31:26] **LNA_BANK_NUM**
64 Mbyte bank number

[25:0] **LNA_MEMORY_ADDR**
Static memory address

BLT_AQ_STA AQ status register



Address: *BDispBaseAddress* +0x0A6C

Type: R

Buffer: Immediate

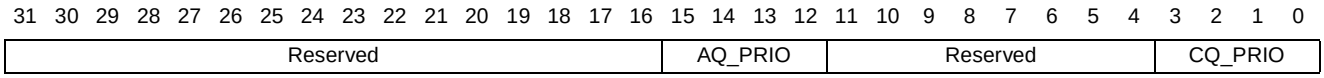
Reset: 0

Description: This register provides the address.

[31:26] **INST_BANK_NUM**
64 Mbyte bank number

[25:0] **INST_MEMORY_ADDR**
Static memory address

BLT_PRI Queue priorities



Address: *BDispBaseAddress* +0x0AF8

Type: R/W

Buffer: Immediate

Reset: 0

Description: This register provides the queue priority when AQ or CQ is active.

[31:16] **Reserved**

[15:12] **AQ_PRIO**
Priority when AQ is active

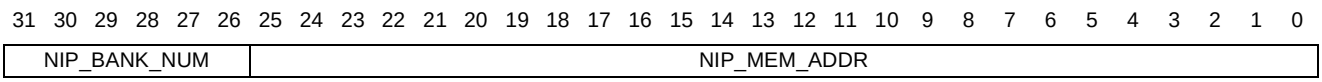
[11:4] **Reserved**

[3:0] **CQ_PRIO**
Priority when CQ is active

25.1 Linked list (node) registers

Note: A node address must be aligned to the source 1 STBus interface width.

BLT_NIP Next instruction pointer



Address: *BDispBaseAddress* + 0x0C00

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the memory location of the next instruction to execute.
The 2D blitter display engine stops when this register is 0 (last node of the link-list).

[31:26] **NIP_BANK_NUM**
64 Mbyte bank number

[25:0] **NIP_MEM_ADDR**
Memory address for the next instruction pointer for current operation

BLT_CIC Current instruction content

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_STBA		S3_SSBA				S2_SSBA				S1_SSBA				Reserved					GROUP_12	Reserved	GROUP_10	GROUP_9	GROUP_8	GROUP_7	GROUP_6	GROUP_5	GROUP_4	GROUP_3	GROUP_2	Reserved	

Address: *BDispBaseAddress* + 0x0C04

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register contains the logical groups present in the current node.

[31:30] **T_STBA**
Indicates which STBA register to use with target:
00: STBA not used (target address is standard BLT_TBA register)
01: Target base address is to be taken in STBA1 register
10: Target base address is to be taken in STBA2 register
10: Reserved

[29:26] **S3_SSBA**
Indicates which SSBA register to use with source3:
0000: SSBA not used (source 3 address is standard BLT_S3BA register)
0001: Source 3 base address is to be taken in SSBA1 register
xxxx: Source 3 base address is to be taken in SSBAn register

[25:22] **S2_SSBA**
Indicates which SSBA register to use with source2:
0000: SSBA not used (source 2 address is standard BLT_S2BA register)
0001: Source 2 base address is to be taken in SSBA1 register
xxxx: Source 2 base address is to be taken in SSBAn register

[21:18] **S1_SSBA**
Indicates which SSBA register to use with source1:
0000: SSBA not used (source 1 address is standard BLT_S1BA register)
0001: Source 1 base address is to be taken in SSBA1 register
xxxx: Source 1 base address is to be taken in SSBAn register

[17:13] **Reserved**

[12] **GROUP_12**: Color key is present or not in loaded words

[11] **Reserved**

[10] **GROUP_10**: Luma filter is present or not in loaded words

[9] **GROUP_9**: RGB / chroma filter is present or not in loaded words

[8] **GROUP_8**: Filter CTL is present or not in loaded words

[7] **GROUP_7**: CLUT is present or not in loaded words

[6] **GROUP_6**: Clipping window is present or not in loaded words

[5] **GROUP_5**: Source 3 is present or not in loaded words

[4] **GROUP_4**: Source 2 is present or not in loaded words

[3] **GROUP_3**: Source 1 is present or not in loaded words

[2] **GROUP_2**: Color fill is present or not in loaded words

[1:0] **Reserved**

BLT_INS Instruction register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLITCOMPIRQ	Reserved																Reserved		OCSC	Reserved	RECT_CLIP	FILCK_FILT	2DRESCALE	CLUTOP	ICSC	SOURCE3	SOURCE2		SOURCE1		

Address: *BDispBaseAddress* + 0x0C08

Type: R/LLU

Buffer: immediate

Reset: 0

Description: This register controls the data flow configuration of the 2D blitter display engine, and the static base addresses indirections.

[31] **BLITCOMPIRQ**
When set to 1, this node generates an IRQ once completed (if enabled)

[30:15] **Reserved**

[14:13] **Reserved**

[12] **OCSC**
Enable output color space converter

[10] **RECT_CLIP**
Enable rectangular clipping

[9] **FLICK_FILT**
Enable flicker filter

[8] **2DRESCALE**
Enable 2D-rescaling engine

[7] **CLUTOP**
Enable CLUT-based operator

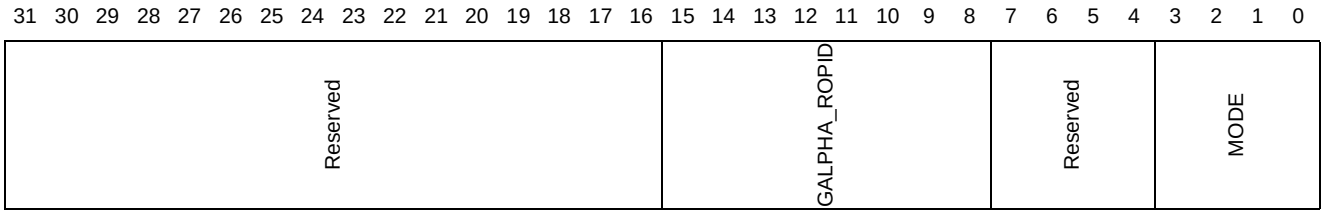
[6] **ICSC**
Enable input color space converter

[5] **SOURCE3**
Source 3 mode
0: Source 3 disabled
1: Source 3 fetched from memory

[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_ACK ALU and color key control



Address: *BDispBaseAddress* + 0x0C80

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register configures the ALU and the color key module.

[31:16] **Reserved**

[15:8] **GALPHA_ROPID**
ALU global alpha (8 bits / 0:128), 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
All non used bits should be set to 0 (i.e GALPHA_ROPID[7:4] in logical operations and all GALPHA_ROPID bits in bypass mode)

[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
0111: Bypass source 2
Others: Reserved

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: *BDispBaseAddress* + 0x0C10

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the byte memory location of the target pixmap (pixel (0,0)), for storing the result of the operation (target data). Pixel (0,0) is always the top-left pixel.

Note: In both XYL standard and transform modes, this register must be programmed with the base address of the target bitmap.

[31:26] **TBA_BANK_NUM**
64 Mbyte bank number

[25:0] **TBA_MEM_ADDR**
Byte address of the first pixel

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			SUBBYTE	CHROMA_NOTLUMA	RGB_ROUND	VSO	HSO	MB_FIELD	Reserved	ALPHA_RANGE	COLOR_FORM					PIXMAP_PITCH															

Address: *BDispBaseAddress* + 0x0C14

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register configures the color format, pitch and raster scan order for the target pixmap.

[31:29] **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] **CHROMA_NOTLUMA**
Component selection to be written into a macro-block organized frame buffer (YCbCr420MB and YCbCr422MB) *Not yet implemented*
0: Write luma component
1: Write chroma components

[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] **MB_FIELD**
Write access mode in macro-block organized frame buffers (YCbCr420MB and YCbCr422MB).*Not yet implemented*
0: Access in frame mode
1: Access in field mode (every other line)

[22] **Reserved**

[21] **ALPHA_RANGE**
8-bit alpha range (for AYCbCr8888, ARGB8565, ARGB8888, ACLUT88 and A8 only)
0: 0:128 (128 means opaque)
1: 0:255 (255 means opaque)

[20:16] **COLOR_FORM**
Pixmap color format
RGB types
00000: RGB565
00001: RGB888
00100: ARGB8565
00101: ARGB8888
00110: ARGB1555
00111: ARGB4444
CLUT types
01000: CLUT1
01001: CLUT2
01010: CLUT4
01011: CLUT8
01100: ACLUT44
01101: ACLUT88
YCbCr types
10000: YCbCr888
10010: YCbCr422R
10101: AYCbCr8888
10011: YCbCr422MB
10100: YCbCr420MB

[15:0] **PIXMAP_PITCH**
Pixmap pitch in byte unit

Note: In both XYL standard and transform modes, 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_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: *BDispBaseAddress* + 0x0C18
Type: R/LLU
Buffer: Immediate
Reset: 0
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).
Note: In XYL transform mode, this register must be filled with the XY coordinates of the left pixel of the target horizontal line.

In XYL standard mode, this register is unused.

Description:

[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)

BLT_T_S1_SZ Target and source 1 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: *BDispBaseAddress* + 0x0C1C

Type: R/LLU

Buffer: Immediate

Reset: 0

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.

Note: In XYL transform mode, this register must be filled with the width (in pixel unit), of the target horizontal line. The height parameter should be set to 1.

In XYL standard mode, this register is not used.

[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

BLT_S1CF Source 1 color fill

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1_COLOR_FILL																															

Address: *BDispBaseAddress* + 0x0C20

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the solid color value to be used for filling or shading from source 1, during an 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_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: *BDispBaseAddress* + 0x0C24

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the solid color value to be used for filling or shading from source 2, during an 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: A source 2 color-fill operation can be done only for raster formats.

BLT_S1BA Source 1 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S1BA_BANK_NUM						S1BA_MEM_ADDR																									

Address: *BDispBaseAddress* + 0x0C28

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the byte memory location of the pixmap (pixel (0,0)), for retrieving graphics data for source 1. Pixel (0,0) is always the upper-left pixel.

Note: In XYL standard mode, this register must be programmed with the base address of the target bitmap.

In XYL transform mode, this register must be programmed with the base address of the source bitmap.

[31:26] **S1BA_BANK_NUM**
64Mbyte bank number

[25:0] **S1BA_MEM_ADDR**
Byte address of the first pixel

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		RGB_EXP	SUBBYTE	Reserved		VSO	HSO	Reserved	A1_SUBST	ALPHA_RANGE	COLOR_FORM					PIXMAP_PITCH															

Address: *BDispBaseAddress* + 0x0C2C

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register configures the color format, pitch and raster scan order for the source 1 pixmap.

[31:30] **Reserved**

[29] **RGB_EXP**
RGB expansion mode:
0: MSBs are duplicated on missing LSBs
1: Missing LSBs are filled with 0

[28] **SUBBYTE**
Sub-byte formats, pixels ordering
0: Screen most right pixel in most significant bits
1: Screen most right pixel in LSBs

[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] **Reserved**

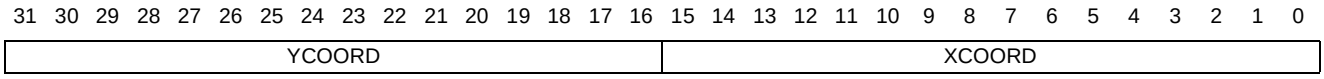
[22] **A1_SUBST**
When in ARGB1555 mode, A1 value substituted between two A8 value
0: A1=0 means transparent alpha, and A1=1 means opaque alpha
1: A1=0 select KEY1.ALPAH0 as alpha value, and A1=1 select KEY2.ALPHA1

[21] **ALPHA_RANGE**
8-bit alpha range (for AYCbCr8888, ARGB8565, ARGB8888, ACLUT88 and A8 only)
0: 0:128 (128 means opaque)
1: 0:255 (255 means opaque)

[20:16] **COLOR_FORM**
Pixmap color format
RGB types
00000: RGB565
00001:RGB888
00100:ARGB8565
00101:ARGB8888
00110:ARGB1555
00111:ARGB4444
CLUT types
01000: CLUT1
01001: CLUT2
01010: CLUT4
01011: CLUT8
01100: ACLUT44
01101: ACLUT88
YCbCr types
10000: YCbCr888
10010: Reserved
10101: AYCbCr8888

[15:0] **PIXMAP_PITCH**
Pixmap pitch in byte unit

BLT_S1XY Source 1 XY location



Address: *BDispBaseAddress* + 0x0C30

Type: R/LLU

Buffer: Immediate

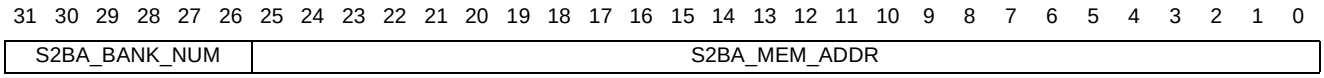
Reset: 0

Description: This register gives the XY location of the first pixel to be transferred, with reference to the top-left corner of the 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_S2BA Source 2 base address



Address: *BDispBaseAddress* + 0x0C38

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the byte memory location address of the pixmap (pixel (0,0)), for retrieving graphics data for source 2. Pixel (0,0) is always the upper-left pixel.

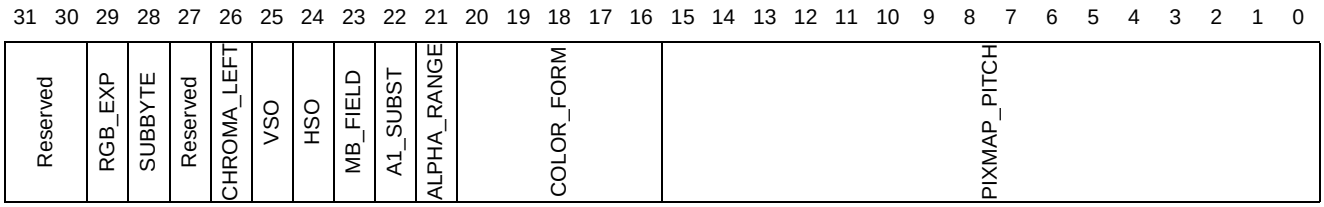
Note: *In XYL standard mode, this register must be programmed with the base address of the clipmask bitmap, if any.*

In XYL transform mode, this register is unused.

[31:26] **S2BA_BANK_NUM**
64Mbyte bank number

[25:0] **S2BA_MEM_ADDR**
Byte address of the first pixel

BLT_S2TY Source 2 type



Address: *BDispBaseAddress* + 0x0C3C

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register configures the color format, pitch and raster scan order for the source 2 pixmap.

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.
In XYL transform mode, this register is unused.

[31:30] **Reserved**

[29] **RGB_EXP:** RGB expansion mode:
0: MSBs are duplicated on missing LSBs 1: Missing LSBs are filled with 0

[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
This bit is meaningful only when accessing sub-byte formats (CLUT1,2 and 4 and A1). For any other format, it should remain set to 0.

[27] **Reserved**

[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)
When the previous sample does not exist, this bit must obviously be set to 0. This remark applies also to color fill operation.

[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)
This bit is meaningful only for MacroBlocks format (420MB, 422MB and HDpip). For any other format, it should remain set to 0.

[22] **A1_SUBST**
When in ARGB1555 mode, A1 value substituted between two A8 value
0: A1=0 means transparent alpha, and A1=1 means opaque alpha
1: A1=0 select KEY1.ALPAH0 as alpha value, and A1=1 select KEY2.ALPHA1

[21] **ALPHA_RANGE**
8-bit alpha range (for AYCbCr8888, ARGB8565, ARGB8888, ACLUT88 and A8 only)
0: 0:128 (128 means opaque)
1: 0:255 (255 means opaque)

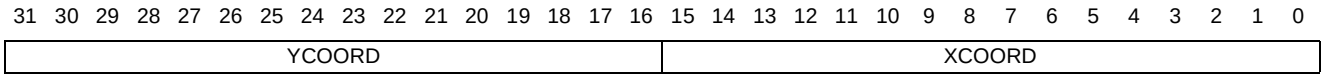
[20:16] **COLOR_FORM**
Pixmap color format

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	10010: YCbCr422R
10101: AYCbCr8888	10011: YCbCr422MB
10100: YCbCr420MB	

For modes YCbCr4;2;2MB and YCbCr4;2;0MB (macroblock based, dual buffer), source 3 and source 2 are both used, and must be programmed with the same color format

[15:0] **PIXMAP_PITCH**
Pixmap pitch in byte unit

BLT_S2XY Source 2 XY location



Address: *BDispBaseAddress* + 0x0C40

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register gives the XY location of the first pixel to be transferred, with reference to the top-left corner of the source 2 bitmap (0,0). This XY location depends on the programmed horizontal and vertical scan order (register BLT_S2TY).

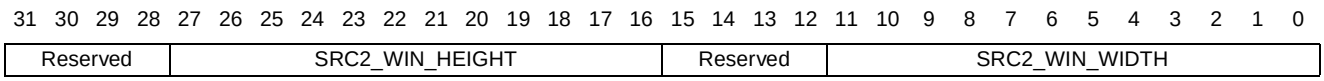
Note: In XYL standard mode, this register contains the 2D-vector that gives the location of the (0,0) clipmask value, with respect to the (0,0) pixel location of the target bitmap. Bits[15:0] provide the horizontal offset, bits[31:16] provide the vertical offset (both values are 16-bit signed integers).

In XYL transform mode, this register is unused.

[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_S2SZ Source 2 window size



Address: *BDispBaseAddress* + 0x0C44

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register gives the width and height of the rectangular window to be transferred.

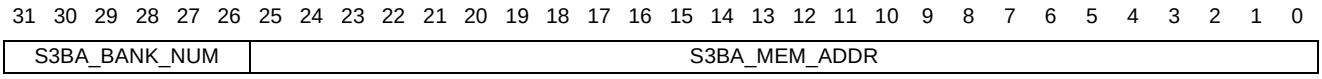
[31:28] **Reserved**

[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_S3BA Source 3 base address



Address: *BDispBaseAddress* + 0x0C48

Type: R/LLU

Buffer: Immediate

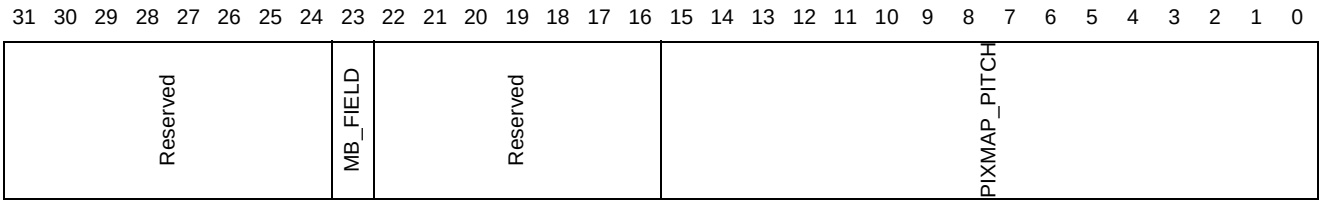
Reset: 0

Description: This register provides the byte memory location address of the pixmap (pixel (0,0)), for retrieving graphics data for source 3. Pixel (0,0) is always the upper-left pixel.

[31:26] **S3BA_BANK_NUM**
64 Mbyte bank number

[25:0] **S3BA_MEM_ADDR**
Byte address of the first pixel

BLT_S3TY Source 3 type



Address: *BDispBaseAddress* + 0x0C4C

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register configures the pitch for the source 3 pixmap.

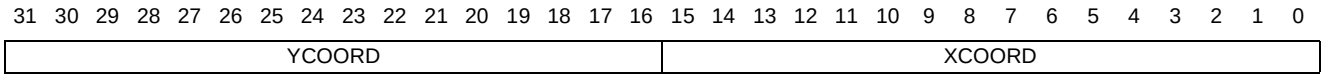
[31:24] **Reserved**

[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)

[22:16] **Reserved**

[15:0] **PIXMAP_PITCH**
Pixmap pitch in byte unit

BLT_S3XY Source 3 XY location



Address: *BDispBaseAddress* + 0x0C50

Type: R/LLU

Buffer: Immediate

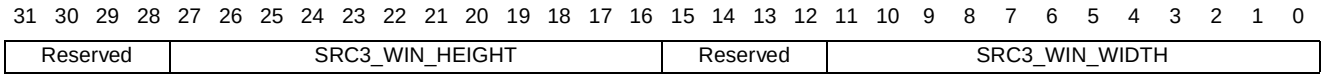
Reset: 0

Description: This register gives the XY location of the first pixel to be transferred, with reference to the top-left corner of the source 3 bitmap (0,0). This XY location depends on the programmed horizontal and vertical scan order (register BLT_S3TY).

[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_S3SZ Source 3 window size



Address: *BDispBaseAddress* + 0x0C54

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register gives the width and height of the rectangular window to be transferred.

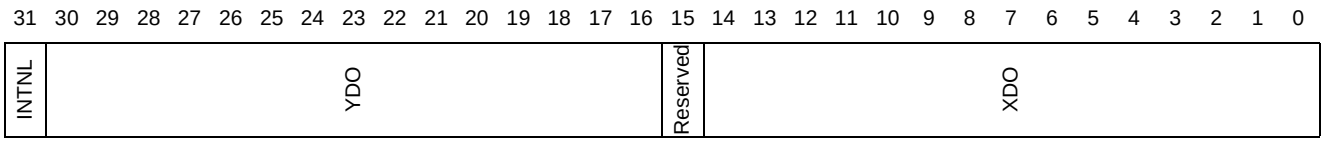
[31:28] **Reserved**

[27:16] **SRC3_WIN_HEIGHT**
Height (in lines) of source 3 window to be transferred

[15:12] **Reserved**

[11:0] **SRC3_WIN_WIDTH**
Width (in pixels) of source 3 window to be transferred

BLT_CWO Clipping window offset



Address: *BDispBaseAddress* + 0x0C58

Type: R/LLU

Buffer: Immediate

Reset: 0

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).

Note: In both XYL standard and transform 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.

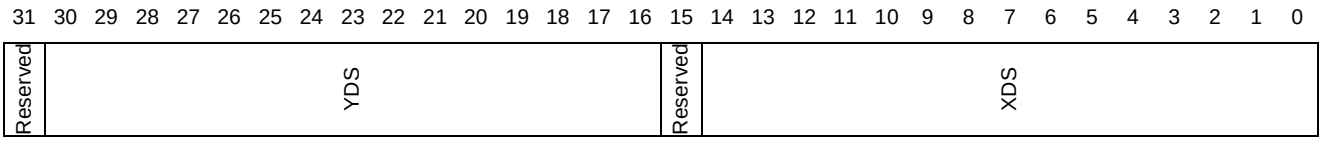
[31] **INTNL**
Internal or external clipping
0: Writing is only allowed inside the window (including the borders)
1: Writing is only allowed outside the window (excluding the borders)

[30:16] **YDO**
Y coordinate (unsigned)

[15] **Reserved**

[14:0] **XDO**
X coordinate (unsigned)

BLT_CWS Clipping window stop



Address: *BDispBaseAddress* + 0x0C5C

Type: R/LLU

Buffer: Immediate

Reset: 0

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).

Note: In both XYL standard and transform modes, 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.

[31] **Reserved**

[30:16] **YDS:** Y coordinate (unsigned)

[15] **Reserved**

[14:0] **XDS:** X coordinate (unsigned)

BLT_CCO CLUT and color conversion operators

31:19	18	17:16	15	14:11	10	9	8	7:4	3	2	1	0
Reserved	CLUT_UPDATE	CLUT_MODE	CLUT_NS2_S1	Reserved	CCO_OUTGFXnVID	CCO_OUTSIGN	CCO_OUTCOL	Reserved	CCO_INVIDnGFX	CCO_INDIR	CCO_INSIGN	CCO_INCOL

Address: *BDispBaseAddress* + 0x0C60

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register controls the two color space converters and the CLUT operator.

Note: The CLUT and the input color space converter cannot be used in XYL mode. But the output color space converter is available.

[31:19] **Reserved**

CLUT operators

[18] **CLUT_UPDATE: Enable CLUT update**

[17:16] **CLUT_MODE**: CLUT operation modes:
00: Color expansion mode ((A)CLUTn >> true color)
01: Color correction (true color >> true color) (CLUTn >> CLUTm)
10: Reserved

[15] **CLUT_NS2_S1**: CLUT Source for color expansion/correction:
0: Color expansion/correction applied to S2
1: Color expansion/correction applied to S1

[14:11] **Reserved**

Color-space converters

[10] **CCO_OUTVIDnGFX:** Output color space converter uses
0: Graphic matrix 1: Video matrix

[9] **CCO_OUTSIGN:** Output color converter chroma format
0: Offset binary 1: 2'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_INVIDnGFX:** Input color space converter uses
0: Graphic matrix
1: Video matrix

[2] **CCO_INDIR:** Input color converter direction
0: YCbCr2RGB 1: RGB2YCbCr

[1] **CCO_INSIGN:** Input color converter chroma format
0: Offset binary 1: 2's complement, signed

[0] **CCO_INCOL:** Input color converter colorimetry
0: ITU-R BT.601 1: ITU-R BT.709

BLT_CML CLUT memory location

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

Address: *BDispBaseAddress* + 0x0C64

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the byte memory location for retrieving 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 Mbyte bank number

[25:4] **CML_MEM_ADDR**
Byte address

[3:0] **Reserved**

BLT_F_RZC_CTL Filter and 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		LUMA_2DVF_MODE		Reserved		LUMA_2DHF_MODE		Reserved				FF_REPEAT_FL	FF_REPEAT_LL	Reserved		AB_TOP	AB_BOT	AB_LEFT	AB_RIGHT	Reserved		FF_MODE		Reserved		2DVF_MODE		Reserved		2DHF_MODE	

Address: *BDispBaseAddress* + 0x0C68

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register configures the 2D resize engine.

[31:30] **Reserved**

[29:28] **LUMA_2DVF_MODE:** Luma path 2D-filter mode (vertical):
00: Vertical resizer disabled
01: Reserved
10: V.resizer enabled Vfilter inactive
11: V.resizer enabled Vfilter active

[27:26] Reserved

[25:24] **LUMA_2DHF_MODE:** Luma path 2D-filter mode (horizontal):
00: Horizontal resizer disabled
01: Reserved
10: H.resizer enabled Hfilter inactive
11: H.resizer enabled Hfilter active

[23:20] Reserved

[19] **FF_REPEAT_FL@:** When set, FF repeats the first incoming line to generate first output line

[18] **FF_REPEAT_LL:** When set, FF repeats the last incoming line to generate last output line

[17:16] **Reserved**

[15] **AB_TOP:** Enable AlphaHBorder on top side of the rectangle

[14] **AB_BOT:** Enable AlphaHBorder on Bottom side of the rectangle

[13] **AB_LEFT:** Enable AlphaVBorder on left side of the rectangle

[12] **AB_RIGHT:** Enable AlphaVBorder on right side of the rectangle

[11:10] Reserved

[9:8] **FF_MODE:** Flicker filter mode
00: Force filter 0
01: Adaptive flicker filtering (RGB format only)
10: Test mode
11: Reserved
When using a fixed coefficient flicker filter (1/4:1/2:1/4), this bit field is of course useless.

[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 Vfilter active on both color and alpha channels

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: *BDispBaseAddress* + 0x0C70
Type: R/LLU
Buffer: Immediate
Reset: 0
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)
Maximum INC value = 0xFC00 and minimum INC value equal 0x0001

[15:0] **HSRC_INC**
Increment value for HSRC (6.10 format)
Maximum INC value = 0xFC00 and minimum INC value equal 0x0001

BLT_RZI Resizer initialization

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

Address: *BDispBaseAddress* + 0x0C74
Type: R/LLU
Buffer: Immediate
Reset: 0
Description: This register provides the scaling factors for both the horizontal and the vertical sample rate converters.

Note: Due to hardware limitations, VNB_REPEAT maximum value is 3, and HNB_REPEAT maximum value is 7.

[31] **Reserved**

[30:28] **VNB_REPEAT**
Number of time first line is repeated when initializing V filter (maximum value = 3)

[27:26] **Reserved**

[25:16] **VSRC_INIT**
Initial value for VSRC decimal part (6.10 format)

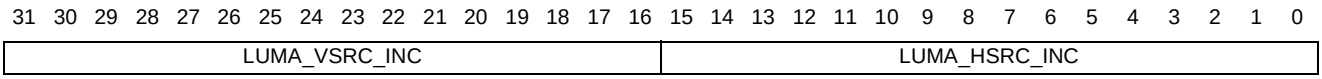
[15] **Reserved**

[14:12] **HNB_REPEAT**
Number of time first pixel is repeated when initializing H filter (maximum value = 7)

[11:10] **Reserved**

[9:0] **HSRC_INIT**
Initial value for HSRC decimal part (6.10 format)

BLT_Y_RSF Luma resize scaling factor



Address: *BDispBaseAddress* + 0x0C80

Type: R/LLU

Buffer: Immediate

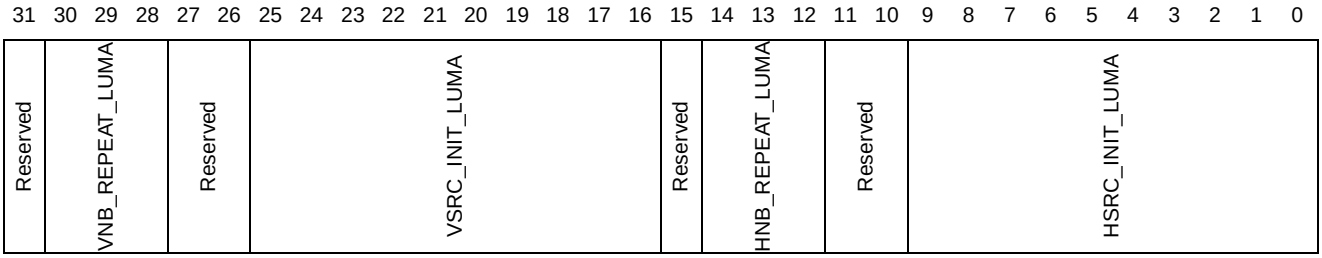
Reset: 0

Description: This register provides the scaling factors for both the horizontal and the vertical sample rate converters.

[31:16] **LUMA_VSRC_INC**
Increment value for VSRC on luma path (6.10 format)
Maximum INC value = 0xFC00 and Minimum INC value equal 0x0001

[15:0] **LUMA_HSRC_INC**
Increment value for HSRC on luma path (6.10 format)
Maximum INC value = 0xFC00 and Minimum INC value equal 0x0001

BLT_Y_RZI Luma resizer initialization



Address: *BDispBaseAddress* + 0x0C84

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register provides the scaling factors for both the horizontal and the vertical sample rate converters.

Note: Due to hardware limitations, VNB_REPEAT_LUMA maximum value is 3, and HNB_REPEAT_LUMA maximum value is 7.

[31] **Reserved**

[30:28] **VNB_REPEAT_LUMA**
Number of time first line is repeated when initializing V filter. (maximum value = 3)

[27:26] **Reserved**

[25:16] **VSRC_INIT_LUMA**
Initial value for VSRC decimal part (6.10 format)

[15] **Reserved**

[14:12] **HNB_REPEAT_LUMA**
Number of time first pixel is repeated when initializing H filter (maximum value = 7)

[11:10] **Reserved**

[9:0] **HSRC_INIT_LUMA**
Initial value for HSRC decimal part (6.10 format)

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
ALPHA0								Reserved																							

Address: *BDispBaseAddress* + 0x0CA0

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register contains the color key minimum values for the RGB components.

[31:24] **ALPHA0**
8 bit alpha value, used when source 1 or source 2 in ARGB1555 mode, A1 = 0 and A1_SUBST = 1

[23:0] **Reserved**

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
ALPHA1								Reserved																							

Address: *BDispBaseAddress* + 0x0CA4

Type: R/LLU

Buffer: Immediate

Reset: 0

Description: This register contains the color key maximum values for the RGB components.

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). It is therefore possible to track a single index, as well as a range of indexes.

[31:0] **ALPHA1**
8 bit alpha value, used when source 1 or source 2 in ARGB1555 mode, A1 = 0 and A1_SUBST = 1

Description:

26 TV out

26.1 Overview

The TV out module processes the YCbCr 444 video and graphics data in memory to provide the following digital video outputs.

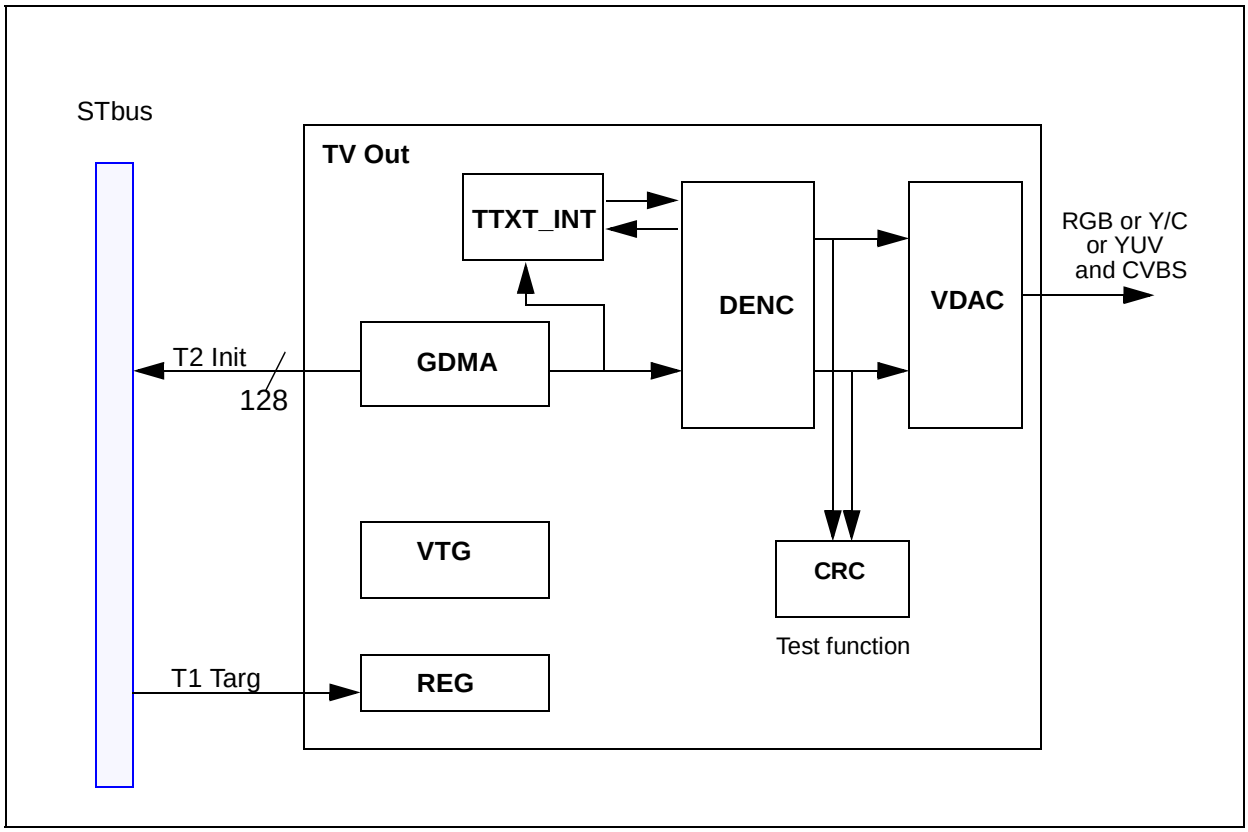
- PAL/NTSC/SECAM encoding with color control processing.
- Macrovision 6.1/7.01 copy protection, enabled/disabled by external anti-fuse.
- Teletext re-insertion in vertical blanking interval.
  - VBI WSS, DVB teletext.
  - VBI closed caption for CMGS-A transmission.
- 10-bit quad video DAC to output the following combinations.
  - CVBS + RGB or YUV.
  - S-VHS(Y/C) + CVBS + CVBS.

All video outputs have the same video content.

The TV out subsystem integrates the DENC, DAC, VTG, TTXT_INT and GDMA. Figure 95 shows the video and graphics dataflow for the STx5119 from memory to output. The process is controlled by the video timing generator (VTG) that provides video synchronization signals to the TV out submodules and to the blitter and CPU.

Note: As shown in Figure 97: TV out block diagram on page 277, the DENC DAC6 is connected to the video DAC4 output.

Figure 95: TV out architecture



The following submodules are integrated into the TV out block:

- graphics DMA, see Chapter 28: *Graphics DMA (GDMA)* on page 282,
- digital encoder, see Chapter 30: *Digital encoder (DENC)* on page 293,
- video timing generator, see Chapter 32: *Video timing generator (VTG)* on page 352,
- video DAC,
- Teletext Interface (TTXT_INT) see Section 26.2: *Teletext* on page 278.
- CRC

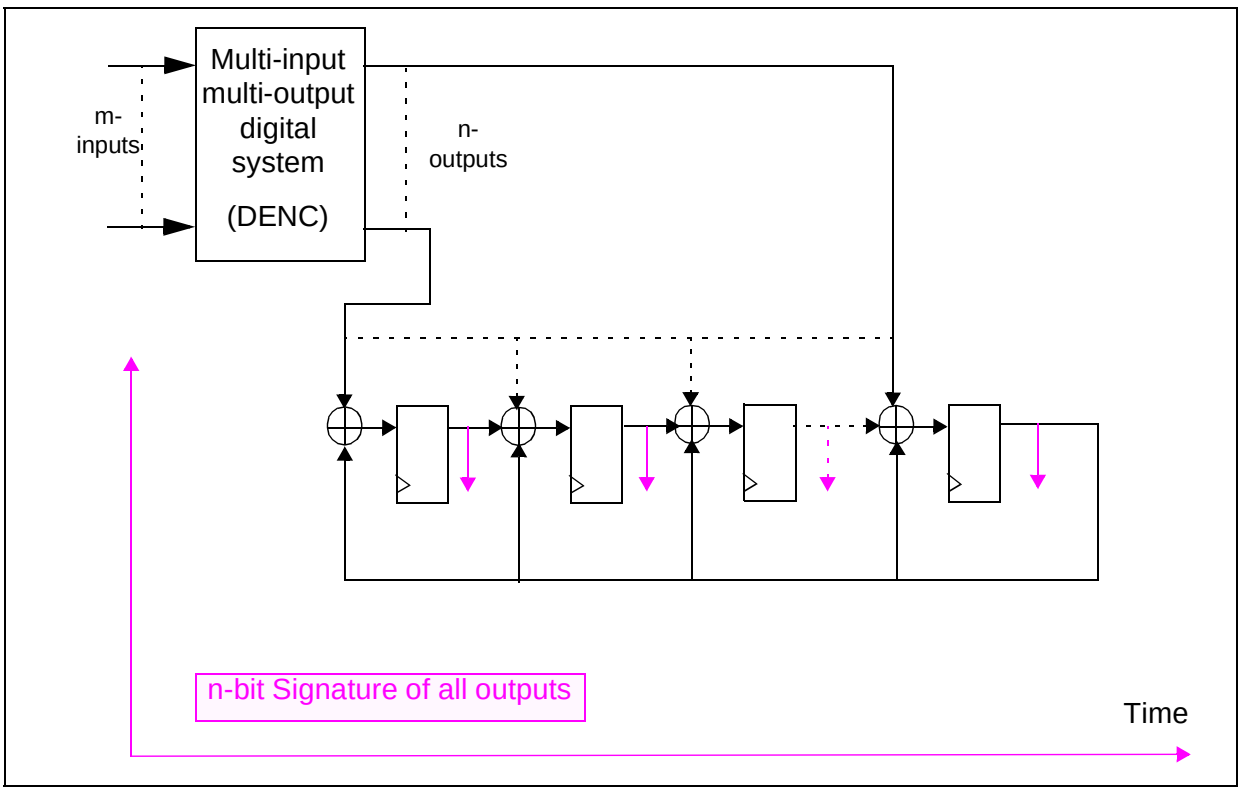
The CRC block is used for compression of the digital output response. The general structure of a CRC is shown in Figure 96. The CRC is an internal-XOR type 24-bit polynomial:

$$g(x) = x24 + x23 + x22 + x17 + 1$$

There is a 10-bit input CRC circuit for each of the four channel outputs. The 4 x 24-bit signatures at the end of each test are compared with reference good signatures and the status register reflects whether the test has passed or failed.

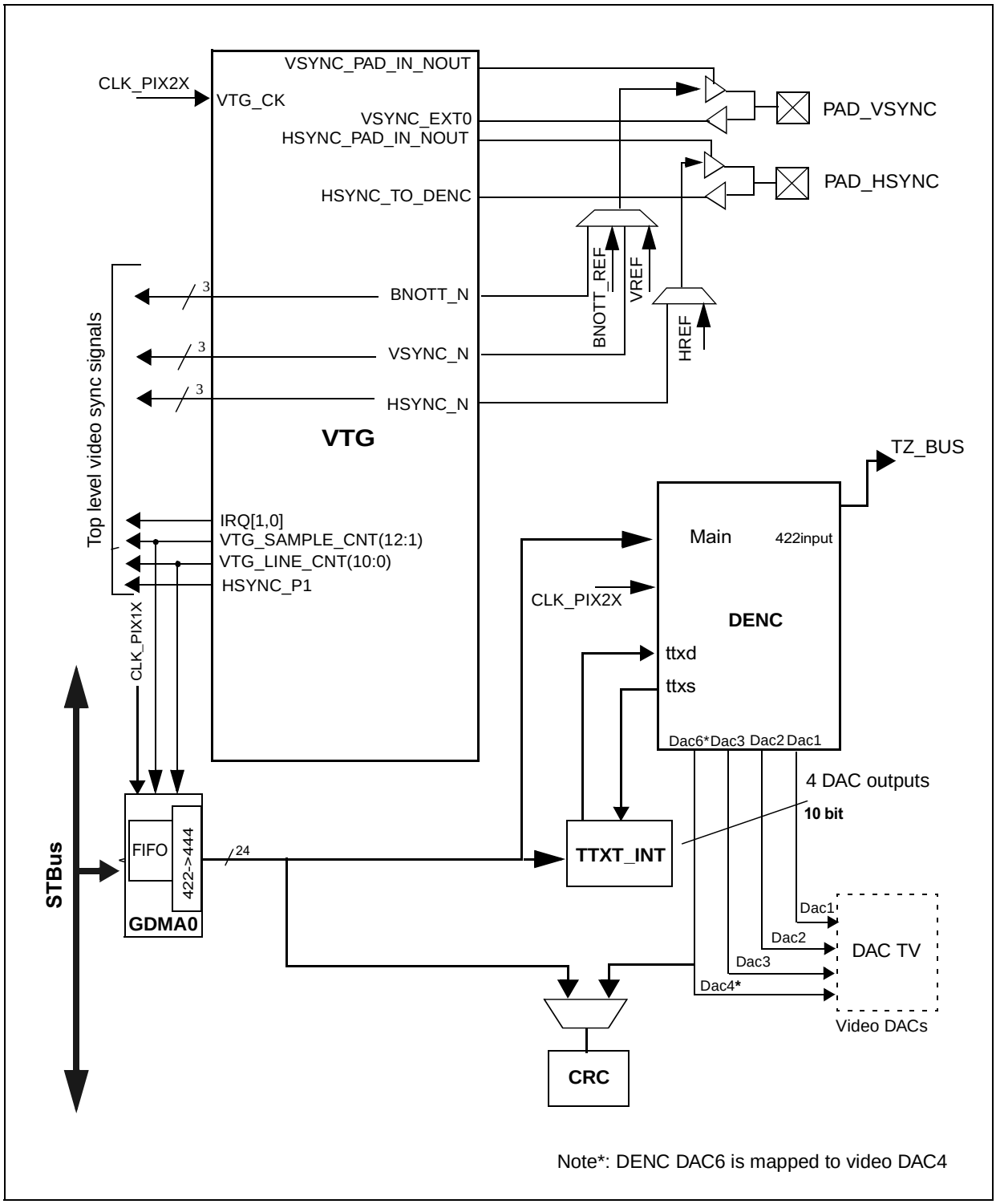
The signature computation, when enabled, starts after the next Vsync active edge and stops after one full frame has been captured. Subsequently, the CRC block can be reset and another one frame of data can be tested.

Figure 96: CRC: General Structure



External to TV out, the blitter engine scales and converts the MPEG decompressed video data and writes the processed active video and graphic data to the external memory in YCrCb format.

Figure 97: TV out block diagram



26.2 Teletext

Teletext data is retrieved from memory, serialized and transferred to the digital encoder. The digital encoder encodes teletext data according to the CCIR/ITU-R Broadcast Teletext System B specification (also known as World System Teletext).

Five dedicated digital encoder registers, DEN_TTX1 to DEN_TTX4, and DEN_TTXM, program the teletext encoding in various areas of the vertical blanking interval (VBI) of each field. Four of these areas (that is, blocks of contiguous teletext lines) can independently be defined within the two VBIs of one frame (for example, two blocks in each VBI, or three blocks in *field1* VBI and one in *field2* VBI). In certain circumstances it is possible to define up to four areas in each VBI.

DENC supports the following most commonly used teletext standards for DVB applications:

- TTXT-B (WST-PAL): 45 bytes - 2 bytes of clock run-in, 1 byte framing code, 42 bytes of data,
- TTXT-C (NABTS-NTSC): 39 bytes - 2 bytes of clock run-in, 1 byte framing code, 36 bytes of data.

There are two teletext operating modes:

- VBI: Teletext data on selected 18 VBI lines (e.g. ITU-R lines 6 to 23 and 318 to 335),
- Full Page: Teletext data replace video data (includes ITU-R lines 24 to 311 and 336 to 623).

26.3 Teletext packet format

One TTXT-B teletext packet (otherwise 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.

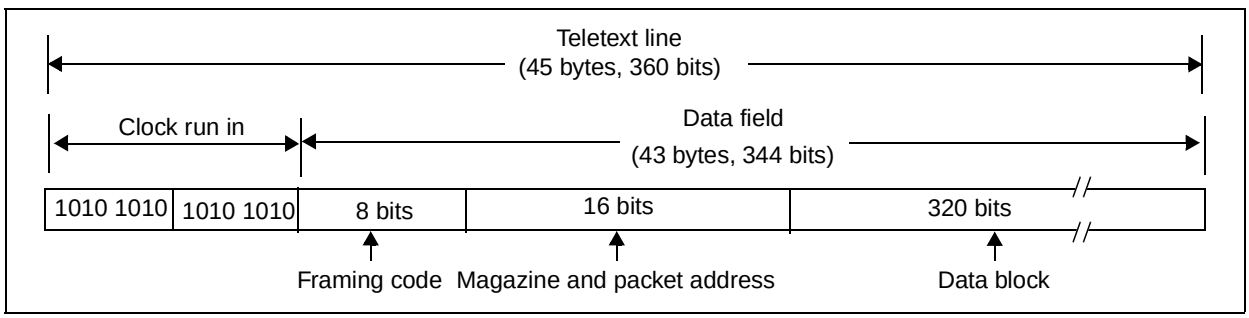
Each teletext packet is composed of the clock run in and the data field, as illustrated in Figure 98.

- The clock run in is composed of two bytes, each with the hexadecimal value 0xAA (binary value 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 98: Teletext packet format



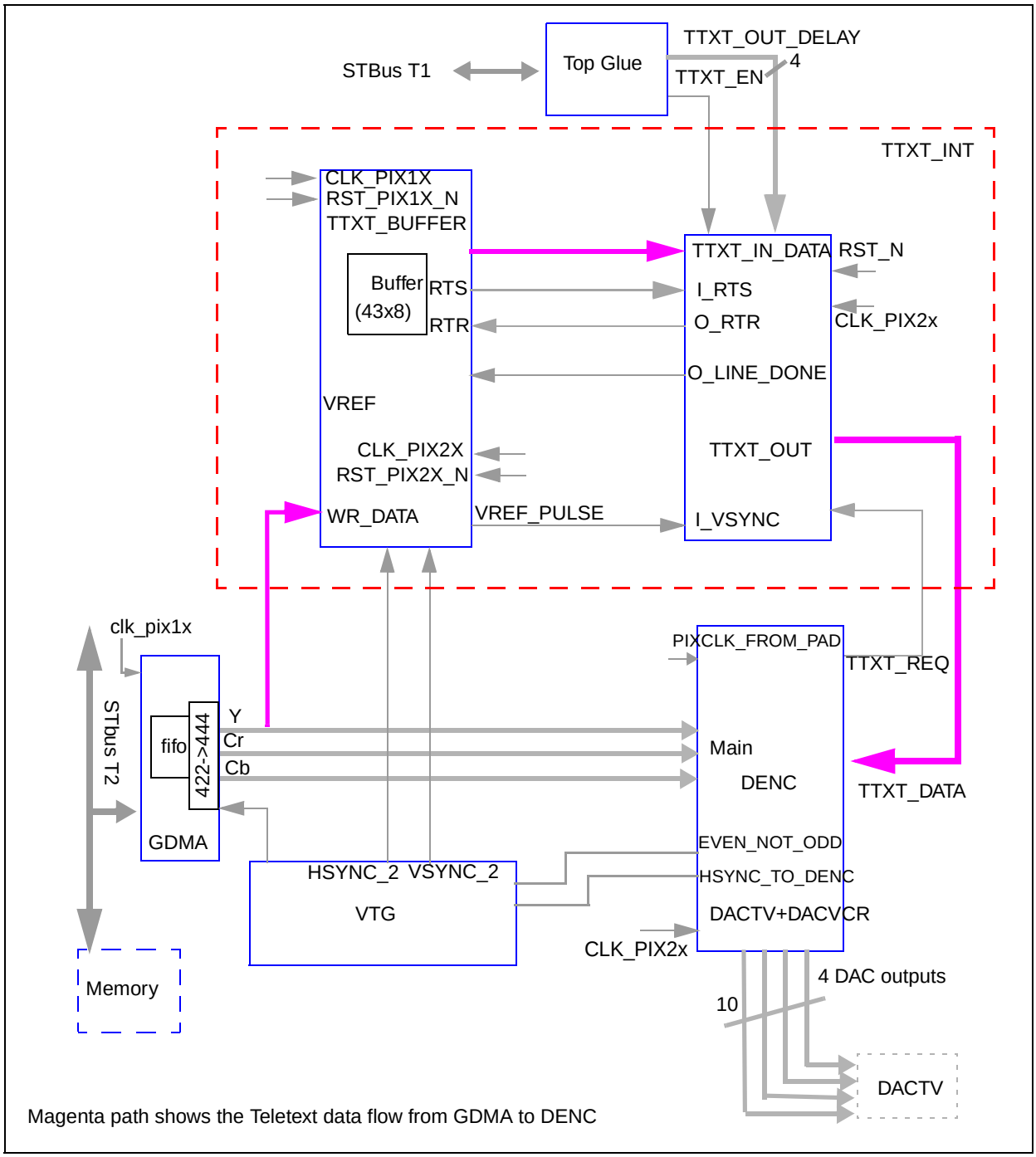
1. Specification for conveying ITU-R Systems B teletext in digital video broadcasting (DVB) bit streams.

26.4 Data transfer sequence

As shown in Figure 99 the teletext interface block (TTXT_INT) contains a 43 bytes buffer (TTXT_BUFFER), which is filled with teletext data by the GDMA at 13.5 Mbytes per second. When requested by the DENC, the teletext serial encoder TTXT_OUT reads this data in bytes from the TTXT_BUFFER, serially encodes it and sends it to the DENC at 6.9375 Mbits per second.

In memory, one line of teletext data is stored as the framing code + data bytes, where data_bytes are defined by the teletext standard. The two bytes of clock run-in are provided by the TTXT_OUT block at the beginning of every teletext line.

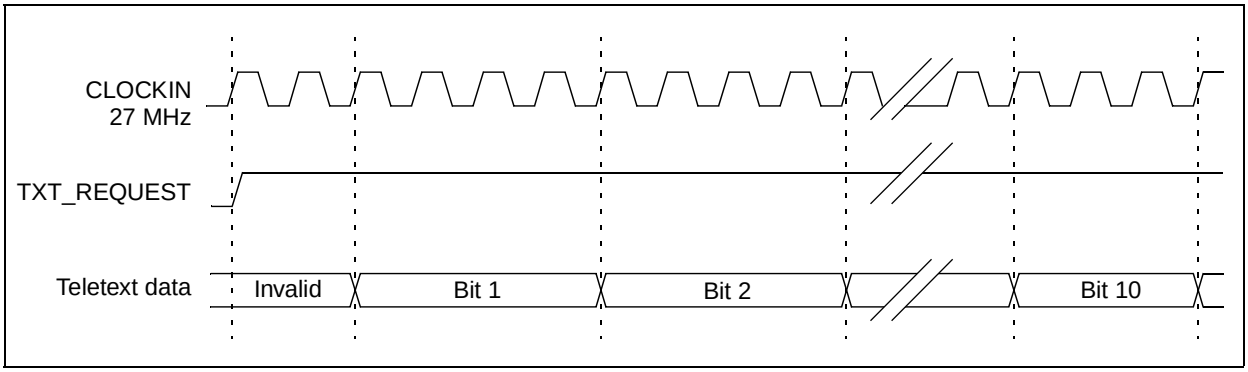
Figure 99: Teletext data transfer flow



The digital encoder issues a teletext request signal to the teletext serial encoder, this is shown by the rising edge of signal TTXT_REQUEST in Figure 100. After a delay, programmable from two to nine master clock periods, the teletext serial encoder transmits the first valid teletext data bit of the teletext packet.

The 360 bits of output data are defined as nine 37-bit sequences, ending with one 27-bit sequence. Within each sequence, each bit is transmitted in four 27 MHz cycles, except bits 10, 19, 28 and 37, which are transmitted in three 27 MHz cycles. This is illustrated in Figure 100 for bits 0 to 10.

Figure 100: Teletext data transfer sequence



The duration of the TTXS window is 1402 reference clock periods (51.926 µs), which corresponds to the duration of 360 teletext bits.

The delay between signal TTXT_REQUEST becoming high and the transfer of the first bit of the teletext packet is between two and nine, 27 MHz clock cycles. This delay is programmed by register bits TTXDEL[2:0] (DEN_TTX1). The value written to this register is increased by two 27 MHz clock cycles, so the value 0 corresponds to an overall delay of two x 27 MHz clock cycles, and the value seven corresponds to a delay of nine x 27 MHz clock cycles.

26.5 Interrupt control

VTG interrupts can be programmed to interrupt the CPU as follows:

- when the current video frame toggles odd to even or even to odd,
- every selected teletext data line per field.

The interrupt status is given by the VTG_ITS status register and masked by the VTG_ITM register. The interrupt bits are reset when the CPU writes to the register VTG_ITS.

27 TV out registers

Addresses are provided as the *TtxtBaseAddress* + offset.

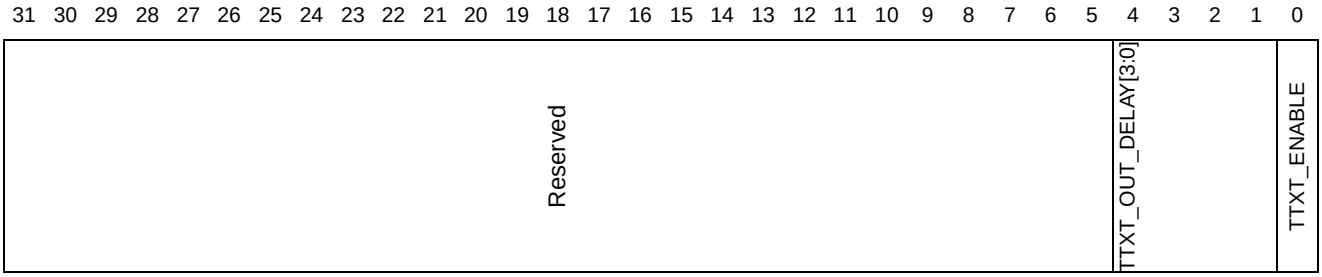
The *TtxtBaseAddress* is:

0x2090 0700.

Table 80: TVOut Teletext registers

Register name	Description	Address offset	Type
TVO_TTXT_CTRL	Teletext control register see page 281	0x0C	R/W

TVO_TTXT_CTRL Teletext control register



Address: *TtxtBaseAddress* + 0x0C

Type: Read/write

Reset: 0: [0], 0010: [4:1]

Description:

[31:5] **Reserved**

[4:1] **TTXT_OUT_DELAY[3:0]**
Delay inside the teletext out block from input TTXT_OUT_REQ to TTXT_OUT_DATA in 27 MHz clocks. Valid values are 2 to 9.

[0] **TTXT_ENABLE**
Enable TTXT_OUT block to digital encoder.

28 Graphics DMA (GDMA)

28.1 Overview

The graphics DMA (GDMA) moves graphics content from memory to the digital encoder. It supports the following list of features:

- link-list based display engine for multiple viewport capabilities,
- YCbCr4:2:2R, YCbCr888 input formats for display viewport,
- byte format for VBI viewport.

The GDMA fetches two premultiplied pixel data formats from the external memory in little endian format:

- YCrCb888 (601 only),
- YCbCr422 (601 only).

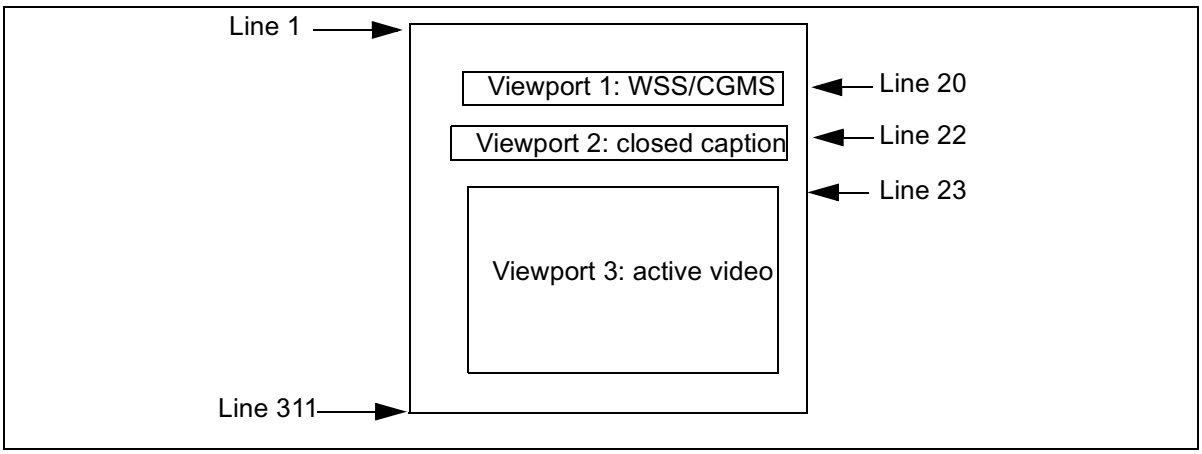
It delivers YCbCr888 to the digital encoder.

The GDMA supports four types of vertical data insertion:

- closed captioning (NTSC:21 and/or 284),(PAL:22 and/or 335),
- wide screen signalling WSS (NTSC:23),(PAL:20 and/or 283),
- copy generation management generation CGMS (NTSC:23),(PAL:20 and/or 283),
- teletext (NABTS/WST): Up to 43 bytes of data per teletext line from memory to TTXT_INT block inside TV out, which serially encodes it and passes it to the DENC

Four invisible viewports are used to insert data during vertical blanking time, plus one viewport for active video. See Figure 101.

Figure 101: PAL example for VDI data insertion by a graphical object from memory



28.2 Viewports

A viewport is a physical rectangular area defined in the screen area by four parameters which locate the top left and bottom right corner. The GDMA controls the multiple viewport display from a display instruction list (link list) stored in the external memory. It can provide VBI or output display content, which can be either in a single viewport or split into several viewports.

When several viewports are defined within a layer, only one viewport can be associated with any given video scan line. This means that there is always one line between two linked viewports. If VBI data is present on adjacent lines, one viewport two lines high should be used.

Where no viewport is defined for a screen location, GDMA output is automatically blanked (the color output is equal to the blanking color set in the GDMA_CTL register).

A pixmap stored in the external memory must be attached to each viewport. For the video nodes this pixmap is the output display composed by the blitter engine. This is defined by the registers GDMA_PML (pixmap memory location) and GDMA_PTY (pixmap memory type).

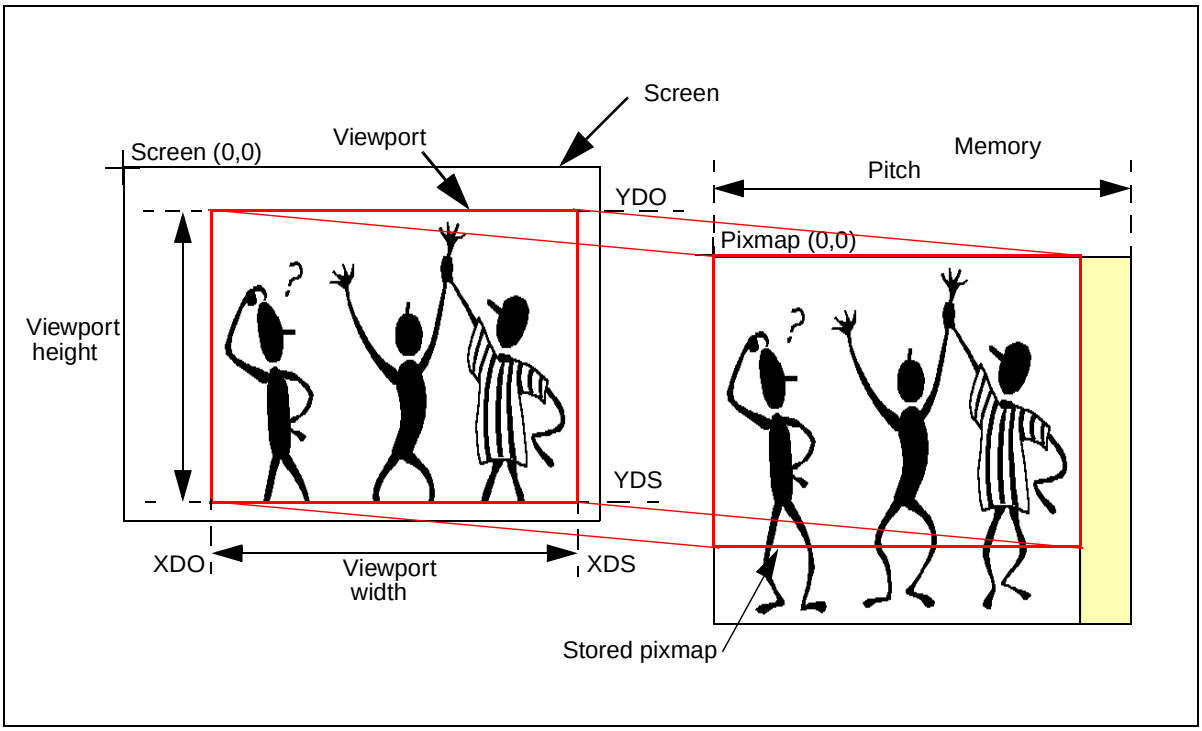
Each viewport has the following parameter list defined in GDMA_VPO and GDMA_VPS:

- **XDO**: horizontal start location of top left pixel,
- **YDO**: vertical start location of top left pixel,
- **XDS**: horizontal start location of bottom right pixel,
- **YDS**: vertical start location of bottom right pixel.

The vertical (YDO,YDS) co-ordinates are field-based.

28.3 Output pixmap display

Figure 102: Output display attached to a viewport, NTSC: 720 x 480, PAL: 720 x 576



Viewports are linked together by a link-list which describes the output display. Each viewport is attached to a bitmap stored in external memory. The link-list describing the output display is built in the external memory. This is constructed as a series of nodes linked by the node pointers GDMA_NVN or GDMA_NIN.

The link-list is field-based for an interlaced display. This means two nodes must be provided for each viewport, one node for the TOP field and another for the BOTTOM field.

There are two types of link-lists:

- VBI link-list,
- video link-list.

VBI data is output before the final video display data (see Figure 103). Each type of link-list uses a different pointer, GDMA_NVN for video data and GDMA_NIN for VBI data.The VBI node should not be vertically adjacent to the video node in memory. This is programmed in the GDMA_LVF register which provides the scan line number for the first video data node to be fetched.

GDMA_LVF is interpreted according to the field or frame configuration. The interlaced environment is field based.

Each pixmap node has the following parameter list:

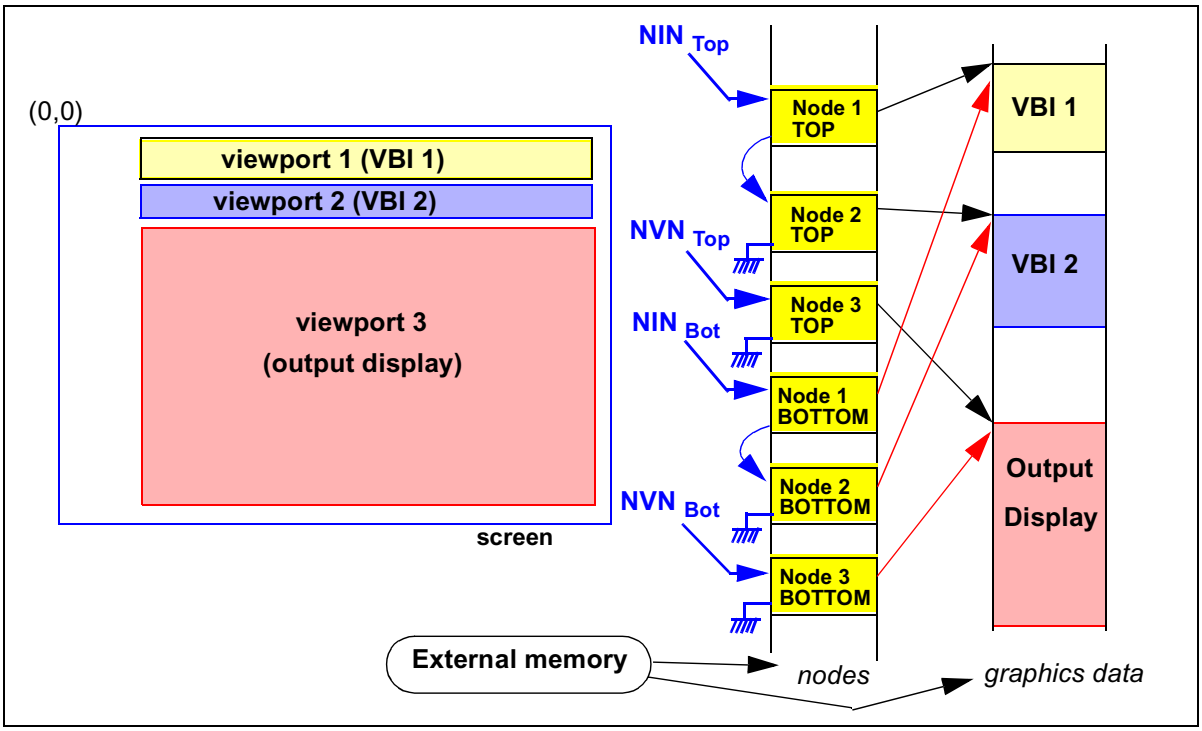
- pixmap color format,
- pixmap pitch,
- linear start address (pixel accurate).

  Width and height are computed from the XDO, XDS, YDO and YDS values.
  Width = XDS - XDO +1 and height = YDS - YDO +1.

  The linear start address is the address for retrieving the pixmap pixel displayed in the upper-left corner of the viewport, that is the address of the window (0,0) pixel.

The YDO and YDS values are specified in reference to frame line numbering, even in an interlaced display.

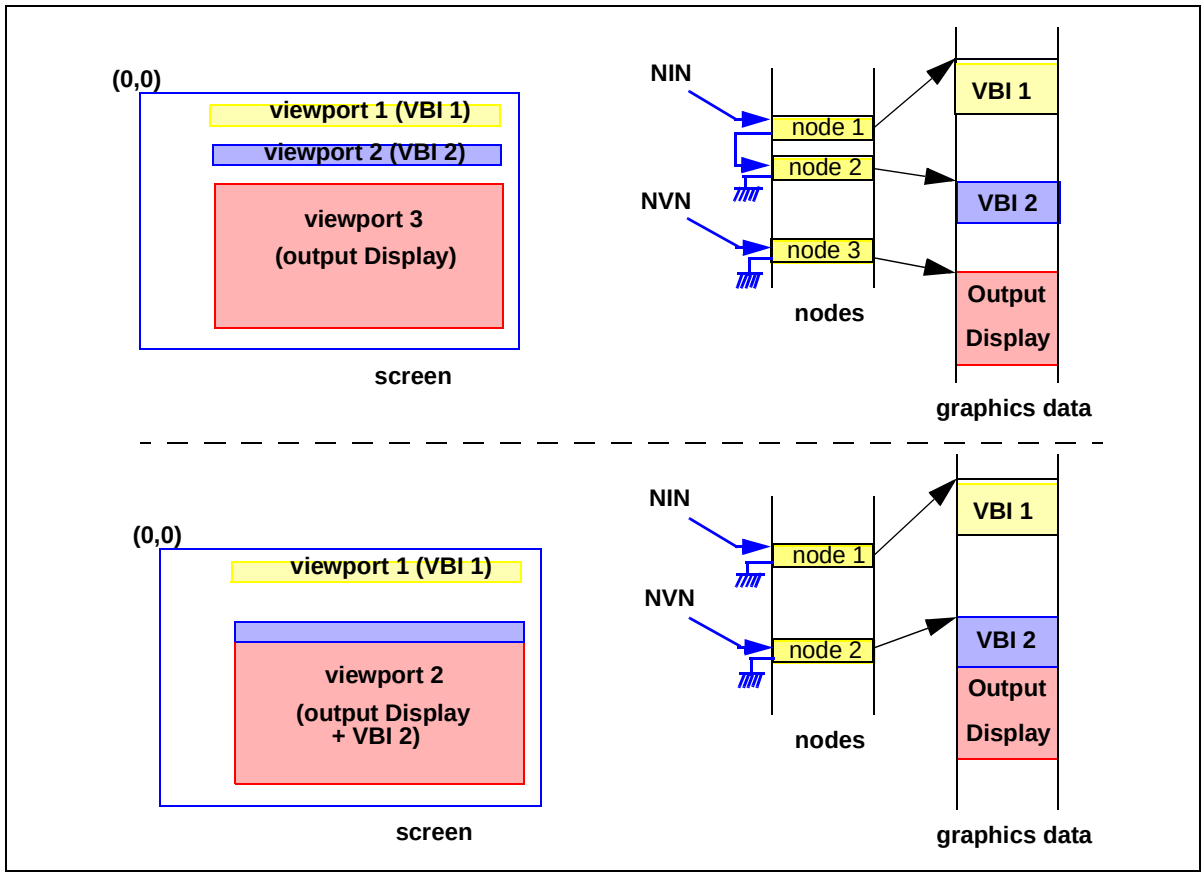
Figure 103: Typical display link-list configuration for interlaced display



Note: Software must leave at least one blank line between two different viewport types.

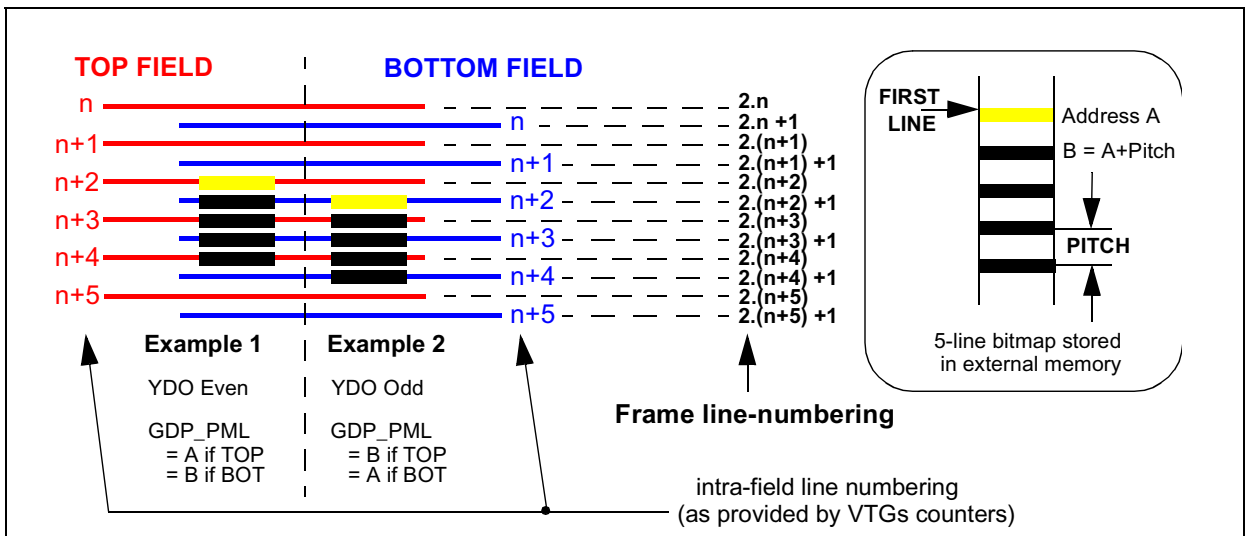
Before switching from a VBI to a video node, the GDMA must load the video node, and then enough video data to fill the pipe. This means that if a VBI node and the video viewport need to be vertically adjacent in memory, the software must increase the video viewport height to include the VBI viewport.

Figure 104: VBI and video merge when viewports are vertically adjacent



In interlaced displays VPO and VPS line number values vary depending on the TOP or BOTTOM field, so that each viewport can be vertically positioned with one line accuracy. See Figure 105.

Figure 105: Line numbering convention in an interlaced display



1 - During the TOP FIELD, the VTG counter must be compared to **(YDO+1 >> 1) & (YDS >> 1)** for detecting the viewport active area.

2 - During the BOTTOM FIELD, the VTG counter must be compared to **(YDO >> 1) & (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.

28.3.1 Starting the display

VBI (VDI) link list

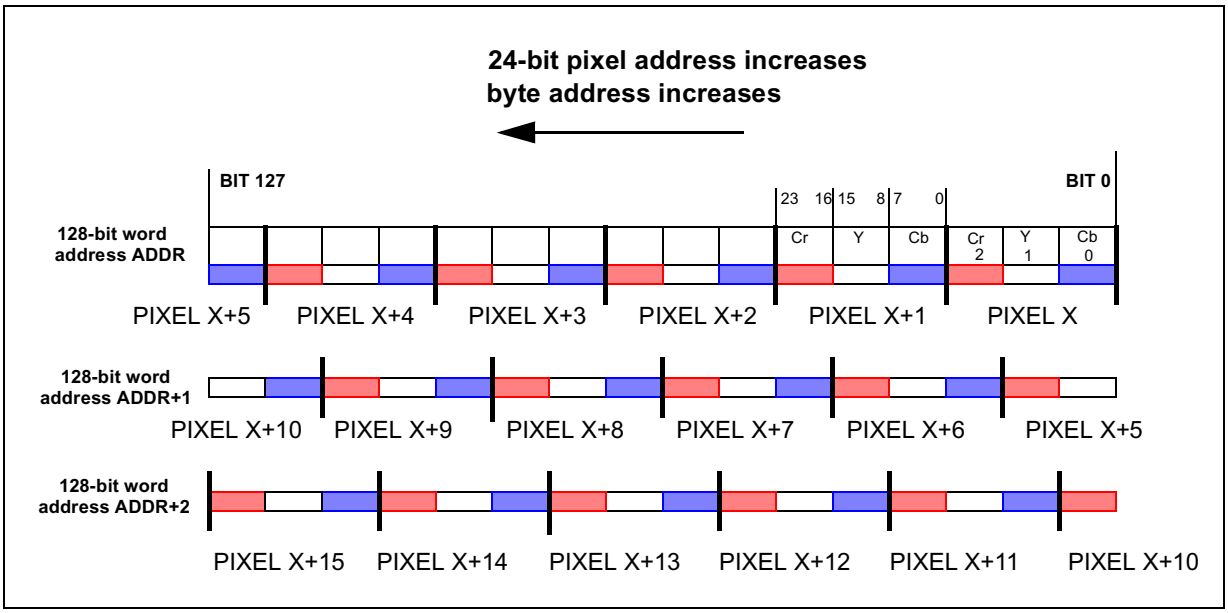
If VBI_PRESENT in GDMA_CTL is set, the GDMA fetches the associated node. Once the line counter reaches the associated line number it then starts fetching the data.

Video link list

To start the display of these nodes, two registers are set. GDMA_NVN points to the first video node, and GDMA_LVF gives the line number to reach before starting to load this node.

28.3.2 Input format

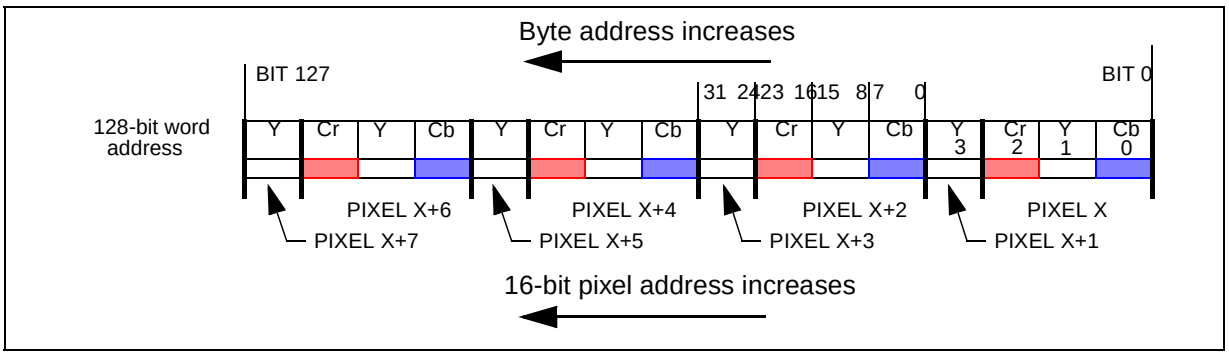
Figure 106: YCbCr888 alignment in a 128-bit word (3-word period / 16-pixel period)



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 107: YCbCr 4:2:2 (16 bpp)

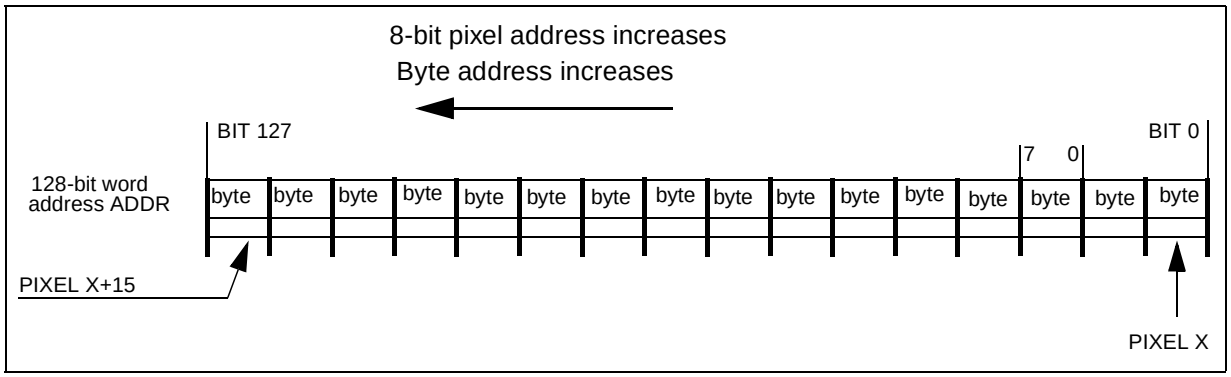


Note: 1 The memory address for pixel [0:0] must be aligned on a 32-bit word address boundary.

2 The pitch value must be a multiple of four bytes.

3 X is even in Figure 107.

Figure 108: Byte alignment in a 128-bit word



Note: The memory address for pixel [0:0] must be byte-aligned.

29 Graphics DMA (GDMA) registers

All the registers are memory mapped. The block occupies a 64 x 32-bit word address range.

Addresses are provided as the *GDMABaseAddress* + offset.

The *GDMABaseAddress* is:

0x2090 0C00

Table 81: Graphics DMA (GDMA) registers

Register name	Description	Address offset	Type
GDMA_CTL	GDMA control register, see page 288	0x0000	R/W
GDMA_NIN	GDMA next insertion node pointer, see page 289	0x0004	R/W
GDMA_NVN	GDMA next video node pointer, see page 289	0x0008	R/W
GDMA_LVF	GDMA line video fetch, see page 290	0x000C	R/W
GDMA_NIP	GDMA next instruction pointer, see page 290	0x0010	R/LLU
GDMA_VPO	GDMA viewport offset, see page 290	0x0014	R/LLU
GDMA_VPS	GDMA viewport stop, see page 291	0x0018	R/LLU
GDMA_PML	GDMA pixmap memory location, see page 291	0x001C	R/LLU
GDMA_PTY	GDMA pixmap type, see page 292	0x0020	R/LLU
GDMA_PKZ	GDMA maximum packet size, see page 292	0x00FC	R/W

GDMA_CTL GDMA control register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHROMA_FORMAT	601_709_SEL	Reserved		VBI_PRESENT	VIDEO_ PRESENT	Reserved		Y								CB								CR							

Address: *GDMABaseAddress* + 0x0000

Type: Read/write

Buffer: Double buffered

Reset: 0x0010 8080

Description: GDMA_CTL provides the operating mode of the GDMA pipe for the current frame display.

[31] **CHROMA_FORMAT**
0: The color space converter handles Cr/Cb as offset binary data
1: The color space converter handles Cr/Cb as 2's signed data

[30] **601_709_SEL**
0: The color space converter uses 601 colorimetry levels
1: The color space converter uses 709 colorimetry levels

[29:28] **Reserved**

[27] **VBI_PRESENT**
0: No VBI node to fetch
1: There is at least one VBI node to fetch

[26] **VIDEO_PRESENT**
0: No video node to fetch
1: There is at least one video node to fetch

[25:24] **Reserved**

[23:16] **Y**: Y blanking value

[15:8] **CB**: Cb blanking value

[7:0] **CR**: Cr blanking value

GDMA_NIN GDMA next insertion 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
NIN_BANK					NEXT_VBI_NODE																							Reserved			

Address: *GDMA BaseAddress* + 0x0004
Type: Read/write
Buffer: Double buffered
Reset: 0
Description: The GDMA_NIN register provides the address of the first node associated to VBI data in memory. If no VBI data has to be displayed by GDMA, this register should remain set to 0.

[31:27] **NIN_BANK:** 64 MByte bank number.

[26:4] **NEXT _VBI_NODE**: next VBI node address.

[3:0] **Reserved**

GDMA_NVN GDMA next video 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
NVN_BANK					NEXT_VIDEO_NODE																							Reserved			

Address: *GDMA BaseAddress* + 0x0008
Type: Read/Write
Buffer: Immediate
Reset: 0
Description: The GDMA_NVN register provides the address of the (first) node associated to video (display) data.

[31:27] **NVN_BANK**: 64 MByte bank number.

[26:4] **NEXT_VIDEO_NODE**: next video node address.

[3:0] **Reserved**

GDMA_LVF GDMA line video fetch

31	30	29	28	27	26	25	24	23	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_FETCH_NUMBER										

Address: *GDMA BaseAddress* + 0x000C

Type: Read/Write

Buffer: Double buffered

Reset: 0

Description: The GDMA_LVF register provides the line number from which GDMA should start to fetch video data (first) node.

[31:11] **Reserved**

[10:0] **LINE_FETCH_NUMBER**: line number to wait before fetching video node.

GDMA_NIP GDMA 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: *GDMA BaseAddress* + 0x0010

Type: Read/Link list update

Buffer: Updated on node fetch

Reset: 0

Description: The GDMA_NIP register provides the address where GDMA should fetch the next node of current linked list.

[31:27] **NIP_BANK_NUM**: 64 MByte bank number.

[26:0] **NIP_MEM_ADDR**: NIP address.

GDMA_VPO GDMA 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: *GDMA BaseAddress* + 0x0014

Type: Read / Linked list update

Buffer: Updated on node fetch

Reset: 0

Description: The GDMA_VPO register provides the x,y location of the viewport top-left pixel, with respect to the current video timebase (top-left corner of the active video window).

[31:27] **Reserved**

[26:16] **YDO**: Y location for the first line of the viewport (top), with respect to frame numbering (vertical start location of the active video window, with respect to line frame-numbering).

[15:12] **Reserved**

[11:0] **XDO**: X location for the first pixel of the viewport (left) (Horizontal start location of the active video window, with respect to VTG horizontal counter).

GDMA_VPS GDMA 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: *GDMA BaseAddress* + 0x0018

Type: Read / Linked list update

Buffer: Updated on node fetch

Reset: 0

Description: The GDMA_VPS register provides the x,y location of the viewport bottom-right pixel, with respect to the current video timebase (bottom-right corner of the active video window).

[31:27] **Reserved**

[26:16] **YDS:** Y location for the last line of the viewport (bottom), with respect to frame numbering (vertical stop location of the active video window, with respect to line frame-numbering).

[15:12] **Reserved**

[11:0] **XDS:** X location for the last pixel of the viewport (right) (horizontal stop location of the active video window, with respect to VTG horizontal counter).

GDMA_PML GDMA 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_NO						PIXMAP_MEM_PTR																									

Address: *GDMABaseAddress* + 0x001C

Type: Read / Linked list update

Buffer: Updated on node fetch

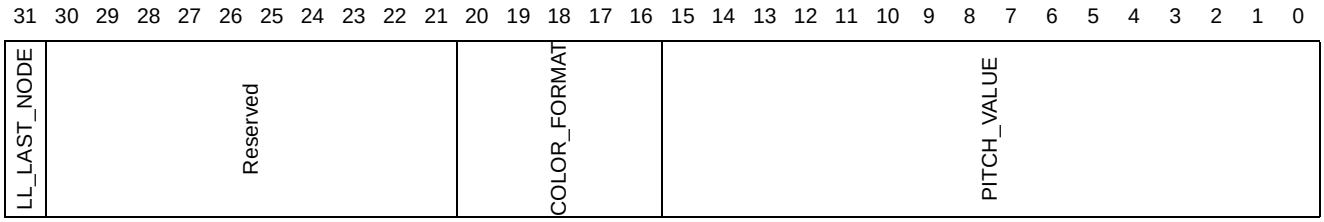
Reset: 0

Description: The GDMA_PML register is a 32-bit register containing the memory location for the first pixel to be displayed (top-left corner).

[31:26] **64MB_BANK_NO**: 64 Mbyte bank number.

[25:0] **PIXMAP_MEM_PTR:** first 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).

GDMA_PTY GDMA pixmap type



Address: *GDMABaseAddress* + 0x0020

Type: Read/Linked list update

Buffer: Updated on node fetch

Reset: 0

Description: The GDMA_PTY register contains the memory pitch for the displayed pixmap, as stored in the memory.

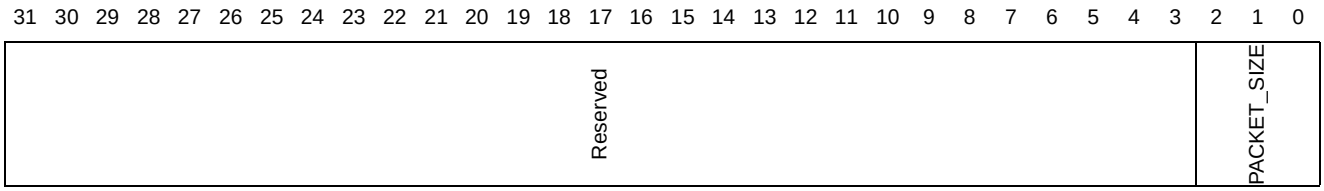
[31] **LL_LAST_NODE:** when set, the current node is the last of the linked list.

[30:21] **Reserved**

[20:16] **COLOR_FORMAT:** color format of current node.
10010: YCbCr 422R 10000: YCbCr 888
11111: Byte

[15:0] **PITCH_VALUE:** memory pitch for the displayed pixmap.
The pitch is the distance inside the memory, in byte units, between two vertically adjacent pixels).

GDMA_PKZ GDMA maximum packet size



Address: *GDMABaseAddress* + 0x00FC

Type: Read/write

Buffer: Immediate

Reset: 0x0

Description: The GDMA_PKZ register is a 3-bit register for controlling the static parameters of the GDMA pipelines.

[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

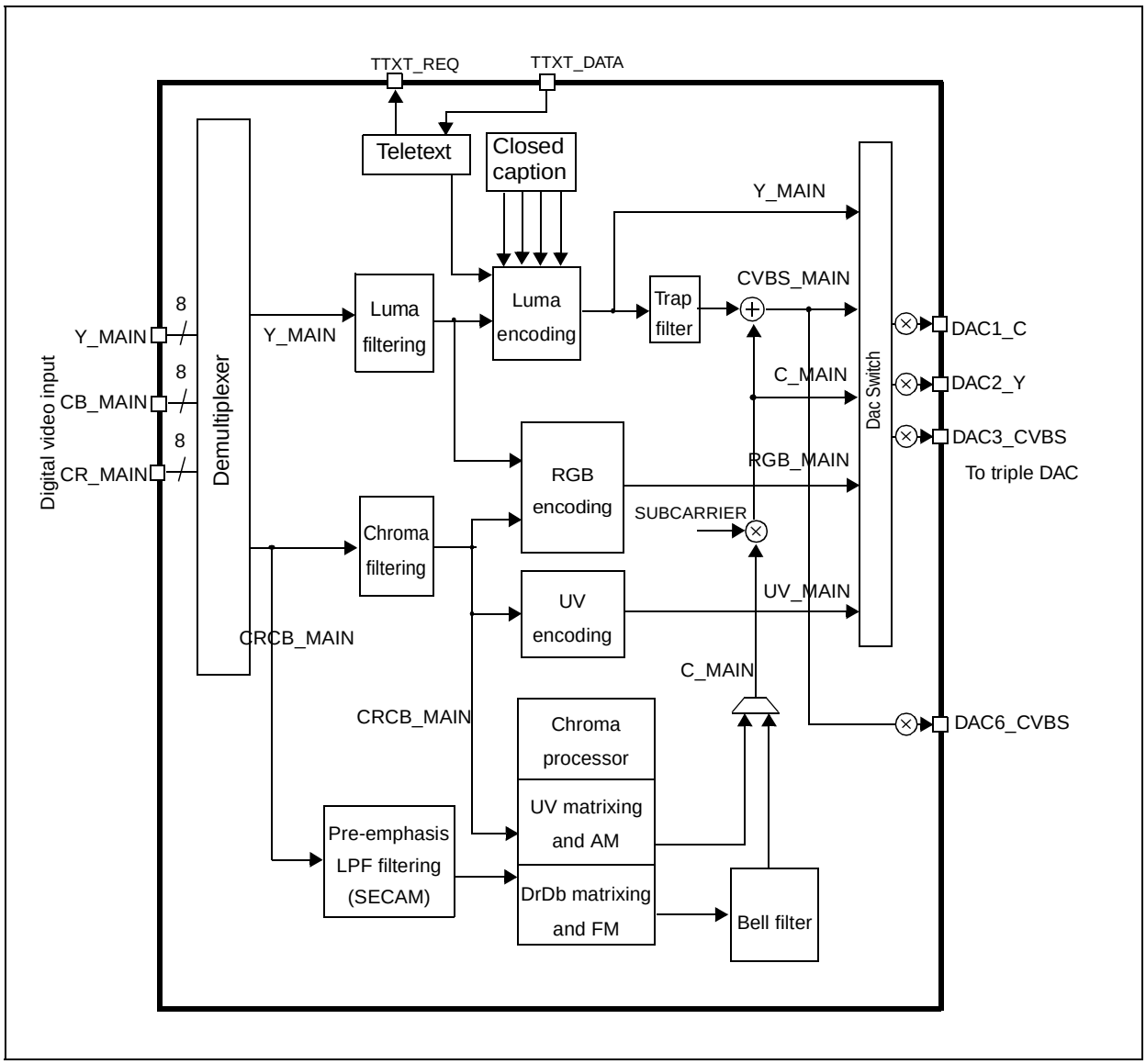
30 Digital encoder (DENC)

The DENC supports PAL/NTSC/SECAM encoding. It gives RGB, CVBS and Y/C outputs.10-bit video DACs are integrated on the device. DENC supports Macrovision 6.1/7.01 copy protection which can be enabled by an anti-fuse. It also supports teletext re-insertion in vertical blanking interval. The VCR output is the same as the TV output.

The digital encoder (DENC) is a high performance PAL/SECAM/NTSC digital encoder. It converts one 4:4:4 digital video stream into a standard analog baseband PAL/SECAM/NTSC signal and into RGB and YPrPb analog components.

The DENC can handle interlaced mode in all standards. It can perform closed-captions, CGMS, WSS, VPS and teletext encoding and allows Macrovision^{TM} 7.01/6.1 copy protection. Four analog output pins are available.

Figure 109: STx5119 digital encoder block diagram

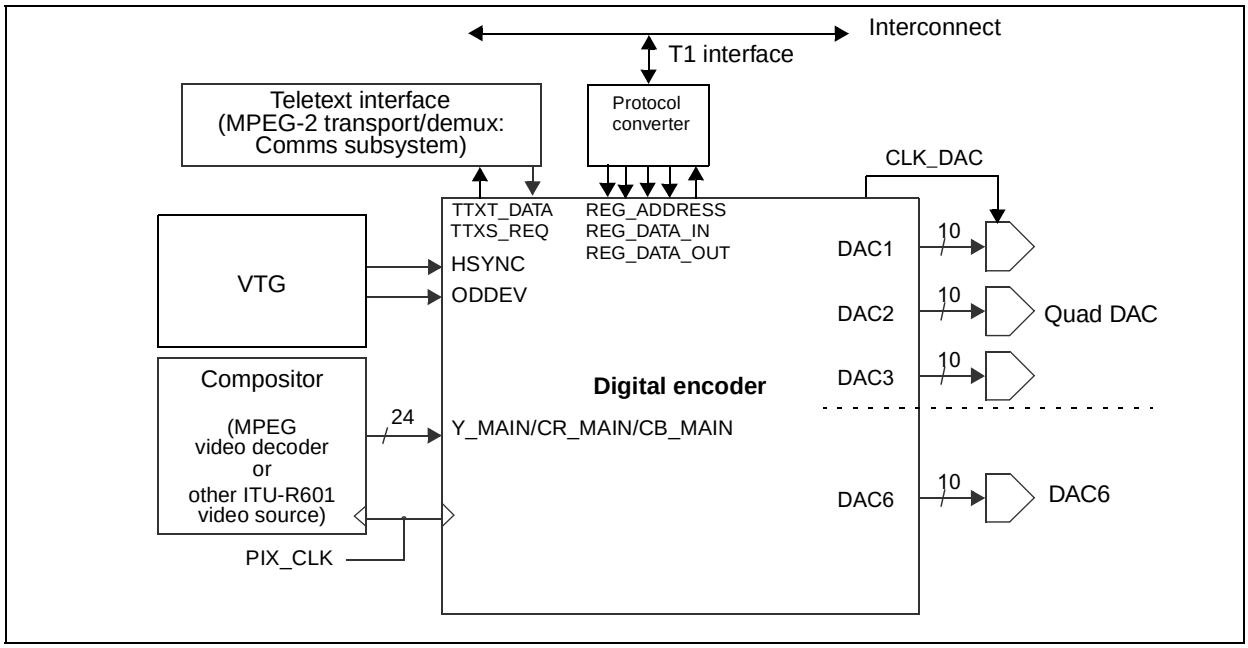


30.1 Main features

- NTSC-M, PAL-BDGHI, N, M, SECAM plus NTSC-4.43 encoding (optional pedestal in PAL and NTSC standards).
- Four simultaneous analog outputs on four 10-bit DACs:
  S-VHS (Y_MAIN/C_MAIN) + CVBS_MAIN + CVBS_MAIN or
  RGB_MAIN + CVBS_MAIN or
  Pr_MAIN + Y_MAIN+ Pb_MAIN + CVBS_MAIN
- Macrovision™ Rev7.01/Rev6.1 copy protection process in NTSC, PAL and SECAM.
- Cross-color reduction by specific trap filtering on luma within CVBS flow for the CVBS output.
- Closed-captioning, CGMS, WSS, VPS encoding and teletext encoding with included sequencer and shaping for CVBS_MAIN coming independently from main video input.
- Digital frame sync input/ODDEV, programmable polarity and relative position.
- Digital horizontal sync input/output (HSYNC), programmable polarity and relative position.
- Full or partial vertical blanking.
- Luma filtering with 2x oversampling and sin y / y correction.
- Chrominance filtering with 4x oversampling to either 1.1 MHz, 1.3 MHz, 1.6 MHz or 1.9 MHz (in PAL and NTSC) and chrominance filtering according to relevant SECAM standards.
- Both luma and chrominance filtering are with programmable coefficients on main video path.
- Wide chrominance bandwidth for RGB encoding (2.45 MHz if CRCB are input at 6.75 MHz and 5.8 MHz if CRCB are input at 13.5 MHz rate).
- Saturation, contrast and brightness control of 4:4:4 main.
- 24-bit direct digital frequency synthesizer for color subcarrier.
- Programmable reset of color subcarrier phase (4 modes).
- Programmable access via parallel port.
- Programmable (4 bits) output levels RGB (-20% to +17%), C versus Y in CVBS and S-VHS (0 to +23.5%), PrPb (-25% to +22%), Y and CVBS (-25% to +22%) and C (-25% to +22%).
- 4:4:4 input can be encoded on both RGB (YPrPb) and CVBS (S-VHS) outputs (programmable).
- Autotest operation mode (on-chip color bar pattern 100/0/75/0).

30.2 Brief overview

Figure 110: Typical DENC application



The DENC can operate either in master mode, where it supplies all sync signals, or in six slave modes, where it locks onto incoming sync signals. The main functions are controlled from a microcontroller via a parallel port.

30.3 Data input format

The digital input is through three separate 8-bit buses carrying Y, Cb and Cr at 13.5 MHz. Input samples are latched in on the rising edge of the clock signal PIX_CLK. Figure 110 illustrates the expected data input format. The video input of 24 bits (Y_MAIN/CB_MAIN/CR_MAIN) must be synchronous. The encoding of the main 4:4:4 input is enabled by programming DAC123_CONF as RGB or YPrPb or Y/C/CVBS analog outputs.

Table 82: Subcarrier frequencies in ITU-R601 mode

Standard	Application	PIX_CLK (MHz)	Pixel rate (MHz)	Field rate (Hz)	Vertical resolution
PAL-B, D, G, H, I PAL-N, SECAM	ITU-R601	27	13.5	50	625
NTSC-M, PAL-M	ITU-R601	27	13.5	60	525

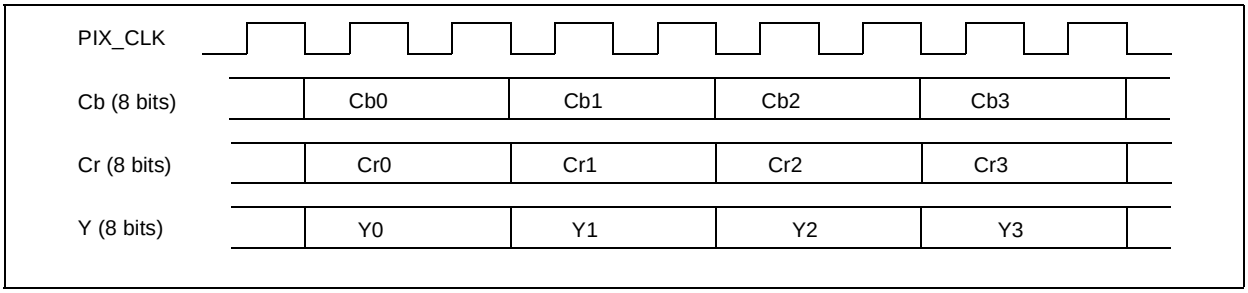
The input pixel data has an integer relationship to the number of clock cycles per horizontal line as detailed in Table 83.

Table 83: Relationship of input pixel data to clock cycles per line

Standard	Application	Pixel clock (MHz)	Total pixels per line	Active pixels per line
PAL-B, D, G, H, I PAL-N, SECAM	ITU-R601	13.5	864	720
NTSC-M, PAL-M	ITU-R601	13.5	858	720

Timing of video inputs is shown in Figure 111.

Figure 111: 4:4:4 input data format



30.4 Video timing

The DENC outputs interlaced video in PAL-BDGHI, PAL-N, PAL-M or NTSC-M standards (NTSC- 4.43 is also possible) and SECAM.

The burst sequences are internally generated, subcarrier generation being performed numerically with PIX_CLK as reference. 4-frame bursts are generated for PAL or 2-frame bursts for NTSC. Rise and fall times of synchronization tips and burst envelope are internally controlled according to the relevant ITU-R and SMPTE recommendations. The 6-frame subcarrier phase sequence is generated in SECAM (see Section 30.11 on page 308).

Figure 114 to Figure 119 depict typical VBI waveforms.

Incoming YCrCb data may be encoded on those lines of the VBI that do not bear line sync pulses or pre/post-equalization pulses. This mode of operation is referred to as partial blanking and is the default set-up. Any VBI data present in digitized form in the incoming YCrCb stream (such as supplementary closed-captions line or StarSight ® data) may be kept in the encoded waveform in this way. Alternatively, the complete VBI may be fully blanked, so no incoming YCrCb data is encoded on these lines.

Complete VBI comprises the following lines:

- for 525/60 systems (SMPTE line numbering convention): lines 1 to 19 and second half of line 263 to line 282,
- for 625/50 systems (CCIR line numbering convention): second half of line 623 to line 22 and lines 311 to 335.

Partial VBI consists of:

- for 525/60 systems (SMPTE line numbering convention): lines 1 to 9 and second half of line 263 to line 272,
- for 625/50 systems (CCIR line numbering convention): second half of line 623 to line 5 and lines 311 to 318.

Full or partial blanking is controlled by configuration bit BLKLI (DEN_CFG1).

Note: 1 Line 282 in 525/60/SMPTE systems is either fully blanked or fully active.

2 Line 23 in 625/60/CCIR systems is always fully active.

In an ITU-R656-compliant digital TV line, the active portion of the digital line is the portion included between the SAV (start of active video) and EAV (end of active video) words. However, this digital active line starts somewhat earlier and may end slightly later than the active line usually defined by analog standards.

The DENC allows two approaches:

- Encodes the full digital line (720 pixels / 1440 clock cycles). In this case, the output waveform reflects the full YCrCb stream included between SAV and EAV.

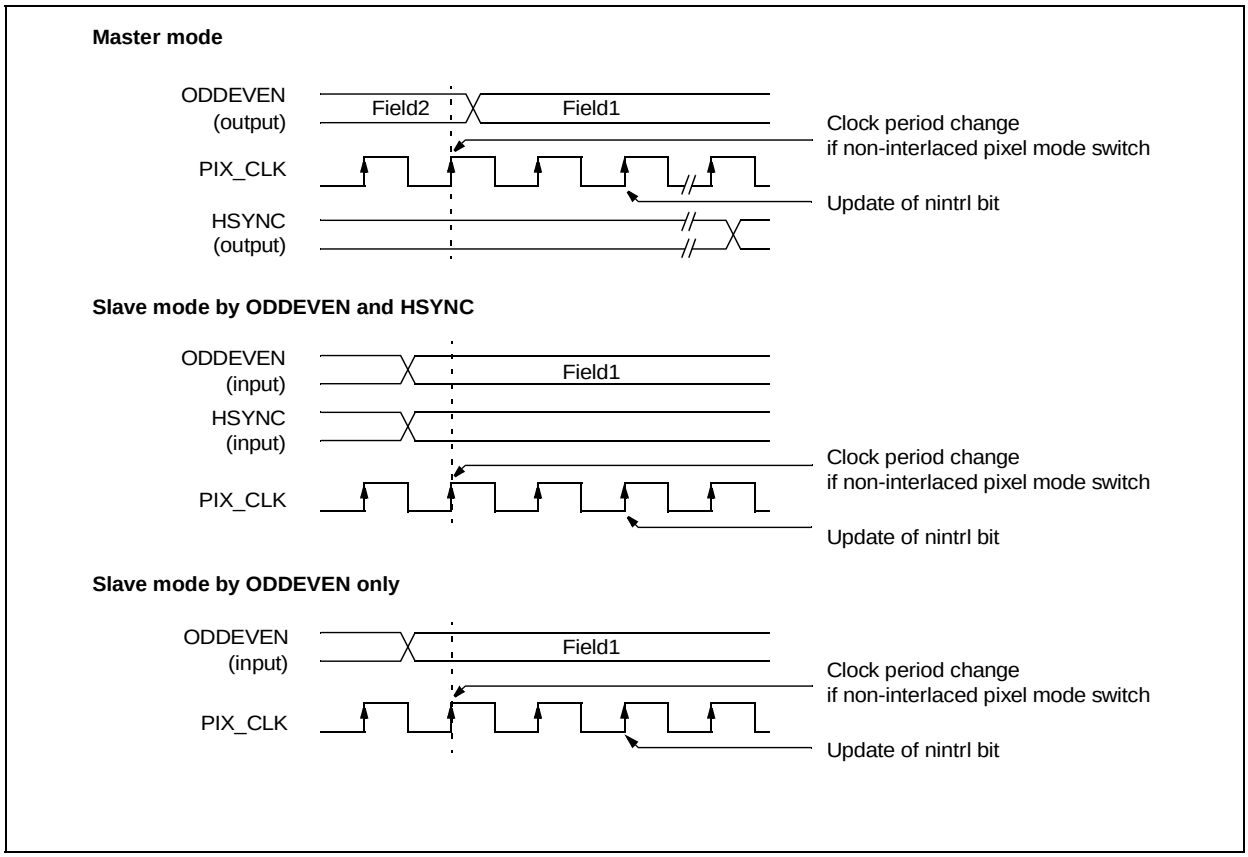
- Drops some YCrCb samples at the extremities of the digital line so that the encoded analog line fits within the analog ITU-R/SMPTE specifications.

Selection between these two modes of operation is performed with bit ALINE (DEN_CFG4).

In all cases, the transitions between horizontal blanking and active video are shaped to avoid too steep edges within the active video. Figure 119 on page 300 gives timings concerning the horizontal blanking interval and the active video interval.

Note: The burst envelope shown here indicates the location from which the first subcarrier positive zero crossing is sought (with respect to the 0_H reference). The normal burst always starts with such a positive zero crossing.

Figure 112: Non-interlaced mode switch



Note: 1 These diagrams are valid with contents of delay and sync-delay register fields equal to the default values.

2 If on-the-fly format changing is required, clock switching must be synchronized onto the start of frame as shown in the above waveform.

Figure 113: PAL-BDGHI, PAL-N typical VBI waveform, interlaced mode (ITU-R625 line numbering)

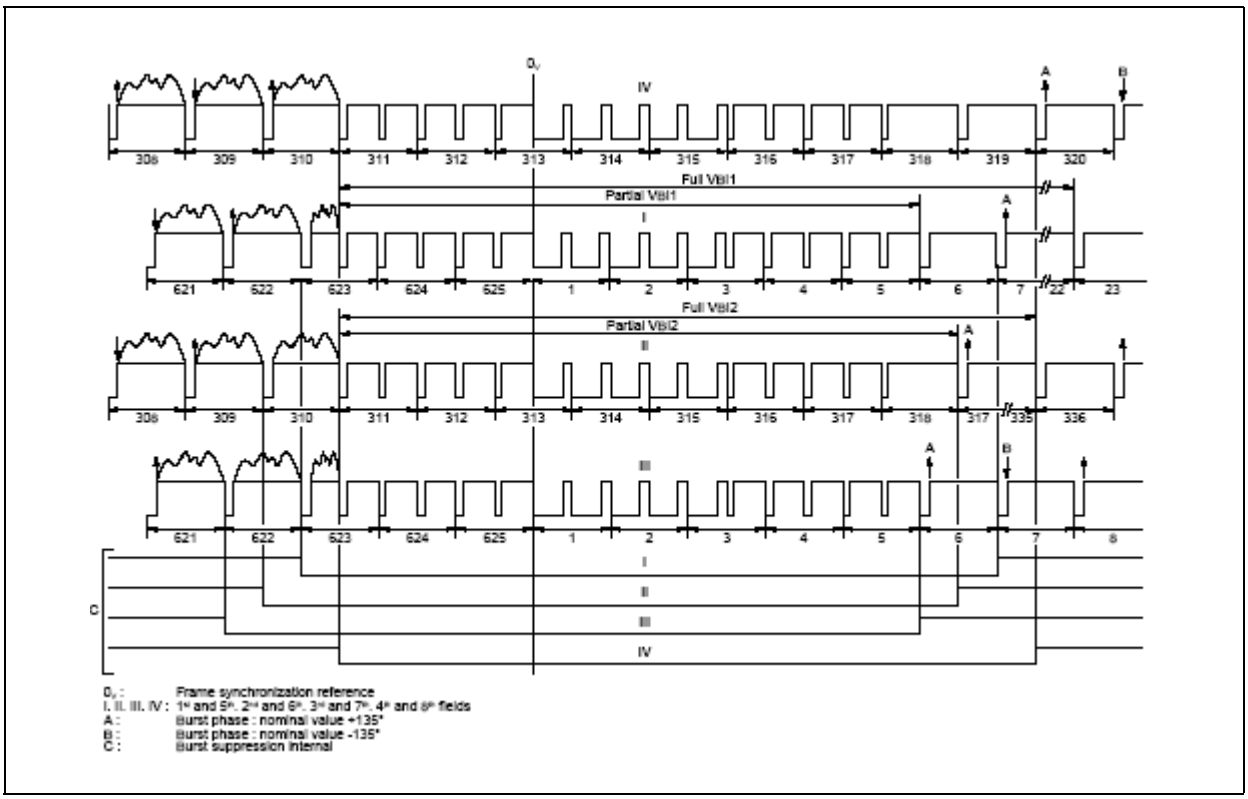


Figure 114: PAL-BDGHI, PAL-N typical VBI waveform, non-interlaced mode (CCIR-like line numbering)

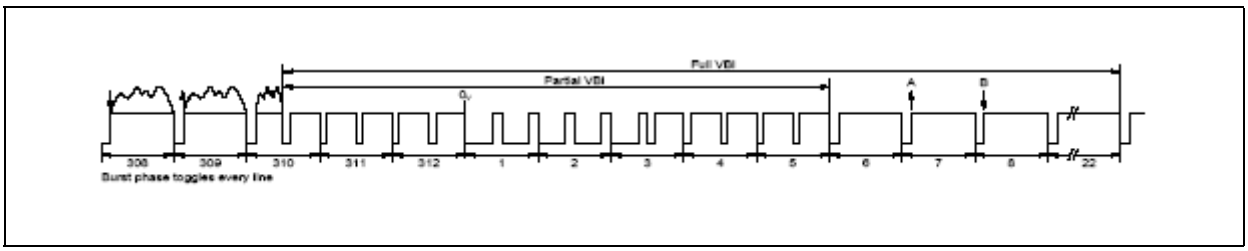


Figure 115: NTSC-M typical VBI waveforms, interlaced mode (SMPTE-525 line numbering)

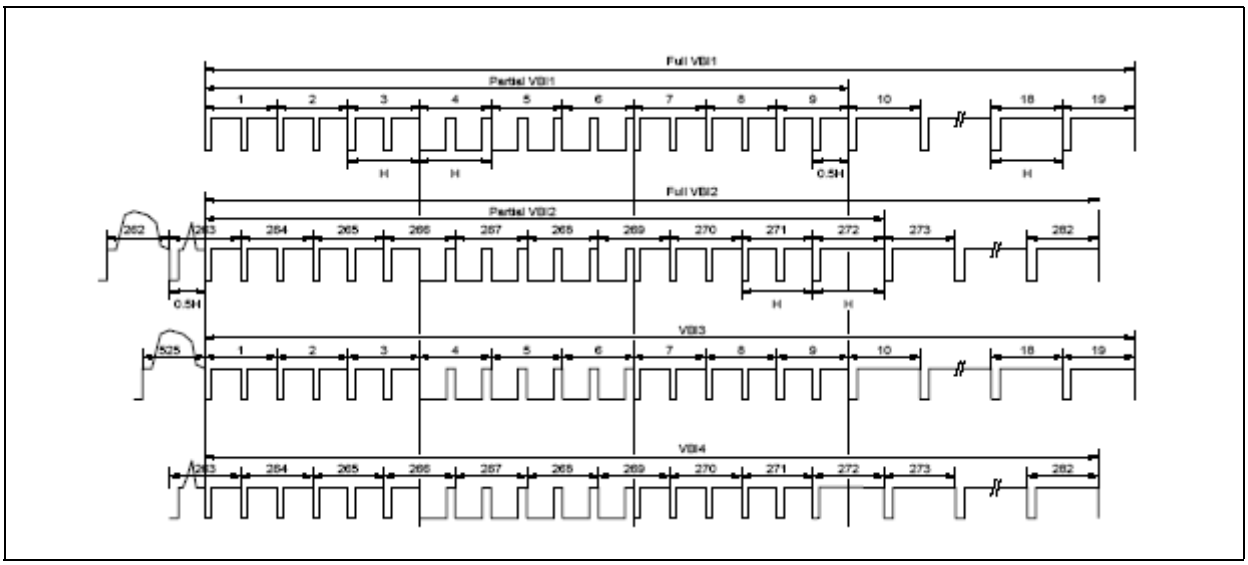


Figure 116: NTSC-M typical VBI waveforms, noninterlaced mode (SMPTE-like line numbering)

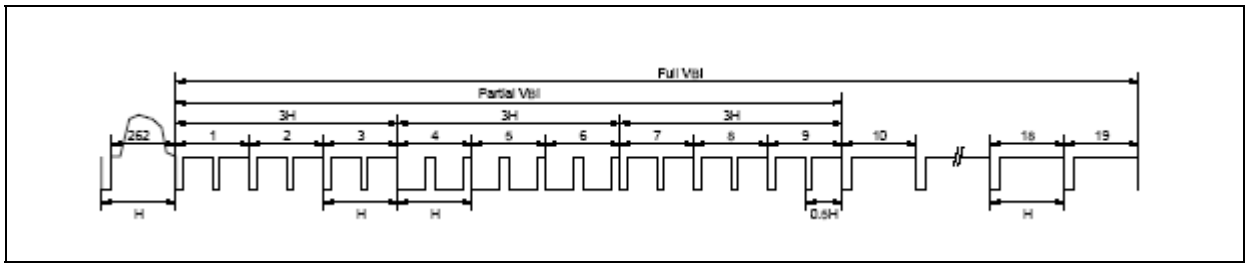


Figure 117: PAL-M typical VBI waveforms, interlaced mode (ITU-R/CCIR-525 line numbering)

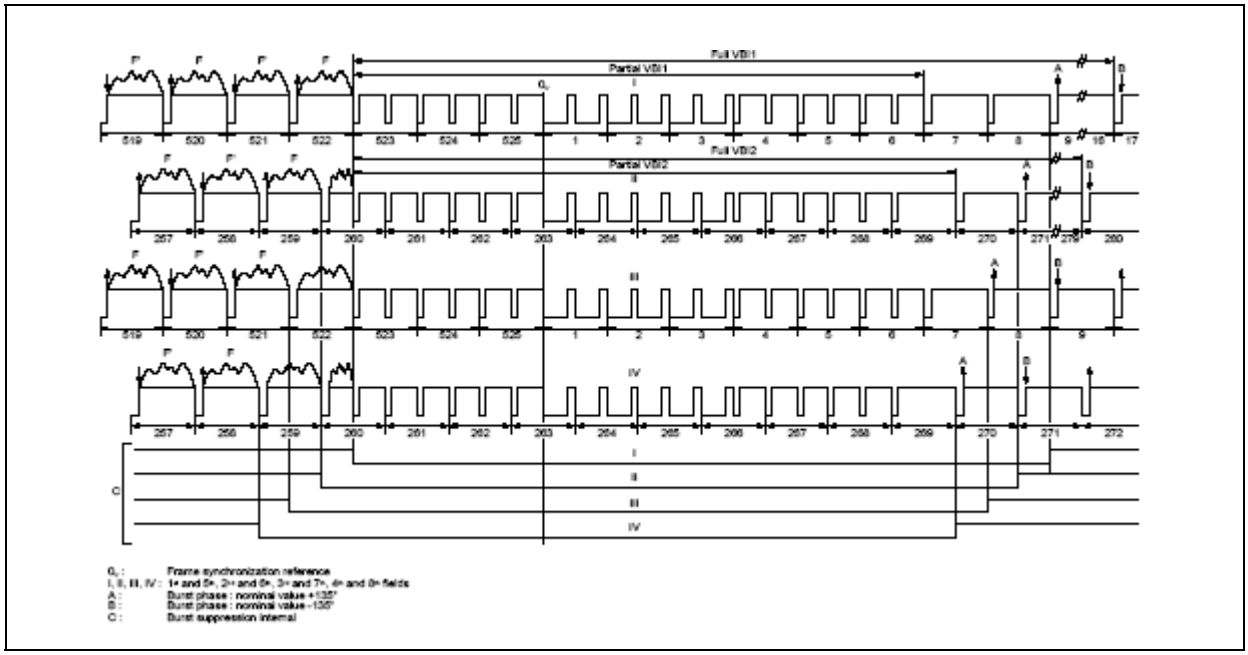


Figure 118: PAL-M typical VBI waveforms, noninterlaced mode (ITU-R/CCIR-like line numbering)

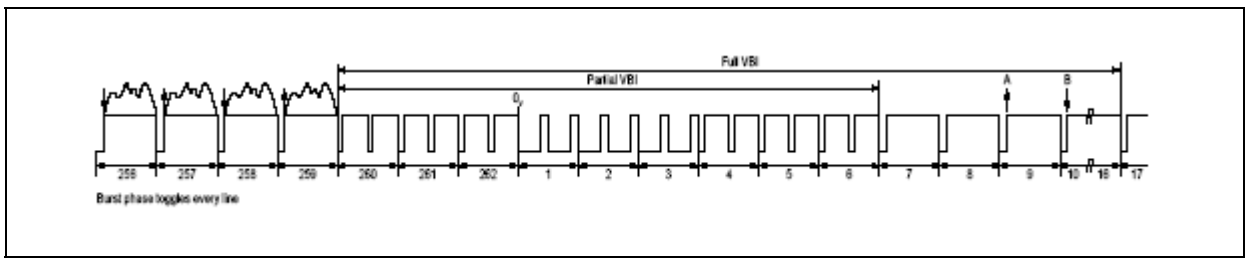


Figure 119: Horizontal blanked interval and active video timings

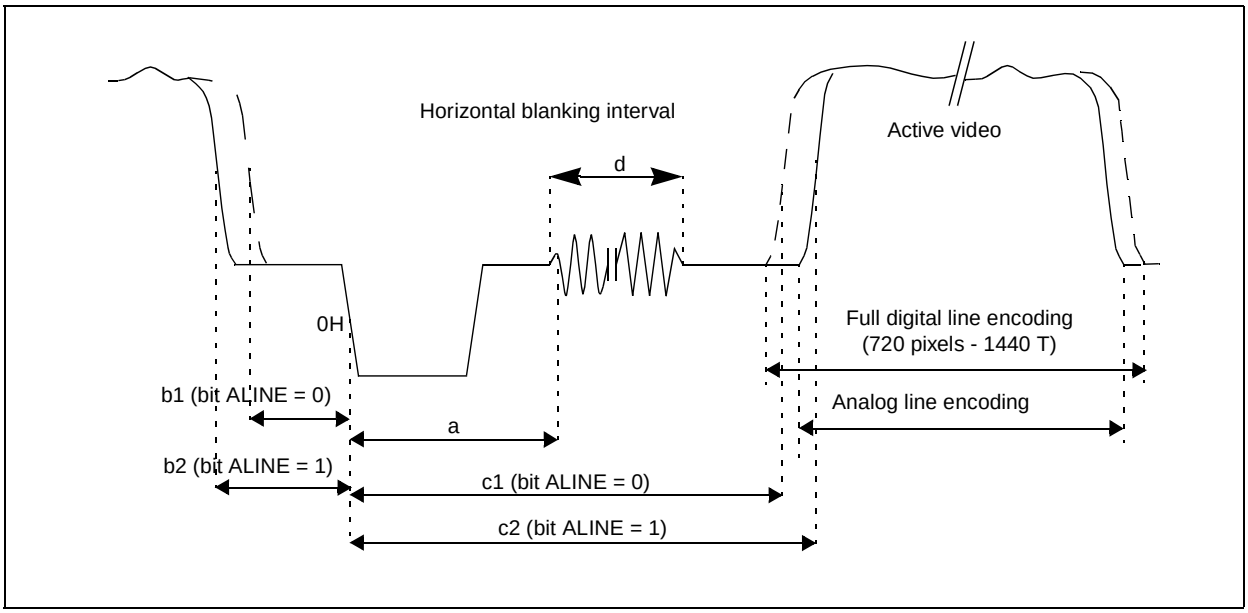


Table 84: Typical values for horizontal blanked interval and active video timings

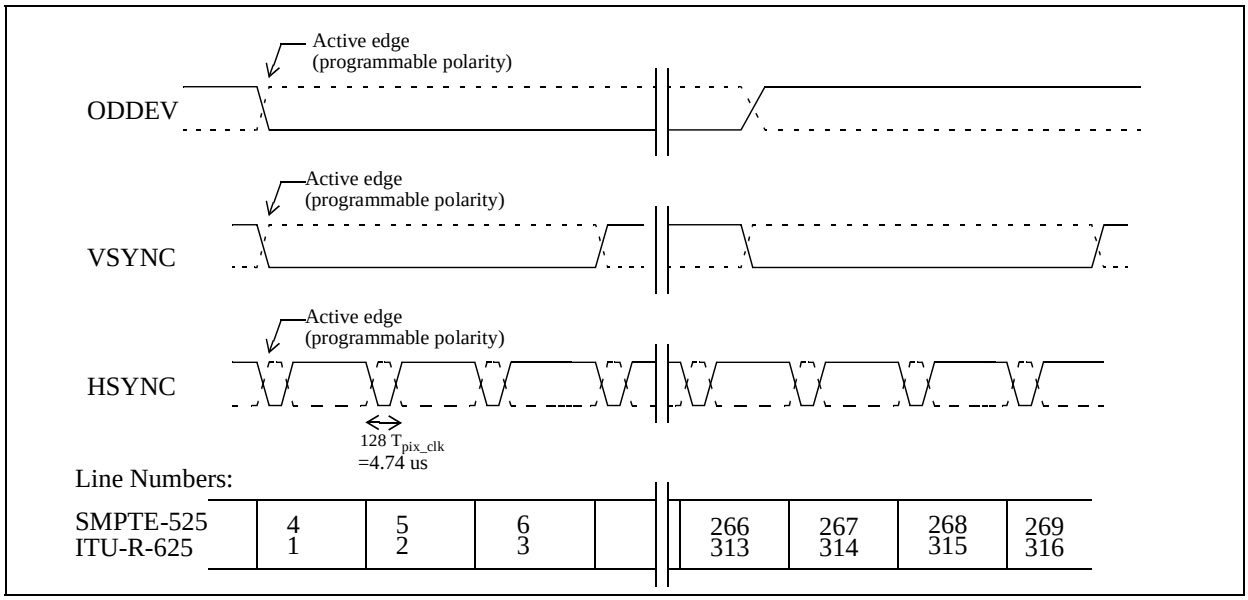
	NTSC-M	PAL-BDGHI	PAL-N	PAL-M	SECAM
a[1]	5.38 µs (even lines) 5.52 µs (odd lines)	5.54 µs (A-type) 5.66 µs (B-type)	5.54 µs (A-type) 5.66 µs (B-type)	5.73 µs (A-type) 5.87 µs (B-type)	5.37 µs
b1	1.56 µs	1.3 µs	1.3 µs	1.56 µs	1.0 µs
b2	1.56 µs	1.52 µs	1.52 µs	1.56 µs	1.52 µs
c1	8.8 µs	9.6 µs	9.6 µs	8.8 µs	9.9 µs
c2	9.41 µs	10.48 µs	10.48 µs	9.41 µs	10.48 µs
d	9 cycles of 3.58 MHz	10 cycles of 4.43 MHz	9 cycles of 3.58 MHz	9 cycles of 3.58 MHz	

1. These are typical values. Real values depend on the static offset programmed for subcarrier generation.

30.5 Master Mode

In this mode, the DENC supplies HSYNC and ODDEV sync signals (with independently programmable polarities) to drive other blocks. Refer to Figure 120 and Figure 121 for timings and waveforms. The DENC starts encoding and counting clock cycles as soon as the master mode has been loaded into the control register (Reg.0). Configuration bits “Syncout_ad[1:0]” (Reg4) allow to shift the relative position of the sync signals by up to 3 clock cycles to cope with any YCrCb phasing.

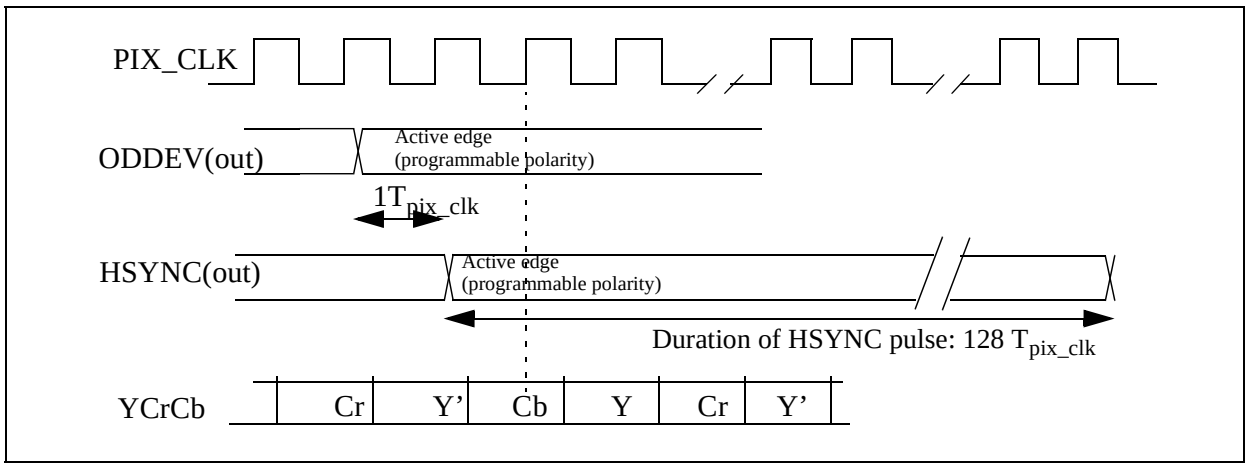
Figure 120: ODDEVEN, VSYNC and HSYNC waveforms



Note: When ODDEV is a sync input, only one edge ("the active edge") of the incoming ODDEV is taken into account for synchronization. The "non-active" edge (2nd edge on this drawing) is not critical and its position may differ by ±H/2 from the location shown.

The HSYNC pulse width indicated is valid when the DENC supplies HSYNC. In those slave modes where it receives HSYNC, only the edge defined as active is relevant, and the width of the HSYNC pulse it receives is not critical.

Figure 121: Master mode sync signals



30.6 Slave modes

30.6.1 Introduction

A number of slave modes are available: ODDEV+HSYNC based (line-based sync), VSYNC+HSYNC based (another type of line-based sync), ODDEV-only based (frame-based sync), VSYNC-only based (another type of frame-based sync), or sync-in-data based (line locked or frame locked).

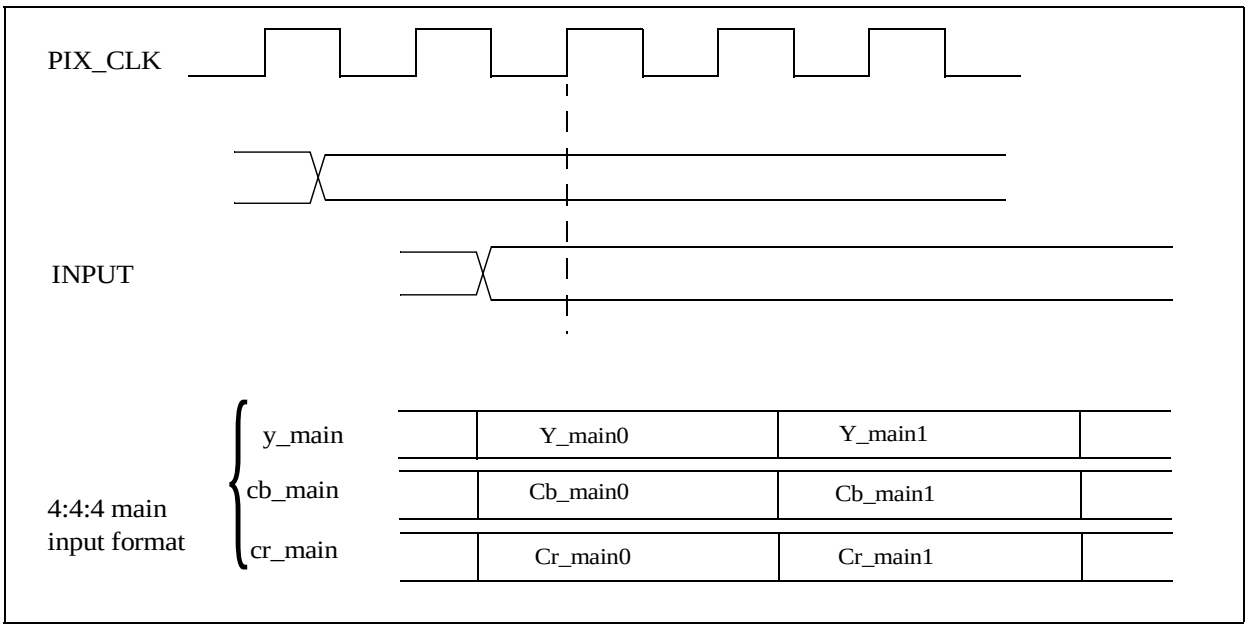
ODDEV refers to an odd/even (also known as not-top/bottom) field flag, HSYNC is a line sync signal, VSYNC is a vertical sync signal. Their waveforms are shown in Figure 120. The polarities of HSYNC and VSYNC/ODDEV are independently programmable in all slave modes.

30.6.2 Line-based synchronization

HSYNC + ODDEV based synchronization

Synchronization is performed on a line-by-line basis by locking onto incoming ODDEV and HSYNC signals (refer to Figure 122 for waveforms and timings). The polarities of the active edges of HSYNC and ODDEV are programmable and independent.

Figure 122: HSYNC + ODDEVEN based slave mode sync signals



The first active edge of ODDEV 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 at the same time as ODDEV). 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 YCrCb 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 YCrCb stream). It is, however, possible to internally delay the incoming sync signals (HSYNC + ODDEV) by up to 3 clock cycles to deal with different data and sync phases, using configuration bits SYNCIN_AD[1:0] (DEN_CFG4).

The DENC is thus fully slaved to the HSYNC signal, which means that lines may contain more or less samples than usual.

- If the digital line is shorter than its nominal value the sample counter and all internal synchronization signals are re-initialized when the early HSYNC arrives.
- If the digital line is longer than its nominal value the sample counter is stopped when it reaches its nominal end-of-line value and waits for the late HSYNC before re-initializing.

The field counter is incremented on each ODDEV transition. The line counter is reset on the HSYNC following each active edge of ODDEV.

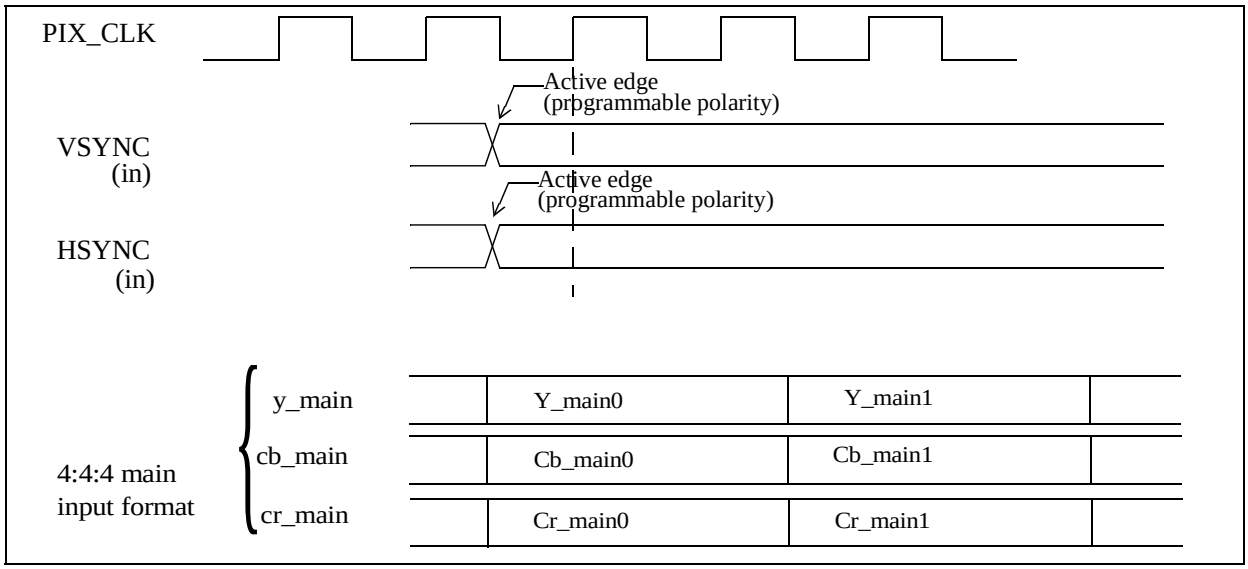
HSYNC+VSYNC Based Synchronization

Synchronization is performed on a line-by-line basis by locking onto incoming VSYNC and HSYNC signals. Refer to Figure 123 for waveforms and timings. The polarities of HSYNC and VSYNC are programmable and independent.

The incoming VSYNC signal is immediately transformed into a waveform identical to the odd/even waveform of an ODDEV signal, therefore the behavior of the core is identical to that described above for ODDEV+HSYNC based synchronization. Again, the phase relationship between HSYNC and the incoming YCrCb data is normally such that the first clock rising edge following the HSYNC active edge samples “Cb” (i.e. a ‘blue’ chroma sample within the YCrCb stream). It is however possible to internally delay the incoming sync signals (HSYNC+VSYNC) by up to 3 clock cycles to cope with different data/sync phases, using configuration bits SYNCIN_AD

The field counter is incremented on each active edge of VSYNC.

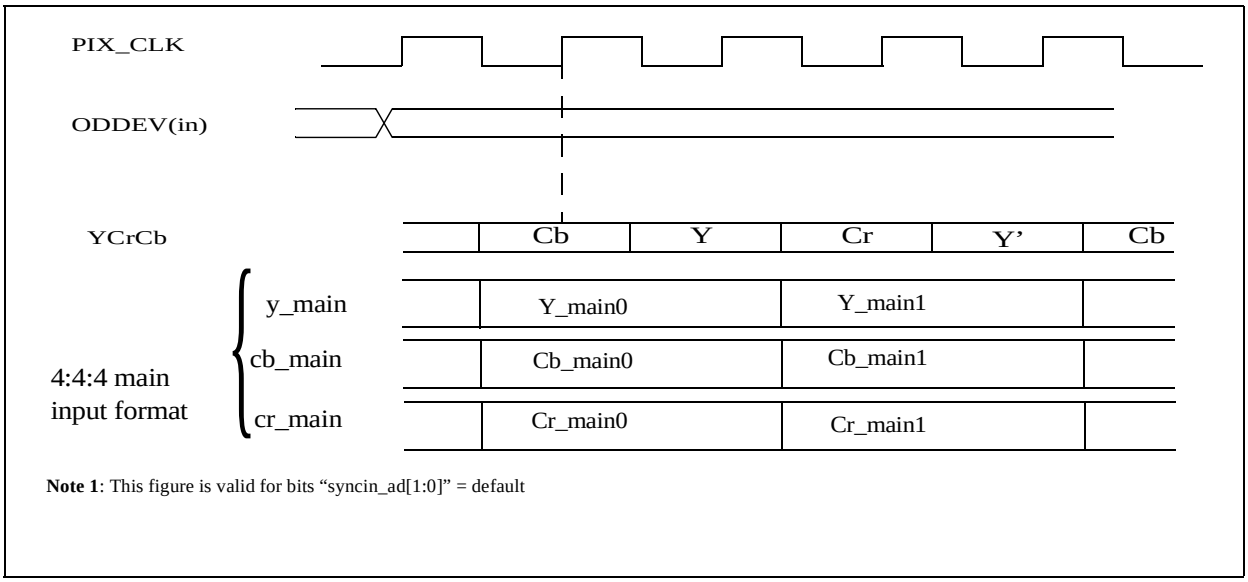
Figure 123: HSYNC + VSYNC Based Slave Mode Sync Signals



Note: This figure is valid for bits “syncin_ad[1:0]” = default

The active edges of HSYNC and VSYNC should normally be simultaneous. It is permissible that HSYNC transitions before VSYNC, but VSYNC must nottransition before HSYNC.

Figure 124: ODDEVEN Based Slave Mode Sync Signals



Note: This figure is valid for bits "syncin_ad[1:0]" = default

30.7 Autotest mode

An autotest mode is available, which causes the DENC to produce a color bar pattern in the appropriate standard, independently of the video input.

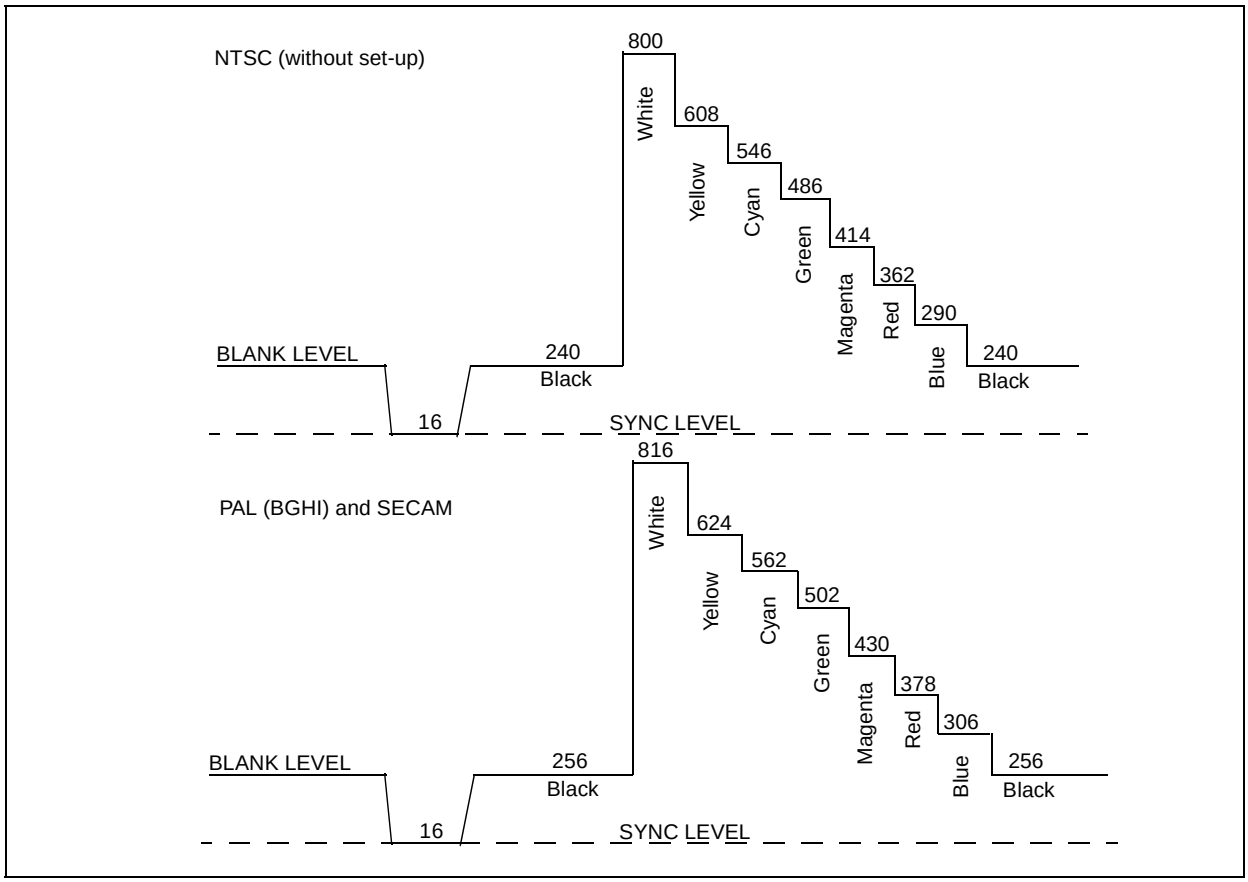
The autotest mode is started by setting field SYNC[2:0] to 111 (DEN_CFG0). As this mode sets the DENC in master mode, VSYNC/ODDEV and HSYNC signals are in output mode. Table 85 shows the decimal values of Y, Cr and Cb corresponding to the autotest color bar.This mode should not be used for quality measurements and is for DENC and DACs testing only.

Table 85: Autotest color bar Y, Cr, Cb decimal values

Color	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 graphically in Figure 125. The figure shows the static values corresponding to the input values in Table 85.

Figure 125: Luma output levels in autotest



30.8 Input demultiplexor

The incoming YCrCb data, as well as Y4 and CrCb in 4:4:4 mode, 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 configuration bits SYNCIN_AD (DEN_CFG4). Brightness, saturation and contrast are then performed on demultiplexed data (refer to registers DEN_BRIGHT to DEN_SATUR).

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. Accordingly, incoming YCrCb samples can be saturated in the input multiplexer with the following rules (after saturation, brightness and contrast stage):

- 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 having to heavily saturate the composite video codes before digital-to-analog conversion if erroneous or unrealistic YCrCb samples are input to the encoder (there may

otherwise be overflow errors in the codes driving the DACs). This avoids generating a distorted output waveform.

However, in some applications, it may be desirable to let extreme YCrCb codes pass through the demultiplexer. This is controlled using bit MAXDYN (DEN_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.

In any case, the YCrCb codes are not overridden for EAV/SAV decoder.

30.9 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 which modulates the baseband chrominance components.

The subcarrier frequency is obtained from the following equation:

$$Fsc = (24\text{-bit increment word} / 2^{24}) \times PIX\_CLK$$

Hardwired increment word values are available for each standard and can be automatically selected. Alternatively in PAL and NTSC (according to bit SELRST in DEN_CFG2), the frequency can be fully customized by programming other values into a dedicated increment word register (DEN_IDFSn). This allows NTSC-4.43 or PAL-M-4.43 to be encoded, for instance.

The following procedure is used.

1. Program the required increment in registers DEN_IDFSn.
2. Set bit SELRST_INC to 1 in register DEN_CFG5.
3. Perform a software reset (DEN_CFG6). Caution: this sets all bits from DEN_STA onwards to their default value. Alternatively, set register DEN_CFG8 bits PH_RST_MODE to 01, in which case the frequency (and phase) is updated on the beginning of next video line.

Note: if a standard change occurs after software reset, the increment value is automatically re-initialized with the hardwired or loaded value according to bit SELRST_INC.

The reset phase of the color subcarrier can also be software-controlled by registers DEN_PDFSn.

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 adjustment can be performed every line, every eighth field, every fourth field, or every second field (VALRST in DEN_CFG2). If SECAM is performed, subcarrier phase is reset every line.

30.10 Burst insertion (PAL and NTSC)

The color reference burst is inserted so it always starts with a positive zero crossing of the subcarrier sine wave (except in some cases where Macrovision anti-copy is active). 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 DEN_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 BURSTEN to 0 (DEN_CFG2).

Two strategies exist for burst insertion: one merely gates and shapes the subcarrier for burst insertion, the other is more elaborate and always starts the burst with a positive-going zero crossing. In the first case, the phase of the subcarrier when the burst starts is not controlled, with the consequence that some of its first and last cycles are more heavily distorted. The second solution guarantees smooth start and end of burst with a maximum of undistorted burst cycles, and can only be beneficial to chroma decoders. It is the solution implemented in the DENC.

While the first option gave constant burst start time but uncontrolled initial burst phase, the second solution guarantees start on a positive-going zero crossing with the consequence that two burst start locations are visible over successive lines, according to the 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 the horizontal sync) from one line to the next according to the line parity. So if the burst always starts with the same phase (positive-going zero crossing), this means the burst is inserted at time X or at time X + $T_{NTSC}/2$ after the horizontal sync tip according to the line parity, where T_{NTSC} is the duration of one cycle of the NTSC burst.

With PAL, a similar rationale holds, and again two burst start locations are possible. The subcarrier phase difference (with respect to 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-..., where A denotes A-type bursts and B denotes B-type bursts, A-type and B-type being 180° out of phase with respect to horizontal sync. So 2 locations are possible, one for A-type, the other for B-type (see Figure 115).

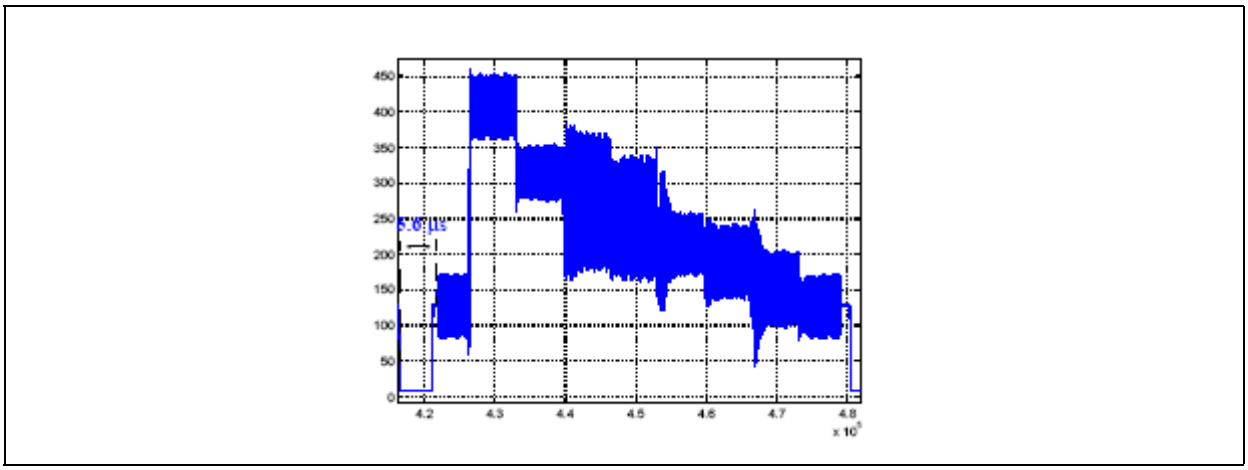
This assumes a periodic reset of the subcarrier is automatically performed (see VALRST[1:0] in DEN_CFG2). Otherwise, over several frames, the start of burst drifts within an interval of half a subcarrier cycle. This is normal and means the burst is correctly locked to the colors encoded. The equivalent effect with a gated burst approach would be the following: the start location would be fixed but the phase with which the burst starts (with respect to the horizontal sync) would drift.

30.11 Subcarrier insertion (SECAM)

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 on Figure 126.

Figure 126: SECAM color bar pattern (blue line)



In odd fields the phase of 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 a 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 bit PHI_12_SECAM in DEN_CFG7). Bit INV_PHI_SECAM allows the inversion of this sequence (π, π, 0, π, π, 0,... instead of 0, 0, π, 0, 0, π,...), in odd fields.

In even fields, the subcarrier sequence is always inverted compared to the odd field sequence.

To enable SECAM mode, a soft reset or loading of DEN_CFG0 (which is not taken into account) must be performed after setting SECAM to 1 in DEN_CFG7.

30.12 Luminance encoding

The demultiplexed Y samples are band-limited and interpolated at PIX_CLK clock rate. The resulting luminance signal is correctly scaled before insertion of any closed-captions, CGMS, VPS, WSS or teletext data, Macrovision copy-protection signals and synchronization pulses.

The interpolation filter compensates for the sin(x)/ x attenuation inherent to digital/analog conversion and greatly simplifies the output stage filter (refer to Figure 127 to Figure 129 for characteristic curves with a 27 MHz reference clock).

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 intended for low-end TV sets which are subject to cross-color if the digital source has a wide luminance bandwidth. The trap filter does not affect the S-VHS luminance output nor the RGB outputs.

Note: If SECAM is performed, a trap filter with 4.43 MHz cut-off frequency should always be enabled on the luma part of the CVBS signal (see bits MAIN_ENTRAP and TRAP_4.43 in DEN_CFG3).

A 7.5 IRE pedestal can be programmed if needed with all standards (see DEN_CFG0 and DEN_CFG7). This allows in particular encoding of Argentinian and non-Argentinian PAL-N, or Japanese NTSC (NTSC with no set-up).

Figure 127: Luma filtering including DAC attenuation

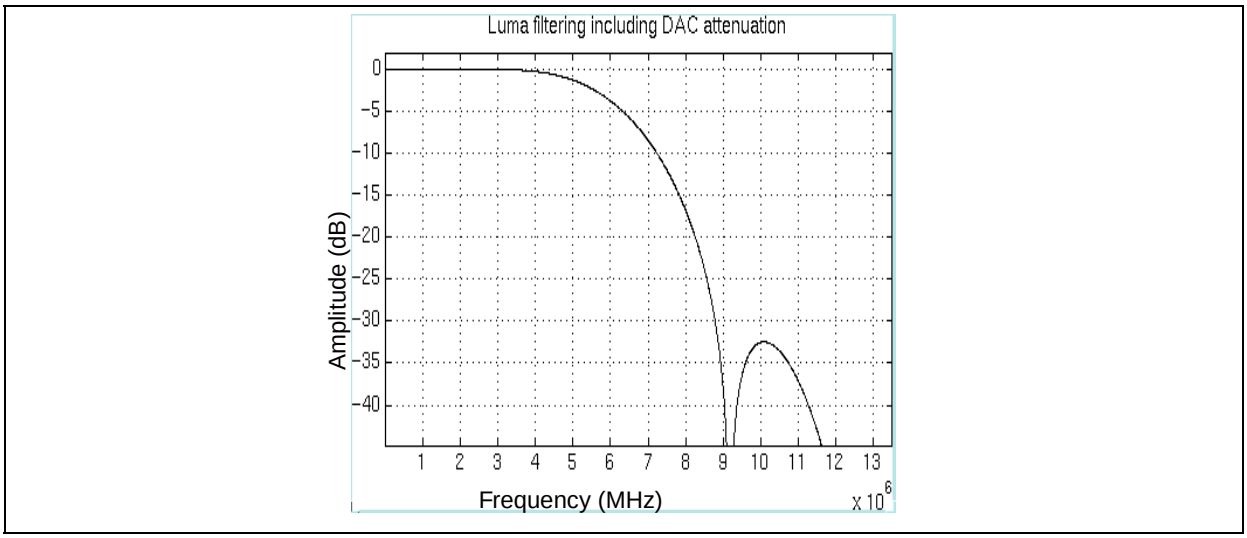


Figure 128: Luma filtering with 3.58 MHz trap, including DAC attenuation

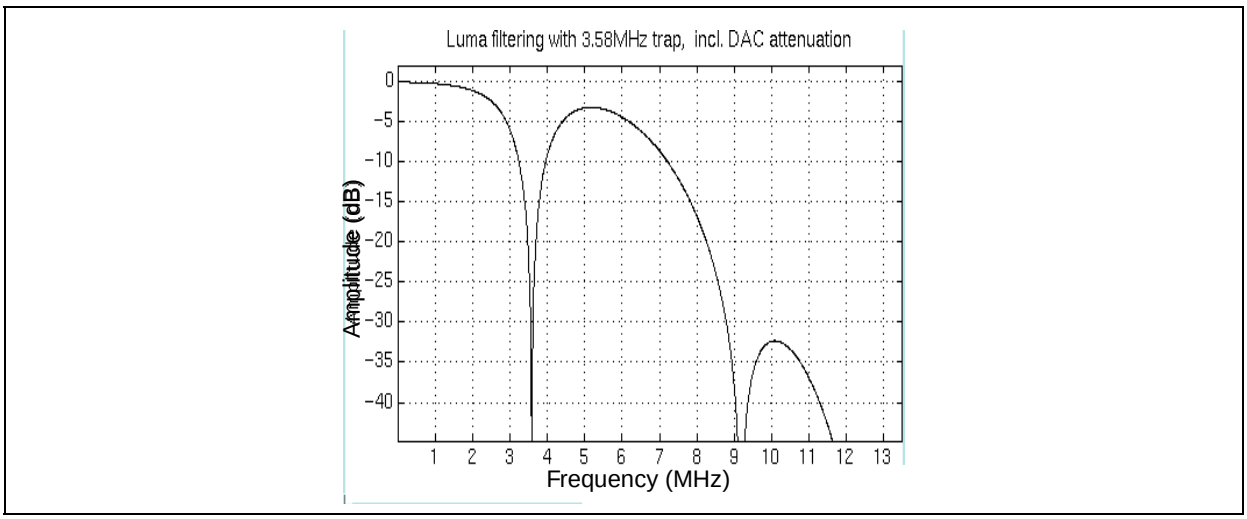
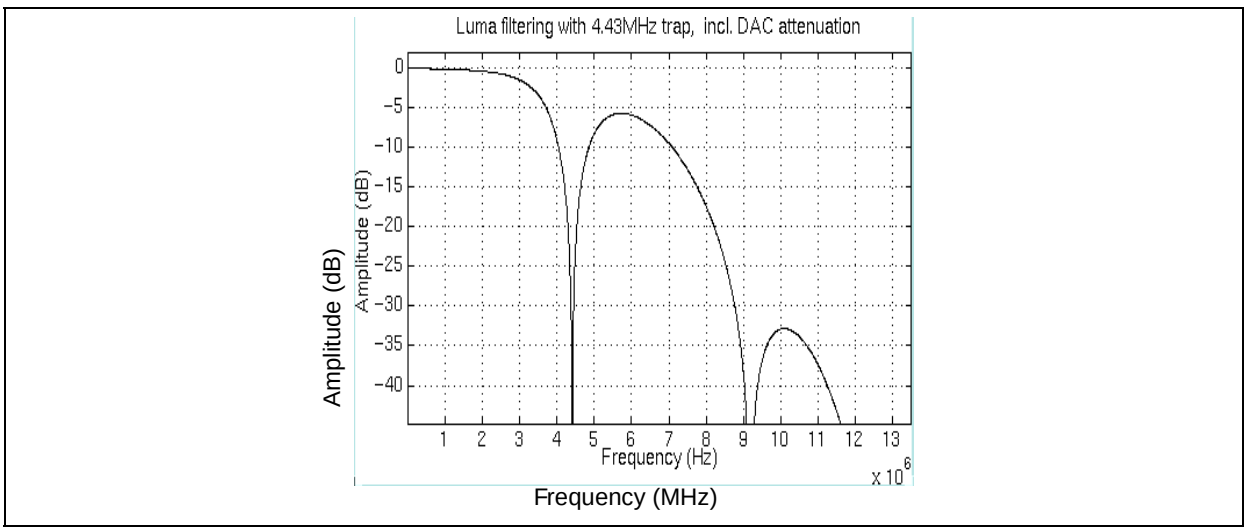


Figure 129: Luma filtering with 4.43 MHz trap, including DAC attenuation



A programmable delay can be inserted on the luminance path to offset any chroma/luma delay introduced by off-chip filtering (chroma and luma transitions being coincident at the DAC output with default delay, see DEN_CFG3 and DEN_CFG4). The luma processing as well as line and field timings in SECAM mode are identical to PAL BDGHI ones.

30.13 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 PIX_CLK clock rate. This processing eases the 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 these include filters recommended by ITU-R 624-4 and SMPTE170-M. The preprogrammed 3-dB bandwidths are 1.1, 1.3, 1.6 or 1.9 MHz, and Figure 134, Figure 135 and Figure 136 show the various frequency responses with a 27 MHz reference clock (see DEN_CFG1 for programming).

The narrower bandwidths are useful against cross-luminance artefacts, the wider bandwidths keep higher chroma contents.

The DENC allows the use of software values for filter coefficients. This can be useful 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 chroma filter is 17 taps with symmetric coefficients and operates at PIX_CLK frequency (27 MHz default).

The upsampling (13.5 to 27 MHz with 27 MHz PIX_CLK) is performed by padding with zeros.

In SECAM, 1.3 MHz low-pass and pre-emphasis filtering are performed on Dr and Db chroma components, before the frequency modulation, according to ITU-R Rec624-4. Bell filtering is performed at the end of 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). Refer to Figure 131 for frequency response of bell filter with subcarrier frequencies and clipping values.

Figure 130: SECAM chroma filtering (pre-emphasis and 1.3 MHz low pass filtering)

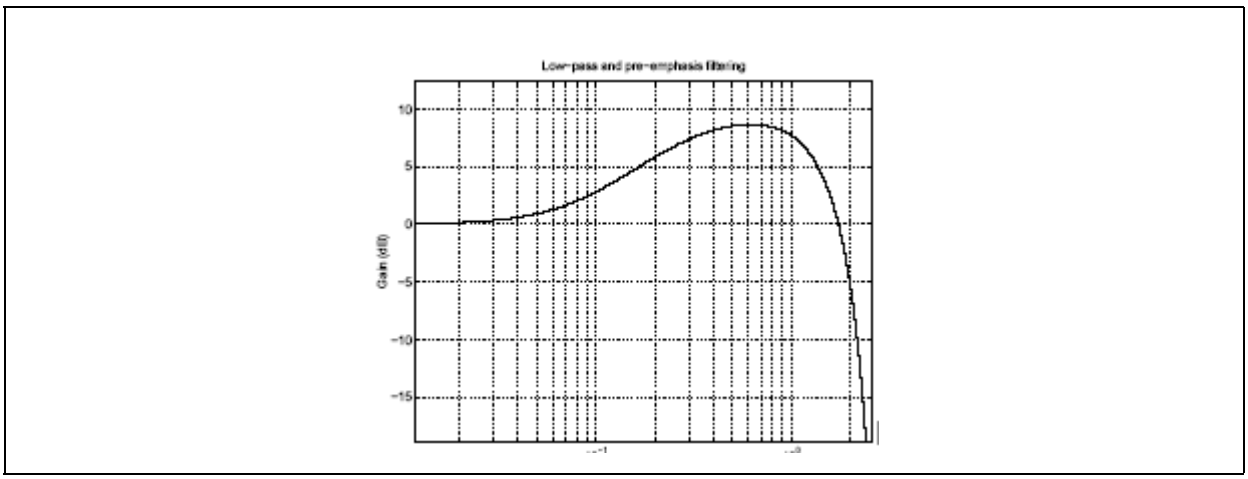


Figure 131: SECAM high frequency subcarrier pre-emphasis (bell filtering), including DAC attenuation

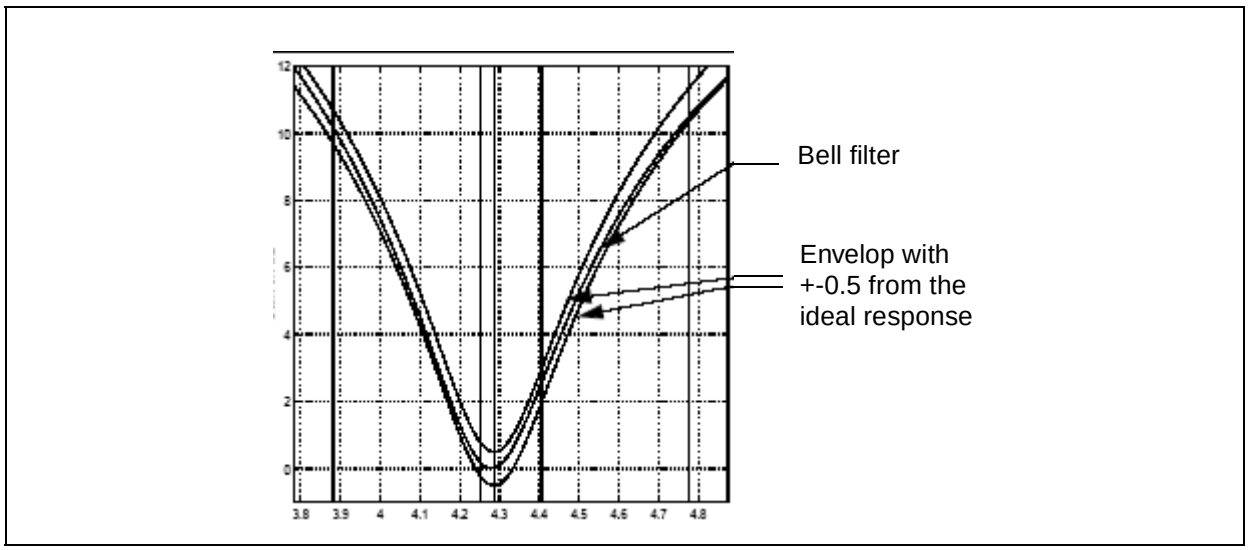
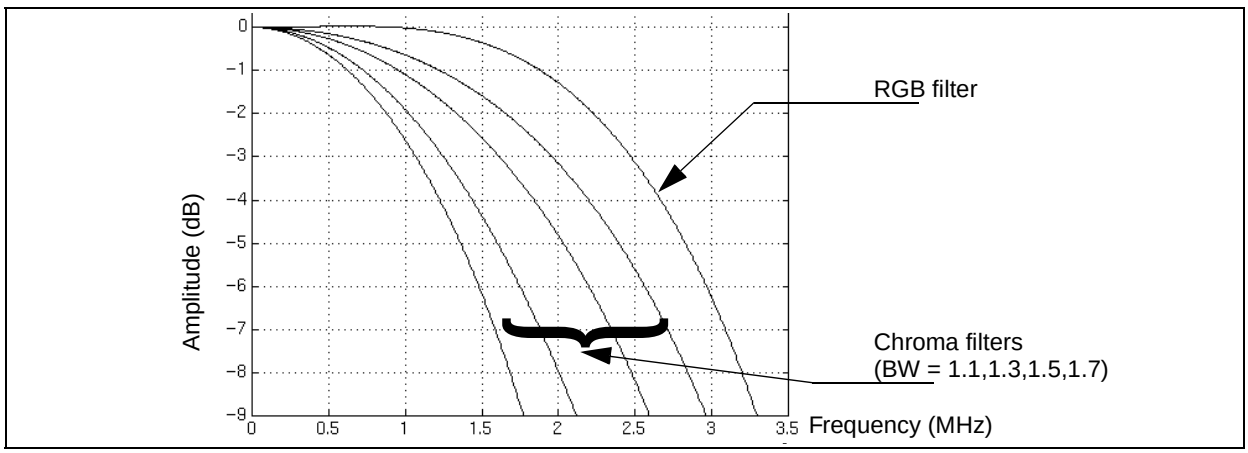


Figure 132: Various chroma filters available and RGB filter



30.14 Composite video signal generation

The composite video signal is created by adding the luminance after trap filtering (optional in PAL and NTSC, see DEN_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 graphic engines).

A color killing function is available, whereby the composite signal contains no chrominance, that is, replicates the trap-filtered luminance.

Note: This function does not suppress the chrominance on the S/VHS outputs (nevertheless suppressing the S-VHS chrominance is possible using bit BKDACn in DEN_CFG5, if chrominance signal is output on DAC n).

In some US applications, CVBS output is IF modulated before the antenna TV set input. Some IF modulators generate delay on the chroma part of the signal. This delay can be compensated in the DENC by adding delay on the luma part of the video signal before generating the composite (CVBS) signal.

Figure 133: 1.1 MHz chroma filter

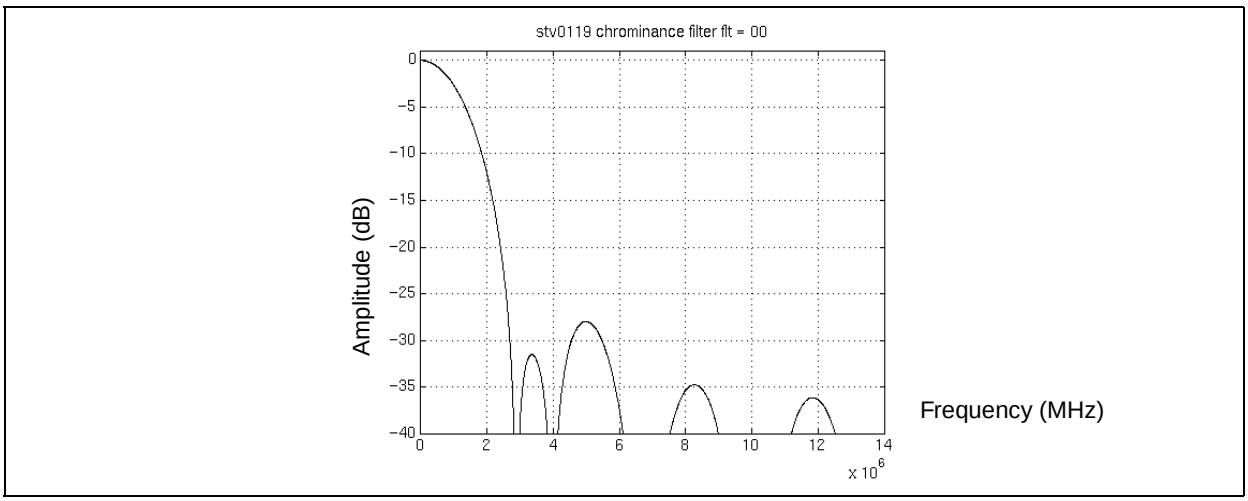


Figure 134: 1.3 MHz chroma filter

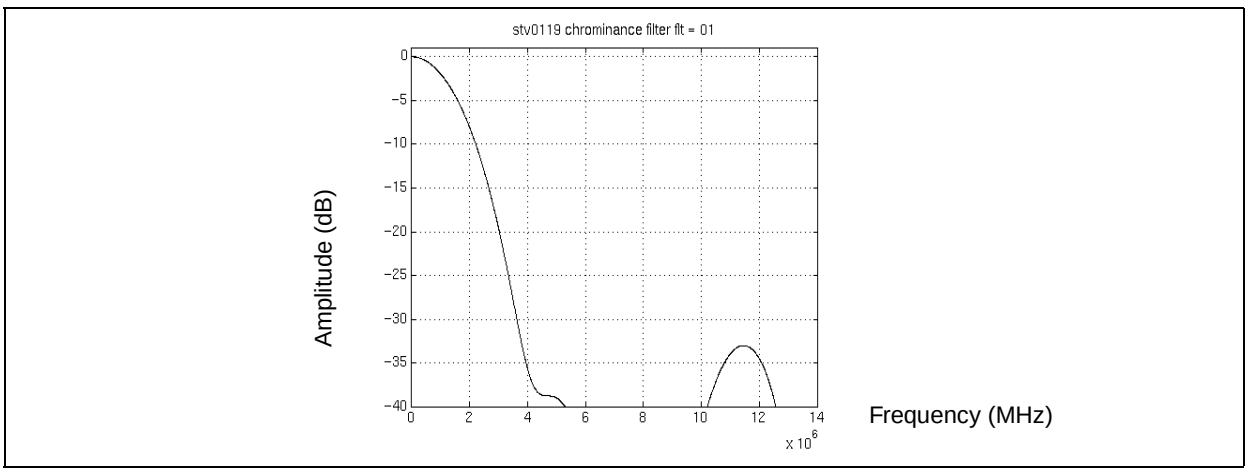


Figure 135: 1.6 MHz chroma filter

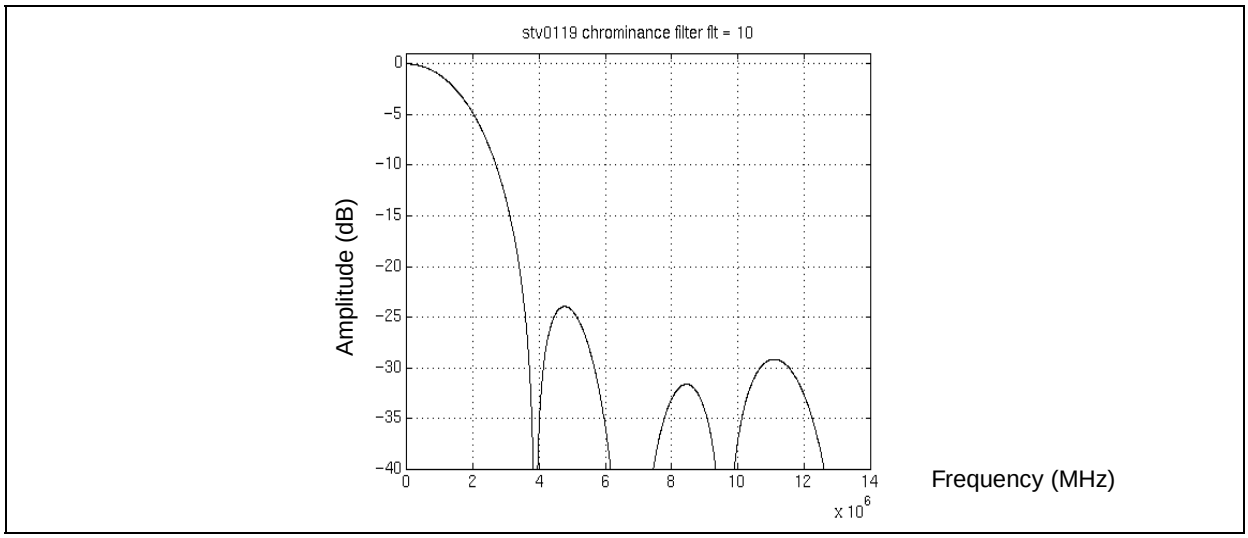
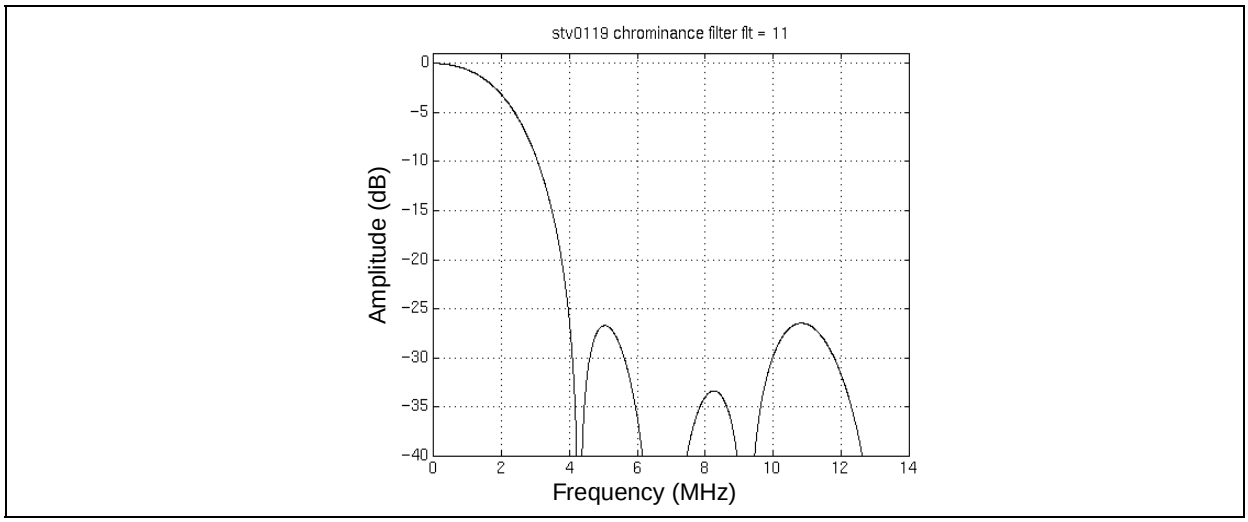


Figure 136: 1.9 MHz chroma filter



30.15 RGB and YPrPb (YUV) encoding

The Cr and Cb samples feed an x 2 interpolation filter. The resulting baseband chroma signal has a 2.45 MHz bandwidth (Figure 132) and is combined with the filtered luma component to generate R,G,B or U,V samples at PIX_CLK frequency rate. These filters are also programmable if the need arises.

Identical filtering to luma filtering (see Figure 127) is performed on all components (YCRCB_MAIN). In this case, DAC 2 output data is encoded from Y_MAIN input if YPRPB_MAIN (YUV_MAIN) configuration is used (see bits DAC123_CONF in DEN_CFG13).

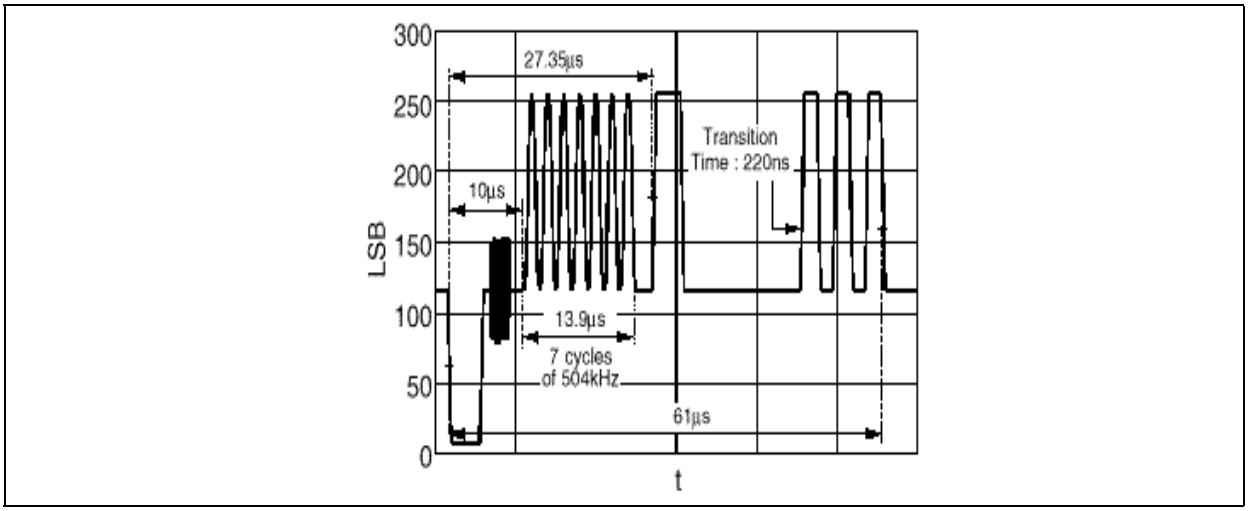
30.16 Closed-captioning

Closed-captions (or data from an extended data service as defined by the closed-captions specification) can be encoded by the DENC. The closed-caption data is delivered to the circuit through the register interface. Two dedicated pairs of bytes (two bytes per field), each pair preceded by a clock run-in and a 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 register DEN_CCCF11 (or DEN_CCCF21) contains the first byte to send (LSB first) after the start bit on the appropriate TV line in field 1 (or field 2), and DEN_CCCF12 (DEN_CCCF22) contains the second byte to send. The TV line number where data is to be encoded is programmable (DEN_CCLIF1, DEN_CCLIF2). Lines that may be selected include those used by the StarSight data broadcast system. Closed-caption data has less priority than CGMS but more priority than Macrovision anticopy signals programmed for the same line.

Figure 137: Example of closed-caption waveform



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 (see Figure 137).

When closed-captioning is on (bits CC1/CC2 in DEN_CFG1), the CPU should load the relevant registers (DEN_CCCF11 and 2, or DEN_CCCF21 and 2) once every frame at most (although there is some margin due to the double-buffering). Bits BUF2_FREE and BUF1_FREE are set in DEN_STA if the closed-caption data registers are loaded too frequently. These can be used to regulate loading rate.

The closed-caption encoder considers that closed-caption data has been loaded and is valid on completion of the write operation into register DEN_CCCF12 for field 1, into DEN_CCCF22 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, the circuit outputs two US-ASCII null characters with odd parity after the start bit.

30.17 CGMS encoding

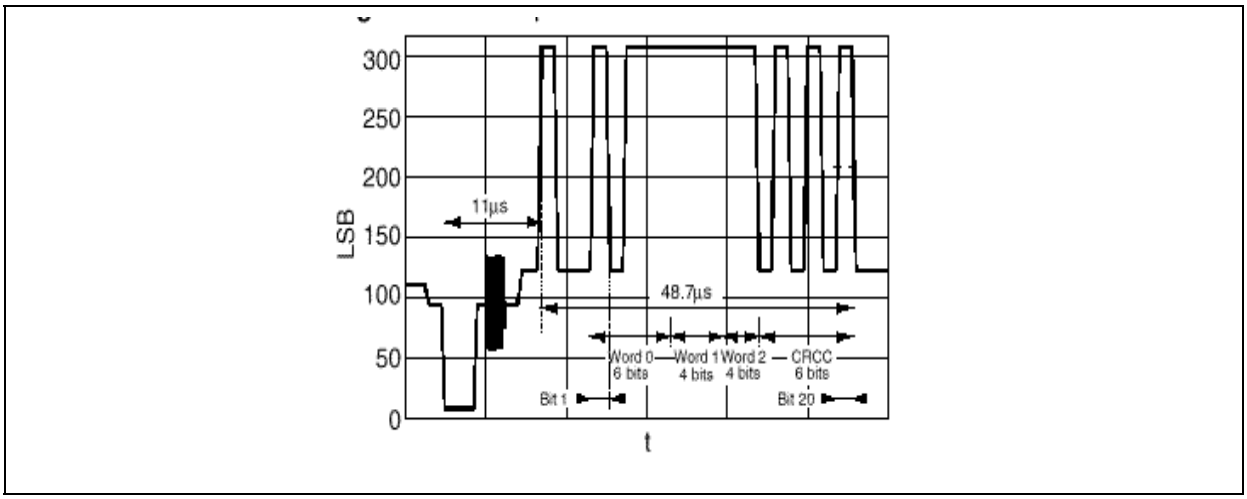
CGMS (copy generation management system, also known as VBID and described by standard CPX-1204 of EIAJ) data can be encoded by the circuit. Three bytes (20 significant bits) are delivered to the chip via the register interface. Two reference bits (1 then 0) are encoded first, followed by 20 bits of CGMS data (including a cyclic redundancy check sequence, not computed

by the device and supplied to it as part of the 20 data bits). The reference bits are generated locally by the DENC. Refer to Figure 138 for a typical CGMS waveform.

When CGMS encoding is enabled, the CGMS (see bit ENCGMS in DEN_CFG3) waveform is present once in each field, on lines 20 and 283 (SMPTE-525 line numbering). CGMS data has priority over any Macrovision anticopy signals programmed for the same line.

The CGMS data register is double-buffered, which means that it can be loaded anytime (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 DEN_CGMS3. CGMS always has priority over any Macrovision anticopy signals programmed for the same line.

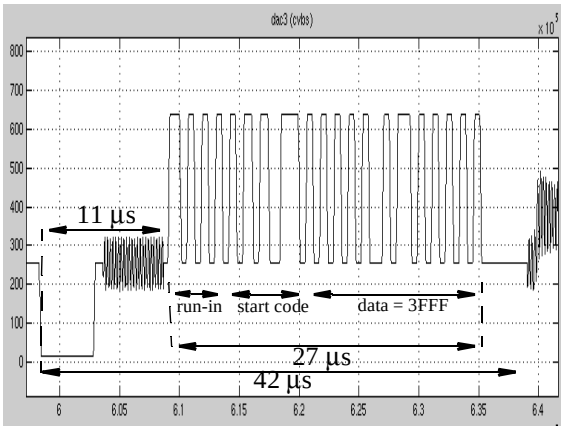
Figure 138: Example of CGMS waveform



30.18 WSS encoding

The DENC allows WSS (wide screen signalling) in 625 line format, according to ETS 300 294 standard. Two bytes are delivered to the circuit through the parallel interface into two dedicated registers (see registers DEN_WSSn).

Figure 139: Example of WSS waveform

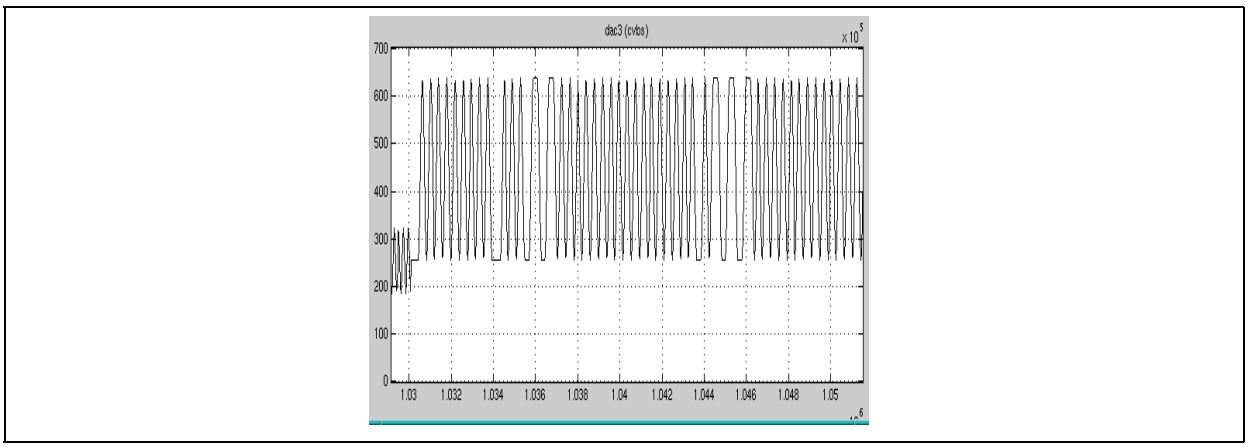


When WSS encoding is enabled (see bit ENWSS in DEN_CFG3), the waveform is present on the first half of line 23, on each frame. Refer to Figure 139 for an example of WSS waveform. Data is preceded by a run-in sequence and a start code generated locally by the DENC. Different WSS data for same line are not supported. The same WSS waveform is simultaneously present for the two CVBS outputs.

30.19 VPS encoding

VPS data (as defined by ETS 300 231 communication from June 1993) can be encoded by the DENC on line 16 (CCIR) for 625 line PAL and SECAM television systems. VPS data is delivered to the circuit through the parallel interface (see registers DEN_DACC to DEN_BRIGHT). 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 ENVPS (DEN_CFG7). Figure 140 shows an example of a VPS waveform.

Figure 140: Example of VPS waveform



30.20 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 CPU (or demultiplexer) must sort out incoming data packets and, in particular, store teletext packets in a buffer, which then passes them to the DENC on request. This is performed by the GDMA + TTXT_INT block inside the TVO block.

30.20.1 Signals exchanged

The DENC and the TTXT_INT block exchange two signals: TTXS (teletext sync) from the DENC to the TTXT_INT block and TTXD (teletext data) from the TTXT_INT block to the DENC.

TTXS is a request signal generated on selected lines. In response to this signal, the TTXT_INT block sends teletext bits to the DENC for insertion of a teletext line into the analog video signal (360 in case of teletext B - WST in PAL or 288 in case of teletext C - NABTS in NTSC).

The teletext system is selected by register DEN_TTXCFG bits TTXT_ABCD[1:0].

The duration of TTXS corresponds to the number of bits to be transmitted for the teletext standard selected (288, 296, 320 or 360 bits) with the protocol described below. In teletext B and 625 line systems (WST), the duration of the TTXS window is 1402 reference clock periods, which corresponds to the duration of 360 teletext bits (see Section 30.20.2).

- [9 x (33 x 4 PIX_CLK + 4 x 3 PIX_CLK) + (25 x 4 PIX_CLK + 2 x 3 PIX_CLK)]

In teletext C and 525 line systems (NABTS), this duration is 1121 master clock periods.

- [7 x (33 x 4 PIX_CLK + 4 x 3 PIX_CLK) + (26 x 4 PIX_CLK + 3 x 3 PIX_CLK)]

Following the TTXS rising edge, the encoder expects data from the TTXT_INT block after a programmable number (2 to 9) of master clock periods (PIX_CLK). Data is transmitted synchronously with the master clock at an average rate of 6.9375 Mbit/s (27 MHz reference clock) according to the protocol described below. It consists, in transmission order, of 16 clock

run-in bits, 8 framing code bits and n that represents one teletext packet (336 bits, 42 bytes, in teletext B - 625 lines and 288 bits, 36 bytes, in teletext C -525 line systems). If more than n bits are transmitted (n depends on the standard selected), they are ignored by the DENC. Two bits change in framing code from one teletext standard to another, so the DENC can blank them and put the right bits instead, according to the standard selected (see DEN_TTXCFG and DEN_DACC). This can be useful if the teletext interface can deliver only the framing code for teletext B-625 line standard (WST).

30.20.2 Transmission protocol

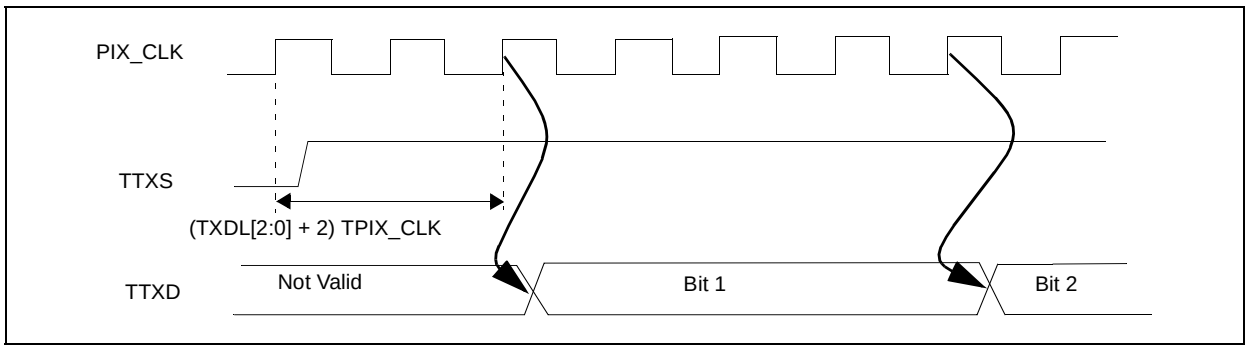
So as to transmit the teletext data bits at an average rate of 6.9375 Mbit/s, which is about 1/3.89 times the master clock frequency (case of 27 MHz master clock), the following scheme is adopted.

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.

30.20.3 Programming

- TTXS rising to first valid sample delay programming
  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 141 depicts this graphically for N = 0). N is programmable from 0 to 7 (that is, overall delay is programmable from two to nine PIX_CLK cycles) via 3 dedicated bits located in register DEN_TTXn: TTXDEL[2:0].

Figure 141: TTXT rising to first-valid-sample delay for TTXDEL[2:0] = 0



- Teletext line selection
  Five dedicated registers (from registers DEN_TTXn for CVBS_main) allow to program teletext encoding on various lines of the vertical blanking interval (VBI) of each field. Each line in VBI can be selected independently. Refer to registers DEN_TTXn for details.
  Moreover, full page teletext encoding is possible using bit FP_TTXT from DEN_TTX1. In this case, teletext is encoded on lines 24 to 311 and 336 to 623 (ITU-R line numbering), in addition to lines already programmed by registers DEN_TTXn. When full page teletext is performed, no video data is encoded (Y_MAIN input stream is ignored). Different teletext data for both the composite videos is not supported.
- Amplitude of encoding
  In 525 lines systems the default teletext amplitude is 70 IRE. If TTXT100IRE is set (DEN_DACC), the teletext amplitude is 100 IRE.

- Teletext pulse shape

  The shape and amplitude of a single pulse are depicted in Figure 142, its relative power spectral density is given in Figure 143 and is basically zero at frequencies above 5 MHz, as required by the World System Teletext specification.

Figure 142: Shape and amplitude of a single teletext symbol

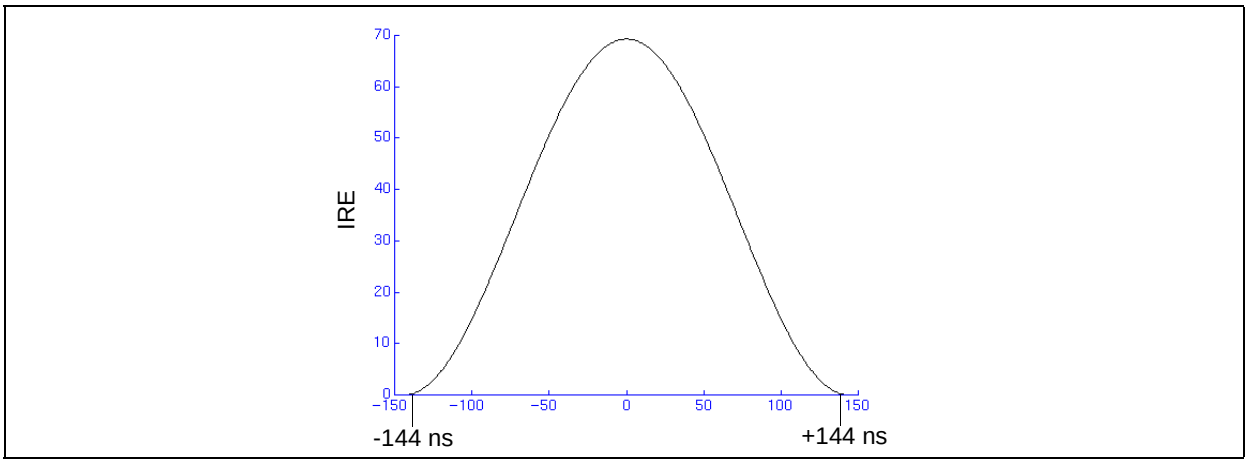


Figure 143: Linear PSD scale

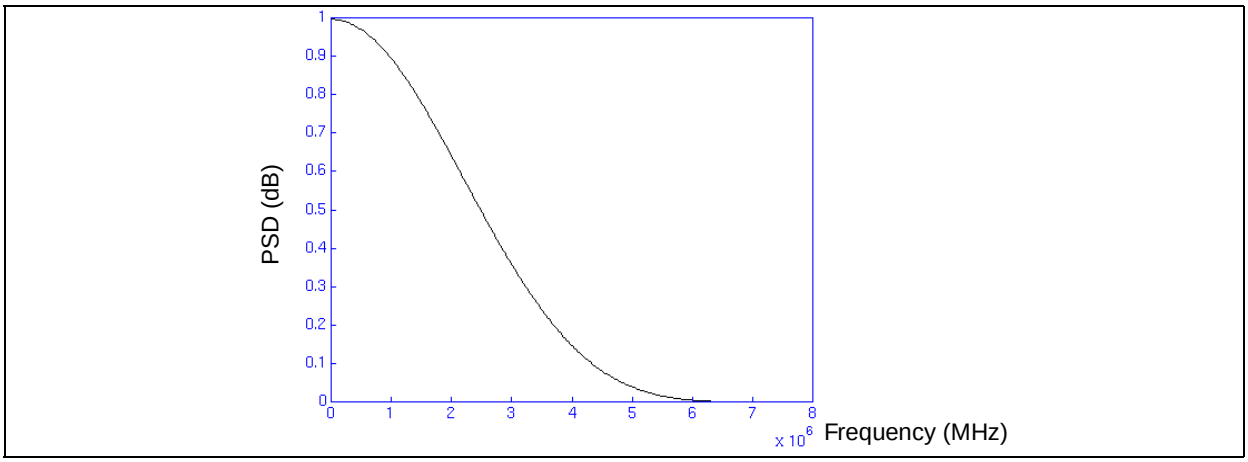
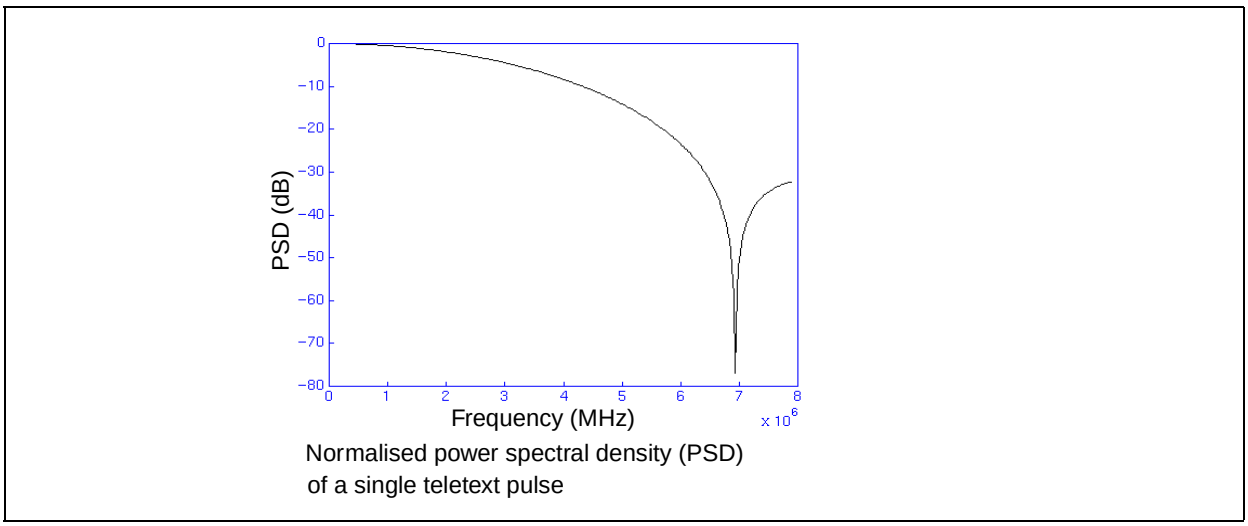


Figure 144: Logarithmic PSD Scale



30.21 Macrovision copy protection process rev7.01/6.1

The chrominance, luminance and composite video signals on CVBS_MAIN and R, G, B video signals can be altered according to the Macrovision™ copy protection process, Revision 7.01. and Revision 6.1.

A programming document is available to those customers who have executed a license or a non-disclosure agreement with Macrovision Corporation.

30.22 CVBS, S-VHS(Y/C), RGB and UV outputs

The STx5119 provides signals to the main output. These signals are designated as follows:

- CVBS: CVBS_MAIN
- S-VHS: Y_MAIN/C_MAIN
- RGB: RGB_MAIN
- UV: UV_MAIN

Video signals can be directed to three analog output pins through 10-bit DACs operating at the reference clock frequency. The available combinations are described by bits DAC123_CONF of DEN_CFG13.

Note: DAC6 is always hardwired to CVBS_MAIN and hence not programmable.

The C to Y peak-to-peak amplitude ratio can be modified in both CVBS and VHS (Y/C) outputs (see DEN_CMULT and TTXS, Register 65 bits c_mult[3:0]).

The 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. It is possible to use the maximum DAC dynamic when RGB is being output by changing RGB_MAXDYN to1 of DEN_CFG13 when DAC123_CONF is 010.

U and V outputs have 0.7 V peak-to-peak amplitude if 100/0/100/0 color bar pattern is input. In all cases, UV outputs can be multiplied by 0.75 to 1.22 factor according to DEN_DAC13 and DEN_DAC34 bits DAC1_MULT[5:0] and DAC1_MULT[5:0] respectively.

A single external analog power supply pair is used for all DACs, but two independent pairs of current and voltage references are needed. Each current reference pin normally connects externally to a resistor tied to the analog ground, whilst each voltage reference pin normally connects to a capacitance tied to the analog ground.

The internal current sources are independent from the positive supply, and the consumption of the DACs is constant whatever the codes converted.

Any unused DAC may be independently disabled by software, in which case its output is at neutral level (blanking for luma and composite outputs, no color for chroma output, black for RGB and UV outputs). For applications where a single CVBS output is required, the RGB/CVBS and S-VHS/UV DAC should be disabled and pins VIDANAREXT[1:0] tied to the analog power supply.

Due to the 3.3 V power supply used, the output swing of the DACs is about 1 Vp-p. Therefore some external gain may be required, which, combined with the recommended output filtering stage, means active filtering. For this active filtering stage to be very simple, it is possible to invert the DAC outputs by programming a user register bit (DEN_CFG5). Code N becomes code 1024 - N, that is, the resulting waveform undergoes a symmetry around the mid-swing code.

31 Digital encoder (DENC) registers

Note: Do not change the value of reserved bits from their default values.

Addresses are provided as the *DENCBaseAddress* + offset.

The *DENCBaseAddress* is: 0x2090 0000.

Table 86: Digital encoder (DENC) registers

Register name	Description	Address offset	Type
DEN_CFG0	Configuration 0, see page 322	0x0000	R/W
DEN_CFG1	Configuration 1, see page 323	0x0004	R/W
DEN_CFG2	Configuration 2, see page 324	0x0008	R/W
DEN_CFG3	Configuration 3, see page 325	0x000C	R/W
DEN_CFG4	Configuration 4, see page 326	0x0010	R/W
DEN_CFG5	Configuration 5, see page 327	0x0014	R/W
DEN_CFG6	Configuration 6, see page 328	0x0018	R/W
DEN_CFG7	Configuration 7, see page 329	0x001C	R/W
DEN_CFG8	Configuration 8, see page 329	0x0020	R/W
DEN_STA	DENC status, see page 330	0x0024	RO
DEN_IDFS1	Increment 1 for digital frequency synthesizer, see page 331	0x0028	R/W
DEN_IDFS2	Increment 2 for digital frequency synthesizer, see page 331	0x002C	R/W
DEN_IDFS3	Increment 3 for digital frequency synthesizer, see page 331	0x0030	R/W
DEN_PDFS1	Static phase offset 1 for digital frequency synthesizer, see page 332	0x0034	R/W
DEN_PDFS2	Static phase offset 2 for digital frequency synthesizer, see page 332	0x0038	R/W
DEN_WSS1	WSS data 1, see page 334	0x003C	R/W
DEN_WSS2	WSS data 2, see page 334	0x0040	R/W
DEN_DAC13	DAC1 and DAC3 multiplying factors, see page 335	0x0044	R/W
DEN_DAC34	DAC3 and DAC4 multiplying factors, see page 335	0x0048	R/W
DEN_LINE0	LTARG[8:0] and LREF[8:0], see page 336	0x0054	R/W
DEN_LINE1		0x0058	R/W
DEN_LINE2		0x005C	R/W
DEN_VPS1	VPS data 1, see page 336	0x0064	R/W
DEN_VPS2	VPS data 2, see page 336	0x0068	R/W
DEN_VPS3	VPS data 3, see page 336	0x006C	R/W
DEN_VPS4	VPS data 4, see page 336	0x0070	R/W
DEN_VPS5	VPS data 5, see page 336	0x0074	R/W
DEN_VPS6	VPS data 6, see page 336	0x0078	R/W
DEN_CGMS1	CGMS data 1, see page 337	0x007C	R/W
DEN_CGMS2	CGMS data 2, see page 337	0x0080	R/W
DEN_CGMS3	CGMS data 3, see page 337	0x0084	R/W

Table 86: Digital encoder (DENC) registers

Register name	Description	Address offset	Type
DEN_TTX1	Teletext block configuration 1, see page 337	0x0088	R/W
DEN_TTX2	Teletext block configuration 2, see page 337	0x008C	R/W
DEN_TTX3	Teletext block configuration 3, see page 337	0x0090	R/W
DEN_TTX4	Teletext block configuration 4, see page 337	0x0094	R/W
DEN_TTX5	Teletext block configuration 5, see page 337	0x0098	R/W
DEN_CCCF11	Closed-caption characters/extended data field 1 (LSBs), see page 339	0x009C	R/W
DEN_CCCF12	Closed-caption characters/extended data field 1 (MSBs), see page 339	0x00A0	R/W
DEN_CCCF21	Closed-caption characters/extended data field 2 (LSBs), see page 339	0x00A4	R/W
DEN_CCCF22	Closed-caption characters/extended data field 2 (MSBs), see page 339	0x00A8	R/W
DEN_CCLIF1	Closed-caption characters/extended line insertion field 1, see page 340	0x00AC	R/W
DEN_CCLIF2	Closed-caption characters/extended line insertion field 2, see page 341	0x00B0	R/W
DEN_TTXCFG	Teletext configuration, see page 342	0x0100	R/W
DEN_DACC	DACC multiplier and TTXS, see page 342	0x0104	R/W
DEN_BRIGHT	Brightness, see page 343	0x0114	R/W
DEN_CONTR	Contrast, see page 343	0x0118	R/W
DEN_SATUR	Saturation, see page 344	0x011C	R/W
DEN_CFCOEF0	Chroma filter coefficient 0 (main video), see page 345	0x0120	R/W
DEN_CFCOEF1	Chroma filter coefficient 1 (main video), see page 345	0x0124	R/W
DEN_CFCOEF2	Chroma filter coefficient 2 (main video), see page 345	0x0128	R/W
DEN_CFCOEF3	Chroma filter coefficient 3 (main video), see page 345	0x012C	R/W
DEN_CFCOEF4	Chroma filter coefficient 4 (main video), see page 345	0x0130	R/W
DEN_CFCOEF5	Chroma filter coefficient 5 (main video), see page 345	0x1034	R/W
DEN_CFCOEF6	Chroma filter coefficient 6 (main video), see page 345	0x0138	R/W
DEN_CFCOEF7	Chroma filter coefficient 7 (main video), see page 345	0x013C	R/W
DEN_CFCOEF8	Chroma filter coefficient 8 (main video), see page 345	0x0140	R/W
DEN_CDEL_LFC	Chroma delay and luma filter control, see page 346	0x0144	R/W
DEN_LFCOEF0	Luma filter coefficient 0, see page 347	0x0148	R/W
DEN_LFCOEF1	Luma filter coefficient 1, see page 347	0x014C	R/W
DEN_LFCOEF2	Luma filter coefficient 2, see page 347	0x0150	R/W
DEN_LFCOEF3	Luma filter coefficient 3, see page 347	0x0154	R/W
DEN_LFCOEF4	Luma filter coefficient 4, see page 347	0x0158	R/W

Table 86: Digital encoder (DENC) registers

Register name	Description	Address offset	Type
DEN_LFCOEF5	Luma filter coefficient 5, see page 347	0x015C	R/W
DEN_LFCOEF6	Luma filter coefficient 6, see page 347	0x0160	R/W
DEN_LFCOEF7	Luma filter coefficient 7, see page 347	0x0164	R/W
DEN_LFCOEF8	Luma filter coefficient 8, see page 347	0x0168	R/W
DEN_LFCOEF9	Luma filter coefficient 9, see page 347	0x016C	R/W
DEN_CFG10	Configuration 10, see page 348	0x0170	R/W
DEN_CFG11	Configuration 11, see page 348	0x0174	R/W
DEN_CFG13	Configuration 13, see page 349	0x017C	R/W
DEN_HUE_CONTROL	Hue control, see page 349	0x01A4	R/W
DEN_DAC2	DAC2 multiplying factors, see page 350	0x01A8	R/W
DEN_DAC6	DAC6 multiplying factors, see page 351	0x01AC	R/W

DEN_CFG0 Configuration 0

7	6	5	4	3	2	1	0
STD1	STD0	SYNC2	SYNC1	SYNC0	POLH	POLV	FREERUN

Address: *DENCBaseAddress* + 0x0000
Type: Read/write
Buffer: Immediate
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 DEN_CFG7 is set, STD1 and STD0 are not taken into account.

[5:3] **SYNC**[2:0]: configuration.
000: BnotTREF-only based slave mode (frame locked)
001: F based slave mode (frame locked)
010: BNOTTREF + HREF based slave mode (line locked)
011: reserved
100: VSYNC- only based slave mode
101: VSYNC + HSYNC based slave mode (line locked)
110: reserved
111: autotest mode (color bar pattern)

[2] **POLH**: horizontal sync. Active edge of HREF selection (when input) or polarity of HREF (when output)
0: HREF is a negative pulse (128 TPIX_CLK wide) or falling edge is active
1: HREF is a positive pulse (128 TPIX_CLK wide) or rising edge is active

[1] **POLV**: vertical sync. Active edge of BnotTREF/Vsync selection.
0: falling edge of BnotTREF flags start of field1 (odd field) or Vsync is active low
1: rising edge of BnotTREF flags start of field1 (odd field) or Vsync is active high

[0] **FREERUN**: Sync: active edge of BNOTTREF. This bit is taken into account in BNOTTREF-only and F-based slave modes. It is irrelevant to other sync modes.
0: disabled 1: enabled

DEN_CFG1 Configuration 1

7	6	5	4	3	2	1	0
BLKLI	FLT1	FLT0	SYNCOK	COKI	SETUP_MAIN	CC2	CC1

Address: *DENCBaseAddress* + 0x0004
Type: Read/write
Buffer: Immediate
Reset: 0x44
Description:

[7] **BLKLI** [1]: Vertical blanking interval selection for active video lines area.

0: (partial blanking) Only the following lines inside the vertical interval are blanked:[2]
NTSC-M: lines [1:9], [263(half):272] (525-SMPTE),
PAL-M: lines [523:6], [260(half):269] (525-CCIR)
other PAL: lines [623(half):5], [311:318] (625-CCIR)
This mode preserves embedded VBI data within the incoming YCrCb, for example teletext (lines [7:22] and [320:335]), wide screen signalling (full line 23), video programing service (line16).

1: (full blanking) All lines inside VBI are blanked
NTSC-M: lines [1:19], [263(half):282] (525-SMPTE),
PAL-M: lines [523:16], [260(half):279] (525-CCIR)
other PAL: lines [623(half):22], [311:335] (625-CCIR)

[6:5] **FLT**[1:0]: UV chroma filter bandwidth selection. Typical applications for 3 dB bandwidth are given for each FLT bit configuration.
00: f_{-3dB} = 1.1 MHz low definition NTSC filter
01: f_{-3dB} = 1.3 MHz low definition PAL filter
10: f_{-3dB} = 1.6 MHz high definition NTSC filter (ATSC compliant) and PAL M/N (ITU-R 624.4 compliant)[2]
11: f_{-3dB} = 1.9 MHz high definition PAL filter: Rec 624 - 4 for PAL BDG/I compliant

[4] **SYNCOK**: Sync signal availability (analog) for input synchronization loss with FREERUN inactive (FREERUN = 0).
0: No sync output signals[2]
1: Output syncs available on YS, CVBS and, when applicable, that is, the same behavior as FREERUN except that video outputs are blanked in the active portion of the line.

[3] **COKI**: color killer.
0: color on[2]
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 DEN_CFG5.BKDAC2.
Note: Depending on the different output configurations chosen by programming bits from DAC123_CONF and DAC456_CONF (DEN_CFG13) pedestal is automatically selected on it.

[2] **SETUP_MAIN**: pedestal enable.
0: blanking level and black level are identical on all lines (for example, Argentinian. PAL-N, Japan NTSC-M, PAL-BDGHI)[2]
1: black level is 7.5 IRE above blanking level on all lines outside VBI (for example, Paraguayan and Uruguayan PAL-N). In all cases, gain factor is adjusted to obtain the required chrominance levels.

[1:0] **CC**[2:1]
00: no closed-caption/extended data encoding[2]
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

1. BLKLI must be set to 0 when closed-captions are to be encoded on the following lines:
in the 525/60 system: before line 20(SMPTE) or before line 283(SMPTE),
in the 625/50 system: before line 23(CCIR) or before line 336(CCIR).
2. Default mode when pin NOT_RESET is active (low level).

DEN_CFG2 Configuration 2

7	6	5	4	3	2	1	0
NINTRL	ENRST	BURSTEN	Reserved	SELRST	RSTOSC_BUF	VALRST1	VALRST0

Address: *DENCBaseAddress* + 0x0008
Type: Read/write
Buffer: Immediate
Reset: 0x20
Description:

[7] **NINTRL**: non-interlaced mode select.
0: interlaced mode (625/50 or 525/60 system)[1]
1: non-interlaced mode(2x312/50 or 2x262/60 system)
Note: NINTRL update is internally taken into account on beginning of next frame. In SECAM mode, only interlaced mode is available.

[6] **ENRST**: cyclic update of DDFS phase.
0: no cyclic subcarrier phase reset[1]
1: cyclic subcarrier phase reset depending on VALRST (see below)

[5] **BURSTEN**: chrominance burst control
0: burst is turned off on CVBS, chrominance output is not affected
1: burst is enabled[1]

[4] **Reserved**

[3] **SELRST**: selects reset values for direct digital frequency synthesizer accumulator.
0: hardware reset values for subcarrier oscillator phase (see registers DEN_PDFSn for values)[1]
1: loaded reset values selected (see contents of registers DEN_PDFSn)

[2] **RSTOSC_BUF** [2]: software phase reset of DDFS (direct digital frequency synthesizer) buffer.
0: no reset[1]
1: when a 0 to 1 transition occurs, either the hardwired default phase value, or the value loaded in DEN_PDFSn (according to bit SELRST), is put to the phase buffer. This value is loaded into accumulator (phase of subcarrier) when bits PH_RST_MODE from DEN_CFG8 are programmed, or when the standard changes or soft reset occurs.

[1:0] **VALRST**[1:0] [3]: period of automatic reset of the oscillator.
00: every line[1] 01: every 2nd field
10: every 4th field 11: every 8th field

1. Default mode when pin NOT_RESET is active (low level).
2. RSTOSC_BUF is automatically set back to 0 after the buffer is loaded.
3. VALRST[1:0] is taken into account only if 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 DEN_PDFSn (depending on the setting of SELRST), to which a 0°, 90°, 180°, or 270° correction is applied according to the field and line on which the reset is performed. Note: If SECAM is performed the oscillator is reset every line.

DEN_CFG3 Configuration 3

7	6	5	4	3	2	1	0
MAIN_ENTRAP	TRAP_4.43	ENCGMS	Reserved	MAIN_DEL_EN	Reserved	Reserved	ENWSS

Address: *DENCBaseAddress* + 0x000C
Type: Read/write
Buffer: Immediate
Reset: 0x00
Description:

[7] **MAIN_ENTRAP**: enable trap filter. When SECAM is performed, the trap filter is always enabled (set to 1).
0: trap filter disabled[1] 1: trap filter enabled

[6] **TRAP_4.43**: enable trap filter. It is taken into account only for main if bit MAIN_ENTRAP is set.
0: select trap filter centered around 3.58 MHz[1] 1: select trap filter centered around 4.43 MHz

[5] **ENCGMS**: CGMS encoding enable
0: disabled[1] 1: enabled
When ENCGMS is set to 1, closed-captions and extended data services should not be programmed on lines 20 and 283 (525/60, SMPTE line number convention).

[4] **Reserved**

[3] **MAIN_DEL_EN**: enable of chroma to luma delay programming on the main 4:4:4 input.
0: disabled (default)
1: enabled (chroma to luma delay is programmed by DEL[3:0] bits from DEN_CDEL_LFC.

[2] **Reserved**, must be set to 0

[1] **Reserved**

[0] **ENWSS**: WSS encoding enable.
0: disabled[1] 1: enabled

1. Default mode when pin NOT_RESET is active (low level).

DEN_CFG4 Configuration 4

7	6	5	4	3	2	1	0
SYNCIN_AD1	SYNCIN_AD0	SYNCOUT_AD1	SYNCOUT_AD2	ALINE	HUE_EN	Reserved	

Address: *DENCBaseAddress* + 0x0010

Type: Read/write

Buffer: Immediate

Reset: 0x00

Description:

[7:6] **SYNCIN_AD**[1:0]: adjustment of incoming sync signals. Used to ensure correct interpretation of incoming video samples as Y, Cr or Cb when the encoder is slaved to incoming sync signals (including F/H flags stripped off ITU-R656/D1 data)

00: nominal[1] 01: +1 PIX_CLK

10: +2 PIX_CLK 11: +3 PIX_CLK

[5:4] **SYNCOUT_AD[1:0]:** adjustment of outgoing sync signals. Used to ensure correct interpretation of incoming video samples as Y, Cr or Cb when the encoder is master and supplies sync signals

00: nominal[2] 01: +1 PIX_CLK

10: +2 PIX_CLK 11: +3 PIX_CLK

[3] **ALINE**: video active line duration control.

0: full digital video line encoding (720 pixels - 1440 clock cycles)[1]

1: active line duration follows ITU-R/SMPTE analog standard requirements

[2] **HUE_EN:** Enables variance in phase of the subcarrier, as programmed in REGISTER_105, 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

[1:0] **Reserved**

1. Default mode when pin NOT_RESET is active (low level)
2. Default mode when pin NOT_RESET is active (low level)

DEN_CFG5 Configuration 5

7	6	5	4	3	2	1	0
SELRST_INC	BKDAC1	BKDAC2	BKDAC3	Reserved	Reserved	BKDAC6	DACINV

Address: *DENCBaseAddress* + 0x0014
Type: Read/write
Buffer: Immediate
Reset: 0x00
Description:

[7] **SELRST_INC**: choice of digital frequency synthesizer increment after soft reset or when PH_RST_MODE = 01 (see register DEN_CFG8)
0: hard wired value (depending on TV standard)[1] 1: soft (value from DEN_IDFSn registers)

[6:4] **BKDACn**: blanking of DACs (n = 1,2,3)
0: DACn in normal operation[1]
1: DACn input code forced to black level (if RGB, UV or C) or blanking level (if Y or CVBS) depending on CONF_OUT Fbits of DEN_CFG8.

[3:2] **Reserved**

[1] **BKDAC6**: blanking of DAC6
0: DAC6 in normal operation[1]
1: DAC6 input code forced to blanking level (CVBS)

[0] **DACINV**: inverts DAC codes to compensate for an inverting output stage in the application.
0: noninverted DAC inputs (outputs)[1] 1: inverted DAC inputs (outputs)

1. Default mode when pin NOT_RESET is active (low level)

DEN_CFG6 Configuration 6

7	6	5	4	3	2	1	0
SOFTRESET	JUMP	DEC_NINC	FREE_JUMP	Reserved		TTX_EN	MAXDYN

Address: *DENCBaseAddress* + 0x0018
Type: Read/write
Buffer: Immediate
Reset: 0x10
Description:

[7] **SOFTRESET**: software reset. This bit is automatically reset after internal reset generation. Software reset is active for 4 PIX_CLK periods. When soft reset is activated, all the device is reset as with hardware reset except for the first nine user registers (DEN_CFG0 to DEN_CFG8).
Registers DEN_IDFSn to DEN_PDFSn (increment and phase of oscillator), DEN_VPSn, DEN_CGMSn and DEN_CCCF1n/DEN_CCCF2n are never reset (hard/soft).

0: no reset[1] 1: software reset

[6:4] **JUMP**, **DEC_NINC**, **FREE_JUMP**: bit JUMP is automatically reset after use.
000: Normal mode (no line skip/insert capability)
ITU-R (CCIR): 313/312 or 263/262
non-interlaced: 312/312 or 262/262
0x1: 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 LREF and LTAR of DEN_LINEn (by default, LREF = 0 and LTAR = 1 which leads to normal mode above).
100: Automatic line insert mode.
The 2nd field of the frame following the writing of this value is increased. Line insertion is done after line 245 in 525/60 and after line 290 in 625/50. LREF and LTAR are ignored[2].
110: Automatic line skip mode. The 2nd field of the frame following the writing of this value is decreased. Line suppression is done after line 245 in 525/60 and after line 290 in 625/50. LREF and LTAR are ignored[2].
1x1: Not to be used.

[3:2] **Reserved**

[1] **TTX_EN**: teletext enable bit

0: disabled[1] 1: enabled

[0] **MAXDYN**[3]: max dynamic magnitude allowed on YCrCb inputs for encoding.

0: 0x10 to 0xEB for Y, 0x10 to 0xF0 for chrominance (Cr,Cb)[1]
1: 0x01 to 0xFE for Y, Cr and Cb

1. Default mode when pin NOT_RESET is active (low level)
2. Two lines are skipped (inserted) in 525/60 and four lines in 625/50 standards.
3. EAV and SAV words are replaced by blanking values before being fed to the luminance and chrominance processing.

DEN_CFG7 Configuration 7

7	6	5	4	3	2	1	0
SECAM	PHI_12_SECAM	INV_PHI_SECAM	Reserved	Reserved	Reserved	ENVPS	Reserved

Address: *DENCBaseAddress* + 0x001C
Type: Read/write
Buffer: Immediate
Reset: 0x00
Description:

[7] **SECAM**: SECAM standard selection.
0: standard selected by STD1 and STD0 bits of DEN_CFG0
1: SECAM standard

[6] **PHI_12_SECAM**: subcarrier phase sequence start point in SECAM mode (refer to Section 30.11: *Subcarrier insertion (SECAM)* on page 308).
0: line 1 1: line 23

[5] **INV_PHI_SECAM**: inversion of subcarrier phase.
0: 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

[4:3] **Reserved**

[2] **Reserved**

[1] **ENVPS**: VPS encoding enable.
0: disable
1: enable

[0] **Reserved**

DEN_CFG8 Configuration8

7	6	5	4	3	2	1	0
PH_RST_MODE1	PH_RST_MODE0	Reserved		BLK_ALL	Reserved		

Address: *DENCBaseAddress* + 0x0020
Type: Read/write
Buffer: Immediate
Reset: 0x20
Description:

[7:6] **PH_RST_MODE**[1:0]: subcarrier phase reset mode. In modes 01 and 10, this bit is automatically reset to 00 after oscillator reset.
00: disabled
01: enabled, phase is updated with value from phase buffer register (see DEN_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 DEN_CFG5).
10: Reserved.
11: Reserved.

[5] **Reserved**. Must be set to 1.

[4] **Reserved,** Must be set to 0.

[3] **BLK_ALL**: blanking of all video lines.
0: disabled
1: enabled (all inputs ignored - 0x80 instead of Cr and Cb and 0x10 instead of Y and Y4)

[2:0] **Reserved**

DEN_STA DENC status

7	6	5	4	3	2	1	0
HOK	ATFR	BUF2_FREE	BUF1_FREE	FIELDCT2	FIELDCT1	FIELDCT0	JUMP

Address: *DENCBaseAddress* + 0x0024
Type: Read only
Buffer: Immediate
Reset: Undefined
Description:

[7] **HOK**: Hamming decoding of frame sync flag embedded within ITU-R656/D1 compliant YCrCb streams. Signal quality detector is issued from Hamming decoding of EAV, SAV from YCrCb.
0: consecutive errors 1: a single or no error[1]

[6] **ATFR**: frame synchronization flag.
0: encoder not synchronized[1] 1: in slave mode: encoder synchronized

[5] **BUF2_FREE**: closed-caption registers access condition for field 2. 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 DEN_CCCF2n are never written more than once between two frame reference signals, then BUF2_FREE is always true (set). Otherwise, access to closed-caption field 2 registers (DEN_CCCF2n) might be temporarily forbidden by resetting BUF2_FREE until the next field 2 closed-caption line occurs.
Note: this bit is false (reset) when two 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)[1]

[4] **BUF1_FREE**: closed-caption registers access condition for field 1. Same as BUF2_FREE but concerns field 1. Reset value: 1 (access authorized)[1]

[3:1] **FIELDCT**[2:0]: digital field identification number.
000: indicates field 1 up to 111: indicates field 8
Note: 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 DEN_CFG6.

1. Default mode when pin NOT_RESET is active (low level).

DEN_IDFSn Increment 1 to 3 for digital frequency synthesizer

7	6	5	4	3	2	1	0
D23	D22	D21	D20	D19	D18	D17	D16
7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

Address: *DENCBaseAddress* + 0x0028 (DEN_IDFS1), 0x002C (DEN_IDFS2), 0x0030 (DEN_IDFS3)

Type: Read/write

Buffer: Immediate

Reset: Undefined

Description: These registers contain the 24-bit increment used by the DDFS if bit SELRST_INC (register DEN_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 use these registers instead of the hard wired values follow the procedure below.

- Load the registers with the required value.
- Set bit SELRST_INC to 1 (register DEN_CFG5).
- Perform a software reset (register DEN_CFG6).

Note: The values loaded in DEN_IDFSn are taken into account after a software reset, and only if bit SELRST_INC = 1.

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, independent of DEN_IDFSn. These hardwired values cannot be read out of the DENC.

	Value	Frequency synthesized	Reference clock
[23:0]	0x21 F07C for NTSC M	f = 3.5795452 MHz	27 MHz
	0x2A 098B for PAL BGHIN	f = 4.43361875 MHz	27 MHz
	0x21 F694 for PAL N	f = 3.5820558 MHz	27 MHz
	0x21 E6F0 for PAL M	f = 3.57561149 MHz	27 MHz
	0x25 5554 for NTSC M square pixel	f = 3.5795434 MHz	24.545454 MHz
	0x26 798C for PAL BGHIN square pixel	f = 4.43361867 MHz	29.5 MHz
	0x1F 15C0 for PAL N square pixel	f = 3.58205605 MHz	29.5 MHz
	0x25 4AD4 for PAL M square pixel	f = 3.57561082 MHz	24.545454 MHz

DEN_PDFSn Static phase offset 1 and 2 for digital frequency synthesizer

Address: *DENCBaseAddress* + 0x0034 (DEN_PDFS1), 0x0038 (DEN_PDFS2)

Type: Read/write

Buffer: Immediate

Reset: Undefined

Description: When SECAM (DEN_CFG8) = 0 (PAL and NTSC)

7	6	5	4	3	2	1	0
Reserved						O23	O22

7	6	5	4	3	2	1	0
O21	O20	O19	O18	O17	O16	O15	O14

In any of the following conditions when SELRST = 0 (DEN_CFG5), the phase accumulator is initialized with the ten MSBs of this register:

- after a 0-to-1 transition of bit RSTOSC (DEN_CFG2),
- after a standard change,
- when cyclic phase readjustment has been programmed (DEN_CFG2.VALRST[1:0]).

The 14 remaining LSBs loaded into the accumulator in these cases are all zeroes (defining the phase reset value with a 0.35° accuracy).
If SELRST = 0 (for example after a hardware reset) the phase offset used every time the DDFS is re initialized is a hardwired value. The hardwired 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.

To use these registers instead of the hardwired values follow the procedure below.

- Load the registers with the required value.
- Set SELRST to 1 (DEN_CFG2).
- Perform a software reset (DEN_CFG6),
  or set RSTOSC_BUF to 1 (DEN_CFG6, this puts the soft phase value into a tampon register) and set PH_RST_MODE[1:0] (DEN_CFG8) to put this value into an accumulator at the beginning of the next line.

These registers are never reset and must be explicitly written to.

Description: When SECAM (DEN_CFG8) = 1 (two SECAM subcarriers)

7	6	5	4	3	2	1	0
B21	B20	B19	B18	B17	B16	B15	B14
7	6	5	4	3	2	1	0
R21	R20	R19	R18	R17	R16	R15	R14

When bit SELRST = 1, this register configures the static phase offset for digital frequency synthesizer for two SECAM subcarriers (8 MSB only) (blue lines - DEN_PDFS1 and red lines - DEN_PDFS2).
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:

- DEN_PDFS1 x 16384 (decimal), or DEN_PDFS1 x 0x4000 (blue lines)
- (256 + DEN_PDFS1 + DEN_PDFS2) x 16384 (decimal), or
  (100 + DEN_PDFS1 + DEN_PDFS2) x 0x4000 (red lines)

If bit SELRST = 0 (for example after a hardware reset) the phase offset used every time the DDFS is re initialized is a hard wired value. 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.

To validate use of these registers instead of the hard wired values, follow the PAL and NTSC mode.

The registers are never reset and must be explicitly written to.

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* + 0x0003C (DEN_WSS1), 0x0040 (DEN_WSS2)

Type: Read/write

Buffer: Double buffered

Reset: Undefined

Description: Refer to Section 30.18: *WSS encoding* on page 315

[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.

DEN_DAC13 DAC1 and DAC3 multiplying factors

7	6	5	4	3	2	1	0
DAC1_MULT[5:0]						DAC3_MULT[5:0]	

Address: *DENCBaseAddress* + 0x0044
Type: Read/write
Buffer: Immediate
Reset: 0x82
Description:

[7:2] **DAC1_MULT[5:0]**: multiplying factor on DAC1_C digital signal before the DACs with 0.78% step.

If R, G or B (DAC123_CONF = 010)	If not R, G or B (DAC123_CONF ≠ 010)
000000: 81.25%	000000: 75.000%
000001: 81.84%,	000001: 75.78%
000010: 82.42%,	000010: 76.56%
000011: 83.01%,	000011: 77.34%
.. ..	
100000: 100%	100000: 100%[1]
.. ..	
111111: 118.16%,	111111: 124.22%

[1:0] **DAC3_MULT**[5:0]: multiplying factor on DAC3_CVBS digital signal before the DACs with 0.78% step + DEN_DAC34[7:4]

If R, G or B (DAC123_CONF = 010)	If not R, G or B (DAC123_CONF ≠ 010)
000000: 81.25%	000000: 75.000%
000001: 81.84%,	000001: 75.78%
000010: 82.42%,	000010: 76.56%
000011: 83.01%,	000011: 77.34%
.. ..	
100000: 100%	100000: 100%[1]
.. ..	
111111: 118.16%,	111111: 124.22%

1. Default value

DEN_DAC34 DAC3 and DAC4 multiplying factors

7	6	5	4	3	2	1	0
DAC3_MULT[3:0]				Reserved			

Address: *DENCBaseAddress* + 0x0048
Type: Read/write
Buffer: Immediate
Reset: 0x00
Description:

[7:4] See DEN_DAC13 description.

[3:0] **Reserved**

DEN_LINEn LTARG[8:0] and LREF[8:0] (n = 0, 1 or 2)

7	6	5	4	3	2	1	0
LTARG8	LTARG7	LTARG6	LTARG5	LTARG4	LTARG3	LTARG2	LTARG1
LTARG0	LREF8	LREF7	LREF6	LREF5	LREF4	LREF3	LREF2
LREF1	LREF0	Reserved					

Address: *DENCBaseAddress* + 0x0054 (DEN_LINE0), 0x0058 (DEN_LINE1), 0x005C (DEN_LINE2)

Type: Read/write

Buffer: Immediate

Reset: LREF[8:0]: 0 0000 0000; LTARG[8:0]: 0 0000 0001

Description: These registers may 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 with no problems. If this function is required, it should be used with caution.

0x054: [7:0]
0x058: [7] **LTARG**: contains the target line binary number.

0x058 [6:0]
0x05C[7:6] **LREF**: contains in binary format the reference line from which a jump is required.

DEN_VPSn VPS data 1 to 6

7	6	5	4	3	2	1	0
S1	S0	Reserved		CNI3	CNI2	CNI1	CNI0
NP7	NP6	D4	D3	D2	D1	D0	M3
M2	M1	M0	H4	H3	H2	H1	H0
MIN5	MIN4	MIN3	MIN2	MIN1	MIN0	C3	C2
C1	C0	NP5	NP4	NP3	NP2	NP1	NP0
PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0

Address: *DENCBaseAddress* + 0x0064 (DEN_VPS1), 0x0068 (DEN_VPS2), 0x006C (DEN_VPS3), 0x0070 (DEN_VPS4), 0x0074 (DEN_VPS5), 0x0078 (DEN_VPS6)

Type: Read/write

Buffer: Immediate

Reset: Undefined

Description: See Section 30.19: *VPS encoding* on page 316.

S[1:0]: sounds.
00: don t know 01: mono
10: stereo 11: dual sound

CNI[3:0]: 4 bits of CNI reserved for enhancement of VPS.

NP[7:6]: network or program CNI.

D[4:0]: day, binary.

M[3:0]: month, binary.

H[4:0]: hour, binary.

MIN[5:0]: minute, binary.

C[3:0]: country, binary.

NP[5:0]: network or program CNI (country and network identification).

PT[7:0]: program type, binary.

DEN_CGMSn CGMS data 1 to 3 (20 bits only)

7	6	5	4	3	2	1	0
Reserved				B1	B2	B3	B4
B5	B6	B7	B8	B9	B10	B11	B12
B13	B14	B15	B16	B17	B18	B19	B20

Address: *DENCBaseAddress* + 0x007C (DEN_CGMS1), 0x0080 (DEN_CGMS2), 0x0084 (DEN_CGMS3)

Type: Read/write

Buffer: Immediate

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)

DEN_TTXn Teletext block definition 1 to 5

7	6	5	4	3	2	1	0
FP_TTXT	TTXDEL2	TTXDEL1	TTXDEL0	TTX_L6	TTX_L7	TTX_L8	TTX_L9
TTX_L10	TTX_L11	TTX_L12	TTX_L13	TTX_L14	TTX_L15	TTX_L16	TTX_L17
TTX_L18	TTX_L19	TTX_L20	TTX_L21	TTX_L22	TTX_L23	TTX_L318	TTX_L319
TTX_L320	TTX_L321	TTX_L322	TTX_L323	TTX_L324	TTX_L325	TTX_L326	TTX_L327
TTX_L328	TTX_L329	TTX_L330	TTX_L331	TTX_L332	TTX_L333	TTX_L334	TTX_L335

Address: *DENCBaseAddress* + 0x0088 (DEN_TTX1), 0x008C (DEN_TTX2), 0x0090 (DEN_TTX3), 0x094 (DEN_TTX4), 0x098 (DEN_TTX5)

Type: Read/write

Buffer: Immediate

Reset: Undefined

Description: See Section 30.20: *Teletext encoding* on page 316.

FP_TTXT: full page teletext mode enable.

0: disabled[1] 1: enabled

TTXDEL[2:0]: teletext data latency. The encoder clocks 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 synchronization signal, supplied by the encoder). Default value: 100

TTX_L*n*: Each of these bits enables teletext on line X (ITU-R line numbering and 625 line systems). For teletext line selections for other standards, refer to Table 87 below. Default value: 0

1. Default mode when pin NOT_RESET is active (low level).

Table 87: Teletext line selections

	PAL BDGHIN line (CCIR 625 numbering)	PAL M line (CCIR 525 numbering)	NTSC M line (SMPTE 525 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
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

DEN_CCCF1n Closed-caption characters/extended data for field 1

7	6	5	4	3	2	1	0
OPC11	C117	C116	C115	C114	C113	C112	C111
OPC12	C127	C126	C125	C124	C123	C122	C121

Address: *DENCBaseAddress* + 0x009C (DEN_CCCF11), 0x00A0 (DEN_CCCF12)

Type: Read/write

Buffer: Immediate

Reset: Undefined

Description:

OPC11: odd parity bit of US-ASCII 7-bit character C11[7:1]

C11[7:1]: first byte to encode in field1

OPC12: odd parity bit of US-ASCII 7-bit character C12[7:1]

C12[7:1]: second byte to encode in field1

Note: Default value: none, but enabling closed-captions without loading these registers issues a null character. Registers DEN_CCCF1n *are never reset.*

DEN_CCCF2n Closed-caption characters/extended data for field 2

7	6	5	4	3	2	1	0
OPC21	C217	C216	C215	C214	C213	C212	C211
OPC22	C227	C226	C225	C224	C223	C222	C221

Address: *DENCBaseAddress* + 0x00A4 (DEN_CCCF21), 0x00A8 (DEN_CCCF22)

Type: Read/write

Buffer: Double buffered

Reset: Undefined

Description:

OPC21: Odd parity bit of US-ASCII 7-bit character C21[7:1]

C21[7:1]: First byte to encode in field 2

OPC22: Odd parity bit of US-ASCII 7-bit character C22[7:1]

C22[7:1]: Second byte to encode in field 2

Note: Default value: none, but closed-captions enabling without loading these registers issue a null character. Registers DEN_CCCF2n *are never reset.*

DEN_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* + 0x00AC

Type: Read/write

Buffer: Double buffered

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).

[7:5] **Reserved**

[4:0] **L1**[4:0]
00000: no line selected for closed-caption encoding
000xx: do not use these codes
....
i code line: (i + 6) (SMPTE) selected for encoding
....
11111: 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.

[7:5] **Reserved**

[4:0] **L1**[4:0]
00000: no line selected for closed-caption encoding
....
i code line: (i + 6) (CCIR) selected for encoding (i > 0)
....
11111: line 37 (CCIR) selected

Note: Default value = 0 1111 line 21 (525/60, 525-SMPTE line number convention), which corresponds to line 21 in I625/50 system,(625-CCIR line number convention).

DEN_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* + 0x00B0

Type: Read/write

Buffer: Double buffered

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), although it is possible to program over a wider range.

[7:5] **Reserved**

[4:0] **L2**[4:0]
00000: no line selected for closed-caption encoding
000xx: do not use these codes
i line: (269 + i) (SMPTE) selected for encoding
....
01111: line 284 (SMPTE) selected for encoding
11111: 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), although it is possible to program over a wider range.

[7:5] **Reserved**

[4:0] **L2**[4:0]
00000: no line selected for closed-caption encoding
i line: (318 + i) (CCIR) selected for encoding
....
10010: line 336 (CCIR) selected for encoding
11111: line 349 (CCIR)

Note: Default value= 01111 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).

DEN_TTXCFG Teletext configuration

7	6	5	4	3	2	1	0
TTXT100IRE	TTXT_ABCD[1:0]		Reserved				

Address: *DENCBaseAddress* + 0x0100
Type: Read/write
Buffer: Double buffered
Reset: 0x50
Description:

[7] **TTXT100IRE**: teletext waveform amplitude.
0: 70 IRE[1] 1: 100 IRE

[6:5] **TTXT_ABCD**[1:0]: teletext standard selection.
00: teletext A
01: teletext B, teletext B in 625 line systems is known as world system teletext
10: teletext C, teletext C in 525 line systems is known as NABTS
11: teletext D

[4:0] **Reserved**. must be set to 10000.

1. Default mode when pin NOT_RESET is active (low level).

DEN_DACC DACC multiplier and TTXS

7	6	5	4	3	2	1	0
DACC_MULT[3:0]				TTX_MASK_OFF	Reserved	Reserved	BCS_EN_MAIN

Address: *DENCBaseAddress* + 0x0104
Type: Read/write
Buffer: Immediate
Reset: 0x01
Description: 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 IREF 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 IREF 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.

[7:4] **DACC_MULT**[3:0]: multiplying factor on C digital output (signal before the DACs and on color part of CVBS signal).
0000: 1.000000 (1.000000 decimal)
0001: 1.000001 (1.015625 decimal)
0010: 1.000010 (1.03120 decimal)
0011: 1.000011 (1.046875 decimal)
...
1111: 1.001111 (1.234375 decimal)

[3] **TTX_MASK_OFF**: masking of 2 bits in teletext framing code, depending on selected teletext standard.
0: enable 1: disable

[2:1] **Reserved**

[0] **BCS_EN_MAIN**: brightness, contrast and saturation control by registers DEN_BRIGHT to DEN_SATUR on 4:4:4 main video input.
0: disable 1: enable

DEN_BRIGHT Brightness

7	6	5	4	3	2	1	0
B[7:0]							

Address: *DENCBaseAddress* + 0x0114

Type: Read/write

Buffer: Immediate

Reset: 0x80

Description: The following formula is used to adjust the luminance intensity of the display video image:

$$Y_{OUT} = Y_{IN} + B - 128$$

Where Y_{IN} is 8-bit input luminance and Y_{OUT} is the result of brightness operation (still on 8 bits). This value is saturated at 235 (16) or 254 (1) according to DEN_CFG6 bit MAXDYN, B: brightness (unsigned value with center at 128, default 128).

DEN_CONTR Contrast

7	6	5	4	3	2	1	0
C[7:0]							

Address: *DENCBaseAddress* + 0x0118

Type: Read/write

Buffer: Immediate

Reset: 0x00

Description: The following formula is used to adjust the relative difference between the display image higher and lower intensity luminance values:

$$Y_{OUT} = \frac{(Y_{IN} - 128)(C + 128)}{128} + 128$$

Where, Y_{IN} is 8-bit input luminance, Y_{OUT} is the result of contrast operation (still on 8 bits). This value is saturated at 235 (16) or 254 (1) according to DEN_CFG6 bit MAXDYN, C: contrast (two's complement value from -128 to 127, default 0).

DEN_SATUR Saturation

7	6	5	4	3	2	1	0
S[7:0]							

Address: *DENCBaseAddress* + 0x011C
Type: Read/write
Buffer: Immediate
Reset: 0x80
Description: The following formula is used to adjust the color intensity of the displayed video image:

$$Cbout = \frac{S(Cbin - 128)}{128} + 128$$

$$Crout = \frac{S(Crin - 128)}{128} + 128$$

Where Crin and Cbin are the 8-bit input chroma, Crout and Cbout are the result of saturation operation (still on 8 bits).

This value is saturated at 240 (16) or 254 (1) according to DEN_CFG6 bit MAXDYN, S: saturation value (unsigned value with centre at 128, default 128).

DEN_CFCOEFn Chroma filter coefficient 0 to 8 (main video)

7	6	5	4	3	2	1	0
MAIN_FLT_S	MAIN_PLG_DIV[1:0]		CH_COEF0[4:0]				
Reserved	CH_COEF8[8]	CH_COEF1[5:0]					
Reserved	CH_COEF2[6:0]						
Reserved	CH_COEF3[6:0]						
CH_COEF4[7:0]							
CH_COEF5[7:0]							
CH_COEF6[7:0]							
CH_COEF7[7:0]							
CH_COEF8[7:0]							

Address: *DENCBaseAddress* + 0x0120 (DEN_CFCOEF0), 0x0124, 0x0128, 0x012C, 0x0130, 0x0134, 0x0138, 0x013C, 0x0140 (DEN_CFCOEF8)

Type: Read/write

Buffer: Immediate

Reset: See table below

Description: These registers contain the nine coefficients and three control bits for the chroma filter, which are described in the table below. Values from these registers are used only when MAIN_FLT_S is set to 1. This bit has highest priority over the rest. 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 (DEN_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 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. The offset values for each coefficient are listed in the table below.

Filter sampling frequency is PIX_CLK (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 DEN_CFCOEFn values, default (reset) or programmed, to determine coefficients.

Coefficient	Offset	Reset value	Reset setting
CH_COEF0[4:0]: coefficient number 0	16	1 0001	C0 = 1
CH_COEF1[5:0]: coefficient number 1	32	10 0111	C1 = 7
CH_COEF2[6:0]: coefficient number 2	64	101 0100	C2 = 20
CH_COEF3[6:0]: coefficient number 3	32	100 0111	C3 = 39
CH_COEF4[7:0]: coefficient number 4	32	0101 1111	C4 = 63
CH_COEF5[7:0]: coefficient number 5	32	0111 0111	C5 = 87
CH_COEF6[7:0]: coefficient number 6	0	0110 1100	C6 = 108
CH_COEF7[7:0]: coefficient number 7	0	0111 1011	C7 = 123
CH_COEF8[8:0]: coefficient number 8	0	0 1000 0000	C8 = 128

DEN_CDEL_LFC Chroma delay and luma filter control

7	6	5	4	3	2	1	0
MAIN_DEL[3:0]				Reserved	PLG_DIV_Y[1:0]		FLT_YS

Address: *DENCBaseAddress* + 0x0144

Type: Read/write

Buffer: Immediate

Reset: 0x22

Description: This register contains the chroma path delay (referenced to luma on S-VHS and CVBS outputs), selection of input format mode, and three bits to control the luma filter.

[7:4] **MAIN_DEL**[3:0] [1]: delay on chroma path with reference to luma path encoding (1 pixel = 2 periods of frequency f_{PIX_CLK}). Delays are made programmable by setting MAIN_DEL_EN = 1 (DEN_CFG3).

0010: -0.5 pixel delay
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
Others
xxx1: 0 pixel delay
xxx0: +0.5 pixel delay
If MAIN_DEL_EN = 0, the delays used are:
0010 when mode is PAL/NTSC in 4:2:2 format on CVBS.
0001 when mode is PAL/NTSC in 4:4:4 or SECAM 4:2:2 format on CVBS.
0000 when mode is SECAM in 4:4:4 format on CVBS.

[3] **Reserved**

[2:1] **PLG_DIV_Y**[1:0]

00: when sum of coefficients = 256
01: when sum of coefficients = 512 (default)
10: when sum of coefficients = 1024
11: when sum of coefficients = 2048

[0] **FLT_YS**: enables the software loading capability of luma coefficients.
0: use hard wired coefficients for the luma filter
1: use register DEN_LFCOEFn values, default (reset) or programmed, to determine coefficients

1. Bit DEL_EN in register DEN_CFG3 selects either default or programmed delays.

DEN_LFCOEFn Luma filter coefficient 0 to 9

7	6	5	4	3	2	1	0
Reserved			LU_COEF8[8]	LU_COEF0[4:0]			
LU_COEF9[9:8]			LU_COEF1[5:0]				
LU_COEF6[8]		LU_COEF2[6:0]					
LU_COEF7[8]		LU_COEF3[6:0]					
LU_COEF4[7:0]							
LU_COEF5[7:0]							
LU_COEF6[7:0]							
LU_COEF7[7:0]							
LU_COEF8[7:0]							
LU_COEF9[7:0]							

Address: *DENCBaseAddress* + 0x0148 (DEN_LFCOEF0), 0x014C, 0x0150, 0x0154, 0x0158, 0x015C, 0x0160, 0x0164, 0x0168, 0x016C (DEN_LFCOEF9)

Type: Read/write

Buffer: Immediate

Reset: See table below

Description: These registers contain the ten coefficients for the luma filter, which are described in the table below. Values from this register is used only when FLT_YS (DEN_CDEL_LFC) is set to 0. The coefficients give the required filter response for a specific application according to the symmetrical RIF filter equation:
$H(z) = A_0 + A_1z^{-1} + A_2z^{-2} + ... + A_8z^{-8} + A_9z^{-9} + A_8z^{-10} + ... + A_2z^{-16} + A_1z^{-17} + A_0z^{-18}$
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.
Sampling frequency of the filter is PIX_CLK (27 MHz, 24.545454 MHz or 29.5 MHz).
The control bits for this filter are in register DEN_CDEL_LFC.

Coefficient	Reset value	Reset setting
LU_COEF0[4:0]: coefficient number 0	0 0001	A0 = +1
LU_COEF1[5:0]: coefficient number 1	11 1111	A1 = -1
LU_COEF2[6:0]: coefficient number 2	111 0111	A2 = -9
LU_COEF3[6:0]: coefficient number 3	000 0011	A3 = +3
LU_COEF4[7:0]: coefficient number 4	0001 1111	A4 = +31
LU_COEF5[7:0]: coefficient number 5	1111 1011	A5 = -5
LU_COEF6[8:0]: coefficient number 6	1 1010 1100	A6 = -84
LU_COEF7[8:0]: coefficient number 7	0 0000 0111	A7 = +7
LU_COEF8[8:0]: coefficient number 8	1 0011 1101	A8 = +317
LU_COEF9[9:0]: coefficient number 9	01 1111 1000	A9 = +504

Working frequency of this filter is the one of PIX_CLK (27, 24.545454 or 29.5 MHz), and the filtering is done on upsampled signal (half PIX_CLK to PIX_CLK frequency by padding by zeros).

DEN_CFG10 Configuration 10

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	RGBSAT_EN	SECAM_IN_MUX	Reserved	

Address: *DencBaseAddress* + 0x0170
Type: Read/write
Buffer: Immediate
Reset: 0x08
Description:

[7:4] **Reserved**

[3] **RGBSAT_EN**: RGB format output not saturated to real colors (241 max and 15 min), but allowed to have more saturated colors (255 max and 0 min).
0: RGB outputs allowed to have more saturated colors.
1: RGB outputs saturated to real colors.

[2] **SECAM_IN_MUX**: SECAM input video select. (refer to section Section 30.12: *Luminance encoding* on page 308). This is bit is enabled if SECAM bit in DEN_CFG7 = 1.
0: main video input is SECAM encoded. 1: auxiliary video input is SECAM encoded.
Note: No Auxiliary video input support

[1:0] **Reserved**

DEN_CFG11 Configuration 11

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	MAIN_IF_DEL	Reserved	

Address: *DencBaseAddress* + 0x0174
Type: Read/write
Buffer: Immediate
Reset: 0x28
Description:

[7:3] **Reserved**

[2] **MAIN_IF_DEL**: 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). See Section 30.14: *Composite video signal generation* on page 312.
0: enabled 1: disabled

[1:0] **Reserved**

DEN_CFG13 Configuration 13

7	6	5	4	3	2	1	0
AUX_NOTMAIN RGB	RGB_MAXDYN	DAC123_CONF2	DAC123_CONF1	DAC123_CONF0	Reserved	Reserved	Reserved

Address: *DencBaseAddress* + 0x017C

Type: Read/write

Buffer: Immediate

Reset: 0x82

Description:

[7] **AUX_NOTMAINRGB**: it is possible to have either main video or auxiliary video as RGB.
0: main video is output as RGB 1: auxiliary video is output as RGB[1]

[6] **RGB_MAXDYN**: it is possible to have different gains when RGB is being output.
0: Black to peak R/G/B swing is 560 decimal values
1: Black to peak R/G/B swing is 821 decimal values
Setting to 1 better uses the maximum dynamic range of the video DACs. Blanking level is 64.

[5:3] **DAC123_CONF**[2:0]: below are the possible combinations to be observed at DAC123.
000: Y_MAIN (DAC2), C_MAIN (DAC1), CVBS_MAIN (DAC3)[1]
001: Reserved
010: R (DAC1), G (DAC2), B (DAC3)
011: PR_MAIN (DAC1), Y_MAIN (DAC2), PB_MAIN (DAC3)
100: Reserved
101: Reserved
110: Reserved
111: Reserved

[2:0] **Reserved,** always set to 001.
CVBS_MAIN(DAC6)

1. Default mode when NOT_RESET is active (low level).
Note: Set this bit to '0' for RGB outputs.

DEN_HUE_CONTROL Hue control

7	6	5	4	3	2	1	0
PHASE	HUE_CONTROL[6:0]						

Address: *DencBaseAddress* + 0x01A4

Type: R/W

Buffer: Immediate

Reset: 0x00

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 the DEN_PDFS1 HUE_EN bit, phase variation would be in a range of +/- 22.324 degrees with increments of 0.17578127.

Note: To make a value programmed in this register effective, immediately after programming HUE_CONTROL, pulse the DEN_CFG4 HUE_EN bit.

Once enabled, to disable any phase shift for burst in the active subcarrier, write the default value into the DEN_HUE_CONTROL register followed by a pulse on DEN_CFG4 HUE_EN bit.

[7] **PHASE**: 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_CONTROL[6:0]**: absolute value of phase adjustment, range 1 to 127.
LSB 0x01 implies 0.17578127 degrees, 0x7F imply 22.324 degrees

DEN_DAC2 DAC2 multiplying factors

7	6	5	4	3	2	1	0
Reserved		DAC2_MULT[5:0]					

Address: *DENCBaseAddress* + 0x01A8

Type: Read/write

Buffer: Immediate

Reset: 0x20

Description:

[7:6] **Reserved**

[5:0] **DAC2_MULT[5:0]**: multiplying factor on DAC2_C digital signal before the DACs with 0.78% step.

f R, G or B (DAC123_CONF = 010)	If not R, G or B (DAC123_CONF ≠ 010)
000000: 81.25%	000000: 75.000%
000001: 81.84%,	000001: 75.78%
000010: 82.42%,	000010: 76.56%
000011: 83.01%,	000011: 77.34%
. .. .	
100000: 100%	100000: 100%[1]
.. ..	
111111: 118.16%,	111111: 124.22%

1. Default value.

DEN_DAC6 DAC6 multiplying factors

7	6	5	4	3	2	1	0
Reserved		DAC6_MULT[5:0]					

Address: *DENCBaseAddress* + 0x01AC
Type: Read/write
Buffer: Immediate
Reset: 0x20
Description:

[7:6] **Reserved**

[5:0] **DAC6_MULT[3:0]**: multiplying factor on DAC6_CVBS digital signal before the DACs with 0.78% step.

000000: 75.000%
000001: 75.78%
000010: 76.56%
000011: 77.34%
.. ..
100000: 100%[1]
.. ..
111111: 124.22%

1. Default value.

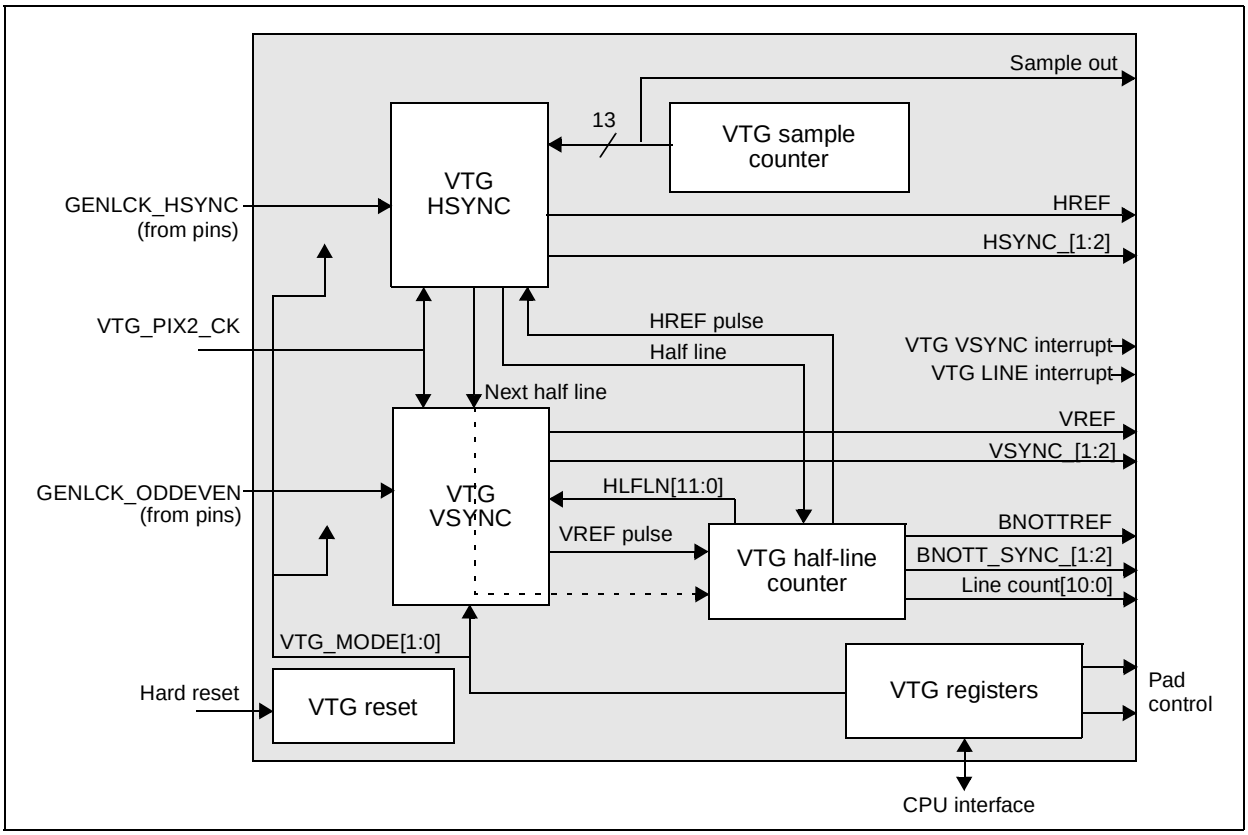
32 Video timing generator (VTG)

The video timing generator (VTG) provides video synchronization signals internally to sub-blocks of the TV out module (GDMA and digital encoder) as well as externally to the blitter and host CPU.

The video timing generator in STx5119 is master. As master It automatically generates synchronization signals and associated counter values. The VTG delivers HSYNC (horizontal synchronization signal) and VSYNC (vertical synchronization signal) starting from hard or soft reset and calibrated by register information such as the number of samples per line and the number of half lines in a field).

Note: The VTG works in interlaced mode only.

Figure 145: Block diagram



32.1 Signal generation

Horizontal and vertical synchronization signals are either fixed or programmable.

32.1.1 Fixed signals

There are three fixed signals, HREF, VREF and BNOTTREF.

- HREF is high during 64 samples and the rising edge occurs when the pixel counter takes the value 0.
- VREF is high during 64 samples and the rising edge occurs when the line counter takes the value 1 in progressive mode or on the top field, in interlaced mode on the bottom field. VREF occurs when the line count takes the value 0.
- BNOTTREF toggles each rising edge of VREF.

32.1.2 Programmable signals

The rising and falling edges of these signals are fully programmable and refer to line count and sample count (see registers VTG_H_HD_n, VTG_TOP_V_VD_n, VTG_BOT_V_VD_n, VTG_TOP_V_HD_n and VTG_TOP_V_HD_n). Vertical delay must match the line counter (for interlaced mode, the first line of the top field is line 1 but the first line of the bottom field is line 0). Horizontal delay must match the sample counter (for interlaced mode on bottom field line 0, VTG_BOT_V_VD_n delay must be CLKLN / 2 to take into account the field changes at the middle of the horizontal excursion).

The sample granularity for horizontal delays allows the edge of the sync to be phased on the rising or falling edge of the pixel clock.

32.1.3 Other signals

The BNOTT_SYNC_n signal toggles each rising edge of VSYNC.

The video timing generator generates two interrupts, one on every VSYNC and the other on a programmed line number.

32.2 Master mode

See Figure 146: *Master mode waveforms* on page 354.

The parity of the half line number (VTG_HLFLN register) determines the position of VREF regarding the HREF position. After a reset, VREF and HREF are activated at the same time to generate a top field (VTG_BNOTT_REF = 0). The half line counter is cleared and the line counter is set to one (VREF and HREF are detected in phase). The vertical count is incremented each time the horizontal count reaches the end of line value (VTG_CLKLN register). HREF is then produced and a pulse activated, every half line increments a half line counter. When this counter is equal to HLFLN - 1, VREF is activated. If HALF_LINE_NUMBER is odd a VREF occurs but HREF is not present. The vertical and half line counters are cleared and a bottom field is generated (VTG_BNOTT_REF = 1).

Next time the half line counter reaches HALF_LINE_NUMBER - 1, both VREF and HREF are generated and a top field is generated. If VREF is activated in phase with or without HREF, the video timing generator works in interlaced mode.

If HALF_LINE_NUMBER is even, VREF is always generated in phase with HREF and the video timing generator works in progressive mode. Each time a VREF occurs the signal BNOTTREF toggles.

From VREF, HREF and BNOTTREF two sets of synchronization signals are generated:

- **set 1**: HSYNC_1, VSYNC_1, BNOTT_SYNC_1,
- **set 2**: HSYNC_2, VSYNC_2, BNOTT_SYNC_2.

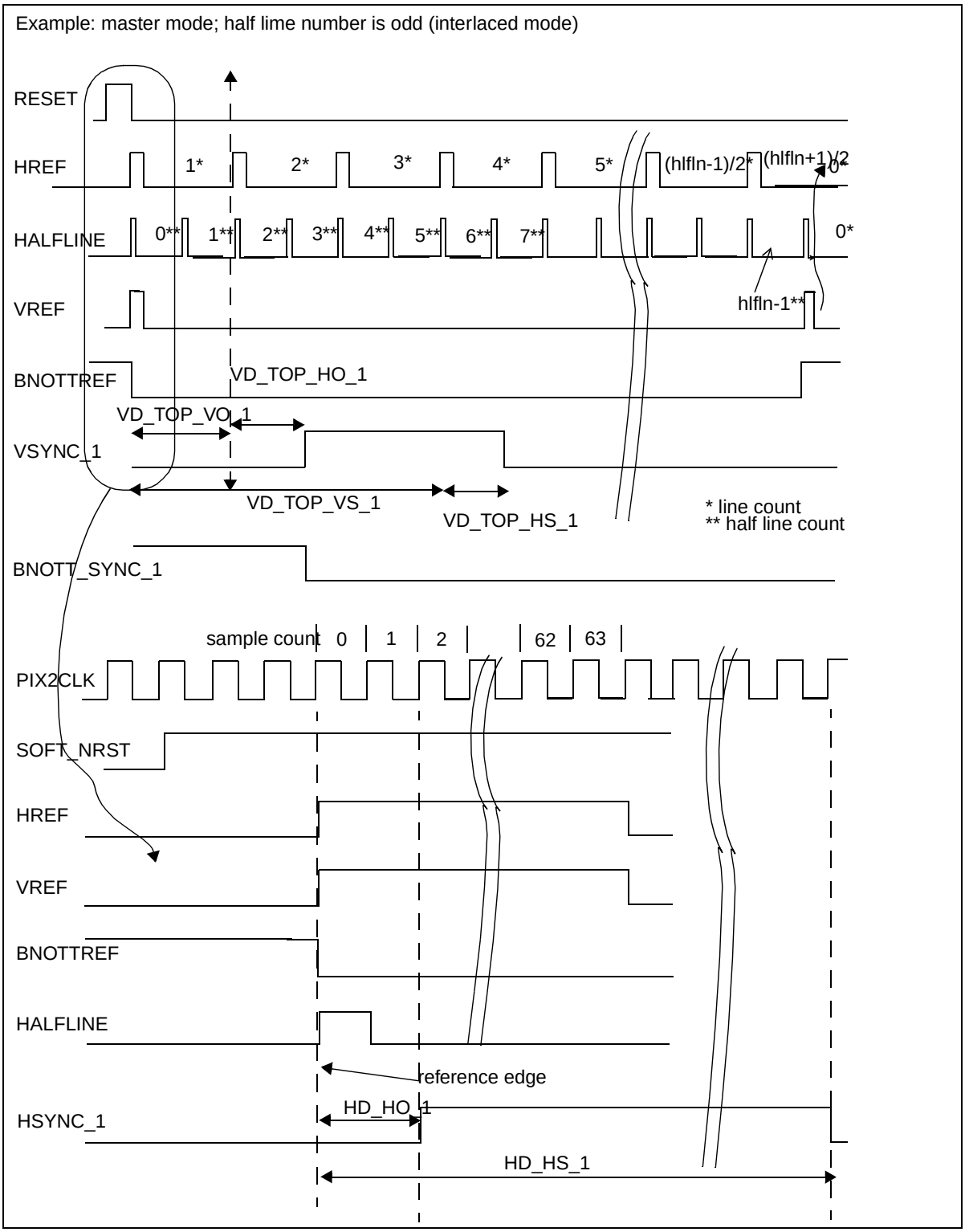
Programmable register values allow the adjustment of the rising and falling edges of the HSYNC_n and VSYNC_n signals. The BNOTT_SYNC_n signals toggle each rising edge of the associated VSYNC_n signal.

In the waveform in Figure 146 only set 1 is drawn. The values of VD_TOP_VO_1, VD_TOP_HO_1, VD_TOP_VS_1 are duplicated for the bottom field. HD_HO_1 and HD_HS_1 are available for the top and bottom field.

These values are duplicated for set 2 with the suffix _2.

Note: A VTG_DISABLE bit is implemented in the VTG_MOD register. This signal disables line and sample counters. After reset this signal is activated and the video timing generator does not run before the bit is reset.

Figure 146: Master mode waveforms



33 Video timing generator (VTG) registers

Addresses are provided as the *VTGBaseAddress* + offset.

The *VTGBaseAddress* is:

0x2090 0400.

Table 88: Video timing generator (VTG) registers

Register name	Description	Address offset	Type
VTG_VID_TFO	Video top field offset, see page 358	0x04	R/W
VTG_VID_TFS	Video top field stop, see page 358	0x08	R/W
VTG_VID_BFO	Video bottom field offset, see page 358	0x0C	R/W
VTG_VID_BFS	Video bottom field stop, see page 359	0x10	R/W
VTG_MOD	Mode of operation, see page 356	0x24	R/W
VTG_CLKLN	Clocks per line, see page 357	0x28	R/W
VTG_H_HD_1	HSYNC_1 sample number falling and rising edge, see page 359	0x2C	R/W
VTG_TOP_V_VD_1	VSYNC_1 vertical TOP field line number falling and rising edge, see page 360	0x30	R/W
VTG_BOT_V_VD_1	VSYNC_1 vertical BOTTOM field line number output falling and rising edge, see page 360	0x34	R/W
VTG_TOP_V_HD_1	VSYNC_1 vertical TOP field sample number rising and falling edge, see page 361	0x38	R/W
VTG_BOT_V_HD_1	VSYNC_1 vertical BOTTOM field sample number rising and falling edge, see page 361	0x3C	R/W
VTG_HLFLN	Half lines per vertical, see page 362	0x40	R/W
VTG_DRST	Video timing generator reset, see page 362	0x48	WO
VTG_H_HD_2	HSYNC_2 sample number falling and rising edge, see page 359	0x5C	R/W
VTG_TOP_V_VD_2	VSYNC_2 vertical TOP field line number falling and rising edge, see page 360	0x60	R/W
VTG_BOT_V_VD_2	VSYNC_2 vertical BOTTOM field line number output falling and rising edge, see page 360	0x64	R/W
VTG_TOP_V_HD_2	VSYNC_2 vertical TOP field sample number rising and falling edge, see page 361	0x68	R/W
VTG_BOT_V_HD_2	VSYNC_2 vertical BOTTOM field sample number rising and falling edge, see page 361	0x6C	R/W
VTG_H_HD_3	HSYNC_3 sample number falling and rising edge, see page 359	0x7C	R/W
VTG_TOP_V_VD_3	VSYNC_3 vertical TOP field line number falling and rising edge, see page 360	0x80	R/W
VTG_BOT_V_VD_3	VSYNC_3 vertical BOTTOM field line number output falling and rising edge, see page 360	0x84	R/W
VTG_TOP_V_HD_3	VSYNC_3 vertical TOP field sample number rising and falling edge, see page 361	0x88	R/W
VTG_BOT_V_HD_3	VSYNC_3 vertical BOTTOM field sample number rising and falling edge, see page 361	0x8C	R/W
VTG_ITM	Interrupt mask, see page 362	0x98	R/W
VTG_ITS	Interrupt status, see page 363	0x9C	R/W

Table 88: Video timing generator (VTG) registers

Register name	Description	Address offset	Type
VTG_STA	Status register, see page 363	0xA0	RO
VTG_LN_STAT	Line number status, see page 363	0xA4	RO
VTG_LN_INTERRUPT	Line value for VTG_IRQ2 output, see page 364	0xA8	R/W

VTG_MOD Mode of operation

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BNOTTREF_POL	VINPUT_POL	HINPUT_POL	VREF_POL	HREF_POL	BNOTTREF_INV_POLARITY	VTG_DISABLE	VSYNC_PAD_IN_NOUT	HSYNC_PAD_IN_NOUT	Reserved	SLAVE_BY_ODDEVEN	ODDEVEN_NOT_VSYNC	Reserved		MODE	

Address: *VTG BaseAddress* + 0x24

Type: Read/write

Reset: 00000200

Description:

[31:16] **Reserved**

[15] **BNOTTREF_POL**
0: external BNOTTREF synchro represents ‘0’ as top field, ‘1’ as bottom field
1: external BNOTTREF synchro represents ‘1’ as top field, ‘0’ as bottom field

[14] **VINPUT_POL:** input synchro to VSYNC_IN/ODDEVEN_IN
Input synchro to VSYNC_IN
0: external vertical synchro is positive (the rising edge is taken into account)
1: external vertical synchro is negative (the falling edge is taken into account)
Input synchro to ODDEVEN_IN
0: ‘0’ represents top field, ‘1’ even field
1: ‘1’ represents top field, ‘0’ even field

[13] **HINPUT_POL**
0: external horizontal synchro is positive (the rising edge is taken into account)
1: external horizontal synchro is negative (the falling edge is taken into account)

[12] **VREF_POL**
0: VREF is a positive pulse 1: VREF is negative pulse

[11] **HREF_POL**
0: HREF is a positive pulse 1: HREF is negative pulse

[10] **BNOTTREF_INV_POLARITY**
0: HSYNC_INTERLACED_1 activated on odd lines if BNOTTREF = 0 and on even lines if BNOTTREF = 1
1: HSYNC_INTERLACED_1 activated on even lines if BNOTTREF = 0 and on odd lines if BNOTTREF = 1

[9] **VTG_DISABLE**
0: sample and line counters are enabled
1: sample and line counters are disabled (reset value)
Used while initializing the module or when it has to be re-configured

[8] **VSYNC_PAD_IN_NOUT:** tri-state command of VSYNC pad of digital encoder

[7] **HSYNC_PAD_IN_NOUT:** tri-state command of HSYNC pad of digital encoder

[6] **Reserved**

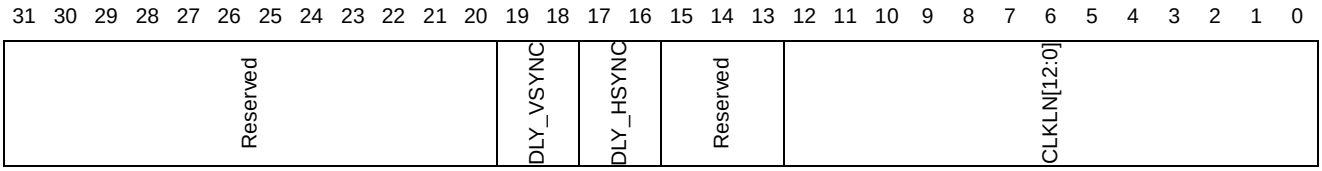
[5] **SLAVE_BY_ODDEVEN**
0: the video timing generator is slave by both horizontal and vertical synchronization signals
1: the video timing generator is slave only by ODDEVEN_IN signal, HSYNC is free running, bit ODDEVEN_NOT_VSYNC is forced to 1

[4] **ODDEVEN_NOT_VSYNC**
0: VSYNC_IN is used as vertical synchronization signal
1: ODDEVEN_IN is used as vertical synchronization signal

[3:2] **Reserved**

[1:0] **MODE**
00: master mode 01: slave by EXT_0
10: Reserved 11: reserved
Caution: if MODE[1:0] = 00 the video timing generator is set in master mode but it starts with a top field only after a soft reset.

VTG_CLKLN Clocks per line



Address: *VTG BaseAddress* + 0x28
Type: Read/write
Reset: 0
Description:

[31:20] **Reserved**

[19:18] **DLY_VSYNC[1:0]**: specifies the delay to be applied to the incoming VSYNC_EXT_0/1 (bit definition same as for DLY_HSYNC)

[17:16] **DLY_HSYNC[1:0]**: specifies the delay to be applied to the incoming HSYNC_EXT_0/1.
00 (default): no delay 01: +1 VTG_CK clock delay
10: +2 VTG_CK clock delay 11: +3 VTG_CK clock delay

[15:13] **Reserved**

[12:0] **CLKLN [12:0]:** specifies the total number of sample clock cycle per horizontal period (horizontal complete line length).
Note: the frequency of the video timing generator clock is twice the pixel frequency then CLKLN is 13 bits width

VTG_VID_TFO Video top field 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					VIDEO_VALID_TVO[10:0]											Reserved			VIDEO_VALID_THO[12:0]													

Address: *VTG BaseAddress* + 0x04

Address: Read/write

Reset: 0

Description: This register provides the x,y location of the top-left video, with respect to the current values of LINECNT and SAMPLECNT

[31:27] **Reserved**

[26:16] **VIDEO_VALID_TVO**: Y location for the first video line (top), with respect to field numbering

[15:13] **Reserved**

[12:0] **VIDEO_VALID_THO**: X location for the first video sample (left), in sample unit

VTG_VID_TFS Top field 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					VIDEO_VALID_TVS[10:0]											Reserved			VIDEO_VALID_THS[12:0]													

Address: *VTG BaseAddress* + 0x08

Address: Read/write

Reset: 0

Description: This register provides the x,y location of the bottom-right video, with respect to the current values of LINECNT and SAMPLECNT

[31:27] **Reserved**

[26:16] **VIDEO_VALID_TVO**: Y location for the first video line (bot), with respect to field numbering

[15:13] **Reserved**

[12:0] **VIDEO_VALID_THS**: X location for the first video sample (right), in sample unit

VTG_VID_BFO Bottom field 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					VIDEO_VALID_BVO[10:0]											Reserved			VIDEO_VALID_BHO[12:0]													

Address: *VTG BaseAddress* + 0x0c

Address: Read/write

Reset: 0

Description: This register provides the x,y location of the top-left video, with respect to the current values of LINECNT and SAMPLECNT

[31:27] **Reserved**

[26:16] **VIDEO_VALID_BVO**: Y location for the first video line (top), with respect to field numbering

[15:13] **Reserved**

[12:0] **VIDEO_VALID_BHO**: X location for the first video sample (left), in sample unit

VTG_VID_BFS Bottom field 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					VIDEO_VALID_BVS[10:0]											Reserved			VIDEO_VALID_BHS[12:0]													

Address: *VTG BaseAddress* + 0x10

Address: Read/write

Reset: 0

Description: This register provides the x,y location of the bottom-right video, with respect to the current values of LINECNT and SAMPLECNT.

[31:27] **Reserved**

[26:16] **VIDEO_VALID_BVS**: Y location for the first video line (bot), with respect to field numbering

[15:13] **Reserved**

[12:0] **VIDEO_VALID_BHS**: X location for the first video sample (right), in sample unit

VTG_H_HD_n HSYNC_n sample number falling and 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					HD_HS_1[12:0]											Reserved			HD_HO_1[12:0]													

Address: *VTG BaseAddress* + 0x2C (VTG_H_HD_1), 0x5C (VTG_H_HD_2), 0x7C (VTG_H_HD_3)

Address: Read/write

Reset: 0

Description: This register determines the rising sample position of the HSYNC_n signals output pulse relative to sample counter value.
The HSYNC_n outputs rise from 0 to 1 level when the sample counter takes the value HD_HO_n and falls from 1 to 0 when the sample counter takes the value HD_HS_n.

Note: The order of HD_HO and HD_HS determines the polarity of the HSYNC signal pin (YUV_HSYNC).
If the frequency of the video timing generator clock is twice the pixel frequency then HDO_n is 13 bits wide.

VTG_TOP_V_VD_n VSYNC_n vertical top field line number falling and 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					VD_TOP_VS_1[10:0]											Reserved					VD_TOP_VO_1[10:0]										

Address: *VTG BaseAddress* + 0x30 (VTG_TOP_V_VD_1), 0x60 (VTG_TOP_V_VD_2), 0x80 (VTG_TOP_V_VD_3)

Address: Read/write

Reset: 00000000

Description: This register determines the line position edges of the VSYNC_n signal pin (YUV_VSYNC) output pulse relative to line counter value.
The VSYNC_n output rises from 0 to 1 level when the line counter is equal to VD_TOP_VO_n and falls from 1 to 0 when the line counter is equal to VD_TOP_VS_n.

Note: *The order of VD_TOP_VO_1 and VD_TOP_VS_1 determines the polarity of the VSYNC_1 signal pin.*
If VD_TOP_VO_1 = VD_TOP_VS_1 the polarity of the VSYNC_1 is determined by the order of VD_TOP_HO_1 and VD_TOP_HS_1.

Note: *The min value of VD_TOP_VO_1 is 0x001.*

VTG_BOT_V_VD_n VSYNC_n vertical bot field line number output falling and 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					VD_BOT_VS_1[10:0]											Reserved					VD_BOT_VO_1[10:0]										

Address: *VTG BaseAddress* + 0x34 (VTG_BOT_V_VD_1), 0x64 (VTG_BOT_V_VD_2), 0x84 (VTG_BOT_V_VD_3)

Address: Read/write

Reset: 00000000

Description: This register determines the line position edges of the VSYNC signal pin (YUV_VSYNC) output pulse relative to line counter value.
The VSYNC_n output will rise from 0 to 1 level when the line counter is equal to VD_BOT_VO_n and will fall from 1 to 0 when the line counter is equal to VD_BOT_VS_n.
The order of VD_BOT_VO_n and VD_BOT_VS_n determines the polarity of the VSYNC_n signal pin; if VD_BOT_VO_n = VD_BOT_VS_n it is the order of VD_BOT_HO_n and VD_BOT_HS_n which determines this polarity.

Note: *The min value of VD_BOT_VO_n is 0x00.*

VTG_TOP_V_HD_n VSYNC_n vertical top field sample number rising and 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			VD_TOP_HS_1[12:0]													Reserved			VD_TOP_HO_1[12:0]												

Address: *VTG BaseAddress* + 0x38 (VTG_TOP_V_HD_1), 0x68 (VTG_TOP_V_HD_2), 0x88 (VTG_TOP_V_HD_3)

Address: Read/write

Reset: 0

Description: This register determines the line position edges of the VSYNC_n signal pin (YUV_VSYNC) output pulse relative to sample counter clear.
The VSYNC_n output will rise from 0 to 1 when the line counter takes the value VD_TOP_HO_n, and will fall from 1 to 0 when the line counter takes the value VD_TOP_HS_n.
The order of VD_TOP_HO_n and VD_TOP_HS_n determines the polarity of the VSYNC_n signal pin if VD_TOP_VO_n = VD_TOP_VS_n, else the order of VD_TOP_VO and VD_TOP_VS determines the polarity of the VSYNC signal pin.

VTG_BOT_V_HD_n VSYNC_n vertical bot field sample number rising and 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			VD_BOT_HS_1[12:0]													Reserved			VD_BOT_HO_1[12:0]												

Address: *VTG BaseAddress* + 0x3C (VTG_BOT_V_HD_1), 0x6C (VTG_BOT_V_HD_2), 0x8C (VTG_TOP_V_HD_3)

Address: Read/write

Reset: 0

Description: This register determines the line position edges of the VSYNC_n signal pin (YUV_VSYNC) output pulse relative to sample counter clear.
The VSYNC_n output will rise from 0 to 1 level when the line counter takes the value VD_BOT_HO_n and will fall from 1 to 0 when the line counter takes the value VD_BOT_HS_n.
The order of VD_BOT_HO_n and VD_BOT_HS_n determines the polarity of the VSYNC_n signal pin if VD_BOT_VO_n = VD_BOT_VS_n, else the order of VD_BOT_VO_n and VD_BOT_VS_n determines the polarity of the VSYNC_n signal pin.
The value given to VD_BOT_HDO_n allows the rising edge of VSYNC_n to be adjusted between two HSYNC_n to be able to start an analog bottom field. Normal value is HD_HO_n + CLKLN / 2.
If HD_HO_n is 0 and VD_BOT_VO_n is 0, take care that line counter is cleared when sample count = CLKLN / 2 and the first value of sample counter is CLKLN / 2 when line counter is 0, then the correct value for VD_BOT_HDO_n must be CLKLN / 2. Under this value VSYNC_n will not start for bottom field.

Note: If the frequency of the video timing generator clock is twice the pixel frequency then HDS is 13 bits width.

VTG_HLFLN Half lines 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[11:0]											

Address: *VTG BaseAddress* + 0x40

Type: Read/write

Reset: 0

Description: This register specifies the number of half lines per vertical period (the vertical period is a field.)
This register should always contain an odd value (for interlaced mode).

VTG_DRST Video timing generator 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
VTG_DRST																															

Address: *VTG BaseAddress* + 0x48

Type: Write

Reset: 0

Description: Writing to this register resets the sample counter and line counter of the video timing generator. The reset is activated once on writing at this address. None of the existing register configurations are lost on writing to this register.
It can be used in any mode (master/slave), and could be invoked, for example, if the module is not working although the configuration is correct.
When activated, this reset activates VREF and HREF. If the video timing generator is in interlaced mode (that is half lines per vertical number is odd) a top field is generated.
In slave mode, after soft reset the output syncs (VREF, HREF) go to their in-active states (as programmed in VTG_MOD register). The module then waits for the programmed active edge of input syncs (HSYNC_IN, VSYNC_IN or ODDEVEN_IN) and when it occurs, it synchronizes on to it and re-generates the synchros with active edges as programmed in VTG_MOD register

VTG_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																								LNT	LNB	Reserved	VST	VSB	Reserved		

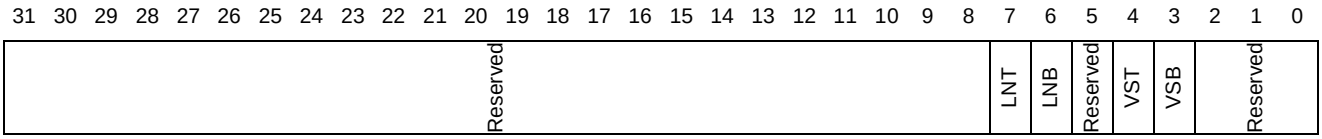
Address: *VTG BaseAddress* + 0x98

Type: Read/write

Reset: 0

Description: Any bit set in this register will enable the corresponding interrupt in VTG_IRQ line. An interrupt is generated whenever a bit in the VTG_STA register changes from 0 to 1 and the corresponding mask bit is set.

VTG_ITS Interrupt status



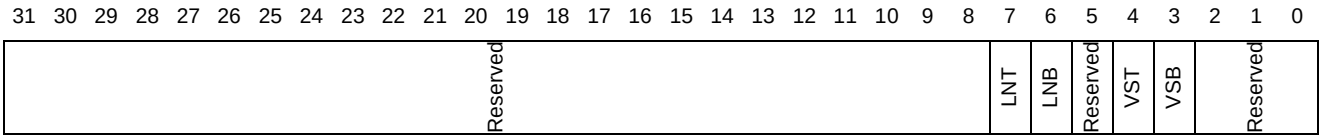
Address: *VTG BaseAddress* + 0x9c

Type: Read/write

Reset: 0

Description: When a bit in the VTG_STA register changes from 0 to 1, the corresponding bit in the VTG_ITS register is set, independent of the state of VTG_ITM. If any set VTG_ITS bit is unmasked, the line VTG_IRQ is asserted. The writing of VTG_ITS clears all bits in the register. When VTG_ITS is zero the VTG_IRQ line returns de-asserted.

VTG_STA Status register



Address: *VTG BaseAddress* + 0xa0

Type: Read

Reset: 0

Description: This register contains a set of bits which represents the status of the video timing generator. Any change from 0 to 1 of any of these bits sets the corresponding bit of the VTG_ITS register, and can thus potentially cause an interrupt on VTG_IRQ_1 or VTG_IRQ_2. VST, VSB, LNT and LNB are pulses and are unlikely ever to be read as a 1.

[31:8] **Reserved**

[7] **LNT**: line number top set for a short time when line counter reaches LINE_NUMBER_TOP value in register VTG_LN_INTERRUPT in the top field.

[6] **LNB**: line number bot set for a short time when line counter reaches LINE_NUMBER_BOT value in register VTG_LN_INTERRUPT in the bottom field.

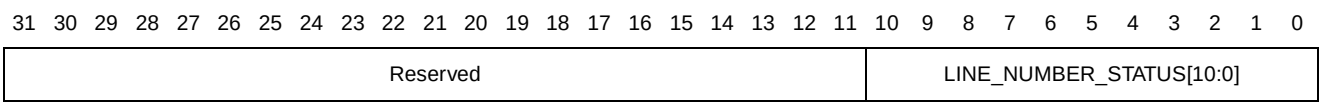
[5] **Reserved**

[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**

VTG_LN_STAT Line number status



Address: *VTG BaseAddress* + 0xA4

Type: Read

Reset: 0

Description: This register contains the value of LINE_COUNTER loaded by LN_STAT_IN.

VTG_LN_INTERRUPT Line value for VTG_IRQ2 output

31	30	29	28	27	26	25	24	23	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_BOT[10:0]											Reserved					LINE_NUMBER_TOP[10:0]										

Address: *VTG BaseAddress* + 0xA8

Type: Read/write

Reset: 0

Description:

[31:27] **Reserved**

[26:16] **LINE_NUMBER_BOT**: line number bottom value to generate a VTG_IRQ2 interrupt on the bottom field.

[15:11] **Reserved**

[10:0] **LINE_NUMBER_TOP**: line number top value to generate VTG_IRQ2 interrupt on the top field.

34 Quadruple video DAC

34.1 Overview

The STx5119 has one quadruple, 10-bit, video DAC that provides output signals in CVBS+RGB, CVBS+YUV or SVHS(Y/C) + CVBS + SCVBS. Figure 147 shows the DAC architecture.

A reference circuit controlled by one external precision resistor sets the full-scale output for the quadruple DAC. The voltage across the resistor is defined by an internal bandgap reference circuit. Dedicated pads are used to ensure good noise immunity.

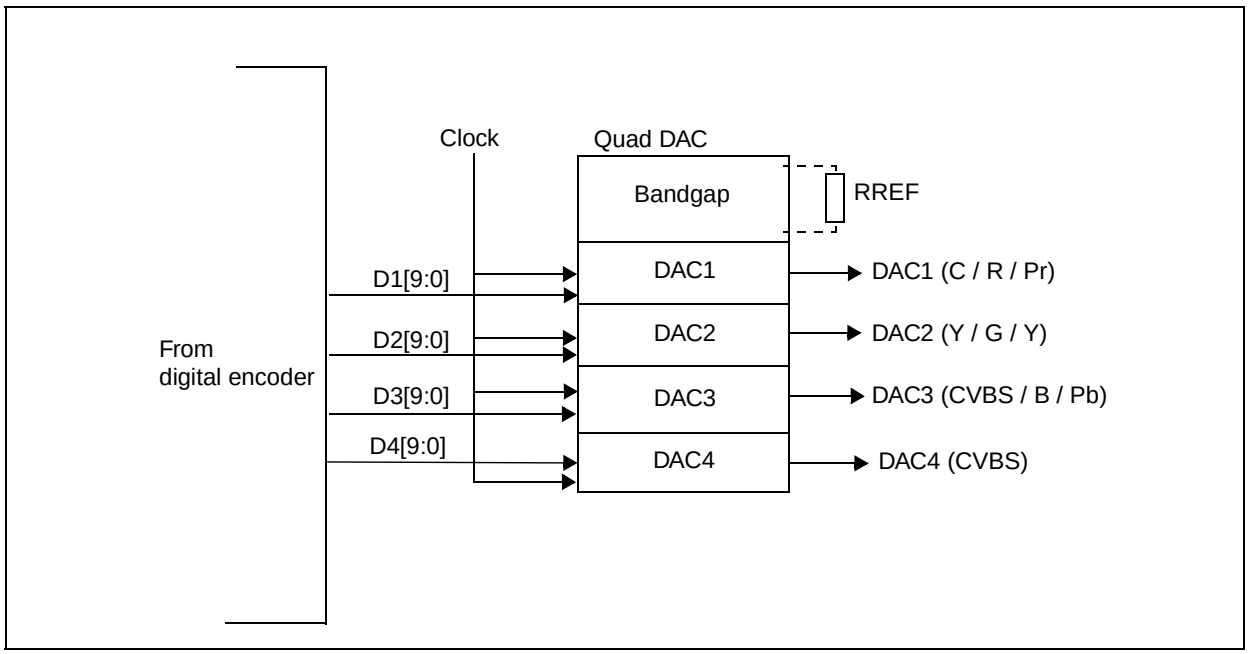
The quadruple DAC uses a 6/4 segmented architecture. The four least significant bits control a binary weighted current matrix and the six most significant bits control a thermometric current matrix. It is able to output 8 mA current.

The generic DAC current sources may provide an output range of up to 1.6 V with good linearity. Sampled data is available on video outputs after one clock period on the next rising clock edge.

Dedicated 3.3 V analog and digital power and ground supplies are used to reduce noise effects. The main digital 1.2 V supply uses the chip core VDD and GND.

The DAC output settings are controlled by DEN_CFG13.

Figure 147: Quadruple video DAC schematic



34.2 Input codes for video application

The table below lists the reference input codes generated by the digital encoder depending on the configuration for the DACs and the standard.

Table 89: Reference input codes

	Y-PAL/SECAM	Y-NTSC	RGB
WHITE(235)	816.00	800.00	616.00
BLACK(16)	256.00	240.00	41.00
SYNC TIP	16.00	16.00	-

Note: CVBS = Y + C, therefore chrominance component has no effect on the CVBS *signal (C is null).*

34.3 Video output voltage level

The resistor R_{ref} connected to the bandgap, controls the DAC output current. For the maximum code (1023 in decimal):

I_{out} (max) = 77.696 / R_{ref}

For example, with a typical value of R_{ref} = 9.7 kΩ, I_{out} (max) = 8 mA for each DAC.

The value of R_{ref} must be greater or equal to 9.7 kΩ so as not to exceed the maximum output current. Due to the sensitive relationship between the DAC and R_{ref}, the tolerance of R_{ref} value must be small, typically 1%.

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} * I_{out} (max)

For example, with a typical load value

R_{load} = 165 Ω,

I_{out} (max) = 8 mA,

V_{out} (max) = 1.32 V.

V_{out} should always be lower than 1.6 V to guarantee linearity.

For any given digital input code, the output voltage of the DAC is given by the following formula:

$V_{out} = D_{in} / 1023 * V_{out}$ (max) = (D_{in} * R_{load} * 77.696) / (R_{ref} * 1023)

$V_{out} = D_{in} * R_{load} * 0.0759 / R_{ref}$

For example, with R_{load} = 165 Ω, R_{ref} = 9.7 kΩ and D_{in} = 802, V_{out} = 1.04 V.

34.4 Video specifications and DAC set up

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_{ref} resistor must be chosen according to the value of R_{load} and the previous formula, to achieve the standard output video range. The minimum value for R_{ref} is 9.7 kΩ.

According to video specifications ITU-R BT 601, the nominal sampling frequency is 27 MHz. The DACs clock is buffered from the digital encoder clock, which is usually generated externally by a DCO. It must be as clean as possible to achieve a good signal to noise ratio.

The video DAC is enabled and managed through bits 9 to 16 in the Interconnect Config register D as shown in Table 90 and Table 91.

Table 90: Video DAC power management

HZ	POFF	Power state
0	0	Normal Mode
1	0	High Impedance
X	1	Power off Mode

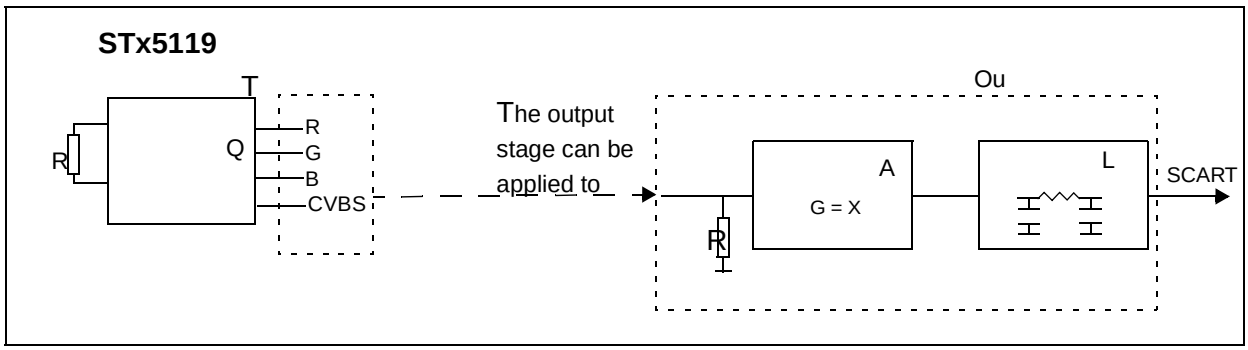
Table 91: Video DAC command management

U,V,W,X[9:0]	CMDS	CMDR	U,V,W,X OUT	GND	DAC power state
code	0	0	Icode	Imax-Icode	Normal Mode
X	0	1	0	Imax	
X	1	0	Imax	0	
X	1	1	Imax / 2	Imax / 2	

34.5 Output stage adaptation and amplification

Figure 148 shows the block diagram of the output stage. 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 output signal level and impedance may be different).

Figure 148: Output stage block diagram



The amplifier gain must be in accordance with one of the Quad-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} = 9.7 kΩ and R_{load} = 165 Ω), a video amplifier with a +6 dB gain should be adequate. The ideal input impedance of the output stage should be greater than R_{load}.

The Quad-DACs have no cut off frequency, therefore, a low pass filter (around 10 MHz) must be applied to remove harmonics of (mainly) 27 MHz. If additional attenuation is applied by the filter due to imperfection in the amplifier (generally degrading the C/L ratio), correction must be applied to preserve 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, additional correction may be required.

35 Audio decoder

35.1 Overview

The audio decoder subsystem consists of a dedicated core and a hardware wrapper that interfaces the core to the PCM player and S/PDIF player through the STBus interconnect. The audio decoder receives the elementary stream (ES) data. PTS extraction, start code detection and the routing of un-decoded Dolby® Digital streams to the S/PDIF is managed by software.

The PCM player is described in Chapter 37: *PCM player* on page 380 and S/PDIF player is described in Chapter 39: *S/PDIF player* on page 388.

35.1.1 Input formats

The audio decoder accepts:

- MPEG-1 layers I and II,
- MPEG-2.

35.1.2 Audio video synchronization

Skip frame, pause blocks and soft mute frame features are used to synchronize audio and video data, these features are implemented by software. Soft muting of the audio DACs is achieved by setting bit 18 in the CONFIG_CONTROL_D register.

35.1.3 Sampling frequencies

Sampling frequencies of F_s = 48, 44.1, 32 kHz and half sampling frequencies are supported.

35.1.4 Audio decode data flow

The audio subsystem architecture is shown in Figure 149, and the decode flow is shown in figure Figure 150.

The decode flow is described below.

1. Following demultiplexing, the PTI stores PES data in a buffer located in external memory.
2. The host CPU software reads the PES bit-buffer, performs PES parsing and detects the sync words for MPEG1-layers I and II. The PTS extracted from the PES header is stored together with the associated sync words addresses in the SyncWord and PTS buffer in external memory.
3. The host CPU software parses the ES data and uses it to verify correct sync word detection. Bit rate and sampling frequency information included in the ES stream can be extracted by the CPU or read from the audio decoder. The audio frame boundaries can be determined and the frame size checked.
4. The audio decoder, which also has an internal sync detector, is fed with ES data starting at the sync word boundary. The decoder latency is approximately 7000 cycles for layer 1 and 9500 cycles for layer 2. Only one frame is present inside the decoder at any instant. The sync word is not 32-bit aligned, software ensures that no false synchronization pattern is present before the correct sync word. The ES data is decoded and the PCM data is fed to the decoder output. The decoder signals when PCM decoded data is available. On detection of a CRC error, the CRC detector generates an interrupt (if enabled). If a CRC error occurs, frame decoding continues (the correct number of samples is produced but with corrupted values) and a flag is raised. Soft-mute is performed by software and not by the decoder core. Soft muting of the audio DACs is achieved by setting bit 18 in the CONFIG_CONTROL_D register.

5. The FDMA uses pacing signals from the decoder to write the compressed data and to read the decoded PCM data.
6. The audio/video synchronization is performed by the sync detector software in the CPU. The software controls the FDMA to skip one or more frames when the decoded data is written to the PCM player.
7. In compressed mode, the data is extracted directly from the circular buffer and formatted according to the IEC-61937 standard.
8. It is also possible to output the decoded PCM data on the S/PDIF interface according to the IEC-60958 standard. Data formatting is managed by the CPU software.

Figure 149: Audio system architecture

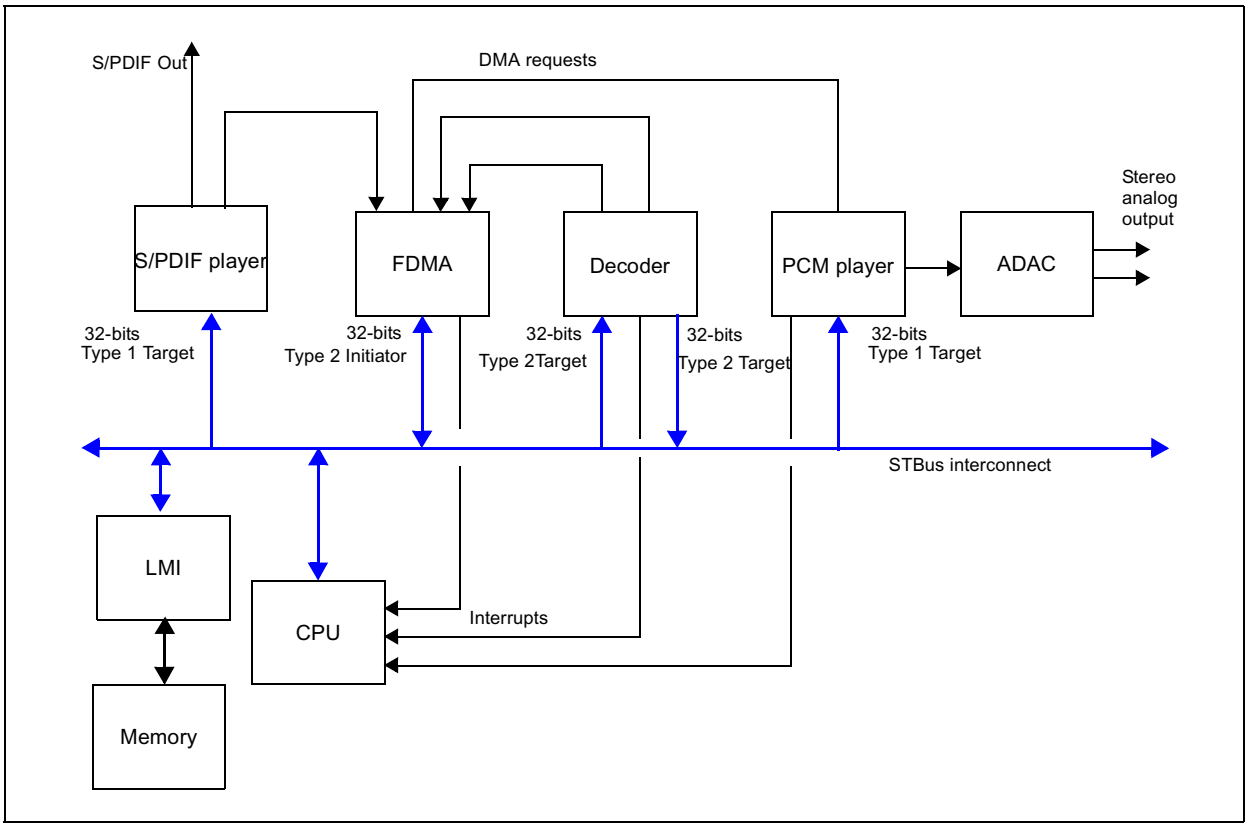
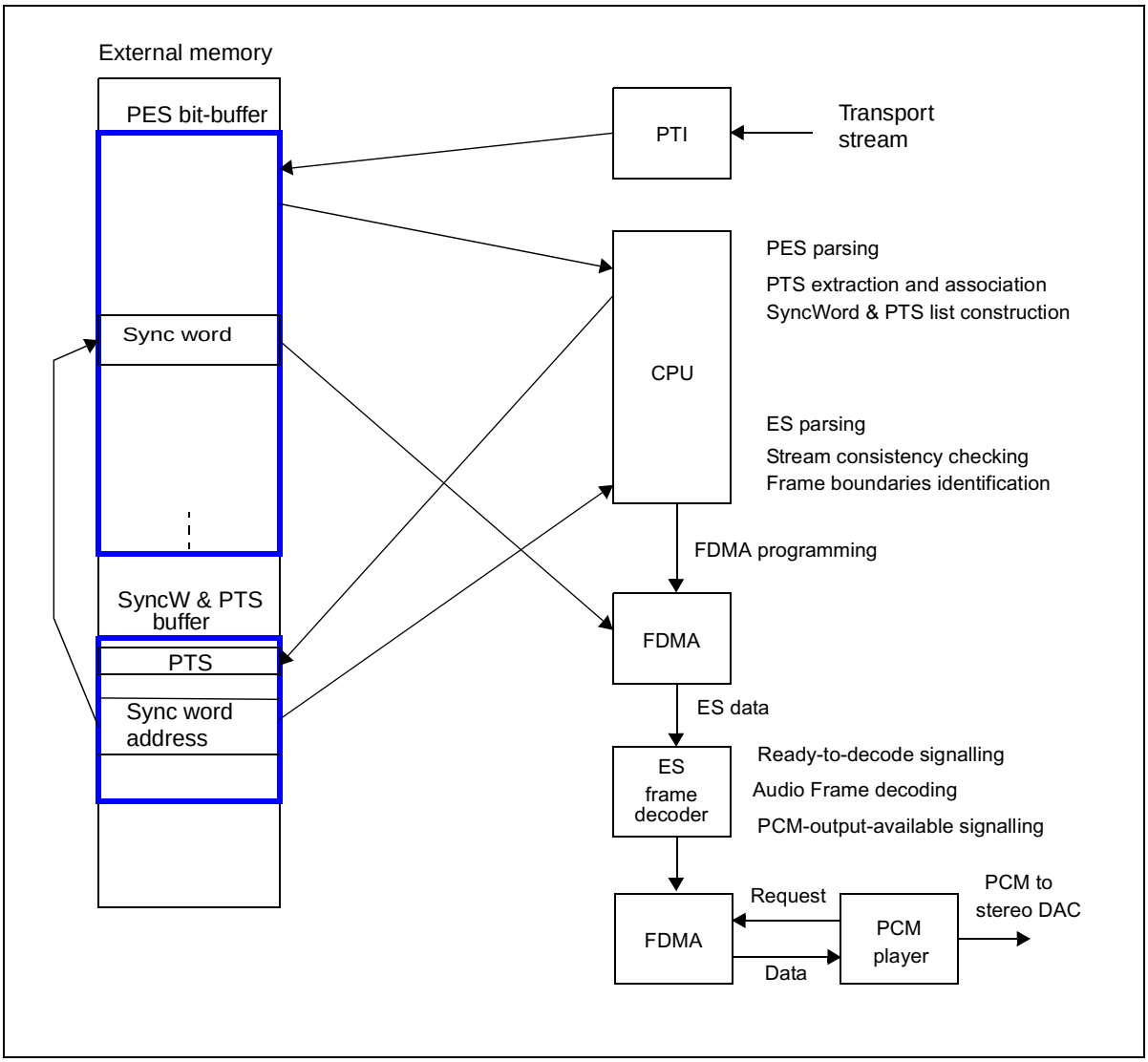


Figure 150: Audio decode flow



35.1.5 Reset

Software resets the audio decoder using the following procedure.

1. Soft reset: 1 is written to register AUD_DEC_CONFIG. The decoder remains in the reset state as along as the soft reset bit is asserted. The decoder comes out of the reset only after the soft reset bit is reset to 0 by the software.
2. Asserting the soft-reset doesn't change the contents of any other register. Other registers are modified separately by writing to them.
3. An interrupt is asserted, if enabled, after the decoder core is soft reset.
4. The soft reset bit in AUD_DEC_STATUS register signals the completion of a soft reset.

The decoder core is ready to decode frames after a hardware or soft reset.

35.1.6 Clocks

The audio decoder uses audio video clock running at 50 MHz. The PCM clock and LR clock are fed to the PCM player and S/PDIF clock is fed to the S/PDIF player.

35.1.7 Decode mode

The audio decoder decodes MPEG1 layer 1 and 2 streams only. During the decoding process the AUD_DEC_STATUS register holds information such as bit rate, sampling frequency and audio mode. The audio decoder is ready to decode after de-assertion of either hard reset or soft reset. No bits have to be set to start the decoding process.

35.1.8 Interrupt register

The audio decoder can be configured to generate an interrupt in a variety of conditions. Enabling the interrupt is done by setting the corresponding bit in the AUD_DEC_INT_ENABLE register. Writing 0 to ENB_INT (bit0) disables all the interrupts. When an interrupt is detected the source of the interrupt can be retrieved by reading the AUD_DEC_INT_STATUS register. The interrupt can be cleared by writing 1 to the corresponding bit of the AUD_DEC_INT_CLEAR register.

35.1.9 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. The intended use is for satellite dish alignment (22 kHz) and not for generating a beep tone through the television speaker.

In STx5119, the audio decoder does not generate the beep tone. The PCM file corresponding to the beep tone is initialized in memory by the software. The CPU moves the data stream corresponding to the beep tone to the PCM player using the FDMA.

36 Audio decoder registers

Addresses are provided as the *AudioBaseAddress* + offset.

The *AudioBaseAddress* is:

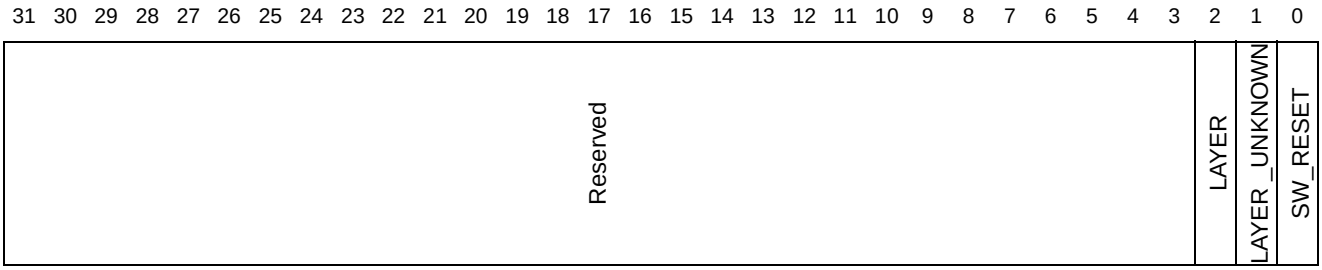
0x2070 0000.

Table 92: Audio decoder registers

Register name	Description	Address offset	Type
AUD_DEC_CONFIG	Configures the Audio decoder, see page 373	0x000	R/W
AUD_DEC_STATUS	Status of the Audio decoder can be read here, see page 374	0x004	RO
AUD_DEC_INT_ENABLE	Register to enable the interrupt, see page 376	0x008	R/W
AUD_DEC_INT_STATUS	Interrupt status register, see page 377	0x00C	RO
AUD_DEC_INT_CLEAR	Interrupt clear register, see page 378	0x010	R/W
AUD_DEC_VOL_CTRL	Volume control register, see page 379	0x014	R/W
AUD_DEC_CD_IN	Compressed data input register, see page 379	0x040 - 0x05C	R/W
AUD_DEC_PCM _OUT	Decoded PCM out register, see page 379	0x060 - 0x07C	R/W

36.1 Audio decoder registers

AUD_DEC_CONFIG Audio decoder configuration



Address: *AudioBaseAddress*+ 0x000

Type: Read/write

Reset: 0: [2:0]

Description: The audio decoder is configured by writing to this register.

[31:3] **Reserved**[1]

[2] **LAYER**:
0: Layer 2
1: Layer 1

[1] **LAYER _UNKNOWN**
0: Layer unknown, decoder detects the layer from the input stream
1: Decoder locks to the layer indicated in the layer bit

[0] **SW_RESET**
0:
1: Soft reset asserted

1. Writes to reserved bits are ignored, reads result in all zeros.

AUD_DEC_STATUS Audio decoder 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
PCMFIFO_STATUS				CDFIFO_STATUS				Reserved			COPY_RIGHT	MODE_EXTN		AUDIO_MODE		SAMPFREQ_INDEX		BITRATE_INDEX				LAYER_ID		VERSION_ID	EMPHASIS		ORIGINAL	CRC_ERROR	FRAME_LOCKED	SW_RESET	Reserved

Address: *AudioBaseAddress*+ 0x004

Type: Read only

Reset: 0000: [13:10] [27:24] [31:28], 00: [6:5] [9:8] [15:14] [17:16] [19:18], all other reset to 0

Description:

[31:28] **PCMFIFO_STATUS**
0000: No words in PCMFIFO
0001: One word in PCMFIFO
0010: Two words in PCMFIFO
.....
0111: Seven words in PCMFIFO
1000: Eight words in PCMFIFO - FIFO full

[27:24] **CDFIFO_STATUS**
0000: No words in CDFIFO
0001: One word in CDFIFO
0010: Two words in CDFIFO
.....
0111: Seven words in CDFIFO
1000: Eight words in CDFIFO - FIFO full

[23:21] **Reserved**

[20] **COPY_RIGHT**
0:
1: Bit stream copyright protected

[19:18] **MODE_EXTN**
see Table 93

[17:16] **AUDIO_MODE**
00: Stereo
01: Joint stereo - intensity mode
10 Dual channel
11: Single channel

[15:14] **SAMPFREQ_INDEX**

For version ID = 1
00: 44.1kHz
01: 48 kHz
10: 32 kHz
11: Reserved

For version ID = 0
00: 22.05 kHz
01: 24 kHz
10: 16 kHz
11: Reserved

[13:10] **BITRATE_INDEX**
see Table 94

[9:8] **LAYER_ID**
10: Layer 2
11: Layer 1
all others unknown

[7] **VERSION_ID**
0: ISO/IEC 13818-3 low sample rate
1: ISO/IEC 11172-3 high sample rate

[6:5] **EMPHASIS**
00: None
01: 50/15 microseconds
10: Reserved
11: CCITT J.17

[4] **ORIGINAL**
1: bit stream is original

[3] **CRC_ERROR**
0:
1: CRC error in the current frame

[2] **FRAME_LOCKED**
0:
1: Frame sync word found and frame locked

[1] **SW_RESET**
0:
1: Soft reset assertion complete

[0] **Reserved**

Mode extension shows the MPEG audio mode used in joint stereo mode. It indicates which sub-bands are in intensity zero. All other sub-bands are coded on stereo. Table 93 lists the values of mode extension bits.

Table 93: Mode extension

Mode Extn	Sub bands in intensity zero	Bound
00	4-31	4
01	8-31	8
10	12-31	12
11	16-31	16

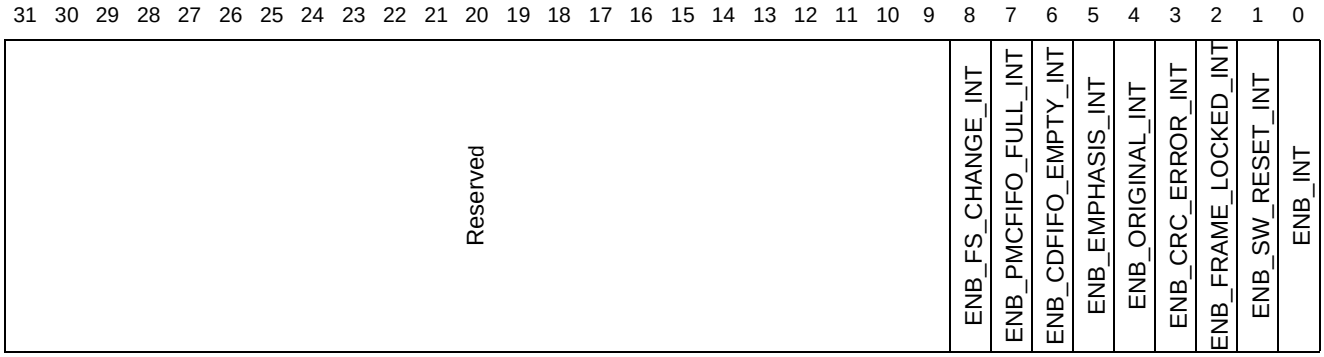
Table 94: Bit rate index

Bit rate Index	Bit rate in Kb/s for FS = 32KHz, 44.1Khz or 48KHz	
	Layer I	Layer II
0000	Free	Free
0001	32	32
0010	64	48
0011	96	56
0100	128	64
0101	160	80
0110	192	96
0111	224	112
1000	256	128
1001	288	160
1010	320	192
1011	352	224
1100	384	256

Table 94: Bit rate index

Bit rate Index	Bit rate in Kb/s for FS = 32KHz, 44.1Khz or 48KHz	
	Layer I	Layer II
1101	416	320
1110	488	384
1111	Forbidden	Forbidden

AUD_DEC_INT_ENABLE Interrupt enable



Address: *AudioBaseAddress*+ 0x008

Type: Read/write

Reset: 0

Description: Interrupts can be enable on different events by setting the bits of this register appropriately.

[31:9] **Reserved**

[8] **ENB_FS_CHANGE_INT**

0: 1: Enables Interrupt on change of sampling freq.

[7] **ENB_PMCFIFO_FULL_INT**

0: 1: Enables Interrupt on PCM FIFO full

[6] **ENB_CDFIFO_EMPTY_INT**

0: 1: Enables Interrupt on CDFIFO empty

[5] **ENB_EMPHASIS_INT**

0: 1: Enables Interrupt on change of de-emphasis stat.

[4] **ENB_ORIGINAL_INT**

0: 1: Enables Interrupt when bit stream is copy

[3] **ENB_CRC_ERROR_INT**

0: 1: Enables interrupt on CRC error

[2] **ENB_FRAME_LOCKED_INT**

0: 1: Enables interrupt on frame sync status change

[1] **ENB_SW_RESET_INT**

0: 1: Enables interrupt after Soft reset completion

[0] **ENB_INT:** Global interrupt disable

0: disables all interrupts 1:

AUD_DEC_INT_STATUS 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																							FS_CHANGE_INT	PMCFIFO_FULL_INT	CDFIFO_EMPTY_INT	EMPHASIS_INT	ORIGINAL_INT	CRC_ERROR_INT	FRAME_LOCKED_INT	SW_RESET_INT	NT_PENDING

Address: *AudioBaseAddress*+ 0x00C

Type: Read only

Reset: 0

Description:

[31:9] **Reserved**

[8] **FS_CHANGE_INT**

0: 1: Sampling frequency change interrupt pending

[7] **PMCFIFO_FULL_INT**

0: 1: PCM FIFO full interrupt pending

[6] **CDFIFO_EMPTY_INT**

0: 1: CDFIFO empty interrupt pending

[5] **EMPHASIS_INT**

0: 1: De-emphasis status change interrupt pending

[4] **ORIGINAL_INT**

0: 1: Original bit stream interrupt pending

[3] **CRC_ERROR_INT**

0: 1: CRC error interrupt pending

[2] **FRAME_LOCKED_INT**

0: 1: Frame sync interrupt pending

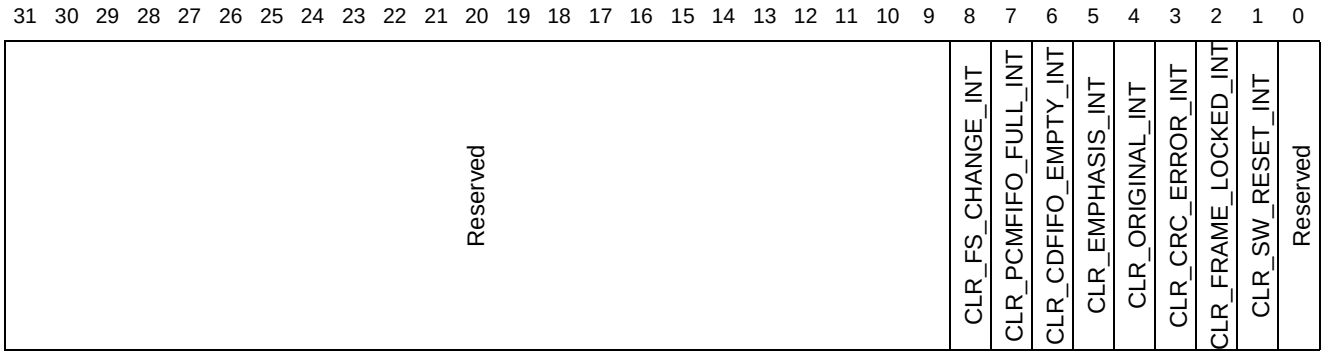
[1] **SW_RESET_INT**

0: 1: Soft reset interrupt pending

[0] **INT_PENDING**

0: 1: Interrupt pending

AUD_DEC_INT_CLEAR Interrupt clear



Address: *AudioBaseAddress*+ 0x010

Type: Write only

Reset: 0

Description: On writing 1, the interrupt is cleared and the corresponding interrupt bit in the interrupt status register is reset. Writing 0 does not affect the status of the interrupt.

[31:9] **Reserved**

[8] **CLR_FS_CHANGE_INT**

0: 1: Clears Sampling frequency change interrupt

[7] **CLR_PCMFIFO_FULL_INT**

0: 1: Clears PCM FIFO full interrupt

[6] **CLR_CDFIFO_EMPTY_INT**

0: 1: Clears CDFIFO empty interrupt

[5] **CLR_EMPHASIS_INT**

0: 1: Clears De-emphasis status change interrupt

[4] **CLR_ORIGINAL_INT**

0: 1: Clears Original bit stream interrupt

[3] **CLR_CRC_ERROR_INT**

0: 1: Clears CRC error interrupt

[2] **CLR_FRAME_LOCKED_INT**

0: 1: Clears Frame sync interrupt

[1] **CLR_SW_RESET_INT**

0: 1: Clears Soft reset interrupt

[0] **Reserved**

AUD_DEC_VOLUME_CONTROL Volume 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															MIX	Reserved		RIGHT						Reserved		LEFT					

Address: *AudioBaseAddress*+ 0x014

Type: Read/write

Reset: 0x00: [0:5] [13:8], 0: [16]

Description: Attenuation of the PCM output is programmed through this register. The attenuation can be programmed from 0dB to 63 dB in steps of 1dB.

[31:17] **Reserved**

[16] **MIX**

0: 1: PCM out is mixed result of right and left channels

[15:14] **Reserved**

[13:8] **RIGHT**
Attenuation for right channel

[7:6] **Reserved**

[5:0] **LEFT**
Attenuation for left channel

AUD_DEC_CD_IN Compressed data in

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

Address: *AudioBaseAddress*+ 0x040 - 0x05C

Type: Write

Reset: 0x00000000

Description: The compressed data is written to a FIFO which is instantiated at this address. The address range of 0x40 to 0x5C points to the same element of the FIFO.

[31:0] **CD_DATA_IN**

AUD_DEC_PCM_OUT Decompressed data out

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

Address: *AudioBaseAddress*+ 0x060 - 0x07C

Type: Read

Reset: 0x00000000

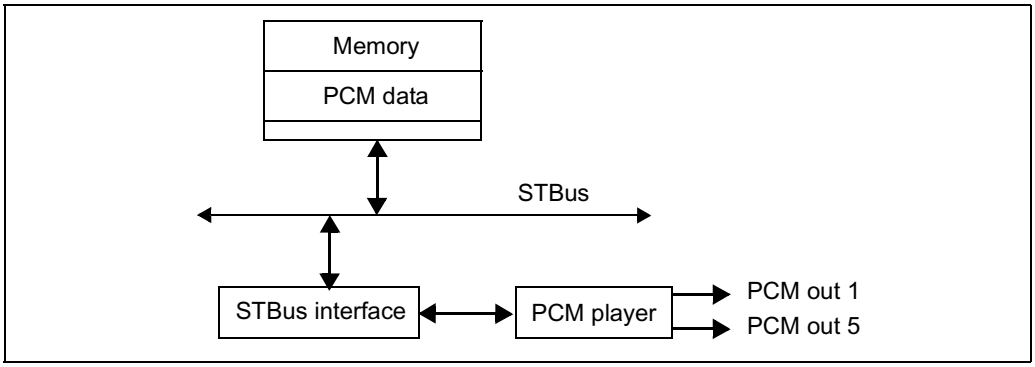
Description: The decompressed PCM data is read from a FIFO which is instantiated at this address. The address RANGE 0x060-0x07C points to the same element of the FIFO.

[31:0] **CD_DATA_OUT**

37 PCM player

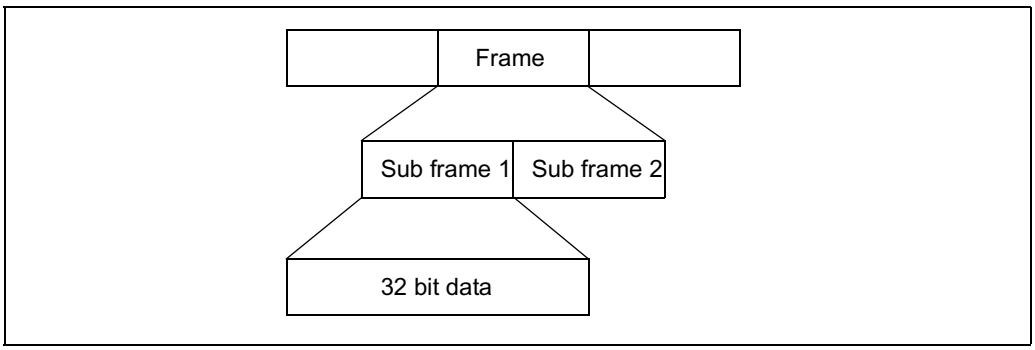
The PCM player plays audio data stored in memory using the appropriate STBus interface. The player is compliant with Phillips I^2S and SONY audio data format. The number of serial outputs is variable from 1 to 5 depending upon the number of stereo audio channels.

Figure 151: System diagram



The basic PCM element consists of a sub-frame made up of a 32 bit word (up up to 32 audio data bits). A frame consists of two sub-frames.

Figure 152: PCM stream



Input

The PCM player receives data and control words from the STBus interface via a simple register interface with a request/acknowledge for clock synchronization.

Output

The output consists of five serial stereo data channels, a bit clock and an LR clock.

Figure 153: PCM player block diagram

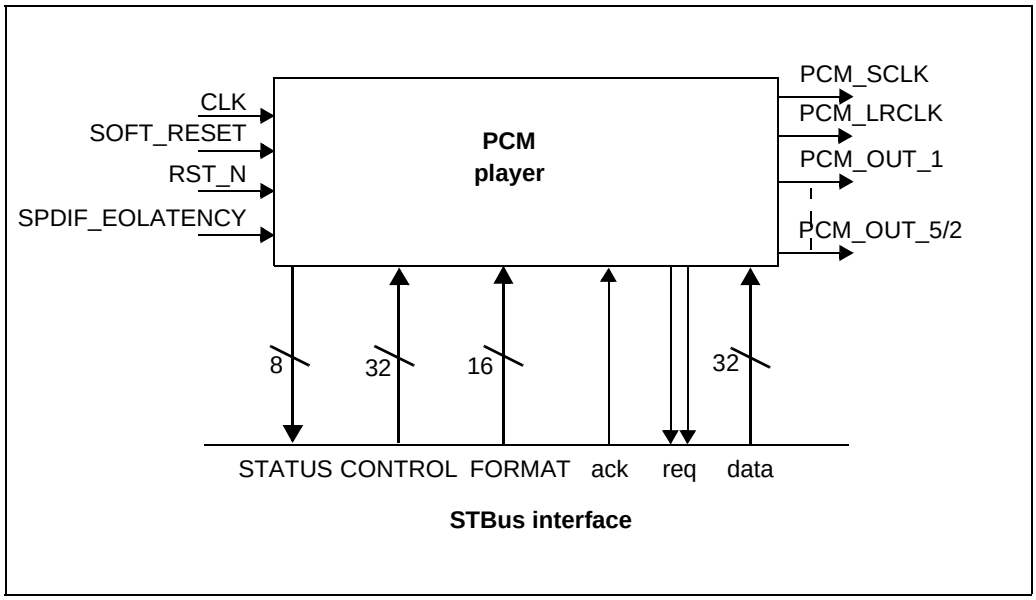
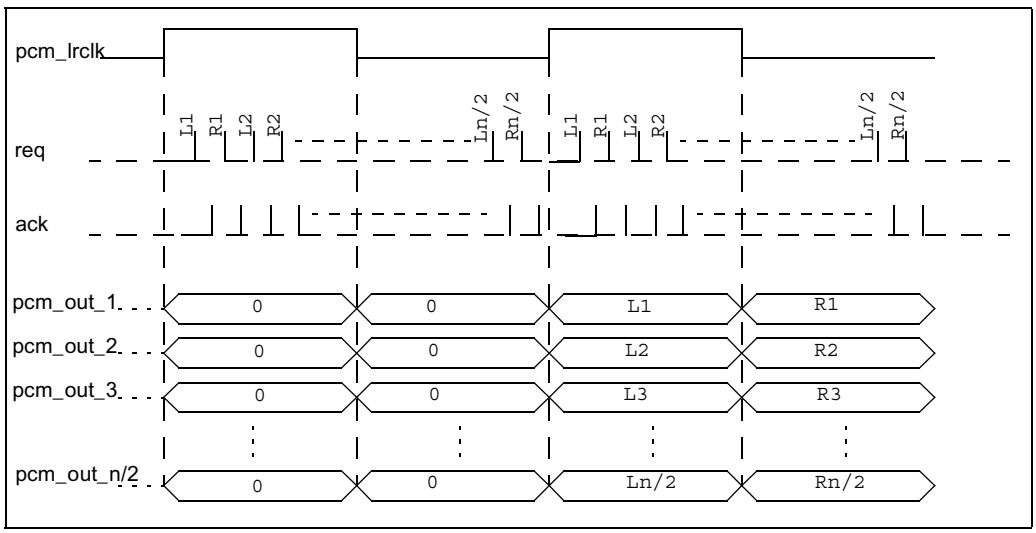
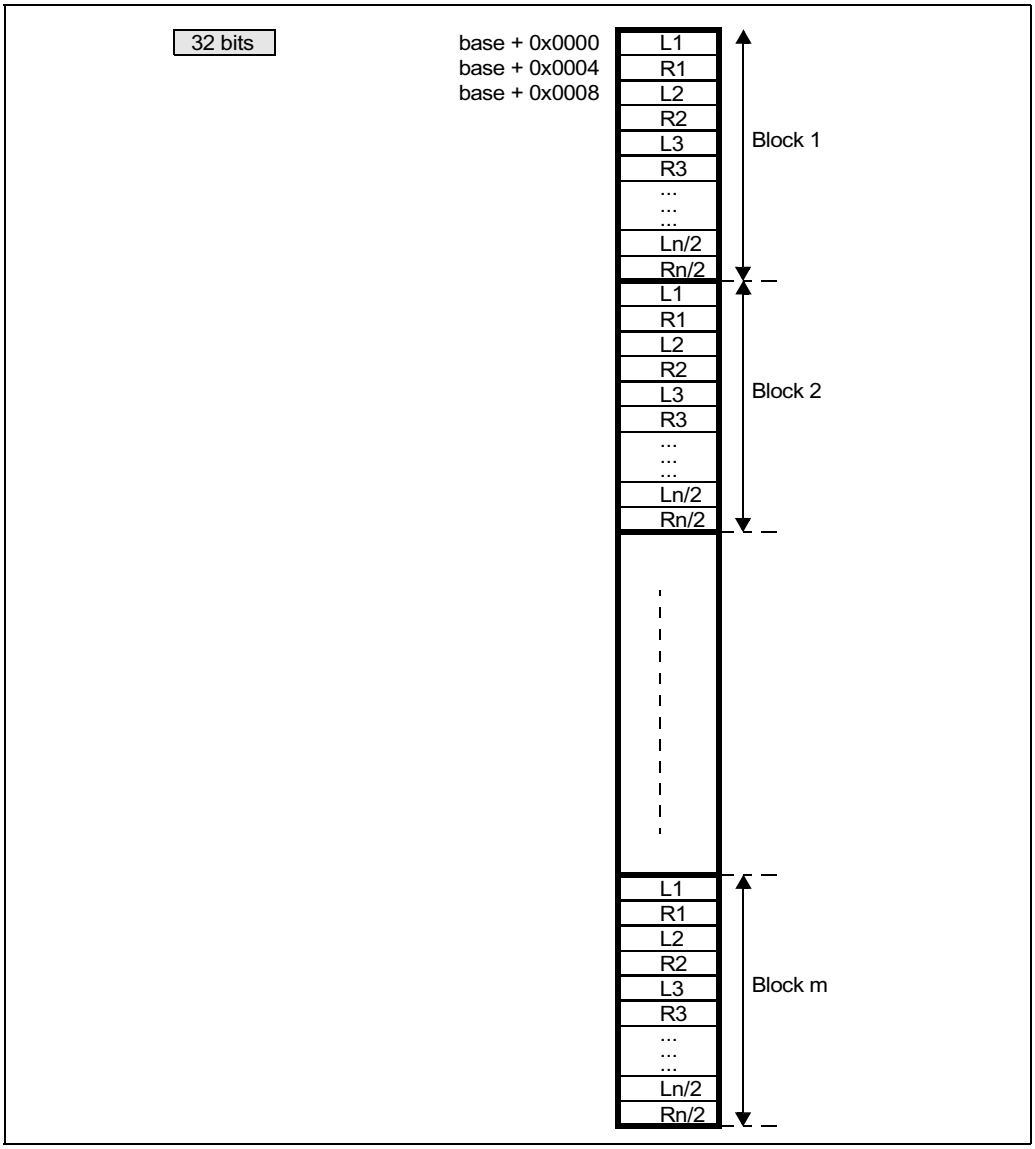


Figure 154: Signal timing



The first sample sent by the STBus interface to the PCM player block is a LEFT sample.

Figure 155: Audio samples memory storage proposal - n audio channels case

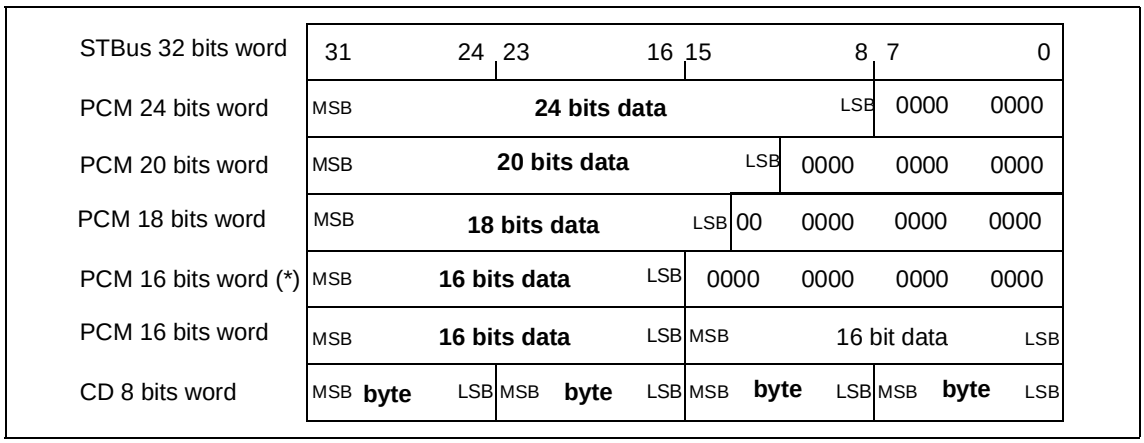


37.1 Data

The PCM data consists of up to 24 bits audio samples.

Data is located on an external memory, and should be aligned and formatted as shown in Figure 156.

Figure 156: STBus data



(*): for S/PDIF player compatibility.

38 PCM player registers

Addresses are provided as the *PCMplayerbase address* + offset.

The *PCMplayer base address* is:

0x2140 0000.

Table 95: PCM player registers

Register name	Description	Address offset	Type
PCMP_CONTROL	PCM player control, see page 385	0x01C	R/W
PCMP_STATUS	PCM player status, see page 386	0x020	R/W
PCMP_FORMAT	PCM player format, see page 387	0x024	R/W
PCMP_DATA	PCM player data, see page 387	0x004	R/W

PCMP_CONTROL PCM player control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMBER OF SAMPLES TO READ																			WAIT_EOLATENCY	DIVIDER[7:0]								ROUNDING	FORMAT	OPERATION	

Address: *PCMplayerbase address* + 0x01C

Type: Read/write

Reset: 0: [31:2], 00: [1:0]

[31:2] NUMBER OF SAMPLES TO READ
Each time the number of samples read by the PCM player is equal to this number, the associated bit in the status register is set to 1

[12] WAIT_EOLATENCY
0: do not wait for SPDIF latency signal
1: Wait for SPDIF latency signal SPDIF_EOLATENCY to be equal to 1 before sending out PCM samples. PCM null data is sent out on the output channels until there is a SPDIF_EOLATENCY signal. SPDIF_EOLATENCY is a pulse signal, synchronous to the PCM player clock.

[11:4] DIVIDER[7:0]
Audio clock divider = (CLK / (2 x PCM_SCLK)). Refer to Table 96 and Table 97.

[3] ROUNDING
0: No rounding
1: 16 bit rounding on PCM audio data output

[2] FORMAT
0: 16 bits / 0
1: 16 bits/16 bits

[0:1] OPERATION
00: PCM OFF
01: PCM MUTE
10: PCM mode ON (bits per sample >= 16)
11: CD mode ON

Table 96: Divider values for Tlrclk = 64 Tsclk cases

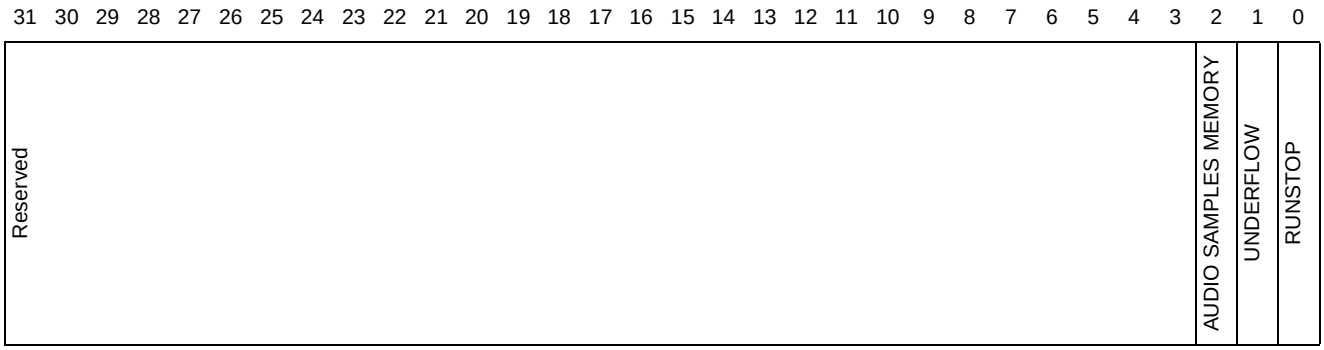
Divider value	DAC clock
0	PCM_SCLK = clk
1	128 x Fs
6	192 x Fs[1]
2	256 x Fs
3	384 x Fs
4	512 x Fs
6	768 x Fs

1. If the dac clock = 192 x Fs, the clock applied to the PCM player block has to be set to 384 x Fs.

Table 97: Divider values for Tlrclk = 32 Tsclk cases

Divider value	DAC clock
0	PCM_SCLK = CLK
2	128 x Fs
3	192 x Fs
4	256 x Fs
6	384 x Fs
8	512 x Fs
12	768 x Fs

PCMP_STATUS PCM player status



Address: *PCMplayerbase address* + 0x020

Type: Read/write

Reset: 0

Description:

[7:3] Reserved

[2] AUDIO SAMPLES MEMORY FULLY READ
1: the number of samples to read defined in the CONTROL_REG[31:13] have been received by the PCM player block (that is, part of the memory can be updated with new samples)

[1] UNDERFLOW
Set to 1 when data underflow detected

[0] RUNSTOP
0: PCM stopped
1: PCM running

PCMP_FORMAT PCM player format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								ORDER	ALIGN	PADDING	SCLKEDGE	LRLEVEL	DATALENGTH[1:0]		BITPERSUBFRAME

Address: *PCMplayerbase address* + 0x024

Type: Read/write

Reset: 0x01

Description: The control bits are directly connected from the STBus interface and are used to configure the serial interface. These bits are re-synchronized inside the PCM player with the PCM player clock.

[7] ORDER
0: data is sent LSB first
1: data is sent MSB first

[6] ALIGN
0: data is in the first SCLK cycles of LRCLK
1: data is in the last SCLK cycles of LRCLK

[5] PADDING
0: left word when PCM_LRCLK low,
1: left word when PCM_LRCLK high

[4] SCLKEDGE
0: data clocked on rising edge of PCM_SCLK
1: data clocked on falling edge of PCM_SCLK

[3] LRLEVEL
0: left word when PCM_LRCLK low,
1: left word when PCM_LRCLK high

[2:1] DATALENGTH[1:0]
00: 24 bit
01: 20 bit
10: 18 bit
11: 16 bit

[0] BITPERSUBFRAME
0: 32 bit per sub frame
1: 16 bit per sub frame

PCMP_DATA PCM player data

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

Address: *PCMplayerbase address* + 0x004

Type: Read/write

Reset: 0x01

Description:

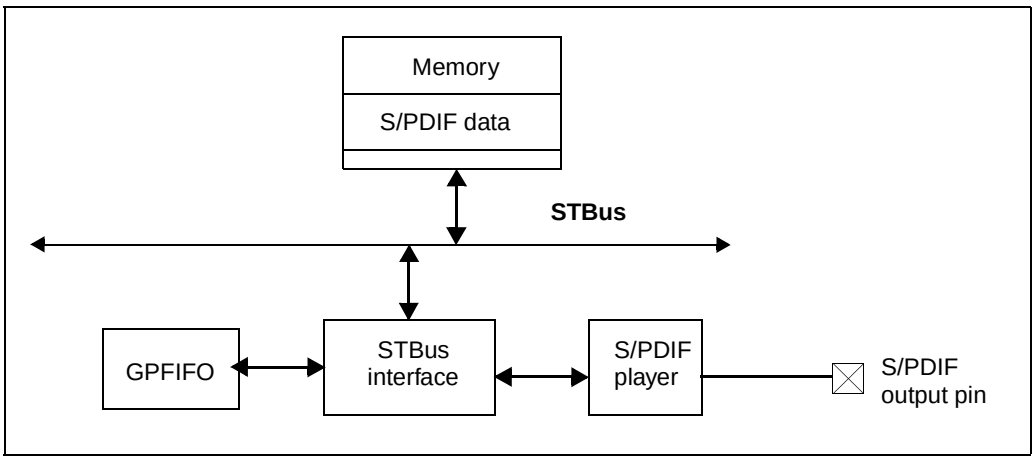
[31:0] **SUBFRAME DATA**

39 S/PDIF player

39.1 Overview

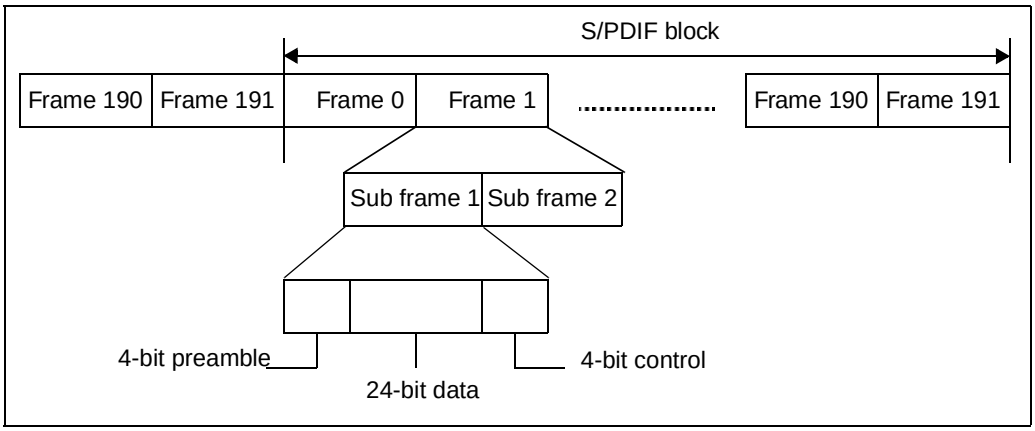
The S/PDIF player plays PCM audio data or audio-encoded bit streams stored in memory in the S/PDIF format using the STBus interface. The player is compliant with IEC-60958 for audio data and IEC-61937 for compressed audio data. The block receives data from memory using the FDMA, formats the data and sends it out on the S/PDIF output pin.

Figure 157: System diagram



The basic S/PDIF element is a subframe that consists of a 32-bit word (a 4-bit preamble, up to 24 audio data bits and 4 control bits). A frame consists of two subframes and a block consists of 192 frames.

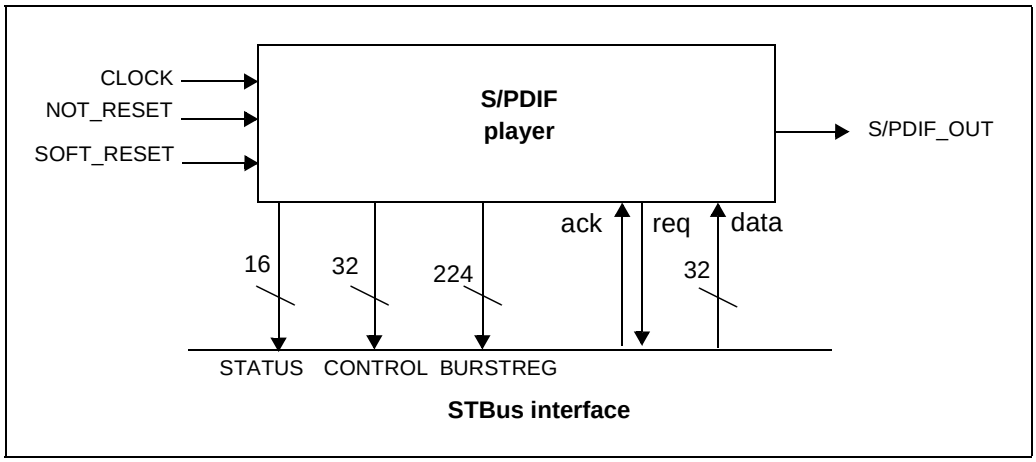
Figure 158: S/PDIF stream



39.1.1 S/PDIF Signals

The S/PDIF player receives data and control words from the STBus interface using a simple register interface with a request/acknowledge signal for clock synchronization. Figure 159 show the S/PDIF player architecture.

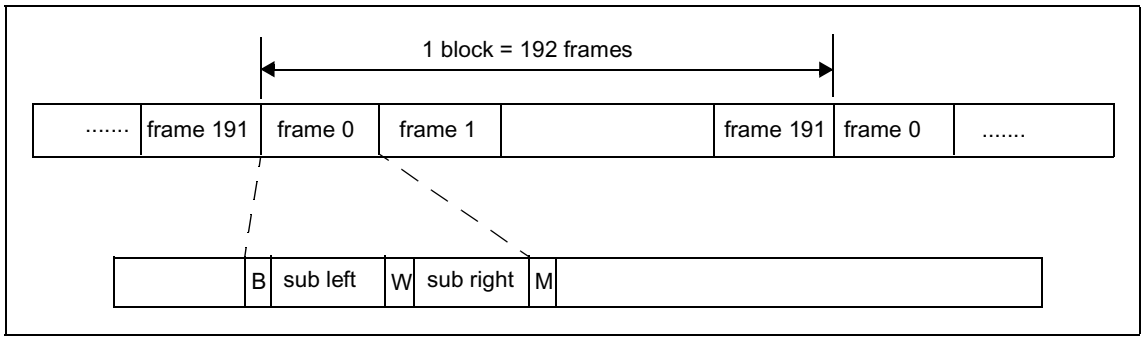
Figure 159: S/PDIF player block diagram



39.2 Audio data mode or encoded mode

39.2.1 Audio data mode

Figure 160: PCM mode frame format



PCM null data

The PCM null data is sent on the S/PDIF output when:

- in S/PDIF audio data mode (SPDIF_CTRL[2:0] = 011) and an underflow is detected (the ACK signal not asserted after a request for data), or
- in S/PDIF MUTE with PCM null data mode (SPDIF_CTRL[2:0] = 001).

The PCM null data is generated by hardware, and the VUC data is read in burst registers (refer to Section 40.1.1: *Burst registers* on page 397).

Switching between modes

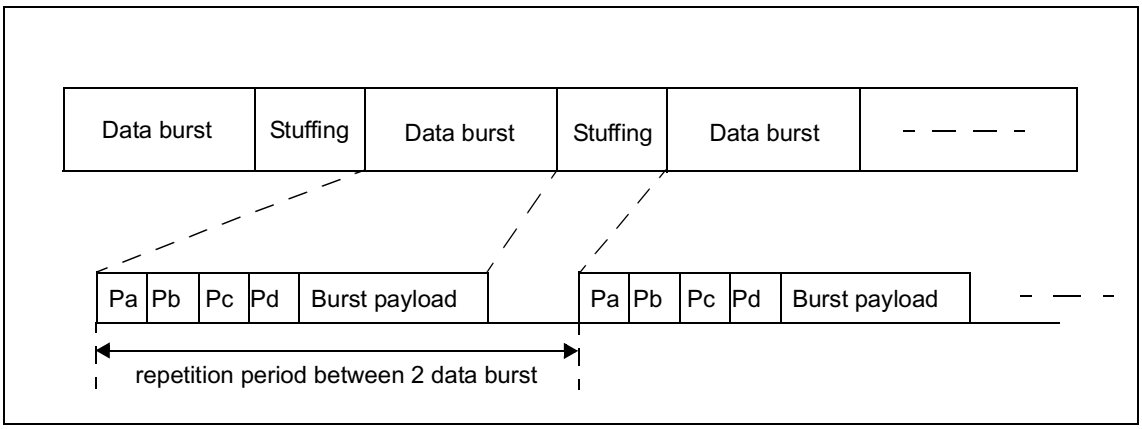
When an underflow occurs before the end of a block, PCM null data is sent until the end of the S/PDIF block is reached. When the end of the block is reached, a check is made to see if data has been received.

- If data has been received: PCM data is sent to the S/PDIF output.
- If data has not been received (still in underflow active): another complete block (192 frames) of null data is sent to the S/PDIF output.
- When S/PDIF MUTE mode is selected before the end of a block, the switch to mute mode is only effective (PCM null data send on S/PDIF output) when the end of a frame is reached. Wait until the end of the block to switch back to S/PDIF audio data mode.
- The switch to S/PDIF off mode is only effective when the end of a block is reached.

39.2.2 Encoded mode

Data burst

Figure 161: Data burst format



The repetition period between two data bursts is defined in SPDIF_BURST_LEN[31:16] register.

Stuffing

The stuffing can be made by software or hardware. The selection is made by bit 14 of the SPDIF_CTRL register.

Soft stuffing

The software enabled stuffing data is received by the S/PDIF player through the data input port.

The data received is as following:

3132	....	16	15	....	4	3	0
0	....	0	0	....	0	VUC	0
0	....	0	0	....	0	VUC	0
0	....	0	0	....	0	VUC	0
0	....	0	0	....	0	VUC	0

Pause data burst

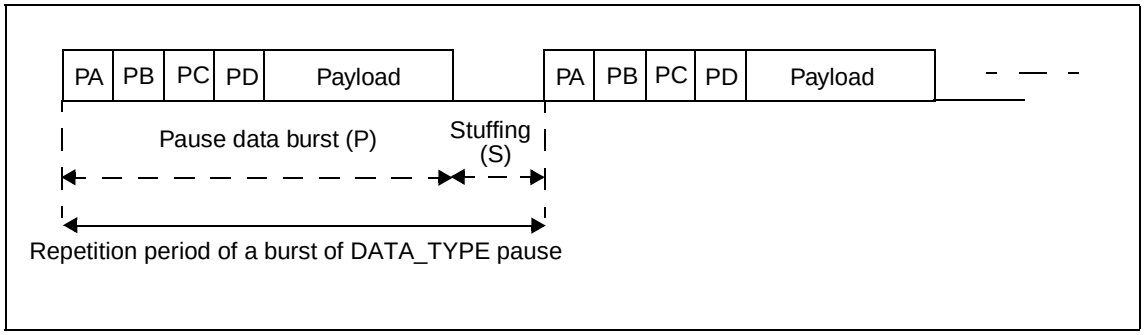
The pause data burst is sent to the S/PDIF output if:

- in S/PDIF encoded mode (SPDIF_CTRL[2:0] = 100), an underflow is detected (the ACK signal not asserted) and an end of data burst is reached,
- in S/PDIF MUTE with pause burst mode (SPDIF_CTRL[2:0] = 010).

The pause data-burst is generated by the hardware.

For generating the pause burst, the PA, PB, PC, PD and the VUC information is read in the burst registers (refer to Section 40.1.1: *Burst registers* on page 397).

Figure 162: Pause burst format



The repetition period between two pause bursts is defined in the SPDIF_BURST_LEN[15:0] register.

The number of pause burst (P + S) repetition periods are defined in the SPDIF_PAUSE_LAT[31:16] register.

- If set to 1, the hardware sends one P+S on the S/PDIF output, before looking if the operation mode has changed, or if data has been received by the S/PDIF player (underflow).
- If set to 3, the hardware sends three P+S on the S/PDIF output, before looking if the operation mode has changed or if data have been received by the S/PDIF player (underflow).

The channel status bit number of the first PA sent on the S/PDIF output is stored in the SPDIF_STATUS register.

- If SPDIF_PAUSE_LAT[31:16] register = 1 each time PA is sent to the S/PDIF output, the PA_C_BIT_NUMBER field of the SPDIF_STATUS register is updated with that channel's status bit number.
- If the SPDIF_PAUSE_LAT[31:16] register = 3 when the first PA, the fourth, the seventh PA (and so on) are sent to the S/PDIF output, the PA_C_BIT_NUMBER field of the SPDIF_STATUS register is updated with the channel status bit number.

The application uses this information to resynchronize the next data to be sent to the S/PDIF player.

Switch between modes

- When an underflow occurs before the end of a data burst, null data is sent until the end of the data burst is reached. When the end of the data burst is reached, a check is made to see if data has been received.
  - If data has been received a data burst is sent on the S/PDIF output.
  - If data has not been received (still in underflow active) a pause data burst is sent on the S/PDIF output.
- When S/PDIF MUTE mode is selected before the end of a data burst, the switch to mute mode is only effective (pause data burst sent on the S/PDIF output) when the end of the data burst is reached. Wait until the end of the pause data burst to switch back to S/PDIF enhanced mode.
- The switch to S/PDIF off mode is only effective when the end of the data burt or pause data burst is reached.

39.2.3 Byte swap

Depending on the endianness of the received data, a byte swap may be required. By setting bit 13 of the SPDIF_CTRL register to 1, a byte swap is applied in encoded mode between bits [31:24] and bits [23:16]. No byte swap is performed on bits [15:0].

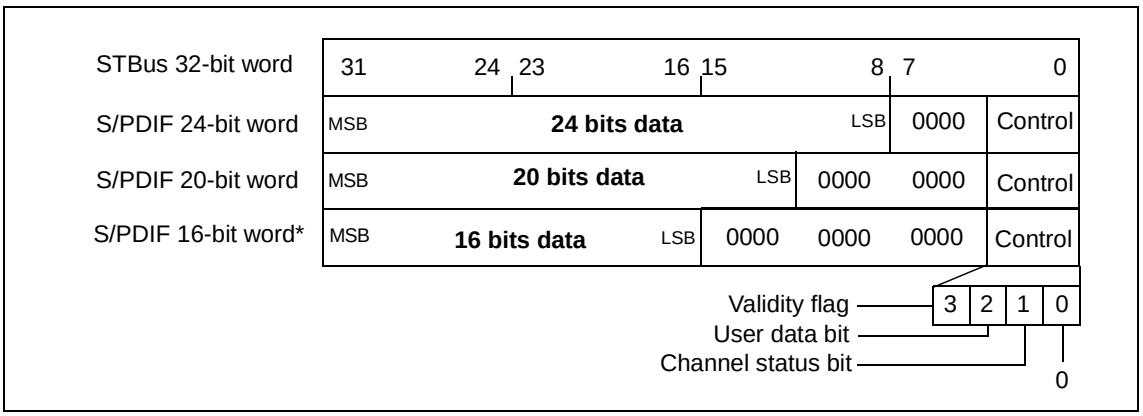
Byte swap is only performed on the data received on the data input bus (PA, PB, PC, PD and burst payload). When playing pause bursts, PA, PB, PC and PD are read in the SPDIF_PA_PB and SPDIF_PC_PD burst registers and no byte swap is performed on PA, PB, PC or PD.This means that these four 16-bit words need to be stored in their associated burst registers with the corrected endianness.

39.3 Data

The S/PDIF data is composed of audio samples or encoded data plus a set of S/PDIF control words (user data, channel status and validity flag).

Data is located in external memory and should be aligned and formatted as shown in Figure 163.

Figure 163: STBus data



Note: This is the only format for S/PDIF encoded mode.

39.4 Interrupts

There are five registers for interrupt management:

- SPDIF_INT_STATUS, read-only interrupt status register,
- SPDIF_INT_STATUS_CLR, clear interrupt status,
- SPDIF_INT_EN, read-only interrupt enable register,
- SPDIF_INT_EN_SET, set interrupt enable,
- SPDIF_INT_EN_CLR, clear interrupt enable.

Table 98 gives the mapping of the SPDIF_INT_STATUS bits.

Table 98: S/PDIF player block interrupt mapping

SPDIF_INT_STATUS	Interrupt source
SPDIF_INT_STATUS[0]	Underflow
SPDIF_INT_STATUS[1]	End of data burst
SPDIF_INT_STATUS[2]	End of block
SPDIF_INT_STATUS[3]	End of latency
SPDIF_INT_STATUS[4]	End of PD data burst
SPDIF_INT_STATUS[5]	Memory block fully read
SPDIF_INT_STATUS[6]	End of PD pause burst

Each time an interrupt occurs, the relevant SPDIF_INT_STATUS bit is set to 1. If the corresponding bit of the SPDIF_INT_EN_SET register is also set to 1 the interrupt request output signal is asserted.

When the system has received an interrupt request, the reading of the SPDIF_INT_STATUS register shows which source has generated the interrupt.

Once the interrupt has been taken into account, software can set the corresponding bit of the SPDIF_INT_STATUS_CLR register to 1 to de-assert the interrupt request output signal and to clear the corresponding bit of the SPDIF_INT_STATUS register.

39.5 Soft reset

When software sets the SPDIF_SOFTRESET register to 1, the GPFIFO completes the transaction in progress and any data inside the FIFO after the soft reset must be considered invalid. After the soft reset the SPDIF_SOFTRESET register is reset to 0.

40 S/PDIF player and GPFIFO registers

Addresses are provided as the *S/PDIFBaseAddress* + offset.

The *S/PDIFBaseAddress* is:

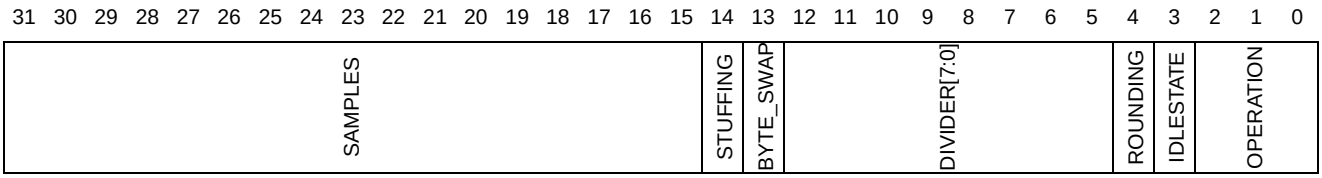
0x2060 0000.

Table 99: S/PDIF player and GPFIFO registers

Register name	Description	Address offset	Type
S/PDIF player registers			
SPDIF_CTRL	Control register, see page 395	0x001C	R/W
SPDIF_STATUS	Status register, see page 396	0x0020	RO
SPDIF_PA_PB	Sync words, see page 397	0x0024	R/W
SPDIF_PC_PD	Encoded control word and data length, see page 397	0x0028	R/W
SPDIF_CL1	Channel status (left) 1, see page 397	0x002C	R/W
SPDIF_CR1	Channel status (right) 1, see page 398	0x0030	R/W
SPDIF_CL2_CR2_U_V	Channel status 2, user values and validity, see page 398	0x0034	R/W
SPDIF_PAUSE_LAT	Pause latency, see page 398	0x0038	R/W
SPDIF_BURST_LEN	Length of data bursts and pauses, see page 399	0x003C	R/W
GPFIFO registers			
SPDIF_SOFTRESET	Soft reset, see page 399	0x0000	R/W
SPDIF_FIFO_DATA	GPFIFO data, see page 399	0x0004	R/W
SPDIF_INT_STATUS	Interrupt status, see page 400	0x0008	RO
SPDIF_INT_STATUS_CLR	Clear interrupt status, see page 400	0x000C	WO
SPDIF_INT_EN	Enable interrupt, see page 401	0x0010	RO
SPDIF_INT_EN_SET	Set enable interrupt register, see page 401	0x0014	WO
SPDIF_INT_EN_CLEAR	Clear enable interrupt register, see page 402	0x0018	WO

40.1 S/PDIF player registers

SPDIF_CTRL Control register



Address: *S/PDIFBaseAddress* + 0x001C

Type: Read/write

Reset: 0

Description: The control bits are resynchronized inside the S/PDIF player with the S/PDIF player clock.

[31:15] **SAMPLES**: Each time the number of samples read by the S/PDIF player is equal to this number, the associated bit in the status register is set to 1

[14] **STUFFING**
0: stuffing made by software in encoded mode 1: stuffing made by hardware in encoded mode

[13] **BYTE_SWAP**
0:No byte swap in encoded mode
1:Byte swap between bits [31:24] and bits [23:16] in encoded mode

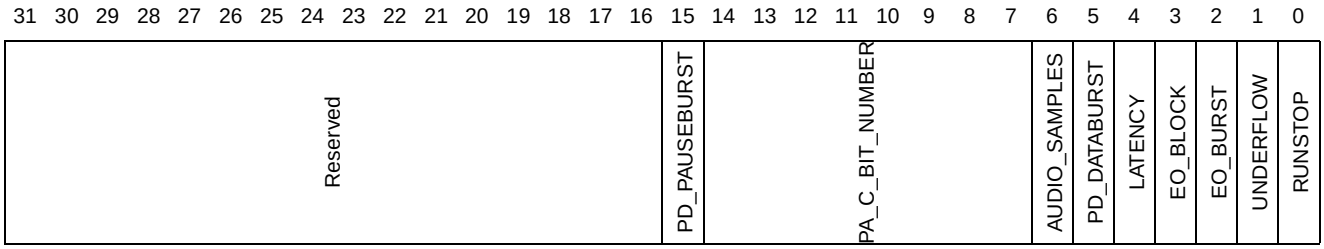
[12:5] **DIVIDER[7:0]**: audio clock divider

[4] **ROUNDING**
0: No rounding 1: 16 bit rounding on PCM audio data output

[3] **IDLESTATE:** Idle state of the S/PDIF output line

[2:0] **OPERATION**
000: S/PDIF OFF 001: S/PDIF MUTE with PCM null data
010: S/PDIF MUTE with pause burst 011: S/PDIF audio data mode
100: S/PDIF encoded mode

SPDIF_STATUS Status register



Address: *S/PDIFBaseAddress* + 0x0020

Type: Read only

Reset: 0

Description: The status bits are changed on the rising edge of the S/PDIF player clock whenever a change occurs. This register must be re-synchronized inside the STBus interface with the STBus clock. These bits should be connected to an interrupt register associated to an interrupt mask to be able to wake up the system if required.

[31:16] **Reserved**

[15] **PD_PAUSEBURST**: Interrupt sent each time PD word is sent on S/PDIF output while outputting a pause burst

[14:7] **PA_C_BIT_NUMBER:** PA channel status bit number when pause data burst sent

[6] **AUDIO_SAMPLES**
1: the number of samples to read defined in the CONTROL_REG[31:15] have been received by the S/PDIF player (that is, part of the memory can be updated with new samples)

[5] **PD_DATABURST:** interrupt sent each time PD word sent on S/PDIF output while outputting a data burst

[4] **LATENCY:** end of subframe counter for latency

[3] **EO_BLOCK:** end of block

[2] **EO_BURST:** end of data burst

[1] **UNDERFLOW:** data underflow

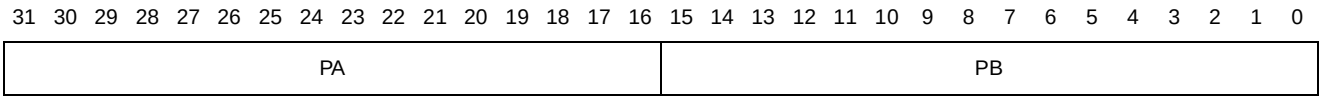
[0] **RUNSTOP**
0: S/PDIF stopped 1: S/PDIF running

*Note: Bits 1, 2, 3, 4, 5, 6 and 15 are pulse signals (set to 1 during one clock cycle). Data underflow is detected when operation = 011 or 100 (that is S/PDIF audio data mode is on, or S/PDIF encoded mode is on), and when no further data is received (*ACK *input signal is not asserted after a request for data).*

40.1.1 Burst registers

The control bits are re-synchronized inside the S/PDIF player with the S/PDIF player clock.

SPDIF_PA_PB Sync words

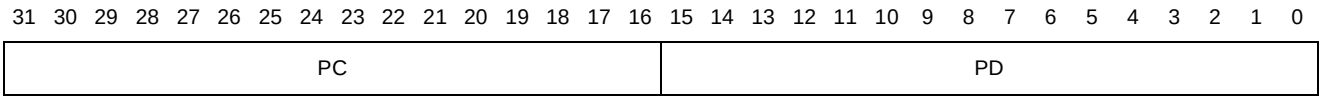


Address: *S/PDIFBaseAddress* + 0x0024
Type: Read/write
Reset: 0
Description: This register is used in encoded mode only and is a function of the encoded algorithm.

[31:16] **PA**: sync word 1

[15:0] **PB**: sync word 2

SPDIF_PC_PD Encoded control word and data length

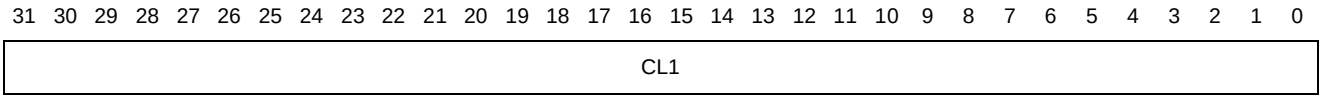


Address: *S/PDIFBaseAddress* + 0x0028
Type: Read/write
Reset: 0
Description: This register is used in encoded mode only and is a function of the encoded algorithm.

[31:16] **PC**: encoded control word in frame unit

[15:0] **PD**: encoded data length in frame unit

SPDIF_CL1 Channel status (left) 1



Address: *S/PDIFBaseAddress* + 0x002C
Type: Read/write
Reset: 0
Description:

[31:0] **CL1**: channel status for left subframes: bits 31 down to 0

SPDIF_CR1 Channel status (right) 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
CR1																															

Address: *S/PDIFBaseAddress* + 0x0030
Type: Read/write
Reset: 0
Description:

[31:0] **CR1**: channel status for right subframes: bits 31 down to 0

SPDIF_CL2_CR2_U_V Channel status, user values and validity‘

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												VALIDITY_RIGHT	VALIDITY_LEFT	USER_RIGHT	USER_LEFT	Reserved				CR2				Reserved				CL2				

Address: *S/PDIFBaseAddress* + 0x0034
Type: Read/write
Reset: 0
Description:

[31:20] **Reserved**

[19] **VALIDITY_RIGHT**: validity for right subframes

[18] **VALIDITY_LEFT**: validity for left subframes

[17] **USER_RIGHT**: user value for right subframes

[16] **USER_LEFT**: user value for left subframes

[15:12] **Reserved**

[11:8] **CR2**: channel status for right subframes: bits 35 down to 32

[7:4] **Reserved**

[3:0] **CL2:** channel status for left subframes: bits 35 down to 32

SPDIF_PAUSE_LAT Pause latency

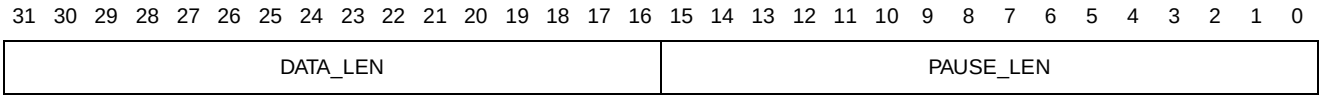
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GAP_LEN																SUBFRAME_MAX															

Address: *S/PDIFBaseAddress* + 0x0038
Type: Read/write
Reset: 0
Description: This register is used in encoded mode only and is a function of the encoded algorithm.

[31:16] **GAP_LEN**: gap length in a pause data burst (in frame unit)

[15:0] **SUBFRAME_MAX**: subframe counter max for latency

SPDIF_BURST_LEN Length of data bursts and pauses



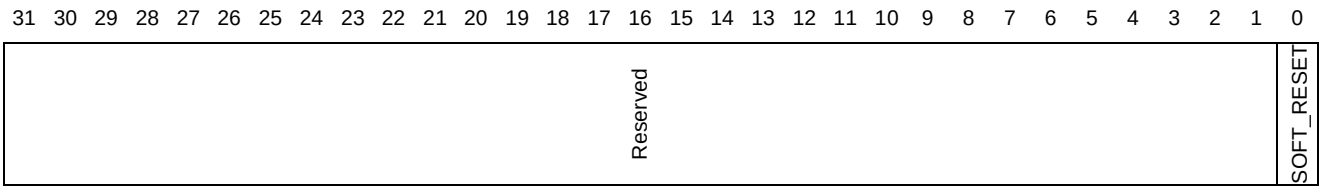
Address: *S/PDIFBaseAddress* + 0x003C
Type: Read/write
Reset: 0
Description: This register is used in encoded mode only and is a function of the encoded algorithm.

[31:16] **DATA_LEN**: total length of data burst in frame units

[15:0] **PAUSE_LEN**: total length of pause data burst in frame units

40.2 GPFIFO registers

SPDIF_SOFTRESET Soft reset

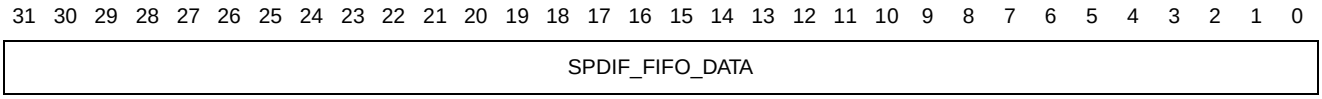


Address: *S/PDIFBaseAddress* + 0x0000
Type: Read/write
Reset: 0
Description:

[31:1] **Reserved**

[0] **SOFT_RESET**: when 1 a soft reset signal is generated to complete safely any transaction in progress and reset all the FSMs and the FIFO pointers. Software must reset the register to 0.

SPDIF_FIFO_DATA GPFIFO data



Address: *S/PDIFBaseAddress* + 0x0004
Type: Read/write
Reset: 0
Description:

[31:0] **SPDIF_FIFO_DATA**: data contained in the first location of the FIFO.

SPDIF_INT_STATUS 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																									SPDIF_INT_STATUS[6:0]						

Address: *S/PDIFBaseAddress* + 0x0008

Type: Read only

Reset: 0

Description: This is a read-only register containing information about the interrupts generated by the possible sources. Each bit is associated to an interrupt source. If the bit is set to 1 the source has generated an interrupt. If the bit is set to 0 this means no interrupt has been generated.
The event making a bit of this register to be set to 1 is an interrupt strobe generated by one of the possible interrupt sources of the application.

[31:7] **Reserved**

[6:0] **SPDIF_INT_STATUS[6:0]**

Each bit is associated to a possible interrupt source. If bit n is set to 1, the nth interrupt source has generated an interrupt.
The bits are mapped as follows:

Bit 0: Underflow
Bit 1: End of data burst
Bit 2: End of block
Bit 3: End of latency
Bit 4: End of PD data burst
Bit 5: Memory block fully read
Bit 6: End of PD pause burst

SPDIF_INT_STATUS_CLR Clear 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																									SPDIF_INT_CLR[6:0]						

Address: *S/PDIFBaseAddress* + 0x000C

Type: Write only

Reset: 0

Description: This register shows when an interrupt has completed. Each bit is associated to an interrupt source. If the bit is set to 1 this means the relative interrupt has completed, and the corresponding bit of the SPDIF_INT_STATUS register has to be reset to 0.

[31:7] **Reserved**

[6:0] **SPDIF_INT_STATUS_CLR[6:0]**

Each bit is associated to a possible interrupt source. If bit n is set to 1, this means the nth interrupt has been serviced and the correspondent bit of the SPDIF_INT_STATUS register has to be reset to 0.

SPDIF_INT_EN Enable 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
Reserved																									SPDIF_INT_EN[6:0]						

Address: *S/PDIFBaseAddress* + 0x0010

Type: Read only

Reset: 0

Description: This register's bits are set and reset by accessing the SPDIF_INT_EN_SET and SPDIF_INT_EN_CLR registers. Its content tells which interrupt sources are enabled and disabled.
The GPFIFO can assert an interrupt only if the relative SPDIF_INT_STATUS register bit ANDed with the SPDIF_INT_STATUS_EN register bit is nonzero.

[31:7] **Reserved**

[6:0] **SPDIF_INT_EN[6:0]**

Each bit is associated to a possible interrupt source. If bit n is set to 1, this means the nth interrupt source is enabled to generate interrupts.

SPDIF_INT_EN_SET Set enable interrupt register

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

Address: *S/PDIFBaseAddress* + 0x0014

Type: Write only

Reset: 0

Description: This register allows bits of the SPDIF_INT_EN register to be set. There is a one-to-one correspondence between the bits of this register and the bits of the SPDIF_INT_EN register. Each bit of the SPDIF_INT_EN_SET register set to 1 forces the corresponding bit of the SPDIF_INT_EN register to be set to 1. Each bit set to 0 has no effect.

[31:7] **Reserved**

[6:0] **SPDIF_INT_EN**

Each bit is associated to a possible interrupt source. If bit n is set to 1, the nth bit of the SPDIF_INT_EN register is set to 1. If bit n is set to 0, there is no effect.

SPDIF_INT_EN_CLR Clear enable interrupt register

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

Address: *S/PDIFBaseAddress* + 0x0018

Type: Write only

Reset: 0

Description: This register allows bits of the SPDIF_INT_EN register to be reset. There is a one-to-one correspondence between the bits of this register and the bits of the SPDIF_INT_EN register. Each bit of the SPDIF_INT_EN_CLR register set to 1, forces the corresponding bit of the SPDIF_INT_EN register to be reset to 0. Each bit set to 0 has no effect.

[31:7] **Reserved**

[6:0] **SPDIF_INT_EN_CLR[6:0]**

Each bit is associated to a possible interrupt source. If bit n is set to 1, the nth bit of the SPDIF_INT_EN register is reset to 0. If bit n is set to 0, there is no effect.

41 Audio DAC

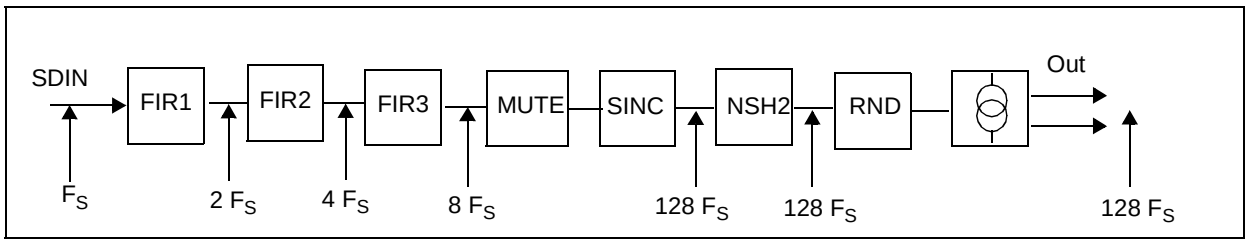
41.1 Description

This audio digital-to-analog converter (DAC) is a high performance stereo audio converter operating at 256 Fs system clock. This DAC accepts 24-bit input data in I^2S format from the PCM player macro block and converts it into a differential current 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 164: Digital flow



The input stream SDIN derived from the audio decoder, sampled at Fs, is first interpolated by two and then filtered by a 75th order FIR filter, FIR1. This signal, at 2 Fs, is interpolated by two and filtered by a 20th order FIR filter, FIR2. The signal, at 4 Fs, 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 Fs which is divided down internally to produce all other clocks.

41.2 Input signals and output pins

41.2.1 Supplies

The audio DAC utilizes two supply levels, 1.2 V and 3.3 V for different stages. Supplies are split into digital and analog for noise immunity as appropriate.

41.2.2 Output signals

Table 100: Audio DAC output signals

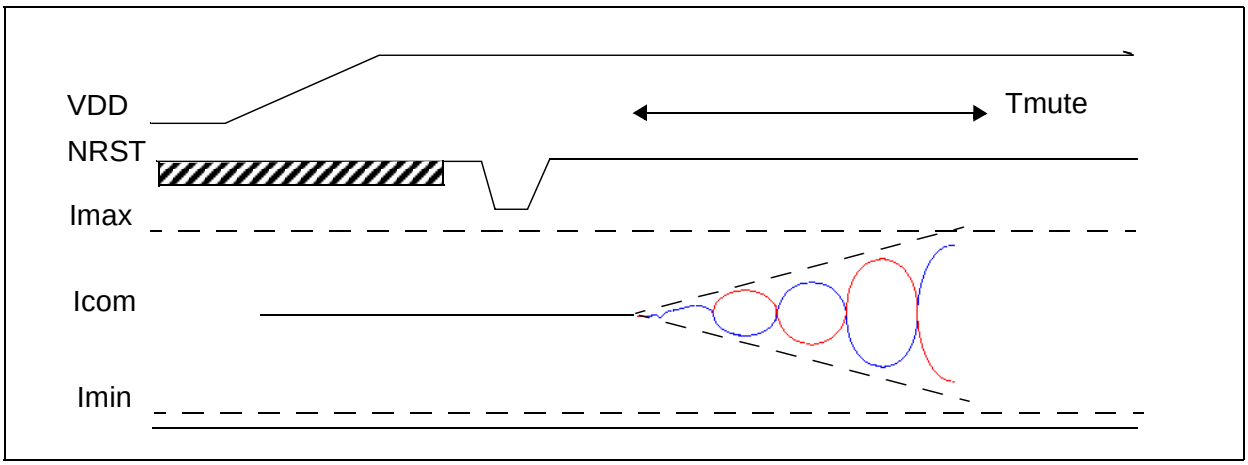
Input/output	Pin name	Description
Output	OUTRPLUS	Right differential positive analog output
	OUTRMINUS	Right differential negative analog output
	OUTLPLUS	Left differential positive analog output
	OUTLMINUS	Left differential negative analog output
	VBGOUT	Bandgap voltage output. This pin should be connected to an external 10uF capacitor (analog ground)

41.2.3 Reset

A low level on NOT_RESETIN or a reset of bit Interconnect Config Control D[NRST] puts the system in reset mode by initializing internal counters and control registers.

At reset, the audio DAC is disabled. Figure 165 shows the recommended sequence.

Figure 165: Reset timing



41.3 Soft mute

The mute function is controlled by the soft mute input. In this mode the output current (OUTRPLUS, OUTRMINUS, OUTLPLUS and OUTLMINUS) is 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. 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 each. This mute function is also controllable through Interconnect Config Control Reg D.

Figure 166: Soft mute sequence

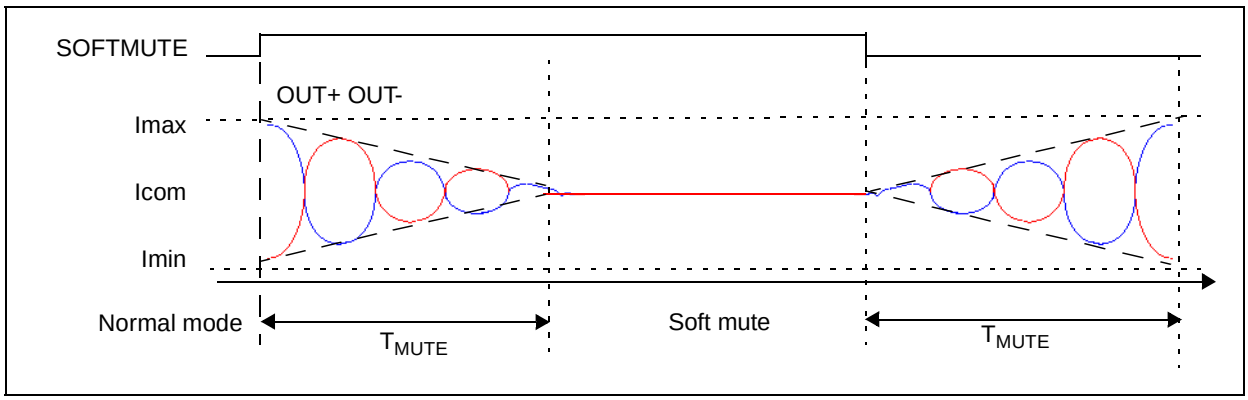


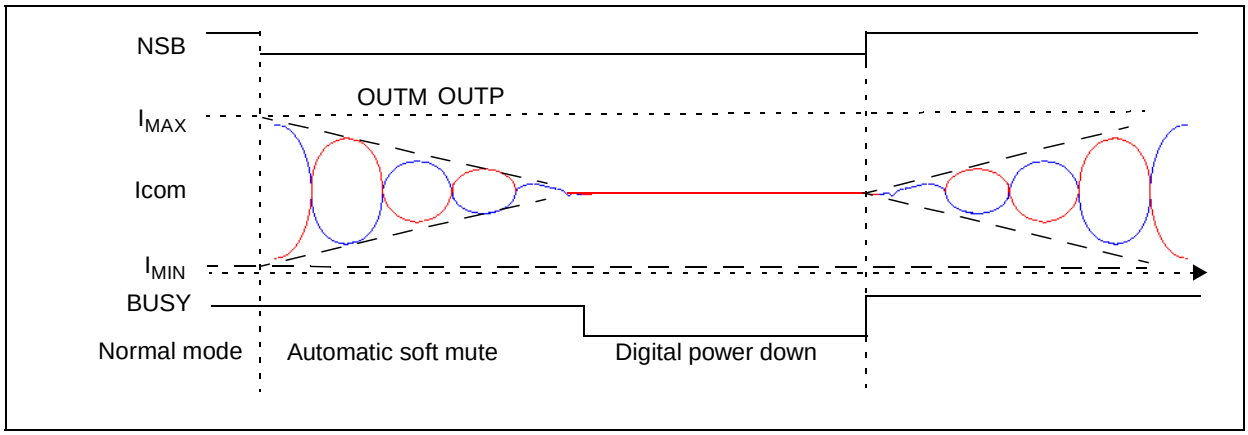
Table 101: Soft mute timings

Symbol	Parameter	Min	Typ	Max	Unit
T_{MUTE}	Total time for mute/unmute sequence				
	Normal mode		1920		T_{LR}
	Double mode		960	1024	T_{LR}

41.4 Digital and analog power down

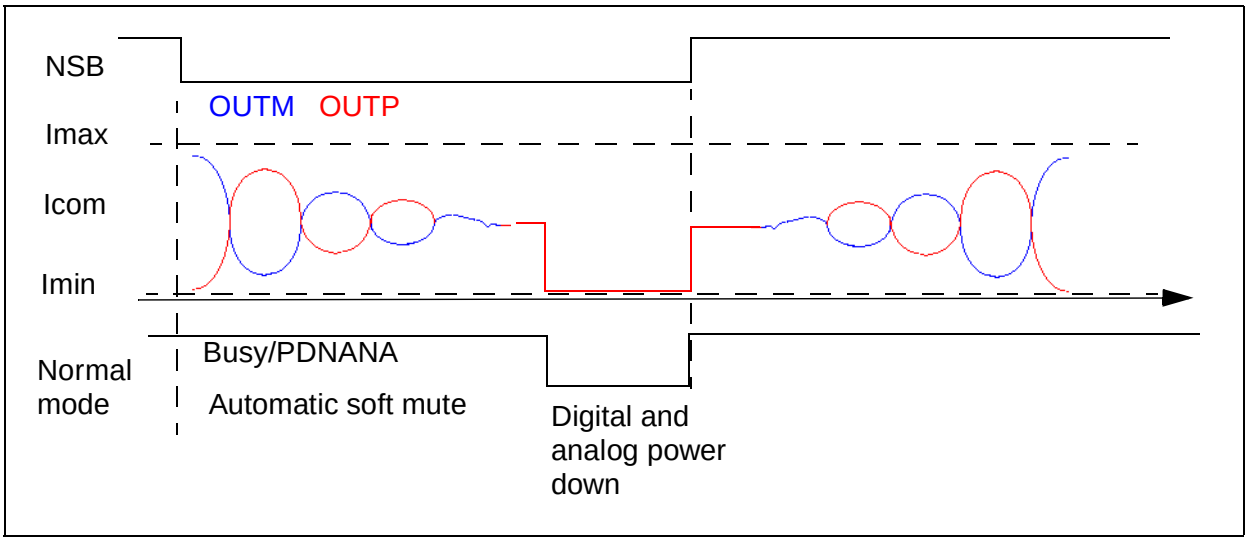
To minimize power consumption the digital and analog parts can be put into power down mode separately. In the digital power up (NSB = 1) and in power down (NSB = 0) there is an automatic soft mute.

Figure 167: Digital power-down sequence



For a digital and analog power down at the same time a pop noise can be heard with the high end application schematic. To avoid the pop noise the very high end application board schematic should be used.

Figure 168: Digital and analog power-down sequence



41.5 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 OUTLMINUS/OUTRMINUS, 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 IREFOUT of the DAC. A typical value for R_{REF} is 200 Ohm. RREF should always be higher than 175 Ohm to get a proper bandgap functionality. Also it is recommended to filter the bandgap reference through a 10uF capacitor to the analog ground.

42 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,
  - S/PDIF output.

1. A channel is paced if an external request causes a single data unit to be transferred per request. The FDMA may require multiple requests to complete the operation.

42.1 Channel structures

There are three types of channel in the STx5119 FDMA:

- standard 0D, 1D, 2D and paced memory channels,
- an SCD/PES parsing channel,
- an S/PDIF 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 2 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.

42.1.1 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.

42.1.2 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. There are no alignment restrictions for the memory side of S/PDIF transfers.

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.

42.2 FDMA timing model

42.2.1 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.

42.2.2 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 169: FDMA request routing

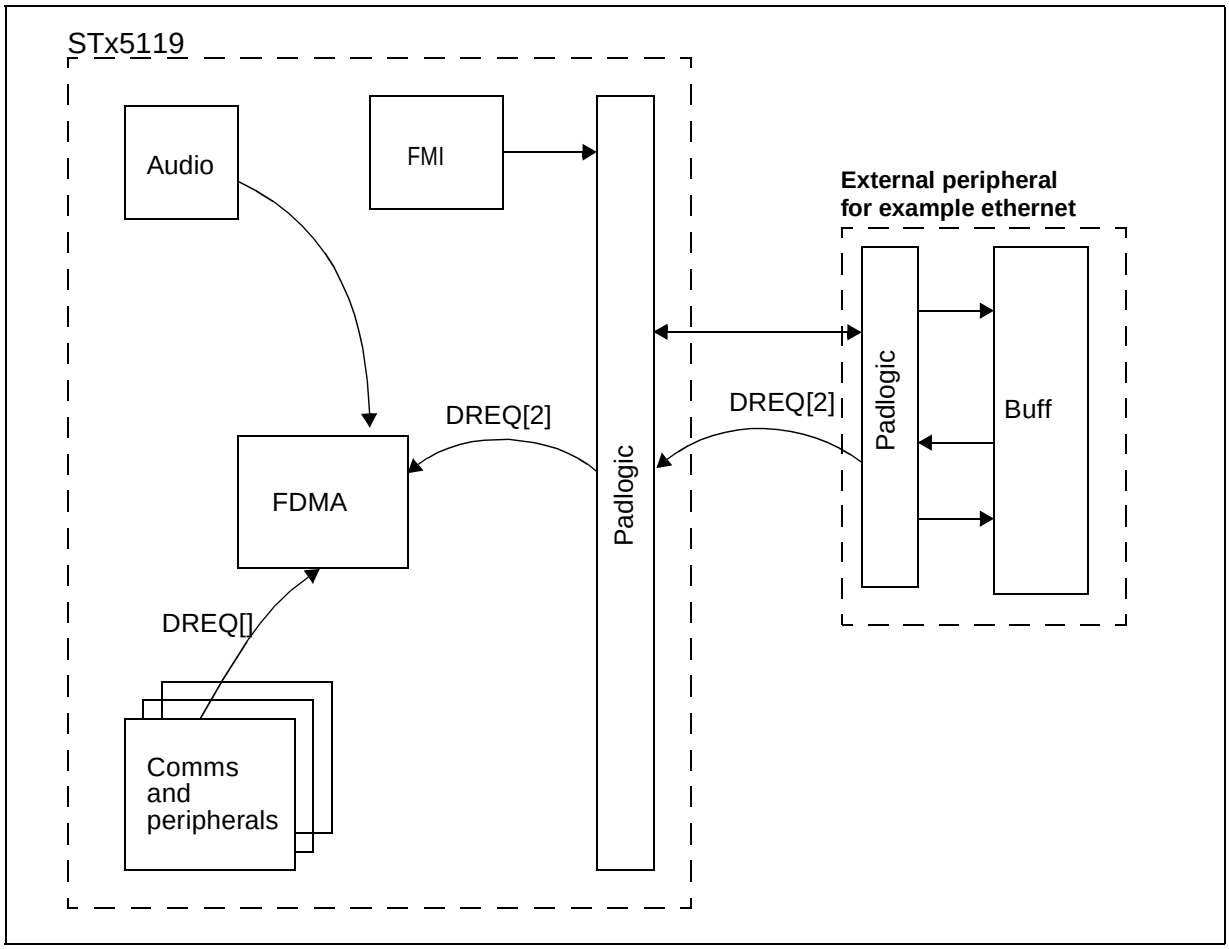


Table 102: FDMA internal handshakes

Block	Block port/signal name	FDMA requests	Description
External	DREQ[0:1]	1,2	DMA reqs from pads
Audio	AUDIO_CD_REQ	3	CD req from Decoder
	AUDIO_DECO_REQ	4	Req. to read decoded data
	AUDIO_PCMO_REQ	5	Req. to write data to PCM player
	AUDIO_SPDIF	6	Req. to write data to SPDIF player
COMMS/UART (x2)	UART[0]_RX_HALF_FULL	7	UART Rx. FIFO Half full
	UART[1]_RX_HALF_FULL	8	
	UART[0]_TX_HALF_EMPTY	9	UART Rx. FIFO Half Empty
	UART[1]_TX_HALF_EMPTY	10	
COMMS/SSC (x2)	SSC[0]_RXBUFF_HALF_FULL	11	SSC Rx. FIFO Half full
	SSC[1]_RXBUFF_HALF_FULL	12	
	SSC[0]_TXBUFF_HALF_EMPTY	13	SSC Rx. FIFO Half Empty
	SSC[1]_TXBUFF_HALF_EMPTY	14	

42.2.3 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.

42.2.4 S/PDIF channel

The S/PDIF channel is treated as a standard paced channel.

42.2.5 PCM player channel

The PCM player channel is treated as a standard paced channel.

42.2.6 Audio decoder channels

Two channels support the audio decoder. One channel is memory to paced that writes the compressed data to the audio decoder. The other channel is paced to memory that writes the decoded data to the memory.

42.3 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.

42.3.1 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.

42.3.2 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 42.4.3: *SCD/PES parsing* on page 416.

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.

42.3.3 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.

42.3.4 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

42.3.5 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.

42.3.6 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.

42.3.7 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.

42.4 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.

42.4.1 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 0D source, or NODE.CONTROL.DST_INC = CONSTANT SOURCE for a 0D destination.

Incrementing (1D)

A structure is 1D if NODE.NBYTES= NODE.LENGTH or if NODE.STRIDE = NODE.LENGTH.

For any 1D transfer the address is incremented by one byte until NODE.NBYTES bytes are transferred.

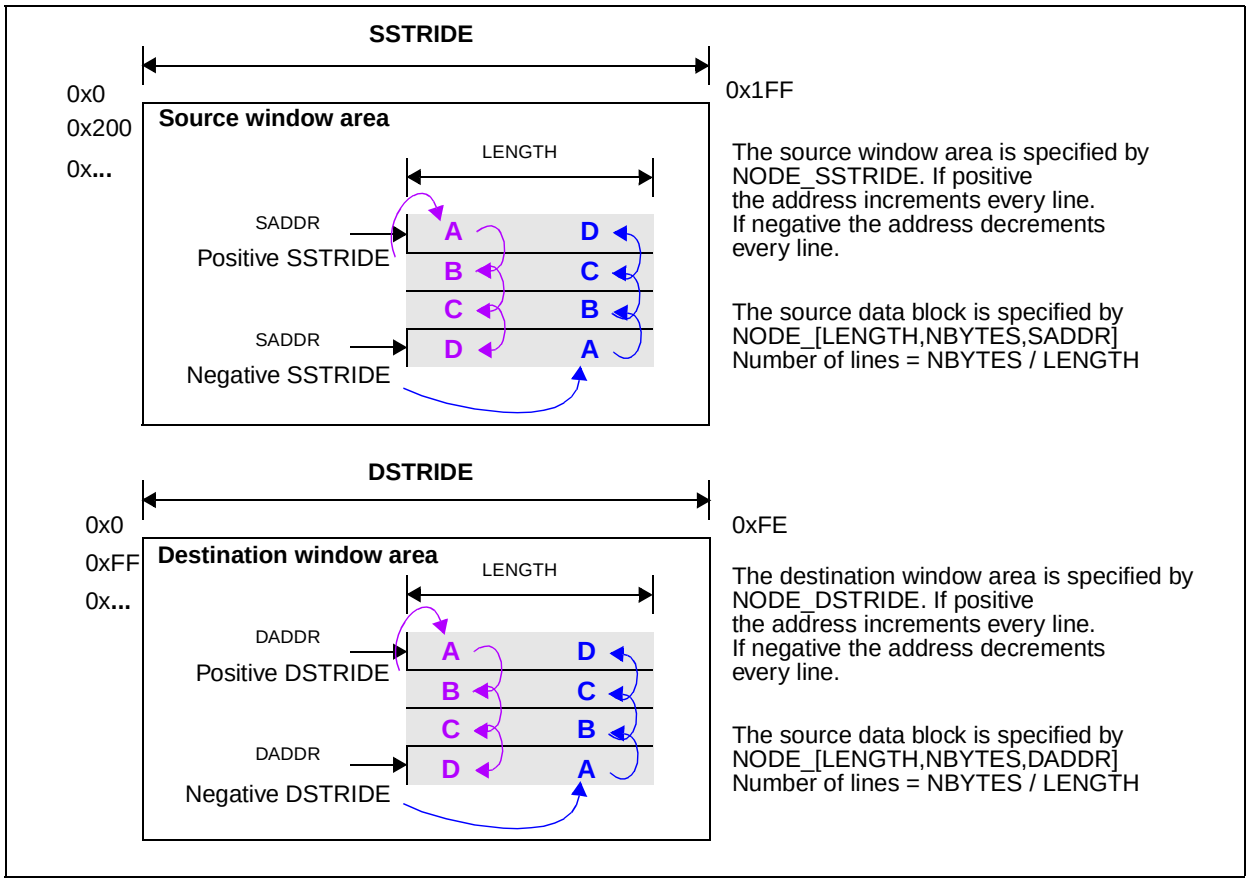
In the case of a incrementing transfer from address NODE.SADDR of NODE.NBYTES bytes, the following bytes at the following addresses are transferred:

NODE.SADDR to NODE.SADDR + (NODE.NBYTES - 1)

Rectangular array (2D)

If NODE.NBYTES ≠ NODE.LENGTH and abs(NODE.[S|D]STRIDE) ≠ NODE.LENGTH then the transfer is 2D

Figure 170: 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 a 2D to 1D transfer) the source or destination stride must be specified (depending on whether the source or destination should be 1D) to be the same as the length.

Table 103: 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 104: Definitions

Acronym	Description	Variable Name
L	Length	`Length`
SS	Source stride	`SourceStride`
DS	Destination stride	`DestinationStride`
NBytes	Number of bytes to transfer	`NumberBytes`

42.4.2 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 42.1: *Channel structures* on page 408). 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 (see Section 4.16: Hold off mechanism on page 33).

42.4.3 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.

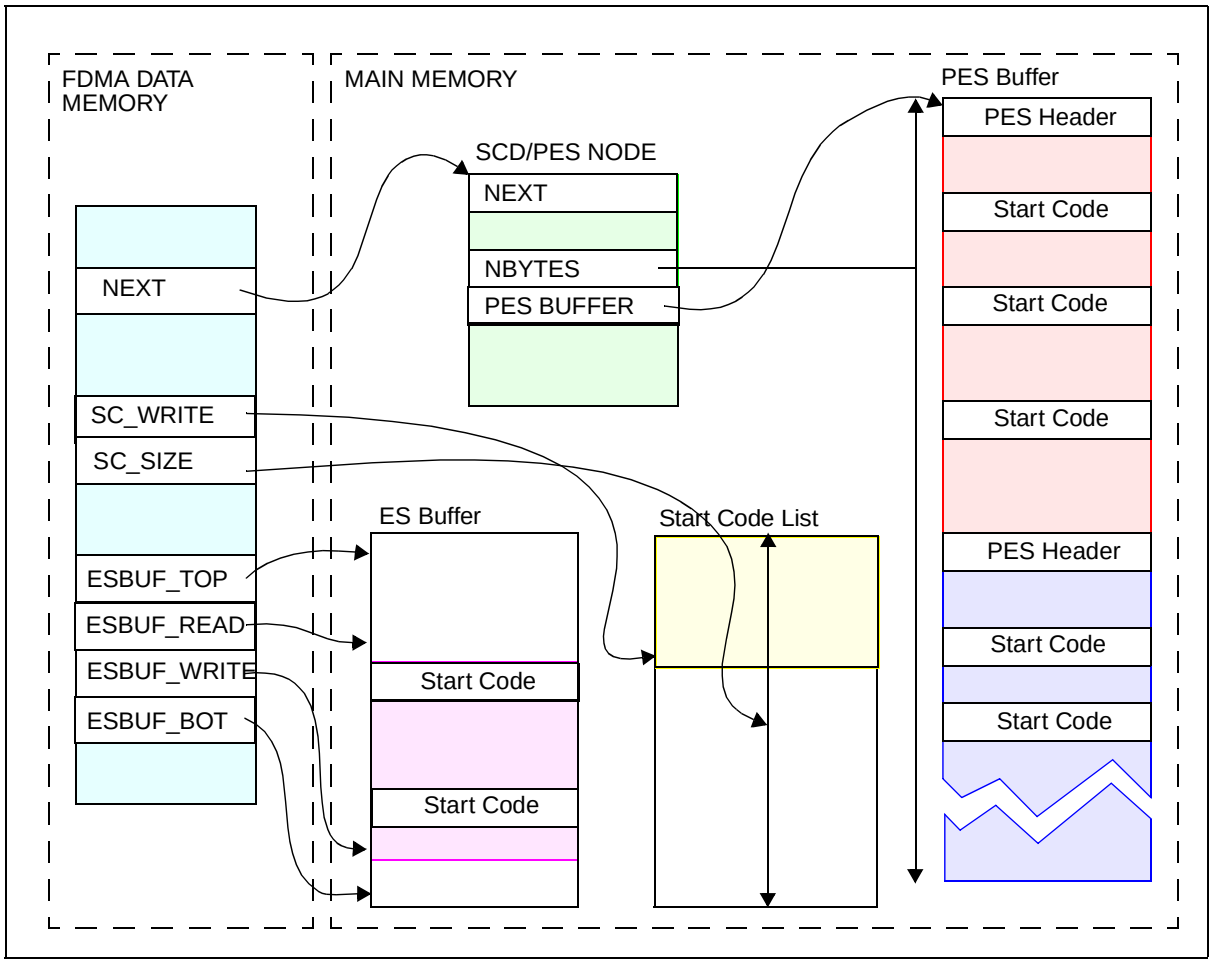
If PES start code detection is enabled (DETECT_ENABLE in PES_CONTROL = 1), the PES is parsed and the payload (ES) is written to a circular ES buffer. If PES start code detection is disabled, PES headers are not parsed and removed, but ES start code detection is carried out and the output is stored in 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 SCn_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 (a fourth is reserved for the S/PDIF channel). 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 171: SCD/PES parsing data structures

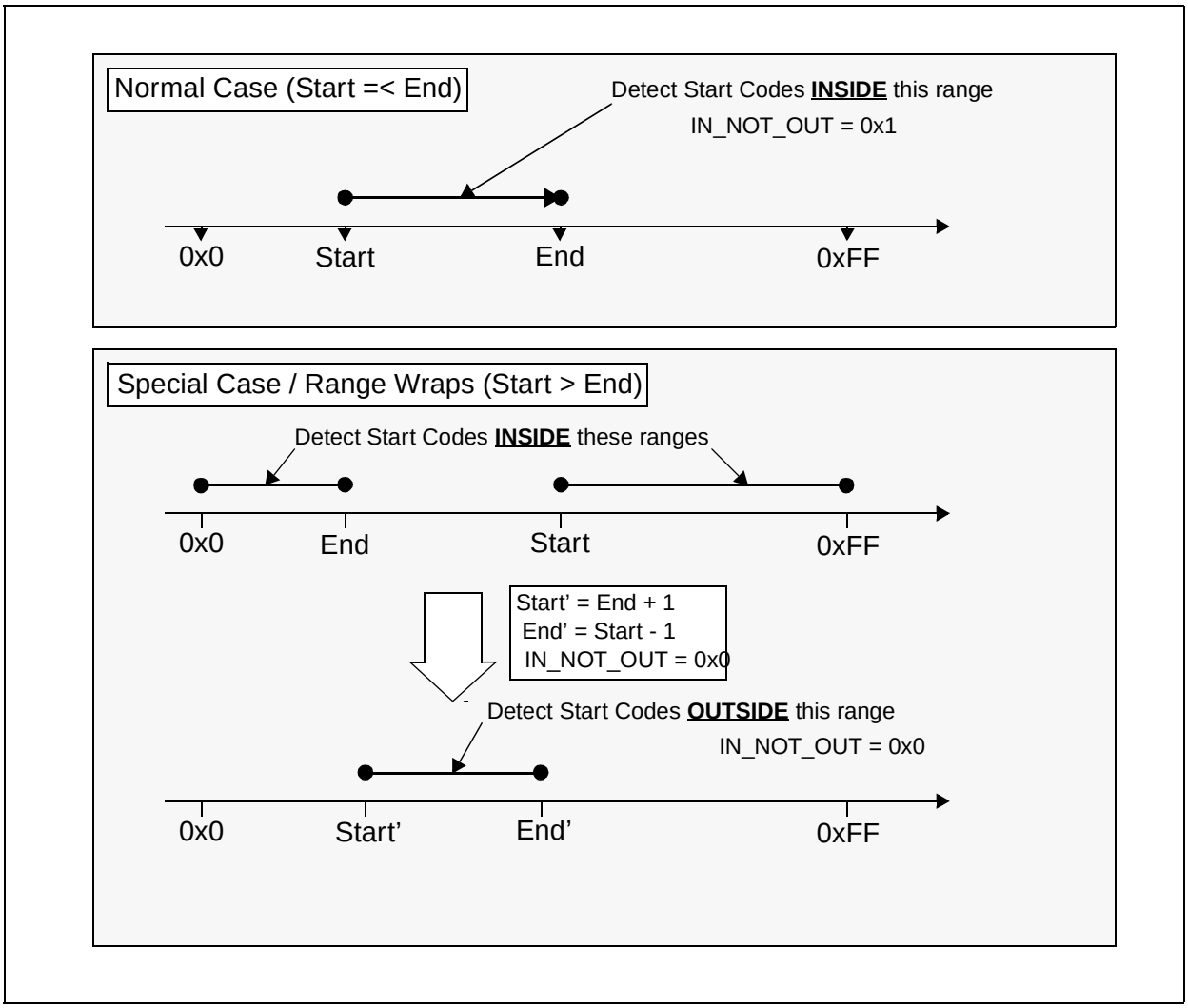


Specifying start code ranges

When specifying the start code ranges the host must ensure that RANGE_END >= RANGE_START (see PES_CONTROLn and SCn_CONTROLm). 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 172: Start code range setup



Overflow handling

There may be occasions when either the start codes list or the ES buffer overflows.

When the start codes list overflows the FDMA sets bit 0 in the SC_WRITE register for the channel and continues parsing the PES data and outputting data to the ES buffer. However, no start code is added to the list. If the incoming data is ES, it is simply copied to the ES buffer.

When the ES buffer overflows, the FDMA signals this by setting bit 0 in the ESBUF_WRITE register for the channel. The FDMA ignores the overflow condition and continues processing data thereby overwriting data already in the ES buffer.

Start codes list

The start codes list is output as a linear array of data structures. Each structure is four words in size (16 bytes) and may contain either start code data or PTS data. A label specifies whether the structure contains a start code or PTS. To know how many entries have been placed in the start codes list, the driver must read SC_WRITE and compare it with the value it had written at the start of the transfer. The difference, divided by 4 gives the number of entries in the start codes list.

42.4.4 Audio data output through S/PDIF player

The S/PDIF channel is a paced channel in which the FDMA formats data before outputting the data to the S/PDIF player. S/PDIF output involves PTI, ST20, FDMA and the S/PDIF player. The ST20 parses the PES stream coming from PTI and sets up a transfer on the S/PDIF channel.

Each node in the linked list contains the data to enable the FDMA to fetch data from a buffer, format it and output it when the S/PDIF player raises a DREQ. The node's source address points to the start of the audio frame. The node also contains the information required by the FDMA to format the incoming data. The formatted data is output to the S/PDIF player when it raises the DREQ. In order to correctly describe the data transfers for a single burst, it may be necessary to use a linked list of two or more nodes. The BURST_END bit indicates to the FDMA when the last set of data for the burst is being transferred. This allows the FDMA to automatically generate the correct stuffing to complete a burst. The node following a node in which BURST_END = 1 is considered the first node of the next burst.

Valid nodes

The S/PDIF channel nodes' control word contains a node valid bit. This bit indicates whether the node is valid. If a node is loaded by FDMA and the valid bit is not set, the FDMA completes the current burst by sending stuffing data and then return the channel to idle. The CPU is interrupted.

This mechanism only works correctly if the words are read in order (from lowest address to highest) when fetching the node from LMI. This is because the audio driver is expected to fill the node structure and validate it once the node structure has been completed.

End of burst

The BURST_END bit, part of the S/PDIF node's CONTROL parameter is used to indicate that the node describes the last node of an S/PDIF burst. The FDMA uses this to know when it should start to output stuffing. The node following a node with BURST_END = 0x1 is considered the start of a burst.

Output of stuffing

The FDMA needs to output stuffing to the S/PDIF player when the data for a burst is finished but the end of the burst hasn't been reached. The FDMA keeps a count of the outstanding frames to output for a burst. When an end of burst node is received and all the data associated with that node is transferred or when an invalid node is received, the FDMA outputs stuffing until the outstanding frames counter goes to zero.

43 Flexible DMA (FDMA) registers

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.

Register addresses are provided as the *FDMABaseAddress* + offset.

The *FDMABaseAddress* is:

0x20D0 0000.

Table 105: Flexible DMA (FDMA) registers

Register name	Description	Address offset	Type
FDMA interface			
FDMA_ID	Hardware ID, see page 423	0x0000	R/I
FDMA_VERSION	Version number, see page 423	0x0004	R/I
FDMA_ENABLE	Enable controller, see page 424	0x0008	R/I
Channel interface			
SW_ID	Revision number, see page 424	0x4000	R/I
FDMA_CMD_STAT[n]	Command and status for channel n, see page 425 (where n = 0 to 15)	0x4030 + n x 4	RO R/W
FDMA_PTR[n]	Pointer to next node for channel n, see page 426 (where n = 0 to 15)	0x4070 + n x 64	RO
FDMA_COUNT[n]	Byte count, see page 426 (where n = 0 to 15)	0x4078 + n x 64	RO
FDMA_REQ_CONTROL[n]	Request control, see page 427	0x4470 + n x 30	WO
Command mailbox			
FDMA_CMD_STAT	Bit 2 x n, 2 x (n+1) asserted indicates pending command for channel n (where n = 0 to 15), see page 428	0x5FC0	RO
FDMA_CMD_SET	When set, bits 2 x n, 2 x (n+1) flag command for channel n (where n = 0 to 15) by setting mailbox status bits 2 x n, 2 x (n+1), see page 428	0x5FC4	WO
FDMA_CMD_CLR	When set, bits 2 x n, 2 x (n+1) acknowledge command for channel n (where n = 0 to 15) by clearing mailbox status bits 2 x n and 2 x (n+1), see page 428	0x5FC8	I
FDMA_CMD_MASK	Enable flag generation for channel n (where n = 0 to 15), see page 429	0x5FCC	WO
Interrupt mailbox			
FDMA_INT_STAT	Bit 2 x n, 2 x (n+1) asserted indicates interrupt for channel n (where n = 0 to 15), see page 429	0x5FD0	RO
FDMA_INT_SET	When set, bits 2 x n, 2 x (n+1) generate an interrupt to the host for channel n (where n = 0 to 15) by setting mailbox status bits 2 x n, 2 x (n+1), see page 430	0x5FD4	I
FDMA_INT_CLR	When set, bits 2 x n, 2 x (n+1) acknowledge interrupt for channel n (where n = 0 to 15) by clearing mailbox status bits 2 x n and 2 x (n+1), see page 430	0x5FD8	WO

Table 105: Flexible DMA (FDMA) registers

Register name	Description	Address offset	Type
FDMA_INT_MASK	Enable interrupt generation for channel n (where n = 0 to 15), see page 430	0x5FDC	WO
Memory-to-memory moves and paced transfer registers			
NODE_NEXT	Pointer to start of next node, see page 431 If 0x0, this is the last node in the list	0x0000[1]	R/W
NODE_CONTROL	Channel control, see page 431	0x0004[1]	R/W
NODE_NBYTES	Number of bytes to transfer, see page 431	0x0008[1]	R/W
NODE_SADDR	Channel source address, see page 432	0x000C[1]	R/W
NODE_DADDR	Channel destination address, see page 432	0x0010[1]	R/W
NODE_LENGTH	2D line length, see page 432	0x0014[1]	R/W
NODE_SSTRIDE	2D source stride, see page 432	0x0018[1]	R/W
NODE_DSTRIDE	2D destination stride, see page 433	0x001C[1]	R/W
S/PDIF registers			
NODE_NEXT	Pointer to next node, see page 433 If 0x0, this is the last node in the list	0x0000[1]	R/W
NODE_CONTROL	S/PDIF node control, see page 434	0x0004[1]	R/W
NODE_NBYTES	Number of bytes to read, see page 434	0x0008[1]	R/W
NODE_SADDR	Source address, see page 435	0x000C[1]	R/W
NODE_DADDR	Destination address, see page 435	0x0010[1]	R/W
NODE_PA_PB	Preamble words a, b, see page 435	0x0014[1]	R/W
NODE_PC_PD	Preamble words c, d, see page 435	0x0018[1]	R/W
NODE_BURSTPERIOD	Burst repeat period in S/PDIF frames, see page 436	0x001C[1]	R/W
NODE_CHANNEL0_STATUS_LOW	Channel 0 status bits low, see page 436	0x0020[1]	R/W
NODE_CHANNEL0_STATUS_HI	Channel 0 status bits high, see page 436	0x0024[1]	R/W
NODE_CHANNEL1_STATUS_LOW	Channel 1status bits low, see page 436	0x0028[1]	R/W
NODE_CHANNEL1_STATUS_HI	Channel 1status bits high, see page 436	0x002C[1]	R/W
SCD/PES parsing registers			
NODE_NEXT	Pointer to start of next node, see page 437 If 0x0, this is the last node in the list	0x0000[1]	R/W
NODE_CONTROL	Node control parameter, see page 437	0x0004[1]	R/W
NODE_NBYTES	Number of bytes to read from PES buffer, see page 438	0x0008[1]	R/W
PESBUFFER	Read pointer to PES buffer, see page 438	0x000C[1]	R/W
Additional data region registers			
SC_WRITE0	Start code list write pointer for region 0, see page 438	0x44F0	R/W
SC_SIZE0	Start code list size for region 0, see page 439	0x44F4	R/W
ESBUF_TOP0	Top address of elementary stream buffer for region 0, see page 439	0x44F8	R/W

Table 105: Flexible DMA (FDMA) registers

Register name	Description	Address offset	Type
ESBUF_READ0	Elementary stream buffer read pointer for region 0, see page 439	0x44FC	R/W
ESBUF_WRITE0	Elementary stream buffer write pointer for region 0, see page 439	0x4500	R/W
ESBUF_BOT0	Elementary stream buffer bottom address for region 0, see page 440	0x4504	R/W
PES_CONTROL0	PES header start code range control for region 0, see page 440	0x4508	R/W
SC1_CONTROL0	Start code range 1 control for region 0, see page 441	0x450C	R/W
SC2_CONTROL0	Start code range 2 control for region 0, see page 441	0x4510	R/W
SCD_STATE0	Start code detector state for region 0, see page 441	0x4514 to 0x452F	R/W
SC_WRITE1	Start code list write pointer for region 1, see page 438	0x4530	R/W
SC_SIZE1	Start code list size for region 1, see page 439	0x4534	R/W
ESBUF_TOP1	Top address of elementary stream buffer for region 1, see page 439	0x4538	R/W
ESBUF_READ1	Elementary stream buffer read pointer for region 1, see page 439	0x453C	R/W
ESBUF_WRITE1	Elementary stream buffer write pointer for region 1, see page 439	0x4540	R/W
ESBUF_BOT1	Elementary stream buffer bottom address for region 1, see page 440	0x4544	R/W
PES_CONTROL1	PES header start code range control for region 1, see page 440	0x4548	R/W
SC1_CONTROL1	Start code range 1 control for region 1, see page 441	0x454C	R/W
SC2_CONTROL1	Start code range 2 control for region 1, see page 441	0x4550	R/W
SCD_STATE1	Start code detector state for region 1, see page 441	0x4554 to 0x456F	R/W
SC_WRITE2	Start code list write pointer for region 2, see page 438	0x4570	R/W
SC_SIZE2	Start code list size for region 2, see page 439	0x4574	R/W
ESBUF_TOP2	Top address of elementary stream buffer for region 2, see page 439	0x4578	R/W
ESBUF_READ2	Elementary stream buffer read pointer for region 2, see page 439	0x457C	R/W
ESBUF_WRITE2	Elementary stream buffer write pointer for region 2, see page 439	0x4580	R/W
ESBUF_BOT2	Elementary stream buffer bottom address for region 2, see page 440	0x4584	R/W
PES_CONTROL2	PES header start code range control for region 2, see page 440	0x4588	R/W
SC1_CONTROL2	Start code range 1 control for region 2, see page 441	0x458C	R/W
SC2_CONTROL2	Start code range 2 control for region 2, see page 441	0x4590	R/W
SCD_STATE2	Start code detector state for region 2, see page 441	0x4594 to 0x45AF	R/W
SC_WRITE3	Start code list write pointer for region 3, see page 438	0x45B0	R/W
SC_SIZE3	Start code list size for region 3, see page 439	0x45B4	R/W

Table 105: Flexible DMA (FDMA) registers

Register name	Description	Address offset	Type
ESBUF_TOP3	Top address of elementary stream buffer for region 3, see page 439	0x45B8	R/W
ESBUF_READ3	Elementary stream buffer read pointer for region 3, see page 439	0x44FC	R/W
ESBUF_WRITE3	Elementary stream buffer write pointer for region 3, see page 439	0x45C0	R/W
ESBUF_BOT3	Elementary stream buffer bottom address for region 3, see page 440	0x45C4	R/W
Start code entry registers			
SC_TYPE	Type of data in this entry, see page 442	0x0000[1]	R/W
SC_ADDRESS	Memory address of start code, see page 442	0x0004[1]	R/W
SC_VALUE	Start code value, see page 442	0x0008[1]	R/W
PTS entry registers			
PTS_TYPE	Type of data in this entry, see page 443	0x0000[1]	R/W
PTS_ADDRESS	Memory address of lowest significant byte of PTS, see page 443	0x0004[1]	R/W
PTS_UPPER	MSB of PTS value, see page 443	0x0008[1]	R/W
PTS_LOWER	LSB of PTS value, see page 443	0x000C[1]	R/W

1. Uses *MemoryOffset* as the base address

43.1 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																															

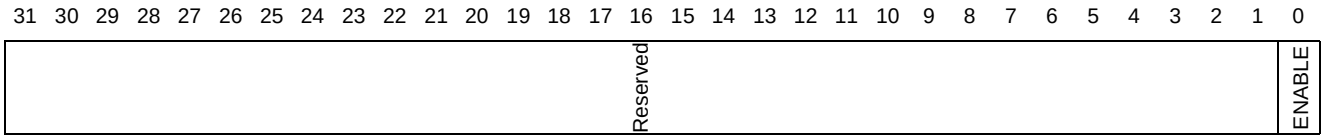
Address: *FDMABaseAddress* + 0x0000
Type: Read/write (writable only during initialization)
Reset: 0
Description: Holds hardware ID number.

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: Read/write (writable only during initialization)
Reset: 0
Description: Holds hardware version number.

FDMA_ENABLE Enable controller



Address: *FDMABaseAddress* + 0x0008

Type: Read/write (writable only during initialization)

Reset: 0

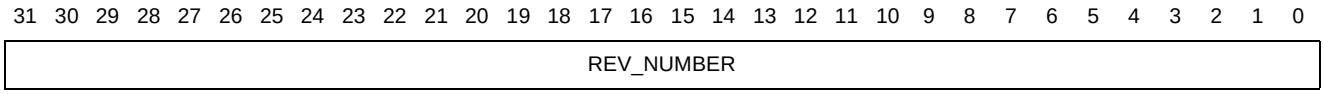
Description:

[31:1] **Reserved**

[0] **ENABLE**
1: Block enabled or running
0: Block stopped, CPU can access embedded memory.

43.2 Channel interface

SW_ID Revision number



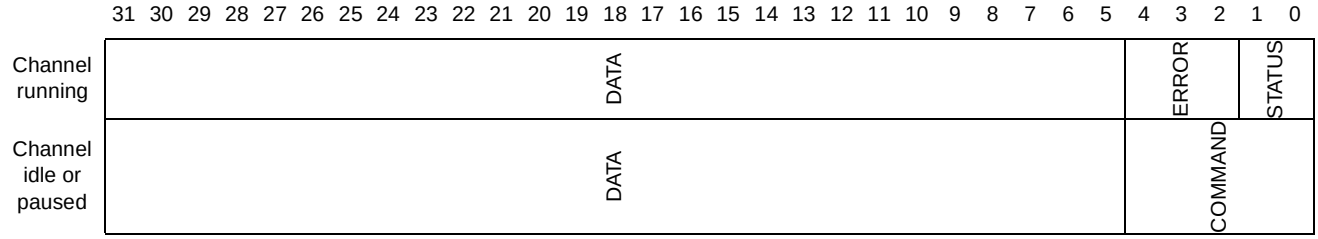
Address: *FDMABaseAddress* + 0x4000

Type: Read/write (writable only during initialization)

Reset: Undefined

Description: Holds low level revision number.

FDMA_CMD_STAT[n] Command and status for channel n



Address: *FDMABaseAddress* + 0x4030 + n x 4 (where n = 0 to 15)

Type: Read only (when channel is running)

Type: Read/write (when channel is idle or paused)

Reset:

Description: The FDMA_CMD_STAT registers are used 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 0x0 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 i is idle
10 -> channel is running
11 -> channel is paused
01 -> reserved

Channel idle or paused

[31:5] **DATA**: Pointer to node if command is START
Reserved otherwise

[4:0] **COMMAND**
00001 -> START and initialize channel n
00000 -> RESTART channel n (no initialization) from where it was paused
Others -> Reserved

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: Read only

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: Read only

Reset: Undefined

Description:

[31:0] **COUNT**: number of bytes remaining to be transferred for current node

FDMA_REQ_CONTROL[n] Request control

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

Address: *FDMABaseAddress* + 0x4470 + n x 4 (where n = 0 to 30)

Type: Write only

Reset: Undefined

Description:

[31:30] **Reserved**

[29] **INCRADDR**: increment address
0x0 - no address increment between transfers
0x1 - 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**
0x0 - software transport stream
0x1 - all other paced channels
Others - reserved

[21:15] **Reserved**

[14] **WNR:** write not read
0x0 - Read from paced peripheral
0x1 - 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] **Reserved**

43.3 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: Read only

Reset: Undefined

Description: Indicates a pending command for channel n.

[2 x n + 1:2 x n] **CHANNELn**

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: Write only

Reset: Undefined

Description: Notify FDMA there is a command for channel n

[2 x n + 1:2 x n] **CHANNELn**

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: Write only 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: Write only

Reset: Undefined

Description: Enable interrupt generation for channel n.

[2 x n + 1:2 x n] **CHANNELn**

00: Disable channel n messaging

11: Enable channel n messaging

Others: Reserved

43.4 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: Read only

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: Write only 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: Write only

Reset: Undefined

Description: When set these bits acknowledge an interrupt for channel n by clearing the relevant status bits.

[n x 2] **CHANNELn:** acknowledge message for channel n

[n x 2 + 1] **ERRORn:** 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: Write only

Reset: Undefined

Description: When set these bits enable interrupt generation for channel n.

[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

43.5 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 Parameters for the transfer

31	30	29	28	27	26	25	24	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 (see Section 43.7: *SCD/PES parsing registers* on page 437

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. For the memory-side of a paced transfer bits 0 to 4 must be 0, that is aligned to 32 bytes.

NODE_DADDR Channel destination address

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

Address: *MemoryOffset* + 0x0010

Type: R/W

Reset: Undefined

Description: The target address for the transfer. For the memory-side of a paced transfer bits 0 to 4 must be 0, that is aligned to 32 bytes.

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.

43.6 S/PDIF 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: Read/write
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 S/PDIF node control

31	30	29	28	27:13	12:8	7:5	4:0
INT_ENB	PAUSE_ENB	NODE_VALID	BURST_END	Reserved	DREQ	TYPE	REQ_MAP

Address: *MemoryOffset* + 0x0004

Type: Read/write

Reset: Undefined

Description:

[31] **INT_ENB**: interrupt at end of node
0x0 No interrupt at end of node 0x1 Interrupt host at end of node

[30] **PAUSE_ENB**: pause at end of node
0x0 Continue processing next node 0x1 Pause at end of this node

[29] **NODE_VALID**: node valid bit
0x0 Node is not valid 0x1 Node is valid

[28] **BURST_END**: end of burst
0x0 - node describes burst start or continuation
0x1 - node describes the last data to transfer for the burst. Once this data has been transferred the FDMA will start sending stuffing until the end of the burst.

[27:13] **Reserved**

[12:8] **DREQ**: DREQ mapping
0x0 and 0x1F reserved 0x1-0x1E - use REQ_CTRL 1:30

[7:5] **TYPE**: node type
0x0 SCD/PES parsing node 0x1 S/PDIF node
0x2 - 0x7 Reserved

[4:0] **REQ_MAP**: REQ_MAP/extended node type
0x1F Extended node type 0x0-0x1E Reserved
(DREQ mapping for standard node)

NODE_NBYTES Number of bytes to read

31:0
NBYTES

Address: *MemoryOffset* + 0x0008

Type: Read/write

Reset: Undefined

Description: Number of bytes to read from buffer.

NODE_SADDR Source address

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

Address: *MemoryOffset* + 0x000C
Type: Read/write
Reset: Undefined
Description: The source address from which the transfer begins. For the memory-side of a paced transfer bits 0 to 4 must be 0, that is aligned to 32 bytes.

NODE_DADDR Destination address

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

Address: *MemoryOffset* + 0x0010
Type: Read/write
Reset: Undefined
Description: The target address for the transfer. For the memory-side of a paced transfer bits 0 to 4 must be 0, that is aligned to 32 bytes.

NODE_PA_PB PA and PB preamble words

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

Address: *MemoryOffset* + 0x0014
Type: Read/write
Reset: Undefined
Description: Preamble data for S/PDIF.

NODE_PC_PD PC and PD preamble words

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

Address: *MemoryOffset* + 0x0018
Type: Read/write
Reset: Undefined
Description: Preamble data for S/PDIF.

NODE_BURSTPERIOD Burst 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
BURSTPERIOD																															

Address: *MemoryOffset* + 0x001C
Type: Read/write
Reset: Undefined
Description: Number of S/PDIF frames in the burst.

NODE_CHANNELn_STATUS_LOW Channel status LSB

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

Address: *MemoryOffset* + 0x0020 (Channel 0), 0x0028 (Channel 1)
Type: Read/write
Reset: Undefined
Description: Channel status bits 31 to 0.

NODE_CHANNELn_STATUS_HIGH channel status MSB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										VALID	USER_STATUS	CHANNEL_STATUS[35:32]			

Address: *MemoryOffset* + 0x0024 (Channel 0), 0x002C (Channel 1)
Type: Read/write
Reset: Undefined
Description:

[31:6] **Reserved**

[5] **VALID**: validity bit

[4] **USER_STATUS**: user status bit

[3:0] **CHANNEL_STATUS[35:32]**: channel status bits 35 to 32

43.7 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: Read/write
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 register

31	30	29	28	27	26	25	24	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: Read/write
Reset: Undefined
Description:

[31] **INT_ENB**: interrupt at end of node
0x0 No interrupt at end of node
0x1 Interrupt host at end of node

[30] **PAUSE_ENB**: pause at end of node
0x0 Continue processing next node
0x1 Pause at end of this node

[29:24] **Reserved**

[23:16] **TAG**: Node tag. This tag is copied to the data structures put into the start codes list.

[15:12] **Reserved**

[11:8] **ADD_DATA**: Additional data associated with node:
0x0 -> Additional Data Region 0
0x1 -> Additional Data Region 1
0x2 -> Additional Data Region 2
Others -> Reserved

[7:5] **TYPE**: node type
0x0 SCD/PES Parsing
0x1 S/PDIF
0x2 - 0x7 Reserved

[4:0] **REQ_MAP**: REQ_MAP/extended node type
0x1F Extended Node Type
0x0-0x1E Reserved (DREQ mapping for standard node)

NODE_NBYTES Number of bytes to be read from PES buffer

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

Address: *MemoryOffset* + 0x0008
Type: Read/write
Reset: Undefined
Description: Number of bytes in PES buffer on which SCD/PES parsing should be performed.

PESBUFFER Read pointer to PES buffer

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

Address: *MemoryOffset* + 0x000C
Type: Read/write
Reset: Undefined
Description: Read pointer to PES buffer.

43.7.1 Additional data regions

SC_WRITEn Start code list 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
SC_WRITE																												Reserved			SC_OVERFLOW

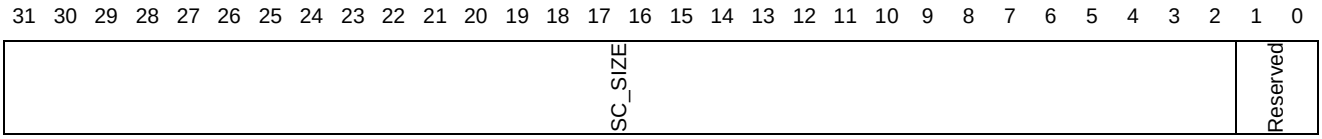
Address: *FDMABaseAddress* + 0x44F0 (Region 0), 0x4530 (Region 1), 0x4570 (Region 2), 0x45B0 (Region 3)
Type: Read/write
Reset: 0
Description:

[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
0x0 - no overflow
0x1 - overflow

SC_SIZEn Start code list size



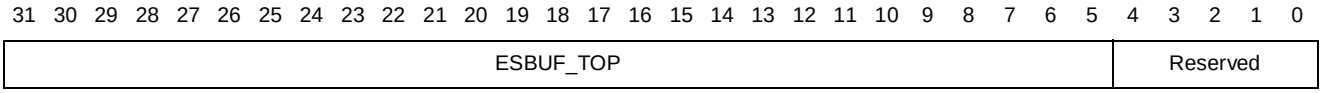
Address: *FDMABaseAddress* + 0x44F4 (Region 0), 0x4534 (Region 1), 0x4574 (Region 2), 0x45B4 (Region 3)

Type: Read/write

Reset: 0

Description: Size of start codes list (in number of words).

ESBUF_TOPn Top address of elementary stream buffer



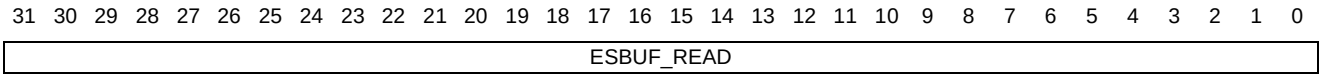
Address: *FDMABaseAddress* + 0x44F8 (Region 0), 0x4538 (Region 1), 0x4578 (Region 2), 0x45B8 (Region 3)

Type: Read/write

Reset: 0

Description: Address of top of elementary stream buffer(32 byte aligned)

ESBUF_READn Elementary stream buffer read pointer



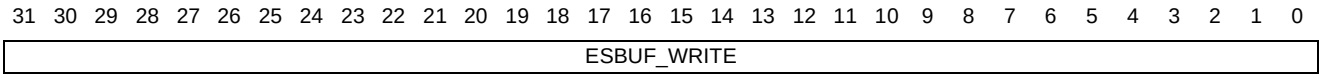
Address: *FDMABaseAddress* + 0x44FC (Region 0), 0x453C (Region 1), 0x457C (Region 2), 0x45BC (Region 3)

Type: Read/write

Reset: 0

Description:

ESBUF_WRITEn Elementary stream buffer write pointer



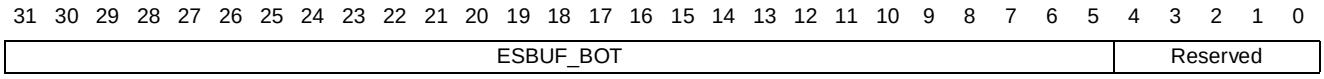
Address: *FDMABaseAddress* + 0x4500 (Region 0), 0x4540 (Region 1), 0x4580 (Region 2), 0x45C0 (Region 3)

Type: Read/write

Reset: 0

Description:

ESBUF_BOTn Elementary stream buffer bottom address



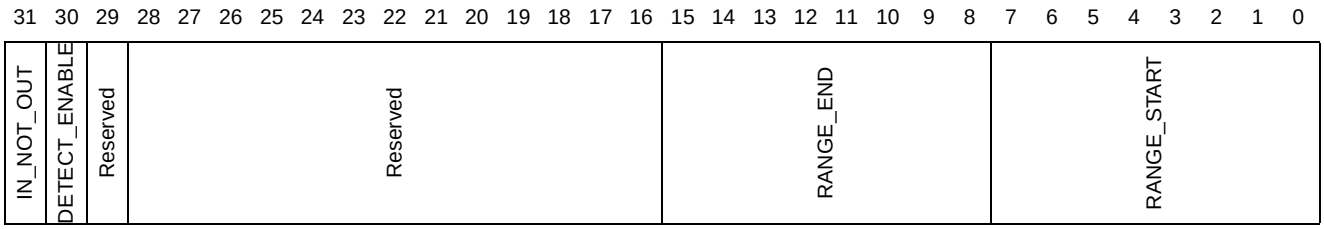
Address: *FDMABaseAddress* + 0x4504 (Region 0), 0x4544 (Region 1), 0x4584 (Region 2), 0x45C4 (Region 3)

Type: Read/write

Reset: 0

Description: ES buffer bottom pointer (32 byte aligned).

PES_CONTROLn PES header start code range control



Address: *FDMABaseAddress* + 0x4508 (Region 0), 0x4548 (Region 1), 0x4588 (Region 2)

Type: Read/write

Reset: 0

Description:

[31] **IN_NOT_OUT:** range mode
0x0 - Detect start codes outside this range
0x1 - Detect start codes within this range

[30] **DETECT_ENABLE:** enable start code detection for this range
0 -> no detection
1 -> detect

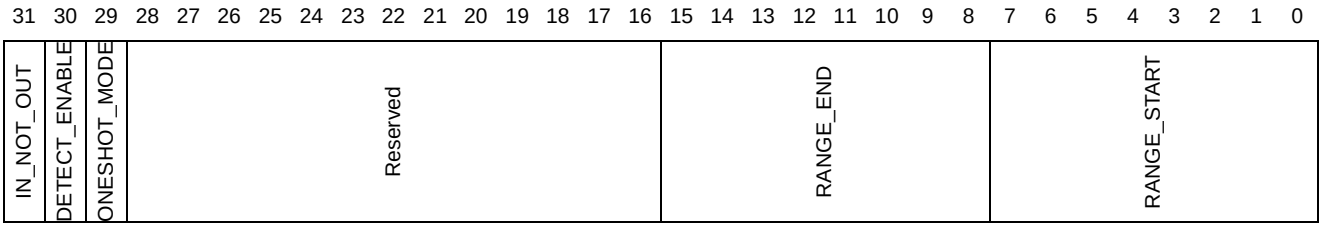
[29] **Reserved**

[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_CONTROLm Start code range n control for region m



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: Read/write

Reset: 0

Description:

[31] **IN_NOT_OUT**: range mode
0x0 - Detect start codes outside this range
0x1 - Detect start codes within this range

[30] **DETECT_ENABLE**: enable start code detection for this range
0 -> no detection
1 -> detect

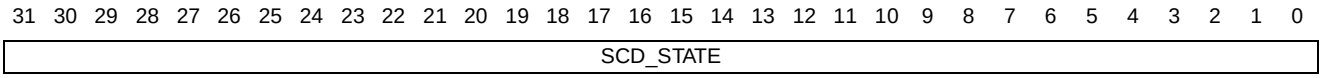
[29] **ONESHOT_MODE**: enable one shot mode
0 -> one shot mode disabled
1 -> one shot mode enabled

[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_STATEn Start code detector state



Address: *FDMABaseAddress* + 0x4514 to 0x452F (Region 0), 0x4554 to 0x456F (Region 1), 0x4594 to 0x45AF (Region 2)

Type: Read/write

Reset: 0

Description: Must be set to 0x0 when a new stream is started on this additional data area.

43.7.2 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: Read/write
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: Read/write
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: Read/write
Reset: Undefined
Description: Value of the start code

43.7.3 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: Read/write
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: Read/write
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: Read/write
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: Read/write
Reset: Undefined
Description: Bits 0 to 31 of PTS.

44 Infrared transmitter/receiver

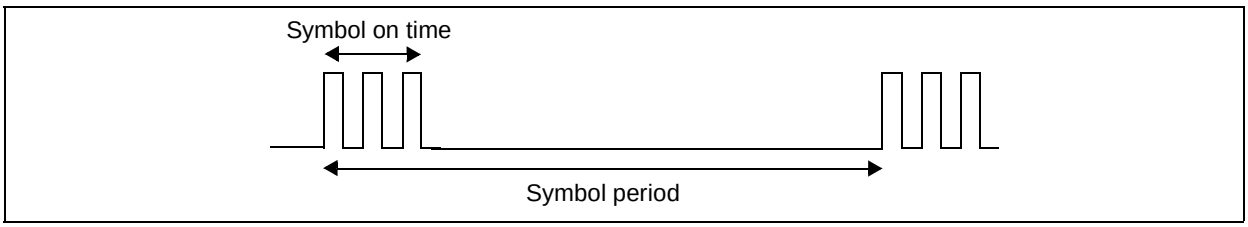
44.1 Overview

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 8-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 173.

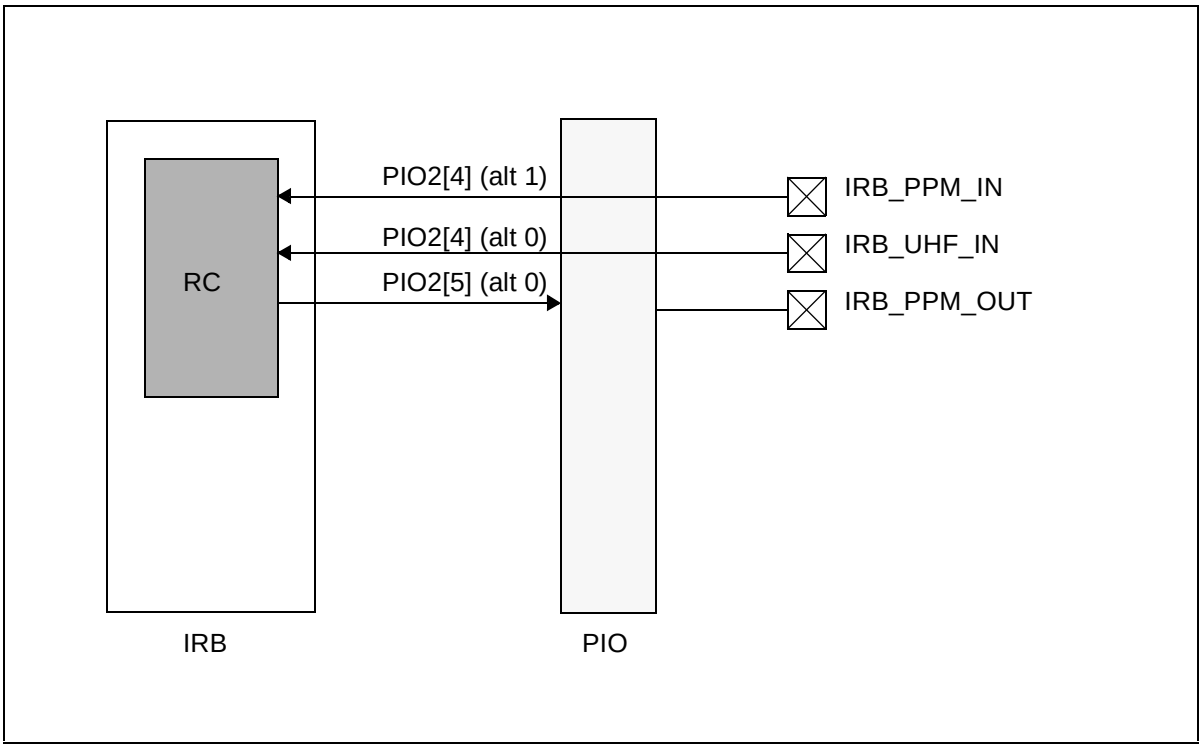
The incoming signal must be detected, and the subcarrier must be suppressed, externally. Only the symbol envelope can be used by the IR and UHF processors. It is sampled at 10 MHz and the sample values are transferred into the input buffer in microseconds.

Figure 173: IR transmitter/receiver symbol



The connectivity of the IR transmitter/receiver is given in Figure 174.

Figure 174: IRB connections diagram



44.2 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 175 shows the IR transmitter/receiver block diagram in a typical circuit configuration.

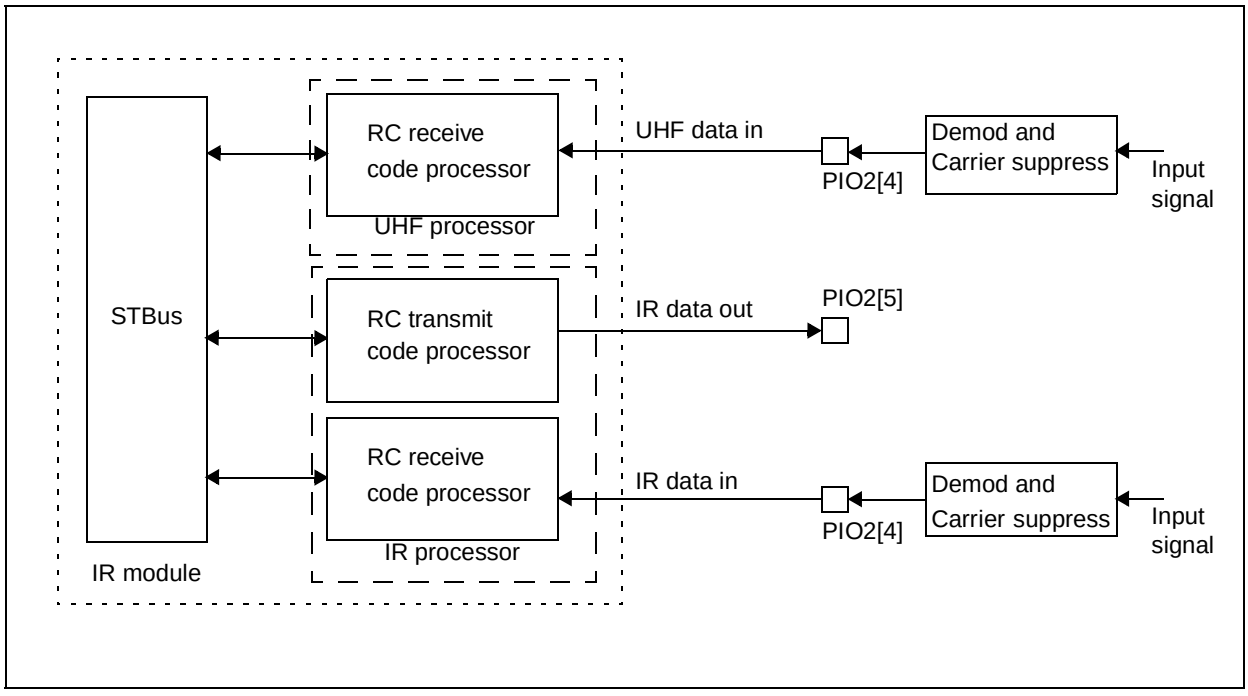
In the transmitter there are two programmable dividers to generate the prescaled clock and the subcarrier clock. The subcarrier clock sets the resolution for the transmitted data. Both receivers contain a sampling rate clock, which samples the incoming data, and is programmed to 10 MHz.

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.

There are two 8-word FIFOs in the RC transmitter and two 8-word FIFOs in each RC receiver. The eighth element in each 8-word FIFO is used internally and is not accessible to the STBus. Therefore, the 8-word FIFO is empty when there are seven empty words and full when it contains seven words. At all times, the fullness level of the 8-word FIFO is given in the corresponding status register.

The pair of FIFOs, for symbol period and symbol on time, in each submodule 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 175: IR transmitter/receiver block diagram and implementation



UHF and IR data in functions are programmed as alternative 0 and alternative 1 functions of PIO2. See Table 7: *PIO2 alternative functions* on page 34.

44.2.1 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 status 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. To clear the interrupt status clear the corresponding clear bit location in the IRB_TX_INTERRUPT_CLEAR_IR register.

Note: The IRB_TX_STATUS_IR register reflects the true status of the block.

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.

44.2.2 RC receive code processor

This section describes the UHF data and the IR data receivers. They are independent and identical. The 10 MHz sampling clock is common to both receivers and is set by register IRB_RX_SAMPLING_RATE_COMMON. This register is programmed with the value 5 for a 50 MHz infrared transmitter/receiver system clock, or with the value 6 for a 60 MHz clock.

Each receiver processes the incoming RC code symbol envelope and stores the values symbol period and symbol on time (in microseconds) in an 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 receive status (IRB_RX_STATUS_IR) 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. To clear the interrupt status clear the corresponding bit in the IRB_RX_INTERRUPT_CLEAR_IR register.

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. The RXOVERRUNSTATUS bit is automatically cleared but if the interrupt was enabled the corresponding interrupt status is cleared by writing 0x01 to the IRB_RX_INTERRUPT_CLEAR_IR register.

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 status bit is cleared automatically when the last value in the FIFO has been read.

Note: There are separate receive interrupt status and interrupt clear locations for IR and UHF.

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.

44.2.3 Noise suppression filter

This filter is programmable in the IR and UHF receiver using registers IRB_RX_NOISE_SUPPRESS_WIDTH_IR or 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.

44.2.4 Reception in low power mode

In low power mode the system clock is switched off and the 27 MHz clock is active. Before entering low power mode the 27 MHz operation should be enabled by setting the clock select bit in IRB_RX_CLOCK_SELECT. It is not mandatory to have an exact 27 MHz clock but only a clock running during low power mode if filtering is needed on wakeup trigger pulses. If filtering of wakeup pulses is not required then this clock need not be used and should be tied to a logic level (either 0 or 1). Care must be taken that the software does not executes a clock switch.

Wakeup trigger pulse generation

A wakeup trigger pulse is generated on the wakeup outputs RC_WAKEUP_TRIGGER and UHF_WAKEUP_TRIGGER on receipt of an active pulse on RC data or UHF data respectively. The polarity of the wake up trigger pulse is active high, irrespective of the active polarity of corresponding input.

The RC_WAKEUP_TRIGGER and UHF_WAKEUP_TRIGGER are generated only after being filtered by the noise suppression filter. Filtering can be disabled by programming the noise suppression width to 0x00. In low power mode the system clock should be switched off. To facilitate filtering by the noise suppression filter, the noise suppression filter should be switched to work with the 27 MHz clock by setting the CLOCK_SELECT bit in the IRB_RX_CLOCK_SELECT register to 1. In 27 MHz mode, if a value n is programmed in the noise suppression filter, all the pulses smaller than n / 27 µs are treated as noise and filtered. This is because the 27 MHz clock is fed directly to the noise suppression filter without any prescaler division.

The wakeup trigger pulse is generated only if the IRB is enabled and the corresponding receive section is enabled by setting the appropriate bit in the receive enable register.

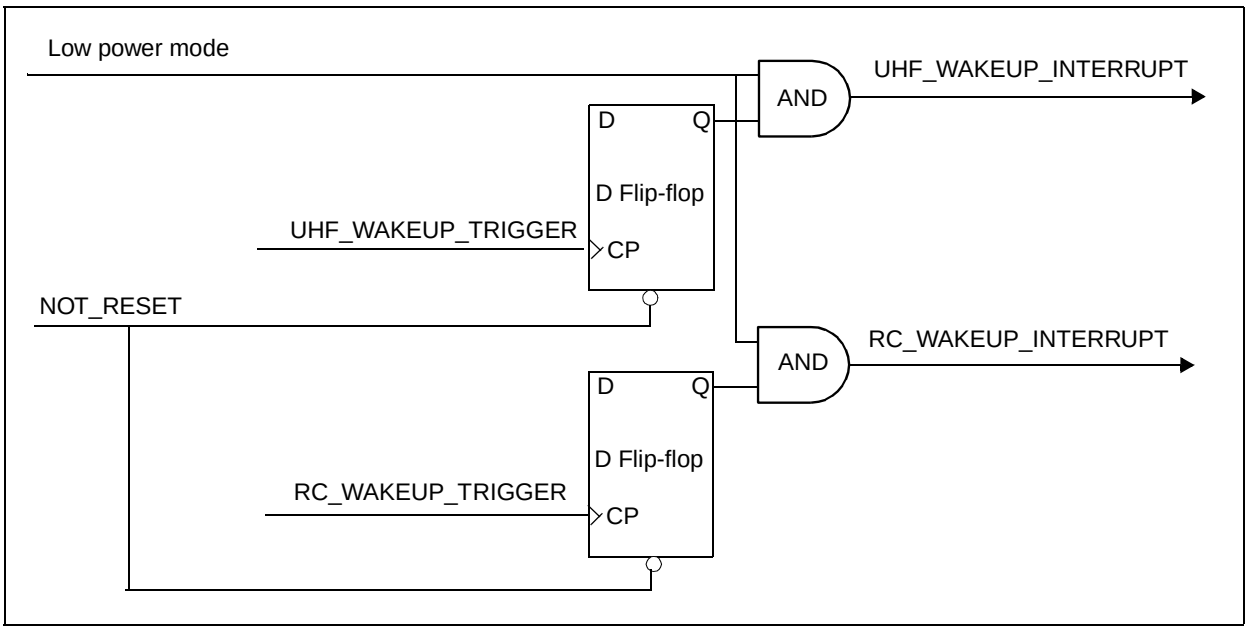
Wakeup interrupt

Once the device is in low power mode, the infra-red remote control device is restored to normal mode by using the wakeup feature.

The wakeup mechanism is shown in Figure 176. In low power mode the signal LOW_POWER_MODE is at logic 1. This signal can be the output of the low power controller's LOWPOWEROUT signal or any other signal which remains statically high when the chip is in low power mode.

On the rising edge of the trigger pulse the flip-flops hold logic 1 and a logic 1 is asserted on the corresponding wakeup interrupt pin. These pins are routed to the interrupt controller which in turn generates the wakeup by interrupt to the low power controller. The low power controller then restores the normal operation.

Figure 176: Wake up interrupt



45 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 the *IRBBaseAddress* + offset.

The *IRBBaseAddress* is:

0x2081 8000.

Table 106: Infrared transmitter/receiver registers

Register name	Description	Address offset	Type
RC transmit interface registers			
IRB_TX_PRE_SCALER_IR	Clock prescaler selection, see page 450	0x00	R/W
IRB_TX_SUB_CARRIER_IR	Subcarrier frequency. programming, see page 450	0x04	R/W
IRB_TX_SYM_PERIOD_IR[1]	Symbol period programming, see page 451	0x08	WO
IRB_TX_ON_TIME_IR[1]	Symbol on time programming, see page 451	0x0C	WO
IRB_TX_INT_EN_IR	Transmit interrupt enable register, see page 451	0x10	R/W
IRB_TX_INT_STATUS_IR	Transmit interrupt status register, see page 452	0x14	RO
IRB_TX_EN_IR	RC transmit enable register, see page 452	0x18	R/W
IRB_TX_INTERRUPT_ CLEAR_IR	Clear interrupt, see page 453	0x1C	WO
IRB_TX_SUB_CARRIER_ WIDTH_IR	Subcarrier frequency programming, see page 453	0x20	R/W
IRB_TX_STATUS_IR	Transmit status, see page 454	0x24	RO
RC receive interface registers for infrared signals			
IRB_RX_ON_TIME_IR[1]	Received pulse time capture, see page 454	0x40	RO
IRB_RX_SYM_PERIOD_IR[1]	Received symbol period capture, see page 455	0x44	RO
IRB_RX_INT_EN_IR	Receive interrupt enable register, see page 455	0x48	R/W
IRB_RX_INT_STATUS_IR	Receive interrupt status register, see page 456	0x4C	RO
IRB_RX_INT_STATUS_UHF	Receive interrupt status register, see page 456	0x8C	RO
IRB_RX_EN_IR	RC receive enable register, see page 456	0x50	R/W
IRB_RX_MAX_SYM_ PERIOD_IR	Maximum RC symbol period register, see page 456	0x54	R/W
IRB_RX_INTERRUPT_ CLEAR_IR	Clear interrupt, see page 457	0x58	WO
IRB_RX_STATUS_IR	Receive status, see page 458	0x6C	R/W
Common to RC and UHF receivers			
IRB_RX_SAMPLING_RATE_ COMMON	Sampling frequency division for UHF and IR frequencies, see page 458	0x64	R/W
IRB_RX_CLOCK_SELECT	Clock selection, see page 459	0x70	R/W
IRB_RX_CLOCK_SELECT_ STATUS	Clock selection status, see page 459	0x74	RO
RC receive interface registers for UHF signals			
IRB_RX_ON_TIME_UHF[1]	Received pulse time capture, see page 454	0x80	RO

Table 106: Infrared transmitter/receiver registers

Register name	Description	Address offset	Type
IRB_RX_SYM_PERIOD_UHF[1]	Received symbol period capture, see page 455	0x84	RO
IRB_RX_INT_EN_UHF	Receive interrupt enable register, see page 455	0x88	R/W
IRB_RX_INT_STATUS_UHF	Receive interrupt status register, see page 456	0x8C	RO
IRB_RX_EN_UHF	RC receive enable register, see page 456	0x90	R/W
IRB_RX_MAX_SYM_PERIOD_UHF	Maximum RC symbol period register, see page 456	0x94	R/W
IRB_RX_INTERRUPT_CLEAR_UHF	Clear interrupt, see page 457	0x98	WO
Noise suppression registers			
IRB_RX_NOISE_SUPPRESS_WIDTH_IR	Noise suppression width, see page 460	0x5C	R/W
IRB_RX_NOISE_SUPPRESS_WIDTH_UHF	Noise suppression width, see page 460	0x9C	R/W
Reverse polarity registers			
IRB_POLINV_REG_IR	IR polarity, see page 460	0x68	R/W
IRB_POLINV_REG_UHF	UHF polarity, see page 460	0xA8	R/W

1. These locations have a quadruple buffer, that is, a FIFO of length 4 words.

45.1 RC transmitter registers

IRB_TX_PRE_SCALER_IR Clock prescaler

7	6	5	4	3	2	1	0
PRESCALEVAL							

Address: *IRBBaseAddress* + 0x00

Type: Read/write

Reset: 0x00

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: Read/write

Reset: 0x00

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: Write only

Reset: 0x0000

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 eight-word 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: Write only

Reset: 0x0000

Description: The value in this register gives the symbol ON time (pulse duration) in periods of the subcarrier clock. This register is eight-word 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 register

7	6	5	4	3	2	1	0
Reserved			TXONEWRORDEMPTY	TXHALFEMPTY	TXEMPTY	TXUNDERRUN	TXIRQENABLE

Address: *IRBBaseAddress* + 0x10

Type: Read/write

Reset: 0x00

Description:

[7:5] **Reserved**
Set to logic 0

[5:4] **TXONEWORDEMPTY**
Select the interrupt enable/ disable when at least one word is empty in FIFO:
0: Interrupt disable 1: Interrupt enable

[3] **TXHALFEMPTY**
Select the FIFO half empty interrupt enable/ disable:
0: Interrupt disable 1: Interrupt enable

[2] **TXEMPTY**
Select the FIFO empty interrupt enable/ disable:
0: Interrupt disable 1: Interrupt enable

[1] **TXUNDERRUN**
Select the underrun interrupt enable/ disable:
0: Interrupt disable 1: Interrupt enable

[0] **TXIRQENABLE**
Select the global transmit interrupt enable/ disable:
0: Interrupt disable 1: Interrupt enable

IRB_TX_INT_STATUS_IR Transmit Interrupt status register

7	6	5	4	3	2	1	0
Reserved			TXONEWRORDEMPTY_STATUS	TXHALFEMPTY_STATUS	TXEMPTY_STATUS	TXUNDERRUN_STATUS	TXIRQENABLE_STATUS

Address: *IRBBaseAddress* + 0x14
Type: Read only
Reset: 0x00
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.

[7:5] **Reserved**
Set to logic 0

[5:4] **TXONEWORDEMPTY_STATUS**
0: Interrupt not pending 1: Interrupt pending

[3] **TXHALFEMPTY_STATUS**
0: Interrupt not pending 1: Interrupt pending

[2] **TXEMPTY_STATUS**
0: Interrupt not pending 1: Interrupt pending

[1] **TXUNDERRUN_STATUS**
0: Interrupt not pending 1: Interrupt pending

[0] **TXIRQENABLE_STATUS**
0: Interrupt not pending 1: Interrupt pending

IRB_TX_EN_IR RC transmit enable register

7	6	5	4	3	2	1	0
Reserved							TXENABLE

Address: *IRBBaseAddress* + 0x18
Type: Read/write
Reset: 0x00
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_INTERRUPT_CLEAR_IRClear interrupt

7	6	5	4	3	2	1	0
Reserved			TXONEWRORDEMPTY_CLR	TXHALFEMPTY_CLR	TXEMPTY_CLR	TXUNDERRUN_CLR	TXIRQENABLE_CLR

Address: *IRBBaseAddress* + 0x1C

Type: Write only

Reset: 0x00

Description: .

[7:5] **Reserved**
Set to logic 0

[5:4] **TXONEWORDEMPTY_CLR**
1: Clear Interrupt when one word is empty in FIFO

[3] **TXHALFEMPTY_CLR**
1: Clear half empty Interrupt

[2] **TXEMPTY_CLR**
1: Clear FIFO empty Interrupt

[1] **TXUNDERRUN_CLR**
1: Clear underrun Interrupt

[0] **TXIRQENABLE_CLR**
1: Clear global transmit Interrupt

IRB_TX_SUB_CARRIER_WIDTH_IR Subcarrier frequency programming

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBCARRIERWIDTHVAL															

Address: *IRBBaseAddress* + 0x20

Type: Read/write

Reset: 0x00

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 should be 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					TXFIFO			Reserved			TXONEWORD	TXHALFEMPTY	TXEMPTY	TXUNDERRUN	Reserved

Address: *IRBBaseAddress* + 0x24

Type: Read only

Reset: 0x00

Description:

[15:11] **Reserved**
Set to logic 0

[10:8] **TXFIFO**
Select the transmit FIFO fullness interrupt:
000: FIFO empty
001: One block in 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] **TXONEWORD**
1: At least one word empty in FIFO

[3] **TXHALFEMPTY**
1: FIFO half empty

[2] **TXEMPTY**
1: FIFO empty

[1] **TXUNDERRUN**
1: FIFO underrun

[0] **Reserved**

45.2 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: Read only

Reset: 0x0000

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 eight-word 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: Read only

Reset: 0x0000

Description: This register holds the detected time (in micro seconds) between the start of two successive received RC pulses. It is eight-word 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 register

7	6	5	4	3	2	1	0
Reserved		RXONEWORDFULL	RXHALFFULL	RXFULL	RXOVERRUN	RXLASTSYMBOL	RXIRQENABLE

Address: *IRBBaseAddress* + 0x48 (IR) *IRBBaseAddress* + 0x88 (UHF)

Type: Read/write

Reset: 0x00

Description:

[7:6] **Reserved**
Set to logic 0

[5] **RXONEWORDFULL**
0: Interrupt disable 1: Interrupt enable

[4] **RXHALFFULL**
0: Interrupt disable 1: Interrupt enable

[3] **RXFULL**
0: Interrupt disable 1: Interrupt enable

[2] **RXOVERRUN**
0: Interrupt disable 1: Interrupt enable

[1] **RXLASTSYMBOL**
0: Interrupt disable 1: Interrupt enable

[0] **RXIRQENABLE**
0: Interrupt disable 1: Interrupt enable

IRB_RX_INT_STATUS Receive Interrupt status register

7	6	5	4	3	2	1	0
Reserved		RXONEWORDFULL_STATUS	RXHALFFULL_STATUS	RXFULL_STATUS	RXOVERRUN_STATUS	RXLASTSYMBOL_STATUS	RXIRQENABLE_STATUS

Address: *IRBBaseAddress* + 0x4C (IR) *IRBBaseAddress* + 0x8C (UHF)

Type: Read only

Reset: 0x00

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.

[7:6] **Reserved**
Set to logic 0

[5] **RXONEWORDFULL_STATUS**
0: Interrupt not pending 1: Interrupt pending

[4] **RXHALFFULL_STATUS**
0: Interrupt not pending 1: Interrupt pending

[3] **RXFULL_STATUS**
0: Interrupt not pending 1: Interrupt pending

[2] **RXOVERRUN_STATUS**
0: Interrupt not pending 1: Interrupt pending

[1] **RXLASTSYMBOL_STATUS**
0: Interrupt not pending 1: Interrupt pending

[0] **RXIRQENABLE_STATUS**
0: Interrupt not pending 1: Interrupt pending

IRB_RX_EN RC receive enable register

7	6	5	4	3	2	1	0
Reserved							RXENABLE

Address: *IRBBaseAddress* + 0x50 (IR) *IRBBaseAddress* + 0x90 (UHF)

Type: Read/write

Reset: 0x00

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 register

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: Read/write

Reset: 0x0000

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_INTERRUPT_CLEAR Clear interrupt

7	6	5	4	3	2	1	0
Reserved		RXONEWRORDFULL_CLR	RXHALFFULL_CLR	RXFULL_CLR	RXOVERRUN_CLR	RXLASTSYMBOL_CLR	Reserved

Address: *IRBBaseAddress* + 0x58 (IR) *IRBBaseAddress* + 0x98 (UHF)

Type: Write only

Reset: 0x00

Description: .

[7:6] **Reserved**
Set to logic 0

[5] **RXONEWORDFULL_CLR**
1: Clear interrupt when one word is received in FIFO

[4] **RXHALFFULL_CLR**
1: Clear HALF FULL INTERRUPT

[3] **RXFULL_CLR**
1: Clear FIFO full Interrupt

[2] **RXOVERRUN_CLR**
1: Clear overrun Interrupt

[1] **RXLASTSYMBOL_CLR**
1: Clear last symbol Interrupt

[0] **Reserved**

IRB_RX_STATUS Receive status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RXFIFO			Reserved		RXATLEASTONEWORD	RXHALFFULL	RXFULL	RXOVERRUN	RXLASTSYMBOL	Reserved

Address: *IRBBaseAddress* + 0x6C (IR) *IRBBaseAddress* + 0xAC (UHF)

Type: Read only

Reset: 0x00

Description:

[15:11] **Reserved**
Set to logic 0

[10:8] **RXFIFO**
Select the receive FIFO fullness interrupt:
000: FIFO empty
001: One block in 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] **RXATLEASTONEWORD**
1: Clears at least one word in FIFO interrupt

[4] **RXHALFFULL**
1: Clears FIFO half full interrupt

[3] **RXFULL**
1: Clears FIFO full interrupt

[2] **RXOVERRUN**
1: Clears FIFO overrun interrupt

[1] **RXLASTSYMBOL**
1: Clears last symbol receive interrupt

[0] **Reserved**

45.3 RC and UHF receiver control

IRB_RX_SAMPLING_RATE_COMMON Sampling frequency division for UHF and IR

7	6	5	4	3	2	1	0
Reserved				SETSAMPLERATE			

Address: *IRBBaseAddress* + 0x64

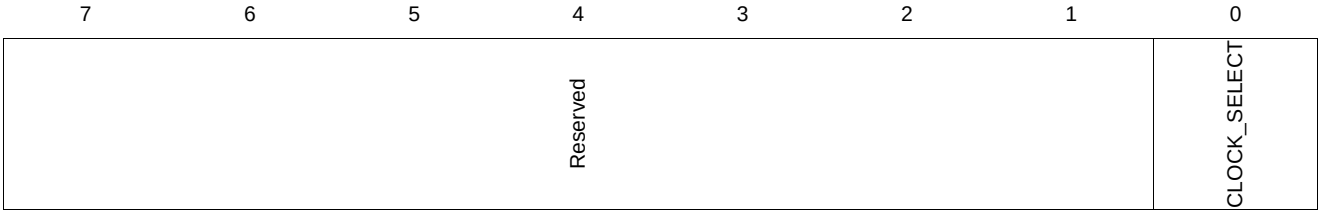
Type: Read/write

Reset: 0x00

Description: The value in this register is the divisor which sets the sampling rate for the RC receive sections to 10 MHz. It must be set to five with an IR transmitter/receiver system clock of 50 MHz or to six with a clock of 60 MHz. The register is common to both the IR and the UHF receive processors.

45.4 Clock selection

IRB_RX_CLOCK_SELECT Clock selection



Address: *IRBBaseAddress* + 0x70

Type: Read/write

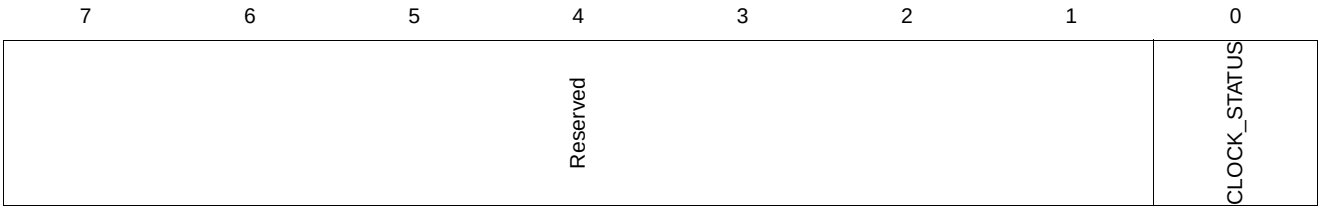
Reset: 0x00

Description: This register selects whether the IR and UHF receivers are clocked by the system clock or the 27 MHz clock. In low power mode the system clock should be switched off. The noise suppression filters are clocked by the 27 MHz clock to ensure the filtering takes place on the received signal even in low power mode.

[7:1] **Reserved**
Set to logic 0

[5:4] **CLOCK_SELECT**
0: System clock 1: 27 MHZ clock

IRB_RX_CLOCK_SELECT_STATUS Clock selection status



Address: *IRBBaseAddress* + 0x74

Type: Read only

Reset: 0x00

Description: This register shows whether the IR and UHF receivers are clocked by the system clock or the 27 MHz clock. After changing the clocking mode by programming the IRB_RX_CLOCK_SELECT register, software should read the IRB_RX_CLOCK_SELECT_STATUS register to see if the clock change has happened.

[7:1] **Reserved**
Set to logic 0

[5:4] **CLOCK_STATUS**
0: System clock 1: 27 MHZ clock

45.5 Noise suppression register

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: Read/write

Reset: 0x00

Description: The value, in microseconds, in this register determines the maximum width of noise pulses which the filter suppresses.

45.6 Reverse polarity registers

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_IR Reverse polarity on IR data

7	6	5	4	3	2	1	0
Reserved							POLARITY

Address: *IRBBaseAddress* + 0x68

Type: Read/write

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

IRB_POLINV_REG_UHF Reverse polarity on UHF data

7	6	5	4	3	2	1	0
Reserved							POLARITY

Address: *IRBBaseAddress* + 0xA8

Type: Read/write

Reset: 0

Description:

[7:1] **Reserved**

[0] **POLARITY**

0: No polarity inversion 1: Polarity of UHF data inverted

This bit should always be set to 1

46 Asynchronous serial controller (ASC)

46.1 Overview

The asynchronous serial controller, also referred to as the UART interface, provides serial communication between the STx5119 and other microcontrollers, microprocessors or external peripherals. The STx5119 provides two ASCs, one of which is used as the smartcard controller.

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.

46.2 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.

46.2.1 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.

46.2.2 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.

46.3 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).

46.3.1 8-bit data frames

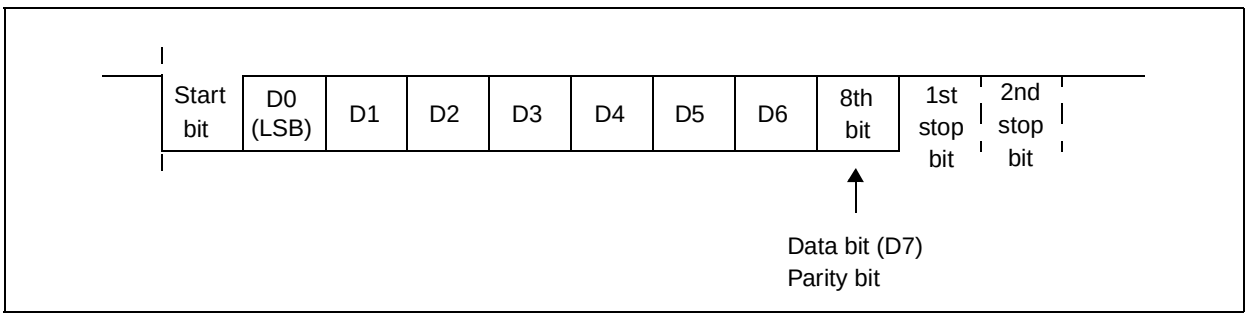
Figure 177 illustrates an 8-bit transmitted data frame. 8-bit frames may use one of the following formats:

- 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 177: 8-bit Tx data frame format

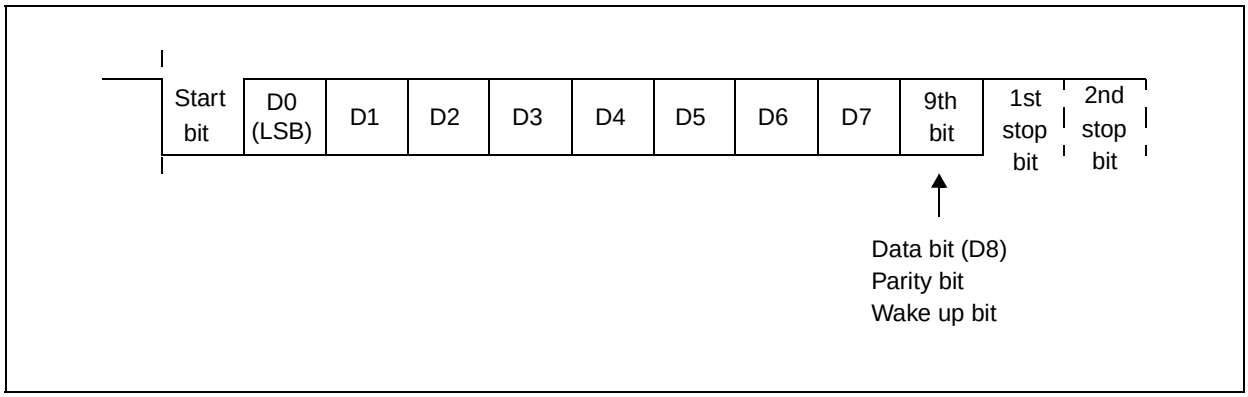


46.3.2 9-bit data frames

Figure 178 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 178: 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.

46.4 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 CTS is high.

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.

46.4.1 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.

46.4.2 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.

46.5 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.

46.5.1 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 46.5).

  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 46.3.1 on page 462), the ninth bit of each input entry (bit 8 of 0 to 9) is undefined.

46.5.2 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.

46.5.3 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.

46.6 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.

46.6.1 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 communications clock, see Chapter 18: *System services* on page 123).

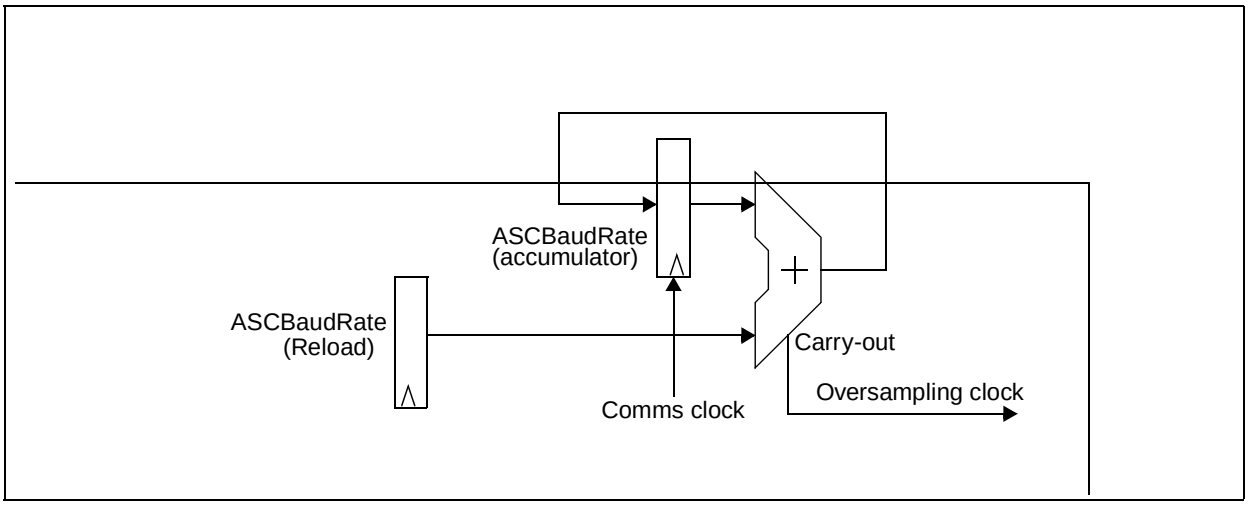
The baudrate counter is clocked by the comms clock. It counts downwards and can be started or stopped by the RUN bit in the ASC_n_CONTROL register. Each underflow of the timer provides one oversampling baudrate clock pulse. The counter is reloaded with the value stored in its 16-bit reload register each 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 179.

Figure 179: 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.

46.7 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 180.

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:

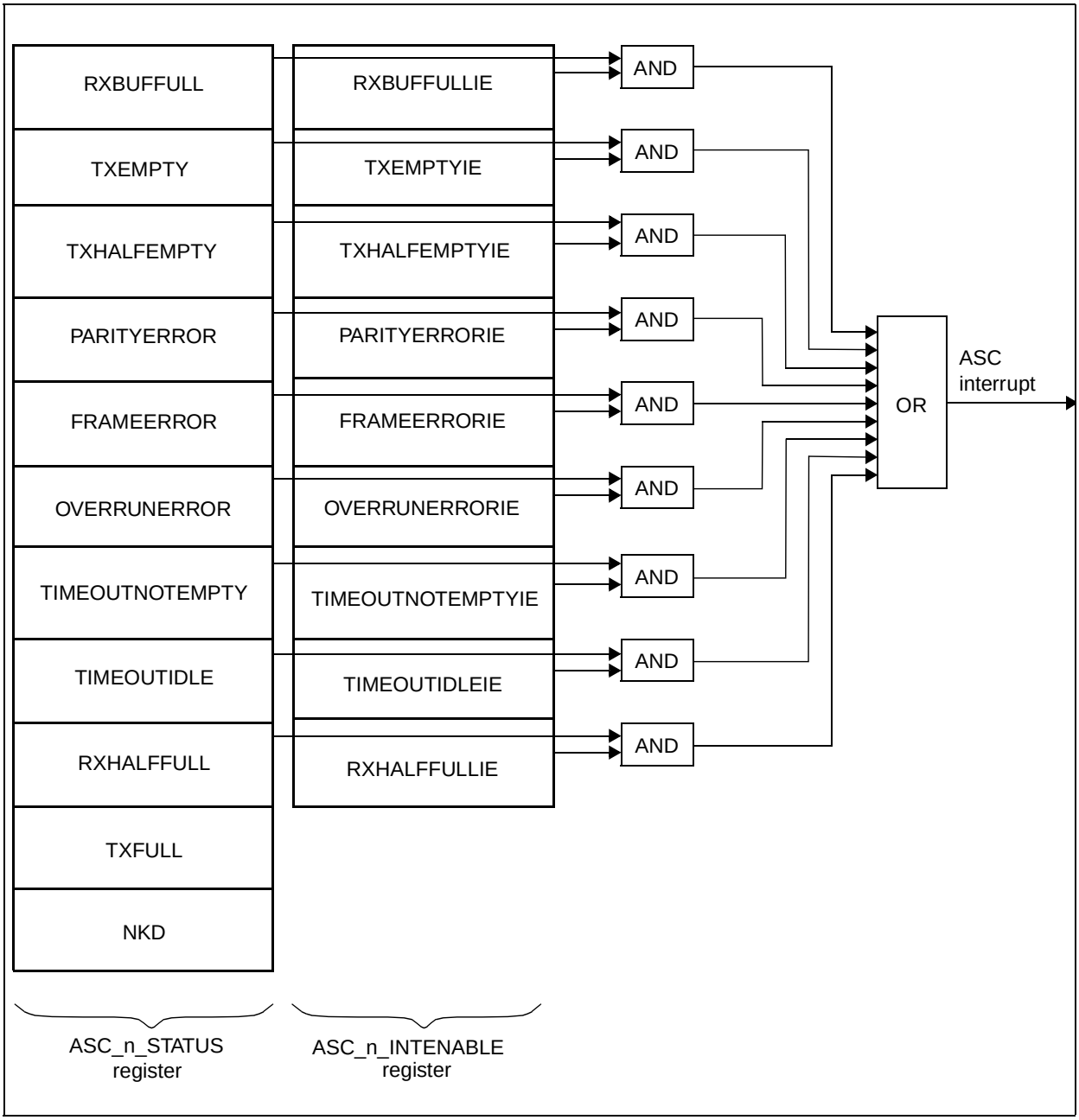
- transmitter interrupt status bits (TXEMPTY and TXHALFEMPTY) are reset when a character is written to the transmitter buffer,
- receiver interrupt status bit (RXBUFFULL) is reset when a character is read from the receive buffer,
- PARITYERROR and FRAMEERROR status bits are reset when all characters 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.

46.7.1 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 180: ASC status and interrupt registers



As shown in Figure 180, 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.

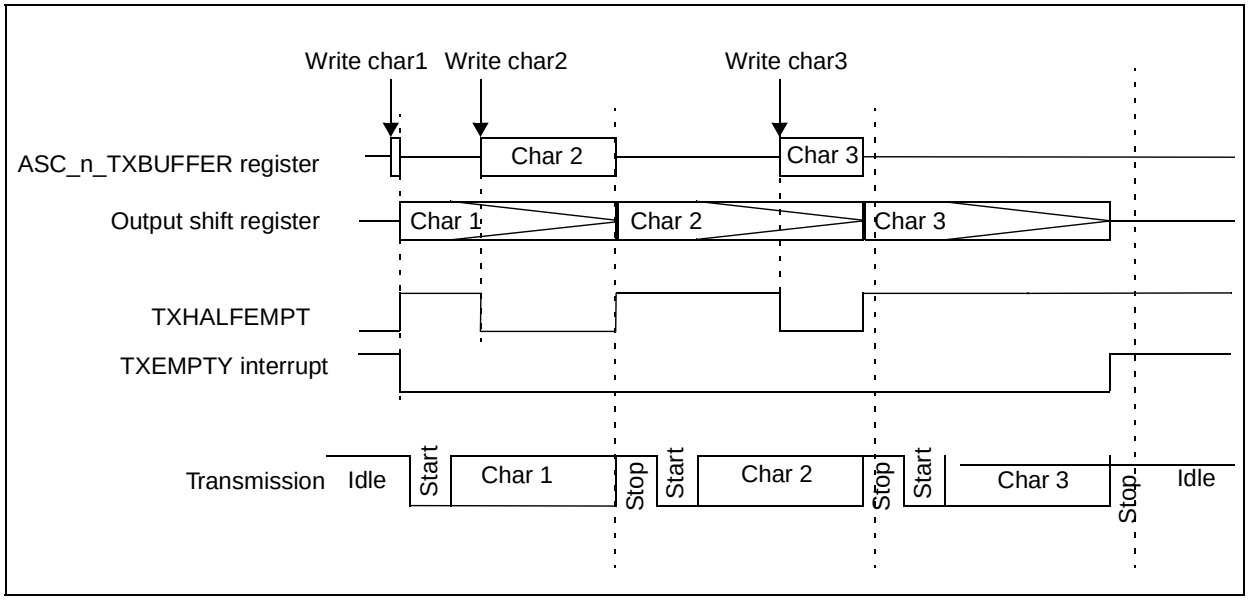
For single transfers it is sufficient to use the transmitter interrupt (TXEMPTY), which indicates that the previously loaded data has been transmitted, except for the last bit of a frame.

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.

46.7.2 Using the ASC interrupts when FIFOs are enabled

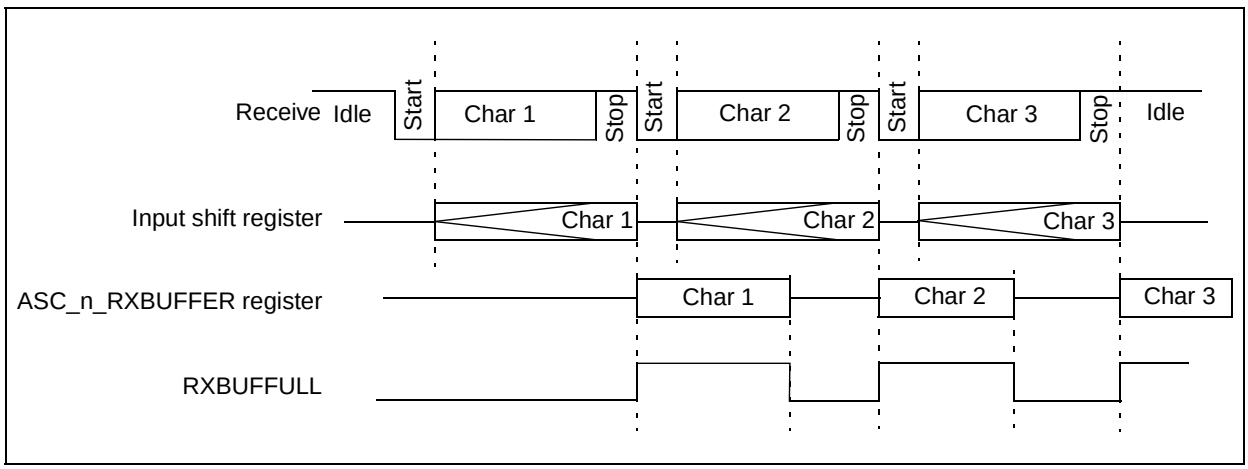
To transmit a large number of characters back to back, the driver routine initially writes 16 characters to ASC_n_TXBUFFER. Then every time a TXHALFEMPTY interrupt fires, it writes eight more. When there is nothing more to send, a TXEMPTY interrupt tells the driver that everything has been transmitted.

Figure 181: 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 182: ASC reception



46.8 ASC operation

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.

46.8.1 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.

46.8.2 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.

46.8.3 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.

46.8.4 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.

47 Asynchronous serial controller (ASC) registers

The registers for each ASC are grouped in a 4-Kbyte block, with the base of the block for ASC number n at the address *ASCnBaseAddress*.

Register addresses are provided as the *ASCnBaseAddress* + offset.

The *ASCnBaseAddresse*s are:

UART0: 0x2083 0000,

UART1: 0x2083 1000,

There is also one enable register located in the infrared blaster block. This is provided as *IRBBaseAddress* + offset. The *IRBBaseAddress* is 0x2011 5000.

Table 107: Asynchronous serial controller (ASC) registers

Register name	Description	Address offset	Type
ASC_n_BAUDRATE	ASCn baudrate generator, see page 475	0x000	R/W
ASC_n_TXBUFFER	ASCn transmit buffer, see page 481	0x004	WO
ASC_n_RXBUFFER	ASCn receive buffer, see page 479	0x008	RO
ASC_n_CONTROL	ASCn control register, see page 476	0x00C	R/W
ASC_n_INTENABLE	ASCn interrupt enable, see page 478	0x010	R/W
ASC_n_STATUS	ASCn interrupt status, see page 480	0x014	RO
ASC_n_GUARDTIME	ASCn guard time, see page 477	0x018	R/W
ASC_n_TIMEOUT	ASCn time out, see page 481	0x01C	R/W
ASC_n_TXRESET	ASCn transmit FIFO reset, see page 482	0x020	WO
ASC_n_RXRESET	ASCn receive FIFO reset, see page 479	0x024	WO
ASC_n_RETRIES	ASCn number of retries on transmission, see page 478	0x028	R/W

ASC_n_BAUDRATE ASCn baudrate generator

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

Address: *ASCnBaseAddress* + 0x00

Type: Read/write

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 108 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 108: 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	0x0091	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 109: 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 register

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: Read/write

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: Read/write

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: Read/write

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: Read/write

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

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

Address: *ASCnBaseAddress* + 0x08

Type: Read only

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: Write only

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: Read only

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: Read/write

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: Write only

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: Write only

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.

48 Synchronous serial controller (SSC)

48.1 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. Two SSCs are provided on the STx5119. The SSC gives full support for the I^2C bus.

The following features are programmable:

- baudrate,
- data width,
- shift direction,
- clock polarity,
- clock phase,
- FIFO or nonFIFO modes.

The SSC shares pins with the parallel input/output (PIO) ports. It supports both half-duplex and full-duplex synchronous communication when used in conjunction with the PIO configuration.

The SSC fully supports the I^2C bus standard. The extra I^2C features include:

- multimaster 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.

48.1.1 Pin connection and control

The SSC uses three signals:

- serial clock SCLK,
- serial data in/out MRST,
- serial data out/in MTSR.

The SCLK, MTSR and MRST signals are provided by two (I^2C) or three (SPI) 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.

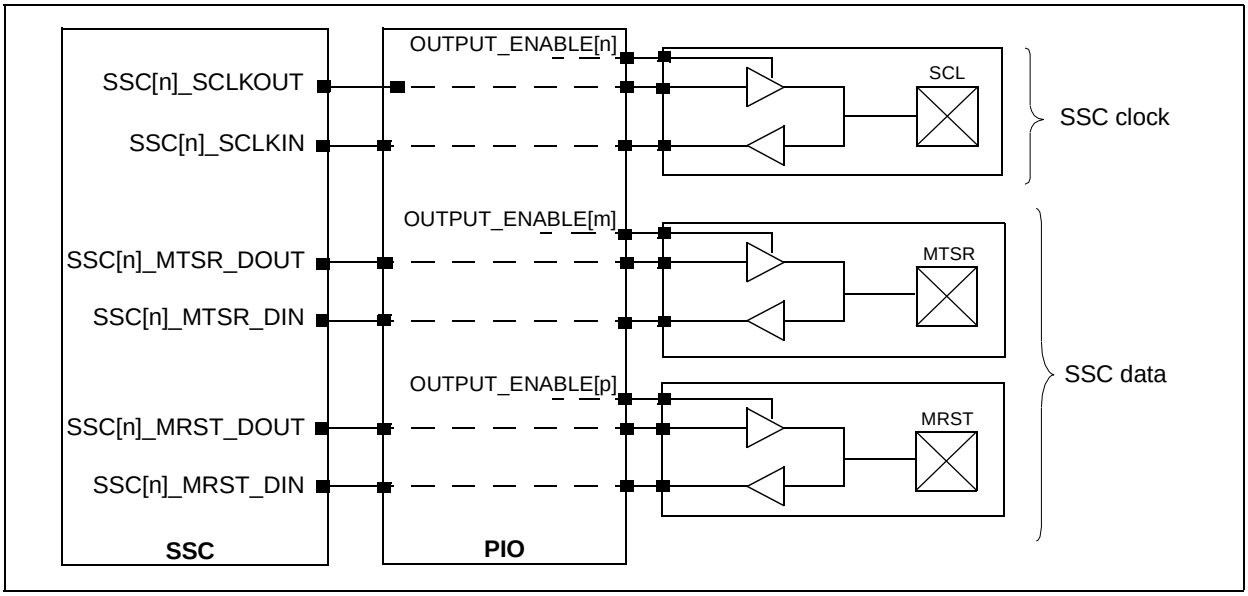
The connections between the SSC ports and the relevant PIO pins are illustrated in Figure 183. Pins are shared with PIO.

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.
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.

See Table 20: *Synchronous serial controller pin mapping* on page 44 for further information on connecting the pins.

Figure 183: SSC to PIO 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).

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 signals dependent on the master or slave mode of the SSC.

48.2 Basic operation

The serial clock output signal is programmable in master mode for baudrate, polarity and phase. This is described in Section 48.2.1: *Clock generation* on page 486.

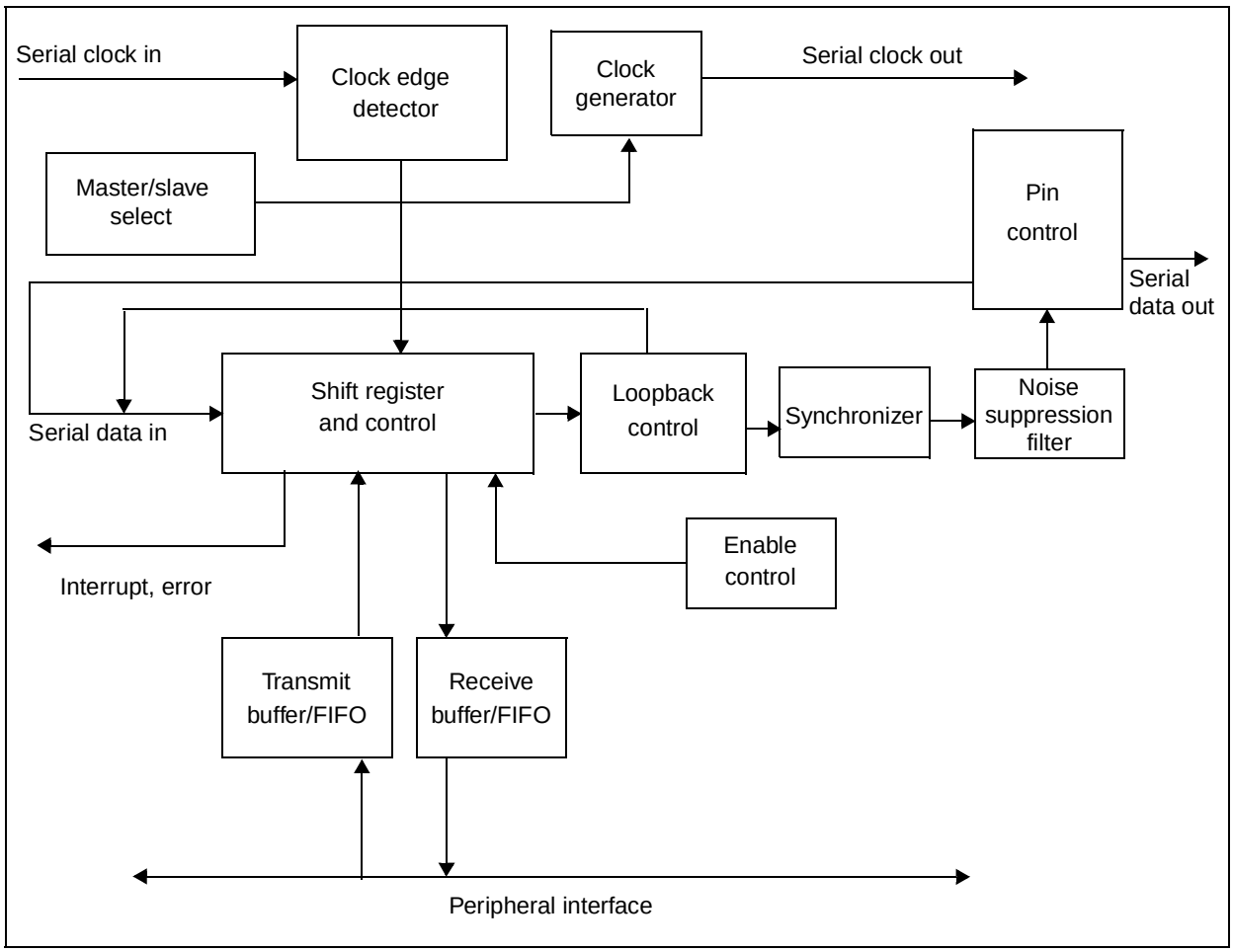
The SSC works by taking the data frame (two to sixteen bits) from a transmission buffer or FIFO and placing it into a shift register. It then shifts the data at the serial clock frequency out of the output pin and synchronously shifts in data coming from the input pin. The number of bits and the direction of shifting (MSB or LSB first) are programmable. This is described in Section 48.2.3: *Shift register* on page 488.

After the data frame has been completely shifted out of the shift register, it transfers the received data frame into the receive buffer or FIFO. The transmit and receive buffers are described in Section 48.2.5: *Transmit and receive buffers (nonFIFO mode)* on page 489 and Section 48.2.6: *Transmit and receive FIFOs (FIFO mode)* on page 489. In FIFO mode the FIFO is 8 bits deep and 16 bits wide while in nonFIFO mode the SSC is double buffered. This allows back-to-back transmission and reception of data frames up to the speed that interrupts can be serviced. The interrupt latency is further reduced in FIFO mode due to the additional 8-bit FIFO depth.

The SSC generates interrupts in a variety of situations:

- the transmission buffer is empty,
- the receive buffer is full,
- the transmit FIFO is half empty,
- the transmit FIFO is full,
- the receive FIFO is half full
- an error occurs.

Figure 184: SSC architecture



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 48.2.7: *Loopback mode* on page 490.

The SSC can be turned on and off by setting the enable control. This is described in Section 48.2.8: *Enabling operation* on page 490. It can be also be set to operate as a bus master or as a bus slave device. This is described in Section 48.2.9: *Master/slave operation* on page 490.

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 48.2.10: *Error detection* on page 490.

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 48.3: *I2C operation* on page 493.

48.2.1 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. At the end of the last clock period the shift register is unloaded into the receive buffer or FIFO.

For I^2C operation there are a number of additional clocking features. These are described in Section 48.3: *I2C operation* on page 493.

Clock control

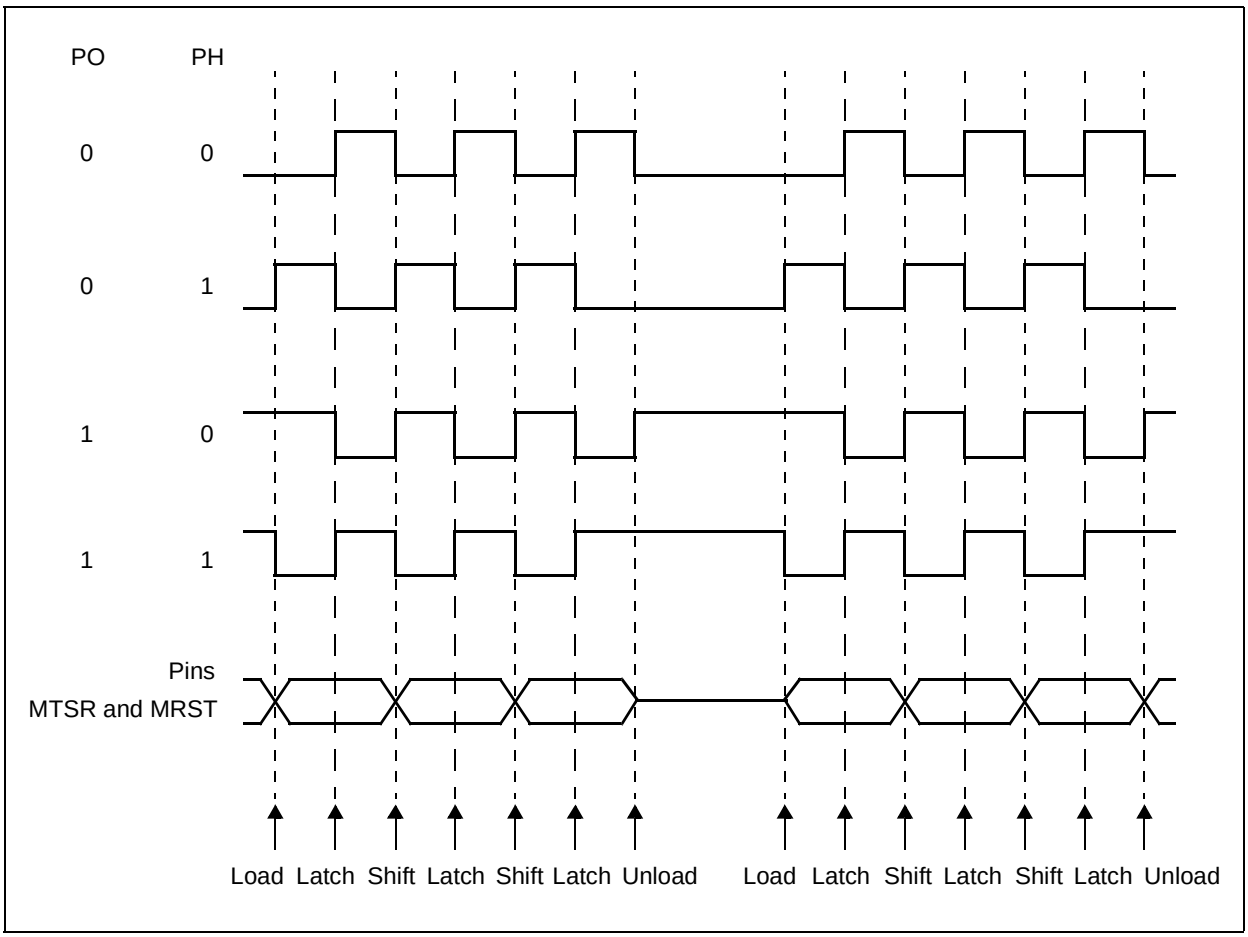
In master mode, the serial clock SCLK, is generated by the SSC according to the setting of the phase bit PH and polarity bit PO in the control register SSCnCON.

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 185.

Figure 185: 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.

48.2.2 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 110.

Table 110: 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.

Programming the SSCnBRG register with 0x0 results in the slowest clock because it has to count the entire range of the 16-bit counter. A slave device need not be programmed with any SSCnBRG value since it is always forced internally to 0. This ensures that the slave clock always follows the master clock and eradicates clock synchronization problems. The minimum programmed SSCnBRG value has to be 0x07.

For operation at even lower frequencies the clock generated according to the baudrate generator value can be further prescaled by a division factor. The prescaler register PRESCALER_BRG can be programmed with this prescaler division factor.

Hence the clock generated can be defined by:

Baudrate = F_{CPU} / (2 * *prsc*)

prsc = F_{CPU} / (2 * baudrate)

where prsc = *brg* * *prescaler_division_factor.*

48.2.3 Shift register

The shift register is loaded with the data in the transmit buffer or FIFO 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.

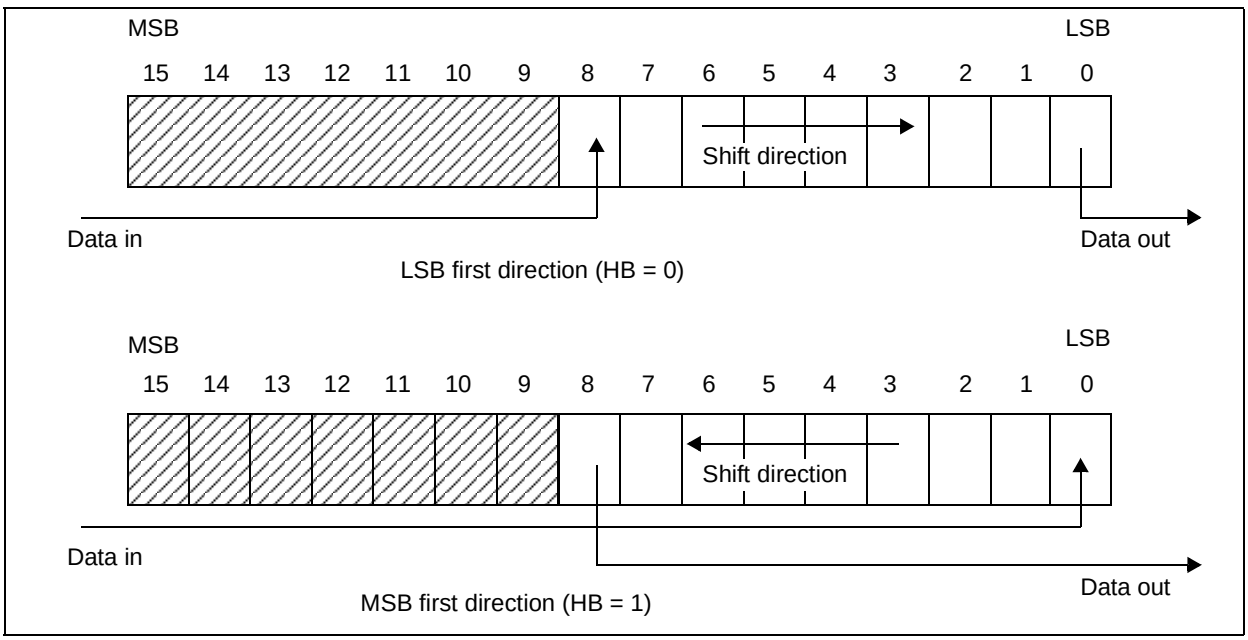
The shift register can shift out LSB first or MSB first. This is programmed by the heading control bit HB in the control register SSCnCON. A logic 1 indicates that the MSB is shifted out first and a logic 0 that the LSB shifts first.

The width of a data frame is also programmable from 2 bits 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 ^{2}C 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 186.

Figure 186: 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 48.2.1: *Clock generation* on page 486). 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^{2}C bus standard. These are described in Section 48.3: *I2C operation* on page 493.

48.2.4 Receive clock and data sampling

The serial clock and 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.

48.2.5 Transmit and receive buffers (nonFIFO mode)

This mode is selected by writing 0 in the ENB_TX_FIFO bit (bit 11) and ENB_RX_FIFO bit (bit 12) respectively of the SSCnCON register. The transmit and receive FIFO's are bypassed in this mode. This is the SSC's default mode.

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 48.2.10: *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 48.2.10: *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.

Note: When the first data pattern is written into the transmit buffer it is directly loaded into the transmit shift register. Again, when a transmit underrun condition occurs, the first data pattern written is loaded into the transmit shift register directly.

48.2.6 Transmit and receive FIFOs (FIFO mode)

This mode is selected by writing 1 in the ENB_TX_FIFO bit (bit 11) and ENB_RX_FIFO bit (bit 12) respectively of the SSCnCON register. In this mode the transmit buffer is bypassed. The transmit FIFO is 16 bits wide, and 8 bits deep. It is possible to write eight data patterns into it. Once filled, the FIFO cannot be overwritten with further data till one or more spaces become available.

The width of the data pattern to be transmitted is selected by setting the data width in the SSCnCON register.

The transmit FIFO may be flushed on a number of conditions, such as receiving a software reset, receiving a NACK, disabling the transmit section, arbitration loss, and bypassing FIFO mode by switching off the ENB_TX_FIFO bit in the SSCnCON register.

The receive FIFO may be flushed on a number of conditions, such as receiving a software reset, disabling the receive section, and bypassing FIFO mode by switching off the ENB_RX_FIFO bit in the SSCnCON register.

If the FIFO is filled with n data patterns (where n <= 8) then after all n data bytes have been transmitted the FIFO empty interrupt is generated, if enabled.

When the transmit FIFO is completely filled with eight data patterns, the FIFO full interrupt is generated, if enabled.

When the receive FIFO is completely filled with eight data patterns, the FIFO cannot be overwritten with further data till one or more data is read by the CPU.

Note: When the first data pattern is written into the transmit FIFO it is directly loaded into the transmit shift register. Again, when a transmit underrun condition occurs, the first data pattern written is loaded into the transmit shift register directly.

48.2.7 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.

48.2.8 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.

48.2.9 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.

48.2.10 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.

In FIFO mode this condition occurs when old data is about to be transmitted from the FIFO. It is cleared by the next write to the transmit FIFO.

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.

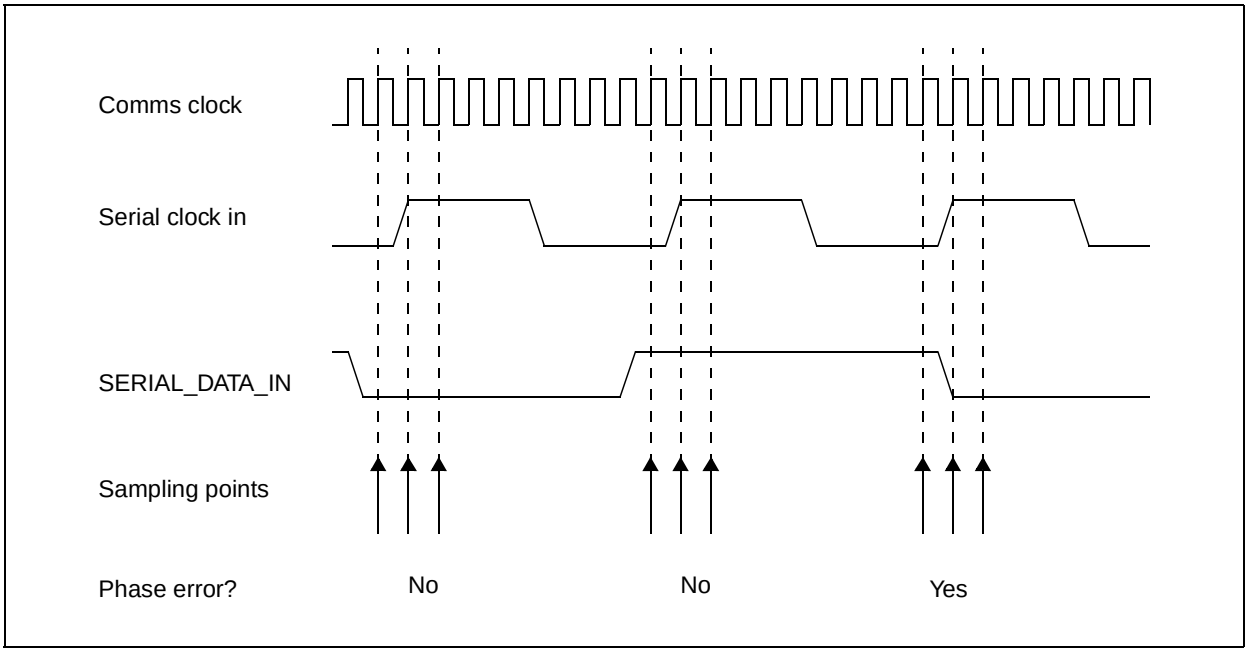
In FIFO mode this condition occurs when the receive FIFO is full and a new data byte is received. The FIFO is not overwritten, but the error condition occurs. Data bytes may be lost if the contents of the receive FIFO are not read.

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 187.

Figure 187: 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.

48.2.11 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.

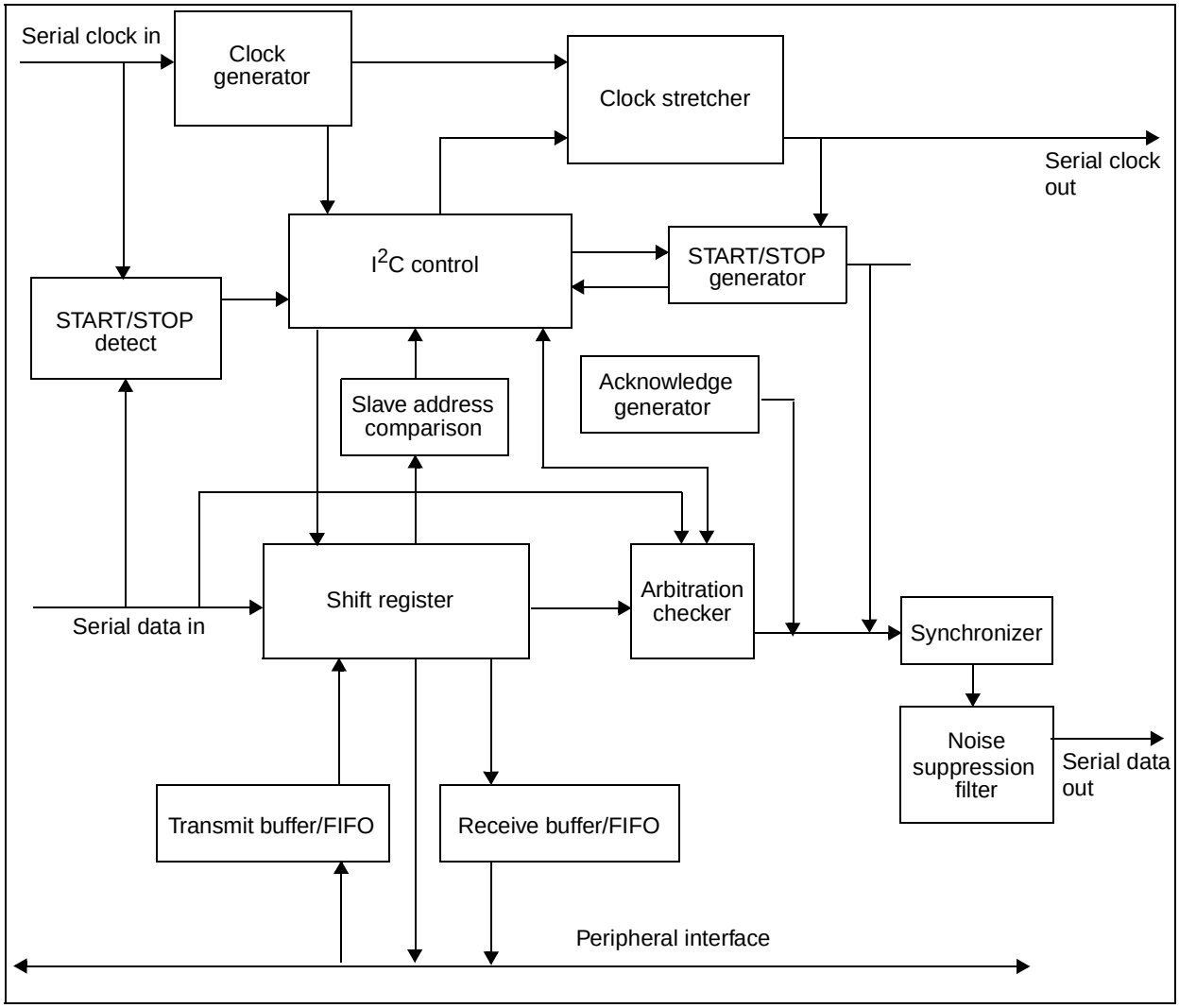
To determine which interrupt condition has occurred, a status register SSCnSTAT is provided which includes a bit for each condition. This is independent of the interrupt enables register SSCnIEN, and determines whether the condition asserts one or more of the interrupt signals.

48.3 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 188.

Figure 188: I2C architecture



48.3.1 I²C control

There are a number of features of the I²C bus protocol that 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 2-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 48.3.2: *Clock synchronization* on page 496). When the I^2C control bit is set, the START and STOP condition detection is performed. In addition bits PH and PO of register SSC_nCON must be set to 1.

Fast mode is supported by setting the appropriate values in the following registers:

- SSCn_START_HOLD_TIME for the I^2C start hold time,
- SSCn_REP_START_HOLD_TIME for the I^2C repeated start hold time,
- SSCn_REP_START_SETUP_TIME for the I^2C repeated start setup time,
- SSCn_DATA_SETUP_TIME for the I^2C data setup time,
- SSCn_STOP_SETUP_TIME for the I^2C stop setup time,
- SSCn_BUS_FREE_TIME for the I^2C bus free time.

Fast mode differs from normal mode by the values set in the above registers. Hence by appropriately programming these registers operations in both modes can be achieved.

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 48.3.4: *Slave address comparison* on page 497). 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 48.3.4: *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 48.3.5: *Clock stretching* on page 498).

To generate a START condition, the I^2C START condition generate bit STRTG in register SSCnI2C, must be set (see Section 48.3.6: *START/STOP condition generation* on page 498). To generate a STOP condition, the I^2C STOP condition generate bit STOPG, must be set (see Section 48.3.6: *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 48.3.7: *Acknowledge bit generation* on page 499).

To indicate to the software that various situations have arisen on the I^2C 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; the interrupt occurs after the second byte acknowledge bit, in the situations where 2 bytes of address are sent; or it occurs after the first byte acknowledge in the situation where only 1 byte is required.

Until the status bit is reset, the SSC holds the clock line low (see Section 48.3.5: *Clock stretching* on page 498). This forces the master device to wait until the software has processed the interrupt.

Interrupt bits are cleared in the CLEAR_STATUS_SSC register. Writing 1 in the corresponding location clears the interrupt.

The status bit is 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 status bit is reset by a read of the status register SSCnSTAT. The STOP interrupt is cleared by performing a clear bit operation on the CLR_SSCSTOP bit (bit 7) of the CLEAR_STATUS_SSC register.

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 48.3.5: *Clock stretching*). This forces the winning master device to wait until the software has processed the interrupt.

The status bit is reset by a read of the status register SSCnSTAT.

To indicate that the I^2C bus is busy (that is, between a START and a STOP condition), the I^2C bus busy bit BUSY in register SSCnSTAT is set. This does not generate an interrupt.

48.3.2 Clock synchronization

The I^2C 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).

The devices 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.

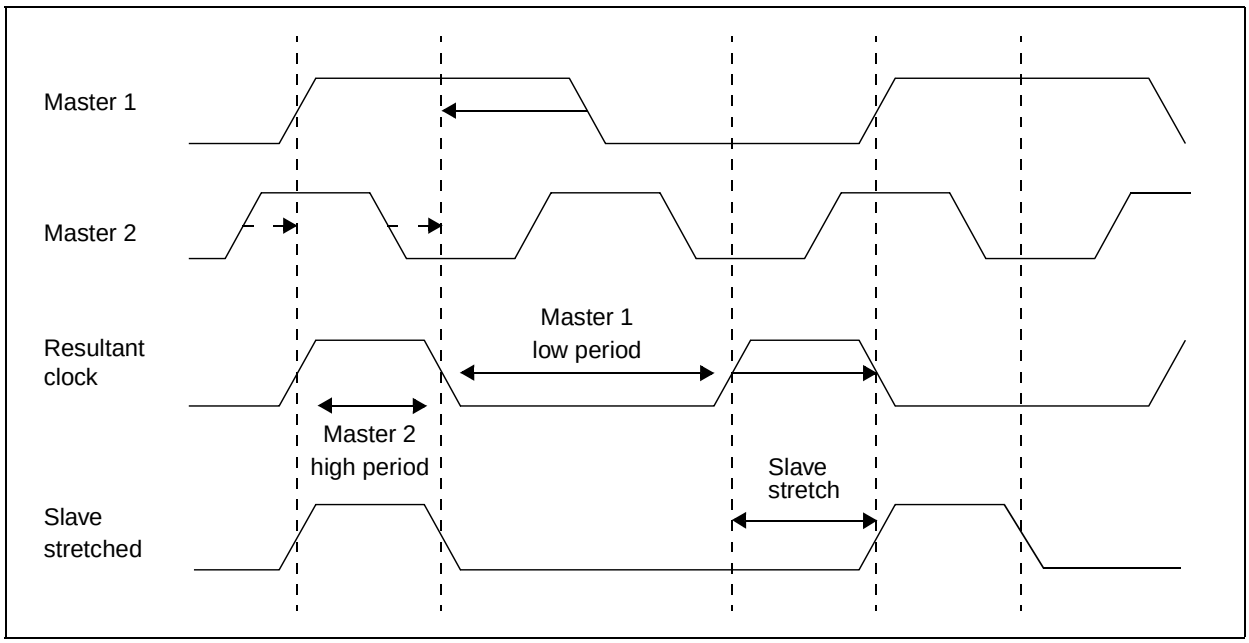
Once the external clock signal goes high, all the master devices begin counting off their high clock pulse. In this case the first master to finish counting attempts to pull the external clock line low and wins (because of the open drain line). The other master devices detect this and abort their high 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 189.

Figure 189: Synchronization and stretching



The SSC implements this clock synchronization mechanism when the I^2C control bit I2CM, is enabled.

For a slave device the output clock is maintained at logic high for the master clock to propagate on the line. However when a clock stretching condition arises on it, it can force a logic low on the serial clock line and hold it low until the clock stretch is removed.

48.3.3 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 2 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 48.3.8: *Arbitration checking* on page 499).

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 hold time and the STOP setup time are programmed. To detect the start and stop times the SSC compares the setup and hold times with the programmed values.

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 status bit is cleared when the status register is read.

48.3.4 Slave address comparison

After a START condition has been detected, the SSC goes into the address comparison phase. If the address does not match, then the SSC ignores further data until a STOP condition is detected.

7-bit addressing

In 7-bit addressing the SSC 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).

10-bit addressing

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.

If the address does not match, then the SSC ignores further data until a STOP condition is detected.

10-bit addressing with repeated start

If the slave has been addressed in the first frame of transmission (with full 2 bytes of address) with the read/write bit LOW (for a write), then the master device needs to send only one address byte in the next frame of transmission to address it for a read operation (read/write bit HIGH).

The SSC only acknowledges it if it has previously been addressed and a STOP condition has not yet occurred. 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.

If the address does not match, then the SSC ignores further data until a STOP condition is detected.

48.3.5 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 48.3.2: *Clock synchronization* on page 496). 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.

- By default the SSC stretches the clock on transmit underrun. In this case software must write into the transmit buffer to release the clock stretch. 0x1FF is written if the SSC is receiving data. The SSC can be programmed to enable clock stretch on the receive FIFO/buffer full condition (by setting bit 13 in the SSCnCON register to 1). Then 0x1FF need not be written into the transmit buffer to disable clock stretch, but the receive buffer must be read instead.
- When the SSC is a master device transmitting to a slave and it receives a NACK condition the clock is stretched after the current byte with no acknowledge until the STOP or a repeated START generation bit is set on it.

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.

48.3.6 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 for a minimum specified period.

The START/STOP generator is controlled by the START condition generate bit STRTG and the STOP condition generate bit STOPG in register SSCn I^2C.

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 which is loaded with a value programmed with the appropriate timing register. See Section 48.3.1: *I2C control* on page 493.

48.3.7 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 or restart for a new 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.

48.3.8 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. Arbitration to decide 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.

49 Synchronous serial controller (SSC) registers

Addresses are provided as the *SSCnBaseAddress* + offset.

The *SSCnBaseAddresses* are:

SSC0: 0x2084 0000,

SSC1: 0x2084 1000.

Table 111: Synchronous serial controller (SSC) registers

Register name	Description	Address offset	Type
Control and configuration registers			
SSCnBRG	SSCn baudrate generation, see page 502	0x000	R/W
SSCnCON	SSCn control, see page 503	0x00C	R/W
SSCnSLAD	SSCn slave address, see page 503	0x01C	WO
SSC status and interrupt registers			
SSCnIEN	SSC interrupt enable, see page 504	0x010	R/W
SSCnSTAT	SSCn status, see page 505	0x014	RO
CLEAR_STATUS_SSC	SSC clear bit operation, see page 506	0x080	WO
I^2C registers			
SSCnI2C	SSC I^2C control, see page 506	0x018	R/W
REP_START_HOLD_TIME	Repeated start hold timer, see page 507	0x020	R/W
START_HOLD_TIME	Start hold timer, see page 507	0x024	R/W
REP_START_SETUP_TIME	Repeated start setup timer, see page 507	0x028	R/W
DATA_SETUP_TIME	Data setup timer, see page 508	0x02C	R/W
STOP_SETUP_TIME	Stop setup timer, see page 508	0x030	R/W
BUS_FREE_TIME	Bus free timer, see page 508	0x034	R/W
Buffer and FIFO registers			
SSCnTBUF	SSCn transmit buffer, see page 509	0x004	WO
SSCnRBUF	SSCn receive buffer, see page 508	0x008	RO
SSCTXFSTAT	Transmit FIFO status, see page 509	0x038	RO
SSCRXFSTAT	Receive FIFO status, see page 509	0x03C	RO
Noise suppression filter registers			
NOISE_SUPPRESS_WIDTH_SSC	Noise suppression width, see page 510	0x100	R/W
NOISE_SUPPRESS_WIDTH_DATAOUT	Programming maximum delay, see page 510	0x108	R/W
Prescaler registers			
PRE_SCALER_BRG	BRG clock prescaler, see page 511	0x040	R/W
PRE_SCALER_SSC	Glitch width suppression, see page 510	0x104	R/W
PRE_SCALER_DATAOUT	Programming prescaler for delay in dataout, see page 511	0x10C	R/W

49.1 Control and configuration registers

SSCnBRG SSC n baudrate generation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRG															

Address: *SSCnBaseAddress* + 0x000

Type: Write only

Reset: 0

Description: This address is dual purpose. When reading, the current 16-bit counter value is returned. When a value is 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.

SSCnCON SSC n control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CLSTRX	RXFIFO	TXFIFO	LPB	EN	MS	SR	PO	PH	HB	BM			

Address: *SSCnBaseAddress* + 0x00C

Type: Read/write

Reset: 0

Description:

[15:14] **Reserved**

[13] **CLSTRX**: enable clock stretch mechanism for receiving devices
0: Enabled by transmitter 1: Enabled by receiver

[12] **RXFIFO**: enable receive side FIFO
0: FIFO bypassed 1: FIFO enabled

[11] **TXFIFO**: enable transmit side FIFO
0: FIFO bypassed 1: FIFO enabled

[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

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* + 0x01C

Type: Write only

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].

49.2 SSC status and interrupt registers

SSCnIEN SSCn 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																	RHFIEN	Reserved	THEIEN	REPSTRTEN	NACKEN	Reserved	ARBLEN	STOPEN	AASEN	Reserved	PEEN	REEN	TEEN	TIEN	RIEN

Address: *SSCnBaseAddress* + 0x010

Type: Read/write

Reset: 0

Description: This register holds the interrupt enable bits, which can be used to mask the interrupts.

[31:15] **Reserved**

[14] **RHFIEN**: Receive FIFO half full interrupt enable
1: Interrupt enabled

[13] **Reserved**

[12] **THEIEN**: Transmit FIFO half empty interrupt enable
1: interrupt enable

[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	RXHF	Reserved	TXHE	REPSTRT	NACK	BUSY	ARBL	STOP	AAS	CLST	PE	RE	TE	TIR	RIR

Address: *SSCnBaseAddress* + 0x014

Type: Read only

Reset: 2, that is, all active bits clear except TIR.

Description:

[15] **Reserved**

[14] **RXHF:**
Receive FIFO half full
1: Receive FIFO is half full

[13] **Reserved**

[12] **TXHE:** Transmit FIFO half empty
1: Transmit FIFO is half empty

[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

CLEAR_STATUS_SSC SSC clear bit operation

31:13	12	11	10	9	8	7	6	5:0
Reserved		CIR_REPSTRT	CIR_NACK	Reserved	CIR_SSCARBL	CIR_SSCSTOP	CIR_SSCAAS	Reserved

Address: *SSCnBaseAddress* + 0x080

Type: Write only

Reset: 0

Description:

[31:12] **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**

49.3 I²C registers

SSCnI2C SSC n I²C control

31:12	11	10:6	5	4	3	2	1	0
Reserved	REPSTRTG	Reserved	TXENB	AD10	ACKG	STOPG	STRTG	I2CM

Address: *SSCnBaseAddress* + 0x018

Type: Read/write

Reset: 0

Description: To suit I²C specifications, bits PH and PO of register SSC_nCON must also be set to 1.

[31:12] **Reserved**

[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

REP_START_HOLD_TIME Repeated start hold timer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP_START_HOLD_TIME															

Address: *SSCnBaseAddress* + 0x020
Type: Read/write
Reset: 1
Description: The value in this register corresponds to the I²C repeated start hold time requirement where:
repeated start hold time = clock_period * REP_START_HOLD_TIME.

START_HOLD_TIME Start hold timer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START_HOLD_TIME															

Address: *SSCnBaseAddress* + 0x024
Type: Read/write
Reset: 1
Description: The value in this register corresponds to the I²C start hold time requirement where:
start hold time = clock_period * START_HOLD_TIME.

REP_START_SETUP_TIME Repeated start setup timer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP_START_SETUP_TIME															

Address: *SSCnBaseAddress* + 0x028
Type: Read/write
Reset: 1
Description: The value in this register corresponds to the I²C repeated start setup time requirement where:
repeated start setup time = clock_period * REP_START_SETUP_TIME.

DATA_SETUP_TIME Data setup timer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_SETUP_TIME															

Address: *SSCnBaseAddress* + 0x02C
Type: Read/write
Reset: 1
Description: The value in this register corresponds to the I^2C data setup time requirement where: data setup time = clock_period * DATA_SETUP_TIME.

STOP_SETUP_TIME Stop setup timer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP_SETUP_TIME															

Address: *SSCnBaseAddress* + 0x030
Type: Read/write
Reset: 1
Description: The value in this register corresponds to the I^2C stop setup time requirement where: stop setup time = clock_period * STOP_SETUP_TIME.

BUS_FREE_TIME Bus free timer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUS_FREE_TIME															

Address: *SSCnBaseAddress* + 0x034
Type: Read/write
Reset: 1
Description: The value in this register corresponds to the I^2C bus free time requirement where: bus free time = clock_period * BUS_FREE_TIME.

49.4 Buffer and FIFO registers

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* + 0x008
Type: Read only
Reset: 0
Description: Receive buffer data RD15 to RD0.

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* + 0x004
Type: Write only
Reset: 0
Description: Transmit buffer data TD15 to TD0.

SSCTXFSTAT Transmit FIFO status

7	6	5	4	3	2	1	0
Reserved					SSCTXFSTAT		

Address: *SSCnBaseAddress* + 0x038
Type: Read only
Reset: 0
Description: This register gives the status of the transmit FIFO.

[7:3] **Reserved**

[2:0] **SSCTXFSTAT**: transmit FIFO status

000: FIFO empty	001: 1 word in FIFO
010: 2 words in FIFO	011: 3 words in FIFO
100: 4 words in FIFO	101: 5 words in FIFO
110: 6 words in FIFO	111: 7 words in FIFO

SSCRXFSTAT Receive FIFO status

7	6	5	4	3	2	1	0
Reserved					SSCTXFSTAT		

Address: *SSCnBaseAddress* + 0x03C
Type: Read only
Reset: 0
Description: This register gives the status of the receive FIFO.

[7:3] **Reserved**

[2:0] **SSCTXFSTAT**: receive FIFO status

000: FIFO empty	001: 1 byte in FIFO
010: 2 bytes in FIFO	011: 3 bytes in FIFO
100: 4 bytes in FIFO	101: 5 bytes in FIFO
110: 6 bytes in FIFO	111: 7 bytes in FIFO

49.5 Noise suppression filter registers

NOISE_SUPPRESS_WIDTH_SSC Noise suppression width

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

Address: *SSCnBaseAddress* + 0x100

Type: Read/write

Reset: 0

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.

NOISE_SUPPRESS_WIDTH_DATAOUT Programming maximum delay

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

Address: *SSCnBaseAddress* + 0x108

Type: Write only

Reset: 0

Description: The NOISE_SUPPRESS_WIDTH_DATAOUT register holds the maximum delay for the output data.

49.6 Prescaler registers

PRE_SCALER_SSC Glitch width suppression

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

Address: *SSCnBaseAddress* + 0x104

Type: Read/write

Reset: 0

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.

PRE_SCALER_BRG BRG clock prescaler

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESCALEVAL															

Address: *SSCnBaseAddress* + 0x040

Type: Read/write

Reset: 1

Description: The value in this register is used to further prescale the clock generated according to the programming of the baudrate. It can be used in conjunction with the programming in the SSCnBRG register to slow down the serial clock generated in order to operate at lower frequencies.

PRE_SCALER_DATAOUT Programming prescaler for delay in dataout

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

Address: *SSCnBaseAddress* + 0x10C

Type: Write only

Reset: 0

Description: This register holds the prescaler division value.

50 Programmable I/O port

50.1 Overview

The COMMS block contains one 7-bit and three 8-bit PIO blocks. These general purpose bocks are memory mapped and each bit can be configured as input, output or bidirectional. The driver may be configured as push-pull or weak pull up.

In addition any input may be compared to a stored value and used to trigger an interrupt.

Any pin may be configured as an alternative function, in this mode the pad may be routed to or from another peripheral. The alternative pin map is configured in the STx5119 by linking the appropriate COMMS bristles.

Alternative functions are detailed in Section 6.2: *Alternative functions (mapped to GPIO pins)* on page 32.

The PIO ports can be controlled by registers, mapped into the device address space. The registers for each port are grouped in a 4 Kbyte block, with the base of the block for port n at the address *PIO*n*BaseAddress*. During reset all of the registers are reset to zero.

Each PIO port has a set of registers. Each bit of a register refers to a 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_RESET_x registers clear bits individually. Writing 1 in these registers resets a corresponding bit in the associated register x; 0 leaves the bit unchanged.

51 Programmable I/O port 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 the *PIOnBaseAddress* + offset.

The *PIOnBaseAddresses* are:

PIO0: 0x2082 0000,
PIO1: 0x2082 1000,
PIO2: 0x2082 2000,
PIO3: 0x2082 3000.

Table 112: Programmable I/O port registers

Register name	Description	Address offset	Type
PIO_PnOUT	PIO output, see page 517	0x00	R/W
PIO_SET_PnOUT	Set bits of PIO_PnOUT, see page 518	0x04	WO
PIO_RESET_PnOUT	Clear bits of PIO_PnOUT, see page 514	0x08	WO
PIO_PnIN	PIO input, see page 516	0x10	RO
PIO_PnC0	PIO configuration, see page 515	0x20	R/W
PIO_SET_PnC0	Set bits of PIO_PnC0, see page 517	0x24	WO
PIO_RESET_PnC0	Clear bits of PIO_PnC0, see page 514	0x28	WO
PIO_PnC1	PIO configuration, see page 515	0x30	R/W
PIO_SET_PnC1	Set bits of PIO_PnC1, see page 517	0x34	WO
PIO_RESET_PnC1	Clear bits of PIO_PnC1, see page 514	0x38	WO
PIO_PnC2	PIO configuration, see page 515	0x40	R/W
PIO_SET_PnC2	Set bits of PIO_PnC2, see page 517	0x44	WO
PIO_RESET_PnC2	Clear bits of PIO_PnC2, see page 514	0x48	WO
PIO_PnCOMP	PIO input comparison, see page 516	0x50	R/W
PIO_SET_PnCOMP	Set bits of PIO_PnCOMP, see page 517	0x54	WO
PIO_RESET_PnCOMP	Clear bits of PnCOMP, see page 514	0x58	WO
PIO_PnMASK	PIO input comparison mask, see page 516	0x60	R/W
PIO_SET_PnMASK	Set bits of PIO_PnMASK, see page 517	0x64	WO
PIO_RESET_PnMASK	Clear bits of PnMASK, see page 514	0x68	WO

PIO_RESET_PnC[2:0] Clear bits of PnC[2:0]

	7	6	5	4	3	2	1	0
0x28	RESET_PC0[7:0]							
0x38	RESET_PC1[7:0]							
0x48	RESET_PC2[7:0]							

Address: *PIOnBaseAddress* + 0x28 (PIO_RESET_PnC0), 0x38 (PIO_RESET_PnC1), 0x48 (PIO_RESET_PnC2)

Type: Write only

Description: PIO_RESET_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_RESET_PnCOMP Clear bits of PnCOMP

7	6	5	4	3	2	1	0
RESET_PCOMP[7:0]							

Address: *PIOnBaseAddress* + 0x58

Type: Write only

Description: PIO_RESET_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_RESET_PnMASK Clear bits of PnMASK

7	6	5	4	3	2	1	0
RESET_PMASK[7:0]							

Address: *PIOnBaseAddress* + 0x68

Type: Write only

Description: PIO_RESET_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_RESET_PnOUT Clear bits of PnOUT

7	6	5	4	3	2	1	0
RESET_POUT[7:0]							

Address: *PIOnBaseAddress* + 0x08

Type: Write only

Description: PIO_RESET_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: Read/write

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 113.

Table 113: 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: Read/write

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_RESET_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: Read only

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: Read/write

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_RESET_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: Read/write

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_RESET_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: Write only

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: Write only

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: Write only

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: Write only

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.

52 Electrical specifications

52.1 Absolute maximum ratings

Table 114: Absolute maximum ratings

Symbol	Parameter	Min	Max	Units
$VDD33_{MAX}$	DC supply voltage		6.4	V
VDD_{MAX}	DC supply voltage		2.1	V
$VI5_{MAX}$	DC voltage on input, output and bidirectional 5 V tolerant pins	GND - 0.5[1]	5.5[2]	V
$VO3_{MAX}$	DC voltage on input, output and bidirectional 3.3 V tolerant pins	GND - 0.5[1]	VDD + 0.5[3]	V
IO_{MAX}	DC output current		20	mA
TS_{MAX}	Storage temperature (ambient)	-55	125	°C
TA_{MAX}	Temperature under bias (ambient)	-55	125	°C

1. AC transient undershoot of 0.7 V below GND -0.5 V (that is -1.2 V) is allowed for a maximum period of 2 ns.
2. AC transient overshoot of 0.7 V above 5.5 V is allowed for a maximum period of 2 ns.
3. AC transient overshoot of 0.7 V above VDD33 + 0.5 V is allowed for a maximum period of 2 ns.

Note: Stresses greater than those listed in Table 114 may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operating sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

52.2 Operating conditions

Table 115: Operating conditions

Symbol	Parameter	Min	Typical	Max	Units	Notes
VI	Input voltage (3.3 V I/O only)	0		3.6	V	1
VI35	Input voltage (3.3 V / 5 V tolerant I/O)	0		5.5	V	2
C_L	Load capacitance per pin			100	pF	
TA	Operating temperature (ambient)	0		70	°C	
PD	Maximum power dissipation at VDD33 = 3.6 V and VDD = 1.32 V			1.35	W	
PD_{LP}	Power dissipation (low power mode)			900	mW	

1. Excursions beyond the supplies are permitted but not recommended. AC transient overshoot and undershoot are permitted up to the limits in Table 114.
2. AC transient overshoot and undershoot is permitted up to the limits in Table 114.

52.3 Standard pads DC specifications

The standard pads concern all digital interfaces except LMI interfaces.

Table 116: Standard pads DC specifications

Symbol	Parameter	Min	Typical	Max	Units	Notes
VDD	Positive core supply voltage	1.08	1.2	1.32	V	
VDD33	Positive I/O supply voltage	3.0	3.3	3.6	V	
V_{IH}	Input logic 1 voltage (3.3 V I/O only)	2.0			V	
V_{IH5}	Input logic 1 voltage (5 V tolerant I/O only)	2.0		5.5	V	
V_{IL}	Input logic 0 voltage			0.8	V	
I_{IN}	Input current (input pin)			±5	µA	1
I_{OZ}	Off state digital output current			±50	µA	
IWP_U	Input weak pull-up current			110	µA	
IWP_D	Input weak pull-down current			110	µA	
V_{OH}	Output logic 1 voltage	2.4			V	2
V_{OL}	Output logic 0 voltage			0.4	V	3
C_{IN}	Input capacitance (input pins)			10	pF	
C_{IO}	Input capacitance (bidirectional pins)			15	pF	
C_{OUT}	Output capacitance			15	pF	
V_{HYS}	Hysteresis voltage (3 V/5 V tolerant I/O)	0.495		0.620	V	4

1. $0 \leq VI \leq VDD33$
2. Iload = 4 mA for all outputs.
3. Iload = 4 mA for all outputs.
4. for all standard pads which are TTL compatible except for I^2C pads which have hysteresis in the range 0.310 to 0.380 V.

52.4 LMI pads DC/AC specifications for SDR-based systems

The LMI pads are compliant to the LVTTL specifications.

Table 117: LMI pads AC/DC specifications

Symbol	Parameter	Condition	Min	Typical	Max	Units
VDD33LMI	Positive I/O supply voltage		3.0	3.3	3.6	V
V_{IL}	DC input logic low				0.8	
V_{IH}	DC input logic high		2			V
V_{OH}	Output logic 1 voltage	VDD33LMI = 3 V I_{out} = -12 mA	2			V
		VDD33LMI = 3 V I_{out} = -2 mA	2.4			
		VDD33LMI = 3V I_{out} = -100 µA	2.8			
V_{OL}	Output logic 0 voltage	VDD33LMI = 3 V I_{out} = 12 mA			0.45	V
		VDD33LMI = 3V I_{out} = 2 mA			0.4	
		VDD33LMI = 3V I_{out} = 100 µA			0.2	
C_{out}	Output capacitance				35	pF
Frequency	Operational frequency				166	MHz
	Output rise slew rate	Between 0.8 and 2 V	1			V/ns
	Output fall slew rate	Between 0.8 and 2 V	1			V/ns
	Output rise time	Between 0.8 and 2 V			1.2	ns
	Output fall time	Between 0.8 and 2 V			1.2	ns

52.5 Targeted power consumption

Table 118: Targeted power consumption

Power	Consumption
VDD 1.2 V	350 mA
VDD33LMI 3.3V	100 mA
VDD 3.3V	20 mA to 120 mA[1]
VDDDAC (total) V	40 mA
VDDAADA V	30 mA

1. The value depends on the number of peripherals and PIO used.

52.6 Audio DAC specifications

Table 119: Audio DAC supply specifications

Symbol	Parameter	Min	Typ	Max	Units	Notes
VDDAADAC	DC analog supply voltage	3.0	3.3	3.6	V	

52.6.1 Audio DAC parameters

Parameter	Typical	Unit
SNR	94 dB at -3 dBFS 114 dB at -60 dBFS	dB
Dynamic Range [1]	-95	dB
Signal to {THD + Noise}	85	dB
Signal to Total Harmonics Distortion	-92	dB
Noise Floor [2]	-108	dBv
Interchannel Isolation	-80 at 1 kHz	dB
Interchannel Gain Mismatch	0.1	dB
Out of band noise	-70	dB

1. The specified value is obtained by adding 60 dB to the measured THD+N at full scale -60dB, with A-Weighting Filter.
2. The specified value is obtained when the digital code is “00H”

52.7 Video DAC parameters

Table 120: Video DAC parameters

Parameter	Min	Typ	Max	Unit
Differential non linearity (DNL)	-1	--	1	LSB
Integral non linearity (INL)	-2	--	2	LSB
Dynamic current[1]	7	8	9	mA
Output voltage (max)	--	--	1.6	V DC
Analogue 3dB bandwidth	6	--	--	MHz
Rise and Fall time	--	--	10	ns
DAC to DAC matching[2]	--	+/- 1	+2	%
Conversion rate	27	--	--	MHz
SNR, BW=10MHz, Fclk=80MHz	--	50	--	dB
THD, Fout=5MHz, Fclk=80MHz	--	-50	--	dB
(Vbandgap)	-4%	1.23	+4%	V
Cross-talk				
Gain variation	--	10	--	%
Syncro/blanking/white levels				

1. The output current of 8mA is with typ bandgap voltage (1.23V) and RREF = 10Kohms
2. These figures can be significantly improved (overall variation ~3%) by using the trimming control in the DENC. The gain control has a resolution of 0.75%

Table 121: Video DAC PSRR figures

Parameter	3.3v DAC RGB/YCC	3.3v Video PLL
Power supply rejection ratio	60db	70dB

53 Timing specification

This chapter contains I/O specifications for the following interfaces:

- system (reset, CLK27, low power mode) on page 523,
- 16-bit LMI (local memory) on page 525,
- 16-bit FMI (flash and peripherals) on page 530,
- transport (PTI) incorporating multiplexed D1 input and YUV out on page 533,
- TAP on page 534,
- PIO port, including digital I/O on page 536,
- digital I/O:
  - (DMA requests on page 536,
  - asynchronous serial controller on page 536,
  - synchronous serial controller on page 537,
  - infrared receiver/transmitter on page 538,
- general power supply requirements on page 538.

Load capacitance on the tester differs according to the way measurements are taken. In standard mode it is evaluated to approximately 50 pF whereas for high speed measurements with 50 Ω termination the figure is around 15 pF.

The default drive strength setting used for all characterization and simulation work is 25 pF.

53.1 System

53.1.1 Reset timing

The $t_{RSTHRSTL}$ value in the following figure and table is for power-on reset when the device is cold.

Figure 190: Reset timings

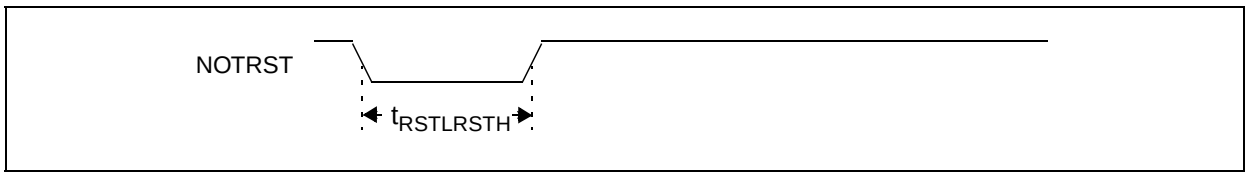


Table 122: Reset timings figures

Symbol	Parameter	Min	Nom	Max	Units
$t_{RSTLRSTH}$	NOTRST pulse width low[1]	10	-	-	ms

1. This is to allow for crystal oscillator start-up times which can be long.

53.1.2 Watchdog reset out

Figure 191: NOT_WDOGRSTOUT timings

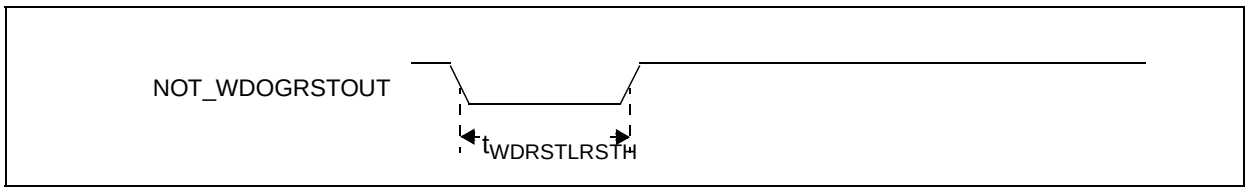


Table 123: NOT_WDOGRSTOUT figures

Symbol	Parameter	Min	Nom	Max	Units
$t_{WDRSTLRSTH}$	NOT_WDRSTOUT pulse width low	10	-	NOT_RESET + 10	ms

53.1.3 Low power mode timings

Table 124: LPCLKIN requirements

	Min	Max
Clock frequency	1/6 of the communications clock	50 kHz

Note: The LPM clock is also available as an output, as an alternative function on PIO2[5].

53.1.4 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 VCO is the option recommended by STMicroelectronics.

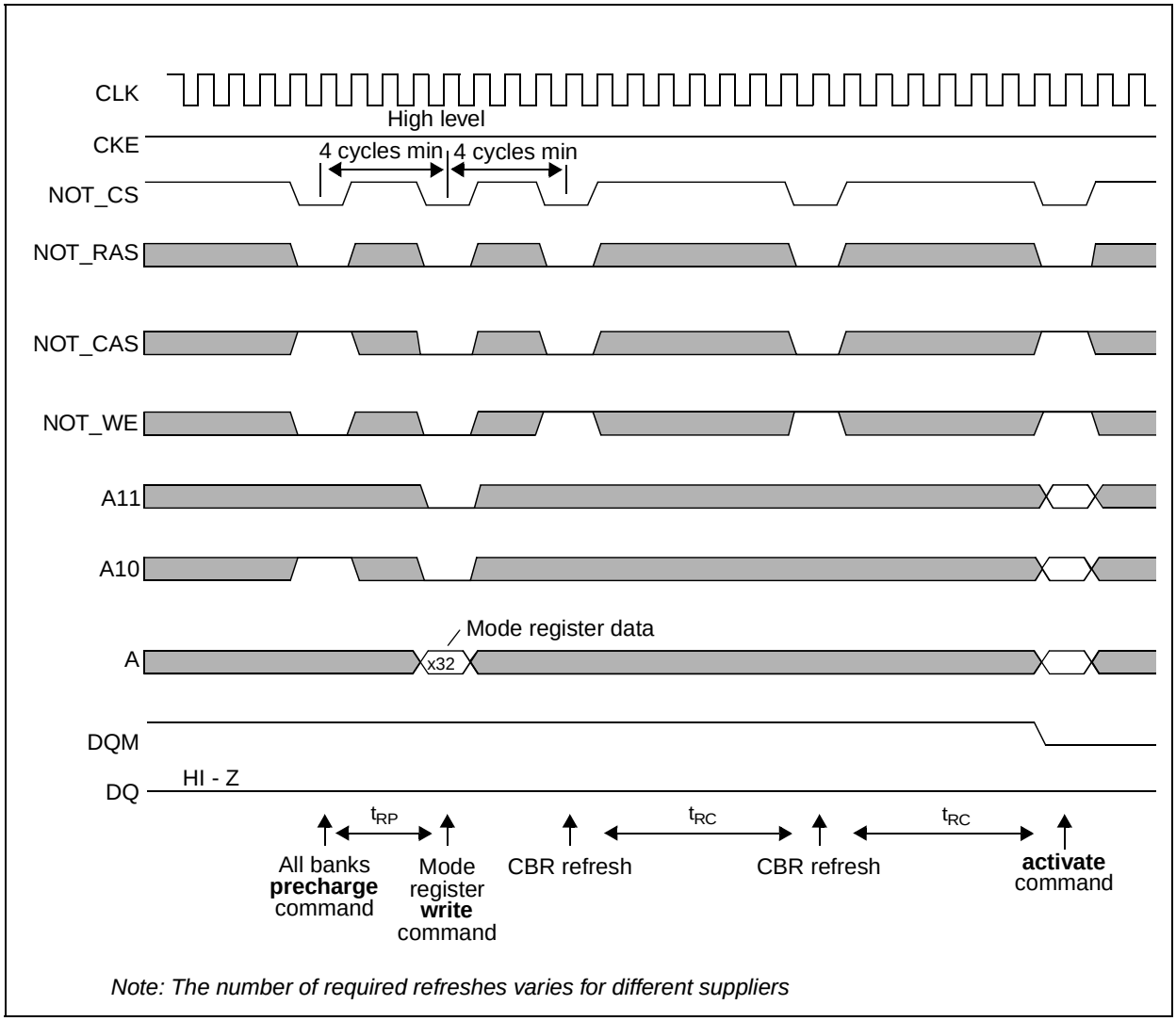
Crystal requirements

The external 27 MHz crystal has the following requirements:

- 27.000 MHz fundamental frequency,
- parallel resonance,
- 40 ppm accuracy.

53.2 LMI (SDRAM) interface specifications

Figure 192: Synchronous DRAM power-on sequence



Note: The number of required refreshes varies for different suppliers

Figure 193: AC parameters for read (synchronous DRAM)

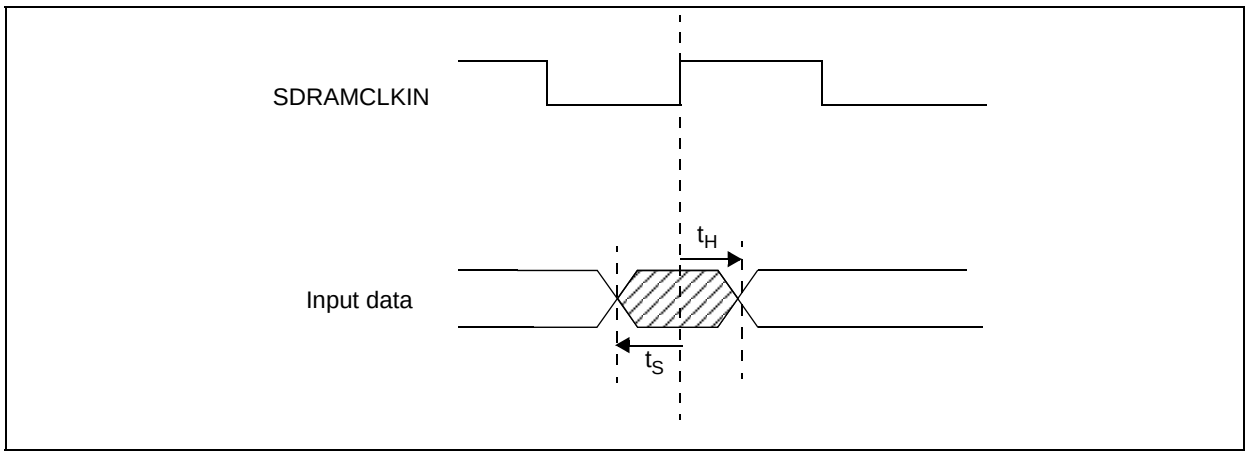


Figure 194: AC parameters for write (synchronous DRAM)

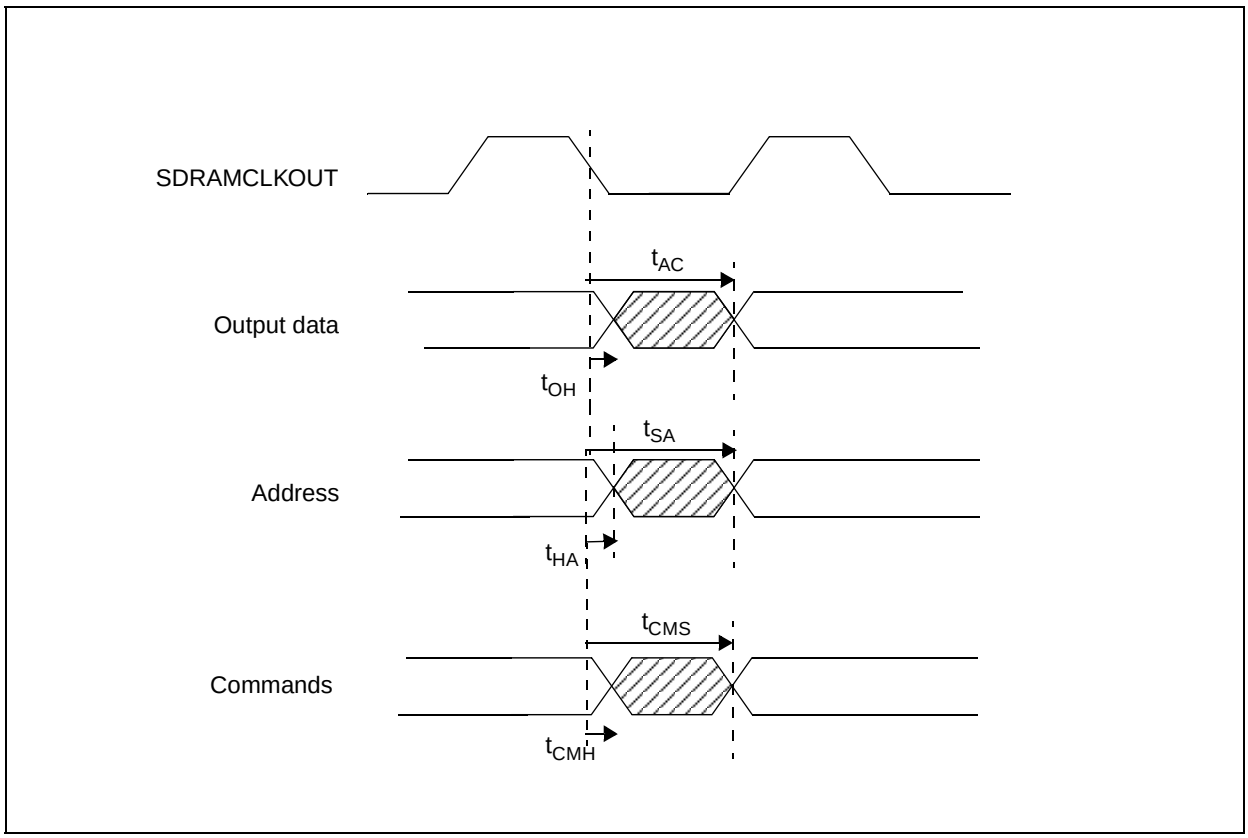
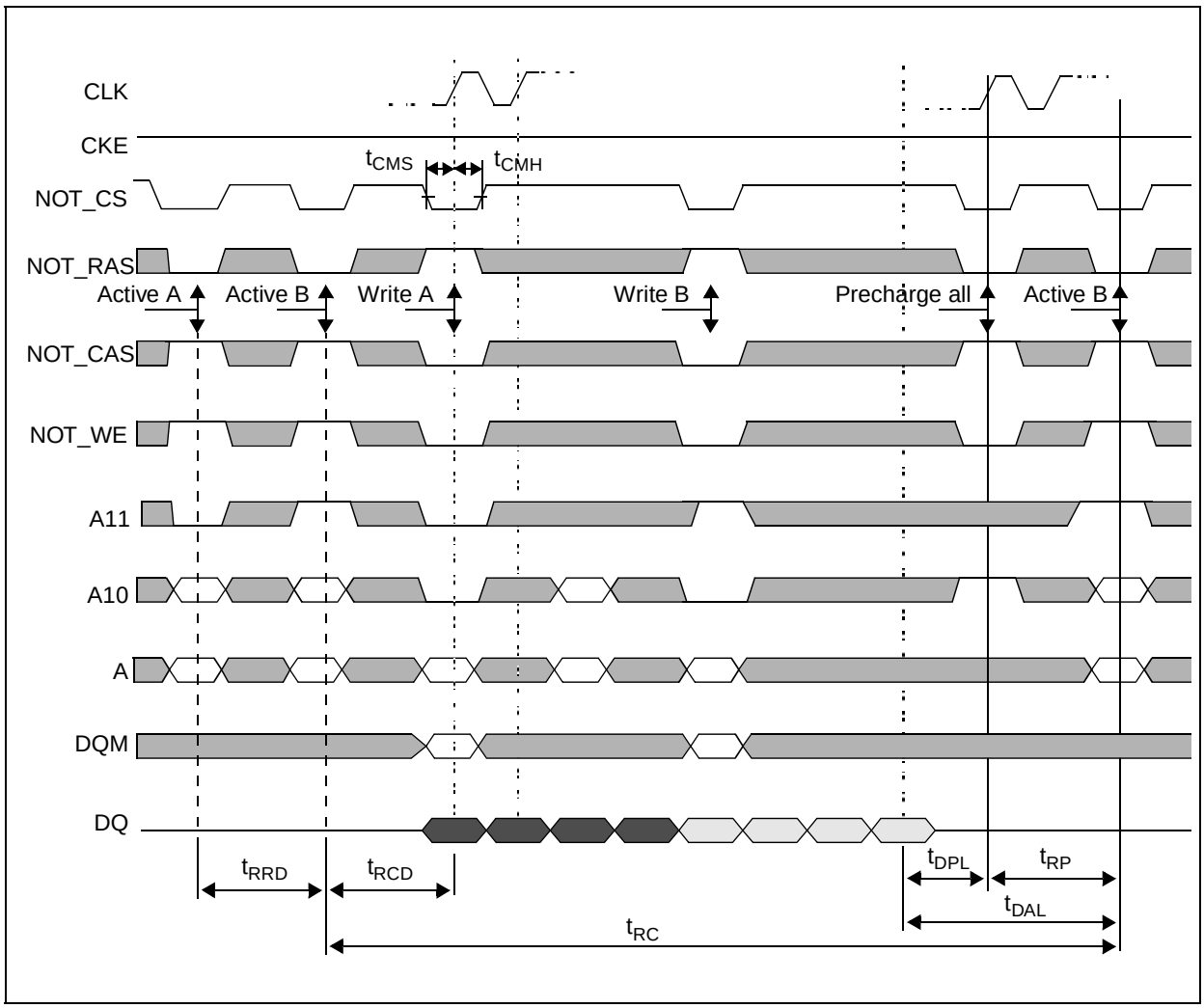


Figure 195: Synchronous DRAM write (burst length = 4, CAS latency = 3)



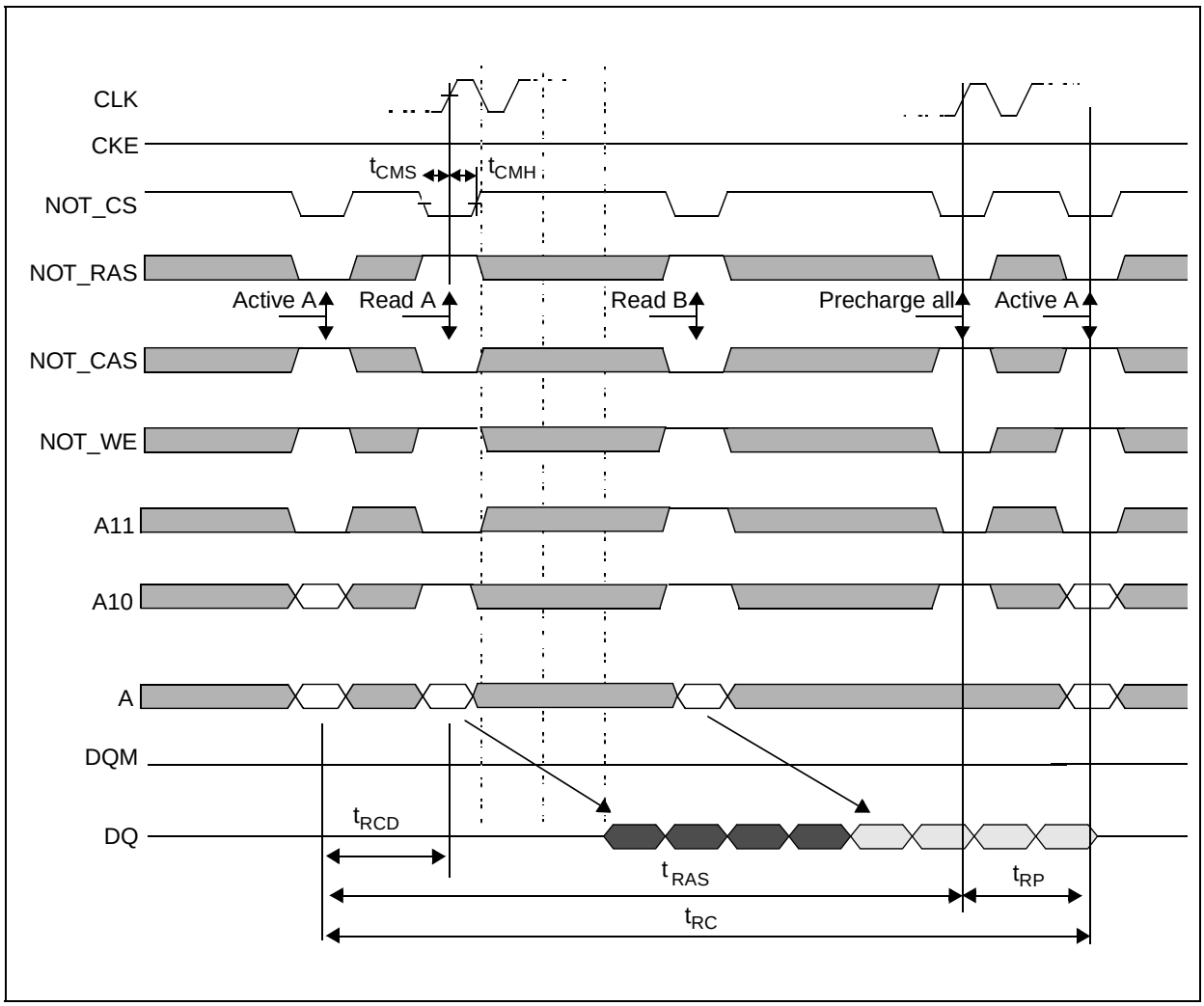
In Table 125, the unit T is the period of the memory subsystem clock (typically 8.23 ns / 121.5 MHz).

Table 125: Synchronous DRAM read and write

Symbol	Parameter	Min	Max	Units	Notes
t_{RP}	**active** to **pre** command period	3		T	
t_{RC}	**ref** to **ref/active** command period	12		T	
t_{CK}	Clock cycle time	7.4		ns	1
t_{CH}	Clock high level width			ns	2
t_{CL}	Clock low level width			ns	b
t_{S}	Data input setup time	1.9		ns	
t_{H}	Data input hold time	1		ns	
t_{AC}	Output data access time		2.0	ns	
t_{OH}	Output data hold time	-2.0		ns	3
t_{SA}	Address access time		2.0	ns	
t_{HA}	Address hold time	-2.0		ns	c
t_{CMS}	Command (NOT_CS, NOT_RAS, NOT_CAS, NOT_WE, NOT_DQM) access time		2.0	ns	
t_{CMH}	Command (NOT_CS, NOT_RAS, NOT_CAS, NOT_WE, NOT_DQM) hold time	-2.0		ns	c
t_{RCD}	Delay time **active** to **read/write** command	4		T	
t_{RRD}	**active(a**) to **active(b)** command period	4		T	
t_{DAL}	**data-out** to **active** command period	5		T	
t_{DPL}	**data-out** to **precharge** command period	2		T	
t_{RAS}	**active** to **precharge** command period	9		T	

1. Maximum clock rate is 135 MHz.
2. A 50% duty cycle is recommended.
3. Negative values mean that the measured event is before the falling edge of MEMCLKOUT.

Figure 196: Synchronous DRAM read (burst length = 4, CAS latency = 3)



53.3 FMI

The FMI supports synchronous accesses to SFlash™ and also access to asynchronous peripherals.

All figures quoted in this section are under the default configuration described at the beginning of this chapter.

53.3.1 Synchronous devices

Outputs are generated and inputs sampled with respect to the rising edge of the bus clock. The static timing characteristics for all FMI pins are shown in Figure 197 and Table 126. These values are static offsets within a bus clock cycle.

Figure 197: FMI interface timings for synchronous transactions

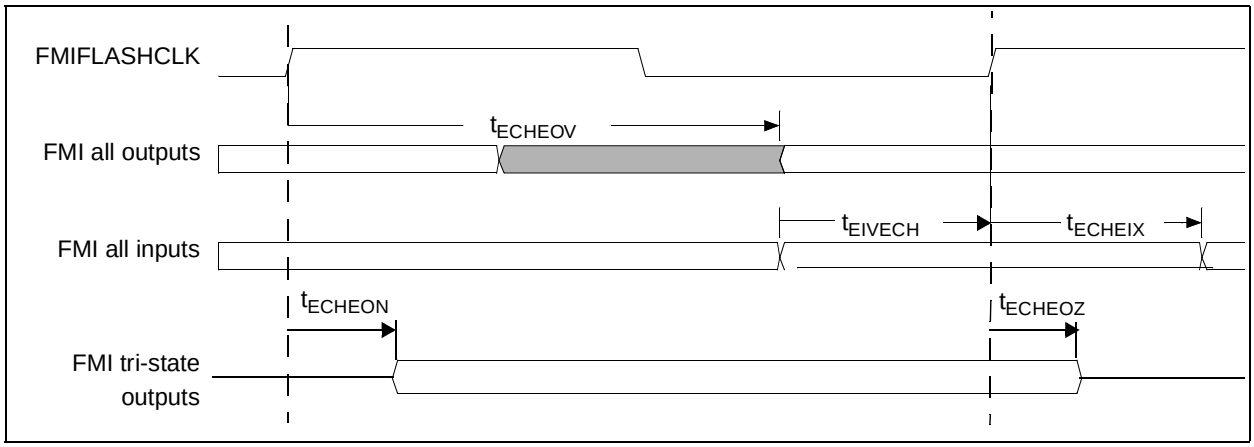


Table 126: FMI synchronous interface parameters

Output values assume a 25 pF external load			
Symbol	**Description**	**Min (ns)**	**Max (ns)**
t_{ECHEOV}	Bus clock rising edge to valid output data	1.0	3.0
t_{EIVECH}	Input valid to rising clock edge (input set-up time)	4.0	-
t_{ECHEIX}	Rising clock edge to input invalid (input hold time)	-1.0	-
t_{ECHEON}	Rising clock edge to tri-state outputs	2	-
t_{ECHEOZ}	Rising clock edge to outputs tri-state	-	4

Table 126 assumes an external load of 25 pF. If larger loads are used t_{ECHEOV} must be derated accordingly, as in Figure 198.

Bus cycle time is programmed as a division (1, 2 or 3) of the FMI subsystem clock frequency. When configured as a clock master, the STx5119 FMI may be programmed over a range of frequencies up to 108 MHz.

53.3.2 Asynchronous memory/peripherals

The FMI 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 closely matched, with a tolerance of ±0.3 ns using equal loads (nominal 25 pF). Any additional skew is dependent on the external load spread. Increasing the load delays the signal. The signals are delayed as in Figure 198.

The input latchpoint for a read access is determined by the number of programmed FMI subsystem clock cycles for the latchpoint. The correction in Figure 198 allows the latchpoint to be measured from the active strobe, assuming the strobe has a 25 pF external load. For larger loads adjust according to the derating graph in Figure 198.

For example if all signals are equally loaded with 25 pF, the time between the address bus switching and a strobe switching is only *n* programmed phases ±0.3 ns. For a write cycle, the time between the address bus switching and the data bus switching is *n* programmed phases of data drive delay ±0.3 ns. For a read cycle the data is latched by the STx5119 at the programmed number of FMI subsystem clock cycles from the end of the access plus a latch point correction time. The latch point correction time is 3 ns + load correction of the output signal used as a reference. In this case this is 3 ±0.3 ns. This ensures that the read hold time is always a minimum of 0 ns, guaranteed by design.

Figure 198: Asynchronous access: read

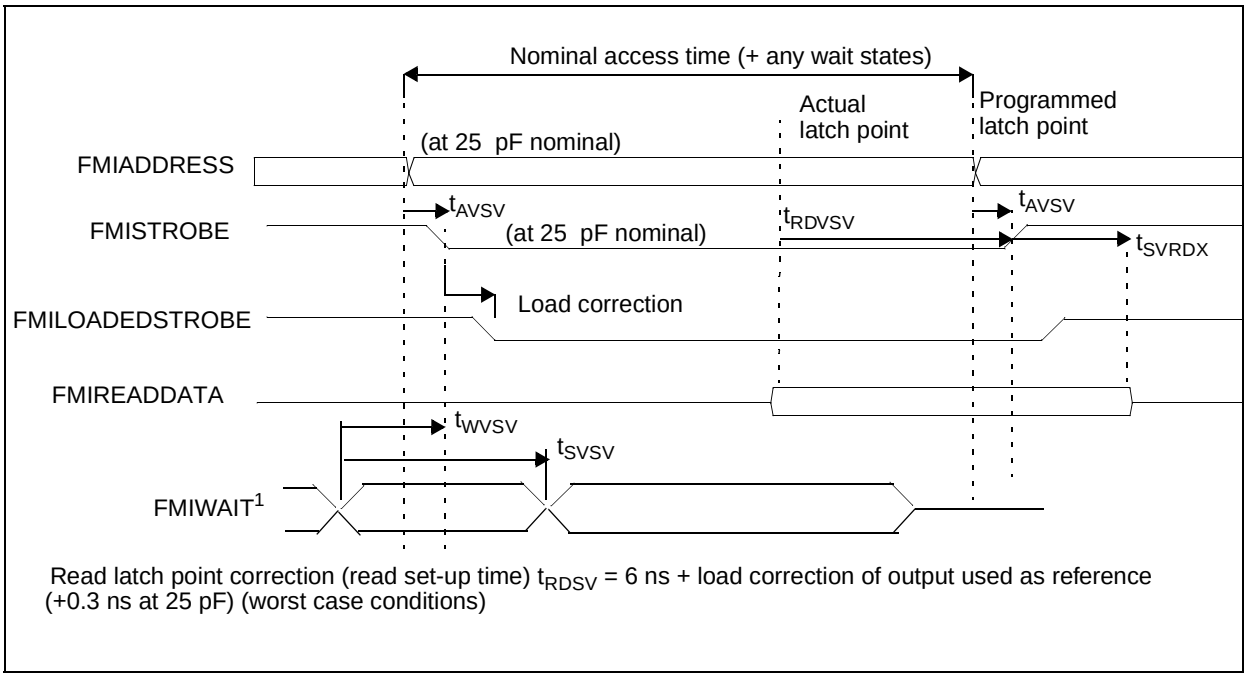


Figure 199: Asynchronous access: write

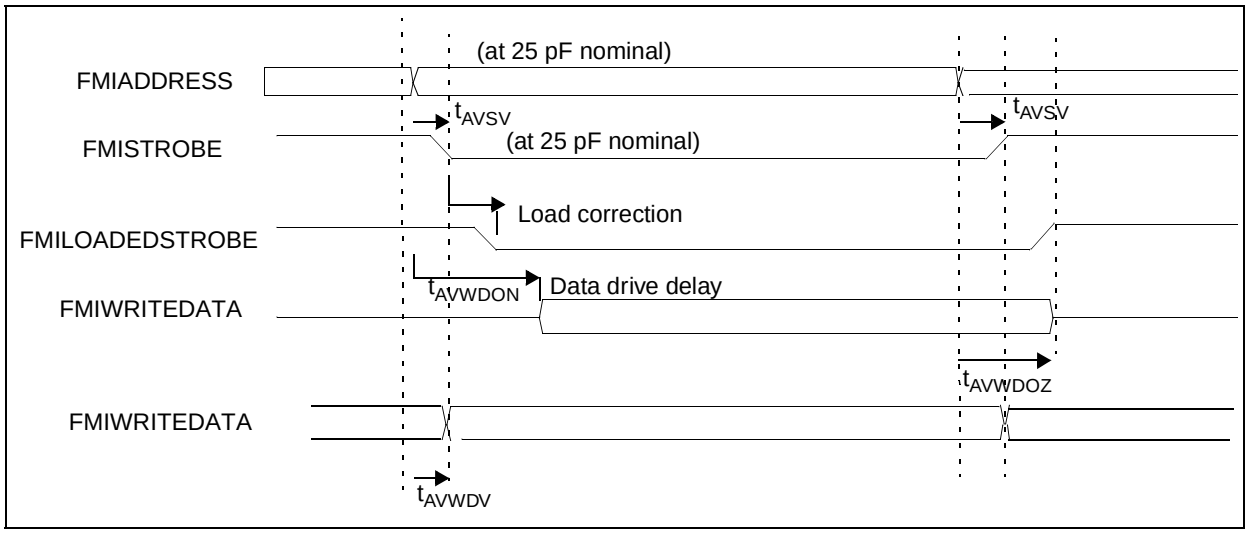


Table 127: FMI asynchronous access mode timings

Values assume a 25 pF external load				
Symbol	**Description**	**Min (ns)**	**Max (ns)**	**Notes**
t_{AVSV}	Address valid to output strobe valid	-0.3	0.3	1,2
t_{RDVSV}	Read data valid to strobe valid (read set-up time)	6.3	-	3
t_{SVRDX}	Read data hold after strobe valid (read hold time)	0	-	4
t_{WVSV}	Wait valid to strobe valid (wait set-up time)	6.3	-	5,6
t_{WVWV}	Minimum wait valid time (before state change)	>1T	-	6
t_{AVWDON}	Address valid to write data valid from tri-state	-	2.3	7
t_{AVWDOZ}	Address valid to write data valid to tri-state	-	4.3	
t_{AVWDV}	Address valid to write data valid from on state	-0.3	0.3	8

1. Skew plus nominal N programmed FMI subsystem clock cycles of strobe delay.
2. Example: if load on address is 25 pF and load on strobe is 50 pF, tAVSV is -0.3 + 0.8 ns minimum, 0.3 + 0.8 ns maximum = 0.5 ns minimum, 1.1 ns maximum.

   If the load on the address is 50 pF and the load on the strobe is 25 pF, tAVSV is -0.3 ns - 0.8 ns minimum, 0.3 ns -0.8 ns maximum = -1.1 ns minimum, -0.5 ns maximum.
3. Skew from nominal programmed read latch point.
4. Minimum values are guaranteed by design.
5. Use an output strobe programmed to fall on the rising edge of the FMI subsystem clock, as a reference. FMIWAIT is sampled at the end of the first and subsequent clock cycles of the access, except for the penultimate cycle.
6. FMIWAIT is synchronized by the FMI pad logic using the FMI 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 FMI subsystem clock cycle, where T is the FMI subsystem clock period.
7. Skew from nominal programmed phases of data drive delay.
8. No data drive delay.

53.4 Transport stream timings

Each live input can tolerate 27 Mbyte/s in parallel mode and 100 Mbit/s in serial mode.

In serial mode, the data is arriving on TSINDATA[7] only. See Table 12: *Transport stream 0 pins* on page 39 for alternative functions.

Table 128: Maximum bit/byte clock frequencies supported by TSIS

Transport steam format	Max bit/byte clock frequency	Period
Parallel	27 MHz	37 ns
Serial	100 MHz	10 ns

Figure 200: Parallel transport stream input timings (27 Mbyte/s input rate)

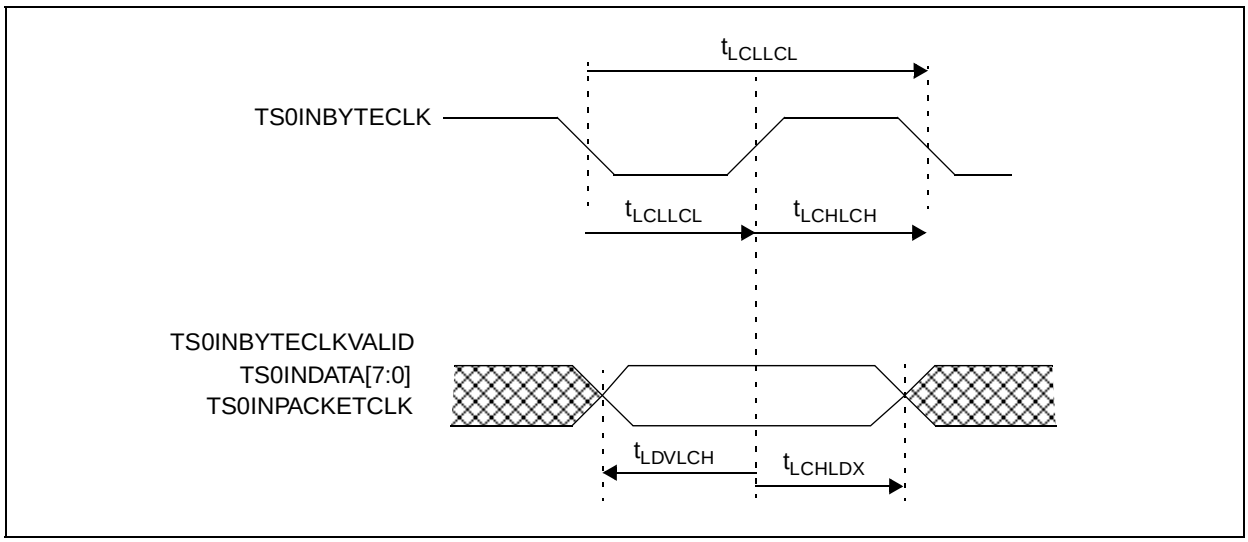


Table 129: TSIN timings: parallel input mode

Symbol	Parameter	Min	Nom	Max	Units
t_{LCLLCL}	TS0INBYTECLK period	37	-	-	ns
t_{LCHLCH}	TS0INBYTECLK pulse width high	10	-	-	ns
t_{LCLLCL}	TS0INBYTECLK pulse width low	10	-	-	ns
t_{LDVLCH}	TS0IN signals valid to TS0INBYTECLK high	10	-	-	ns
t_{LCHLDX}	TS0IN signals hold after TS0INBYTECLK high	0	-	-	ns

53.5 TAP timings

Figure 201: TAP timings

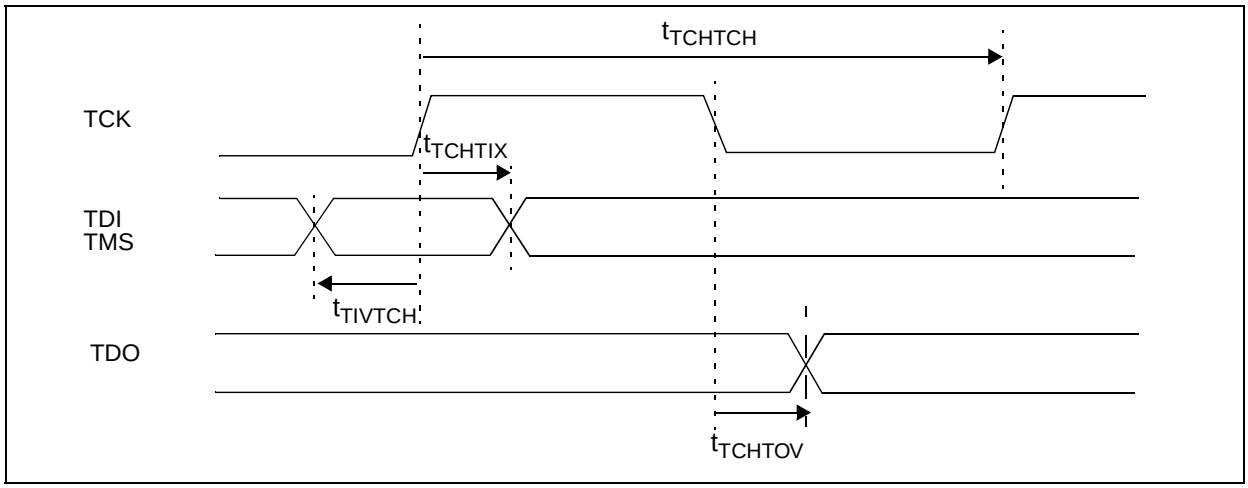


Table 130: TAP timings

Symbol	Parameter	Min	Nom	Max	Units
t_{TCHTCH}	TCK period	20	-	-	ns
t_{TIVTCH}	TAP inputs set-up time[1]	4	-	-	ns
t_{TCHTIX}	TAP input hold time[1]	4	-	-	ns
t_{TCHTOV}	TCK low to TAP output valid	-	-	20	ns

1. Set-up and hold time should be equivalent to 30% of the given TCK period.

There are three other DCU signals, NOT_TRST, DCUTRIGGERIN and DCUTRIGGEROUT. The latter two have dedicated pins on the STx5119. These signals can be considered asynchronous and therefore have no timing constraints. DCUTRIGGERIN need only be asserted long enough for the CPU to sample it.

53.6 Audio

53.6.1 S/PDIF

The S/PDIF output is Manchester encoded and hence self-clocking. There are no requirements on this signal relative to other chip signals.

53.6.2 PCM decoder output interface

The PCM decoder output interface can be clocked off both edges.

Figure 202: PCM data output

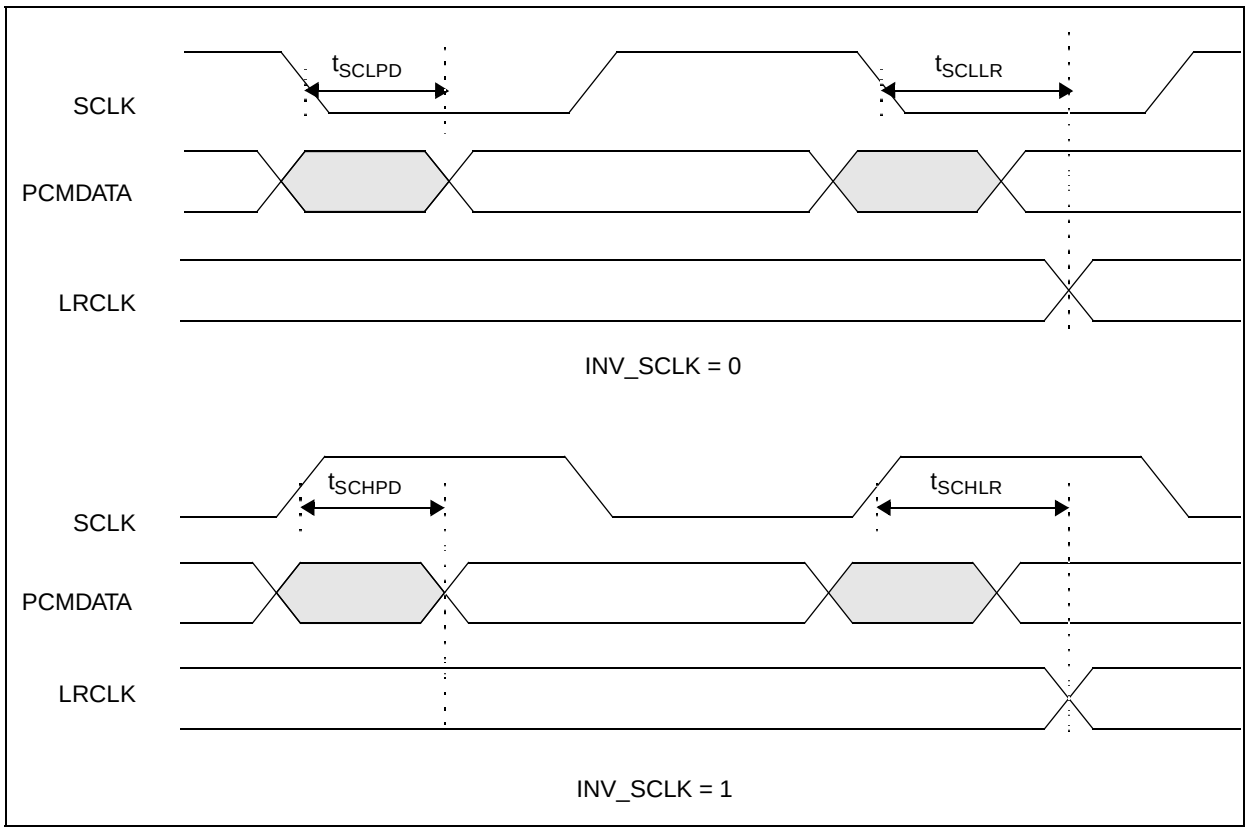


Table 131: PCM data output

Symbol	Parameter	Min	Max	Units
t_{SCLPD}	SCLK low to PCMDATA valid	-	10	ns
t_{SCLLR}	SCLK low to LRCLK	-	10	ns
t_{SCHPD}	SCLK high to PCMDATA valid	-	50	ns
t_{SCHLR}	SCLK high to LRCLK	-	50	ns

53.7 General PIO

Figure 203: General PIO timings

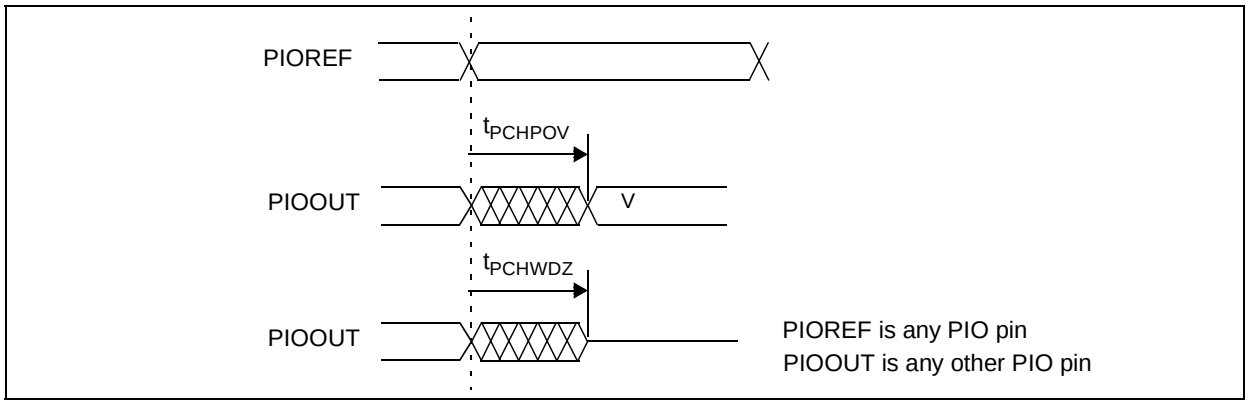


Table 132: General PIO timing parameters

Symbol	Parameter	Min	Max	Units
t_{PCHPOV}	PIOREF transition to PIO output valid	-20.0	0.0	ns
t_{PCHWDZ}	PIO tri-state after PIOREF transition	-20.0	5.0	ns

53.8 Digital I/O

53.8.1 DMA requests

A pair of request signals for DMA (DMAREQ[1:0]). These are used mostly by the FDMA and are asynchronous. Note that INTERRUPT2 and INTERRUPT3 are the alternative function of these pins.

53.8.2 ASC

There are two UARTs. These are, by definition, asynchronous and have no skew or timing constraints. The mapping of the UART pins is shown in Table 18: *Asynchronous serial controllers pin mapping* on page 43.

53.8.3 SSC

There are two SSC interfaces. The maximum bit rate for SPI is 10 Mbit/s.The SSC supporting SPI can be configured to sample on either the rising or falling edge. Data is sampled using a majority scheme based on communications clock sampling, therefore timing constraints are defined in periods of the communications clock. The pin mapping is shown in Table 20: *Synchronous serial controller pin mapping* on page 44.

Figure 204: SSC clock relationship waveform

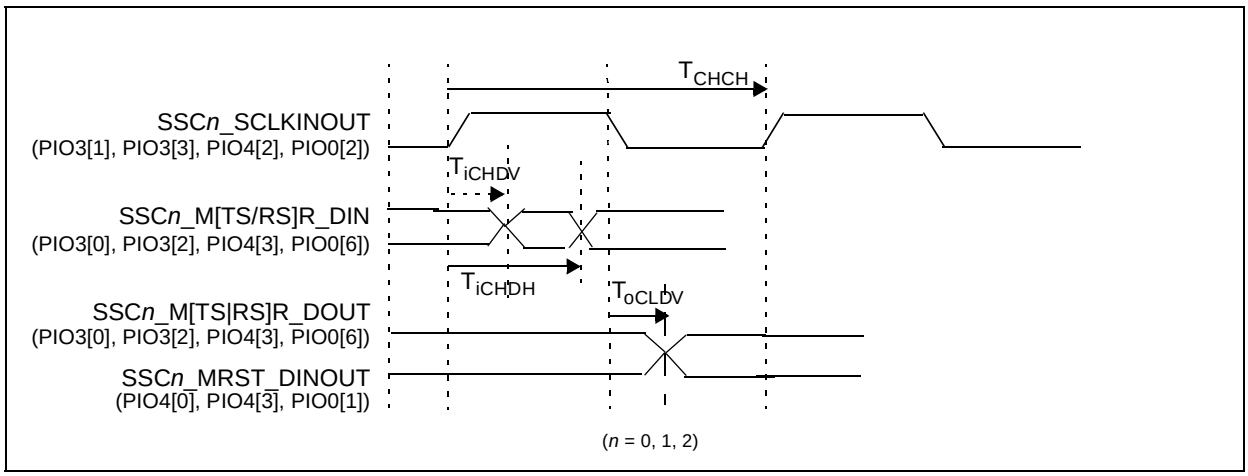


Table 133: SSC timing parameters

Symbol	Parameter	Min[1]	Nom	Max[1]	Units
t_{CHCH}	Input clock period (rising-rising)	-	200	-	ns
t_{iCHDV}	Required time elapsed from rising (capture) clock edge to input data stable.	-	-	2T	ns
t_{iCHDH}	Required time elapsed from rising (capture) clock edge for input data to hold.	6T + 10	-	-	ns
t_{oCLDV}	Required time elapsed from falling (launch) clock to output data stable.	2T	-	3T + 10	ns

1. T is 1 period communications clock, 16.7 ns for standard 60 MHz communications frequency.

Figure 205: SSC I²C clock relationship waveform

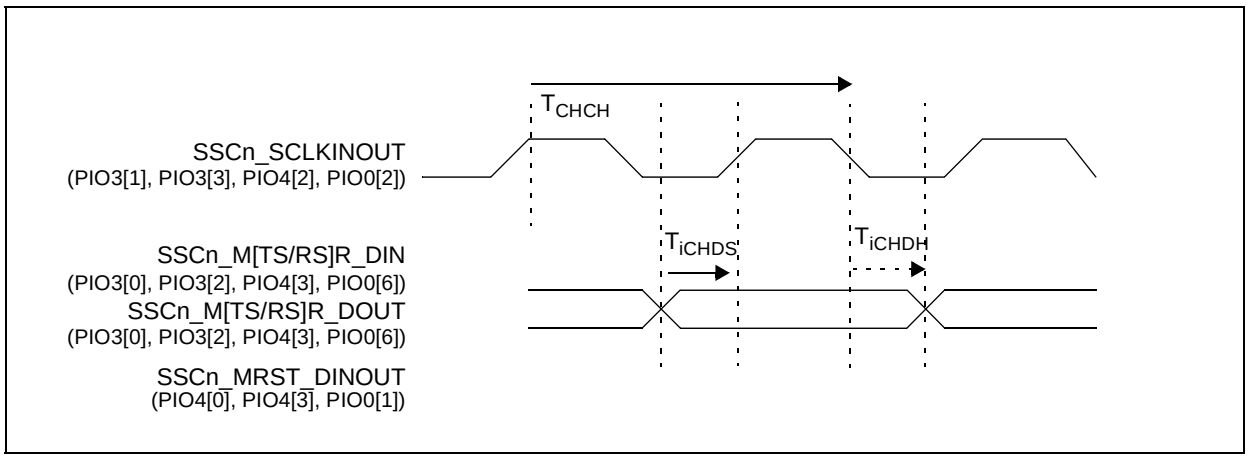


Table 134: SSC I²C timing parameters

Symbol	Parameter	High speed			Fast mode			STD mode			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
t_{CHCH}	Input clock period	589	-	-	2500	-	-	10000	-	-	ns
t_{iCHDS}	Data[1] set-up time	10	-	-	100	-	-	250	-	-	ns
t_{iCHDH}[1]	Data hold time	0[2]	-	150	300	-	900	300	-	3450	ns
T_{CL}	SCLK low period	320	-	-	1300	-	-	4700	-	-	ns
T_{CH}	SCLK high period	120	-	-	600	-	-	4000	-	-	ns
TrTf[3]	SDA/SCLK rise/ fall	20	-	80	20+ $0.1C_b$[4]	-	300	-	-	1000	ns

1. The data is launched during the low period of the clock but captured during the high period of the clock. There is no clock inversion.
2. A device must internally provide a Data hold time to bridge the undefined part between VIH and VIL of the falling edge of the SCLK signal.
3. Both SDA and SCLK are considered data signals and are synchronized w.r.t the communications clock. All parameters are fully programmable in H9 IP block.
4. Capacitive load C_b, for clock and data bus is 400 pF maximum for I²C. (10 pF per I²C device) Other devices, such as a serial EEPROM, have a typical value of 8 pF.

53.8.4 IRB

There are two inputs and two outputs. No constraints on timing or skew. Mapping is shown in Table 22: *Miscellaneous* on page 46.

53.9 General power supplies

Table 135: Power supply requirements

Supply	Pin name	Comments
3.3 V I/O supply	VDD33	± 10%
1.2 V core supply	VDD12	± 10%
3.3 V LMI supply	VDD33LMI	± 10%
3.3 V audio DAC supplies	VDDAADAC	60 mV ripple on 5 V[1]
3.3 V video DAC RGB	VDDVDACRGB	30 mV ripple
3.3 V video DAC YCC	VDDVDACYCC	30 mV ripple
1.2 V frequency synthesizers	VCCAFS	30 mV ripple
1.2 V system PLLs	VCCAPLL1 VCCAPLL2	50 mV ripple 50 mV ripple

1. Figures are from simulation data.

Note: Only VDD is shown in the table. Where a supply has a particular ripple requirement it is assumed that the corresponding VSS/GND pin is subject to the same.

53.9.1 Signal overshoot and undershoot.

Maximum undershoot is -1.2 V for 2 ns maximum. Maximum overshoot is 4.3 V for 2 ns maximum (VDD33).

54 Index of registers

Confidential