

Low-cost set-top box decoder

DATASHEET

■ Enhanced ST20 32-bit VL-RISC CPU

- 180 MHz, 8 Kbyte instruction cache, 8 Kbyte data cache and 8 Kbyte SRAM

■ Shared memory interface

- 135 MHz, 16-bit wide SDRAM interface, 64 and 128 Mbit support

■ Programmable external memory interface

- 6 separately configurable banks, 8/16-bits wide
- SRAM, SDRAM, SFlash™ support
- PIO mode HDD or DVB-CI support

■ Programmable transport interfaces (PTI)

- 2 input static MUX
- single transport stream deMUX: DVB and/or DIRECTV®
- integrated DES-ECB, DVB and ICAM descramblers
- support for low cost DVB-CI interface

■ Package 35 x 35 PBGA388

■ MPEG2 MP@ML video decoder

- greater than 2x decoding speed
- trick modes including smooth fast-forward and rewind
- fully programmable horizontal and vertical SRCs

■ Graphics/display

- 5 display planes
- 2, 4 and 8 bpp CLUT graphics, 256 x 30 bits (AYCbCr) CLUT entries. Link list control
- alpha blending, antialiasing, antiflicker, antiflicker filters

- 2-D paced BLT engine with fill function

- subpicture decoder

- display compositor with separate OSD control for TV and VCR outputs

■ PAL/NTSC/SECAM encoder

- RGB, CVBS, Y/C and YUV outputs with 10-bit DACs
- encoding of CGMS, Teletext, WSS, VPS, closed caption

■ Audio subsystem

- MPEG-1 layers I/II decoding (MP3 option)
- Dolby® Digital 5.1 decoding downmixed to 2-channels
- Dolby® Pro Logic® compatible output
- PCM mixing and sample rate conversion
- SRS®/TruSurround® virtual surround sound
- IEC958/IEC1937 digital audio output interface
- integrated stereo audio DAC system

■ On-chip peripherals

- 5 ASCs (UARTs) with Tx and Rx FIFOs
- 6 x 8-bit banks of parallel I/O
- 2 smartcard interfaces and clock generators
- 2 SSCs for I²C master/slave interface
- 4 PWM channels
- teletext serializer and DMA module
- multi-channel infrared transmitter/receiver
- modem analog front-end interface
- IEEE1284 interface
- low-power / RTC / watchdog controller

■ JTAG/TAP interface

Figure 1: STi5516 block diagram

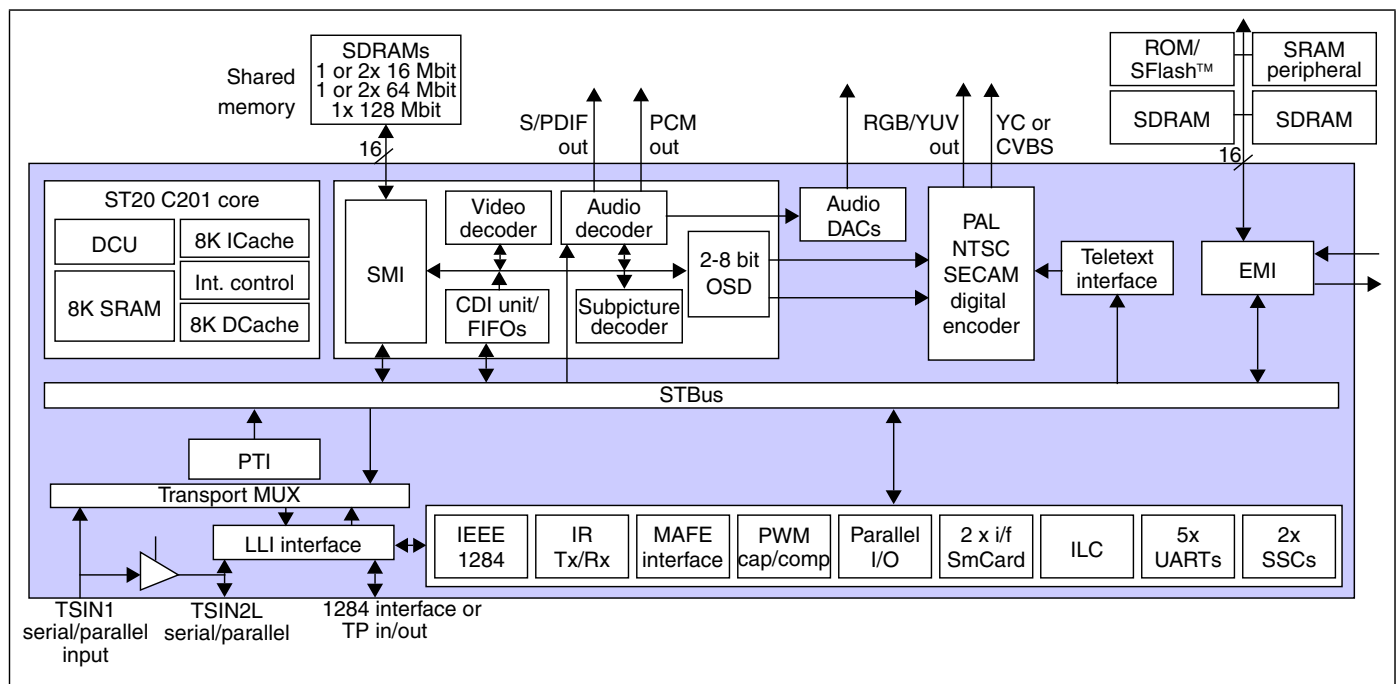


Table of contents

Chapter 1	Introduction	14
Chapter 2	Architecture	15
2.1	Overview	15
2.2	Omega2 (STBus) interconnect	16
2.3	Processor core	16
2.4	Memory subsystem	16
2.5	Transport stream processing	17
2.6	LLI interface	18
2.7	MPEG graphics and display architecture	19
2.8	Graphics and display	20
2.9	Digital encoder	21
2.10	Audio subsystem	21
2.11	Modem	22
2.12	Internal peripherals	23
2.13	EMI programmable output drive	23
2.14	Clock generation	23
2.15	Smartcard interface	23
Chapter 3	Audio and video summary specification	24
3.1	Functional limitations	24
3.2	Summary specification	24
Chapter 4	Pin list	26
4.1	Pin-out	26
4.2	STi5516 pin list	28
4.3	PIO pins and alternative functions	36
4.4	Reset states	40
Chapter 5	Package specifications	43
Chapter 6	Register base addresses	46

Confidential

Chapter 7	Register summary	48
Chapter 8	Central processing unit (CPU)	81
8.1	Overview	81
8.2	Registers used in sequential integer processes	81
8.3	Processes and concurrency	82
8.4	Priority	83
8.5	Process communications	84
8.6	Timers	84
8.7	Traps and exceptions	86
Chapter 9	Central processing unit (CPU) registers	91
9.1	Machine registers	91
9.2	Other machine registers	91
9.3	Register details	93
Chapter 10	Instruction set	97
10.1	Overview	97
10.2	Instruction cycles	97
10.3	Instruction characteristics	98
10.4	Instruction set tables	99
Chapter 11	Interrupt system	110
11.1	Overview	110
11.2	Interrupt controller	111
11.3	Interrupt level controller	114
11.4	Interrupt assignments	115
Chapter 12	Interrupt system registers	117
12.1	Interrupt level controller registers	123
Chapter 13	Memory map	128

Chapter 14	Memory	131
14.1	External memory	131
14.2	On-chip SRAM memory	131
14.3	Cacheing	132
Chapter 15	Memory registers	135
Chapter 16	External memory interface (EMI)	141
16.1	Overview	141
16.2	Operation	142
16.3	Default/reset configuration	144
16.4	Peripheral interface with synchronous flash memory support	145
16.5	SDRAM interface	151
Chapter 17	External memory interface (EMI) registers	168
17.1	Configuration register format for peripherals	173
17.2	Configuration register format for SDRAM	177
Chapter 18	Padlogic	180
18.1	Overview	180
18.2	EMI padlogic	180
18.3	TRI_PTI MUXing	191
18.4	CDREQ MUXing	192
Chapter 19	Padlogic registers	193
19.1	EMI general purpose configuration outputs	193
19.2	Monitor registers	194
19.3	Configuration registers	194
Chapter 20	EMI buffer	199
Chapter 21	EMI buffer registers	200

Chapter 22	System services	204
22.1	Overview	204
22.2	Power-on hard reset	204
22.3	Bootstrap	204
Chapter 23	Diagnostic controller (DCU)	205
23.1	Overview	205
23.2	Diagnostic hardware	205
23.3	Access features	206
23.4	Software debugging features	207
23.5	Controlling the diagnostic controller	208
23.6	Peeking and poking the host from the target	210
Chapter 24	Diagnostic controller (DCU) registers	211
24.1	General registers	211
24.2	Jump trace registers	218
24.3	Compare registers	221
24.4	Capture registers	223
24.5	Sequencing registers	224
24.6	Work space range enable registers	225
Chapter 25	Test access port	227
Chapter 26	Test access port registers	228
Chapter 27	Data flow	229
27.1	Overview	229
27.2	Audio: PCM mixing	229
27.3	Video: standard decode	230
Chapter 28	Programmable transport interface (PTI)	232
28.1	Overview	232
28.2	PTI functions	234
28.3	PTI architecture	235
28.4	PTI operation	238
28.5	Interrupt handling	240

28.6	DMA operation	242
28.7	Section filter (SF)	244
28.8	Compatibility with PTI1	250
Chapter 29	Programmable transport interface (PTI) registers	251
29.1	DMA registers	251
29.2	Input interface registers	259
29.3	PTI configuration registers	261
29.4	Section filter registers	265
29.5	Transport controller mode register	267
Chapter 30	Transport stream multiplexor (TSMUX)	269
30.1	Overview	269
30.2	Architecture	269
30.3	Transport stream routing	270
30.4	PTI MUXing	271
30.5	Transport output	271
30.6	Local byte clock	272
30.7	TS timing information	272
30.8	TSMUX_SWTS	272
Chapter 31	Transport stream multiplexor (TSMUX) registers	274
Chapter 32	IEEE1394 link layer interface (LLI)	278
32.1	Overview	278
32.2	Block diagram	278
32.3	Data streams	279
32.4	Pin function	280
Chapter 33	IEEE1394 link layer interface (LLI) registers	281
Chapter 34	MPEG video decoder	283
34.1	Overview	283
34.2	Decoder operation	283
34.3	Configuration and control	284
34.4	Reset	284

34.5	Bit buffer and start code detection (video)	285
34.6	Video decoding pipeline control	287
34.7	Quantization table loading	288
34.8	Memory mapping of data	289
34.9	Using picture pointers	299
34.10	Video pipeline	299
34.11	PES parser	303
34.12	Enhanced trick modes	304
Chapter 35	MPEG video decoder registers	306
35.1	Configuration and control (CFG) register information	328
35.2	PES parser (PES) register Information	330
Chapter 36	Subpicture decoder	333
36.1	Overview	333
36.2	Buffer management and pointers	334
36.3	Subpicture decoder operation	334
36.4	Subpicture display	336
Chapter 37	Subpicture decoder registers	338
Chapter 38	Display planes	344
38.1	Overview	344
38.2	Background color	345
38.3	Still picture plane	346
38.4	MPEG video plane	348
38.5	On-screen display (OSD)	358
38.6	Subpicture or cursor plane	373
38.7	Mixing display planes	373
Chapter 39	Display planes registers	376
39.1	Background color registers	376
39.2	Still picture plane registers	377
39.3	On-screen display registers	382

Chapter 40	2-D block move	397
40.1	Overview	397
40.2	Copying blocks of data	397
Chapter 41	2-D block move registers	399
Chapter 42	Teletext DMA	402
42.1	Overview	402
42.2	Teletext packet format	402
42.3	Data transfer sequence	403
42.4	Interrupt control	403
Chapter 43	Teletext DMA registers	404
Chapter 44	Digital encoder	407
44.1	Overview	407
44.2	Video timing	407
44.3	Reset procedure	411
44.4	Slave modes	411
44.5	Input demultiplexor	416
44.6	Subcarrier generation	417
44.7	Burst insertion (PAL and NTSC)	418
44.8	Subcarrier insertion (SECAM)	419
44.9	Luminance encoding	420
44.10	Chrominance encoding	421
44.11	Composite video signal generation	423
44.12	RGB and UV encoding	425
44.13	Closed captioning	425
44.14	CGMS encoding	426
44.15	WSS encoding	427
44.16	VPS encoding	427
44.17	Teletext encoding	428
44.18	CVBS, S-VHS, RGB and UV outputs	430

Chapter 45	Digital encoder registers	432
Chapter 46	Triple video DAC	458
46.1	Overview	458
46.2	Input codes for video application	459
46.3	Video output voltage level	459
46.4	Video specifications and DAC setup	460
46.5	Output stage adaptation and amplification	460
Chapter 47	Audio decoder	461
47.1	Overview	461
47.2	Decoding process	465
47.3	Operation	466
47.4	Decoding states	467
47.5	Stream parsers	468
47.6	Decoding modes	469
47.7	PCM output	472
47.8	S/PDIF output	477
47.9	Interrupts	478
47.10	Audio/video synchronization	479
47.11	PCM beep tone	480
47.12	PCM mixing	481
47.13	VCR output	482
Chapter 48	Audio decoder registers	483
48.1	Audio DSP start up registers	483
48.2	Audio DSP version registers	484
48.3	RS232 activation registers	485
48.4	Audio DSP setup and input registers	485
48.5	PCM configuration registers	486
48.6	ADC input/second input registers	488
48.7	PCM mixing registers	493
48.8	VCR configuration registers	495
48.9	S/PDIF output setup registers	496
48.10	Audio command registers	499

48.11	Audio interrupt registers	502
48.12	Audio DSP decoding algorithm registers	509
48.13	Audio DSP system synchronization registers	510
48.14	Postdecoding and Pro Logic® registers	512
48.15	Bass redirection registers	514
48.16	Dolby® Digital configuration registers	516
48.17	MPEG configuration registers	522
48.18	LPCM registers	527
48.19	LPCM downmix coefficients	528
48.20	PCM beep tone registers	532
48.21	Pink noise register	534
48.22	General mode configuration registers	535
Chapter 49	Audio decoder interface (AUDIF)	537
49.1	Overview	537
49.2	PCM input module (PCMI)	538
49.3	PCM output module (PCMO)	538
Chapter 50	Audio decoder interface (AUDIF) registers	542
Chapter 51	Audio DAC	547
51.1	Description	547
51.2	Input signals and output pins	548
51.3	Soft mute	549
51.4	Output stage filtering	549
Chapter 52	Clock generator	551
52.1	Overview	551
52.2	Maximum clock frequencies and restrictions	552
52.3	Modes of operation	554
52.4	System clocks	556
52.5	Programmable dividers	558
52.6	PCM clock	560
52.7	Smartcard clocks	561
52.8	Auxiliary clock	561

Chapter 53	Clock generator registers	562
53.1	Programmable dividers registers	566
53.2	Shift register for SDLL_CLOCK[2:0]	568
53.3	Audio DAC, DSS smartcard, auxiliary clock registers	569
53.4	Low power mode registers	570
53.5	CPU tick timer register	572
Chapter 54	Low power module (LPM) and power-down mode	573
54.1	Power-down mode	573
54.2	Real-time counter	573
54.3	Watchdog counter	574
Chapter 55	Low power module (LPM) registers	575
Chapter 56	PWM and counter module	577
56.1	External interface	577
56.2	PWM outputs	577
56.3	Capture inputs	578
56.4	Compare (programmable timer) facilities	578
56.5	Capture/compare counter, prescaling and clocking	578
Chapter 57	PWM and counter module registers	579
Chapter 58	Modem analog front-end interface (MAFEIF)	587
58.1	Overview	587
58.2	Using the MAFEIF to connect to a modem	587
58.3	Software	588
Chapter 59	Modem analog front-end interface (MAFEIF) registers	589
Chapter 60	Infrared transmitter/receiver	593
60.1	Overview	593
60.2	Functional description	593

Chapter 61	Infrared transmitter/receiver registers	596
61.1	RC transmitter registers	596
61.2	RC receiver registers	599
61.3	Noise suppression register	602
61.4	RC and UHF receiver control	602
61.5	Reverse polarity registers	603
Chapter 62	Asynchronous serial controller (ASC)	604
62.1	Overview	604
62.2	Control	604
62.3	Data frames	605
62.4	Transmission	607
62.5	Reception	608
62.6	Baudrate generation	610
62.7	Interrupt control	612
62.8	Smartcard operation	615
Chapter 63	Asynchronous serial controller (ASC) registers	617
Chapter 64	Smartcard interface	625
64.1	Overview	625
64.2	External interface	625
64.3	Smartcard clock generator	626
Chapter 65	Smartcard interface registers	627
Chapter 66	Synchronous serial controller (SSC)	628
66.1	Overview	628
66.2	Basic operation	628
66.3	I ² C operation	636
Chapter 67	Synchronous serial controller (SSC) registers	644
Chapter 68	Parallel I/O port	651
Chapter 69	Parallel I/O port registers	652

Chapter 70	IEEE 1284 port (PC parallel port)	658
70.1	Overview	658
70.2	IEEE 1284 port pins	659
70.3	IEEE 1284 mode	660
70.4	Transport stream mode	662
Chapter 71	IEEE 1284 port (PC parallel port) registers	666
Chapter 72	Electrical specifications	679
72.1	Absolute maximum ratings	679
72.2	Operating conditions	680
72.3	DC specifications	681
72.4	Audio DAC specifications	682
72.5	Video DAC specifications	682
Chapter 73	Timing specification	683
73.1	SMI SDRAM	683
73.2	Audio PCM interface timings	688
73.3	EMI timings	689
73.4	PIO timings	693
73.5	Reset timings	694
73.6	Clock timings	695
73.7	TAP timings	696
73.8	Transport stream timings	697
73.9	Teletext timings	699
73.10	IEEE 1284 timings	700
Chapter 74	Revision history	701
Chapter 75	Index of registers	704

1 Introduction

The STi5516 is a single transport derivative of the STi5514 Omega set-top box decoder intended mainly for use in nonhard disk drive set-top boxes. However support for a simple PIO mode hard disk drive interface is included, mapped on to the external memory interface (EMI). Also included is support for a single DVB-CI port. The device is also intended as an upgrade path for customers using the STi5512. New features have been included to increase performance and reduce system cost, whilst they have been specified so that a single PCB can be designed to take either the STi5516A or STi5516 device. (See *STi5516A/STi5516 Product Preview* for differences between the STi5514, STi5516A and STi5516 devices)

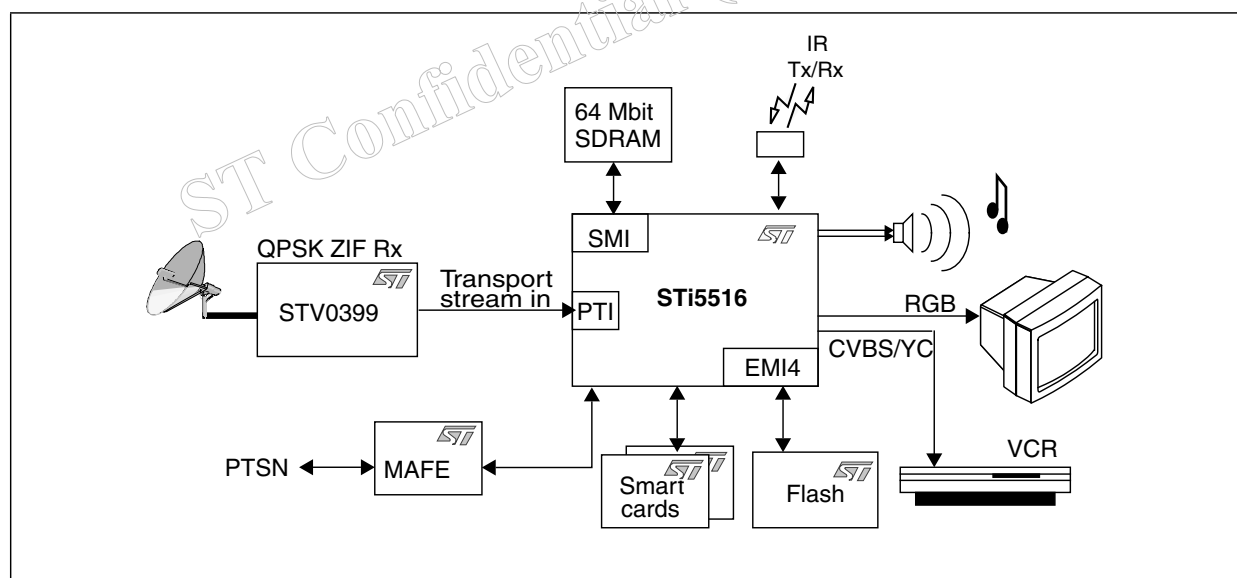
Performance is enhanced with a 180 MHz, ST20 VL-RISC CPU and increased cache sizes. This increased performance allows system cost to be reduced by implementing low speed modem processing entirely in software. In applications requiring a high speed modem, the controller code can be run on the ST20 CPU, thus reducing the cost of the external data pump.

The additional performance also allows memory usage to be optimized and unified in a single 64-Mbit or 128-Mbit SDRAM for most pay TV applications.

The STi5516 contains all of the most commonly used descramblers, and thus allows a single platform to be developed to cover the entire mainstream set-top box market.

The STi5516 is suitable for use in satellite, terrestrial and cable applications. A typical application is shown in [Figure 2](#).

Figure 2: Basic pay TV



2 Architecture

2.1 Overview

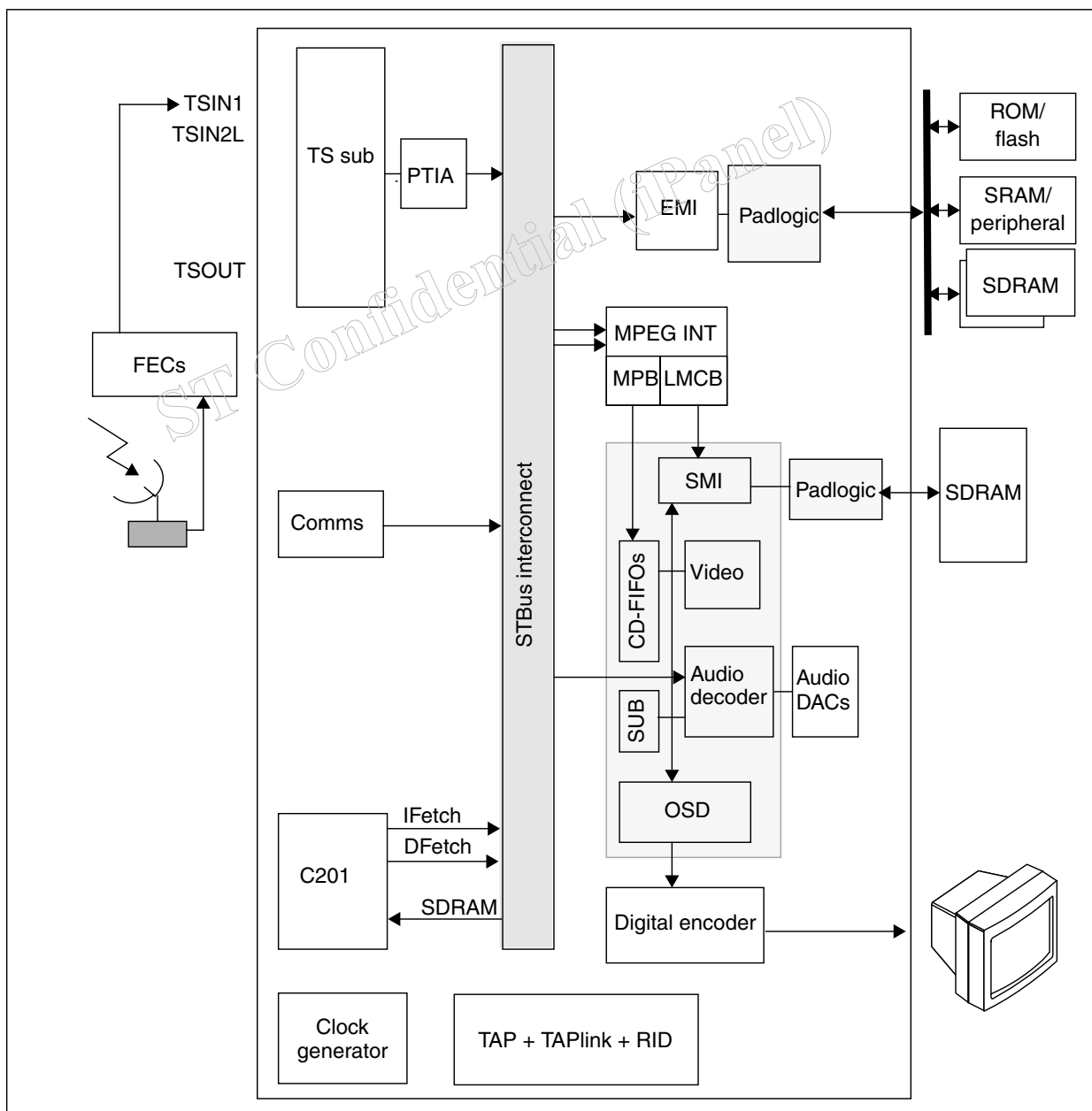
The STi5516 is derived from the STi5514 with the following functional blocks removed:

- Two PTIs,
- GPDMA,
- HDDI (hard disk drive interface),
- DES (for HDD encryption).

The STi5516 EMI only has a 16-bit wide data port and does not support MPX.

The architecture of the STi5516 is illustrated in [Figure 3](#). This chapter briefly describes each of the major functional blocks.

Figure 3: STi5516 architecture



2.2 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.

2.3 Processor core

The ST20-C201 processor core is composed of the ST20C2+ CPU, a diagnostic controller unit (for low intrusion, real-time debugging), memory (8-Kbyte instruction cache, 8-Kbyte data cache and 8 Kbyte SRAM) and a 16 priority-level interrupt controller.

2.4 Memory subsystem

The STi5516 has two memory interfaces.

The SMI is the STi5516's local memory interface and is used for all the STi5516's data requirements in unified memory applications, including graphics, video and audio buffers.

For high-performance, nonunified memory systems, additional data SDRAM can be placed on the EMI. Instructions can execute in place from flash/SFlash™ on the EMI or can be copied to SDRAM on the SMI for unified memory applications or to SDRAM on the EMI for the highest performance systems.

The sections below overview the different memory interfaces.

2.4.1 Shared memory interface (SMI)

The SMI is a 16-bit wide data bus with a peak bandwidth of 270 Mbyte/s. It supports one or two banks of 16-Mbit SDRAMs, or one bank of 64-Mbit SDRAM, or one bank of 128-Mbit SDRAM. The SMI provides a fully cacheable address space for data and instructions, with data cacheability controlled in 512 Kbyte blocks for up to 8 Mbytes.

2.4.2 External memory interface (EMI)

This fourth generation EMI provides a glueless interface to SDRAM, SRAM, flash, SFlash™ and peripherals, in up to six configurable banks over a 16-bit wide interface.

Bus cycle strobe timings can be programmed from 0 to 15 phases for slower peripherals.

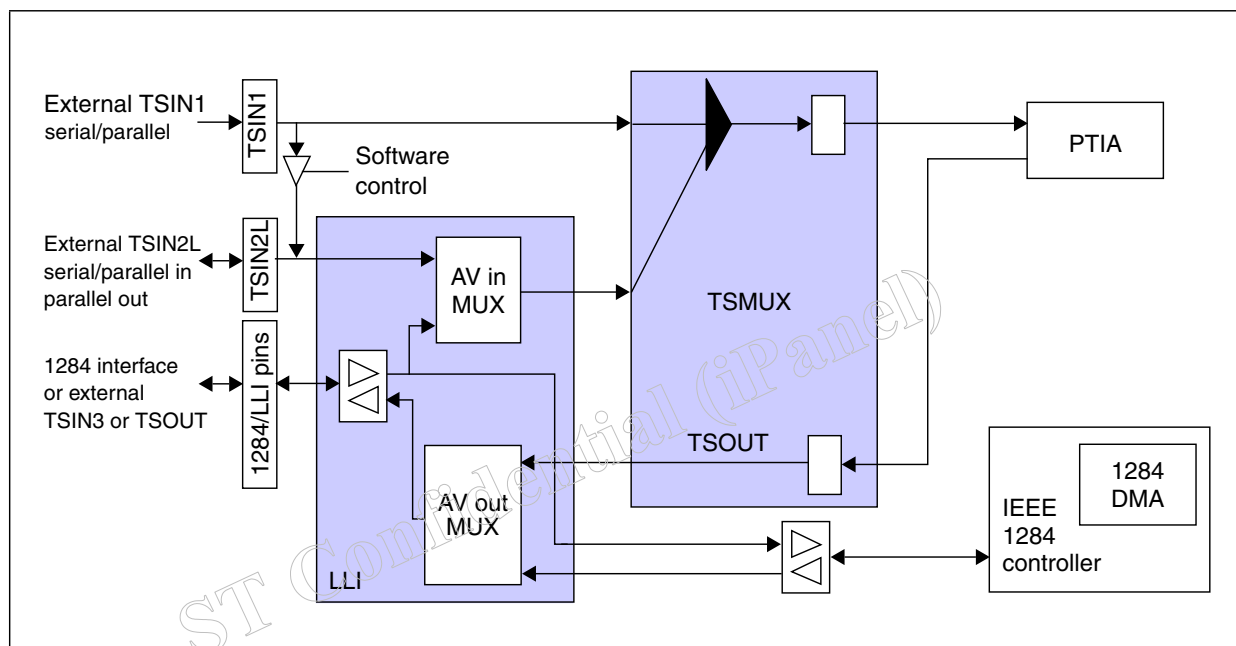
2.5 Transport stream processing

The STi5516 contains a programmable transport interface (PTI) block for concurrent demultiplexing and descrambling of transport streams (TS). Transport streams are input via the two parallel/serial TS inputs. A third parallel TS input interface is available via the link layer interface if neither IEEE1284 nor a transport stream output is required.

2.5.1 Transport stream input/output

A single transport stream is routed to the PTI by the TSMUX block. Any of the three externally supplied input streams can be routed to the PTI. This is illustrated in Figure 4.

Figure 4: TSMUX inputs and outputs



The PTI has an output that allows the entire transport stream or selected packets to be output to an external device such as a DVCR or IEEE1394 link layer controller. The output pins can be tri-stated under software control to support low cost DVB-CI implementations and similar module interfaces. Also under software control, the transport stream input on TSIN1 can be output directly via TSIN2L. This is again to support low cost DVB-CI implementations. The returned stream from the CI module can be input on the 1284 LLI pins (TSIN3).

Note: Only one transport stream can be routed to the single PTI at any one time. The TSMUX cannot dynamically multiplex two streams, so one TS input can be processed by the PTI.

2.5.2 Programmable transport interface (PTI)

The PTI (PTI3) 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.

- DIRECTV® and DVB transport streams can be handled by the PTI with data rates up to 120 Mbit/s.
- The PTI performs PID/SCID filtering to select audio, video and data packets to be processed. More than 48 PID/SCID slots are supported.
- The PTI can descramble streams using DES-ECB or DVB ciphers. NDS specific streams are also supported by the integration of ICAM functionality.

- The PTI has a section filter core that filters DVB standard sections. Four filtering modes are available:
 - wide match mode: 32 x 16-byte filters,
 - long match mode: 64 x 8-byte filters,
 - MAC match mode: 32 x 8-byte filters plus 32 x 6-byte MAC address filter,
 - positive/negative mode: 32 x 8-byte filters with positive/negative filtering at the bit level.

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

Note: The section filter core can also be used to filter nonDVB section data, for DIRECTV® format data streams.

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

2.6 LLI interface

The LLI interface has two functions.

- It provides a dedicated serial/parallel TS input (TSIN2L) that is routed to the TSMUX.
- It selects whether an IEEE1284 data or a parallel transport stream is available on the bidirectional 1284/LLI pins. In this case, the TS is either an output stream (TSOUT) from the PTI (arriving from the TSMUX) or an input stream (TSIN3).

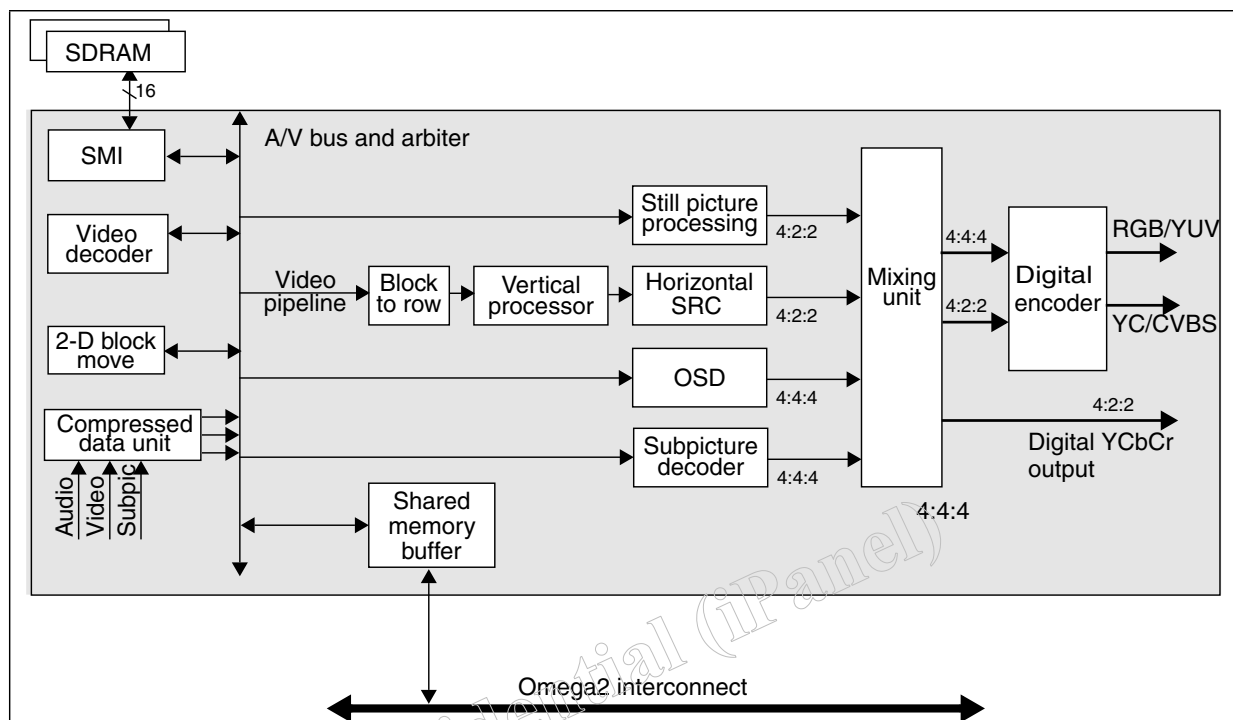
The LLI interface selects either TSIN3 or TSIN2L to route to the TSMUX.

The 1284/LLI interface is used to interface with another IEEE1284 device (in 1284 mode) or to send/receive transport packets to/from an IEEE1394 link layer controller or DVCRs, for example.

2.7 MPEG graphics and display architecture

The MPEG graphics and display architecture shown in the following diagram provides the graphics, video-stream processing and display capabilities of the STi5516.

Figure 5: Graphics and display subsystem



2.7.1 Compressed data unit

The compressed data unit accepts three memory mapped stream inputs, for audio PES/ES, video PES/ES and subpicture units (SPUs). It can also directly accept MPEG-1 system streams or MPEG-2 program streams. It performs PTS/DTS extraction and association, and DSM trick mode bit extraction and association, before writing to the bit buffers via the audio, video, subpicture CD FIFOs.

2.7.2 Video display

The video display pipeline includes highly programmable horizontal and vertical sample-rate converters (SRCs) for smooth video resizing with interpolated filtering. Horizontal and vertical resizing can both be programmed over a x0.25 to x4 range. The video display pipeline also provides chroma upsampling and subpixel and subline pan and scan as part of the filtering and display process.

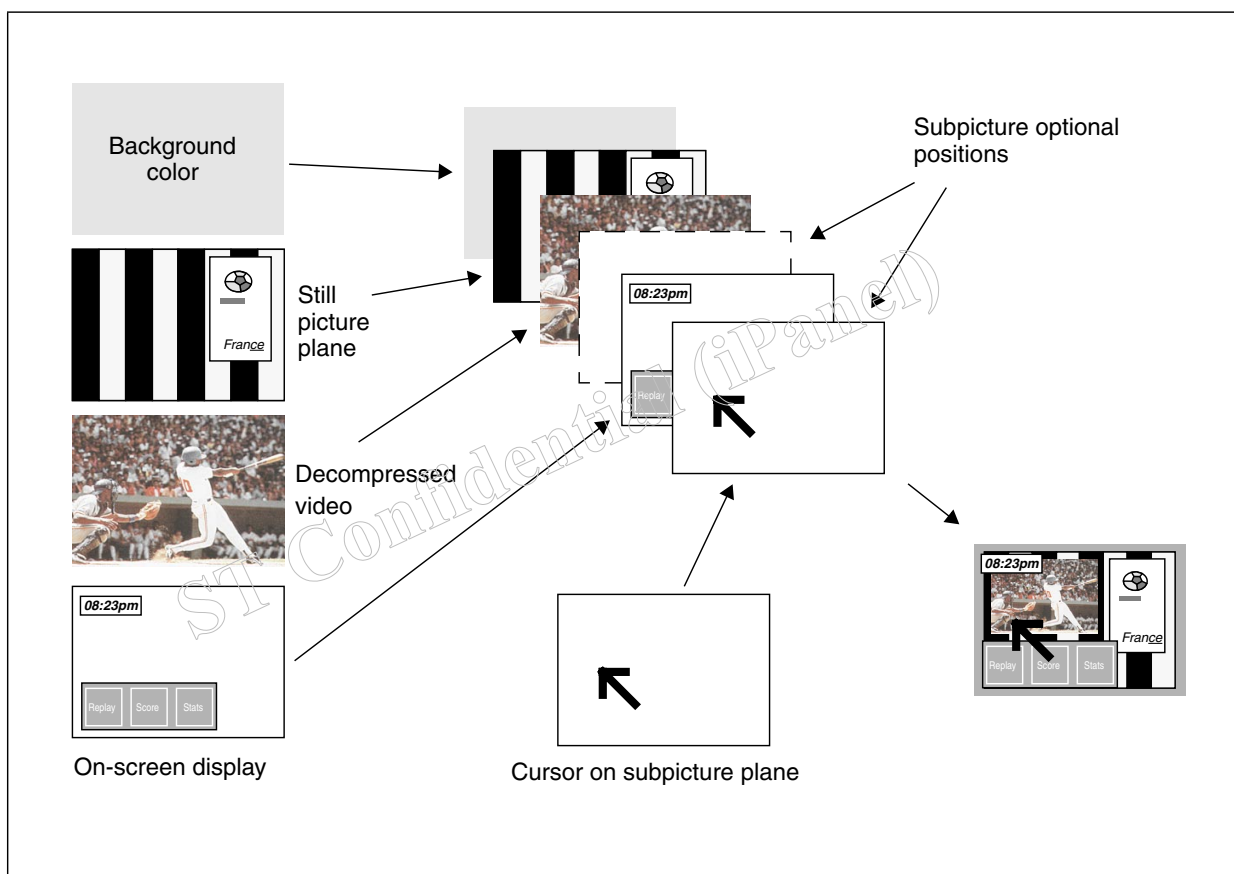
2.8 Graphics and display

2.8.1 Display planes

The following display planes are available:

- background color,
- picture plane,
- video plane,
- on-screen display (OSD) plane,
- subpicture/cursor plane.

Figure 6: Display planes



2.8.2 Picture plane

The picture plane can be used for still pictures and graphics display behind the video plane. Images are stored using YCbCr 4:2:2 format. The picture plane can perform zoom in (x2) and zoom out (x1/2) and hardware unroll or wipes. A 2-D paced BLT engine is also provided for manipulating images and tiling operations (wallpaper effect).

2.8.3 OSD plane

The OSD plane is managed as a set of regions with a specification comprising configuration, bit map and palette information for each region. Each region can be independently specified with a resolution of 2 bpp, 4 bpp or 8 bpp. Regions can be frame-based or field-based. Each region palette can support up to 256 colors with up to 24-bit resolution per color entry. A vertical interfield, antiflicker filter is provided to reduce flicker on interlace displays.

2.8.4 Display mixing

Display planes are composited using alpha blending between planes. Mixing of the OSD plane with the lower layers is achieved using an alpha blending component per region or using an individual 6-bit alpha component per color to support antialiased fonts.

Mixing of OSD regions and the subpicture/cursor layer on the composite outputs (YC and CVBS), intended for a VCR, may be disabled at the same time as outputting all the layers to the component (RGB/YUV) and digital video outputs.

2.9 Digital encoder

The digital encoder converts a 4:2:2 digital video stream from the display mixer into a standard analog baseband PAL/SECAM/NTSC signal. It also converts a 4:4:4 digital video stream from the digital mixer into RGB and YUV components. The digital encoder can also input a 4:2:2 digital video stream from the digital video output port, when the port is used in input mode.

Note: These functions are multiplexed on to parallel input/output (PIO) ports.

The digital encoder can handle interlaced mode in all standards. Both square pixel and 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.

Two integrated tri-DACs provide six analog TV outputs on which it is possible to output either (S-VHS(Y/C) + CVBS + RGB) or (S-VHS(Y/C)+ CVBS + YUV) or (Y1 + C1 + CVBS1 + C2 + Y2 + CVBS2).

The encoder can operate in master mode or in one of several slave modes, where it locks on to incoming sync signals.

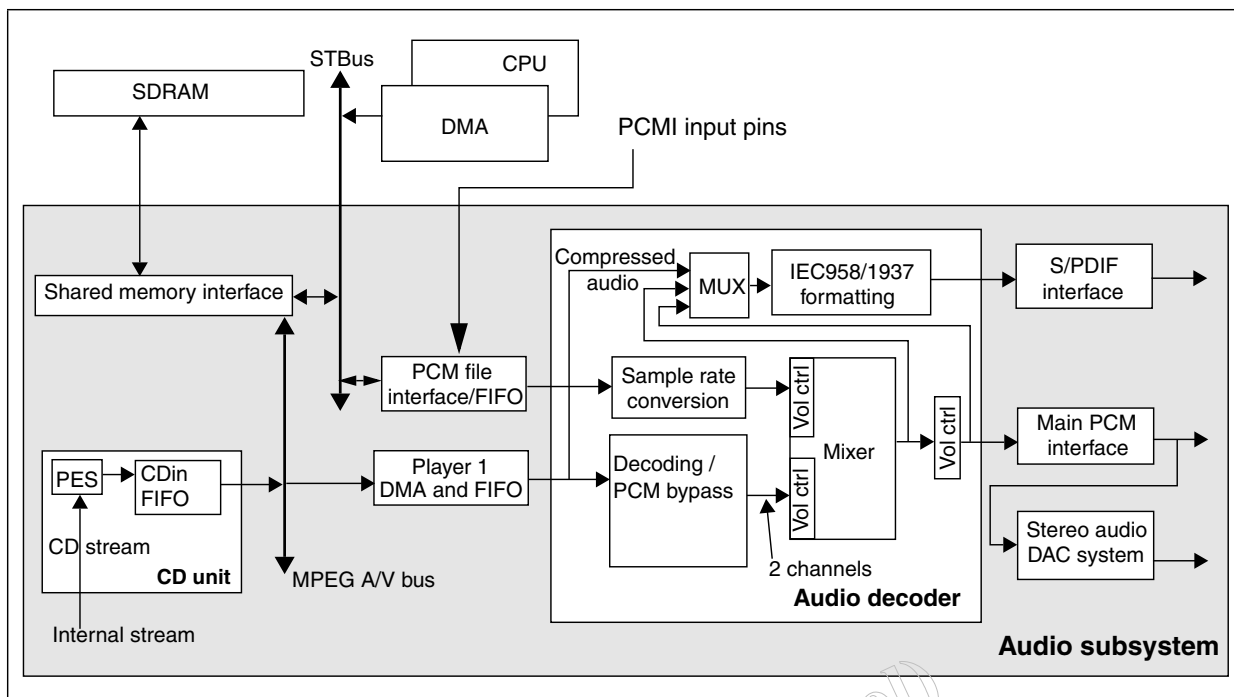
2.10 Audio subsystem

The audio subsystem supports Dolby® Digital 5.1 (Dolby® AC3) audio decoding and mixing with internal PCM files. Decoding of MPEG-1 layers I and II are also supported. MP3 is supported as product variant option. Decoded multi-channel audio is downmixed before emerging as stereo or Dolby® Pro Logic® encoded audio.

The integrated DACs provide analog stereo output directly from the device. Alternatively the output can be from an external stereo DAC.

Multichannel streams are not decoded for full multichannel output, but can be passed through to the IEC958 output for external decoding. sound. Audio sample rates of 32 kHz, 44.1 kHz and 48 kHz are supported. The audio subsystem is illustrated in [Figure 7](#).

Figure 7: Audio subsystem



The audio subsystem consists of the units listed below.

- Compressed data interface: inputs compressed data audio streams and buffers them in an audio bit buffer in local memory. The stream source is internal (from the PTI, for example).
- CD player: manages the CD stream bit buffer; it reads the CD stream and sends it to the DSP for decoding, mixing and output at the same time as formatting the stream and sending it to the IEC958 output.
- PCM file interface: receives a PCM file, from a PCM file buffer using the PTI channel 3 DMA as PCM file player and sends it to the DSP for sample rate conversion, mixing and output. It is also possible to write directly from the CPU to the PCM file interface.
- Audio decoder 24-bit audio digital signal processor: processes audio streams sent to it by the CD player and PCM file player.
- IEC958 output interface: outputs IEC958/IEC1937 formatted CD or PCM audio received from the audio DSP.
- 1-channel PCM output interface: outputs PCM audio received from the audio DSP.
- Integrated 24-bit stereo audio DAC system.
- Audio PLL/ digital frequency synthesizer: generates the PCM and sample rate clocks.
- Programmable tone generation for dish alignment.

2.11 Modem

Standard drivers are available for V22bis software modem and V34/V90 controllerless modem ported to the STi5516 architecture.

2.12 Internal peripherals

The STi5516 has many dedicated internal peripherals for digital TV receiver applications, including:

- five ASCs (UARTs), two of which are generally used by the smartcard controllers,
- teletext serializer and DMA,
- two SSCs for I²C master/slave interfaces,
- an IEEE1284 interface module with DMA (Note: 1284 electrical drivers are not present in the device),
- six GPIO ports,
- four PWM channels,
- a multichannel, infrared blaster/decoder interface module,
- a modem analog front-end interface (MAFE),
- an interrupt level controller,
- a low-power/RTC/watchdog controller,
- DCU toolset support,
- a JTAG/TAP interface.

2.13 EMI programmable output drive

The EMI output drive of the STi5516 is programmable on a bus-by-bus basis.

2.14 Clock generation

All system clocks are generated using the clock generator block. This contains a high-frequency PLL (600 MHz) that is divided down to produce a series of phase-related programmable clock channels. The guaranteed phase relationship between these channels simplifies interconnect bridging between different subsystem modules and gives lower latency compared to a fully asynchronous clocking scheme.

The STi5516 has a clock master; the two channels driving the SDRAM/flash clock outputs are systematically phase aligned to optimize the external bus performance of the EMI.

2.15 Smartcard interface

The STi5516 smartcard interface is ISO7816, EMV2000 and NDS compliant with the addition of an ST8004 smartcard interface. Note for some markets the ST8004 functions can be implemented using lower-cost discrete solutions.

3 Audio and video summary specification

3.1 Functional limitations

Functional limitations of the STi5516 different from this datasheet can be found in the *STi5516 buglist*, reference number ADCS 7428678. This datasheet must be used in conjunction with the bug list.

3.2 Summary specification

Table 1: Summary specification

Video decoder	
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@main level (MP@ML), main profile @ low level (MP@LL) Simple profile@main level (SP@ML)
Maximum picture size	Width: 4080 Number of macroblocks: 16383
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 vertical
PTI compressed data input (2 x parallel/serial)	Parallel peak input rate: 160 Mbit/s Serial peak input rate: 100 Mbit/s Maximum sustained average input rate: 120 Mbit/s
SDRAM interface	External SDRAM used for storage of picture buffers, bit buffer and on-screen display definitions. 16-bit data bus, one or two banks, refresh handled by decoder Configurations supported for SDRAM: 1 M x 16 (1 bank), 2 M x 16 (2 banks), 4 M x 16 (1 bank) for a single 64 Mbit SDRAM, 8 M x 16 (1 bank), for 128 Mbit SDRAM.
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	
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 antialiasing filter Vertical chroma reconstruction or luma filtering up to 4-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)	A bitmap that is separately definable for each field or frame which can be superimposed on the final picture output OSD defined as rectangular regions, each with unique palette defining 4, 16 or 256 colors (including transparency) Each region has a blending factor, selectively applied to each color in the palette The number of regions limited by the memory space allocated to the OSD Region definitions can be organized as a linked list A block move facility is available for reduction of microcontroller loading
Audio decoder	
Bit streams accepted	Dolby® Digital, MPEG-1 layers I and II and PCM MPEG2 PES streams for MPEG-1, Dolby® Digital, MP3 (product option) and Linear PCM (LPCM) S/PDIF input data (IEC-60958 or IEC-61937 standards) is accepted if an external circuitry extracts the PCM clock from the stream
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
Audio clock	27 MHz nominal
Compressed data input	Byte-mode input - burst rate up to 28.5 Mbyte/s
PCM output	16-, 18-, 20- or 24-bit PCM output, I ² S and other popular formats supported
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

4 Pin list

4.1 Pin-out

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

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	EMI SDRAM CLK	EMIFLASH CLK	EMIDATA [14]	EMIDATA [13]	EMIDATA [10]	EMIDATA [7]	EMIDATA [3]	EMIDATA [0]	EMIADDR [23]	EMIADDR [19]	EMIADDR [16]	EMIADDR [13]	EMIADDR [9]
B	VDD33	VDD33	EMIDATA [15]	EMIDATA [12]	EMIDATA [9]	EMIDATA [6]	EMIDATA [2]	EMIADDR [25]	EMIADDR [22]	EMIADDR [18]	EMIADDR [15]	EMIADDR [12]	EMIADDR [8]
C	VDD33	VDD33	VDD33	EMIDATA [11]	EMIDATA [8]	EMIDATA [5]	EMIDATA [1]	EMIADDR [24]	EMIADDR [21]	EMIADDR [17]	EMIADDR [14]	EMIADDR [11]	EMIADDR [7]
D	VDD33	VDD33	VDD33	VDD33	VDD33	EMIDATA [4]	GND	VDD33	EMIADDR [20]	VDD18	VDD33	EMIADDR [10]	EMIADDR [6]
E	GND	GND	GND	VDD33									
F	GND	GND	GND	VDD33									
G	VDD18	VDD18	VDD18	GND									
H	NOT_EMI ACKREQ	NC	EMIBOOT MODE[0]	VDD18									
J	NOT_EMI CAS	NOT_EMI RAS	NOT_EMI REQNGT	VDD33									
K	NOT_EMI CSD	NOT_EMI CSC	NOT_EMI CSB	NOT_EMI CSA									
L	NOT_EMI BE [1]	NOT_EMI BE [0]	NOT_EMI CSF	NOT_EMI CSE									
M	NOT_EMI OE	VDD33	VDD33	VDD33									
N	VDD33	EMIRD NOTWR	NOT_EMI LBA	EMIWAIT NOT TREADY									
P	DCU TRIGGER IN	VDD18	VDD18	VDD18									
R	VDD18	VDD18	DCU TRIGGER OUT	VDD18									
T	INTER RUPT[3]	INTER RUPT[2]	INTER RUPT[1]	INTER RUPT[0]									
U	PIO0[1]	PIO0[0]	PIO0[2]	VDD33									
V	PIO0[6]	PIO0[5]	PIO0[4]	PIO0[3]									
W	PIO1[2]	PIO1[1]	PIO1[0]	PIO0[7]									
Y	PIO1[5]	PIO1[4]	PIO1[3]	GND									
A A	VSSAA DAC	VDDAA DAC	VDDASA DAC	PIO1[6]									
A B	OUTM LEFT	VCCAA DAC	GNDAA DAC	OUTM RIGHT									
A C	VCCASA DAC	IREF	OUTP LEFT	OUTP RIGHT	GND	VDD33	TDI	LPCLKIN	SHIELDV DAC	GNDV/DAC YCC	VREFDAC RGB	GNDV/DAC RGB	VDD18
A D	VBGFIL	PIO2[1]	PIO2[5]	PIO2[7]	PIO3[6]	PIO3[4]	TMS	LPCLK OSC	VREFDAC YCC	CVOUT	IREFDAC RGB	BOUT	SCLK
A E	PIO1[7]	PIO2[2]	PIO2[6]	PIO3[1]	PIO3[0]	NOT_TRST	NOT_RESET	RTCVDD	IREFDAC YCC	YOUT	GOUT	VDDV/DAC RGB	PCMCLK
A F	PIO2[0]	PIO2[3]	PIO2[4]	PIO3[3]	PIO3[2]	TDO	TCK	NOT_WDOG RSTOUT	COUT	VDDV/DAC YCC	ROUT	NC	LRCLK

GND	GND	GND
GND	GND	GND
GND	GND	GND
GND	GND	GND
GND	GND	GND
GND	GND	GND

Note: 1 NC (not connected): indicates that the pin has no electrical connection, and may be used to route power or signals on the PCB.

2 Do not connect: pin is reserved. It may have an electrical connection and must not be used.

14	15	16	17	18	19	20	21	22	23	24	25	26	
EMIADDR [5]	GND	VDDGEN FSYN	NC	CLK27MA	VDD18	VDD33	GND	Do not connect	GND	GND	GND	GND	A
EMIADDR [4]	GND	GNDGEN FSYN	VDD AUDIO FSYN	VDD18	VDD33	GND	GND	VDD18	GND	GND	GND	NOT_P1284 STROBE	B
EMIADDR [3]	VDDVPLL	CLK SPEED SEL	GND AUDIO FSYN	VDD18	VDD33	GND	GND	GND	GND	GND	P1284 BUSY	NOT_P1284 ACK	C
EMIADDR [2]	GND	GND	AUXCLK OUT	VDD18	VDD33	GND	GND	GND	GND	NOT_P1284 AUTOFD	P1284 SELECT	P1284 PERROR	D
									VDD33	NOT_P1284 SELECTIN	NOT_P1284 INIT	NOT_P1284 FAULT	E
									P1284 DATA[4]	P1284 DATA[5]	P1284 DATA[6]	P1284 DATA[7]	F
									GND	P1284 DATA[1]	P1284 DATA[2]	P1284 DATA[3]	G
									TSIN2L DATA[5]	TSIN2L DATA[6]	TSIN2L DATA[7]	P1284 DATA[0]	H
									VDD33	TSIN2L DATA[3]	TSIN2L PACKET CLK	TSIN2L DATA[4]	J
									GND	TSIN2L DATA[0]	TSIN2L DATA[1]	TSIN2L DATA[2]	K
									VDD18	TSIN2L BYTE CLK	TSIN2L ERROR	TSIN2L BYTE CLKVALID	L
									TSIN1 DATA[4]	TSIN1 DATA[5]	TSIN1 DATA[6]	TSIN1 DATA[7]	M
									TSIN1 DATA[0]	TSIN1 DATA[1]	TSIN1 DATA[2]	TSIN1 DATA[3]	N
									TSIN1 BYTE CLK	TSIN1 PACKET CLK	TSIN1 ERROR	TSIN1 BYTE CLKVALID	P
									VDD18	VDD18	VDD18	VDD18	R
									SMIMEM CLKIN	SMIDATA ML	SMIDATA MU	NC	T
									NOT_SMI CAS	NOT_SMI RAS	NOT_SMIWE	SMIMEM CLKOUT	U
									VDD33	SMIDATA [15]	NOT_SMI CS0	NOT_SMI CS1	V
									SMIDATA [11]	SMIDATA [12]	SMIDATA [13]	SMIDATA [14]	W
									VDD33	SMIDATA [8]	SMIDATA [9]	SMIDATA [10]	Y
									GND	SMIDATA [5]	SMIDATA [6]	SMIDATA [7]	A A
									VDD33	SMIDATA [2]	SMIDATA [3]	SMIDATA [4]	A B
GND	SPDIF	VDD18	PIO4[4]	VDD33	PIO5[3]	GND	VDD18	VDD33	SMIADDR [12]	SMIADDR [13]	SMIDATA [0]	SMIDATA [1]	A C
DO NOT CONNECT	PIO3[5]	PIO4[1]	PIO4[5]	PIO5[1]	PIO5[4]	PIO5[7]	GND	VDD33	SMIADDR [8]	SMIADDR [9]	SMIADDR [10]	SMIADDR [11]	A D
PCM DATA[1]	PIO3[7]	PIO4[2]	PIO4[7]	PIO5[2]	PIO5[5]	NOT_HSYNC	GND	VDD33	SMIADDR [4]	SMIADDR [5]	SMIADDR [6]	SMIADDR [7]	A E
DO NOT CONNECT	PIO4[0]	PIO4[3]	PIO5[0]	PIO4[6]	PIO5[6]	EVENNOT ODD	GND	VDD33	SMIADDR [0]	SMIADDR [1]	SMIADDR [2]	SMIADDR [3]	A F

Confidential

4.2 STi5516 pin list

Signal names are prefixed by NOT_ if they are active low; otherwise they are active high.
All signals are only 3.3 V capable unless otherwise indicated as 1.8 V or 5 V tolerant.

Table 2: I/O load capacitance and DC loading

Pad type	Functional pin group	Maximum load capacitance (pF)	Drive (mA)	Notes
C4	Others	75	4	
S8	SDRAM/EMI	75	8	
S8b	SDRAM/EMI	75	8	
E8	EMI (programmable)	35	8	a
		25	6	a
		15	4	a
		5	2	a
P4	PIO	400	4	

a. Typical load, but the maximum is 75 pF

Table 3: Analog power supply pins

Pin	Location	Number	Function
VDDVDACRGB	AE12	1	3.3 V power supply for RGB video DAC
VDDVDACYCC	AF10	1	3.3 V power supply for YCC video DAC
GNDVDACRGB	AC12	1	Ground for RGB video DAC
GNDVDACYCC	AC10	1	Ground for YCC video DAC
SHIELDVDAC	AC9	1	Shield ground for 2 x video DACs
IREFDACRGB	AD11	1	RGB video DAC current reference
IREFDACYCC	AE9	1	YCC video DAC current reference
VREFDACRGB	AC11	1	RGB video DAC voltage reference
VREFDACYCC	AD9	1	YCC video DAC voltage reference
VDDVPLL	C15	1	3.3 V power for video PLL
VDDAUDIOFSYN	B17	1	1.8 V dedicated power for low jitter audio clock frequency synthesizer
GNDAUDIOFSYN	C17	1	Dedicated ground for low jitter audio clock frequency synthesizer
VDDGENFSYN	A16	1	1.8 V dedicated power for nonaudio clock frequency synthesizer
GNDGENFSYN	B16	1	Dedicated ground for nonaudio clock frequency synthesizer
VDDAADAC	AA2	1	3.3 V power for audio DAC
VSSAADAC	AA1	1	Ground for audio DAC command switches

Table 3: Analog power supply pins

Pin	Location	Number	Function
VDDASADAC	AA3	1	3.3 V power for audio DAC substrate
VCCAADAC	AB2	1	3.3 V power for audio DAC command switches
GNDAADAC	AB3	1	Ground for audio DAC
VCCASADAC	AC1	1	3.3 V power for audio DAC command switches substrate
IREF	AC2	1	Audio DAC output reference current
VBGFIL	AD1	1	Audio DAC filtered output reference voltage

Table 4: Digital power supply pins

Pin	Location	Number	Function
VDD18	a	24	1.8 V power supply
VDD33	b	35	3.3 V power supply
RTCVDD	AE8	1	Low power controller 1.8 V power supply
GND	c	76	Ground for power supplies

- a. A19, B18, B22, C18, D10, D18, G1 to G3, H4, L23, P2 to P4, R1, R2, R4, R23 to R26, AC13, AC16 and AC21.
- b. A20, B1, B2, B19, C1 to C3, C19, D1 to D5, D8, D11, D19, E4, E23, F4, J4, J23, M2 to M4, N1, U4, V23, Y23, AB23, AC6, AC18, AC22, AD22, AE22 and AF22.
- c. A15, A21, A23 to A26, B15, B20, B21, B23 to B25, C20 to C24, D7, D15, D16, D20 to D23, E1 to E3, F1 to F3, G4, G23, K23, L11 to L16, M11 to M16, N11 to N16, P11 to P16, R11 to R16, T11 to T16, Y4, AA23, AC5, AC14, AC20, AD21, AE21 and AF21.

Table 5: RTC pins

Pin	Location	I/O	Function
LPCLKIN ^a	AC8	I	Low power clock input
LPCLKOSC ^A	AD8	I/O	Low power clock oscillator

- a. 1.8 V tolerant

Table 6: System pins

Pin	Location	I/O	Function	Pad type
CLK27MA ^A	A18	I	Selectable input clock to PLL or for x1 mode	C4
CLKSPEEDSEL ^a	C16	I	PLL speed select	C4
AUXCLKOUT ^a	D17	O	Auxiliary clock for general use	C4
NOT_RESET ^B	AE7	I	System reset	-
NOT_WDOGRSTOUT ^a	AF8	O	Internal watchdog timer reset.	C4

a. 5 V tolerant

b. 1.8 V tolerant

Table 7: JTAG pins

Pin	Location	I/O	Function	Pad type
TDI ^A	AC7	I	Boundary scan test data input	C4
TMS ^a	AD7	I	Boundary scan test mode select	C4
TCK ^a	AF7	I	Boundary scan test clock	-
NOT_TRST ^a	AE6	I	Boundary scan test logic reset	C4
TDO ^a	AF6	O	Boundary scan test data output	C4

a. 5 V tolerant

Table 8: DCU pins

Pin	Location	I/O	Function	Pad type
DCUTRIGGERIN ^a	P1	I	External trigger input to DCU	C4
DCUTRIGGEROUT ^A	R3	O	Signal to trigger external debug circuitry	C4

a. 5 V tolerant

Table 9: EMI pins

Pin	Location	I/O	Function	Pad type
NOT_EMIRAS or NOT_CI_IORD ^a	J2	O	Row address strobe for SDRAM	C4
NOT_EMICAS or NOT_CI_IOW ^a	J1	O	Column address strobe for SDRAM	E8
NOT_EMICSA	K4	O	Peripheral chip select A	E8
NOT_EMICSB	K3	O	Peripheral chip select B	E8
NOT_EMICSC	K2	O	Peripheral chip select C	E8
NOT_EMICSD	K1	O	Peripheral chip select D	E8
NOT_EMICSE	L4	O	Peripheral chip select E	E8
NOT_EMICSF	L3	O	Peripheral chip select F	E8
NOT_EMIBE[1:0]	L1, L2	O	External device data bus byte enable. 1 bit per byte of the data bus.	E8
NOT_EMIOE or NOT_CI_OE	M1	O	External device output enable.	E8
NOT_EMILBA or NOT_CI_WEA	N3	O	Flash device load burst address.	E8
EMIWAITNOTTREADY ^b	N4	I	External memory device target ready indicator	C4
EMIRDNOTWR	N2	O	External read/write access indicator. Common to all devices.	E8
EMIDATA[15:0]	c	I/O	External common data bus.	E8
EMIADDR[25:2] ^d	e	O	External common address bus	E8
NOT_EMIREQGNT	J3	O	Bus request/grant indicator	E8
NOT_EMIAKREQ ^b	H1	I	Bus grant/request indicator	C4
EMIBOOTMODE0 ^b	H3	I	External power-up port size indicator	C4
EMISDRAMCLK	A1	O	SDRAM clock	E8
EMIFLASHCLK	A2	O	Peripheral clock	E8

a. Or equivalent ATA HDD interface signal.

b. 5 V tolerant

c. B3, A3, A4, B4, C4, A5, B5, C5, A6, B6, C6, D6, A7, B7, C7 and A8.

d. EMIADDR[19:20] are used as ATA HDD interface function: ATA CS0 and CS1. There is no interconnect configuration control register bit to select this function. The addresses are just reused as chip selects.

e. B8, C8, A9, B9, C9, D9, A10, B10, C10, A11, B11, C11, A12, B12, C12, D12, A13, B13, C13, D13, A14, B14, C14 and D14.

Table 10: Transport stream 2 pins

Pin	Location	I/O	Function	Pad type
TSIN2LBYTECLK ^a	L24	I/O	Transport stream bit clock	C4
TSIN2LBYTECLKVALID ^a	L26	I/O	Transport stream bit clock valid edge	C4
TSIN2LERROR ^a	L25	I/O	Transport stream packet error	C4
TSIN2LPACKETCLK ^a	J25	I/O	Transport stream packet strobe	C4
TSIN2LDATA[7:0] ^a	b c	I/O	Transport stream data	C4

a. 5 V tolerant

b. H25, H24, H23, J26, J24, K26, K25 and K24

c. TSIN2LDATA7 is used for data input in serial mode.

Table 11: Transport stream 1 pins

Pin	Location	I/O	Function	Pad type
TSIN1BYTECLK ^a	P23	I	Transport stream bit/byte clock	C4
TSIN1BYTECLKVALID ^a	P26	I	Transport stream bit/byte clock valid edge	C4
TSIN1ERROR ^a	P25	I	Transport stream packet error	C4
TSIN1PACKETCLK ^a	P24	I	Transport stream packet strobe	C4
TSIN1DATA[7:0] ^a	b, c	I	Transport stream data in	C4

a. 5 V tolerant

b. M26, M25, M24, M23, N26, N25, N24 and N23.

c. TSIN1DATA7 is used for data input in serial mode.

Confidential

Table 12: Programmable I/O pins

Pin	Location	I/O	Function	Pad type
PIO0[0:7] ^A	b	I/O	Parallel input/output pin or alternative function	P4
PIO1[0:7] ^a	c	I/O		P4
PIO2[0:7] ^a	d	I/O		P4
PIO3[0:7] ^a	e	I/O		P4
PIO4[0:7] ^a	f	I/O		P4
PIO5[0:7] ^a	g	I/O		P4

a. 5 V tolerant

b. U2, U1, U3, V4, V3, V2, V1 and W4

c. W3, W2, W1, Y3, Y2, Y1, AA4 and AE1

d. AF1, AD2, AE2, AF2, AF3, AD3, AE3 and AD4

e. AE5, AE4, AF5, AF4, AD6, AD15, AD5 and AE15

f. AF15, AD16, AE16, AF16, AC17, AD17, AF18 and AE17

g. AF17, AD18, AE18, AC19, AD19, AE19, AF19 and AD20

Table 13: Digital audio pins^a

Pin	Location	I/O	Function	Pad type
SCLK ^B	AD13	O	Serial clock	C4
PCMDATA[1] ^b	AE14	O	PCM data out	C4
PCMCLK ^b	AE13	I/O	External PCM clock input or internal PCM clock output	C4
LRCLK ^b	AF13	O	Left/right clock	C4
SPDIF ^b	AC15	O	Digital audio output	C4

a. Note: Digital audio input pins PCMI_SCLK, PCMI_DATA and PCMI_LRCLK are alternate functions for NOT_CD_REQ[1], I1284HOSTLOGICHIGH, and NOT_CD_REQ[0], on PIO port 3 bits [6:4]

b. 5 V tolerant

Table 14: AVSDRAM pins (SMI)

Pin	Location	I/O	Function	Pad type
SMIADDR[13:0]	a	O	Audio/video core SDRAM address bus	S8
SMIDATA[15:0]	b	I/O	Audio/video core SDRAM data bus	S8
NOT_SMICS0	V25	O	Audio/video core SDRAM chip select for 1st SDRAM	S8
NOT_SMICS1	V26	O	Audio/video core SDRAM chip select for 2nd 16 Mbit SDRAM	S8
NOT_SMICAS	U23	O	Audio/video core SDRAM column address strobe	S8
NOT_SMIRAS	U24	O	Audio/video core SDRAM row address strobe	S8
NOT_SMIWE	U25	O	Audio/video core SDRAM write enable	S8
SMIMEMCLKIN	T23	I	Audio/video core SDRAM memory clock input	S8b
SMIMEMCLKOUT	U26	O	Audio/video core SDRAM memory clock output	S8
SMIDATAML	T24	O	Audio/video core SDRAM data bus lower byte enable	S8
SMIDATAMU	T25	O	Audio/video core SDRAM data bus upper byte enable	S8

- a. AC24, AC23, AD26, AD25, AD24, AD23, AE26, AE25, AE24, AE23, AF26, AF25, AF24 and AF23.
- b. V24, W26, W25, W24, W23, Y26, Y25, Y24, AA26, AA25, AA24, AB26, AB25, AB24, AC26 and AC25.

Table 15: IEEE 1284/1394 pins

Pin	Location	I/O	Function	Pad type
P1284DATA[7:0] ^a	b	I/O	1284 data or 1394 AV data	I14
NOT_P1284SELECTIN ^a	E24	I/O	1284 or 1394 AV control signals	I14
NOT_P1284INIT ^a	E25	I/O		I14
NOT_P1284FAULT ^a	E26	I/O		I14
NOT_P1284AUTOFD ^a	D24	I/O		I14
P1284SELECT ^a	D25	I/O		I14
P1284PERROR ^a	D26	I/O		I14
P1284BUSY ^a	C25	I/O		I14
NOT_P1284ACK ^a	C26	I/O		I14
NOT_P1284STROBE ^a	B26	I/O		I14

- a. 5 V tolerant
- b. F26, F25, F24, F23, G26, G25, G24 and H26.

Table 16: Interrupt pins

Pin	Location	I/O	Function	Pad type
INTERRUPT[3:0] ^A	b	I/O	External interrupts	C4

a. 5 V tolerant

b. T1, T2, T3 and T4.

Table 17: Analog audio DAC (digital-to-analog converter) pins

Pin	Location	I/O	Function
OUTLEFT	AC3	O	Left channel, differential positive current output
OUTMLEFT	AB1	O	Left channel, differential negative current output
OUTRIGHT	AC4	O	Right channel, differential positive current output
OUTMRIGHT	AB4	O	Right channel, differential negative current output

Table 18: Analog video DAC pins

Pin	Location	I/O	Function
ROUT	AF11	O	Red output
GOUT	AE11	O	Green output
BOUT	AD12	O	Blue output
COUT	AF9	O	Chroma output
CVOUT	AD10	O	Composite video output
YOUT	AE10	O	Luma output

Table 19: Digital video pins

Pin	Location	I/O	Function	Pad type
NOT_HSYNC ^A	AE20	I/O	Horizontal sync	C4
EVENNOTODD ^a	AF20	I/O	Vertical sync	C4

a. 5 V tolerant

Confidential

4.3 PIO pins and alternative functions

To improve flexibility and to allow the STi5516 to fit into different set-top box application architectures, the input and output signals from some of the peripherals 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 these pins to be configured as general purpose PIO if the associated peripheral input or output is not required in that particular application.

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

Figure 8: I/O port pins

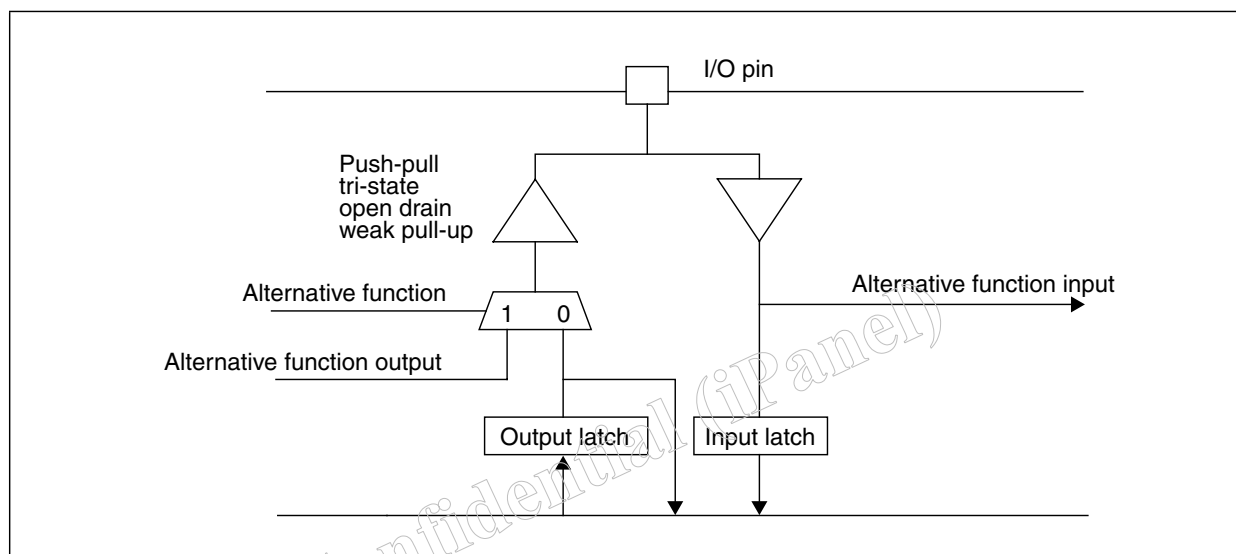


Table 20 to Table 25 show the assignment of the alternative functions to the PIO bits. Brackets () in the table indicate suggested or possible pin usages as a PIO, not an alternative function connection.

Table 20: Port 0 PIO signal assignments

Port 0 bit	Input	Output
Bit 0		SC0_DATAOUT or ASC0_TXD ^a
Bit 1	SC0_DATAIN or ASC0_RXD ^a	
Bit 2	SC0_CG_EXTCLK	
Bit 3		SC0CG_CLK or SMCDSS_CLK ^b
Bit 4		(SC0_RESET)
Bit 5		(SC0_NOT_SETVCC)
Bit 6		SC0_DIR or ASC0_NOTOE ^c , (SC0_NOT_SETVPP)
Bit 7	(SC0_DETECT)	

- ASC0 TX/RX data becomes smartcard TX/RX data when the ASC module is used in smartcard mode.
- Output function between PIO or smartcard module clock generator alternate function and clock generator module frequency synthesizer clock is selected by bit 28 in the interconnect register CONFIG_CONTROL_A (SMCARDA_DSSSMCLK_NOT_PIOBIT3). If the DSS smartcard mode is selected (bit 28 = 1), this overrides the normal PIO or PIO alternate function output.
- When ASC0 is used in nonsmartcard mode, the smartcard direction signal becomes an active low ASC TX output enable signal. The signals are in fact the same, that is, SC0_DIR = 0 means smartcard TX is active.

Table 21: Port 1 PIO signal assignments

Port 1 bit	Input	Output
Bit 0		SC1_DATAOUT or ASC1_TXD ^a
Bit 1	SC1_DATAIN or ASC1_RXD ^a	
Bit 2	SC1_CG_EXTCLK	
Bit 3		SC1CG_CLK or SMCDSS_CLK ^b
Bit 4		(SC1_RESET)
Bit 5	YC[1]	YC[1], (SC1_NOT_SETVCC)
Bit 6		SC1_DIR or ASC1_NOTOE ^c , (SC1_NOT_SETVPP)
Bit 7	YC[0] (SC1_DETECT)	YC[0]

- ASC1 TX/RX data becomes smartcard TX/RX data when the ASC module is used in smartcard mode.
- Output function between PIO or smartcard module clock generator alternate function and clock generator module frequency synthesizer clock is selected by bit 29 in the interconnect register CONFIG_CONTROL_A (SMCARDB_DSSSMCLK_NOT_PIOBIT3). If the DSS smartcard mode is selected (bit 29 = 1) this overrides the normal PIO or PIO alternate function output.
- When ASC1 is used in nonsmartcard mode the smartcard direction signal becomes an active low ASC TX output enable signal. The signals are in fact the same, that is, SC1_DIR = 0 means smartcard TX is active.

Table 22: Port 2 PIO signal assignments

Port 2 bit	Input	Output
Bit 0		MAFE_HC1 or NOT_ASC4_RTS ^a
Bit 1		MAFE_DOUT or ASC4_TXD ^a
Bit 2	MAFE_DIN or ASC4_RXD ^a	
Bit 3	MAFE_FS or NOT_ASC4_CTS ^a	
Bit 4	MAFE_SCLK	
Bit 5	PWM_CAPTURE0	
Bit 6		PWM_COMPARE0
Bit 7		PWM_OUT0

a. Controlled by bit MAFE_OR_UART4_SEL in interconnect register CONFIG_CONTROL_D (bit 20).

Table 23: Port 3 PIO signal assignments

Port 3 bit	Input	Output
Bit 0	SSC0_MTSR_DIN or SSC0_MRST_DIN	SSC0_MTSR_DOUT or SSC0_MRST_DOUT ^a
Bit 1	SSC0_SCLKIN	SSC0_SCLKOUT
Bit 2	SSC1_MTSR_DIN or SSC1_MRST_DIN	SSC1_MTSR_DOUT or SSC1_MRST_DOUT ^b
Bit 3	SSC1_SCLK	SSC1_SCLK
Bit 4	NOT_CD_REQ[0] or PCMI_LRCLK	I1284PERILOGICHIGH
Bit 5	Slave mode I1284HOSTLOGICHIGH or PCMI_DATA	Master mode I1284HOSTLOGICHIGH
Bit 6	NOT_CD_REQ[1] or PCMI_SCLK	I1284INNOTOUT
Bit 7		PWM_OUT1

- a. Output function selected by bit 24 in interconnect configuration register CONFIG_CONTROL_B (COMMS_SSC0_DOUT_MRST_NOTMTSR_MUXSEL)
- b. Output function selected by bit 25 in interconnect configuration register CONFIG_CONTROL_B (COMMS_SSC1_DOUT_MRST_NOTMTSR_MUXSEL)

Confidential

Table 24: Port 4 PIO signal assignments

Port 4 bit	Input	Output
Bit 0	TTXTREQUEST or OSDENABLE	OSDENABLE ^a
Bit 1	CFC	TXTDATAOUT ^b
Bit 2	YC[7]	YC[7]
Bit 3	ASC2_RXD	
Bit 4		ASC2_TXD ^c
Bit 5	PWM_CAPTURE2 or YC[6]	YC[6]
Bit 6	SCCG_EXTCLK	PWM_COMPARE2
Bit 7		PWM_OUT2

- a. OSDENABLE output function is selected rather than PIO by interconnect configuration register CONFIG_CONTROL_C, bit 2 (CONFIG_OTHER_ALT_PIOPORT4[0]). The output can then be turned off by the MPEG video decoder to use OSDENABLE as an input.
- b. TXTDATAOUT function selected by interconnect configuration register CONFIG_CONTROL_C, bit 3 (CONFIG_OTHER_ALT_PIOPORT4[1]).
- c. After reset, register CONFIG_CONTROL_C bit 4 must be set to 1 to pass PIO data or to use ASC2_TXD in alternate output PIO mode.

Table 25: Port 5 PIO signal assignments

Port 5 bit	Input	Output
Bit 0	IRB_IR_IN ^a	
Bit 1	IRB_UHF_IN ^b	
Bit 2		Infrared transmitter/receiver drive PPM
Bit 3		Infrared transmitter/receiver drive jack (0 or z) open drain jack output ^{c, d}
Bit 4	YC[5]	ASC3_TXD or YC[5] ^e
Bit 5	ASC3_RXD or YC[4]	YC[4]
Bit 6	NOT_ASC3_CTS or YC[3]	YC[3]
Bit 7	YC[2]	NOT_ASC3RTS or YC[2] ^e

- a. The wake-up from low power mode function is enabled by interconnect configuration register CONFIG_CONTROL_D, bit 21 (RC_IRDA_DATA_IN_EN).
- b. The wake-up from low power mode function is enabled by interconnect configuration register CONFIG_CONTROL_D, bit 22 (UHF_IN_EN).
- c. PIO needs to be configured as open drain in alternate output mode to use infrared transmitter/receiver drive jack.
- d. After reset, bit 5 in CONFIG_CONTROL_C must be set to 1 to pass PIO data or use infrared transmitter/receiver drive jack in alternate function mode.
- e. Output function selected by bit 11 in CONFIG_CONTROL_E (CONFIG_OTHER_ALT_PIO_YC)

4.4 Reset states

Table 26: Pin reset states

Pin name	Normal pin direction	Reset pin direction	Pin I/O value (Hex)	Pull up/ pull down ^a
System				
CLK27MA	In	In	Z	-
CLKSPEEDSEL	In	In	Z	-
AUXCLKOUT	Out	Out	b	-
NOT_RESET	In	In	Z	-
NOT_WDOGRSTOUT	Out	Out	0	-
JTAG				
TDI	In	In	Z	Up
TMS	In	In	Z	Up
TCK	In	In	Z	-
NOT_TRST	In	In	Z	-
TDO	Out	In	Z	-
DCU				
DCUTRIGGERIN	In	In	Z	-
DCUTRIGGEROUT	Out	Out	0	-
Transport stream 1				
TSIN1BITCLK	In	In	Z	-
TSIN1BITCLKVALID	In	In	Z	-
TSIN1ERROR	In	In	Z	-
TSIN1PACKETCLK	In	In	Z	-
TSIN1DATA[7:0]	In	In	ZZ	-
Transport stream 2L				
TSIN2LBYTECLK	In/Out	In	Z	-
TSIN2LBYTECLKVALID	In/Out	In	Z	-
TSIN2LERROR	In/Out	In	Z	-
TSIN2LPACKETCLK	In/Out	In	Z	-
TSIN2LDATA[7:0]	In/Out	In	ZZ	-
Interrupts				
INTERRUPT[3:0]	In/Out	In	Z	-

Confidential

Table 26: Pin reset states

Pin name	Normal pin direction	Reset pin direction	Pin I/O value (Hex)	Pull up/ pull down ^a
System EMI				
NOT_EMIRAS	Out	In	Z	-
NOT_EMICAS	Out	In	Z	-
NOT_EMICS[A:F]	Out	In	Z	-
NOT_EMIBE[1:0]	Out	In	Z	-
NOT_EMIOE	Out	In	Z	-
NOT_EMILBA	Out	In	Z	-
EMIWAITNOTTREADY	In	In	Z	-
EMIRDNOTWR	Out	In	Z	-
EMIDATA[15:0]	In/Out	In	ZZZZ	-
EMIADDR[25:2]	Out	In	ZZZZZZ	-
NOT_EMIREQGNT	Out	Out	0	-
NOT_EMIACKREQ	In	In	Z	-
EMIBOOTMODE0	In	In	Z	-
EMISDRAMCLK	Out	Out	c	-
EMIFLASHCLK	Out	Out		-
Programmable I/O				
PIO0[7:0]	In/Out	In	ZZ	Up
PIO1[7:0]	In/Out	In	ZZ	Up
PIO2[7:0]	In/Out	In	ZZ	Up
PIO3[7:0]	In/Out	In	ZZ	Up
PIO4[7:0]	In/Out	In	ZZ	Up
PIO5[7:0]	In/Out	In	ZZ	Up
Digital audio				
SCLK	Out	Out	0	-
PCMDATA1	Out	Out	0	-
PCMCLK	In/Out	In	Z	-
LRCLK	Out	Out	0	-
SPDIF	Out	Out	0	-
IEEE 1284/1394 (all signals in this section refer to footnote ^c)				
P1284DATA[7:0]	In/Out	In	ZZ	-
NOT_P1284SELECTIN	In/Out	In	Z	-
NOT_P1284INIT	In/Out	In	Z	-
NOT_P1284FAULT	In/Out	Out	0	-

Confidential

Table 26: Pin reset states

Pin name	Normal pin direction	Reset pin direction	Pin I/O value (Hex)	Pull up/ pull down ^a
NOT_P1284AUTOFD	In/Out	In	Z	-
P1284SELECT	In/Out	Out	0	-
P1284PERROR	In/Out	Out	0	-
P1284BUSY	In/Out	Out	0	-
NOT_P1284ACK	In/Out	Out	0	-
NOT_P1284STROBE	In/Out	In	Z	-
AV SDRAM				
SMIADDR[13:0]	Out	In	Z	-
SMIDATA[15:0]	In/Out	In	Z	-
NOT_SMICS0	Out	In	Z	-
NOT_SMICS1/SMIADDR[14]	Out	In	Z	-
NOT_SMICAS	Out	In	Z	-
NOT_SMIRAS	Out	In	Z	-
NOT_SMIWE	Out	In	Z	-
SMIMEMCLKIN	In	In	Z	Up
SMIMEMCLKOUT	Out	Out	d	-
SMIDATAML	Out	In	Z	-
SMIDATAMU	Out	In	Z	-

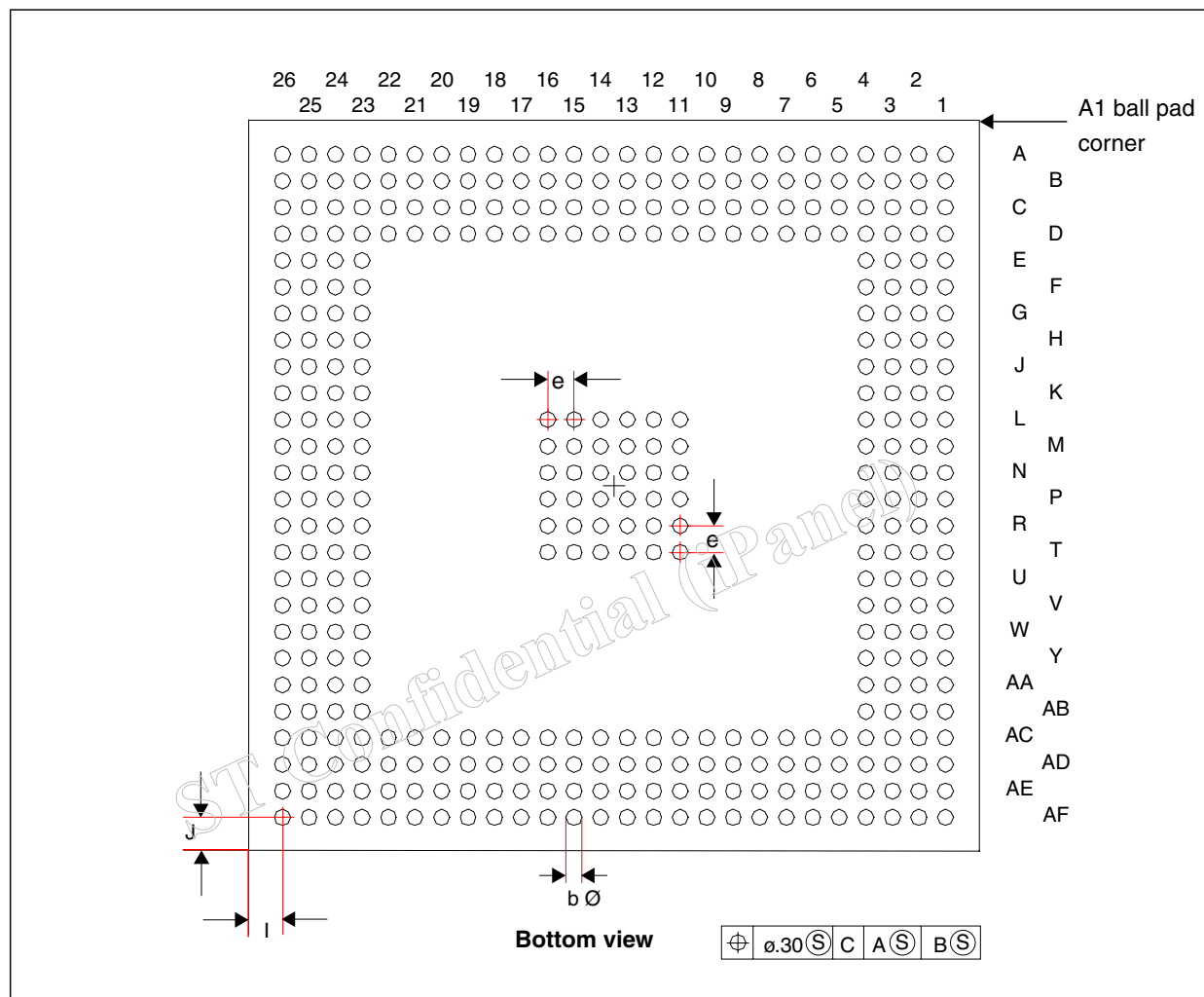
- All resistors are 50 kΩ..
- When in reset mode, the frequency synthesizer that drives AUX_CLK_OUT is set to bypass mode. Therefore the output follows the master clock and toggles.
- When in reset the I/O value for the EMI clocks is the same as the internal clock signals. In reset the internal clock does not stop running so the pad output transitions regularly from 0 to 1.
- When in reset the SMIMEMCLKOUT outputs a regular 27 MHz waveform.

Confidential

5 Package specifications

The STi5516 is packaged in a 388-pin ball grid array (BGA).

Figure 9: 388-pin BGA package



Confidential

Confidential

[illegible]

Table 27: Package dimensions

Dimension	Millimeters			Inches		
	Minimum	Typical	Maximum	Minimum	Typical	Maximum
A	2.21	2.33	2.52	0.087	0.092	0.099
A1	0.50	0.60	0.70	0.020	0.024	0.028
A2	1.12	1.17	1.22	0.044	0.046	0.048
b	0.60	0.76	0.90	0.024	0.030	0.036
D	34.80	35.00	35.20	1.371	1.379	1.387
D1		30.00	30.70		1.181	1.210
D2		-				
e		1.27 BASIC			0.050 BASIC	
E	34.80	35.00	35.20	1.371	1.379	1.387
E1		30.00	30.70		1.181	1.210
E2		-				

Controlling dimensions: Millimeters

Reference document: JEDEC M0151 Revision A

ST Confidential (iPanel)

Confidential

6 Register base addresses

Table 28: Register base addresses

Module	Base address	Register summary
2-D block move	0x0000 0000	Table 29: 2-D block move registers on page 48
Asynchronous serial controller: UART0	0x2010 3000	Table 30: Asynchronous serial controller (ASC) registers on page 48
Asynchronous serial controller: UART1	0x2010 4000	
Asynchronous serial controller: UART2	0x2010 5000	
Asynchronous serial controller: UART3	0x2010 6000	
Asynchronous serial controller: UART4	0x2011 4000	
Audio decoder	0x0000 0800	Table 31: Audio decoder registers on page 49
Audio interface	0x2050 0000	Table 32: Audio interface registers on page 53
Clock generator	0x2001 3000	Table 33: Clock generator registers on page 54
Memory: DCache	0x3000 5000	Table 44: Memory registers on page 69
Memory: ICache	0x3000 4000	
Diagnostic controller	0x3000 3000	Table 34: Diagnostic controller (DCU) registers on page 56
Digital encoder	0x0000 1800	Table 35: Digital encoder registers on page 59
EMI	0x2020 0000	Table 37: EMI registers on page 63
EMI banks: Bank 5	0x7F80 0000	
EMI banks: Bank 4	0x7F00 0000	
EMI banks: Bank 3	0x7000 0000	
EMI banks: Bank 2	0x6000 0000	
EMI banks: Bank 1	0x5000 0000	
EMI banks: Bank 0	0x4000 0000	
EMI buffer	0x202F F800	Table 38: External memory interface (EMI) buffer registers on page 64
IEEE1284 port	0x2012 5000	Table 39: IEEE 1284 port registers on page 65
Infrared blaster	0x2011 5000	Table 41: Infrared transmitter/receiver registers on page 66
Interconnect	0x2001 0000	Table 47: Padlogic registers on page 73

Confidential

Table 28: Register base addresses

Module	Base address	Register summary
Interrupt controller	0x3000 0000	Table 42: Interrupt system registers on page 67
Interrupt level controller	0x2011 1000	
Link layer interface	0x2030 0000	Table 40: IEEE1394 link layer interface (LLI) registers on page 65
Low power module	0x2010 0000	Table 43: Low power module (LPM) registers on page 69
MAFEI	0x2011 3000	Table 45: Modem analog front-end interface (MAFEIF) registers on page 70
MPEG video decoder	0x0000 0000	Table 46: MPEG video decoder registers on page 70
On-screen display	0x0000 0000	Table 36: Display planes registers on page 61
PIO: PIO0	0x2010 C000	Table 48: Parallel I/O port registers on page 73
PIO: PIO1	0x2010 D000	
PIO: PIO2	0x2010 E000	
PIO: PIO3	0x2010 F000	
PIO: PIO4	0x2011 0000	
PIO: PIO5	0x2011 2000	
PTI	0x2002 0000	Table 49A: Programmable transport interface (PTI) registers: DMA on page 74 Table 49B: Programmable transport interface (PTI) registers: others on page 76
PWM	0x2010 B000	Table 50: PWM and counter module registers on page 77
Smartcard interface: SCCG0	0x2010 7000	
Smartcard interface: SCCG1	0x2010 8000	
Still picture plane	0x0000 0000	Table 36: Display planes registers on page 61
Subpicture decoder	0x0000 1000	Table 52: Subpicture decoder registers on page 78
Synchronous serial controller: SSC0	0x2010 9000	Table 53: Synchronous serial controller (SSC) registers on page 79
Synchronous serial controller: SSC1	0x2010 A000	
Teletext DMA	0x2012 4000	Table 54: Teletext DMA registers on page 79
TSMUX	0x2040 0000	Table 55: Transport stream multiplexor (TSMUX) registers on page 80

7 Register summary

All areas not allocated are reserved. Reserved areas must never be accessed.

Table 29: 2-D block move registers

Register name	Description	Address offset	Type
USD_BMC	Block move control, see page 399	0x009E and 0x009F	R/W
USD_BMH	Block move height, see page 400	0x00A2 and 0x00A3	R/W
USD_BMW	Block move width, see page 400	0x00A0 and 0x00A1	R/W
USD_BRP	Memory read pointer, see page 400	0x0088 to 0x008A	R/W
USD_BSK	Block skip, see page 401	0x00A4, 0x00A5 and 0x00A6	R/W
USD_BWP	Memory write pointer, see page 401	0x008C to 0x008E	R/W
USD_PAT	Block move pattern, see page 401	0x00A7 to 0x00AE	R/W

Table 30: Asynchronous serial controller (ASC) registers

Register name	Description	Address offset	Type
ASC_ENABLE	Asynchronous I/O enable register, see page 617	0x00D0 ^a	R/W
ASC_n_BAUDRATE	ASCn baudrate generator, see page 618	0x0000	R/W
ASC_n_CONTROL	ASCn control register, see page 620	0x000C	R/W
ASC_n_GUARDTIME	ASCn guard time, see page 621	0x0018	R/W
ASC_n_INTENABLE	ASCn interrupt enable, see page 621	0x0010	R/W
ASC_n_RETRIES	ASCn number of retries on transmission, see page 622	0x0028	R/W
ASC_n_RXBUFFER	ASCn receive buffer, see page 622	0x0008	RO
ASC_n_RXRESET	ASCn receive FIFO reset, see page 622	0x0024	WO
ASC_n_STATUS	ASCn interrupt status, see page 623	0x0014	RO
ASC_n_TIMEOUT	ASCn time out, see page 624	0x001C	R/W
ASC_n_TXBUFFER	ASCn transmit buffer, see page 624	0x0004	WO
ASC_n_TXRESET	ASCn transmit FIFO reset, see page 624	0x0020	WO

a. Uses *IRBBBaseAddress*

Table 31: Audio decoder registers

Register name	Description	Address offset	Type
Audio decoder start procedure			
AUD_BREAKPOINT	Set breakpoints, see page 483	0x002B	WO
AUD_CLOCKCMD	Start DSP clock, see page 483	0x003A	WO
AUD_INT_RAM	Status of hardware registers, see page 484	0x00FF	RO
Version			
AUD_VERSION	Version, see page 484	0x0000	RO
AUD_IDENT	Identify, see page 484	0x0001	RO
AUD_SOFTVER	Software version, see page 484	0x0071	R/W
RS232 activation			
AUD_RS232_INTERF_ECHO	Enable RS232 input, see page 485	0x00EF	R/W
First input configuration			
AUD_SIN_SETUP	Input data setup, see page 485	0x000C	R/W
AUD_CAN_SETUP	A/D converter setup, see page 486	0x000D	R/W
Output configuration			
AUD_PCMDIVIDER	Input divider for PCM clock, see page 486	0x0054	R/W
AUD_PCMCONF	PCM configuration, see page 487	0x0055	R/W
AUD_PCMCROSS	Cross PCM channels, see page 487	0x0056	R/W
AUD_SFREQ	Sampling frequency, see page 488	0x0005	R/W
ADC input configuration			
AUD_ADCIN_PLAY	Switch second input processing, see page 488	0x00AD	R/W
AUD_SFREQ2	PCM sampling frequency for the second input, see page 489	0x0094	R/W
AUD_ADCIN_MODE	Configure the input mode of PCM data, see page 489	0x0095	R/W
AUD_ADCIN_CFG	Configure the second input hardware processing, see page 490	0x00B1	R/W
AUD_ADCIN_USERSETUP	Size of second input data, see page 490	0x00AB	R/W
AUD_ADCIN_USERSETUP2	Second input data position, see page 491	0x00AC	R/W
AUD_ADCIN_LEFT_VOL	Second input left attenuation, see page 491	0x0088	R/W
AUD_ADCIN_RIGHT_VOL	Second input right attenuation, see page 491	0x0089	R/W
AUD_ADCIN_SHIFT	Second input level shift, see page 492	0x00FC	R/W
PCM-mixing			
AUD_PCMMIX_UPDATE	PCM mixing configuration, see page 493	0x00B2	R/W
AUD_PCMMIX_FIRSTINPUT_VOLUME	Volume can be applied on main channels, see page 494	0x00B3	R/W
AUD_PCMMIX_MIX_COEFFICIENT	Mixing level of second input, see page 494	0x00BA	R/W

Table 31: Audio decoder registers

Register name	Description	Address offset	Type
AUD_PCMMIX_SRC_MSB	Input of the MSB of a new SRC coefficient, see page 494	0x00B8	R/W
AUD_PCMMIX_SRC_LSB	Input LSB of a new SRC coefficient, see page 494	0x00B9	R/W
AUD_PCMMIX_SRC_HANDSHAKE	Allow input of a new SRC coefficient, see page 495	0x00B7	R/W
AUD_PCMMIX_ACK	Activate DSP acquisition of a new SRC coefficient, see page 495	0x0016	R/W
Auxiliary output (PCMOUT3)			
AUD_VCR_OUTPUT	Possible configurations for the VCR output, see page 495	0x00AE	R/W
S/PDIF output setup			
AUD_SPDIF_CMD	S/PDIF control, see page 496	0x005E	R/W
AUD_SPDIF_CAT	Category code, see page 496	0x005F	R/W
AUD_SPDIF_CONF	S/PDIF PCMCLK register, see page 497	0x0060	R/W
AUD_SPDIF_STATUS	S/PDIF status bit, see page 498	0x0061	R/W
AUD_SPDIF_REP_TIME	S/PDIF repetition time of a pause frame, see page 498	0x0075	R/W
AUD_SPDIF_LATENCY	Latency value, see page 498	0x007E	R/W
AUD_SPDIF_DTDI	S/PDIF data type information, see page 499	0x007F	R/W
Command			
AUD_SOFTREST	Soft reset, see page 499	0x0010	WO
AUD_PLAY	Play, see page 500	0x0013	R/W
AUD_MUTE	Mute, see page 500	0x0014	R/W
AUD_RUN	Run decoding, see page 500	0x0072	R/W
AUD_SKIP_MUTE_CMD	Skip or mute commands, see page 501	0x0073	R/W
AUD_SKIP_MUTE_VALUE	Skip frames or mute blocks of frame, see page 502	0x0074	R/W
Interrupt			
AUD_INTE	Interrupt enable, see page 502	0x0007, 0x0008	R/W
AUD_INT	Interrupts (L and H), see page 503	0x0009, 0x000A	RO
Interrupt status			
AUD_SYNC_STATUS	Synchronization status, see page 504	0x0040	RO
AUD_ANCCOUNT	Ancillary data, see page 504	0x0041	RO
AUD_HEAD4	Header 4, see page 505	0x0042	RO
AUD_HEAD3	Header 3, see page 505	0x0043	RO

Table 31: Audio decoder registers

Register name	Description	Address offset	Type
AUD_HEADLEN	Frame length, see page 506	0x0044, 0x0045	RO
AUD_PTS	PTS, see page 506	0x0046 to 0x004A	R/W
AUD_ERROR	Error code, see page 506	0x000F	RO
Decoding algorithm			
AUD_DECODESEL	Decoding algorithm, see page 509	0x004D	R/W
AUD_STREAMSEL	Stream selection, see page 510	0x004C	R/W
System synchronization			
AUD_PACKET_LOCK	Packet lock, see page 510	0x004F	R/W
AUD_ID_EN	Enable audio ID, see page 510	0x0050	R/W
AUD_ID	Audio ID, see page 511	0x0051	R/W
AUD_ID_EXT	Audio extension, see page 511	0x0052	R/W
AUD_SYNC_LOCK	Sync lock, see page 511	0x0053	R/W
Postdecoding and Pro Logic®			
AUD_PDEC	Postdecoder control, see page 512	0x0062	R/W
AUD_PL_AB	Pro Logic® autobalance, see page 512	0x0064	R/W
AUD_PL_DWNX	Pro Logic® decoder downmix, see page 513	0x0065	R/W
AUD_DWSMODE	Downsampling filter, see page 513	0x0070	R/W
Bass redirection			
AUD_VOLUME0	Volume of first channel, see page 514	0x004E	R/W
AUD_VOLUME1	Volume of second channel, see page 514	0x0063	R/W
AUD_OCFG	Out put configuration, see page 515	0x0066	R/W
AUD_CHAN_IDX	Channel index, see page 515	0x0067	R/W
Dolby® Digital configuration			
AUD_AC3_DECODE_LFE	Decode LFE, see page 516	0x0068	R/W
AUD_AC3_COMP_MOD	Compression mode, see page 516	0x0069	R/W
AUD_AC3_HDR	High dynamic range, see page 516	0x006A	R/W
AUD_AC3_LDR	Low dynamic range, see page 517	0x006B	R/W
AUD_AC3_RPC	Repeat count, see page 517	0x006C	R/W
AUD_AC3_KARAMODE	Karaoke downmix, see page 518	0x006D	R/W
AUD_AC3_DUALMODE	Dual downmix, see page 519	0x006E	R/W
AUD_AC3_DOWNMIX	Downmix, see page 519	0x006F	R/W
AUD_AC3_STATUS0	Dolby® Digital status 0, see page 519	0x0076	RO
AUD_AC3_STATUS1	Dolby® Digital status 1, see page 520	0x0077	RO

Table 31: Audio decoder registers

Register name	Description	Address offset	Type
AUD_AC3_STATUS2	Dolby® Digital status 2, see page 520	0x0078	RO
AUD_AC3_STATUS3	Dolby® Digital status 3, see page 520	0x0079	RO
AUD_AC3_STATUS4	Dolby® Digital status 4, see page 521	0x007A	RO
AUD_AC3_STATUS5	Dolby® Digital status 5, see page 521	0x007B	RO
AUD_AC3_STATUS6	Dolby® Digital status 6, see page 521	0x007C	RO
AUD_AC3_STATUS7	Dolby® Digital status 7, see page 522	0x007D	RO
MPEG-1 and MPEG-2 configuration			
AUD_MP_SKIP_LFE	Channel skip, see page 522	0x0068	R/W
AUD_MP_PROG_NUMBER	Program number, see page 522	0x0069	R/W
AUD_MP_DUALMODE	MPEG setup dual mode, see page 523	0x006E	R/W
AUD_MP_DRC	Dynamic range control, see page 523	0x006A	R/W
AUD_MP_CRC_OFF	CRC check off, see page 523	0x006C	R/W
AUD_MP_MC_OFF	Multichannel, see page 523	0x006D	R/W
AUD_MP_DOWNMIX	MPEG downmix, see page 524	0x006F	R/W
AUD_MP_STATUS0	MPEG status 0, see page 525	0x0076	RO
AUD_MP_STATUS1	MPEG status 1, see page 525	0x0077	RO
AUD_MP_STATUS2	MPEG status 2, see page 525	0x0078	RO
AUD_MP_STATUS3	MPEG status 3, see page 526	0x0079	RO
AUD_MP_STATUS4	MPEG status 4, see page 526	0x007A	RO
AUD_MP_STATUS5	MPEG status 5, see page 526	0x007B	RO
MP3 configuration			
AUD_CRC_OFF	CRC checking, see page 527	0x006C	RO
LPCM configuration			
AUD_DOWNSAMPLING	Audio downsampling, see page 527	0x0070	R/W
AUD_CHANNEL_ASSIGNMENT	Channel assignment, see page 527	0x00A8	R/W
AUD_MULTI_CHANNEL	Multichannel structure, see page 528	0x00A9	R/W
AUD_LPCM_DOWNMIX	LPCM downmix, see page 528	0x006F	R/W
AUD_DM_COEFT_0 to AUD_DM_COEFT_13	LPCM downmix coefficients, see page 528 to page 532	0x0096 to 0x00A3	R/W
PCM beep tone configuration			
AUD_BEEP_FREQ	PCM beep tone frequency, see page 532	0x0068	R/W
AUD_BT_CHANNELCONF	PCM beep tone channel configuration, see page 533	0x0069	R/W

Confidential

Table 31: Audio decoder registers

Register name	Description	Address offset	Type
Pink noise configuration			
AUD_PN_CHANNELCONF	PCM pink noise channel configuration, see page 534	0x0069	R/W
De-emphasis			
AUD_DEEMPH	De-emphasis, see page 535	0x00B5	R/W
Downmix			
AUD_DOWNMIX	Downmix values, see page 536	0x006F	R/W
Dual mode			
AUD_DUALMODE	Dual mode, see page 536	0x006E	R/W

Table 32: Audio interface registers

Register name	Description	Address offset	Type
Load/store4			
AUDIF_GCF	General configuration, see page 542	0x0000	R/W
AUDIF_MUX	Audio MUX configuration, see page 543	0x0004	R/W
AUDIF_PCMICFG	PCMI configuration, see page 544	0x0008	R/W
AUDIF_PCMOCFG	PCMO configuration, see page 545	0x000C	R/W

Table 33: Clock generator registers

Register name	Description	Address offset	Type
PLL_FS configuration			
PLLFSDIV0	PLL_FS predivider and feedback divider configuration, see page 564	0x0000	R/W
PLLFSDIV1	PLL_FS postdivider and electrical set up configuration, see page 564	0x0004	R/W
C201_CLK configuration			
CLKDIV0_CONFIG0	PLL_CLOCK[0] divider configuration register 0, see page 567	0x0010	R/W
CLKDIV0_CONFIG1	PLL_CLOCK[0] divider configuration register 1, see page 567	0x0014	R/W
CLKDIV0_CONFIG2	PLL_CLOCK[0] divider configuration register 2, see page 567	0x0018	R/W
STBUS_CLK configuration			
CLKDIV1_CONFIG0	PLL_CLOCK[1] divider configuration register 0, see page 567	0x0020	R/W
CLKDIV1_CONFIG1	PLL_CLOCK[1] divider configuration register 1, see page 567	0x0024	R/W
CLKDIV1_CONFIG2	PLL_CLOCK[1] divider configuration register 2, see page 567	0x0028	R/W
COMMS_CLK configuration			
CLKDIV2_CONFIG0	PLL_CLOCK[2] divider configuration register 0, see page 567	0x0030	R/W
CLKDIV2_CONFIG1	PLL_CLOCK[2] divider configuration register 1, see page 567	0x0034	R/W
CLKDIV2_CONFIG2	PLL_CLOCK[2] divider configuration register 2, see page 567	0x0038	R/W
MEMCLK_CLK configuration			
CLKDIV4_CONFIG0	PLL_CLOCK[4] divider configuration register 0, see page 567	0x0050	R/W
CLKDIV4_CONFIG1	PLL_CLOCK[4] divider configuration register 1, see page 567	0x0054	R/W
CLKDIV4_CONFIG2	PLL_CLOCK[4] divider configuration register 2, see page 567	0x0058	R/W
CLK2 configuration			
CLKDIV5_CONFIG0	PLL_CLOCK[5] divider configuration register 0, see page 567	0x0060	R/W
CLKDIV5_CONFIG1	PLL_CLOCK[5] divider configuration register 1, see page 567	0x0064	R/W
CLKDIV5_CONFIG2	PLL_CLOCK[5] divider configuration register 2, see page 567	0x0068	R/W

Confidential

Table 33: Clock generator registers

Register name	Description	Address offset	Type
CLK3 configuration			
CLKDIV6_CONFIG0	PLL_CLOCK[6] divider configuration register 0, see page 567	0x0070	R/W
CLKDIV6_CONFIG1	PLL_CLOCK[6] divider configuration register 1, see page 567	0x0074	R/W
CLKDIV6_CONFIG2	PLL_CLOCK[6] divider configuration register 2, see page 567	0x0078	R/W
DSP_CLK configuration			
CLKDIV7_CONFIG0	PLL_CLOCK[7] divider configuration register 0, see page 567	0x0080	R/W
CLKDIV7_CONFIG1	PLL_CLOCK[7] divider configuration register 1, see page 567	0x0084	R/W
CLKDIV7_CONFIG2	PLL_CLOCK[7] divider configuration register 2, see page 567	0x0088	R/W
Flash clock configuration			
SDLLDIV1_CONFIG0	SDLL_CLOCK[1] configuration register 0, see page 567	0x00D0	R/W
SDLLDIV1_CONFIG1	SDLL_CLOCK[1] configuration register 1, see page 567	0x00D4	R/W
SDLLDIV1_CONFIG2	SDLL_CLOCK[1] configuration register 2, see page 567	0x00D8	R/W
SDRAM clock configuration			
SDLLDIV2_CONFIG0	SDLL_CLOCK[2] configuration register 0, see page 567	0x00E0	R/W
SDLLDIV2_CONFIG1	SDLL_CLOCK[2] configuration register 1, see page 567	0x00E4	R/W
SDLLDIV2_CONFIG2	SDLL_CLOCK[2] configuration register 2, see page 567	0x00E8	R/W
PCMCLKIN configuration			
SYNTH0_CONFIG0	SYNTH_CLOCK[0] digital frequency synthesizer configuration register 0, see page 569	0x0120	R/W
SYNTH0_CONFIG1	SYNTH_CLOCK[0] CLOCK[8] digital frequency synthesizer configuration register 1, see page 570	0x0124	R/W
SMART_CARD_CLK configuration			
SYNTH1_CONFIG0	SYNTH_CLOCK[1] digital frequency synthesizer configuration register 0, see page 569	0x0130	R/W
SYNTH1_CONFIG1	SYNTH_CLOCK[1] digital frequency synthesizer configuration register 1, see page 570	0x0134	R/W

Table 33: Clock generator registers

Register name	Description	Address offset	Type
AUX_CLK_OUT configuration			
SYNTH2_CONFIG0	SYNTH_CLOCK[2] digital frequency synthesizer configuration register 0, see page 569	0x0140	R/W
SYNTH2_CONFIG1	SYNTH_CLOCK[2] digital frequency synthesizer configuration register 1, see page 570	0x0144	R/W
LP_CLK configuration			
LPMODE0	LP_CLOCK configuration register 0, see page 570	0x020C	R/W
LPMODE1	LP_CLOCK configuration register 1, see page 571	0x0210	R/W
LPMODE2	LP_CLOCK configuration register 2, see page 572	0x0214	R/W
REGISTER_LOCK	Lock configuration registers, see page 562	0x0300	R/W
Other registers			
SHIFT_CONFIG	Shift configuration for SDLL channels, see page 568	0x00F0	R/W
DIVIDER_MODE	Mode transitions register, see page 562	0x00F8	R/W
REDUCED_POWER	Reduced power mode, see page 563	0x00FC	R/W
CLOCK_SEL1	PLL clock MUX selection register, see page 565	0x0200	R/W
TICKTIMER	CPU_TICK configuration register, see page 572	00208	R/W

Table 34: Diagnostic controller (DCU) registers

Register name	Description	Address offset	Type
DCU_CAPABILITY	Indicates the capabilities of this version of the DCU, see page 211	0x0000	RO
DCU_CONTROL	Control CPU and DCU behavior, see page 212	0x0004	R/W
DCU_SIGNALLING	Communicate trigger events, see page 213	0x0008	R/W
DCU_STATUS	CPU and DCU status summary, see page 214	0x000C	RO
DCU_TRIGGER_IN	Trigger status, see page 215	0x0010	RO
DCU_COMPARE_STATUS	Matched status of compare blocks, see page 215	0x0014	RO
DCU_SEQUENCING_CONFIGURATION	Indicates the connectivity of the sequencing block, see page 216	0x0040	RO
DCU_TRIGGER_IN_PROPERTIES	Properties for TRIGGER_IN, see page 217	0x0080	R/W

Table 34: Diagnostic controller (DCU) registers

Register name	Description	Address offset	Type
DCU_JUMPTRACE_PROPERTIES	Properties for jump trace, see page 218	0x0100	R/W
DCU_JUMPTRACE_FROM_LPTR	IPTR prior to jump, see page 219	0x0104	RO
DCU_JUMPTRACE_TO_LPTR	Current (next) IPTR, see page 219	0x0108	RO
DCU_JUMPTRACE_LAST_LPTR	Previous IPTR stored to jump trace, see page 219	0x010C	RO
DCU_JUMPTRACE_LAST_CAPTURE0	Previous capture0 stored to jump trace, see page 220	0x0110	RO
DCU_JUMPTRACE_BYTES	Buffer for incomplete jump trace words, see page 220	0x0114	RO
DCU_JUMPTRACE_START_ADDRESS	Pointer to start of rolling jump trace buffer, see page 220	0x0118	R/W
DCU_JUMPTRACE_END_ADDRESS	Pointer to end of rolling jump trace buffer, see page 220	0x011C	R/W
DCU_JUMPTRACE_ADDRESS	Pointer to rolling jump trace buffer, see page 221	0x0120	R/W
DCU_COMPARE0_PROPERTIES	Properties for general compare block 0, see page 221	0x0200	R/W
DCU_COMPARE0_VALUE1	General compare block 0, first comparison value, see page 222	0x0204	R/W
DCU_COMPARE0_VALUE2	General compare block 0, second comparison value, see page 222	0x0208	R/W
DCU_COMPARE1_PROPERTIES	Properties for general compare block 1, see page 221	0x0210	R/W
DCU_COMPARE1_VALUE1	General compare block 1, first comparison value, see page 222	0x0214	R/W
DCU_COMPARE1_VALUE2	General compare block 1, second comparison value, see page 222	0x0218	R/W
DCU_COMPARE2_PROPERTIES	Properties for general compare block 2, see page 221	0x0220	R/W
DCU_COMPARE2_VALUE1	General compare block 2, first comparison value, see page 222	0x0224	R/W
DCU_COMPARE2_VALUE2	General compare block 2, second comparison value, see page 222	0x0228	R/W
DCU_CAPTURE0_PROPERTIES	Properties for capture block 0, see page 223	0x0400	R/W
DCU_CAPTURE0_VALUE	Captured value of capture block 0, see page 223	0x0404	R/W
DCU_CAPTURE1_PROPERTIES	Properties for capture block 1, see page 223	0x0408	R/W
DCU_CAPTURE1_VALUE	Captured value of capture block 1, see page 223	0x040C	R/W
DCU_SEQUENCING_ENABLE	Enable mask register for hardware single step, see page 224	0x0500	R/W

Table 34: Diagnostic controller (DCU) registers

Register name	Description	Address offset	Type
DCU_SEQUENCING_DISABLE	Disable mask register for hardware single step, see page 225	0x0504	R/W
DCU_SEQUENCING_TRIGGER_IN_ENABLE	Enable mask register for TRIGGER_IN, see page 224	0x0508	R/W
DCU_SEQUENCING_TRIGGER_IN_DISABLE	Disable mask register for TRIGGER_IN, see page 225	0x050C	R/W
DCU_SEQUENCING_JUMPTTRACE_ENABLE	Enable mask register for jump trace, see page 224	0x0510	R/W
DCU_SEQUENCING_JUMPTTRACE_DISABLE	Disable mask register for jump trace, see page 225	0x0514	R/W
DCU_SEQUENCING_COMPARE0_ENABLE	Enable mask register for compare0, see page 224	0x0518	R/W
DCU_SEQUENCING_COMPARE0_DISABLE	Disable mask register for compare0, see page 225	0x051C	R/W
DCU_SEQUENCING_COMPARE1_ENABLE	Enable mask register for compare1, see page 224	0x0520	R/W
DCU_SEQUENCING_COMPARE1_DISABLE	Disable mask register for compare1, see page 225	0x0524	R/W
DCU_SEQUENCING_COMPARE2_ENABLE	Enable mask register for compare2, see page 224	0x0528	R/W
DCU_SEQUENCING_COMPARE2_DISABLE	Disable mask register for compare2, see page 225	0x052C	R/W
DCU_SEQUENCING_CAPTURE0_ENABLE	Enable mask register for capture0, see page 224	0x0530	R/W
DCU_SEQUENCING_CAPTURE0_DISABLE	Disable mask register for capture0, see page 225	0x0534	R/W
DCU_SEQUENCING_CAPTURE1_ENABLE	Enable mask register for capture1, see page 224	0x0538	R/W
DCU_SEQUENCING_CAPTURE1_DISABLE	Disable mask register for capture1, see page 225	0x053C	R/W
DCU_WRANGE_ENABLE_ONLY_IN_RANGE	Enable properties for WPTR range enable block, see page 225	0x0600	R/W
DCU_WRANGE_ENABLE_ONLY_OUT_OF_RANGE	Enable properties for WPTR range enable block, see page 226	0x0604	R/W
DCU_WRANGE_LOWER	Lower value for WPTR range enable block, see page 226	0x0608	R/W
DCU_WRANGE_UPPER	Upper value for WPTR range enable block, see page 226	0x060C	R/W
DCU_HOSTED_MEMORY_VIA_TAP	Target initiated PEEK/POKE to host, see	0x0800 to 0x0FFC	R/W

Confidential

Table 35: Digital encoder registers

Register name	Description	Address offset	Type
DEN_CFG0	Configuration 0, see page 432	0x0000	R/W
DEN_CFG1	Configuration 1, see page 433	0x0001	R/W
DEN_CFG2	Configuration 2, see page 434	0x0002	R/W
DEN_CFG3	Configuration 3, see page 435	0x0003	R/W
DEN_CFG4	Configuration 4, see page 436	0x0004	R/W
DEN_CFG5	Configuration 5, see page 437	0x0005	R/W
DEN_CFG6	Configuration 6, see page 437	0x0006	R/W
DEN_CFG7	Configuration 7, see page 438	0x0007	R/W
DEN_CFG8	Configuration 8, see page 439	0x0008	R/W
DEN_STA	Status, see page 440	0x0009	RO
DEN_IDFS[1:3]	Increment for digital frequency synthesizer, see page 441	0x000A, 0x000B, 0x000C	R/W
DEN_PDFS[1:2]	Static phase offset for digital frequency synthesizer(s), see page 442	0x000D, 0x000E	R/W
DEN_WSS[1:2]	WSS data registers, see page 444	0x000F, 0x0010	R/W
DEN_DAC13	DAC1 and DAC3 multiplying factors, see page 445	0x0011	R/W
DEN_DAC45	DAC4 and DAC5 multiplying factors, see page 445	0x0012	R/W
DEN_DAC6C	DAC6 and DACC multiplying factors, see page 446	0x0013	R/W
DEN_CID	Digital encoder version identification number, see page 446	0x0018	RO
DEN_VPS1	VPS data registers, see page 447	0x0019, 0x001A, 0x001B, 0x001C, 0x001D, 0x001E	R/W
DEN_CGMS[1:3]	CGMS data registers, see page 448	0x001F, 0x0020, 0x0021	R/W
DEN_TTX[1:5]	Teletext block, see page 448	0x0022, 0x0023, 0x0024, 0x0025, 0x0026	R/W
DEN_CCF1	Closed caption characters and extended data for field 1, see page 450	0x0027, 0x0028	R/W
DEN_CCF2	Closed caption characters and extended data for field 2, see page 450	0x0029, 0x002A	R/W

Table 35: Digital encoder registers

Register name	Description	Address offset	Type
DEN_CLF1	Closed caption and extended data line insertion for field 1, see page 451	0x002B	R/W
DEN_CLF2	Closed caption and extended data line insertion for field 2, see page 452	0x002C	R/W
DEN_REG_64	Teletext configuration, see page 453	0x0040	R/W
DEN_REG_65	DAC2 multiplier and teletext, see page 453	0x0041	R/W
DEN_REG_69	Brightness, see page 454	0x0045	R/W
DEN_REG_70	Contrast, see page 454	0x0046	R/W
DEN_REG_71	Saturation, see page 454	0x0047	R/W
DEN_CFCOE[0:8]	Chroma coefficient 0 to 8, see page 455	0x0048, 0x0049, 0x004A, 0x004B, 0x004C, 0x004D, 0x004E, 0x004F, 0x0050	R/W
DEN_CDEL_LFC	Chroma delay and luma filter control, see page 456	0x0051	R/W
DEN_LFCOE[0:9]	Luma filter coefficient 0 to 9, see page 457	0x0052, 0x0053, 0x0054, 0x0055, 0x0056, 0x0057, 0x0058, 0x0059, 0x005A, 0x005B	R/W

Confidential

Table 36: Display planes registers

Register name	Description	Address offset	Type
Background color registers (offset from video base address)			
BCK_Y	Background color Y, see page 376	0x0098	R/W
BCK_U	Background color U, see page 376	0x0099	R/W
BCK_V	Background color V, see page 376	0x009A	R/W
Still picture plane registers			
TDL_CSO	TDL chrominance offset, see page 377	0x00EC	R/W
TDL_DCF	Still picture display configuration, see page 377	0x00F4	R/W
TDL_LSO	TDL luminance offset, see page 378	0x00EA	R/W
TDL_LSR	Still picture SRC luma/chroma resolution, see page 378	0x00EB, 0x00ED	R/W
TDL_SCN	Still picture scan vector, see page 379	0x00B0, 0x00B1	R/W
TDL_SWT	Picture switch line number for unrolling, see page 380	0x00C8, 0x00C9	R/W
TDL_TDW	Still picture width in macroblocks, see page 379	0x0086	R/W
TDL_TEP	Buffer pointer for the displayed even still picture, see page 380	0x00E6, 0x00E7, 0x00E8	R/W
TDL_TEP2	Buffer pointer for the displayed even still picture 2, see page 380	0x0093, 0x0094, 0x0095	R/W
TDL_TOP	Buffer pointer for the displayed odd still picture, see page 380	0x00E3, 0x00E4, 0x00E5	R/W
TDL_TOP2	Buffer pointer for the displayed odd still picture 2, see page 381	0x00B2, 0x00B3, 0x00B4	R/W
TDL_YDO	Still picture display start Y offset, see page 382	0x00EE, 0x00EF	R/W
TDL_YDS	Still picture display Y end, see page 382	0x00C6, 0x00C7	R/W
TDL_XDO	Still picture display start X offset, see page 381	0x00F0, 0x00F1	R/W
TDL_XDS	Still picture display X end, see page 381	0x00F2, 0x00F3	R/W
On screen display registers			
VID_DFP	Displayed luma frame pointer, see page 387	0x000C, 0x000D	R/W
VID_XFW	Displayed frame width, see page 395	0x0028	R/W

Table 36: Display planes registers

Register name	Description	Address offset	Type
OSD_TOP	OSD top field pointer, see page 384	0x002A	Serial R/W
OSD_BOT	OSD bottom field pointer, see page 383	0x002B	Serial R/W
VID_PAN	Pan/scan horizontal vector integer part, see page 391	0x002C, 0x002D	R/W
VID_656	Enable 656 mode, see page 385	0x0032	R/W
OSD_ACT	Active signal, see page 382	0x003E	R/W
VID_656B	656 status word set up, see page 385	0X0045	R/W
VID_YDS	Display Y end, see page 396	0x0046, 0x0047	R/W
VID_DFC	Displayed chroma frame pointer, see page 387	0x0058, 0x0059	R/W
VID_LSO	SRC luminance offset, see page 389	0x006A	R/W
VID_LSR	SRC luma/chroma resolution, see page 389	0x006B, 0x006D	R/W
VID_CSO	SRC chrominance offset, see page 385	0x006C	R/W
VID_YDO	Display Y offset, see page 395	0x006E, 0x006F	R/W
VID_XDO	Display X offset, see page 394	0x0070, 0x0071	R/W
VID_XDS	Display X end, see page 394	0x0072, 0x0073	R/W
VID_DCF	Display configuration, see page 386	0x0074, 0x0075	R/W
VID_SCN	Pan/scan vertical vector, see page 391	0x0087	R/W
VID_OUT	Output of 4:2:2 display, see page 391	0x0090	R/W
OSD_CFG	OSD configuration, see page 384	0x0091	R/W
OSD_BDW	OSD boundary weight, see page 383	0x0092	R/W
VID_MWV	Mix weight video, see page 390	0x009B	R/W
VID_MWS	Mix weight still picture, see page 390	0x009C	R/W
VID_MWSV	Mix weight still/video, see page 390	0x009D	R/W
VID_XDO_656	Display X offset (in 656 mode only), see page 394	0x00B5, 0x00B6	R/W
VID_YDO_656	Display Y offset (in 656 mode only), see page 396	0x00B8, 0x00B9	R/W
VID_DIS	Video configuration, see page 388	0x00D6	R/W

Table 36: Display planes registers

Register name	Description	Address offset	Type
VID_VFLMODE	Configure luminance of block row, see page 393	0x00DA, 0x00DB, 0x00DC, 0x00DD, 0x00DE	R/W
VID_XDS_656	Display X end (in 656 mode only), see page 395	0x00F5, 0x00F6	R/W
VID_YDS_656	Display Y end (in 656 mode only), see page 396	0x00F8 to 0x00F9	R/W
VID_VFCMODE	Configure chrominance of block row, see page 392	0x00FA, 0x00FB, 0x00FC, 0x00FD, 0x00FE	R/W

Table 37: EMI registers

Name	Function	Address offset	Type
EMI_STATUSCFG	EMI status configuration register, see page 169	0x0010	RO
EMI_STATUSLOCK	EMI status configuration lock register, see page 169	0x0018	RO
EMI_LOCK	EMI lock register, see page 169	0x0020	R/W
EMI_GENCFG	EMI general purpose register, see page 170	0x0028	R/W
EMI_SDRAMNOPGEN	EMI NOP generate register, see page 170	0x0030	WO
EMI_SDRAMMODEREG	EMI SDRAM mode register, see page 170	0x0038	WO
EMI_SDRAMINIT	EMI SDRAM initialize register, see page 171	0x0040	WO
EMI_REFRESHINIT	EMI refresh interval setting register, see page 171	0x0048	WO
EMI_FLASHCLKSEL	EMI flash burst clock select register, see page 171	0x0050	WO
EMI_SDRAMCLKSEL	EMI SDRAM clock select register, see page 172	0x0058	WO
EMI_CLKENABLE	EMI clock enable register, see page 172	0x0068	WO
EMI_CONFIGDATA0	EMI configuration 0 data register for banks 0 to 5, see page 173 and page 177	0x0100 (bank 0), 0x0140 (bank 1), 0x0180 (bank 2), 0x01C0 (bank 3), 0x0200 (bank 4), 0x0240 (bank 5)	R/W

Table 37: EMI registers

Name	Function	Address offset	Type
EMI_CONFIGDATA1	EMI configuration 1 data register for banks 0 to 5, see page 174 and page 178	0x0108 (bank 0), 0x0148 (bank 1), 0x0188 (bank 2), 0x01C8 (bank 3), 0x0208 (bank 4), 0x0248 (bank 5)	R/W
EMI_CONFIGDATA2	EMI configuration 2 data register for banks 0 to 5, see page 175 and page 178	0x0110 (bank 0), 0x0150 (bank 1), 0x0190 (bank 2), 0x01D0 (bank 3), 0x0210 (bank 4), 0x0250 (bank 5)	R/W
EMI_CONFIGDATA3	EMI configuration 3 data register for banks 0 to 5, see page 176 and page 179	0x0118 (bank 0), 0x0158 (bank 1), 0x0198 (bank 2), 0x01D8 (bank 3), 0x0218 (bank 4), 0x0258 (bank 5)	R/W

Table 38: External memory interface (EMI) buffer registers

Name	Function	Address offset	Type
BANK_0_BASE_ADDRESS	Bits 27 to 22 of base address of external memory bank 0, see page 200	0x0000	R/W
BANK_1_BASE_ADDRESS	Bits 27 to 22 of base address of external memory bank 1, see page 200	0x0010	R/W
BANK_2_BASE_ADDRESS	Bits 27 to 22 of base address of external memory bank 2, see page 201	0x0020	R/W
BANK_3_BASE_ADDRESS	Bits 27 to 22 of base address of external memory bank 3, see page 201	0x0030	R/W
BANK_4_BASE_ADDRESS	Bits 27 to 22 of base address of external memory bank 4, see page 202	0x0040	R/W
BANK_5_BASE_ADDRESS	Bits 27 to 22 of base address of external memory bank 5, see page 202	0x0050	R/W
BANKS_ENABLED	Value of the total number of enabled banks (not configured by the bank programming), see page 203	0x0060	R/W

Table 39: IEEE 1284 port registers

Register name	Description	Address offset	Type
1284CHECKSUM	1284 check sum, see page 666	0x0028	RO
1284CONTROL	1284 DMA control, see page 667	0x0008	WO
1284DATAIN	1284 data input, see page 668	0x0020	RO
1284DATAOUT	1284 data output, see page 669	0x0024	WO
1284DMAADDRESS	1284 DMA address, see page 669	0x0040	WO
1284DMACONTROL	1284 DMA control, see page 670	0x0048	R/W
1284DMACOUNT	1284 DMA count, see page 670	0x0044	R/W
1284DMATOKEN	1284 DMA token, see page 671	0x0030	R/W
1284INTACK	1284 interrupt acknowledge, see page 672	0x0054	WO
1284INTENABLE	1284 interrupt enable, see page 673	0x004C	R/W
1284INTSTATUS	1284 interrupt status, see page 674	0x0050	RO
1284MODEENABLE	1284 mode enable, see page 675	0x0000	WO
1284PACKETSIZE	1284 packet size, see page 675	0x002C	WO
1284PININ	1284 pin input, see page 676	0x0010	RO
1284PININENABLE	1284 pin input enable, see page 676	0x0014	R/W
1284PININVALUE	1284 pin output, see page 677	0x0018	R/W
1284PINOUT	1284 pin out, see page 677	0x001C	R/W
1284PULSEWIDTH	1284 pulse width, see page 678	0x0004	WO
1284STATUS	1284 status, see page 678	0x000C	RO

Table 40: IEEE1394 link layer interface (LLI) registers

Register name	Description	Address offset	Type
LLI_CONTROL	LLI control register Sets LLI mode of operation by selecting stream and clock sources, see page 281	0x0000	R/W
LLI_BYTECLOCK	Sets the frequency of the locally generated BYTECLOCK as a ratio of the system clock, see page 282	0x0008	R/W
LLI_BYTECLOCKSELECT	Bit used to select TSOSBYTECLK, see page 282	0x0018	R/W

Table 41: Infrared transmitter/receiver registers

Register name	Description	Address offset	Type
RC transmit interface registers			
IRB_TX_PRE_SCALER_IR	Clock prescaler selection, see page 596	0x0000	R/W
IRB_TX_SUB_CARRIER_IR	Subcarrier frequency programming, see page 599	0x0004	R/W
IRB_TX_SYM_PERIOD_IR ^a	Symbol period programming, see page 596	0x0008	WO
IRB_TX_ON_TIME_IR ^a	Symbol on time programming, see page 597	0x000C	WO
IRB_TX_INT_EN_IR	Transmit Interrupt enable register, see page 597	0x0010	R/W
IRB_TX_INT_STATUS_IR	Transmit Interrupt status register, see page 598	0x0014	RO
IRB_TX_EN_IR	RC transmit enable register, see page 598	0x0018	R/W
IRB_TX_CLR_UNDERRUN_IR	Clears the underrun status, see page 599	0x001C	WO
IRB_TX_SUB_CARRIER_WIDTH_IR	Subcarrier frequency programming, see page 599	0x0020	R/W
RC receive interface registers for infrared signals			
IRB_RX_ON_TIME_IR ^a	Received pulse time capture, see page 599	0x0040	RO
IRB_RX_SYM_PERIOD_IR ^a	Received symbol period capture, see page 600	0x0044	RO
IRB_RX_INT_EN_IR	Receive interrupt enable register, see page 600	0x0048	R/W
IRB_RX_INT_STATUS_IR	Receive interrupt status register, see page 601	0x004C	RO
IRB_RX_EN_IR	RC receive enable register, see page 601	0x0050	R/W
IRB_RX_MAX_SYM_PERIOD_IR	Maximum RC symbol period register, see page 601	0x0054	R/W
IRB_RX_CLR_OVERRUN_IR	Clears the overrun status, see page 602	0x0058	WO
IRB_RX_NOISE_SUPPRESS_WIDTH_IR	Noise suppression width, see page 602	0x005C	R/W
Common to RC and UHF receivers			
IRB_RX_SAMPLING_RATE_COMMON	Sampling frequency division for UHF and IR frequencies, see page 602	0x0064	R/W
RC receive interface registers for UHF signals			
IRB_RX_ON_TIME_UHF ^a	Received pulse time capture, see page 599	0x0080	RO
IRB_RX_SYM_PERIOD_UHF ^a	Received symbol period capture, see page 600	0x0084	RO
IRB_RX_INT_EN_UHF	Receive interrupt enable register, see page 600	0x0088	R/W
IRB_RX_INT_STATUS_UHF	Receive interrupt status register, see page 601	0x008C	RO

Table 41: Infrared transmitter/receiver registers

Register name	Description	Address offset	Type
IRB_RX_EN_UHF	RC receive enable register, see page 601	0x0090	R/W
IRB_RX_MAX_SYM_PERIOD_UHF	Maximum RC symbol period register, see page 601	0x0094	R/W
IRB_RX_CLR_OVERRUN_UHF	Clears the overrun status, see page 602	0x0098	WO
IRB_RX_NOISE_SUPPRESS_WIDTH_UHF	Noise suppression width, see page 602	0x009C	R/W
Reverse polarity registers			
IRB_POLINV_REG_IR	IR polarity, see page 603	0x0068	R/W
IRB_POLINV_REG_UHF	UHF polarity, see page 603	0x00A8	R/W

a. These locations have a quadruple buffer, that is, a FIFO of length 4 words.

Table 42: Interrupt system registers

Register name	Description	Address offset	Type
INC_CLEAR_EXEC	Clear a bit of the INC_EXEC register, see page 117	0x0108	WO
INC_CLEAR_MASK	Clear a bit of the interrupt enable mask, see page 117	0x00C8	WO
INC_CLEAR_PENDING	Clear a bit of the pending register, see page 117	0x0088	WO
INC_EXEC	Interrupts executing, see page 118	0x0100	R/W
INC_HANDLERWPTRn	Interrupt handler work space pointer, see page 119	0x0000, 0x0004, 0x0008, 0x000C, 0x0010, 0x0014, 0x0018, 0x001C, 0x0020, 0x0024, 0x0028, 0x002C, 0x0030, 0x0034, 0x0038, 0x003C	R/W
INC_MASK	Interrupt enable mask, see page 120	0x00C0	R/W
INC_PENDING	Interrupt pending, see page 121	0x0080	R/W
INC_SET_EXEC	Set a bit of the INC_EXEC register, see page 121	0x0104	WO
INC_SET_MASK	Set an interrupt enable mask, see page 121	0x00C4	WO
INC_SET_PENDING	Set a bit of the pending register, see page 122	0x0084	WO

Table 42: Interrupt system registers

Register name	Description	Address offset	Type
INC_TRIGGERMODEn	Interrupt trigger mode, see page 122	0x0040, 0x0044, 0x0048, 0x004C, 0x0050, 0x0054, 0x0058, 0x005C, 0x0060, 0x0064, 0x0068, 0x006C, 0x0070, 0x0074, 0x0078, 0x007C	R/W
ILC_INPUT_INTERRUPT	Input interrupt register, see page 123	0x0080 and 0x0084	RO
ILC_STATUS	Status registers, see page 123	0x0200 and 0x0204	RO
ILC_CLEAR_STATUS	Clear status locations, see page 124	0x0280 and 0x0284	WO
ILC_ENABLE	Enable registers, see page 124	0x0400 and 0x0404	R/W
ILC_CLEAR_ENABLE	Clear enable locations, see page 125	0x0480 and 0x0484	WO
ILC_SET_ENABLE	Set enable locations, see page 125	0x0500 and 0x0504	WO
ILC_WAKEUP_ENABLE	Wake up enable registers, see page 126	0x0604	R/W
ILC_WAKEUP_ACTIVE_LEVEL	Wake up level registers, see page 126	0x0684	R/W
ILC_PRIORITYn	Priority registers, see page 127	0x0800 + (n x 0x0800)	R/W
ILC_MODEn	Mode registers, see page 127	0x0800 + (n x 0x0008) + 0x0004	R/W

Confidential

Table 43: Low power module (LPM) registers

Register name	Description	Address offset	Type
LPM_TIMER	Low power timer, see page 575	0x0400 and 0x0404	R/W
LPM_TIMERSTART	Low power timer start, see page 575	0x0408	WO
LPM_ALARM	Low power alarm, see page 575	0x0410 and 0x0414	R/W
LPM_ALARMSTART	Low power alarm start, see page 576	0x0418	WO
LPM_WDENABLE	Watchdog enable, see page 576	0x0510	R/W
LPM_WDFLAG	Watchdog flag, see page 576	0x0514	RO

Table 44: Memory registers

Name	Function	Address offset	Type
CACHEING_ENABLE	Global cacheing enable, see page 135	0x0000	R/W
INVALIDATE	Start invalidate, see page 135	0x0010	WO
FLUSH	Start flush (DCache only), see page 136	0x0014	WO
STATUS	Monitor status, see page 136	0x0018	RO
REGION_0_ENABLE	Configure region 0 cacheability, see page 136	0x0020	R/W
REGION_1_BLOCK_ENABLE	Configure region 1 cacheability, see page 137	0x0028	R/W
REGION_1_TOP_ENABLE		0x002C	R/W
REGION_2_ENABLE	Configure region 2 cacheability, see page 138	0x0030	R/W
REGION_3_BLOCK_ENABLE	Configure region 3 cacheability, see page 139	0x0038	R/W
REGION_3_BANK_ENABLE		0x003C	R/W

Confidential

Table 45: Modem analog front-end interface (MAFEIF) registers

Register name	Description	Address offset	Type
MOD_CONTROL	MAFEIF control, see page 589	0x0000	R/W
MOD_STATUS	MAFEIF status, see page 589	0x0004	RO
MOD_INT_ENABLE	Interrupt enable, see page 590	0x0008	R/W
MOD_ACK	Acknowledge, see page 590	0x000C	WO
MOD_BUFFER_SIZE	Buffer size, see page 590	0x0010	R?W
MOD_MAFE_CTRL	MAFEIF control, see page 590	0x0014	WO
MOD_MAFE_STATUS	MAFEIF status, see page 591	0x0018	RO
MOD_RECEIVE0_POINTER	Receive memory buffer 0 start address, see page 591	0x0020	R?W
MOD_RECEIVE1_POINTER	Receive memory buffer 1 start address, see page 591	0x0024	R/W
MOD_TRANSMIT0_POINTER	Transmit memory buffer 0 start address, see page 591	0x0028	R?W
MOD_TRANSMIT1_POINTER	Transmit memory buffer 1 start address, see page 592	0x002C	R/W

Table 46: MPEG video decoder registers

Register name	Description	Address offset	Type
MPEGCONTROL	MPEG control register, see page 306	0x5000	R/W
VID_ABG	Start of audio bit buffer, see page 307	0x001C and 0x001D	R/W
VID_ABL	Audio bit buffer level, see page 307	0x001E and 0x001F	RO
VID_ABS	Audio bit buffer stop, see page 307	0x0020 and 0x0021	R/W
VID_ABT	Audio bit buffer threshold, see page 308	0x0022 and 0x0023	R/W
VID_BFC	Backward chroma pointer, see page 308	0x005E and 0x005F	R/W
VID_BFP	Backward frame pointer, see page 308	0x0012 and 0x0013	R/W
VID_CD_COUNT	Bit buffer input counter, see page 309	0x0067	RO
VID_CTL	Decoding control, see page 309	0x0002	R/W
VID_CWL	CD write launch, see page 309	0x0031	R/W
VID_DFS	Decoded frame size, see page 310	0x0024	R/W
VID_DFW	Decoded frame width, see page 310	0x0025	R/W
VID_FFC	Forward chroma frame pointer, see page 310	0x005C and 0x005D	R/W

Table 46: MPEG video decoder registers

Register name	Description	Address offset	Type
VID_FFP	Forward luma frame pointer, see page 311	0x0010 and 0x0011	R/W
VID_HDF	Header data FIFO, see page 311	0x0066	RO
VID_HDS	Header search, see page 312	0x0069	R/W
VID_ITM	Interrupt mask, see page 312	0x003C, 0x0060 and 0x0061	R/W
VID_ITS	Interrupt status, see page 313	0x003D, 0x0062 and 0x0063	RO
VID_LDP	Load pointer, see page 313	0x003F	R/W
VID_PFH	Picture f-parameters horizontal, see page 314	0x0004	R/W
VID_PFV	Picture f-parameters vertical, see page 314	0x0005	R/W
VID_PPR1	Picture parameters 1, see page 315	0x0006	R/W
VID_PPR2	Picture parameters 2, see page 316	0x0007	R/W
VID_PTH	Panic threshold, see page 316	0x002E and 0x002F	R/W
VID_QMW	Quantization matrix data, see page 317	0x0076	WO
VID_LCK	Memory configuration lock, see page 317	0x007B	R/W
VID_RFC	Reconstructed chroma frame pointer, see page 317	0x005A and 0x005B	R/W
VID_RFP	Reconstructed frame pointer, see page 318	0x000E and 0x000F	R/W
VID_SCDCOUNT	Bit buffer output controller, see page 318	0x0068	RO
VID_SPB	Subpicture buffer begin, see page 318	0x0050 and 0x0051	R/W
VID_SPE	Subpicture buffer end, see page 319	0x0052 and 0x0053	R/W
VID_SPREAD	Subpicture read pointer, see page 319	0x004E	R/W
VID_SPWRITE	Subpicture write pointer, see page 320	0x004F	R/W
VID_SRA	Audio soft reset, see page 320	0x0035	R/W
VID_SRV	Video soft reset, see page 320	0x0039	R/W
VID_STA	Status, see page 321	0x003B, 0x0064 and 0x0065	R/W
VID_STL	SCD trick mode launch, see page 322	0x0030	R/W
VID_TIS	Task instruction, see page 323	0x0003	WO
VID_TP_CD	CD pointer load address, see page 323	0x00CA, 0x00CB and 0x00CC	R/W

Table 46: MPEG video decoder registers

Register name	Description	Address offset	Type
VID_TP_CDLIMIT	CD write limit address, see page 324	0x00E2, 0x00E1 and 0x00E0	R/W
VID_TP_CD_RD	CD pointer address, see page 324	0x0085, 0x0084 and 0x0083	RO
VID_TP_SCD	SCD pointer load address, see page 324	0x00C0, 0x00C1 and 0x00C2	R/W
VID_TP_SCD_CURRENT	SCD pointer current address, see page 325	0x00CD, 0x00CE and 0x00CF	RO
VID_TP_SCD_RD	SCD pointer VLD load address, see page 325	0x0080, 0x0081 and 0x0082	RO
VID_TP_SCDLIMIT	SCD read limit address, see page 318	0x00BA, 0x00BB and 0x00BC	R/W
VID_TP_VLD	VLD pointer load address, see page 325	0x00D4 and 0x00D5	R/W
VID_TP_VLD_RD	VLD pointer current address, see page 325	0x00C3, 0x00C4 and 0x00C5	RO
VID_TRF	Temporal reference, see page 326	0x0056 and 0x0057	R/W
VID_VBG	Start of video bit buffer, see page 326	0x0014 and 0x0015	R/W
VID_VBL	Video bit buffer level, see page 327	0x0016 and 0x0017	RO
VID_VBS	Video bit buffer stop, see page 327	0x0010 and 0x0019	R/W
VID_VBT	Video bit buffer threshold, see page 327	0x001A and 0x001B	R/W
CFG_CCF	Chip configuration, see page 328	0x0001	R/W
CFG_CDR	Compressed data input, see page 328	0x0044	WO
CFG_DRC	DRAM configuration, see page 329	0x0038	R/W
CFG_GCF	General configuration, see page 329	0x003A	R/W
CFG_MCF	Memory refresh interval, see page 330	0x0000	R/W

Confidential

Table 46: MPEG video decoder registers

Register name	Description	Address offset	Type
PES_CF1	PES audio decoding control, see page 330	0x0040	R/W
PES_CF2	PES video parser control, see page 331	0x0041	R/W
PES_TM1	DSM trick mode, see page 331	0x0042	RO
PES_TM2	PES parser status, see page 331	0x0043	RO
PES_TS	PES time stamps, see page 332	0x0049 to 0x004D	RO

Table 47: Padlogic registers

Register name	Description	Address offset	Type
EMI_GENCFG	General purpose configuration outputs, see page 193	0x0028 ^a	R/W
CONFIG_DEVICEID_REG	Device ID register, see page 194	0x0010 ^b	RO
CONFIG_CONTROL_A	Interconnect configuration control register A, see page 194	0x0000 ^b	R/W
CONFIG_CONTROL_B	Interconnect configuration control register B, see page 195	0x0004 ^b	R/W
CONFIG_CONTROL_C	Interconnect configuration control register C, see page 196	0x0008 ^b	R/W
CONFIG_CONTROL_D	Interconnect configuration control register D, see page 197	0x000C ^b	R/W
CONFIG_CONTROL_E	Interconnect configuration control register E, see page 198	0x0028 ^b	R/W

a. Uses *EMIBaseAddress*b. Uses *InterconnectBaseAddress*

Table 48: Parallel I/O port registers

Register name	Description	Address offset	Type
PIO_CLEAR_PnC[2:0]	Clear bits of PnC[2:0], see page 652	0x0028 to 0x0048	WO
PIO_CLEAR_PnCOMP	Clear bits of PnCOMP, see page 652	0x0058	WO
PIO_CLEAR_PnMASK	Clear bits of PnMASK, see page 653	0x0068	WO
PIO_CLEAR_PnOUT	Clear bits of PnOUT, see page 653	0x0008	WO
PIO_PnC[2:0]	PIO configuration, see page 654	0x0020 to 0x0040	R/W
PIO_PnCOMP	PIO input comparison, see page 655	0x0050	R/W
PIO_PnIN	PIO input, see page 655	0x0010	RO
PIO_PnMASK	PIO input comparison mask, see page 655	0x0060	R/W

Table 48: Parallel I/O port registers

Register name	Description	Address offset	Type
PIO_PnOUT	PIO output, see page 656	0x0000	R/W
PIO_SET_PnC[2:0]	Set bits of PnC[2:0], see page 656	0x0024 to 0x0044	WO
PIO_SET_PnCOMP	Set bits of PnCOMP, see page 656	0x0054	WO
PIO_SET_PnMASK	Set bits of PnMASK, see page 656	0x0064	WO
PIO_SET_PnOUT	Set bits of PnOUT, see page 657	0x0004	WO

Table 49A: Programmable transport interface (PTI) registers: DMA

Register name	Description	Address offset		Type
		PTI3	PTI1	
PTI_DMA0BASE	Base address for channel 0, see page 252	0x1000	0x1000	R/W
PTI_DMA0TOP	Top of the circular buffer for channel 0, see page 256	0x1004	0x1010	R/W
PTI_DMA0WRITE	Absolute write pointer for channel 0, see page 256	0x1008	0x1020	R/W
PTI_DMA0READ	Absolute read pointer for channel 0, see page 254	0x100C	0x1030	R/W
PTI_DMA0SETUP	Channel 0 holdoff enable, burst/word enable (PTI3 mode only), see page 255	0x1010	Not used	R/W
PTI_DMA0HOLDOFF	Write size and holdoff limits for channel 0, (PTI3 mode only) see page 253	0x1014	Not used	R/W
PTI_DMA0STATUS	Reports the current status, see page 251	0x1018	0x1040	R/W
PTI_DMAENABLE	Global DMA channel enables, see page 258	0x101C	0x1050	R/W
PTI_DMADCADDR	CD FIFO page address (PTI1 mode only), see page 257	Not used	0x1060	WO
PTI_DMA1BASE	Base address for channel 1, see page 252	0x1020	0x1004	R/W
PTI_DMA1TOP	Top of the circular buffer for channel 1, see page 256	0x1024	0x1014	R/W
PTI_DMA1WRITE	Absolute write pointer for channel 1, see page 256	0x1028	0x1024	R/W
PTI_DMA1READ	Absolute read pointer for channel 1, see page 254	0x102C	0x1034	R/W
PTI_DMA1BURST	Byte or word transfers (PTI1 mode only), see page 252	Not used	0x1044	R/W
PTI_DMA1SETUP	Byte not word mode and block move for channel 1 (PTI3 mode only), see page 255	0x1030	Not used	R/W
PTI_DMA1HOLDOFF	Write size and holdoff limits for channel 1, see page 253	0x1034	0x1054	R/W
PTI_DMA1CDADDR	Configuration of CD FIFO address for channel 1, see page 257	0x1038	0x1064	R/W

Table 49A: Programmable transport interface (PTI) registers: DMA

Register name	Description	Address offset		Type
		PTI3	PTI1	
PTI_DMASECSTART	Start address the of current section, see page 258	0x103C	0x1074	R/W
PTI_DMA2BASE	Base address for channel 2, see page 252	0x1040	0x1008	R/W
PTI_DMA2TOP	Top of the circular buffer for channel 2, see page 256	0x1044	0x1018	R/W
PTI_DMA2WRITE	Absolute write pointer for channel 2, see page 256	0x1048	0x1028	R/W
PTI_DMA2READ	Absolute read pointer for channel 2, see page 254	0x104C	0x1038	R/W
PTI_DMA2BURST	Byte or word transfers (PTI1 mode only), see page 252	Not used	0x1048	R/W
PTI_DMA2SETUP	Byte not word mode and block move for channel 2 (PTI3 mode only), see page 255	0x1050	Not used	R/W
PTI_DMA2HOLDOFF	Write size and holdoff limits for channel 2, see page 253	0x1054	0x1058	R/W
PTI_DMA2CDADDR	Configuration of CD FIFO address for channel 2, see page 257	0x1058	0x1068	R/W
PTI_DMAFLUSH	Tell the DMA to flush ready for a soft reset, see page 258	0x105C	0x1078	R/W
PTI_DMA3BASE	Base address for channel 3, see page 252	0x1060	0x100C	R/W
PTI_DMA3TOP	Top of the circular buffer for channel 3, see page 256	0x1064	0x101C	R/W
PTI_DMA3WRITE	Absolute write pointer for channel 3, see page 256	0x1068	0x102C	R/W
PTI_DMA3READ	Absolute read pointer for channel 3, see page 254	0x106C	0x103C	R/W
PTI_DMA3BURST	Byte or word transfers (PTI1 mode only), see page 252	Not used	0x104C	R/W
PTI_DMA3SETUP	Byte not word mode and blockmove for channel 3 (PTI3 mode only), see page 255	0x1070	Not used	R/W
PTI_DMA3HOLDOFF	Write size and holdoff limits for channel 3, see page 253	0x1074	0x105C	R/W
PTI_DMA3CDADDR	Configuration of CD FIFO address for channel 3, see page 257	0x1078	0x106C	R/W
PTI_DMAPTIPROG	Switch between PTI1 and PTI3 memory maps for DMA, see page 259	0x107C	-	R/W

Confidential

Table 49B: Programmable transport interface (PTI) registers: others

Register name	Description	Address offset	Type
Input interface registers			
PTI_IIFALTFIFOCOUNT	IIF alternative FIFO count, see page 259	0x2004	RO
PTI_IIFALTLATENCY	IIF alternative output latency, see page 259	0x2010	WO
PTI_IIFFIFOCOUNT	IIF count, see page 260	0x2000	RO
PTI_IIFFIFOENABLE	IIF FIFO enable, see page 260	0x2008	R/W
PTI_IIFSYNCDROP	IIF sync drop, see page 260	0x2018	WO
PTI_IIFSYNCKLOCK	IIF sync lock, see page 260	0x2014	WO
PTI_IIFSYNCCONFIG	IIF sync configuration, see page 261	0x201C	WO
PTI_IIFSYNCPERIOD	IIF sync period, see page 261	0x2020	WO
PTI configuration registers			
PTI_AUDPTS	Audio presentation time stamp, see page 261	0x0040 and 0x0044	RO
PTI_INTACK	PTI interrupt acknowledgment, see page 262	0x0020 to 0x002C	WO
PTI_INTENABLE	PTI interrupt enable, see page 262	0x0010 to 0x001C	R/W
PTI_INTSTATUSn	PTI interrupt status, see page 263	0x0000 to 0x000C	RO
PTI_VIDPTS	Video presentation time stamp, see page 263	0x0048 to 0x004C	RO
PTI_STCTIMER	Set STC timer, see page 264	0x0050 to 0x0054	WO
Section filter registers			
PTI_SFFILTERDATAn	Section filter data, see page 265	0x4000 to 0x41F8	R/W
PTI_SFFILTERMASKn	Section filter mask, see page 266	0x4004 to 0x41FC	R/W
PTI_SFNOTMATCHn	Section filter not match mode for CAM A, see page 267	0x4400 to 0x447C	R/W
Transport controller mode register			
PTI_TCMODE	Transport controller mode, see page 267	0x0030	R/W

Confidential

Table 50: PWM and counter module registers

Register name	Description	Address offset	Type
PWM_nCAPTUREEDGE	PWM n capture event definition, see page 579	0x0030 to 0x003C	R/W
PWM_nCAPTUREVAL	PWM n capture value, see page 580	0x0010 to 0x001C	RO
PWM_nCOMPAREOUTVAL	PWM n compare output value, see page 581	0x0040 to 0x004C	R/W
PWM_nCOMPAREVAL	PWM n compare value, see page 581	0x0020 to 0x002C	R/W
PWM_nVAL	PWM n pulse width, see page 582	0x0000 to 0x000C	R/W
PWM_CAPTURECOUNT	PWM capture and compare counter, see page 582	0x0064	R/W
PWM_CONTROL	PWM control register, see page 583	0x0050	R/W
PWM_COUNT	PWM output counter, see page 583	0x0060	R/W
PWM_INTACK	PWM interrupt acknowledge, see page 584	0x005C	WO
PWM_INTENABLE	PWM interrupt enable, see page 585	0x0054	R/W
PWM_INTSTATUS	PWM interrupt status, see page 586	0x0058	RO

Table 51: Smartcard interface registers

Register name	Description	Address offset	Type
SCI_n_CLKCON	Smartcard n clock control, see page 627	0x0004	WO
SCI_n_CLKVAL	Smartcard n clock, see page 627	0x0000	WO

Table 52: Subpicture decoder registers

Register name	Description	Address offset	Type
SPD_CTL1	Control register 1, see page 338	0x0000	R/W
SPD_CTL2	Control register 2, see page 339	0x0002	R/W
SPD_HCN	Highlight region contrast, see page 339	0x0016 and 0x0017	R/W
SPD_HCOL	Highlight region color, see page 339	0x0014 and 0x0015	R/W
SPD_HLEX	Highlight region end X, see page 340	0x0010 and 0x0011	R/W
SPD_HLEY	Highlight region end Y, see page 340	0x0012 and 0x0013	R/W
SPD_HLSX	Highlight region start X, see page 340	0x000C and 0x000D	R/W
SPD_HLSY	Highlight region start Y, see page 340	0x000E and 0x000F	R/W
SPD_LUT	Main look-up table, see page 341	0x0003	WO
SPD_SPR	Soft reset, see page 341	0x0001	R/W
SPD_SXD0	Subpicture display area, see page 341	0x0024 and 0x0025	R/W
SPO_SXD1	Subpicture display area, see page 342	0x0028 and 0x0029	R/W
SPD_SYD0	Subpicture display area, see page 342	0x0026 and 0x0027	R/W
SPD_SYD1	Subpicture display area, see page 342	0x002A and 0x002B	R/W
SPD_XD0	Subpicture X offset, see page 343	0x0004 and 0x0005	R/W
SPD_YD0	Subpicture Y offset, see page 343	0x0006 and 0x0007	R/W

Table 53: Synchronous serial controller (SSC) registers

Register name	Description	Address offset	Type
SSCnBRG	SSC n baudrate generation, see page 644	0x0000	R/W
SSCnCON	SSC n control, see page 645	0x000C	R/W
SSCnI2C	SSC n I ² C control, see page 646	0x0018	R/W
SSCnIEN	SSCn interrupt enable, see page 647	0x0010	R/W
SSCnRBUF	SSC n receive buffer, see page 647	0x0008	RO
SSCnSLAD	SSC n slave address, see page 648	0x001C	WO
SSCnSTAT	SSC n status, see page 648	0x0014	RO
SSCnTBUF	SSC n transmit buffer, see page 649	0x0004	WO
CLEAR_STATUS_SSC	SSC clear bit operation, see page 649	0x0080	R/W
NOISE_SUPPRESS_WIDTH_SSC	Noise suppression width, see page 649	0x0100	R/W
PRE_SCALER_SSC	Clock prescaler, see page 650	0x0104	R/W

Table 54: Teletext DMA registers

Register name	Description	Address offset	Type
TTXT_ABORT	Teletext abort, see page 404	0x0024	R/W
TTXT_ACKODDEVEN	Teletext acknowledge odd or even, see page 404	0x0020	R/W
TTXT_DMAADDRESS	Teletext DMA address, see page 404	0x0000	R/W
TTXT_DMACOUNT	Teletext DMA count, see page 405	0x0004	R/W
TTXT_INTENABLE	Teletext interrupt enable, see page 405	0x001C	R/W
TTXT_INTSTATUS	Teletext interrupt status, see page 406	0x0018	R/W
TTXT_MODE	Teletext mode, see page 406	0x0014	R/W
TTXT_OUTDELAY	Teletext output delay, see page 406	0x0008	R/W

Table 55: Transport stream multiplexor (TSMUX) registers

Register name	Description	Address offset	Type
TSMUX_TSiSnMODE	Input modes for transport streams 0 to 2, see page 274	0x0008 to 0x0010	R/W
TSMUX_SWTS	Software stream data register (SWTS), see page 275	0x0028	R/W
TSMUX_SWTSCONFIG	SWTS configuration, see page 275	0x0030	R/W
TSMUX_PTIA_SOURCE	Transport stream input source for PTI A, see page 274	0x0040	R/W
TSMUX_TSOUT_SOURCE	PTI source for transport stream output (TSOUT), see page 276	0x00A0	R/W
TSMUX_TSOUT_CLKSOURCE	Source for transport stream output clock, see page 276	0x00A8	R/W
TSMUX_CLOCKGEN	Speed select for locally generated byte clock, see page 277	0x00B0	R/W

Confidential

ST Confidential (iPanel)

8 Central processing unit (CPU)

8.1 Overview

The central processing unit (CPU) is the ST20 32-bit processor core. It contains instruction processing logic, instruction and data pointers, and a three registers evaluation stack. It can directly access the high speed on-chip memory, which can store data or programs. Where larger amounts of memory are required, the processor can access memory via the external memory interface (EMI). The processor provides high performance features:

- fast integer multiply: 4 cycle multiply,
- fast bit shift: single cycle barrel shifter,
- byte and part word handling,
- scheduling and interrupt support,
- 64-bit integer arithmetic support.

The scheduler provides a single level of preemption. In addition, multilevel preemption is provided by the interrupt subsystem. Additionally, there is a per priority trap handler to improve the support for arithmetic errors and illegal instructions, refer to [Section 8.7: Traps and exceptions](#) on page 86.

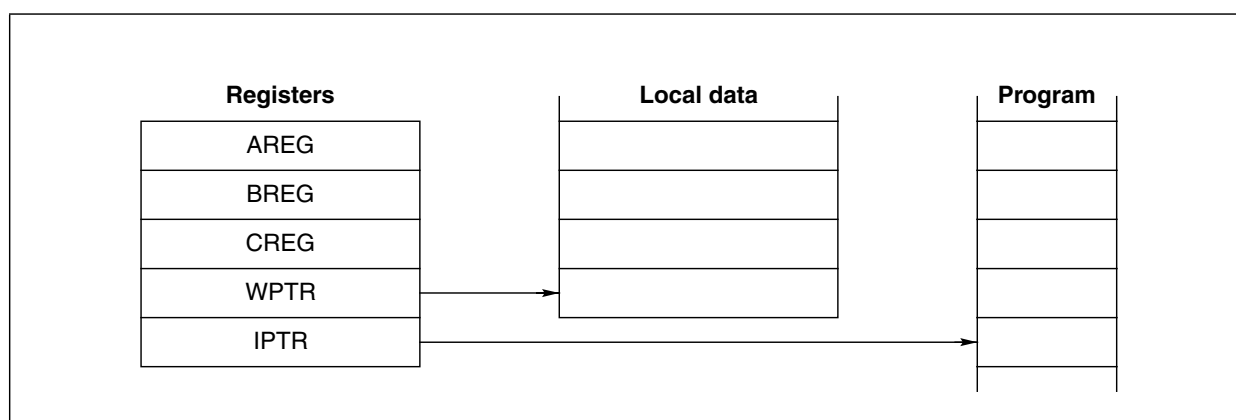
8.2 Registers used in sequential integer processes

The CPU contains six registers which are used in the execution of a sequential integer process. The six registers are:

- the work space pointer (WPTR) which points to an area of store where local data is kept,
- the instruction pointer (IPTR) which points to the next instruction to be executed,
- the status register (STATUS),
- the AREG, BREG and CREG registers which form an evaluation stack.

The AREG, BREG and CREG registers are the sources and destinations for most arithmetic and logical operations. Loading a value into the stack pushes BREG into CREG, and AREG into BREG, before loading AREG. Storing a value from AREG, pops BREG into AREG and CREG into BREG. CREG is left undefined.

Figure 11: Registers used in sequential integer processes



Expressions are evaluated on the evaluation stack, and instructions refer to the stack implicitly. For example, the **add** instruction adds the top two values in the stack and places the result on the top of the stack. The use of a stack removes the need for instructions to explicitly specify the location of their operands. No hardware mechanism is provided to detect that more than three

values have been loaded on to the stack; it is easy for the compiler to ensure that this never happens.

Note: A location in memory can be accessed relative to the work space pointer, enabling the work space to be of any size. The use of shadow registers provides fast, simple and clean context switching.

8.3 Processes and concurrency

This section describes the default behavior of the CPU.

Note: This behavior can be altered, for example by disabling timeslicing or installing a user scheduler.

A process starts, performs a number of actions, and then either stops without completing or terminates complete. Typically, a process is a sequence of instructions. The CPU can run several processes in parallel (concurrently). Processes may be assigned either high or low priority, and there may be any number of each.

The processor has a microcoded scheduler which enables any number of concurrent processes to be executed together, sharing the processor time. This removes the need for a software kernel, although kernels can still be written if desired. At any time, a process may be:

- active:
 - being executed,
 - interrupted by a higher priority process,
 - on a list waiting to be executed,
- inactive:
 - waiting to input,
 - waiting to output,
 - waiting until a specified time.

The scheduler operates so that inactive processes do not consume any processor time. Each active high priority process executes until it becomes inactive. The scheduler allocates a portion of the processor's time to each active low priority process in turn (see [Section 8.4: Priority on page 83](#)). Active processes waiting to be executed are held in two linked lists of process work spaces, one of high priority processes and one of low priority processes. Each list is implemented using two registers, one of which points to the first process in the list, the other to the last. In the linked process list shown in [Figure 12](#), process *S* is executing and *P*, *Q* and *R* are active, awaiting execution. Only the low priority process queue registers are shown; the high priority process ones behave in a similar manner.

Figure 12: Linked process lists

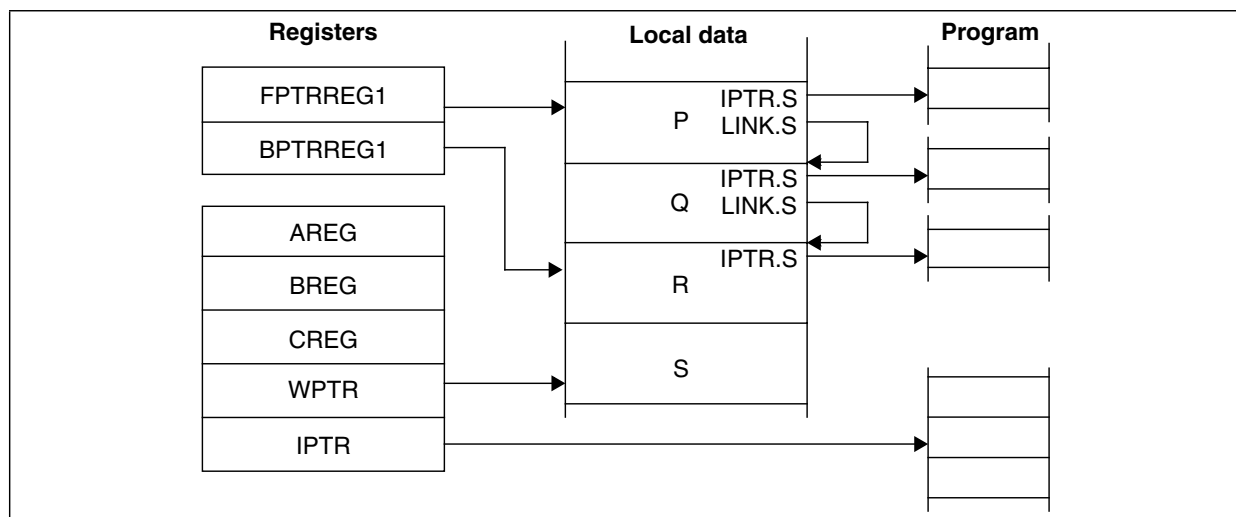


Table 56: Priority queue control registers

Function	High priority	Low priority
Pointer to front of active process list	FPTRREG0	FPTRREG1
Pointer to back of active process list	BPTRREG0	BPTRREG1

Each process runs until it has completed its action or is descheduled. In order for several processes to operate in parallel, a low priority process is only permitted to execute for a maximum of two timeslice periods. After this, the machine deschedules the current process at the next timeslicing point, adds it to the end of the low priority scheduling list and executes the next active process. The timeslice period is 1 ms.

There are only certain instructions at which a process may be descheduled. These are known as descheduling points. A process may only be timesliced at certain descheduling points. These are known as timeslicing points and are defined in such a way that the operand stack is always empty. This removes the need for saving the operand stack when timeslicing. As a result, an expression evaluation can be guaranteed to execute without the process being timesliced part way through.

Whenever a process is unable to proceed, its instruction pointer is saved in the process work space and the next process is taken from the list.

The processor core provides a number of special instructions to support the process model, including **startp** (start process) and **endp** (end process). When a main process executes a parallel construct, **startp** is used to create the necessary additional concurrent processes. A **startp** instruction creates a new process by adding a new work space to the end of the scheduling list and enabling the new concurrent process to be executed together with the ones already being executed. When a process is made active it is always added to the end of the list, and thus cannot preempt processes already on the same list.

The correct termination of a parallel construct is assured by use of the **endp** instruction. This uses a data structure that includes a counter of the parallel construct components which have still to terminate. The counter is initialized to the number of components before the processes are started. Each component ends with an **endp** instruction which decrements and tests the counter. For all but the last component, the counter is nonzero and the component is descheduled. For the last component, the counter is zero and the main process continues.

8.4 Priority

This section describes default behavior of the CPU.

Note: This behavior can be altered, for example, by disabling timeslicing and priority interrupts.

The processor can execute processes at one of two priority levels, one level for urgent (high priority) processes, one for less urgent (low priority) processes. A high priority process always executes in preference to a low priority process if both are able to do so.

High priority processes are expected to execute for a short time. If one or more high priority processes are active, then the first on the queue is selected and executes until it has to wait for a communication, a timer input, or it completes processing.

If no process at high priority is active, but one or more processes at low priority are active, then one is selected. Low priority processes are periodically timesliced to provide an even distribution of processor time between tasks which use a lot of computation.

If there are n low priority processes, then the maximum latency from the time at which a low priority process becomes active to the time when it starts processing is the order of $2n$ timeslice periods. It is then able to execute for between one and two timeslice periods, less any time taken

by high priority processes. This assumes that no process monopolizes the time of the CPU; that is, it has frequent timeslicing points.

The specific condition for a high priority process to start execution is that the CPU is idle or running at low priority and the high priority queue is nonempty.

If a high priority process becomes able to run while a low priority process is executing, the low priority process is temporarily stopped and the high priority process is executed. During execution of the high priority process, the state of low priority process is preserved in the shadow registers (see *Shadow registers* on page 92). When no further high priority processes are able to run, the state of the interrupted low priority process is reloaded from the shadow registers and the interrupted low priority process continues executing. Instructions are provided on the processor core to allow a high priority process to store the shadow registers to memory and to load them from memory. Instructions are also provided to allow a process to exchange an alternative process queue for either priority process queue. These instructions allow extensions to be made to the scheduler for custom run time kernels.

A low priority process may be interrupted after it has completed execution of any instruction. In addition, to minimize the time taken for an interrupting high priority process to start executing, the potentially time consuming instructions are interruptible. Also some instructions may be aborted, and restarted when the process next becomes active (see [Chapter 10: Instruction set](#) on page 97).

8.5 Process communications

Communication between processes takes place over channels, and is implemented in hardware. Communication is point-to-point, synchronized and unbuffered. As a result, a channel needs no process queue, no message queue and no message buffer. A channel between two processes executing on the same CPU is implemented by a single word in memory; a channel between processes executing on different processors is implemented by point-to-point links. The processor provides a number of operations to support message passing, the most important being **in** (input message) and **out** (output message).

The **in** and **out** instructions use the address of the channel to determine whether the channel is internal or external. This means that the same instruction sequence can be used for both hard and soft channels, allowing a process to be written and compiled without knowledge of where its channels are implemented.

Communication takes place when both the input and output processes are ready. Consequently, the process which first becomes ready must wait until the second one is also ready. The input and output processes only become active when the communication has completed. A process performs an input or output by loading the evaluation stack with a pointer to a message, the address of a channel, and a count of the number of bytes to be transferred, and then executing an **in** or **out** instruction.

Note: The STi5516 does not have an OS Link for interprocessor point-to-point links, nor does it implement hard channels for any of its peripherals. The only use of channels is for soft channels between two processes executing on the CPU.

8.6 Timers

There are two 32-bit hardware timer clocks which tick periodically. These are independent of any on-chip peripheral real time clock. The timers provide accurate process timing, allowing processes to deschedule themselves until a specific time.

One timer is accessible only to high priority processes and is incremented every microsecond, cycling completely in 4295 seconds. The other is accessible only to low priority processes and

runs 64 times slower, giving 15625 ticks per second. It has a full period of approximately 76 hours.

Actual timer speeds are derived from the external 27 MHz input. The periods may be calculated as follows:

The high priority clock period is derived from the clock generator TICKTIMER register (see register *TICKTIMER* on page 572) the default is 27 MHz. This is divided by 27 to give a 1 μ s high priority tick.

The low priority clock period equals the high priority clock period x 64.

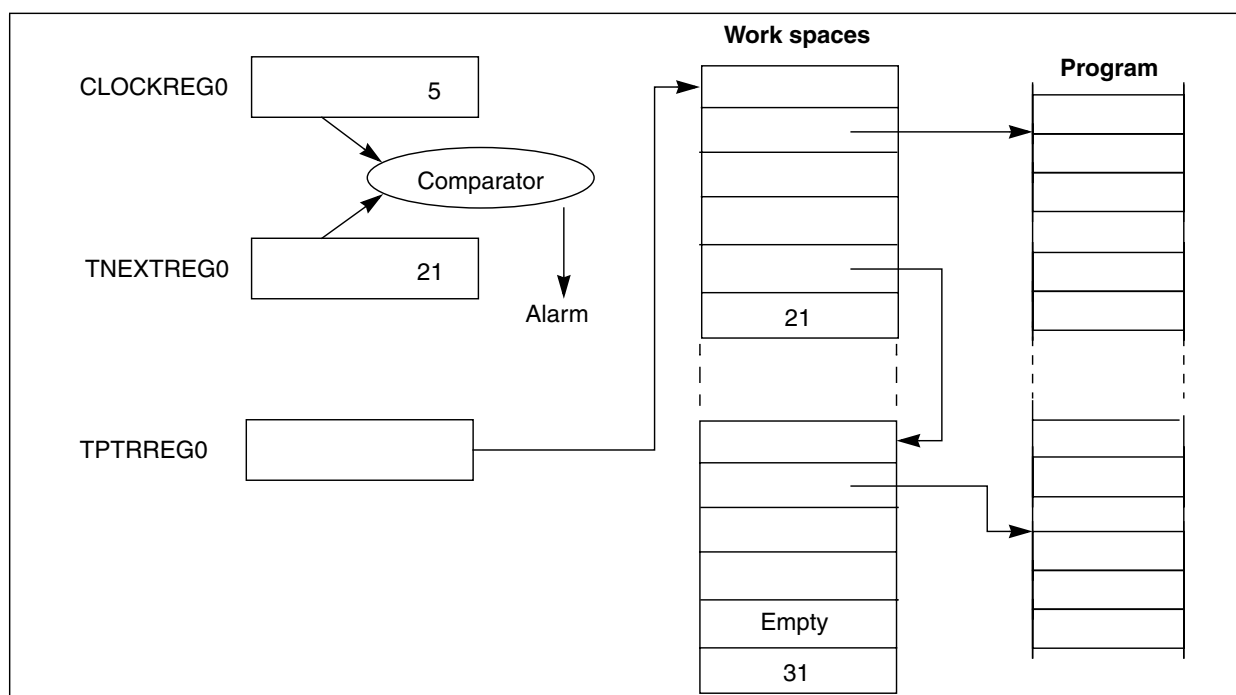
Table 57: Timer registers

Register	Function
CLOCKREG0	Current value of high priority (level 0) process clock.
CLOCKREG1	Current value of low priority (level 1) process clock.
TNEXTREG0	Indicates time of earliest event on high priority (level 0) timer queue.
TNEXTREG1	Indicates time of earliest event on low priority (level 1) timer queue.
TPTRREG0	High priority timer queue.
TPTRREG1	Low priority timer queue.

The current value of the processor clock can be read by executing an **ldtimer** (load timer) instruction. A process can arrange to perform a **tin** (timer input), in which case it is ready to execute after a specified time has been reached. The **tin** instruction requires a time to be specified. If this time is in the past then the instruction has no effect. If the time is in the future then the process is descheduled. When the specified time is reached the process becomes active. In addition, the **ldclock** (load clock) and **stclock** (store clock) instructions allow total control over the clock value. The **clockenb** (clock enable) and **clockdis** (clock disable) instructions allow each clock to be individually stopped and restarted.

Figure 13 shows two processes waiting on the timer queue, one waiting for time 21, the other for time 31.

Figure 13: Timer registers



8.7 Traps and exceptions

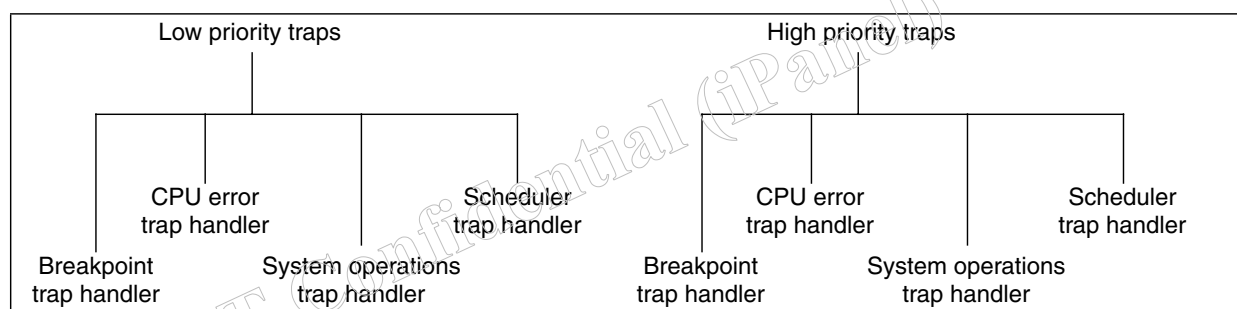
A software error, such as arithmetic overflow or array bounds violation, can cause an error flag to be set in the CPU. The flag can be ignored or the CPU stopped. Stopping the CPU on an error means that the error cannot cause further corruption. As well as containing the error in this way it is possible to determine the state of the CPU and its memory at the time the error occurred. However, the error flag feature is not directly supported in the software toolset as it has been superseded by using the error trap handler. If a trap handler process is installed, a variety of traps or exceptions can be trapped and handled by software. A user supplied trap handler routine can be provided for each high and low process priority level. The handler is started when a trap occurs and is given the reason for the trap. The trap handler is not re-entrant and must not cause a trap itself within the same group. All traps can be individually masked.

Note: The software toolset installs a null error trap handler by default. Therefore a breakpoint can be set to stop the CPU.

8.7.1 Trap groups

The trap mechanism is arranged on a per priority basis. For each priority there is a handler for each group of traps, as shown in Figure 14.

Figure 14: Trap arrangement



There are four groups of traps, as detailed below.

- **Breakpoint:** This group consists of the BREAKPOINT trap. The breakpoint instruction (**j0**) calls the breakpoint routine via the trap mechanism.
- **Errors:** The traps in this group are INTEGERERROR and OVERFLOW. OVERFLOW represents arithmetic overflow, such as arithmetic results which do not fit in the result word. INTEGERERROR represents errors caused when data is erroneous, for example when a range checking instruction finds that data is out of range.
- **System operations:** This group consists of the LOADTRAP, STORETRAP and ILLEGALOPCODE traps. The ILLEGALOPCODE trap is signalled when an attempt is made to execute an illegal instruction. The LOADTRAP and STORETRAP traps allow a kernel to intercept attempts by a monitored process to change or examine trap handlers or trapped process information. It enables a program to signal to a kernel that it wishes to install a new trap handler.
- **Scheduler:** The scheduler trap group consists of the EXTERNALCHANNEL, INTERNALCHANNEL, TIMER, TIMESLICE, RUN, SIGNAL, PROCESSINTERRUPT and QUEUEEMPTY traps. The PROCESSINTERRUPT trap signals that the machine has performed a priority interrupt from low to high. The QUEUEEMPTY trap indicates that there is no further executable work to perform. The other traps in this group indicate that the hardware scheduler wants to schedule a process on a process queue, with the different traps enabling the different sources of this to be monitored.

The scheduler traps enable a software scheduler kernel to use the hardware scheduler to implement a multipriority software scheduler.

Note: Scheduler traps are different from other traps as they are caused by the microscheduler rather than by an executing process.

Trap groups encoding is shown in Table 58. These codes are used to identify trap groups to various instructions.

Table 58: Trap group codes

Trap group	Code
Breakpoint	0
CPU errors	1
System operations	2
Scheduler	3

In addition to the trap groups mentioned above, the CAUSEERROR flag in the STATUS register is used to signal when a trap condition has been activated by the **causeerror** instruction. It can be used to indicate when trap conditions have occurred due to the user setting them, rather than by the system.

8.7.2 Events that can cause traps

Table 59 summarizes the events that can cause traps and gives the encoding of bits in the trap STATUS and ENABLE registers.

Table 59: Trap causes and status/enable codes

Trap cause	Status/ enable bit number	Trap group	Comments
Breakpoint	0	0	When a process executes the breakpoint instruction (j0) then it traps to its trap handler.
Integer error	1	1	Integer error other than integer overflow, for example, explicitly checked or explicitly set error.
Overflow	2	1	Integer overflow or integer division by zero.
Illegal opcode	3	2	Attempt to execute an illegal instruction. This is signalled when opr is executed with an invalid operand.
Load trap	4	2	When the trap descriptor is read with the ldtraph instruction or when the trapped process status is read with the ldtrapped instruction.
Store trap	5	2	When the trap descriptor is written with the sttraph instruction or when the trapped process status is written with the sttrapped instruction.
Internal channel	6	3	Scheduler trap from internal channel.
External channel	7	3	Scheduler trap from external channel.
Timer	8	3	Scheduler trap from timer alarm.
Timeslice	9	3	Scheduler trap from timeslice.
Run	10	3	Scheduler trap from runp (run process) or startp (start process).

Table 59: Trap causes and status/enable codes

Trap cause	Status/ enable bit number	Trap group	Comments
Signal	11	3	Scheduler trap from signal .
Process interrupt	12	3	Start executing a process at a new priority level.
Queue empty	13	3	Caused by no process active at a priority level.
Cause error	15 (Status only)	Any, encoded 0 to 3	Signals that the causeerror instruction set the trap flag.

8.7.3 Trap handlers

For each trap handler there is a trap handler structure and a trapped process structure. Both the trap handler structure and the trapped process structure are in memory and can be accessed via instructions.

The trap handler structure specifies what should happen when a trap is taken, see [Table 60](#).

The trapped process structure saves some of the state of the process that was running when the trap was taken, see [Table 61](#).

In addition, for each priority there is an ENABLES register and a STATUS register. The ENABLES register contains flags to enable each cause of trap. The STATUS register contains flags to indicate which trap conditions have been detected. The ENABLES and STATUS register bit encoding is given in [Table 59 on page 87](#).

Table 60: Trap handler structure

Register	Comments	Location
IPTR	IPTR of trap handler process.	Base + 3
WPTR	WPTR of trap handler process. A null WPTR indicates that a trap handler has not been installed.	Base + 2
STATUS	Contains the STATUS register that the trap handler starts with.	Base + 1
ENABLES	A word which encodes the trap enable and global interrupt masks, which is ANDed with the existing masks to allow the trap handler to disable various events while it runs.	Base + 0

Table 61: Trapped process structure

Register	Comments	Location
IPTR	Points to the instruction after the one that caused the trap condition.	Base + 3
WPTR	WPTR of the process that was running when the trap was taken.	Base + 2
STATUS	The relevant trap bit is set, see Table 58 on page 87 for trap codes.	Base + 1
ENABLES	Interrupt enables.	Base + 0

A trap is taken at an interruptible point if a trap is set and the corresponding trap enable bit is set in the ENABLES register. If the trap is not enabled then nothing is done with the trap condition. If the trap is enabled then the corresponding bit is set in the STATUS register to indicate the trap condition has occurred.

When a process takes a trap the processor saves the existing IPTR, WPTR, STATUS and ENABLES in the trapped process structure. It then loads IPTR, WPTR and STATUS from the equivalent trap handler structure and ANDs the value in ENABLES with the value in the structure. This allows the user to disable various events while in the handler, in particular a trap handler must disable all the traps of its trap group to avoid the possibility of a handler trapping to itself.

The trap handler then executes. The values in the trapped process structure can be examined using the **ldtrapped** instruction. When the trap handler has completed its operation it returns to the trapped process via the **tret** (trap return) instruction. This reloads the values saved in the trapped process structure and clears the trap flag in STATUS.

*Note: When a trap handler is started, AREG, BREG and CREG are not saved. The trap handler must save the AREG, BREG, CREG registers using **stl** (store local).*

8.7.4 Trap instructions

Trap handlers and trapped processes can be set up and examined via the **ldtraph**, **sttraph**, **ldtrapped** and **sttrapped** instructions. Table 62 describes the instructions that may be used when dealing with traps.

Table 62: Instructions which may be used when dealing with traps

Instruction	Meaning	Use
ldtraph	Load trap handler	Load the trap handler from memory to the trap handler descriptor.
sttraph	Store trap handler	Store an existing trap handler descriptor to memory.
ldtrapped	Load trapped	Load replacement trapped process status from memory.
sttrapped	Store trapped	Store trapped process status to memory.
traphenb	Trap enable	Enable traps.
trapdis	Trap disable	Disable traps.
tret	Trap return	Used to return from a trap handler.
causeerror	Cause error	Program can simulate the occurrence of an error.

The first four instructions transfer data to or from the trap handler structures or trapped process structures from or to an area in memory. In these instructions, AREG contains the trap group code (see Table 58 on page 87) and BREG points to the 4-word area of memory used as the source or destination of the transfer. In addition, CREG contains the priority of the handler to be installed or examined in the case of **ldtraph** or **sttraph**. **ldtrapped** and **sttrapped** apply only to the current priority.

If the LOADTRAP trap is enabled then **ldtraph** and **ldtrapped** do not perform the transfer but set the LOADTRAP trap flag. If the STORETRAP trap is enabled then **sttraph** and **sttrapped** do not perform the transfer but set the STORETRAP trap flag.

The trap enable masks are encoded by an array of bits (see Table 59 on page 87) which are set to indicate which traps are enabled. This array of bits is stored in the lower half word of the ENABLES register. The ENABLES register consists of the following subregisters:

- TRAPENABLESREG[0] for high priority trap enables,
- TRAPENABLESREG[1] for low priority trap enables,
- GLOBALENABLESREG bits, used to control timeslicing and interruptability.

Traps are enabled or disabled by loading a mask into AREG with bits set to indicate which traps are to be affected and the priority to affect in BREG. Executing **trapenb** ORs the mask supplied in AREG with the trap enables mask in the ENABLES register for the priority in BREG. Executing **trapdis** negates the mask supplied in AREG and ANDs it with the trap enables mask in the ENABLES register for the priority in BREG. Both instructions return the previous value of the trap enables mask in AREG.

8.7.5 Restrictions on trap handlers

There are various restrictions that must be placed on trap handlers to ensure that they work correctly.

- Trap handlers must not deschedule or timeslice. Trap handlers alter the ENABLES masks, therefore they must not allow other processes to execute until they have completed.
- Trap handlers must have their ENABLE masks set to mask all traps in their trap group to avoid the possibility of a trap handler trapping to itself.
- Trap handlers must terminate via the **tret** (trap return) instruction. The only exception to this is that a scheduler kernel may use RESTART to return to a previously shadowed process.

Confidential

ST Confidential (iPanel)

9 Central processing unit (CPU) registers

9.1 Machine registers

Two sets of registers are explained here. The first set of registers are known as state registers. They fully define the state of the executing process. The second set of registers are used for multitasking, real-time clocks and trap and interrupt handling.

9.1.1 Process state registers

The state of an executing process at any instant is defined by the contents of the machine registers listed in [Table 63](#).

Table 63: Process state registers

Register	Full name or description process modes
STATUS	Status register
WPTR	Work space pointer, contains the address of the stack of the currently executing process
IPTR	Instruction pointer register, pointer to next instruction to be executed
AREG	Evaluation stack register A
BREG	Evaluation stack register B
CREG	Evaluation stack register C

9.2 Other machine registers

There are several other registers which are important, but which are not part of the process state. These are presented in [Table 64](#).

Table 64: Other machine registers

Register	Full name and description
PROCQUEUEFPTR[0]	High priority front pointer register Contains pointer to first process on the high priority scheduling list
PROCQUEUEFPTR[1]	Low priority front pointer register Contains pointer to first process on the low priority scheduling list
PROCQUEUEBPTR[0]	High priority back pointer register Contains pointer to last process on the high priority scheduling list
PROCQUEUEBPTR[1]	Low priority back pointer register Contains pointer to last process on the low priority scheduling list
CLOCKREG[0]	High priority clock register Contains current value of high priority clock
CLOCKREG[1]	Low priority clock register Contains current value of low priority clock
TPTRREG[0]	High priority timer list pointer register Contains pointer to the first process on the high priority timer list

Table 64: Other machine registers

Register	Full name and description
TPTRREG[1]	Low priority timer list pointer register Contains pointer to the first process on the low priority timer list
TNEXTREG[0]	High priority alarm register Contains the time of the first process on the high priority timer queue
TNEXTREG[1]	Low priority alarm register Contains the time of the first process on the low priority timer queue
ENABLES	Trap and global interrupt enables register

9.2.1 CLOCKENABLES

CLOCKENABLES is a pair of flags which enable the timers CLOCKREG to tick. Bit zero of CLOCKENABLES controls CLOCKREG[0] and bit 1 controls CLOCKREG[1]. In each case, the timer ticks if the CLOCKENABLES bit is set to 1. CLOCKENABLES can be set using the **clockenb** instruction and cleared using **clockdis**.

9.2.2 Shadow registers

When a high priority process interrupts a low priority process, the state of the currently executing process needs to be saved. For this purpose, two sets of process state registers are provided, one each for high and low priority. On interrupt, the processor switches to using the high priority registers, leaving the low priority registers to preserve the low priority state.

A high priority process may manipulate the low priority shadow registers with the instructions **ldshadow** and **stshadow**. In the definitions of these instructions, the process state registers have a subscript (for example AREG[low priority]) indicating the priority. If the process state registers are referred to without subscripts then the current priority is implied.

9.2.3 Error flags

The other machine flags referred to in the instruction definitions are listed in [Table 65](#).

Table 65: Error flags

Flag name	Description
ERRORFLAG	Untrapped arithmetic error flags
HALTONERRORFLAG	Halt the processor if the ERRORFLAG is set

ERRORFLAG is a pair of flags, one for each priority, set by the processor if an integer error or integer overflow error occurs and the corresponding trap is not enabled. The processor immediately halts if the HALTONERRORFLAG is also set, or continues otherwise. The ERRORFLAGS may also be set by the **seterr** instruction or tested and cleared by the **testerr** instruction. The **stoperr** instruction stops the current process if the ERRORFLAG is set. The low priority ERRORFLAG is copied to the high priority when the processor switches from low to high priority. The HALTONERRORFLAG may be set by the **sethalterr** instruction, cleared by **clrrhalterr** and tested by **testhalterr**.

9.3 Register details

STATUS

Status 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																
STATUS_VALID		INTERRUPTED_OPERATION_STATUS				Reserved				TIMESLICE_ENABLE		TRAP_GROUP_STATUS		SCHEDULER_TRAP_RETURN_PRIORITY_STATUS		CAUSEERROR_STATUS		Reserved		QUEUE_EMPTY_TRAP_STATUS		PROCESS_INTERRUPT_TRAP_STATUS		SIGNAL_TRAP_STATUS		RUN_TRAP_STATUS		TIMESLICE_TRAP_STATUS		TIMER_TRAP_STATUS		EXTERNAL_CHANNEL_TRAP_STATUS		INTERNAL_CHANNEL_TRAP_STATUS		STORE_TRAP_TRAP_STATUS		LOAD_TRAP_TRAP_STATUS		ILLEGAL_OPCODE_TRAP_STATUS		INTEGER_OVERFLOW_TRAP_STATUS		INTEGER_ERROR_TRAP_STATUS		BREAKPOINT_TRAP_STATUS	

Type: Read/write

Reset: Undefined

Description: The STATUS register contains status bits which describe the current state of the process and any errors which may have been detected.

[31] **STATUS_VALID**: Status valid bit

[30:26] **INTERRUPTED_OPERATION_STATUS**: Interrupted operation status bits:

00000: None	00001: move	00010: devmove	00011: move2dall
00100: move2dzero	00101: move2dnonzero	00110: in	00111: out
01000: tin	01001: tin restart	01010: taltwt	01011: taltwt restart
01100: dist	01101: dist restart	01110: enbc	01111: disc
10000: resetch			

[25:21] **Reserved**

[20] **TIMESLICE_ENABLE**: TIMESLICE enable bit

[19:18] **TRAP_GROUP_STATUS**: Trap group status bits

00: Breakpoint 01: Error 10: System 11: Scheduler

[17:16] **SCHEDULER_TRAP_RETURN_PRIORITY_STATUS**: Scheduler trap return priority status bits:

00: High priority 01: Low priority

[15] **CAUSEERROR_STATUS**: CAUSEERROR status bit

[14] **Reserved**

[13] **QUEUE_EMPTY_TRAP_STATUS**: QUEUEEMPTY trap status bit

[12] **PROCESS_INTERRUPT_TRAP_STATUS**: PROCESSINTERRUPT trap status bit

[11] **SIGNAL_TRAP_STATUS**: SIGNAL trap status bit

[10] **RUN_TRAP_STATUS**: RUN trap status bit

[9] **TIMESLICE_TRAP_STATUS**: TIMESLICE trap status bit

[8] **TIMER_TRAP_STATUS**: TIMER trap status bit

[7] **EXTERNAL_CHANNEL_TRAP_STATUS**: EXTERNALCHANNEL trap status bit

[6] **INTERNAL_CHANNEL_TRAP_STATUS**: INTERNALCHANNEL trap status bit

[5] **STORETRAP_TRAP_STATUS**: STORETRAP trap status bit

[4] **LOADTRAP_TRAP_STATUS**: LOADTRAP trap status bit

[3] **ILLEGAL_OPCODE_TRAP_STATUS**: ILLEGALOPCODE trap status bit

[2] **INTEGER_OVERFLOW_TRAP_STATUS**: OVERFLOW trap status bit

[1] **INTEGER_ERROR_TRAP_STATUS**: INTEGERERROR trap status bit

[0] **BREAKPOINT_TRAP_STATUS**: BREAKPOINT trap status bit

Enables register

31	30	29	28	27	26	25	24	Reserved															
LOW_PRIORITY_TIMER_ALARM_ENABLE																							
LOW_PRIORITY_EXTERNAL_EVENT_ENABLE																							
LOW_PRIORITY_TIMESLICE_ENABLE																							
LOW_PRIORITY_PROCESS_PREEMPTION_ENABLE																							
HIGH_PRIORITY_TIMER_ALARM_ENABLE																							
HIGH_PRIORITY_EXTERNAL_EVENT_ENABLE																							
HIGH_PRIORITY_TIMESLICE_ENABLE																							
HIGH_PRIORITY_PROCESS_PREEMPTION_ENABLE																							
DIAGNOSTICS_ENABLE																							
Reserved																							
QUEUE_EMPTY_TRAP_ENABLE																							
PROCESS_INTERRUPT_TRAP_ENABLE																							
SIGNAL_TRAP_ENABLE																							
RUN_TRAP_ENABLE																							
TIMESLICE_TRAP_ENABLE																							
TIMER_TRAP_ENABLE																							
EXTERNAL_CHANNEL_TRAP_ENABLE																							
INTERNAL_CHANNEL_TRAP_ENABLE																							
STORE_TRAP_TRAP_ENABLE																							
LOAD_TRAP_TRAP_ENABLE																							
ILLEGAL_OPCODE_TRAP_ENABLE																							
INTEGER_OVERFLOW_TRAP_ENABLE																							
INTEGER_ERROR_TRAP_ENABLE																							
BREAKPOINT_TRAP_ENABLE																							

Type: Read/write

Reset: Undefined

Description: The ENABLES register consists of two subregisters:

- TRAPENABLESREG[priority] (the TRAPENABLESREG for the current priority), bits (0 to 15), which is used to control the taking of traps. There are two TRAPENABLESREGs:
 - TRAPENABLESREG[0] for high priority trap enables,
 - TRAPENABLESREG[1] for low priority trap enables.
- GLOBALENABLESREG, bits 16 to 31, which is used to control timeslicing and interruptability. These bits are normally set to 1.

Note: While TRAPENABLESREG is dependent on the priority, GLOBALENABLESREG is a single register visible to both priorities.

Bits of either priority TRAPENABLESREG may be set using the **trapenb** instruction and cleared using **trapdis**. Bits of GLOBALENABLESREG may be set using the instruction **gintenb** and cleared using **gintdis**.

Note: While the bits of GLOBALENABLESREG appear in ENABLES as bits 16 to 31, when accessed directly using **gintenb** and **gintdis** then they appear as bits 0 to 15. Bit 15 of TRAPENABLESREG, the diagnostics enable bit, does not control any trap directly but is used to signal to a diagnostic control unit (DCU) whether breakpoints and watchpoints are enabled. This is designed to stop the DCU monitoring activity for breakpoints and watchpoints while the processor is executing a breakpoint trap handler. This bit should be set (that is diagnostics enabled) by the initialization code after booting and should only be cleared on entry to breakpoint trap handlers.

9.3.1 GLOBALENABLESREG

GLOBALENABLESREG is modified (as part of the ENABLES register) on entry to trap handlers. This means that by setting the ENABLES mask in the trap handler state to clear all these bits, a trap handler (and in particular a scheduler trap handler) can have full atomic access to the processor state.

The GLOBALENABLESREG register allows various aspects of the built in scheduling system to be globally enabled or disabled. This register can be modified using the **gintdis** (global interrupt disable) and **gintenb** (global interrupt enable) instructions. GLOBALENABLESREG bits 4 and 1 (HIGH_PRIORITY_TIMESLICE_ENABLE and LOW_PRIORITY_PROCESS_PREEMPTION_ENABLE) are allocated meaning in the ENABLES register, although they have no actual effect and are treated as reserved.

Each bit exists in a low and high priority version.

- **Process preemption enable**

This controls whether the built in microscheduler allows this priority to pre-empt a lower priority process. This only makes sense for the high priority process pre-emption enable bit and the low priority version has no meaning. If pre-emption is disabled and the machine is at low priority then any high priority processes that become ready are placed on the high priority queue and are only released for execution when this bit is enabled. This is unlike the behavior of INTDIS, INTENB which only allow pre-emption to be disabled locally until the next process deschedules.

- **Timeslice enable**

This controls whether timeslicing is enabled. Timeslicing is limited to low priority processes so the high priority version of this bit has no meaning. This enables timeslicing to be globally disabled for all low priority processes, unlike **settimeslice** which only affects behavior until the next process deschedules.

- **External event enable**

This controls whether the processor responds to external events (that is, the response from subsystems and external interrupts) at the appropriate priority. If disabled then any such events remain pending in the command bus interface or the subsystem until external events are re-enabled. External events which occur while this bit is disabled are not ignored but are deferred until the CPU is again prepared to accept them.

- **Timer alarm enable**

This controls whether the processor responds to processes on the appropriate timer queue becoming ready. If disabled then any process which becomes ready is left on the timer queue and is taken off when timer alarms are re-enabled.

Note:

Timer alarms should only be disabled for less than half a cycle of the clock register. If the alarms are disabled for more than half a clock cycle, processes may become ready during the disabled time but appear to be in the past when re-enabled. This causes the timer alarm to be missed and also may result in the violation of ordering rules in the timer queues. Since the timer alarm cannot be disabled more than 35 minutes at high priority and 38 hours at low priority, this should not be too severe a limitation.

Both **gintenb** and **gintdis** take a mask value in AREG (only the bottom eight bits are significant). **gintenb** ORs this value into GLOBALENABLESREG to ensure that the actions specified by the bits set in AREG are enabled. **gintdis** performs a bitwise negation on the bottom eight bits of AREG and then ANDs this into

GLOBALENABLESREG to ensure that the actions specified by the bits set in AREG are disabled. Both instructions return the value of GLOBALENABLESREG prior to their execution in AREG.

Thus timeslicing can be globally disabled by:

```
ldc 2; gintdis
```

and later re-enabled by:

```
ldc 2; gintenb
```

Also at any point the value of GLOBALENABLESREG can be examined by:

```
ldc 0; gintenb
```

If all the enables bits in GLOBALENABLESREG are cleared (that is all disabled) then the process executing can be guaranteed to have exclusive control over the processor as:

- no process can preempt it,
- it cannot be descheduled by timeslicing,
- no actions are taken in response to external events,
- no actions are taken for processes on the timer queue becoming ready.

This means that with respect to the rest of the system any actions performed by the process on the processor state is atomic. In particular, the process can safely manipulate objects such as scheduling queues without the danger other parts of the system attempting to modify them (for example via a reschedule request from a subsystem) while the state may be inconsistent. While all global enables are disabled the process must not deschedule for any reason. Thus it must not execute instructions that could deschedule. Although the process may have exclusive access to the state of the processor it cannot claim exclusive access over the memory system (as subsystems may be performing DMAs). However in most circumstances through the use of semaphores, protocols can be established to ensure that such a process has exclusive access to the structures which it needs to access atomically.

10 Instruction set

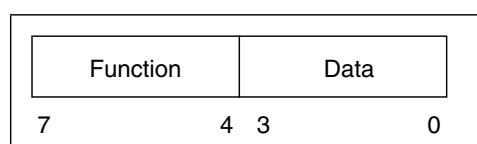
10.1 Overview

This chapter provides information on the ST20-C201 instruction set. It contains tables listing all the instructions and, where applicable, provides details of the number of processor cycles taken by an instruction.

The instruction set has been designed for simple and efficient compilation of high level languages. All instructions have the same format, designed to give a compact representation of the operations occurring most frequently in programs.

Each instruction consists of a single byte divided into two 4-bit parts. The four most significant bits (MSB) of the byte are a function code and the four least significant bits (LSB) are a data value, as shown in Figure 15.

Figure 15: Instruction format



For further information on the instruction set refer to the *ST20C2/C4 Core Instruction Set Reference Manual* (document number 7301761).

10.2 Instruction cycles

Timing information is available for some instructions. However, it should be noted that many instructions have ranges of timings which are data dependent.

Where included, timing information is based on the number of clock cycles assuming any memory accesses are to two cycle internal memory and no other subsystem is using memory. Actual time is dependent on the speed of external memory and memory bus availability.

The actual time can be increased as in the conditions below.

- The instruction requires a value on the register stack from the final memory read in the previous instruction. The current instruction stalls until the value becomes available.
- The first memory operation in the current instruction can be delayed while a preceding memory operation completes. Any two memory operations can be in progress at any time. Any further operation stalls until the first completes.
- Memory operations in current instructions can be delayed by access from either instruction fetches or subsystems to the memory interface.
- There can be a delay between instructions while the instruction fetch unit fetches and partially decodes the next instruction. This is the case whenever an instruction causes the instruction flow to jump.

Note: *The instruction timings given refer to standard behavior and may be different if, for example, traps are set by the instruction.*

10.3 Instruction characteristics

Table 68 on page 99 gives the basic function code of each of the primary instructions. Where the operand is less than 16, a single byte encodes the complete instruction. If the operand is greater than 15, one prefix instruction (**prefix**) is required for each additional four bits of the operand. If the operand is negative the first prefix instruction is **nfix**. Examples of **prefix** and **nfix** coding are given in Table 66 below.

Table 66: Prefix coding

Mnemonic	Function code	Memory code
ldc 0x3	0x4	0x43
ldc 0x35		
is coded as		
prefix 0x3	0x2	0x23
ldc 0x5	0x4	0x45
ldc 0x987		
is coded as		
prefix 0x9	0x2	0x29
prefix 0x8	0x2	0x28
ldc 0x7	0x4	0x47
ldc -31 (ldc 0xFFFF FFE1)		
is coded as		
nfix 0x1	0x6	0x61
ldc 0x1	0x4	0x41

Any instruction which is not in the instruction set tables is an invalid instruction and is flagged illegal, returning an error code to the trap handler if loaded and enabled.

10.4 Instruction set tables

In the instruction set tables, [Table 68: Primary functions on page 99](#) to [Table 89: Clock instructions on page 109](#), the following conventions apply:

m = memory speed,

d = peripheral space access time.

The Notes column indicates the features of an instruction as described in [Table 67](#).

Table 67: Instruction features

Identifier	Feature
E	Instruction can set an INTEGERERROR trap
L	Instruction can cause a LOADTRAP trap
S	Instruction can cause a STORETRAP trap
O	Instruction can cause an OVERFLOW trap
I	Interruptible instruction
A	Instruction can be aborted and later restarted
D	Instruction can deschedule
T	Instruction can timeslice

Table 68: Primary functions

Function code	Memory code	Mnemonic	Processor cycle	Name	Notes
0x000	0x0	j	4 if not breakpoint or timeslice 5 if breakpoint 5 + 2m if timeslice	Jump	D, T
0x001	0x1	ldlp	1	Load local pointer	
0x002	0x2	pfix	0 to 1	Prefix	
0x003	0x3	ldnl	1 + m	Load nonlocal	
0x004	0x4	ldc	0 or 1 (can be grouped)	Load constant	
0x005	0x5	ldnlp	m + 1	Load nonlocal pointer	
0x006	0x6	nfix	0 to 1	Negative prefix	
0x007	0x7	ldl	0: Grouped 1: Register cache, not grouped m + 1: General case	Load local	
0x008	0x8	adc	1	Add constant	O
0x009	0x9	call	4 + 4m	Call	

Table 68: Primary functions

Function code	Memory code	Mnemonic	Processor cycle	Name	Notes
0x00A	0xA	cj	1 if jump not taken 4 if jump is taken	Conditional jump	
0x00B	0xB	ajw	2	Adjust work space	
0x00C	0xC	eqc	1	Equals constant	
0x00D	0xD	stl	m	Store local	
0x00E	0xE	stnl	1 + m	Store nonlocal	
0x00F	0xF	opr	0	Operate	

Table 69: Processor initialization operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x22FA	testpranal	2	Test processor analyzing	
0x23FE	saveh	1 + 2m	Save high priority queue registers	
0x23FD	savel	1 + 2m	Save low priority queue registers	
0x21F8	sthf	1	Store high priority front pointer	
0x25F0	sthb	1	Store high priority back pointer	
0x21FC	stlf	1	Store low priority front pointer	
0x21F7	stlb	1	Store low priority back pointer	
0x25F4	sttimer	2	Store timer	
0x27FE	ldmemstartval	1	Load value of mem start address	

Table 70: Arithmetic/logical operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x24F6	and	1	And	
0x24FB	or	1	Or	
0x23F3	xor	1	Exclusive or	
0x23F2	not	1	Bitwise not	
0x24F1	shl	1	Shift left	
0x24F0	shr	1	Shift right	
0x00F5	add	1	Add	O
0x00FC	sub	1	Subtract	O
0x25F3	mul	4	Multiply	O
0x27F2	fmul	6 no overflow 7 overflow	Fractional multiply	O
0x22FC	div	5 if division by $\pm 2^n$ 37 otherwise	Divide	A, O

Table 70: Arithmetic/logical operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x21FF	rem	5 if divisor is $\pm 2^n$ 40 otherwise	Remainder	A, O
0x00F9	gt	1	Greater than	
0x25FF	gtu	1	Greater than unsigned	
0x00F4	diff	1	Difference	
0x25F2	sum	1	Sum	
0x00F8	prod	4	Product	
0x26F8	satadd	2	Saturating add	
0x26F9	satsub	2	Saturating subtract	
0x26FA	satmul	5	Saturating multiply	

Table 71: Long arithmetic operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x21F6	ladd	2	Long add	O
0x23F8	lsub	2	Long subtract	O
0x23F7	lsum	2	Long sum	
0x24FF	ldiff	2	Long diff	
0x23F1	lmul	5 or 6	Long multiply	
0x21FA	ldiv	3 overflow 38 no overflow	Long divide	A, O
0x23F6	lshl	2	Long shift left	
0x23F5	lshr	2	Long shift right	
0x21F9	norm	2 if result 64 4 if result 32 to 64 5 if result 0 to 31	Normalize	
0x26F4	slmul	5	Signed long multiply	
0x26F5	sulmul	5	Signed times unsigned long multiply	

Confidential

Table 72: General operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x00F0	rev	1	Reverse	
0x23FA	xword	4	Extend to word	
0x25F6	cword	2 if no error 3 if error	Check word	E
0x21FD	xdbl	2	Extend to double	
0x24FC	csngl	2 if no error 3 if error	Check single	E
0x24F2	mint	1	Minimum integer	
0x25FA	dup	1	Duplicate top of stack	
0x27F9	pop	1	Pop processor stack	
0x68FD	reboot	Not applicable	Reboot	

Table 73: Indexing/array operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x00F2	bsub	1	Byte subscript	
0x00FA	wsub	1	Word subscript	
0x28F1	wsubdb	1	Form double word subscript	
0x23F4	bcnt	1	Byte count	
0x23FF	wcnt	2	Word count	
0x00F1	lb	m	Load byte	
0x23FB	sb	1 + m	Store byte	
0x24FA	move	2 + (r + w) m	Move message	I

Table 74: Timer handling operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x22F2	ldtimer	1	Load timer	
0x22FB	tin	3 if time is in past 7 + N (2 + 2m) + 6m if time is in future Where N is depth in timer queue of process insertion	Timer input	D, I
0x24FE	talt	1 + 2m	Timer ALT start	

Confidential

Table 75: Input and output operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x00F7	in	6 + 3m internal channel not ready 8 + (3 + r + w) m internal channel ready and scheduling queue empty 7 + (4 + r + w) m internal channel ready and queue not empty 8 + m external channel	Input message	D, I
0x00FB	out	6 + 3m internal channel not ready 6 + 5m internal channel to process in enabling ALT state 8 + 5m internal channel to waiting ALT sched queue empty 7 + 6m internal channel to waiting ALT queue not empty 6 + 5m internal channel to ready ALT 9 + (3 + r + w) m internal channel ready and scheduling queue empty 8 + (4 + r + w) m internal channel ready and queue not empty 8 + m external channel	Output message	D, I
0x00FF	outword	As for out but with additional 1 + m	Output word	D, I
0x00FE	outbyte	As for outword	Output byte	D, I
0x24F3	alt	1 + m	ALT start	
0x24F4	altwt	2 + m if ready 6 + 2m if not ready	ALT wait	D
0x24F5	altend	5 + m	ALT end	
0x24F9	enbs	1 if guard is false 1 + m if guard is true	Enable skip	
0x23F0	diss	2 if guard is false 2 + 2m if guard is true	Disable skip	
0x21F2	resetch	2 + 2m internal channel 4 + maximum (c, 2 + m) + 2m external channel	Reset channel	A
0x24F8	enbc	1 if guard is false 2 + 2m if guard true, internal channel not ready 3 + m if guard true, internal channel already enabled 3 + 2m if guard true, internal channel ready for output 3 + 2m + maximum (c, m) if guard true, external channel	Enable channel	I

Confidential

Table 75: Input and output operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x22FF	disc	2 if guard is false 4 + 2m if guard true and internal channel not ready 6 + m if guard true and internal channel already disabled 6 + 3m if guard true and internal channel is first one ready 7 + 2m if guard true and internal channel ready but not first 10 + c + m if guard true and external channel not ready 10 + c + 3m if guard true and external channel is first one ready 11 + c + 2m if guard true and external channel ready but not first	Disable channel	I

Table 76: Control operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x22F0	ret	5 + m	Return	
0x21FB	ldpi	1	Load pointer to instruction	
0x23FC	gajw	1	General adjust work space	
0x00F6	gcall	5	General call	
0x22F1	lend	3 + 3m if last iteration 5 + 4m if not last iteration	Loop end	D, T

Table 77: Scheduling operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x00FD	startp	4 + 3m if trapped 4 + m not trapped, scheduling queue empty 3 + 2m not trapped, scheduling queue not empty	Start process	
0x00F3	endp	4 + 2m not last process 6 + 2m last process to end	End process	D
0x23F9	runp	5 + 2m if trapped 4 if not trapped and queue empty 3 + m if not trapped and queue not empty	Run process	
0x21F5	stopp	3 + m	Stop process	D
0x21FE	ldpri	2	Load current priority	

Table 78: Error handling operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x21F3	csub0	1 if no error 2 if error	Check subscript from 0	E
0x24FD	ccnt1	2 if no error 3 if error	Check count from 1	E
0x22F9	testerr	2	Test error false and clear	
0x21F0	seterr	2	Set error	
0x25F5	stoperr	4+m if error flag set 2 if not	Stop on error (no error)	D
0x25F7	clrhalt	1	Clear halt on error	
0x25F8	sethalt	1	Set halt on error	
0x25F9	testhalt	2	Test halt on error	

Table 79: 2-D block move operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x25FB	move2dinit	4	Initialize data for 2-D block move	
0x25FC	move2dall	$7 + 4 \times \text{rows} + (\text{r} + \text{w}) \text{ m}$	2-D block copy	I
0x25FD	move2dnonzero	$7 + 4 \times \text{rows} + (\text{r} + \text{w}) \text{ m}$	2-D block copy nonzero bytes	I
0x25FE	move2dzero	$7 + 4 \times \text{rows} + (\text{r} + \text{w}) \text{ m}$	2-D block copy zero bytes	I

Table 80: CRC and bit operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x27F4	crcword	36	Calculate crc on word	A
0x27F5	crcbyte	12	Calculate crc on byte	A
0x27F6	bitcnt	3	Count bits set in word	
0x27F7	bitrevword	1	Reverse bits in word	
0x27F8	bitrevnbits	2	Reverse bottom n bits in word	

Table 81: Floating point support operation codes

Memory code	Mnemonic	Processor cycle	Name	Notes
0x27F3	cferr	2 if no error 3 if error	Check floating point error	E
0x29FC	fpsterr	1	Load value true (FPU not present)	
0x26F3	unpacksn	10	Unpack single length floating point number	A
0x26FD	roundsn	7	Round single length floating point number	
0x26FC	postnormsn	9	Postnormalize correction of single length floating point number	A
0x27F1	ldinf	1	Load single length infinity	

Table 82: Range checking and conversion instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x2CF7	cir	2 if no error 3 if error	Check in range	E
0x2CFC	ciru	2 if no error 3 if error	Check in range unsigned	E
0x2BFA	cb	2 if no error 3 if error	Check byte	E
0x2BFB	cbu	2 if no error 3 if error	Check byte unsigned	E
0x2FFA	cs	2 if no error 3 if error	Check 16	E
0x2FFB	csu	2 if no error 3 if error	Check 16 unsigned	E
0x2FF8	xsword	3	Sign extend 16 to word	
0x2BF8	xbword	3	Sign extend byte to word	

Table 83: Indexing/array instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x2CF1	ssub	1	16 subscript	
0x2CFA	ls	m	Load 16	
0x2CF8	ss	1 + m	Store 16	
0x2BF9	lby	m	Load byte and sign extend	
0x2FF9	lsx	m	Load 16 and sign extend	

Table 84: Device access instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x2FF0	devlb	2 + m if to memory space 3 + d if to device space	Device load byte	A
0x2FF2	devls	2 + m if to memory space 3 + d if to device space	Device load 16	A
0x2FF4	devlw	2 + m if to memory space 3 + d if to device space	Device load word	A
0x62F4	devmove	2 + m if to memory space 3 + d if to device space	Device move	I
0x2FF1	devsb	2 + m if to memory space 3 + d if to device space	Device store byte	A
0x2FF3	devss	2 + m if to memory space 3 + d if to device space	Device store 16	A
0x2FF5	devsw	2 + m if to memory space 3 + d if to device space	Device store word	A

Table 85: Semaphore instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x60F5	wait	2 + 2m semaphore available 6 + 5m semaphore not available but queue empty 6 + 6m semaphore not available and queue not empty	Wait	D
0x60F4	signal	3 + 3m count not 0 5 + 3m count 0, size semaphore queue 1, empty sched queue 4 + 4m count 0, size semaphore queue 1, queue not empty 6 + 4m count 0, size semaphore queue > 1, empty sched queue 5 + 5m count 0, size semaphore queue > 1, queue not empty	Signal	

Confidential

Table 86: Scheduling support instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x60F0	swapqueue	4	Swap scheduler queue	
0x60F1	swaptimer	3 if swapping in empty queue 4 + m if swapping in nonempty queue	Swap timer queue	
0x60F2	insertqueue	3 queue was empty 4 + m queue not empty	Insert at front of scheduler queue	
0x60F3	timeslice	3 + 2m if trapped 5 + m otherwise	Timeslice	D
0x60FC	ldshadow	7 + m = if enables load selected 7 + 3 + 3m if status, WPTR, IPTR load selected 7 + 1 + 3m if stack load selected 7 + 5 + 4m if other registers load selected	Load shadow registers	A
0x60FD	stshadow	7 + m if enables store selected 7 + 3m if status, WPTR, IPTR store selected 7 + 3m if stack store selected 7 + 4m if other register store selected	Store shadow registers	A
0x62FE	restart	11 + 10m	Restart	
0x62FF	causeerror	6 if scheduler trap not set 7 + m if scheduler trap set	Cause error	
0x61FF	iret	4 >+ m if returning to idle machine priority 7 + 6m if returning to nonidle machine priority	Interrupt return	
0x2BF0	settimeslice	2	Set timeslicing status	
0x2CF4	intdis	2	Interrupt disable	
0x2CF5	intenb	2	Interrupt enable	
0x2CFD	gintdis	5	Global interrupt disable	
0x2CFE	gintenb	5	Global interrupt enable	

Confidential

Table 87: Trap handler instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x26FE	ldtraph	3 + 8m if not trapped 4 if trapped	Load trap handler	L
0x2CF6	ldtrapped	3 + 8m if not trapped 4 if trapped	Load trapped process status	L
0x2CFB	sttrapped	3 + 8m if not trapped 4 if trapped	Store trapped process status	S
0x26FF	sttraph	3 + 8m if not trapped 4 if trapped	Store trap handler	S
0x60F7	trapenb	4	Trap enable	
0x60F6	trapdis	4	Trap disable	
0x60FB	tret	4 + 5m return from program trap 6 + 5m return from scheduler trap	Trap return	

Table 88: Processor initialization and no operation instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x68FC	ldprodid	1	Load product identity	
0x63F0	nop	1	No operation	

Table 89: Clock instructions

Memory code	Mnemonic	Processor cycle	Name	Notes
0x64FF	clockenb	2	Clock enable	
0x64FE	clockdis	2	Clock disable	
0x64FD	ldclock	2	Load clock	
0x64FC	stclock	3	Store clock	

11 Interrupt system

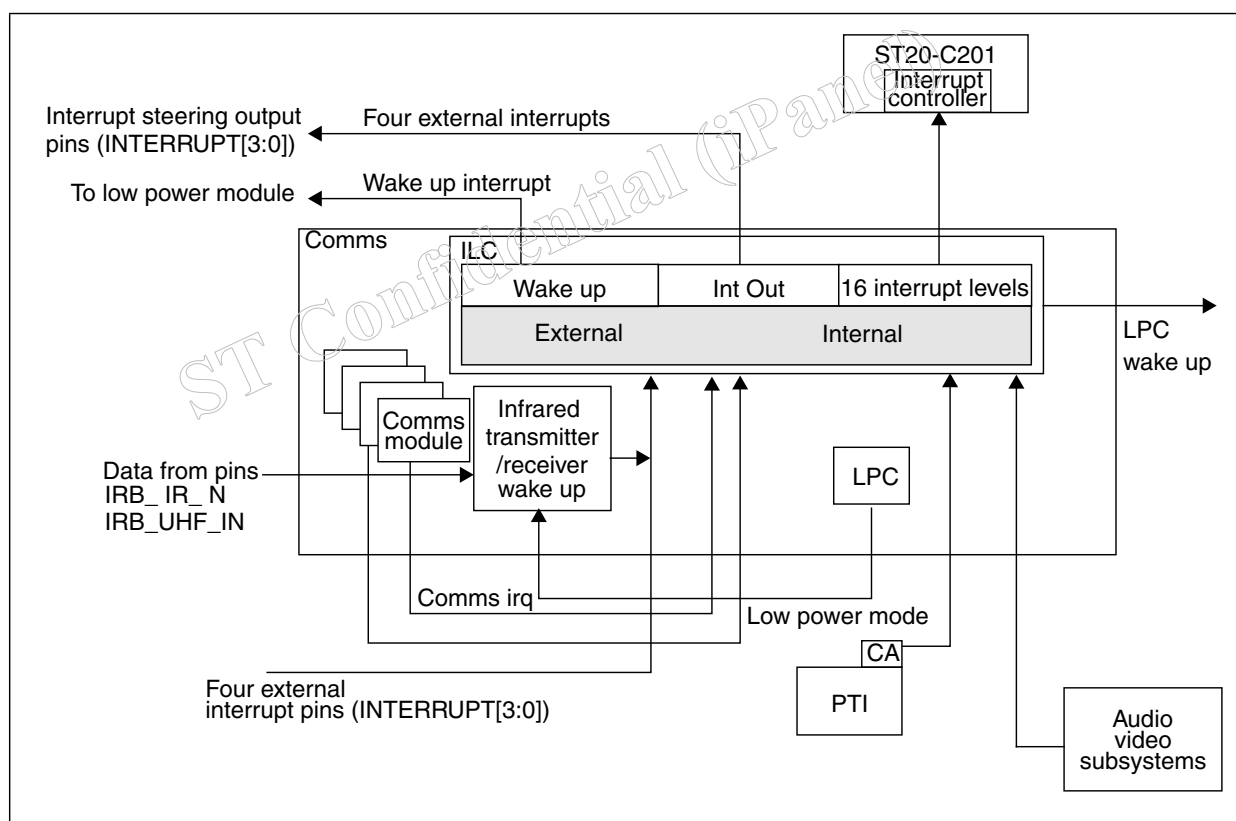
11.1 Overview

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 11.2: Interrupt controller on page 111](#)) and an on-chip interrupt level controller (see [Section 11.3: Interrupt level controller on page 114](#)). The interrupt level controller multiplexes the 38 incoming interrupt sources on to the 16 programmable interrupt level inputs of the interrupt controller. Multiplexing is controlled by software. This is illustrated in [Figure 16](#).

Figure 16: Interrupt network



Pins **INTERRUPT[3:0]** can be connected to both the interrupt steering outputs or the four external interrupt inputs. The direction is controlled by the interconnect configuration control register B [29:26] **EXT_INTERRUPT_ENABLES [3:0]**

11.2 Interrupt controller

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

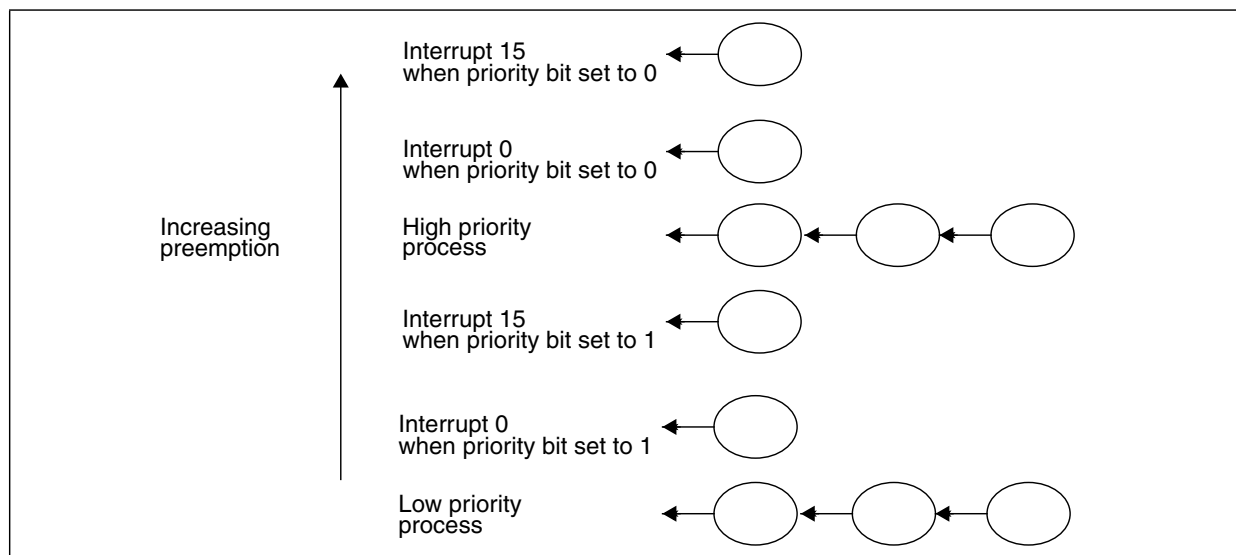
All interrupts have a higher priority than the low priority process queue. Each interrupt can be programmed to be at a lower or higher priority than the high priority process queue by writing to the priority bit in the INC_HANDLERWPTR registers. Interrupts which are specified as higher priority must 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 a 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 that these interrupt sources are multiplexed on to (by the interrupt level controller) 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 of the INC_MASK, INC_PENDING and INC_TRIGGERMODE registers is supported via the operating system run time library functions of STLite/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 17: Interrupt priority



11.2.1 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.

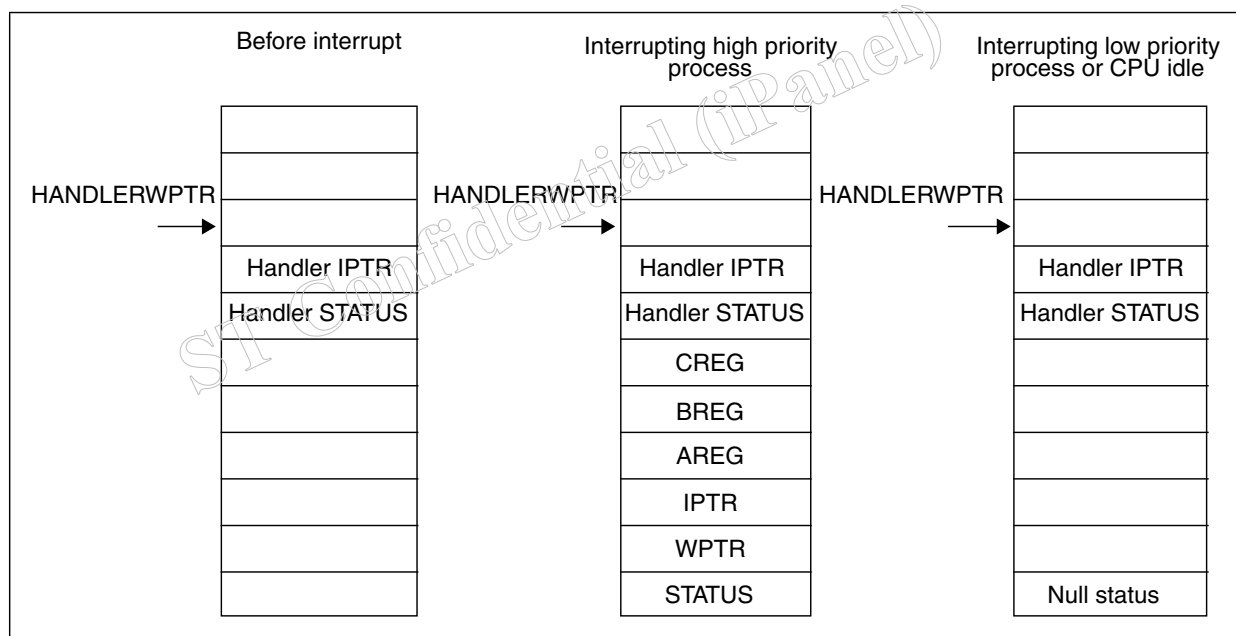
The operating system run time library (STLite/OS20) supports the setting and programming of the vector table.

11.2.2 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 18](#). Each interrupt level has its own work space.

Figure 18: 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 11.2.4: Preemption and interrupt priority on page 113](#)).

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.

11.2.3 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.

11.2.4 Preemption 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.

11.2.5 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 2-D block move instructions.
A workaround is to store the state of the shadow registers used by the 2-D block move instruction into memory, with the **stshadow** instruction. The state must be restored after the 2-D block move instruction by using the **ldshadow** instruction.
- Interrupt handlers must not cause program traps.
(However, they may be trapped by a scheduler trap)

11.3 Interrupt level controller

The interrupt level controller multiplexes 34 internal and 5 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 90 on page 115](#) assigns each of the interrupt sources to a number *n* from 0 to 34. Software assigns a signal *n* to 1 of the 16 interrupt levels by writing the priority of the required input in the register ILC_PRIORITY. Each of the 35 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 STi5516 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.

11.4 Interrupt assignments

All interrupts are active high. Interrupts from the internal peripherals and external pins are assigned as in [Table 90](#).

Table 90: STi5516 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	PIO4	Compare function
5	SSC0	SSC0TIR, SSC0RIR, SSC0EIR I ² C MASTER
6	SSC1	SSC1TIR, SSC1RIR, SSC1EIR I ² C MASTER
7	ASC3	ASC3TIR, ASC3TBIR, ASC3RIR, ASC3EIR
8	ASC2	ASC2TIR, ASC2TBIR, ASC2RIR, ASC2EIR
9	ASC1	ASC1TIR, ASC1TBIR, ASC1RIR, ASC1EIR
10	ASC0	ASC0TIR, ASC0TBIR, ASC0RIR, ASC0EIR
11	PWM and capture	PWM_OUT[3:0], PWM_CAPTURE0, PWM_CAPTURE2, PWM_COMPARE0 and PWM_COMPARE2
12	P1284	
13	TTXT	Teletext DMA interrupt
14	PTI	
15	Reserved	
16	Modem DMA	MAFE interface interrupt
17	PIO5	Compare function
18	IR blaster	Tx, Rx interrupts
19	Reserved	
20	Reserved	
21	Video Decoder	
22	Audio Decoder	
23 to 34	Reserved	
35	ASC4	ASC4TIR, ASC4TBIR, ASC4RIR, ASC4EIR
36 to 47	Reserved	

Table 90: STi5516 interrupt assignments

INT n	Peripheral	Description of the function
48 (EXT0)	EXTINTIN0	From external device(s)
49 (EXT1)	EXTINTIN1	
50 (EXT2)	EXTINTIN2	
51 (EXT3)	EXTINTIN3	
52 (EXT4)	IR wake up	From pins via pulse stretcher
Interrupt outputs below this line (inputs above)		
48 (EXT0)	EXTINTOUT0	To external device(s)
49 (EXT1)	EXTINTOUT1	
50 (EXT2)	EXTINTOUT2	
51 (EXT3)	EXTINTOUT3	

Confidential

ST Confidential (iPanel)

12 Interrupt system registers

Addresses are provided as the *BaseAddress* + offset.

The *BaseAddresses* are:

IntControllerBaseAddress: 0x3000 0000,

ILCBaseAddress: 0x2011 1000.

A register summary is given in [Table 42: Interrupt system registers on page 67](#).

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	CLR

Address: *IntControllerBaseAddress* + 0x108

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	CLR

Address: *IntControllerBaseAddress* + 0x0C8

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

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	CLR

Address: *IntControllerBaseAddress* + 0x088

Type: Write only

Description: When set, bits 0 to 15 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.

Confidential

INC_EXEC**Interrupts executing**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEX15	INTEX14	INTEX13	INTEX12	INTEX11	INTEX10	INTEX9	INTEX8	INTEX7	INTEX6	INTEX5	INTEX4	INTEX3	INTEX2	INTEX1	INTEX0

Address: *IntControllerBaseAddress* + 0x100

Type: Read/write

Reset: 0

Description: This register records whether interrupt levels 0 to 15 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.

Confidential

ST Confidential (iPanel)

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
HANDLERWPTR0[31:2]																Res		P													
HANDLERWPTR1[31:2]																Res		P													
HANDLERWPTR2[31:2]																Res		P													
HANDLERWPTR3[31:2]																Res		P													
HANDLERWPTR4[31:2]																Res		P													
HANDLERWPTR5[31:2]																Res		P													
HANDLERWPTR6[31:2]																Res		P													
HANDLERWPTR7[31:2]																Res		P													
HANDLERWPTR8[31:2]																Res		P													
HANDLERWPTR9[31:2]																Res		P													
HANDLERWPTR10[31:2]																Res		P													
HANDLERWPTR11[31:2]																Res		P													
HANDLERWPTR12[31:2]																Res		P													
HANDLERWPTR13[31:2]																Res		P													
HANDLERWPTR14[31:2]																Res		P													
HANDLERWPTR15[31:2]																Res		P													

Address: *IntControllerBaseAddress* + 0x000 (INC_HANDLERWPTR0) to 0x03C (INC_HANDLERWPTR15)

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

Address: *IntControllerBaseAddress* + 0x0C0

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.

[31:17] **Reserved**

[16] **GLOEN**

1: Interrupt setting determined by the corresponding INTEN bit
0: All interrupts disabled

[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.

Confidential

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* + 0x080

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	SET	SET	SET	SET

Address: *IntControllerBaseAddress* + 0x104

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	SET

Address: *IntControllerBaseAddress* + 0x0C4

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	SET

Address: *IntControllerBaseAddress* + 0x084

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
Reserved																TRIGGER0															
Reserved																TRIGGER1															
Reserved																TRIGGER2															
Reserved																TRIGGER3															
Reserved																TRIGGER4															
Reserved																TRIGGER5															
Reserved																TRIGGER6															
Reserved																TRIGGER7															
Reserved																TRIGGER8															
Reserved																TRIGGER9															
Reserved																TRIGGER10															
Reserved																TRIGGER11															
Reserved																TRIGGER12															
Reserved																TRIGGER13															
Reserved																TRIGGER14															
Reserved																TRIGGER15															

Address: *IntControllerBaseAddress* + 0x040 (INC_TRIGGERMODE0) to 0x07C (INC_TRIGGERMODE15)

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.

12.1 Interrupt level controller registers

ILC_INPUT_INTERRUPT Input 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						EXT4	EXT3	EXT2	EXT1	EXT0											Reserved						INT35	INT34	INT33	INT32
NT31	NT30	NT29	NT28	NT27		NT25	NT24	NT23	NT22	NT21	NT20	NT19	NT18	NT17	NT16	NT15	NT14	NT13	NT12	NT11	NT10	INT9	INT8	INT7	INT6		INT5	INT4	INT3	INT2	INT1	INT0			

Address: $ILCBaseAddress + 0x080$ and $0x084$

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.

Note: Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.

ILC_STATUS Status 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						EXT4	EXT3	EXT2	EXT1	EXT0											Reserved						INT35	INT34	INT33	INT32
NT31	NT30	NT29	NT28	NT27		NT26	NT25	NT24	NT23	NT22	NT21	NT20	NT19	NT18	NT17	NT16	NT15	NT14	NT13	NT12	NT11	NT10	INT9	INT8	INT7		INT6	INT5	INT4	INT3	INT2	INT1	INT0		

Address: $ILCBaseAddress + 0x200$ and $0x204$

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 input interrupt registers from $0x200$ to $0x204$. The contents of this register are set to logic 0 on reset.

Note: Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.

ILC_CLEAR_STATUS Clear status locations

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved						EXT4	EXT3	EXT2	EXT1	EXT0						Reserved							INT35	INT34	INT33	INT32
INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: $ILCBaseAddress + 0x280$ and $0x284$

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 $0x284$.

Note: Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.

ILC_ENABLE Enable 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						EXT4	EXT3	EXT2	EXT1	EXT0						Reserved							INT35	INT34	INT33	INT32
INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Address: $ILCBaseAddress + 0x400$ and $0x404$

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 $0x404$.

Note: Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.

ILC_CLEAR_ENABLE Clear enable locations

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
INT31 INT30 INT29 INT28 INT27 INT26	Reserved	INT25 INT24 INT23 INT22 INT21	Reserved	EXT4 EXT3 EXT2 EXT1 EXT0	Reserved	INT15 INT14 INT13 INT12 INT11 INT10	Reserved	INT9 INT8 INT7 INT6 INT5 INT4	INT3 INT2 INT1 INT0	Reserved	INT35 INT34 INT33 INT32	Reserved	INT31 INT30 INT29 INT28 INT27 INT26	Reserved	EXT4 EXT3 EXT2 EXT1 EXT0	Reserved	INT15 INT14 INT13 INT12 INT11 INT10	Reserved	INT9 INT8 INT7 INT6 INT5 INT4	INT3 INT2 INT1 INT0	Reserved	INT35 INT34 INT33 INT32	Reserved	INT31 INT30 INT29 INT28 INT27 INT26	Reserved	EXT4 EXT3 EXT2 EXT1 EXT0	Reserved	INT15 INT14 INT13 INT12 INT11 INT10	Reserved	INT9 INT8 INT7 INT6 INT5 INT4	INT3 INT2 INT1 INT0	Reserved	INT35 INT34 INT33 INT32

Address: $ILCBaseAddress + 0x480$ and $0x484$

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.

Note: Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.

ILC_SET_ENABLE Set enable locations

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NT31 NT30 NT29 NT28 NT27 NT26 NT25	Reserved	NT24 NT23 NT22 NT21	Reserved	EXT4 EXT3 EXT2 EXT1 EXT0	NT15 NT14 NT13 NT12 NT11 NT10	Reserved	INT9 INT8 INT7 INT6 INT5 INT4	INT35 INT34 INT33 INT32	INT3 INT2 INT1 INT0																						

Address: $ILCBaseAddress + 0x500$ and $0x504$

Type: Write only

Reset: Undefined

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

There are two locations at $0x500$ and $0x504$ corresponding to respective enable registers.

Note: Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.

ILC_WAKEUP_ENABLE Wake up enable 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											EXT4	EXT3	EXT2	EXT1	EXT0	Reserved															

Address: *ILCBaseAddress* + 0x604

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: *Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.*
To wake up a device from low-power mode each source can be woken up independently, see register CONFIG_CONTROL_D, bits 21 and 22 in [Section 19.3: Configuration registers](#).

ILC_WAKEUP_ACTIVE_LEVEL Wake up level 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											EXT4	EXT3	EXT2	EXT 1	EXT 0												Reserved					

Address: *ILCBaseAddress* + 0x684

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 0x684 corresponds to respective interrupt numbers.

Note: *Interrupt EXT4 is not a valid output, see [Section 11.4: Interrupt assignments on page 115](#) for a list of interrupt numbers.*

Confidential

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: $ILCBaseAddress + 0x800 + (n \times 0x008)$

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 $0x800 + (n \times 0x008)$.

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: $ILCBaseAddress + 0x800 + (n \times 0x008) + 0x004$

Type: Read/write

Reset: 0

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

- Program the trigger modes to appropriate trigger levels by writing the values as shown below. Use the syntax 0xnnnn nnnn for the address.
- Include the values outside the address range (and say what happens when they are accessed).

[31:3] **Reserved**

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

0x00: No trigger mode

0x02: Low level trigger mode

0x04: Falling edge trigger mode

0x06: No trigger mode

0x01: High level trigger mode

0x03: Rising edge trigger mode

0x05: Any edge trigger mode

0x07: No trigger mode

The mode register is set to 0x00 on reset. The address of mode register corresponding to interrupt number n is at $0x800 + (n \times 0x008) + 0x004$.

13 Memory map

The STi5516 memory space is divided into four main regions:

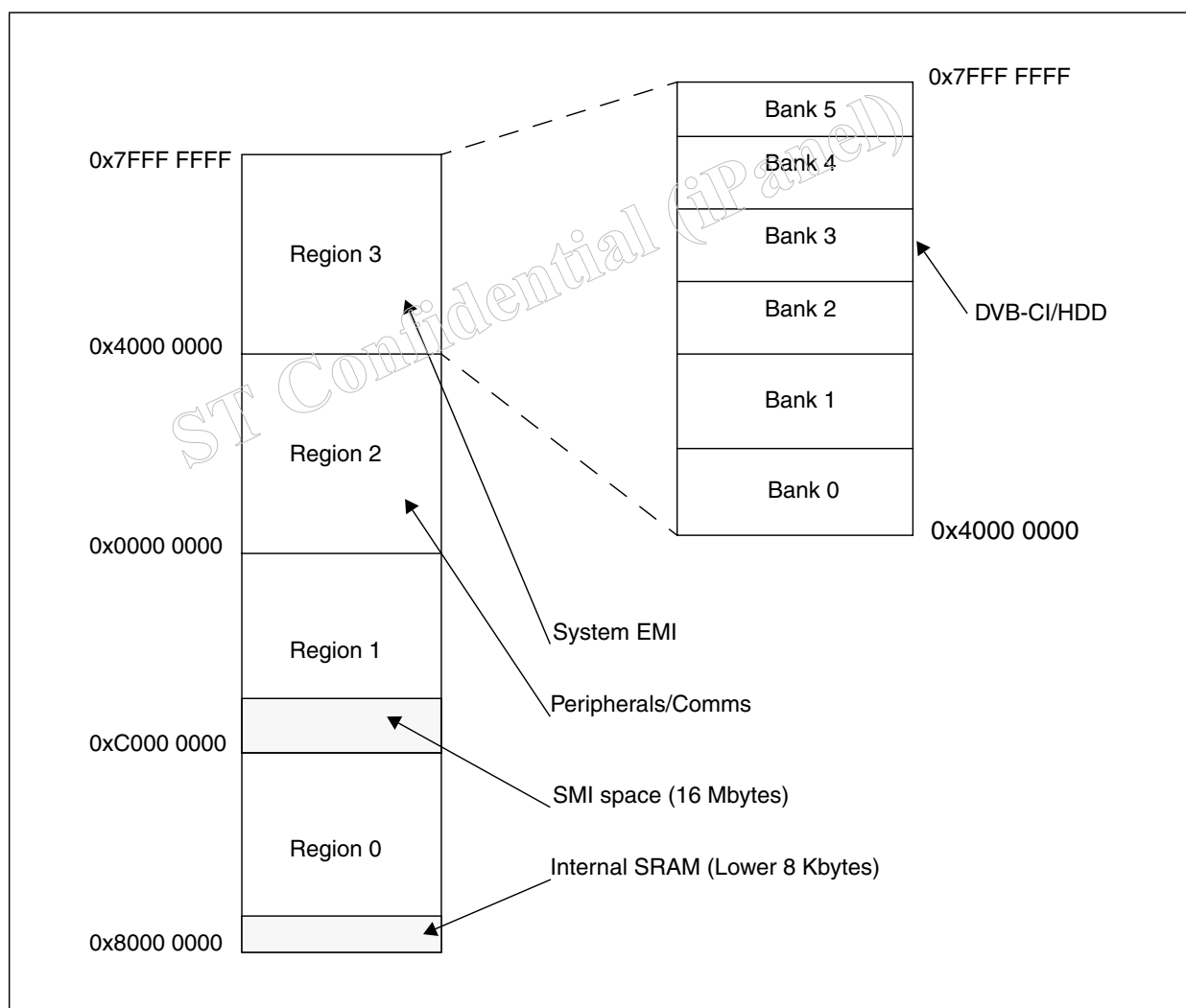
- region 0: addresses which map on to the on-chip SRAM,
- region 1: addresses which map on to the off-chip SDRAM connected to the local SMI,
- region 2: addresses which map on to the on-chip peripheral configuration registers,
- region 3: addresses which map on to the external memory interface (EMI).

Region 3 is decoded via the EMI buffer into six banks, the boundaries between these banks can be programmed using six configuration registers. Address bits 29 down to 22 are compared with the programmed values to determine the bank of the current access.

The largest programmable size of a bank is 256 Mbytes. The minimum is 4 Mbytes.

Figure 19 illustrates the top level address map breakdown.

Figure 19: Memory map



Confidential

Table 91: Typical EMI configuration

Bank	From	To	Size (Mbytes)	Type
5	0x7F80 0000	0x7FFF FFFF	8	ROM
4	0x7F00 0000	0x7F7F FFFF	8	SRAM
3	0x7000 0000	0x7EFF FFFF	240	Peripheral
2	0x6000 0000	0x6FFF FFFF	256	Peripheral
1	0x5000 0000	0x5FFF FFFF	256	SDRAM
0	0x4000 0000	0x4FFF FFFF	256	SDRAM

Table 92: Region 2 (comms/peripherals) breakdown

Address range		Size	Region type	Subregion type ^a
From	To			
0x3000 6000	0x3FFF FFFF	256 Mbytes	Peripherals	Reserved core expansion
0x3000 5000	0x3000 5FFF	4 Kbytes		DCache configuration
0x3000 4000	0x3000 4FFF	4 Kbytes		ICache configuration
0x3000 3000	0x3000 3FFF	4 Kbytes		DCU
0x3000 0000	0x3000 2FFF	12 Kbytes		Interrupt controller (INTC) includes future expansion
0x2060 0000	0x206F FFFF	250 Mbytes		Reserved
0x2050 0000	0x205F FFFF	1 Mbyte		Audio interface
0x2040 0000	0x204F FFFF	1 Mbyte		TSMUX
0x2030 0000	0x203F FFFF	1 Mbyte		Link layer interface
0x2020 0000	0x202F FFFF	1 Mbyte		System EMI configuration
0x2010 0000	0x201F FFFF	1 Mbyte	Comms	Comms base address and range
0x2010 0000	0x2010 0FFF	4 Kbytes		Low power module (LPM)
0x2010 3000	0x2010 6FFF	16 Kbytes		Asynchronous serial controller (0 to 3)
0x2010 7000	0x2010 8FFF	8 Kbytes		Smartcard clock generator (0 to 1)
0x2010 9000	0x2010 AFFF	8 Kbytes		Synchronous serial controller (0 to 1)
0x2010 B000	0x2010 BFFF	4 Kbytes		PWM
0x2010 C000	0x2011 0FFF	20 Kbytes		PIO (0 to 4)
0x2011 1000	0x2011 1FFF	4 Kbytes		Interrupt level controller (ILC)
0x2011 2000	0x2011 2FFF	4 Kbytes		PIO5
0x2011 3000	0x2011 3FFF	4 Kbytes		MAFE
0x2011 4000	0x2011 4FFF	4 Kbytes		Asynchronous serial controller 4

Confidential

Table 92: Region 2 (comms/peripherals) breakdown

Address range		Size	Region type	Subregion type ^a
From	To			
0x2011 5000	0x2011 5FFF	4 Kbytes	Comms	Infrared blaster
0x2011 6000	0x2011 6FFF	4 Kbytes		Reserved
0x2012 4000	0x2012 4FFF	4 Kbytes		Teletext
0x2012 5000	0x2012 5FFF	4 Kbytes		1284 interface
0x2006 0000	0x200F FFFF	64 Kbytes	Peripherals	Reserved
0x2005 0000	0x2005 FFFF	64 Kbytes		
0x2004 0000	0x2004 FFFF	64 Kbytes		
0x2003 0000	0x2003 FFFF	64 Kbytes		
0x2002 0000	0x2002 FFFF	64 Kbytes		PTI A
0x2001 C000	0x2001 FFFF	16 Kbytes		Reserved
0x2001 B000	0x2001 BFFF	4 Kbytes		
0x2001 A000	0x2001 AFFF	4 Kbytes		
0x2001 9000	0x2001 9FFF	4 Kbytes		
0x2001 8000	0x2001 8FFF	4 Kbytes		
0x2001 7000	0x2001 7FFF	4 Kbytes		
0x2001 6000	0x2001 6FFF	4 Kbytes		
0x2001 5000	0x2001 5FFF	4 Kbytes		
0x2001 4000	0x2001 4FFF	4 Kbytes		Clock generator configuration
0x2001 3000	0x2001 3FFF	4 Kbytes		
0x2001 2000	0x2001 2FFF	4 Kbytes	Peripherals	Reserved
0x2001 1000	0x2001 1FFF	4 Kbyte		
0x2001 0C00	0x2001 0FFF	1 Kbyte		
0x2001 0800	0x2001 0BFF	1 Kbyte		
0x2001 0400	0x2001 07FF	1 Kbyte		
0x2001 0000	0x2001 03FF	1 Kbyte		Interconnect configuration
0x2000 0400	0x2000 FFFF	63 Kbytes		Reserved
0x2000 0000	0x2000 03FF	1 Kbyte		INTC

a. Spaces which are reserved or not used allow accesses but return unknown data.
Reserved spaces must not be programmed as this causes unpredictable behavior.

Confidential

14 Memory

14.1 External memory

14.1.1 Programmable CPU memory interface

The programmable CPU memory interface (commonly referred to as the EMI) decodes region 3 of the address space into six banks, into which different external memories and peripherals can be mapped. Two of the banks support SDRAM and one bank is normally used for boot ROM.

- Default locations 0x4000 0000 to 0x5FFF FFFF (banks 0 and 1) are generally used for SDRAM, but may be used for any external memory or peripherals.
- The default locations 0x6000 0000 to 0x6FFF FFFF (bank 2) may be used for any external memory or peripherals except SDRAM.
- Default locations 0x7000 0000 to 0x7FFF FFFF (banks 3, 4 and 5) may be used for any external memory or peripherals except SDRAM, but are generally used for boot ROM. When booting from ROM, the system boots from the predefined location BOOTENTRY (0x7FFF FFFE) at the top of memory space.

Depending on the way the EMI is programmed, the accessing some areas of memory causes special access characteristics (strokes for example) to be generated.

The EMI provides address decoding, address and data buses, timing strobes, enabling signals and refresh where appropriate.

14.1.2 Shared SDRAM memory

The shared SDRAM memory occupies a maximum of 128 Mbits of region 1, and is shared with the MPEG decoders. OSD bitmaps, for example, are stored in this memory. Depending on the configuration, the shared SDRAM memory may only use 64 Mbits, backwardly compatible systems use 32 or 16 Mbits of space.

For details of the shared SDRAM memory interface configuration and set-up, refer to [Chapter 16: External memory interface \(EMI\)](#) and [Chapter 17: External memory interface \(EMI\) registers on page 168](#).

14.2 On-chip SRAM memory

This internal memory module, known as on-chip memory, contains 8 Kbytes of SRAM, which is mapped into the lowest 8 Kbytes of memory space from MININT (0x8000 0000) extending upwards, as shown in [Chapter 13: Memory map](#), [Figure 19: Memory map on page 128](#).

Part of the lowest 4 Kbytes of memory is committed to system use; see [Chapter 13: Memory map on page 128](#) for details. The remainder of the lowest 4 Kbytes of memory is uncommitted and can be used to store on-chip data, stack or code for time-critical routines.

The upper 4 Kbytes of the on-chip memory is also uncommitted SRAM, and is contiguous with the lower 4 Kbytes.

Locations between 0x8000 2000 and 0xBFFF FFFF should not be addressed.

14.3 Caching

Cache can be used to reduce the average access delay imposed on the CPU when it accesses a memory location to read or write. Some locations should not be cached, for example those to which other modules have direct memory access.

The STi5516 cache subsystem provides:

- 8 Kbytes of direct-mapped write-back data cache,
- 8 Kbytes of direct-mapped read-only instruction cache.

The cache configuration is held in memory-mapped registers. The registers must be accessed using the device access instructions.

Device access instructions can also be used to force access to external memory without going through the cache. These instructions can be used to resolve any cache coherency issues. Device writes do not change the value in the cache.

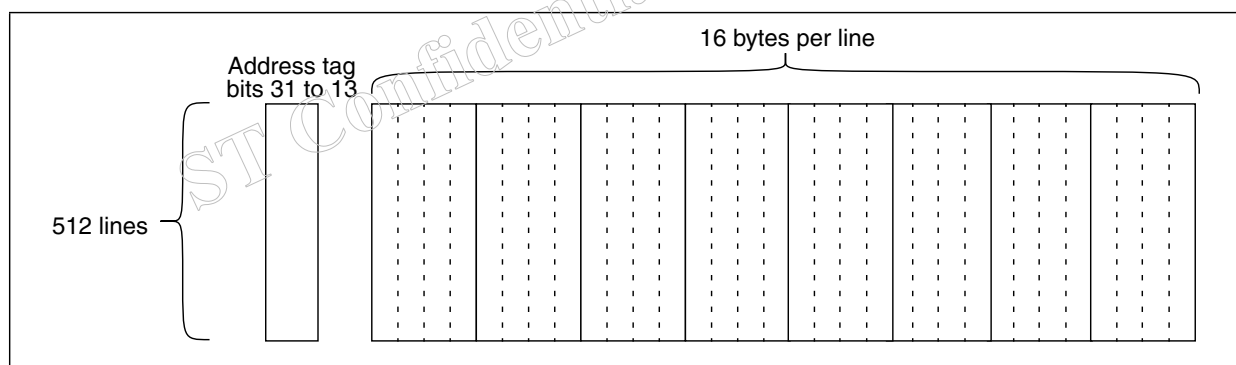
Registers are provided to configure areas of memory that are cacheable or noncacheable for data access, as described in *Cacheable and noncacheable memory locations* on page 134.

Note: The correct cache initialization sequences, described in *Cache initialization* on page 133, must be used before the caches are enabled.

14.3.1 Outline of operation

The cache is four 32-bit words (16 bytes) wide and 512 lines (8 Kbytes, 1024 words) high. It is direct-mapped (sometimes called 'one way set associative'). This is shown in Figure 20.

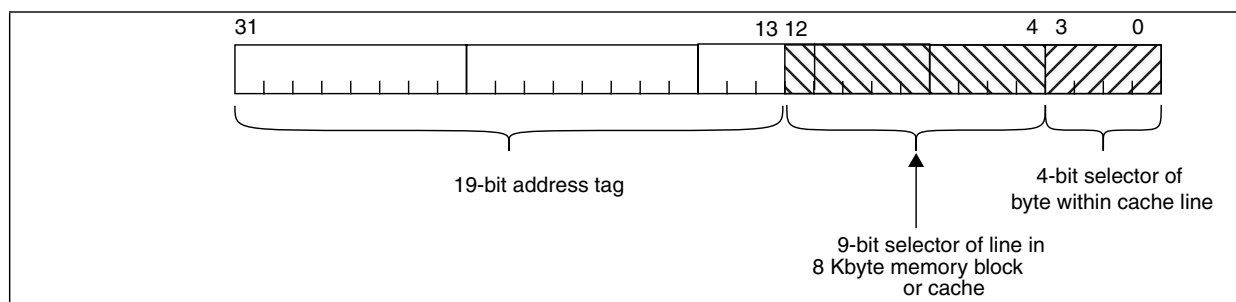
Figure 20: 8 Kbyte data or instruction cache



Each line of the cache can only store data from specific four-word sections of memory at 8 Kbyte intervals, with the bottom line of the cache coinciding with the four words just above each 8 Kbyte boundary. Thus the line number of the cache pinpoints the four-word section of memory within a 8 Kbyte block, that is, bits 4 to 12 of the address.

The 19 most significant bits of the address selects the 8 Kbyte block. These 19 bits are stored in 512 tag registers, with one tag register corresponding to each cache line. The significant parts of the address, when using the cache, are shown in Figure 21.

Figure 21: Address fields when using cache



If a request is made to access a cacheable memory location, and a copy of that location is held in cache, then the access is said to have made a cache hit. A hit is identified by comparing the address bits 13 to 31 with the address tag for the cache line given by the address bits 4 to 12. If the cache is hit, then the access is completed by the cache subsystem. If the cache is missed, the appropriate cache line is written back to memory, and if necessary, the new location in memory is read into that cache line. All cache reads and writes to memory are complete lines due to the efficiency of accessing the memory in burst mode.

14.3.2 Cache initialization

Before the caches are enabled, they must be correctly initialized. To do this the cache must first be invalidated before it is accessed. To ensure this occurs, the invalidate bit of each cache must be set with the cache disabled and then the enable bit set to enable the cache.

This sequence has the effect of forcing a cache to be invalid, which initializes the cache state before any other accesses are considered by it.

14.3.3 Cache subsystem control

The cache subsystem registers control cache functions such as flushing and invalidation, and are used to mark sections of memory space as cacheable or not cacheable. Registers should be accessed using the device access instructions.

14.3.4 Data cache

Flushing the cache means forcing a write back to memory of every dirty line in the cache. A dirty line is a line of cache that has been written to since it was loaded or last written back. Only the data cache can be flushed; the instruction cache never needs flushing since it is read-only.

To flush the data cache, set the FLUSH register to 1. It is automatically reset to 0 on completion of the task. Any memory accesses that are cacheable, which were started before the flush of the data cache is complete, is blocked until it is completed.

14.3.5 Instruction cache

The instruction cache can be selected by writing 1 to the CACHEING_ENABLE register; the default condition is no instruction cache.

The instruction cache must be enabled before it is used therefore, by default, it is disabled.

Invalidating a cache marks every line as not containing valid data. This is done by setting the INVALIDATE register to 1. This register is automatically reset to 0 on completion of the task.

Any instruction fetches that are cacheable, and were started before completion of the invalidation of the instruction cache, is blocked until it is completed.

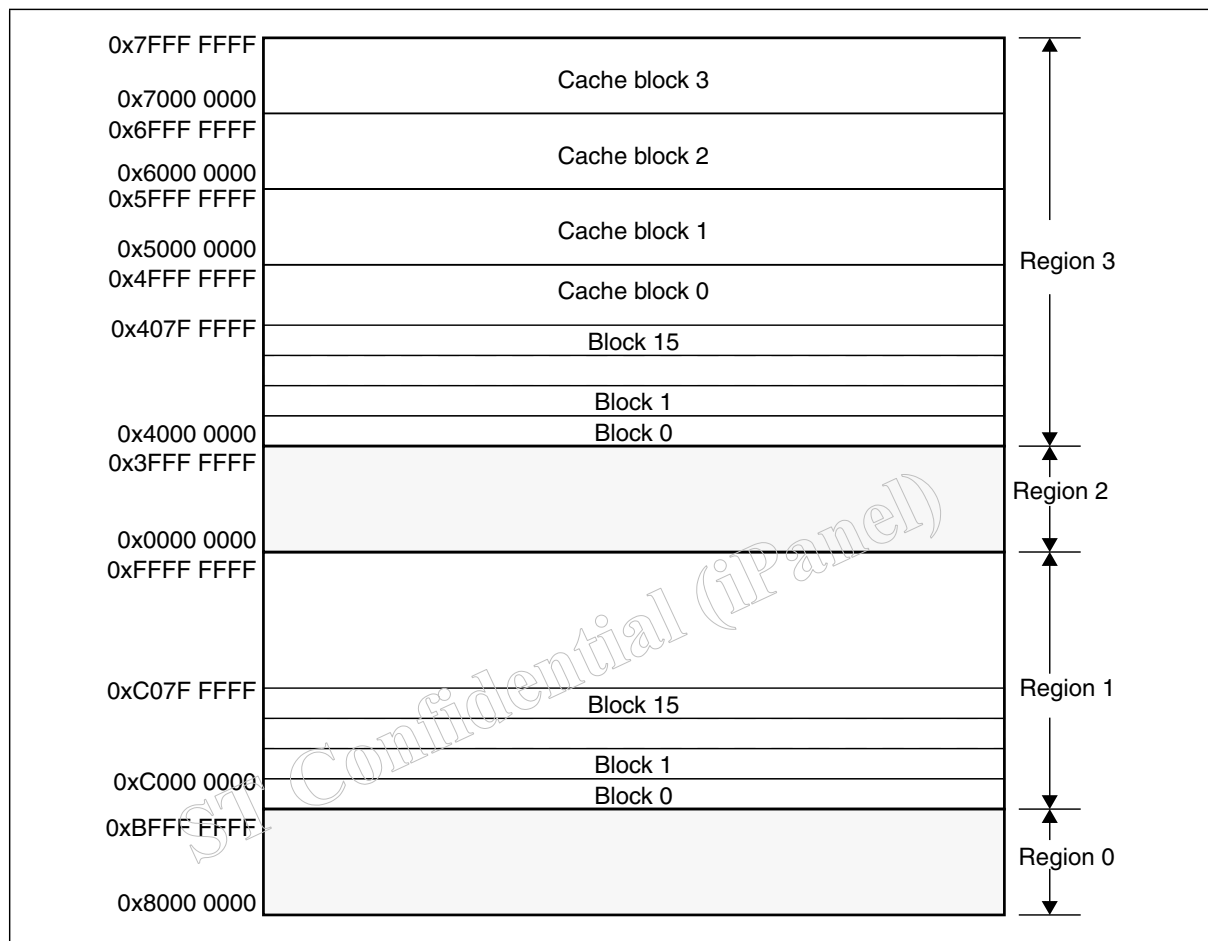
If the instruction cache is enabled, the cache contents is random and must be invalidated by setting the invalidate bit first before being enabling.

The STATUS register is read-only, and shows the current state of the caches.

14.3.6 Cacheable and noncacheable memory locations

It may be desirable for some locations in memory to not be cached. For example, where other units have direct memory access, the cache could get out of step with the memory, that is, the cache could become incoherent.

Figure 22: Cacheability memory map



The six EMI banks do not directly map to the cacheable blocks shown in [Figure 22](#). However it is possible to program them to correspond by specifying the top address for each bank and the number of banks enabled (see [Chapter 20: EMI buffer on page 199](#) for further details). The cacheable regions are fixed by the cacheability control registers, REGION_1_BANK_ENABLE and REGION1_TOP_ENABLE for SMI and REGION_3_BLOCK_ENABLE and REGION_3_BANK_ENABLE for EMI.

The registers may be either programmed at start up or configured dynamically. If they are to be set up once and for all then it is recommended that they are locked afterwards, using the lock register, to ensure that they cannot be accidentally overwritten. If, however, they are to be changed dynamically during operation then the following procedure should be used:

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

15 Memory registers

Addresses are provided as the *BaseAddress* + offset.

The *BaseAddress* are:

DCacheBaseAddress: 0x3000 5000,

ICacheBaseAddress: 0x3000 4000.

A register summary is given in [Table 44: Memory registers on page 69](#).

CACHEING_ENABLE Cacheing enable 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																CACHE_ENABLE															

Address: *ICache or DCache BaseAddress* + 0x00

Type: Read/write

Reset: 0

Description:

[31:1] **Reserved**

[0] **CACHE_ENABLE**

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

0: No access is cacheable

INVALIDATE Invalidate 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																INVALIDATE															

Address: *ICache or DCache BaseAddress* + 0x10

Type: Write only

Reset: 0

Description:

[31:1] **Reserved**

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

FLUSH**Flush 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																FLUSH															

Address: *DCache BaseAddress + 0x14*

Type: Write only

Reset: 0

Description:

[31:1] **Reserved**

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

STATUS**Status 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																READY			FLUSH_START			INV_START									

Address: *ICache or DCache BaseAddress + 0x18*

Type: Read only

Reset: 0

Description:

[31:3] **Reserved**

[2] **READY**: Status of cache

1 means ready and that any flush or invalidate process has finished

[1] **FLUSH_START**: Status of flush process

0: Reset when ready, flush is complete when ready returns to 1. This is not applicable to ICACHE.

[0] **INVALIDATE_START**: Status of invalidate process

0: Reset when ready, flush is complete when ready returns to 1. An invalidate is also performed on reset.

REGION_0_ENABLE**Region 0 enable 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																REGION_ENABLE															

Address: *ICache or DCache BaseAddress + 0x20*

Type: Read/write

Reset: 0

Description:

[31:1] **Reserved**

[0] **REGION_ENABLE**

1: Reserved

0: Region 0 (0x8000 0000 to 0xBFFF FFFF) is not cached

Confidential

REGION_1_BLOCK_ENABLE Region 1 enable 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																BLOCK_15	BLOCK_14	BLOCK_13	BLOCK_12	BLOCK_11	BLOCK_10	BLOCK_9	BLOCK_8	BLOCK_7	BLOCK_6	BLOCK_5	BLOCK_4	BLOCK_3	BLOCK_2	BLOCK_1	BLOCK_0

Address: *ICache or DCache BaseAddress + 0x28*

Type: Read/write

Reset: 0

Description:

[31:16] **Reserved**[15] **BLOCK_15**

1: Block 15 (0xC078 0000 to 0xC07F FFFF) uses the cache

0: It is not cacheable

[14] **BLOCK_14**

1: Block 14 (0xC070 0000 to 0xC077 FFFF) uses the cache

0: It is not cacheable

[13] **BLOCK_13**

1: Block 13 (0xC068 0000 to 0xC06F FFFF) uses the cache

0: It is not cacheable

[12] **BLOCK_12**

1: Block 12 (0xC060 0000 to 0xC067 FFFF) uses the cache

0: It is not cacheable

[11] **BLOCK_11**

1: Block 11 (0xC058 0000 to 0xC05F FFFF) uses the cache

0: It is not cacheable

[10] **BLOCK_10**

1: Block 10 (0xC050 0000 to 0xC057 FFFF) uses the cache

0: It is not cacheable

[9] **BLOCK_9**

1: Block 9 (0xC048 0000 to 0xC04F FFFF) uses the cache

0: It is not cacheable

[8] **BLOCK_8**

1: Block 8 (0xC040 0000 to 0xC047 FFFF) uses the cache

0: It is not cacheable

[7] **BLOCK_7**

1: Block 7 (0xC038 0000 to 0xC03F FFFF) uses the cache

0: It is not cacheable

[6] **BLOCK_6**

1: Block 6 (0xC030 0000 to 0xC037 FFFF) uses the cache

0: It is not cacheable

[5] **BLOCK_5**

1: Block 5 (0xC028 0000 to 0xC02F FFFF) uses the cache

0: It is not cacheable

[4] **BLOCK_4**

1: Block 4 (0xC020 0000 to 0xC027 FFFF) uses the cache

0: It is not cacheable

[3] **BLOCK_3**

1: Block 3 (0xC018 0000 to 0xC01F FFFF) uses the cache

0: It is not cacheable

[2] **BLOCK_2**

1: Block 2 (0xC010 0000 to 0xC017 FFFF) uses the cache

0: It is not cacheable

[1] **BLOCK_1**

1: Block 1 (0xC008 0000 to 0xC00F FFFF) uses the cache

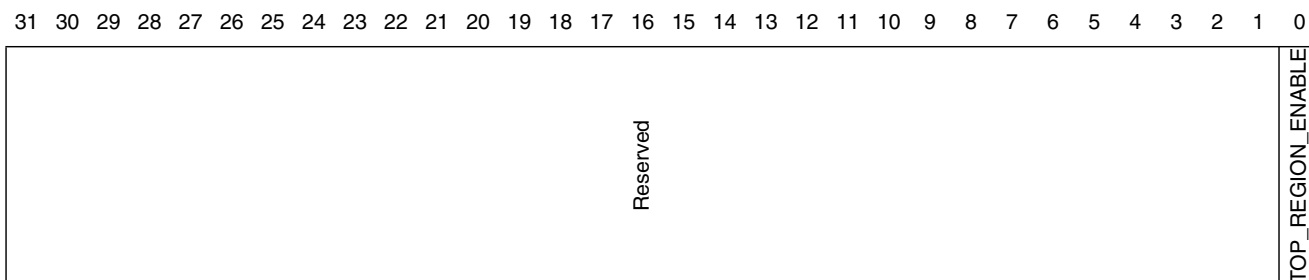
0: It is not cacheable

[0] **BLOCK_0**

1: Block 0 (0xC000 0000 to 0xC007 FFFF) uses the cache

0: It is not cacheable

Confidential

REGION_1_TOP_ENABLE **Region 1 top enable register**

Address: *ICache or DCache BaseAddress + 0x2C*

Type: Read/write

Reset: 0

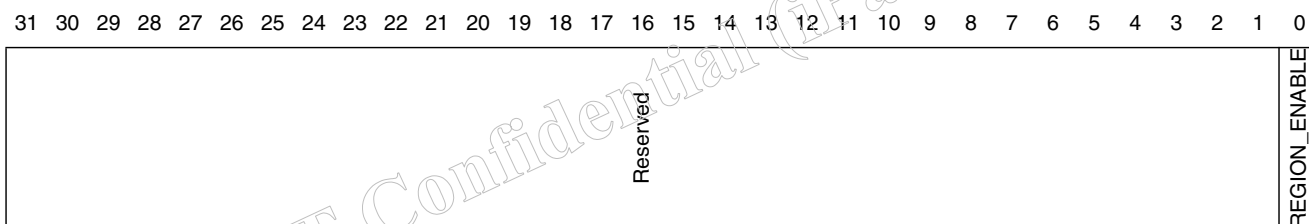
Description:

[31:1] **Reserved**

[0] **TOP_REGION_ENABLE**

1: All of top region 1(0xC080 0000 to 0xFFFF FFFF) is cacheable. Writing 1 to this register causes all memory above 0xC080 0000 to be cached.

0: This area is not cached

REGION_2_ENABLE **Region 2 enable register**

Address: *ICache or DCache BaseAddress + 0x30*

Type: Read/write

Reset: 0

Description:

[31:1] **Reserved**

[0] **REGION_ENABLE**

1: All of Region 2 (0x0000 0000 to 0x3FFF FFFF) is cacheable. Do not write 1 to this register as it causes the device to malfunction.

0: This area is not cached

Confidential

REGION_3_BLOCK_ENABLERegion 3 enable 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																BLOCK_15	BLOCK_14	BLOCK_13	BLOCK_12	BLOCK_11	BLOCK_10	BLOCK_9	BLOCK_8	BLOCK_7	BLOCK_6	BLOCK_5	BLOCK_4	BLOCK_3	BLOCK_2	BLOCK_1	BLOCK_0

Address: ICache or DCache BaseAddress + 0x38

Type: Read/write

Reset: 0

Description:

[31:16] **Reserved**[15] **BLOCK_15**

1: Block 15 (0x4078 0000 to 0x407F FFFF) uses the cache 0: It is not cacheable

[14] **BLOCK_14**

1: Block 14 (0x4070 0000 to 0x4077 FFFF) uses the cache 0: It is not cacheable

[13] **BLOCK_13**

1: Block 13 (0x4068 0000 to 0x406F FFFF) uses the cache 0: It is not cacheable

[12] **BLOCK_12**

1: Block 12 (0x4060 0000 to 0x4067 FFFF) uses the cache 0: It is not cacheable

[11] **BLOCK_11**

1: Block 11 (0x4058 0000 to 0x405F FFFF) uses the cache 0: It is not cacheable

[10] **BLOCK_10**

1: Block 10 (0x4050 0000 to 0x4057 FFFF) uses the cache 0: It is not cacheable

[9] **BLOCK_9**

1: Block 9 (0x4048 0000 to 0x404F FFFF) uses the cache 0: It is not cacheable

[8] **BLOCK_8**

1: Block 8 (0x4040 0000 to 0x4047 FFFF) uses the cache 0: It is not cacheable

[7] **BLOCK_7**

1: Block 7 (0x4038 0000 to 0x403F FFFF) uses the cache 0: It is not cacheable

[6] **BLOCK_6**

1: Block 6 (0x4030 0000 to 0x4037 FFFF) uses the cache 0: It is not cacheable

[5] **BLOCK_5**

1: Block 5 (0x4028 0000 to 0x402F FFFF) uses the cache 0: It is not cacheable

[4] **BLOCK_4**

1: Block 4 (0x4020 0000 to 0x4027 FFFF) uses the cache 0: It is not cacheable

[3] **BLOCK_3**

1: Block 3 (0x4018 0000 to 0x401F FFFF) uses the cache 0: It is not cacheable

[2] **BLOCK_2**

1: Block 2 (0x4010 0000 to 0x4017 FFFF) uses the cache 0: It is not cacheable

[1] **BLOCK_1**

1: Block 1 (0x4008 0000 to 0x400F FFFF) uses the cache 0: It is not cacheable

[0] **BLOCK_0**

1: Block 0 (0x4000 0000 to 0x4007 FFFF) uses the cache 0: It is not cacheable

Confidential

16 External memory interface (EMI)

16.1 Overview

The EMI is a general purpose external memory interface which allows the system to support a number of memory types, external process interfaces and devices. This includes glueless support for up to six independent memories or devices. The EMI allows external devices to become master of the memory bus to support features such as external DMAs and bus mastering.

The main features include:

- up to 100 MHz operating frequency,
- support for up to six external memory banks,
- SDRAM support with possible subdecoding (connectable banks 0 to 5),
- burst flash support (AMD AM29BL162C, STMicroelectronics M58LW064, Intel 28F160F3 compatible) (connectable banks 0 to 5),
- peripheral support (SRAM and ROM) (connectable banks 0 to 5),
- external bus multimaster (DMA and other) support.

Note: Refer to [Section 18.2.15: Bank configurations on page 191](#) for specific STi5516 settings.

The EMI memory map is divided into six regions (EMI banks) which may be independently configured to accommodate one of SRAM, ROM, burst flash, or SDRAM.

Each bank can only accommodate one type of device, but different device types can be placed in different banks to provide glueless support for mixed memory systems.

EMI endianness is fixed at system reset 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.

A maximum of two banks may be configured as SDRAM at the same time, though these banks may be address subdecoded to provide glueless connection to several devices. If only one bank is dedicated to SDRAM, then subdecoding two subbanks is possible. If two SDRAM banks are used, both banks allow subdecoding for up to four subbanks, two for each bank.

An additional EMI bank (bank 5) may also be address subdecoded for other peripheral devices, for example flash.

The external data bus can be configured to be either 16 or 8 bits wide on a per bank basis.

Table 93: Possible configurations for each EMI bank

EMI bank	SDRAM	Other peripheral	EMI bank subdecoding	DVB-CI, hard disk drive ^a
0	✓	✓	✓ (SDRAM only)	-
1	✓	✓	✓ (SDRAM only)	-
2	-	✓	-	-
3	-	✓	-	✓
4	-	✓	-	-
5	-	✓	✓	-

a. Simple PIO mode hard disk drive support, memory-mapped on the EMI

Note: An EMI bank can be address subdecoded into two subbanks. Each subbank has the configuration properties of the bank that is subdecoded.

Another bank's chip select strobe is used as the subbank chip select, so that there may be two chip selects per bank, one for each subbank. This allows up to four SDRAM devices to be address-mapped into the address spaces of two EMI banks and two other peripherals to be mapped into the address space of another bank.

16.2 Operation

The EMI is a highly flexible memory device which is able to support a large range of memory components gluelessly. It accepts memory operations from the system and, depending on the address of the operation, either accesses its internal configuration space or one of the possible six external memory banks.

The position, size, clock frequency and memory type supported is dependent on how the associated control registers, EMI_BANKS[0:5], are programmed.

Following reset, all banks start with the same configuration which allows the system to boot from a large range of nonvolatile memory devices.

As part of the boot process, the user should program the EMI configuration registers to match the memory supported in that system, defining the memory size, the location in the address and the device type connected.

16.2.1 Bank programming

Please refer to [Section 18.2.15: Bank configurations on page 191](#) for full bank programming details.

16.2.2 Clock reconfiguration for synchronous interfaces

Following reset, the clocks for synchronous interfaces are disabled. This is due to the default reset assuming a memory which may be accessed asynchronously.

To access the synchronous memory, the user sets up the configuration state associated with that bank. The user then programs the required clock ratio in the register EMI_xxxCLKSEL associated with that memory type.

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

16.2.3 Operating in master mode

On the STi5516 device the EMI is always the bus master. The internal bus arbitration signals HOLD_REQ and HOLD_ACK are not active, HOLD_ACK is held low and HOLD_REQ is held low by the EMI block.

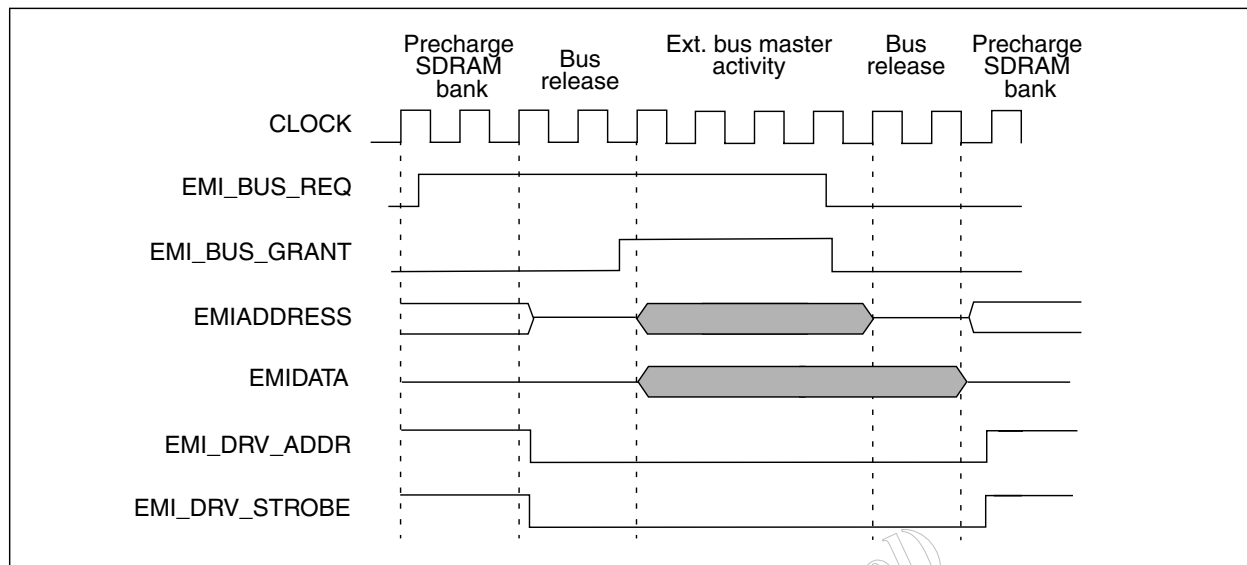
Note: To enable the EMI to work in master mode for synchronous devices, the SHIFT_CONFIG register must be set. See [Chapter 52: Clock generator and the register SHIFT_CONFIG on page 568](#) for programming details.

In this mode, the EMI is the bus master. A bus request from an external device (DMA or any device which is nonMPX) can occur asserting the signal BUS_REQ. The purpose of this is to allow the STi5516 to boot without interruption before another agent can access the bus.

Once the EMI_BUS_REQ has been enabled, and before an external bus request is granted, the generic EMI block ensures that the current precharge time for the previous SDRAM bank (if present) access is satisfied. The EMI block then checks if the bus release time is satisfied for the previous access. The bus release time is the time required for an external device to tri-state the

data bus. During this period the EMI_DRV_ADDR and EMI_DRV_STROBE signals are set to low. If the bus release time has been satisfied, then the request is granted by taking EMI_BUS_GRANT high.

Figure 23: Signal timings



External memory access is terminated by taking EMI_BUS_REQ low. The external agent uses the bus release time (which is fixed) to tri-state the address and data bus before the EMI drives it again. Since the EMI has no idea of the external activity, the slowest memory device is applied before allowing further external accesses. If the bus release time is satisfied the EMI starts to drive the bus again taking the EMI_DRV_ADDR and EMI_DRV_STROBE signals high. If there is a SDRAM bank, it is then assumed that precharging is required before being accessed (in the case of two SDRAM banks the EMI chooses the highest precharge time to satisfy).

Whilst in the release bus state, the EMI may signal its intent to use the external buses by taking the EMI_RFSHPEND signal high. The external bus is then relinquished by the external agent taking EMI_BUS_REQ low. The external agent must ensure that EMI_BUS_REQ is not taken high again before either the precharge time or largest bus release time has elapsed. If EMI_BUS_REQ is raised too early, pending EMI accesses are not given the chance to execute. This is because external device requests are treated with the higher priority.

16.3 Default/reset configuration

Following reset, a default configuration setting is loaded into all six banks. This allows the EMI to access data from a slow ROM memory. The default settings are detailed in the following sections.

16.3.1 Default configuration for asynchronous boot

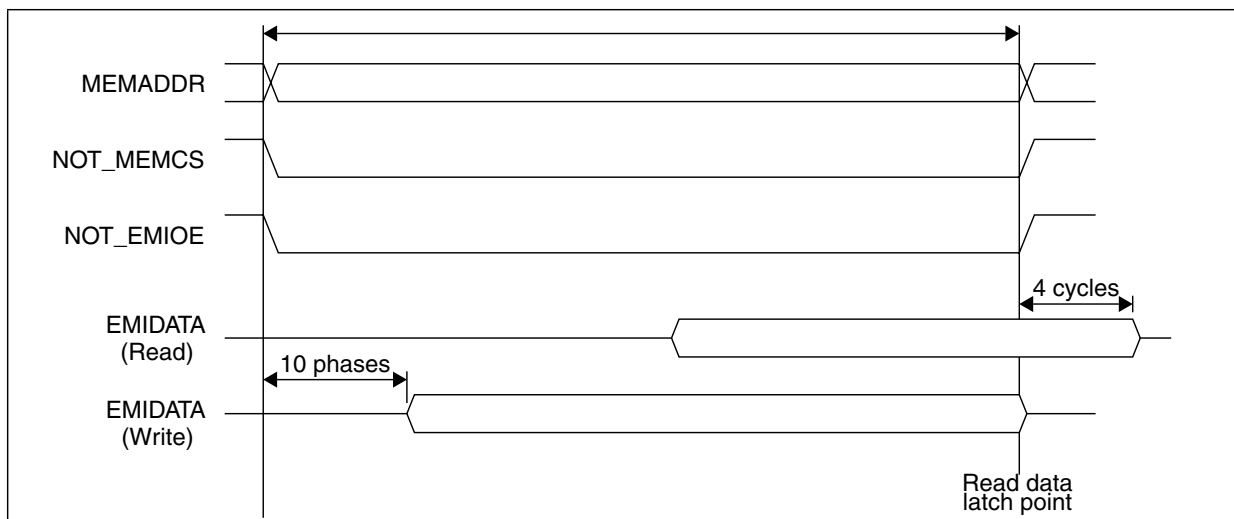
The default configuration setting is loaded into all six banks on reset.

Table 94: Default configuration

Parameter	Default value
DATADRIVEDELAY	10 phases
BUSRELEASETIME	4 cycles
CSACTIVE	Active during read-only
OEACTIVE	Active during read-only
BEACTIVE	Inactive
PORTSIZE	Value of the signal EMI4_PRTSZ_INIT
DEVICETYPE	Peripheral
ACCESSTIMERREAD	(18 + 2 = 20 cycles)
CSE1TIMERREAD	0 phases
CSE2TIMERREAD	0 phases
OEE1TIMERREAD	0 phases
OEE2TIMERREAD	0 phases
LATCHPOINT	End of access cycle
WAITPOLARITY	Active high
CYCLENOTPHASE	Phase
BE1TIMERREAD	3 phases
BE2TIMERREAD	3 phases

The remaining configuration parameters are not relevant for an asynchronous boot, that is the aim of the default configuration.

Figure 24: Default asynchronous configuration



16.4 Peripheral interface with synchronous flash memory support

16.4.1 Overview

A generic peripheral (for example SRAM, EPROM, SFlash™) access is provided which is suitable for direct interfacing to a wide variety of SRAM, ROM, flash, SFlash™ and other peripheral devices. No subdecoding is possible with banks configured to hold a peripheral configuration.

Note: Please refer to [Section 18.2.15: Bank configurations on page 191](#) for specific STi5516 settings.

Figure 25 shows a generic access cycle and the allowable values for each timing field are shown.

Figure 25: Generic access cycle

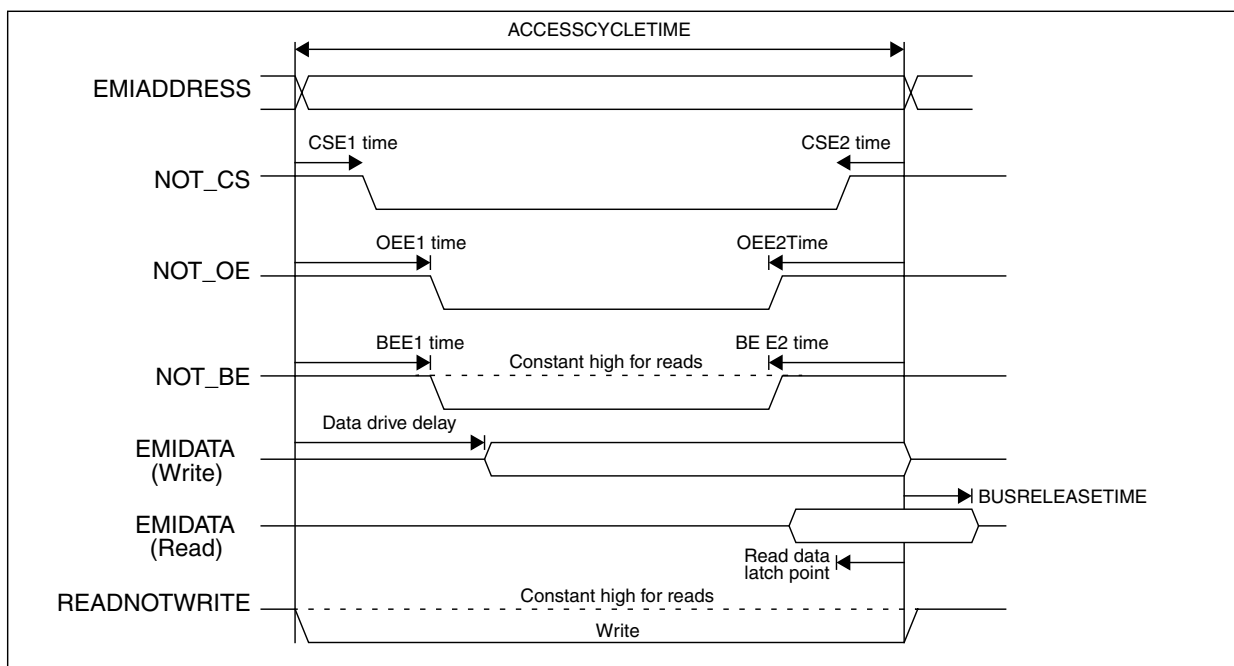


Table 95: Strobe timing parameters for peripheral

Name	Programmable value
ACCESSTIME	2 cycles + 0 to 125 cycles
BUSRELEASETIME	0 to 15 cycles
DATADRIVEDELAY	0 to 31 phases after start of access cycle
CSE1TIME	Falling edge of CS. 0 to 15 phases or cycles after start of access cycle
CSE2TIME	Rising edge of CS. 0 to 15 phases or cycles before end of access cycle
OEE1TIME	Falling edge of OE. 0 to 15 phases or cycles after start of access cycle
OEE2TIME	Rising edge of OE. 0 to 15 phases or cycles before end of access cycle.
BEE1TIME	Falling edge of BE. 0 to 15 phases or cycles after start of access cycle
BEE2TIME	Rising edge of BE. 0 to 15 phases or cycles before end of access cycle
LATCHPOINT	0: End of access cycle. 1 to 16: 1 to 16 cycles before end of access cycle.

Separate configuration parameters are available for reads and writes. In addition, each strobe can be configured to be active on read, write, neither or both.

Table 96: 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

16.4.2 Synchronous burst flash support

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

The EMI supports the following memory devices:

- AMD AM29BL162C,
- STMicroelectronics M58LW064A/B,
- Intel 28F800F3/ 28F160F3.

In [Table 97](#) there is a brief description and comparison of main the features of the flash memories that the EMI can interface with.

Note: Not all memory features are supported. When a feature is not supported, this is highlighted.

Table 97: STMicroelectronics, AMD, Intel flash supported features comparison

	AM29BL162C	STMicroelectronics M58LW064A/B	Intel 28F800F3/ 28F160F3
Size	16 Mbits	64 Mbits	8/16 Mbits
Maximum^a operating frequency	40 MHz	60 MHz	60 MHz
Data bus	16 bits fixed	16/32 bits	16 bits fixed
Main operations	Asynch single access write Synch burst read Asynch single access read	Asynch single access write Synch burst read Asynch single access read Asynch page read Not supported by EMI	Asynch single access write Synch burst read Asynch single access read Asynch page read Not supported by EMI Synch single access read Not supported by EMI
Burst size	32 word	1-2-4-8 words ^b or continuous Set by burst config register	4 to 8 words or continuous Set by read config register
Burst style^c	Linear burst -32 words	Sequential burst	Linear burst
X-latency^d	70-90-120 ns	7-8-9-10-12 ^e cycles	2-3-4-5-6 cycles
Y-latency^f	1 cycle	1-2 cycles	1 cycle
Burst suspend/resume^g	Yes via burst address advance (BAA) input	Yes via burst address advance (B) input	No automatic advance
Ready/busy pin^h	Yes (RD/BY)	Yes (RD/BY)	No
Ready for burstⁱ	No	Yes (R)	Yes (W)

- The flash operating frequency, clock divide ratios and system frequency should be consistent with the maximum operating frequency.
- A burst length of eight words is not available in the x32 data bus configuration.
- 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 asynch write operations. On STMicroelectronics and Intel devices the burst is enabled synchronously via the burst configuration register.
- 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
- 10 to 12 only for F = 50 MHz
- Y-latency is the time elapsed from the current valid data that is output to the next data valid in output during a burst
- In AMD and STMicroelectronics devices, BAA (or B) can be tied active. This means that the address advance during a burst is noninterruptable (Intel likewise). EMI assumes these pins are tied active and does not generate a BAA signal.
- 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
- 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.

EMI implements a superset of operational modes so that it is compatible with most of the main functions listed for the three flash families. The following sections contain a brief description of the EMI flash interface functionality.

16.4.3 Operating mode

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 EMI_FLASHCLKSEL register.

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

2 32 words burst size is partially supported by the EMI; the burst is interrupted when the required data has been read.

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

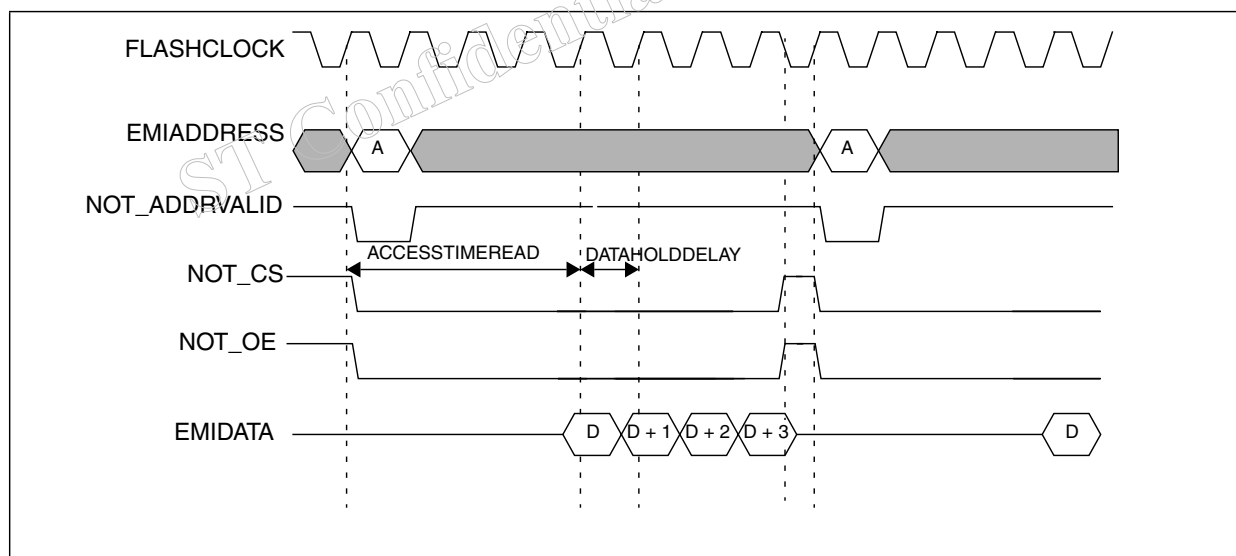
4 Interleaved burst mode is not supported by EMI because of the implementation of multiple reads only using synchronous burst mode (feature provided by all the three families of flash chips adopted).

EMI supports a asynchronous single write.

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

In [Figure 26](#) a typical burst access with burst length of four words is shown.

Figure 26: Synchronous burst mode flash read (burst length = 4)



The ACCESSSTIMEREAD 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.

16.4.4 Burst interrupt and burst reiteration

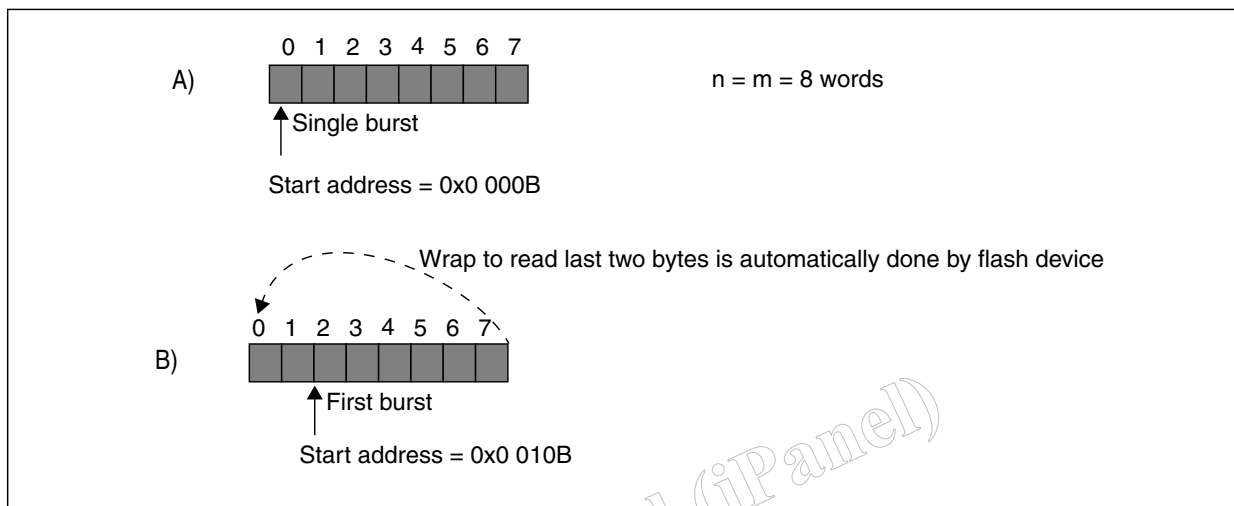
The EMI interrupts the burst after the required amount of data has been read; thus making the chip select of the burst device inactive. This operation is allowed by all the three families of flash devices (burst read interrupt for an STMicroelectronics device, standby for Intel, terminate current burst 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 (EMI_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 EMI on the STBus interface, there are three possible outcomes.

- **$n = m$**

The EMI performs one burst access during which it gets the exact number of words as requested (see example A on [Figure 27](#) with $n = m = 8$). Depending on the starting address, there is possibly a wrap that is automatically completed by the flash device. The wrap occurs when the starting address is not aligned on an n -byte word boundary.

Figure 27: Burst on a flash with a single access



- **$n > m$**

If the starting address is aligned on an m -byte word boundary, the EMI gets m bytes from a single burst sequence as explained in the previous paragraph. Then the transfer on flash is interrupted making the chip select inactive. This terminates the burst transfer and puts the memory device in standby mode, waiting for a new request and starting address for a new burst.

If the starting address is not aligned on an m -byte word boundary, a first burst on the flash executes until the m -byte word boundary is crossed. The burst on the flash is interrupted and there follows another burst with a starting address that wraps to an m -byte boundary (directly given by STBus interface) to read the remaining data. After all the required bytes have been read, the burst access on flash can be interrupted.

- **$n < m$**

The EMI needs to perform more burst accesses until it gets the required m words.

If the starting address is aligned on an n -byte word boundary, there are a series of flash burst accesses until the exact number of bytes is met.

If the starting address is not aligned on an n -byte word boundary, there is a first access on flash to read data until the n -byte word boundary is met. This access is then interrupted and new series of accesses are started on a new address provided by STBus (that eventually wraps at the m -bytes boundary). This is repeated until the exact number of bytes is reached. This happens in the middle of the last flash burst that is interrupted in the usual manner.

16.4.5 Synchronous burst enable

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

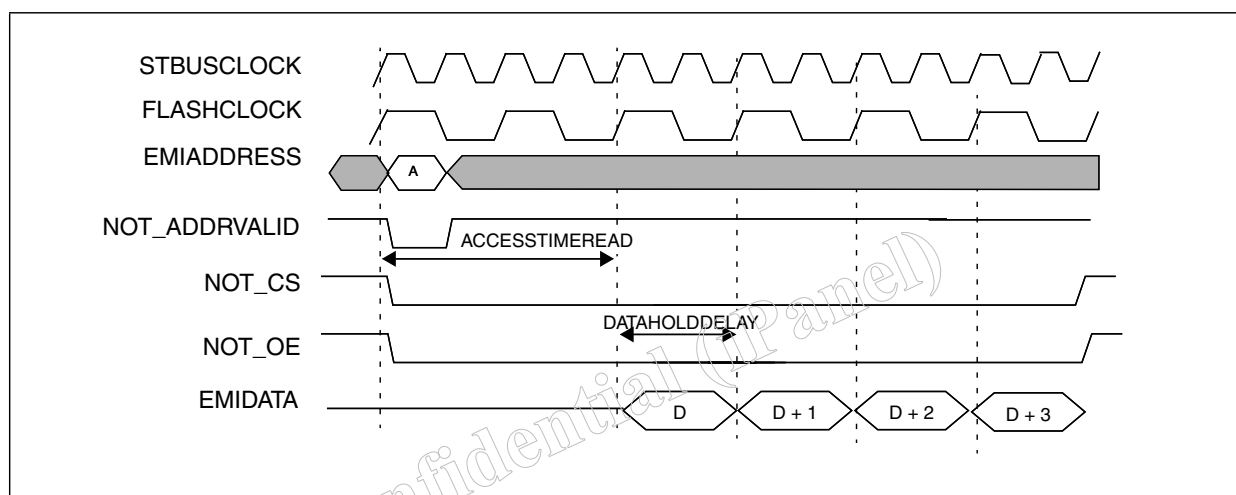
[Table 97: STMicroelectronics, AMD, Intel flash supported features comparison on page 147](#), shows that for STMicroelectronics and Intel devices to operate in synchronous burst mode, the configuration parameters must be set in a special configuration register inside the memory

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.

16.4.6 Support for lower clock rates

Many SFlash™ devices operate in the 30 to 50 MHz clock range (Table 97: *STMicroelectronics, AMD, Intel flash supported features comparison on page 147*) whereas the EMI operates up to a clock frequency of 100 MHz. To deal with this difference, the EMI runs in a lower speed mode. The hardware in the EMI needed for this mode forces accesses to always start on the rising edge of the slower clock. It is up to the user to configure the other EMI timings, to set up and latch, on the appropriate edge of this slower clock.

Figure 28: Half speed EMI SFlash™ clock



16.4.7 Initialization sequence

Peripheral interfaces are used immediately after reset to boot the device. Therefore, the default state must be correct for either synchronous or normal ROM. An SFlash™ device can be interfaced to normal ROM strobes with the addition of only the address valid signal and the clock. When the CPU has run the initial bootstrap, it can configure both the SFlash™ device and the EMI to make use of the burst features.

Note: The flash devices are in asynchronous read mode after reset.

Caution:

The process of changing from default configuration to synchronous mode is not interruptible. Therefore the CPU must not be reading from the device at the same time as changing the configuration as there is a small window where the EMI's configuration is inconsistent with the memory device's.

16.4.8 Flash subbank decoding

As shown in Table 97: *STMicroelectronics, AMD, Intel flash supported features comparison on page 147*, the maximum size of memory chip for SFlash™ is 64 Mbits. This may not be enough for some of the variants that use the EMI. Hence subdecoding of the address space for the boot bank flash address (bank 5) may be needed. As for the normal peripherals, the chip select is the signal that starts a transaction. Therefore, the best choice in term of flexibility, is to leave the address/chip select subdecoding task to the padlogic (product dependent) as usual.

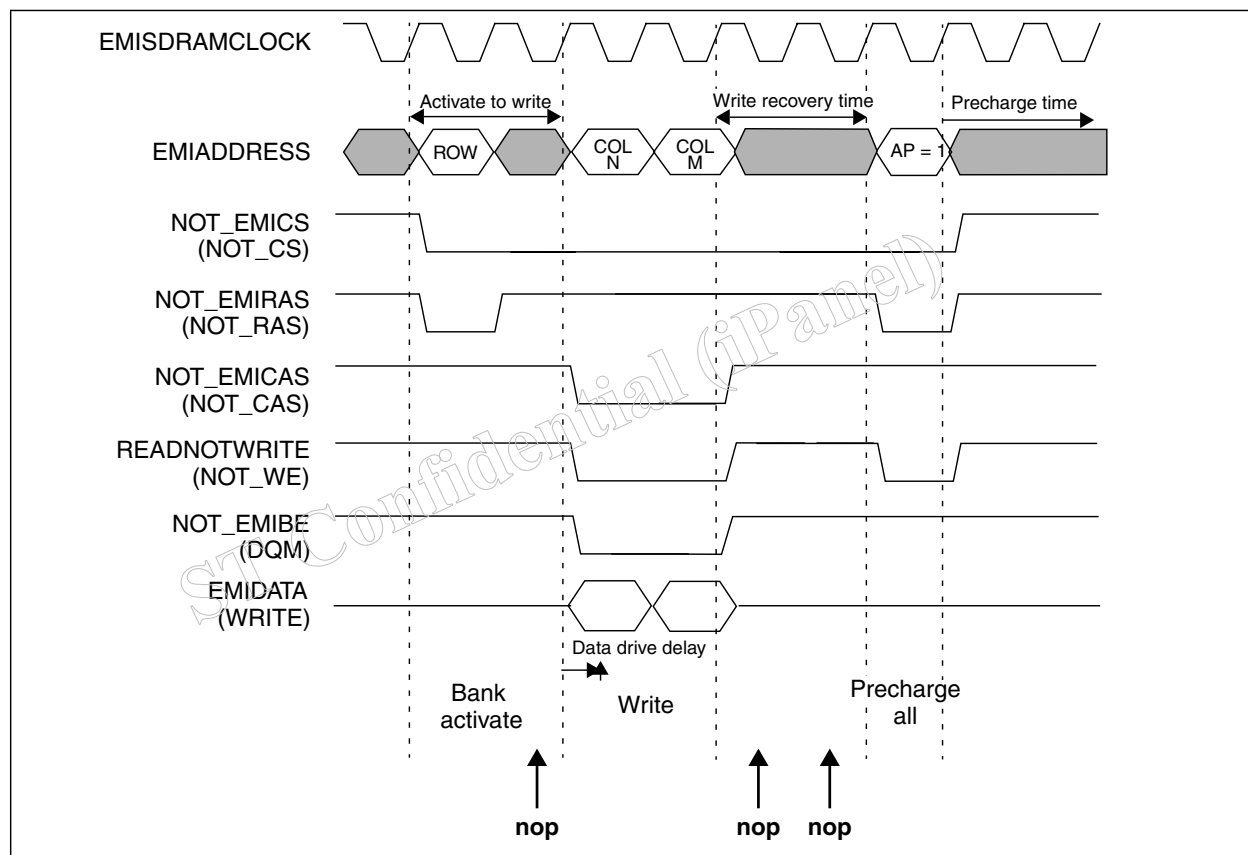
16.5 SDRAM interface

All signals on this interface are synchronous to the SDRAM clock, which can be set to the full STBus clock, 1/2 or 1/3 of this clock. The set up of the EMI_SDRAMCLKSEL sets the SDRAM clock.

16.5.1 Typical access

The following diagram describes a typical write access to an SDRAM. The waveforms show what should appear on the pads of a device containing the generic EMI. This example shows a bank activation, due to a page miss, then two write accesses in the same bank which are performed in page mode.

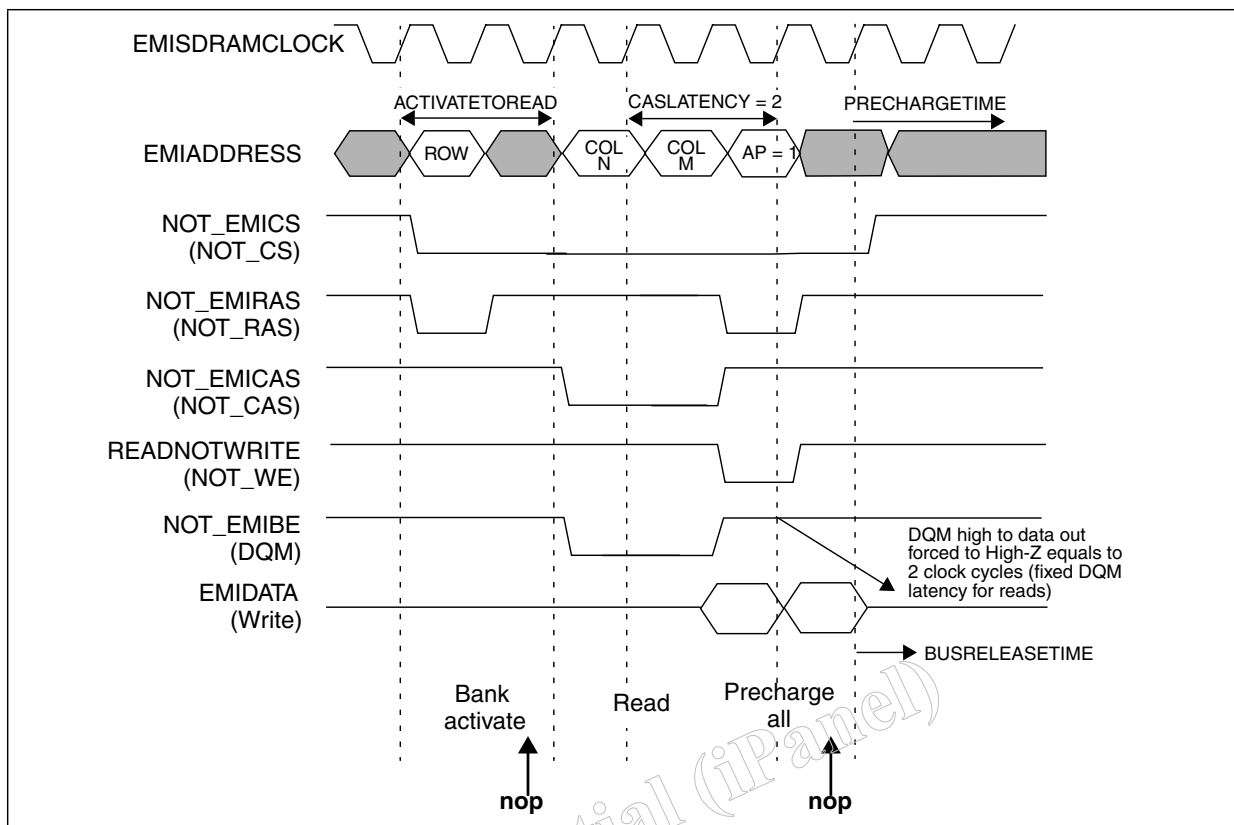
Figure 29: Generic SDRAM write access



A precharge is then completed in anticipation of another bank activation command. If, as in this example, only one SDRAM word is to be written, then the NOT_EMIBE signal is used as a data mask so that only the correct word is updated.

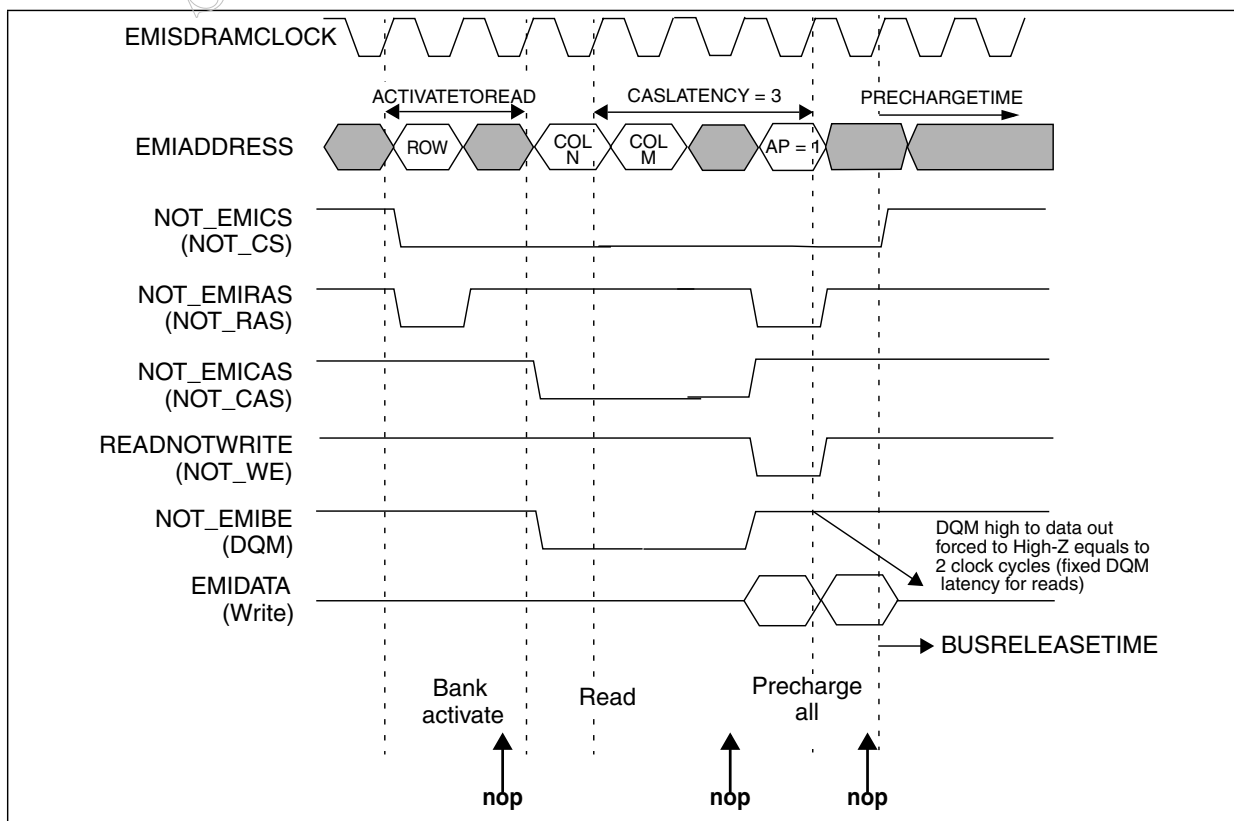
The following figure shows a bank activation, due to a page miss, then two read accesses in the same bank are performed in page mode.

Figure 30: SDRAM read accesses with CASLATENCY = 2 cycles



A precharge is then completed in anticipation of another bank activation command. If, as in this example, only one SDRAM word is to be read, then the NOT_EMIBE signal is used as a data output enable.

Figure 31: SDRAM read accesses with CASLATENCY = 3 cycles



16.5.2 Description of signals

EMIADDRESS

This is driven with the ROW address during an **activate** command, and the COLUMN address (the full address bus without being shifted) during a read or write operation.

The address lines used for the mode selection are controlled using the PORTSIZE configuration register settings, because the address lines effectively shift as the bus width alters.

EMIDATA

For writes, data is driven for each cycle of the burst access, either immediately after start of access cycle, or one phase later (DATADRIVEDELAY configuration parameter). For reads, the data is latched for each cycle of the read, this occurs a number of cycles after the command is sent to the SDRAM (CASLATENCY parameter).

NOT_EMIRAS: NOT_RAS strobe

Normally this is high, however it is low for a bank activate, precharge, refresh cycle. Additionally it is low for the SDRAM mode register initialization cycle. NOT_EMIRAS is shared by all the SDRAM present in the system.

NOT_EMICAS: NOT_CAS strobe

Normally this is high, however it is low in a read, write or refresh cycle. Additionally it is low for the SDRAM mode register initialization cycle. NOT_EMICAS is shared by all the SDRAM present in the system.

NOT_EMICS[A:F]: NOT_CS strobes

These signals select which device an access is destined for. They are normally high and one is asserted low in the cycle when an access is to be made.

NOT_EMIBE[3:0]: DQM strobes

Normally this is high. NOT_EMIBE is asserted low during read and write accesses to enable which bytes/words are accessed.

DQM is a multiple function signal defined as data mask for both reads and writes. During reads, DQM performs synchronous output enable. During writes, DQM performs write data masking. The DQM latency is different for reads and writes. For reads, DQM latency is defined as the difference between the clock when DQM is asserted and the clock when the output bus has been forced to High-Z; its value is always two clock cycles. For writes, DQM latency is defined as the difference between the clock when DQM is asserted and the clock when the write input data is inhibited; its value is always 0.

NOT_EMIBE[n] is only active if the corresponding byte is to be accessed. For single byte accesses only one is active. The behavior of these signals is dependent on the PORTSIZE configuration bits.

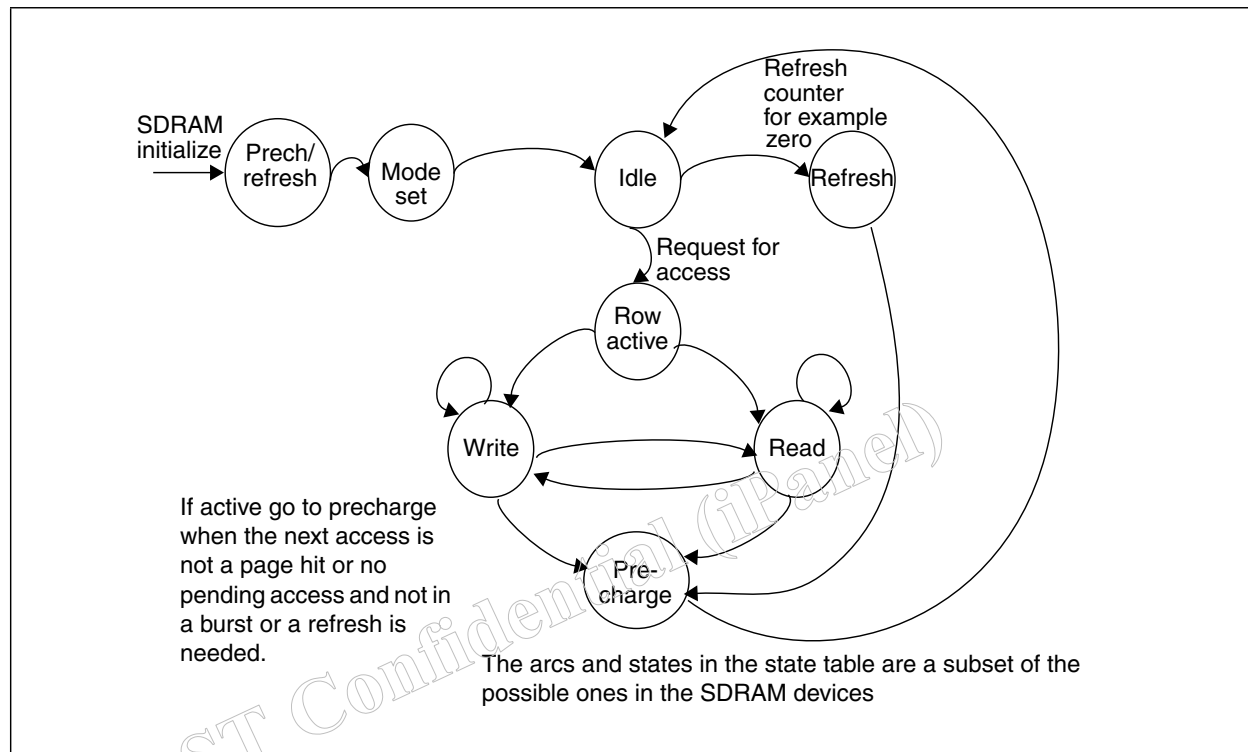
READNOTWRITE: NOT_WE strobe

This signal indicates that the current cycle is a read cycle and is normally high. It is asserted low for the **prechargeall** and **write** commands. Additionally it is low for the SDRAM mode register initialization cycle.

16.5.3 SDRAM controller states

The following diagram shows how the generic EMI controls an SDRAM. These states are a subset of the commands implemented by an SDRAM. The diagram describes the functionality though does not explain the actual implementation, that is these states may not all be implemented by one block within the EMI. For instance the refresh controller takes over from the SDRAM controller when it decides it is time for a refresh operation.

Figure 32: SDRAM controller states in the current EMI implementation



16.5.4 Supported SDRAM commands

The generic EMI supports the following SDRAM commands. The relevant NOT_EMICS signal is active for all these operations.

Table 98: SDRAM commands with NOT_EMICS active

Command	Strobes state			Mode bits		Address	Notes
	NOT_RAS	NOT_CAS	NOT_WE	BS	A10 AP		
prechargeall	0	1	0	X	1	X	Used to close pages after a burst in simple page mode
activate	0	1	1	1/0	1/0	Row addr	Opens a page (row active)
write	1	0	0	1/0	0	Col. addr	Write words Uses byte enables to mask bytes
read	1	0	1	1/0	0	Col. addr	Read words
modeset	0	0	0	1/0	1/0	Mode data	Done once only at start up
refresh	0	0	1	X	X	X	CBR style refresh (autorefresh) or self-refresh ^a
nooperation	1	1	1	X	X	X	Does nothing Same as having CS inactive

a. If clock enable is high, this is autorefresh, else this is self-refresh

16.5.5 Supported operations applicable to a single bank of SDRAM

Using the above set of commands, the current implementation of the EMI supports the operations listed in [Table 99](#).

Table 99: Supported commands for a single SDRAM access

SDRAM current state	Command	Notes
Idle	desl	desl means that NOT_EMICS is inactive.
	activate	The bank specified by the address pins and row address are activated.
	refresh	SDRAM enters in refresh mode.
	self_refresh	The self_refresh command is issued and EMI goes into power down mode.
	mode set	SDRAM enters in mode register set cycle.
Row active	nop	When the number of cycles between the activate of the page and the read/write command is > 1 cycle.
	read	A read operation starts.
	write	A write operation starts.
Read	read	This command is issued for each read access in the same row (page mode).
	nop	This command is performed when a write access follows the current read access and is in the same row (page mode) or before a prechargeall command when CASLATENCY is > 2 clock cycles. If during the cycle in which the nop is issued, a new request comes and it is in the same row, a read or a write command is generated.
	prechargeall	This command occurs each time there is either a new access (read or write) but to a different row address (page miss), an EMI bank switch, no new pending requests from the EMI buffer, a pending refresh or a DMA access. The command is issued in the cycle after the current read, only if CASLATENCY is < 3 clock cycles.
Write	write	This command is issued for each write access in the same row (page mode).
	read	This command occurs when a read access is followed by a write access and within the same row (page mode).
	nop	This command occurs each time there is either a new access (read or write) but to a different row address (page miss), an EMI bank switch, no new pending requests from the EMI buffer, a pending refresh or a DMA access. The command is followed by a prechargeall after the write recovery time is expired. If, in the cycle in which the nop is issued, a new request comes and it is in the same row, a read or a write command is generated.
Precharge all	desl	desl means NOT_EMICS is inactive.
	nop	This command is performed until desl command.
Refresh	desl	desl means NOT_EMICS is inactive.

16.5.6 Multiple banks

If subbank decoding is used, the whole EMI block can support up to four arrays of SDRAM, two in each of EMI banks 0 and 1. An array for an EMI bank with a 16-bit wide data port is equivalent to a 16-bit wide data bus for a physical SDRAM device. The NOT_EMIRAS and NOT_EMICAS strobes are shared by all the devices. EMI subbank selection is made with the EMI_GENCFG bits, EMI_GENCFG[2] for bank 0 and EMI_GENCFG[3] for bank 1. To subdecode either of these banks set the corresponding EMI_GENCFG bit to 1.

One SDRAM bank

In this case only one bank is connected to an SDRAM.

Table 100: Chip selects for bank 0

Number of EMI subbanks	Pin I/O
1 ^a	NOT_EMICSA
2	NOT_EMICSA (lower address) NOT_EMICSC (upper address)

a. The EMI bank is not subdecoded

Table 101: Chip selects for bank 1

Number of EMI subbanks	Pin I/O
1 ^a	NOT_EMICSB
2	NOT_EMICSB (lower address) NOT_EMICSD (upper address)

a. The EMI bank is not subdecoded

Two SDRAM banks.

Table 102: Chip selects for two SDRAM banks

EMI bank	Number of EMI subbanks	Pin I/O name
0	1 ^a	NOT_EMICSA
1	1 ^a	NOT_EMICSB
0	2	NOT_EMICSA (lower address) NOT_EMICSC (upper address)
1	1 ^a	NOT_EMICSB
0	1 ^a	NOT_EMICSA
1	2	NOT_EMICSB (lower address) NOT_EMICSD (upper address)
0	2	NOT_EMICSA (lower address) NOT_EMICSC (upper address)
1	2	NOT_EMICSB (lower address) NOT_EMICSD (upper address)

a. The EMI bank is not address subdecoded

One peripheral bank

Bank 5 can be subdecoded using GENCFG[11]. Set to 1 to enable subdecoding.

Number of EMI subbanks	GENCFG[11]	Pin I/O
1 ^a	0	NOT_EMICSF
2	1	NOT_EMICSE (lower address) NOT_EMICSF (upper address)

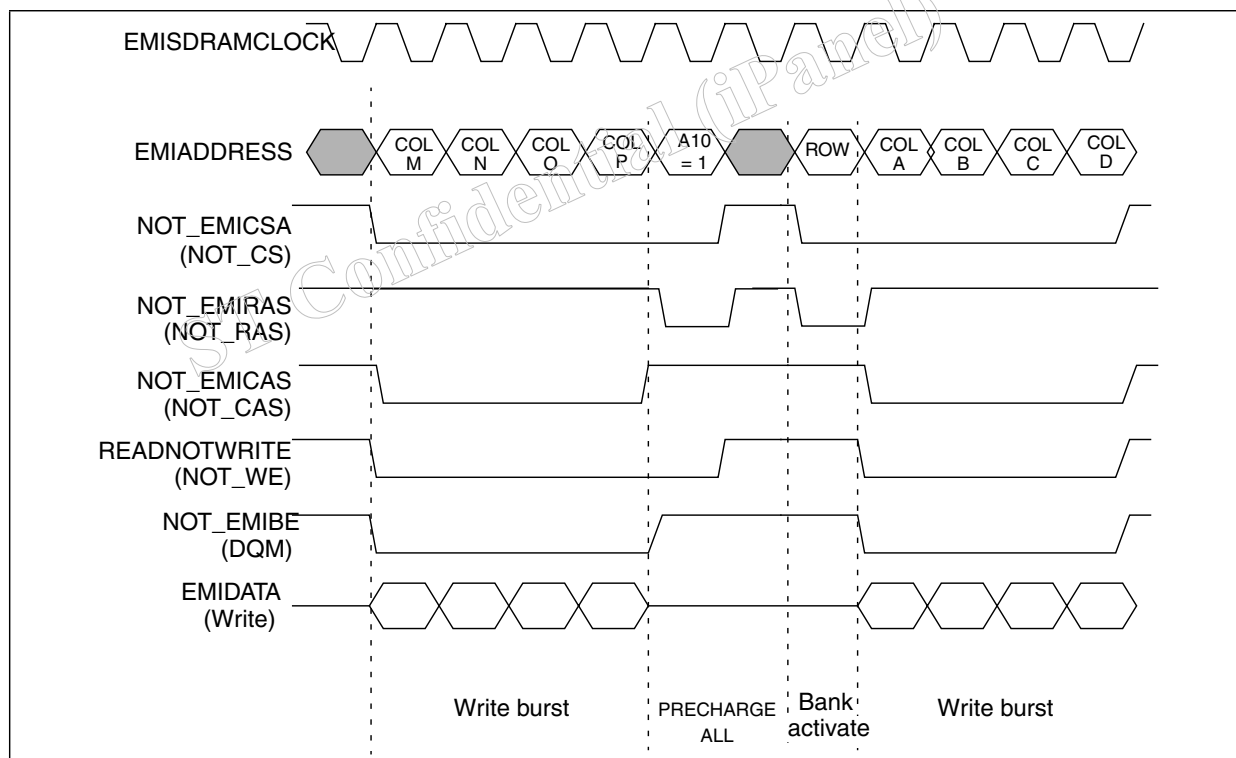
a. The EMI bank is not subdecoded

16.5.7 Burst access behavior

A page hit occurs when a new memory access is in the range of an address in a page already activated. The EMI always accesses the SDRAM device in page mode, if enabled (RASBITS not all to zero) independently by the value of the burst length specified in the mode register.

The EMI does not implement the JEDEC standard for SDRAM devices about random column accesses (also called 2n rule).

Figure 33: Case of burst write accesses (BL = 2, port size = 16 bits)



The **activate** command is used to open a page and allow read and write operations to be completed.

The EMI analyzes the access for a page hit, and if no following access is available, the SDRAMs are precharged. This incurs a precharge and activate delay before making the next access.

16.5.8 SDRAM accesses example

Figure 34: Single WRITE

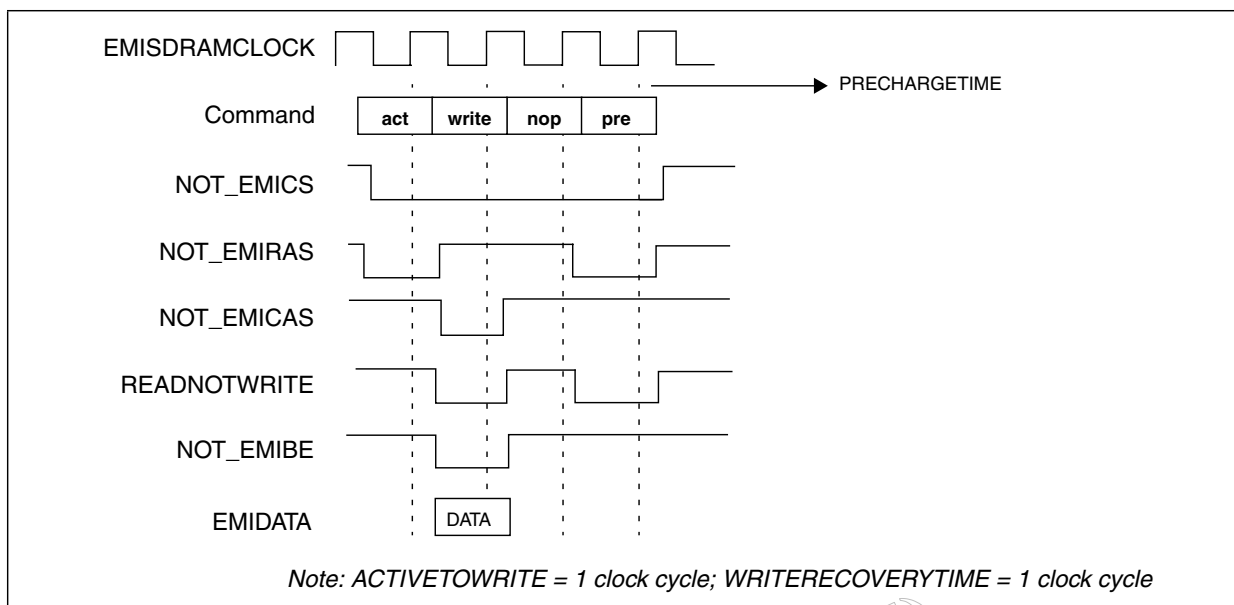
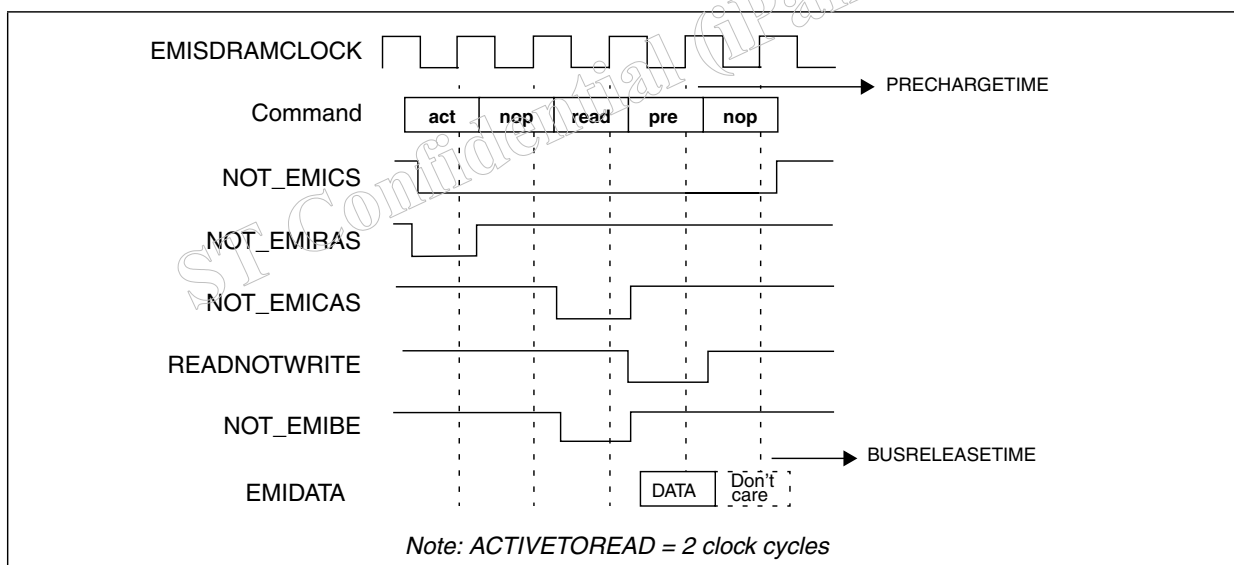


Figure 35: Single READ and CASLATENCY = 1 cycle



Confidential

Figure 36: Pagemode READs and CASLATENCY = 1 cycle

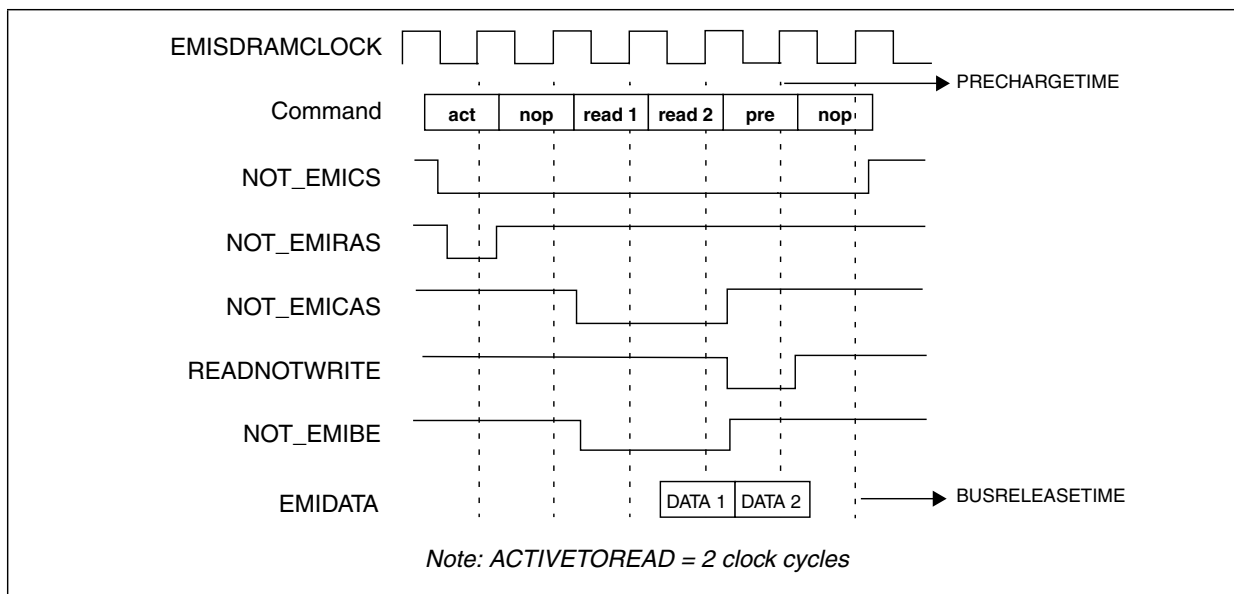
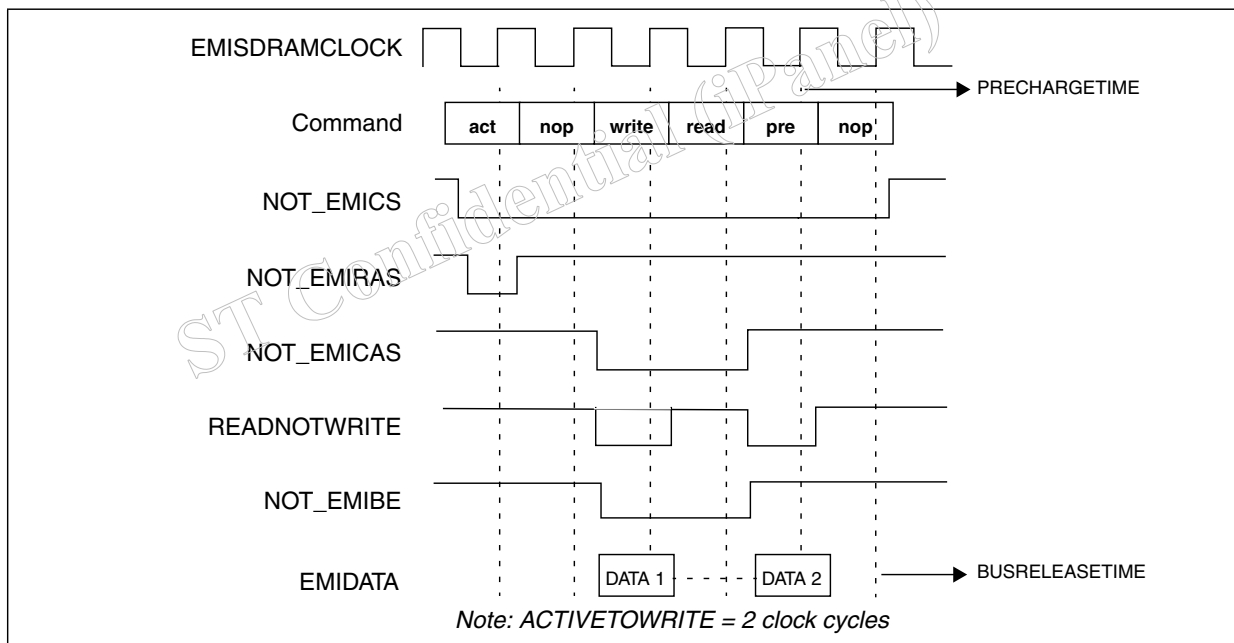


Figure 37: Pagemode READ after WRITE and CASLATENCY = 1 cycle



Confidential

Figure 38: Pagemode WRITE after READ and CASLATENCY = 1 cycle

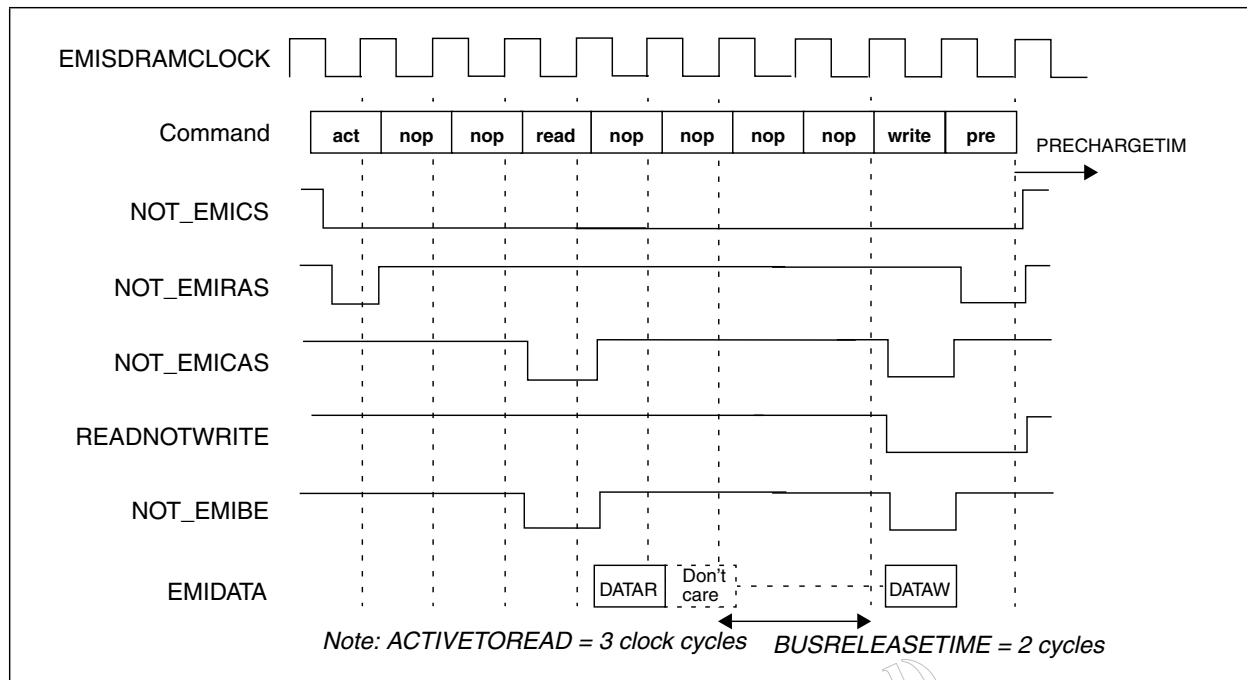


Figure 39: Single READ and CASLATENCY = 2 cycles

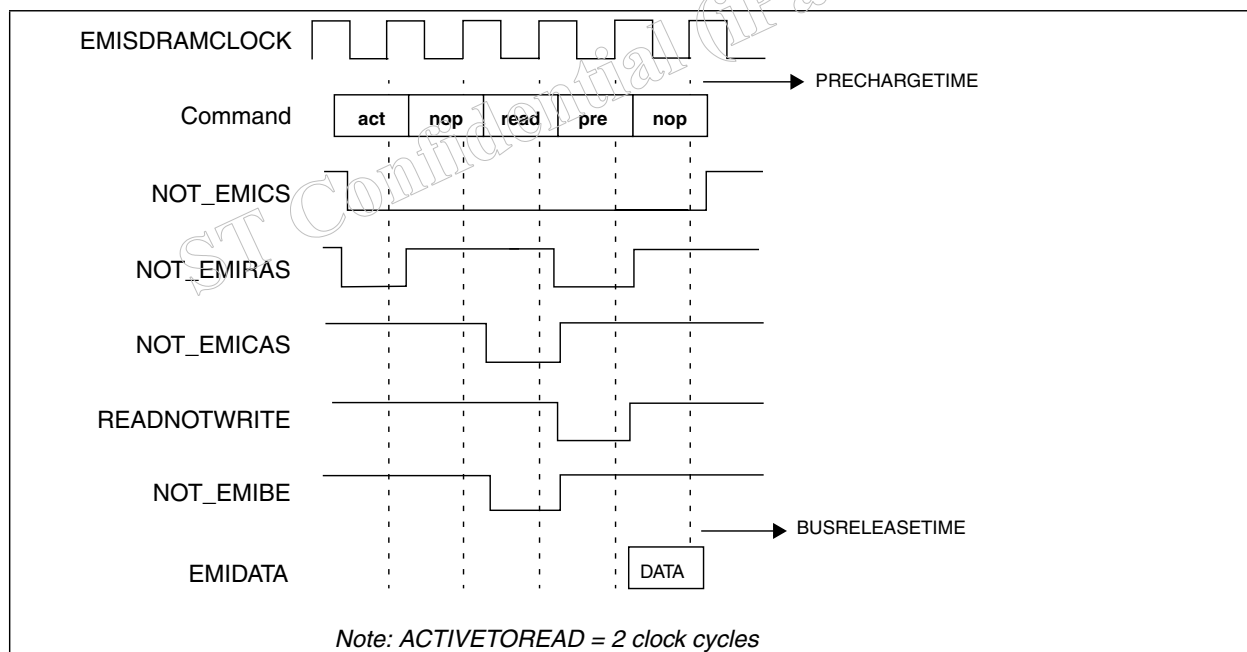


Figure 40: Pagemode READ and CASLATENCY = 2 cycles

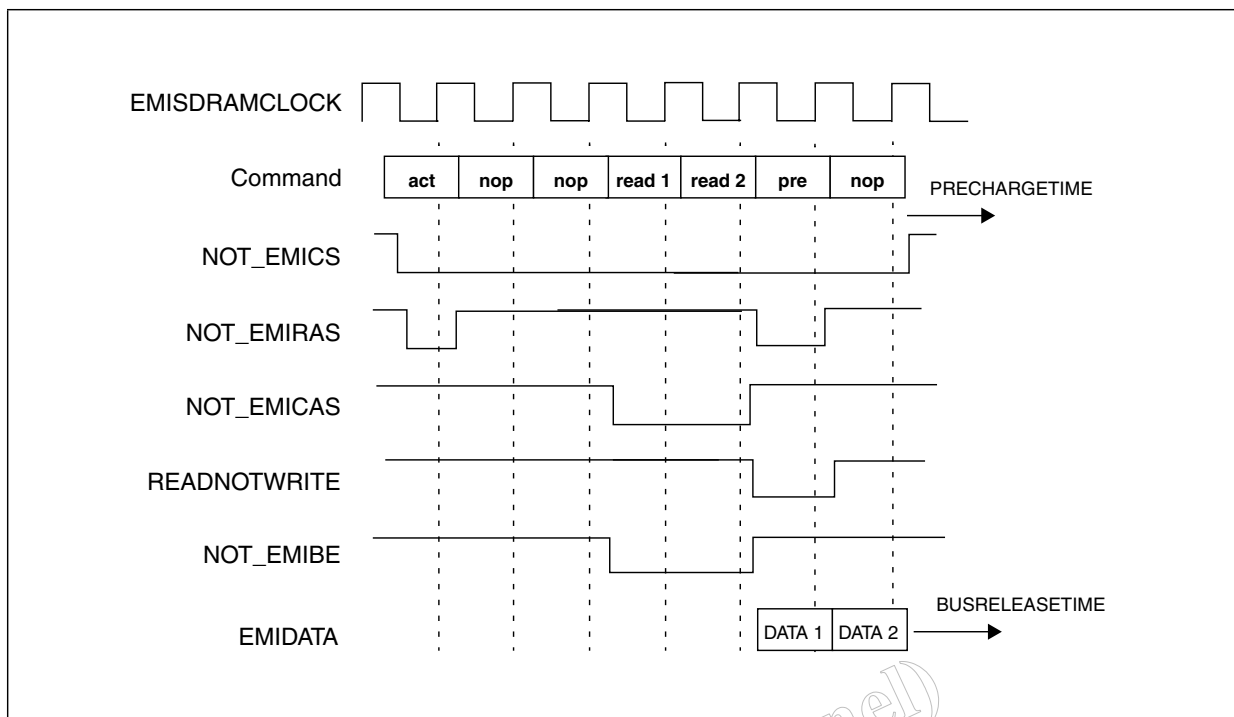
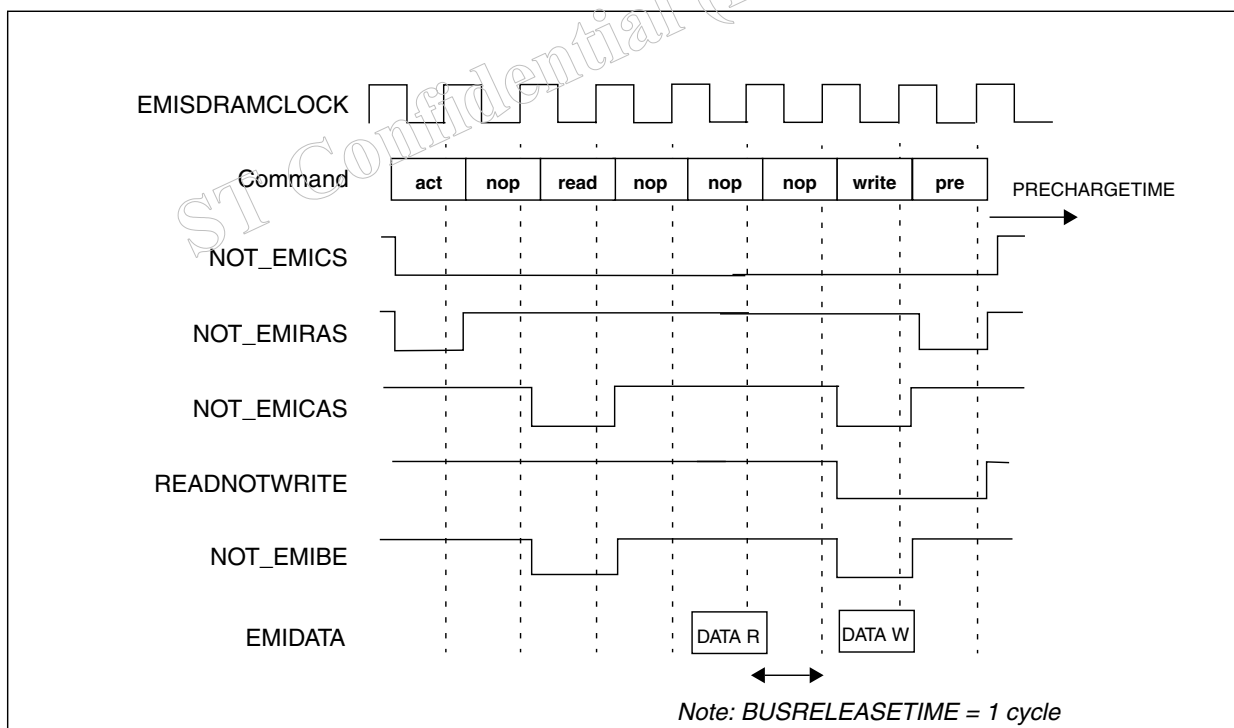


Figure 41: Pagemode WRITE after READ and CASLATENCY = 2 cycles



Confidential

Figure 42: Single READ and CASLATENCY = 3 cycles

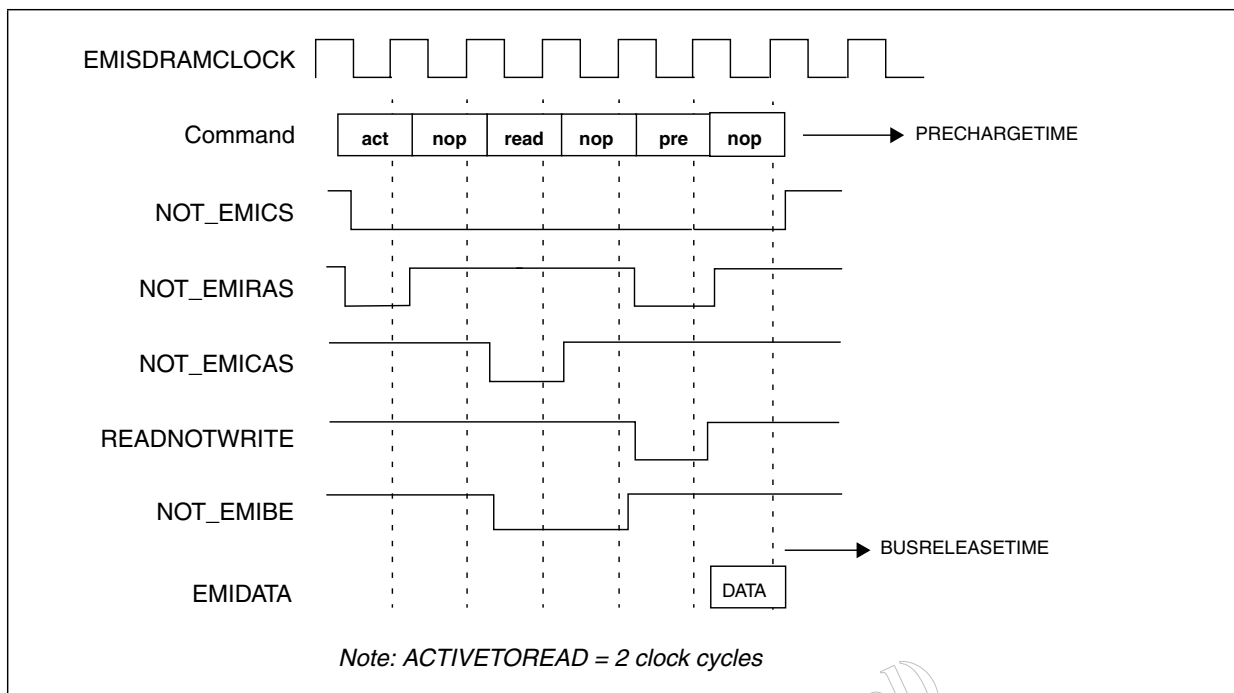


Figure 43: Pagemode READ and CASLATENCY = 3 cycles (pagehit during the nop)

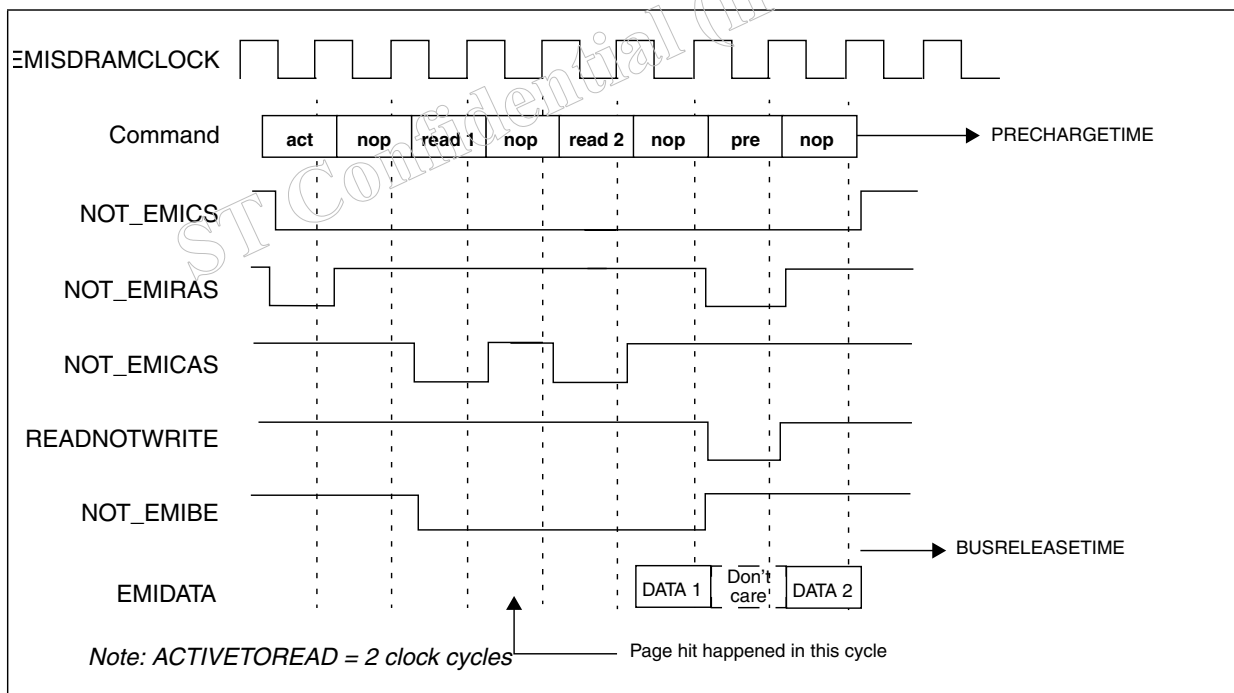
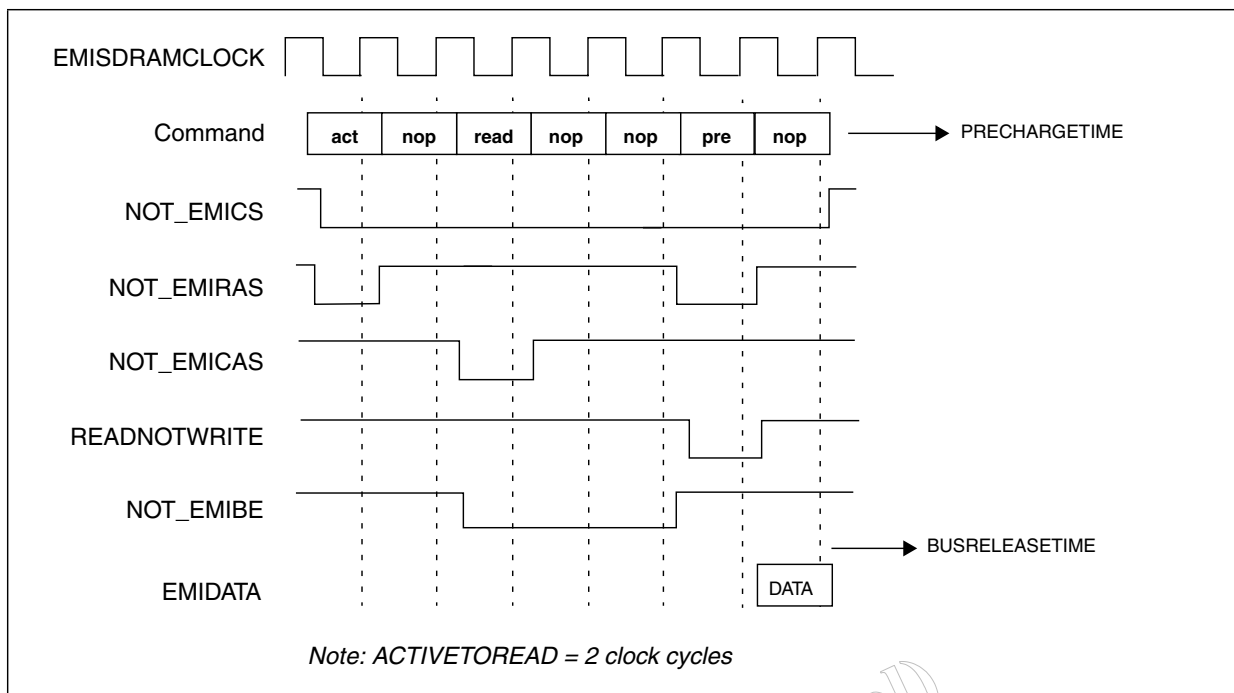


Figure 44: Single READ and CASLATENCY = 4 cycles



16.5.9 SDRAM EMI bank subdecoding and address selection

Each SDRAM bank may be subdecoded into two EMI subbanks using the SUBBANKS configuration parameter. The size and timing of each EMI subbank is identical. The size of each subbank is specified using the SUBBANKSIZE configuration parameter in EMI_CONFIGDATA0. The address bits used to select a NOT_EMICS[A:D] strobe depend on the number and size of the subbanks.

The SUBBANKSIZE is the total amount of memory cells contained in one physical SDRAM memory device, for example 64 Mbits or 128 Mbits. In Table 103 the address A means EMIADDRESS.

Table 103: EMI subbank decoding

Number of EMI subbanks	EMI subbank size	EMI subbank selection address	Strobe selection
2	16 Mbits	A ₂₁	0: NOT_EMICSA 1: NOT_EMICSB
	32 Mbits	A ₂₂	
	64 Mbits	A ₂₃	
	128 Mbits	A ₂₄	
	256 Mbits	A ₂₅	

CS[0:3] are internal EMI signals. These are mapped to the external signals CSA, CSB, CSC and CSD in the padlogic. See Section 18.2.10: Chip select output strobe generation on page 186.

An address line, connected to the SDRAM, is used as a precharge mode signal. It is always address pin 10 on the physical SDRAM and the address line from the EMI (the AP bit) is set to EMIADDRESS[12].

Table 104: Internal SDRAM bank selection address and BS mapping to EMIADDRESS pins

Number of DRAM banks ^a	SDRAM memory size	DRAM bank selection address ^b	BS	SDRAM port size	
				16-bit	8-bit
2	16 Mbit	A ₂₀	BS0	EMI_MEM_ADDR[17]	EMI_MEM_ADDR[18]
	32 Mbit	A ₂₁			
	64 Mbit	A ₂₂			
	128 Mbit	A ₂₃			
	256 Mbit	A ₂₄			
4	16 Mbit	A ₂₀ to A ₁₉	BS[1:0]	EMI_MEM_ADDR[18:17]	EMI_MEM_ADDR[19:18]
	32 Mbit	A ₂₁ to A ₂₀			
	64 Mbit	A ₂₂ to A ₂₁			
	128 Mbit	A ₂₃ to A ₂₂			
	256 Mbit	A ₂₄ to A ₂₃			

a. Physical SDRAM devices, number of internal DRAM banks

b. Internal EMI address

16.5.10 SDRAM refresh cycle

The SDRAM bank is periodically refreshed at intervals specified by the REFRESHINTERVAL configuration parameter. All subbanks are refreshed in the same access. After the last **refresh** command is issued, the EMI waits for REFRESHTIME + PRECHARGETIME cycles before starting a new access with an **activate** command.

Figure 45: Generic refresh access for SDRAM bank (case one SDRAM bank)

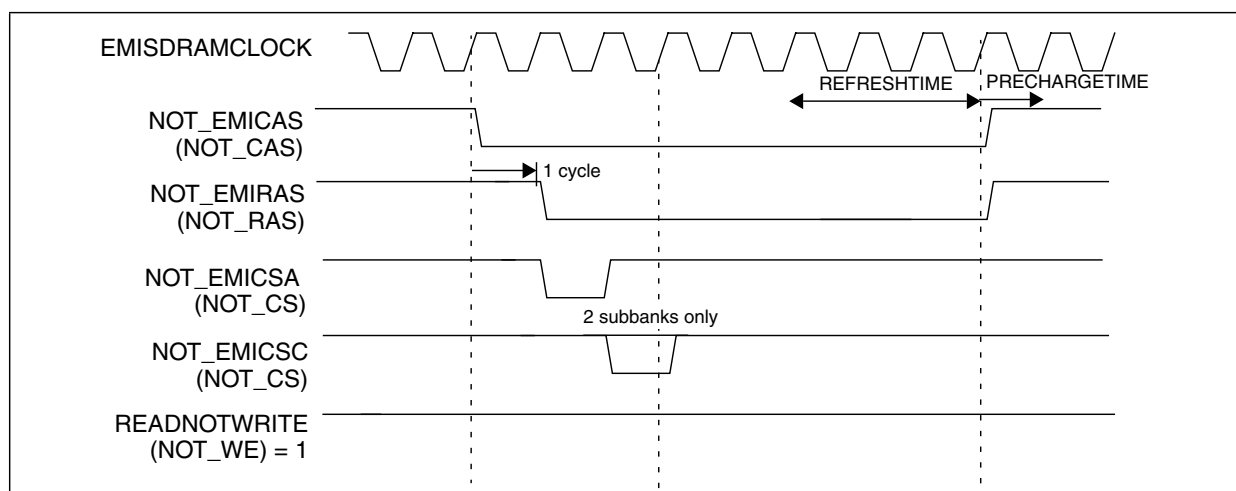
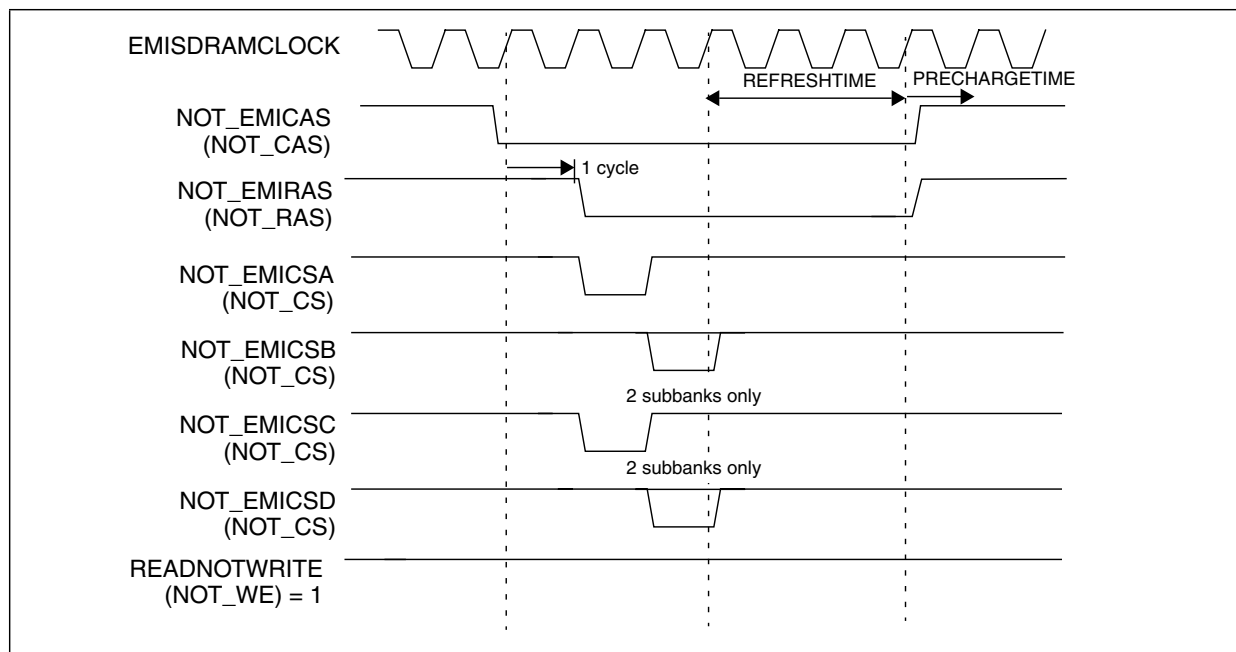


Figure 46: Generic refresh access for SDRAM bank (case two SDRAM banks)



The precharge time commences after the end of the refresh access. The SDRAM bank is precharged before the refresh access starts.

Table 105: Precharge and refresh times

Name	Programmable value	50 MHz	100 MHz
PRECHARGETIME	1 to 16 cycles	20 to 320 ns	10 to 160 ns
REFRESHINTERVAL	8 to 4096 cycles	160 ns to 82 μ s	80 ns to 41 μ s
REFRESHTIME	1 to 32 cycles	20 to 640 ns	10 to 320 ns

When two EMI banks are configured as SDRAM, the two devices could have different configuration values for REFRESHTIME. The EMI adopts the greatest configured refresh time from the two banks. A similar approach is adopted for precharge time.

16.5.11 Initialization

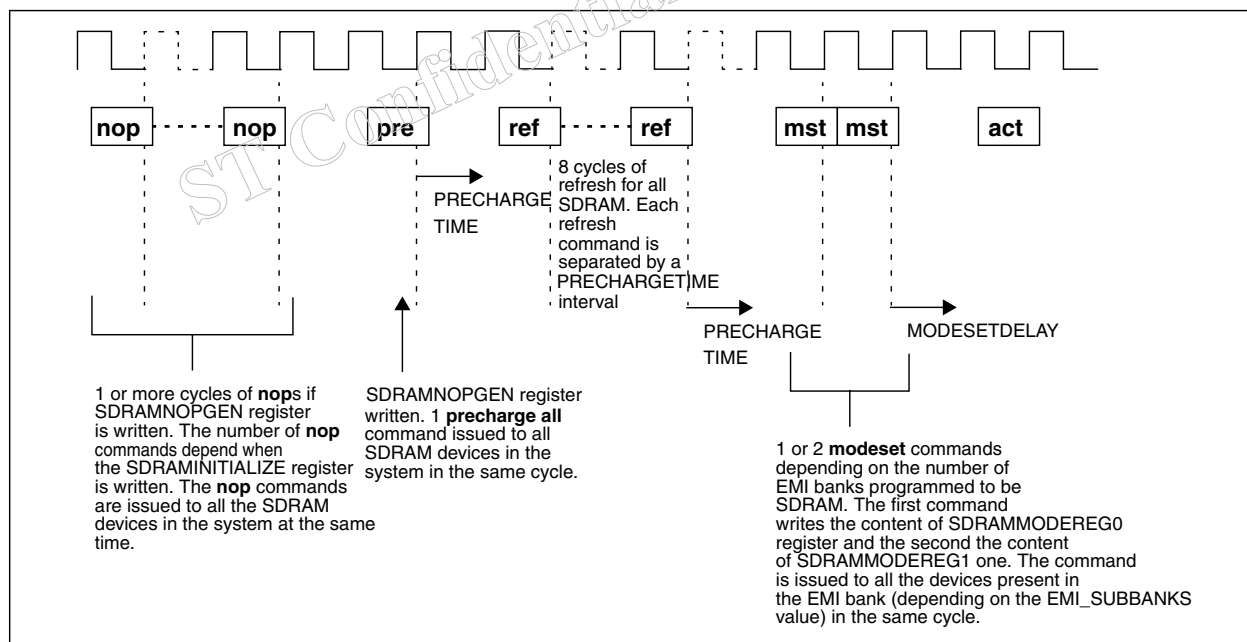
This must only be done after all other configuration registers have been programmed.

1. The JEDEC and PC100 standards recommends the application of **nop** input conditions for a minimum of 100 to 200 μs after stable power and stable clock. The EMI deals with this recommendation and provides a new register called SDRAMNOPGEN. Once it is written, it generates **nop** commands to all the SDRAM devices until the EMI_SDRAMINIT register is written. The system guarantees the maintenance of this condition for 100 to 200 μsec . To write to this register is optional. If it is written after EMI_SDRAMINIT, the write has no effect.

Note: Most SDRAMs have a power up to first precharge which is a minimum delay of 100 to 200 μs . This is a system requirement and the EMI_SDRAMINIT should not be written to until this time has elapsed.

2. The EMI_SDRAMINIT register is written to and an SDRAM bank is configured.
3. All SDRAM EMI banks are precharged using the **prechargeall** command. Eight refresh cycles are completed. Any SDRAM in the system is refreshed at the same time, as shown in Figure 47.
4. After the eight refresh cycles, the data written to the SDRAM mode registers is copied on to the bottom 16 address lines and a mode register set operation is executed (one or two depending upon the number of the EMI banks configured as SDRAM).
5. After the MODESETDELAY cycle (interval between setting the mode register and executing a **activate** command), the SDRAM is ready to accept an **activate** command. In case of two SDRAM banks, the MODESETDELAY used comes from BANK0.

Figure 47: EMI SDRAM initialization



17 External memory interface (EMI) registers

The EMI is allocated a 256 Mbyte region of the available memory map which is separated into two spaces. One being configuration space used to control the behavior of the EMI, the other a memory region which is mapped on to six user configurable EMI banks.

Each EMI_BANK[n] (n = [0:5]) contains a set of four 32-bit registers that are used to configure each bank depending on the type of device that is connected.

[Table 106](#) describes how the configuration region of each bank is divided. The address is offset from the EMI bank base address.

Table 106: EMI bank[n] bank configuration register formats

Register name	Description	Address offset	Type
EMI_CONFIGDATA0	Configuration data 0 register, see page 173	0x0000	RW
EMI_CONFIGDATA1	Configuration data 1 register, see page 174	0x0008	RW
EMI_CONFIGDATA2	Configuration data 2 register, see page 175	0x0010	RW
EMI_CONFIGDATA3	Configuration data 3 register, see page 176	0x0018	RW
Reserved		0x0020 to 0x0038	

The type and organization of each set of bank registers is dependent on the value in the DEVICETYPE bit (EMI_CONFIGDATA0) which defines the type of memory or device attached to that bank.

The EMI has two memory type configurations:

- peripheral interface,
- SDRAM interface.

The peripheral 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 EMI_CONFIGDATA0, EMI_CONFIGDATA1 and EMI_CONFIGDATA2. To configure for SFlash™, set bit 26 (WE_USE_OE_CONFIG) in EMI_CONFIGDATA0 to and set EMI_CONFIGDATA3.

Both memory types and their associated control registers are described in [Chapter 16: External memory interface \(EMI\)](#) on [page 141](#).

Addresses are provided as the *EMIBaseAddress* + offset.

The *EMIBaseAddress* is:

0x2020 0000.

A register summary is given in [Table 37: EMI registers](#) on [page 63](#).

EMI_STATUSCFG**EMI status configuration 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																								CONFIGURATION_UPDATED			

Address: *EMIBaseAddress* + 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.

EMI_STATUSLOCK**EMI status configuration lock 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																								CONFIGURATION_LOCKED			

Address: *EMIBaseAddress* + 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.

EMI_LOCK**EMI lock 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																								PROTECT			

Address: *EMIBaseAddress* + 0x0020

Type: Read/write

Reset: Undefined

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

Confidential

EMI_GENCFG**EMI general purpose 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
GENCFG[6:31]																REWAIT_RETIME				GENCFG[0:3]											

Address: *EMIBaseAddress* + 0x0028

Type: Read/write

Reset: Undefined

Description: Used to propagate general purpose outputs. If bit [n] is set, then the general purpose output GENCFG[n] is set. See EMI_GENCFG in [Chapter 19: Padlogic registers](#) for more details about this register.

The WAIT_RETIME bits set the number of retime stages (from 0 to 2) for the EMIWAITNOTTREADY signal. The signal can be dynamically changed between a single retime input stage or a double retime input stage. Both retime stages on the retimed input are synchronized to the current clock.

If EMIWAITNOTTREADY is set at the beginning of the access, $ACCESSTIMERREAD > LATCHPOINT + (2 + WAIT\_RETIME)$.

EMI_SDRAMNOPGEN**EMI nop generate 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																NOPGEN															

Address: *EMIBaseAddress* + 0x0030

Type: Write only

Reset: Undefined

Description: When an SDRAM is in the system it generates **nop** commands during the initialization phase until a SDRAM initialize is issued.

EMI_SDRAMMODEREG**EMI SDRAM mode 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
REG1																REG0															

Address: *EMIBaseAddress* + 0x0038

Type: Write only

Reset: Undefined

Description:

[31:16] **REG1**: SDRAM mode register for the SDRAM EMI bank 1
If only one SDRAM bank is defined, this register is reserved.

[15:0] **REG0**: SDRAM mode register for the SDRAM EMI bank 0
If only one SDRAM bank is defined, it is assumed to be EMI bank 0.

EMI_SDRAMINIT**EMI SDRAM initialize 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																SDRAMINIT	

Address: *EMIBaseAddress* + 0x0040

Type: Write only

Reset: Undefined

Description: Initialize any SDRAM in the system. This bit to be set after the setting of SDRAMMODEREG.

EMI_REFRESHINIT**EMI refresh interval setting 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																REFRESH_INTERVAL	

Address: *EMIBaseAddress* + 0x0048

Type: Write only

Reset: Undefined

Description: This register defines the interval between successive refreshes in clock cycles. Valid values are in the range 0x007 to 0xFFF corresponding to an interval of between 8 and 4096 cycles. Values outside this range may lead to undefined behavior.

EMI_FLASHCLKSEL**EMI flash burst clock select 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																FLASH_CLOCK_SELECT	

Address: *EMIBaseAddress* + 0x0050

Type: Write only

Reset: Undefined

Description:

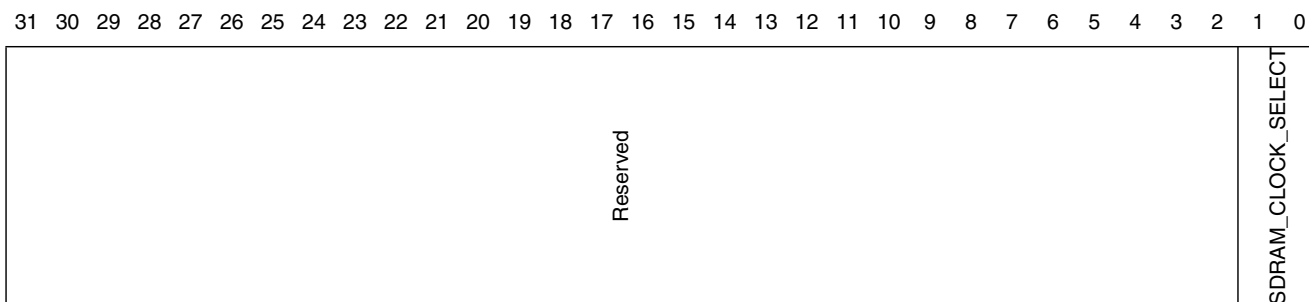
[31:2] **Reserved**[1:0] **FLASH_CLOCK_SELECT**: Set clock ratio for burst flash clock

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

EMI_SDRAMCLKSEL**EMI SDRAM clock select register**Address: *EMIBaseAddress* + 0x0058

Type: Write only

Reset: 10

Description:

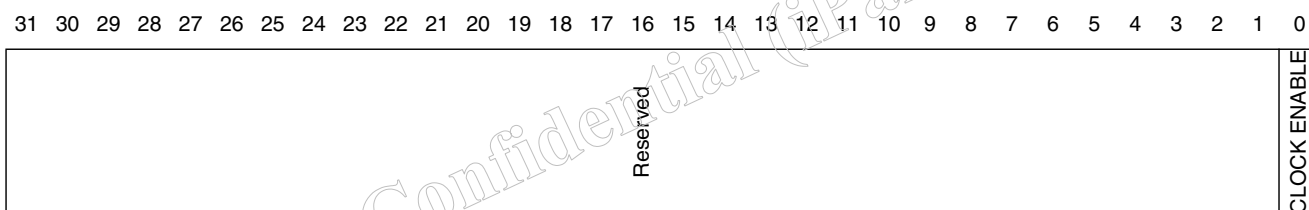
[31:2] **Reserved**[1:0] **SDRAM_CLOCK_SELECT**: Set clock ratio for SDRAM clock

00: 1:1 SDRAM operates at STBus clock

01: 1:2 SDRAM operates at 1/2 of STBus clock

10: 1:3 SDRAM operates at 1/3 of STBus clock

11: Reserved

EMI_CLKENABLE**EMI clock enable register**Address: *EMIBaseAddress* + 0x0068

Type: Write only

Reset: 00

Description:

[31:1] **Reserved**[0] **CLOCK_ENABLE**: Update flash and SDRAM clocks

1: Flash and SDRAM clocks are updated

This operation should only occur once. Further writes to this register lead to undefined behavior.

Confidential

17.1 Configuration register format for peripherals

The following are the configuration registers formats for peripherals. Any bit in the configuration registers which is defined as reserved should be set to 0.

EMI_CONFIGDATA0 EMI configuration data 0 register (peripheral 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					WE_USE_OE_CONFIG		WAITPOLARITY		LATCHPOINT				DATADRIVEDELAY				BUSRELEASETIME				CSACTIVE		OEACTIVE		BEACTIVE		PORTSIZE		DEVICETYPE		

Address: *EMIBaseAddress* + 0x100 (Bank 0), 0x140 (Bank 1), 0x180 (Bank 2), 0x1C0 (Bank 3), 0x200 (Bank 4), 0x240 (Bank 5)

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 STMicroelectronics M58LW064A/B). It requires a configurable READNOTWRITE signal for asynch 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

00010: 2 clock cycles before end of access cycle

00100: 4 cycles before end of access cycle

00110: 6 cycles before end of access cycle

01000: 8 cycles before end of access cycle

01010: 10 cycles before end of access cycle

01100: 12 cycles before end of access cycle

01110: 14 cycles before end of access cycle

10000: 16 cycles before end of access cycle

00001: 1 STBus clock cycle before end of access

00011: 3 cycles before end of access cycle

00101: 5 cycles before end of access cycle

00111: 7 cycles before end of access cycle

01001: 9 cycles before end of access cycle

01011: 11 cycles before end of access cycle

01101: 13 cycles before end of access cycle

01111: 15 cycles before end of access cycle

Other: Reserved

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

[14:11] **BUSRELEASETIME**: 0 to 15 cycles

[10:5] **CSACTIVE, OEACTIONE, BEACTIVE**: See in [Chapter 16: External memory interface \(EMI\)](#)
[Table 96: Active code settings](#)

[4:3] **PORTSIZE**

00: Reserved

10: 16-bit

01: Reserved

11: 8-bit

[2:0] **DEVICETYPE**

001: Normal peripheral

Other: Reserved

100: Burst flash

EMI_CONFIGDATA1**EMI configuration data 1 register (peripheral 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
CYCLENOTPHASEREAD	ACCESSTIMERREAD							CSE1TIMERREAD				CSE2TIMERREAD				OEE1TIMERREAD				OEE2TIMERREAD				BEE1TIMERREAD				BEE2TIMERREAD			

Address: *EMIBaseAddress* + 0x108 (Bank 0), 0x148 (Bank 1), 0x188 (Bank 2), 0x1C8 (Bank 3), 0x208 (Bank 4), 0x248 (Bank 5)

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] **ACCESSTIMERREAD**

2 to 127 STBus clock cycles: Value 0 and 1 are reserved

[23:20] **CSE1TIMERREAD**

Falling edge of CS: 0 to 15 phases/cycles after start of access cycle

[19:16] **CSE2TIMERREAD**

Rising edge of CS: 0 to 15 phases/cycles before end of access cycle

[15:12] **OEE1TIMERREAD**

Falling edge of OE: 0 to 15 phases/cycles after start of access cycle

[11:8] **OEE2TIMERREAD**

Rising edge of OE: 0 to 15 phases/cycles before end of access cycle

[7:4] **BEE1TIMERREAD**

Falling edge of BE: 0 to 15 phases/cycles after start of access cycle

[3:0] **BEE2TIMERREAD**

Rising edge of BE: 0 to 15 phases/cycles before end of access cycle

EMI_CONFIGDATA2

EMI configuration data 2 register (peripheral 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
CYCLENOTPHASEWRITE	ACCESSTIMEWRITE							CSE1TIMEWRITE				CSE2TIMEWRITE				OEE1TIMEWRITE				OEE2TIMEWRITE				BEE1TIMEWRITE				BEE2TIMEWRITE			

Address: *EMIBaseAddress* + 0x110 (Bank 0), 0x150 (Bank 1), 0x190 (Bank 2), 0x1D0 (Bank 3), 0x210 (Bank 4), 0x250 (Bank 5)

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 (EMI_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 (EMI_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

EMI_CONFIGDATA3

EMI configuration data 3 register (peripheral 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					STROBEONFALLING	Reserved															BURST_SIZE			DATA LATENCY			DATAHOLDDDELAY	BURSTMODE			

Address: *EMIBaseAddress* + 0x118 (Bank 0), 0x158 (Bank 1), 0x198 (Bank 2), 0x1D8 (Bank 3), 0x218 (Bank 4), 0x258 (Bank 5)

Type: Read/write

Reset: 0

Description:

[31:27] **Reserved**

[26] **STROBEONFALLING^a**

0: Strobes for burst generated on rising edge of the flash clock

1: Strobes 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^b

110: 128

111: Reserved

Only valid in burst mode

[6:2] **DATA LATENCY**

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] **DATAHOLDDDELAY**

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 DATAHOLDDDELAY are relevant for strobe generation timing during read operations

- Configuration of EMI_CONFIGDATA0, EMI_CONFIGDATA1, EMI_CONFIGDATA2 relates only to the asynchronous behavior (normal peripheral and normal asynchronous behavior of flash). These registers must be programmed in terms of STBus clock cycle. EMI_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.
- 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 EMI, 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.

EMI_CONFIGDATA3 does not need to be configured for a normal asynchronous peripheral.

The strobe on falling feature of EMI means only that strobes, data and address are generated on the falling edge of the SFlash™ clock. This does not imply that the same signals are sampled on the falling edge by the memories. The EMI assumes memory always samples on the rising edge anyway. The strobes on falling feature has been implemented only to possibly extend the hold time of half a cycle to help padlogic implementation.

17.2 Configuration register format for SDRAM

The following are the configuration registers formats for SDRAM. Any bit in the configuration registers which is defined as reserved should be set to 0.

EMI_CONFIGDATA0 EMI configuration data 0 register (SDRAM 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					STROBEONFALLING	Reserved											BUSRELEASETIME	EMI_SUBBANKS	SUBBANKSIZE	SHIFTAMOUNT	PORTSIZE	DEVICETYPE									

Address: *EMI*BaseAddress + 0x100 (Bank 0), 0x140 (Bank 1), 0x180 (Bank 2), 0x1C0 (Bank 3), 0x200 (Bank 4), 0x240 (Bank 5)

Type: Read/write

Reset: 0

Description:

[31:27] **Reserved**

[26] **STROBEONFALLING**

0: Strokes/Data/Address for SDRAM generated on rising edge of the clock (recommended mode)
1: Strokes/Data/Address for SDRAM generated on falling edge of the clock

[25:15] **Reserved**

[14:13] **BUSRELEASETIME**

1 to 4 SDRAM clock cycles

[12:11] **EMI_SUBBANKS**

00: 1, that is, bank is not subdecoded

01: 2

10, 11: Reserved

[10:8] **SUBBANKSIZE**

The physical size of the SDRAM device used in each EMI bank or subbank

Even if the subbank is selected in EMI_SUBBANKS (bits 12:11), the subbank size must be set to match the physical size of the SDRAM device used on this bank.

000: 16 Mbit

001: 32 Mbit

010: 64 Mbit

011: 128 Mbit

100: 256 Mbit

101, 110, 111: Reserved

[7:5] **SHIFTAMOUNT**

Column address width

0: 7

1: 8

...

[4:3] **PORTSIZE**

00: Reserved

01: Reserved

10: 16-bit

11: 8-bit

[2:0] **DEVICETYPE**

Sets the format of the configuration register

010: SDRAM

EMI_CONFIGDATA1**EMI configuration data 1 register (SDRAM 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	RASBITS

Address: *EMIBaseAddress* + 0x108 (Bank 0), 0x148 (Bank 1), 0x188 (Bank 2), 0x1C8 (Bank 3), 0x208 (Bank 4), 0x248 (Bank 5)

Type: Read/write

Reset: 0

Description:

[31:23] **Reserved**

[22:0] **RASBITS**: Page address mask for address bits [29:7]

For a 16-bit data bus port width, memory address bits [6:1] of the internal memory address are assumed to be 0 by default. For example for a 16-bit data bus (8-bit SDRAM column size) set RASBITS[1:0] to 0 and RASBITS[22:2] to 1.

EMI_CONFIGDATA2**EMI configuration data 2 register (SDRAM 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	PRECHARGE TIME	Reserved

Address: *EMIBaseAddress* + 0x110 (Bank 0), 0x150 (Bank 1), 0x190 (Bank 2), 0x1D0 (Bank 3), 0x210 (Bank 4), 0x250 (Bank 5)

Type: Read/write

Reset: 0

Description:

[31:8] **Reserved**

[7:4] **PRECHARGE TIME**

1 to 16 SDRAM clock cycles

[3:0] **Reserved**

Confidential

EMI_CONFIGDATA3

EMI configuration data 3 register (SDRAM 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												MODESETDELAY		REFRESHTIME				ACTIVATETOREAD				ACTIVATETOWRITE				CASLATENCY				DRAM_BANKS		WRITERECOVERYTIME	

Address: *EMIBaseAddress* + 0x118 (Bank 0), 0x158 (Bank 1), 0x198 (Bank 2), 0x1D8 (Bank 3), 0x218 (Bank 4), 0x258 (Bank 5)

Type: Read/write

Reset: 0

Description:

[31:20] **Reserved**

[19:18] **MODESETDELAY**

3 to 6 SDRAM clock cycles

[17:13] **REFRESHTIME**

1 to 32 SDRAM clock cycles

[12:10] **ACTIVATETOREAD**

1 to 8 SDRAM clock cycles

[9:7] **ACTIVATETOWRITE**

1 or 8 SDRAM clock cycles

[6:4] **CASLATENCY**

1 to 8 SDRAM clock cycles

[3] **DRAM_BANKS**

Internal DRAM arrays for a physical SDRAM device

0: 2 DRAM banks, 1: 4 DRAM banks

Set to 0 for a 16-Mbit SDRAM device with two internal arrays and set to 1 for a 64-Mbit or 128-Mbit SDRAM device with four internal arrays.

[2:0] **WRITERECOVERYTIME**

000: Reserved

....

001: 1 SDRAM clock cycle

111: 7 cycles

Confidential

18 Padlogic

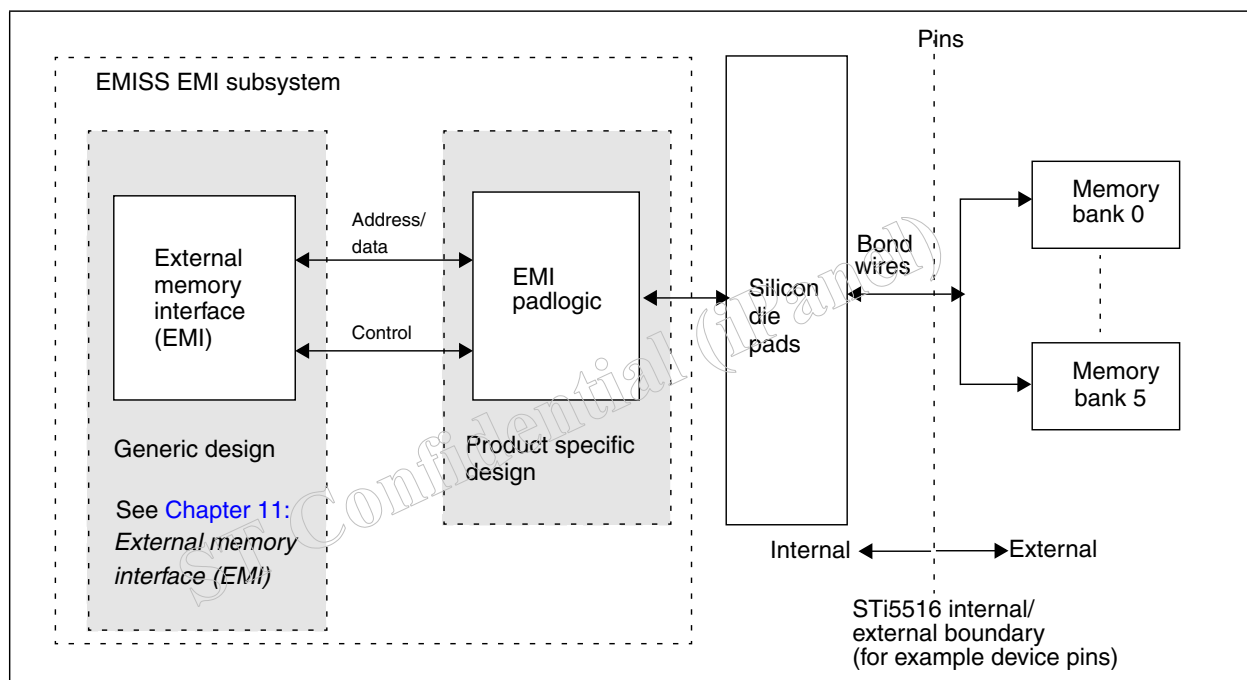
18.1 Overview

This chapter describes:

- the STi5516 specific EMI padlogic that lies between the generic EMI and the device pins,
- general nonEMI padlogic including system configuration registers.

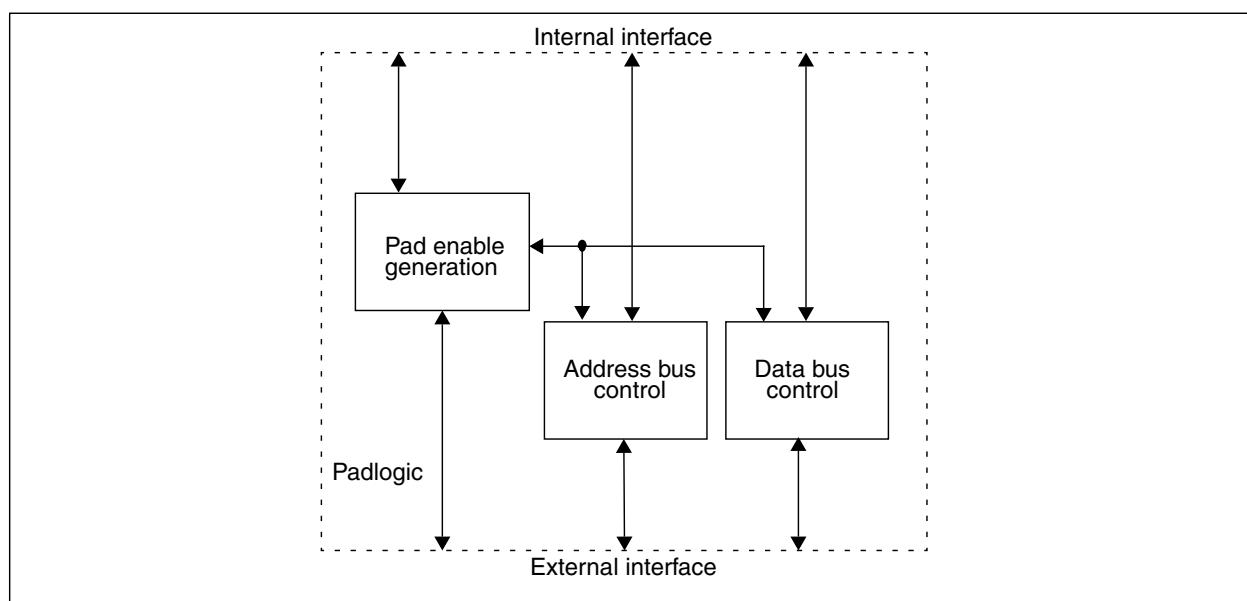
18.2 EMI padlogic

Figure 48: Generic EMI and EMI padlogic architecture



The EMI padlogic is detailed in Figure 49.

Figure 49: EMI padlogic architecture



18.2.1 Features

- **Pad enable generation**

Generates the necessary internal pad driver signals for the address and data buses. Where necessary, it combines the two phases of output signals, generated internally by the generic EMI, into a single output signal for each of the general memory interface control signals. It also provides tri-state capability for bi-directional signals.

- **Address bus control**

Combines the two phases of address bus, generated internally by the generic EMI, into a single address bus output.

- **Data bus control**

Combines the two phases of the generic EMI write data bus into a single data bus, before combining the separate internal read/write data buses into a single output bus.

- **Programmable drive strength pads**

It is possible to select the drive strength of the EMI pins to match board loading characteristics.

Note: The generic EMI chapter does not explicitly show the two phase EMI signals. These can be seen in [Figure 50: EMI and EMI padlogic interfaces on page 182](#). The generic EMI generates two phase signals so that address, data and control strobes can be made to change state on the rising or falling edges of the STBus clock, thus achieving better resolution with slow clock frequencies.

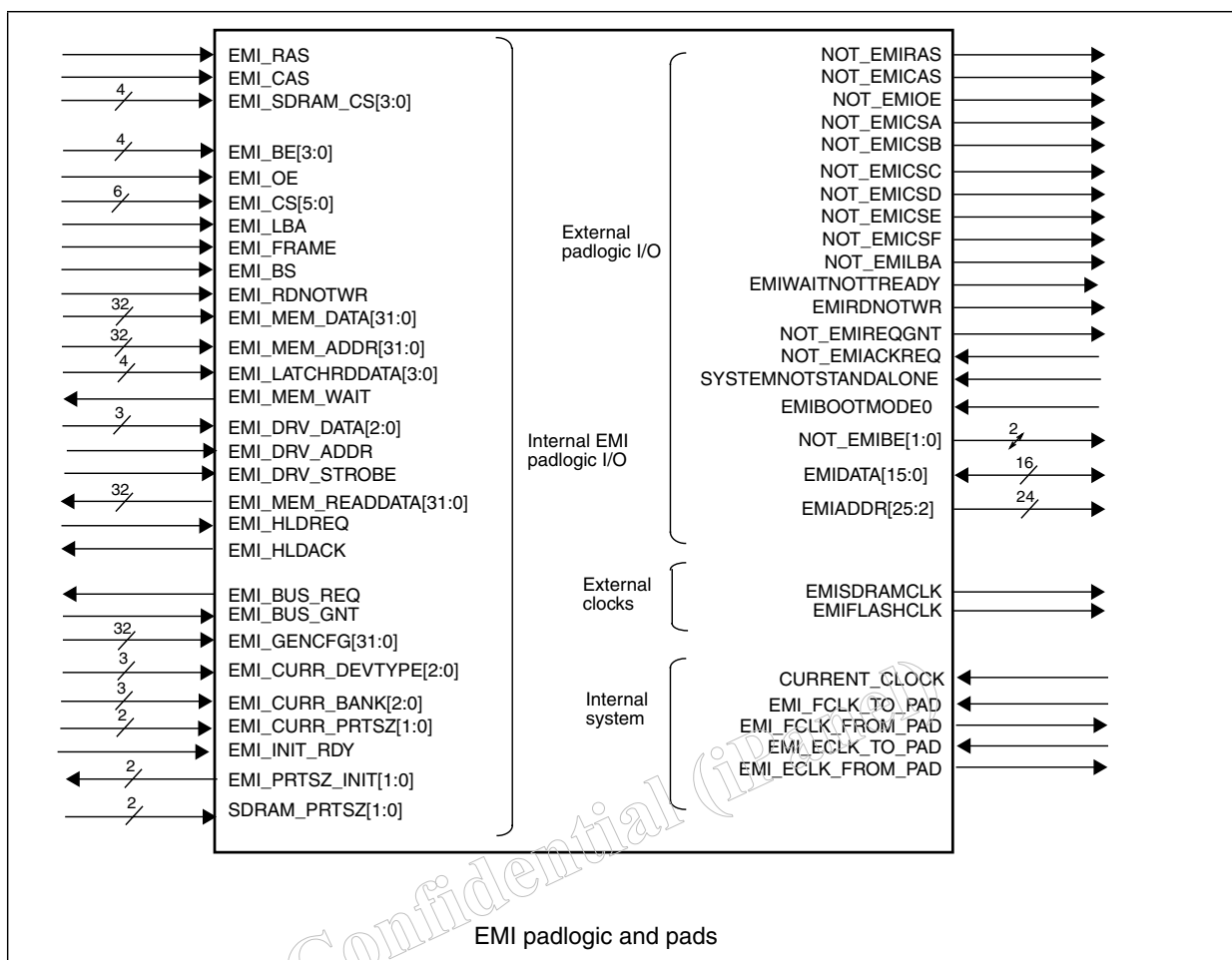
For each strobe, the EMI outputs two signals to the pad, one for each phase. At the pad, these two signals are latched and multiplexed with the current clock (for example, clock ruling current access) by the EMI padlogic, to provide signals which can change on both edges of the clock. The PHI1/PHI2 strobe pair is only useful for peripheral type accesses when the EMI is running at a low speed (50 MHz for example). In this case there is a gain of half a cycle (10 ns) due to strobe generation on falling edge of the clock. If the EMI runs at 100 MHz, a 10 ns time window is equal to a clock cycle, so strobe generation on rising edges would guarantee the same timing resolution due to the improved clock speed.

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

EMI padlogic interfaces

In [Figure 50](#), all internal interfaces to the EMI are shown on the left and interfaces to the external pins on the right.

Figure 50: EMI and EMI padlogic interfaces



EMI_GENCFG register

The EMI_GENCFG register is part of the generic EMI register set. It is a general purpose configuration register which provides 32 signals out of the EMI. These signals are then input to the EMI padlogic as EMI_GENCFG[31:0]. The outputs are used by the EMI padlogic to control such features as chip select type, address bus shifting for different bank data port widths, input signal retiming and tri-state control of control signals. The use of this register is summarized in [Chapter 19: Padlogic registers on page 193](#).

18.2.2 Strobe generation

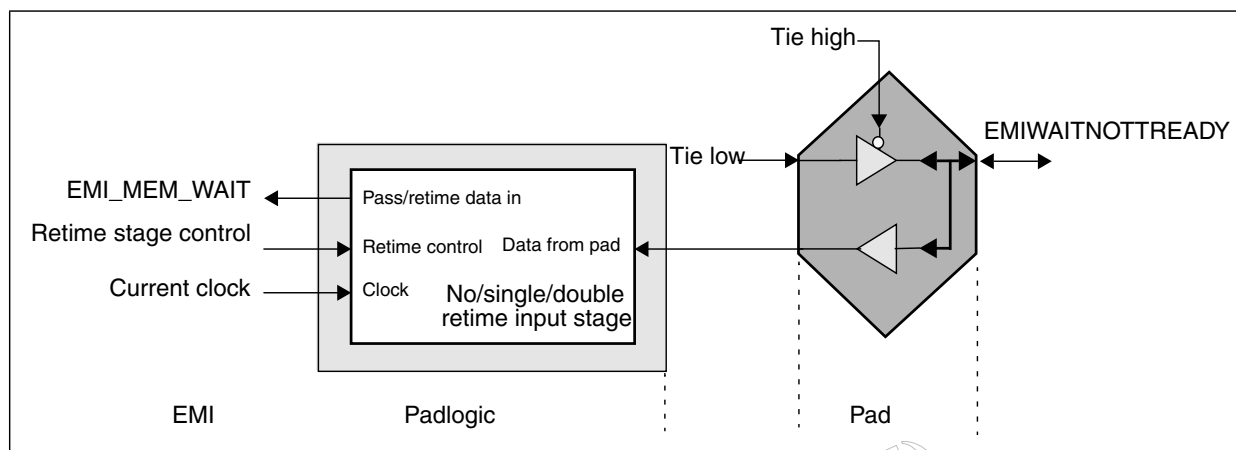
The strobe generator retimes the selected signal with the current clock, and so generates an output that can change state on a rising or falling edge of the current clock. The relationship of the strobes is programmed inside the EMI (described in [Chapter 16: External memory interface \(EMI\)](#)), to 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 EMI 100 MHz subsystem clock).

At 100 MHz the extra resolution is not generally needed, and for synchronous interfaces, such as SDRAM, the EMI does not support negative edge transition.

18.2.3 EMIWAITNOTTREADY input strobe

EMIWAITNOTTREADY may be passed to the generic EMI 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. EMIWAITNOTTREADY uses the internal EMI control signal EMI_MEM_WAIT.

Figure 51: Synchronous input strobe padlogic connections



When accessing a device, the number of retime stages is programmable through EMI_GENCFG[5:4] as shown in [Table 107](#).

Table 107: EMIWAITNOTTREADY retime stage control mapping

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

The boot indicator (EMI_GENCFG[20]), an EMI 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.

18.2.4 EMIBOOTMODE0 input strobe

EMIBOOTMODE0 is passed directly to the EMI, without going through a retime element.

18.2.5 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 EMI and are then passed to the padlogic. If necessary they are also inverted.

Figure 52: Output strobe padlogic connections

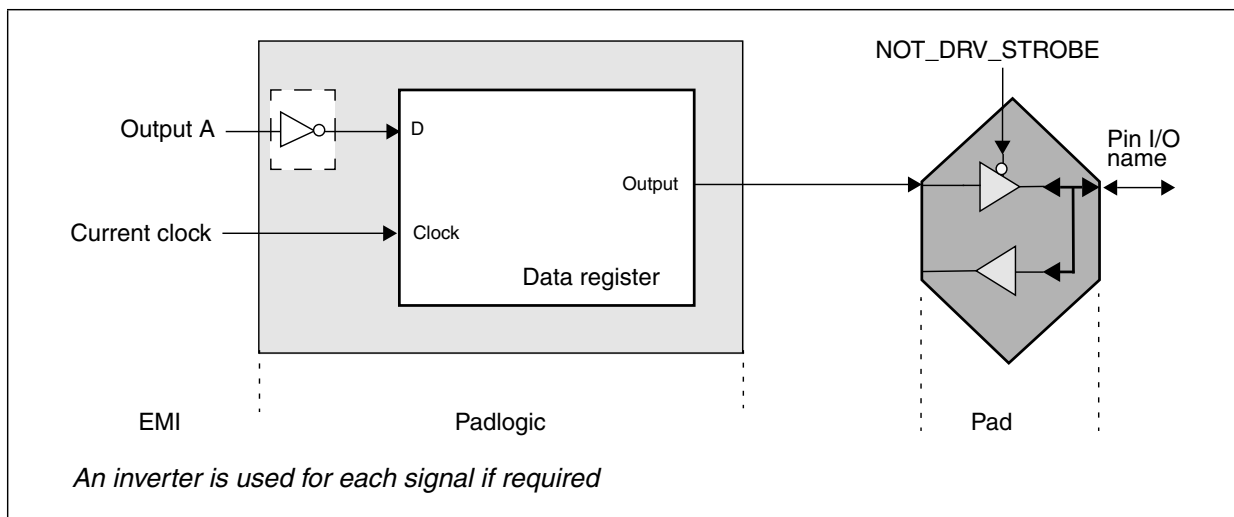


Table 108 details the signal connection for the internal EMI control signals to the external device pins.

Table 108: Output signal to pin connections

Generic EMI output signal name	Pin I/O name
EMI_RAS	NOT_EMIRAS
EMI_CAS	NOT_EMICAS
EMI_BE[0]	NOT_EMIBE[0]
EMI_BE[1]	NOT_EMIBE[1]
EMI_OE	NOT_EMIOE
EMI_LBA	NOT_EMILBA
EMI_RDNOTWR	EMIRDNOTWR

18.2.6 Bus control input strobe

The bus acknowledge/request input pad drives the EMI bus request input (thereby indicating that a slave device has requested access to the bus).

18.2.7 DVB-CI support on EMI bank 3

When bit `CONFIG_EMISS_DVB_ENABLE` of register `CONFIG_CONTROL_E` = 1, DVB-CI support is enabled for bank 3. When DVB-CI is enabled the following mapping is applied to the strobes:

- NOT_EMIRAS: NOT_CI_IORD,
- NOT_EMICAS: NOT_CI_IOWR,
- NOT_EMILBA: NOT_CI_OE,
- EMIRDNOTWR: NOT_CI_WE.

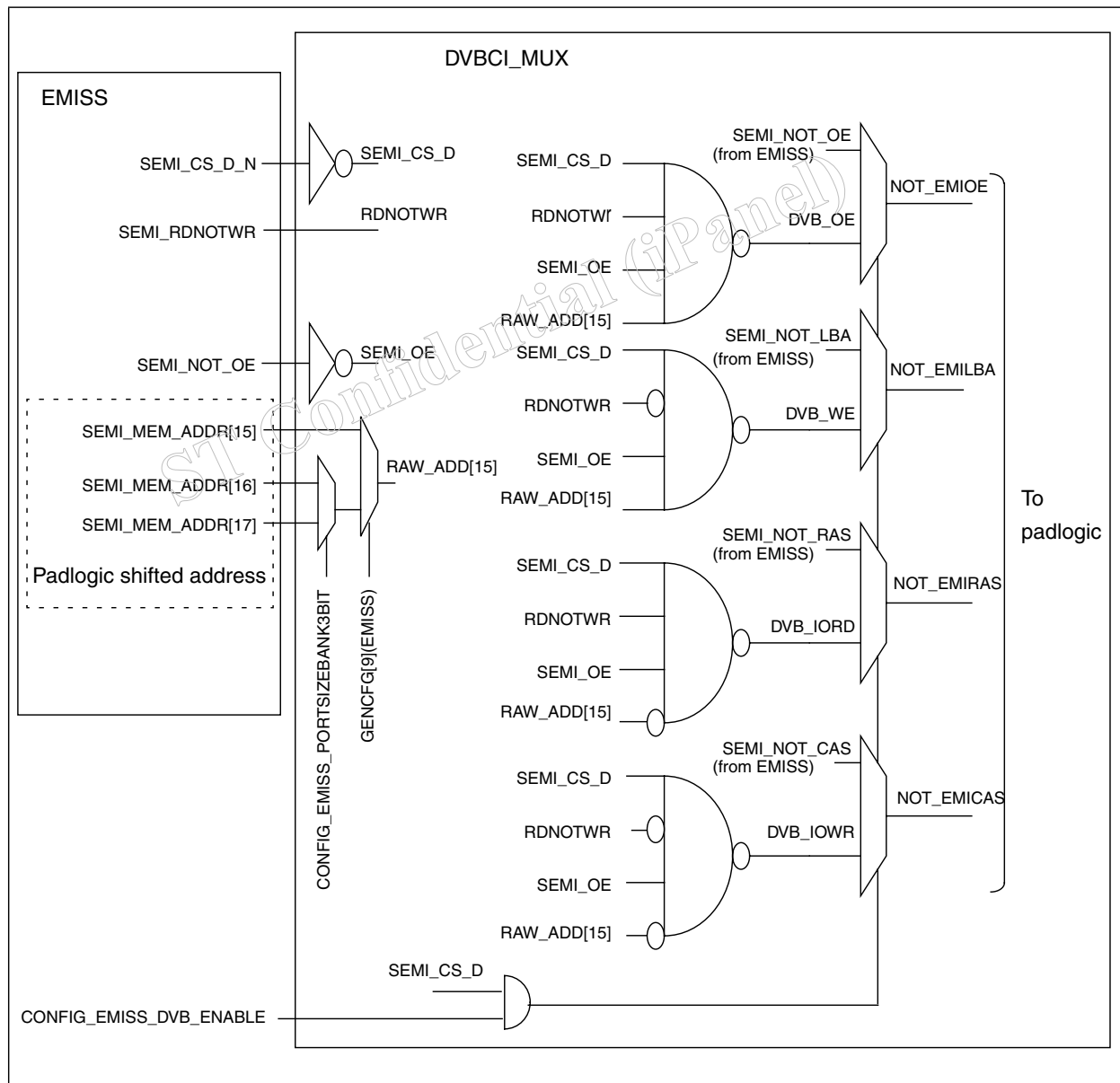
Note: Bit `CONFIG_EMISS_PORTSIZEBANK3BIT` of register `CONFIG_CONTROL_E` must be set to match the port size of the peripheral connected.

Also the four read/write signals: DVB_OE[n], DVB_WE[n], DVB_IORD[n], DVB_IOWR[n] are generated using the following Boolean equations:

- $DVB\_IORD[n] = \text{NOT}(\text{SEMI\_CS\_D\_N} \text{ and } \text{SEMI\_RDNOTWR} \text{ and } \text{SEMI\_NOT\_OE} \text{ and } \text{NOT}(\text{SEMI\_MEM\_ADDR}[15]))$,
- $DVB\_IOWR[n] = \text{NOT}(\text{SEMI\_CS\_D\_N} \text{ and } \text{NOT}(\text{SEMI\_RDNOTWR}) \text{ and } \text{SEMI\_NOT\_OE} \text{ and } \text{NOT}(\text{SEMI\_MEM\_ADDR}[15]))$,
- $DVB\_OE[n] = \text{NOT}(\text{SEMI\_CS\_D\_N} \text{ and } \text{SEMI\_RDNOTWR} \text{ and } \text{SEMI\_NOT\_OE} \text{ and } \text{SEMI\_MEM\_ADDR}[15])$,
- $DVB\_WE[n] = \text{NOT}(\text{SEMI\_CS\_D\_N} \text{ and } \text{NOT}(\text{SEMI\_RDNOTWR}) \text{ and } \text{SEMI\_NOT\_OE} \text{ and } \text{SEMI\_MEM\_ADDR}[15])$

See [Section 18.2.5: Output strobe generation on page 183](#) for details about the EMI control signal connections.

Figure 53: DVB-CI connections



Confidential

18.2.8 HDDI

HDDI ATA mode uses NOT_CI_IORD and NOT_CI_IOWR as the ATA IORD and IOWR signals. EMI_MEMADDR[19:20] are used as ATA CS0 and CS1 respectively. To use HDDI interface mode put the device into DVD-CI mode (set CONFIG_CONTROL_E, bit 17 to 1).

18.2.9 Bus control output strobe generation

The bus grant output pad is driven by the EMI bus grant output (thereby indicating to a slave device that its data bus request has been accepted).

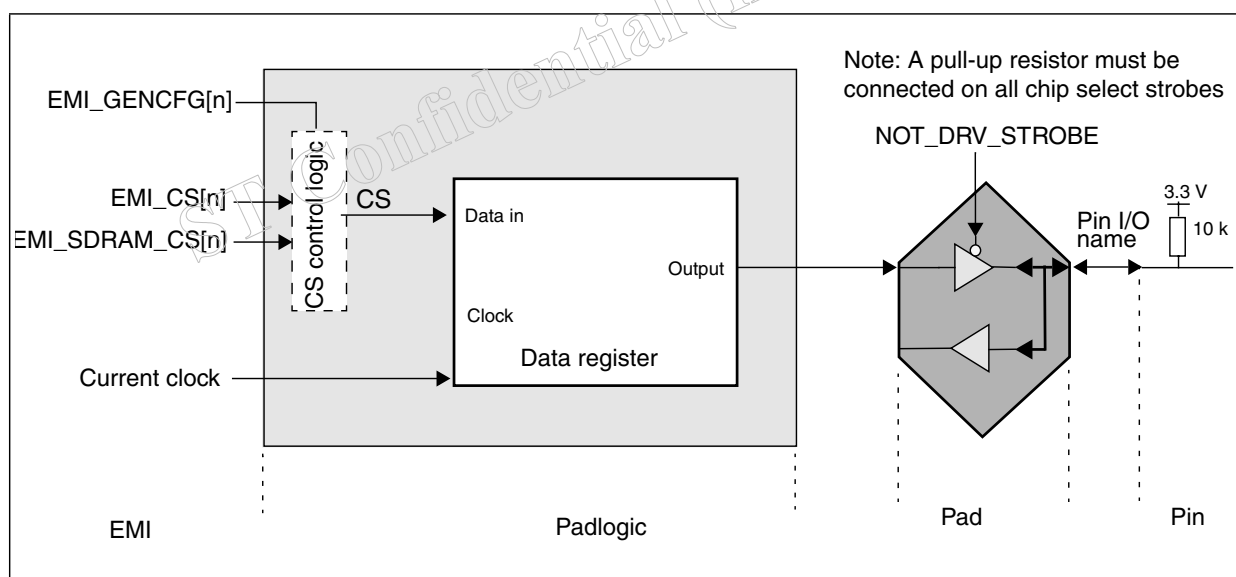
18.2.10 Chip select output strobe generation

The EMI provides the padlogic with ten chip selects for external devices (six dedicated banks 0 to 5; two dedicated to SDRAM in bank 0; two dedicated to SDRAM in bank 1). The padlogic controls which of the peripheral/SDRAM chip selects are passed through to the external chip selects. [Table 109: CSA to CSD control logic parameters](#) and [Table 111: CSE and CSF control logic parameters](#) show the control parameters of the CS control logic that perform the function of chip select switching.

SDRAM subdecoding

The use of external SDRAM devices only affects the chip selects A, B, C and D (normally assigned to banks 0 to 3).

Figure 54: SDRAM chip select connections



Confidential

Table 109: CSA to CSD control logic parameters

CS control logic setup	CS control logic output	Chip select function	Inverter needed	Pin I/O name
EMI_GENCFG[0]: 0	EMI_CS[0]	Bank 0 peripheral chip select	Yes	NOT_EMICSA
EMI_GENCFG[0]: 1	EMI_SDRAM_CS[0]	Bank 0 SDRAM chip select ^a		
EMI_GENCFG[1]: 0	EMI_CS[1]	Bank 1 peripheral chip select	Yes	NOT_EMICSB
EMI_GENCFG[1]: 1	EMI_SDRAM_CS[2]	Bank 1 SDRAM chip select ^a		
EMI_GENCFG[2]: 0	EMI_CS[2]	Bank 2 peripheral chip select	Yes	NOT_EMICSC
EMI_GENCFG[2]: 1	EMI_SDRAM_CS[1]	Bank 0 subdecoded upper address SDRAM chip select		
EMI_GENCFG[3]: 0	EMI_CS[3]	Bank 3 peripheral chip select	Yes	NOT_EMICSD
EMI_GENCFG[3]: 1	EMI_SDRAM_CS[3]	Bank 1 subdecoded upper address SDRAM chip select		

a. If there is no subdecoding, the normal chip select applies. However, if subdecoding is used it becomes the lower address subbank chip select.

Bank 5 subdecoding

To support contiguous memory address space in the boot bank, bank 5 supports subdecoding when a peripheral device (for example, flash) is in bank 5. The bank 5 subdecoding is controlled by the logic block referred to as the CS control logic in [Figure 55](#).

Figure 55: Bank 5 subdecoding chip select connections

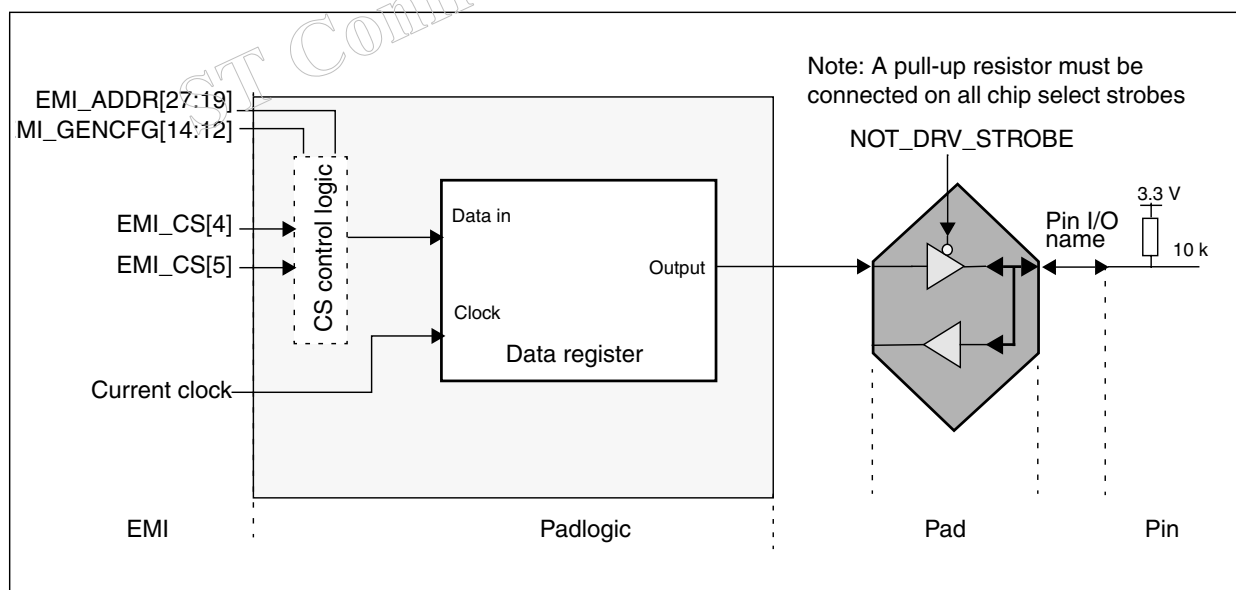


Table 110: Bank 5 subbank size control logic

Subbank size control (EMI_GENCFG[14:12])	Subbank size	Subbank 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 111: CSE and CSF control logic parameters

Bank 5 subdecoding control (EMI_GENCFG[11])	Subbank selection control	Outputs	Function	Inverter needed	Pin I/O name
0: Disabled	-	EMI_CS[4]	Bank 4 chip select	Yes	NOT_EMICSE
1: Enabled	0: Lower bank	EMI_CS[5]	Bank 5 lower subdecoded chip select		
	1: Upper bank	Inactive			
0: Disabled	-	EMI_CS[5]	Bank 5 chip select	Yes	NOT_EMICSF
1: Enabled	0: Lower bank	Inactive	Bank 5 upper subdecoded chip select		
	1: Upper bank	EMI_CS[5]			

18.2.11 Address bus shifting

The least significant address bus bit (bit 0) of any external device is connected to the STi5516 least significant address bus bit (bit 2). The port size for the EMI in the STi5516 may be either 16- or 8-bit, and address bus shifting must always be enabled. Bits 1 and 0 of the internal address bus are not connected externally.

Address bus shifting for banks 0 to 4 is enabled through the register EMI_GENCFG (bits 6 to 10). These bits should always be set to 1. For bank 5 the EMIBOOTMODE0 pin sets the port size and cannot be disabled. Bank 5 shifts the address bus according to the boot bank size values set, so the STi5516 device can successfully access the boot memory with the correct address and data bus size. The AP bit is always set to MEM_ADDRESS[12].

Address shifting also occurs during an SDRAM **modeset** instruction. To ensure the correct address shifting function is implemented, configuration bit 8 in register CONFIG_CONTROL_E must be set before the **modeset** instruction is issued (0 for a 16-bit port size and 1 for an 8-bit port size). CONFIG_CONTROL_E specifies the port size for all SDRAMs connected.

Table 112: Address shift mapping to internal address lines for banks 0 and 1

Device type		Port size		EMIADDR[25:2] mapping
Name	CONFIGDATA0 setting	Size	CONFIGDATA0 setting	
SDRAM ^a	010	16-bit	10	[24:1]
		8-bit	11	[23:0]
Peripheral	001	16-bit	10	[24:1]
		8-bit	11	[23:0]
SFlash TM	100	16-bit	10	[24:1]
		8-bit	11	[23:0]

a. If banks 0 and 1 are configured as SDRAM the BA select signals are shifted to the EMIADDR[18:17] pins.

Table 113: Address shift mapping of internal address lines for banks 2, 3, and 4

Device type		Port size		EMIADDR[25:2] mapping
Name	CONFIGDATA0 setting	Size	CONFIGDATA0 setting	
SDRAM	010	16-bit	10	Not available
		8-bit	11	
Peripheral	001	16-bit	10	[24:1]
		8-bit	11	[23:0]
SFlash TM	100	16-bit	10	[24:1]
		8-bit	11	[23:0]

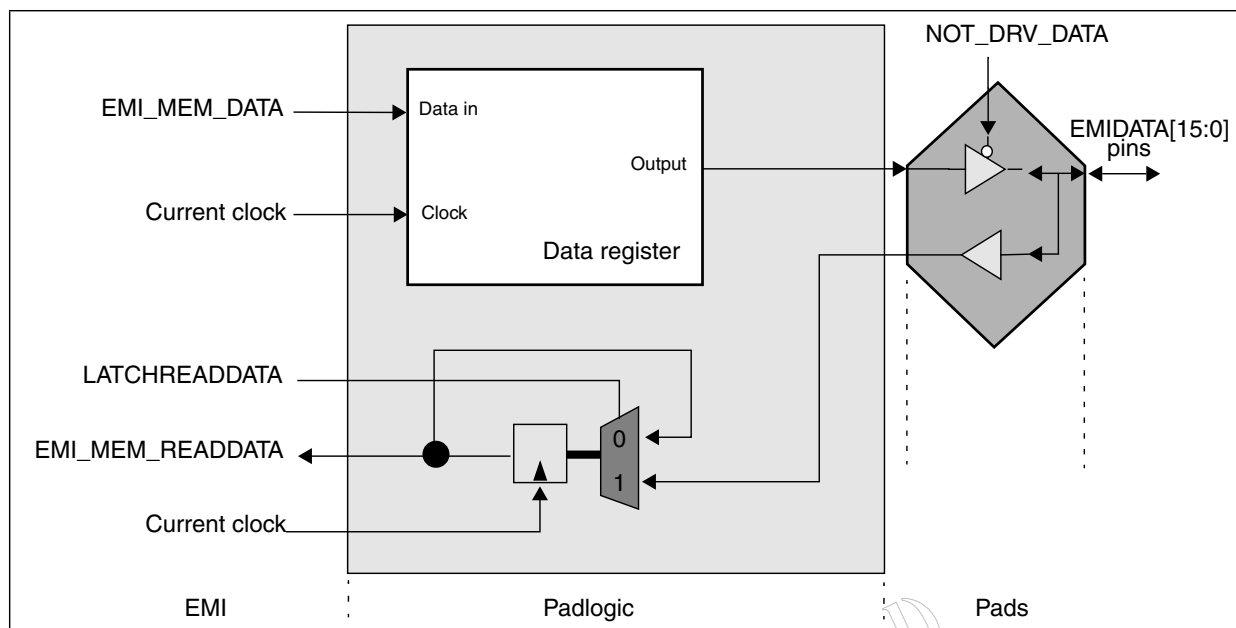
Table 114: Address shift mapping of internal address lines for bank 5

Device type		Port size		EMIADDR[25:2] mapping
Name	CONFIGDATA0 setting	Size	EMIBOOTMODE0	
SDRAM	010	16-bit	0	Not available
		8-bit	1	
Peripheral	001	16-bit	0	[24:1]
		8-bit	1	[23:0]
SFlash TM	100	16-bit	0	[24:1]
		8-bit	1	[23:0]

Note: EMI banks 2, 3, 4 and 5 do not support SDRAM devices.

18.2.12 Data bus control

Figure 56: Data bus two phase combination with output control



18.2.13 External clocks

The EMI padlogic is used to pass the necessary EMI clocks to the external devices. The SDRAM and peripheral (SFlash™) clocks operate as outputs only.

Figure 57: External clocks

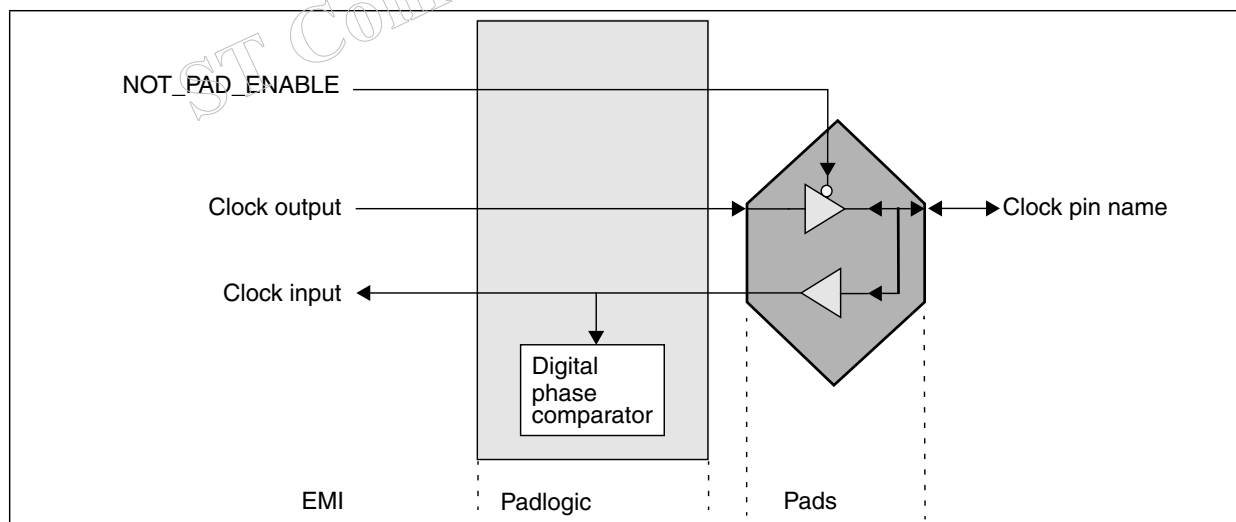


Table 115: External clock control

Clock output	Clock input	Pad enable	Pad direction	Clock pin name
EMI_ECLK_TO_PAD	EMI_ECLK_FROM_PAD	Tie low	Output	EMISDRAMCLK
EMI_FCLK_TO_PAD	EMI_FCLK_FROM_PAD	Tie low	Output	EMIFLASHCLK

Within the EMI padlogic there are two digital phase comparators, one for each of the device clocks (EMISDRAMCLK and EMIFLASHCLK). Each digital phase comparator samples the output clock relative to the internal version and provides feedback to the clock generator module, to ensure clock edge alignment.

18.2.14 EMI booting

During boot sequence, the EMI padlogic samples the boot mode input pin. This static input determines the boot bank (bank 5) port size. The STi5516 only supports a port size of 16 bits (EMIBOOTMODE0 = 0) or 8 bits (EMIBOOTMODE0 = 1).

18.2.15 Bank configurations

Each of the EMI banks can be configured separately to support different types of devices. There are restrictions on certain banks as can be seen in [Table 116](#).

Only banks 0 and 1 can be programmed to be of SDRAM type. Banks 0 and 1 can be subdecoded only when they have been configured as SDRAM. Subdecoding in banks 2, 3 and 4 is not supported.

Table 116: Bank configuration and subdecoding options

STi5516 allowable bank configurations						
	Bank0	Bank1	Bank2	Bank3	Bank4	Bank5
SDRAM device allowed	Y	Y	N	N	N	N
SDRAM subbank	Y	Y	-	-	-	-
Peripheral device allowed	Y	Y	Y	Y	Y	Y
Peripheral subbank	N	N	N	N	N	Y
DVB-CI/HDD	N	N	N	Y	N	N

Note: A maximum of two SDRAM devices only can be subdecoded into a single bank, although the generic EMI can support a maximum of four SDRAM devices. Subdecoding of four SDRAM devices is not supported in STi5516.

18.3 TRI_PT1 MUXing

There are two sources for the PTI's A_PTS_LATCH and V_PTS_LATCH. These are:

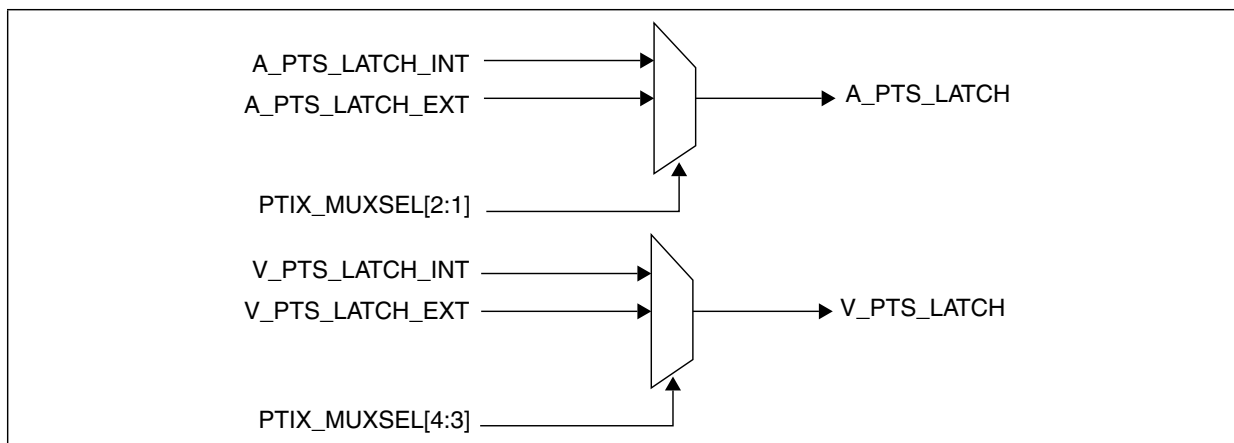
- one internal set of A_PTS_LATCH and V_PTS_LATCH.
- one external set of A_PTS_LATCH and V_PTS_LATCH (on the PIO alternate function).

Therefore MUXing is required to select the possible sources for the PTI.

Table 117: PTI MUX selects

Bit[2:1]	A_PTS_LATCH	Bit[4:3]	V_PTS_LATCH
0x	A_PTS_LATCH_INT	0x	V_PTS_LATCH_INT
10	A_PTS_LATCH_EXT	10	V_PTS_LATCH_EXT
11	Not applicable	11	Not applicable

Figure 58: MUXing for each PTI



18.4 CDREQ MUXing

There are several sources of CDREQ and DMAREQ available, these are:

- three internal CD request signal which are routed to the PTI (from the AV decoder),
- two external CD request signals that can be routed to the PTI (alternate pin functions),
- the audio interface generates DMAREQ signals that can be routed to the PTI,
- the SW transport input of the TSMUX generates a DMAREQ signal that can be routed to the PTI.

The targets for these CDREQs are:

- three NOT_CDREQs for the PTI.

Note: NOT_CDREQ is an active low signal indicating that the target is ready to accept new data.

The MUX control for each target is given in Table 118.

Table 118: CDREQ_SRC_MUXSEL source select for each target

Source	Virt CDREQ	Control[3:0] ^a
NOT_CDREQ_UNUSED_INP ^b	-	0x00
NOT_VIDEO_CDREQ_INT	-	0x01
NOT_AUDIO_CDREQ_INT	-	0x10
NOT_SP_CDREQ_INT	-	0x11
NOT_CDREQ_EXT[0]	0	1000
NOT_CDREQ_EXT[1]	1	1001
NOT_AIF_PCMO_REQ	2	1010
NOT_AIF_PCMI_REQ	3	1011
TSSUB_SWTS_REQ	4	1100
Unused	-	1101
Unused	-	1110
Unused	-	1111

a. From general purpose configuration bit

b. Inverted from general purpose configuration bit

19 Padlogic registers

Addresses are provided as the *BaseAddress* + offset.

The *BaseAddresses* are:

EMIBaseAddress: 0x2020 0000,

InterconnectBaseAddress: 0x2001 0000.

A register summary is given in [Table 47: Padlogic registers on page 73](#).

19.1 EMI general purpose configuration outputs

EMI_GENCFG

General purpose configuration outputs

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

EMI_GENCFG[31:0]

Address: *EMIBaseAddress* + 0x028

Type: Read/write

Reset: 0

Description: These control outputs are padlogic internal signals and are not passed to output pins.

[31:21]Reserved

[20]EMI_GENCFG[20]: Boot indicator

0: System in boot mode (default)

1: System in normal operation

[19:15]Reserved

[14:12]EMI_GENCFG[14:12]: Bank 5 subbank size

Refer to [Chapter 16: External memory interface \(EMI\) Bank 5 subdecoding on page 187](#)

[11]EMI_GENCFG[11]: Bank 5 subdecoding

0: Subdecoding disabled (default)

1: Subdecoding enabled

[10]EMI_GENCFG[10]: Bank 4 address shift

0: Default, always reset to 1(address shift enabled)

[9]EMI_GENCFG[9]: Bank 3 address shift

0: Default, always reset to 1(address shift enabled)

[8]EMI_GENCFG[8]: Bank 2 address shift

0: Default, always reset to 1(address shift enabled)

[7]EMI_GENCFG[7]: Bank 1 address shift

0: Default, always reset to 1(address shift enabled)

[6]EMI_GENCFG[6]: Bank 0 address shift

0: Default, always reset to 1(address shift enabled)

[5:4]EMI_GENCFG[5:4]: EMIWAITNOTTREADY retime control

See [Table 107: EMIWAITNOTTREADY retime stage control mapping on page 183](#)

[3]EMI_GENCFG[3]: CSD type

0: Other (default)

1: Bank 1 subdecoded SDRAM

[2]EMI_GENCFG[2]: CSC type

0: Other (default)

1: Bank 0 subdecoded SDRAM

[1]EMI_GENCFG[1]: CSB type

0: Other (default)

1: Bank 1 SDRAM

[0]EMI_GENCFG[0]: CSA type

0: Other (default)

1: Bank 0 SDRAM

Confidential

19.2 Monitor registers

CONFIG_DEVICEID_REG Device ID 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

CONFIG_DEVICEID_REG

Address: *InterconnectBaseAddress* + 0x010

Type: Read only

Reset: Undefined

Description: This register returns the device ID.

19.3 Configuration registers

Note: Reserved configuration register bits must not be changed from the default as this causes the device to malfunction.

CONFIG_CONTROL_A Interconnect configuration control register A

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

Reserved	SMCARDDB_DSSSMCLK_NOT_PIOBIT3	SMCARDDB_DSSSMCLK_NOT_PIOBIT3	CDREQ_UNUSED_INP	PTIA_CDREQ3_MUXSEL[3:0]	PTIA_CDREQ2_MUXSEL[3:0]	PTIA_CDREQ1_MUXSEL[3:0]	Reserved	PTIA_AVC_MUXSEL[4:1]	Reserved

Address: *InterconnectBaseAddress* + 0x00

Type: Read/write

Reset: 0

Description: .

[31:30]Reserved

[29]SMCARDDB_DSSSMCLK_NOT_PIOBIT3: Override of PIO1bit3 to DSS smartcard clock

[28]SMCARDDB_DSSSMCLK_NOT_PIOBIT3: Override of PIO0 bit3 to DSS smartcard clock

[27:15]CDREQ_UNUSED_INP, PTIA_CDREQ3_MUXSEL, PTIA_CDREQ2_MUXSEL, PTIA_CDREQ1_MUXSEL

See [Section 18.4: CDREQ MUXing on page 192](#) for CDREQ and DMAREQ sources.

[14:5]Reserved

[4:1]PTIA_AVC_MUXSEL

See [Section 18.3: TRI_PTI MUXing on page 191](#) for A_PTS_LATCH and V_PTS_LATCH sources.

[0]Reserved

Confidential

CONFIG_CONTROL_B Interconnect configuration control register B

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

TTXTOUTREQSEL_ALTNOTDENC	1284_MASTER_MODE	EXT_INTERRUPT_ENABLE[3:0]	COMMS_SSC1_DOUT_MRST_NOTMTSR_MUXSEL	COMMS_SSC0_DOUT_MRST_NOTMTSR_MUXSEL	Reserved																									

Address: *InterconnectBaseAddress* + 0x04

Type: Read/write

Reset: 0

Description:

[31] TTXTOUTREQSEL_ALTNOTDENC

Selects between digital encoder and alternate function pad control of TTXT out request

1: PIO alternate

0: Digital encoder

To block: TTXT

[30] I1284_MASTER_MODE

This is additional logic to the LLI1/1284 to allow for 1284 master mode. In this mode, pairs of the 1284 interface swap functionality and direction

The five pairs of signals are:

1284NOTSELECTIN

1284NOTFAULT

1284NOTINIT

1284SELECT

1284NOTAUTOFD

1284PERROR

1284HOSTLOGICH

1284BUSY

1284NOTSTROBE

1284NOTACK

1: Enable master mode

0: Disable master mode

[29:26] EXT_INTERRUPT_ENABLE[3:0]: Control of external interrupt direction

0: Input

[25] COMMS_SSC1_DOUT_MRST_NOTMTSR_MUXSEL

Selects between SSC1 MRST and MTSR serial data out

1: SSC1 MRST

0: SSC1 MTSR

To block: PIO alternate

[24] COMMS_SSC0_DOUT_MRST_NOTMTSR_MUXSEL

Selects between SSC0 MRST and MTSR serial data out

1: SSC1 MRST

0: SSC1 MTSR

To block: PIO alternate

[23:0] Reserved

Confidential

CONFIG_CONTROL_C Interconnect configuration control register 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
Reserved														VIDEODAC_ENABLE		Reserved															
														AUDIODAC_MUTE																	
Reserved														CONFIG_OTHER_ALT_IRB_DRIVE_NOTIRCNTL_RX_RST		Reserved															
														CONFIG_OTHER_ALT_IRD_NOTIRC_DOUT																	
														CONFIG_OTHER_ALT_PIOPORT4[1]																	
														CONFIG_OTHER_ALT_PIOPORT4[0]																	
														AUDIOPCMDATA_1SEL_3_NOT0																	
Reserved														AUDIOPCMDATA_ADACSEL_3_NOT0		Reserved															

Address: *InterconnectBaseAddress* + 0x08

Type: Read/write

Reset: 0

Description:

[31:15] **Reserved**[14] **VIDEODAC_ENABLE:** Video DAC enable

1: Enabled 0: Disabled

[13] **AUDIODAC_MUTE:** Audio DAC mute

1: Mute 0: Not muted

[12:6] **Reserved**[5] **CONFIG_OTHER_ALT_IRB_DRIVE_NOTIRCNTL_RX_RST**

Always set this bit to 1

[4] **CONFIG_OTHER_ALT_IRD_NOTIRC_DOUT**

Always set this bit to 1

[3] **CONFIG_OTHER_ALT_PIOPORT4[1]**

1: CFC as input 0: PIO

See [Table 25: Port 5 PIO signal assignments on page 39](#)
(bit 1)[2] **CONFIG_OTHER_ALT_PIOPORT4[0]**

1: OSDEN as bidirectional, video output enable controls direction.

See [Table 25: Port 5 PIO signal assignments on page 39](#)
(bit 0)

0: PIO

[1] **AUDIOPCMDATA_1SEL_3_NOT0**

Bits 1 and 0 allow main and VCR PCM data combinations to be output on PCM output and audio DAC.

1: Audio PCMD[3] 0: Audio PCMD[0] Output: PCMDATA1 pin

[0] **AUDIOPCMDATA_ADACSEL_3_NOT0**

1: Audio PCMD[3] 0: Audio PCMD[0] Output: internal audio DAC input

Confidential

CONFIG_CONTROL_D Interconnect configuration control register 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								IRDA_ASC_RX_DATA_IN_EN	UHF_IN_EN	RC_IRDA_DATA_IN_EN	MAFE_OR_UART4_SEL	PCMCLK_FROM_PAD	Reserved	SMI_DATAIN_DEL_CTRL	Reserved	AUDIODAC_DIGITAL_STANDBY	Reserved	AUDIODAC_ANALOG_PWRDN	Reserved															

Address: *InterconnectBaseAddress* + 0x0C

Type: Read/write

Reset: 0

Description:

[31:24] **Reserved**

[23:21] **IRDA_ASC_RX_DATA_IN_EN, UHF_IN_EN, RC_IRDA_DATA_IN_EN**

Comms infrared transmitter/receiver input enables (for wake up masking)

1: Input enabled

[20] **MAFE_OR_UART4_SEL**

1: Select ASC4 instead of MAFE on PIO2

[19] **PCMCLK_FROM_PAD**

Selects PCMCLK to clock generator to come from external pad

[18] **Reserved**

[17:16] **SMI_DATAIN_DEL_CTRL**

Controls delay insertion on SMI data in path

11: Maximum delay

00: Minimum delay

[15] **Reserved**

[14] **AUDIODAC_DIGITAL_STANDBY**

[13] **Reserved**

[12] **AUDIODAC_ANALOG_PWRDN**

1: Power down analog DACs

0 must be written to this bit to be forward compatible with the STi5517, where the bit is 1 by default.

[11:0] **Reserved**

Confidential

CONFIG_CONTROL_E Interconnect configuration control register 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													CONFIG_EMISS_PORTSIZEBANK3BIT	CONFIG_EMISS_DVBCI_ENABLE	CONFIG_IRB_UART2RXD_BYPASS	CONFIG_IRDACTRL_DIN_DISABLE	CONFIG_AUDIFREMAP_OVERRIDE	Reserved	CONFIG_PCMDIVCLK_OVERRIDE	CONFIG_OTHER_ALT_PIO_YC	TO_PAD_TS2L_OE	Reserved	EMI_SDRAM_PRTSZ	Reserved	Reserved	EMI_CTRL_DS_PROG[1:0]	EMI_DATA_DS_PROG[1:0]	EMI_ADDR_DS_PROG[1:0]			

Address: *InterconnectBaseAddress* + 0x28

Type: Read/write

Reset: 0

Description:

[31:17] **Reserved**[18] **CONFIG_EMISS_PORTSIZEBANK3BIT**

0: 16-bit

1: 8-bit

The value of this bit must match the port size of the peripheral connected.

[17] **CONFIG_EMISS_DVB_ENABLE**

0: Disabled

1: DVB-CI or HDD mode enabled

See [Section 18.2.7: DVB-CI support on EMI bank 3 on page 184](#)[16] **CONFIG_IRB_UART2RXD_BYPASS**

0: Infrared transmitter/receiver to UART2RXD

1: PIO4[3] to UART2 directly

[15] **CONFIG_IRDACTRL_DIN_DISABLE**

Always set to 1

[14] **CONFIG_AUDIFREMAP_OVERRIDE**

0: Remapping

1: No remapping

[13] **Reserved**[12] **CONFIG_PCMDIVCLK_OVERRIDE**

0: Functional mode (divider on the clock)

1: Dolby® certification (no divider on the clock)

[11] **CONFIG_OTHER_ALT_PIO_YC**

0: Normal (after PIO)

1: YC

[10] **TO_PAD_TS2L_OE**: Routing out TSIN1

0: TSIN2L pins are inputs

1: TSIN1 signals routed out on TSIN2L pins

[9] **Reserved**[8] **EMI_SDRAM_PRTSZ**: Set size of SDRAM connected to EMI

0: 16-bit (default after reset)

1: 8-bit

This value must be written before performing an SDRAM **modeset** command on the EMI.[7:6] **Reserved**[5:4] **EMI_CTRL_DS_PROG[1:0]**: Set drive strength on EMI control (all other) pads

11: 15 pF

10: 35 pF

01: 5 pF

00: 25 pF (default after hard reset)

[3:2] **EMI_DATA_DS_PROG[1:0]**: Set drive strength on EMI data pads

11: 15 pF

10: 35 pF

01: 5 pF

00: 25 pF (default after hard reset)

[1:0] **EMI_ADDR_DS_PROG[1:0]**: Set drive strength on EMI address pads

11: 15 pF

10: 35 pF

01: 5 pF

00: 25 pF (default after hard reset)

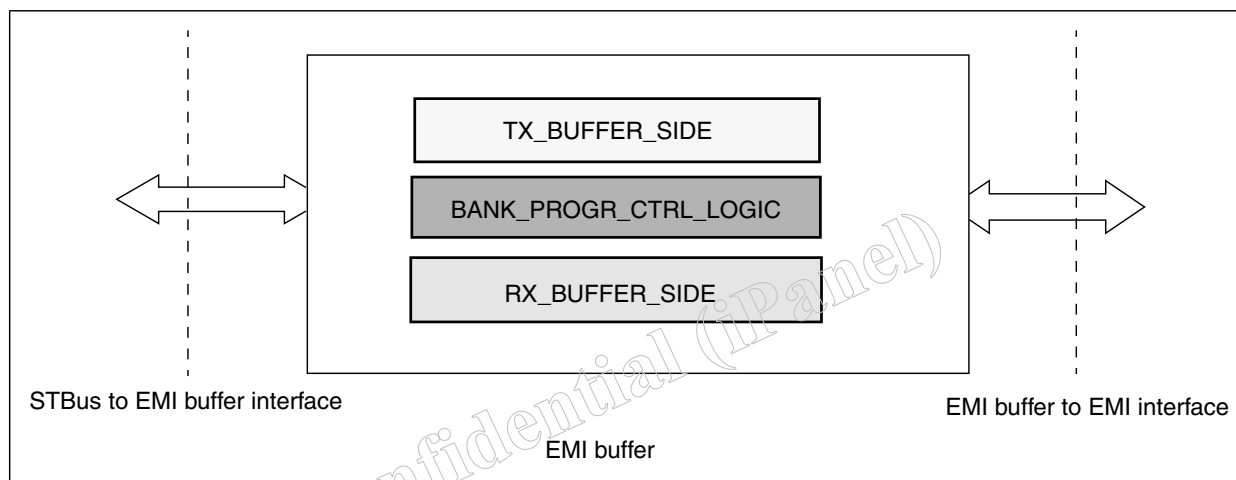
Confidential

20 EMI buffer

The EMI buffer is located between the STBus and the EMI. It handles the regular exchange of information between these two blocks, buffering all data from the STBus interconnect before passing it on to the EMI block. In particular the block performs the functions listed below.

- Guarantees a continuous data flow to the EMI block. This means that the block must be able to transmit a packet from the STBus interconnect block towards the EMI block without a gap between two consecutive cells in the packet.
- Support for EMI bank size programmability as well as the generation of the right external bank to access.

Figure 59: EMI buffer



The TX_BUFFER_SIDE handles the request packet from the STBus block to the EMI block. The EMI buffer deals with the timing constraints and continuous data flow in a burst transaction.

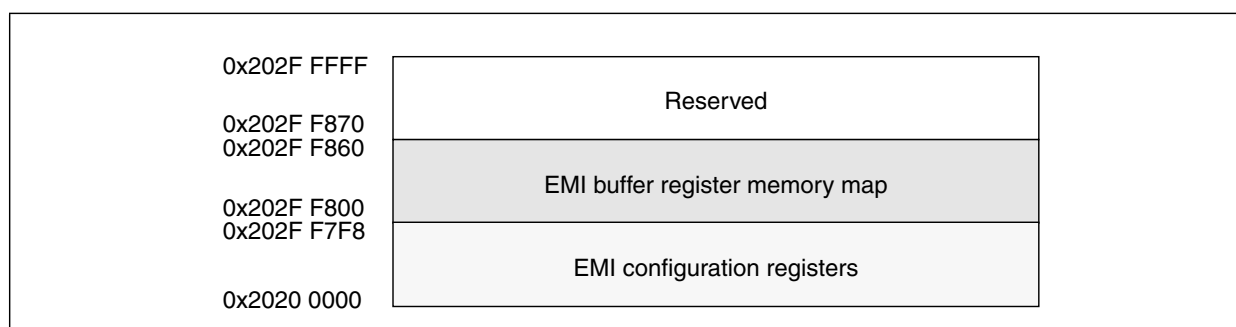
The RX_BUFFER_SIDE completes the same operation as the TX_BUFFER_SIDE for the response packet request from the EMI block to the STBus block.

The BANK_PROGR_CTRL_LOGIC controls the internal registers of the EMI buffer block containing the top address of the external memory banks. This block also provides the total number of external memory banks enabled at the same time (after setting an internal register).

The EMI buffer block is characterized by the following seven internal registers:

- six related to the accessible external memory banks (each composed of 8 bits),
- one related to the value of the total number of banks registers enabled at the same time (composed of 3 bits).

Figure 60: EMI memory map



21 EMI buffer registers

Addresses are provided as the *EMIBufferBaseAddress* + offset.

The *EMIBufferBaseAddress* is:

0x202F F800.

A register summary is given in [Table 38: External memory interface \(EMI\) buffer registers on page 64](#).

All EMI buffer registers are nonvolatile.

BANK_0_BASE_ADDRESS External memory bank 0 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												BANK_0_BASE_ADDRESS			

Address: *EMIBufferBaseAddress* + 0x000

Type: Read/write

Reset: 0x00

Description: Contains bits 27 to 22 of the base address of external memory bank 0. Accesses to this address space cause transfer on EMI bank0 (0x4000 0000 to 0x4FFF FFFF).

Note: Do not change from the reset state when booting from ROM.

BANK_1_BASE_ADDRESS External memory bank 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
Reserved																												BANK_1_BASE_ADDRESS			

Address: *EMIBufferBaseAddress* + 0x010

Type: Read/write

Reset: 0x04

Description: Contains bits 27 to 22 of the base address of external memory bank 1. Accesses to this address space cause transfer on EMI bank1 (0x5000 0000 to 0x5FFF FFFF).

Confidential

BANK_2_BASE_ADDRESS External memory bank 2 base address

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

Reserved																								BANK_2_BASE_ADDRESS			

Address: *EMIBufferBaseAddress* + 0x020

Type: Read/write

Reset: 0x08

Description: Contains bits 27 to 22 of the base address of external memory bank 2.
 Accesses to this address space cause transfer on EMI bank2 (0x6000 0000 to 0x6FFF FFFF).

BANK_3_BASE_ADDRESS External memory bank 3 base address

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

Reserved																								BANK_3_BASE_ADDRESS			

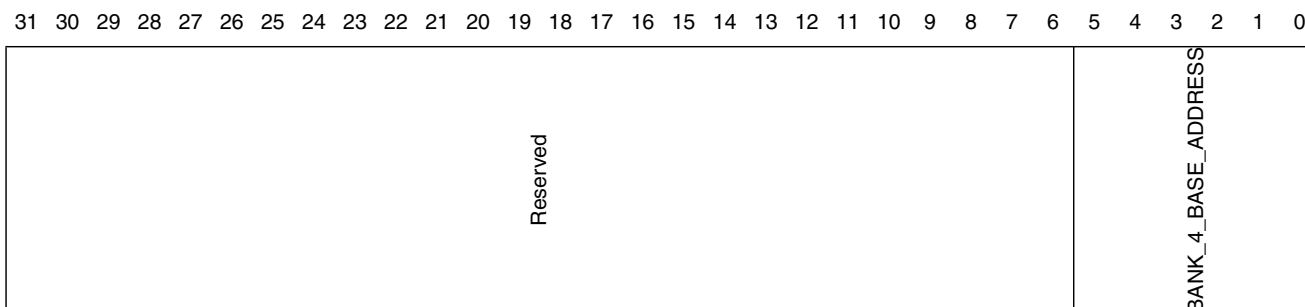
Address: *EMIBufferBaseAddress* + 0x030

Type: Read/write

Reset: 0x0C

Description: Contains bits 27 to 22 of the base address of external memory bank 3.
 Accesses to this address space cause transfer on EMI bank 3 (0x7000 0000 to 0x7FFF FFFF).

Confidential

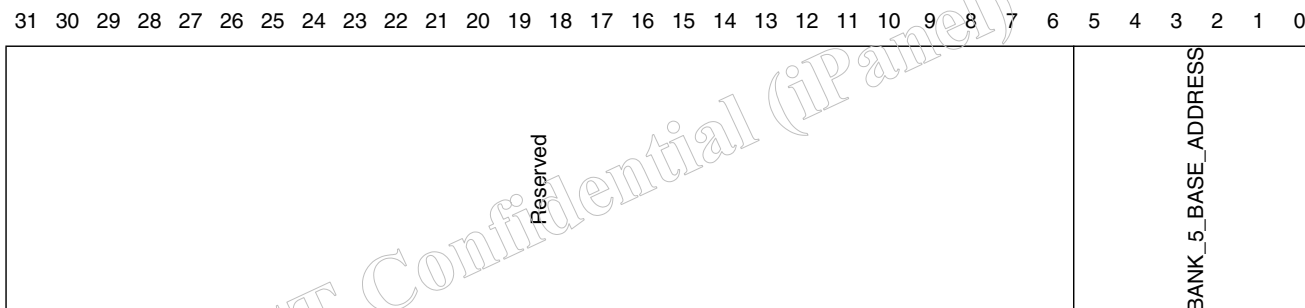
BANK_4_BASE_ADDRESS External memory bank 4 base address

Address: *EMIBufferBaseAddress* + 0x040

Type: Read/write

Reset: 0x10

Description: Contains bits 27 to 22 of the base address of external memory bank 4. Accesses to this address space cause transfer on EMI bank 4 (0x7F00 0000 to 0x7F7F FFFF).

BANK_5_BASE_ADDRESS External memory bank 5 base address

Address: *EMIBufferBaseAddress* + 0x050

Type: Read/write

Reset: 0x14

Description: Contains bits 27 to 22 of the base address of external memory bank 5. Accesses to this address space cause transfer on EMI bank 5 (0x7F80 0000 to 0x7FFF FFFF).

Confidential

BANKS_ENABLED**Enabled bank 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																										BANKS_ENABLED	

Address: *EMIBufferBaseAddress* + 0x060

Type: Read/write

Reset: 0x06

Description: Contains the total number of bank registers enabled.

At reset all the banks are enabled.

6 (110) = All banks enabled

5 (101) = Banks 5 down to 1 enabled

4 (100) = Banks 5 down to 2 enabled

3 (011) = Banks 5 down to 3 enabled

2 (010) = Banks 5 down to 4 enabled

1 (001) = Bank 5 only 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 five banks are enabled, BANK0 is disabled, then BANK1 region contains its own area plus the BANK0 area.

ST Confidential (201403)

Confidential

22 System services

22.1 Overview

The system services module includes all of the necessary logic to initialize the device. Device initialization and debugging can also be done with the diagnostic controller unit (DCU); see [Section 23: Diagnostic controller \(DCU\) on page 205](#).

22.2 Power-on hard reset

The NOT_RESET pin provides a power-on or hard reset function. It must be asserted (low) before the clocks and power supply are stable. When the NOT_RESET pin is asserted (regardless of any other inputs), all modules are asynchronously forced into their power on reset state.

The NOT_RESET pin should only be de-asserted (high) after both of the following events have taken place:

- the clocks and power are stable, to guarantee well defined behavior,
- the NOT_TRST TAP reset pin has been asserted.

When the NOT_RESET pin is de-asserted, the CPU enters its boot sequence. The sequence starts only after the rising edge of the NOT_RESET pin is internally synchronized and the clocks are stable.

Bootstrap code can either be in off-chip ROM or can be received through the DCU.

22.3 Bootstrap

The STi5516 can be bootstrapped from the diagnostic controller (DCU) or ROM.

22.3.1 Booting from the DCU

The STi5516 can be booted from the DCU at any time by setting up the test access port (TAP) to do so. The procedure is explained in [Section 23: Diagnostic controller \(DCU\) on page 205](#).

If the device is not set up to boot from DCU, the STi5516 boots from ROM as soon as it leaves the reset state.

22.3.2 Booting from ROM

Any value other than 0 on the EMIBOOTMODE0 pin causes the STi5516 to boot from ROM as it comes out of reset.

Boot code is run from a slow external ROM placed in bank 3 at the top of memory. The ROM width is 16 bits wide. When booting from ROM, the value in the configuration registers for the PORTSIZE for bank 3 is disregarded.

When booting from ROM, the STi5516 starts to execute code from the top two bytes in external memory, at address 0x7FFF FFFE, which should contain a backward jump to a bootstrap program in ROM.

23 Diagnostic controller (DCU)

23.1 Overview

The STi5516 diagnostic controller unit (DCU) is used to boot the CPU and to control and monitor all of the systems on the chip, via the standard IEEE 1194.1 test access port. The DCU includes on-chip hardware with ICE (in-circuit emulation) and LSA (logic state analyzer) features to facilitate verification and debugging of software running on the on-chip CPU in real time. It is an independent hardware module with a private link from the host to support real-time diagnostics.

The DCU has the following features.

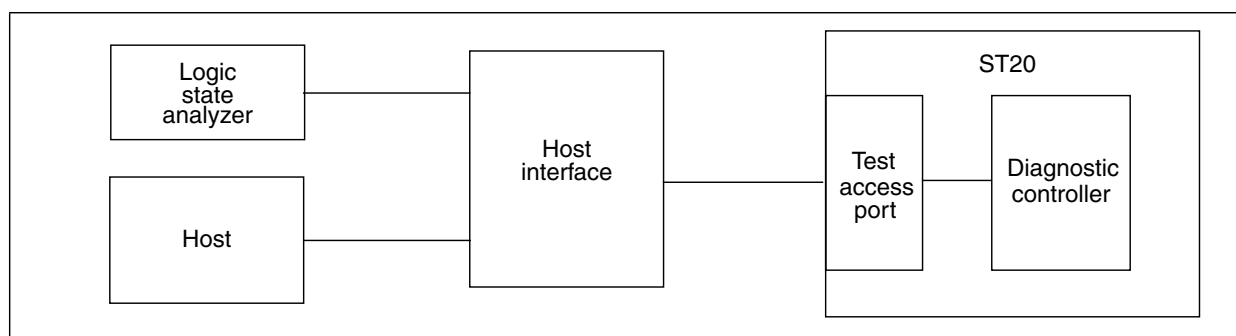
- Unified compare blocks, each of which is capable of breakpoint, breakcount, breakrange, inverse breakrange, watchrange, inverse watchrange, datawatch or watchcount operations. The DCU on STi5516 contains four compare blocks.
- Four capture blocks. Each capture block can store the value of the instruction pointer, workspace pointer, data address or write data value when a particular event occurs (for example, match from a compare block).
- A jump trace which can store:
 - instruction pointer to the source of the code jump,
 - instruction pointer to the destination of the code jump,
 - value in a capture block,
 - cycle count.

The events which cause these to be stored are selected from: code jump, context change (change between threads), event from a compare block and store of a capture block. The DCU jump trace stores the information in a compressed format.

23.2 Diagnostic hardware

The on-chip diagnostic controller assists in debugging, while either reducing or eliminating the intrusion into the target code space, the CPU utilization and impact on the application. As shown in [Figure 61](#), the DCU and TAP provide a means of connecting a diagnostic host to a target board with a suitable JTAG port connector and interface.

Figure 61: Debugging hardware



The diagnostic controller provides the following facilities for debugging from a host:

- control of target CPU and subsystems including CPU boot,
- hardware breakpoint, watchpoint, datawatch and single instruction step,
- complex trigger sequencing and choice of subsequent actions,
- nonintrusive jump trace and instruction pointer profiling,
- access to the memory of the target while the device is powered up, regardless of the state of the CPU,
- full debugging of ROM code.

When running multitasking code on the target, one or more processes can be single stepped or stopped while others continue running in real time. In this case, the running threads can be interrupted by incoming hardware interrupts, with a low latency.

The host can communicate with the DCU via a private link, using the five standard test pins.

Target software allows access to the diagnostic facilities and access through the DCU to the host memory.

A logic state analyzer can be connected to the DCUTRIGGERIN and DCUTRIGGEROUT pins. The response to DCUTRIGGERIN and the events that cause a TRIGGEROUT signal can be controlled by the host or by target software.

The diagnostic controller provides debugging facilities with much less impact on the software and target performance. In particular:

- nonintrusive attachment to the host system,
- no intrusion into the performance of the CPU or any subsystems,
- no intrusion into the code space, so the application builder does not need to add a debugging kernel,
- no intrusion into any on-chip functional modules, including any communications facilities,
- no functional external connection pins are used.

The connections between the diagnostic controller and other on-chip modules and external hardware may vary between ST20 variants.

23.3 Access features

23.3.1 Access to target memory and peripheral registers from the host

Full read and write access to the entire on-chip and external memory space, and the register space, is available via the TAP. This is independent of the state of the CPU.

23.3.2 Access from the target CPU process

The CPU itself can program its own diagnostic controller. Further access may be explicitly prevented by the lock mechanism so that the application being debugged cannot interfere with the breakpoint and watchpoint settings. When the breakpoint or watchpoint match occurs, then the diagnostic controller may release the lock according to settings in the DCU_CONTROL register.

23.3.3 Access to host memory from target

If the target CPU accesses any address in the top half of the DCU memory space, these accesses are mapped on to host memory via the TAP as target initiated PEEK and POKE messages. Peek and poke accesses are specifically enabled by separate property bits.

23.4 Software debugging features

23.4.1 Control of the target CPU including boot

Various state information about the target CPU may be monitored, and the CPU may be controlled from the diagnostic controller via the TAP. The control of the CPU extends to stalling, forcing a trap and booting.

23.4.2 Nonintrusive IPTR profiling

A copy of the IPTR is visible as a read-only register in the diagnostic controller. This register may be read at any time. Reading this register is not intrusive on the CPU or its memory space.

23.4.3 Events

Support is provided by the diagnostic controller to trigger actions when certain predefined events occur.

Table 119: Software debugging events

Event	Action
Breakpoint	The function of the breakpoint is to break before the instruction is executed, but only if it really was going to be executed. A 32-bit comparator is used to compare the breakpoint register against the instruction pointer of the next instruction to be executed. The matched instruction is not executed and the CPU state, including all CPU registers, is defined as at the start of the instruction. The previous instruction is run to completion.
Breakpoint range	The function of a breakpoint range is equivalent to any single breakpoint but where the breakpoint address can be anywhere within a range of addresses bound by lower and upper register values.
Watchpoint	The function of a watchpoint is to trigger after a memory access is made to an address within the range specified by a pair of 32-bit registers. The CPU pipeline architecture allows for the CPU to continue execution of instructions without necessarily waiting for a write access to complete. So, by the time a watchpoint violation has been detected, the CPU may have executed a number of instructions after the instruction which caused the violation. If the subsequent action is to stall the CPU or to take a hardware trap, then the last instruction executed before the stall or trap may not be the instruction which caused the violation.
Datawatch	The function of a datawatch is to trigger after a data value specified in one 32-bit register is written to a memory word address specified in another 32-bit register. The subsequent action is equivalent to a watchpoint.

Following a watchpoint match, or any other condition detectable by the diagnostic controller, the subsequent action may be programmed to do one of the following:

- stall the CPU, that is, inhibit further instructions from being executed by the CPU,
- wait until the end of the current instruction, then signal a hardware trap,
- signal an immediate hardware trap,
- continue without intrusion.

In addition, the diagnostic controller may take any combination of the following actions:

- signal on TRIGGEROUT to a logic state analyzer,
- send a triggered message via the TAP to the host,
- unlock access by the target CPU.

23.4.4 Hardware single instruction step

The function of single stepping one CPU instruction is performed by using a breakpoint range over the code to be single stepped. The DCU includes a mechanism to prevent the breakpoint trap handler single stepping itself. By selecting an inverse range, the effect of single stepping one high level instruction can be achieved.

23.4.5 Jump trace

Jump tracing monitors code jumps, where a jump is any change in execution flow from the stream of consecutive instructions stored in memory. A jump may be caused by a program instruction, an interrupt or a trap.

When the jump occurs, a 32-bit DCU register is loaded with the origin of the jump. This value points to the instruction which would have been executed next if the jump had not occurred. The CPU may not have completed the instruction prior to the change in flow. The diagnostic controller can be set to trace the origin of each jump, the destination, or both.

The DCU copies the details of each jump to a rolling trace buffer in memory. The trace buffer may be located in host memory, but using target memory has less impact on performance. The tracing facility has two modes.

- Low intrusion: in this mode, the DCU uses dead memory cycles to write the trace into the buffer. This means that the CPU is not delayed, but some trace information may be lost.
- Complete trace: in this mode, the CPU is stalled on every jump to ensure the data can be written to the buffer. This means that no trace information is lost, but the CPU performance is affected.

23.4.6 Logic state analyzer (LSA) support

Two signals, TRIGGERIN and TRIGGEROUT, are provided to support diagnostics with an external LSA. The action by the DCU on receiving a TRIGGERIN signal is programmable. The selection of internal events which trigger a TRIGGEROUT signal is also programmable.

23.4.7 Trigger combinations and sequences

Complex trigger conditions can be programmed. For example, the fifth time that breakpoint 3 is encountered or to enable a watchpoint when a breakpoint occurs.

There is no software intrusion imposed by this mechanism.

23.5 Controlling the diagnostic controller

This section gives a summary of host communications with the diagnostic controller.

The diagnostic controller has direct access to:

- the instruction pointer,
- a selection of CPU state control signals,
- the memory bus,
- memory-mapped peripheral configuration registers.

This access does not depend on the state of the CPU. Access to nonmemory-mapped peripheral configuration registers is via the CPU, and for this the CPU must be active and running the appropriate handler.

The host can give two commands to the diagnostic controller: **peek** and **poke**. **peek** reads memory locations or configuration registers, and **poke** writes to memory locations or configuration registers. The diagnostic controller responds to a **peek** command with a PEEKED message, giving the contents of the peeked addresses.

The diagnostic controller has registers, which are accessed from the host using **peek** and **poke** commands. The registers are used to control breakpoints, watchpoints, datawatch, tracing and other facilities.

The target CPU can also access these registers using the normal **load** and **store** instructions, so the target software running on the CPU can program its own diagnostic controller. A lock is provided to prevent CPU access, which can be released by the diagnostic controller when a breakpoint or watchpoint match occurs.

In addition, the target CPU can peek and poke the host via the diagnostic controller by reading or writing addresses in the top half of the memory space of the diagnostic controller. This facility can be disabled.

Various different types of CPU events can be selected as trigger events. When a trigger event occurs, the diagnostic controller can send a triggered message.

The four types of message are summarized in [Table 120](#) below. The messages are distinguished by the two least significant bits of the message header byte.

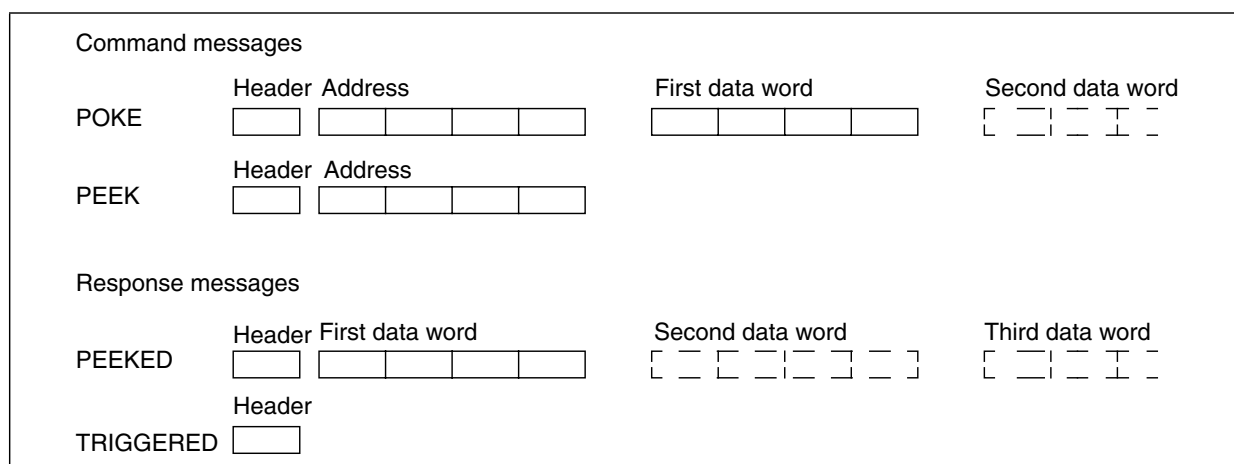
Table 120: Diagnostic controller message types

Message type	Direction	Bit 1	Bit 0	Meaning
POKE	Command	0	0	Write to one or more addresses
PEEK	Command	0	1	Read from one or more addresses
PEEKED	Opposite of peek command	1	0	The result of a peek command
TRIGGERED	DCU to host	1	1	A trigger event has occurred

Messages may be initiated from either the host or the target. Target initiated messages, which constitute asynchronous or unsolicited messages, can be enabled by a property bit.

Messages are composed of a header byte followed by zero or more data bytes, depending on the type of message. The formats for the four message types are shown in [Figure 62](#).

Figure 62: Message formats



23.6 Peeking and poking the host from the target

The target CPU can peek and poke the host via the diagnostic controller. This is done by reading or writing a single word to a block of addresses within the DCU register block. The DCU then sends a PEEK or POKE message to the host. After a host PEEK, the target CPU waits until the host responds with a peeked message, which the DCU returns to the CPU as memory read data.

Peeking and poking the host from the target can be enabled or disabled. After reset, these bits are cleared, so peek and poke from the target are disabled.

Confidential

ST Confidential (iPanel)

24 Diagnostic controller (DCU) registers

All the functions of the DCU are controlled by values in its registers. The base address of the DCU is programmable depending on the value of the *DCUBaseAddress* input.

The addresses given in the following tables are offsets from this DCU base address. The register allocation allows for up to 32 blocks (any except jump trace). For any particular implementation, the value in the capability register and sequence configuration register can be used to calculate the address of all registers. [Table 34: Diagnostic controller \(DCU\) registers](#) shows addresses assuming three compare blocks, two capture blocks, one TRIGGER_IN block, one jump trace, one work space range compare block, with full connectivity of sequencing.

Addresses are provided as the *DCUBaseAddress* + offset.

The *DCUBaseAddress* is:

0x3000 3000.

A register summary is given in [Table 34: Diagnostic controller \(DCU\) registers on page 56](#).

24.1 General registers

Note: All unused register bits have value zero when read and should be written with zero.

DCU_CAPABILITY

Capability 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
MAJOR_VERSION				MINOR_VERSION				Reserved				NUM_W_RANGE				NUM_CAPTURE				NUM_COMPARE				JTR_SUPPORT		NUM_TRIG_IN					

Address: *DCUBaseAddress* + 0x000

Type: Read only

Reset: See table

Description: The capability register provides information about the hardware configuration of a particular implementation of the DCU to allow software to see what hardware is available.

[31:29] **MAJOR_VERSION**

DCU major version number

Reset (depends on implementation): 1

[28:26] **MINOR_VERSION**

DCU minor version number

Reset (depends on implementation): 0

[25:21] **Reserved**

[20:16] **NUM_W_RANGE**

The number of work space range detect blocks

Reset (depends on implementation): 1

[15:11] **NUM_CAPTURE**

Number of capture blocks

Reset (depends on implementation): 2

[10:6] **NUM_COMPARE**

Number of compare blocks

Reset (depends on implementation): 3

[5] **JTR_SUPPORT**

1 if jump trace supported, 0 if not.

Reset (depends on implementation): 1

[4:0] **NUM_TRIG_IN**

The number of TRIGGER_IN blocks

Reset (depends on implementation): 1

DCU_CONTROL

Control 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
DCU_CONTROL_OUT																IN_TRAP_LOCK	IN_TRAP	BYTE_ENABLES[3:0]				DEVICE_ACCESS	DISABLE	TARGET_PEEK_ENABLE	TARGET_POKE_ENABLE	WRITE_LOCK	HOSTED_BUSY	TRIGGER	SINGLE_STEP	CPU_TRAP	CPU_STALL

Address: *DCUBaseAddress* + 0x004

Type: Read/write

Reset: 0 (BYTE_ENABLES = 1111, CPU_STALL depends on HOST_CONNECTED signal)

Description:

- [31:16] **DCU_CONTROL_OUT**: General purpose outputs for functions like CPU reset, subsystem reset and stall. Output pins from the DCU reflect whatever value is written to DCU control out. The connection of these pins are product specific and include controls required to reset or stall the product during debugging.
- [15] **IN_TRAP_LOCK**: Disables IN_TRAP from being cleared automatically on an IPTR jump. This bit is set automatically on entering the diagnostics trap handler. It is cleared by the user at the end of the trap handler to re-enable the DCU functions on exiting the trap handler. Interrupts need to be turned off before clearing this bit to ensure that the first jump with this bit clear is the trap return.
- [14] **IN_TRAP**: Disables the DCU functions. This is automatically set when the CPU invokes the diagnostics trap handler and is automatically cleared on the next NEW_IPTR_LOADED that IN_TRAP_LOCK is not set. The return from the trap handler.
- [13:10] **BYTE_ENABLES**: Selects which bytes of host memory access are to be written/read. Host accesses are only to the bytes of the word which are specified in this field. To do a part word access, this field must have only the required bits set. To do a full word host access, all bits must be set.
- [9] **DEVICE_ACCESS**: Causes DCU memory accesses to be device accesses (to bypass the cache).
- [8] **DISABLE_GROUPING**: Disables the instruction fetch in the CPU from grouping instructions.
- [7] **TARGET_PEEK_ENABLE**
Enables reads to the hosted memory area to be translated to peeks to the host when not executing the diagnostics trap handler (always enabled while IN_TRAP is high). While not enabled, reads from hosted addresses return undefined data.
- [6] **TARGET_POKE_ENABLE**
Enables writes to the hosted memory area to be translated to pokes to the host when not executing the diagnostics trap handler (always enabled while IN_TRAP is high). While not enabled, writes to hosted addresses are ignored.
- [5] **WRITE_LOCK**
When set, CPU write accesses to the DCU register space have no effect except when the CPU is executing the diagnostics trap handler (while IN_TRAP is high).
- [4] **HOSTED_BUSY**: High if the host interface is in use or when the host is not connected. Indicates that an access to hosted memory would cause the CPU to hang and wait for the hosted memory interface to become available. No behavior on write to this bit.
- [3] **TRIGGER**: software input to the signalling scheme (see *DCU_SIGNALLING* on page 213). This can be used for sending trigger messages to the host or via TRIGGER_OUT.
- [2] **SINGLE_STEP**
While high, the DCU requests a TRAP_AT_NEXT_INSTRUCTION for every INSTRUCTION_STARTED or NEW_IPTR_LOADED seen (unless IN_TRAP is set), causing a hardware single instruction step. Can be disabled by WRANGE enable.
- [1] **CPU_TRAP**
A write of 1 to this bit causes the CPU to take a diagnostics trap at the next interrupt point. This bit automatically clears itself immediately.
- [0] **CPU_STALL**: Stalls CPU at the next interrupt point and remain stalled until this bit is cleared. The reset value depends on the signal HOST_CONNECTED, so if a host is connected at the end of reset, this is high and stalls the CPU for a boot from DCU otherwise it resets low and so does not stall the CPU as it comes out of reset.

DCU_SIGNALLING

Signalling 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		WIDTH								POLARITY	SIGNAL_VIA_TRIGGER_OUT		SIGNAL_VIA_TAP		REPEATABLE		TRIGGERED		TRIGGER_ON_ILLEGAL_ACCESS		TRIGGER_ON_MONITOR														

Address: *DCUBaseAddress* + 0x008

Type: Read/write

Reset: 0

Description: The trigger on inputs of the signalling register consist of the monitor inputs which are enabled, trigger from TRIGGER_IN block, jump trace full, compare matched or software trigger from the control register.

[31] **Reserved**

[30:23] **WIDTH**

This field specifies the number of extra clock cycles to hold TRIGGEROUT high when it is being used to generate pulses (0: Single cycle pulse).

[22] **POLARITY**

If this bit is high, TRIGGEROUT is inverted (active low).

[21:20] **SIGNAL_VIA_TRIGGER_OUT**

00: Don't signal via TRIGGEROUT.

01: TRIGGEROUT pulses high for one cycle on any new TRIGGER_ON condition.

10: TRIGGEROUT is a level, the OR of any enabled conditions.

11: TRIGGEROUT is the same value as TRIGGERED.

[19] **SIGNAL_VIA_TAP**

If set, then rising edge of the triggered bit causes a triggered message to be sent to the host.

[18] **REPEATABLE**

If this is set then triggered is automatically cleared the cycle after it is set, otherwise it remains set (that is, for multiple triggers rather than one shot).

[17] **TRIGGERED**

Reflects that one of the TRIGGER_ON events that signalling is sensitive to has occurred.

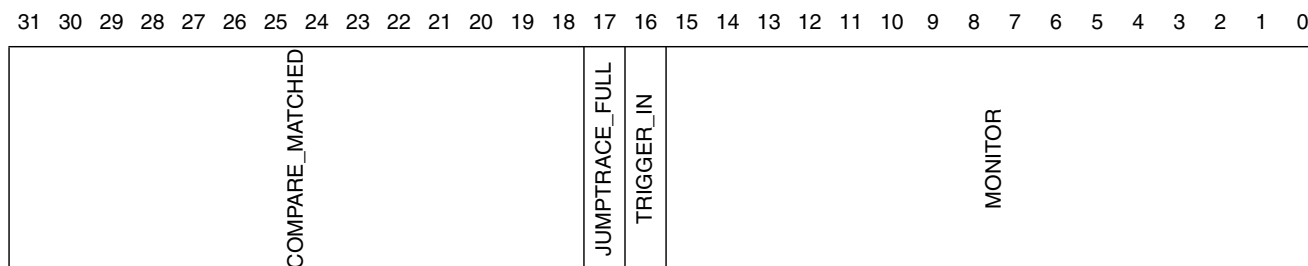
This gets set when any of the TRIGGER_ON inputs goes high. If repeatable is set then it is cleared automatically the next cycle, otherwise it remains set until it is cleared by a write of 0 to this register bit (a write of 1 is ignored).

[16] **TRIGGER_ON_ILLEGAL_ACCESS**

If set, detects an access to registers or hosted memory when not enabled via the control register.

[15:0] **TRIGGER_ON_MONITOR**

This mask selects the monitor bits to which signalling is sensitive. Triggered becomes set following the rising of any monitor for which the mask value is set.

DCU_STATUS**Status register**

Address: *DCUBaseAddress + 0x00C*

Type: Read only

Reset: 0

Description: The TRIGGER_IN, JUMPTRACE_FULL and COMPARE_MATCHED bits of the status register are identical to the corresponding TRIGGERED, FULL and MATCHED bits in the TRIGGER_IN, jump trace and compare properties registers. They are included in this status register to simplify polling the DCU to find the cause of a trap or stall. For typical implementations of the DCU this provides enough information. For implementations where there are more than one TRIGGER_IN block or more than 14 compare blocks, TRIGGER_IN status and compare status are provided as all of these bits don't fit in the status register.

[31:18] **COMPARE_MATCHED** [*n*]

Reflects the value of each of the match bits in the compare blocks. The number of bits depends on how many compare blocks there are.

[17] **JUMPTRACE_FULL**

High if JUMPTRACE_FULL bit is set.

[16] **TRIGGER_IN**

Value of TRIGGERIN[0] input.

[15:0] **MONITOR**

Value of bits on monitor input.

Bits 3:0 contain a decoded CPU run state, which is different between the ST20-C1 and ST20-C2. The idle encoding is the same

MONITOR[0]: C1 = user

C2 = booting

MONITOR[1]: C1 = trap

C2 = halted

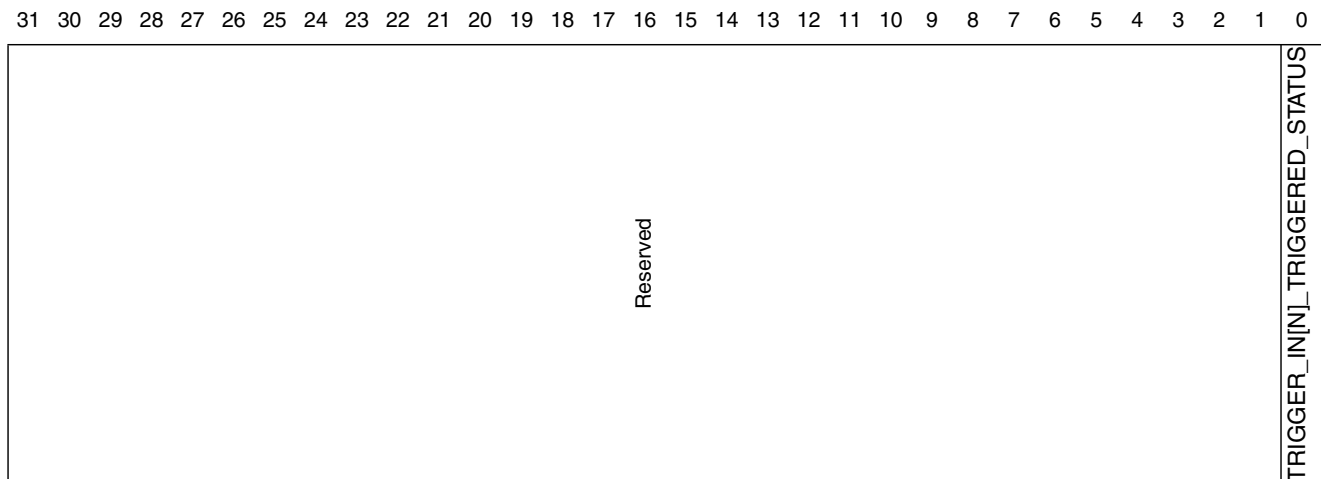
MONITOR[2]: C1 = idle

C2 = idle

MONITOR[3]: C1 = interrupt

C2 = running

Bits 15:4 are general purpose and product specific.

DCU_TRIGGER_IN**Trigger in status register**

Address: *DCUBaseAddress* + 0x010

Type: Read only

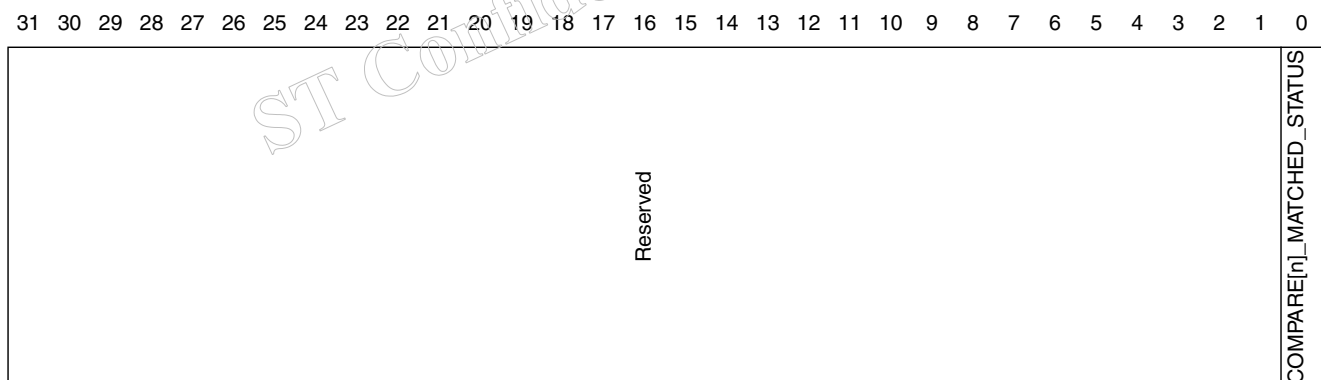
Reset: 0

Description:

[31:1] **Reserved**

[0] **TRIGGER_IN[n]_TRIGGERED_STATUS**

Reflects the value of the triggered bits of all the TRIGGER_IN blocks.

DCU_COMPARE_STATUS Compare status register

Address: *DCUBaseAddress* + 0x014

Type: Read only

Reset: 0

Description:

[31:1] **Reserved**

[0] **COMPARE[n]_MATCHED_STATUS**

Reflects the value of the matched bits of all the compare blocks.

DCU_SEQUENCING_CONFIGURATION Sequencing configuration 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								COMPARE_EVENTS				JUMPTRACE_EVENTS	TRIGGER_IN_EVENTS				CAPTURE				COMPARE				JUMPTRACE	TRIGGER_IN					

Address: *DCUBaseAddress* + 0x040

Type: Read only

Reset: 0

Description: The sequencing configuration register provides information about the hardware configuration of a particular implementation of the DCU to allow software to find the register and bit allocations for the sequencing and wrangle blocks.

[31:27] **Reserved**

[26:22] **COMPARE_EVENTS**

Number of compare blocks that contribute to events detected by the sequencing blocks.

Reset (depends on implementation): 3

[21] **JUMPTRACE_EVENTS**

Number of jump trace blocks that contribute to events detected by the sequencing blocks.

Reset (depends on implementation): 1

[20:16] **TRIGGER_IN_EVENTS**

Number of TRIGGER_IN blocks that contribute to events detected by the sequencing blocks.

Reset (depends on implementation): 1

[15:11] **CAPTURE**

Number of capture blocks that can be controlled by sequencing and work space range compare.

Reset (depends on implementation): 2

[10:6] **COMPARE**

Number of compare blocks that can be controlled by sequencing and work space range compare.

Reset (depends on implementation): 3

[5] **JUMPTRACE**

Number of jump trace blocks that can be controlled by sequencing and work space range compare.

Reset (depends on implementation): 1

[4:0] **TRIGGER_IN**

Number of TRIGGER_IN blocks that can be controlled by sequencing and work space range compare.

Reset (depends on implementation): 1

Confidential

DCU_TRIGGER_IN_PROPERTIES Trigger in properties 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																										TRIGGER_TYPE	TRIGGER_ON	TRAP_ON	STALL_ON	CAPTURED	ENABLE

Address: $DCUBaseAddress + (0x080 + 4 \times n)$

Type: Read/write

Reset: 0

Description:

[31:7] **Reserved**

[6:5] **TRIGGER_TYPE**

00: Trigger on high level

01: Trigger on low level

10: Trigger on rising edge

11: Trigger on falling edge

[4] **TRIGGER_ON**

If set, causes a trigger out when TRIGGER_IN is seen. Depends on setting of signalling register.

[3] **TRAP_ON**

If set, causes the CPU to take a diagnostics trap when TRIGGER_IN is seen.

[2] **STALL_ON**

If set, causes the CPU to stall when TRIGGER_IN is seen.

[1] **CAPTURED**

Gets set when TRIGGER_IN is seen. If STALL_ON or TRAP_ON are set then it stays set until cleared by the host or the CPU, otherwise it clears at the next cycle (a write of 1 is ignored).

[0] **ENABLE**

Enables this block

24.2 Jump trace registers

DCU_JUMPTTRACE_PROPERTIES Jump trace properties 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							MISSED_COUNT	VALID_BYTES	Reserved	DATA_TO_WRITE	WRAPPED	RESET_JUMPTTRACE	FLUSH	WRITE_ENABLE	INC_WHEN_STALLED	NONINTRUSIVE	STORE_ON_CAPTURE	STORE_ON_COMPARE	STORE_ON_CONTEXT	STORE_ON_IPTR	STORE_COUNT	STORE_CAPTURE	STORE_TO_IPTR	STORE_FROM_IPTR	TRIGGER_ON_FULL	TRAP_ON_FULL	STALL_ON_FULL	JUMPTTRACE_FULL	ENABLE		

Address: *DCUBaseAddress* + 0x100

Type: Read/write

Reset: See table for details

Description:

[31:25] **Reserved**

[24:22] **MISSED_COUNT (Read only)**

Contains the missed count field (used as bit 34:32 of the count when storing cycle counts).

Not hard reset, 000: Soft reset

[21] **VALID_BYTES (Read only)**

Indicates which bytes in the jump trace bytes register are valid. (11: 3 left, 10: 2 left, 01: 1 left, 00: None or 4 waiting to be written to memory.) Not hard reset, 00: Soft reset

[20] **Reserved**

[19] **DATA_TO_WRITE (Read only)**

Indicates that the jump trace has more complete words to write to memory. Not hard reset, 0: Soft reset

[18] **WRAPPED**

Gets set when the jump trace buffer becomes full. Remains set until cleared by a write to this register (a write of 1 is ignored). Not hard reset, 0: Soft reset

[17] **RESET_JUMPTTRACE**

Generates a soft reset for the jump trace block. Causes the cycle and missed count to reset to zero.

Resets the last values to zero and clears all buffering in the jump trace block. This bit must be written to prior to using jump trace. 0: Hard reset, 0: Soft reset

[16] **FLUSH**

Forces the jump trace to write any remaining complete words to the jump trace buffer, regardless of the state of the full bit. This is required to force a write out of data which is contained in internal buffering in the jump trace block when the trace buffer is full. 0: Hard reset, 0: Soft reset

[15] **WRITE_ENABLE**

Allows the jump trace block to do memory accesses. This is low following hard reset to prevent any data in the jump trace's internal buffering being written to memory. 0: Hard reset, 1: Soft reset

[14] **INC_WHEN_STALLED**

If high, the cycle count used by jump trace increments regardless of the state of the CPU, if low it only increments when the CPU is not stalled. 0: Hard reset, 0: Soft reset

[13] **NONINTRUSIVE**

If set, the DCU does not stall the CPU if there is no available memory bandwidth to store the jump trace information and so is nonintrusive but loses trace data. It records the fact that data has been lost in the jump trace buffer. 0: Hard reset, 0: Soft reset

[12] **STORE_ON_CAPTURE:** Trace on update of capture0 event. 0: Hard reset, 0: Soft reset

[11] **STORE_ON_COMPARE:** Trace on match of compare block 0 (used for general trace). 0: Hard reset, 0: Soft reset

[10] **STORE_ON_CONTEXT:** Trace on context change events (used for call profiling)
0: Hard reset, 0: Soft reset

[9] **STORE_ON_IPTR:** Trace on IPTR jump events (used for instruction jump trace)
0: Hard reset, 0: Soft reset

Confidential

- [8] **STORE_COUNT**: If set, causes a cycle count (and missed field count) to be stored in the jump trace.
Not hard reset, 0: Soft reset
- [7] **STORE_CAPTURE**: If set, causes the value in capture block 0 to be stored to the jump trace buffer
Not hard reset, 0: Soft reset
- [6] **STORE_TO_IPTR**: If set, causes the IPTR_IN jump trace IPTR register to be stored in the jump trace
Not hard reset, 0: Soft reset
- [5] **STORE_FROM_IPTR**: If set, causes the IPTR_IN jump trace from IPTR register to be stored in the jump trace.
Not hard reset, 0: Soft reset
- [4] **TRIGGER_ON_FULL**: Depends on setting of signalling register. 0: Hard reset, 0: Soft reset
- [3] **TRAP_ON_FULL**
If set, causes the CPU to take a diagnostics trap when the jump trace buffer becomes full.
0: Hard reset, 0: Soft reset
- [2] **STALL_ON_FULL**: If set, stalls the CPU while jump trace full is set. 0: Hard reset, 0: Soft reset
- [1] **JUMPTRACE_FULL**
Goes high to indicate that the jump trace buffer is full. If TRAP_ON_FULL or STALL_ON_FULL are set then it remains high until a write to this register, otherwise it automatically goes low in the next clock cycle (a write of 1 is ignored). Not hard reset, 0: Soft reset
- [0] **ENABLE**: Enables this block. 0: Hard reset, 0: Soft reset

DCU_JUMPTRACE_FROM_LPTR Jump trace from LPTR 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

JUMPTRACE_FROM																													

Address: *DCUBaseAddress* + 0x104

Type: Read only

Reset: Not reset

Description: IPTR before jump.

DCU_JUMPTRACE_TO_LPTR Jump trace to LPTR 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

JUMPTRACE_LPTR																													

Address: *DCUBaseAddress* + 0x108

Type: Read only

Reset: Not reset

Description: Contains the current value of IPTR.

DCU_JUMPTRACE_LAST_LPTR Jump trace last LPTR 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

JUMPTRACE_LAST_IPTR																													

Address: *DCUBaseAddress* + 0x10C

Type: Read only

Reset: No hard reset, 1 for soft reset

Description: Contains the value of the last IPTR written to the jump trace.

DCU_JUMPTRACE_LAST_CAPTURE0 Jump trace last capture0 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

JUMPTRACE_LAST_CAPTURE0

Address: *DCUBaseAddress* + 0x110

Type: Read only

Reset: No hard reset, 1 for soft reset

Description: Contains the value of the last capture0 written to the jump trace.

DCU_JUMPTRACE_BYTES Jump trace bytes 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

BYTE3

BYTE2

BYTE1

BYTE0

Address: *DCUBaseAddress* + 0x114

Type: Read only

Reset: No hard reset, 0 for soft reset

Description: Contains the value of the last IPTR written to the jump trace.

[31:24] **BYTE3**[23:16] **BYTE2** This is the byte buffer which contains incomplete words to write to the jump trace. The validity of these bytes is indicated by the state of the valid bytes field in the jump trace properties register.[15:8] **BYTE1**[7:0] **BYTE0****DCU_JUMPTRACE_START_ADDRESS** Jump trace start address 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

JUMPTRACE_START_ADDRESS

Res

Address: *DCUBaseAddress* + 0x118

Type: Read/write

Reset: No hard reset, 0 for soft reset

Description: Start of rolling jump trace buffer address.

DCU_JUMPTRACE_END_ADDRESS Jump trace end address 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

JUMPTRACE_END_ADDRESS

Res

Address: *DCUBaseAddress* + 0x11C

Type: Read/write

Reset: No hard reset, 0 for soft reset

Description: End of rolling jump trace buffer address, (jump trace buffer includes this address).

Confidential

DCU_JUMPTTRACE_ADDRESS Jump trace address 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	
JUMPTTRACE_ADDRESS																																Res

Address: *DCUBaseAddress* + 0x120
 Type: Read/write
 Reset: No hard reset, 0 for soft reset
 Description: Current rolling jump trace buffer address.

24.3 Compare registers**DCU_COMPAREn_PROPERTIES** Compare properties 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																READ_NOT_WRITE	WATCH_READS	WATCH_WRITES	BYTE_ENABLES				FUNCTION				TRIGGER_ON_MATCH	TRAP_ON_MATCH	STALL_ON_MATCH	MATCHED	ENABLE

Address: *DCUBaseAddress* + (0x200 + 0x10 x n)

Type: Read/write

Reset: See table

Description:

[31:15] **Reserved**

[14] **READ_NOT_WRITE**

While doing watchpoint compares, reflects whether the last access was a read (high) or write (low). Low for breakpoint compares. Reset: 0

[13] **WATCH_READS**: Set to enable detection of read accesses for watchpoints. Reset: 0

[12] **WATCH_WRITES**: Set to enable detection of write accesses for watchpoints. Reset: 0

[11:8] **BYTE_ENABLES**

Used during datawatch and counted watchpoint to select which bytes to compare. Reset: 1111

[7:5] **FUNCTION**

Selects which type of comparison function is to be performed.

000: Breakpoint

001: Counted breakpoint

010: Breakrange

011: Inverse breakrange

100: Datawatch

101: Counted watchpoint

110: Watchrange

111: Inverse watchrange

Reset: 000

[4] **TRIGGER_ON_MATCH**

Send a trigger message to host or via TRIGGEROUT when the compare matches. Depends on setting of signalling register. Reset: 0

[3] **TRAP_ON_MATCH**

Causes the CPU to take a diagnostics trap when a match occurs. Reset: 0

[2] **STALL_ON_MATCH**

Causes the CPU to stall when matched is set. Reset: 0

[1] **MATCHED**

Set when the comparison function matches. Remains set if TRAP or STALL are set, otherwise is automatically cleared on the next comparison (a write of 1 is ignored). Reset: 0

[0] **ENABLE**: Enables the compare block. Reset: 0

DCU_COMPAREn_VALUE1 Compare value1 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

VALUE1																													

Address: $DCUBaseAddress + (0x204 + 0x10 \times n)$

Type: Read/write

Reset: 0

Description: This register holds the first value which is used in the comparison operations.

DCU_COMPAREn_VALUE2 Compare value2 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

VALUE2																													
Reserved																COUNT													

Address: $DCUBaseAddress + (0x208 + 0x10 \times n)$

Type: Read/write

Reset: 0

Description: This register holds the first value which is used in the comparison operations.

[31:0] **VALUE2**

This register holds the second value which is used in the comparison operations.

[31:16] **Reserved**[15:0] **COUNT**

Used as a count value for the counted comparison operations.

Confidential

24.4 Capture registers

DCU_CAPTUREn_PROPERTIES Capture properties 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																REASON		SELECT		CAPTURED		ENABLE									

Address: $DCUBaseAddress + (0x400 + 0x08 \times n)$

Type: Read/write

Reset: 0

Description:

[31:5] **Reserved**

[4] **REASON (Read only):** If storing address or data, is high if the last access was a read, otherwise low.
Reset: 0

[3:2] **SELECT:** Selects which input bus value to capture.

00: IPTR

01: WPTR

10: Address

11: Data

[1] **CAPTURED**

Goes high to indicate that a value has been captured. Stays set until cleared by a write to this register (a write of 1 is ignored).

[0] **ENABLE:** Enables this block.

While enabled, the value on the selected bus is captured on every change. When enable is low the value remains constant. This can be used to capture a value on an event by using the sequencing block to disable this function on the event.

DCU_CAPTUREn_VALUE Capture 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
VALUE																															

Address: $DCUBaseAddress + (0x404 + 0x08 \times n)$

Type: Read only

Reset: Not reset

Description: Contains the value which has been captured.

24.5 Sequencing registers

DCU_SEQUENCING_ENABLE Sequencing enable 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																										ON_COMPARE2MATCHED	ON_COMPARE2MATCHED	ON_COMPARE2MATCHED	ON_SIGNALLINGTRIGGERED	ON_JUMPTRACE_FULL	ON_TRIGGER_IN

Address: *DCUBaseAddress* + (0x500 + 0x08 x n)

Type: Read/write

Reset: 0

Description:

[31:6] **Reserved**

[5] **ON_COMPARE2MATCHED**: When set, enables the feature when the matched bit of compare2 is set.

[4] **ON_COMPARE1MATCHED**: When set, enables the feature when the matched bit of compare1 is set.

[3] **ON_COMPARE0MATCHED**: When set, enables the feature when the matched bit of compare0 is set.

[2] **ON_SIGNALLINGTRIGGERED**: When set, enables the feature when the triggered bit in the signalling register is set.

[1] **ON_JUMPTRACE FULL**: When set, enables the feature when jump trace full is set.

[0] **ON_TRIGGER_IN**: When set, enables the feature on TRIGGER_IN.

Confidential

DCU_SEQUENCING_DISABLE Disable 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																								ON_COMPARE2MATCHED	ON_COMPARE1MATCHED	ON_COMPARE0MATCHED	ON_SIGNALLINGTRIGGERED	ON_JUMPTTRACE_FULL	ON_TRIGGER_IN

Address: $DCUBaseAddress + (0x504 + 8 \times n)$

Type: Read/write

Reset: 0

Description: The positioning of the bits in these registers is implementation dependant, for any implementation, the positioning can be found by examining the sequencing configuration register, see *DCU_SEQUENCING_CONFIGURATION* on page 216.

[31:6] Reserved[5] **ON_COMPARE2MATCHED:** When set, disables the feature when the matched bit of compare 2 is set.[4] **ON_COMPARE1MATCHED:** When set, disables the feature when the matched bit of compare 1 is set.[3] **ON_COMPARE0MATCHED:** When set, disables the feature when the matched bit of compare 0 is set.[2] **ON_SIGNALLINGTRIGGERED:** When set, disables the feature when the triggered bit in the signalling register is set.[1] **ON_JUMPTTRACE FULL:** When set, disables the feature when jump trace full is set.[0] **ON_TRIGGER_IN:** When set, disables the feature on TRIGGER_IN.**24.6 Work space range enable registers****DCU_WRANGE_ENABLE_ONLY_IN_RANGE Wrange enable only in-range 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																								CAPTURE1	CAPTURE0	COMPARE2	COMPARE1	COMPARE0	JUMPTTRACE	TRIGGER_IN	SINGLE_STEP

Address: $DCUBaseAddress + (0x600 + 0x10 \times n)$

Type: Read/write

Reset: 0

Description:

[31:8] Reserved[7] **CAPTURE1:** Only enable capture1 block if in range.[6] **CAPTURE0:** Only enable capture0 block if in range.[5] **COMPARE2:** Only enable compare2 block if in range.[4] **COMPARE1:** Only enable compare1 block if in range.[3] **COMPARE0:** Only enable compare0 block if in range.[2] **JUMPTTRACE:** Only enable jump trace block if in range.[1] **TRIGGER_IN:** Only enable trigger in block if in range.[0] **SINGLE_STEP:** Only enable single step if work space pointer is in range lower reg $\leq$ WPTR $<$ upper reg

DCU_WRANGE_ENABLE_ONLY_OUT_OF_RANGE**Wrange enable only out of range 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																								CAPTURE1	CAPTURE0	COMPARE2	COMPARE1	COMPARE0	JUMPTRACE	TRIGGER_IN	SINGLE_STEP

Address: $DCUBaseAddress + (0x604 + 0x10 \times n)$

Type: Read/write

Reset: 0

Description: The positioning of the bits in these registers is implementation dependant. For any implementation, the positioning can be found by examining the sequencing configuration register. See *DCU_SEQUENCING_CONFIGURATION* on page 216.

[31:8] **Reserved**

[7] **CAPTURE1**: Only enable capture1 block if not in range.

[6] **CAPTURE0**: Only enable capture0 block if not in range.

[5] **COMPARE2**: Only enable compare2 block if not in range.

[4] **COMPARE1**: Only enable compare1 block if not in range.

[3] **COMPARE0**: Only enable compare0 block if not in range.

[2] **JUMPTRACE**: Only enable jump trace block if not in range.

[1] **TRIGGER_IN**: Only enable trigger in block if not in range.

[0] **SINGLE_STEP**: Only enable single step if work space pointer is not in range (that is, WPTR < lower reg or WPTR ≥ upper reg).

DCU_WRANGE_LOWER Wrange lower 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
LOWER																															Res

Address: $DCUBaseAddress + (0x608 + 0x10 \times n)$

Type: Read/write

Reset: 0

Description: Contains the lower work space pointer value for the range comparison.

DCU_WRANGE_UPPER Wrange upper 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
UPPER																															Res

Address: $DCUBaseAddress + (0x60C + 0x10 \times n)$

Type: Read/write

Reset: 0

Description: Contains the upper work space pointer value for the range comparison.

25 Test access port

The STi5516 test access port (TAP) conforms to IEEE standard 1149.1.

The TAP has pins as listed in [Table 121](#). TDO can be overdriven to the power rails, and TCK can be stopped in either logic state.

Table 121: STi5516 TAP pins

Pin	In/out	Pull up/down	Description
TDI	In	Up	Test data input
TDO	Out		Test data output
TMS	In	Up	Test mode select
TCK	In		Test clock
NOT_TRST	In	Down	Test logic reset

The instruction register is 5 bits long with no parity. The pattern 00001 is loaded into the register during the Capture-IR state.

There are four defined public instructions, see [Table 122](#). All other instruction codes are reserved.

Table 122: Instruction codes

Instruction code ^a					Instruction	Selected register
0	0	0	0	0	extest	BOUNDARY_SCAN
0	0	0	1	0	idcode	IDENTIFICATION
0	0	0	1	1	sample/preload	BOUNDARY_SCAN
1	1	1	1	1	bypass	BYPASS

a. MSB... LSB; LSB closest to TDO.

26 Test access port registers

There are three test data registers; BYPASS, BOUNDARY_SCAN and IDENTIFICATION. These registers operate according to 1149.1. The operation of the BOUNDARY_SCAN register is defined in the BSDL description.

The identification code is 0xnD41 D041 where n is the 4-bit silicon revision number.

Table 123: Identification code

Bit 31						Bit 0 ^a	
Mask rev	ST20 family	Variant			STMicroelectronics manufacturers id		b
n	D	4	1	D	0	4	1

a. Closest to TDO

b. Defined as 1 in IEEE 1149.1 standard

27 Data flow

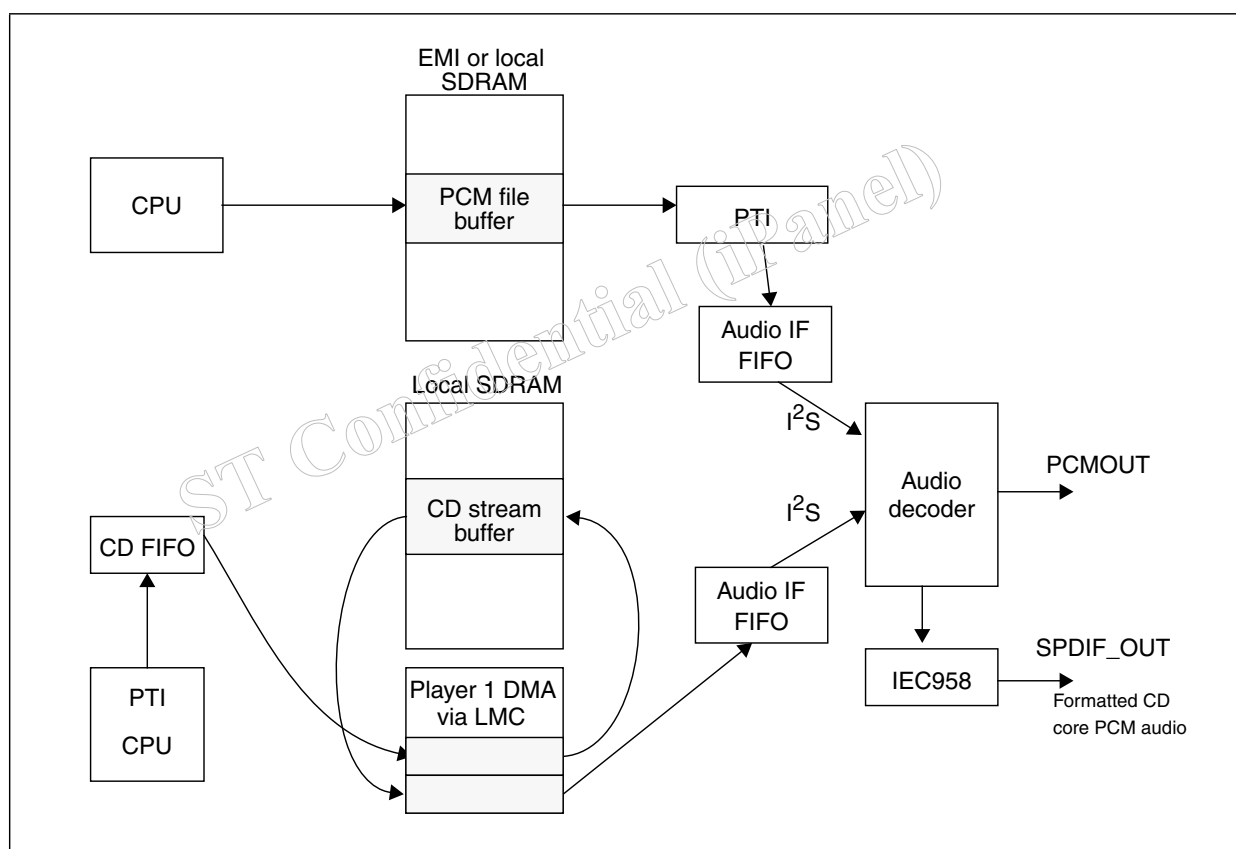
27.1 Overview

This chapter describes the normal data flow through the STi5516 from the incoming transport stream to the outgoing analog video and PCM audio. It details how the picture and sound modules are used together. The individual modules are described in the appropriate chapters.

27.2 Audio: PCM mixing

The audio core maintains two types of buffer in external memory, containing either compressed or PCM file data.

Figure 63: CD and PCM data flow



The audio subsystem supports multichannel audio decode and mixing with PCM files. This is completed in software, inside the audio decoder core.

27.2.1 Compressed data

Compressed data normally arrives at the CD FIFOs having been extracted from a transport stream by a PTI, the data may have been back buffered in external memory before being written by the PTI DMA engine (channels 1 to 3). This data is extracted from the FIFO by the player 1 DMA engine which sets up a CD stream buffer in the local SDRAM. A second player 1 DMA channel can play this buffer back to the audio decoder core.

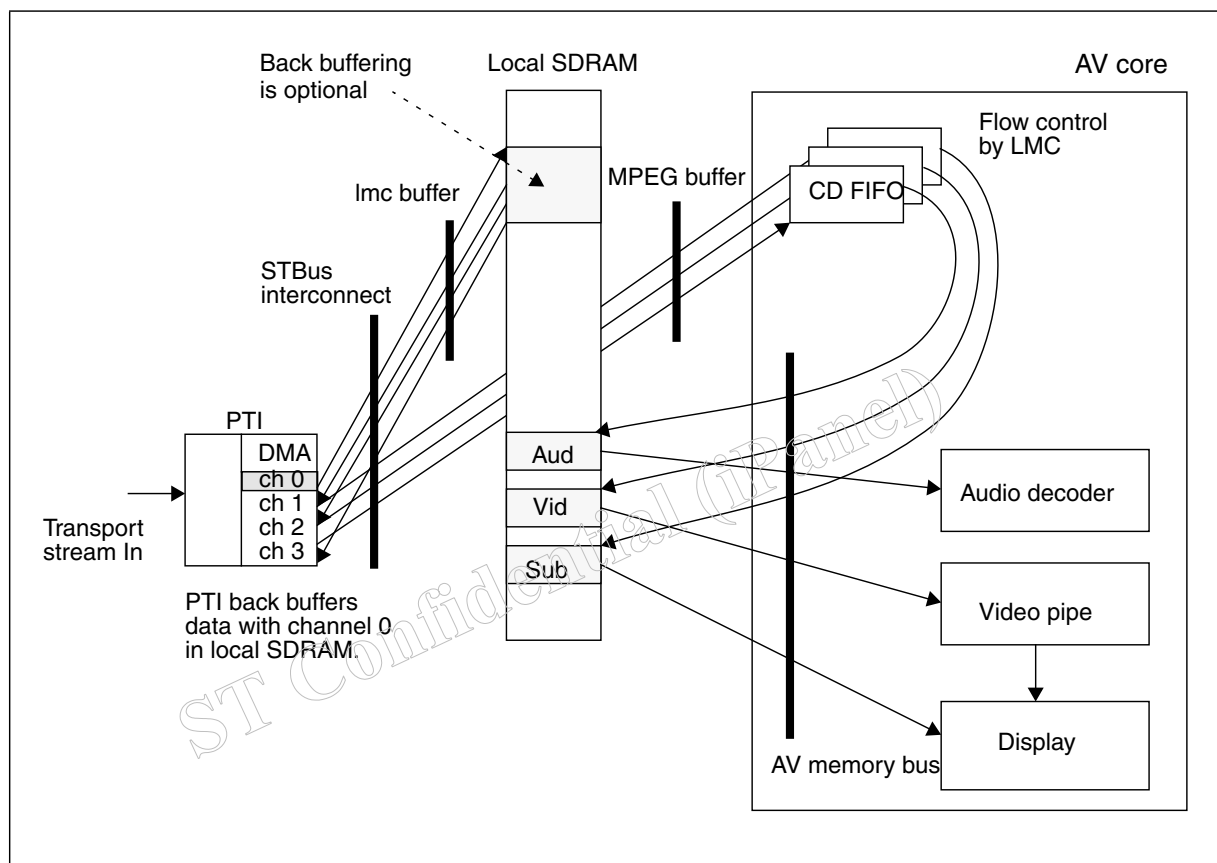
In addition it is possible for the CPU to transfer data directly into the CD stream buffer.

27.2.2 PCM files

A PCM file buffer may be located in external memory, either local SDRAM or EMI mapped SDRAM. The buffer is written from CPU initiators and played back by a PTI DMA channel which targets the audio decoder core via the audio interface.

27.3 Video: standard decode

Figure 64: Standard decode data flow



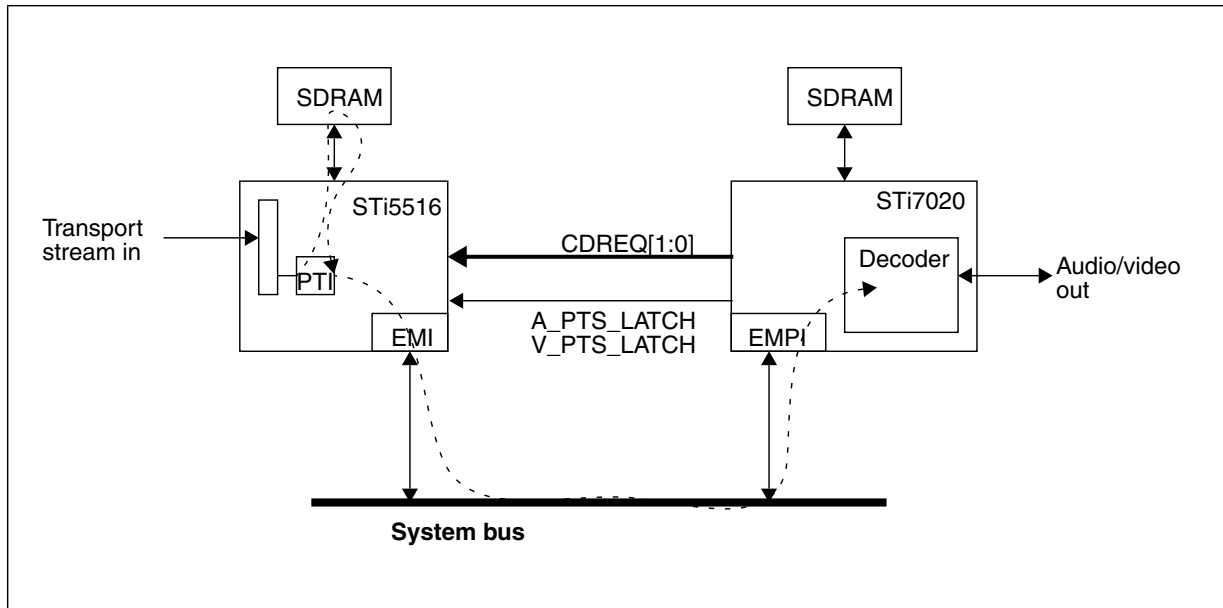
Transport stream packets are parsed by one of the PTI blocks, the transport stream is routed from the pins of the chip via the TSSUB/TSMUX block. The PTI channel 0 DMA engine is capable of performing four word burst (16 byte) accesses to the local SDRAM and can be used to back buffer the transport stream data into a circular buffer although this is optional.

PTI channels 1 to 3 can perform single word read and write accesses (no burst support) and read the channel 0 buffer and write to the CD FIFO channels of the AV core that buffer audio, video and subtitle information.

The CD FIFO function is implemented by the local memory controller (LMC) of the AV core which sets up circular buffers; again these are held in the local SDRAM.

When paired with a STi7020 high definition decoder the STi5516 embedded (SD) core is bypassed and the buffered PTI data is DMAed to the target device via the EMI.

Figure 65: Two chip HD application data flow



Confidential

ST Confidential (iPanel)

28 Programmable transport interface (PTI)

28.1 Overview

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

The PTI maintains an internal system time clock timer (STC) which keeps track of the encoder clock. The STC is clocked off the external 27 MHz clock which comes out of the set-top box VCXO (internal or external). The set-top box clock recovery software drives the VCXO in such a way that its output clock is locked on to the encoder's clock.

The ST20 sets up the PTI and loads its program through the type 1 interface. On reception of a new transport packet the PTI writes its content to the ST20 memory space through the type 2 interface.

The signals A_PTS_LATCH and V_PTS_LATCH come from the internal or external video decoder (depending on mode, standalone SD or companion STi7020 HD decoder) and are routed to the PTI in the STi5516. These two strobes are synchronous to the external 27 MHz clock. When the internal or external AV decoder starts displaying an audio or video frame it asserts A_PTS_LATCH or V_PTS_LATCH high for one cycle of the external 27 MHz clock. The PTI uses this signal to latch its local system clock into the AVPTS registers.

The three CDREQ signals are used as pacing signals for the PTI DMAs.

Figure 66: PTI inputs and outputs

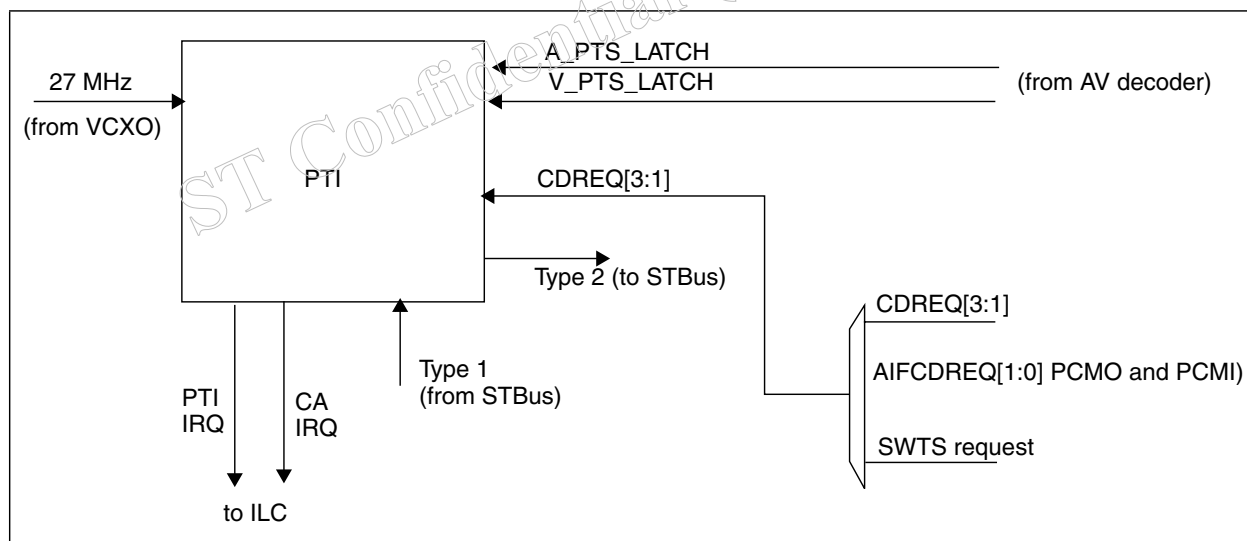


Table 124: PTI services

Service	Provider	Comment
Clock	Clock generator (PLL channel 1)	Operates from STBUS_CLK (100 MHz)
Clock_27	External VCXO	Nominally 27 MHz
Interrupts x 2	Comms/ILC	PTI and associated CA device.
Pacing	CD FIFOs	CDREQ signals from CD FIFOs to PTI/DMA, CDREQ[3:1] External and internal CDREQs are MUXed together with audio interface requests and the SWTS request to allow the PTI DMA to be used as a more general purpose DMA engine.
T1 target	STBus	Register configuration
T2 initiator	STBus	PTI/DMA target
Reset	C201	Global system asynchronous reset
Test	TAP	Standard scan, BIST for embedded FIFO and SRAM structures.

The PTI module parses, descrambles and demultiplexes the transport stream, using a mixture of hardware and software running on an application specific processor called the transport controller (TC). The TC gives the PTI the level of flexibility normally associated with software based demultiplexing of transport streams without the overhead of this processing being placed on the ST20 CPU.

The PTI is configured by registers and programmed by two blocks of static shared memory contained within the PTI, one block containing instructions and the other data. The data block contains structures shared with the ST20 CPU plus structures private to the TC. Code for the TC is downloaded into the PTI instruction memory by a PTI software driver running on the ST20.

The functionality of the PTI is therefore defined by a combination of the PTI hardware, the software running on the TC, and the software driver running on the ST20. This arrangement allows great flexibility by changing the code to be run. Many parameters of the code are modified to change the behavior and features of the PTI. The TC code and PTI driver software are provided by STMicroelectronics. Different versions of these software components are available, with support for generic MPEG-2/DVB transport stream parsing, descrambling and demultiplexing.

Specific details of the data structures and mechanisms used to communicate between the TC and the PTI driver running on the ST20 are contained in the documentation for these software components.

PTI operation is controlled by software supplied by STMicroelectronics. An API is available for those who wish to modify the software or create new applications.

The remainder of this chapter is a description of the hardware components of the PTI and the features and operation implemented by STMicroelectronics software.

28.2 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 MPEG2 transport streams with a data rate of 120 Mbit/s or greater,
- framing of transport packets (sync byte detection),
- descrambling to DVB standard - transport or PES level,
- section filtering up to 64x8 bytes or 32x16 bytes using four programmable filtering modes,
- CRC checking of sections (CRC32) and PES data (CRC16),
- DMA and buffering of streams in circular buffers in memory.

This behavior is changed with TC software:

- DMA of three data streams to audio and video MPEG decoders via buffers in memory,
- DMA of one data stream directly to decoders,
- DMA support for block moves,
- fast search for packet ID (PID) using dedicated hardware engine,
- time stamp checking in two formats.

Software extends the hardware's capability:

- PID filtering of more than 48 PIDs,
- eight descrambling key pairs per PTI,
- adaptation field parsing - PCR detection and time stamping,
- special purpose section filters allowing total flexibility in processing transport streams,
- demultiplexing of transport stream by PID, supported by hardware,
- communication to ST20 CPU of buffer state.

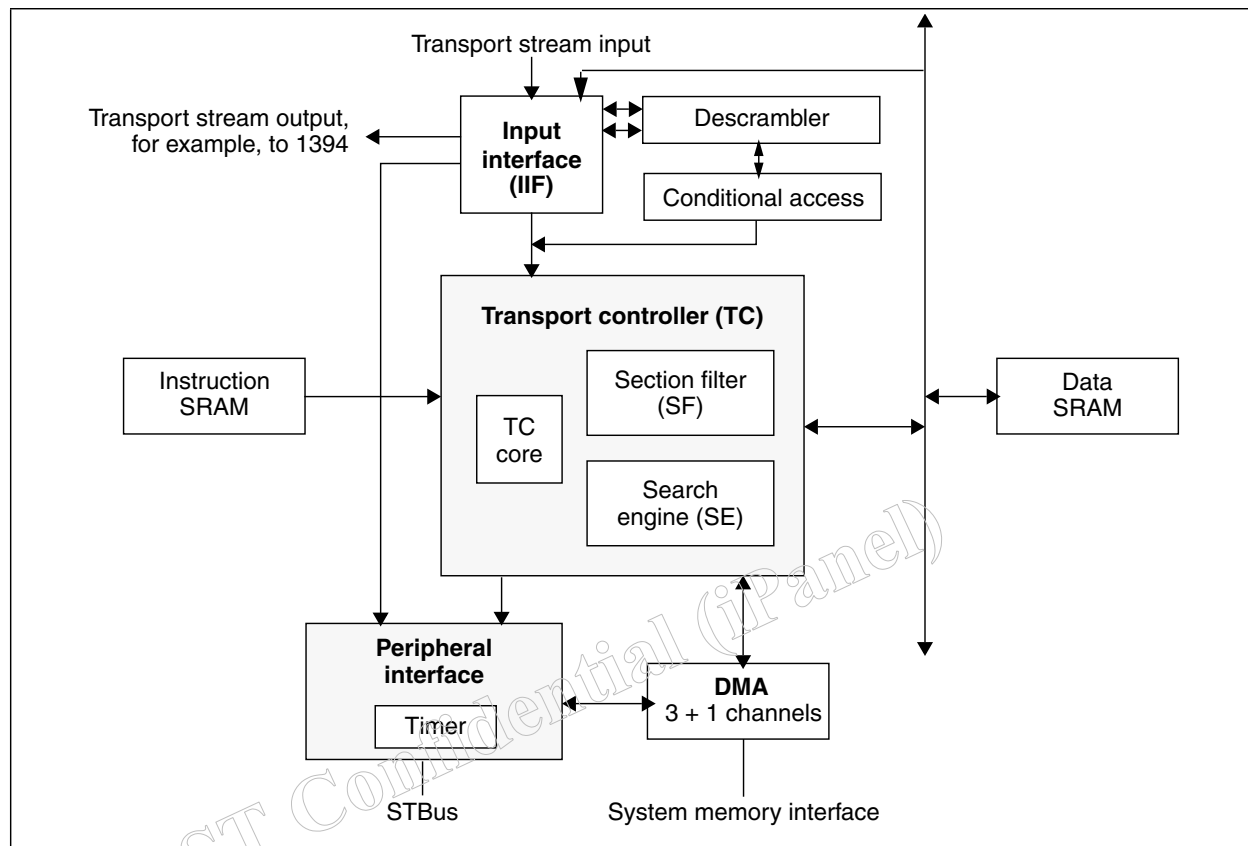
In addition to these transport device functions, the interface copies the entire transport stream or transport packets with selected PIDs from the transport stream through to the PTI output stream interface. Different PID groups or packet elements are tagged using packet tag signals which are output with the transport stream. This allows the input or output of transport streams using other interface standards, for example to provide a bidirectional transport stream interface to an external IEEE 1394 AV link layer device connected to the shared IEEE 1284/1394 and transport stream I/O pins.

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.

28.3 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 67](#).

Figure 67: PTI architecture



28.3.1 Transport controller (TC)

The TC incorporates the TC core which implements the transport deMUX software. The TC core is a simple 16-bit RISC CPU which uses hardware accelerators to implement PID searching (SE) and section filtering (SF). There is also hardware support for CRC checking.

The TC is responsible for parsing transport packets delivered via the IIF, and controls the entire PTI operation via memory mapped registers. Processed transport data is passed out from the TC to the dedicated DMAs where it is written to ST20 memory via the STBus.

Search engine (SE)

The SE performs a table look-up on the PID to determine whether the current packet is required. It supports searching on up to 48 PIDs and also implements range-checking.

Section filter (SF)

The SF is a hardware accelerator block dedicated to the section filtering task. It supports a number of standard section filtering schemes (short matching mode, long matching mode, MAC mode, positive negative mode) and any software based filtering scheme programmed using the PTI_SFFILTERMASK, PTI_SFFILTERDATA and PTI_SFNOTMATCH registers. Standard filtering is implemented using a pair of CAMs.

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.

28.3.2 Shared memory

The PTI contains 4 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.

28.3.3 Input interface (IIF)

The IIF provides the TC with a stream of descrambled transport packets for parsing. It also allows data to be copied directly to an output port, for example, one supporting the IEEE 1394 protocol.

The IIF is responsible for inputting the synchronous data stream to the PTI and passing data to the TC for processing. The start of a packet is detected either from the incoming packet clock or by sync byte detection. Under the control of TC software, the IIF routes data via the descrambler and conditional access modules to the TC input register via a special FIFO called the H- or Header FIFO. The H-FIFO, which sits between the IIF and the TC, helps to decouple descrambling from section filtering and prevents the descrambler from stalling if section filtering is late. This improves the overall performance of the system.

In order to allow time for transport data to be processed within the TC, the IIF sets a fixed time period between packet input and data output on the alternate output port. This is known as latency and is programmable in a register.

Descrambling is performed by the descrambler and conditional access modules, under the control of the TC. For each packet to be descrambled the TC loads a key into the descrambler and then requests a number of bytes. Descrambling schemes supported are:

- DVB,
- DES (DSS),
- NDS proprietary.

The IIF, like other DMA modules, is controlled by registers. Parameters programmed in these registers include:

- latency period,
- operating conditions for sync byte detection,
- packet length up to 256 bytes long (default 188),
- data transfer parameters, for example, number of bytes to be passed to the TC or whether descrambling is required,
- alternate output parameters.

28.3.4 DMA

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

There are four DMA channels. One channel (channel 0) is used for writing data to ST20 memory under the control of the TC, so that data is sorted into a number of circular buffers. Channels 1 to 3 are dedicated to managing these buffers; data is written directly from these buffers to the video and audio decoders under the control of signals generated by the decoder FIFOs. These three channels are generic DMA engines controlled by the ST20 or TC. A programmable delay (the holdoff time) is built into the data transfer to ensure all data has reached the decoder before the FIFO signal is read for the next transfer. Also, a programmable write length ensures that all data has storage room in the target upon arrival.

Channel 0 inputs single words or 4-word bursts. It may also be configured to output data directly to the decoders.

Channels 1 to 3 are normally set to write whole words but are configured to write to byte-wide devices.

The DMA block is controlled by a number of registers which are programmed by the TC software. Functions controlled by these registers include:

- setting up and manipulating the circular data buffers,
- configuring write channel (channel 0) operation,
- configuring channels 1 to 3 to write to the decoders (or memory),
- channel 0 status,
- memory map selection (programming mode).

28.3.5 Peripheral interface

The peripheral interface allows the PTI to interact with the ST20 and communicate data between them. It handles all address decoding (for example, DMA writes and interrupts) and allows joint access to the instruction and data SRAMs and configuration registers. It also implements the timer and time stamp functions and soft reset (see [Section 28.4.2: Soft reset on page 238](#)).

28.3.6 Timer module

The timer module in the PER contains the system time clock (STC) and three time stamp registers:

- packet start time register,
- audio PTS (presentation time stamp) latch register,
- video PTS latch register.

These registers are loaded with the STC value according to the events shown in [Table 125](#).

Table 125: 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.

28.4 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.

28.4.1 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 IIFIFOENABLE register bit must be set high to enable the input FIFO.

28.4.2 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.

28.4.3 Typical operation

When the TC is running, the software waits for a transport packet to arrive. Having detected the start of a packet, the TC code then sets up the PTI hardware search engine to perform PID filtering. The search engine searches a contiguous block of PID values within data SRAM for a match with the PID in the incoming transport packet header. If the packet is to be rejected, the software discards the packet and waits for the next packet to arrive. If the packet is one to be processed, the TC examines the data structure for the PID in the data SRAM to determine what other processing is required. The type of transport packet is recorded in this data structure by the ST20. The PTI driver sets variables in this data structure to configure the TC software to perform various tasks.

Typical tasks might be:

- descrambling at the transport or the PES level with a key pair,
- section filtering, with a set of filters and section CRC checking for streams containing sections,
- directing the output stream either to a circular buffer in memory or to a compressed data FIFO of an audio or video decoder,
- enabling or disabling a stream,
- appending or indexing extra information for further data processing by the ST20.

Having examined the PID data structures, the TC sets up the rest of the hardware in the PTI to perform the required descrambling and DMA operations before starting to parse the rest of the packet. Processing varies depending on the contents of the transport packet, which includes:

- PES data,
- section data,
- adaptation fields,
- continuity count fields,
- time stamp.

Typical processing for different packet types and fields is described in the rest of this section.

PES data

Transport packets which contain PES data and are not rejected by PID filtering, are CRC checked and descrambled if required. The PES data is DMA transferred either into a circular buffer or to a decoder compressed data FIFO. The DMA features of the PTI buffer a PES stream in memory and then transfer the data to a decoder without the CPU being involved. Optionally an interrupt is generated to the ST20 when the buffer for a PES stream has data added to it and the state of the buffer changes from empty to nonempty. An interrupt is raised and an error flag set in the data SRAM if the buffer overflows. In such cases, the most recent data is lost.

Section data

Transport packets which contain section data and are not rejected by PID filtering, are descrambled if required and 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): 64 filters of 8 bytes each,
- long match mode (LMM): 32 filters of 16 bytes each.

MAC match mode (MMM) and positive/negative matching mode are 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 is also 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.

28.4.4 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.

Groups of PIDs, or parts of transport packets, are flagged using packet tag signals which are output by the PTI along with the transport stream. These signals are controlled by the TC software and are updated for each byte output.

28.5 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 68 shows the TC and the ST20 processes and mechanism described above.

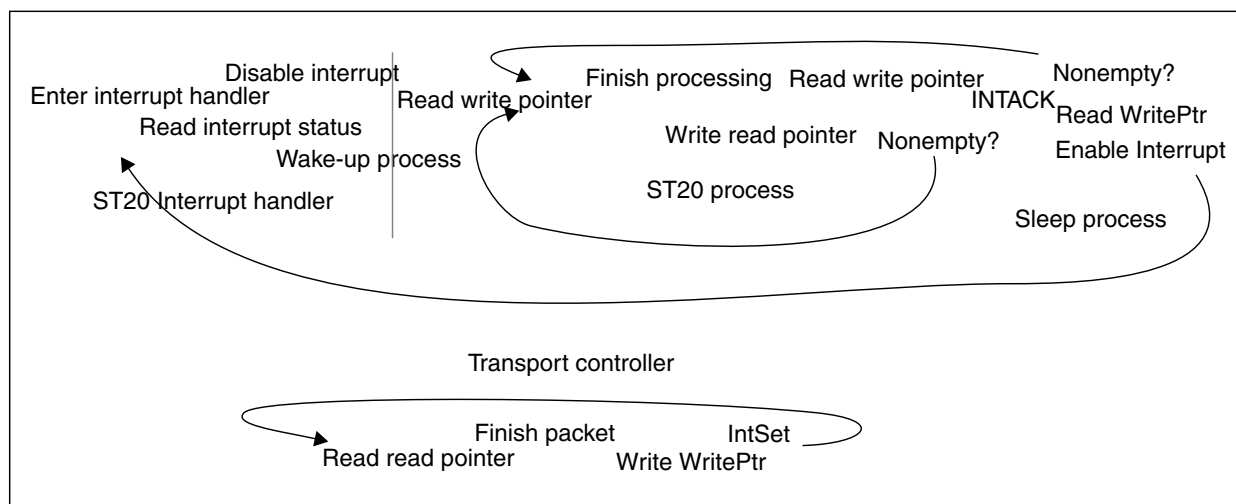
Note: *At any given time each process is at any point during the critical regions of code. There is no implied timing for each step of a process only an ordering of steps.*

After the buffer has been refilled the TC sets the interrupt status bit causing the PTI interrupt handler to be run. When the interrupt handler finds the buffer process semaphore status bit is set then the interrupt handler signals to the semaphore to restart the process and disable that interrupt bit. Therefore, the process itself disables the interrupt at the PTI level, and only enables it when it is about to sleep.

An error condition would be handled in a similar manner.

The association of interrupt bits with particular conditions and events is determined by the TC code and the corresponding PTI driver running on the ST20 CPU.

Figure 68: PTI buffer interrupt handling



28.6 DMA operation

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

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

Channel 0 is configured to write data directly to the decoders by specifying the address to be written to in the appropriate buffer registers. When using this feature channel 0 should be set to holdoff mode.

It is also possible to configure channel 0 to ignore the decoder ready signal. This feature should be used with caution as it allows data to be written without flow control.

Channels 1 to 3 are used to read from circular buffers and write the data to a device such as an audio or video decoder mapped to a fixed address in ST20 memory, in response to a request signal. Each time the request signal is active (low) and there is data to be read from the buffer (that is, the read pointer is not equal to the write pointer), a programmable number of bytes is transferred. This is followed by a hold-off time during which the request is not sampled, allowing the request signal time to become valid again.

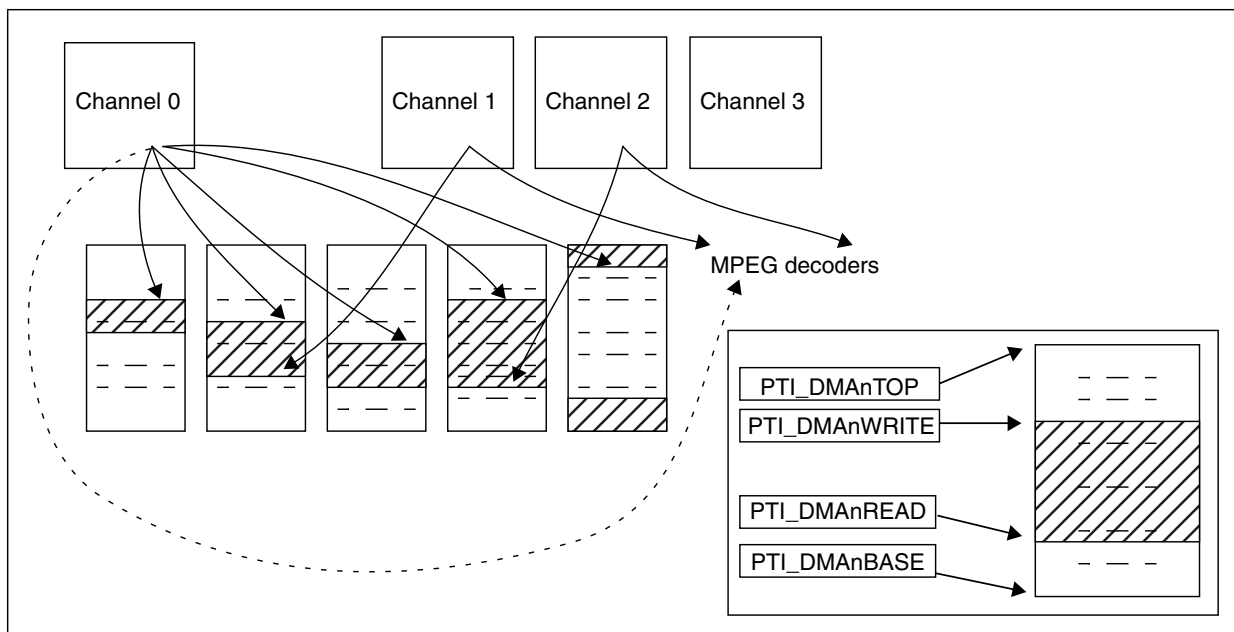
Block moves are supported on channels 1 to 3 using a special DMA mechanism which increments the input address after each write.

During operation the read and write pointers are examined by the hardware to determine if there is data in the buffers to be transferred. If there is data in the buffer and the request line for that channel is active then the data is transferred and the read pointer updated. If channel 0 DMA is writing into the buffer then the write pointer is updated by the TC; otherwise the ST20 CPU updates the write pointer after adding data to a buffer.

At the end of inputting each transport packet via channel 0, the DMA write pointer of the corresponding buffer data structure in the PTI data SRAM is updated. If the transport packet did not contain the end of a complete data unit such as a section, a temporary write pointer variable is used. This is done so that the ST20 process only sees a complete unit of data to be processed. The temporary write pointer is available for reading by the TC software in a special register. When the data unit is complete, the write pointer used by the ST20 process is updated and an interrupt is set to signal to the ST20 process that data is in that buffer. This mechanism of updating the write pointers and interrupting, is not used in the special case that the buffers are being transferred by DMA to an audio, video or other decoder.

Channels 1 to 3 have the write pointers updated either by the TC software after data has been placed in the corresponding buffer by DMA channel 0, or, by the ST20 CPU if this is writing data into the buffer that DMA channels 1 to 3 are reading from.

Figure 69: DMA channels



28.6.1 Circular buffers

The inset in Figure 69 shows how the buffer pointer registers are used to implement circular buffers for the four DMA channels. The base register points to the base word of the buffer, and must be 16-byte aligned, so bits 0 to 3 must be zero. The top register points to the top byte of the buffer. If a circular buffer is being used then this address must be one byte below a 16-byte aligned address, so bits 0 to 3 must be 1. The buffer for channel 0 only is reduced to a single address by setting the top register equal to the base register. In this case, the data is written to a fixed address defined by the write pointer and the write pointer is not updated.

The read and write buffers point to the next word to be read or written respectively.

At initialization the read and write pointers are set to the same value, so that the buffers are empty. The base and top pointers are initialized to point to the beginning and end of the buffers.

28.6.2 Channel arbitration

Only one of the four DMA channels may have access to the memory bus at any time. To ensure smooth flow of data and to avoid mutual lockout, there is a fair and efficient arbitration scheme between the four DMA channels.

The scheme used is the least recently used (LRU) method, that is, the channel that has not requested an access for the longest time is guaranteed the next access. This ensures that none of the channels locks out the others, and has the advantage that the LRU arbiter does not waste any clock cycles on an inactive channel.

Although channel 0 appears to have no priority over channels 1 to 3 to write incoming transport data, in fact the performance of channel 0 is enhanced because:

- it transfers data four times faster than channels 1 to 3 using 4-word bursts,
- it never performs read accesses, which are inherently slower because they cannot be posted ahead.

28.6.3 Flushing the DMA

Before doing a PTI soft reset (see Section 28.4.2: *Soft reset* on page 238), 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.

28.6.4 Performance

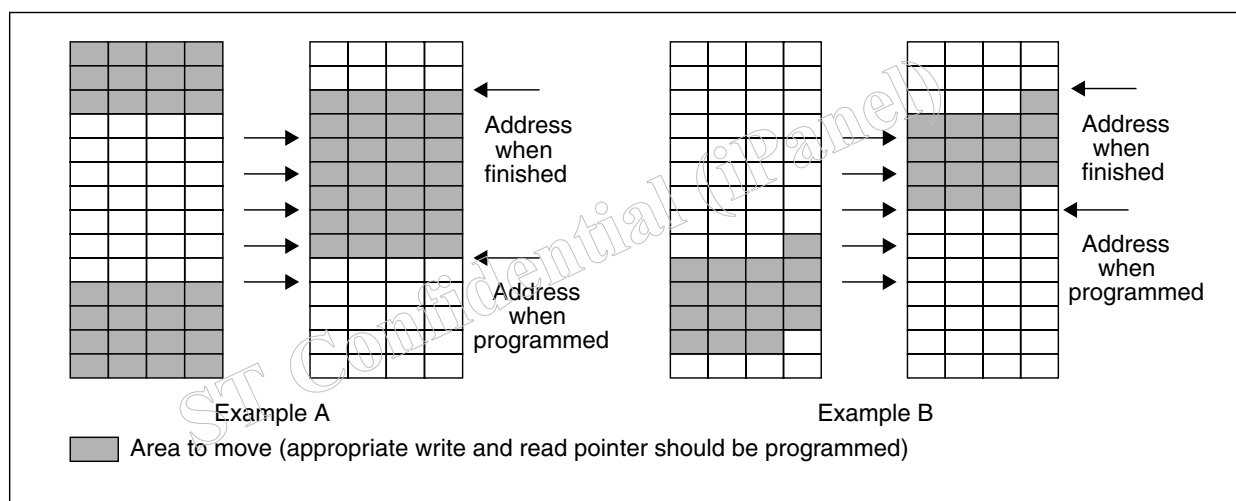
Each DMA channel performs one read, write or burst access when it has been granted by the memory arbiter. When more than one channel is active, one performs an access while another is waiting for valid pulses to come back from the interconnect. So the accesses are interleaved, guaranteeing a high performance.

Performance is enhanced by being able to post writes to the STBus that is, a read or write is initiated without waiting for a valid signal on the previous access. Reads cannot be posted since the next operation may be a write that depends on the result of the previous read.

28.6.5 Block move

In normal operation DMA channels 1 to 3 read from a circular buffer and write to a fixed address (the memory-mapped address of the decoder). Channels 1 to 3 are also configured to perform a block move by setting a bit in the channels status register. This causes the decoder address to be incremented after each write and effectively performs a circular- to- linear blockmove function. This is illustrated in Figure 70.

Figure 70: DMA blockmove



28.7 Section filter (SF)

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 specification¹.

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 64 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

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

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:

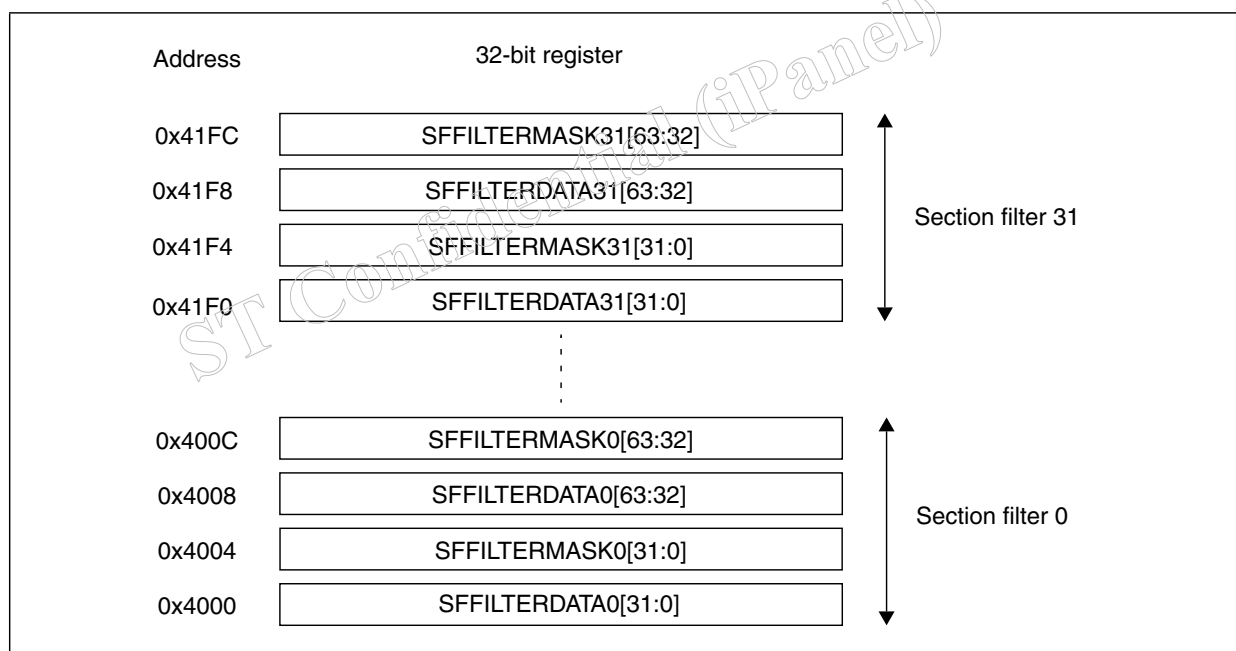
- all sections,
- no sections,
- only those sections that have the section syntax indicator bit set via bits in the PID data structure read by the TC code.

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.

The filter matching is programmable by means of a set of registers. Each 8-byte section filter entry is composed of four 32-bit words in memory, with each group of four words aligned on a 4-word boundary. Within the 4-word group, the section filter is composed of two 32-bit words dedicated to the storage of filter data, plus two 32-bit words dedicated to the storage of masking information. An overall view of the section filter data is shown in Figure 71, as it appears in the memory map. This memory map is for a single CAM. The second CAM is extended from 0x4200 to 0x43FC. If 8-byte schemes (SMM) are used the filters are independent. In 16-byte schemes the filters are an extension of those shown in Figure 71.

Figure 71: Section filter memory map



Only one word of a filter is updated at a time, so updates of the section filters should be carried out when one of the following is true:

- a false match does not matter,
- the filter is not in use for any PIDs,
- the PIDs that use the filter have been disabled.

The section filter registers are in a block in the peripheral space in the ST20 memory map. The address of the base of the block is called *SFBaseAddress*, which has the value given by:

$$SFBaseAddress = PTIBaseAddress + 0x4000$$

Filtering is performed by testing whether:

$$packet \text{ bit AND } FilterMask = FilterData \text{ bit AND } FilterMask$$

If this is true then the packet passes through the filter.

28.7.1 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 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.

28.7.2 Filtering modes

Standard filtering modes supported are:

- short match mode (SMM): matches on CAM A or CAM B,
- long match mode (LMM): matches on CAM A line n and CAM B line n,
- positive/negative matching mode: matches on CAM A line n and not CAM B line n,
- MAC match mode (MMM): matches on CAM A and CAM B.

Standard filtering is performed by means of two 64-bit wide CAMs (A and B), the outputs of which are ANDed to give the final result. CAM A has an additional register (PTI_SFNOTMATCH) containing five bits of data and one bit to enable not matching. CAM lines are enabled or disabled using mask registers. Almost any CAM-based filtering match over 18 bytes can be programmed. The mapping of data to CAM A and CAM B for each of these modes is shown in [Table 126](#) to [Table 129](#).

Table 126: SMM mapping

Section bytes	CAM A	CAM B
5[5:1] ^a	PTI_SFNOTMATCHn[4:0]	Not used
0	PTI_SFDATAAn[63:56]	PTI_SFDATAAn[63:56]
1	Not used	
2		
3	PTI_SFDATAAn[55:48]	PTI_SFDATAAn[55:48]
4	PTI_SFDATAAn[47:40]	PTI_SFDATAAn[47:40]
5	PTI_SFDATAAn[39:32]	PTI_SFDATAAn[39:32]
6	PTI_SFDATAAn[31:24]	PTI_SFDATAAn[31:24]
7	PTI_SFDATAAn[23:16]	PTI_SFDATAAn[23:16]

Table 126: SMM mapping

Section bytes	CAM A	CAM B
8	PTI_SFDATA _n [15:8]	PTI_SFDATA _n [15:8]
9	PTI_SFDATA _n [7:0]	PTI_SFDATA _n [7:0]
10	Not used	
11		
12		
13		
14		
15		
16		
17		

a. Used when bit EN_n of register PTI_SFNOTMATCH_n = 1

Table 127: LMM mapping

Section bytes	CAM A	CAM B
5[5:1] ^a	PTI_SFNOTMATCH _n [4:0]	Not used
0	PTI_SFDATA _n [63:56]	Not used
1	Not used	
2		
3	PTI_SFDATA _n [55:48]	Not used
4	PTI_SFDATA _n [47:40]	
5	PTI_SFDATA _n [39:32]	
6	PTI_SFDATA _n [31:24]	
7	PTI_SFDATA _n [23:16]	
8	PTI_SFDATA _n [15:8]	
9	PTI_SFDATA _n [7:0]	
10	Not used	PTI_SFDATA _n [63:56]
11		PTI_SFDATA _n [55:48]
12		PTI_SFDATA _n [47:40]
13		PTI_SFDATA _n [39:32]
14		PTI_SFDATA _n [31:24]
15		PTI_SFDATA _n [23:16]
16		PTI_SFDATA _n [15:8]
17		PTI_SFDATA _n [7:0]

a. Used when bit EN_n of register PTI_SFNOTMATCH_n = 1

Table 128: Positive/negative matching mode mapping

Section bytes	CAM A	CAM B
5[5:1] ^a	PTI_SFNOTMATCHn[4:0]	Not used
0	PTI_SFDATAAn[63:56]	PTI_SFDATAAn[63:56]
1	Not used	
2		
3	PTI_SFDATAAn[55:48]	PTI_SFDATAAn[55:48]
4	PTI_SFDATAAn[47:40]	PTI_SFDATAAn[47:40]
5	PTI_SFDATAAn[39:32]	PTI_SFDATAAn[39:32]
6	PTI_SFDATAAn[31:24]	PTI_SFDATAAn[31:24]
7	PTI_SFDATAAn[23:16]	PTI_SFDATAAn[23:16]
8	PTI_SFDATAAn[15:8]	PTI_SFDATAAn[15:8]
9	PTI_SFDATAAn[7:0]	PTI_SFDATAAn[7:0]
10	Not used	
11		
12		
13		
14		
15		
16		
17		

a. Used when bit ENn of register PTI_SFNOTMATCHn = 1

Table 129: MAC match mode mapping

Section bytes	CAM A	CAM B	
5[5:1] ^a	PTI_SFNOTMATCHn[4:0]	Not used	
0	PTI_SFDATAAn[63:56]	Not used	
1	Not used		
2			
3			PTI_SFDATAAn[63:56]
4			PTI_SFDATAAn[55:48]
5	PTI_SFDATAAn[55:48]	Not used	
6	PTI_SFDATAAn[47:40]		
7	PTI_SFDATAAn[39:32]		
8	Not used		
9		PTI_SFDATAAn[47:40]	
10		PTI_SFDATAAn[39:32]	
11		PTI_SFDATAAn[31:24]	
12	PTI_SFDATAAn[31:24]	Not used	
13	PTI_SFDATAAn[23:16]		
14	PTI_SFDATAAn[15:8]		
15	PTI_SFDATAAn[7:0]		
16	Not used	PTI_SFDATAAn[15:8]	
17		PTI_SFDATAAn[7:0]	

a. Used when bit ENn of register PTI_SFNOTMATCHn = 1

28.7.3 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.

28.7.4 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 matching results,
- CAM mask data,
- CAM not matching results,
- selection of memory map (programming mode).

28.8 Compatibility with PTI1

Address mappings for the DMA have changed from the PTI1. To allow software compatibility two programming modes have been defined.

PTI1 mode

PTI1 mode allows PTI1 code to run on the PTI3 without modification. PTI1 mode is the default on start up.

PTI3 mode

PTI3 mode uses the new memory map without transformation and allows access to the new features.

Compatibility is achieved by the following methods.

- **Using a register format which is a superset of the previous format**
PTI3 registers retain the same bit positions as in the PTI1, and any new bits are disabled on reset. Default values, where set, are also the same.
- **Defining an alternative memory map**
In PTI1 mode PTI1 addresses and many of the new PTI3 registers are remapped in a way which is transparent to the programmer.

Some of the new DMA registers can be accessed from PTI1 mode. CAM configuration does not need any change to make it compatible as the PTI1 had a single CAM memory and no version match registers. The first CAM holds the same address in PTI3.

Programming modes are selected for the DMA by setting register PTI_DMPTI3PROG to 1 for PTI3 mode and 0 for PTI1 mode (the default on reset). See [Table 49A: Programmable transport interface \(PTI\) registers: DMA](#) on page 74 for the differences in the register maps between PTI1 and PTI3 modes.

29 Programmable transport interface (PTI) registers

For compatibility between PTI1 and PTI3, two address maps are provided for the DMA registers. Input interface, PTI configuration and section filter registers remain the same for both PTI1 and PTI3.

Addresses are provided as the *PTIBaseAddress* + offset.

The *PTIBaseAddress* is:

0x2002 0000.

A register summary is given in [Table 49A: Programmable transport interface \(PTI\) registers: DMA on page 74](#) and [Table 49B: Programmable transport interface \(PTI\) registers: others on page 76](#).

29.1 DMA registers

PTI_DMA0STATUS

DMA channel 0 status

7	6	5	4	3	2	1	0
Reserved						DMA0OVERFLOW	Reserved

Address: *PTIBaseAddress* + 0x1040 (PTI1)

Address: *PTIBaseAddress* + 0x1018 (PTI3)

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.

[7:2] **Reserved**

[1] **DMA0OVERFLOW**

If 1, the channel 0 circular buffer overflowed. Reset by writing 1 to this bit.

[0] **Reserved**

Write 0. Read data undefined.

PTI_DMAnBASE
DMA buffer base address

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000/ 0x1000	DMA0BASE																															
0x1004/ 0x1020	DMA1BASE																															
0x1008/ 0x1040	DMA2BASE																															
0x100C/ 0x1060	DMA3BASE																															

Address: *PTIBaseAddress* + 0x1000, 0x1004, 0x1008 and 0x100C (PTI1)

Address: *PTIBaseAddress* + 0x1000, 0x1020, 0x1040 and 0x1060 (PTI3)

Type: Write only

Description: Each of these registers holds the base address of the DMA buffer for the corresponding DMA channel. This address must be aligned to a 16-byte boundary, so bits 3 to 0 must be written as 0. Bits 31 to 30 of this address must be the same as bits 31 to 30 of the corresponding PTI_DMAnTOP address register. The DMA channel 0 register would be set up for each PID in the PID data structure in the PTI data memory.

PTI_DMAnBURST
DMA byte or word transfers

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1044	Reserved																															DMA1BURST
0x1048	Reserved																															DMA2BURST
0x104C	Reserved																															DMA3BURST

Address: *PTIBaseAddress* + 0x1044 to 0x104C (PTI1 only)

Type: Write only

Description: The PTI_DMA[1:3]BURST registers determine the size of the DMA transfer that occurs each time the corresponding NOT_CDREQ[1:3] signal is low. 0 means write one word at a time. 1 means write one byte at a time.

Confidential

PTI_DMAnHOLDOFF

DMA hold off time

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1014	Reserved								DMA0WRITESIZE								Reserved								DMA0HOLDOFF							
0x1054	Reserved								DMA1WRITESIZE								Reserved								DMA1HOLDOFF							
0x1034	Reserved								DMA2WRITESIZE								Reserved								DMA2HOLDOFF							
0x1058	Reserved								DMA2WRITESIZE								Reserved								DMA2HOLDOFF							
0x1054	Reserved								DMA3WRITESIZE								Reserved								DMA3HOLDOFF							
0x105C	Reserved								DMA3WRITESIZE								Reserved								DMA3HOLDOFF							
0x1074	Reserved								DMA3WRITESIZE								Reserved								DMA3HOLDOFF							

Address: *PTIBaseAddress* + 0x1054, 0x1058 and 0x105C (PTI1)

Address: *PTIBaseAddress* + 0x1014, 0x1034, 0x1054 and 0x1074 (PTI3)

Type: Read/write

Reset: 0 (DMAnHOLDOFF)
4 (DMAnWRITESIZE)

Description: The PTI_DMA[1:3]HOLDOFF registers are used to specify the delay time between the end of a burst of data being transferred and resampling the NOT_CDREQ[1:3] signals before another transfer is started on a DMA channel. The time is in units of byte clock cycles in the range 0 to 31 cycles.

[31:24] **Reserved**

[23:16] **DMAnWRITESIZE**

Number of bytes that can be written by the channel before having to wait for all valid pulses to be returned.

[15:8] **Reserved**

[7:0] **DMAnHOLDOFF**

Number of clock cycles to count between receiving a valid pulse after writing to the CD FIFO, and assuming CD_REQ[n] is valid.

Confidential

PTI_DMA n READ

DMA buffer read address

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1030 0x100C	DMA0READ																															
0x1034 0x102C	DMA1READ																															
0x1038 0x104C	DMA2READ																															
0x103C 0x106C	DMA3READ																															

Address: $PTIBaseAddress + 0x1030$ to $0x103C$ (PTI1)

Address: $PTIBaseAddress + 0x100C$ to $0x106C$ (PTI3)

Type: Read/write except DMA0READ which is write-only

Reset: Undefined

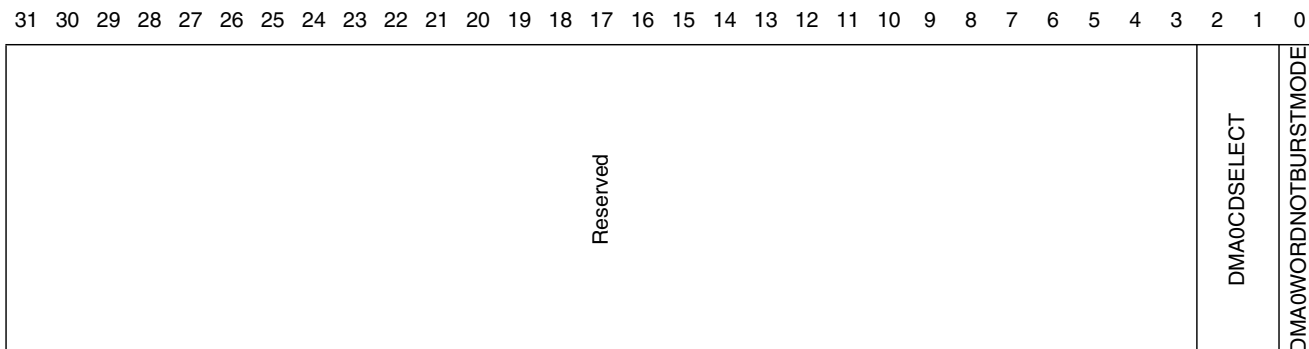
Description: Each of these registers holds the read address within the DMA buffer for the corresponding DMA channel. The address in the register is always a pointer to the next byte to be read except when the buffer is empty. The pointer must remain between the addresses defined by the base and top register. The channel 0 register would be initialized for each PID in the PID data structure in the PTI data shared memory, and an updated read pointer would be written into the same data structure. The read pointer is initialized to be equal to the write pointer. As data is read from the buffer, the hardware updates the read pointer. For channels 1 to 3, the read pointer is automatically wrapped round, and cannot overtake the write pointer. On reaching the top pointer address the read pointer wraps around to continue reading from the base address. If the read pointer is equal to the write pointer then no data is read. For channel 0, if the buffer is not being read by any of the DMA channels 1 to 3, then the ST20 software must perform the checks below.

- If the read address is not less than the top address, then the read address must be wrapped round to the base address.
- If the read address is equal to the write address then the buffer is empty and data should not be read.

Confidential

PTI_DMA0SETUP

DMA channel 0 byte not word mode and block move



Address: *PTIBaseAddress* + 0x1010 (PTI3 only)

Type: Read/write

Reset: 0

Description:

[31:3] **Reserved**

[2:1] **DMA0CDSELECT**

Selects which NOT_CDREQ is checked when working in CD FIFO mode.

00: NOT_CDREQ is not checked

01: NOT_CDREQ[1] signal is checked

10: NOT_CDREQ[2] signal is checked

11: NOT_CDREQ[3] signal is checked

When 00 is selected channel 0 works under CD_REQ and holdoff mode, but it is assumed that the CDREQ signal is always high when holdoff time is reached.

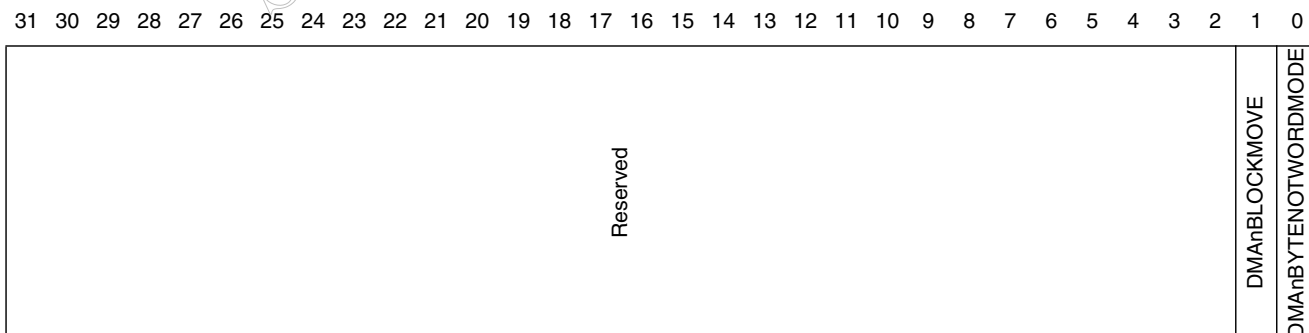
[0] **DMA0WORDNOTBURSTMODE**

0: Enable 4-word-bursts

1: Write word-at-a-time

PTI_DMAAnSETUP

DMA channels 1 to 3 byte not word mode and block move



Address: *PTIBaseAddress* + 0x1030, + 0x1050 and + 0x1070 (PTI3 only)

Type: Read/write

Reset: 0

Description:

[31:2] **Reserved**

[1] **DMAAnBLOCKMOVE**

0: CDADDR remains fixed.

1: CDADDR is incremented after each write. In this mode the CD FIFO mode is off, and so NOT_CDREQ is ignored and *Holdoff* and *Writelength* are not considered.

[0] **DMAAnBYTENOTWORDMODE**

0: Write word-at-a-time

1: Write byte-at-a-time

DMA buffer top address

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1010 0x1004	DMA0TOP																															
0x1014 0x1024	DMA1TOP																															
0x1018 0x1044	DMA2TOP																															
0x101C 0x1064	DMA3TOP																															

DMA buffer write address

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1020 0x1008	DMA0WRITE																															
0x1024 0x1028	DMA1WRITE																															
0x1028 0x1048	DMA2WRITE																															
0x102C 0x1068	DMA3WRITE																															

PTI_DMADCADDR**DMA CD FIFO page 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
DMACDADDR																		Reserved													

Address: *PTIBaseAddress* + 0x1060 (PTI1 only)

Type: Write only

Reset: Undefined

Description: DMA channels 1 to 3 must all write to the same 16-Kbyte page in memory. The PTI_DMADCADDR register holds the address of the base of the 16-Kbyte page. The page must be 16-Kbyte aligned, so the least significant 14 bits of PTI_DMADCADDR must be 0.
For DMA channels 1 to 3, certain aspects of the output are fixed in the DMA channels. The destination addresses for the written data are at fixed offsets from the output base address given by DMACDADDR. The offsets are shown in [Table 130](#). This table also shows which decoder is connected to each DMA channel.

Table 130: DMA CD FIFO page address

DMA channel	Write address offset from DMACDAddr	Target decoder	DMA request signal
1	0x2000	Video decoder	NOT_CDREQ1
2	0x3000	Audio decoder	NOT_CDREQ2
3	0x0000	Subpicture decoder	NOT_CDREQ3

PTI_DMAncDADDR**Configuration of CD FIFO address for channels 1 to 3**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1064 0x1038	DMA1CDADDR																															Res
0x1068 0x1058	DMA2CDADDR																															Res
0x106C 0x1078	DMA3CDADDR																															Res

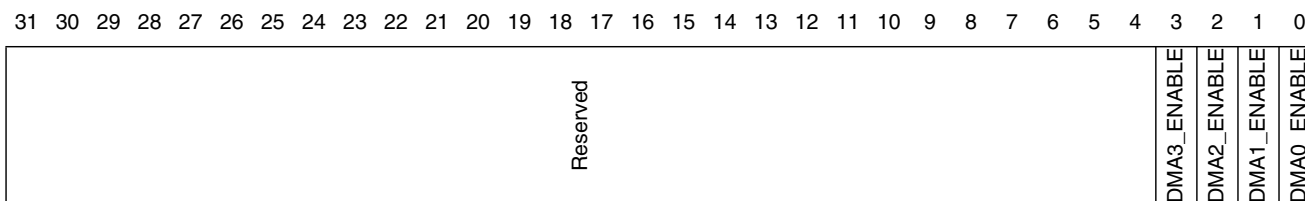
Address: *PTIBaseAddress* + 0x1064, + 0x1068, + 0x106C (PTI1)

Address: *PTIBaseAddress* + 0x1038, + 0x1058, + 0x1078 (PTI3)

Type: Write only

Reset: Undefined

Description: PTI_DMAncDADDR gives the address of the corresponding CD FIFO for channels 1 to 3. The addresses must be aligned to a word boundary: The two LS bits are assumed to be 0. DMAncDADDR is only defined for MPEG channels (MDC channels).

PTI_DMAENABLE
DMA enable

 Address: *PTIBaseAddress* + 0x1050 (PTI1)

 Address: *PTIBaseAddress* + 0x101C (PTI3)

Type: Write only

Reset: Undefined

Description: This register controls the enabling of the four DMA channels. Disabling channels 1 to 3 does not result in any data being lost inside the DMA controller but data may be lost by buffer overflow. Disabling channel 0 may result in lost data depending on the input data rate to the PTI and the length of time that the channel is disabled.

 [31:4] **Reserved**

 [3] **DMA3_ENABLE**

Enable (1) or disable(0) channel 3.

 [2] **DMA2_ENABLE**

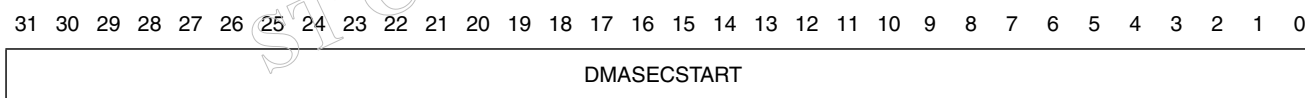
Enable (1) or disable(0) channel 2.

 [1] **DMA1_ENABLE**

Enable (1) or disable(0) channel 1.

 [0] **DMA0_ENABLE**

Enable (1) or disable(0) channel 0.

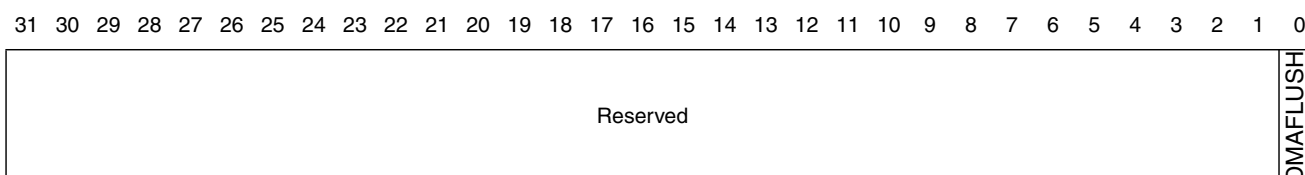
PTI_DMASECSTART
Address of start of section

 Address: *PTIBaseAddress* + 0x1074 (PTI1)

 Address: *PTIBaseAddress* + 0x103C (PTI3)

Type: Read/write

Reset: 0

Description: PTI_DMASECSTART gives the address of the start of the section currently being read by channel 0.

PTI_DMAFLUSH
Flush ready for a soft reset

 Address: *PTIBaseAddress* + 0x1078 (PTI1)

 Type: *PTIBaseAddress* + 0x105C (PTI3)

Type: Read/write

Reset: 0

Description: 1 is written to PTI_DMAFLUSH when a flush is required. It is reset by writing 0 to the bit.

PTI_DMAPTI3PROG**Memory map for PTI**

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

Reserved																																		DMAPT3PROG

Address: *PTIBaseAddress* + 0x107C

Type: Read/write

Reset: 0

Description: Select either PTI1 or PTI3 memory map.

0: Use PTI1 memory map (default)

1: Use PTI3 memory map

29.2 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* + 0x2004

Type: Read only

Reset: Undefined

Description: The number of bytes in the alternative FIFO. This FIFO does not overflow since there is flow control on its input.

PTI_IIFALTLATENCY**IIF alternative output latency**

31 30 29 28 27 26 25 24 23 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* + 0x2010

Type: Write only

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.

Confidential

PTI_IFFIFOCOUNT IIF count

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

Address: *PTIBaseAddress* + 0x2000

Type: 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_IFFIFOENABLE IIF FIFO enable

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

Address: *PTIBaseAddress* + 0x2008

Type: 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

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

Address: *PTIBaseAddress* + 0x2018

Type: Write only

Reset: Undefined

Description: The number of successive erroneous sync bytes found before the lock is treated as lost.

PTI_IIFSYNCLOCK IIF sync lock

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

Address: *PTIBaseAddress* + 0x2014

Type: Write only

Reset: Undefined

Description: The number of successive correct sync bytes found before the sync detection is locked.

PTI_IIFSYNCCONFIG **IIF sync configuration**

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

Address: *PTIBaseAddress* + 0x201C

Type: Write only

Reset: Undefined

Description:

[31:2] **Reserved**[1:0] **SYNC**

00: Default, if SYNC is activated use the internal clock

01: Use SOP

10: Use incoming TS_PACKETCLK

PTI_IIFSYNCPERIOD **IIF sync period**

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

Address: *PTIBaseAddress* + 0x2020

Type: Write only

Reset: Undefined

Description:

[31:8] **Reserved**[7:0] **SYNCPERIOD**Specifies the expected number of TS_IN_BYTECLK_PULSE cycles between SYNC bytes
On reset set to 188.**29.3 PTI configuration registers****PTI_AUDPTS** **Audio presentation time stamp**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0040																AUDPTS[31:0]															
0x0044																Reserved															
																AUDPTS[32]															

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_INTACK 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_INTENABLE 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.

Confidential

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

	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	
0x0048	VIDPTS[31:0]																																
0x004C	Reserved																VIDPTS[32]																

Address: *PTIBaseAddress* + 0x0048 and 0x004C

Type: Read only

Reset: Undefined

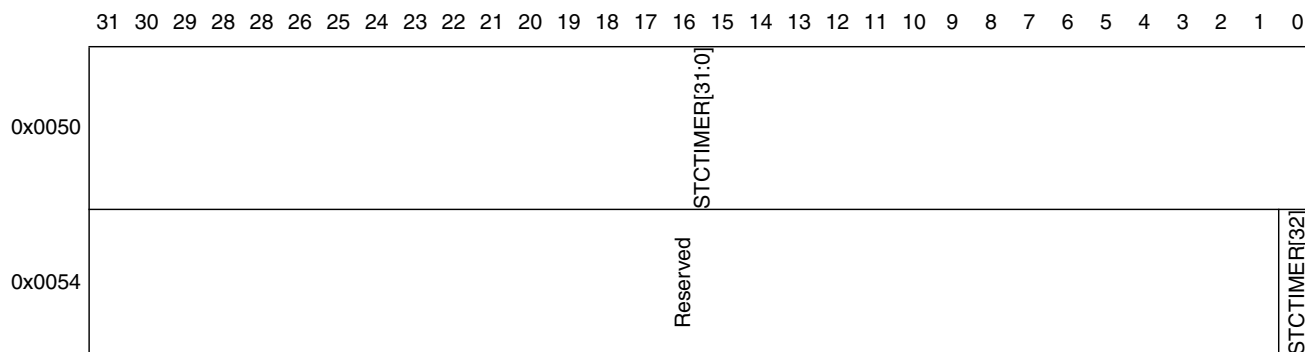
Description: The STC value is latched into this register at VSYNC.
The audio and video PTS registers are used by driver software to synchronize the audio and video. The time stamps are 33-bit values, so the most significant bit is held at a separate address.

Note: *Because the PTI divides the 27 MHz clock by 300, the PTI_VIDPTS register timebase is 90 kHz.*

Confidential

PTI_STCTIMER

Set STC timer



Address: *PTIBaseAddress* + 0x0050 to 0x0054

Type: Write only

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.

Confidential

ST Confidential (iPanel)

29.4 Section filter registers

PTI_SFFILTERDATAn Section filter 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
0x4000 0x4200	SFFILTERDATA0[31:0]																															
0x4008 0x4208	SFFILTERDATA0[63:32]																															
0x4010 0x4210	SFFILTERDATA1[31:0]																															
0x4018 0x4218	SFFILTERDATA1[63:32]																															
...	...																															
0x41F0 0x43F0	SFFILTERDATA31[31:0]																															
0x41F8 0x43F8	SFFILTERDATA31[63:32]																															

Address: *PTIBaseAddress* + 0x4000, 0x4008, 0x4010, up to 0x41F8 (CAM A)
PTIBaseAddress + 0x4200, 0x4208, 0x4210, up to 0x43F8 (CAM B)

Type: Read/write

Reset: Undefined

Description: The PTI_SFFILTERDATAn registers are composed of four 32-bit words. Within the four word group the section filter has two 32-bit words dedicated to the storage of data, and two 32-bit words dedicated to the storage of masking information.

There is one SFFILTERDATA register for each of the 32 section filters. Each register is split into two 32-bit registers; SFFILTERDATA[31:0] holds the least significant word, and SFFILTERDATA[63:32] holds the most significant word. This enables the least significant or most significant word to be written independently without affecting the other word.

Only one word of a filter is updated at a time, so updates of the section filters should be carried out when one of the following is true:

- a false match does not matter,
- the filter is not in use for any PIDs,
- the PIDs that use the filter have been disabled.

Filtering is performed by testing whether *packet* bit AND *FilterMask* = *FilterData* bit AND *FilterMask*. If this is true then the packet passes through the filter.

PTI_SFFILTERMASKn Section filter 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
0x4004 0x4204	SFFILTERMASK0[31:0]																															
0x400C 0x420C	SFFILTERMASK0[63:32]																															
0x4014 0x4214	SFFILTERMASK1[31:0]																															
0x401C 0x421C	SFFILTERMASK1[63:32]																															
...	...																															
0x41F4 0x43F4	SFFILTERMASK31[31:0]																															
0x41FC 0x43FC	SFFILTERMASK31[63:32]																															

Address: *PTIBaseAddress* + 0x4004, 0x400C, 0x4014 up to 0x41FC (CAM A)
PTIBaseAddress + 0x4204, 0x420C, 0x4214 up to 0x43FC (CAM B)

Type: Read/write

Reset: Undefined

Description: The PTI_SFFILTERMASK registers are 64-bit registers holding the filter masks. There is one PTI_SFFILTERMASK register for each of the 32 section filters. Each register is split into two 32-bit registers; PTI_SFFILTERMASK[31:0] holds the least significant word, and PTI_SFFILTERMASK[63:32] holds the most significant word. This enables the least significant or most significant word to be written independently without affecting the other word.

Bits set to 1 in the PTI_SFFILTERMASK registers enable the corresponding bits in the PTI_SFFILTERMASK registers to be compared with the packet. Bits set to 0 in the mask disable comparison between the corresponding bits, and may be regarded as *don't care* bits.

Only one word of a filter is updated at a time, so updates of the section filters should be carried out when one of the following is true:

- a false match does not matter,
- the filter is not in use for any PIDs,
- the PIDs that use the filter have been disabled.

Filtering is performed by testing whether:

packet bit AND *FilterMask* = *FilterData* bit AND *FilterMask*

If this is true then the packet passes through the filter.

PTI_SFNOTMATCHn Not match mode for CAM A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4400	Reserved																										EN0	DATA0				
0x4404	Reserved																										EN1	DATA1				
...	...																															
0x4478	Reserved																										EN30	DATA30				
0x447C	Reserved																										EN31	DATA31				

Address: *PTIBaseAddress* + 0x4400, 0x4404, 0x4408 up to 0x447C (CAM A only)

Type: Read/write (only readable when ENn set to 1)

Reset: Undefined

Description: For each line of CAM A there is a not match option. This consists of a 5-bit DATA field which is checked against the version number in a section and must not match for the CAM A line to record a hit. There is an enable bit (ENn) for each not match element.

29.5 Transport controller mode register**PTI_TCMODE Transport controller 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																																TCSINGLESTEP	TCRESETIPTR	TCENABLE

Address: *PTIBaseAddress* + 0x0030

Type: Read/write

Reset: Undefined

Description: The PTI_TCMODE register allows the ST20 software to control the transport controller. Under normal operation the software should only set the TCENABLE bit to start the TC executing code.

[31:3] **Reserved**

[2] **TCSINGLESTEP**: Reset TCENABLE after each instruction.

[1] **TCRESETIPTR**: Reset the instruction pointer of the TC (level sensitive).

[0] **TCENABLE**

1 enables the TC

0 disables the TC disabling instruction fetch. Reading a zero after a single step means the TC has finished executing the previous instruction.

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.

Single stepping the TC through its instructions

- 1 Set the TCENABLE and TCSINGLESTEP bits.
- 2 Wait for the TCENABLE bit to be cleared.

Repeat this procedure for further single steps.

Confidential

ST Confidential (iPanel)

30 Transport stream multiplexor (TSMUX)

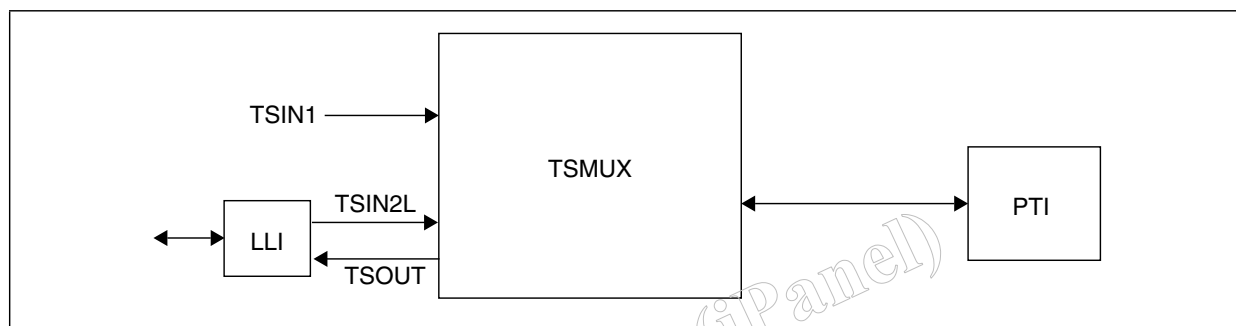
30.1 Overview

Digital set-top boxes and digital TVs may require multiple program stream inputs, for example cable/terrestrial/satellite TV. The transport stream multiplexor (TSMUX) is designed to route two input transport streams independently to the on-chip PTI.

The TSMUX also supports input from an internal transport stream (software writable transport stream, SWTS) and allows the PTI output to be multiplexed to an output stream.

The operation of the TSMUX in terms of the main inputs and outputs is shown in [Figure 72](#).

Figure 72: TSMUX inputs and outputs

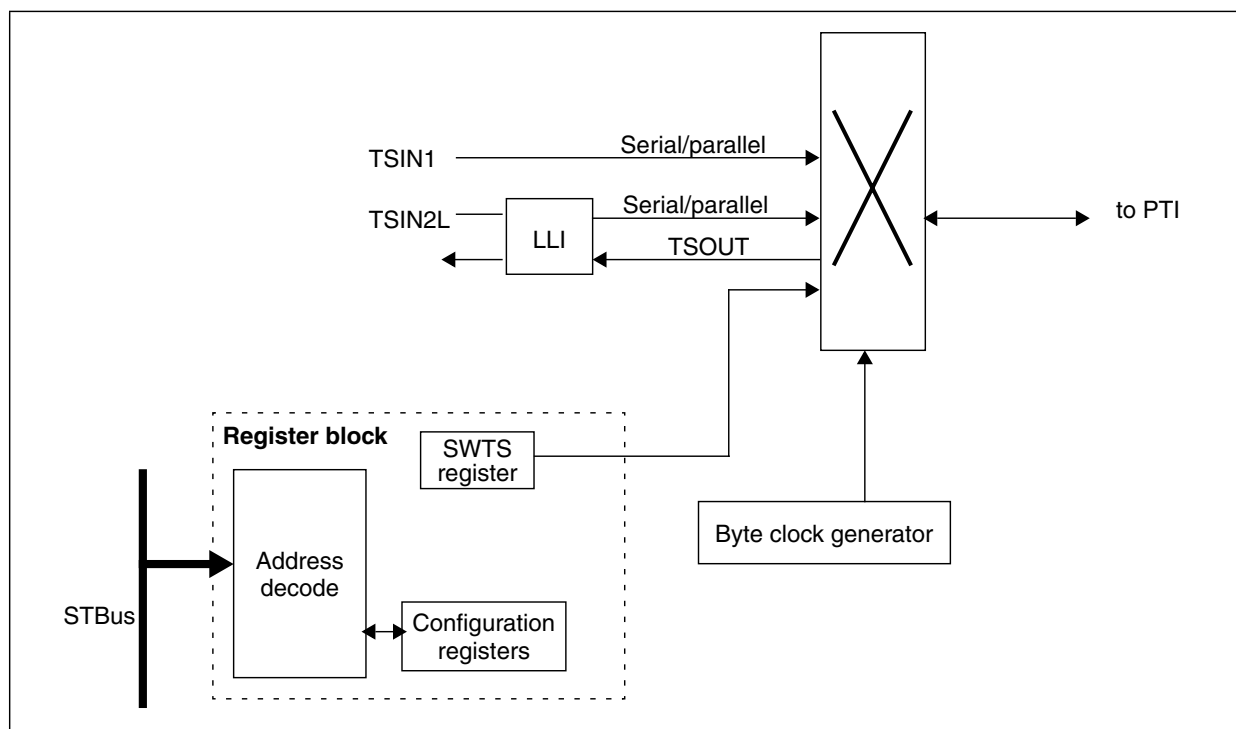


30.2 Architecture

The architecture of the TSMUX is shown in [Figure 73](#), which also shows how transport streams are routed.

The design is organized as a number of independent blocks which are notionally linked together to form the TSMUX, all controlled by the CPU using the STBus interface.

Figure 73: TSMUX architecture



30.2.1 Transport stream inputs

Transport streams from any of the input stages, or from the TSMUX_SWTS register are routed to multiplexors that are connected to the PTI.

TSIN2L is routed via the LLI which multiplexes signals between a 1394 AV link layer interface and a 1284 controller interface.

TSIN1 and TSIN2L support serial or parallel transport streams, selectable via the TSMUX_TSIS[1:2]MODE registers.

The TSIN1 input can be routed directly to TSIN2L pins as an output for DVBCI integration. This is controlled by bit TO_PAD_TS2L_OE in register CONFIG_CONTROL_E.

30.2.2 Transport stream output (TSOUT)

The transport stream output from the PTI may be routed to the TSMUX transport output, for example, to write a stream to a device on a 1394 bus. The outgoing stream is multiplexed via the LLI on to a 1394 AV link layer interface.

30.2.3 SWTS

In HDD applications a transport stream 'software input' is retrieved from a disk and injected into the PTI via a special register. Transport stream bytes are written sequentially to the register by the CPU as words or bytes, buffered in FIFOs, and then converted to a standard byte-wide transport stream. The TSMUX then routes the reconstructed transport stream the PTI.

30.2.4 Packet synchronization

The PTI interface carries timing information that allows the PTI to set a fixed latency period between TSIN and TSOUT, this ensures that there is sufficient time to process the data. The latency value is programmable in the PTI_IIFALTLATENCY register.

30.3 Transport stream routing

The TSMUX routes transport streams from the two input streams, and the TSMUX_SWTS register, to the PTI input interface. It also controls the routing of transport outputs from the PTI back to the main TSMUX transport output (TSOUT).

30.3.1 Incoming stream protocol

The channel decoder IC must contain the following signals:

- **TSBYTECLK**

This is used to clock in all the other signals. When TS input data is parallel (the TSIS[1:2]MODE bit in the TSMUX_TSIS[1:2]MODE register = 0) the TS byte clock should be set in relation to the STBus clock:

STBus clock $\geq 3 \times$ TS byte clock.

When TSINPUT data is serial (the TSIS[1:2]MODE bit in the TSMUX_TSIS[1:2]MODE register = 1) the TSBYTECLK should be set to a maximum of 100 MHz only if using a DVB stream. In other circumstances the same relation between the STBus clock and TS byte clock remains:

STBus clock $\geq 3 \times$ TS byte clock.

- **TSDATA[7:0] or TSDATA[7]**

The data byte or bit to be input. Data is valid when TSVALID is high.

- **TSVALID**

When TSVALID is high TSDATA contains a valid byte.

- **TSPACKETCLK**

Indicates the first byte of a packet. TSMUX interprets it differently in synchronous and asynchronous mode.

- **TSPACKETERROR**

Error flag for the transport packet. Sampled on the TSBYTECLK rising edge that samples the first byte of the transport packet.

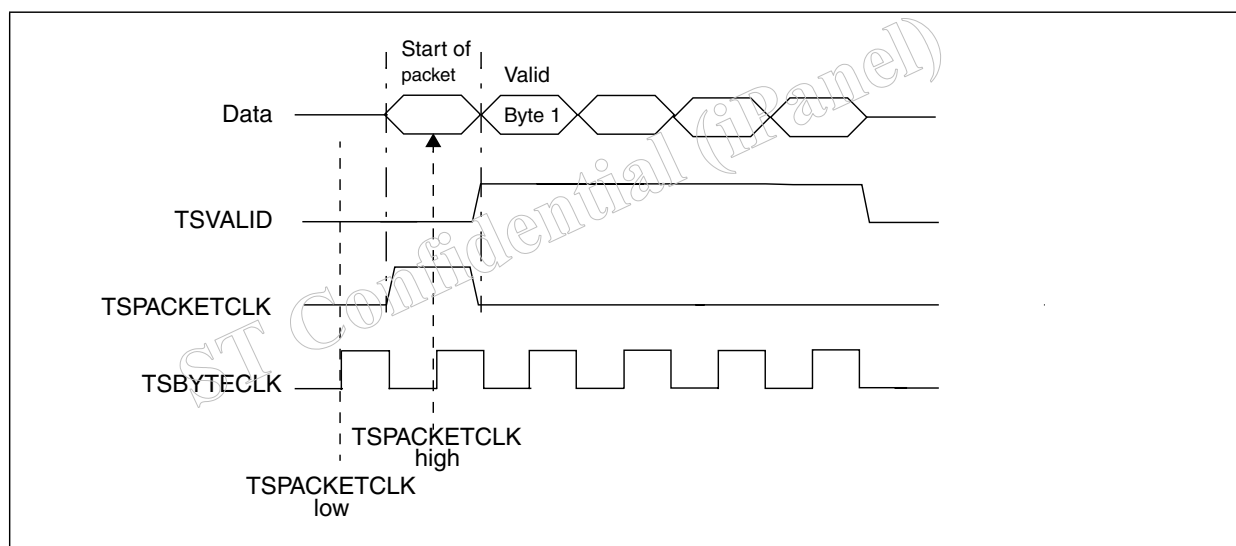
30.3.2 Asynchronous vs. synchronous packet detection

Packet detection is set to asynchronous or synchronous operation. For DSS protocols (using FEC STV0299 only) use asynchronous operations and for DVB and DSS (without FEC STV0299) use synchronous.

A rising edge on TSPACKETCLK indicates the first byte in a packet (start-of-packet).

In asynchronous mode the start-of-packet is detected by the next TSBYTECLK which finds TSVALID high after TSPACKETCLK goes high. In synchronous mode the start-of-packet is detected when both TSPACKETCLK and TSVALID are high on a TSBYTECLK tick after the previous tick found TSPACKETCLK low. This is illustrated in [Figure 74](#).

Figure 74: Asynchronous packet detection



30.4 PTI MUXing

The MUXing stage routes the input transport streams to the PTI. Transport streams from either of the two inputs, or TSMUX_SWTS, are routed to the PTI. The MUX is controlled by a configuration register which can be programmed from software.

30.5 Transport output

The output stage synchronizes a PTI output transport stream to a selected TSBYTECLK, allowing it to be sent to an off-chip target. The choice of the source for the transport output stream (TSOUT) is controlled by the TSOUT_SRC register.

The source for the output clock is selected from any of the incoming TSIS input clocks, or the internally generated byte clock (local byte clock). The choice of output clock is controlled by the TSMUX_TSOUTCLKSOURCE register.

30.6 Local byte clock

The TSMUX supports an internally generated byte clock (LOCALCLK) which can be used instead of the input byte clocks to clock out the transport streams from the PTI.

The period of this clock is programmable in the TSMUX_CLOCKGEN register. This register specifies the period of the locally generated byte clock expressed in SYSCLK cycles. If the defined period is even then the locally generated clock goes high for $n / 2$ cycles and low for $n / 2$ cycles. If the defined period is odd then the locally generated clock goes high for $n / 2$ cycles and low for $n / 2 + 1$ cycles.

The maximum frequency of the local clock is STBus clock $\geq 4 \times$ LOCALCLK.

30.7 TS timing information

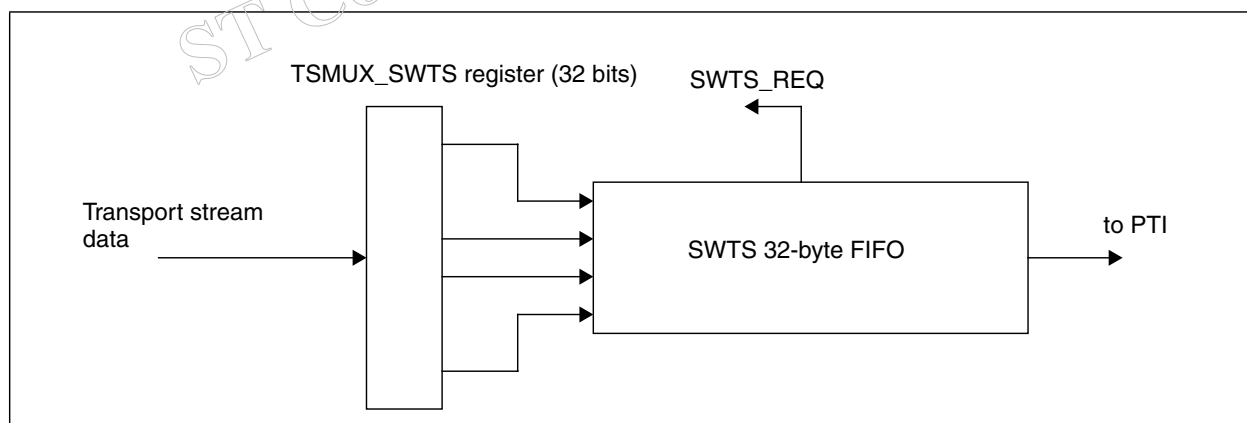
Input byte rate and start-of-packet signals are passed directly to the PTI. Using this information a PTI defines a fixed latency between TSIN and TSOUT to allow the PTI sufficient time to process the data. This is programmable in the PTI_IIFALTLATENCY register in terms of clock cycles, to support the varying processing times required by different applications (depending on the type and extent of processing required and the use of hardware vs. software).

The latency counter is reset at the start of each incoming packet. The start of an incoming packet is defined in the IIFSYNCCONFIG register.

30.8 TSMUX_SWTS

SWTS is the software writable transport stream register. The SWTS system can be used to copy a transport stream from memory to the PTI.

Figure 75: SWTS architecture



30.8.1 Communication model

Bytes or words are written to the TSMUX_SWTS register using the CPU. The data is transferred to a 32-byte FIFO before being converted to a byte-wide transport stream for input to the PTI MUXes.

Transfer of data from the SWTS register to the FIFO is paced using the SWTS_REQ signal that is routed through the PTI DMA channel. This is asserted when the FIFO has room for at least 16 bytes.

SWTS_REQ is made available to software in bit 10 of the SWTSCONFIG register.

As soon as there is information available in the FIFO it is streamed out to the PTI MUX. The speed of byte transfer from the FIFO can be controlled using the SWTS_PACE field in the SWTSCONFIG register.

30.8.2 Data format

Data is assumed to be little endian that is, the LS byte contains the byte which is sent out first to the PTI. Either words or bytes can be written to the register. The SWTS only sends valid bytes to the PTI. The value returned on reading the register is either the current stored value or, if empty, the latest stored value.

ST Confidential (iPanel)

Confidential

31 Transport stream multiplexor (TSMUX) registers

All TSMUX registers are 32-bit read/write registers, aligned on 8-byte boundaries for future compliance with 64-bit architectures.

Addresses are provided as the *TSMUXBaseAddress* + offset.

The *TSMUXBaseAddress* is:

0x2040 0000.

A register summary is given in [Table 55: Transport stream multiplexor \(TSMUX\) registers on page 80](#).

TSMUX_TSISnMODE Set TSIS[1:2] stream input 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
0x08	Reserved																TSIS2_SYNC		TSIS1_SYNC		TSIS2_MODE		TSIS1_MODE									
0x10	Reserved																TSIS2_SYNC		TSIS1_SYNC		TSIS2_MODE		TSIS1_MODE									

Address: *TSMUXBaseAddress* + 0x08 to 0x10

Type: Read/write

Reset: 0

Description: The TSMUX_TSISnMODE registers set the modes of operation of TSIS1 and TSIS2:

- parallel or serial mode,
- synchronous or asynchronous packet detection.

[31:2] **Reserved**

[1] **TSISn_SYNC**

0: Packet detector in asynchronous mode

1: Packet detector in synchronous mode

[0] **TSISn_MODE**

0: Parallel mode

1: Serial mode

TSMUX_PTIASOURCE Input source for the PTI

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

Address: *TSMUXBaseAddress* + 0x40

Type: Read/write

Reset: 0

Description: The TSMUX_PTIASOURCE register defines the transport stream input source (TSIS) for the PTI.

[31:3] **Reserved**

[2:0] **TSINPUTA**

000: Reserved

001: TSIS1

010: TSIS2

011: Reserved

100: Reserved

101: SWTS

TSMUX_SWTS**SWTS data 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

SWTS_DATA

Address: *TSMUXBaseAddress* + 0x28

Type: Read/write

Description: The TSMUX_SWTS register is the data register for writing transport streams from memory to the PTI.

Data may be written as words or bytes. On reading the register it *contains either the current stored value or, if empty, the latest stored value.*

Byte transfer from TSMUX_SWTS is little endian, that is the LS byte is sent out first to the PTI. Bits within bytes are arranged in a big endian fashion, that is, earliest bit is bit 7.

TSMUX_SWTSCONFIG**Configure SWTS**

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

Reserved

SWTS_REQ

SWTS_PACE

Address: *TSMUXBaseAddress* + 0x30

Type: Read/write

Reset: 0

Description: Transport data from SWTS enters a 32-byte FIFO. The FIFO is popped byte-by-byte and converted to a valid transport stream before being transferred to a PTI. The speed of byte transfer can be defined using SWTS_PACE (bits [9:0]) which sets the wait time between two valid bytes coming out of the FIFO.

The FIFO is kept supplied with bytes from SWTS. SWTS_REQ is asserted when at least half the FIFO is empty and stays asserted until the FIFO is over half full. In software-controlled data transfer SWTS_REQ can be read by the program to determine if the FIFO is ready to receive the next block of data. In DMA-controlled transfers the DMA controller receives the SWTS_REQ signal directly.

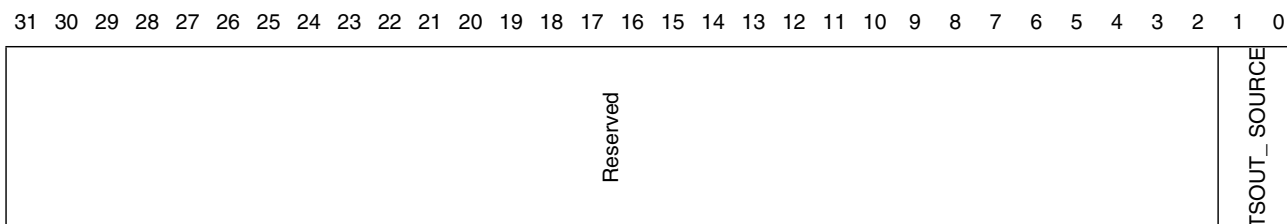
[31:11] **Reserved**[10] **SWTS_REQ**

Request for data from the FIFO. Asserted when at least half the SWTS FIFO is empty, stays asserted until the FIFO is over half full.

[9:0] **SWTS_PACE**

Paces the transfer of bytes from SWTS to the PTI. SWTS_PACE sets the minimum number of wait states in numbers of clock cycles, between consecutive byte transfers from the SWTS FIFO to the PTI.

Confidential

TSMUX_TSOUTSOURCE PTI source for TSOUT

Address: *TSMUXBaseAddress* + 0xA0

Type: Read/write

Reset: 0

Description: The TSMUX_TSOUTSOURCE register defines the PTI source for TSOUT. Bits 0 to 1 are used to define the PTI.

[31:2] **Reserved**

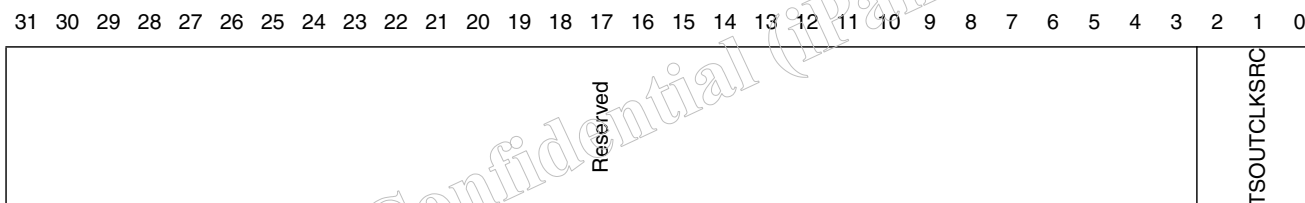
[1:0] **TSOUT_SOURCE**

Source for TSOUT. Always set to 0.

00: PTI

01: Reserved

10: Reserved

TSMUX_TSOUTCLKSOURCE TSOUT byte clock source

Address: *TSMUXBaseAddress* + 0xA8

Type: Read/write

Reset: 0

Description: The TSMUX_TSOUTCLKSOURCE register defines the source for the TSOUT byte clock. Either of the two incoming TSIS input clocks can be selected, or the internally generated clock can be used.

[31:3] **Reserved**

[2:0] **TSOUTCLKSRC**

Source for TSOUT byte clock:

000: Reserved

001: TSBYTECLK1

010: TSBYTECLK2

011: Reserved

100: Reserved

101: Locally generated clock

Confidential

TSMUX_CLOCKGEN Local clock speed select

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

Address: *TSMUXBaseAddress* + 0xB0
Type: Read/write
Reset: 0
Description: The TSMUX_CLOCKGEN register sets the period of the locally generated byte clock.
 [31:10] **Reserved**
 [9:0] **CLOCKGEN_DIVIDE**
 Sets the period of the locally generated byte clock (LOCALCLK) in SYSCLK cycles. The minimum value for CLOCKGEN_DIVIDE is 4.
 If the period is even the clock goes high for $n / 2$ cycle and low for $n / 2$ cycles. If the period is odd the clock goes high for $n / 2$ cycle and low for $(n / 2) + 1$ cycles.

Confidential

ST Confidential (iPanel)

32 IEEE1394 link layer interface (LLI)

32.1 Overview

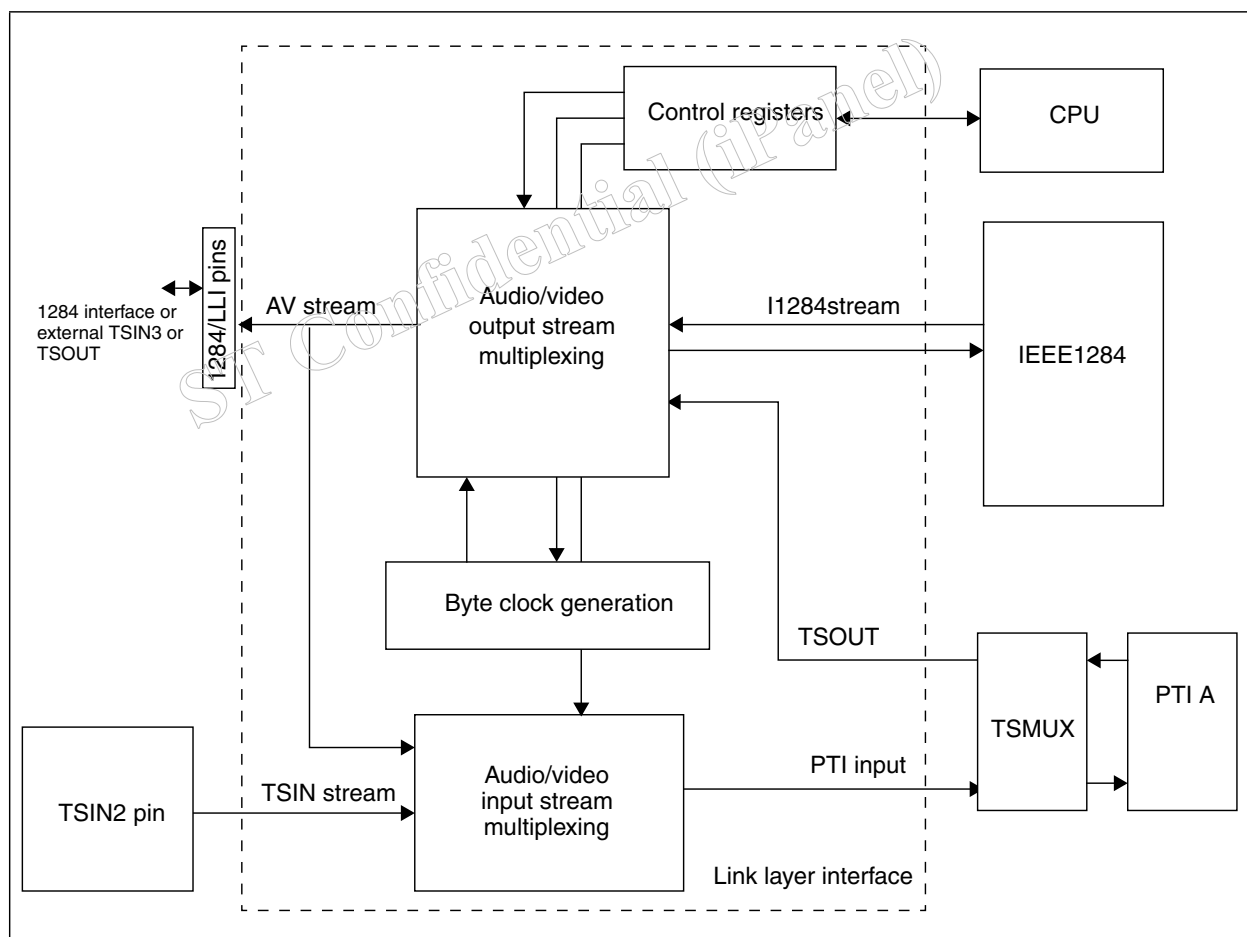
The LLI block operates as a multiplexor, selecting data streams from a number of sources and routing them to the correct subsystem and pins. It has an input port for transport streams, called the TSIN, and a bidirectional high speed data port, called the LLI port.

The LLI port can be configured as an IEEE1284 interface or an interface to an external IEEE1394 link layer controller. The LLI pins are shared between these functions. In this chapter, the port may be referred to as the IEEE1394 port or the IEEE port depending on the context.

The LLI also supplies a locally generated byte clock based on the STBus clock, with a frequency down to 10 MHz.

32.2 Block diagram

Figure 76: LLI block diagram



32.3 Data streams

32.3.1 AV stream

The AV stream is defined as any data passing between the I1284 pins and the AV input and output multiplexors. It may be an input or output stream, and may be clocked by either the external TSIN stream clock or by a locally generated byte clock.

Depending on the settings of the LLI_CONTROL register bits, the AV stream may be:

- an outgoing stream from the PTI output port, via the TSMUX, to the I1284 pins,
- a set of IEEE1284 signals or a transport stream from the I1284 peripheral.

The AV stream can be:

- an incoming transport stream on the I1284 pins going to the PTI input,
- a set of IEEE1284 signals to the I1284 peripheral.

Note: The input and output AV stream multiplexors are independent and the AV stream input is taken directly from the I1284 pins, so that some loopback modes are possible.

For example, a transport stream output from the I1284 peripheral can be input to the PTI.

32.3.2 PTI input stream

The PTI input stream is the stream from the LLI to the PTI.

32.3.3 PTI output stream

This is defined as the transport stream from the PTI block to the AV output multiplexor. It consists of the same signal group as the input stream, with the addition of PACKETTAG[3:0] signals but without the PACKETERROR signal.

32.3.4 I1284 stream

This is as defined as the transport or I1284 format information stream from the I1284 micro.

32.3.5 TSIN stream

This is defined as the stream from the TSIN pins to the AV input multiplexor.

32.4 Pin function

The function and direction of dataflow through the I1284 pins depends on the setting of the bits in the LLI_CONTROL register.

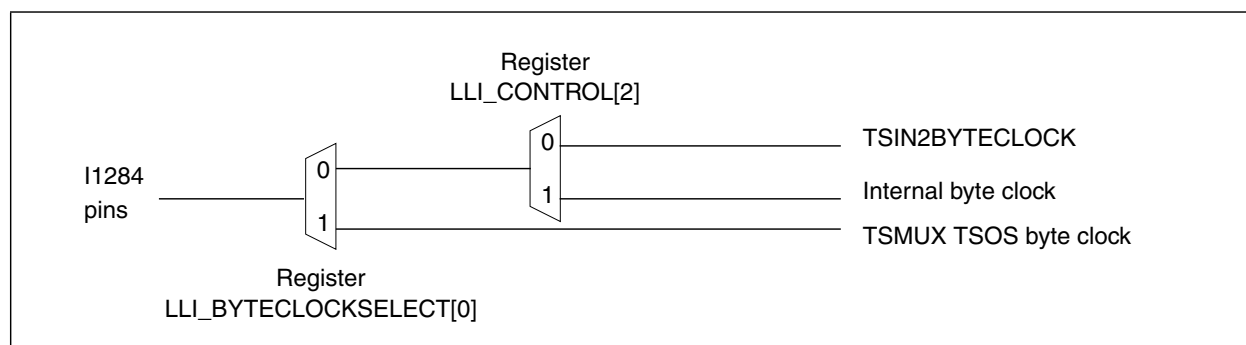
Table 131 details the functions of each of the I1284 pins with different settings of the LLI_CONTROL register bits. Figure 77 shows the selection of the byte clock according to the settings of bit 2 (LOCALNOTTSCLOCK) in register LLI_CONTROL and bit 0 (BYTECLKSELECT) in register LLI_BYTECLOCKSELECT.

Table 131: Pin functions

External pin name	Function and pin direction					
	AVNOT1284 = 0		AVNOT1284 = 1			
			AVOUTNOTIN = 0		AVOUTNOTIN = 1	
P1284DATA[7:0]	1284DATA[7:0]	I/O	AVINPUTDATA[7:0]	I ^a	AVOUTPUTDATA[7:0]	O
NOT_P1284SELECTIN	1284NOTSELECTIN	I	No function	I	GND	O
NOT_P1284INIT	1284NOTINIT	I	No function	I	AVPACKETTAG3	
NOT_P1284FAULT	1284NOTFAULT	O	No function	I	AVPACKETTAG2	
NOT_P1284AUTOFD	1284NOTAUTOFD	I	No function	I	AVPACKETTAG1	
P1284SELECT	1284SELECT	O	No function	I	AVPACKETTAG0	
P1284PERROR	1284PERROR	O	AVINPUTBYTE CLKVALID	I ^a	AVOUTPUTBYTE CLKVALID	O
P1284BUSY	1284BUSY	O	AVINPUT PACKETCLK	I ^a	AVOUTPUTPACKET CLK	O
NOT_P1284ACK	1284NOTACK	O	AVBYTECLK	O ^b	AVBYTECLK	O ^b
NOT_P1284STROBE	1284NOTSTROBE	I	AVINPUT PACKETERROR	I ^a	GND	O

- These signals are only fed into the PTI inputs when the LLI_CONTROL register bit AVNOTTS is set to 1.
- The BYTECLK output on this pin is selected by the LLI_CONTROL register bit LOCALNOTTSCLOCK. Setting this register bit to 0 selects the clock input on the TSIN2BYTECLOCK pin to be output on this pin. Setting the register bit to 1 selects the local byte clock generator signal to be output on this pin. This is overridden if the TSOS BYTECLK is set.

Figure 77: Byte clock selection



33 IEEE1394 link layer interface (LLI) registers

Addresses are provided as the *LLIBaseAddress* + offset.

The *LLIBaseAddress* is:

0x2030 0000.

A register summary is given in [Table 40: IEEE1394 link layer interface \(LLI\) registers on page 65](#).

LLI_CONTROL

LLI control register

7	6	5	4	3	2	1	0
Reserved				AVOUTNOTIN	LOCALNOTTSCLOCK	AVNOT1284	AVNOTTS

Address: *LLIBaseAddress* + 0x00

Type: Read/write

Reset: 0

Description:

[7:4] **Reserved**

[3] **AVOUTNOTIN**

Selects the direction of the AV stream when operating in transport mode:

0: AV stream pins are inputs

1: AV stream pins are outputs

[2] **LOCALNOTTSCLOCK**

Selects the clock source to be output:

0: Select TSINBYTECLK

1: Select locally generated BYTECLK

[1] **AVNOT1284**

Selects the AV output stream source:

0: Select I1284 signals (bypass mode)

1: Select AV stream to/from I1284 pins

[0] **AVNOTTS**

Selects the PTI input stream:

0: Select TSIN stream

1: Select AV input stream

Confidential

LLI_BYTECLOCK**LLI byte clock register**

7	6	5	4	3	2	1	0
Reserved						DIVISION_RATIO	

Address: *LLIBaseAddress* + 0x08

Type: Read/write

Reset: 00

Description:

[7:2] **Reserved**

[1:0] **DIVISION_RATIO**

Value by which to divide the system clock when producing the byte clock:

00: Divide by 4

01: Divide by 6

10: Divide by 8

11: Divide by 10

The internal BYTE CLOCK is free running and is programmed to have a period which is a fixed multiple of the system clock period, as defined by the byte clock ratio register.

Changing the clock source during operation may lead to glitches occurring on the clock and should therefore only be performed before initialization of any external IEEE1394 LLI devices which use the clock.

LLI_BYTECLOCKSELECT **LLI byte clock select register**

7	6	5	4	3	2	1	0
Reserved							BYTECLKSELECT

Address: *LLIBaseAddress* + 0x18

Type: Read/write

Reset: 00

Description: When the BYTECLKSELECT bit is set, the BYTECLK from the TSOS within the TSMUX is selected as the BYTECLK going off-chip through the IEEE1394 port. This bit when set overrides the selection of LOCALNOTTSCLOCK in register LLI_CONTROL.

Confidential

34 MPEG video decoder

34.1 Overview

The MPEG video decoder decompresses an MPEG 2-bit stream and constructs a picture stream. The display functions are described in [Chapter 36: Subpicture decoder on page 333](#) and [Chapter 38: Display planes on page 344](#). The MPEG video decoder registers are described in [Chapter 35: MPEG video decoder registers on page 306](#).

34.2 Decoder operation

The video decoder is a picture decoder; it decodes one picture and then stops until instructed to decode the next picture in the video bit stream.

Normally, the decoding of a new picture starts in response to the start of display of a new picture. The registers whose contents can change from picture to picture are double banked and are updated automatically when decoding starts. The bit stream is read from the bit buffer into the variable length decoder (VLD), from where a picture can be built. Any required predictors are fetched from the appropriate area of the external memory, and the reconstructed picture is written back into the area of memory assigned to the decoded picture.

While a picture is being decoded, the start code detector locates the start of the next picture header. The CPU then uses this to set up the double banked registers to decode the next picture.

All of these tasks can be synchronized using interrupts generated on start code hits and vertical sync signals.

34.2.1 Start code search

The video decoder is able to decode in its entirety a video bit stream from the slice layer downwards. The higher layers (that is, picture and upwards) are decoded by the driver in order to extract the information needed for decoding and set up the appropriate video decoder registers and quantization tables. Since the header information is byte aligned and requires minimal interpretation, this task represents only a small load on the CPU.

The start code detector parses the bit stream stored in the bit buffer and locates start codes corresponding to picture layer and above. When one of these start codes has been found, the start code detector stops and raises an interrupt.

The CPU is then able to read the header data following the start code. The start code detector starts automatically whenever the decoding of a new picture starts and on user command. In normal operation, start code parsing is performed one picture in advance of decoding.

34.2.2 Bandwidth reduction mode

Bandwidth reduction mode is the only decoding mode that is supported by the device. In this mode, where I, P and B frames are decoded into and displayed from frame buffers in external memory, the decoder uses three frame buffers in external memory. This mode optimizes memory bandwidth use.

Note: B frame on the fly decoding mode is not supported in the STi5516.

34.3 Configuration and control

The MPEG control register MPEGCONTROL controls the speed of CPU accesses to the audio, video, subpicture and digital encoder registers. It also controls the nature and speed of accesses to the CD FIFOs. For normal applications, this register should be left in its default state.

Bits [14:12] (CSHOLDLSB and CSSETUPLSB) are used to slow down register accesses by adding extra cycles on the internal register select signals. Bits [10:8] (CDHOLDLSB and CDSETUPLSB) are used to slow down CD FIFO accesses by adding extra cycles on the internal FIFO strobe signals.

There are two mechanisms to aid in the control of traffic to CD FIFOs:

- a 5-bit interaccess delay between access groups,
- ability to stall data transfers (not recommended).

The interaccess value delays the return of the signal IC_R_REQ to the memory controller when accessing the MPEGAV FIFOs. This signal indicates to the requesting system that an access has completed, and if it is delayed stops the requester generating new requests to transfer data. This effectively regulates the maximum data throughput to the MPEGAV FIFOs.

If the NOTxxEN bit is set, then the MPEG buffer samples the NOT_xxREQ signal on each byte transfer. If the NOT_xxREQ signal is inactive (high) and enabled, then the MPEG buffer stalls the transfer until the signal becomes active. If the enable is not set, then data is transferred regardless of the status of the request signal. Data transferred when NOT_xxREQ is inactive is lost. Since the MPEG registers and MPEG FIFOs share a single data port there is a danger of deadlock occurring if clearing the FIFO is dependent on accessing the MPEG registers and NOT_xxREQ is enabled. It is therefore not recommended to use the stall feature.

The MPEGCONTROL register is shared between all three registers, and controls the maximum throughput to all MPEG FIFOs rather than each FIFO.

A write to this register must be done as a 32-bit word. The read for this register is completed using three byte reads, (LD1). The active byte enable determines the byte lane used during the access. The strobe cycle timings default to one cycle. The base address is 0x0000 5000.

34.4 Reset

Hard reset is a global signal and is described in [Chapter 22: System services on page 204](#). The following types of soft reset can be used for the video decoder.

- Total soft reset is generated by setting and resetting bit SRS (VID_CTL) and register AUD_RES.

They must be set for a duration of at least 1 μ s.

- Audio, video and subpicture subsystems may be individually soft reset by setting and resetting VID_SRA and AUD_RES, VID_SRV and SPD_SPR respectively.
- Pipeline reset is generated by setting and resetting bit PRS (VID_CTL).

It must be set for a duration of at least 40 ns.

After a soft reset, all processes concerning decoding and bit buffer control are reset. Any data remaining in the bit buffer, the compressed data FIFO and the start code detector FIFO are lost.

The interrupt unit is reset. All registers maintain their contents and the display process is not disturbed. A soft reset would normally be used when the decoding of the current bit stream must be terminated and it is required to restart on a new sequence.

After a hard, soft or a video soft reset, the first task performed by the pipeline when it has been enabled is always a search for the beginning of a new sequence. The bit buffer data is flushed until the first picture start code following a sequence start code is detected by the pipeline, at which time it stops. At this point normal picture decoding behavior is resumed. After a hard or a soft reset, the first search performed by the start code detector in response to the first DSYNC is

always a search for a sequence start code, after which it stops. After this, the start code detector operates normally.

A pipeline reset terminates the decoding of the current picture. The remaining bits of the picture are flushed from the bit buffer until the next picture start code is detected by the pipeline. At this point normal behavior is resumed, that is, the pipeline waits for the next picture decoding instruction. No other part of the circuit is affected by a pipeline reset. A pipeline reset would normally be used as part of a manual error recovery procedure. A pipeline reset has no effect if the decoding pipeline is in its idle state.

34.5 Bit buffer and start code detection (video)

34.5.1 Bit buffer

The transfer of compressed data is carried out using the link DMA engines. Compressed data can be taken from any memory space visible to the CPU and transferred to the relevant elementary stream decoder.

34.5.2 Start code detection

The start code detector operates in parallel with the decoding pipeline. The purpose of this unit is to allow external access to the header data which follows start codes in the input bit stream. Compressed data is read twice from the bit buffer; once into the pipeline, and once into the start code detector through the 128-byte header FIFO. The transfer of data into the header FIFO does not affect the bit buffer level; only the data transfer into the pipeline can reduce the bit buffer level.

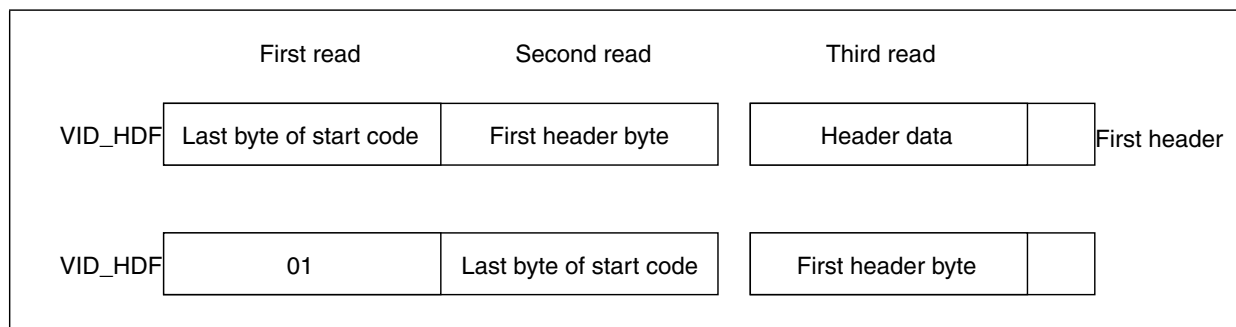
Start code detection is initiated in two ways.

- Automatically whenever the internal event DSYNC occurs. DSYNC is derived from VSYNC as described in [Section 34.10.1: Decoding task on page 299](#).
A DSYNC is generated every time the pipeline starts a new picture decoding task.
- By software writing to the VID_HDS register with bit HDS set (VID_HDS).

When start code detection has been started, data is read continuously from the bit buffer into the header FIFO and parsed by the start code detector, which receives the FIFO output data. When a start code is detected, the data scanning stops and the status bit SCH (VID_STA) becomes 1. When a start code has been detected, it can be identified by reading the VID_HDF register. The start code detector detects all start codes other than the codes from 0x0000 0102 through to 0x0000 01AF. The first slice start code 0x0000 0101 can be optionally detected to help driver development.

The register VID_HDF should always be read twice to return a 16-bit value. The most significant byte is read first. After detection of a start code, VID_HDF returns one of the 16-bit values shown in [Figure 78](#).

Figure 78: States of VID_HDF after detection of a start code



The first step is to examine the first byte read from VID_HDF. If this contains 0x01, then the start code can be identified by a second read at the same address. If the first byte is not 0x01 then it must be the last byte of the start code and the second byte is the first byte of the header data. In both cases, subsequent reads from VID_HDF give access to the header data which follows the start code.

Scanning for start codes recommences on either the next DSYNC or a write to HDS (VID_HDS). Whenever a start code has been detected, the VID_HDF register must be read in order for the start code detector to restart correctly. The number of reads before a manual or automatic (DSYNC) restart must always be even.

The first start code search after a hard or soft reset is a search for a sequence header start code; all other start codes are ignored. When this start code has been read, all subsequent searches look for any start codes other than slice start codes.

The two status bits HFE (header FIFO empty) and HFF (header FIFO full) in VID_STA indicate the state of the header FIFO. Reading from HDF must never be performed if HFE is 1. HFF is set whenever the header FIFO contains at least 66 bytes.

The start code detector can also be programmed to stop on the first slice of the picture. This allows the use of the start code search even after reception of the picture start code. All header data that is not used by the application can then be skipped without risk, in order to jump to the next picture start code.

This mode is enabled by setting bit SOS (VID_HDS). To differentiate between first slice start code (00 00 01 01) and other start codes, it is possible to detect at which position (MSB or LSB) the last byte of start code is positioned in the VID_HDF register. When set, register bit SCM (VID_HDS) indicates that the last byte of start code is held by the MSB of VID_HDF. It is zero otherwise.

34.5.3 Handling time stamps

The video decoder accepts MPEG-1 system streams and MPEG-2 packet streams.

For video/audio elementary streams, time stamps contained in the video packet headers are associated with the picture decode time. This is important because the number of pictures which may be stored in the bit buffer at any instant is unknown. Therefore there is a variable delay between the input of a picture into the bit buffer and its entry into the decoding pipeline.

There is a 24-bit counter at the input and the output of the CD FIFO (bit buffer) header FIFO chain, as shown in [Figure 79](#). Each time a byte is written into the CD FIFO the counter CDCOUNT is incremented. Each time a 16-bit word is read from the header FIFO, the counter SCDCOUNT is incremented. Both of the counters are reset by a hard or soft reset. Both are modulo 2^{24} , that is, the state following 0xFF FFFF is 0x00 0000.

The diagram illustrates the ST20 video decoder architecture. It features an **ST20 arbiter and bus** connected to three CD FIFOs: **Audio CD FIFO**, **Subpicture CD FIFO**, and **Video CD FIFO (128 bytes)**. The **Video CD FIFO** is also connected to a **PES parser**. The **Video CD FIFO** feeds into the **SDRAM arbiter (LMC)**, which is connected to **SDRAM**. The **SDRAM** contains a **Video bit-buffer**. The **SDRAM arbiter (LMC)** also feeds into the **Video decoder**, which has **VLD** and **SCD** sections. The **Video decoder** is connected to the **SDRAM** and the **Video bit-buffer**. The **Video decoder** also receives input from the **Subpicture CD FIFO** and the **Audio CD FIFO**. The **Video decoder** outputs are connected to the **SDRAM** and the **Video bit-buffer**. The **Video decoder** also receives input from the **Subpicture CD FIFO** and the **Audio CD FIFO**. The **Video decoder** outputs are connected to the **SDRAM** and the **Video bit-buffer**. The **Video decoder** also receives input from the **Subpicture CD FIFO** and the **Audio CD FIFO**. The **Video decoder** outputs are connected to the **SDRAM** and the **Video bit-buffer**.

The time stamp association information is available in the PES_TS register. The same mechanism is implemented for the DSM trick mode association (register PES_TMn).

Note: The video decoder only operates in bandwidth reduction mode.

Finally, the decoded picture is written back into the memory, from where it can be accessed by the display unit for output.

The pipeline is also able to skip through picture data for various reasons. The different possibilities are:

- **Skip to next sequence**

This occurs unconditionally on the first instruction execution after a hard or soft reset (see [Section 34.4: Reset on page 284](#)). Compressed data is skipped until the first picture start code following a sequence start code is found. The pipeline then indicates task completion and waits for a new instruction.

- **Skip to next picture**

This occurs either after a pipeline reset (see [Section 34.4: Reset on page 284](#)) or when the decoding instruction specifies that one or two pictures should be skipped (see [Section 34.10.1: Decoding task on page 299](#)). In the first case, compressed data is skipped until the next picture start code is found, after which the pipeline indicates task completion and waits for a new instruction. In the second case, after the skipping operation the decoding of the following picture is started immediately.

- **Skip to next slice**

This occurs after automatic error concealment (see [Section 34.10.2: Error recovery and missing macroblock concealment on page 301](#)). Compressed data is skipped until the next slice start code in the picture is found, after which normal decoding resumes.

Before starting to decode a sequence, certain static parameters must be set up.

These are:

- MPEG-1 or MPEG-2 mode selection,
- bit MP2 (VID_PPR2) must be set for an MPEG-2 sequence, reset for an MPEG-1 sequence,
- decoded picture size.

Register VID_DFW must be set up with the picture width in macroblocks, and register VID_DFS must be set up with the number of macroblocks in the picture.

Decoding is enabled by setting bit EDC (VID_CTL).

34.7 Quantization table loading

The two quantization matrices (intra and nonintra) used by the inverse quantizer must be initialized by the user. There are no built in quantization matrices. Therefore, they must be loaded either with default matrices or with those extracted from the bit stream by the ST20.

The quantization tables are double buffered. This enables one or both tables to be updated without disturbing the decoding task in progress.

The video decoder maintains two bits which record whether one or both of the tables have been modified. A modified table is automatically brought into operation at the start of the next decoding operation, that is, when the next DSYNC occurs.

After a hard reset, the same pair of tables is always selected. The data previously loaded into the tables is not affected. Other types of reset have no effect on the quantization tables.

The quantization tables are written at the address held in the register VID_QMW. Bit QMI (VID_HDS) is used to select the intra or nonintra quantization table; when it is set, the intra table is selected; when clear, the nonintra table is selected.

34.8 Memory mapping of data

Three types of external SDRAM can be mapped from the STi5516:

- 1 or 2 x 16-Mbit SDRAM,
- 1 x 64-Mbit SDRAM,
- 1 x 128-Mbit SDRAM.

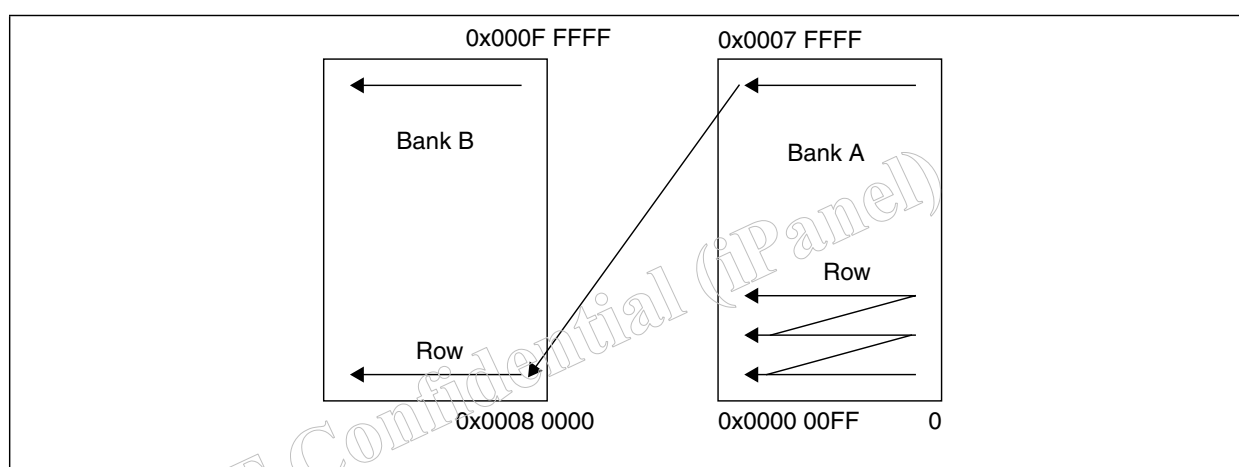
This section discusses video decoder memory (SDRAM) addressing, 32-bit word addressing for the CPU and 64-bit word addressing for FIFO for both of these types of external SDRAM.

34.8.1 Mapping 1 or 2 x 16-Mbit SDRAM

Video decoder memory SDRAM addressing (for 1 or 2 x 16-Mbit SDRAM)

The locations in an SDRAM are addressed row by row, bank A then bank B, as shown [Figure 80](#).

Figure 80: Standard addressing in a SDRAM (16-bit words) for 1 or 2 x 16-Mbit SDRAM



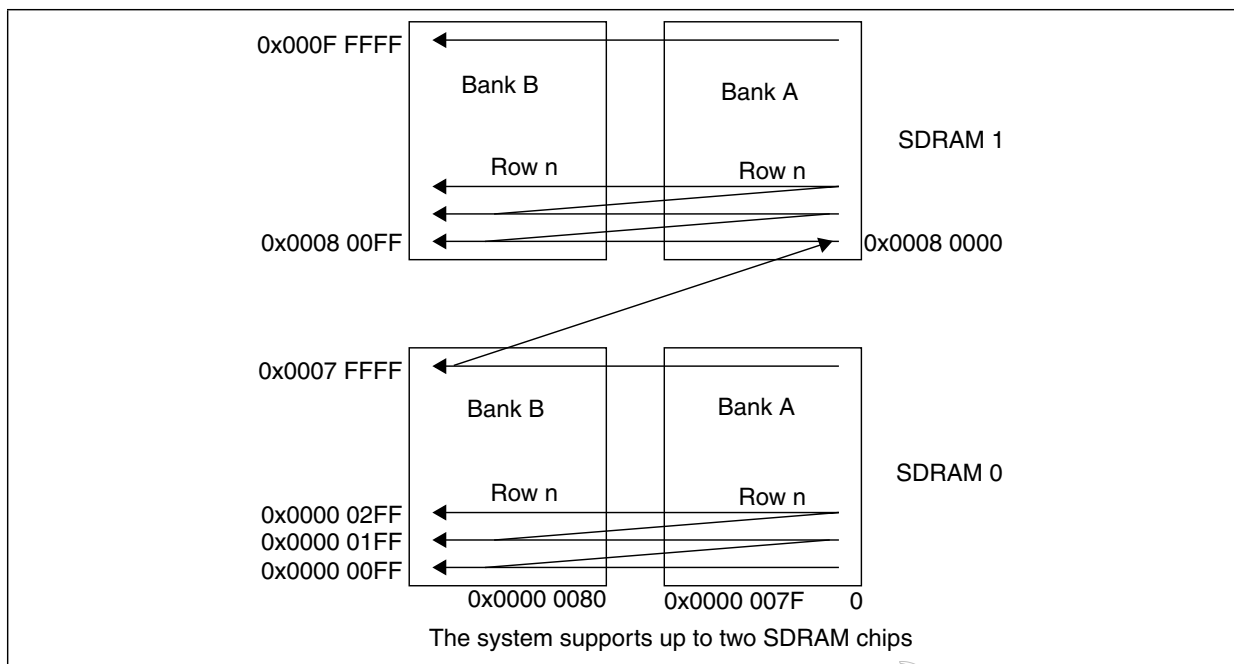
32-bit word addressing for the CPU (for 1 or 2 x 16-Mbit SDRAM)

The CPU accesses the SDRAM by using a 19-bit address for each 32-bit word. It is the task of the SDRAM memory controller to remap the logical address space of the CPU on to the SDRAM address space.

The logical address map seen by the CPU is different from the one described above. For each row, both banks are used. The addresses seen by the CPU through the SDRAM interface are counted in the following order:

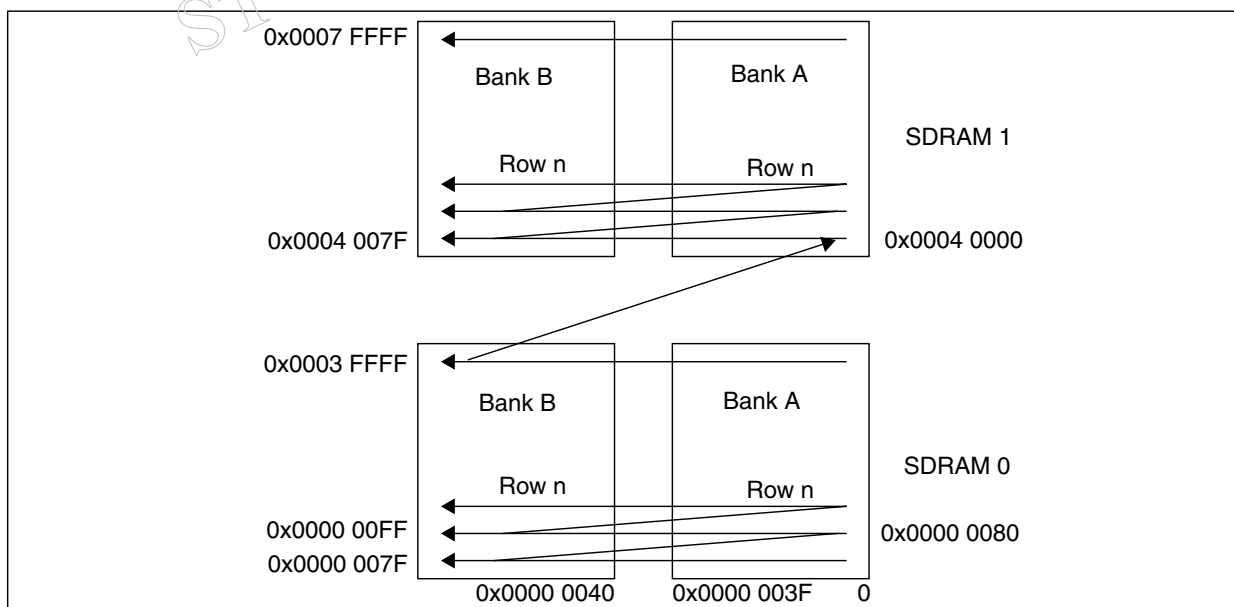
Bank A row0 → Bank B row0 → Bank A row1 → Bank B row1 →... and so on, as illustrated in [Figure 81](#).

When using a second SDRAM chip, addresses continue in a similar way, starting from the next address above the first SDRAM, as illustrated in [Figure 81](#). A maximum of two SDRAM chips are supported.

Figure 81: 32-bit word addressing, as seen by the CPU for 1 or 2 x 16-Mbit SDRAM**64-bit word addressing for FIFO processes (for 1 or 2 x 16-Mbit SDRAM)**

The video decoder uses circular buffers mapped into external SDRAM which act as software FIFOs. The processes pertaining to these circular buffers are managed with a 64-bit granularity. The memory mapping for these buffers is similar to that of the CPU and is shown in [Figure 82](#).

When using a second SDRAM chip, addresses continue in a similar way, starting from the next address above the first SDRAM. A maximum of two SDRAM chips is supported.

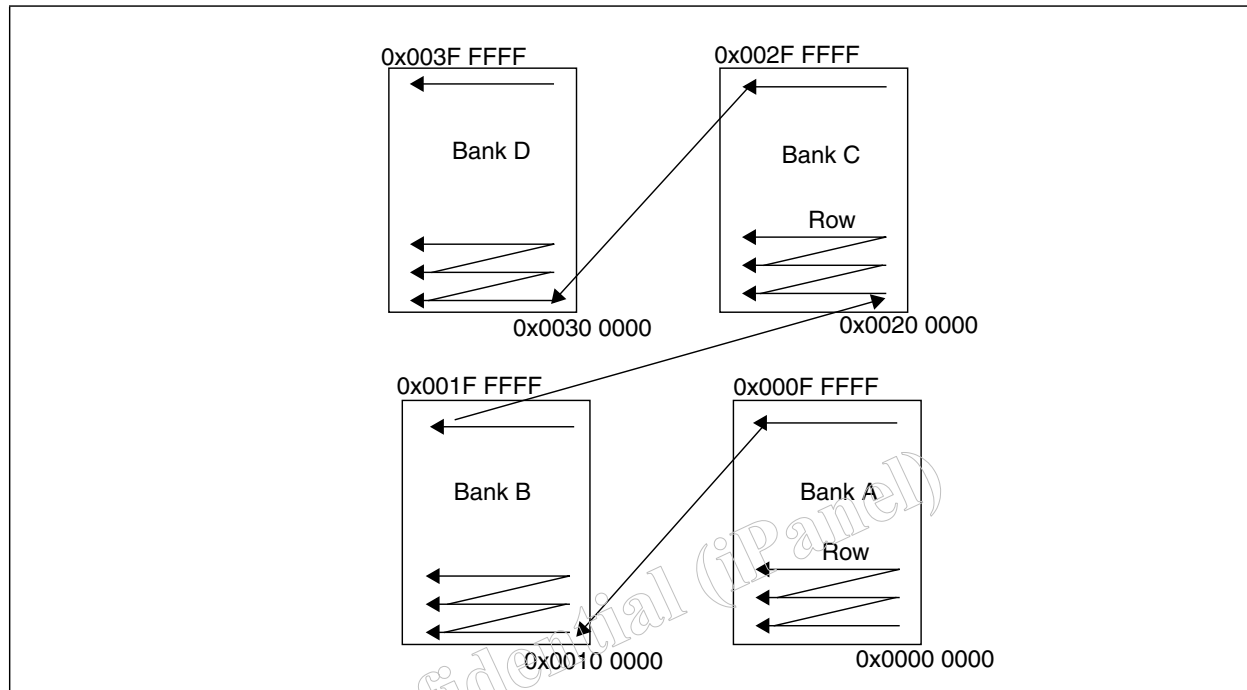
Figure 82: 64-bit word addressing for FIFO processes for 1 or 2 x 16-Mbit SDRAM

34.8.2 Mapping 1 x 64-Mbit SDRAM

Video decoder memory SDRAM addressing (for 1 x 64-Mbit SDRAM)

The locations in an SDRAM are addressed row by row, bank A, B, C and then bank D, as shown in [Figure 83](#).

Figure 83: Standard addressing in a SDRAM (16-bit words) for 1 x 64-Mbit SDRAM



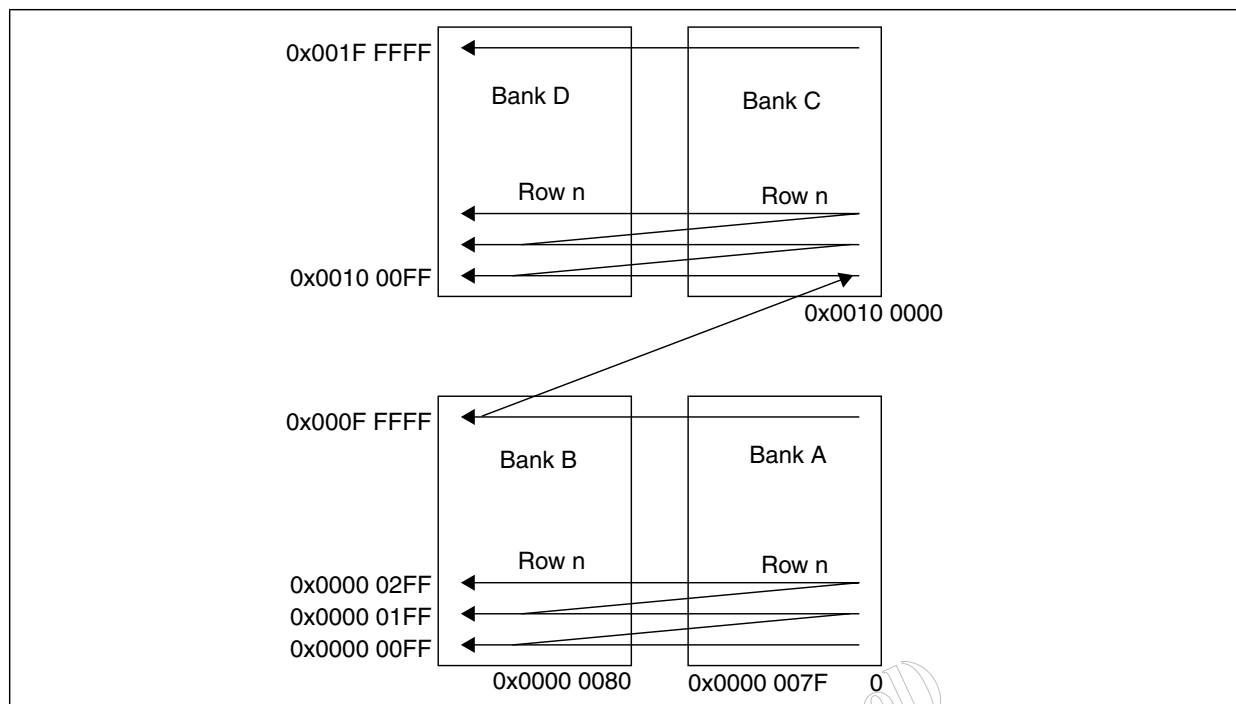
32-bit word addressing for the CPU (for 1 x 64-Mbit SDRAM)

The CPU accesses the SDRAM by using a 21-bit address for each 32-bit word. It is the task of the SDRAM memory controller to remap the logical address space of the CPU on to the SDRAM address space.

The logical address map seen by the CPU is different from the one described above. For each row, both banks are used. The addresses seen by the CPU through the SDRAM interface are counted in the following order:

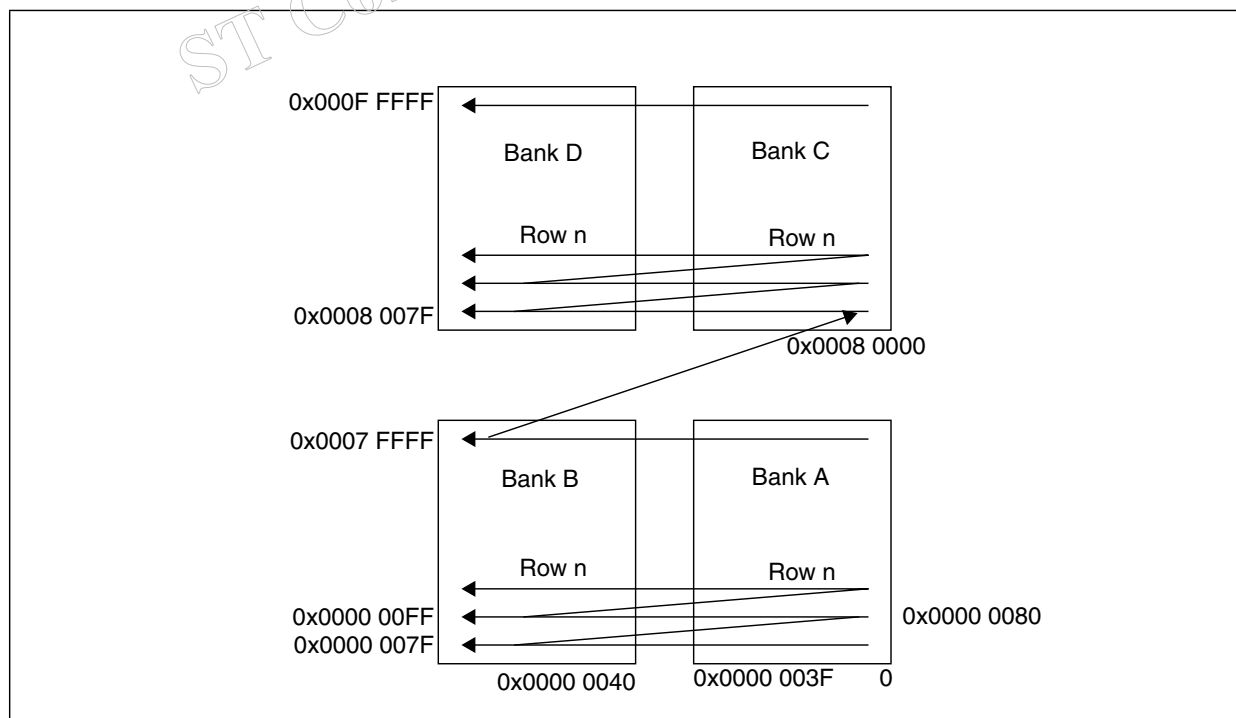
Bank A row0 → Bank B row0 → Bank A row1 → Bank B row1 → ... → Bank A row 4005 → Bank B row 4005, → Bank C row 0 → Bank D row 0... and so on, as illustrated in [Figure 84](#).

Figure 84: 32-bit word addressing, as seen by the CPU for 1 x 64-Mbit

**64-bit word addressing for FIFO processes (for 1 x 64-Mbit SDRAM)**

The video decoder uses circular buffers mapped into external SDRAM which act as software FIFOs. The processes pertaining to these circular buffers are managed with a 64-bit granularity. The memory mapping for these buffers is similar to that of the CPU and is shown in Figure 85.

Figure 85: 64-bit word addressing for FIFO processes for 1 x 64-Mbit SDRAM

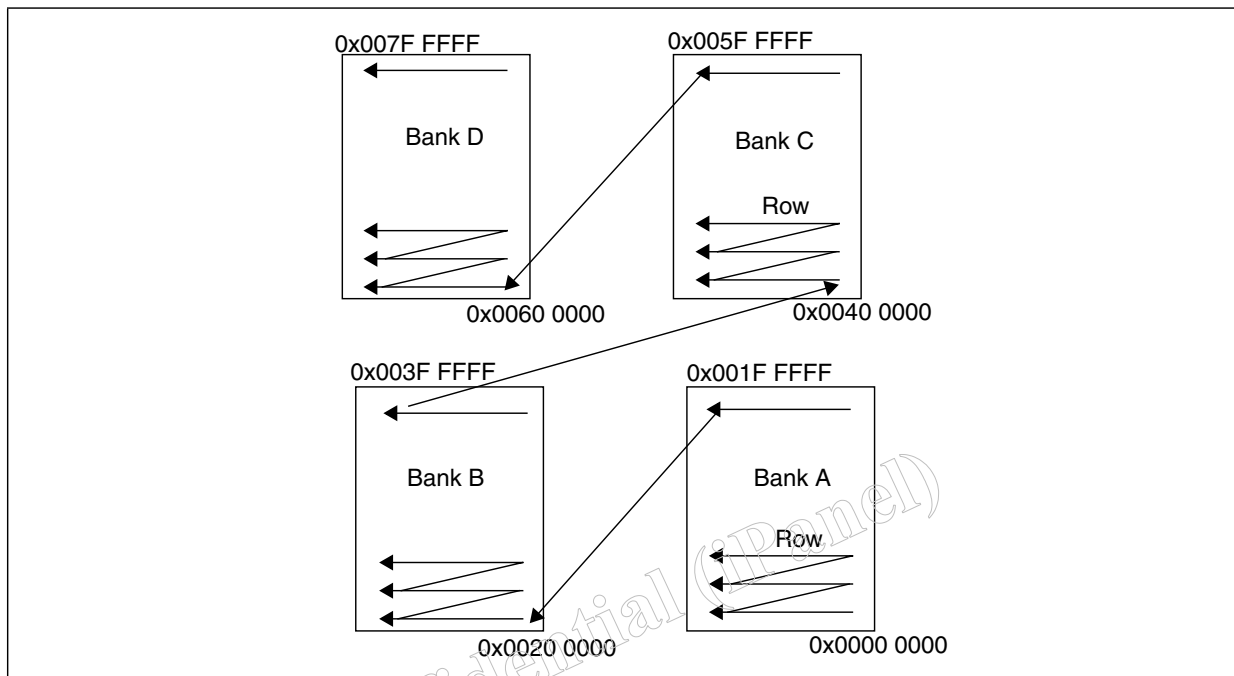


34.8.3 Mapping 1 x 128-Mbit SDRAM

Video decoder memory SDRAM addressing (for 1 x 128-Mbit SDRAM)

The locations in an SDRAM are addressed row-by-row, bank A, B, C then bank D, as shown in [Figure 86](#).

Figure 86: Standard addressing in a SDRAM (16-bit words) for 1 x 128-Mbit SDRAM



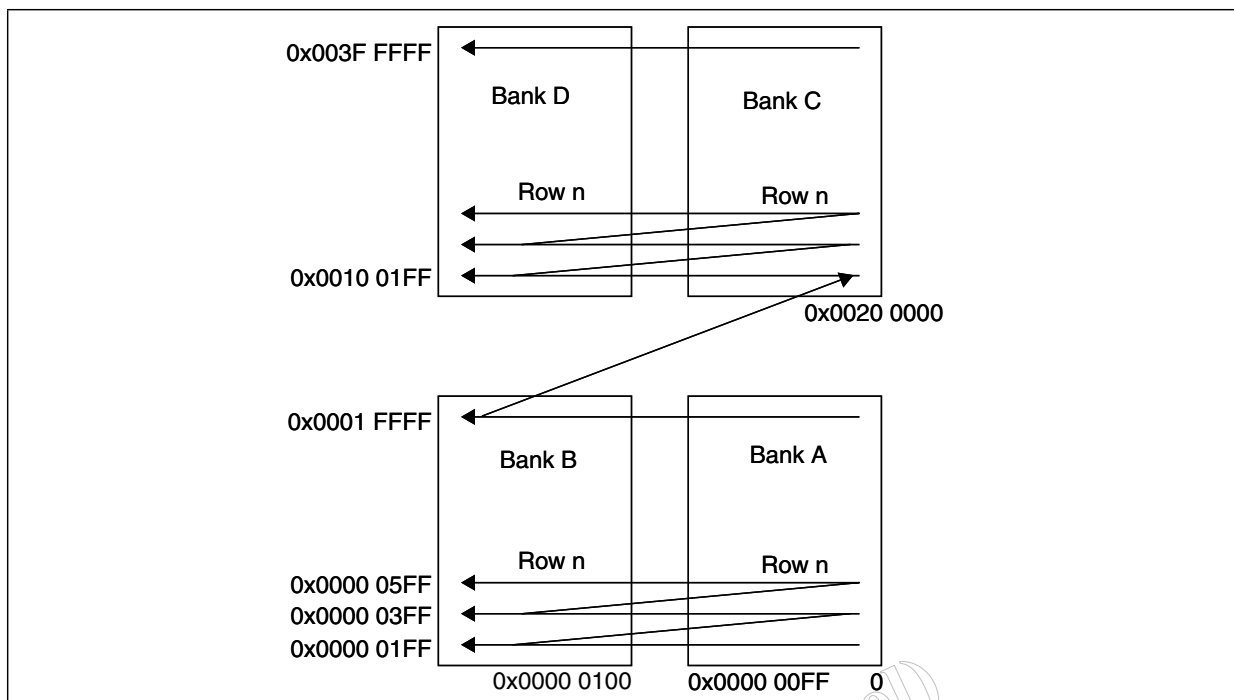
32-bit word addressing for the CPU (for 1 x 128-Mbit SDRAM)

The CPU accesses the SDRAM by using a 22-bit address for each 32-bit word. It is the task of the SDRAM memory controller to remap the logical address space of the CPU onto the SDRAM address space.

The logical address map seen by the CPU is different from the one described above. For each row, both banks are used. The addresses seen by the CPU through the SDRAM interface are counted in the following order:

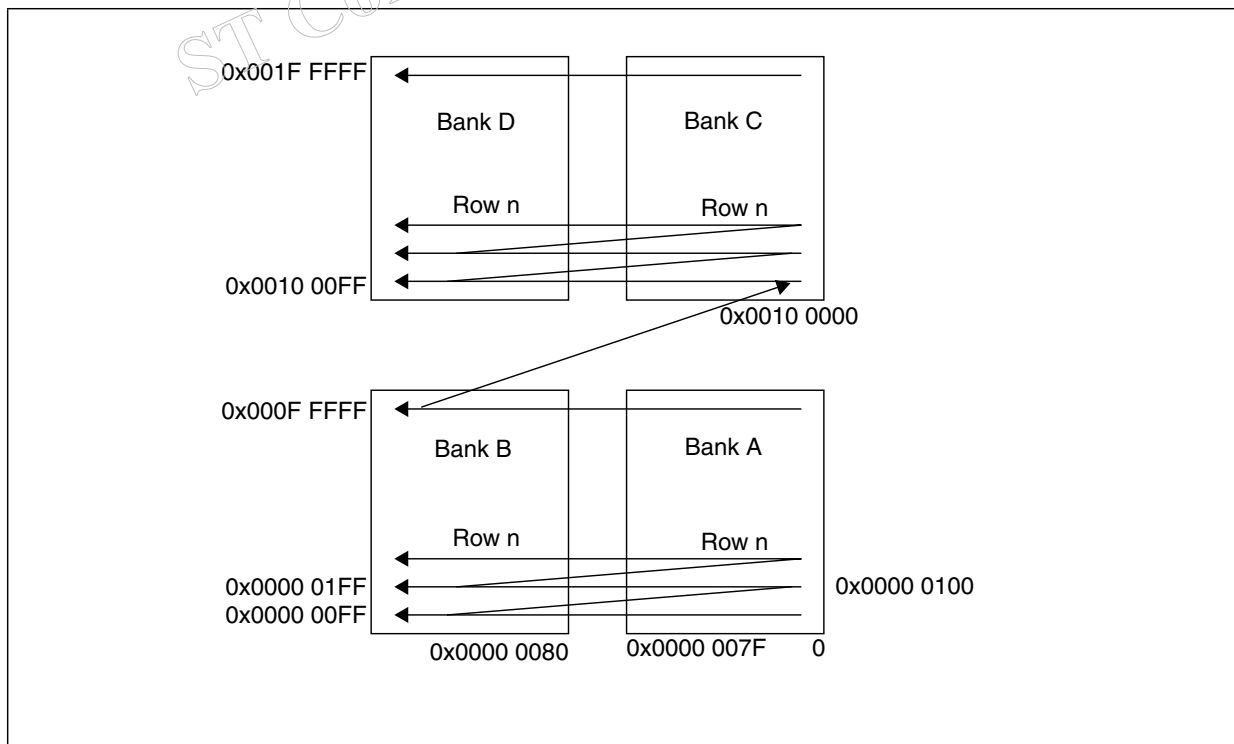
Bank A row0 → Bank B row0 → Bank A row1 → Bank B row1 → ... → Bank A row 4096 → Bank B row 4096 → Bank C row 0 → Bank D row 0... and so on, as illustrated in [Figure 87](#).

Figure 87: 32-bit word addressing for FIFO processes for 1 x 128-Mbit SDRAM

**64-bit word addressing for FIFO processes (for 1 x 128-Mbit SDRAM)**

The video decoder uses circular buffers mapped into external SDRAM which act as software FIFOs. The processes pertaining to these circular buffers are managed with a 64-bit granularity. The memory mapping for these buffers is similar to that of the CPU and is shown in Figure 88.

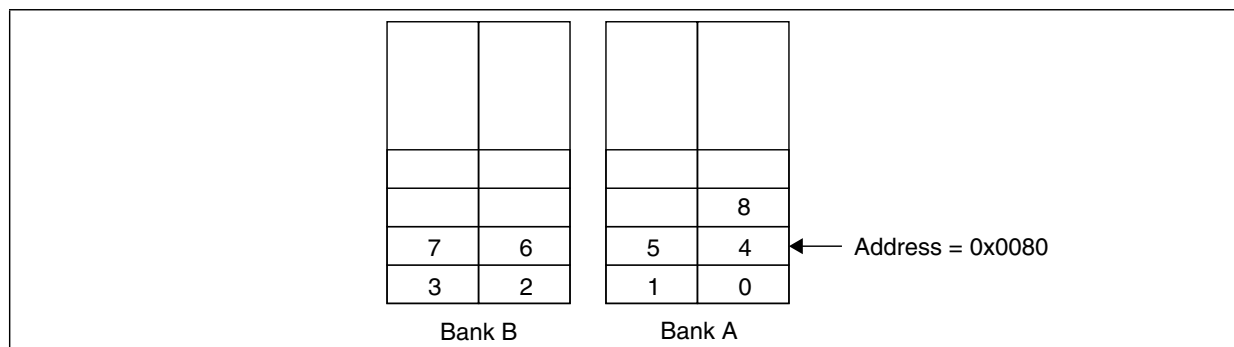
Figure 88: 64-bit word addressing for FIFO processes for 1 x 128-Mbit SDRAM



34.8.4 Memory segments for 16- or 64-Mbit SDRAM

The circular buffer start and end pointers are programmed by the user, in segments, where each segment is 256 bytes. The values in the configuration registers are numbers of segments. For example, a value of 4 means 4×256 bytes = 1 Kbyte or 128×64 -bit words. This would result in a pointer pointing to a 64-bit word address of 128 (0x0080). This address would be physically mapped to the first word in the second row of bank A of SDRAM 0, as shown in Figure 89.

Figure 89: SDRAM segments as seen by the user



Arrangement of pixel pairs inside a luma SDRAM row

Every SDRAM row in a luma frame contains 256 16-bit words and can store up to two luma macroblocks. Every 16-bit word contains a pair of horizontally adjacent luma pixels. The row itself stores a pair of horizontally adjacent luma macroblocks. The pixel pairs are arranged in line order; the first 16 words store the first line of pixels for the two macroblocks, the next 16 words store the second line and so on. This is shown in Table 132.

Table 132: Arrangement of pixel pairs in a luma SDRAM row

16-bit word addresses in SDRAM row	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
Arrangement of luma pixel pairs	Y = 2 pixels	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Macroblock 0								Macroblock 1							
16-bit word addresses in SDRAM row	0xF0	0xF1	0xF2	0xF3	0xF4	0xF5	0xF6	0xF7	0xF8	0xF9	0xFA	0xFB	0xFC	0xFD	0xFE	0xFF
Arrangement of luma pixel pairs	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Arrangement of pixel pairs inside a chroma SDRAM row

Every SDRAM row in a chroma frame contains 256 16-bit words and can store up to four chroma macroblocks. Every 16-bit word contains a pair of horizontally adjacent 8-bit chroma pixels.

The row stores pixel pairs in line order for macroblocks 0 and 1 and then macroblocks 2 and 3. The Cb and Cr words are interleaved two by two in the linear addressing order, as shown in Table 133.

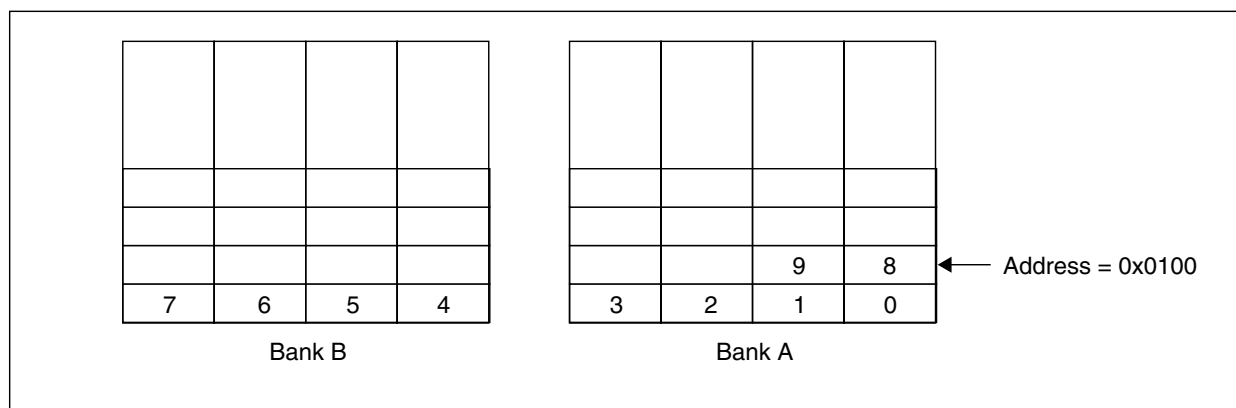
Table 133: Arrangement of pixel pairs in a chroma SDRAM row

16-bit word addresses in SDRAM row	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
Arrangement of luma pixel pairs	Cb = 2 pixels	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
	Macroblock 0								Macroblock 1							
16-bit word addresses in SDRAM row	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77	0x78	0x79	0x7A	0x7B	0x7C	0x7D	0x7E	0x7F
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
16-bit word addresses in SDRAM row	0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
	Macroblock 0								Macroblock 1							
16-bit word addresses in SDRAM row	0xF0	0xF1	0xF2	0xF3	0xF4	0xF5	0xF6	0xF7	0xF8	0xF9	0xFA	0xFB	0xFC	0xFD	0xFE	0xFF
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr

34.8.5 Memory segments for 128-Mbit SDRAM

The circular-buffer start and end pointers are programmed by the user, in segments, where each segment is 256 bytes. The values in the configuration registers are numbers of segments. These values must be a multiple of 8 for 128-Mbit SDRAM. For example, a value of 8 means $8 \times 256 \text{ bytes} = 2 \text{ Kbytes}$ or $256 \times 64\text{-bit words}$. This would result in a pointer pointing to a 64-bit word address of 256 (0x0100). This address would be physically mapped to the first word in the second row of bank A of SDRAM 0, as shown in Figure 90.

Figure 90: SDRAM segments (128-Mbit SDRAM) as seen by the user



Arrangement of pixel pairs inside a luma SDRAM row

Every SDRAM row of one bank in a luma frame contains 512 16-bit words and can store up to four luma macroblocks. Every 16-bit word contains a pair of horizontally adjacent luma pixels. The row itself stores 2 pairs of horizontally adjacent luma macroblocks. The pixel pairs are arranged in line order; the first 16 words store the first line of pixels for the two macroblocks, the next 16 words the second line and so on, as shown in [Table 134](#).

Table 134: Arrangement of pixel pairs in a chroma SDRAM row

16-bit word addresses in SDRAM row	0x000	0x001	0x002	0x003	0x004	0x005	0x006	0x007	0x008	0x009	0x00A	0x00B	0x00C	0x00D	0x00E	0x00F
Arrangement of luma pixel pairs	Y = 2 pixels	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Macroblock 0								Macroblock 1							
16-bit word addresses in SDRAM row	0x0F0	0x0F1	0x0F2	0x0F3	0x0F4	0x0F5	0x0F6	0x0F7	0x0F8	0x0F9	0x0FA	0x0FB	0x0FC	0x0FD	0x0FE	0x0FF
Arrangement of luma pixel pairs	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
16-bit word addresses in SDRAM row	0x100	0x101	0x102	0x103	0x104	0x105	0x106	0x107	0x108	0x109	0x10A	0x10B	0x10C	0x10D	0x10E	0x10F
Arrangement of luma pixel pairs	Y = 2 pixels	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Macroblock 0								Macroblock 1							
16-bit word addresses in SDRAM row	0x1F0	0x1F1	0x1F2	0x1F3	0x1F4	0x1F5	0x1F6	0x1F7	0x1F8	0x1F9	0x1FA	0x1FB	0x1FC	0x1FD	0x1FE	0x1FF
Arrangement of luma pixel pairs	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Arrangement of pixel pairs inside a chroma SDRAM row

Every SDRAM row of one bank in a chroma frame contains 512 16-bit words and can store up to 8 chroma macroblocks. Every 16-bit word contains a pair of horizontally adjacent 8-bit chroma pixels.

The row stores pixel pairs in line order for macroblocks 0 and 1 and then macroblocks 2 and 3. The Cb and Cr words are interleaved two by two in linear addressing order, as in [Table 135](#).

Table 135: Arrangement of pixel pairs in a chroma SDRAM row

16-bit word addresses in SDRAM row	0x000	0x001	0x002	0x003	0x004	0x005	0x006	0x007	0x008	0x009	0x00A	0x00B	0x00C	0x00D	0x00E	0x00F
Arrangement of luma pixel pairs	Cb = 2 pixels	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
	Macroblock 0								Macroblock 1							
16-bit word addresses in SDRAM row	0x070	0x071	0x072	0x073	0x074	0x075	0x076	0x077	0x078	0x079	0x07A	0x07B	0x07C	0x07D	0x07E	0x07F
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
16-bit word addresses in SDRAM row	0x080	0x081	0x082	0x083	0x084	0x085	0x086	0x087	0x088	0x089	0x08A	0x08B	0x08C	0x08D	0x08E	0x08F
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
	Macroblock 2								Macroblock 3							
16-bit word addresses in SDRAM row	0x0F0	0x0F1	0x0F2	0x0F3	0x0F4	0x0F5	0x0F6	0x0F7	0x0F8	0x0F9	0x0FA	0x0FB	0x0FC	0x0FD	0x0FE	0x0FF
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
16-bit word addresses in SDRAM row	0x100	0x101	0x102	0x103	0x104	0x105	0x106	0x107	0x108	0x109	0x10A	0x10B	0x10C	0x10D	0x10E	0x10F
Arrangement of luma pixel pairs	Cb = 2 pixels	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
	Macroblock 4								Macroblock 5							
16-bit word addresses in SDRAM row	0x170	0x171	0x172	0x173	0x174	0x175	0x176	0x177	0x178	0x179	0x17A	0x17B	0x17C	0x17D	0x17E	0x17F
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
16-bit word addresses in SDRAM row	0x180	0x181	0x182	0x183	0x184	0x185	0x186	0x187	0x188	0x189	0x18A	0x18B	0x18C	0x18D	0x18E	0x18F
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr
	Macroblock 6								Macroblock 7							
16-bit word addresses in SDRAM row	0x1F0	0x1F1	0x1F2	0x1F3	0x1F4	0x1F5	0x1F6	0x1F7	0x1F8	0x1F9	0x1FA	0x1FB	0x1FC	0x1FD	0x1FE	0x1FF
Arrangement of luma pixel pairs	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr	Cb	Cb	Cr	Cr

34.9 Using picture pointers

Before decoding a picture, the following frame buffer pointers must be set up.

- VID_RFC, VID_RFP for reconstructed frame pointers for chroma and luma.
These pointers define the memory buffer to which the decoded picture is written.
- VID_FFC, VID_FFP for forward prediction frame pointers for chroma and luma.
These pointers define the areas in memory from which the predictors are fetched.
- VID_BFC, VID_BFP for backward prediction frame pointers for chroma and luma.
These pointers define the areas in memory from which the predictors are fetched.

The displayed frame pointers, VID_DFC, VID_DFP, are described in [Section 36.4: Subpicture display on page 336](#).

The following rules must be followed when using prediction frame pointers.

- Pictures are always stored as frames of interleaved lines. Therefore, to access a field (top or bottom), the starting address of the frame must be defined.
- Pframe picture (frame, field or dual-prime prediction): VID_FFP and VID_FFC are set to the address of the predictor frame (in which the two predictor fields lie). VID_BFP and VID_BFC are not used.
- Bframe picture (frame or field prediction): VID_FFP and VID_FFC are set to the address of the forward predictor frame (in which the two predictor fields lie). VID_BFP and VID_BFC are set to the address of the backward predictor frame (in which the two predictor fields lie).
- Pfield picture (field, 16 x 8 or dual-prime prediction): When decoding either field, VID_FFP and VID_FFC are set to the address of the previous decoded I or P frame. VID_BFP and VID_BFC are not used.
- Bfield picture (field or 16 x 8 prediction): VID_FFP and VID_FFC are set to the address of the frame in which the two forward predictor fields lie. VID_BFP and VID_BFC are set to the address of the frame in which the two backward predictor fields lie.
- Ipictures: For Ipicture decoding, no predictors are necessary, but VID_FFP and VID_FFC must be set to the address of the last decoded Ipicture or Ppicture for use by the automatic error concealment function.

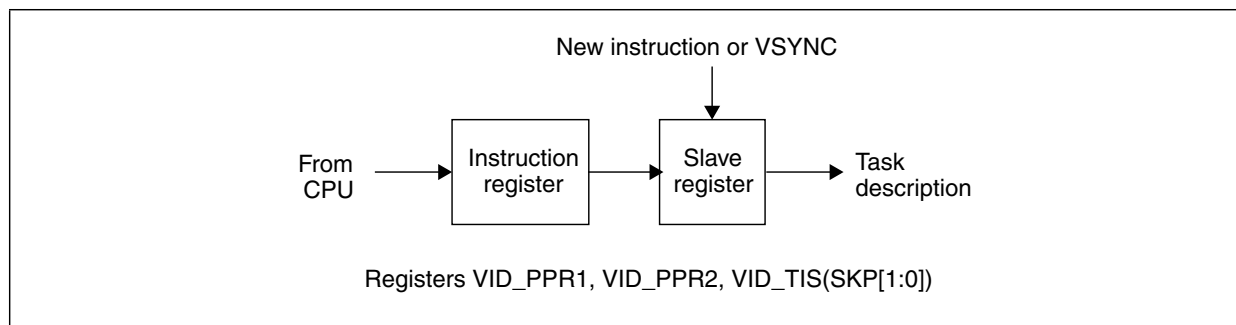
34.10 Video pipeline

34.10.1 Decoding task

A task is a single picture decoding operation. A task is specified by the task description or instruction, which is set up before the decoding of each picture. A task starts when the internal signal DSYNC is generated. A task completes (the decoder becomes idle) when the picture is entirely reconstructed in the memory and the picture header of the following picture is detected by the pipeline. The instruction is double buffered, so that during execution of a decoding task, the instruction for the next task can be set up by the CPU. When the next instruction is activated, a DSYNC can be generated and the next decoding task started. The buffering mechanism is illustrated in [Figure 91](#). Some instruction bits are latched by VSYNC, others by a signal from the pipeline controller new instruction.

The instruction is written into registers VID_PPR1 and VID_PPR2. If a new instruction is not written, the task descriptor is the same as the previous one.

Figure 91: Instruction buffering



Normally, it is VSYNC starts the execution of a new instruction and, thus, the generation of DSYNC. If however, a VSYNC occurs before task completion (that is, before the pipeline becomes idle), the start of the next task is delayed until the present one is completed. In this way, the picture decoding can extend beyond the nominal period by one or two VSYNC periods.

Three status bits in VID_STA (and thus interrupts) are associated with pipeline control:

- PSD indicates the occurrence of a DSYNC,
- PII indicates that the pipeline is idle,
- DEI indicates that the decoder is idle, that is, the pipeline is idle and the next picture start code has been found.

The operation of the pipeline controller is shown in [Figure 92](#). The abbreviations used in the diagram are explained in [Table 136: State transition abbreviations on page 301](#).

Figure 92: Task control state diagram

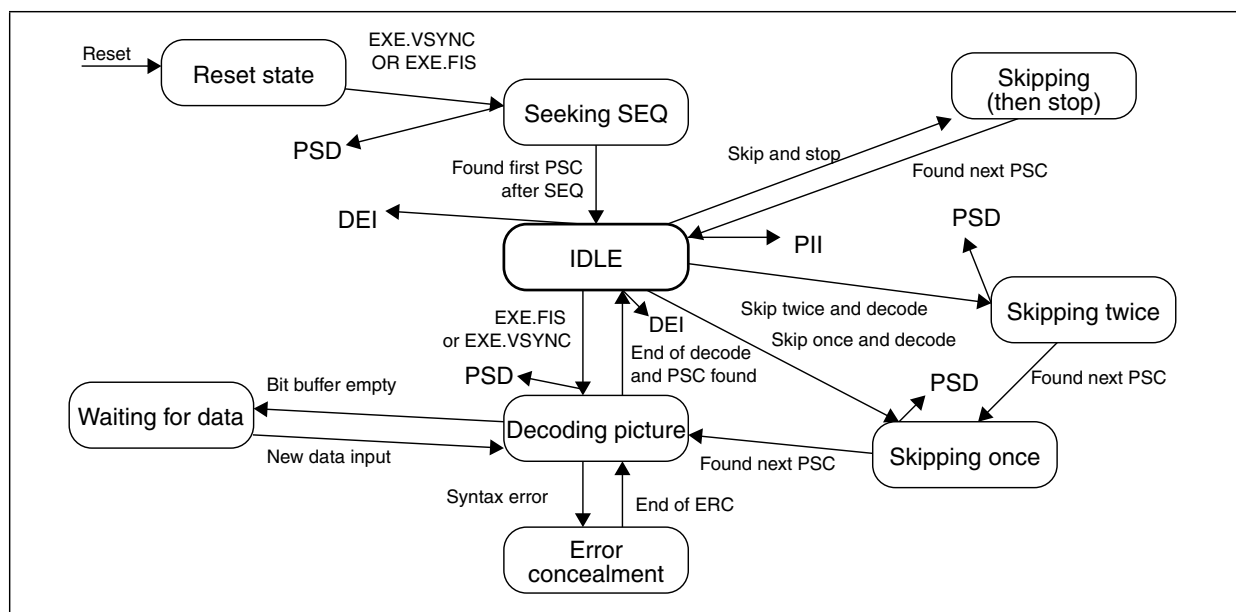


Table 136: State transition abbreviations

Abbreviation	Meaning
ERC	Automatic error concealment
EXE.FIS	Both EXE and FIS are set in VID_TIS
EXE.VSYNC	Bit EXE set when external VSYNC occurs
PII	Pipeline idle
DEI	Decoder idle interrupt generated
PSC	Picture start code
PSD	Pipeline start decode interrupt generated
SEQ	Sequence start code
Skip and decode	VID_TIS = EXE SKP[01] and VSYNC occurred
Skip and stop	VID_TIS = EXE SKP[11] and VSYNC occurred
Skip twice and decode	VID_TIS = EXE SKP[10] and VSYNC occurred

The instruction bits which affect state transitions are EXE and FIS in VID_TIS. The events to which the controller responds are:

- VSYNC, either a VSYNC top or a VSYNC bottom,
- IDLE representing the idle state of the pipeline.

34.10.2 Error recovery and missing macroblock concealment

For the video decoder, there are four levels of error detection and recovery:

- bit stream syntax error detection with the option of automatic missing macroblock concealment,
- bit stream semantic error detection with the option of automatic concealment or skip to the next picture,
- pipeline overflow or underflow error detection,
- user initiated skip to next sequence using soft reset.

Syntax error detection and concealment

In normal operation of the STi5516, error concealment must always be enabled, that is, EDC (VID_CTL) should be reset.

If the VLD detects a syntax error in the bit stream, the pipeline copies macroblocks from the previous picture using the motion vectors reconstructed for the previous row of macroblocks in the current picture, while scanning the bit stream until a slice start code is detected. At this point normal decoding resumes. If the slice in which the error occurred was the last one in the picture, concealment continues until the end of the picture, at which time the pipeline stops normally (assuming that the following picture start code is intact).

The concealment macroblocks are accessed using the pointers VID_FFP and VID_BFP. Lost macroblocks in the first row are copied directly from the previous pictures (that is, as Pmacroblocks with zero motion vectors). If an intrapicture is coded with concealment motion vectors, these are used. If not, then the concealment is a simple copy from the previous picture using zero vectors. Even in intrapictures, the pointer VID_FFP must be set up.

Table 137 gives the rules that are used for fetching concealment macroblocks.

Table 137: Rules for fetching concealment macroblocks

Picture type	Macroblock type	Fetch rule
Ipicture	Imacroblock without vectors	Copy with zero motion.
	Imacroblock with vectors	Copy as forward predicted macroblock.
Ppicture	Imacroblock without vectors	Copy with zero motion.
	Imacroblock with vectors	Copy as forward predicted macroblock.
	Pmacroblock	Copy using stored vector.
	Pfield macroblock	Copy in field mode using both vectors.
	Skipped macroblock	Copy with zero vector.
	Dual-prime macroblock	Copy using stored vector.
Bpicture	Imacroblock without vectors	Copy with zero motion.
	Imacroblock with vectors	Copy as forward predicted macroblock.
	Forward macroblock	Copy using stored vector.
	Backward macroblock	Copy using stored backward vector.
	Bidirectional macroblock	Only the forward vectors are stored, concealed as forward macroblock.
	Skipped macroblock	Copy in frame mode using the same mode and vectors as the previous macroblock.

Overflow or underflow error

An overflow error occurs whenever the pipeline reconstructs more macroblocks than are defined by the decoded picture size, VID_DFS. This can occur when the input data to the decoder contains undetected errors. This condition is signalled by bit SER (VID_STA). Decoding is automatically halted when this error is detected. In order to restart decoding, a pipeline reset must be performed.

An underflow error occurs whenever the pipeline reconstructs less macroblocks than are defined by the decoded picture size, VID_DFS. This condition is signalled by bit PDE (VID_STA). Decoding is automatically halted when this error occurs. In order to restart decoding, a pipeline reset must be performed.

34.11 PES parser

34.11.1 Description

The PES parser is situated between the ST20 arbiter/bus and the compressed data FIFOs of the video/audio core. It has a 100 Mbit/s (maximum burst) bit rate, and allows the following input streams:

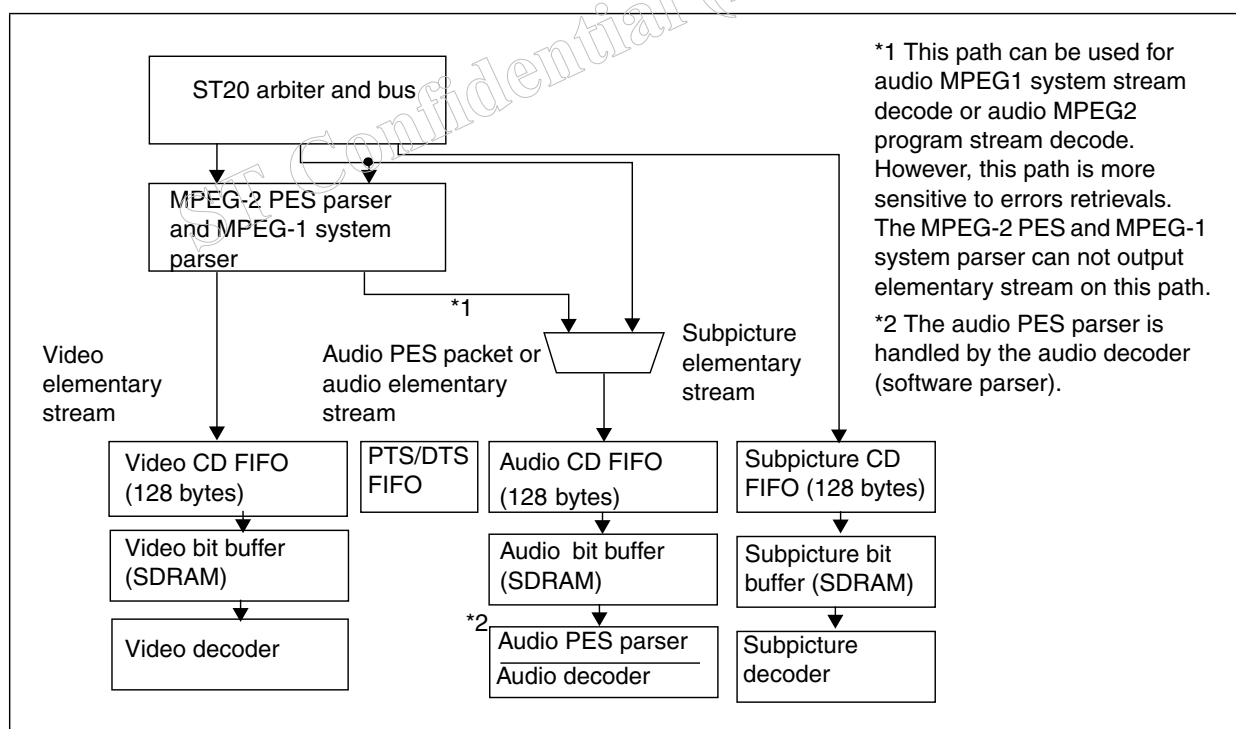
- MPEG-2 PES (Packetized PES), ISO 13818-1,
- MPEG-1 system layer (ISO 11172-1).

The MPEG-2 PES and MPEG-1 system parser accepts PES streams in the same way that pure audio or video streams are accepted.

For packetized elementary stream data which is demultiplexed from a transport stream (MPEG-2), the data stream consists of concatenated, incomplete packets of audio, and video PES. To handle this configuration, the STi5516 contains two separate parsers: one for the audio (audio PES parser in audio decoder) and one for the video (MPEG-2 PES and MPEG-1 system parser).

As the audio or video data is input, it is demultiplexed by each parser and the audio/video streams are placed in their respective buffers. For program stream data or MPEG-1 systems stream data, the audio and video packets are complete so that a single parser (MPEG-2 PES and MPEG-1 system parser) can be used. The packets are internally separated into video and audio streams. If required, the two parsers can still be used but the packets must be separated by the ST20 (recommended mode). See [Figure 93](#).

Figure 93: System parser internal architecture



When the MPEG-2 PES and MPEG-1 system parser is configured to accept MPEG-2 PES audio/video packets (mode 3), the parser extracts audio and video bit streams in accordance with the programmed stream ID. For the audio stream this is contained in PES_CF1; for the video stream in PES_CF2. Any audio or video packets which are not selected for decode (because their stream IDs do not match the programmed values) are discarded. The audio PES are output on path 1 (see [Figure 93](#)).

When used for decoding program streams or MPEG-1 system streams, the audio, video and system level data is automatically separated internally to the MPEG-2 PES and MPEG-1 system parser. Time stamp association is supported by the decoder.

During parsing, decode or display time stamps (DTSs or PTSs, selected by SDT (PES_CF1)) are stored in an internal FIFO. When the image corresponding to these time stamps is decoded (or, in the case of video, about to be decoded) the corresponding time stamp is made available and a flag or interrupt is given.

34.11.2 Functional modes

The parsers are enabled by setting bit SS (PES_CF2) for the video parser, and registers AUD_STREAMSEL/AUD_DECODESEL for the audio parser. Depending on the required mode, one or both of the parsers are required.

Four different modes can be configured with the two mode bits of register PES_CF2[7:6].

- Mode 0
Automatic configuration. The parser examines the incoming stream and self-configures for decode. The mode selected can be read back from PES_TM2[1].
- Mode 1
MPEG-1 system stream decode. Single data strobe input format. The audio elementary stream is extracted by the MPEG-2 PES parser and MPEG-1 system parser block and sent to the CD audio FIFO.
- Mode 2
MPEG-2 PES decode. Twin data strobe input format. The video PES stream and the audio PES stream are sent separately and respectively to MPEG-2 PES parser and MPEG-1 system parser block and audio CD FIFO. This is the most common way to enter data into the circuit.
- Mode 3
MPEG-2 whole PES audio/video packets. Single data strobe input format. This is used to decode MPEG-2 program streams. The audio PES are output on path 1 (see [Figure 93](#)) extracted by the MPEG-2 PES parser and MPEG-1 system parser block and send to the CD audio FIFO.

The video parser is reset by setting SS (PES_CF2) to 0.

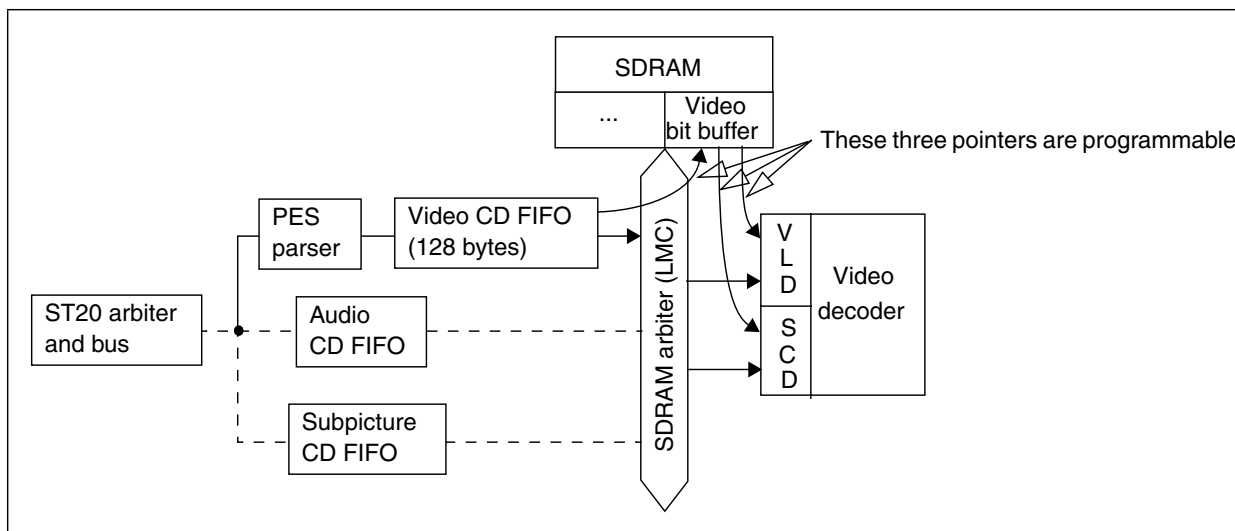
34.12 Enhanced trick modes

DVD trick modes, especially backward mode, require more video decoding flexibility than standard MPEG applications. The STi5516 supports the following trick mode features:

- programmable video CD (compressed data) FIFO pointer,
- programmable SCD (start code detector) pointer,
- programmable VLD (variable length decoder) pointer.

[Figure 94](#) illustrates the video decoder features which enhance DVD trick modes.

Figure 94: Enhanced trick mode support



34.12.1 Programming a video CD FIFO pointer

The video CD FIFO destination is programmed inside the SDRAM shared memory (up to 128 Mbits).

1. Set register bit TM (VID_LDP) to 1.
2. Flush the FIFO by writing 64 bytes (0xFF) to the CD FIFO.
3. Write a new 20-bit CD pointer value to the VID_TP_CD register (this is a three x 8-bit register).
4. Set bit CWL of register VID_CWL to 1. This starts the FIFO reset mechanism.
Status bit CWR of the VID_ITS register indicates that the CD FIFO is ready for transfer into SDRAM.

Register VID_TP_CALINIT can be used for overwrite protection.

34.12.2 Programming an SCD pointer

1. Set register bit TM (VID_LDP) to 1.
2. Write a new 20-bit SCD pointer value to the VID_TP_SCD register (this is a three x 8-bit register).
3. Set bit STL of register VID_STL to 1. This starts the FIFO reset mechanism.
Status bit SWR of the VID_ITS register indicates that the SCD FIFO is ready for transfer into SDRAM.

34.12.3 Programming a VLD pointer

1. Set register bit TM (VID_LDP) to 1.
2. Set register VID_TRF with the temporal reference, and set register VID_TP_VLD with a new read pointer.
3. Set bit TR_TML of register VID_TRF to 1.
Status bit TR_OK of register VID_ITS indicates that the VLR read pointer has been loaded into the memory controller, and that the VLD is ready to decode the selected picture.

35 MPEG video decoder registers

Addresses are provided as the *VideoBaseAddress* + offset.

The *VideoBaseAddress* is:

0x0000 0000.

A register summary is given in [Table 46: MPEG video decoder registers on page 70](#).

MPEGCONTROL

MPEG 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
Reserved													CSHOLDMSB	CSSETUPMSB	CDHOLDMSB	CDSETUPMSB	CSHOLDLSB	CSSETUPLSB	Reserved	CDHOLDLSB	CDSETUPLSB	DELAY							NOTSPCEN	NOTAUDEN	NOTVIDCSEN

Address: *VideoBaseAddress* + 0x5000

Type: Read/write

Reset: 0

Description: The recommended value for this register is 0x007 7100. This means compressed data (CD) transfers are one STBus clock cycle longer. If this is unacceptable then CD hold could be reduced by 1 cycle. In this case the register should be set to 0x007 3500. The third option is the maximum setting of 0x007 F700.

[31:19] **Reserved**

[18] **CSHOLDMSB**: Chip select hold time (MSB)

Extra cycle with chip select high (recommended value: 1)

[17] **CSSETUPMSB**: Chip select setup time (MSB)

Number of extra cycles with chip select low (recommended value: 1)

[16] **CDHOLDMSB**: CD strobes hold time (MSB)

Extra cycle with FIFO strobe high (recommended value: 1)

[15] **CDSETUPMSB**: CD strobes setup time (MSB)

Number of extra cycles with FIFO strobe low (recommended value: 0)

[14] **CSHOLDLSB**: Chip select hold time (LSB)

Extra cycle with chip select high (recommended value: 1)

[13:12] **CSSETUPLSB**: Chip select setup time (LSB)

Number of extra cycles with chips select low (recommended value: 11)

[11] **Reserved**

[10] **CDHOLDLSB**: CD strobes hold time (LSB)

Extra cycle with FIFO strobe high (recommended value: 0)

[9:8] **CDSETUPLSB**: CD strobes setup time (LSB)

Number of extra cycles with FIFO strobe low (recommended value: 01)

[7:3] **DELAY**

Interaccess delay: Delays the IC_R_REQ signal

[2] **NOTSPCEN**

Enable NOT_SPC_REQ signal (use not recommended)

[1] **NOTAUDEN**

Enable NOT_AUD_REQ signal (use not recommended)

[0] **NOTVIDCSEN**

Enable NOT_VIDCS_REQ signal (use not recommended)

Confidential

VID_ABG**Start of audio bit buffer**

	7	6	5	4	3	2	1	0
0x1C	ABG[15:8]							
0x1D	ABG[7:0]							

Address: *VideoBaseAddress* + 0x001C and 0x001D

Type: Read/write

Reset: 0

Description: The register holds the starting address of the audio bit stream buffer, defined in units of 2 Kbits. If the audio bit buffer starts at address 0, then this register does not need to be set up, since its reset state is 0. When all the registers corresponding to the settings of the bit buffer are done (VID_ABG, VID_ABS), an audio soft reset must be done for values to be taken in account. In other words, it must only be changed before the first compressed data of a new sequence is input, and never during the decoding of a sequence.

VID_ABL**Audio bit buffer level**

	7	6	5	4	3	2	1	0
0x1E	ABL[15:8]							
0x1F	ABL[7:0]							

Address: *VideoBaseAddress* + 0x001E and 0x001F

Type: Read only

Reset: 0

Description: This register holds the current level of occupation of the audio bit buffer, defined in units of 2 Kbits. It can be read at any time for the monitoring of the audio bit buffer level. When VID_ABL is greater than or equal to the value held in the VID_ABT register, the status bit ABF (audio bit buffer full) in VID_STA becomes set. When VID_ABL is zero, the status bit ABE (audio bit buffer empty) in VID_STA becomes set.

VID_ABS**Audio bit buffer stop**

	7	6	5	4	3	2	1	0
0x20	ABS[15:8]							
0x21	ABS[7:0]							

Address: *VideoBaseAddress* + 0x0020 and 0x0021

Type: Read/write

Reset: 0

Description: This register holds the address of the top of the audio bit buffer, defined in units of 2 Kbits. The space allocated to the audio bit buffer starts at the address defined by the VID_ABG register, or by default 0. The end address of the audio bit buffer is: $(128 \times \text{ABS}) + 127$. VID_ABS must only be changed before the first compressed data of a new sequence is input, and never during the decoding of a sequence.

VID_ABT**Audio bit buffer threshold**

	7	6	5	4	3	2	1	0
0x22	ABT [15:8]							
0x23	ABT [7:0]							

Address: *VideoBaseAddress* + 0x0022 and 0x0023

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

This register holds the level of occupancy of the audio bit buffer, in units of 2 Kbits.

When reached it causes the status bit ABF (VID_STA) to become set.

If the bit PBO (CFG_CCF) is set, then transfer of data to the audio bit buffer is

prevented if the bit buffer level is at or above the level defined in the VID_ABT register.

If VID_ABT is set to a value equal to the top of the bit buffer, then this automatic

mechanism ensures that overflow never occurs.

VID_BFC**Backward chroma pointer**

	7	6	5	4	3	2	1	0
0x5E	BFC [15:8]							
0x5F	BFC [7:0]							

Address: *VideoBaseAddress* + 0x005E and 0x005F

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register holds the start address of the chrominance buffer of the backward prediction frame picture, defined in units of 256 bytes.

Note: *If using 128-Mbit of SDRAM the value must be a multiple of 8, see [Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296](#).*

VID_BFP**Backward frame pointer**

	7	6	5	4	3	2	1	0
0x12	BFP [15:8]							
0x13	BFP [7:0]							

Address: *VideoBaseAddress* + 0x0012 and 0x0013

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register holds the start address of the luminance buffer of the backward prediction frame picture, defined in units of 256 bytes.

Note: *If using 128-Mbit of SDRAM the value must be a multiple of 8, see [Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296](#).*

VID_CDCOUNT**Bit buffer input counter**

	7	6	5	4	3	2	1	0
1st cycle	CDCOUNT [23:16]							
2nd cycle	CDCOUNT [15:8]							
3rd cycle	CDCOUNT [7:0]							

Address: *VideoBaseAddress* + 0x0067

Type: Serial read-only

Reset: 0

Description: These three registers are accessed serially. They hold the number of bytes input to the bit buffer.

The byte pointer is reset by a hardware reset, a global soft reset or a video soft reset.

VID_CTL**Decoding control**

7	6	5	4	3	2	1	0
ERU	ERS	Reserved			PRS	SRS	EDC

Address: *VideoBaseAddress* + 0x0002

Type: Read/write

Description: Edge triggered synchronization

[7] **ERU**

Enable pipeline reset on picture decode error. When this bit is set, a pipeline reset is automatically generated in case of Picture Decode Error (less than DFS macroblocks decoded).

[6] **ERS**

Enable pipeline reset on severe error. When this bit is set, a pipeline reset is automatically generated in case of Severe Error (more than DFS macroblocks decoded).

[5:3] **Reserved**

[2] **PRS**

Pipeline reset. In order to generate a pipeline reset, this bit must be kept set for a duration of at least four SDRAM clock cycles (40 ns for a 100 MHz SDRAM clock).

[1] **SRS**

Soft reset. In order to generate a soft reset, this bit must be kept set for a duration of at least 54 SDRAM clock cycles (540 ns for a 100 MHz primary clock).

[0] **EDC**

Enable decoding. This bit must be set to allow decoding.

VID_CWL**CD write launch**

7	6	5	4	3	2	1	0
Reserved							CWL

Address: *VideoBaseAddress* + 0x0031

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

[7:1] **Reserved**

[0] **CWL**

Launch compress data write FIFO reset procedure for trick modes
Automatically set to zero once reset procedure is completed

VID_DFS**Decoded frame size**

	7	6	5	4	3	2	1	0
1st cycle	Reserved		DFS[13:8]					
2nd cycle	DFS[7:0]							

Address: *VideoBaseAddress* + 0x0024

Type: Serial read/write

Reset: 0

Description: DSYNC synchronization
The circularity is reset by a software reset.

1st cycle [7:6] **Reserved**

1st cycle [5:0] **DFS**

and 2nd cycle [7:0] This register field 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.

VID_DFW**Decoded frame width**

	7	6	5	4	3	2	1	0
	DFW[7:0]							

Address: *VideoBaseAddress* + 0x0025

Type: Read/write

Reset: 0

Description: DSYNC synchronization
This register is set up with the width in macroblocks of the decoded picture. This is derived from the HORIZONTAL_SIZE value transmitted in the sequence header.

VID_FFC**Forward chroma frame pointer**

	7	6	5	4	3	2	1	0
0x5C	FFC[15:8]							
0x5D	FFC[7:0]							

Address: *VideoBaseAddress* + 0x005C and 0x005D

Type: Read/write

Reset: 0

Description: DSYNC synchronization
This register holds the start address of the forward prediction chroma frame picture buffer, defined in units of 256 bytes.

Note: If using 128-Mbit of SDRAM the value must be a multiple of 8, see [Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296](#).

VID_FFP**Forward luma frame pointer**

	7	6	5	4	3	2	1	0
0x10	FFP[15:8]							
0x11	FFP[7:0]							

Address: *VideoBaseAddress* + 0x0010 and 0x0011

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register holds the start address of the forward prediction luma frame picture buffer, defined in units of 256 bytes.

Note: If using 128-Mbit of SDRAM the value must be a multiple of 8, see [Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296](#).

VID_HDF**Header data FIFO**

	7	6	5	4	3	2	1	0
1st cycle	HDF[15:8]							
2nd cycle	HDF[7:0]							

Address: *VideoBaseAddress* + 0x0066

Type: Serial read-only

Reset: Undefined

Description: When the start code detector has found a start code, the header data FIFO must be read in order to identify the start code and if required to obtain the header data. The start code identification procedure is described in [Section 34.5.2: Start code detection on page 285](#).

Before reading the header FIFO, status bit HFE (VID_STA) should be checked to ensure that it is not empty. If bit HFF (VID_STA) is set then the header FIFO contains at least 66 bytes of data.

As the start code detection is managed with 16-bit words, two successive reads are needed to access the whole 16-bit word; you must always perform an even number of reads before start code search restart (by software or with DSYNC signal).

The byte pointer is reset by a hardware reset, a global soft reset or a video reset.

VID_HDS

Header search

	7	6	5	4	3	2	1	0
Read cycle	Reserved				SCM	Reserved		
Write cycle	Reserved					SOS	QMI	HDS

Address: *VideoBaseAddress* + 0x0069

Type: Read only bit SCM, write-only HDS, QMI and SOS.

Reset: 0

Description: No synchronization

This register controls the header search when it is written to, and returns the location of the start code when read.

Read cycle: [7:4] **Reserved**

Read cycle: [3] **SCM**

Start code on MSB. This bit indicates in which byte the start code is located. If this bit is set then VID_HDF[15:8] contains the start code; otherwise VID_HDF[7:0] contains the start code.

Read cycle: [2:0] **Reserved**

Write cycle: [7:3] **Reserved**

Write cycle: [2] **SOS**

Stop on first slice. This bit when set allows the start code detector to stop on the first slice start code of a picture (0x0000 0101). To allow mismatches, this bit must be used in conjunction with SCM (VID_HDS).

Write cycle: [1] **QMI**

This bit is used to control access to the inverse quantizer tables.

1: Select the intra table

0: Select the nonintra table.

For example, to write a new intra table, write QMI (VID_HDS) = 1 then write 64 weights to VID_QMW.

Write cycle: [0] **HDS**

Writing 1 to this bit starts a header search. Completion of the header search is indicated by the setting of bit SCH (VID_STA).

VID_ITM

Interrupt mask

	7	6	5	4	3	2	1	0
0x3C	CWR	ERR	SWR	TR_OK	ABF	SFF	AFF	ABE
0x60	PDE	SER	BMI	HFF	PNC	ERC	DEI	PII
0x61	PSD	VST	VSB	BBE	BBF	HFE	CFF	SCH

Address: *VideoBaseAddress* + 0x003C, 0x0060 and 0x0061

Type: Read/write

Reset: 0 (all interrupts disabled)

Description: No synchronization

Any bit set in this register enables the corresponding interrupt. An interrupt is generated whenever a bit in the VID_STA register changes from 0 to 1 and the corresponding mask bit is set.

VID_ITS**Interrupt status**

	7	6	5	4	3	2	1	0
0x3D	CWR	ERR	SWR	TR_OK	ABF	SFF	AFF	ABE
0x62	PDE	SER	BMI	HFF	PNC	ERC	DEI	PII
0x63	PSD	VST	VSF	BBE	BBF	HFE	CFF	SCH

Address: *VideoBaseAddress* + 0x003D, 0x0062 and 0x0063

Type: Read only

Reset: 0

Description: After the clocks have been enabled, the state changes to be the same as that of VID_STA.

When a bit in the VID_STA register changes from 0 to 1, the corresponding bit in the VID_ITS register is set, irrespective of the state of VID_ITM. If the corresponding bit of VID_ITM is set, the interrupt is asserted. Reading the most significant byte of VID_ITS clears it, leaving IRQ in its de-asserted (high) state.

See [Chapter 11: Interrupt system](#) for more information on interrupt handling.

VID_LDP**Load pointer**

7	6	5	4	3	2	1	0
Reserved				SBS	VSF	TM	LDP

Address: *VideoBaseAddress* + 0x003F

Type: Read/write

Reset: 0

Description: No synchronization.

[7:3] **Reserved**

[3] **SBS**

This bit stops an SCD read

1: SCD pointer address = SCD read limit pointer address

0: SCD pointer address = CD write limit pointer address

[2] **VSF**

This bit stops a VLD read

1: VLD pointer address = CD pointer address

0: VLD pointer address = CD write limit address

[1] **TM**

0: Trick modes are not selected

1: Trick modes are selected

[0] **LDP**

When this bit is set, the current start code detector pointer is stored in an internal register. When a PSC hit occurs, the current start code detector pointer has to be stored for further use by the VLD (if a B frame is to be processed on the fly). This bit has to be set then reset, before the PSD interrupt corresponding to the picture which has to be decoded on the fly (that is, during the SCH interrupt).

B frame on the fly decoding is not supported in the STi5516.

VID_PFH**Picture f-parameters horizontal**

7	6	5	4	3	2	1	0
BFH[3:0]				FFH[3:0]			

Address: *VideoBaseAddress* + 0x0004

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register contains parameters of the picture to be decoded. These parameters are extracted from the bit stream.

[7:4] **BFH**

MPEG-1

Set to BACKWARD_F_CODE of the picture header.

Set to FULL_PEL_BACKWARD_VECTOR of the picture header.

MPEG-2

Set to BACKWARD_HORIZONTAL_F_CODE of the picture coding extension.

[3:0] **FFH**

MPEG-1

Set to FORWARD_F_CODE of the picture header.

Set to FULL_PEL_FORWARD_VECTOR of the picture header.

MPEG-2

Set to FORWARD_HORIZONTAL_F_CODE of the picture coding extension.

VID_PFV**Picture F-parameters vertical**

7	6	5	4	3	2	1	0
BFV[3:0]				FFV[3:0]			

Address: *VideoBaseAddress* + 0x0005

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register contains parameters of the picture to be decoded. These parameters are extracted from the bit stream.

In MPEG-1 mode (that is, when MP2 (VID_PPR2) is reset), VID_PFV is not used.

[7:4] **BFV**

The BACKWARD_VERTICAL_F_CODE of the picture coding extension.

[3:0] **FFV**

The FORWARD_VERTICAL_F_CODE of the picture coding extension.

Confidential

VID_PPR1**Picture parameters 1**

7	6	5	4	3	2	1	0
Reserved	OTF	PCT[1:0]		DCP[1:0]		PST[1:0]	

Address: *VideoBaseAddress* + 0x0006

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register contains parameters of the picture to be decoded. These parameters are extracted from the bit stream.

In MPEG-1 mode (that is, when MP2 (VID_PPR2) is reset), only PCT has to be set; the other bits must be reset.

[7] **Reserved**

[6] **OTF**

When set, this bit directs the decoded data in the block-to-row memory to be displayed directly. The picture data is not reconstructed in memory.

[5:4] **PCT**

Set to the two least significant bits of PICTURE_CODING_TYPE in the picture header.

[3:2] **DCP**

Set equal to INTRA_DC_PRECISION of the picture coding extension. The value 11, defining a precision of 3 bits, is not allowed.

[1:0] **PST**

Set to the PICTURE_STRUCTURE bits of the MPEG-2 picture coding extension.

00: Frame picture. This value is illegal in the MPEG-2 variable.

01: Top field.

10: Bottom field.

11: Frame picture.

VID_PPR2

Picture parameters 2

7	6	5	4	3	2	1	0
Reserved	MP2	TFF	FRM	CMV	QST	IVF	AZZ

Address: *VideoBaseAddress* + 0x0007

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register contains parameters of the picture to be decoded. These parameters are extracted from the bit stream.

In MPEG-1 mode, all bits must be reset to 0.

[7] **Reserved**

[6] **MP2**: MPEG-2 mode

When this bit is set, the STi5516 expects an MPEG-2 video bit stream. If it is reset, then an MPEG-1 bit stream is expected.

[5] **TFF**: Set equal to the TOP_FIELD_FIRST bit of the MPEG-2 picture coding extension

[4] **FRM**: Set equal to the FRAME_PRED_FRAME_DCT bit of the picture coding extension

[3] **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

[2] **QST**: Set equal to the Q_SCALE_TYPE bit of the picture coding extension

[1] **IVF**: Set equal to the INTRA_VLC_FORMAT bit of the picture coding extension

[0] **AZZ**: Set equal to the ALTERNATE_SCAN bit of the picture coding extension

VID_PTH

Panic threshold

	7	6	5	4	3	2	1	0
0x2E	Reserved		FPAN	PEN	NFW	PTH[10:8]		
0x2F	PTH[7:0]							

Address: *VideoBaseAddress* + 0x002E and 0x002F

Type: Read/write

Reset: 0

Description: VSYNC synchronization

The panic threshold defines a block-to-row DRAM fullness level. In the block-to-row converter, there are two counters indicating the number of luma and chroma words stored. If the luma counter pointer is less than PTH or the chroma counter pointer is less than PTH divided by 2, a PANIC flag is set.

The panic threshold is programmed in units of DRAM cells.

0x2E: [7:6] **Reserved**

0x2E: [5] **FPAN**: Force panic mode

When set, this bit forces panic mode while decoding. For test purposes only.

0x2E: [4] **PEN**: Panic mode enable

When set, this bit allows the decoding to set the panic flag and enter the panic mode.

0x2E: [3] **NFW**: Near forward

This bit allows the user to select which prediction direction is retained for bidirectional macroblocks in panic mode. Forward if set, backward otherwise.

0x2E: [2:0] **PTH**: Panic mode threshold.
and 0x2F [7:0]

Confidential

VID_QMW**Quantization matrix data**

7	6	5	4	3	2	1	0
QMW[7:0]							

Address: *VideoBaseAddress* + 0x0076

Type: Write only

Reset: Undefined

Description: No synchronization

This address is used to load the quantization coefficients in the order in which they appear in the bit stream, that is, zigzag order. The bit QMI (VID_HDS) defines which matrix (intra or inter) is written.

For example, to write a new intra table, write QMI = 1, and then write 64 waits to VID_QMW.

VID_LCK**Memory configuration register lock**

7	6	5	4	3	2	1	0
Reserved							LCK

Address: *VideoBaseAddress* + 0x007B

Type: Read/write

Reset: 0

Description:

[7:1] **Reserved**

[0] **LCK**

When set, this bit locks the current values of registers CFG_DRC, CFG_MCF and CFG_CCF.

VID_RFC**Reconstructed chroma frame pointer**

	7	6	5	4	3	2	1	0
0x5A	RFC[15:8]							
0x5B	RFC[7:0]							

Address: *VideoBaseAddress* + 0x005A and 0x005B

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register holds the start address of the reconstructed (decoded) chroma frame picture buffer, defined in units of 256 bytes.

Note: *If using 128-Mbit of SDRAM the value must be a multiple of 8, see [Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296](#).*

VID_RFP**Reconstructed frame pointer**

	7	6	5	4	3	2	1	0
0x0E	RFP[15:8]							
0x0F	RFP[7:0]							

Address: *VideoBaseAddress* + 0x000E and 0x000F

Type: Read/write

Reset: 0

Description: DSYNC synchronization

This register holds the start address of the reconstructed (decoded) luma frame picture buffer, defined in units of 256 bytes.

Note: If using 128-Mbit of SDRAM the value must be a multiple of 8, see [Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296](#).

VID_SCDCOUNT**Bit buffer output counter**

	7	6	5	4	3	2	1	0
1st cycle	SCDCOUNT [23:16]							
2nd cycle	SCDCOUNT [15:8]							
3rd cycle	SCDCOUNT [7:0]							

Address: *VideoBaseAddress* + 0x0068

Type: Serial read-only

Reset: 0

Description: These three registers are accessed serially. This register holds the number of 16-bit words output from the bit buffer into the start code detector.
The byte pointer is reset by a hardware reset, a global soft reset or a video reset.

VID_TP_SCDLIMIT**SCD read limit address**

	7	6	5	4	3	2	1	0
0xBA	Reserved						SRL_ADDRESS[17:16]	
0xBB	SRL_ADDRESS[15:8]							
0xBC	SRL_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00BA, 0x00BB, 0x00BC

Type: Read/write

Reset: 0

Description: This register holds the SCD read limit pointer, only used in trick mode if VID_LDP[3] is set, to stop SCD read access when the SCD pointer reaches it.

Confidential

VID_SPB**Subpicture buffer begin**

	7	6	5	4	3	2	1	0
0x50	Reserved			SPB[12:8]				
0x51	SPB[7:0]							

Address: *VideoBaseAddress* + 0x0050 and 0x0051

Type: Read/write

Reset: 0

Description: No synchronization

This register holds the start address of the subpicture circular buffer and is programmed in units of 2 Kbytes.

VID_SPE**Subpicture buffer end**

	7	6	5	4	3	2	1	0
0x52	Reserved			SPE[12:8]				
0x53	SPE[7:0]							

Address: *VideoBaseAddress* + 0x0052 and 0x0053

Type: Read/write

Reset: 0

Description: No synchronization

This register holds the stop address of the subpicture circular buffer and is programmed in units of 2 Kbytes.

VID_SPREAD**Subpicture read pointer**

	7	6	5	4	3	2	1	0
1st cycle	Reserved			SPR[20:16]				
2nd cycle	SPR[15:8]							
3rd cycle	SPR[7:0]							

Address: *VideoBaseAddress* + 0x004E

Type: Serial read/write

Reset: 0

Description: VSYNC synchronization

These three registers are accessed serially. The byte pointer is reset by a hardware reset. This is the absolute address in the memory. This register is used when the software needs to set the read address of the subpicture decoder and is programmed in units of 64-bit words. This value is taken into account after a VSYNC.

Confidential

VID_SPWRITE**Subpicture write pointer**

	7	6	5	4	3	2	1	0
1st cycle	Reserved			SPW[20:16]				
2nd cycle	SPW[15:8]							
3rd cycle	SPW[7:0]							

Address: *VideoBaseAddress* + 0x004F

Type: Serial read/write

Reset: 0

Description: No synchronization

These three registers are accessed serially. The byte pointer is reset by a hardware reset. This is the absolute address in the memory. This register is used when the software needs to set the write address of the subpicture decoder and is programmed in units of 64-bit words.

It is recommended to stop loading subpicture data to the subpicture decoder FIFO before changing the value of this register.

VID_SRA**Audio soft reset**

	7	6	5	4	3	2	1	0
	Reserved							SRA

Address: *VideoBaseAddress* + 0x0035

Type: Read/write

Reset: 0

Description: No synchronization

In order to generate an audio soft reset (bit buffer), this bit must be kept set for a duration of at least 54 SDRAM clock cycles (540 ns for a 100 MHz primary clock).

VID_SRV**Video soft reset**

	7	6	5	4	3	2	1	0
	Reserved							SRV

Address: *VideoBaseAddress* + 0x0039

Type: Read/write

Reset: 0

Description: No synchronization

In order to generate a video soft reset, this bit must be kept set for a duration of at least 54 SDRAM clock cycles (540 ns for a 100 MHz primary clock).

Confidential

VID_STA

Status

	7	6	5	4	3	2	1	0
0x3B	CWR	ERR	SWR	TR_OK	ABF	SFF	AFF	ABE
0x64	PDE	SER	BMI	HFF	PNC	ERC	DEI	PII
0x65	PSD	VST	VSF	BBE	BBF	HFE	BFF	SCH

Address: *VideoBaseAddress* + 0x003B, 0x0064 and 0x0065

Type: Read only

Reset: See table.

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.
The status vector is sampled internally at the start of the read cycle accessing the most significant byte of VID_STA (address 0x3B). VST, VSB and PSD are pulses and are unlikely ever to be read as 1.
The status bits are described in the table below. Reset shows the state after clocks have been enabled.

0x65: [7]**PSD**: Pipeline starting to decode

This bit is set for a short period at the instant the pipeline starts decoding a picture. Reset: 0

0x65: [6]**VST**: VSYNC top

This bit is set for a short time at the beginning of the top field, corresponding to the falling edge of the B/T signal. Reset: 0

0x65: [5]**VSB**: VSYNC bottom

This bit is set for a short time at the beginning of the bottom field, corresponding to the rising edge of the B/T signal. Reset: 0

0x65: [4]**BBE**: Video bit buffer empty

This bit is set when the bit buffer contains no data. Reset: 1

0x65: [3]**BBF**: Video bit buffer full

This bit is set when the bit buffer level (= VID_VBL) is greater than or equal to the value loaded into the VID_VBT register. Reset: 1

0x65: [2]**HFE**: Header FIFO empty

This bit is set when the header FIFO is empty. Reset: 1

0x65: [1]**BFF**: Video compressed data (bit stream) FIFO full

This bit is set when the video CD FIFO is full. This bit is equivalent to the signal NOT_CDREQ. Reset: 0

0x65: [0]**SCH**: Start code hit

This bit is set whenever the first 16-bit word available in the header FIFO contains one of the start codes recognized. While data is being read from the header FIFO, this bit can be tested to determine whether the next word contains a start code. Reset: 0

0x64: [7]**PDE**: Picture decoding error or underflow error

This bit is set when less than the programmed number of macroblocks (defined by DFS) have been decoded, either due to a data or a programming error. Decoding is halted automatically when this error condition is detected. This bit is reset by all three types of reset. Reset: 0

0x64: [6]**SER**: Severe error or overflow error

This bit is set when more than the programmed number of macroblocks (defined by DFS) have been decoded, either due to a data or a programming error. Decoding is halted automatically when this error condition is detected. This bit is reset by all three types of reset. Reset: 0

0x64: [5]**BMI**: Block move idle

This bit is set when a block move operation has terminated. It is automatically reset at the start of a block move. Reset: 1

0x64: [4]**HFF**: Header FIFO full

This bit is set when the header FIFO contains at least 66 bytes. Reset: 0

0x64: [3]**PNC**: Panic

This bit is set when decoding is in late compare to display. Reset: 0

0x64: [2]**ERC**: Error concealment

This bit is set when an error is detected in the bit stream and the mechanism of error concealment is active.
Reset: 0

0x64: [1]**DEI**: Decoder idle

This bit is set when the STi5516 is not in the course of decoding a picture, that is, when the pipeline is inactive. It becomes low when the decoding of a picture starts and high when picture decoding is complete.
Reset: 1

0x64: [0]**PII**: Pipeline idle

This bit is set when the STi5516 is ready to decode a picture, that is, when the pipeline is inactive and it has found the next picture start code. It becomes low when the decoding of a picture starts and high when picture decoding is complete and the next picture start code has been found by the pipeline. Reset: 1

0x3B: [7]**CWR**: Start code detector write ready

Reset: 0

0x3B: [6]**ERR**: Inconsistency error in PES parser

Reset: 0

0x3B: [5]**SWR**: CD write ready

0x3B: [4]**TR_OK**: VLD pointer loaded and VLD ready to decode picture

0x3B: [3]**ABF**: Audio bit buffer full. Reset: 0

0x3B: [2]**SFF**: Subpicture compressed data (bit stream) FIFO full

This bit is set when the subpicture CD FIFO is full. Reset: 0

0x3B: [1]**AFF**: Audio compressed data (bit stream) FIFO full

This bit is set when the audio CD FIFO is full. Reset: 0

0x3B: [0]**ABE**: Audio bit buffer empty

This bit is set when the audio bit buffer contains no data. Reset: 0

VID_STL

SCD trick mode launch

7	6	5	4	3	2	1	0
Reserved					DSS	INR	STL

Address: *VideoBaseAddress* + 0x0030

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

[7:3] **Reserved**

[2] **DSS**

Disable search of sequence start code after soft reset

[1] **INR**

Inhibit redecode mode.

1: Start code search is never launched automatically (neither by the block itself nor by the standard process)

0: Automatically starts header search after DSYNC

[0] **STL**

Launch start code detector FIFO reset procedure for trick modes

Automatically set to zero once reset procedure is completed

VID_TIS

Task instruction

7	6	5	4	3	2	1	0
Reserved		SKP[1:0]		OVW	FIS	RPT	EXE

Address: *VideoBaseAddress* + 0x0003

Type: Write only

Reset: 0

Description: VSYNC synchronization
This register contains six bits of the decoding task instruction.

[7:6] **Reserved**

[5:4] **SKP**

These bits are part of the instruction register, and are thus synchronized with VSYNC. They define the skipping of one or two pictures. If skipping is required, then these bits must be set up as part of the instruction for the picture which is decoded immediately after the skipped pictures.

00: No skip (default)

01: Skip one picture, decode next

10: Skip two pictures, decode next

11: Skip one picture then stop

[3] **OVW**

This bit must be set when the displayed picture and the reconstructed picture share the same buffer (that is, VID_DFP = VID_RFP). It enables the overwrite mode which ensures that the reconstructed picture does not overwrite data which has not yet been displayed.

[2] **FIS**

Force instruction. If this bit is set, the task described by this register is launched immediately. This bit is reset after its action. Its effect is the same as the effect of a VSYNC.

[1] **RPT**

When this bit is set, the task duration is two VSYNC periods. When the frame display rate is equal to the picture decoding rate, RPT is generally always high. In the case of a task launched by FIS, RPT is not taken in account.

[0] **EXE**

When this bit is not set, no decoding or skipping task is executed for one or two VSYNC periods, depending on the state of RPT. If set, the next task (decoding or skipping) is executed. EXE is internally cleared when the task starts its execution, that is, setting this bit activates only one execution.

VID_TP_CD

CD pointer load address

	7	6	5	4	3	2	1	0
0xCA	Reserved						CD_POINTER_ADDRESS[17:16]	
0xCB	CD_POINTER_ADDRESS[15:8]							
0xCC	CD_POINTER_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00CA, 0x00CB, 0x00CC

Type: Read/write

Description: This register holds the CD pointer address during trick modes, defined in units of 512 bits.

VID_TP_CDLIMIT

CD write limit address

	7	6	5	4	3	2	1	0
0xE0	Reserved						CD_WRITE_LIMIT_ADDRESS[17:16]	
0xE1	CD_WRITE_LIMIT_ADDRESS[15:8]							
0xE2	CD_WRITE_LIMIT_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00E2, 0x00E1, 0x00E0

Type: Read/write

Description: Contains the CD write limit value; the CD cannot write beyond this limit. The address is not latched and is a multiple of the CAS cycle length. In trick modes, a special bit buffer management system is applied where the CD write flow is as follows:

- CD pointer address = CD write limit = request CD disable.

For VLD read flow:

- VLD pointer address = CD write limit => VLD request disable if FMUFLAGS/VSB is reset
- VLD pointer address = CD pointer address => VLD request disable if FMUFLAGS/VSB is set

For SCD read flow:

- VSCD pointer address = CD write limit => SD request disable

Note: *CD write limit must be in the same format as the address used during comparisons.*

VID_TP_CD_RD

CD pointer address

	7	6	5	4	3	2	1	0
0x83	Reserved			CD_POINTER_ADDRESS[20:16]				
0x84	CD_POINTER_ADDRESS[15:8]							
0x85	CD_POINTER_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x0085, 0x0084, 0x0083

Type: Read only

Description: Holds the current CD pointer address, defined in units of 64 bits.

VID_TP_SCD

SCD pointer load address

	7	6	5	4	3	2	1	0
0xC0	Reserved						SCD_POINTER_ADDRESS[17:16]	
0xC1	SCD_POINTER_ADDRESS[15:8]							
0xC2	SCD_POINTER_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00C0, 0x00C1, 0x00C2

Type: Read/write

Description: Holds the address to load into the SCD pointer during trick modes. The address is defined in units of 512 bits.

VID_TP_SCD_CURRENT SCD pointer current address

	7	6	5	4	3	2	1	0
0xCD	Reserved			SCD_POINTER_ADDRESS[20:16]				
0xCE	SCD_POINTER_ADDRESS[15:8]							
0xCF	SCD_POINTER_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00CD, 0x00CE, 0x00CF

Type: Read only

Description: Holds the current SCD address, defined in units of 64 bits.

VID_TP_SCD_RD SCD pointer VLD load address

	7	6	5	4	3	2	1	0
0x80	Reserved							SCD_POINTER_ADDRESS_FOR_VLD[16]
0x81	SCD_POINTER_ADDRESS_FOR_VLD[15:8]							
0x82	SCD_POINTER_ADDRESS_FOR_VLD[7:0]							

Address: *VideoBaseAddress* + 0x0080, 0x0081, 0x0082

Type: Read only

Description: Holds the address to be loaded into the VLD pointer during trick modes. When the start code detector finds a start code, this register holds the bit buffer address, defined in units of 1 Kbit.

VID_TP_VLD VLD pointer load address

	7	6	5	4	3	2	1	0
0xD3	Reserved							VLD_POINTER_ADDRESS[16]
0xD4	VLD_POINTER_ADDRESS[15:8]							
0xD5	VLD_POINTER_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00D4, 0x00D5

Type: Read/write

Reset: 0

Description: This register holds the address to load to the VLD address pointer address during trick modes. It is defined in units of 1 Kbit.

VID_TP_VLD_RD VLD pointer current address

	7	6	5	4	3	2	1	0
0xC3	Reserved			VLD_POINTER_ADDRESS[20:16]				
0xC4	VLD_POINTER_ADDRESS[15:8]							
0xC5	VLD_POINTER_ADDRESS[7:0]							

Address: *VideoBaseAddress* + 0x00C3, 0x00C4, 0x00C5

Type: Read only

Description: Holds the current VLD pointer address, defined in units of 64 bits.

VID_TRF**Temporal reference**

	7	6	5	4	3	2	1	0
0x56	Reserved			TML	DC2	DTR	TRF[9:8]	
0x57	TRF[7:0]							

Address: *VideoBaseAddress* + 0x0056 and 0x0057

Type: Read/write

Reset: 0

Description: DSYNC synchronization

Description: This register contains extensions of the decoding task instruction. It is used only when decoding a B frame on the fly.

0x56: [7:5] **Reserved**

0x56: [4] **TML**: TRICK_MODE_LAUNCH

This signal is autoreinitialized, that is it returns automatically to zero (it generates only a pulse).

0x56: [3] **DC2**: Redecode same B frame twice

When this bit is set it signals the VLD that the next picture has to be decoded twice. This bit is used when decoding a B frame on the fly. It has to be set by the user before the first PSD interrupt corresponding to a B frame and reset on the according PSD interrupt.

0x56: [2] **DTR**: Disable temporal reference comparison

This bit is used only when decoding a B frame on the fly. On every B frame redecode the VLD uses the temporal reference field TRF[9:0] to resynchronize on the previously decoded picture. When this bit is set this comparison mechanism is disabled.

0x56: [1:0] **TRF**: Temporal reference

0x57: [7:0] This field holds the temporal reference of the current decoded B frame. On every B frame redecode the VLD uses the temporal reference field TRF to resynchronize on the previously decoded picture. TRF is used only when bit DTR is reset. B frame on the fly decoding is not supported in the STi5516.

VID_VBG**Start of video bit buffer**

	7	6	5	4	3	2	1	0
0x14	VBG[15:8]							
0x15	VBG[7:0]							

Address: *VideoBaseAddress* + 0x0014 and 0x0015

Type: Read/write

Reset: 0

Description: No synchronization

The register holds the starting address of the video bit buffer, defined in units of 2 Kbits. If the video bit buffer starts at address 0, then this register does not need to be set up, since its reset state is 0. When all the registers corresponding to the settings of the bit buffer are done (VID_VBG, VID_VBS), a video soft reset must be done for values to be taken in account. In other words it must only be changed before the first compressed data of a new sequence is input, and never during the decoding of a sequence.

VID_VBL Video bit buffer level

	7	6	5	4	3	2	1	0
0x16	VBL[15:8]							
0x17	VBL[7:0]							

Address: *VideoBaseAddress* + 0x0016 and 0x0017

Type: Read only

Reset: 0

Description: This register holds the current level of occupation of the video bit buffer, defined in units of 2 Kbits. It can be read at any time for the monitoring of the video bit buffer level. When VID_VBL is greater than or equal to the value held in the VID_VBT register, the status bit BBF (video bit buffer full) in VID_STA becomes set. When VID_VBL is zero, the status bit BBE (video bit buffer empty) in VID_STA becomes set.

VID_VBS Video bit buffer stop

	7	6	5	4	3	2	1	0
0x18	VBS[15:8]							
0x19	VBS[7:0]							

Address: *VideoBaseAddress* + 0x0018 and 0x0019

Type: Read/write

Reset: 0

Description: No synchronization

This register holds the address of the top of the video bit buffer, defined in units of 2 Kbits. The space allocated to the video bit buffer starts at the address defined by the VID_VBG register, or, by default, 0. The end address of the video bit buffer is (in units of 64 bits words):

$$(32 \times \text{VID\_VBS}) + 31$$

VID_VBS must only be changed before the first compressed data of a new sequence is input, and never during the decoding of a sequence.

VID_VBT Video bit buffer threshold

	7	6	5	4	3	2	1	0
0x1A	VBT[15:8]							
0x1B	VBT[7:0]							

Address: *VideoBaseAddress* + 0x001A and 0x001B

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

This register holds the level of occupancy of the video bit buffer, in units of 2 Kbits, which when reached causes the status bit BBF (VID_STA) to become set, that is, if $\text{VID\_VBL} \geq \text{VID\_VBT}$, then BBF is set.

If the bit PBO (CFG_CCF) is set, then transfer of data from the video CD FIFO to the bit buffer is prevented if the bit buffer level is at or above the level defined in the VID_VBT register. If VID_VBT is set to a value equal to the top of the bit buffer, then this automatic mechanism ensures that overflow never occurs.

35.1 Configuration and control (CFG) register information

CFG_CCF

Chip configuration

7	6	5	4	3	2	1	0
Reserved	EOU	PBO	Reserved			EDI	EVI

Address: *VideoBaseAddress* + 0x0001

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

[7] **Reserved**

[6] **EOU**

Enable overflow/underflow errors.

When this bit is reset overflow and underflow errors are not treated internally. This bit must be set for normal operation.

[5] **PBO**

Prevent bit buffer overflow.

When this bit is set, bit buffer overflow (and thus the loss of data) is prevented by disabling the transfer of data from the compressed data FIFO to the bit buffer whenever the bit buffer level reaches the threshold defined in the VID_VBT or VID_ABT register.

[4:2] **Reserved**

[1] **EDI**

Enable SDRAM interface.

When this bit is reset the SDRAM interface is put into its high impedance state. This bit must be set for normal operation and for reduced power mode. It is reset in low power mode.

[0] **EVI**

Enable video interface.

When this bit is reset the video interface is put into its high impedance state and the internal PIXCLK disabled. This bit must be set for normal operation and for reduced power mode (if the display interface is used). It is reset in low power mode.

CFG_CDR

Compressed data input

7	6	5	4	3	2	1	0
CFG_CDR[7:0]							

Address: *VideoBaseAddress* + 0x0044

Type: Write only

Reset: Undefined

Description: No synchronization

This register is a compressed data input register. It is used to input compressed data using a different path from the usual DMA path. Depending on the value of CDR[1:0] (CFG_GCF) the data stored in this register enters either the audio, video or subpicture compressed data FIFO.

CFG_DRC

DRAM configuration

7	6	5	4	3	2	1	0
MY_DEVICE		MRS	Reserved	P[1:0]		ERQ	SDR

Address: *VideoBaseAddress* + 0x0038

Type: Read/write

Reset: 0

Description: No synchronization

[7:6] **MY_DEVICE**

Memory device

00: 16 Mbits SDRAM (one or two units)

01: 128 Mbits SDRAM

10: 64 Mbits SDRAM

11: Reserved

[5] **MRS**

Mode register set

When this bit is set the special power-on reset sequence and mode register programming is carried out.

When this bit is reset the procedure is inhibited. Mode used is: burst length = 4 and CAS latency = 3.

[4] **Reserved**

[3:2] **P[1:0]**

CLK3 phase to MEMCLK

The following procedure has to be executed to make sure the SDRAM interface is properly initialized:

P[1:0] = 0, write four 32-bit words to SDRAM, read back the four words from SDRAM, if they do not read back correctly, increment the value of P[1:0] until the read is correct.

[1] **ERQ**

Enable processes requests

This bit when reset disables all processes requests to the local memory controller. This bit must be set for normal operation.

[0] **SDR**

Synchronous DRAM mode

This bit must always be written as 1.

CFG_GCF

General configuration

7	6	5	4	3	2	1	0
PXD	Reserved		CDR[1:0]		Reserved		

Address: *VideoBaseAddress* + 0x003A

Type: Read/write

Reset: 0

Description: No synchronization

[7] **PXD**

Add one PIXCLK delay

When this bit is set the active video is delayed by one PIXCLK cycle with respect to NOT_HSYNC. This is to allow its horizontal position to be defined more precisely than is possible with VID_XDO and VID_XDS, which have a resolution of two PIXCLK cycles. Changing the value of PXD also has the effect of inverting the phasing of the Y/C output samples with respect to NOT_HSYNC.

[6:5] **Reserved**

[4:3] **CDR**

Compressed data register

00: Automatic strobe generation when writing to CD FIFOs

01: Audio

10: Subpicture

11: Video

When one internal strobe is chosen, it has to be driven by writing to register CFG_CDR.

[2:0] **Reserved**

CFG_MCF**Memory refresh interval**

7	6	5	4	3	2	1	0
Reserved	RFI[6:0]						

Address: *VideoBaseAddress* + 0x0000

Type: Read/write

Reset: 0

Description: No synchronization

This register defines the SDRAM refresh interval in units of 32 SDRAM clock periods.

For example if 2048 rows must be refreshed every 32 ms, with an SDRAM clock of 108 MHz, the following value must be stored in RFI[6:0]:

$$(32 \times 10^{-3} / 2048) \times (108 \times 10^6 / 32) = 52$$

If the register is set to 0 then refresh is disabled.

35.2 PES parser (PES) register Information**PES_CF1****PES audio decoding control**

7	6	5	4	3	2	1	0
SDT	FAD	IAI	AUD_ID[4:0]				

Address: *VideoBaseAddress* + 0x0040

Type: Read/write

Reset: 0

Description: No synchronization

This register controls the audio stream decoding and some miscellaneous parser functions.

[7] **SDT**
Store DTS not PTS

[6] **FAD**
1: Audio compressed data taken directly from audio DMA and not from MPEG2 PES parser/MPEG1 system parser

[5] **IAI**
Ignore audio stream id

[4:0] **AUD_ID[4:0]**
Audio stream id

Confidential

PES_CF2**PES video parser control**

7	6	5	4	3	2	1	0
MOD[1:0]		SS	IVI	VID_ID[3:0]			

Address: *VideoBaseAddress* + 0x0041

Type: Read/write

Reset: 0

Description: No synchronization
This register controls the video stream parser.

[7:6] **MOD[1:0]**

00: Automatic configuration. The parser configures itself to decode the incoming stream.

01: MPEG-1 system stream with one data strobe input format

10: MPEG-2 PES separate data strobes input format (standard mode).

11: MPEG-2 whole PES video and whole PES audio one after the other with single data strobe input format.

[5] **SS**

This bit, when set, indicates that the stream sent to the parser is a system stream. To decode a pure video or audio stream this bit should be set to zero.

[4] **IVI**

Ignore video stream identifier.

[3:0] **VID_ID[3:0]**

Video stream identifier.

PES_TM1**DSM trick mode**

7	6	5	4	3	2	1	0
PES_TM1[7:0]							

Address: *VideoBaseAddress* + 0x0042

Type: Read only

Reset: 0

Description: No synchronization
This register stores the DSM trick mode bits. This register may be used only if the ES rate flag of the bit stream is set to zero.

PES_TM2**PES parser status**

7	6	5	4	3	2	1	0
Reserved						M2	DSA

Address: *VideoBaseAddress* + 0x0043

Type: Read only

Reset: 2

Description: No synchronization

[7:2] **Reserved**

[1] **M2**

MPEG-2 not MPEG-1. This bit indicates, in automatic mode, if the current stream being decoded is an MPEG-2 or an MPEG-1 stream.

[0] **DSA**

DSM association flag. This bit, when set indicates that the picture header present in the start code detector is associated to DSM values present in the PES_TM1 register.

Confidential

PES_TS		PES time stamps							
		7	6	5	4	3	2	1	0
0x49	PES_TS [7:0]								
0x4A	PES_TS [15:8]								
0x4B	PES_TS [23:16]								
0x4C	PES_TS [31:24]								
0x4D	Reserved						TSA		PES_TS [32]

Address:

Type:

Reset:

Description:

VideoBaseAddress + 0x0049 to 0x004D

Read only

0

No synchronization

0x4D: [7:2]

0x4D: [1]

0x4D: [0]

0x49 to 0x4C: [7:0]

Reserved

TSA

PES_TS[32:0]

Time stamps

Time stamp association. When this bit is set it indicates that the picture to be decoded next (from which the header is available in the start code detector) has an associated time stamp available in PES_TS.

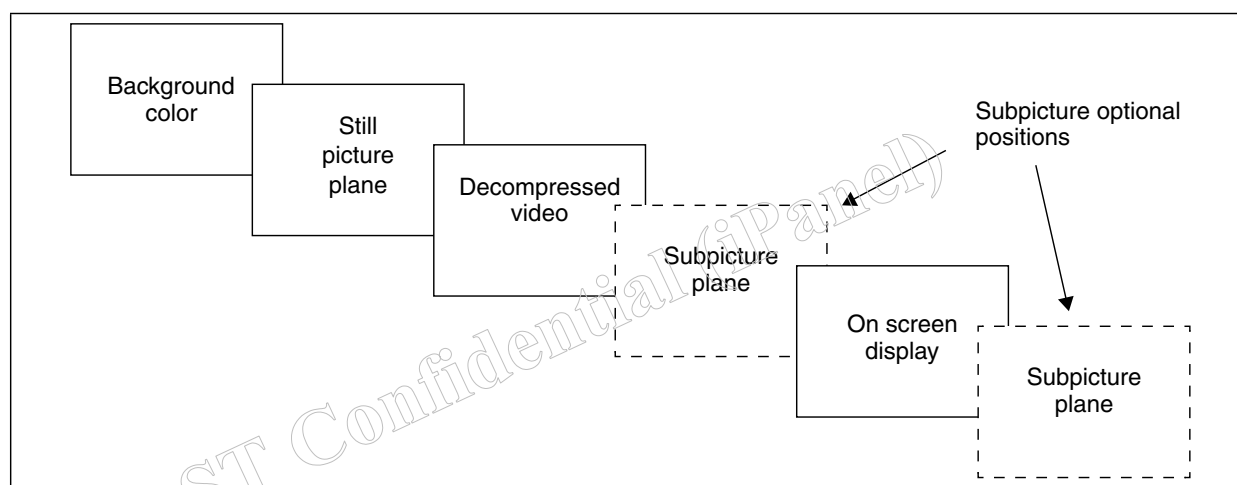
Confidential

36 Subpicture decoder

36.1 Overview

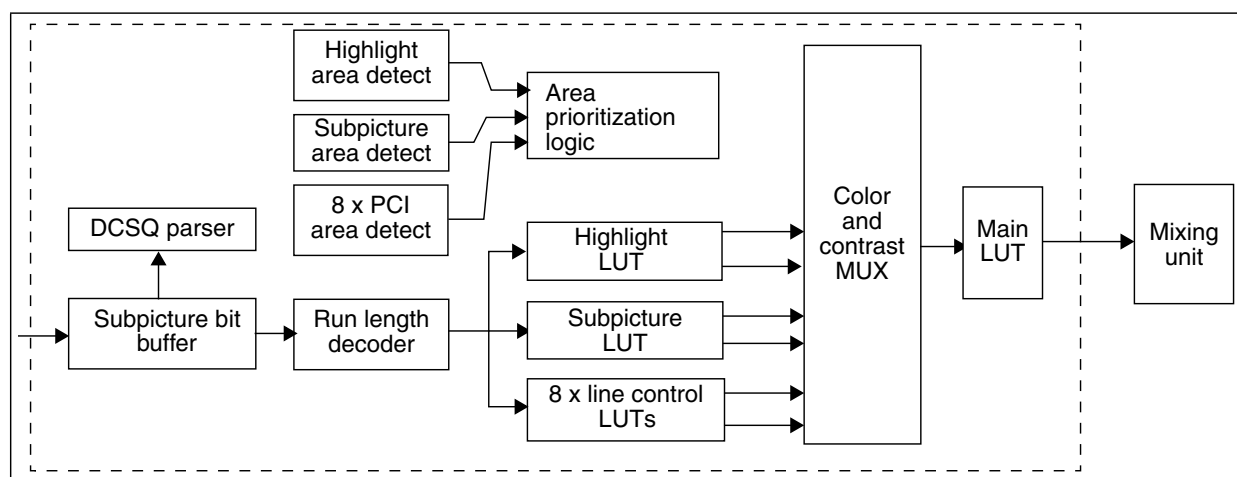
A hardware subpicture decoder is integrated in the STi5516. The subpicture bit buffer that contains subpicture units (SPU) is integrated in SDRAM external memory and has a programmable size. Its position and size can be randomly chosen in multiples of 2 Kbytes. The subpicture bit buffer is set up at power on reset. During player operation, its size and location are constant. Compressed data is input into the bit buffer using a DMA or by a CPU write. Once control is given to the subpicture decoder, it is autonomous until stopped by software control. The subpicture decoder can decode complete subpicture units consisting of a subpicture unit header, compressed pixel data and the display control sequence table without any interaction from the CPU.

Figure 95: Display planes



The subpicture decoder can also be used as a hardware cursor unit. The priority of the subpicture is first raised by programming a register so it is in front of all the other display planes. A cursor can be defined using an optionally compressed (run length encoded) bit map stored in external SDRAM. The bit map can be any size up to a full screen. Per-pixel alpha blending factors can be defined for each cursor to provide anti-aliasing with the background. The cursor is then moved around using register writes into X and Y coordinate registers. [Figure 96](#) shows the architecture of the subpicture decoder.

Figure 96: Subpicture decoder architecture

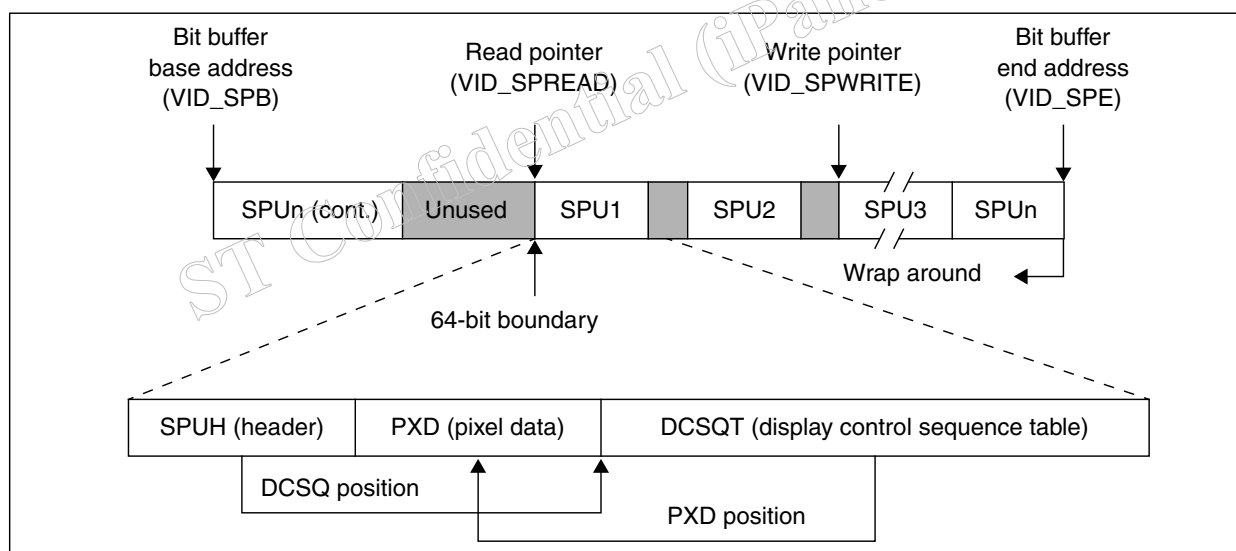


36.2 Buffer management and pointers

There are four programmable registers to control the subpicture bit buffer read and write processes, as shown in [Figure 97](#).

- Bit buffer base address (VID_SPB)
This is an offset relative to the ST20 SDRAM base address. It is programmed in units of 2 Kbytes.
- Bit buffer end address (VID_SPE)
This address is an offset relative to the ST20 SDRAM base address. It is programmed in units of 2 Kbytes.
- Bit buffer read pointer (VID_SPREAD)
It is set by software for each subpicture unit. This is done before control is given to the subpicture hardware decoder. This register is double buffered. The shadow register is updated with each field VSYNC event. This pointer is an offset relative to the ST20 SDRAM base address. It is programmed in units of 64-bit words.
- Bit buffer write pointer (VID_SPWRITE)
It is set by the ST20 before transferring each subpicture unit into the bit buffer. This pointer is an offset relative to the ST20 SDRAM base address. It is programmed in units of 64-bit words.

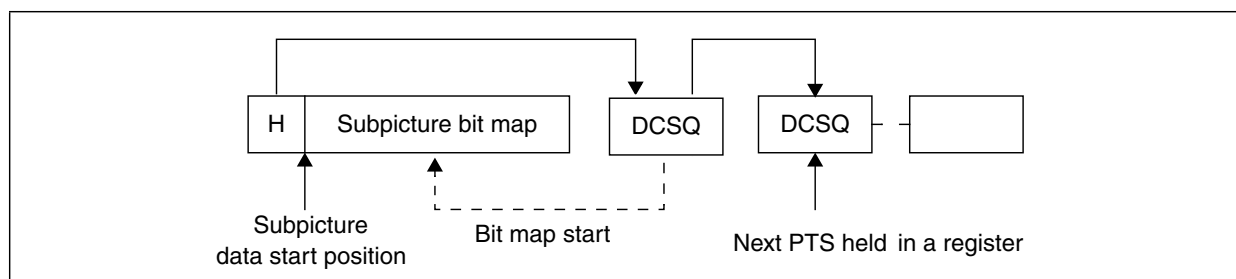
Figure 97: Buffer management



36.3 Subpicture decoder operation

Each subpicture unit data buffer start position is programmed using the register VID_SPWRITE. Subsequently the subpicture header, pixel data and display control sequences are sent via FIFOs to the subpicture decoder. The write into FIFOs is done by DMA or by CPU write. Only data belonging to the subpicture unit (SPUH, PXD, DCSQT) is transferred into the subpicture bit buffer. Subpicture pack headers are removed by the software demultiplexor. The decoder reads the header of the first packet (see [Figure 98](#)) and jumps to the first display control sequence using the command pointer.

Figure 98: Subpicture unit structure



The instructions found in the DCSQ packets enable the subpicture unit to program the palettes and set mixing factors for each region. The DCSQ packets also contain a time stamp, which indicates which image the subpicture information refers to. This information is related to a local time for this subpicture unit. The micro should enable a given subpicture unit at the right global time via registers.

Register bits BOT_TOP(SPD_CTL1) can be used to set which field the subpicture commands are executed on: any field, top field or bottom field. The default is any field.

The overall control of the subpicture decoder is performed by software.

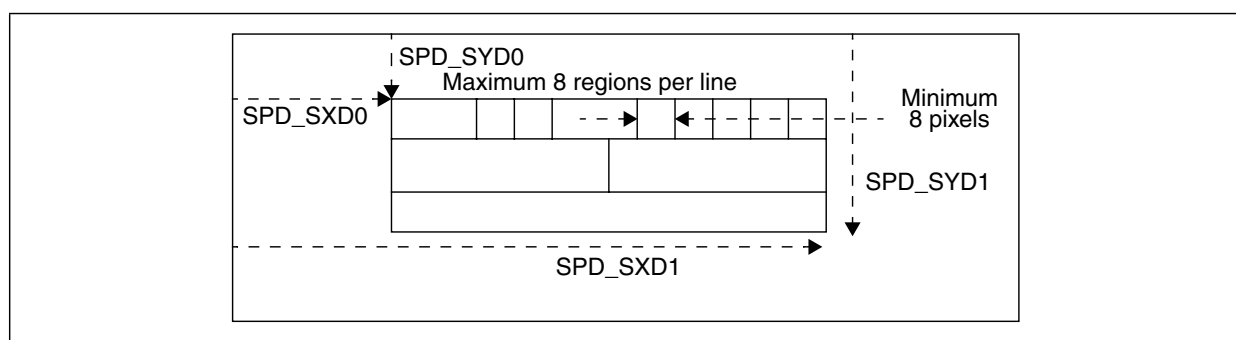
The final information in the DCSQ packet is the region size (rectangle) and the relative position, in bytes, of the bit map start.

A key point here is that the subpicture decoder must read beyond the end of the DCSQ packet in order to verify the next PTS. With this information held in a register, the subpicture decoder knows, in advance, when to change the DCSQ or bit map information. The subpicture unit simply executes the same DCSQ until the image corresponding to the next time stamp is reached.

This is done at the beginning of every field so that the subpicture decoder can load all the relevant information from DCSQ before the first subpicture pixel is required.

The subpicture region declaration is held in registers in the decoder so that the subpicture decoder is turned on and off at the correct position on the screen (see Figure 99). The bit map start pointer indicates where, in the bit map data, to start decoding. When the correct image, corresponding to the local time stamp contained in the DCSQ, should be displayed the subpicture controller enables the subpicture decode for that image.

Figure 99: Subpicture region declaration



A pause mode is defined in the subpicture decoder. As explained previously, the subpicture decoder is autonomous within a subpicture unit.

This means that the DCSQ switching is timed automatically using an internal 90 kHz clock. During video trick modes, where the video stream may be frozen or slowed down the same thing should be possible with the subpicture decoder in order to maintain the synchronization between the two streams.

A pause mode is implemented for the subpicture decoder which stops the 90 kHz counter and therefore pauses the subpicture decoder. This is controlled using the P field in the SPD_CTL1

register and is synchronized to the VSYNC signal. This control bit can therefore be used as a pause and a single step control bit.

The subpicture decoder registers are put together in the subpicture memory map except:

- subpicture software reset (register SPD_SPR),
- subpicture pause mode (P bit of SPD_CTL1),
- subpicture FIFO full (bit 18 of VID_ITS and VID_STA register).

36.4 Subpicture display

36.4.1 Look up tables

There are eleven look up tables inside the subpicture decoder:

- one highlight LUT (2 to 4 bits mapping),
- one subpicture LUT (2 to 4 bits mapping),
- eight PCI LUTs (2 to 4 bits mapping),
- one main LUT (4 to 24 bits mapping).

The subpicture and PCI LUTs are automatically supplied by the decoder itself (subpicture commands contained in the SPU). The highlight and main LUTs need to be loaded by the ST20 (SPD_HCN, SPD_HCOL, SPD_LUT registers).

The output of the subpicture main LUT is mixed with the other planes. The contrast value between these two sources is set by the **set_contr_dcsq** command, by the PCINFs of a **chg_colcon** command or by highlight color information (the highlight LUT has the highest priority, followed by the PCI LUTs. The subpicture LUT has the lowest priority). The mixed video is a 24-bit Y, Cr, Cb video signal where:

- $Y_{MIXED} = [Y_{PLANES} \times (16 - k) + Y_{SUBP} \times k] / 16,$
- $Cr_{MIXED} = [Cr_{PLANES} \times (16 - k) + Cr_{SUBP} \times k] / 16,$
- $Cb_{MIXED} = [Cb_{PLANES} \times (16 - k) + Cb_{SUBP} \times k] / 16,$
- $k = 0$ if contrast value from high light, subpicture, PCI LUTs = 0,
- $k = \text{contrast value} + 1$ if contrast value > 0.

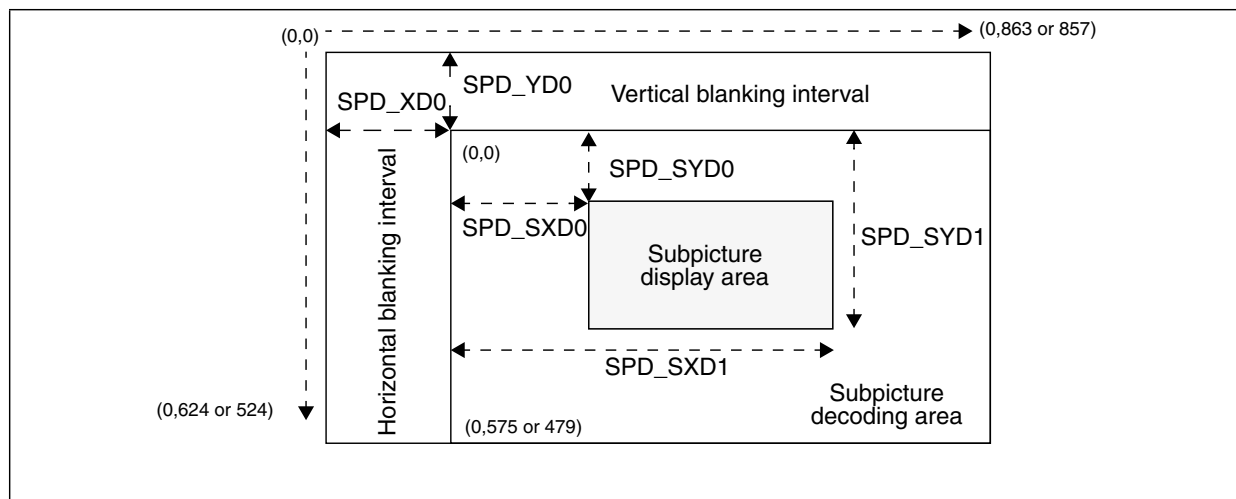
36.4.2 Subpicture areas

The active subpicture decoding area can be 720 x 576 or 720 x 480 pixels. In order to align the subpicture decoding area with the video decoding area, the upper left corner of the active subpicture decoding area has to be set by software, using the SPD_XD0 and SPD_YD0 registers.

Note: The minimum values for SPD_XD0 and SPD_YD0 are 3 and 2 respectively. If the programmed values are less than this, no subpicture is displayed.

The same semantics have been defined as for the video decoder, as shown in [Figure 100](#) below. The active subpicture display area is defined in a similar manner, using the SPD_SXD0, SPD_SYD0, SPD_SXD1 and SPD_SYD1 registers. The highlighted area is defined by the SPD_HLSX, SPD_HLSY, SPD_HLEX and SPD_HLEY registers and is set by software.

Figure 100: Subpicture areas



Confidential

ST Confidential (iPanel)

37 Subpicture decoder registers

Addresses are provided as the *SubPictureBaseAddress* + offset.

The *SubPictureBaseAddress* is:

0x0000 1000.

A register summary is given in [Table 52: Subpicture decoder registers on page 78](#).

SPD_CTL1

Control register 1

7	6	5	4	3	2	1	0
BOT_TOP	P	B	H	V	D	S	

Address: *SubPictureBaseAddress* + 0x00

Type: Read/write

Reset: 0

Description: VSYNC (field) synchronization

This register contains control bits for the subpicture decoder.

[7:6] BOT_TOP

- 00: Subpicture commands execute on any field
- 01: New subpicture command execute on top field
- 10: New subpicture command execute on bottom field
- 11: Reserved

[5] P: Subpicture pause

When set, the 90 kHz clock in subpicture is paused.

[4] B: Bypass

When set, the run length decoder is put into transparent mode. This allows standard 2-bit per pixel bit maps to be fed into the subpicture decoder.

[3] H: Highlight enable

When set, highlighting is turned on inside the subpicture display area.

[2] V: Display active

When reset, the subpicture display is turned off. However, decoding continues even when the display is disabled. When decoding is disabled using the decoder active bit then the display is automatically disabled.

[1] D: Decoder active

This bit is set by the decoder when, at shadow register update, the subpicture start bit is sampled high. When the decoder active bit is reset decoding is disabled and can only be re-enabled by the decoder start bit.

[0] S: Subpicture decoder start command

When this bit is set it indicates the start of a new subpicture unit. The subpicture decoder then resets the local time reference counter. The state is sampled on each VSYNC.

SPD_CTL2**Control register 2**

7	6	5	4	3	2	1	0
Reserved							RC

Address: *SubPictureBaseAddress* + 0x02

Type: Read/write

Reset: 0

Description: No synchronization

This register contains control bits for the subpicture decoder.

[7:1] **Reserved**

[0] **RC:** Reset LUT #2

This bit, when set, resets the autoincrement address counter.

SPD_HCN**Highlight region contrast**

	7	6	5	4	3	2	1	0
0x16	E2[3:0]				E1[3:0]			
0x17	P[3:0]				B[3:0]			

Address: *SubPictureBaseAddress* + 0x16 and 0x17

Type: Read/write

Reset: 0

Description: VSYNC synchronization

These registers contain the highlight contrast map values.

SPD_HCOL**Highlight region color**

	7	6	5	4	3	2	1	0
0x14	E2[3:0]				E1[3:0]			
0x15	P[3:0]				B[3:0]			

Address: *SubPictureBaseAddress* + 0x14 and 0x15

Type: Read/write

Reset: 0

Description: VSYNC synchronization

These registers contain the highlight color map values.

Confidential

SPD_HLEX Highlight region end X

	7	6	5	4	3	2	1	0
0x10	Reserved						SPD_HLEX[9:8]	
0x11	SPD_HLEX[7:0]							

Address: *SubPictureBaseAddress* + 0x10 and 0x11

Type: Read/write

Reset: 0

Description: VSYNC synchronization
Highlight region end position X coordinate.

SPD_HLEY Highlight region end Y

	7	2	1	0
0x12	Reserved		SPD_HLEY[9:8]	
0x13	SPD_HLEY[7:0]			

Address: *SubPictureBaseAddress* + 0x12 and 0x13

Type: Read/write

Reset: 0

Description: VSYNC synchronization
Highlight region end position Y coordinate.

SPD_HLSX Highlight region start X

	7	6	5	4	3	2	1	0
0x0C	Reserved						SPD_HLSX[9:8]	
0x0D	SPD_HLSX[7:0]							

Address: *SubPictureBaseAddress* + 0x0C and 0x0D

Type: Read/write

Reset: 0

Description: VSYNC synchronization
Highlight region start position X coordinate.

SPD_HLSY Highlight region start Y

	7	6	5	4	3	2	1	0
0x0E	Reserved						SPD_HLSY[9:8]	
0x0F	SPD_HLSY[7:0]							

Address: *SubPictureBaseAddress* + 0x0E and 0x0F

Type: Read/write

Reset: 0

Description: VSYNC synchronization
Highlight region start position Y coordinate.

Confidential

SPD_LUT**Main lookup table**

7 6 5 4 3 2 1 0

SPD_LUT[7:0]							

Address: *SubPictureBaseAddress* + 0x03

Type: Write only

Reset: 0

Description: No synchronization

This register allows input of the main lookup table. Writing to this register autoincrements the address in the lookup table. For each color, starting with color 0, the Y component (8-bit) is written first followed by U and V. The process continues for each color up to 16 colors.

SPD_SPR**Soft reset**

7 6 5 4 3 2 1 0

Reserved							SPR

Address: *SubPictureBaseAddress* + 0x01

Type: Read/write

Reset: 0

Description: No synchronization

This register resets the subpicture decoder.

[7:1] **Reserved**

[0] **SPR:** Subpicture reset

When this bit is set the subpicture decoder and its FIFOs are reset. Note that the subpicture decoder is also reset by a hard reset or a global soft reset.

SPD_SXD0**Subpicture display area**

7 6 5 4 3 2 1 0

0x24	Reserved						SPD_SXD0[9:8]	
0x25	SPD_SXD0[7:0]							

Address: *SubPictureBaseAddress* + 0x24 and 0x25

Type: Read/write

Reset: 0

Description: VSYNC synchronization

These registers contain the parameters which define the subpicture display area within the subpicture decode area. The value represents an offset from the corresponding parameter used to define the subpicture decode area.

For example: The true horizontal start position of the subpicture display is equal to XDO + SXDO.

SPD_SXD1**Subpicture display area**

	7	6	5	4	3	2	1	0
0x28	Reserved						SPD_SXD1[9:8]	
0x29	SPD_SXD1[7:0]							

Address: *SubPictureBaseAddress* + 0x28 and 0x29

Type: Read/write

Reset: 0

Description: VSYNC synchronization

These registers contain the parameters which define the subpicture display area within the subpicture decode area. The value represents an offset from the corresponding parameter used to define the subpicture decode area.

SPD_SYD0**Subpicture display area**

	7	6	5	4	3	2	1	0
0x26	Reserved						SPD_SYD0[9:8]	
0x27	SPD_SYD0[7:0]							

Address: *SubPictureBaseAddress* + 0x26 and 0x27

Type: Read/write

Reset: 0

Description: VSYNC synchronization

These registers contain the parameters which define the subpicture display area within the subpicture decode area. The value represents an offset from the corresponding parameter used to define the subpicture decode area.

SPD_SYD1**Subpicture display area**

	7	6	5	4	3	2	1	0
0x2A	Reserved						SPD_SYD1[9:8]	
0x2B	SPD_SYD1[7:0]							

Address: *SubPictureBaseAddress* + 0x2A and 0x2B

Type: Read/write

Reset: 0

Description: VSYNC synchronization

These registers contain the parameters which define the subpicture display area within the subpicture decode area. The value represents an offset from the corresponding parameter used to define the subpicture decode area.

Confidential

SPD_XD0**Subpicture X offset**

	7	6	5	4	3	2	1	0
0x04	Reserved						SPD_XD0[9:8]	
0x05	SPD_XD0[7:0]							

Address: *SubPictureBaseAddress* + 0x04 and 0x05

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register value sets the horizontal position of the left hand side of the active subpicture decode region. The position is measured in number of pixels from the left hand edge of the screen.

The minimum values for SPD_XD0 and SPD_YD0 are 3 and 2 respectively. If the programmed values are less than this, no subpicture is displayed.

SPD_YD0**Subpicture Y offset**

	7	6	5	4	3	2	1	0
0x06	Reserved						SPD_YD0[9:8]	
0x07	SPD_YD0[7:0]							

Address: *SubPictureBaseAddress* + 0x06 and 0x07

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register value sets the vertical position of the top of the active subpicture decode region. The position is measured in number of pixels from the top edge of the screen.

The minimum values for SPD_XD0 and SPD_YD0 are 3 and 2 respectively. If the programmed values are less than this, no subpicture is displayed.

Confidential

38 Display planes

38.1 Overview

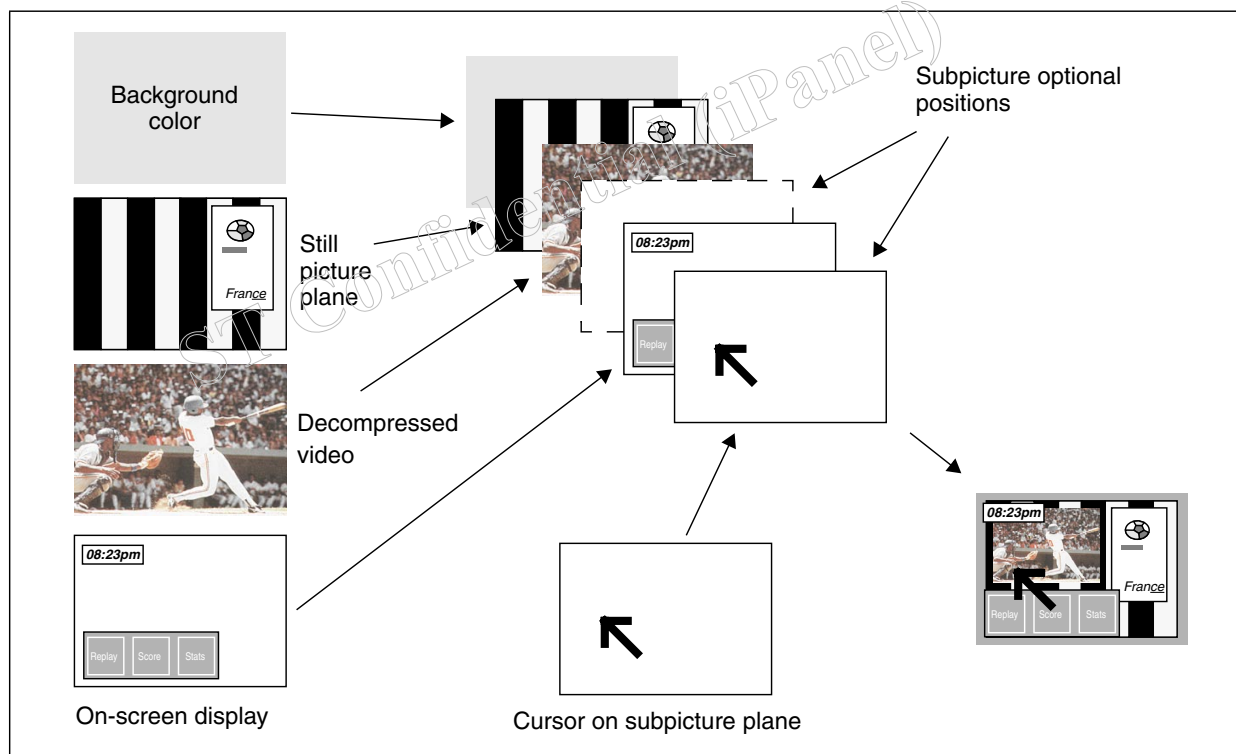
The graphics and display subsystem reads, processes, overlays and mixes pixel data stored in various buffers in SDRAM, and produces a combined image for display on a TV. The buffers are called the display planes.

The subsystem assumes five display planes as follows:

- background color (see [Section 38.2: Background color on page 345](#)),
- still picture and graphics plane, 4:2:2 YCbCr (see [Section 38.3: Still picture plane on page 346](#)),
- MPEG video plane (see [Section 38.4: MPEG video plane on page 348](#)),
- on screen display plane (OSD) (see [Section 38.5: On-screen display \(OSD\) on page 358](#)),
- subpicture or cursor plane (see [Section 38.6: Subpicture or cursor plane on page 373](#)).

The display planes are shown in [Figure 101](#).

Figure 101: Display planes

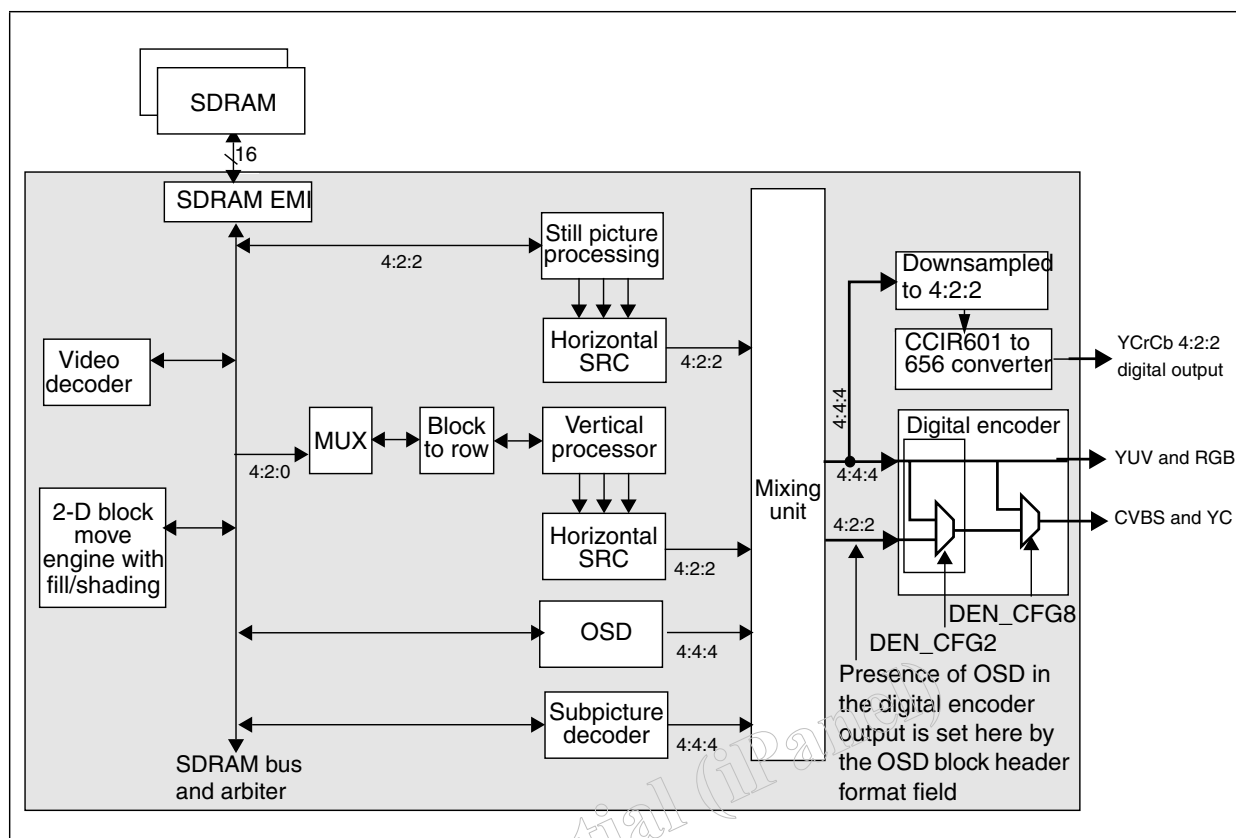


The display planes are normally overlaid in the order shown above, with the background color at the back and the subpicture used as a cursor plane at the front. The position of the subpicture plane can be programmed by register bit SPO in register VID_OUT as:

- the most forward layer to be a cursor plane or a second on screen display plane (see [Figure 101](#)),
- behind the OSD plane, in front of the MPEG video as a second on screen display plane.

[Figure 102](#) is a simplified block diagram of the graphics and display subsystem.

Figure 102: Graphics and display subsystem



The planes can be blended together using the mixing unit, as described in [Section 38.7: Mixing display planes](#) on page 373.

The mixing unit has two outputs as illustrated in [Figure 102](#).

- A 4:2:2 output that is input to the digital encoder to generate CVBS and YC output. This format is generally used for VCR recording. The OSD and subpicture display planes can be disabled from this output as described in [4:2:2 output control](#) on page 375.
- A 4:4:4 output, used in either of the following ways.
 - As an input to the digital encoder to generate the YUV and RGB signals. This is generally used for TV display, and includes all of the available display planes.
 - As a digital signal that bypasses the digital encoder and is output to the YUV pins YC[0:7]. This signal is downsampled to 4:2:2 format by removing the chroma. It is used, for example, for connection to an external graphics device.

The routing of the 4:2:2 and 4:4:4 signals from the mixing unit, through the digital encoder to the DACs, is controlled by the DEN_CFG2 and DEN_CFG8 registers. When the YCrCb 4:2:2 signal is used, the presence of OSD in the digital encoder output is selected or deselected by the OSD block header format field S (see [Table 148: OSD block header format](#) on page 368). When the YCrCb 4:4:4 signal is used, OSD is always present in the digital encoder output.

38.2 Background color

The background color is always the back plane of the video plus the still picture, with all the other display planes on top. The color of the background is defined using the three registers BCK_Y, BCK_U and BCK_V.

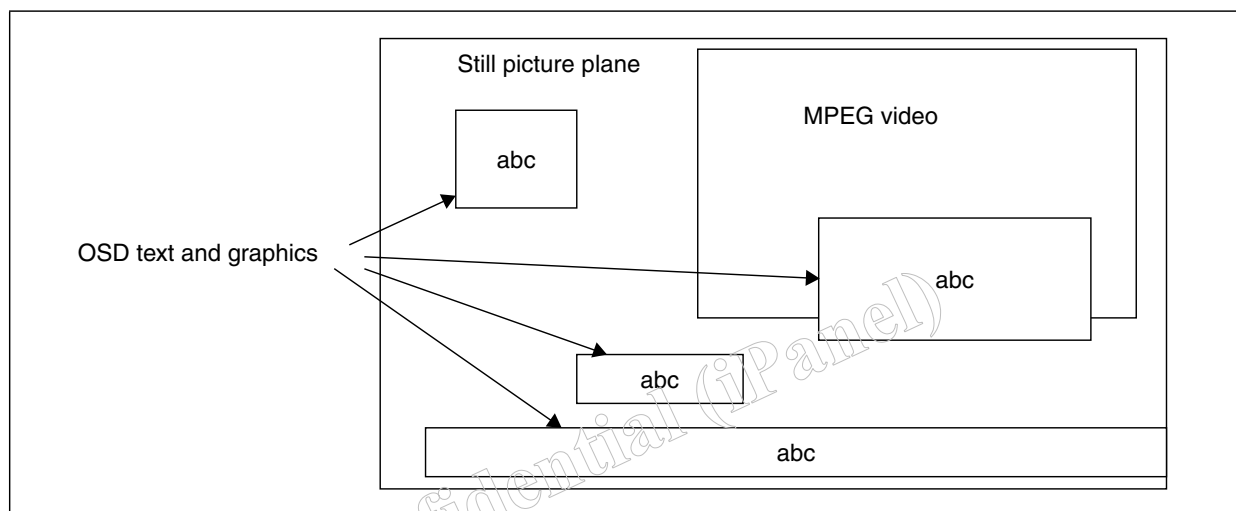
38.3 Still picture plane

The still picture plane is sometimes referred to as the third display layer (TDL).

The still picture plane can hold a picture stored in memory as a video backdrop. The picture can be, for example, a JPEG or MPEG still image. The display priority has the still picture backdrop as the second layer from the bottom, above the background color. The MPEG video, the subpicture and then the OSD are above the still picture plane. A typical example using three of the layers is shown in [Figure 103](#).

The still picture plane can be faded in and out of the background color and then mixed with the video plane.

Figure 103: Use of the still picture plane

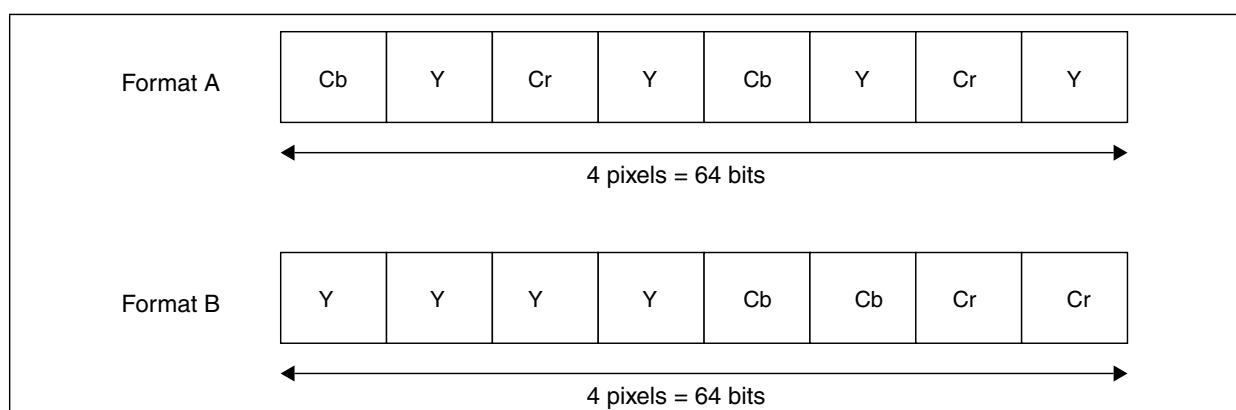


38.3.1 Still pictures

The still picture is stored in 4:2:2 raster scan format, ordered either as format A or format B in the figure below. The format is selected by register bit TDL_DCF[4].

Format A has alternating luma and chroma data types; format B has grouped luma and chroma data types like the video format. Using format B speeds up the software conversion of the video format to the still picture 4:2:2 raster format, as the data can be copied in blocks of 32 bits.

Figure 104: Formats of the still picture plane in memory



The still picture plane pipeline is a duplicate of the video plane pipeline, without the block-to-row converter or the vertical filter. The following features operate in the same way as the video plane:

- horizontal sample rate conversion,
- picture start X/Y coordinates,
- programmable size.

The start point of the displayed picture with respect to the decoded picture (the picture start X/Y co-ordinate or pan/scan vector) is programmed by the register TDL_SCN. The vertical resolution is by line, and the horizontal resolution is by 1/8 pixel. Any part of the decoded picture that falls above or to the left of these coordinates is cut off.

The displayed still picture location and size is defined by registers TDL_XDO, TDL_XDS, TDL_YDO and TDL_YDS. Register TDL_TDW gives the number of macroblocks displayed in one row. The picture width must be stored as a multiple of 64 pixels.

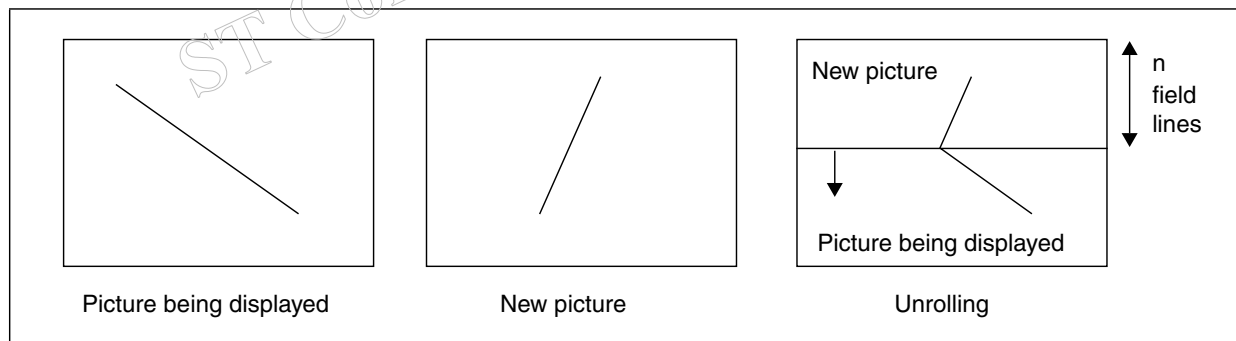
38.3.2 Upscaling and downscaling the picture

Although there is no vertical filter, any part of the decoded still picture can be displayed with scaling, by a factor of 2 or 0.5 in the vertical and horizontal directions. The vertical scaling is enabled by setting bit UDS in register TDL_DCF. The field line repeat mode for upsampling is selected by setting bit UND and line dropping mode for downsampling is selected by clearing UND. Horizontal scaling is performed by the sample rate converter using TDL_LSR. The starting point, location and size of the display are programmed as for unscaled pictures.

38.3.3 Picture unrolling

When a new still picture is needed, it may be progressively unrolled on to the screen to replace an existing displayed still picture. This effect is shown in [Figure 105](#).

Figure 105: Still picture unrolling



The two still pictures to be displayed must be stored in memory and must have the same size and the same resolution. The displayed still picture is addressed by TDL_TOP and TDL_TEP, and the new one by TDL_TOP2 and TDL_TEP2. When the unrolling is enabled, the still picture display controller switches automatically from the new picture to the old picture when the programmed number of lines have been displayed. The number of lines before the switch (n in [Figure 105](#)) is set by software, so the software can control the speed of the unrolling. The number of lines is held in TDL_SWT, and when this value is greater than zero the unrolling starts.

38.4 MPEG video plane

38.4.1 Introduction

The MPEG video plane displays a moving image decoded from an MPEG video stream by the MPEG video decoder. The MPEG video is displayed as the third layer, below subpicture and OSD, and above background color and still picture plane. The picture data is received either from the display frame buffer area of the external memory.

The data is passed as an MPEG macroblock into the block-to-row converter, which generates a line based raster from the frame store. It also performs pan/scan operation and vertical postprocessing of the decoded video. The block-to-row converter is described in [Section 38.4.3: Block-to-row converter on page 353](#).

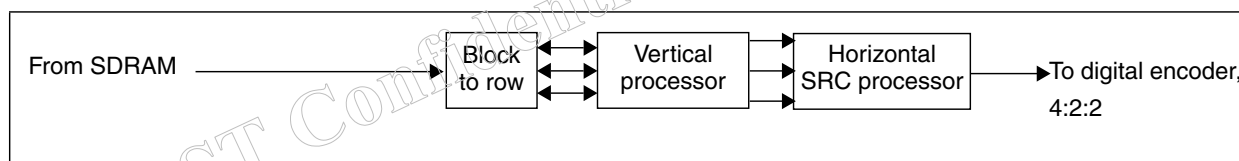
The output of the block-to-row converter is fed to the horizontal sample rate converter (SRC), which is an 8-tap filter with two functions:

- upscaling and downscaling of pixel data when the displayed line length is greater or smaller than the decoded picture width,
- implementation of the fractional part of the pan scan horizontal offset.

The SRC outputs are upsampled lines each having equal numbers of luminance and chrominance samples. The SRC can be bypassed if necessary. The sample rate converter is described in [Section 38.4.2: MPEG video sample rate converter \(horizontal filtering\) on page 349](#).

[Figure 106](#) illustrates the MPEG video plane pipeline.

Figure 106: MPEG video plane pipeline

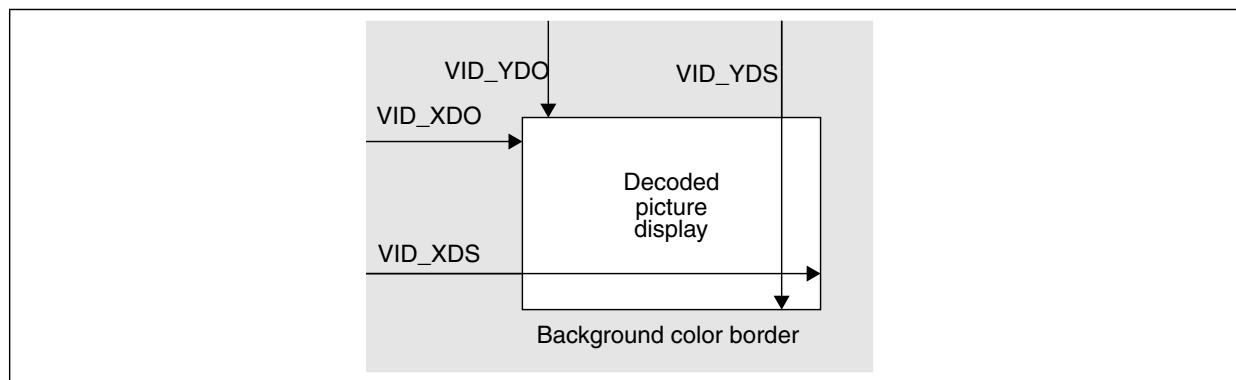


Setting up an MPEG video display

The VID_DFP and VID_DFC registers must be set up with the base address of the buffer containing the picture to be displayed. This register is double buffered; when a new value is written it is taken into account on the occurrence of a VSYNC. Therefore, it is possible to write a new value for this pointer every field, although it would normally be updated only once per frame.

The picture stored in the buffer is always treated as a frame. If ever no display is required, bit EVD in register VID_DCF may be reset and a constant black value is output. The size and location of the display window is defined by registers VID_XDO, VID_XDS, VID_YDO and VID_YDS. The register values define the horizontal and vertical boundaries of the displayed picture, as shown in [Figure 107](#).

Figure 107: Display window positioning



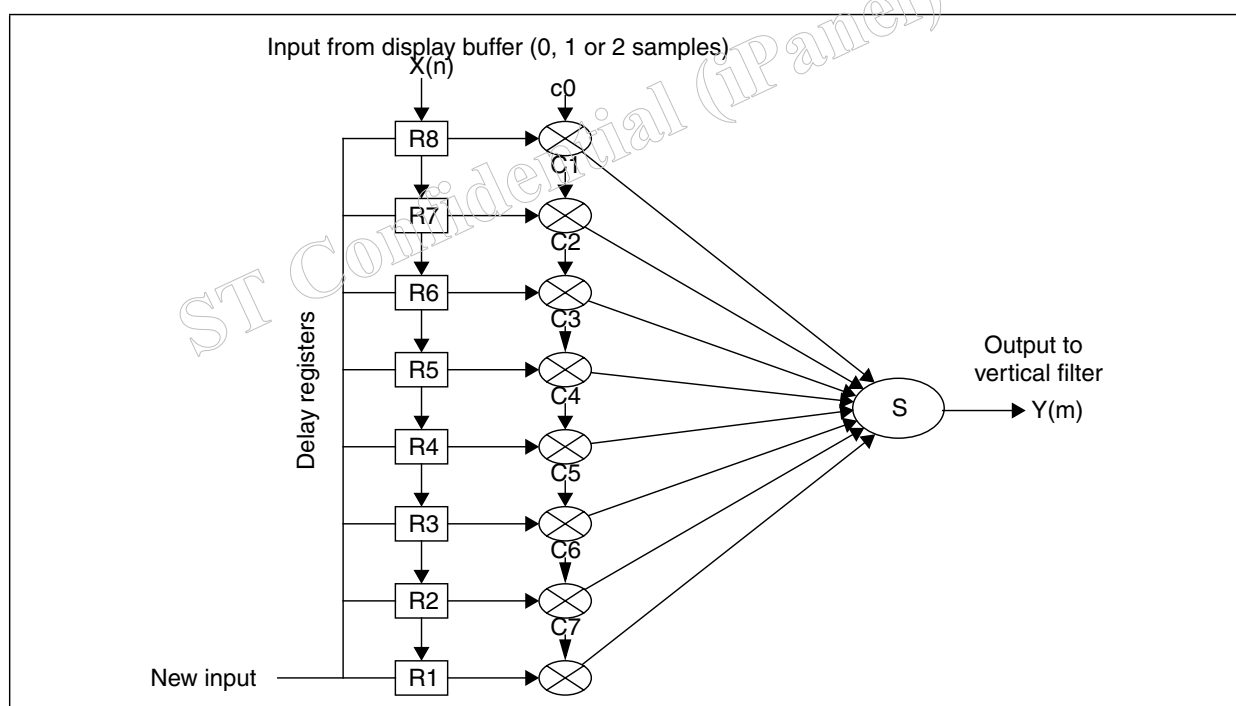
38.4.2 MPEG video sample rate converter (horizontal filtering)

The sample rate converter upsamples or downsamples picture data to increase or decrease the number of horizontal samples in a line.

- Upsampling is required if the horizontal size of the display is greater than the decoded picture width. For example, to display a 720 pixel width 16:9 source image on a 720 pixel width 4:3 display, then 540 pixels selected from each source line must be upsampled to 720.
- Downsampling is required when the resolution of the display is less than that of the decoded image. For example when square pixels are required for an NTSC image the 720 pixel wide image decoded must be downsampled to 640 pixels.

To enable the SRC, register bit DSR in register VID_DCF must be reset to zero; if this bit is set to 1, the SRC is bypassed and the horizontal resolution of the decoded picture is not changed. The sample rate converter can change the sampling rate by a programmable factor. The upsampling ratio is limited to 8 and the downsampling to less than or equal to a factor of 2. The same filter is used both for upsampling and downsampling. If either extreme limits is approached, artifacts may appear in the displayed image. The SRC operates by directly interpolating samples required for the new sampling rate by using those of the decoded picture data read from the display buffer. This is performed by an 8-tap interpolation filter with the structure shown in [Figure 108](#).

Figure 108: 8-tap interpolation filter



The filter has three sets of delay registers multiplexed between the Y , C_B and C_R samples. It has eight sets of coefficients, each set defining one of eight subpixel interpolation positions. Consider an upsampling example, for subpixel position 0, the output is aligned with stored sample R4, for subpixel position 1, the output corresponds to an interpolated pixel position one eighth of the distance from sample R4 to sample R5, and so on. The number of inputs clocked into the SRC is equal to the number of samples used in each line of the source image, and the number of outputs generated is equal to the number of samples displayed. Thus the rate of generation of outputs are greater than the input data rate in the case of upsampling and less in the case of downsampling.

MPEG video SRC operation

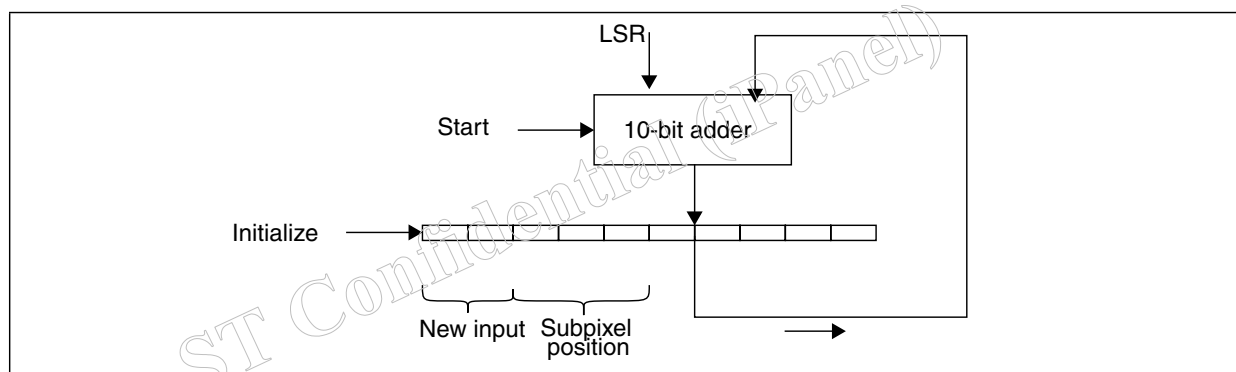
The sample rate converter works in the following manner: The SRC takes a block of M samples of the input signal denoted as $x(n')$, $n' = 0, 1, 2, 3, \dots, M - 1$ and computes a block of L output samples $y(m')$, $m' = 0, 1, 2, \dots, L - 1$.

For each output sample time m' , $m' = 0, 1, 2, \dots, L - 1$ the eight samples in the filter are multiplied with one of the eight sets of filter coefficients the products are accumulated to give the output $y(m')$. Each time the quantity $m'M / L$ increases by one, one sample from the input buffer is shifted into the filter.

The coefficient set used depends on the position of the sample being generated relative to the original samples of the source image. Thus after L output values are computed M input samples have been shifted into the filter delay registers.

The SRC upsampling and downsampling factor is set up in the VID_LSR register. The resampling factors for the luminance and chrominance components are exactly the same. The resampling factor is equal to L / M . The value programmed into VID_LSR is $256 \times M / L$. This value is used to determine both the rate of input of data into the filters and the sequence of subpixel interpolation positions. The mechanism by which this is achieved is shown in Figure 109.

Figure 109: Up/down sampling filter control



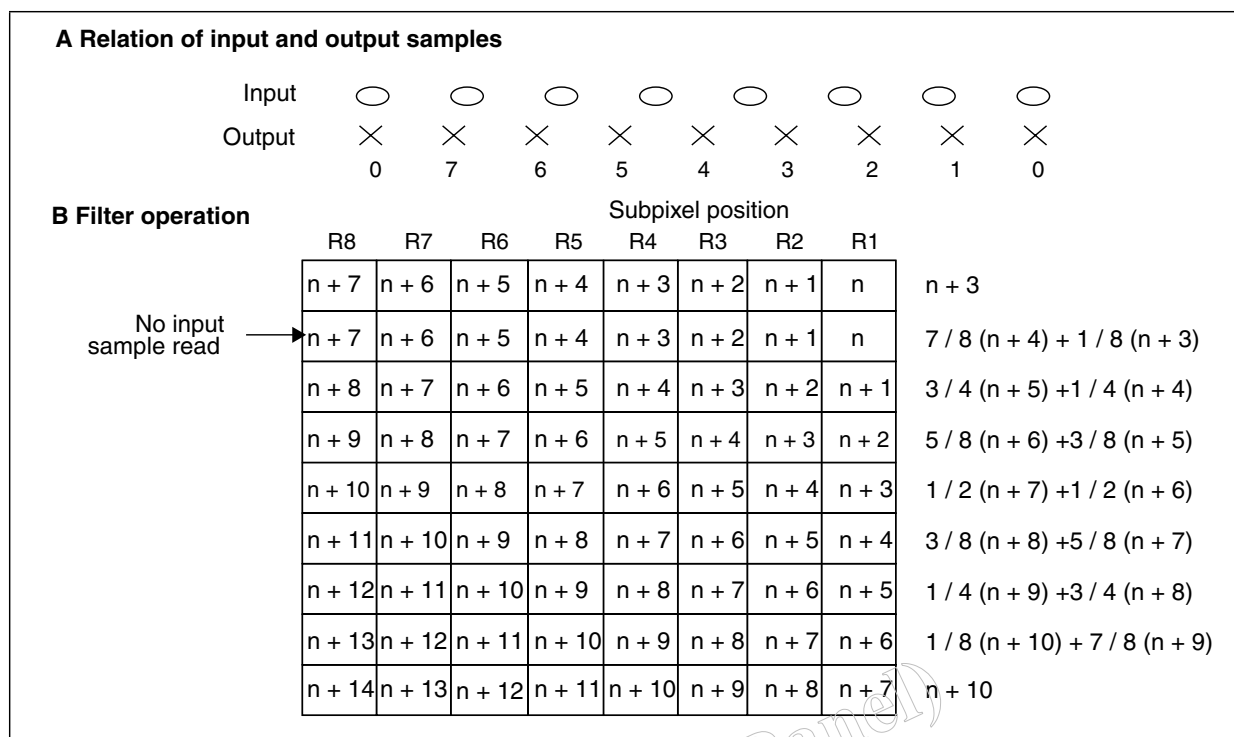
MPEG video upsampling example

The example in Figure 110, illustrates the operation of the sample rate converter when the upsampling ratio is 8:7. For every eight samples clocked out of the filters, seven samples are clocked in.

To illustrate the interpolation positions, at the right of Figure 110 are shown the outputs which would occur with a simple linear interpolation (that is, a 2-tap filter). The actual SRC output values are the 8-tap filter outputs with coefficients appropriate to subpixel positions 0, 7, 6, 5, 4, 3, 2, 1, 0.

The SRC output is limited to lie within the range $[1:254]$, so the codes 0x00 and 0xFF are never output, giving compatibility with ITU-R 656.

Figure 110: SRC example for 8:7 upsampling



The VID_LSR value is added into an accumulator register at a rate equal to the filter output rate. The top two bits indicate how many new inputs are to be loaded into the filter (0, 1 or 2). The next three bits of the accumulator register are used to select the subpixel position. For example, with an upsampling factor of 8:7, the VID_LSR value is $(256 / 8) \times 7 = 224$. The sequence of values in the accumulator register are as shown in Table 138, assuming that it is initialized to zero.

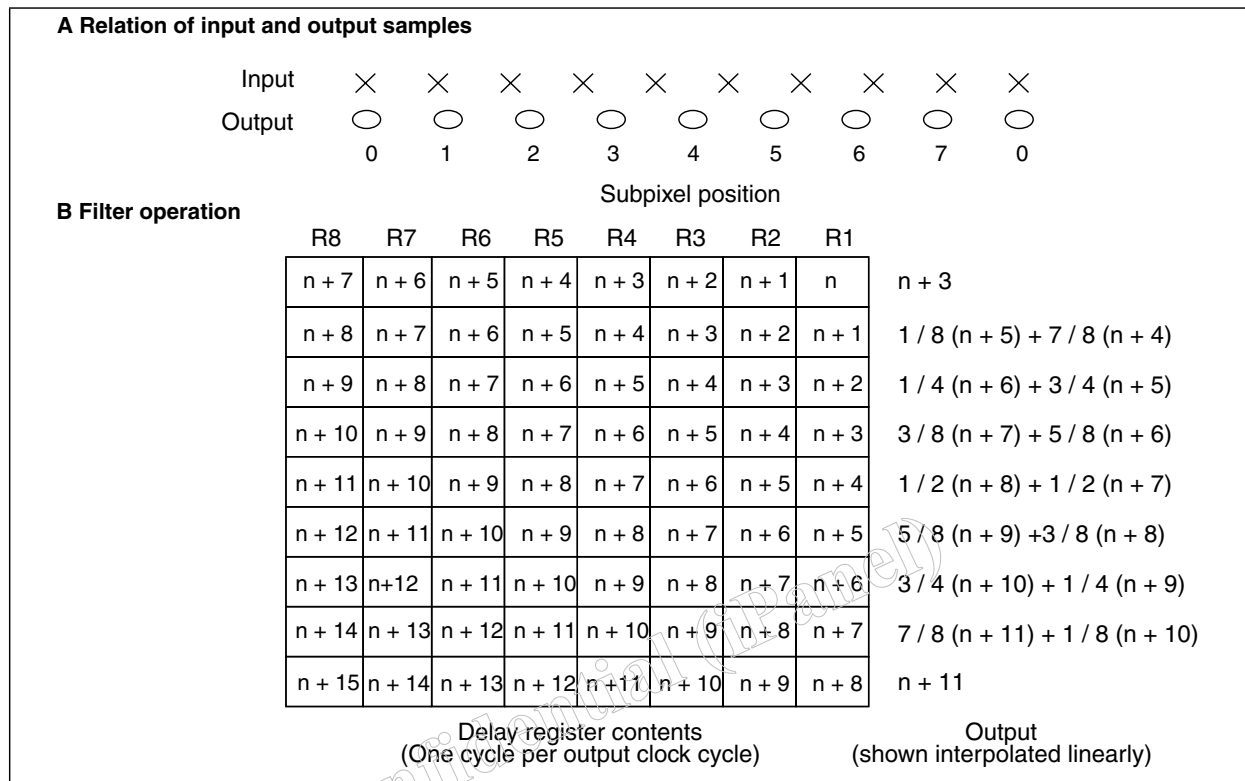
Table 138: Accumulator register sequence for upsampling example

Accumulator register	New input	Subpixel position
0	Yes	0
224	No	7
192	Yes	6
160	Yes	5
128	Yes	4
96	Yes	3
64	Yes	2
32	Yes	1
0	Yes	0

The VID_LSR value thus defines a cycle of subpixel positions as well as the rate of data input. If a value of less than 32 is loaded into VID_LSR, that is, an upsampling ratio of greater than 8 is defined, there could be repeated values in the filter output. This may cause unacceptable display artifacts.

MPEG video downsampling example

The example shown in Figure 111, illustrates the operation of the sample rate converter when the downsampling ratio is 9:8 (720:640).

Figure 111: SRC example for 9:8 downsampling

The VID_LSR value required for a downsampling ratio of 9:8 is $256 \times 9 / 8 = 288$.

Table 139: Accumulator register sequence for downsampling example

Accumulator register	Subpixel position	Number of inputs
0	0	1
288	1	
576	2	
864	3	
128	4	
416	5	
704	6	
992	7	
0	0	2

At the start of a line, the three sets of delay registers R1, R2 and R3 are loaded with the black value ($Y = 16$, $C_B = C_R = 128$).

The first output is thus derived from the inputs stored in registers R4 to R8. At the end of a line, the last eight input samples are stored in registers R1 to R8.

The last valid interpolation is between the samples stored in R4 and R5. Correct interpolation is not possible beyond this except in the case where the next output is in subpixel position 0. This output is valid since coefficient C0 is zero for this position and the invalid sample beyond the end of the line is ignored.

There is thus no valid interpolation possible between the last four input samples. This is illustrated in Figure 112 below, in which 544 pixels are upsampled to 721, in which the upsampling ratio is 4:3. The VID_LSR register would be loaded with the value 192.

The number of valid outputs generated can be calculated as follows:

The ratio between the number of input and output samples is 256:VID_LSR. Given that the last output sample cannot occupy a position beyond the fourth last input sample, the following inequality is always true:

$$\text{VID\_LSR} (n - 1) \leq 256 (m - 4)$$

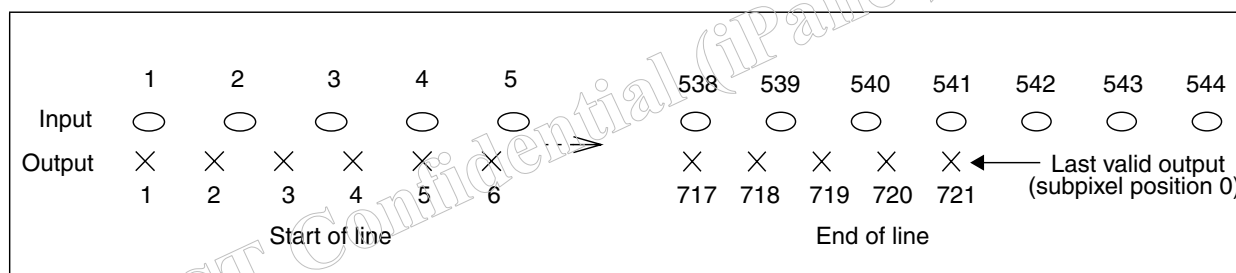
where n is the number of output samples and m is the number of input samples. The value of n is thus given by:

$$n = [256 (m - 4) / \text{VID\_LSR} + 1]$$

where [x] indicates the integer part of x.

The value programmed into the VID_XDS register must be such that all samples beyond the last valid one are masked.

Figure 112: Upsampling from 544 to 720



38.4.3 Block-to-row converter

The block-to-row converter generates a line based raster scan of individual video components (YCbCr) from an MPEG macroblock organized frame store in 4:2:0 or 4:2:2 format.

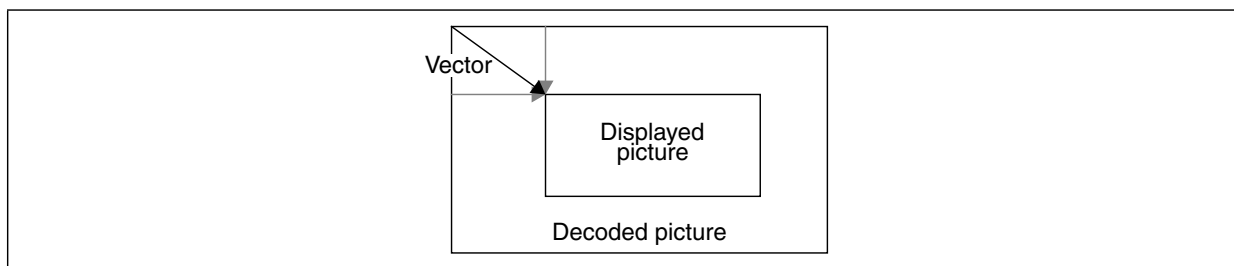
The block-to-row converter also performs:

- the horizontal/vertical pan/scan operation;
- vertical postprocessing of decoded video such as:
 - vertical zoom out x2, x3 (x8 / 3), and x4,
 - vertical programmable filtering with any zoom in, and zoom out up to x2,
- horizontal zoom out x2 (for zoom out by more than x2).

Pan/scan vectors

When the display window has a smaller horizontal dimension than the decoded picture, a vector can be programmed to define the starting point of the displayed picture, as shown in [Figure 113](#).

Figure 113: Pan/scan vector

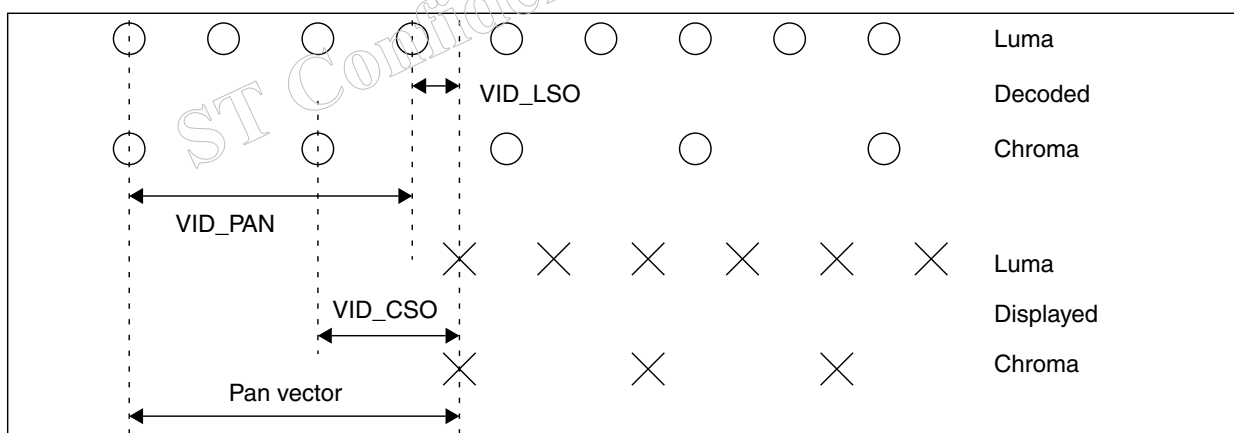


This vector defines the point in the decoded picture which corresponds to the top left corner of the displayed picture. The displayed picture size and location is defined by the numbers programmed in registers VID_XDO, VID_XDS, VID_YDO and VID_YDS.

The pan/scan vector components are programmed into the registers VID_PAN, VID_LSO, VID_CSO, VID_SCN, VID_VFLMODE and VID_VFCMODE. These registers are double buffered; when a new value is written it is taken into account on the occurrence of a VSYNC. Thus it is possible to write a new value of the pan/scan vector for every field.

The integer part of the horizontal component of the pan/scan vector is loaded into the VID_PAN register, and the fractional part defines the contents of the VID_LSO and VID_CSO registers. The relationship between these quantities is illustrated in [Figure 114](#).

Figure 114: Components of the pan/scan vector



The numbers loaded into the VID_LSO and VID_CSO registers are used to initialize the luminance and chrominance upsampling control registers at the start of every line. VID_LSO is set up directly with the value of the fractional part of the pan/scan vector horizontal component. VID_CSO is set up with half of this number, plus 128 if the integer part is an odd number. The resolution to which the horizontal component can be defined is 1/8 pel.

The vertical component of the pan/scan vector is programmed into VID_SCN in units of framestore macroblock rows (that is, units of 16 lines). A vertical scan, with a resolution of 1/8 line, can then be performed by programming the OFFEVEN and OFFODD bits of the VID_VFLMODE and VID_VFCMODE registers.

Vertical filter

The block-to-row converter has an output filter for vertical postprocessing of the video. This vertical filter performs the chroma reconstruction from 4:2:0 to 4:2:2 format, and upsamples and downsamples the luma and chroma components. Any zoom, from any zoom in to zoom out by 4, can be performed in addition to the following basic modes:

- zoom in by 2,
- zoom out by 2, 3 or 4,¹
- zoom out by n (where n lines are produced for each line stored, $1/16 \leq n \leq 2$),
- 16:9 and 14:9 letter box filtering.

The programmable PAL/NTSC vertical filter optimizes video quality according to the type of source, full or half resolution, interlaced or progressive. One luma filter operation and one chroma filter operation can run at the same time. The chroma filter mode is programmed in the register *VID_VFCMODE* on page 392 register, and the luma filter mode is programmed in register *VID_VFLMODE* on page 393. Table 140: *Vertical filter modes* on page 357 lists the recommended configurations.

The interface between the block-to-row converter and the next block (horizontal sample rate converter (SRC)), has three video components. The video data output is in 4:2:2 format.

Successive samples are presented at the relevant video component port (Y, C_r or C_b) synchronously with the relevant output sample clocking signal.

Horizontal compression

For zoom out by three or four, a frame store must not only be compressed vertically by three or four, but also horizontally by three or four. The SRC is able to downsample up to a factor of two.

For zoom out by three or four horizontally, the block-to-row converter must preprocess its output data for the SRC. This is set by the HORIZDN bit (bit 39) of the *VID_VFCMODE* and *VID_VFLMODE* registers.

Filter modes

The vertical filter supports an unlimited number of configurations, however, recommended configurations are listed in Table 140 .

1. STMicroelectronics recommends that the register bit ZOOMOUTX2 (in *VID_DIS*) is set to save bandwidth while performing a zoom out, as a zoom out by 2 in the block-to-row is only then necessary.

Figure 115: Filter mode examples

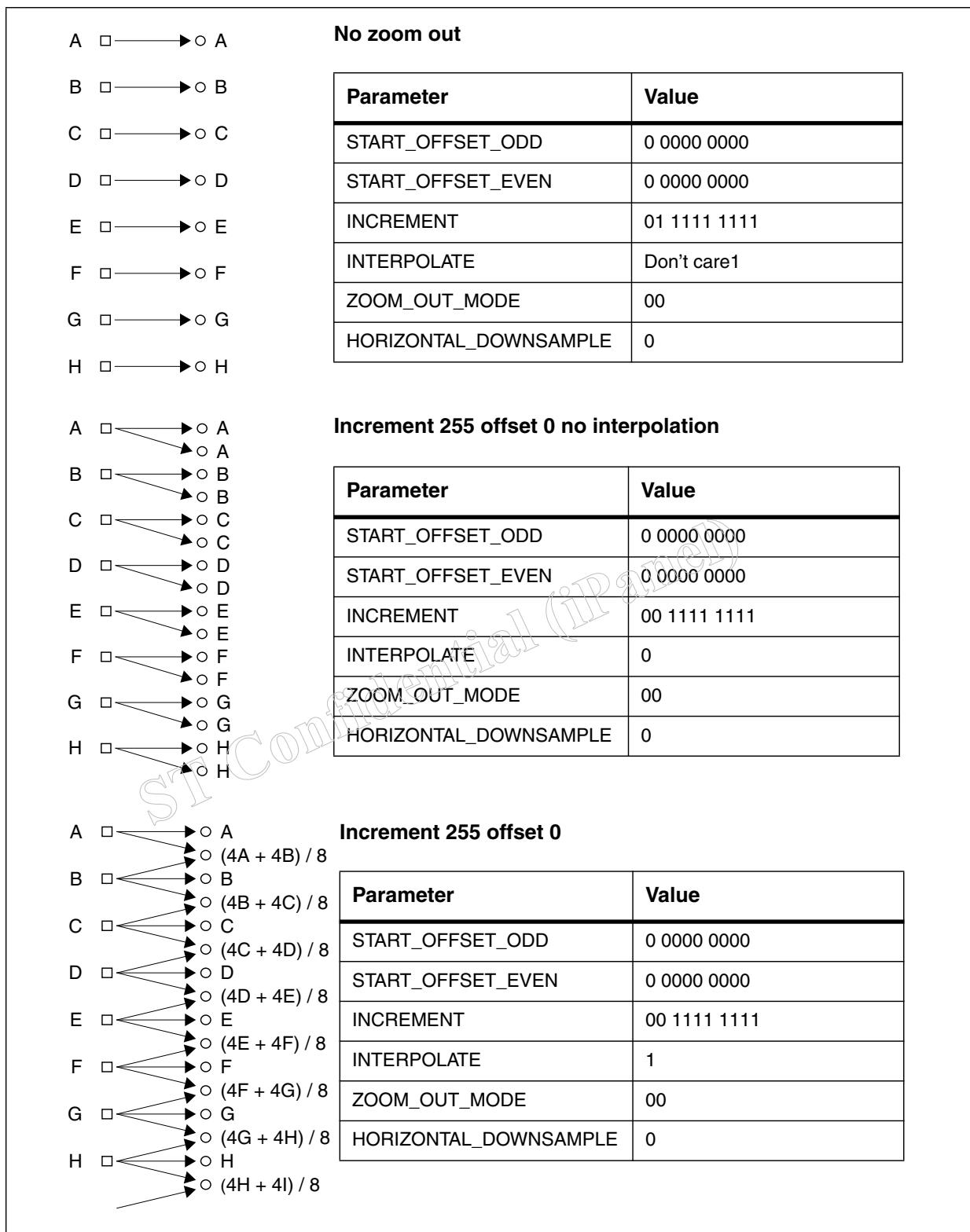


Table 140: Vertical filter modes

Display	Source resolution		Luminance register (VID_VFLMODE)						Chrominance register (VID_VFCMODE)						VID_DIS	
	Spatial	Temporal ^a	[39]	[38]	[37:36]	[35:23]	[22:10]	[9:0]	[39]	[38]	[37:36]	[35:23]	[22:10]	[9:0]	[1]	[0]
			HORIZDN	INTP	ZOOMOUT	OFFODD	OFFEVEN	INCREMENT	HORIZDN	INTP	ZOOMOUT	OFFODD	OFFEVEN	INCREMENT	FBS	FBS
Full screen	Full	I or P	0	0	0	512	512	511	0	1	0	64	192	255	0	0
		I or P	0	0	0	512	512	511	0	1	0	640	384	511	0	0
		P	0	1	0	1536	1024	1023	0	1	0	640	384	511	1	1
	Half	I or P	0	1	0	255	127	253	0	1	0	0	0	126	0	0
		P	0	0	0	127	0	511	0	0	0	0	0	255	1	1
16:9 letter box	Full	I or P	0	1	0	597	512	681	0	1	0	106	192	340	0	0
		P	0	1	1	853	512	681	0	1	0	725	384	681	1	1
	Half	I	0	1	0	255	127	339	0	1	0	0	0	169	0	0
		P	0	1	0	127	0	679	0	1	0	0	0	339	1	1
14:9 letter box	Full	I field-based	0	1	0	0	36	583	0	1	0	0	0	291	0	0
480 to 576	Full	I or P	0	1	0	512	469	425	0	1	0	192	42	212	0	0
576 to 480	Full	I or P	0	1	0	512	563	611	0	1	0	192	90	305	0	0
zoom in x2	Full	I or P	0	1	0	384	512	255	0	1	0	0	192	127	0	0
	Full	P	0	1	0	256	191	511	0	1	0	0	0	255	1	1
zoom in x4	Full	I	0	1	0	255	511	127	0	1	0	0	0	63	0	0
zoom in x4	Half	I	0	1	0	255	288	63	0	1	0	0	0	31	0	0
zoom in x2	Half	P	0	1	0	128	0	255	0	1	0	0	0	127	1	1
zoom out x2	Full	I or P	0	1	0	0	0	1019	0	1	0	0	0	509	0	0
	Half	I	0	1	0	255	255	509	0	1	0	0	0	253	0	0
	Half	P	0	1	1	127	0	509	0	1	0	0	0	509	1	1
zoom out x3	Full	I or P	1	0	2	0	0	573	1	0	1	0	0	382	0	0
	Half	I	0	1	1	0	0	380	0	1	0	0	0	380	0	0
	Half	P	0	0	2	0	0	573	0	0	1	0	0	382	1	1
zoom out x4	Full	I or P	1	0	3	0	0	509	1	0	3	0	0	254	0	0
	Half	I	0	1	0	0	0	1019	0	1	0	0	0	509	0	0
	Half	P	0	0	3	255	64	507	0	0	3	0	0	253	1	1

a. I = Interfaced, P = Progressive

38.4.4 Degradation mode

Sometimes the device is used in a configuration where the available bandwidth on the SDRAM interface is limited. The possible reasons for this include a low clock frequency to use cheaper SDRAMs, or the processor making heavy use of the SDRAM. MPEG decode and display, being a real time process and also a heavy user of SDRAM memory bandwidth then requires a graceful degradation mode.

A small piece of hardware is implemented in the decoder to measure the effective distance (in pixels) between the display process and the decode process. Under conditions of limited bandwidth the decoder becomes late and therefore may get caught by the real time limited display process.

Degradation mode can be enabled and disabled using the register VID_PTH. A threshold or minimum allowable distance between the decode and display processes can also be set. If this threshold is crossed the decoder automatically insures that any bi-directionally predicted macroblock access results in only a single prediction access to external memory thus reducing the bandwidth required by the decoder and allowing recovery.

38.5 On-screen display (OSD)

38.5.1 Introduction

The integrated on screen display (OSD) unit overlays the video picture with software generated graphics. The display priority puts the OSD in front of the MPEG video. It can be configured to be either in front of or behind the subpicture plane by clearing or setting bit SPO in register VID_OUT respectively. The OSD can be enabled by setting bit ENA in register OSD_CFG. The OSD bit map is defined with respect to the display area and is independent of the decoded picture size and any pan/scan offset.

Note: The OSD region cannot be defined with an odd number of pixels. It must be defined with an even number of pixels. The output from the OSD is in 4:4:4 format.

Register bit OSD_CFG[5] controls the polarity of the ENOT0 signal going to the OSD block.

The OSD bit map is field-based and has the following features:

- linked list memory management mode or STi3500A compatibility mode,
- selectable 2, 4 or 8 bits per pixel palette modes giving 4, 16 or 256 palette colors,
- either 6-bit luma resolution and 4-bit chroma resolution per component or 8-bit luma resolution and 8-bit chroma resolution,
- programmable 4-bit mixing factor for each OSD region to blend the video plane and OSD data,
- when anti-aliasing is enabled, each color in an OSD region can be assigned a separate 6-bit mixing factor for mixing with video,
- optional antiflicker and antiflutter filters,
- half resolution mode.

The OSD unit uses color look up tables (LUTs), also called palettes, with 2-bit, 4-bit or 8-bit input. A 2-bit LUT means that four colors can be used at once, and each pixel of the bit map occupies only 2 bits of memory. A 4-bit LUT gives 16 colors and an 8-bit LUT gives 256 colors. The palette of 4, 16 or 256 predefined colors is loaded into the SDRAM by software using the shared memory interface. The palette modes are described in [Section 38.5.6: Color palette on page 364](#).

The output from the LUT can be 14-bit pixels (6-bit Y, 4-bit CB, 4-bit CR) or 24-bit pixels (8-bit Y, 8-bit CB, 8-bit CR) plus 1 bit or 6 bits for transparency control. The color modes are described in [Section 38.5.6: Color palette on page 364](#).

The OSD can consist of a number of display regions, each with its own palette and characteristics. The number of OSD regions resident in memory at any time is limited only by the amount of memory available. Each region has a specification, stored in memory, which contains a header, possibly including a palette, and a bit map. The specifications for the regions can be contiguous or linked in a list structure.

OSD data is stored as one 64-bit SDRAM word and the bit map data in each specification is contiguous with the palette information, as shown in [Figure 121: OSD specification on page 363](#).

The bit map refers to the 2-, 4- or 8-bit color definitions in the palette to create the required picture.

During the display of an image a small state machine first reads the OSD region start and stop addresses then it picks up the palette from SDRAM and loads it into the LUT. When the display reaches the OSD start position (defined in the header) the bit map is sent pixel by pixel to the LUT and the display switches from video to the output of the LUT or a mixture of both.

This process continues until the defined OSD stop position. Thus, for the defined OSD region, the video display is overlaid by the colors which are defined by a combination of the LUT and the bit map.

38.5.2 Using the OSD

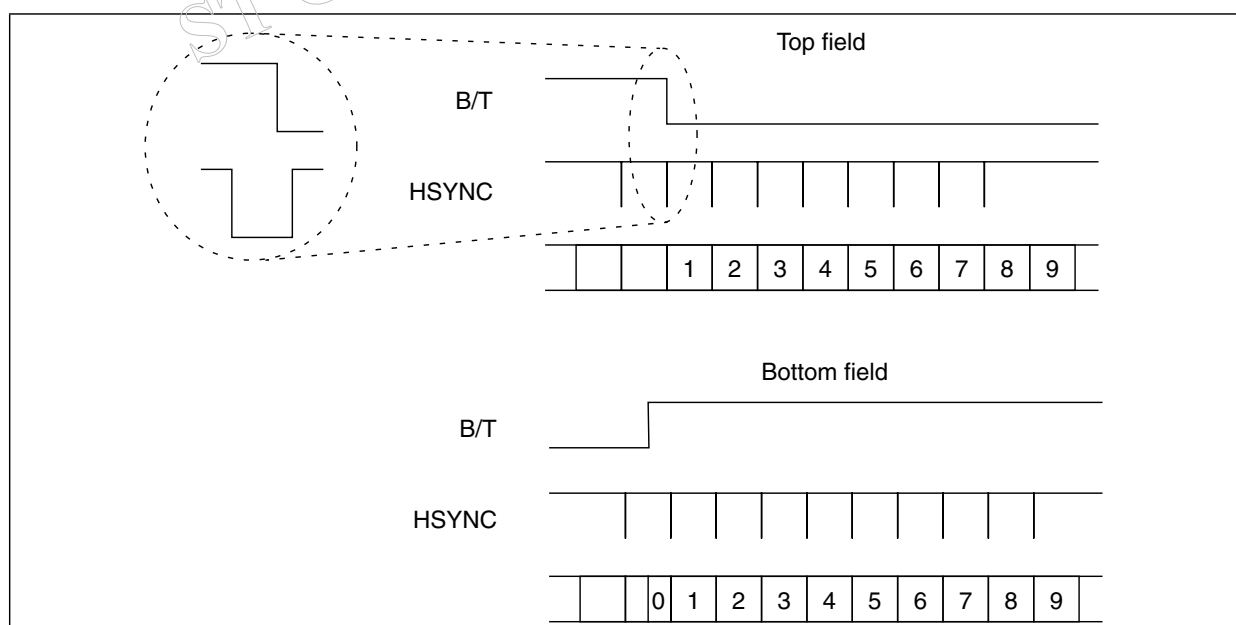
The OSD is enabled if register bit ENA in register OSD_CFG is set.

The OSD top field start address is defined by register OSD_TOP, and the bottom field by register OSD_BOT.

The line numbers, used to define the top and bottom of an OSD region, are the internal (field) line numbers illustrated in [Figure 116](#).

OSD specifications can be written into the SDRAM using the ST20 or the block move DMA. They can be rapidly moved within SDRAM using the SDRAM block move function.

Figure 116: Internal line numbering



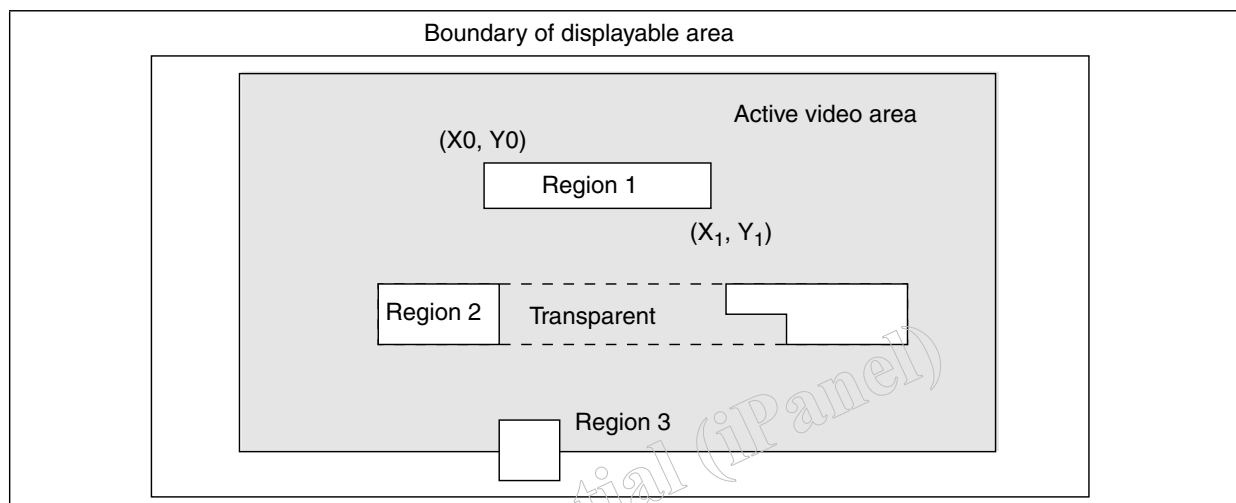
It is therefore possible to share the same OSD specification for both fields of a frame. In this case, OSD_TOP and OSD_BOT are loaded with the same address.

38.5.3 OSD regions

The OSD function can be used to display a user defined bit map over any part of the displayable (that is, nonblanked) screen, independent of the size and location of the active video area (defined by VID_XDO, VID_XDS, VID_YDO, VID_YDS). This bit map can be defined independently for each field.

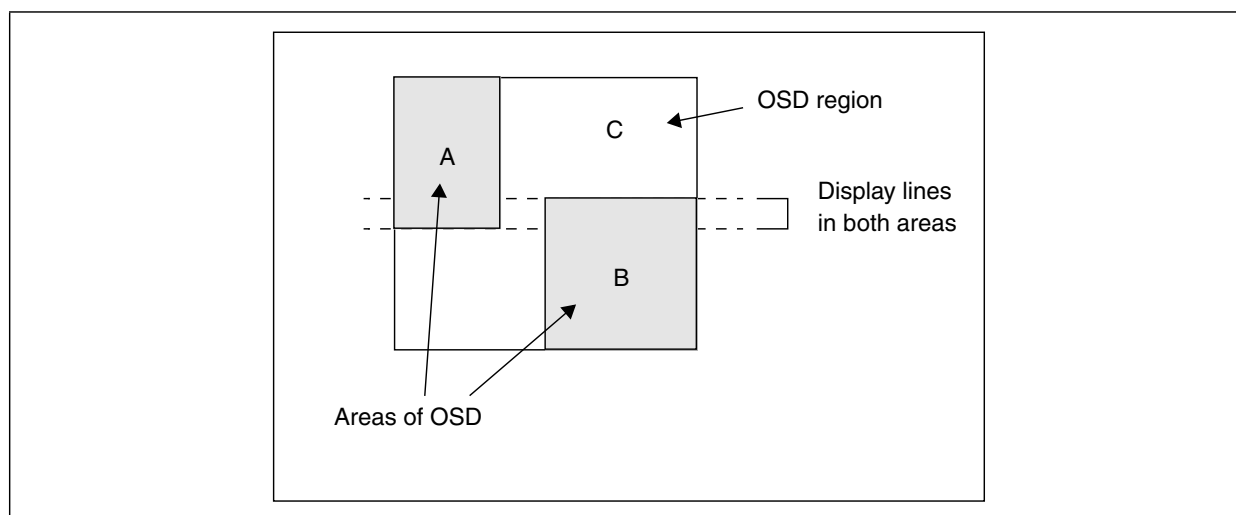
The OSD consists of one or more regions in the display. Each region is a rectangle, and can have its own palette and other properties. Figure 117 shows examples of OSD regions. Region 3 shows that the OSD can be outside the active video area.

Figure 117: OSD regions



No display line can be included in more than one active OSD region, so only one OSD region can be active on a line. If two areas of OSD are required which include the same display line then one region must be defined which includes both areas. For example, Figure 118 shows two areas, marked A and B, with some display lines used in both areas. If these areas are to be active at the same time then one region, marked C, must be defined, which includes both areas. The area of C outside A and B can be defined as transparent.

Figure 118: Two display areas using the same display lines

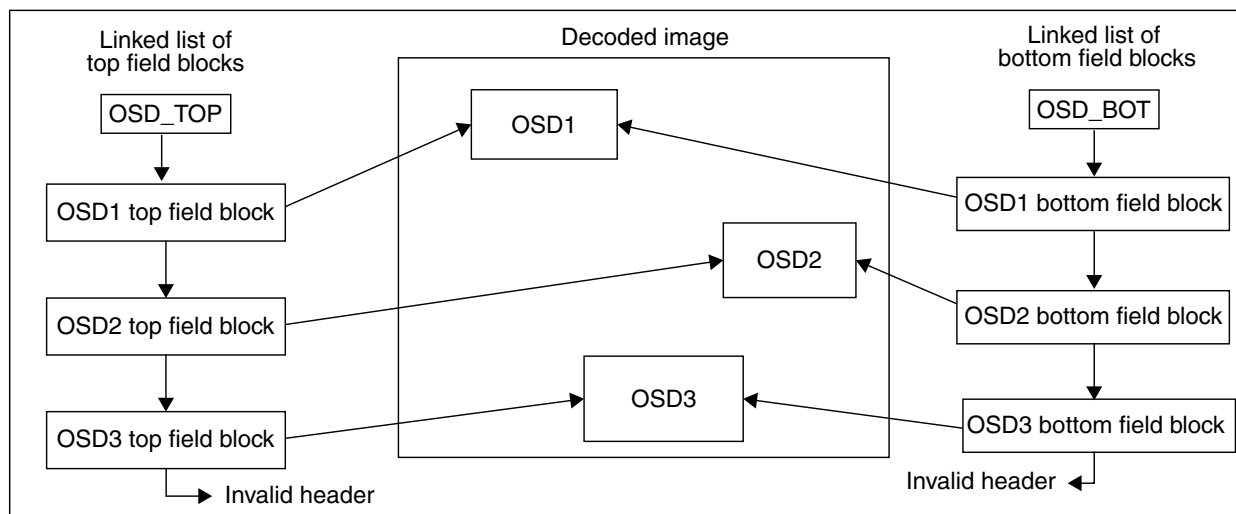


38.5.4 OSD specification

In normal mode, an OSD specification is two lists of blocks of 64-bit words, stored in SDRAM. One list is for the top field and one for the bottom. Generally the lists are linked lists, as shown in Figure 119. The order of the blocks in the list is the order of the regions from top to bottom of the

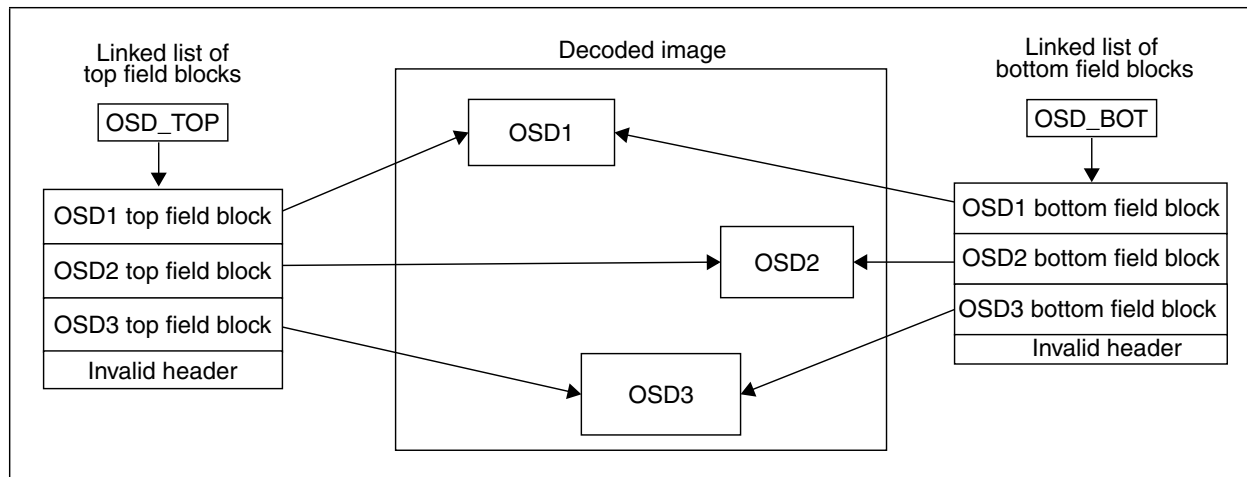
display. The last block in an OSD specification must point to an invalid header, which gives a starting line beyond the displayable area (for example 0xFFFF FFFF FFFF FFFF). This invalid header ends the OSD.

Figure 119: Linked list structure for OSD data



The only case in which linked lists are not used is when the palette mode is two bits for two pixels, that is, the palette mode flags are set to $M = 1$, $Q = 1$, $E = 0$, as described in [Section 38.5.6: Color palette on page 364](#). In this case the blocks must be contiguous in memory, so the start of each block must immediately follow in memory the end of the previous block, as shown in [Figure 120](#). No linked list is allowed after a two bits for two pixels zone.

Figure 120: Block structure for OSD data in two bits per two pixels mode



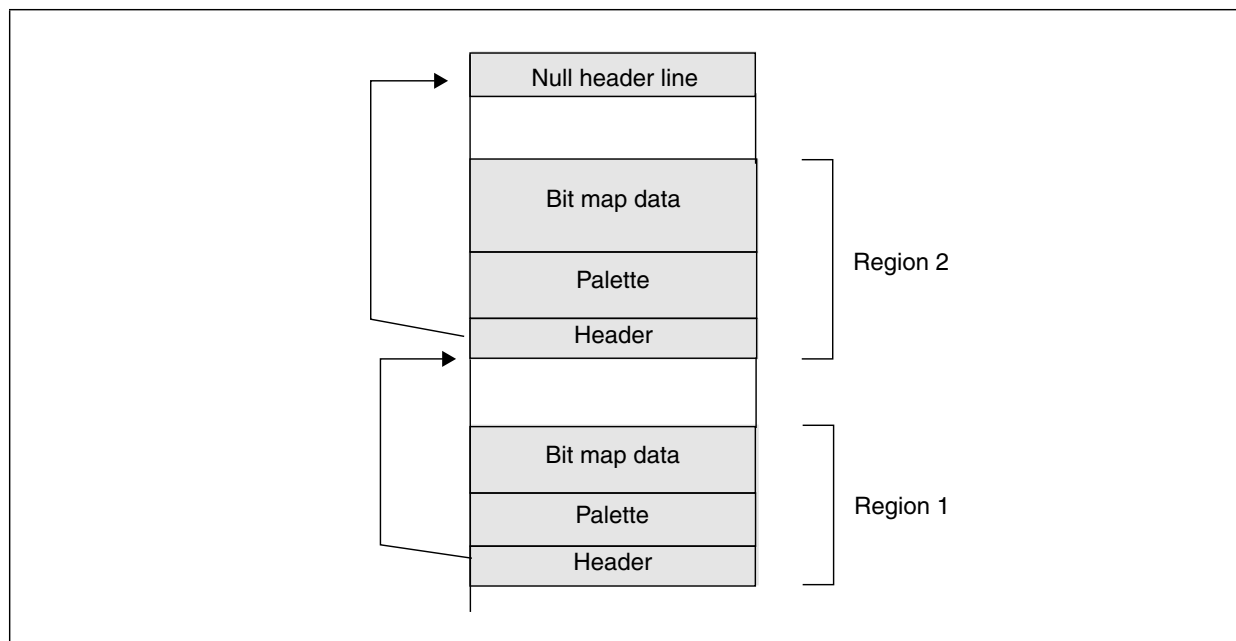
Each block defines one field of one region and includes a header, an optional palette and a bit map. Each block must be aligned on a 64-bit (one memory word) boundary and the first block of each field must be aligned on a 16-word boundary.

The bit map is stored such a way that we can access the OSD lines independently from each other: each OSD line starts at the beginning of a SDRAM 64-bits word and has a given length defined in the OSD header. This is obtained by stuffing the end of the line if required. In any cases only part of the last SDRAM word is stuffed, that is less than 64 bits.

To have each OSD line starting with the first pixel of a SDRAM word allows to point to all the lines or only on one in two or one in three and so on until one in seven.

Figure 121 shows a linked list of two OSD blocks in both modes.

Figure 121: OSD specification



Each region has associated with it a palette defining 4, 16 or 256 colors, used by the bit map. If required, one of these colors can be transparent, allowing the background to show through. Each region may have its own palette, or if a sequence of regions uses the same palette then the palette need only be defined in the first region of the sequence.

The header of each block contains a definition of the boundaries of the region, a pointer to the next region and other control information. The format of the palette depends on the palette mode, as described in [Section 38.5.6: Color palette on page 364](#). The formats are given in [Section 38.5.8: OSD block header format on page 368](#).

38.5.5 OSD region position

The position of each region of the OSD is defined in the header of the specification block. The positions of the left and right edge samples of an OSD region are defined as follows, in units of PIXCLK cycles from the falling edge of HSYNC:

- left edge position = $(2 \times X\_LEFT) + 8$,
- right edge position = $(2 \times X\_RIGHT) + 9$.

where X_LEFT and X_RIGHT are the values defined in the header of the OSD region specification. This is illustrated in [Figure 122](#) and [Figure 123](#). The first sample output in an OSD region is always a C_B value.

Figure 122: OSD region horizontal positioning in 4:4:4 output

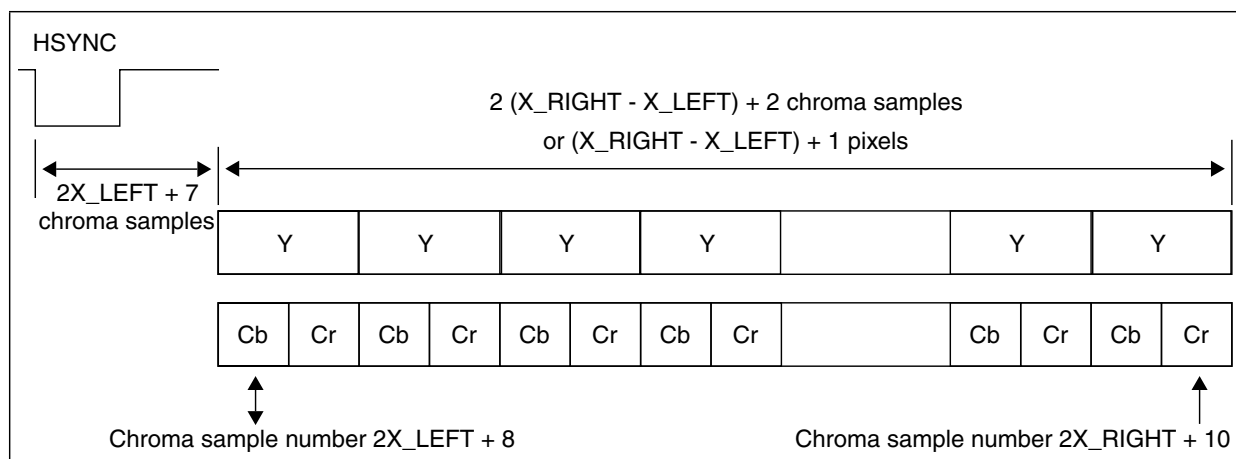
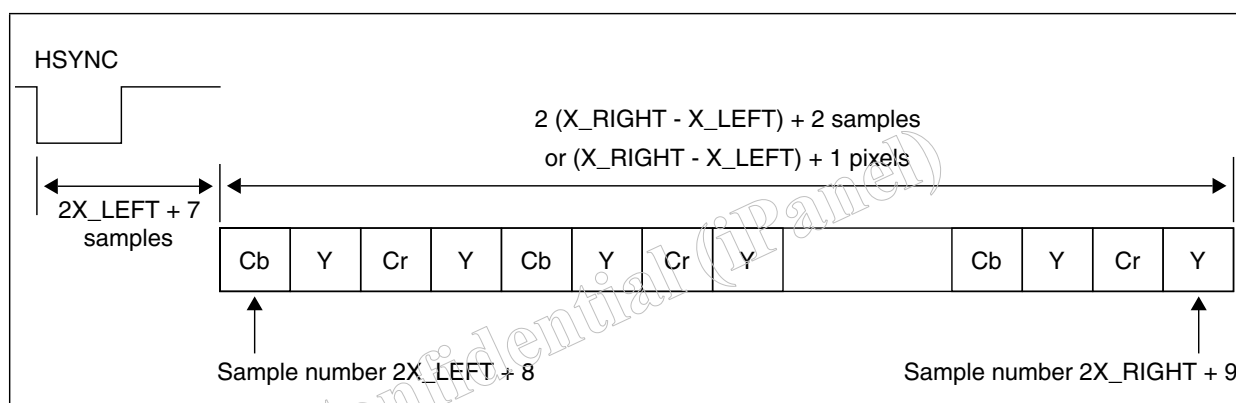


Figure 123: OSD region horizontal positioning in 4:2:2 output



The top and bottom of the regions are defined by the values Y_TOP and Y_BOTTOM , which are also in the block header. These values are specified in units of display lines. The top line specified in the first word of an OSD region specification must be greater than or equal to 3.

38.5.6 Color palette

Each specification block after the first can either define a new palette or use the same palette as the preceding region. If a new palette is defined then it is held in SDRAM immediately after the header and before the bit map. The P flag in the header defines whether the palette follows the header; when $P = 0$ the palette for the region is immediately after header, when $P = 1$ the palette is the same as for the preceding region.

Palette modes

The palette mode defines the bits per pixel in the bit map and the pixel resolution. The palette mode can be different for each OSD region, and is defined by the E, M, and Q flags in the OSD region specification header. Q defines the pixel resolution, allowing half resolution modes to save memory while retaining the color resolution. The meaning of each combination of these flags is given in [Table 141](#).

Table 141: M, Q and E palette mode header flags

E	M	Q	Resolution (pixels)	Bits per pixel	Number of colors
0	0	0	1	2	4
0	1	1	2		
0	1	0	1	4	16
0	0	1	2		
1	0	0	1	8	256
1	1	1	2		
1	1	0	Reserved		
1	0	1	Reserved		

To reduce the size of the bit maps while retaining the color resolution, a half-resolution mode is provided, as shown in [Table 141](#). In half resolution, each pixel in the bit map defines the color of two adjacent pixels in the same line in the display.

Color modes

The pixel color mode defines the format of the output from the palette. Three pixel color modes are supported as listed in [Table 142](#). This table shows how many bits are used for each color element and how many bits for the mixing factor MIXWEIGHT, which determines the effect of overlaying the picture. Anti-aliasing is supported only with 24-bit color.

24-bit or 14-bit color for the region field is selected by the bit TC in the header of the specification block. Anti-aliasing is selected by the bit AA in the header.

Table 142: OSD color modes

Mode	Color resolution	Mixing	Reference
24-bit color with anti-aliasing	Y[7:0], CB[7:0], CR[7:0]	MIXWEIGHT[5:0] defined for each color.	Table 143
24-bit color	Y[7:0], CB[7:0], CR[7:0]	MIXWEIGHT[3:0] defined for the region.	Table 144
14-bit color	Y[5:0], CB[3:0], CR[3:0]	MIXWEIGHT[3:0] defined for the region.	Table 145

The format of each line of the palette depends on the color mode. [Table 143](#) to [Table 145](#) show the format of the palette lines for each color mode.

Table 143: Palette line format in 24-bit color with anti-aliasing

Bit field	Bits	Description
CR[7:0]	[7:0]	Cr chroma value
CB[7:0]	[15:8]	Cb chroma value
Y[7:0]	[23:16]	Y luma value
W[5:0]	[29:24]	Mix weight. See Section 38.7: Mixing display planes on page 373 .
Reserved	[31:30]	Reserved. Write 0.

Table 144: Palette line format in 24-bit color without anti-aliasing

Bit field	Bits	Description
CR[7:0]	[7:0]	Cr chroma value
CB[7:0]	[15:8]	Cb chroma value
Y[7:0]	[23:16]	Y luma value
T	[24]	Transparency: 0: Do not blend video with OSD for this color. 1: Blend video with OSD for this color using the mix weight.
Reserved	[31:25]	Reserved. Write 0

Table 145: Palette line format in 14-bit color mode

Bit field	Bits	Description
CR[3:0]	[3:0]	Cr chroma value
CB[3:0]	[7:4]	Cb chroma value
T	[8]	Transparency: 0: Do not blend video with OSD for this color. 1: Blend video with OSD for this color using the mix weight.
Reserved	[9]	Reserved. Write 0.
Y[5:0]	[15:10]	Y luma value

Standard colors

Table 146 shows the 14-bit Y, CR and CB values nearest to the standard color bar colors.

Table 146: Standard colors in 14-bit color

Standard color	Y	CR	CB
White	0x3B	0x08	0x08
Black	0x04	0x08	0x08
Red	0x10	0x0D	0x06
Green	0x1C	0x04	0x04
Blue	0x09	0x07	0x0D
Yellow	0x28	0x09	0x03
Cyan	0x22	0x03	0x0A
Magenta	0x15	0x0C	0x0D

Table 147 shows the 24-bit Y, CR and CB values nearest to the standard color bar colors.

Table 147: Standard colors in 24-bit color

Standard color	Y	CR	CB
White	0xEC	0x80	0x80
Black	0x10	0x80	0x80
Red	0x40	0xD4	0x64
Green	0x70	0x40	0x48
Blue	0x24	0x74	0xD4
Yellow	0xA0	0x8C	0x2C
Cyan	0x88	0x2C	0x9C
Magenta	0x54	0xC8	0xB8

38.5.7 OSD bit map

The bit map for an OSD region follows the palette if defined, or the header if no palette is defined.

The bit map defines the OSD pixels, in left to right order within lines, and follows the lines in top to bottom order. The number of bits per pixel is 2, 4 or 8 depending on the palette mode. The value for each pixel gives the line of the palette, which defines the color for the pixel. As all the different sources are mixed in 4:4:4 format, there is no risk of a miscolored boundary between OSD, video, still picture and subpicture. A homogeneous decimation is applied to avoid the problem on the 4:2:2 format.

38.5.8 OSD block header format

The OSD specification block location is defined by the OSD_BOT or OSD_TOP registers, in units of 256 bytes. This means that the full address of the first block must be a multiple of 64.

Bit S in [Table 148](#) controls the presence of the OSD on a region basis for the 4:2:2 path to the digital encoder, from which YC and CVBS signals are generated. If this bit is 1, the OSD region is not present; if it is set to 0, the OSD region is present provided the OSD is included in the 4:2:2 output, as defined by OSD_CFG. This is to allow selective recording of OSD regions. This bit does not affect the 4:4:4 path to the digital encoder from which the RGB and YCbCr signals are generated.

[Table 148](#) shows the layout of the header, which occupies one 64-bit word. It shows the layout in graphical form, with each line representing a quarter of a 64-bit word. The header contains the OSDP[18:0] pointer. This pointer defines the address of the next block in the linked list to load from memory, as described in [Section 38.5.4: OSD specification on page 360](#). If OSDP[18:0] = 0, then the next OSD block is read contiguously from memory, after the current OSD block. This is STi3500 compatibility mode.

Table 148: OSD block header format

Field	Size	Bits	Description	Reference
M	1	[63]	Palette mode.	Table 141 .
Q	1	[62]		
E	1	[61]		
TC	1	[60]	1: Select 24-bit color mode. 0: Select 14-bit color mode.	Table 142
P	1	[59]	0: A new palette follows the header. 1: The palette is the same as the previous region.	
AA	1	[58]	Select anti-aliasing. Anti-aliasing can only be used with 24-bit color.	Section 38.5.6
S	1	[57]	OSD region not included in CVBS output in dual output mode.	Below
Y_TOP	9	[56:48]	Position of the top of the OSD region.	Section 38.5.5
MIXWEIGHT	4	[47:44]	Mixing weight a4 with planes behind when anti-aliasing is disabled. When anti-aliasing is enabled, write 0.	Chapter 39
OSDP[6:4]	3	[43:41]	Pointer to the next region specification.	Below
Y_BOTTOM	9	[40:32]	Position of the bottom of the OSD region.	Section 38.5.5
OSDP[12:7]	6	[31:26]	Pointer to the next region specification.	Below
X_LEFT	10	[25:16]	Position of the left of the OSD region.	Section 38.5.5
OSDP[18:13]	6	[15:10]	Pointer to the next region specification.	Below
X_RIGHT	10	[9:0]	Position of the right of the OSD region.	Section 38.5.5

M	Q	E	TC	P	AA	S	Y_TOP
MIXWEIGHT				OSDP[6:4]			Y_BOTTOM
OSDP[12:7]						X_LEFT	
OSDP[18:13]						X_RIGHT	

An OSD region can be placed anywhere in the whole 64-Mbit SDRAM area by setting register bit LLG in OSD_CFG.

- When LLG = 0, the OSD linked-list pointer can point to any word in the lower 32 Mbits of the 64 Mbit SDRAM area.
- When LLG = 1, the OSD linked-list pointer can point to any alternate word in all of the 64 Mbit SDRAM area. This effectively doubles the available area by halving the placement precision.

The block must be 16-word aligned, so the OSDP[18:0] pointer must be a multiple of 16. Therefore, the least significant 4 bits of OSDP are always zero, and are not included in the header

38.5.9 OSD specification block examples

This section shows the format for some complete specification blocks.

Normal mode

Table 149 shows a specification using 2 bits per pixel in the bit map with 1 pixel resolution and 14-bit color. Only the first 8 pixels of the bit map are shown. The palette occupies one 64-bit word, and the bit map occupies one 64-bit word for every 32 pixels.

Table 149: 2 bits per pixel, 14-bit color OSD region specification

Bits within a 16-bit quarter word															Description	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
M = 0	Q = 0	E = 0	Tc = 0	P = 0	A = 0	S	Y_TOP									Word 0
MIXWEIGHT				OSDP[6:4]			Y_BOTTOM									
OSDP[12:7]						X_LEFT										
OSDP[18:13]						X_RIGHT										
PALETTE0 Y						0	T0	PALETTE0 CB				PALETTE0 CR				
PALETTE1 Y						0	T1	PALETTE1 CB				PALETTE1 CR				Word 1
PALETTE2 Y						0	T2	PALETTE2 CB				PALETTE2 CR				
PALETTE3 Y						0	T3	PALETTE3 CB				PALETTE3 CR				
Bit map for 8 OSD pixels																

Table 150 shows a specification using 4 bits per pixel in the bit map with 2 pixel resolution and 14-bit color. Only the first 4 pixels of the bit map are shown. Each entry in the bit map uses 4 bits, but defines two display pixels of the same color.

The palette occupies four 64-bit words, and the bit map occupies one 64-bit word for every 16 bit map pixels.

Table 150: 4 bits per pixel, 2 pixel resolution 14-bit color OSD region specification

Bits within a 16-bit quarter word															Description	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
M = 0	Q = 1	E = 0	Tc = 0	P = 0	AA = 0	S	Y_TOP								Word 0	
MIXWEIGHT				OSDP[6:4]			Y_BOTTOM									
OSDP[12:7]						X_LEFT										
OSDP[18:13]						X_RIGHT										
PALETTE0 Y						0	T0	PALETTE0 CB				PALETTE0 CR			Word 1	
PALETTE1 Y						0	T1	PALETTE1 CB				PALETTE1 CR				
PALETTE2 Y						0	T2	PALETTE2 CB				PALETTE2 CR				
PALETTE3 Y						0	T3	PALETTE3 CB				PALETTE3 CR				
PALETTE4 Y						0	T4	PALETTE4 CB				PALETTE4 CR			Word 3	
PALETTE5 Y						0	T5	PALETTE5 CB				PALETTE5 CR				
PALETTE6 Y						0	T6	PALETTE6 CB				PALETTE6 CR				
PALETTE7 Y						0	T7	PALETTE7 CB				PALETTE7 CR				
PALETTE8 Y						0	T8	PALETTE8 CB				PALETTE8 CR			Word 4	
PALETTE9 Y						0	T9	PALETTE9 CB				PALETTE9 CR				
PALETTE10 Y						0	T10	PALETTE10 CB				PALETTE10 CR				
PALETTE11 Y						0	T11	PALETTE11 CB				PALETTE11 CR				
PALETTE12 Y						0	T12	PALETTE12 CB				PALETTE12 CR			Word 5	
PALETTE13 Y						0	T13	PALETTE13 CB				PALETTE13 CR				
PALETTE14 Y						0	T14	PALETTE14 CB				PALETTE14 CR				
PALETTE15 Y						0	T15	PALETTE15 CB				PALETTE15 CR				
Bit map for 4 OSD pixels															Bit map word	

Confidential

Table 151 shows a specification using 8 bits per pixel in the bit map with full resolution and 14-bit color. Only the first two pixels of the bit map are shown. Each pixel in the bit map uses 8 bits. The palette occupies 64 x 64-bit words (that is, 512 bytes), and the bit map occupies one 64-bit word for every eight bit map pixels.

Table 151: 8 bits per pixel, 14-bit color OSD region specification

Bits within a 16-bit quarter word																Description
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
M = 0	Q = 0	E = 1	Tc = 0	P = 0	AA = 0	S	Y_TOP									Word 0
MIXWEIGHT				OSDP[6:4]			Y_BOTTOM									
OSDP[12:7]						X_LEFT										
OSDP[18:13]						X_RIGHT										
PALETTE0 Y						0	T0	PALETTE0 CB				PALETTE0 CR			Word 1	
PALETTE1 Y						0	T1	PALETTE1 CB				PALETTE1 CR				
PALETTE2 Y						0	T2	PALETTE2 CB				PALETTE2 CR				
PALETTE3 Y						0	T3	PALETTE3 CB				PALETTE3 CR				
PALETTE4 Y						0	T4	PALETTE4 CB				PALETTE4 CR			Word 2	
...								...				...			...	
PALETTE252 Y						0	T12	PALETTE252 CB				PALETTE252 CR			Word 64	
PALETTE253 Y						0	T13	PALETTE253 CB				PALETTE253 CR				
PALETTE254 Y						0	T14	PALETTE254 CB				PALETTE254 CR				
PALETTE255 Y						0	T15	PALETTE255 CB				PALETTE255 CR				
Bit map for 2 OSD pixels with 8 bits per pixel or 8 bits per 2 pixel																Bit map word

Table 152 shows a specification using 2 bits per pixel in the bit map with full resolution and 24-bit color, without anti-aliasing. Only the first 8 pixels of the bit map are shown. Each entry in the bit map uses 2 bits. The palette occupies two 64-bit words, and the bit map occupies one 64-bit word for every 32 bit map pixels.

The palette for 4-bit and 8-bit colors would be similar, but with 16 or 256 color lines in the palette instead of four.

Table 152: 2 bits per pixel 24-bit color without anti-aliasing OSD region specification

Bits within a 16-bit quarter word															Description	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
M = 0	Q = 0	E = 0	Tc = 1	P = 0	AA = 0	S	Y_TOP								Word 0	
MIXWEIGHT				OSDP[6:4]			Y_BOTTOM									
OSDP[12:7]						X_LEFT										
OSDP[18:13]						X_RIGHT										
0 (Reserved)							T0	PALETTE0 Y								Word 1
PALETTE0 CB							PALETTE0 CR									
0 (Reserved)							T1	PALETTE1 Y								
PALETTE1 CB							PALETTE1 CR									
0 (Reserved)							T2	PALETTE2 Y								Word 2
PALETTE2 CB							PALETTE2 CR									
0 (Reserved)							T3	PALETTE3 Y								
PALETTE3 CB							PALETTE3 CR									
Bit map for 8 OSD pixels															Bit map word	

Table 153 shows a specification using two bits per pixel in the bit map with full resolution and 24-bit color with anti-aliasing. Only the first eight pixels of the bit map are shown. Each entry in the bit map uses two bits. The palette and bit map occupy the same memory as without anti-aliasing. The palette for 4-bit and 8-bit colors would be similar, but with 16 or 256 color lines in the palette instead of four.

Table 153: 2 bits per pixel 24-bit color with anti-aliasing OSD region specification

Bits within a 16-bit quarter word															Description
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
M = 0	Q = 0	E = 0	Tc = 1	P = 0	AA = 1	S	Y_TOP								Word 0
0 (Reserved)				OSDP[6:4]			Y_BOTTOM								
OSDP[12:7]						X_LEFT									
OSDP[18:13]						X_RIGHT									
0		MIXWEIGHT0						PALETTE0 Y							Word 1
PALETTE0 CB								PALETTE0 CR							
0		MIXWEIGHT1						PALETTE1 Y							
PALETTE1 CB								PALETTE1 CR							
0		MIXWEIGHT2						PALETTE2 Y							Word 2
PALETTE2 CB								PALETTE2 CR							
0		MIXWEIGHT3						PALETTE3 Y							
PALETTE3 CB								PALETTE3 CR							
Bit map for 8 OSD pixels															Bit map word

38.6 Subpicture or cursor plane

The subpicture or cursor plane displays the output from the subpicture decoder, described in [Chapter 36: Subpicture decoder on page 333](#). The subpicture can be either in front of or behind the OSD, depending on whether the bit SPO in register VID_OUT is 1 or 0 respectively.

The subpicture decoder can also be used as a hardware cursor unit. The subpicture should be configured to be in front of the OSD. A cursor can be defined using an optionally compressed (run length encoded) bit map stored in external SDRAM. The bit map can be any size up to a full screen. Per pixel alpha blending factors can be defined for each cursor to provide anti-aliasing with the background. The cursor is then moved around using register writes into X and Y coordinate registers.

38.7 Mixing display planes

The blending of the elements of the final picture is performed by a mixing unit, which is shown in [Figure 124](#) and [Figure 125 on page 375](#). The mixing of the display planes is controlled by five mix weights, α_1 to α_5 . The mix weights α_1 to α_3 values control the mixing of the back three planes, the background color, the still picture plane and the video plane. These are 8-bit values

held in registers, as shown in Table 154. The three back planes are generated in 4:2:2 format and are mixed in that format and then converted to 4:4:4 for mixing with the subpicture and OSD.

Table 154: Control of mixing factors α_1 to α_3

Factor	Register	Mixed planes	Plane displayed when mix weight is 0x00	Plane displayed when mix weight is 0xFF
α_1	VID_MWS	The background color and the still picture plane.	Still picture plane	Background color
α_2	VID_MWV	The background color and MPEG video.	MPEG video	Background color
α_3	VID_MWSV	The video and the still picture plane.	MPEG video	Still picture plane

The result of mixing the back three planes can be blended in turn with the subpicture and OSD. The mixing formula is: $\text{display} = (1 - a) \times \text{source1} + a \times \text{source2}$.

The order of this mixing depends on whether the subpicture is in front of or behind the OSD. The OSD mix weight is α_4 and is defined in the OSD palette. If anti-aliasing is disabled, then the mix weight is a 4-bit value, defined for each OSD region, and mixing can be enabled or disabled for each color. If anti-aliasing is enabled, then the mix weight is a 6-bit value defined for each color.

The subpicture mix weight is α_5 , which is a 4-bit value defined per pixel type. These values can be controlled by the subpicture bit stream except in the highlight area which is controlled by SPD_HCN.

Figure 124: Mixing with the subpicture in front

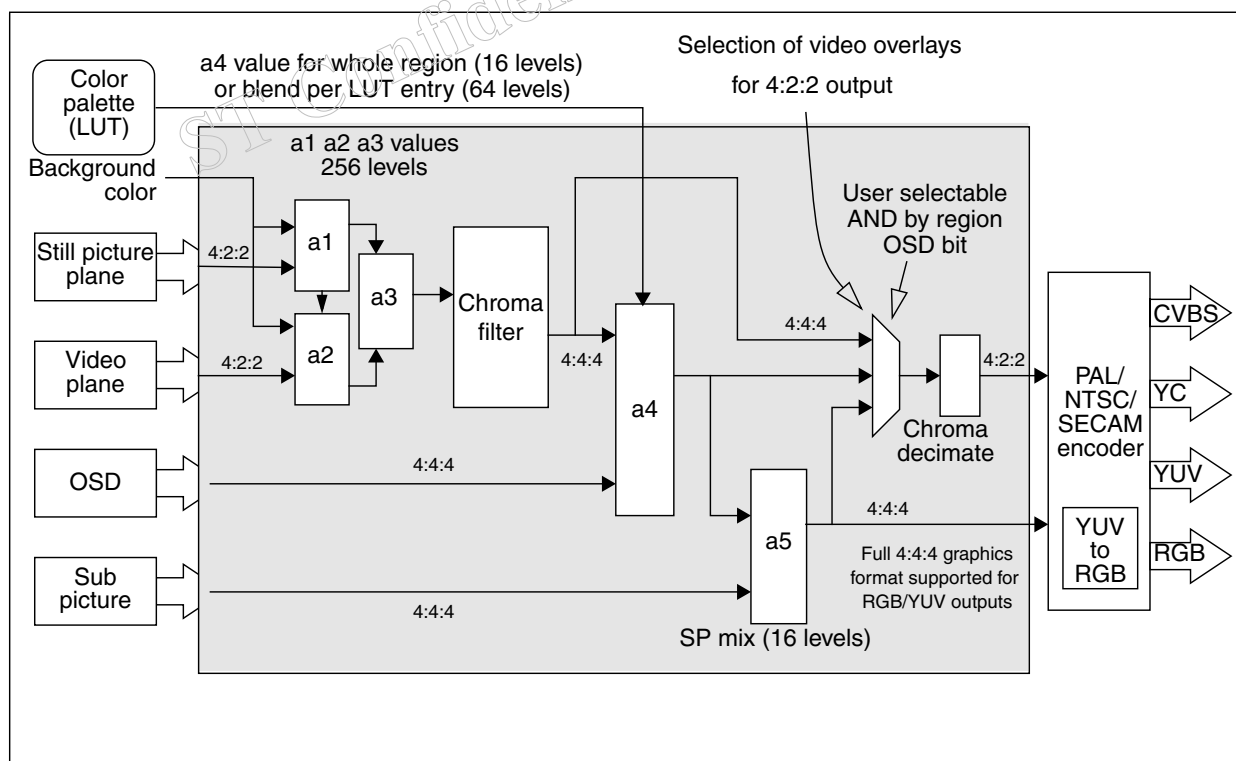
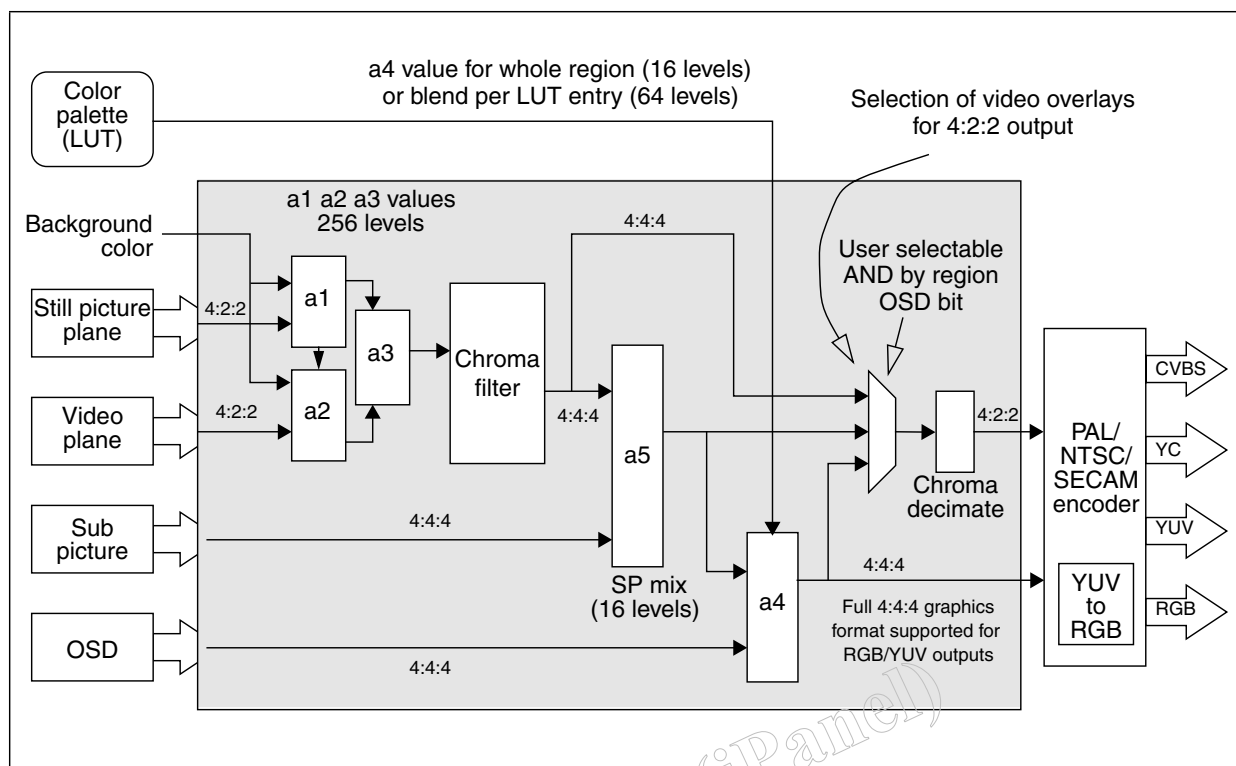


Figure 125: Mixing with the OSD in front



38.7.1 4:2:2 output control

The 4:2:2 output is used by the digital encoder to generate CVBS and YC output, and is generally used for recording by VCR. The OSD and subpicture can be omitted from this output, as controlled by the registers VID_OUT and OSD_CFG and by the S bits in the OSD region headers. The combined actions of these fields depend on whether the OSD is behind or in front of the subpicture. The 4:4:4 output is unaffected.

Bit LAY in register VID_OUT defines the number of planes in front of the video which are initially included. Bit REC_OSD in register OSD_CFG turns on or off the OSD, as shown in Table 155. If bit REC_OSD of register OSD_CFG = 1 (OSD present) then the S bit in an OSD region header can alternatively be used to turn off that OSD region. When the OSD and subpicture planes are present, bit SPO of register VID_OUT defines which plane is in front. The combination of bit functions is given in Table 155.

Table 155: Encoding of LAY, SPO (VID_OUT) and REC_OSD (OSD_CFG)

LAY bit	REC_OSD bit	Subpicture in front (SP0 = 1)	OSD in front (SP0 = 0)
00 to 01	Any	Video only	Video only
10	1	Video + OSD	Video + subpicture
	0	Video only	Video + subpicture
11	1	Video + OSD + subpicture	Video + subpicture + OSD
	0	Video + subpicture	Video + subpicture

39 Display planes registers

Addresses are provided as the *VideoBaseAddress* + offset.

The *VideoBaseAddress* is:

0x0000 0000.

A register summary is given in [Table 36: Display planes registers on page 61](#).

39.1 Background color registers

BCK_Y

Background color Y

7 6 5 4 3 2 1 0

BCK_Y[7:0]

Address: *VideoBaseAddress* + 0x98

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register stores the value of the Y component of the background color mixable with the video or the still picture plane.

BCK_U

Background color U

7 6 5 4 3 2 1 0

BCK_U[7:0]

Address: *VideoBaseAddress* + 0x99

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register stores the value of the U component of the background color mixable with the video or the still picture plane.

BCK_V

Background color V

7 6 5 4 3 2 1 0

BCK_V[7:0]

Address: *VideoBaseAddress* + 0x9A

Type: Read/write

Reset: FF

Description: VSYNC synchronization

This register stores the value of the V component of the background color mixable with the video or the still picture plane.

Confidential

39.2 Still picture plane registers

TDL_CSO

TDL chrominance offset

7 6 5 4 3 2 1 0

CSO[7:0]

Address: *VideoBaseAddress* + 0xEC

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a value calculated from the fractional part of the TDL horizontal pan vector, integer part being in TDL_SCN HSCN vector. If no pan vector is defined, this register can be left in its reset (default) state.

TDL_DCF

TDL configuration

7 6 5 4 3 2 1 0

Reserved

TRANSP

FORMAT

UND

UDS

DSR

ETDL

Address: *VideoBaseAddress* + 0xF4

Type: Read/write

Reset: 0

Description: VSYNC synchronization

[7:4] **Reserved**

[5] **TRANSP**

This bit sets the TDL transparency mode. It can be set only if TDL_DCF[1] is also set.

0: Basic TDL

1: TDL transparent mode. TDL pixels with luma and chroma set to 0x00 are considered transparent.

[4] **FORMAT**

The still picture is stored in 4:2:2 raster scan format, the format order is set by this bit:

0: Format order is: Cb Y Cr Y Cb Y Cr Y

1: Format order is: Y Y Y Y Cb Cb Cr Cr

[3] **UND: Up not down**

This bit is valid only if UDS is set. When UND = 1 then a vertical size of the still picture plane is multiply by 2 by repeating each line. When UND is reset then the vertical size of the still picture plane is divide by 2 by line dropping.

[2] **UDS: Up/down scaling**

When this bit is set, enable the vertical resizing of the still picture plane.

[1] **DSR: Disable SRC**

When this bit is set, both luminance and chrominance SRCs (sample rate converters) are disabled.

[0] **ETDL: TDL enable video display**

When this bit is reset, the still picture plane has a constant value of Y = 16, CB = CR = 128. In this case no fading is possible between still picture plane and video picture.

Confidential

TDL_LSO**TDL luminance offset**

7 6 5 4 3 2 1 0

LSO[7:0]

Address: *VideoBaseAddress* + 0xEA

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a value calculated from the fractional part of the TDL horizontal pan vector, the integer part being in TDL_SCN HSCN vector. If no pan vector is defined, this register can be left in its reset (default) state.

TDL_LSR**SRC luma/chroma resolution**

7 6 5 4 3 2 1 0

0xEB

TDL_LSR[7:0]

0xED

Reserved

TDL_LSR[8]

Address: *VideoBaseAddress* + 0xEB, 0xED

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the upsampling factor of the luminance SRC (sample rate converter). It applies on chroma also. The upsampling factor is equal to 256/LSR. Below are given some examples of upsampling factors, where in each case the displayed picture has a nominal width of 720 pels¹. Also shown are the numbers of valid pels generated, n.

Table 156: Upsampling factors

Decoded picture width	LSR	n
640	228	715
640	227	718
544	193	717
544	192	721
480	170	717
480	169	722
352	125	713
352	124	719
704	250	717
704	249	720

Note: Downsampling is not supported

1. Displayed picture widths other than 720 can of course be supported.

TDL_SCN**Still picture plane scan vector**

	7	6	5	4	3	2	1	0
0xB0	Reserved	TDL_HSCN[5:0]						TDL_SCN[8]
0xB1	TDL_SCN[7:0]							

Address: *VideoBaseAddress* + 0xB0 to 0xB1

Type: Read/write

Reset: 0

Description: VSYNC synchronization

0xB0: [7] **Reserved**

0xB0: [6:1] **TDL_HSCN**: Location of the first displayed pixel of the still picture plane in the decoded picture.
If TDL_HSCN = 11, then the 11 left pixels of each field of the still picture plane are not displayed.

0xB0: [0] **TDL_SCN**: Location of the first displayed line of the still picture plane in the decoded picture.

0xB1: [7:0] If TDL_SCN = 11, then the 11 top lines of each field of the still picture plane are not displayed.

TDL_SWT**Switch line number**

	7	6	5	4	3	2	1	0
0xC8	Reserved						TDL_SWT[9:8]	
0xC9	TDL_SWT[7:0]							

Address: *VideoBaseAddress* + 0xC8 to 0xC9

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the value of the line on which we switch from TDL_TEP to TDL_TEP2 or TDL_TOP to TDL_TOP2. This is the unrolling function.

TDL_TDW**Third display width**

	7	6	5	4	3	2	1	0
Reserved	TDL_TDW[6:0]							

Address: *VideoBaseAddress* + 0x86

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a value defined in units of 16 pixels.

TDL_TEP**TDL even pointer**

	7	6	5	4	3	2	1	0
0xE6	Reserved					TDL_TEP[18:16]		
0xE7	TDL_TEP[15:8]							
0xE8	TDL_TEP[7:0]							

Address: *VideoBaseAddress* + 0xE6 to 0xE8

Type: Read/write

Reset: Undefined

This register holds the source address of the even field of the third display layer. TDL_TEP[18:0] is defined in units of 128 bytes (64 pixels). Therefore, the two LSBs must be programmed to zero.

TDL_TEP2**TDL even pointer 2**

	7	6	5	4	3	2	1	0
0x93	Reserved					TEP2[18:16]		
0x94	TDL_TEP2[15:8]							
0x95	TDL_TEP2[7:0]							

Address: *VideoBaseAddress* + 0x93 to 0x95

Type: Read/write

Reset: Undefined

This register holds the second source address of the even field of the picture plane. See unrolling feature for the right usage of TDL_TEP and TDL_TEP2. TDL_TEP2[18:0] is defined in units of 128 bytes (64 pixels). Therefore, the two LSBs must be programmed to zero.

TDL_TOP**TDL odd pointer**

	7	6	5	4	3	2	1	0
0xE3	Reserved					TDL_TOP[18:16]		
0xE4	TDL_TOP[15:8]							
0xE5	TDL_TOP[7:0]							

Address: *VideoBaseAddress* + 0xE3 to 0xE5

Type: Read/write

Reset: Undefined

This register holds the source address of the odd field of the third display layer. TDL_TOP[18:0] is defined in units of 128 bytes (64 pixels). Therefore, the two LSBs must be programmed to zero.

Confidential

TDL_TOP2**TDL odd pointer**

	7	6	5	4	3	2	1	0
0xB2	Reserved						TOP2[18:16]	
0xB3	TDL_TOP2[15:8]							
0xB4	TDL_TOP2[7:0]							

Address: *VideoBaseAddress* + 0xB2 to 0xB4

Type: Read/write

Reset: Undefined

This register holds the second source address of the odd field of the picture plane. See unrolling feature for the right usage of TDL_TOP and TDL_TOP2. TDL_TOP2[18:0] is defined in units of 128 bytes (64 pixels). Therefore, the two LSBs must be programmed to zero.

TDL_XDO**TDL X offset**

	7	6	5	4	3	2	1	0
0xF0	Reserved						TDL_XDO[9:8]	
0xF1	TDL_XDO[7:0]							

Address: *VideoBaseAddress* + 0xF0 to 0xF1

Type: Read/write

Reset: 0

Description: VSYNC synchronization
TDL_XDO[9:0] is set up with a number defining the beginning of the left border of the display. This offset is measured from the active (first) edge of HSYNC and is specified in units of PIXCLK cycles. The horizontal offset, XDO', is given by: $XDO' = TDL\_XDO + 11$

TDL_XDS**TDL X end**

	7	6	5	4	3	2	1	0
0xF2	Reserved						TDL_XDS[9:8]	
0xF3	TDL_XDS[7:0]							

Address: *VideoBaseAddress* + 0xF2 to 0xF3

Type: Read/write

Reset: 0

Description: VSYNC synchronization
This register is set up with a number defining the right boundary of the picture display window, expressed in units of PIXCLK cycles. The value of XDS[9:0] must be equal to TDL_XDO[9:0] plus the width of the display window. The actual offset, XDS', is given by: $XDS' = TDL\_XDS + 11$.

TDL_YDO**TDL Y offset**

	7	6	5	4	3	2	1	0
0xEE	Reserved						TDL_YDO[9:8]	
0xEF	TDL_YDO[7:0]							

Address: *VideoBaseAddress* + 0xEE to 0xEF

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a number defining the first line of the picture display, that is, the top edge of the display window. Lines are counted from the active (first) edge of VSYNC. In an interlaced display, the same value of TDL_YDO would be used for both fields.

TDL_YDS**TDL Y end**

	7	6	5	4	3	2	1	0
0xC6	Reserved						TDL_YDS[9:8]	
0xC7	TDL_YDS[7:0]							

Address: *VideoBaseAddress* + 0xC6 to 0xC7

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a number defining the bottom line of the display window counted in one field. In an interlaced display, the same value of TDL_YDS would be used for both fields.

39.3 On-screen display registers**OSD_ACT****Active signal**

	7	6	5	4	3	2	1	0
Reserved	OAM	OAD[5:0]						

Address: *VideoBaseAddress* + 0x3E

Type: Read/write

Reset: 0

Description:

[7] **Reserved**[6] **OAM**

OSD active signal mode

When this bit is set, OSD active signal is an input. When it is reset, OSD active signal is an output. Note that OSD active signal must never be driven when bit EVI of register VID_CTL = 1 and bit OAM of register VID_OSD = 0.

[5:0] **OAD**

OSD active signal delay

These bits are used to define the delay of the OSD active signal corresponding to output OSD pixels.

OSD_BDW**OSD boundary weight**

7	6	5	4	3	2	1	0
Reserved		OSD_BDW[5:0]					

Address: *VideoBaseAddress* + 0x92

Type: Read/write

Reset: 0

These bits represent the value of MIX_WEIGHT at the horizontal border of an OSD region or at a horizontal border of a transparent region within an OSD region. The OSD_BDW value is used in OSD filter formulae and has therefore no meaning when the OSD is in normal mode.

OSD_BOT**OSD bottom field pointer**

7	6	5	4	3	2	1	0
1st cycle	OBP[15:8]						
2nd cycle	OBP[7:0]						

Address: *VideoBaseAddress* + 0x2B

Type: Serial read/write

Reset: 0

Description: VSYNC bottom synchronization

This register is written or read in two cycles:

- first cycle: OBP[15:8],
- second cycle: OBP[7:0].

The circularity is reset by a hardware reset or a bottom field VSYNC. The register holds the start address, in units of 256 bytes of the current OSD specification buffer for the bottom field. This specification is decoded during bottom fields when OSD is enabled.

OSD_CFG**OSD configuration**

7	6	5	4	3	2	1	0
Reserved	REC_OSD	OSD_POL	LLG	Reserved	ENA	NOR	FIL

Address: *VideoBaseAddress* + 0x91

Type: RW

Reset: 0

Description:

- [7] **Reserved**
- [6] **REC_OSD**
When set, include the OSD in the 4:2:2 output.
- [5] **OSD_POL**
When set, this bit inverts the internal ENOT0 signal used by the OSD.
- [4] **LLG**
Linked list granularity:
0: OSD linked-list pointer can point to any word in the lower 32 Mbits of the 64-Mbit SDRAM area,
1: OSD linked-list pointer can point to any alternate word in all of the 64-Mbit SDRAM area.
This effectively doubles the available area by halving the placement precision.
- [3] **Reserved**
- [2] **ENA**
OSD enable. 0: OSD disabled, 1: OSD enabled
- [1] **NOR**
OSD normal mode:
0: OSD normal mode active, filter modes specified in bit FIL of register OSD_CFG are not active
1: Filter mode specified in bit FIL of register FIL (OSD_CFG.) is active
- [0] **FIL**
OSD filter mode: to activate one of the two filter modes bit NOR must be set.
0: Antiflicker mode active, 1: Antiflutter mode active

OSD_TOP**OSD top field pointer**

	7	6	5	4	3	2	1	0
1st cycle	OTP[15:8]							
2nd cycle	OTP[7:0]							

Address: *VideoBaseAddress* + 0x2A

Type: Serial read/write

Reset: 0

Description: VSYNC top synchronization

This register is written or read in two cycles:

- first cycle: OTP[15:8],
- second cycle: OTP[7:0].

The circularity is reset by a hardware reset or a top VSYNC. The register holds the start address, in units of 256 bytes, of the current OSD specification buffer for the top field. This specification is decoded during top fields when OSD is enabled.

VID_656**Enable 656 mode**

7	6	5	4	3	2	1	0
Reserved	POU	POY	ADDYDS	ADDYDO	POE	E6M	

Address: *VideoBaseAddress* + 0x32

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

[7:6] **Reserved**

[5] **POU**

If set to 1 changes the polarity of UNOTV

[4] **POY**

If set to 1 changes the polarity of YNOTC

[3] **ADDYDS**

If set to 1 add 1 to YDS_656 value (YDS_656 + 1)

[2] **ADDYDO**

If set to 1 add 1 to YDO_656 value (YDO_656 + 1)

[1] **POE**

If set to 1 changes the polarity of ENOTO to generate synchro pattern F

[0] **E6M**

Enable CCIR 656 mode

0: Basic digital video output is sent to port YC[7:0] (CCIR601)

1: Synchronous patterns are inserted in video stream according to CCIR 656 recommendations

VID_656B**Control synchronization patterns**

7	6	5	4	3	2	1	0
Reserved				FFZ		SF[2:0]	

Address: *VideoBaseAddress* + 0x45

Type: Read/write

Reset: 0

Description: Edge triggered synchronization

[7:4] **Reserved**

[3] **FFZ**

1: Force synchro pattern F to zero (used for progressive mode)

[2:0] **SF[2:0]**

These three register bits allow the F synchro pattern to be generated 0 to 7 lines after the field active area

000: 0 lines after the field active area... up to 111: 7 lines after the field active area

VID_CS0**SRC chrominance offset**

7 6 5 4 3 2 1 0

CSO[7:0]

Address: *VideoBaseAddress* + 0x6C

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a value calculated from the fractional part of the pan vector. If no pan vector is defined, this register can be left in its reset (default) state. The method of calculation of the CSO value is given in the VID_PAN register description.

VID_DCF**Display configuration**

7 6 5 4 3 2 1 0

0x74	Reserved		BLL	BFL	FNF	Reserved		FRZ
0x75	FPO	BCK_ACT	EVD	Reserved	DSR	B2R_POL	Reserved	

Address: *VideoBaseAddress* + 0x74 and 0x75

Type: Read/write

Reset: 0

Description: VSYNC synchronization

0x74: [7:6] **Reserved**

0x74: [5] **BLL**: Blank last line

If this bit is set, the last active line of a picture is blanked. (Used in letter box format display).

0x74: [4] **BFL**: Blank first line.

If this bit is set the first active line of a picture is blanked. (Used in letter box format display).

0x74: [3] **FNF**: Frame not field

Always set to 1.

0x74: [2:1] **Reserved**

0x74: [0] **FRZ**: Freeze the display

When bit 0 is set, the current decoded field is displayed continuously on both fields until this bit is reset by the user. This bit is also used to control three-to-two pull-down operation. Note that the actual effect of this bit is to simply freeze the current polarity of the internal B/NOTT signal.

0x75: [7] **FPO**: Field polarity

When the display is frozen (FRZ = 1), this bit chooses whether to display the top or bottom field.

0x75: [6] **BCK_ACT**

1: The background color fills the screen outside the video and TDL active area.

0: Blanking outside the video and TDL active area

0x75: [5] **EVD**: Enable video display

When this bit is reset, the video output has a constant value of Y = 16, CB = CR = 128. OSD is still displayed.

0x75: [4] **Reserved**

0x75: [3] **DSR**: Disable SRC

When this bit is set, both luminance and chrominance SRCs (sample rate converters) are disabled. In this case no horizontal filtering can occur, as would be required when the horizontal resolution of the decoded picture is equal to the horizontal resolution of the display.

0x75: [2] **B2R_POL**: Active video polarity (field parity).

0x75: [1:0] **Reserved**

VID_DFC**Displayed chroma frame pointer**

	7	6	5	4	3	2	1	0
0x58	VID_DFC[15:8]							
0x59	VID_DFC[7:0]							

Address: *VideoBaseAddress* + 0x58 and 0x59

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the start address, defined in units of 256 bytes, of the chroma frame which is currently being displayed. When a new value is written this is used at the start of the next field. When VID_DFC is set to same value as VID_RFC (that is, the decoder is writing the reconstructed picture into the buffer which is being displayed), bit OVW of register VID_TIS must be set.

VID_DFP**Displayed luma frame pointer**

	7	6	5	4	3	2	1	0
0x0C	VID_DFP[15:8]							
0x0D	VID_DFP[7:0]							

Address: *VideoBaseAddress* + 0x0C and 0x0D

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the start address, defined in units of 256 bytes, of the luma frame which is currently being displayed. When a new value is written this is used at the start of the next field.

When VID_DFP is set to the same value as VID_RFP (that is, the decoder is writing the reconstructed picture into the buffer which is being displayed), bit OVW of register VID_TIS must be set.

VID_DIS

Video configuration

7	6	5	4	3	2	1	0
Reserved	CPU_PRIORITY	DISPLAY_PRIORITY	ZOOMOUTX2	FBS			

Address: *VideoBaseAddress* + 0xD6

Type: Read/write

Reset: 0

Description:

[7] **Reserved**

[6:3] **CPU_PRIORITY and DISPLAY_PRIORITY**

These bits set the priority for the memory accesses of the CPU, still picture plane and MPEG video planes. The register can be set according to the application.

In the default (5510 compatible) configuration, the CPU takes high priority and the still picture and video take lower priority. In all other configurations, still picture-plane memory accesses take high priority.

'x' indicates that the bit value is not important.

CPU_PRIORITY	DISPLAY_PRIORITY	Still picture plane priority	MPEG video plane priority	CPU priority
xx	00 (default)	Medium	Medium	High
1x	01	High	Low	High
01	01	High	Low	Medium
00	01	High	Low	Low
1x	10	High	Medium	High
01	10	High	Medium	Medium
00	10	High	Medium	Low
1x	11	High	High	High
01	11	High	High	Medium

[2] **ZOOMOUTX2**

When set, this bit can be used with a block-to-row filter (set by registers VID_VFLMODE and VID_VFCMODE) to increase the zoom-out by X2, without increasing the bandwidth. This bit sets-up a line drop, where alternate lines are read into the memory.

[1:0] **FBS**

Selects frame-based or field-based vertical filtering. These bits are used in conjunction with registers VID_VFCMODE and VID_VFLMODE to set the vertical filter mode.

Bit field	Luma	Chroma	Type
00:	Field	Field	Field-based filtering
01:	Field	Frame	Luma field-based, chroma frame-based filtering
10:	Frame	Field	Luma frame-based, chroma field-based filtering
11:	Frame	Frame	Frame-based filtering

VID_LSO**SRC luminance offset**

7 6 5 4 3 2 1 0

LSO[7:0]

Address: *VideoBaseAddress* + 0x6A

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a value calculated from the fractional part of the pan vector. If no pan vector is defined, this register can be left in its reset (default) state. The method of calculation of the LSO value is given in the VID_PAN register description.

VID_LSR**SRC luma/chroma resolution**

7 6 5 4 3 2 1 0

0x6B

LSR[7:0]

0x6D

Reserved

LSR[8]

Address: *VideoBaseAddress* + 0x6B and 0x6D

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the upsampling factor of the luminance SRC (sample rate converter). It applies on chroma also.

The upsampling factor is equal to 256/LSR. [Table 157](#) gives some examples of upsampling factors, where in each case the displayed picture has a nominal width of 720 pels. Also shown are the numbers of valid pels generated, n, calculated as shown in [Section 38.4.2: MPEG video sample rate converter \(horizontal filtering\)](#) on page 349. Displayed picture widths other than 720 are supported.

Table 157: Example upsampling factors

Decoded picture width	LSR	n
640	228	715
640	227	718
544	193	717
544	192	721
480	170	717
480	169	722
352	125	713
352	124	719
704	250	717
704	249	720

VID_MWS**Mix weight still picture**

7 6 5 4 3 2 1 0

MWS[7:0]

Address: *VideoBaseAddress* + 0x9C

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the value of the mix between the background color and the still picture plane. When VID_MWS = 0xFF then the background color is being displayed and at 0 the still picture plane is being displayed.

VID_MWSV**Mix weight still/video**

7 6 5 4 3 2 1 0

MWSV[7:0]

Address: *VideoBaseAddress* + 0x9D

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register holds the value of the mix between the video and the still picture plane. When VID_MWSV = 0xFF then the still picture plane is being displayed and at 0 the video is being displayed.

VID_MWV**Mix weight video**

7 6 5 4 3 2 1 0

MWV[7:0]

Address: *VideoBaseAddress* + 0x9B

Type: Read/write

Reset: 0

Description: VSYNC

This register holds the value of the mix between the background color and the video. When VID_MWV = 0xFF then the background color is being displayed and at 0 the video is being displayed.

Confidential

VID_OUT**Output of 4:2:2 display**

7	6	5	4	3	2	1	0
Reserved					SPO	LAY	

Address: *VideoBaseAddress* + 0x90

Type: Read/write

Reset: 1100₂

Description: VSYNC synchronization

This register controls the content of the 4:2:2 output from the display mixing unit and the order of the OSD and subpicture planes.

[7:3] **Reserved**

[2] **SPO**: Place the subpicture plane in front of the OSD plane.

[1:0] **LAY**: The number of layers to be displayed in front of the video in the 4:2:2 output.

VID_PAN**Pan/scan horizontal vector integer part**

	7	6	5	4	3	2	1	0
0x2C	Reserved					PAN[10:8]		
0x2D	PAN[7:0]							

Address: *VideoBaseAddress* + 0x2C and 0x2D

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with the integer part of the horizontal pan/scan vector. The horizontal pan/scan vector defines, in the decoded picture, the location of the first displayed luminance sample relative to the first luminance sample in the line. The VID_LSO and VID_CSO registers are set up with the fractional part of the horizontal pan/scan vector, as follows:

- PSV: $\lfloor \text{horizontal pan / scan vector} \rfloor$, where $\lfloor x \rfloor$ indicates the integer part of x .
- VID_LSO: $256 \times (\text{horizontal pan / scan vector} - \text{PSV})$
- VID_CSO: VID_LSO / 2 (+ 1 if PSV is odd)

VID_SCN**Pan/scan vertical vector**

7	6	5	4	3	2	1	0
Reserved		SCN[8:3]					

Address: *VideoBaseAddress* + 0x87

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register defines the position of the top left corner of the displayed picture with respect to the decoded picture. The position is defined by this register in macroblocks, that is two fields of eight lines.

Confidential

VID_VFCMODE

Configure chrominance of block row

	7	6	5	4	3	2	1	0
0xFA	HORIZDN	INTP	ZOOMOUT[37:36]		OFFODD[35:32]			
0xFB	OFFODD[31:24]							
0xFC	OFFODD[23]	OFFEVEN[22:16]						
0xFD	OFFEVEN[15:10]						INCREMENT[9:8]	
0xFE	INCREMENT[7:0]							

Address: *VideoBaseAddress* + 0xFA, 0xFB, 0xFC, 0xFD, 0xFE

Type: Read/write

Reset: 0

Description: VSYNC synchronization.

0xFA: [7] **HORIZDN**

Horizontal downsample by 2 (neighboring samples are averaged)
Interpolation is disabled by setting the HORIZDN bit.

0xFA: [6] **INTP**

When set, the interpolation field forces a line repeat (or line drop) instead of 2-bit interpolation.

0xFA: [5:4] **ZOOMOUT**

The INCREMENT bit is used for downsampling up to zoom out x2, for zoom out x3 and x4, ZOOMOUT must be used in as below. For zoom out between x2 and x3, or x3 and x4, for example x2.1 and x2.2. ZOOMOUT must be used in conjunction with INCREMENT.

00: No zoom out

01: Zoom out by 2

10: Zoom out by 3 ($\frac{8}{3}$)

11: Zoom out by 4

0xFA: [3:0] **OFFODD**

0xFB: [7:0] Vertical scans for odd fields. Adjusts and superposes the luma and chroma filtering. The offset is a variable distance from the first line.

0xFC: [6:0] **OFFEVEN**

0xFD: [7:2] Vertical scans for even fields. Adjusts and superposes the luma and chroma filtering. The offset is a variable distance from the first line.

0xFD: [1:0] **INCREMENT**

0xFE: [7:0] The INCREMENT bit is used for downsampling up to zoom out x2, and in conjunction with ZOOMOUT for fractional zoom out between x2 and x3, or x3 and x4, for example x2.1 and x2.2.

Define the vertical ratio of the zoom, and given by the formula:

$$\text{increment} = \left(\frac{\text{framestoresize}}{\text{desiredsize}} \times 512 \right) - 1$$

Where framestoresize is the number of lines in the framestore, and desiredsize is the number of lines in the display region.

For example, to display a 525 line framestore on to a 625 display

$$\text{region: increment} = \left(\frac{525}{625} \times 512 \right) - 1 = 429$$

The formula is only true in case no offset is performed due to OFFEVEN and OFFODD bits of the VID_VLFMODE and VID_VFCMODE registers

VID_VFLMODE

Configure luminance of block row

	7	6	5	4	3	2	1	0
0xDA	HORIZDN	INTP	ZOOMOUT[37:36]		OFFODD[35:32]			
0xDB	OFFODD[31:24]							
0xDC	OFFODD[23]	OFFEVEN[22:16]						
0xDD	OFFEVEN[15:10]						INCREMENT[9:8]	
0xDE	INCREMENT[7:0]							

Address: *VideoBaseAddress* + 0xDA, 0xDB, 0xDC, 0xDD, 0xDE

Type: Read/write

Reset: 0

Description: VSYNC synchronization

0xDA: [7] **HORIZDN**

Horizontal downsample by 2 (neighboring samples are averaged)
Interpolation is disabled by setting the HORIZDN bit.

0xDA: [6] **INTP**

When set, the interpolation field forces a line repeat (or line drop) instead of 2-bit interpolation.

0xDA: [5:4] **ZOOMOUT**

The INCREMENT bit is used for downsampling up to zoom out x2, for zoom out x3 and x4, ZOOMOUT must be used in as below. For zoom out between x2 and x3, or x3 and x4, for example x2.1 and x2.2. ZOOMOUT must be used in conjunction with INCREMENT.

00: No zoom out

01: Zoom out by 2

10: Zoom out by 3 ($\frac{8}{3}$)

11: Zoom out by 4

0xDA: [3:0] **OFFODD**

0xDB: [7:0] Vertical scans for odd fields. Adjusts and superposes the luma and chroma filtering. The offset is a variable distance from the first line.

0xDC: [6:0] **OFFEVEN**

0xDD: [7:2] Vertical scans for even fields. Adjusts and superposes the luma and chroma filtering. The offset is a variable distance from the first line.

0xDD: [1:0] **INCREMENT**

0xDE: [7:0] The INCREMENT bit is used for downsampling up to zoom out x2, and in conjunction with ZOOMOUT for fractional zoom out between x2 and x3, or x3 and x4, for example x2.1 and x2.2.

Define the vertical ratio of the zoom, and given by the formula:

$$\text{Increment} = \left(\frac{\text{framestoresize}}{\text{desiredsize}} \times 512 \right) - 1$$

Where framestoresize is the number of lines in the framestore, and desiredsize is the number of lines in the display region.

For example, to display a 525 line framestore on to a 625 display region:

$$\text{Increment} = \left(\frac{525}{625} \times 512 \right) - 1 = 429$$

This formula is only true in case no offset is performed due to OFFEVEN and OFFODD bits of the VID_VFLMODE and VID_VFCMODE registers

VID_XDO**Display X offset**

	7	6	5	4	3	2	1	0
0x70	Reserved							XDO[9:8]
0x71	XDO[7:0]							

Address: *VideoBaseAddress* + 0x70 to 0x71

Type: Read/write

Reset: 0

Description: VSYNC synchronization

VID_XDO is set up with a number defining the beginning of the left border of the display. This offset is measured from the active (first) edge of HSYNC and is specified in units of PIXCLK cycles.

The horizontal offset, XDO', is given by: $XDO' = VID\_XDO + 4$.

The offset, XDO cannot be less than 153 video decoder clock cycles.

VID_XDO_656**Display X offset (in 656 mode only)**

	7	6	5	4	3	2	1	0
0xB5	Reserved							XDO_656[9:8]
0xB6	XDO_656[7:0]							

Address: *VideoBaseAddress* + 0xB5 to 0xB6

Type: Read/write

Reset: 0

Description: VSYNC synchronization

VID_XDO_656 is set up with a number defining the beginning of the left border of the display. This offset is measured from the active (first) edge of HSYNC and is specified in units of PIXCLK cycles.

The horizontal offset, XDO_656, is given by: $XDO\_656' = VID\_XDO\_656 + 4$.

The offset, XDO_656 cannot be less than 153 video decoder clock cycles.

VID_XDS**Display X end**

	7	6	5	4	3	2	1	0
0x72	Reserved							XDS[9:8]
0x73	XDS[7:0]							

Address: *VideoBaseAddress* + 0x72 to 0x73

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a number defining the right boundary of the picture display window, expressed in units of PIXCLK cycles. The value of VID_XDS must be equal to VID_XDO plus the width of the display window.

The actual offset, XDS', is given by:

$XDS' = VID\_XDS + 4$.

Confidential

VID_XDS_656**Display X end (in 656 mode only)**

	7	6	5	4	3	2	1	0
0xF5	Reserved						XDS_656[9:8]	
0xF6	XDS_656[7:0]							

Address: *VideoBaseAddress* + 0xF5 to 0xF6

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a number defining the right boundary of the picture display window, expressed in units of PIXCLK cycles. The value of VID_XDS_656 must be equal to VID_XDO_656 plus the width of the display window.

The actual offset, XDS_656', is given by:

$XDS\_656' = VID\_XDS\_656 + 4.$

VID_XFW**Displayed frame width**

	7	6	5	4	3	2	1	0
0x28	XFW[7:0]							

Address: *VideoBaseAddress* + 0x28

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a value equal to the width in macroblocks of the displayed picture. This is derived from the horizontal size value transmitted in the sequence header.

VID_YDO**Display Y offset**

	7	6	5	4	3	2	1	0
0x6E	Reserved						YDO[9:8]	
0x6F	YDO[7:0]							

Address: *VideoBaseAddress* + 0x6E to 0x6F

Type: Read/write

Reset: 0

Description: VSYNC synchronization

0x6E: [7:2] **Reserved**0x6E: [1:0] **YDO**

0x6F: [7:0] This value defines the first line of the picture display, that is, the top edge of the display window. Lines are counted from the active (first) edge of VSYNC.

In an interlaced display, the same value of VID_YDO must be used for both fields.

VID_YDO_656**Display Y offset (in 656 mode only)**

	7	6	5	4	3	2	1	0
0xB8	Reserved						YDO_656[9:8]	
0xB9	YDO_656[7:0]							

Address: *VideoBaseAddress* + 0xB8 to 0xB9

Type: Read/write

Reset: 0

Description: VSYNC synchronization

0xB8: [7:2] **Reserved**0xB8: [1:0] **YDO_656**

0xB9: [7:0] This value defines the first line of the picture display, that is, the top edge of the display window. Lines are counted from the active (first) edge of VSYNC.

In an interlaced display, the same value of VID_YDO_656 must be used for both fields.

VID_YDS**Display Y end**

	7	6	5	4	3	2	1	0
0x46	Reserved						YDS[9:8]	
0x47	YDS[7:0]							

Address: *VideoBaseAddress* + 0x46 to 0x47

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a number defining the bottom line of the display window. The value of YDS must be equal to:

 $\text{VID\_YDO} + (\text{display size} / 2) - 1$

In an interlaced display, the same value of VID_YDS must be used for both fields.

VID_YDS_656**Display Y end (in 656 mode only)**

	7	6	5	4	3	2	1	0
0xF8	Reserved						YDS_656[9:8]	
0xF9	YDS_656[7:0]							

Address: *VideoBaseAddress* + 0xF8 to 0xF9

Type: Read/write

Reset: 0

Description: VSYNC synchronization

This register is set up with a number defining the bottom line of the display window. The value of YDS_656 must be equal to:

 $\text{VID\_YDO\_656} + (\text{display size} / 2) - 1$

In an interlaced display, the same value of VID_YDS_656 must be used for both fields.

Confidential

40 2-D block move

40.1 Overview

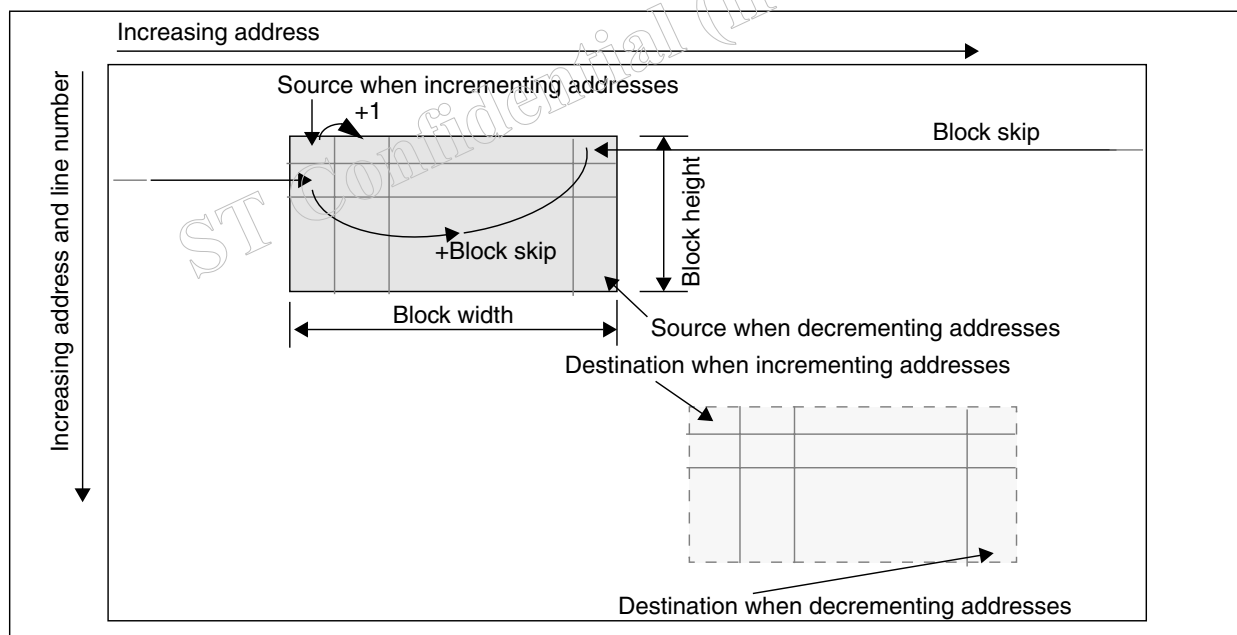
This module copies rectangular blocks of data from one area in display memory to another. The width and height of the destination block must be the same as the width and height of the source block. If the source area overlaps the destination area, then the result depends on the copy direction. To copy correctly, the copy direction must first be setup by the user (see [Figure 127: How to handle overlaps on page 398](#)). The module cannot access devices other than the MPEG/Graphics SDRAM.

The copying is done by DMA engines, so the CPU can perform other tasks during the copying. The interface between the CPU and the 2-D block move module is provided using a set of registers and an interrupt to signal when a copy has completed.

40.2 Copying blocks of data

To perform a 2-D block move, the module must first be initialized with the source and destination addresses and the dimensions of the block to be copied. The terminology used for 2-D block moves is shown below.

Figure 126: 2-D block move



The source and destination addresses are the start of the source and destination areas and must be 64-bit aligned. Normally the addresses are incremented at each step, so the top left corner of the source and destination blocks should be given. If the addresses are set to decrement by writing 1 to D_A (USD_BMC) then the bottom right hand corners must be given.

The source and destination addresses are indicated by the initial value of the registers USD_BRP and USD_BWP. These registers give the offsets in 64-bit words from the base of the SDRAM, *SDRAM_base*. Thus if the full source and destination addresses are *source_address* and *dest_address* then the following values should be written in the following registers:

- $\text{USD\_BRP} = (\text{source\_address} - \text{SDRAM\_base}) / 8,$
- $\text{USD\_BWP} = (\text{dest\_address} - \text{SDRAM\_base}) / 8.$

The block width is held in USD_BMW. It is given in 64-bit words, and can be any value in the range of 1 to 65535. The block height is held in USD_BMH and is given in lines. It can be any value in the range of 1 to 1023 lines.

The block skip is defined as the number of 64-bit words from the end of one line of the block to the beginning of the next. That is, the width of the displayed picture on the screen, less than the width of the block to be copied, plus or minus one pixel, depending on bit D_A as shown in Figure 127. The block skip is held in register USD_BSK.

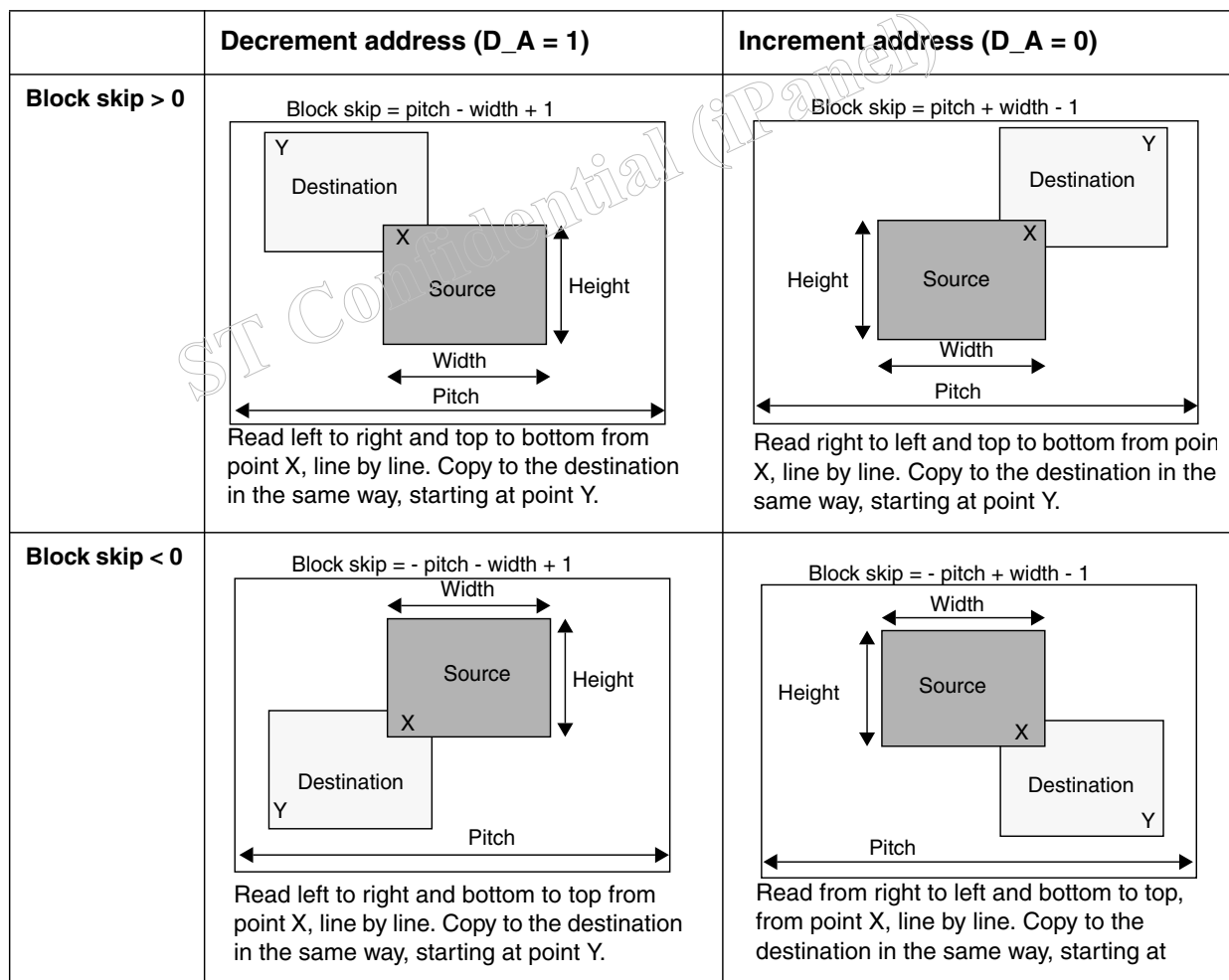
To start the block move, 1 is written to EXE (USD_BMC). Depending on the value of fields in the control register USD_BMC, the 2-D block move engine can be set up to start on various types of events:

- the start of display of the odd field,
- the start of display of the even field,
- after the display of a given line number.

Setting CPAT (USD_BMC) causes the pattern in register USD_PAT to be copied to the whole of the destination rectangular block as a fill function.

At the end of the block move operation an interrupt is generated.

Figure 127: How to handle overlaps



41 2-D block move registers

Addresses are provided as the *VideoBaseAddress* + offset.

The *VideoBaseAddress* is:

0x0000 0000.

A register summary is given in [Table 29: 2-D block move registers on page 48](#).

USD_BMC

Block move control

	7	6	5	4	3	2	1	0
0x9E	NUM_LINE[7:0]							
0x9F	Reserved	D_A	NIR	CPAT	RST	VSE	VSO	EXE

Address: *VideoBaseAddress* + 0x9E and 0x9F

Type: Read/write

Reset: 0x00

Description: This register controls the 2-D block move engine.

0x9E: [7:0] **NUM_LINE**

When NUM_LINE is reached, the block move starts.

0x9F: [7] **Reserved**

0x9F: [6] **D_A**

Decrement address.

0: Increment the address by one after each read or write

1: Decrement the address by one after each read or write

This has to be combined with the right skip value to generate a block move from right bottom to left top.

0x9F: [5] **NIR**

No increment read.

When this bit is set, the read address never changes. This feature can be used to fill the memory with a pattern stored in memory.

0x9F: [4] **CPAT**

Copy pattern.

When this bit is set the data written is specified by register. See [USD_PAT on page 401](#).

0x9F: [3] **RST**

Reset block move.

When this bit is set the block move is reset. This bit must be reset before continuing (after a minimum of 200 ns).

0x9F: [2] **VSE**

Vertical synchronization even.

When this bit is set the block move waits for the even VSYNC to begin. This can be combined with line number.

0x9F: [1] **VSO**

Vertical synchronization odd.

When this bit is set the block move waits for the odd VSYNC to begin. This can be combined with line number. If you set VSE and VSO, the block move begins at next VSYNC.

0x9F: [0] **EXE**

Execute.

This bit has to be set to launch the block move. This bit is internally cleared when the block move begins.

Confidential

USD_BMH**Block move height**

	7	6	5	4	3	2	1	0
0xA2	Reserved						USD_BMH[9:8]	
0xA3	USD_BMH[7:0]							

Address: *VideoBaseAddress* + 0xA2 and 0xA3

Type: Read/write

Reset: Undefined

Description: When a 2-D block move is to be executed, this register holds the number of lines of the block to be moved.

USD_BMW**Block move width**

	7	6	5	4	3	2	1	0
0xA0	USD_BMW[15:8]							
0xA1	USD_BMW[7:0]							

Address: *VideoBaseAddress* + 0xA0 to 0xA1

Type: Read/write

Reset: 0

Description: This register holds the number of 64-bit words to be moved in a block move operation. The write to the higher address A1 (the least significant byte) with USD_BMW nonzero enables (but does not launch) the 2-D block move mode. This register must be written before the associated USD_BRP. The byte pointer is reset by a hardware reset.

USD_BRP**Memory read pointer**

	7	6	5	4	3	2	1	0
0x88	Reserved			USD_BRP[20:16]				
0x89	USD_BRP[15:8]							
0x8A	USD_BRP[7:0]							

Address: *VideoBaseAddress* + 0x88 to 0x8A

Type: Read/write

Reset: Undefined

Description: This register holds the source address of the block to be moved. USD_BRP defines a 21-bit memory word address (a memory word is 64 bits). The pointer is read sequentially from 0x88 to 0x8A

USD_BSK**Block skip**

	7	6	5	4	3	2	1	0
0xA4	Reserved			USD_BSK[20:16]				
0xA5	USD_BSK[15:8]							
0xA6	USD_BSK[7:0]							

Address: *VideoBaseAddress* + 0xA4, 0xA5, 0xA6

Type: Read/write

Reset: Undefined

Description: When a 2-D block move is to be executed, this register holds the number of 64-bit words to skip before finding the beginning of the next line to move. As this register has 21 bits, it is able to contain negative values in two's complement in order to read the source from bottom to top. This is useful to handle source/destination overlap in memory.

USD_BWP**Memory write pointer**

	7	6	5	4	3	2	1	0
0x8C	Reserved			USD_BWP[20:16]				
0x8D	USD_BWP[15:8]							
0x8E	USD_BWP[7:0]							

Address: *VideoBaseAddress* + 0x8C to 0x8E

Type: Read/write

Reset: Undefined

Description: When a block move is to be executed, this register holds the destination address where the block is to be moved. USD_BWP defines a 21-bit memory word address, where a memory word is 64 bits. The pointer is written sequentially from 0x8C to 0x8E.

USD_PAT**Block move pattern**

	7	6	5	4	3	2	1	0
0xA7	PATTERN[63:56]							
0xA8	PATTERN[55:48]							
0xA9	PATTERN[47:40]							
0xAA	PATTERN[39:32]							
0xAB	PATTERN[31:24]							
0xAC	PATTERN[23:16]							
0xAD	PATTERN[15:8]							
0xAE	PATTERN[7:0]							

Address: *VideoBaseAddress* + 0xA7 to 0xAE

Type: Read/write

Reset: Undefined

Description: This register is to be used to fill a part of memory with a pattern. It is a 64-bit pattern.

42 Teletext DMA

42.1 Overview

Teletext data is retrieved from memory, serialized and transferred to the digital encoder by a dedicated teletext DMA. The digital encoder encodes teletext data according to the CCIR/ITU-R Broadcast Teletext System B specification (also known as World System Teletext).

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.

42.2 Teletext packet format

One 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. The DMA reads in multiples of 46 bytes and transfers lines of 45 bytes to the digital encoder.

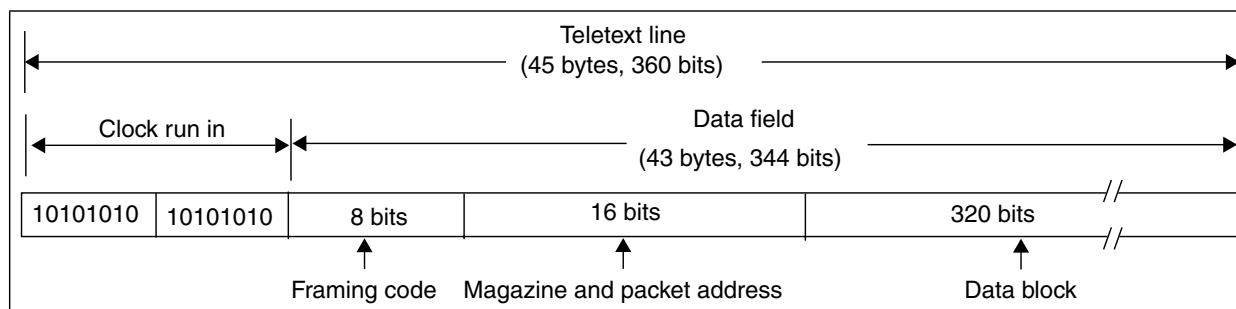
Each teletext packet is composed of the clock run in and the data field, as illustrated in Figure 128.

- 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 0xE4¹. The data is transmitted in order, from the LSB to the MSB of each byte in memory.

Figure 128: Teletext packet format



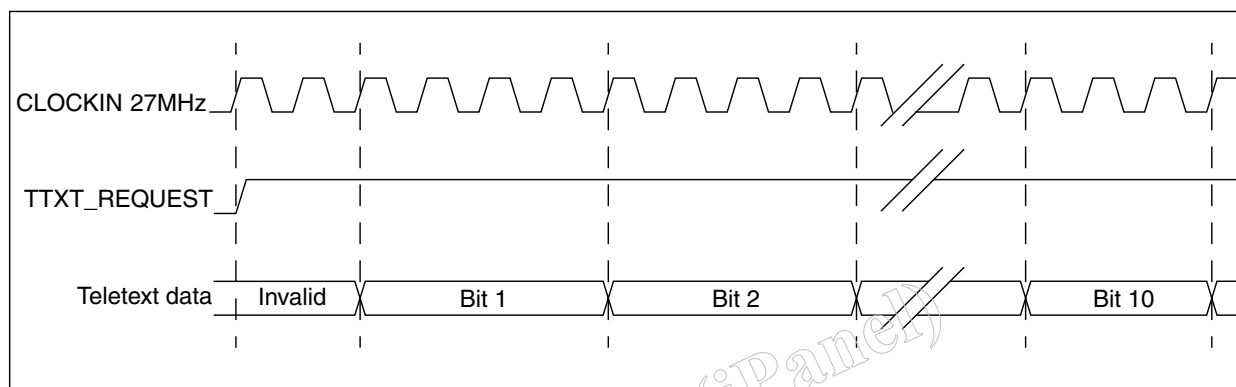
1. Specification for conveying ITU-R Systems B Teletext in Digital Video Broadcasting (DVB) bit streams.

42.3 Data transfer sequence

The digital encoder issues a teletext request signal to the teletext DMA, this is shown by the rising edge of signal TTX_REQUEST in Figure 129. After a delay, programmable from two to nine master clock periods, the teletext DMA transmits the first valid teletext data bit of the teletext packet.

The 360 bits of output data are defined as nine 37-bit sequences, ending with one 27-bit sequence. Within each sequence, each bit is transmitted in four 27 MHz cycles, except bits 10, 19, 28 and 37, which are transmitted in three 27 MHz cycles. This is illustrated in the figure below for bits 0 to 10.

Figure 129: Teletext data transfer sequence



The duration of the TTXS window is 1402 reference clock periods (51.926 μ s), which corresponds to the duration of 360 teletext bits.

The delay between signal TTX_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.

42.4 Interrupt control

Teletext interrupts can be programmed by the TTX_INTENABLE register to interrupt the CPU whenever one of the following occurs:

- a teletext data transfer is complete,
- the current video frame toggles odd to even or even to odd.

The interrupt status is given by the TTX_INTSTATUS register and masked by the TTX_INTENABLE register. The interrupt bits are reset when the CPU writes to the acknowledge register, or when a DMA operation is completed.

43 Teletext DMA registers

Addresses are provided as the *TtxtBaseAddress* + offset.

The *TtxtBaseAddress* is:

0x2012 4000.

A register summary is given in [Table 54: Teletext DMA registers on page 79](#).

TTXT_ABORT

Teletext abort

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

ABORT

Address: *TtxtBaseAddress* + 0x24

Type: Write only

Description: A write of any value to this address causes the teletext interface to abort the current operation. The state of the teletext operation is reset, and the teletext data transfer is interrupted. The DMA engine is reset only after the current word read or write is complete.

TTXT_ACKODDEVEN

Teletext acknowledge odd or even

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

ACKODDEVEN

Address: *TtxtBaseAddress* + 0x20

Type: Write only

Description: This register acknowledges the odd/even toggle interrupt. Any write to the TTXT_ACKODDEVEN register clears the ODD and EVEN bits of the TTXT_INTSTATUS register.

TTXT_DMAADDRESS

Teletext DMA 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

DMAADDRESS

Address: *TtxtBaseAddress* + 0x00

Type: Read/write

Reset: Undefined

Description: The TTXT_DMAADDRESS register is a 32-bit register, which specifies the base address in memory for the DMA transfer from memory.

[31:0] **DMAADDRESS**

The base address for DMA transfer of data to or from memory.

Confidential

TTXT_DMACOUNT**Teletext DMA 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											DMACOUNT									

Address: *TtxtBaseAddress* + 0x04

Type: Read/write

Reset: Undefined

Description: The TTXT_DMACOUNT register specifies the number of bytes to be transferred from memory during the DMA operation. A write to this register also starts the teletext output operation.

- For teletext output operation, this value must be:
46 bytes x *number_of_teletext_lines_to_send*.
- For teletext input operation, the value must be:
42 bytes x *number_of_teletext_lines_to_receive*.

TTXT_INTENABLE**Teletext interrupt enable**

7 6 5 4 3 2 1 0

Reserved				EVENENABLE		ODDENABLE	INOUTCOMPLETEEN

Address: *TtxtBaseAddress* + 0x1C

Type: Read/write

Reset: Undefined

Description: The TTXT_INTENABLE register allows masking of the TTXT_INTSTATUS register.

[7:3] **Reserved**[2] **EVENENABLE**

Enable even field interrupt.

[1] **ODDENABLE**

Enable odd field interrupt.

[0] **INOUTCOMPLETEEN**

Enable teletext input or output operation completed interrupt.

Confidential

TTXT_INTSTATUS**Teletext 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																EVEN		ODD		INOUTCOMPLETE											

Address: *TtxtBaseAddress* + 0x18

Type: Read only

Reset: Undefined

Description: The TTXT_INTSTATUS register gives the current state of the teletext operations. If the appropriate bits in the interrupt enable register are set then interrupts can be driven by the state of this register.

[31:3] **Reserved**

[2] **EVEN:** Current (video encoder) field is EVEN

[1] **ODD:** Current (video encoder) field is ODD

[0] **INOUTCOMPLETE:** Teletext input or output operation completed

TTXT_MODE**Teletext 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																ODDEVEN		MODE													

Address: *TtxtBaseAddress* + 0x14

Type: Read/write

Reset: Undefined

Description: The TTXT_MODE register sets the mode of the teletext interface to teletext data output. It also specifies whether teletext data from memory is for odd or even fields.

[31:2] **Reserved**

[1] **ODDEVEN:** Specify odd or even fields of teletext data

0: Teletext data from memory is for even fields

1: Teletext data from memory is for odd fields.

[0] **MODE:** Teletext interface mode

0: Teletext output enabled (this bit must be programmed to 0)

TTXT_OUTDELAY**Teletext output 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																DELAY															

Address: *TtxtBaseAddress* + 0x08

Type: Read/write

Reset: Undefined

Description: This register programs the delay, in 27 MHz clock periods, from the rising edge of TTXT_REQUEST to the first valid teletext data bit, that is TTXT_DATA starting to transmit.

44 Digital encoder

44.1 Overview

This the final stage of the video pipeline of the device is a high performance PAL/SECAM/NTSC digital encoder referred to as the digital encoder. The digital encoder converts a 4:2:2 digital video stream into a standard analog baseband PAL/SECAM/NTSC signal and into RGB and YUV analog components.

The digital encoder can perform closed captions, CGMS, WSS, teletext and VPS encoding and allows Macrovision™ 7.01/ 6.1 copy protection. Six analog output pins are available, on which it is possible to output one of the following:

- (S-VHS[Y/C]+ CVBS + RGB),
- (S-VHS[Y/C]+ CVBS + V + Y + U),
- (Y1 + C1 + CVBS1 + C2 + Y2 + CVBS2).

The encoder can operate in master mode or in one of several slave modes, where it locks on to incoming synchronous signals. An autotest mode is also provided.

Note: The digital encoder handles interlaced mode only.

44.2 Video timing

The burst sequences are internally generated, subcarrier generation being performed numerically with CKREF 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 44.8: Subcarrier insertion \(SECAM\) on page 419](#)).

[Figure 131](#) to [Figure 134](#) depict typical vertical blanking interval (VBI) waveforms.

It is possible to encode incoming YCrCb data on those lines of the VBI that do not bear line sync pulses or pre-equalization and postequalization pulses (see [Figure 131](#) to [Figure 134](#)). This mode of operation is referred to as *partial blanking* and is the default set up. It allows the encoded waveform to keep any VBI data present in digitized form in the incoming YCrCb stream (for example, supplementary closed caption lines or StarSight® data). In SECAM mode, only Y data are encoded, Cr and Cb are ignored. Alternatively, the complete VBI may be *fully blanked*, so no incoming YCrCb data is encoded on these lines. Full or partial blanking is set by register bit BLKLI (DEN_CFG1).

For 525/60 systems, with the SMPTE line numbering convention:

- complete VBI consists of lines 1 to 19 and the second half of lines 263 to 282,
- partial VBI consists of lines 1 to 9 and the second half of lines 263 to 272,
- line 282 is either fully blanked or fully active.

For 625/50 systems, with the CCIR line numbering convention:

- complete VBI consists of the second half of lines 623 to 22 and lines 311 to 335,
- partial VBI consists of the second half of lines 623 to 5 and lines 311 to 318,
- line 23 is always fully active.

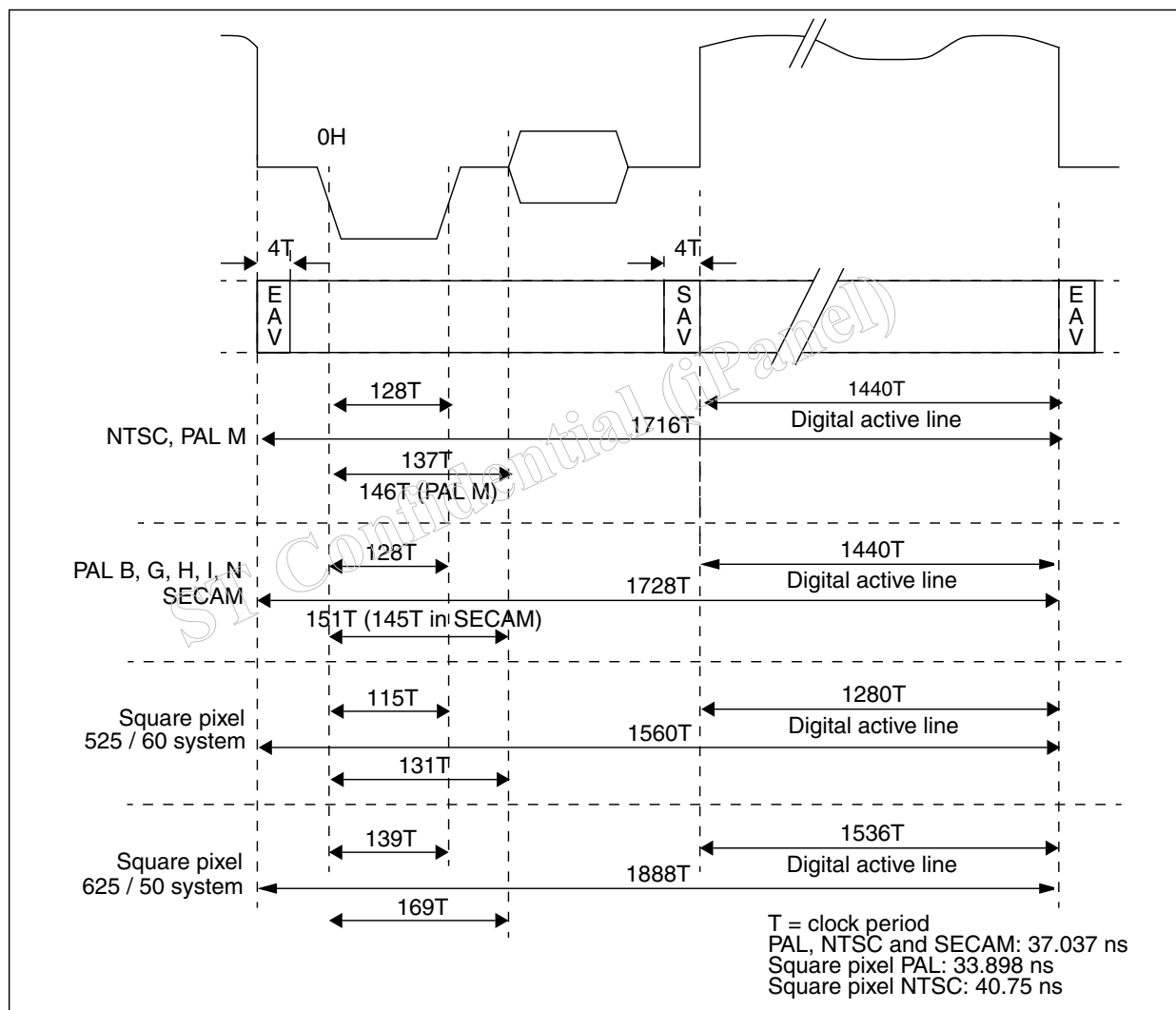
In an ITU-R656 compliant digital TV line, the active portion of the digital line is the portion included between the start of active video (SAV) and end of active video (EAV) words. However, this digital active line starts somewhat earlier and may end slightly later than the active line usually defined by analog standards.

The digital encoder allows two approaches.

- Encoding 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.
- Dropping some YCrCb samples at the extremities of the digital line so that the encoded analog line fits within the analog ITU-R/SMPTE specifications.

In all cases, the transitions between horizontal blanking and active video are shaped to avoid too steep edges within the active video. Figure 134 gives typical timings concerning the horizontal blanking interval and the active video interval.

Figure 130: Input data format (ITU-R656 /D1 4:2:2)



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 131: PAL-BDGHI, PAL-N typical VBI waveform, interlaced mode (ITU-R625 line numbering)

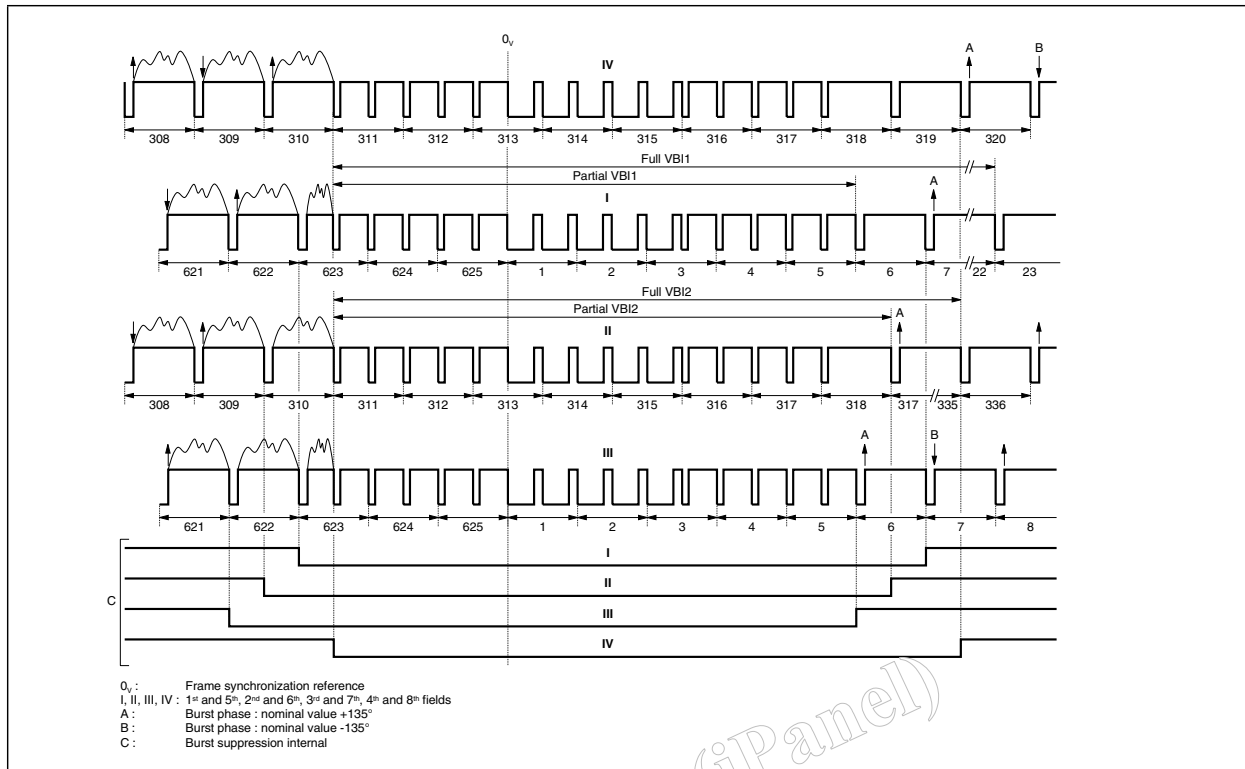


Figure 132: NTSC-M typical VBI waveforms, interlaced mode (SMPTE-525 line numbering)

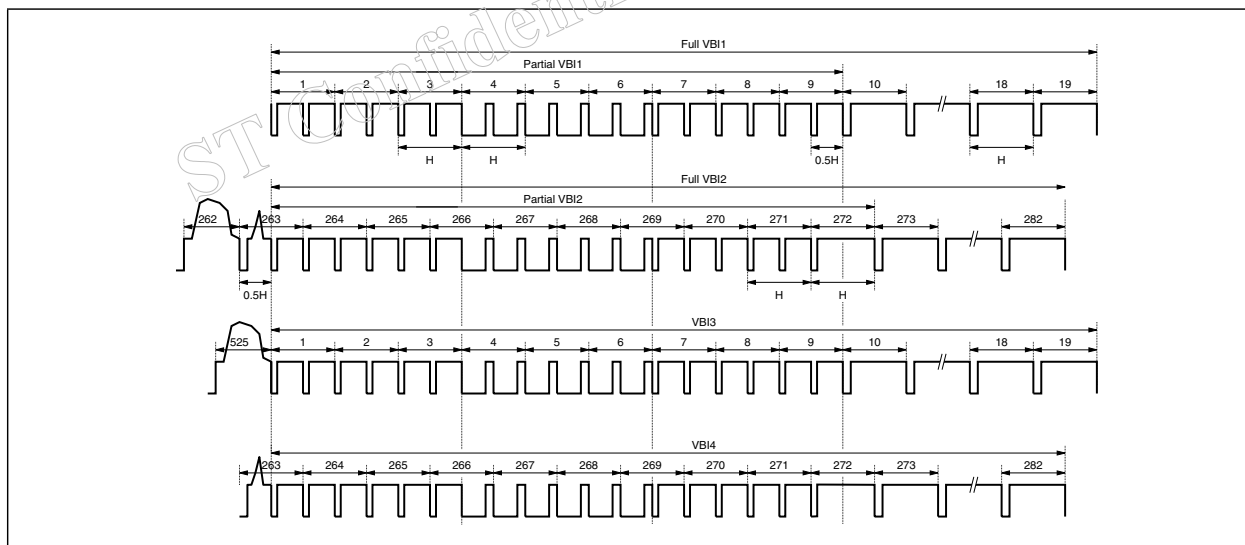


Figure 133: PAL-M typical VBI waveforms, interlaced mode (ITU-R/CCIR-525 line numbering)

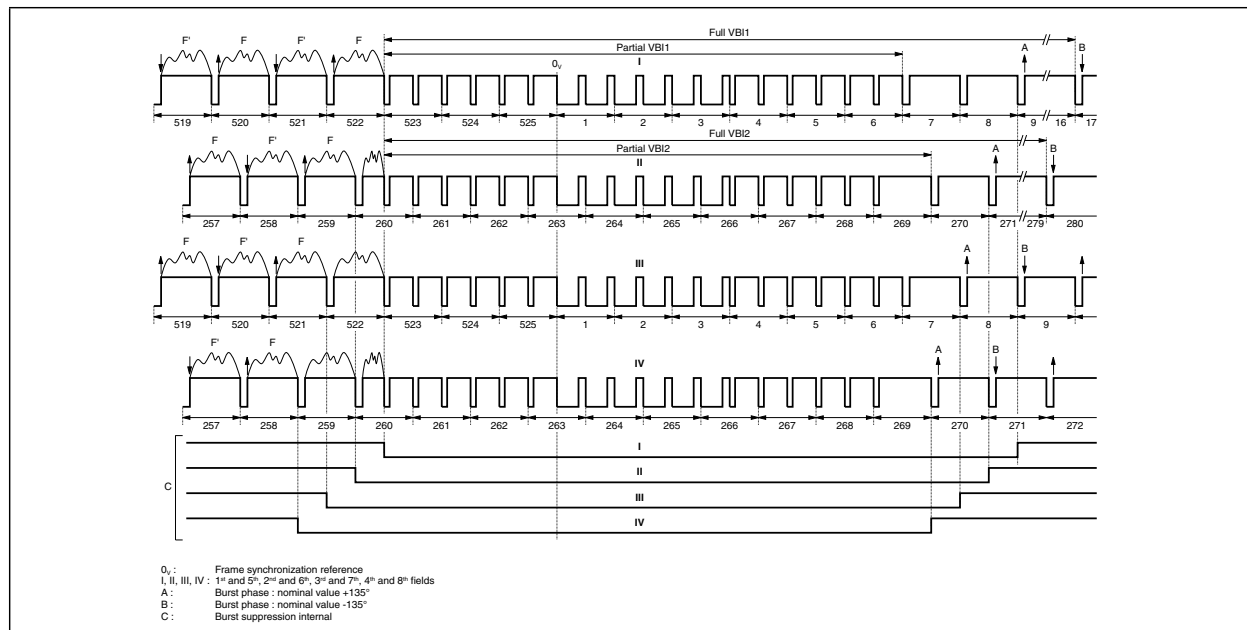


Figure 134: Horizontal blanking interval and active video timings

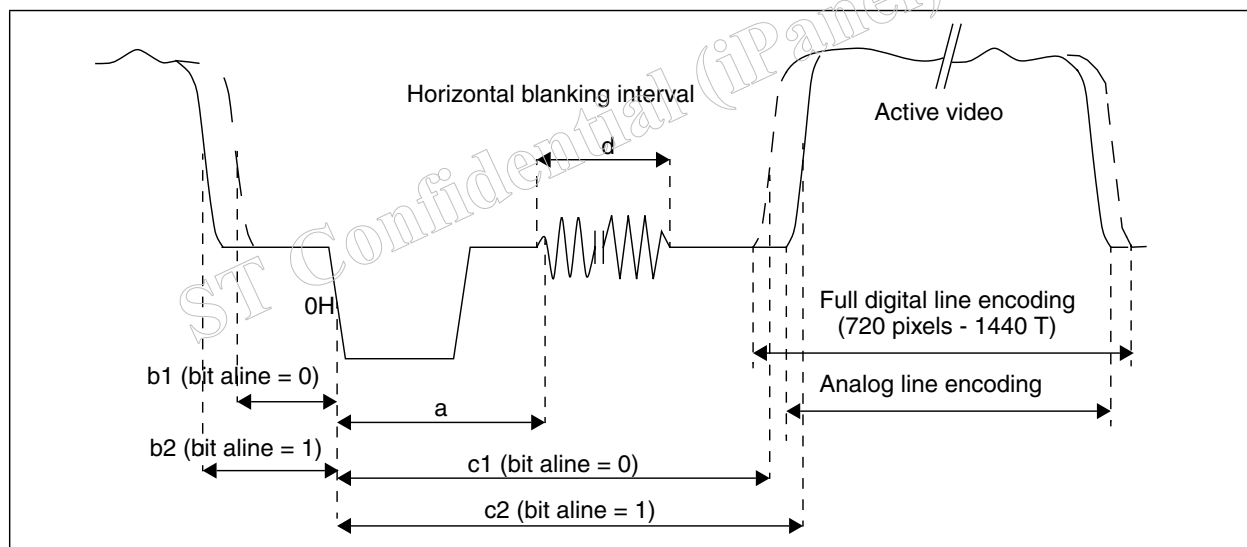


Table 158: Horizontal blanking interval and active video typical timings

	NTSC-M	PAL-BDGI	PAL-N	PAL-M	SECAM
a ^a	5.38 μ s (even lines) 5.52 μ s (odd lines)	5.54 μ s (A-type) 5.66 μ s (B-type)	5.54 μ s (A-type) 5.66 μ s (B-type)	5.73 μ s (A-type) 5.87 μ s (B-type)	5.60 μ s
b1	1.56 μ s	1.3 μ s	1.3 μ s	1.56 μ s	1.0 μ s
b2	1.56 μ s	1.52 μ s	1.52 μ s	1.56 μ s	1.52 μ s
c1	8.8 μ s	9.6 μ s	9.6 μ s	8.8 μ s	9.9 μ s
c2	9.41 μ s	10.48 μ s	10.48 μ s	9.41 μ s	10.48 μ s
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	-

a. These are typical values, actual values depend on the static offset programmed for subcarrier generation.

44.3 Reset procedure

A hardware reset sets the digital encoder in HSYNC + ODDEVEN (line locked) slave mode, for NTSC-M, interlaced ITU-R601 encoding closed captioning, WSS, VPS and CGMS encoding are all disabled.

The configuration can then be customized by writing into the appropriate registers. A few registers are never reset, their contents are unknown until the first loading. See the register information in [Chapter 45: Digital encoder registers on page 432](#).

It is also possible to perform a software reset by setting the seventh bit in the DEN_CFG6 register. The device responds in a similar way as after a hardware reset except that the configuration registers and a few other registers are not altered.

44.4 Slave modes

44.4.1 Introduction

The following slave modes are available:

- ODDEVEN(VSYNC) + HSYNC based (line based sync),
- ODDEVEN(VSYNC)-only based (frame based sync),
- sync in data based (line locked or frame locked).

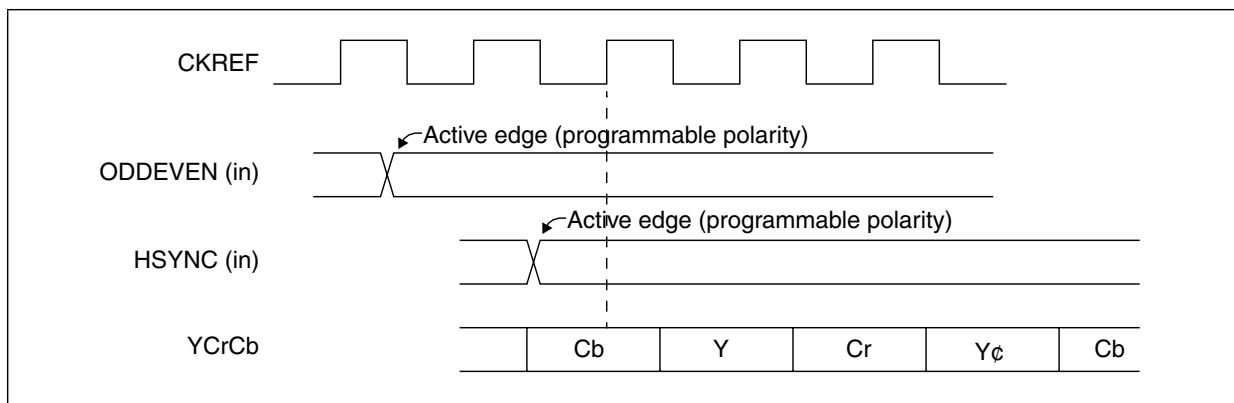
ODDEVEN refers to an odd/even field flag, also known as BottomTop. HSYNC is a line sync signal and VSYNC is a vertical sync signal. Their waveforms are depicted in [Figure 135](#). The polarities of HSYNC and ODDEVEN(VSYNC) are independently programmable in all slave modes. In all slave modes, ODDEVEN(VSYNC) or HSYNC signals, must be related to CKREF, the principal digital encoder clock. In other words, there is *no* genlocking performed by the digital encoder.

44.4.2 Line based synchronization

ODDEVEN + HSYNC based synchronization

Synchronization is performed on a line by line basis by locking on to incoming ODDEVEN and HSYNC signals. Refer to [Figure 135](#) for waveforms and timings. The polarities of the active edges of HSYNC and ODDEVEN are programmable and independent. The first active edge of ODDEVEN initializes the internal line counter but encoding of the first line does not start until an HSYNC active edge is detected (at the earliest, an HSYNC transition may be at the same time as ODDEVEN). At that 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 + ODDEVEN) by up to three clock cycles to cope with different data/sync phases, using configuration bits SYNCIN_AD in DEN_CFG4. The digital encoder 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 is reinitialized when the early HSYNC arrives and all internal synchronization signals are reinitialized.
- 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 reinitializing.

Figure 135: HSYNC + ODDEVEN based slave mode sync signals

Note: This figure is valid for bits SYNCIN_AD[1:0] = default

HSYNC + VSYNC based synchronization

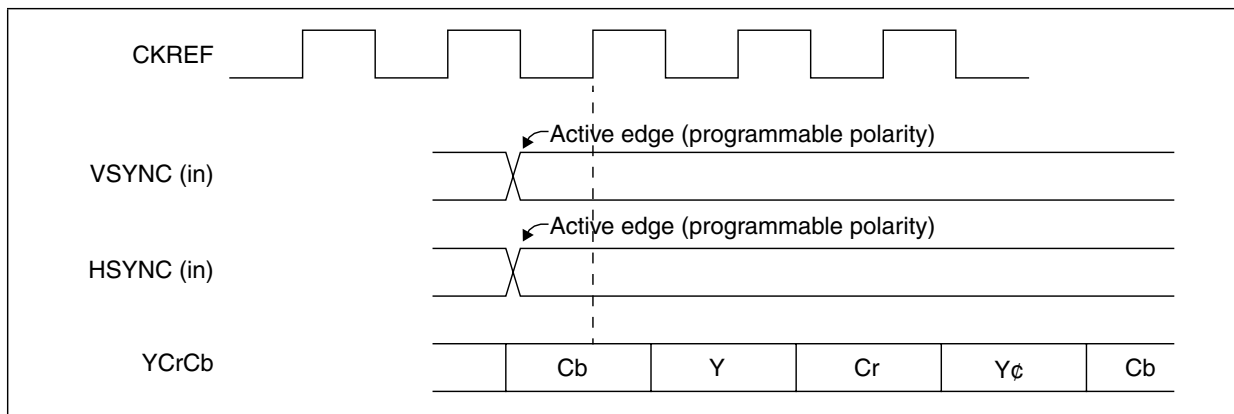
Synchronization is performed on a line by line basis by locking on to incoming VSYNC and HSYNC signals. Refer to [Figure 136](#) for waveforms and timings. The polarities of HSYNC and VSYNC are programmable and independent.

The incoming VSYNC signal is immediately transformed into a waveform identical to the odd/even waveform of an ODDEVEN signal, therefore, the behavior with this synchronization is identical to that described above for ODDEVEN + 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 (that is, a blue chroma sample within the YCrCb stream). It is however possible to internally delay the incoming sync signals (HSYNC + VSYNC) by up to three clock cycles to cope with different data/sync phasing, using configuration bits SYNCIN_AD (DEN_CFG4).

44.4.3 Frame based synchronization

ODDEVEN-only based synchronization

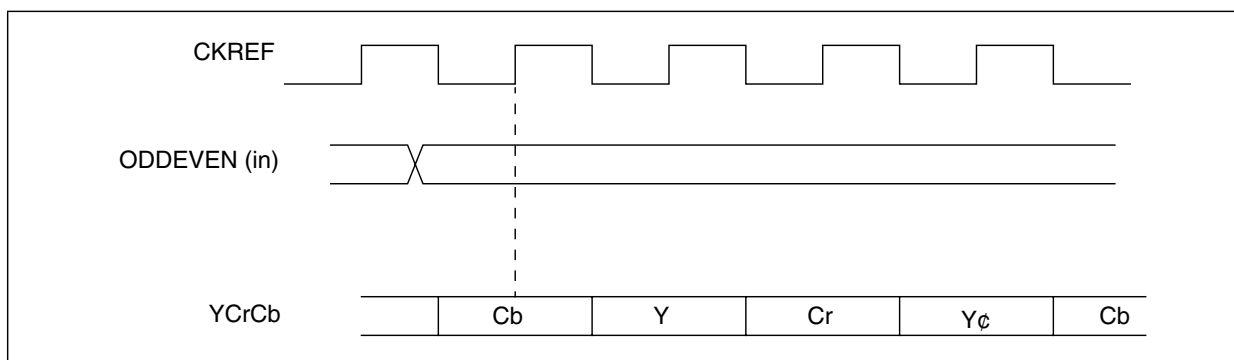
Synchronization is performed on a frame by frame basis by locking on to an incoming ODDEVEN signal. A line sync signal is derived internally and is also issued to the outside as HSYNC. Refer to [Figure 137](#) for waveforms and timings. The phase relationship between ODDEVEN and the incoming YCrCb data is normally such that the first clock rising edge following the ODDEVEN active edge samples Cb (that is, a blue chroma sample within the YCrCb stream). It is however possible to internally delay the incoming ODDEVEN signal by up to three clock cycles to cope with different data/sync phasing, using configuration bits SYNCIN_AD in DEN_CFG4.

Figure 136: 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 not transition before HSYNC.

Figure 137: ODDEVEN based slave mode sync signals



Note: This figure is valid for bits `SYNCIN_AD[1:0]` = default

The first active edge of ODDEVEN triggers generation of the analog sync signals and encoding of the incoming video data. Frames being supposed to be of constant duration, the next ODDEVEN active transition is expected at a precise time after the last ODDEVEN detected.

So, once an active ODDEVEN edge has been detected, checks that the following ODDEVEN are present at the expected instants are performed.

Encoding and analog sync generation carry on unless three successive fails of these checks occur.

In that case, three behaviors are possible, according to the configuration programmed in registers `DEN_CFG[1:2]`.

- If **FREERUN** is enabled:
The digital encoder carries on outputting the digital line sync HSYNC and generating analog video just as though the expected ODDEVEN edge had been present. However, it resynchronizes on to the next ODDEVEN active edge detected, whatever its location.
- If **FREERUN** is disabled but bit **SYNCOK** is set in the configuration registers:
The digital encoder sets the active portion of the TV line to black level but carries on outputting the analog sync tips (on Ys and CVBS) and the digital line sync signal HSYNC. When programmed, Macrovision™ pseudosync pulses and AGC pulses are also present in the analog sync waveform.
- If **FREERUN** is disabled and bit **SYNCOK** is not set:
All analog video is at black level and neither analog sync tips nor digital line sync are output.

This mode is a frame based sync mode, as opposed to a field based sync mode. This means that only one type of edge (rising or falling, according to programming) is of interest to the digital encoder; the other one is ignored.

VSNC-only based synchronization

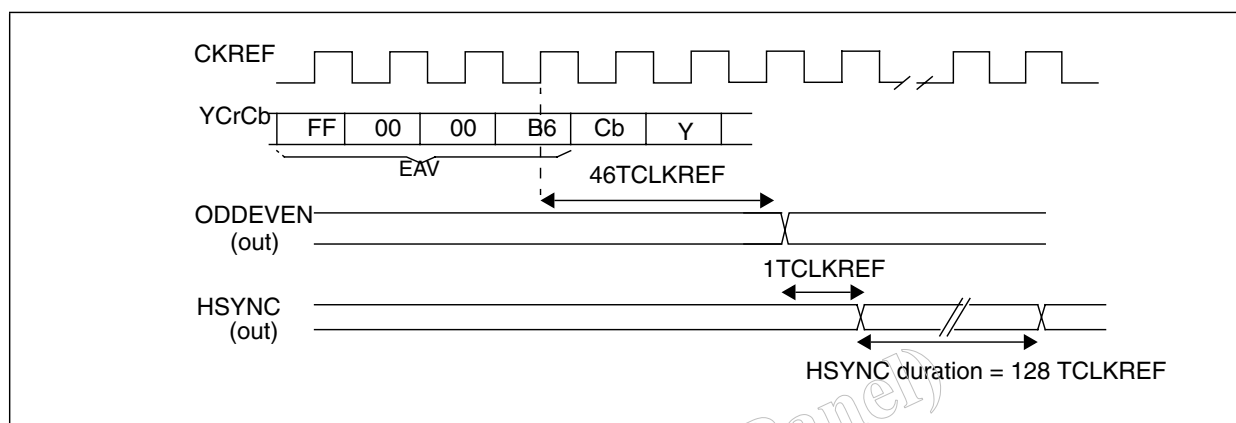
Synchronization is performed on a frame by frame basis by locking on to an incoming VSYNC signal.

An auxiliary line sync signal HSYNC must also be fed to the digital encoder, which uses it to reconstruct from VSYNC and HSYNC information an internal odd/even waveform identical to that of an ODDEVEN signal. Therefore, the behavior with this synchronization is identical to that described above for ODDEVEN-only based synchronization (except that nothing is output on HSYNC pin since it is an input port in this mode).

Note: HSYNC is an input but has no other use than allowing the digital encoder to decide whether an incoming VSYNC pulse flags an odd or an even field. In other words, the digital encoder does not lock on to HSYNC in this mode since this is NOT a line locked mode.

The phase relationship between VSYNC and the incoming YCrCb data is normally such that the first clock rising edge following the VSYNC 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 (VSYNC + HSYNC) by up to three clock cycles to cope with different data/sync phasing, using configuration bits SYNCIN_AD (DEN_CFG4).

Figure 138: Data (EAV) based slave mode sync signals



44.4.4 Sync in data based synchronization

End of frame word-based synchronization

Synchronization is performed by extracting the 1 to 0 transitions of the F flag (end of frame) from the EAV (end of active video) sequence embedded within ITU-R656 / D1 compliant digital video streams. Both a frame sync signal and a line sync signal are derived and are made available externally as ODDEVEN and HSYNC. Refer to [Figure 138](#) for waveforms and timings.

The first successful detection of the F flag triggers generation of the analog sync signals and encoding of the incoming video data. Frames being supposed to be of constant duration, the next EAV word containing the F flag is expected at a precise time after the latest detection.

So, once an active F flag has been detected, checks that the following flags are present within the incoming video stream at the expected times are performed.

Encoding and analog sync generation carry on unless there are three successive fails of these checks. Then, depending on the programmed configuration, one of the following three events occurs.

- If FREERUN is enabled
The digital encoder carries on generating the digital frame and line syncs (ODDEVEN and HSYNC) and generating analog video just as though the expected F flag had been present. However, it resynchronizes on to the next 'F' flag detected within the incoming ITU-R656/ D1 video stream.
- If FREERUN is disabled but the bit SYNCOK is set in the configuration registers
The digital encoder sets the active portion of the TV line to black level but carries on outputting the analog sync tips (on Ys and CVBS) and the digital frame and line sync signals ODDEVEN and HSYNC. (When programmed, Macrovision™ pseudosync pulses and AGC pulses are also present in the analog sync waveform).
- If FREERUN is disabled and the bit SYNCOK is not set
All analog video is at black level and neither analog sync tips nor digital frame/line sync are output.

The SAV and EAV words are Hamming decoded.

After detection of two successive errors, a bit is set in the status register to inform the microcontroller of the poor transmission quality.

End of line word-based synchronization

Synchronization is performed by extracting the F and H flags from the SAV (start of active video) and EAV (end of active video) words embedded within ITU-R656/D1 compliant digital video streams. A line sync signal and a frame sync signal are derived from these flags and are issued to the outside on the HSYNC and ODDEVEN/VSYNC pins in output mode. These signals are also used by the digital encoder, which treats them as incoming ODDEVEN and HSYNC signals in HSYNC + ODDEVEN based synchronization.

Autotest mode

An autotest mode is available, which causes the digital encoder to produce a color bar pattern, in the appropriate standard, independently from the video input.

The autotest mode is started by setting to 7 the 3-bit field SYNC in the register DEN_CFG0. As this mode sets the digital encoder in master mode, VSYNC/ODDEVEN and HSYNC signals are in output mode. In [Table 159](#), the decimal values of Y, Cr and Cb are shown corresponding to the autotest color bar.

Note: *The digital encoder autotest mode is used only to have a picture on the screen, even when the block(s) before the digital encoder are not working correctly. It generates the color bar pattern internally. This color bar pattern is not the standard one because only 6 bits and not 8 for each color level have been used and the transitions between the colors do not respect the video standards. For this reason the autotest color bar pattern should not be used for validation (video quality measurements).*

Table 159: Autotest colors

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 140](#) and [Figure 142](#). Both figures show the static values corresponding to the input values in [Table 159](#).

Figure 139: Luminance output levels in autotest for NTSC without set up

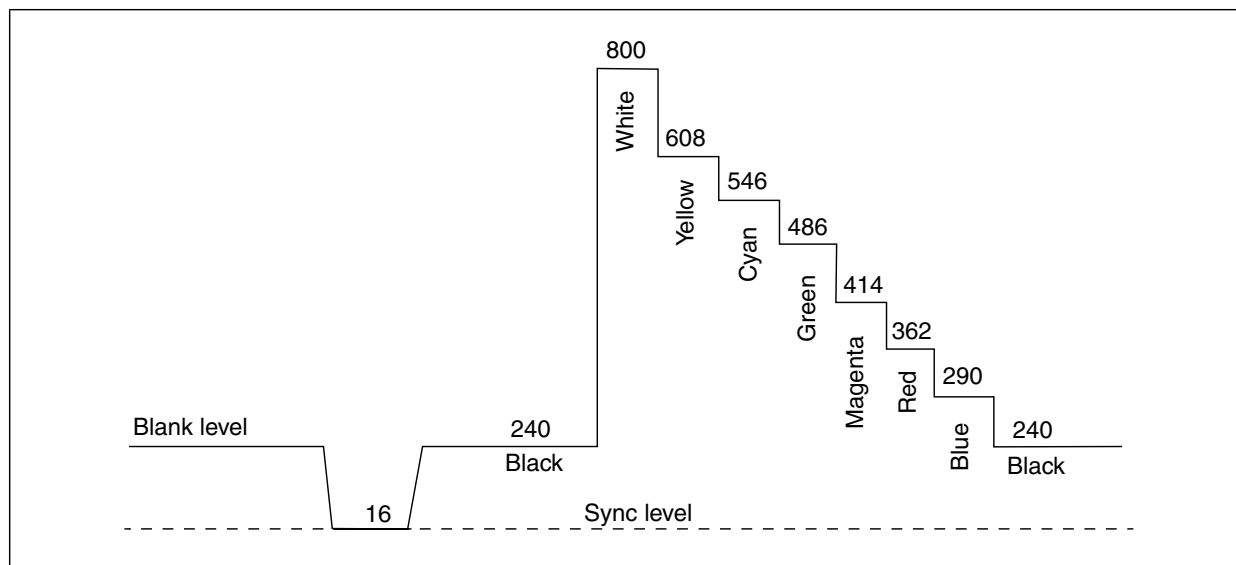
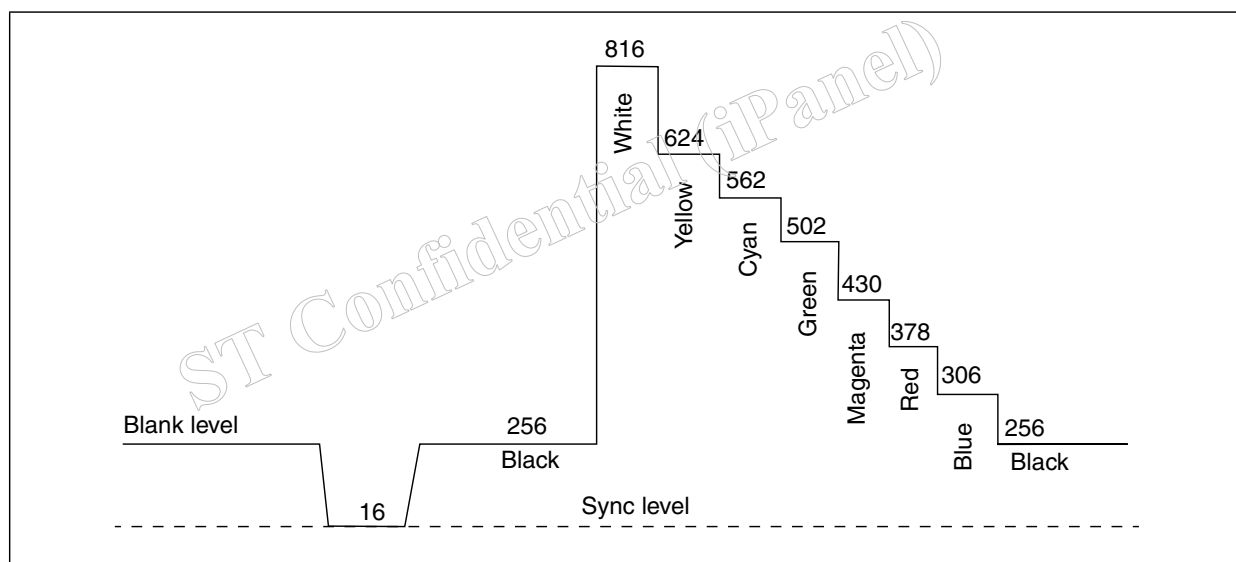


Figure 140: Luminance output levels in autotest for PAL (BGHI) and SECAM



44.5 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 the contents of configuration bits SYNCIN_AD (slave modes) or SYNCOUT_AD (master mode). Brightness, saturation and contrast are then performed on demultiplexed data, refer to registers DEN_REG_69, DEN_REG_70 and DEN_REG_71.

The ITU-R601 recommendation defines the black luma level as $Y = 16$ and the maximum white luma level as $Y = 235$. Similarly, it defines 225 quantification levels for the color difference components (Cr, Cb), centered around 128. After the saturation, brightness and contrast stage, the incoming YCrCb samples can be saturated in the input multiplexor with the following rules:

For Cr or Cb samples:	$Cr, Cb > 240$	=> Cr,Cb saturated at 240
	$Cr, Cb < 16$	=> Cr,Cb saturated at 16
For Y samples:	$Y > 235$	=> Y saturated at 235
	$Y < 16$	=> Y saturated at 16

This avoids having to heavily saturate the composite video codes before digital to analog conversion in case erroneous or unrealistic YCrCb samples are input to the encoder (there may otherwise be overflow errors in the codes driving the DACs), and therefore avoids generating a distorted output waveform.

However, in some applications, it may be desirable to let extreme YCrCb codes pass through the demultiplexor. This is controlled using bit MAXDYN in register 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 the EAV/SAV decoder.

44.6 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 base band chrominance components.

The subcarrier frequency is obtained from the following equation:

$$F_{sc} = (\text{Increment\_Word} / 2^{24}) \times CKREF$$

where *Increment_Word* is a 24-bit value.

Hard wired *Increment_Word* values are available for each standard and can be automatically selected. Alternatively (according to bit SELRST_INC in DEN_CFG5), the frequency can be fully customized by programming other values into a dedicated *Increment_Word* register, DEN_IDFS. This allows, for instance, the encoding of NTSC-4.43 or PAL-M-4.43.

This is done with the following procedure.

1. Program the required increment in DEN_IDFS.
2. Set bit SELRST_INC to 1 in register DEN_CFG5.
3. Perform a software reset using register DEN_CFG6. This sets all bits in all digital encoder registers except DEN_CFGn to their default value.

Alternatively, set DEN_CFG8 bits PH_RST_MODE[1:0] to 01. Then the frequency (and phase) update is done on the beginning of the next video line.

Note: *If a standard change occurs after the software reset, the increment value is automatically reinitialized with the hard wired or loaded value according to bit SELRST*

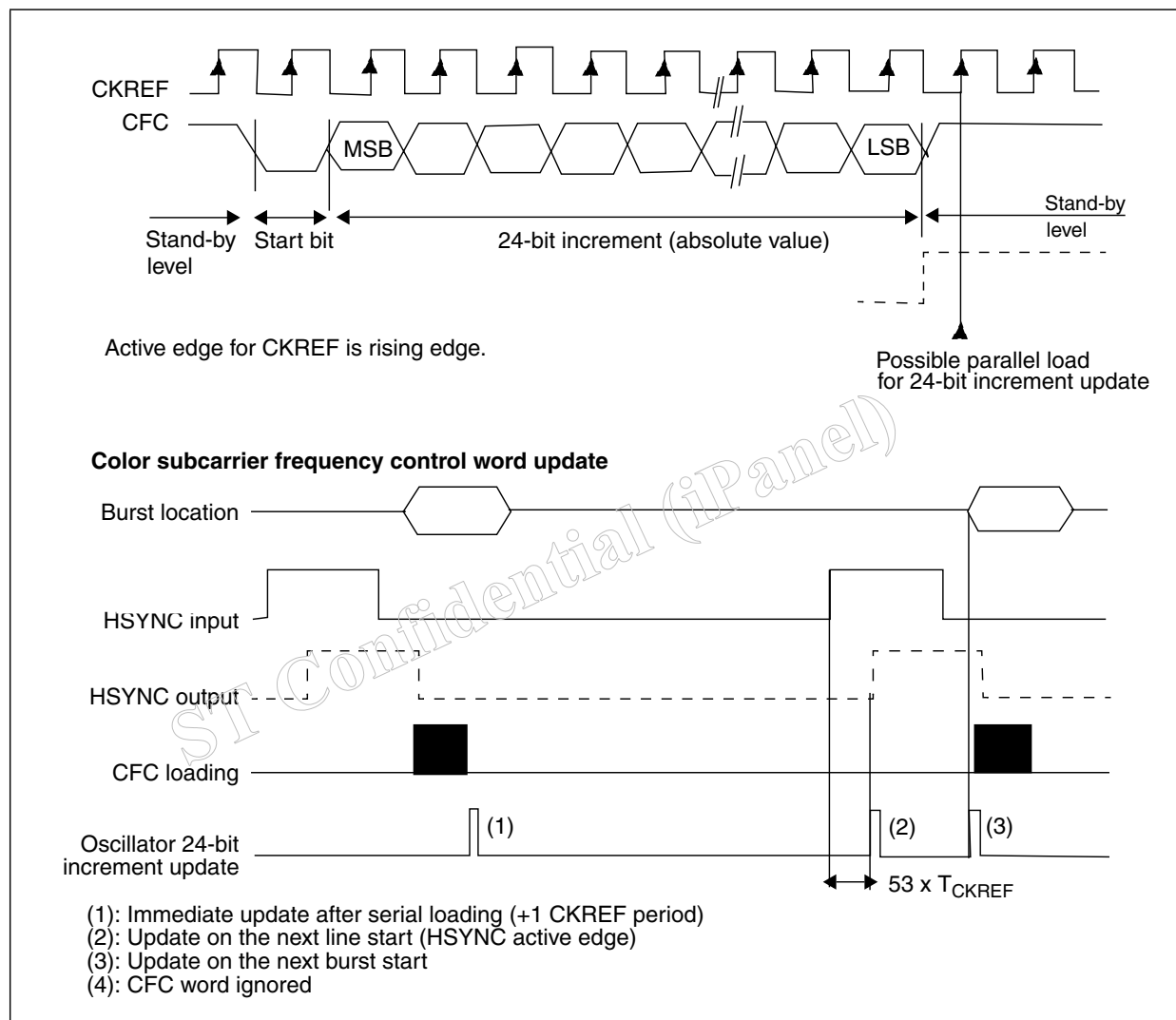
The reset phase of the color subcarrier can also be software controlled by register DEN_PDFS.

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 the PAL and NTSC subcarrier phase adjustment can be performed every line, every eight fields, every four fields, or every two fields (DEN_CFG2 bits VALRST[1:0]). If SECAM is performed, the subcarrier phase is reset every line. When performing external genlocking, the reference frequency of the regenerated clock may slightly deviate depending on the line length measurement. To prevent this drift from corrupting

the colors, the color frequency control line (CFC pin) can be used to update the 24-bit increment of the DDFS and keep the color subcarrier stable (see [Figure 141](#)).

Note: If this feature is not used, the CFC pin must be grounded. The digital encoder can also detect phase reset information coming up to 10 clock cycles after transmission of the CFC's LSB. (see DEN_CFG8 bits PH_RST_MODE[1:0]). It can also automatically reset the phase after CFC loading (see [DEN_CFG8](#) description).

Figure 141: Color subcarrier frequency control (CFC) transmission format



44.7 Burst insertion (PAL and NTSC)

The color reference burst is inserted so as to always start with a positive zero crossing of the subcarrier sine wave. The first and last half cycles have a reduced amplitude so that the burst envelope starts and ends smoothly.

The burst contains 9 or 10 sine cycles of 4.43361875 MHz or 3.579545 MHz (depending on the standard programmed in the register DEN_CFG0) as follows:

- **NTSC-M** 9 cycles of 3.579545 MHz,
- **PAL-BDGI** 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 DEN_CFG2 configuration bit BURSTEN to 0.

Burst insertion is performed by always starting the burst with a positive-going zero crossing. This guarantees a smooth start and end of burst with a maximum of undistorted burst cycles and can only be beneficial to chroma decoders.

This avoids an uncontrolled initial burst phase, and guarantees a start on a positive-going zero crossing with the consequence that two burst start locations are visible over successive lines, according to 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 $X + T_{NTSC} / 2$,

where T_{NTSC} is the duration of one cycle of the NTSC burst

after the horizontal sync tip according to the line parity.

With PAL, a similar rationale holds, and again there are two possible burst start locations. The subcarrier phase difference (with respect to the horizontal sync) from one line to the next in that case is either 0° or 180° with the 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 the horizontal sync. So two locations are possible, one for A type, the other for B type.

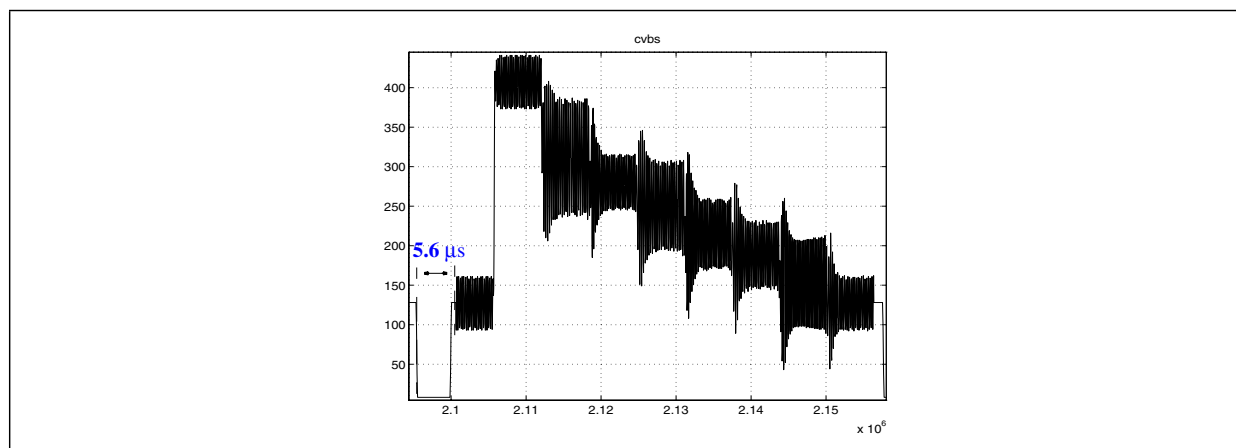
This assumes a periodic reset of the subcarrier is automatically performed (see bits VALRST[1:0] in DEN_CFG2). Otherwise, over several frames, the start of burst drifts within an interval of half a subcarrier's cycle. This is normal and means the burst is correctly locked to the colors encoded. The equivalent effect with a gated burst approach is that the start location is fixed but the burst start phase (with respect to the horizontal sync) is drifting.

44.8 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 nonmodulated subcarrier is shown in the [Figure 142](#).

Figure 142: 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 , ... comparing to a sine wave starting at the same point - 5.6 μ s after a horizontal synch pulse (inverted on one line out of every three and also at each frame). This sequence begins from line 1 or line 23 of the first field (see GEN_SECAM bit of register [DEN_CFG7](#)). 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 sequence of subcarrier is always inverted with respect to the odd field one.

To enable SECAM mode, program 1 in DEN_CFG7.7 (MSB) and then perform a soft reset or loading of DEN_CFG0.

44.9 Luminance encoding

The demultiplexed Y samples are band limited and interpolated at CKREF clock rate. The resulting luminance signal is properly scaled before insertion of any closed captions, CGMS, VPS, teletext or WSS data and synchronization pulses.

The interpolation filter compensates for the $\sin(x)/x$ attenuation inherent in digital to analog conversion and greatly simplifies the output stage filter. See [Figure 143](#), [144](#) and [145](#) for characteristic curves.

In addition, the luminance that is added to the chrominance to create the composite CVBS signal can be trap filtered at 3.58 MHz (NTSC) or 4.43 MHz (PAL). This supports applications oriented towards low end TV sets which are subject to cross color if the digital source has a wide luminance bandwidth (for example, some DVD sources). Note that the trap filter does not affect the S-VHS luminance output nor the RGB outputs. If SECAM is performed, enable the trap filter with 4.43 MHz cut off frequency on the luma part of the CVBS signal (see DEN_CFG3 bits ENTRAP and TRAP_4.43).

A 7.5 IRE pedestal can be programmed if needed with all standards (see registers [DEN_CFG1](#) and [DEN_CFG7](#)). This allows in particular to encode Argentinian and non-Argentinian PAL-N, or Japanese NTSC (NTSC with no set up).

Figure 143: Luma filtering including DAC attenuation

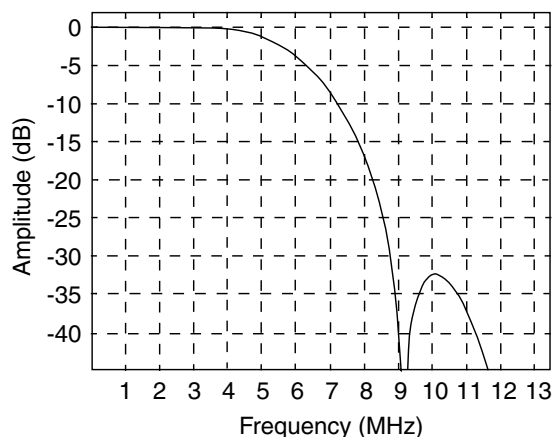


Figure 144: Luma filtering with 3.58 MHz trap, including DAC attenuation

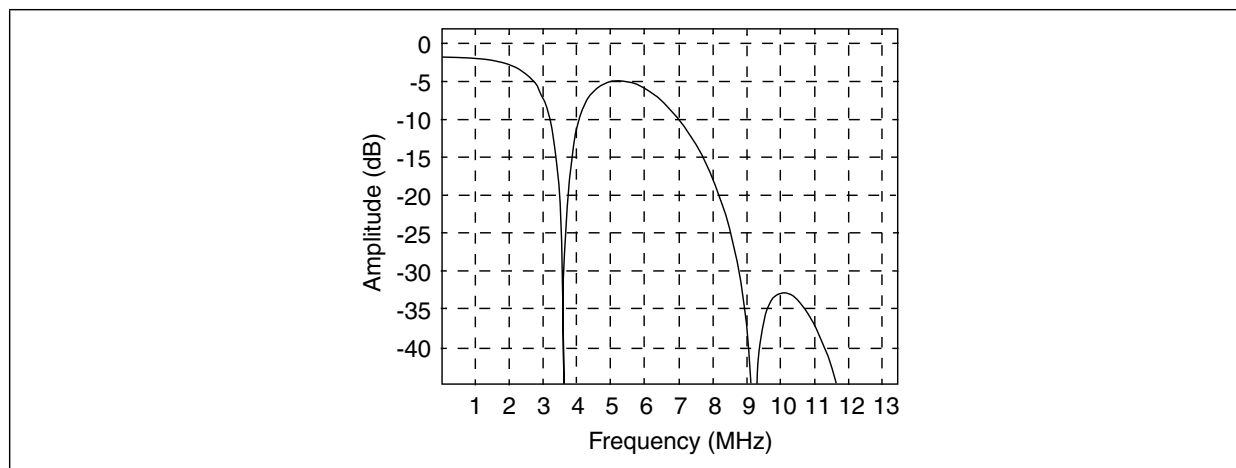
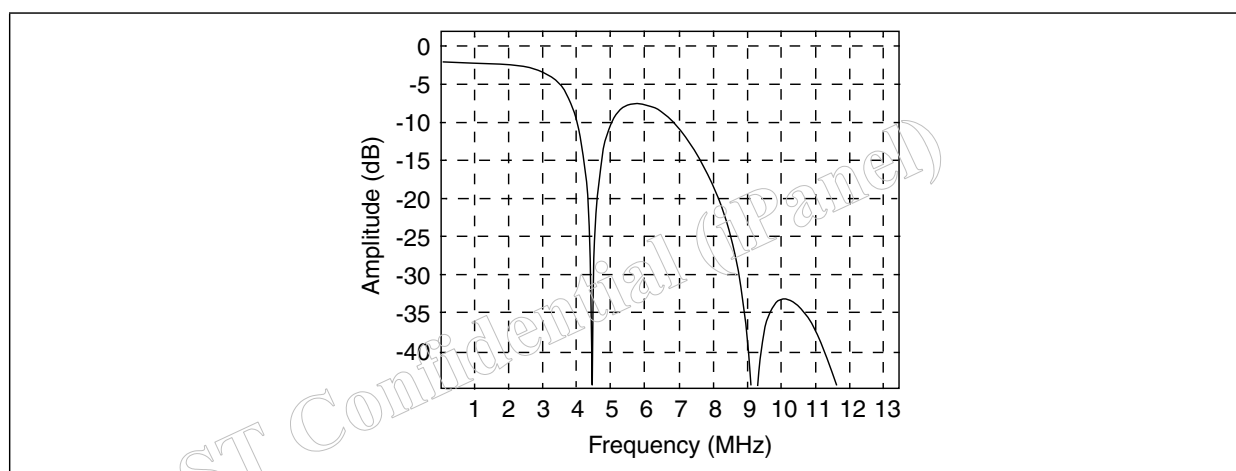


Figure 145: Luma filtering with 4.43 MHz trap, including DAC attenuation



The luma channel has a 19th order filter with coefficients programmable by registers DEN_LFCOE[0:9]. This filter is described in [Section 44.10: Chrominance encoding on page 421](#).

Register bit FLT_YS (DEN_CDEL_LFC) selects either the register or the hard wired values for the filter coefficients.

The luma processing as well as line and field timings in SECAM mode are identical to PAL BDGHI ones.

44.10 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 CKREF clock rate. This processing eases the filtering following digital to analog conversion and allows more accurate encoding.

A set of four different filters is available in PAL and NTSC for chroma filtering to fit a wide variety of applications in the different standards and include filters recommended by ITU-R 624-4 and SMPTE170-M.

The available 3 dB bandwidths are 1.1, 1.3, 1.6 or 1.9 MHz. See [Figure 148](#), [149](#), [150](#), [151](#) and [152](#) for the various frequency responses and register [DEN_CFG1](#) for programming. The narrower bandwidths are useful against cross luminance artifacts, the wider bandwidths allow higher chroma contents.

Alternatively, a filter with programmable coefficients (see registers [DEN_CFCOEF\[0:8\]](#)) can be used. This is necessary when other clock frequencies are used (24.545454 and 29.5 MHz clocks in square pixel mode), or for specific applications where another frequency response is needed. The order of chroma filter is 17, with symmetric coefficients and works at frequency CKREF (default 27 MHz). If the 4:2:2 input format is used on CVBS and S-VHS outputs, then the first upsampling by 2 (6.75 to 13.5 MHz sampling frequency with 27 MHz CKREF) is done by repeating the samples (10 20 30... at 6.75 MHz gives 10 10 20 20 30 30... at 13.5 MHz clock frequency). This is equivalent to filtering by $\cos(\pi \times f / (2 \times f_{ck}))$, where f_{ck} is the CKREF clock frequency. The second upsampling (13.5 to 27 MHz with 27 MHz CKREF) is done by padding with zeros.

Register bit FLT_S (DEN_CFCOEF0) selects either the register or the hard wired values for the filter coefficients.

In SECAM, 1.3 MHz low pass and pre-emphasis filtering are performed on Dr and Db chroma components, before the frequency modulation, according to ITU-R Rec624-4.

Refer to [Figure 146](#) for frequency response of these filters. Bell filtering is performed at the end of frequency modulation stage.

Peak to peak amplitude of modulated chrominance signal at the central frequency (4 279.7 kHz) is 22,88% of the black/white interval (22.88 IRE).

The chrominance path can be delayed, with respect to luma path, for S-VHS and CVBS outputs. Register bit DEL_EN (DEN_CFG3) selects either the default delays or the programmable delay. The default delays are preprogrammed for the PAL, SECAM and NTSC modes in 4:2:2 and 4:4:4 format on CVBS.

Refer to [Figure 147](#) for frequency response of bell filter with subcarrier frequencies and clipping values.

Figure 146: SECAM chroma filtering (pre-emphasis and 1.3 MHz low pass filtering)

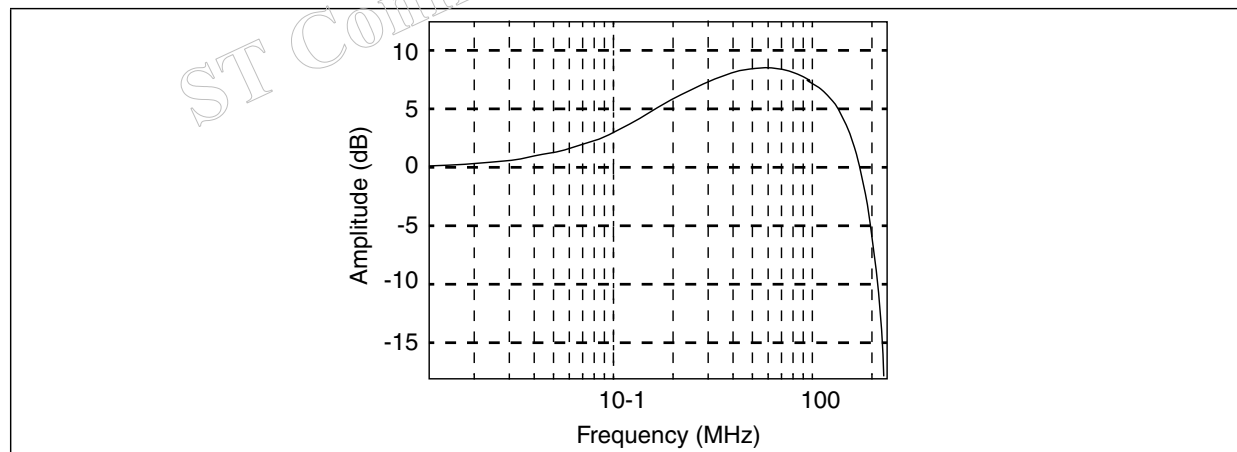


Figure 147: SECAM high frequency subcarrier pre-emphasis (Bell filtering), including DAC attenuation

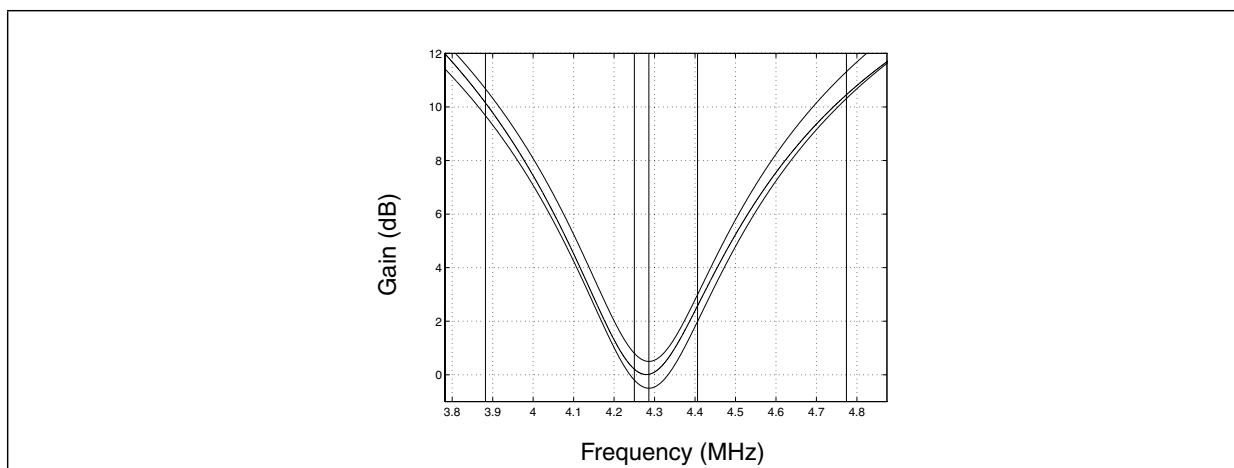
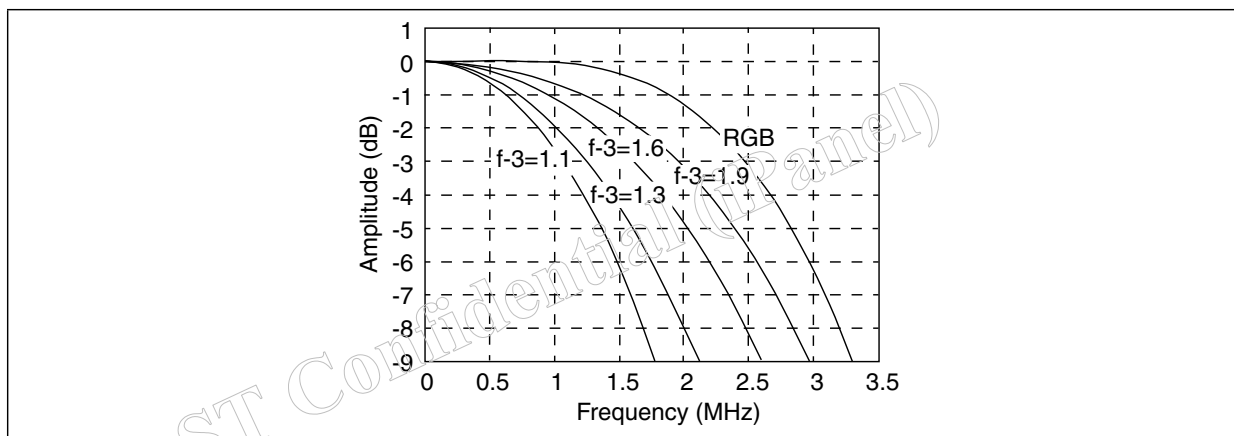


Figure 148: Various chroma filters available and RGB filter



44.11 Composite video signal generation

The composite video signal is created by adding the luminance (after trap filtering, optional in PAL and NTSC, see register [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 graphics engines.

A color killing function is available, whereby the composite signal contains no chrominance, that is, it replicates the trap filtered luminance. This function does not suppress the chrominance on the S/VHS outputs, but suppressing the S-VHS chrominance is possible using bit BKDACn in [DEN_CFG5](#), where the chrominance signal is outputted on DAC n.

Figure 149: 1.1 MHz chroma filter

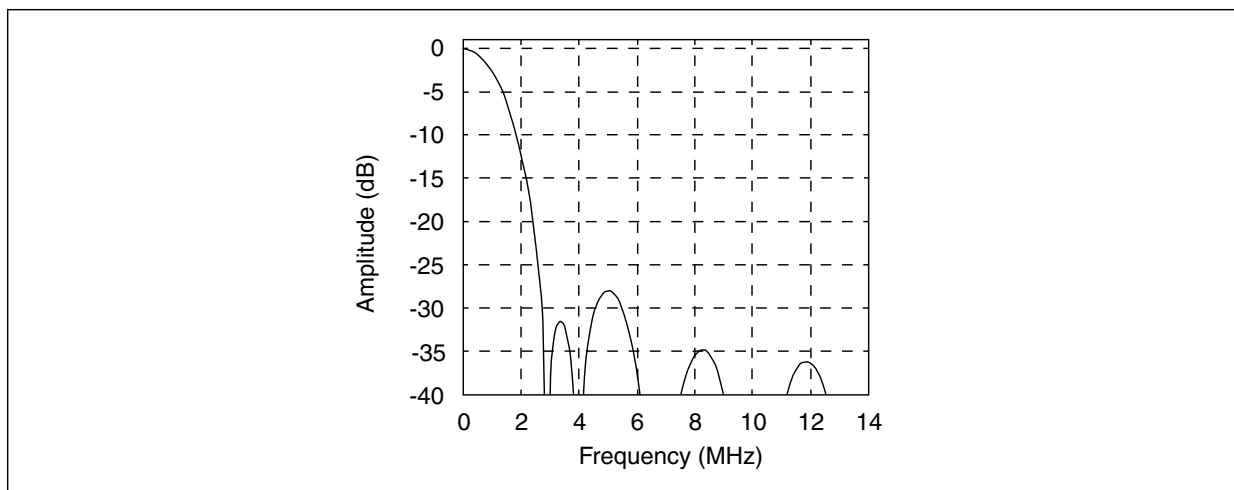


Figure 150: 1.3 MHz chroma filter

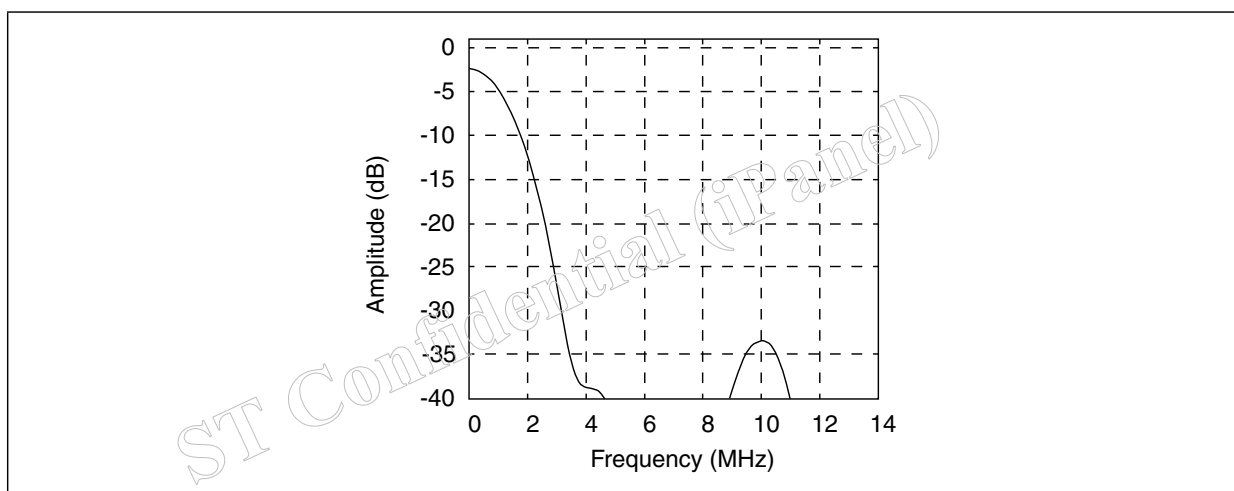


Figure 151: 1.6 MHz chroma filter

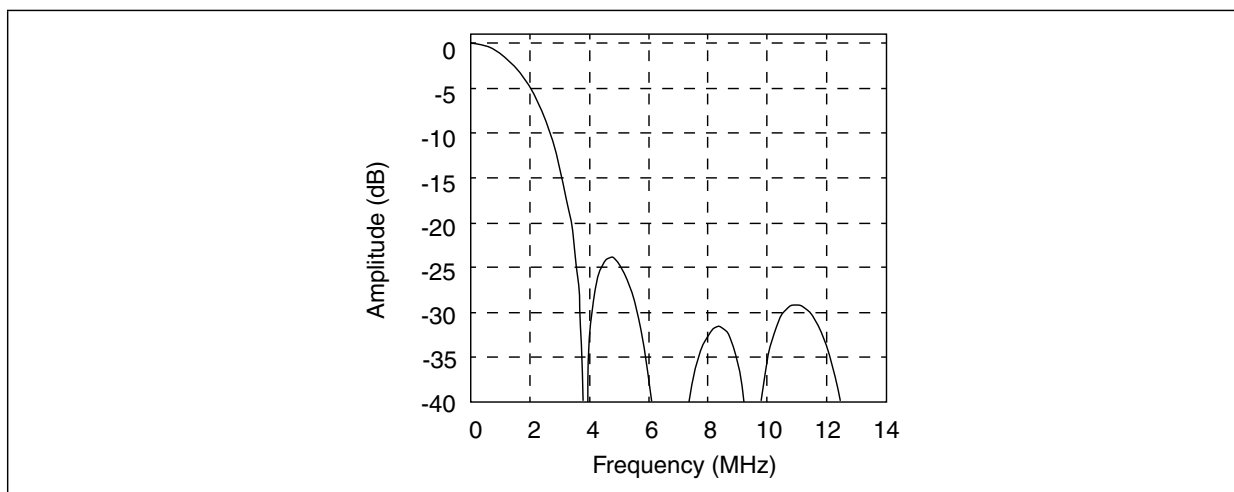
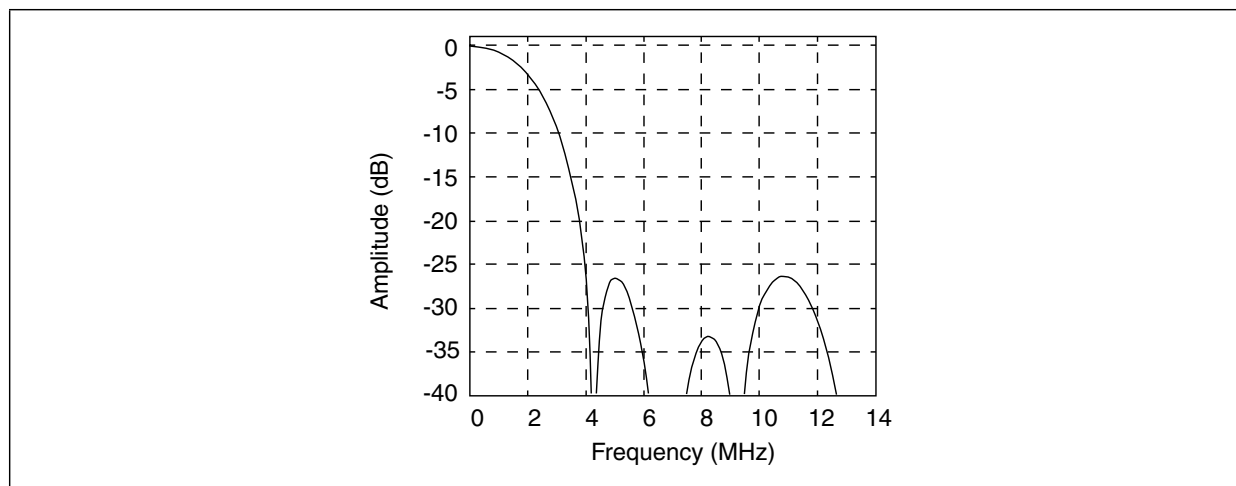


Figure 152: 1.9 MHz chroma filter

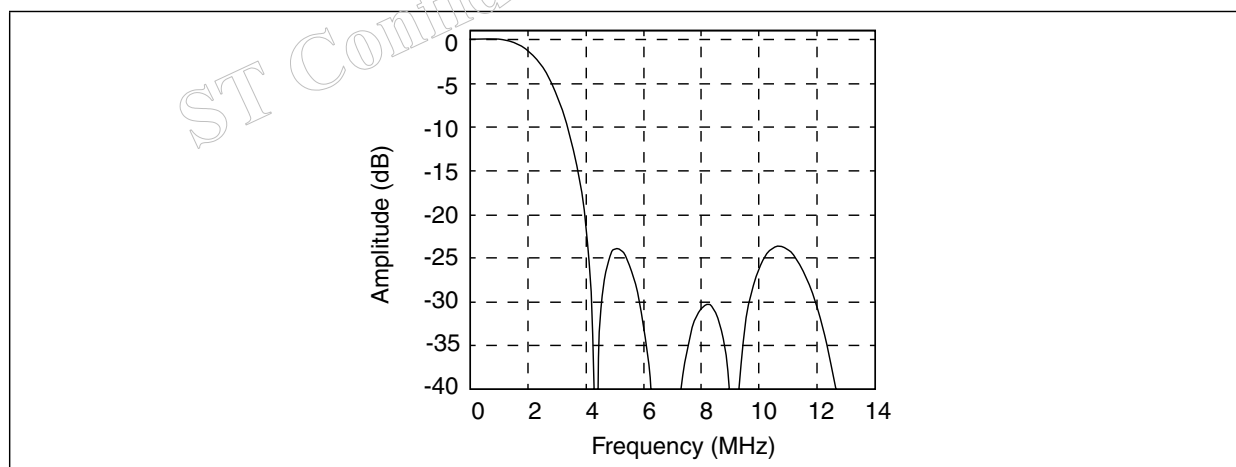


44.12 RGB and UV encoding

After demultiplexing, the Cr and Cb samples feed a 4 times interpolation filter. The resulting base band chroma signal has a 2.45 MHz bandwidth (Figure 153) and is combined with the filtered luma component to generate R, G, B or U, V samples at 27 MHz.

If Y4 and CrCb inputs are used, the filtering identical to luma filtering (see Figure 143) is performed on all components (Y4, Cr and Cb). In this case DAC5 output data encoded from Y4 input if YUV configuration is used (see DEN_CFG8 bits CONF_OUT1 and CONF_OUT0).

Figure 153: RGB chroma filtering



44.13 Closed captioning

Closed captions (or data from an extended data service as defined by the closed captions specification) can be encoded by the circuit. The closed caption data is delivered to the circuit through the register interface. Two dedicated pairs of bytes (2 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 digital encoder.

Closed caption data registers are double buffered so that loading can be performed anytime, even during line 21/284 or any other selected line.

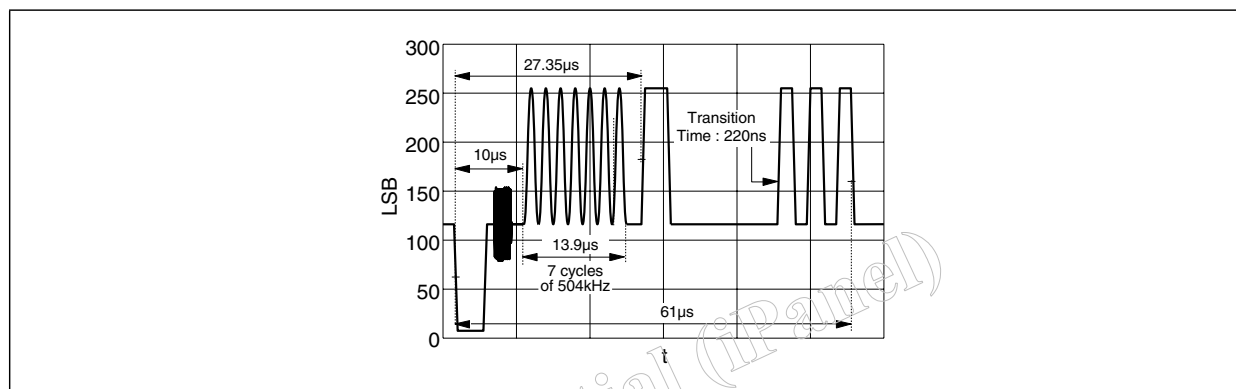
User register DEN_CCF1 and DEN_CCF2 each contain the first and second byte to send (LSB first) after the start bit on the appropriate TV line, where DEN_CCF1 refers to field 1 and DEN_CCF2 to field 2. The TV line number where data is to be encoded is programmable using

registers DEN_CLF1 and DEN_CLF2. Lines that may be selected include those used by the StarSight® data broadcast system. Closed caption data has priority over any CGMS signals programmed for the same line.

The internal clock run in generator is based on a direct digital frequency synthesizer. The nominal instantaneous data rate is 503.496 kHz (that is 32 times the NTSC line rate). Data low corresponds nominally to 0 IRE, data high corresponds to 50 IRE at the DAC outputs.

When closed captioning is on (bits CC1 and CC2 in DEN_CFG1), the CPU should load the relevant registers (DEN_CCF1 or DEN_CCF2) once every frame at most (although there is in fact some margin due to the double buffering). Two bits are set in the DEN_STA register in case of attempts to load the closed caption data registers too frequently; these can be used to regulate the loading rate.

Figure 154: Example of closed caption waveform



The closed caption encoder considers that closed caption data has been loaded and is valid on completion of the write operation into DEN_CCF1 for field1, or DEN_CCF2 for field 2. If closed caption encoding has been enabled and no new data bytes have been written into the closed caption data registers when the closed caption window starts on the appropriate TV line, then the circuit outputs two US-ASCII NULL characters with odd parity after the start bit.

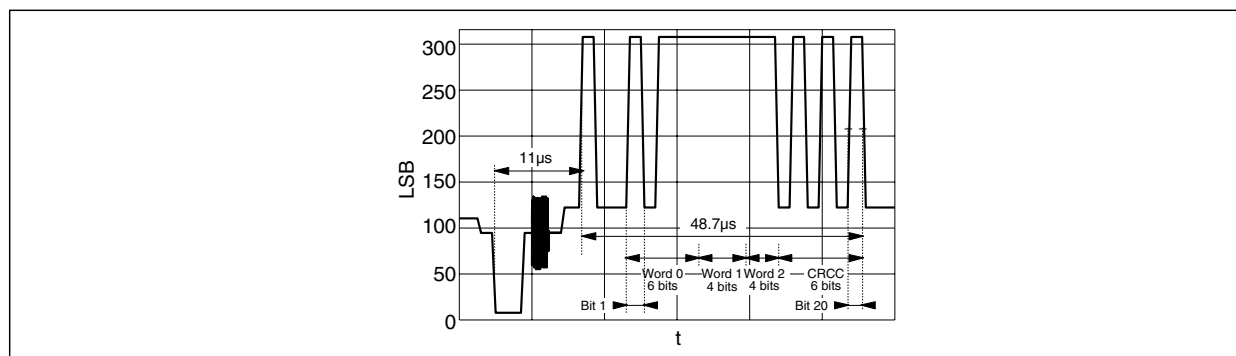
44.14 CGMS encoding

CGMS stands for copy generation management system, and is also known as VBI and described by standard CPX-1204 of EIAJ. CGMS data can be encoded by the digital encoder.

Three bytes, containing twenty significant bits, are delivered to the chip via the register interface. Two reference bits (1 then 0) are encoded first, followed by 20 bits of CGMS data. This includes a cyclic redundancy check sequence, which is not computed by the device but is supplied to it as part of the 20 data bits. The reference bits are generated locally by the digital encoder.

Figure 155 shows a typical CGMS waveform.

Figure 155: Example of CGMS waveform



CGMS encoding is enabled by setting bit ENCGMS in register DEN_CFG3. When enabled, the CGMS waveform is present once in each field, on lines 20 and 283 (SMPTE-525 line numbering).

The CGMS data register is double buffered, which means that it can be loaded at any time (even during line 20/283) without any risk of corrupting CGMS data that could be in the process of being encoded. The CGMS encoder considers that new CGMS data has been loaded and is valid on completion of the write operation into register DEN_CGMS[1:3].

44.15 WSS encoding

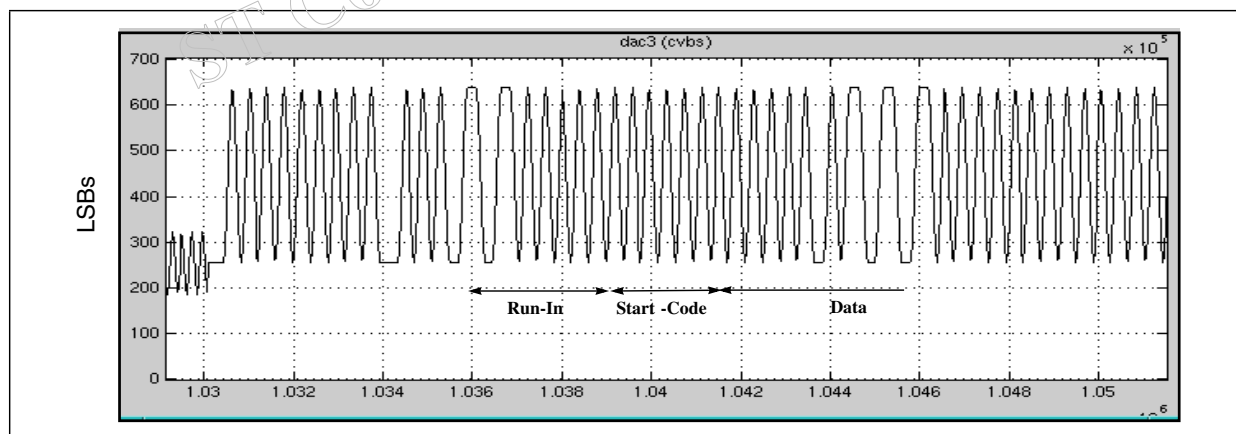
The digital encoder allows WSS (wide screen signalling) in 625 line format, complying with the ETS 300 294 standard. Two bytes are delivered to the circuit through the register interface into two dedicated registers (see register DEN_WSS[1:2]).

WSS encoding is enabled using bit ENWSS in register DEN_CFG3. When WSS encoding is enabled, a waveform is present on the first half of line 23 of each frame. Data is preceded by a run in sequence and a start code generated locally by the digital encoder.

44.16 VPS encoding

VPS data encoding is defined by *ETS 300 231 communication, June 1993*. VPS data can be encoded by the digital encoder on line 16 (CCIR) for 625 line PAL and SECAM television systems. The VPS data is delivered to the circuit using registers DEN_VPS1. The data transmission is preceded by a clock run in and a start code generated by the digital encoder. The clock frequency is 5 MHz. This feature is enabled by setting the ENVPS bit of register DEN_CFG7. Figure 156 shows an example of VPS waveform.

Figure 156: Example of VPS waveform



44.17 Teletext encoding

The digital encoder can encode teletext according to the CCIR/ITU-R broadcast teletext system B specification (also known as world system teletext), and NABTS (North American basic teletext specification) EIA-516.

In DVB applications, teletext data is embedded within DVB streams as MPEG data packets. The transport layer processing IC (ST20) sorts incoming data packets and stores teletext packets in a buffer. It then passes them to the digital encoder on request.

44.17.1 Signal exchange

The digital encoder and the teletext buffer exchange two signals, TTXS (teletext synchronization) going from the digital encoder to the teletext buffer and TTXD (teletext data) going from the teletext buffer to the digital encoder.

The TTXS signal is a request signal generated on selected lines. In response to this signal, the teletext buffer is expected to send teletext bits to the digital encoder for insertion of a teletext line into the analog video signal. The number of teletext bits sent, depends on the teletext system being used (selected by register bit TTXD_ABCD (DEN_REG_64)) 360 bits are sent for teletext B, WST in PAL and SECAM, or 288 for Teletext C, NABTS in NTSC.

The duration of the TTXS window corresponds to the number of bits being sent (see Transmission Protocol below).

- For Teletext B and 625 line systems, the TTXS window duration is 1402 reference clock periods (corresponding to 360 bits).
- For Teletext C and 525 line systems (NABTS), this duration is 1121 master clock periods.

Following the TTXS rising edge, the encoder expects data from the teletext buffer after a programmable number (two to nine) of 27 MHz master clock periods. Data is transmitted synchronously with the master clock at an average rate of 6.9375 Mbit/s according to the protocol described below. In order of transmission, it consists of 16 clock run in bits, 8 framing code bits and one teletext packet of 336 or 228 bits (depending on the teletext system being used). If more than one packet of bits (336 or 228) are transmitted, they are ignored by the digital encoder. By default, register bit TTX_MASK_OFF (DEN_REG_65) masks the two bits of teletext framing code, allowing the code to be set by the digital encoder according to the selected teletext standard.

44.17.2 Transmission protocol

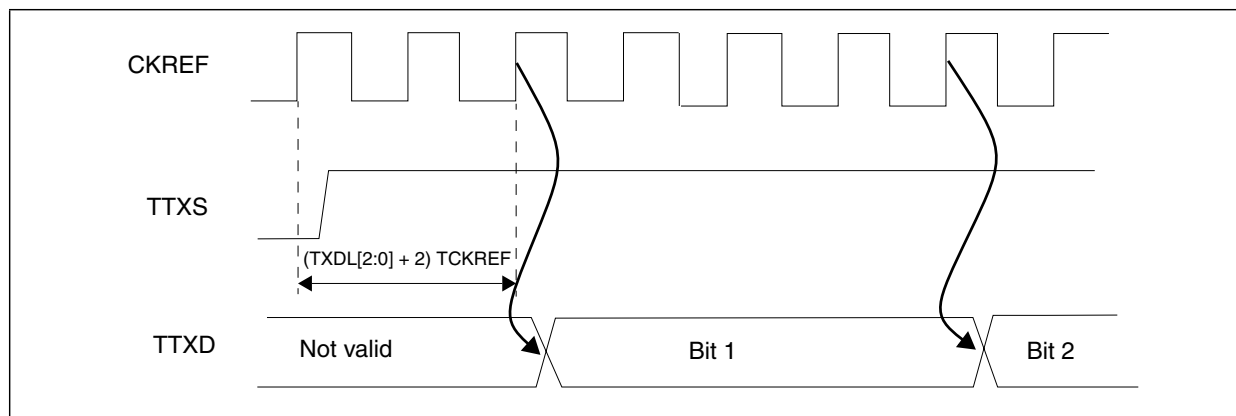
In order to transmit the teletext data bits at an average rate of 6.9375 Mbit/s, which is about 1 / 3.89 times the master clock frequency, the following scheme is adopted.

The 360-bit packet is regarded as nine 37-bit sequences plus one 27-bit sequence. In every sequence, each teletext data bit is transmitted as a succession of four identical samples at 27 Msample/s, except for the 10th, 19th, 28th and 37th bits of the sequence which are transmitted as a succession of three identical samples.

44.17.3 Programming TTXS rising to first valid sample

The encoder expects the teletext buffer to clock out the first teletext data sample on the $(2 + n)^{\text{th}}$ rising edge of the master clock following the rising edge of TTXS. [Figure 157](#) depicts this graphically for $n = 0$.

Figure 157: TTX rising to first valid sample delay for TXDL[2:0] = 0



N is programmable from 0 to 7 by register bits TTXDEL[2:0] (DEN_TTX1). The value written in TXDL[2:0] is two less than the overall delay in CKREF cycles, so a value of 0 for TXDL[2:0] corresponds to an overall delay of two cycles, and a value of seven corresponds to a delay of nine cycles.

44.17.4 Programming teletext line selection

Five dedicated registers, DEN_TTX[1:5], program teletext encoding in various lines in the vertical blanking interval (VBI) of each field. In this way, each line in VBI can be selected independently.

Full page teletext encoding is set by register bit FP_TTXT (DEN_TTX1). teletext is encoded on lines 24 to 311 and 336 to 623 (ITU-R line numbering). This is in addition to the lines already programmed in the VBI. When full page teletext is performed, no video data is encoded (YCrCb, Y4 and CrCb input streams are ignored).

44.17.5 Teletext pulse shape

The shape and amplitude of a single teletext pulse is shown in Figure 158. Its relative power spectral density is shown in Figure 159 and Figure 160. It is zero at frequencies above 5 MHz, as required by the world system teletext specification.

Figure 158: Shape and amplitude of a single teletext symbol

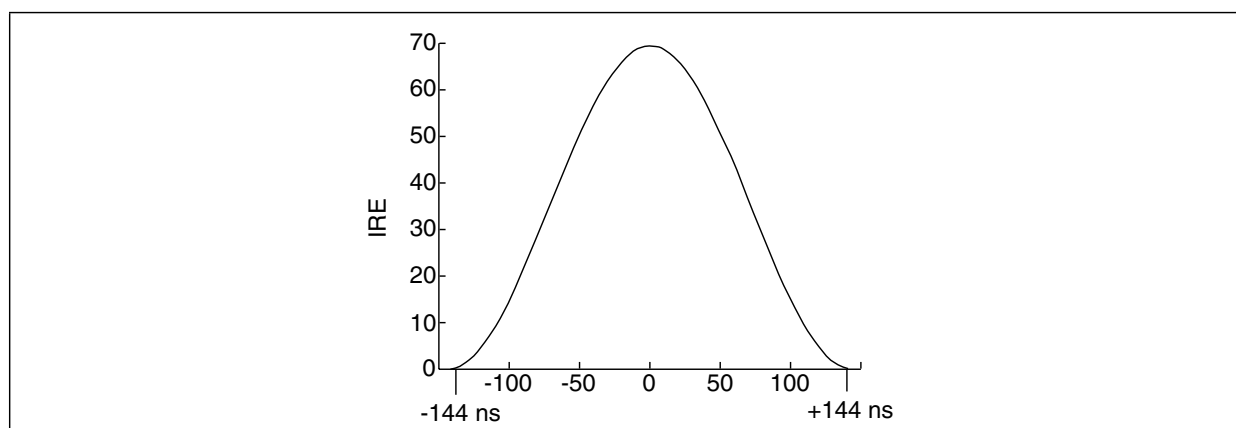


Figure 159: Linear PSD scale

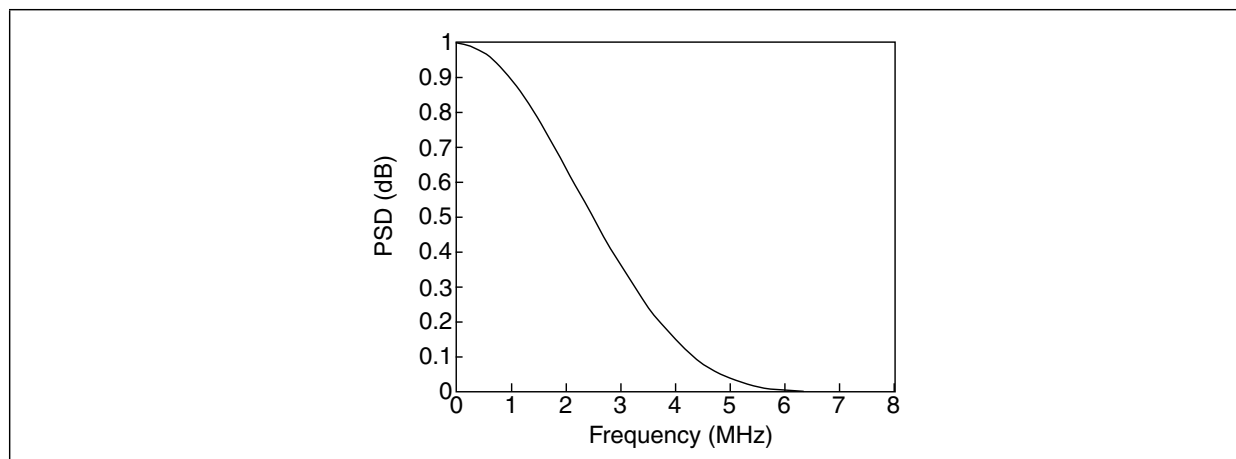
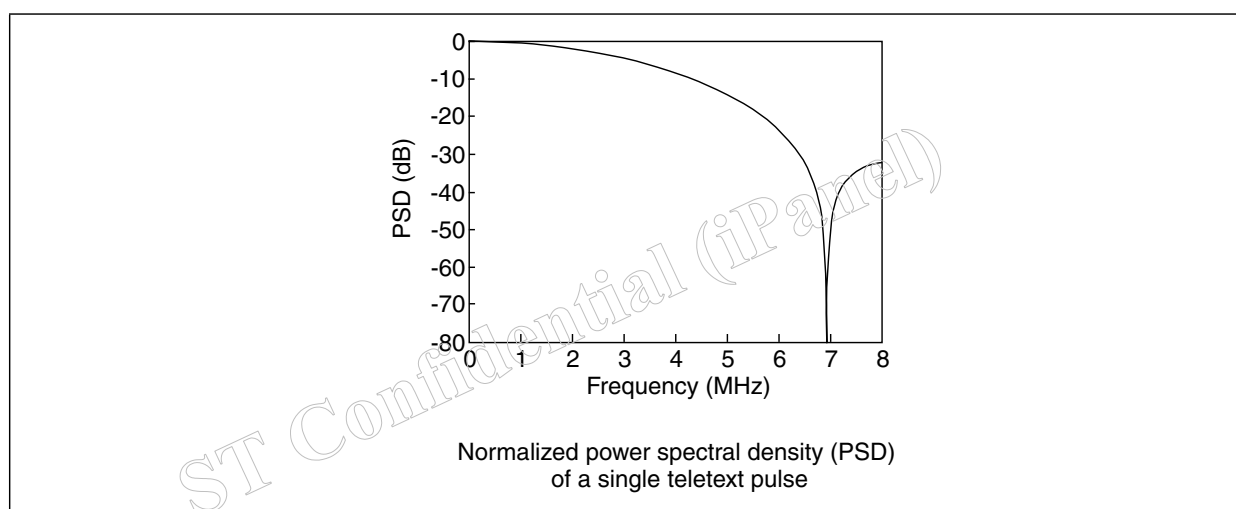


Figure 160: Logarithmic PSD scale



44.18 CVBS, S-VHS, RGB and UV outputs

Six out of eight video signals can be directed to six analog output pins through a 10-bit digital to analog converters operating at the reference clock frequency. The available combinations are:

S-VHS (Y/C) + CVBS + RGB, or S-VHS (Y/C) + CVBS + U + Y2 + V, or Y1 + C1 + CVBS1 + C2 + Y2 + CVBS2.

These combinations are controlled by bits CONF_OUT1 and CONF_OUT0 in register DEN_CFG8, as shown in [Table 160](#).

Table 160: Encoding of CONF_OUT

CONF_OUT1	CONF_OUT0	DAC1	DAC2	DAC3	DAC4	DAC5	DAC6	Notes
0	0	Y	C	CVBS	C	Y	CVBS	
0	1	Y	C	CVBS	V	Y	U	
1	x	Y	C	CVBS	R	G	B	Default

The register bits SET4:4:4 (DEN_CFG3) and SET444_CVBS (DEN_CDEL_LFC) select the outputs CVBS/S-VHS or RGB/YUV for each of the two encoded input formats.

The C to Y peak to peak amplitude ratio can be modified in both CVBS and VHS (Y/C) outputs (see MULT_RGB_C).

Default peak to peak amplitude of UV and RGB outputs is set to 70% of Y or CVBS peak to peak amplitude, for 100/0/100/0 color bar pattern, and can be modified using the multiplying factors in registers DEN_DAC45 and DEN_DAC6C.

If DEN_CFG7[2] bit UV_LEV is 0 (default value) U and V outputs have 0.7 V peak to peak amplitude if 100/0/100/0 color bar pattern is inputted. If this bit is 1 U and V outputs are those defined by ITU-R 624-4 for PAL and NTSC standards ($V_{pp}/U_{pp} = 1.4$). In that case U peak to peak amplitude is 0.61 V (0.57 V if DEN_CFG7 bit SETUPYUV is set) and V peak to peak amplitude is 0.86 V (0.80 V if register DEN_CFG7 bit SETUPYUV is set). In all these cases UV outputs can be multiplied by 0.75 to 1.22 factor according to register bits DAC4_MULT (DEN_DAC45) and DAC6_MULT (DEN_DAC6C).

A single external analog power supply pair is used for all DACs, but two independent pairs of current and voltage references are needed. A 10 Kbytes resistor must be connected between each IREF and VREF pin.

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 triple DAC should be disabled, pin IREFDACRGB should be tied to analog ground and pin VREFDACRGB should be left unconnected.

Due to the 3.3 V power supply used, the output swing of the DACs is about 1 V_{p-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 bit of DEN_CFG5. Code N becomes code 1024-N, that is, the resulting waveform undergoes a symmetry around the midswing code.

45 Digital encoder registers

The main functions are controlled using a 32-bit register interface with the CPU. In all digital encoder registers bits 31 to 8 are reserved.

Addresses are provided as the *DENCBASEAddress* + offset.

The *DENCBASEAddress* is:

0x0000 1800.

A register summary is given in [Table 35: Digital encoder registers on page 59](#).

DEN_CFG0

Configuration0

7	6	5	4	3	2	1	0
STD1	STD0	SYNC2	SYNC1	SYNC0	POLH	POLV	FREERUN

Address: *DENCBASEAddress* + 0x000

Type: Read/write

Reset: 1001 0010

Description:

[31:8] **Reserved**

[7:6] **STD[1:0]:** Standard

00: PAL BDGHI

10: NTSC M ^{a b}

01: PAL N (see bit setup)

11: PAL M

[5:3] **SYNC[2:0]:** Configuration

000: ODDEVEN-only based slave mode (frame locked)

001: F based slave mode (frame locked)

010: ODDEVEN + HSYNC based slave mode (line locked)

011: F+ H based slave mode (line locked)

100: VSYNC-only based slave mode (frame locked) ^c

101: VSYNC + HSYNC based slave mode (line locked)

110: Master mode

111: Autotest mode (color bar pattern)

[2] **POLH:** Synchro: Active edge of HSYNC selection (when input) or polarity of HSYNC (when output)

0: HSYNC is a negative pulse (128 TCKREF wide) or falling edge is active

1: HSYNC is a positive pulse (128 TCKREF wide) or rising edge is active

[1] **POLV:** Synchro: Active edge of ODDEVEN/VSYNC selection (when input)

0: Falling edge of ODDEVEN flags start of field1 (odd field) or VSYNC is active low

1: Rising edge of ODDEVEN flags start of field1 (odd field) or VSYNC is active high

[0] **FREERUN**

Bit FREERUN is taken into account in ODDEVEN-only, VSYNC-only or F-based slave modes. It is irrelevant to other synchronization modes.

Synchro: Active edge of ODDEVEN/VSYNC selection (when input)

0: Disabled

1: Enabled

- 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, the STD1 and STD0 bits are not taken into account.
- In VSYNC-only based slave mode (SYNC[2:0]: 100), HSYNC is nevertheless needed as an input.

DEN_CFG1

Configuration1

7	6	5	4	3	2	1	0
BLKLI	FLT1	FLT0	SYNCOK	COKI	SETUP	CC2	CC1

Address: *DENCBASEAddress* + 0x001

Type: Read/write

Reset: 0100 0100

Description:

[7] **BLKLI**^a

Vertical blanking interval selection for active video lines area.

0: (Partial blanking) Only the following lines inside the vertical interval are blanked:^b

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 programming 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)^b

11: f-3dB = 1.9 MHz high definition PAL filter: Rec 624 - 4 for PAL BDG/I compliant

[4] **SYNCOK**

Sync signal availability (analog and digital) for input synchronization loss with FREERUN inactive (FREERUN = 0)

0:^b No synchro output signals

1: Output synchro signals available on YS, CVBS and, when applicable, HSYNC (if output port), ODDEVEN (if output port),: 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^b

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 BKDAC2 ([DEN_CFG5](#)).

[2] **SETUP**

Pedestal enable

0: Blanking level and black level are identical on all lines (for example, Arg. PAL-N, Japan NTSC-M, PAL-BDGH)^b

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^b

01: Closed caption/extended data encoding enabled in field 1 (odd)

10: Closed caption/extended data encoding enabled in field 2 (even)

11: Closed caption/extended data encoding enabled in both fields

a. BLKLI must be set to 0 when closed captions are to be encoded on the following lines:

in 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)

b. Default mode when DENC_NRST pin is active (low level)

DEN_CFG2

Configuration2

7	6	5	4	3	2	1	0
Reserved	ENRST	BURSTEN	SET4:4:4	SELRST	RSTOSC_BUF	VALRST1	VALRST0

Address: *DENCBaseAddress* + 0x002

Type: Read/write

Reset: 0010 0000

Description:

[7] **Reserved**

[6] **ENRST**

Cyclic update of DDFS phase

0: No cyclic subcarrier phase reset^a

1: Cyclic subcarrier phase reset depending on VALRST1 and VALRST0 (see above)

[5] **BURSTEN**

Chrominance burst control

0: Burst is turned off on CVBS, chrominance output is not affected

1: Burst is enabled^a

[4] **SET4:4:4**

Selects the 4:2:2 or 4:4:4 video input from the mixing unit

0: YCrCb (4:2:2) input:^a

The presence of OSD on the digital encoder output can be selected or deselected by OSD block header field S.

1: YCrCb (4:4:4) input: In this mode OSD is always present on the digital encoder output.

[3] **SELRST**

Selects reset values for direct digital frequency synthesizer

0: Hardware reset values for subcarrier oscillator phase (see [DEN_PDFS\[1:2\]](#) register descriptions for values)^a

1: Loaded reset values selected (see contents of the [DEN_PDFS\[1:2\]](#) registers)

[2] **RSTOSC_BUF^b**

Software phase reset of DDFS (direct digital frequency synthesizer) buffer

0: No reset^a

1: When a 0 to 1 transition occurs, either the hard wired default phase value, or the value loaded in the [DEN_PDFS\[1:2\]](#) register (according to bit SELRST), is put to phase buffer. This value is loaded into accumulator (phase of subcarrier) when bits PH_RST_OSC from [DEN_CFG8](#) are programmed, or when the standard changes or soft reset occurs.

[1:0] **VALRST[1:0]^c**

00: Automatic reset of the oscillator every line^a

01: Automatic reset of the oscillator every 2nd field

10: Automatic reset of the oscillator every 4th field

11: Automatic reset of the oscillator every 8th field

a. Default mode when DENC_NRST pin is active (low level)

b. RSTOSC_BUF is automatically set back to 0 after the buffer is loaded

c. VALRST[1:0] is taken into account only if bit ENRST is set. Resetting the oscillator means here forcing the value of the phase accumulator to its nominal value to avoid accumulating errors due to the finite number of bits used internally. The value to which the accumulator is reset is either the hard wired default phase value or the value loaded in the [DEN_PDFS\[1:2\]](#) registers (according to bit SELRST), to which a 00, 900, 1800, or 2700 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

Configuration3

7	6	5	4	3	2	1	0
ENTRAP	TRAP_4.43	ENCGMS	CK_IN_PHASE	DEL_EN	YCRCB_REF_CLK_S	Reserved	ENWSS

Address: *DENCBaseAddress* + 0x003

Type: Read/write

Reset: 00000000

Description:

- [7] **ENTRAP**^a
Enable trap filter
0: Trap filter disabled^a
1: Trap filter enabled
- [6] **TRAP_4.43**
Enable trap filter: TRAP_4.43 is taken into account only if bit ENTRAP is set
0: Select the trap filter centered around 3.58 MHz^a
1: Select the trap filter centered around 4.43 MHz
- [5] **ENCGMS**^b
CGMS encoding enable
0: Disabled^a
1: Enabled
- [4] **CK_IN_PHASE**^c
Choice of active edge of CKREF (master clock) that samples incoming YCrCb data. This bit can be used to analyze the cause of application synchronization problems.
0: CKREF falling edge^a
1: CKREF rising edge
- [3] **DEL_EN**
Enable of chroma to luma delay programming:
0: Disabled (default)
1: Enabled (chroma to luma delay is programmed by DEL[3:0] bits from register 81.
- [2] **YCRCB_REF_CLK_S**
Select whether the clock and data comes from the system or the pad.
0: Internal clock and video data
1: Clock from PIO1[2] and data from yc[0:7]
- [1] **Reserved**
- [0] **ENWSS**
WSS encoding enable
0: Disabled^d
1: Enabled

- a. When SECAM is performed trap filter is always enabled (ENTRAP = 1).
- b. 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).
- c. This bit is typically used when synchronization problems appear in application.
- d. Default mode when DENC_NRST pin is active (low level)

DEN_CFG4

Configuration4

7	6	5	4	3	2	1	0
SYNCIN_AD1	SYNCIN_AD0	SYNCOU_AD1	SYNCOU_AD0	ALINE	DEL4 [2:0]		

Address: *DENCBASEAddress* + 0x004

Type: Read/write

Reset: 0000 0000

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^a

01: +1 CKREF

10: +2 CKREF

11: +3 CKREF

[5:4] **SYNCOU_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^a

01: +1 CKREF

10: +2 CKREF

11: +3 CKREF

[3] **ALINE**

Video active line duration control

0: Full digital video line encoding (720 pixels - 1440 clock cycles)^a

1: Active line duration follows ITU-R/SMPTE analog standard requirements

[2:0] **DEL4[2:0]**

Delay on luma path w.r.t chroma path on YUV and RGB outputs when 4:4:4 input is used

010: + 2 pixel delay on luma

001: + 1 pixel delay on luma

000: + 0 pixel delay on luma

111: - 1 pixel delay on luma

110: - 2 pixel delay on luma

Other configurations: + 0 pixel delay on luma

a. Default mode when DENC_NRST pin is active (low level)

Confidential

DEN_CFG5**Configuration5**

7	6	5	4	3	2	1	0
SEL_RST_INC	BKDACC1	BKDACC2	BKDACC3	BKDACC4	BKDACC5	BKDACC6	DACINV

Address: *DENCBASEAddress* + 0x005

Type: Read/write

Reset: 0000 0000

Description:

[7] **SEL_RST_INC**

Choice of digital frequency synthesizer increment after soft reset or when PH_RST_MODE = 01 (see the [DEN_CFG8](#) register)

0: Hard wired value (depending on TV standard)^a 1: Soft (value from [DEN_IDFS\[1:3\]](#) registers)

[6:1] **BKDACCn**: Blanking of DACs (n = 1, 2, 3, 4, 5 or 6)

0: DAC N in normal operation^a

1: DAC N input code forced to black level (if RGB, UV or C) or blanking level (if Y or CVBS) depending on CONF_OUT bits of the [DEN_CFG8](#) register.

[0] **DACINV**: Inverts DAC codes to compensate for an inverting output stage in the application

0: Noninverted DAC inputs (outputs)^a 1: Inverted DAC inputs (outputs)

a. Default mode when DENC_NRST pin is active (low level)

DEN_CFG6**Configuration6**

7	6	5	4	3	2	1	0
SOFTRESET	Reserved			CFC0		TTX_EN	MAXDYN

Address: *DENCBASEAddress* + 0x006

Type: Read/write

Reset: 0001 0000

Description:

[7] **SOFTRESET**^a: Software reset

0: No reset^c

1: Software reset

[6:4] **Reserved**

[3:2] **CFC[1:0]**: Color frequency control via CFC line

00: Update mode disabled (update is carried out by loading [DEN_IDFS\[1:3\]](#) registers)^c

01: Update of increment for DDFS just after serial loading via CFC

10: Update of increment for DDFS on next active edge of HSYNC

11: Update of increment for DDFS just before next color burst

[1] **TTX_EN**: Teletext enable bit

0: Disabled^c

1: Enabled

[0] **MAXDYN**^b: Maximum dynamic magnitude allowed on YCrCb inputs for encoding

0: 0x10 to 0xEB for Y, 0x10 to 0xE0 for chrominance (Cr, Cb)^c 1: 0x01 to 0xFE for Y, Cr and Cb

a. Bit SOFTRESET is automatically reset after internal reset generation. Software reset is active for four CKREF 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).

b. EAV and SAV words are replaced by blanking values before being fed to the luminance and chrominance processing.

c. Default mode when DENC_NRST pin is active (low level)

DEN_CFG7

Configuration7

7	6	5	4	3	2	1	0
SECAM	GEN_SECAM	INV_PHI_SECAM	Reserved	SETUPYUV	UV_LEV	ENVPS	SQPIX

Address: *DENCBASEAddress* + 0x007

Type: Read/write

Reset: 00000000

Description:

[7:6] **SECAM and GEN_SECAM**

Where the bit order is SECAM, GEN_SECAM:

00: Standard selected by STD1 and STD0 bits of DEN_CFG0 (PAL or NTSC)^a

01: Invalid on STi5516

10: SECAM standard, the subcarrier phase sequence start point is line1

11: SECAM standard, the subcarrier phase sequence start point is line23

If master mode is performed in the SECAM standard, bit SECAM must be set before SYNC2, SYNC1 and SYNC0 bits of DEN_CFG0. If DEN_CFG0 is not programmed, then a soft reset must be performed after programming SECAM. In SECAM, the trap filter should always be enabled (see DEN_CFG3 register).

[5] **INV_PHI_SECAM**

Inversion of subcarrier phase

0: 0, 0, π ,... in odd fields and π , π , 0 in even fields^a

1: π , π , 0 in odd fields and 0, 0, π ,... in even fields

[4] **Reserved**

[3] **SETUPYUV**

Pedestal enable on YUV outputs:

This bit is significant only if register DEN_CFG8 bit CONF_OUT[1:0] = 01 (YUV outputs). If this is the case, the Y output on DAC5_G_Y has a 7.5 IRE pedestal. Furthermore, if UV_LEV = 1 then the U and V levels also depend on the value of SETUPYUV.

0: Disable^a

1: Enable

[2] **UV_LEV**

UV output level

0: Same peak to peak amplitude for both U and V (default value is 0.7 Vpp for 100/0/100/0 color bar pattern)^a

1: U and V outputs as defined by ITU-R624-4

[1] **ENVPS**

VPS encoding enable

0: Disable^a

1: Enable

[0] **SQPIX**

Square pixel mode enable

0: Disable^a

1: Enable

a. Default value

Confidential

DEN_CFG8

Configuration8

7	6	5	4	3	2	1	0
PH_RST_MODE1	PH_RST_MODE0	CONF_OUT1	CONF_OUT0	BLK_ALL	TTX_NOTMV	Reserved	

Address: *DENCBASEAddress* + 0x008

Type: Read/write

Reset: 0010 0000

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^b

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: Enabled: phase is updated with value from the [DEN_IDFS\[1:3\]](#) registers on the next increment updating by CFC (depending on CFC loading moment and [DEN_CFG6](#) CFC[1:0] bits).

11: Enabled: phase is reset after detecting of RST bit on CFC line, up to nine CKREF after loading of CFC s LSB.

[5:4] **CONF_OUT[1:0]**

Digital encoder output configuration

		DAC1	DAC2	DAC3	DAC4	DAC5	DAC6
0	0	Y	C	CVBS	C	Y	CVBS
0	1	Y	C	CVBS	V	Y	U
1	x ^b	Y	C	CVBS	R	G	B

If CONF_OUT[1:0] = 01 and register [DEN_CFG2](#) bit SET_4:4:4 = 1, then the DAC5 output Y comes from the Y4 input.

[3] **BLK_ALL**

Blanking of all video lines

0: Disabled^b

1: Enabled (all inputs ignored - 0x80 instead of Cr and Cb and 0x10 instead of Y and Y4)

[2] **TTX_NOTMV**

After reset, the default value of this bit is zero, however, for devices using teletext, this bit must then be reprogrammed to 1. This is to avoid the occurrence of teletext glitches in SECAM mode and loss of teletext data in all modes.

Priority of ancillary data on a VBI line. Note, higher priority data overwrites lower priority data.

0: Priority is: CGMS > closed caption > Macrovision™^a > WSS > VPS > teletext^b

1: Priority is: teletext > CGMS > closed caption > WSS > VPS > Macrovision™

[1:0] **Reserved**

- If there is no Macrovision™ programmed on the line, then VBI line blanking (when register [DEN_CFG1](#) bit BLKI = 1) overwrites VPS or teletext, and no VPS or teletext data is encoded.
- Default value

DEN_STA**Status (read-only)**

7	6	5	4	3	2	1	0
HOK	ATFR	BUF2_FREE	BUF1_FREE	FIELDCT2	FIELDCT1	FIELDCT0	JUMP

Address: *DENCBASEAddress + 0x009*

Type: R0

Reset: Undefined

Description:

[7] **HOK^a**

Hamming decoding of frame sync flag embedded within ITU-R656 / D1 compliant YCrCb streams

0: Consecutive errors

1: A single or no error^b

[6] **ATFR**

Frame synchronization flag

0: Encoder not synchronized^b

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 the [DEN_CCF2](#) registers are never written more than once between two frame reference signals, then bit BUF2_FREE is always true (set). Otherwise, closed caption field2 registers ([DEN_CCF2](#)) access might be temporarily forbidden by resetting bit BUF2_FREE until the next field2 closed caption line occurs.

Note that 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)^b

[4] **BUF1_FREE**

Closed caption registers access condition for field 1

Same as BUF2_FREE but concerns field 1.

Reset value: 1 (access authorized)^b

[3:1] **FIELDCT[2:0]**

Digital field identification number

000: Indicates field 1

...

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 [DEN_CFG6](#) register

a. Signal quality detector is issued from Hamming decoding of EAV, SAV from YCrCb

b. Default mode when DENC_Nrst pin is active (low level)

Confidential

DEN_IDFS[1:3]**Increment for digital frequency synthesizer**

	7	6	5	4	3	2	1	0
0x00A	D23	D22	D21	D20	D19	D18	D17	D16
0x00B	D15	D14	D13	D12	D11	D10	D9	D8
0x00C	D7	D6	D5	D4	D3	D2	D1	D0

Address: *DENCBASEAddress*+ 0x00A (DEN_IDFS1), 0x00B (DEN_IDFS2), 0x00C (DEN_IDFS3)

Type: Read/write

Reset: Undefined

Description: These three registers contain the 24-bit increment used by the DDFS, only 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_IDFS[1:3] are taken into account after a software reset, and only if bit SELRST_INC = 1 (register [DEN_CFG5](#))*

These registers are never reset and must be explicitly written, to ensure that they contain sensible information.

On hardware or software reset the DDFS is initialized with a hardwired increment, independent of DEN_IDFS[1:3]. These hardwired values cannot be read out of the digital encoder.

Bit field	Description	Frequency synthesized (MHz)	Ref. clock (MHz)
DEN_IDFS[1:3]	D[23:0]: 0x21 F07C for NTSC M	f = 3.5795452	27
	D[23:0]: 0x2A 098B for PAL BGHIN	f = 4.43361875	27
	D[23:0]: 0x21 F694 for PAL N	f = 3.5820558	27
	D[23:0]: 0x21 E6F0 for PAL M	f = 3.57561149	27
	D[23:0]: 0x25 5554 for NTSC M square pixel	f = 3.5795434	24.545454
	D[23:0]: 0x26 798C for PAL BGHIN square pixel	f = 4.43361867	29.5
	D[23:0]: 0x1F 15C0 for PAL N square pixel	f = 3.58205605	29.5
	D[23:0]: 0x25 4AD4 for PAL M square pixel	f = 3.57561082	24.545454

DEN_PDFS[1:2]

Static phase offset for digital frequency synthesizer

SECAM mode

	7	6	5	4	3	2	1	0
0x00D	B21	B20	B19	B18	B17	B16	B15	B14
0x00E	R21	R20	R19	R18	R17	R16	R15	R14

Address: *DENCBASEAddress* + 0x00D (DEN_PDFS1), 0x00E (DEN_PDFS2),

Type: Read/write

Reset: Undefined

Description: When bit SECAM in register [DEN_CFG7](#) = 1 and bit SELRST in register [DEN_CFG2](#) = 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:

- blue lines: $\text{DEN\_PDFS1} \times 16384$ (decimal), or $\text{DEN\_PDFS1} \times 0x4000$,
- red lines: $(256 + \text{DEN\_PDFS1} + \text{DEN\_PDFS2}) \times 16384$ (decimal), or $(100 + \text{DEN\_PDFS1} + \text{DEN\_PDFS2}) \times 0x4000$.

If bit SELRST = 0 (for example after a hardware reset) the phase offset used every time the DDFS is reinitialized is a hard wired value. The hard wired values cannot be read out of the digital encoder. 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.

NonSECAM mode

	7	6	5	4	3	2	1	0
0x00D	Reserved						O23	O22
0x00E	O21	O20	O19	O18	O17	O16	O15	O14

In the following conditions, when bit SECAM in register [DEN_CFG7](#) = 0 and bit SELRST in register [DEN_CFG2](#) = 0, the phase accumulator can be 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 (see bits VALRST[1:0] of [DEN_CFG2](#)).

The fourteen remaining LSBs loaded into the accumulator in these cases are all zeroes (defining the phase reset value with a 0.35° accuracy).

If bit SELRST = 0 (for example after a hardware reset) the phase offset used every time the DDFS is reinitialized is a hard wired value. The hard wired values cannot be read out of the digital encoder. 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 hard wired values follow the procedure below.

- 1 Load the registers with the required value.
- 2 Set bit SELRST to 1 ([DEN_CFG2](#)).
- 3 Perform a software reset ([DEN_CFG6](#)), or set [DEN_CFG6](#) bit RSTOSC_BUF to 1. This puts the soft phase value into a tampon register and sets register [DEN_CFG8](#) bit PH_RST_MODE[1:0] 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.

DEN_WSS[1:2]

WSS_BIT[15:0]: WSS data registers

	7	6	5	4	3	2	1	0
0x00F	WSS15	WSS14	WSS13	WSS12	WSS11	WSS10	WSS9	WSS8
0x010	WSS7	WSS6	WSS5	WSS4	WSS3	WSS2	WSS1	WSS0

Address: *DENCBaseAddress* + 0x00F (DEN_WSS1), 0x010 (DEN_WSS2)

Type: Read/write

Reset: Undefined

Description:

- [15:13] **WSS[15:13]**
0: Reserved
- [12] **WSS[12]**
0: Camera mode
1: Film mode
- [11:8] **WSS[11:8]^a**
1000: Full format 4:3
0010: Box 14:9 top
0100: Box 16:9 top
1110: Full format 4:3 (shoot and protect 14:9 center)
0001: Box 14:9 center
1011: Box 16:9 center
1101: > box 16:9 center
0111: Full format 16:9 (anamorphic)
- [7:5] **WSS[7:5]**
0: Reserved
- [4:3] **WSS[4:3]**
00: No open subtitles
10: Subtitles out of image area
01: Subtitles in active image area
11: Reserved
- [2] **WSS[2]**
0: No subtitles within teletext
1: Subtitles within teletext
- [1:0] **WSS[1:0]**
0: Reserved

a. WSS11 is an odd parity bit

Confidential

DEN_DAC13**DAC1 and DAC3 multiplying factors**

7	6	5	4	3	2	1	0
DAC1_MULT[3:0]				DAC3_MULT[3:0]			

Address: *DENCBaseAddress* + 0x011

Type: Read/write

Reset: 1000 1000

Description:

- [7:4] **DAC1_MULT[3:0]**: Multiplying factor on DAC1_Y digital signal before the DACs with 3.125% step
 0000: 75.000% DAC1_Y value compared to default 0001: 78.125% DAC1_Y value compared to default
 0010: 81.250% DAC1_Y value compared to default 0011: 84.375% DAC1_Y value compared to default
 1000^a: 100% DAC1_Y value compared to default
 1111: 121.875% DAC1_Y value compared to default
- [3:0] **DAC3_MULT[3:0]**: Multiplying factor on DAC3_CVBS digital signal before the DACs with 3.125% step
 0000: 75.00% DAC3_CVBS compared to default 0001: 78.125% DAC3_CVBS compared to default
 0010: 81.25% DAC3_CVBS compared to default 0011: 84.375% DAC3_CVBS compared to default
 1000^a: 100% DAC3_CVBS compared to default
 1111: 121.875% DAC3_CVBS compared to default

a. Default value

DEN_DAC45**DAC4 and DAC5 multiplying factors**

7	6	5	4	3	2	1	0
DAC4_MULT[3:0]				DAC5_MULT[3:0]			

Address: *DENCBaseAddress* + 0x012

Type: Read/write

Reset: 1000 1000

Description:

- [7:4] **DAC4_MULT[3:0]**
 Multiplying factor on DAC4_R_V_C digital signal before the DACs: DAC4_R_V_C value (in% of default value)
 0000: if R (CONF_OUT = 1x) = 80.49%, if V or C (CONF_OUT = 0) = 75.000%
 0001: if R (CONF_OUT = 1x) = 82.93% if V or C (CONF_OUT = 0) = 78.125%
 0010: if R (CONF_OUT = 1x) = 85.37% if V or C (CONF_OUT = 0) = 81.250%
 0011: if R (CONF_OUT = 1x) = 87.81% if V or C (CONF_OUT = 0) = 84.375%
 ...
 1000: if R (CONF_OUT = 1x) = 100%^a If V or C (CONF_OUT = 0) = 100%^a
 ...
 1111: if R (CONF_OUT = 1x) = 117.07% if V or C (CONF_OUT = 0) = 121.875%
- [3:0] **DAC5_MULT[3:0]**
 Multiplying factor on DAC5_G_Y digital signal before the DACs: DAC5_G_Y value (in% of default value)
 0000: if G(CONF_OUT = 1x) = 80.49% if Y(CONF_OUT = 0x) = 75.000%
 0001: if G(CONF_OUT = 1x) = 82.93% if Y(CONF_OUT = 0x) = 78.125%
 0010: if G(CONF_OUT = 1x) = 85.37% if Y(CONF_OUT = 0x) = 81.250%
 0011: if G(CONF_OUT = 1x) = 87.81% if Y(CONF_OUT = 0x) = 84.375%
 ...
 1000: if G(CONF_OUT = 1x) = 100%^a if Y(CONF_OUT = 0x) = 100%^a
 1111: if G(CONF_OUT = 1x) = 117.07% if Y(CONF_OUT = 0x) = 121.875%

a. Default mode when DENC_NRST pin is active (low level)

DEN_DAC6C**DAC6 and C multiplying factors**

7	6	5	4	3	2	1	0
DAC6_MULT[3:0]				C_MULT[3:0]			

Address: *DENCBASEAddress* + 0x013

Type: Read/write

Reset: 1000 0000

Description:

[7:4] DAC6_MULT[3:0]

Multiplying factor on DAC6_B_CVBS digital signal before the DACs:

DAC6_B_CVBS value (in% of default value)

0000: If B(CONF_OUT = 1x) = 80.49%

If CVBS (CONF_OUT = 01) = 75.000%

0001: If B(CONF_OUT = 1x) = 82.93%

If CVBS (CONF_OUT = 01) 78.125%

0010: If B(CONF_OUT = 1x) = 85.37%

If CVBS (CONF_OUT = 01) 81.250%

0011: If B(CONF_OUT = 1x) = 87.81%

If CVBS (CONF_OUT = 01) 84.375%

...

1000: If B(CONF_OUT = 1x) = 100%

If CVBS (CONF_OUT = 01) 100%

...

1111: If B(CONF_OUT = 1x) = 117.07%

If CVBS (CONF_OUT = 01) 121.875%

[3:0] C_MULT[3:0]

Multiplying factor on C digital output (before DACs) and on color part of CVBS signal:

0000: 1.000000 (1.000000 dec) (factor value (C_MULT))

0001: 1.000001 (1.015625 dec) (factor value (C_MULT))

0010: 1.000010 (1.031250 dec) (factor value (C_MULT))

0011: 1.000011 (1.046875 dec) (factor value (C_MULT))

...

1111: 1.001111 (1.234375 dec) (factor value (C_MULT))

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 inputted. 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. If bit UV_LEV from register [DEN_CFG7](#) is set, default peak to peak amplitude is 86% (80% if SETUPYUV = 1) for V output and 67% (57% if SETUPYUV = 1) for U output, of default Y or CVBS peak to peak amplitude if 100/0/100/0 color bar pattern is inputted, according to ITU-R 624-4 definition of UV signals.

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 inputted. In other words, when IREF is set to deliver 1Vpp for CVBS on DAC3 for example (and DAC3_MULT = 1000), when switched to B, DAC6 delivers 0.7 Vpp.

DEN_CID**CHIPID: digital encoder version identification number**

7	6	5	4	3	2	1	0
CHIPID							

Address: *DENCBASEAddress* + 0x018

Type: RO

Reset: Undefined

Description:

[7:0] CHIPID: 1010 0000

DEN_VPS1

VPS: VPS data registers

	7	6	5	4	3	2	1	0
0x019	S1	S0	Reserved	Reserved	CNI3	CNI2	CNI1	CNI0
0x01A	NP7	NP6	D4	D3	D2	D1	D0	M3
0x01B	M2	M1	M0	H4	H3	H2	H1	H0
0x01C	MIN4	MIN3	MIN3	MIN2	MIN1	MIN0	C3	C2
0x01D	C1	C0	NP5	NP4	NP3	NP2	NP1	NP0
0x01E	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0

Address: *DENCBASEAddress* + 0x019 (DEN_VPS1), 0x01A (DEN_VPS2), 0x01B (DEN_VPS3), 0x01C (DEN_VPS4), 0x01D (DEN_VPS5), 0x01E (DEN_VPS6)

Type: Read/write

Reset: Undefined

Description:

0x19: [7:0] **S[1:0]**

Sounds

00: Don't know

10: Stereo

01: Mono

11: Dual sound

0x19: [5:4] **Reserved**

0x19: [3:0] **CNI[3:0]**

4 bits of CNI reserved for enhancement of VPS

0x1A: [7:6] **NP[7:6]**

Network or program CNI

0x1A: [5:1] **D[4:0]**

Day, binary

0x1A: [0] **M[3:0]**

0x1B: [7:5] Month, binary

0x1B: [4:0] **H [4:0]**

Hour, binary

0x1C: [7:2] **MIN[5:0]**

Minute, binary

0x1C: [1:0] **C[3:0]**

0x1D: [7:6] Country, binary

0x1D: [5:0] **NP[5:0]**

Network or program CNI (country and network identification)

0x1E: [7:0] **PT[7:0]**

Program type, binary

Confidential

DEN_CGMS[1:3]**CGMS_BIT[1:20]: CGMS data registers (20 bits only)**

	7	6	5	4	3	2	1	0
0x01F	Reserved				B1	B2	B3	B4
0x020	B5	B6	B7	B8	B9	B10	B11	B12
0x021	B13	B14	B15	B16	B17	B18	B19	B20

Address: *DENCBASEAddress* + 0x01F (DEN_CGMS1), 0x020 (DEN_CGMS2), 0x021 (DEN_CGMS3)

Type: Read/write

Reset: Undefined

Description:

0x1F: [7:4] **Reserved**

0x0F8: [3:1] **B1 - B3**: Word0A

0x0F8: [0] **B4 - B6**: Word0B

0x100: [7:6]

0x100: [5:2] **B7 - B10**: Word1

0x100: [1:0] **B11 - B14**: Word2

0x108: [7:6]

0x108: [5:0] **B15 - B20**: CRC (not internally computed)

DEN_TTX[1:5]**TTX__[1:5] DEF teletext block definition**

	7	6	5	4	3	2	1	0
0x022	FP_TTXT	TTXDEL2	TTXDEL1	TTXDEL0	TTX_L6	TTX_L7	TTX_L8	TTX_L9
0x023	TTX_L10	TTX_L11	TTX_L12	TTX_L13	TTX_L14	TTX_L15	TTX_L16	TTX_L17
0x024	TTX_L18	TTX_L19	TTX_L20	TTX_L21	TTX_L22	TTX_L23	TTX_L318	TTX_L319
0x025	TTX_L320	TTX_L321	TTX_L322	TTX_L323	TTX_L324	TTX_L325	TTX_L326	TTX_L327
0x026	TTX_L328	TTX_L329	TTX_L330	TTX_L331	TTX_L332	TTX_L333	TTX_L334	TTX_L335

Address: *DENCBASEAddress* + 0x022 (DEN_TTX1), 0x023 (DEN_TTX2), 0x024 (DEN_TTX3), 0x025 (DEN_TTX4), 0x026 (DEN_TTX5)

Type: Read/write

Reset: Undefined

Description:

0x22: [7] **FP_TTXT**: Full page teletext mode enable

0: Disabled^a

1: Enabled

0x22: [6:4] **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).

100: Default value

0x22: [3:0] **TTX_Ln**

0x23 to 0x26: [7:0] Each of these bits enables teletext on line X (ITU-R line numbering and 625 line systems). teletext line selections for other standards refer to the table below.

a. Default mode when DENC_NRST pin is active (low level).

Table 161: Teletext block definition for PAL and NTSC standards

TTX_[1:4]_DEF bit field	PAL BDGHIN line (CCIR 625 line numbering)	PAL M line (CCIR 525 line numbering)	NTSC M line (SMPTE 525 line numbering)
TTX_L6	6	6	9
TTX_L7	7	7	10
TTX_L8	8	8	11
TTX_L9	9	9	12
TTX_L10	10	10	13
TTX_L11	11	11	14
TTX_L12	12	12	15
TTX_L13	13	13	16
TTX_L14	14	14	17
TTX_L15	15	15	18
TTX_L16	16	16	19
TTX_L17	17	17	20
TTX_L18	18	18	21
TTX_L19	19	19	22
TTX_L20	20	20	23
TTX_L21	21	21	24
TTX_L22	22	22	25
TTX_L23	23	23	26
TTX_L318	318	268	271
TTX_L319	319	269	272
TTX_L320	320	270	273
TTX_L321	321	271	274
TTX_L322	322	272	275
TTX_L323	323	273	276
TTX_L324	324	274	277
TTX_L325	325	275	278
TTX_L326	326	276	279
TTX_L327	327	277	280
TTX_L328	328	278	281
TTX_L329	329	279	282
TTX_L330	330	280	283
TTX_L331	331	281	284

Confidential

Table 161: Teletext block definition for PAL and NTSC standards

TTX_[1:4]_DEF bit field	PAL BDGHIN line (CCIR 625 line numbering)	PAL M line (CCIR 525 line numbering)	NTSC M line (SMPTE 525 line numbering)
TTX_L332	332	282	285
TTX_L333	333	283	286
TTX_L334	334	284	287
TTX_L335	335	285	288

DEN_CCF1**CCCF1: closed caption characters/extended data for field 1**

	7	6	5	4	3	2	1	0
0x027	OPC11	C117	C116	C115	C114	C113	C112	C111
0x028	OPC12	C127	C126	C125	C124	C123	C122	C121

Address: *DENCBASEAddress* + 0x027 (DEN_CCF11), 0x028 (DEN_CCF12)

Type: Read/write

Reset: Undefined

Description:

0x27: [7] **OPC11**: Odd parity bit of US-ASCII 7-bit character c11[7:1]

0x27: [6:0] **C11[7:1]**: First byte to encode in field1

0x28: [7] **OPC12**: Odd parity bit of US-ASCII 7-bit character c12[7:1]

0x28: [6:0] **C12[7:1]**: Second byte to encode in field1

- Default value: none, but closed captions enabling without loading these registers issue a null character.
- DEN_CCF1 registers are never reset.

DEN_CCF2**CCCF2: closed caption characters/extended data for field 2**

	7	6	5	4	3	2	1	0
0x029	OPC21	C217	C216	C215	C214	C213	C212	C211
0x02A	OPC22	C227	C226	C225	C224	C223	C222	C221

Address: *DENCBASEAddress* + 0x029 (DEN_CCF21), 0x02A (DEN_CCF22)

Type: Read/write

Reset: Undefined

Description:

0x29: [7] **OPC21**: Odd parity bit of US-ASCII 7-bit character c21[7:1]

0x29: [6:0] **C21[7:1]**: First byte to encode in field2

0x2A: [7] **OPC22**: Odd parity bit of US-ASCII 7-bit character c22(7, 1)

0x2A: [6:0] **C22[7:1]**: Second byte to encode in field2

- Default value: none, but closed captions enabling without loading these registers issue a null character.
- DEN_CCF2 registers are never reset.

Confidential

DEN_CLF1**CCLIF1: closed caption/extended data line insertion for field 1**

7	6	5	4	3	2	1	0
Reserved			L14	L13	L12	L11	L10

This register programs the TV line number of the closed caption/extended data encoded in field1

Address: *DENCBASEAddress* + 0x02B

Type: Read/write

Reset: 0000 1111

Description: **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

- Default value = 0 1111 line 21 (525/60, 525-SMPTE line number convention), which corresponds to line 21 in 625/50 system, (625-CCIR line number convention)

DEN_CLF2

CCLIF2: closed caption/extended data line insertion for field 2

7	6	5	4	3	2	1	0
Reserved			L24	L23	L22	L21	L20

This register programs the TV line number of the closed caption/extended data encoded in field 1

Address: *DENCBASEAddress* + 0x02C

Type: Read/write

Reset: 0000 1111

Description: 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)

- 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_REG_64**TTX_CONF**

7 6 5 4 3 2 1 0

TTXT100IRE	TTXT_ABCD[1:0]	Reserved

Address: *DENCBaseAddress* + 0x040

Type: Read/write

Reset: 0101 0000

Description:

[7] **TTXT100IRE**: Teletext waveform amplitude

0: 70 IRE^a

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^a

11: Teletext D

[4:0] **Reserved. Do not change these values.**

a. Default mode when DENC_NRST pin is active (low level)

DEN_REG_65**DAC2MULT and TTXS**

7 6 5 4 3 2 1 0

DAC2_MULT[3:0]			TTX_MASK_OFF		Reserved	BCS_EN_4	BCS_EN_2

Address: *DENCBaseAddress* + 0x041

Type: Read/write

Reset: 1000 0010

Description:

[7:4] **DAC2_MULT[3:0]**: Multiplying factor on DAC2_C digital signal before the DACs with 3.125% step

0000: 75.000% DAC2_C value compared to default 0001: 78.125% DAC2_C value compared to default

0010: 81.250% DAC2_C value compared to default 0011: 84.375% DAC2_C value compared to default...

1000:^a 100% DAC2_C value compared to default.... 1111: 121.875% DAC2_C value compared to default

[3] **TTX_MASK_OFF**: Masking of 2 bits in teletext framing code, depending on selected teletext standard

0: Enable^a

1: Disable

[2] **Reserved**

[1] **BCS_EN_4**: Brightness, contrast and saturation control by [DEN_REG_69](#) to [DEN_REG_71](#) on 4:4:4 input

0: Disable

1: Enable^a

[0] **BCS_EN_2**: Brightness, contrast and saturation control by [DEN_REG_69](#) to [DEN_REG_71](#) on 4:2:2 input

0: Disable^a

1: Enable

a. Default mode when DENC_NRST pin is active (low level)

Confidential

DEN_REG_69**Brightness**

7 6 5 4 3 2 1 0

B[7:0]

Address: *DENCBASEAddress + 0x045*

Type: Read/write

Reset: 1000 0000

Description: The register contents are used by the following formula to adjust the luminance intensity of the display video image:

$$Y_{out} = Y_{in} + B - 128$$

Where Y_{in} is 8-bit input luminance and Y_{out} is the result of 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_REG_70**Contrast**

7 6 5 4 3 2 1 0

C[7:0]

Address: *DENCBASEAddress + 0x046*

Type: Read/write

Reset: 0000 0000

Description: The register contents are used by the following formula to adjust the relative difference between the display image higher and lower intensity luminance values:

$$Y_{out} = \frac{(Y_{in} - 128)(C + 128)}{128} + 128$$

Where, Y_{in} is 8-bit input luminance, Y_{out} is the result of 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_REG_71**Saturation**

7 6 5 4 3 2 1 0

S[7:0]

Address: *DENCBASEAddress + 0x047*

Type: Read/write

Reset: 1000 0000

Description: The register contents are used by the following formula to adjust the color intensity of the displayed video image:

$$Cr_{out} = \frac{S(Cr_{in} - 128)}{128} + 128$$

$$Cb_{out} = \frac{S(Cb_{in} - 128)}{128} + 128$$

Where Cr_{in} and Cb_{in} are the 8-bit input chroma, Cr_{out} and Cb_{out} 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).

Confidential

DEN_CFCOE[0:8]

Chroma filter coefficient 0 to 8

	7	6	5	4	3	2	1	0
0x048	FLT_S	PLG_DIV1	PLG_DIV0	CH_COEF0[4:0]				
0x049	Reserved	CH_COEF8[8]			CH_COEF1[5:0]			
0x04A	Reserved	CH_COEF2[6:0]						
0x04B	Reserved	CH_COEF3[6:0]						
0x04C	CH_COEF4[7:0]							
0x04D	CH_COEF5[7:0]							
0x04E	CH_COEF6[7:0]							
0x04F	CH_COEF7[7:0]							
0x050	CH_COEF8[7:0]							

Address: *DENCBBaseAddress* + 0x048 (DEN_CFCOE0), 0x049, 0x04A, up to 0x050 (DEN_CFCOE8)

Type: Read/write

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. The coefficients are chosen to give the required filter response for a specific application according to the symmetrical RIF filter equation:

$$H(z) = c_0 + c_1 z^{-1} + c_2 z^{-2} + c_7 z^{-7} + c_8 z^{-8} + c_7 z^{-9} + c_2 z^{-14} + c_1 z^{-15} + c_0 z^{-16}$$

The register reset (or default) values give the coefficients for the SECAM square pixel mode. Each register value is calculated by adding an offset value to the desired coefficient value, according to the relationship:

register value = offset + actual coefficient value.

For instance, to obtain a *coefficient value* of 5 for c_4 , which has an offset of 32, the register DEN_CFCOE4 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.

The sampling frequency of the filter is CKREF (27 MHz., 24.5454 MHz. or 29.5 MHz.).

0x48: [7]FLT_S

0: Use hard wired coefficients for the chroma filter (reset) or programmed, to determine coefficients 1: Use register DEN_CFCOE0..8 values, default

0x48: [6:5]PLG_DIV[1:0]

00: When sum of coefficients = 512 (dec.) 01: When sum of coefficients = 1024 (dec. reset val)
10: When sum of coefficients = 2048 (dec.) 11: When sum of coefficients = 4096 (dec.)

0x48: [4:0]CH_COEF0[4:0]: Coefficient number 0; offset value = 16, reset value = 10001, corresponding to $c_0 = 1$

0x49: [7]Reserved: Reset value = 0

0x49: [6]CH_COEF8[8]: Coefficient number 8 bit 8; reset value = 0, see CH_COEF8[7:0]

0x49: [5:0]CH_COEF1[5:0]: Coefficient number 1; offset value = 32, reset value = 100111, corresponding to $c_1 = 7$

0x4A: [7]Reserved: Reset value = 0

0x4A: [6:0]CH_COEF2[6:0]: Coefficient number 2; offset value = 64, reset value = 1010100, corresponding to $c_2 = 20$

0x4B: [7]Reserved: Reset value = 0

0x4B: [6:0]CH_COEF3[6:0]: Coefficient number 3: Offset value = 32, reset value = 1000111, corresponding to $c_3 = 39$

0x4C: [7:0]CH_COEF4[7:0]: Coefficient number 4: Offset value = 32, reset value = 01011111, corresponding to $c_4 = 63$

0x4D: [7:0]CH_COEF5[7:0]: Coefficient number 5: Offset value = 32, reset value = 01110111, corresponding to $c_5 = 87$

0x4E: [7:0]CH_COEF6[7:0]: Coefficient number 6: Offset value = 0, reset value = 01101100, corresponding to $c_6 = 108$

0x4F: [7:0]CH_COEF7[7:0]: Coefficient number 7: Offset value = 0, reset value = 01111011, corresponding to $c_7 = 123$

0x50: [7:0]CH_COEF8[7:0]^a: Coefficient number 8: Offset value = 0, reset value = 10000000, corresponding to $c_8 = 128$

a. The MSB of this coefficient value can be found in the DEN_CFCOE1 register.

DEN_CDEL_LFC**Chroma delay and luma filter control**

7	6	5	4	3	2	1	0
DEL[3:0]				SET444_CVBS	PLG_DIV_Y[1:0]		FLT_YS

Address: *DENCBASEAddress* + 0x051

Type: Read/write

Reset: 0010 0010

Description: This register contains the chroma path delay (referenced to luma one on S-VHS and CVBS outputs), selection of input format mode, and three bits to control the luma filter. The following table describes the register's functions.

[7:4] **DEL[3:0]** ^a

Delay on chroma path with reference to luma path encoding (1 pixel = 2 periods of frequency f_{CKREF}):

0010: -0.5 pixel delay (default for PAL/NTSC in 4:2:2 format on CVBS)

0011: -1.0 pixel delay

0100: -1.5 pixel delay

0101: -2.0 pixel delay

1100: +2.5 pixel delay

1101: +2.0 pixel delay

1110: +1.5 pixel delay

1111: +1.0 pixel delay

0000: +0.5 pixel delay (default for SECAM in 4:4:4 format on CVBS)

0110, 1000, 1010: +0.5 pixel delay

0001: 0.0 pixel delay (default for PAL/NTSC in 4:4:4 format or SECAM in 4:2:2 format on CVBS)

0111, 1001, 1011: 0.0 pixel delay

[3] **SET444_CVBS**

Select input format to digital encoder for S-VHS and CVBS outputs:

0: 4:2:2 input format (default)

1: 4:4:4 input format

[2:1] **PLG_DIV_Y[1:0]**

00: When sum of coefficients = 256 (dec.)

01: When sum of coefficients = 512 (dec. reset val)

10: When sum of coefficients = 1024 (dec.)

11: When sum of coefficients = 2048 (dec.)

[0] **FLT_YS**

0: Use hard wired coefficients for the luma filter

1: Use register DEN_LFCOEF[0:9] values, default (reset) or programmed, to determine coefficients

a. Bit DEL_EN in register [DEN_CFG3](#) selects either default or programmed delays.

DEN_LFCOEF[0:9]**Luma filter coefficient[0:9]**

	7	6	5	4	3	2	1	0
0x052	Reserved		LU_COEF8[8]	LU_COEF0[4:0]				
0x053	LU_COEF9[9:8]		LU_COEF1[5:0]					
0x054	LU_COEF6[8]	LU_COEF2[6:0]						
0x055	LU_COEF7[8]	LU_COEF3[6:0]						
0x056	LU_COEF4[7:0]							
0x057	LU_COEF5[7:0]							
0x058	LU_COEF6[7:0]							
0x059	LU_COEF7[7:0]							
0x05A	LU_COEF8[7:0]							
0x05B	LU_COEF9[7:0]							

Address: *DENCBASEADDRESS* + 0x052 (DEN_LFCOEF0), 0x053, 0x054, up to 0x05B (DEN_LFCOEF9)

Type: Read/write

Reset: See table below

Description: These registers contain the ten coefficients for the luma filter, which are described in the table below. The coefficients give the required filter response for a specific application according to the symmetrical RIF filter equation:

$$H(z) = l_0 + l_1 z^{-1} + l_2 z^{-2} + l_8 z^{-8} + l_9 z^{-9} + l_8 z^{-10} + l_2 z^{-16} + l_1 z^{-17} + l_0 z^{-18}$$

The values of the coefficients LU_COEF0 through to LU_COEF7 must be entered in two's complement form and the remainder as normal positive values.

The sampling frequency of the filter is CKREF (27 MHz, 24.5454 MHz or 29.5 MHz).

The control bits for this filter are in register [DEN_CDEL_LFC](#).

0x5B: [7:0] **LU_COEF9[7:0]**^a: Coefficient number 9; reset value = 1111 1000, corresponding to l9 = +504

0x5A: [7:0] **LU_COEF8[7:0]**^a: Coefficient number 8; reset value = 0011 1101, corresponding to l8 = +317

0x59: [7:0] **LU_COEF7[7:0]**^a: Coefficient number 7; reset value = 0000 0111, corresponding to l7 = +7

0x58: [7:0] **LU_COEF6[7:0]**^a: Coefficient number 6; reset value = 1010 1100, corresponding to l6 = -84

0x57: [7:0] **LU_COEF5[7:0]**: Coefficient number 5; reset value = 1111 1011, corresponding to l5 = -5

0x56: [7:0] **LU_COEF4[7:0]**: Coefficient number 4; reset value = 0001 1111, corresponding to l4 = +31

0x55: [7] **LU_COEF7[8]**: Coefficient number 7 bit 8; reset value = 0, see LU_COEF7[7:0]

0x55: [6:0] **LU_COEF3[6:0]**: Coefficient number 3; reset value = 000 0011, corresponding to l3 = +3

0x54: [7] **LU_COEF6[8]**: Coefficient number 6 bit 8; reset value = 1, see LU_COEF6[7:0]

0x54: [6:0] **LU_COEF2[6:0]**: Coefficient number 2; reset value = 111 0111, corresponding to l2 = -9

0x53: [7:6] **LU_COEF9[9:8]**: Coefficient number 9 bits 8 and 9; reset value = 01, see LU_COEF9[7:0]

0x53: [5:0] **LU_COEF1[5:0]**: Coefficient number 1; reset value = 11 1111, corresponding to l1 = -1

0x52: [7:6] **Reserved**

Reset: 0

0x52: [5] **LU_COEF8[8]**: Coefficient number 8 bit 8; reset value = 1, see LU_COEF8[7:0]

0x52: [4:0] **LU_COEF0[4:0]**: Coefficient number 0; reset value = 0 0001, corresponding to l0 = +1

a. The MSB(s) of these coefficient values can be found in one of the above DEN_LFCOEF registers.

46 Triple video DAC

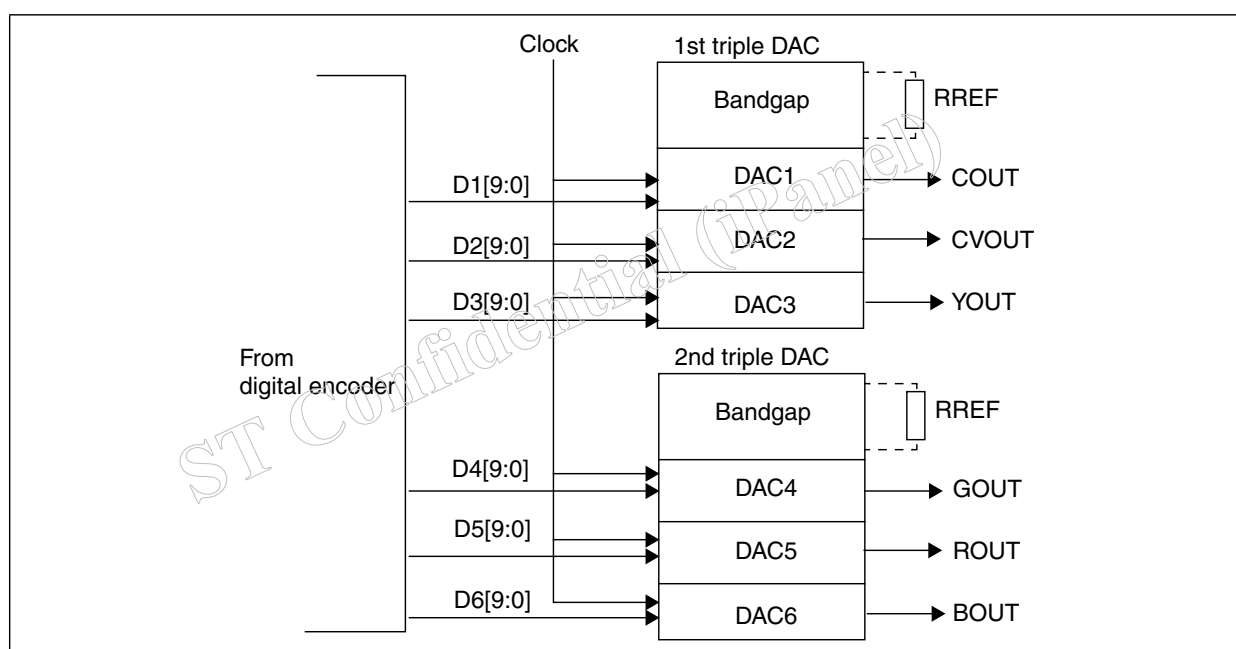
46.1 Overview

There are two on-chip 3 x 10-bit digital to analog converters (triple DACs) used for video output. One provides output signals in CVBS, Y, C, and the other in RGB. [Figure 161](#) shows the DAC arrangement.

An external reference resistor is associated with the bandgap voltage to generate a reference current. This resistor is connected between the VREF pin of the bandgap and a dedicated IREF pin to achieve high noise immunity.

The global segmented architecture is presented in [Figure 161](#). Current sources provide an output range of 1.4 V with good linearity due to a 3.3 V power supply. Sampled data is available on video outputs after one clock period (on the next rising clock edge).

Figure 161: Triple video DAC schematic



The triple video DAC has its own analog ground supplies for noise reduction. The analog section is supplied by a 3.3 V supply distinct from the 3.3 V of the rest of the chip. The digital section is common to the digital 1.8 V core supply of the chip.

46.2 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 162: Reference input codes

	Y-PAL/SECAM	Y-NTSC	RGB
WHITE(235)	816.00	802.00	602.00
BLACK(16)	256.00	240.00	41.00
SYNC TIP	16.00	16.00	-

Note: CVBS = Y + C, therefore chrominance component has no effect on the CVBS signal (C is null).

46.3 Video output voltage level

The resistor R_{ref} connected to the bandgap controls the DAC output current. For the maximum code (1023 in decimal):

$$I_{out} (max) = 80.539 / R_{ref}$$

For example, with a typical value of $R_{ref} = 10 \text{ k}\Omega$, $I_{out} (max) = 8 \text{ mA}$ for each DAC.

The value of R_{ref} must be greater or equal to $10 \text{ k}\Omega$ 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):

$$V_{out} (max) = R_{load} \times I_{out} (max)$$

For example, with a typical load value

$$R_{load} = 165 \Omega,$$

$$I_{out} (max) = 8 \text{ mA},$$

$$V_{out} (max) = 1.32 \text{ V}.$$

V_{out} should always be lower than 1.4 V to guarantee linearity.

For any given digital input code, the output voltage of the DAC is given by the following formula:

$$V_{out} = D_{in} / 1023 \times V_{out} (max) = (D_{in} \times R_{load} \times 80.539) / (R_{ref} \times 1023)$$

$$V_{out} = D_{in} \times R_{load} \times 0.0787 / R_{ref}$$

For example, with $R_{load} = 165 \Omega$, $R_{ref} = 10 \text{ k}\Omega$ and $D_{in} = 802$, $V_{out} = 1.04 \text{ V}$.

46.4 Video specifications and DAC setup

In Y-PAL/SECAM, the output video range between code 16 (synchronization level) and code 816 (white level) should be: $V_{out}(816) - V_{out}(16) = 1\text{ V}$. 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 10 k Ω .

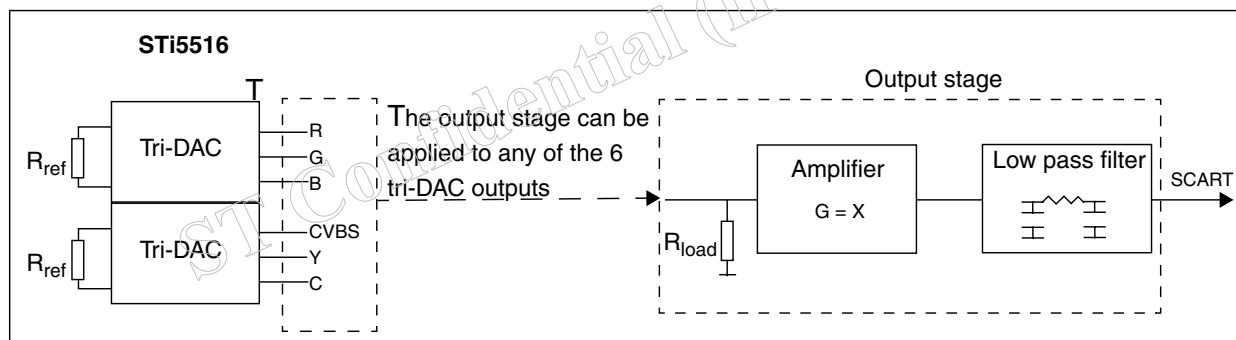
According to video specifications ITU-R BT 601, the nominal sampling frequency is 27 MHz. The DACs clock is buffered from the digital encoder clock, which is usually generated externally by a VCXO. It must be as clean as possible to achieve a good signal to noise ratio.

To enable the video DAC, bit 14 (VIDEODAC_ENABLE) of interconnect configuration control register C must be set to 1. See *CONFIG_CONTROL_C* on page 196.

46.5 Output stage adaptation and amplification

A block diagram of the output stage is shown below. The purpose of the output stage depends on the application and the required price to performance ratio. The output stage is connected directly to a SCART connector or to other components (in which case the output signal level and impedance may be different).

Figure 162: Output stage block diagram



The amplifier gain must be in accordance with the one of the tri-DACs (defined by R_{load} and R_{ref}). If the amplifier gain cannot be set to the standard output video range by R_{ref} and R_{load} , it can be tuned. In common applications (with $R_{ref} = 10\text{ k}\Omega$ and $R_{load} = 165\ \Omega$), 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 tri-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 of the amplifier (generally degrading the C/L ratio), correction must be applied to preserve a good performance. Also, to guarantee a 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.

47 Audio decoder

47.1 Overview

47.1.1 Input formats

The audio decoder accepts:

- Dolby® Digital,
- MPEG-1 layers I and II,
- MP3 (a product variant option),
- PCM formats,
- PES streams for MPEG-1 and Dolby® Digital.

S/PDIF input data (IEC-60958 or IEC-61937 standards) is accepted if external circuitry extracts the PCM clock from the stream and decodes the biphasic mark of S/PDIF.

Note: MPEG2 audio format is not supported in STi5516. It may be referenced in the audio decoder register set, but MPEG2 decoding modes should not be used.

47.1.2 Audio video synchronization

Skip frame, pause blocks and soft mute frame features can be used to synchronize audio and video data. PTS audio extraction is also supported.

47.1.3 Output formats

The device outputs up to four channels of PCM data and appropriate clocks for external digital-to-analog converters:

- two PCM data channels on a PCM output (PCMOUT3) for VCR,
- two PCM data channels on a PCM output (PCMOUT0) for single stereo,
- three clocks for the external DACs:
 - PCMCLK,
 - SCLK,
 - LRCLK.

Postprocessing options on the VCR and single stereo channels are:

- Pro Logic® compatible downmix,
- 2 front/0 rear downmix,
- L/R hardware copy,
- mixing between the first and second input,
- a copy of the second input.

Note: The top level configuration bits 1 and 0 in register CONFIG_CONTROL_C (AUDIOPCMOUT1_SEL_VCR_CANDSUB and AUDIODAC_SEL_VCR_NOT_MAINLR) allow MUXing of these internal signals to the PCMOUT1 pin.

Programmable downmix enables 1, 2, 3, 4 or 5 channel outputs. Data can be output in either I²S or Sony formats. The decoder can format output data according to the IEC-60958 standard (for L/R channels or LVCR/RVCR PCM outputs, 16 or 24 bits) or the IEC-61937 standard (for compressed data), with a sampling frequency, $F_s = 96, 48, 44.1$ or 32 kHz.

The VCR output is not available with MP3.

47.1.4 Sampling frequencies

Sampling frequencies of $F_s = 96, 48, 44.1, 32$ kHz and half sampling frequencies are supported. A downsampling filter (96/48 kHz) for PCM data is available. MP3 also supports sampling frequencies of 24, 22.05, 16, 11.025 and 8 kHz.

47.1.5 Special modes

The decoder supports dual mode for MPEG and Dolby® Digital. It is karaoke aware and capable in Dolby® Digital and MPEG formats, according to DVD specifications. It includes a Dolby® surround compatible downmix and a Pro Logic® decoder.

A pink noise generator is also available to help position speakers accurately for optimal surround sound setup.

PCM beep tone is a special mode used for set-top boxes. It generates a triangular wave signal of variable frequency and amplitude on the left and right channels. The intended use is for satellite dish alignment (22 kHz) and not for generating a beep tone through the television speaker. This is activated by using PCM mixing to play any type of sound effects mixed simultaneously with MPEG stream decoded audio.

In global mute mode, the decoder decodes the incoming bit stream normally, but the PCM and S/PDIF outputs are softmuted. This mode is used to prepare a period of decoding mode that synchronizes audio and video data without sounding the audio.

47.1.6 PCM mixing

The device has PCM mixing capability with associated sample rate conversion. It includes a second PCM input for the reception of PCM data.

Note: It is not possible to mix a second PCM input when the main input is stopped.

47.1.7 Start procedure

1. Set register **AUD_BREAKPOINT** (address 0x2B) to 8.
2. Set register **AUD_CLOCKCMD** (0x3A) to 0 to start the DSP clock.
3. Wait until register **AUD_INT_RAM** (0xFF) indicates the hard registers are initialized (**AUD_INT_RAM** = 1).
4. Load the host register's configuration, including the LPCM downmix coefficients.
5. Set register **AUD_RUN** to 1 to start the request for data and the decoding.
6. Set **AUD_PLAY** to 1 and **AUD_MUTE** to 0. The clocks are started as soon as the first data is ready to be output, that is after one block decoding.

47.1.8 Architecture

The audio decoder has a programmable core, optimized for audio decoding algorithms. Dedicated hardware performs bit stream depacking and IEC data formatting.

Figure 163: Architecture and data flow

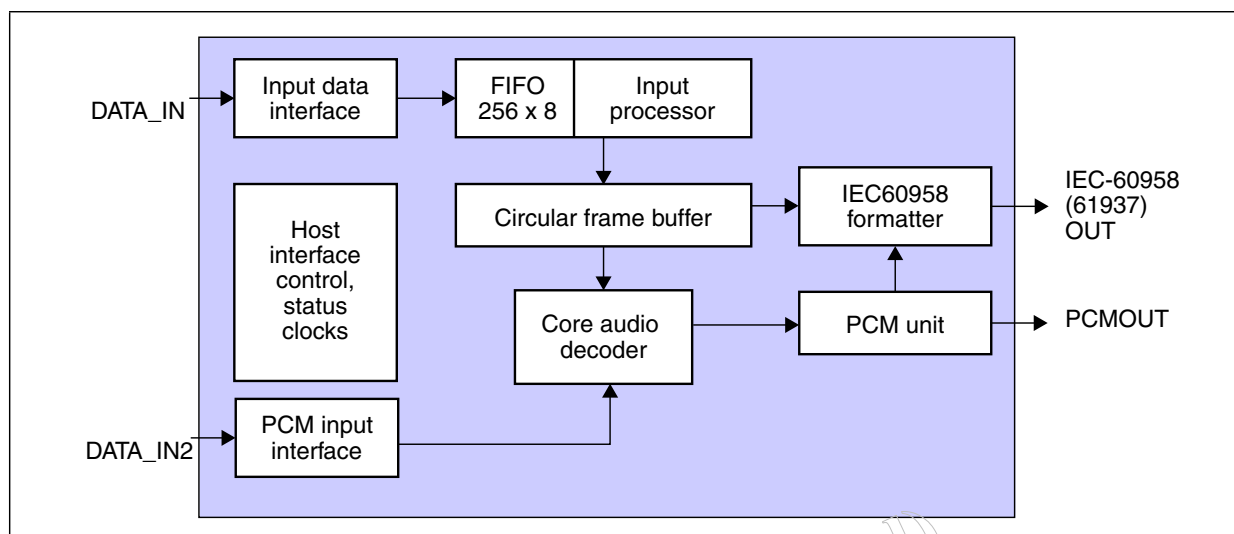
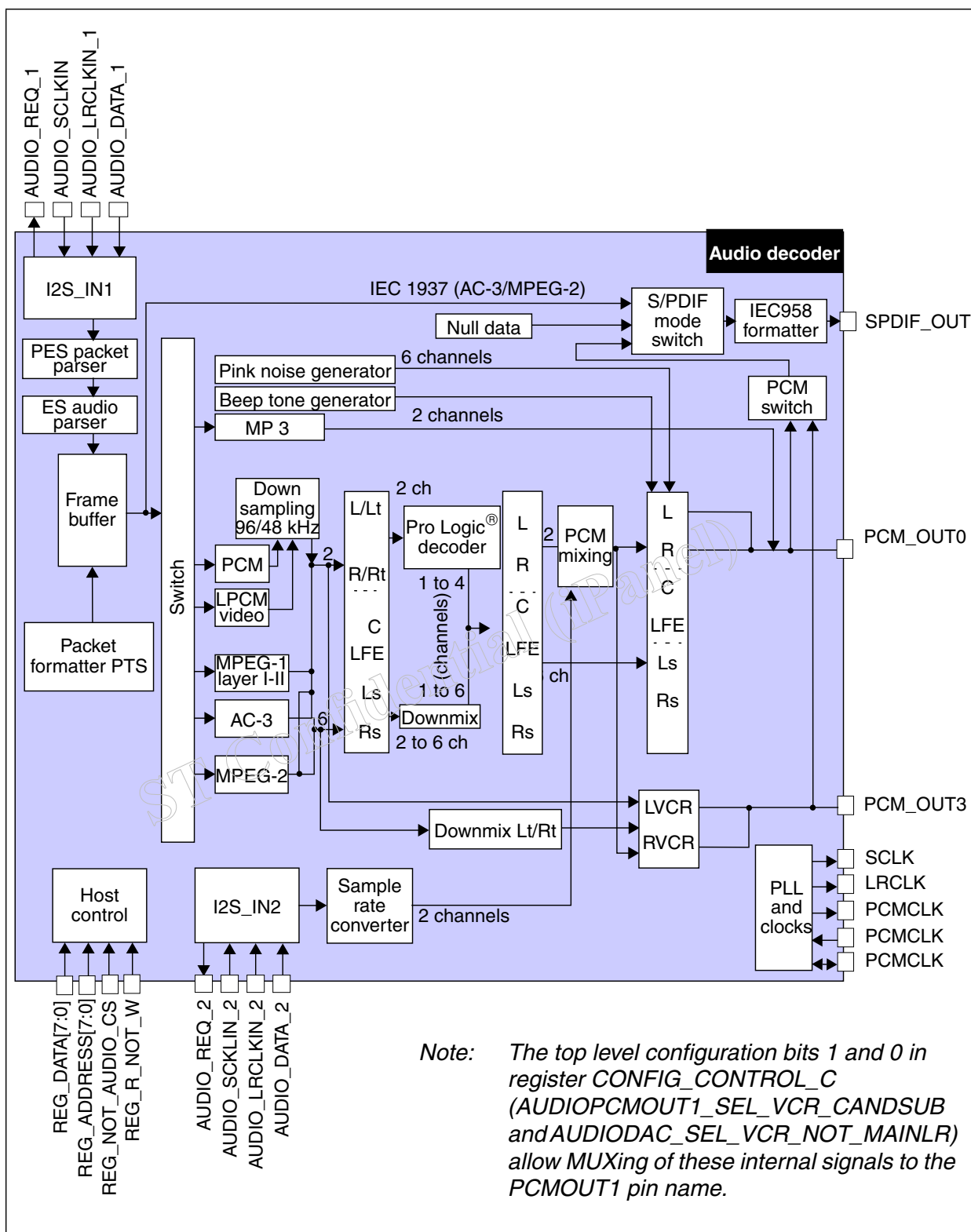


Figure 163 illustrates the audio decoder data flow. The compressed bit stream is transferred from the audio bit buffer (which is mapped into external SDRAM memory) to the audio decoder, via the MPEG DMA, which filters a 64-byte FIFO. When the FIFO is filled, data is transmitted from the FIFO to the audio decoder.

1. The input processor (composed of a packet parser and an audio parser) unpacks the bit stream (packet parser) and verifies its syntax (audio parser).
2. The compressed audio frames with their associated information (PTS) are stored in the circular frame buffer.
3. While a second frame is stored in the circular frame buffer, the audio core decoder extracts and decodes the first frame into audio samples.
4. A PCM input captures incoming PCM samples and copies them to the main memory. A sample rate converter processes and mixes them to the left/right PCM output.
5. The PCM unit converts the samples to PCM format.
6. Simultaneously, the IEC unit transmits compressed or noncompressed data.
 - In compressed mode, data is extracted directly from the circular buffer and formatted according to the IEC-61937 standard.
 - In noncompressed mode, the IEC unit outputs either the left and right, or the LVCR and RVCR, PCM channels formatted by the PCM unit according to the IEC-60958 standard.

Figure 164: Audio decoder block diagram



Confidential

47.2 Decoding process

Decoding is performed in the following stages. The configuration registers can activate or bypass each stage:

Table 163: Audio decoding stages

Parsing	The input processor parses the bit stream. Parsing discards all of the nonaudio information so that only the elementary audio stream (Dolby® Digital, MPEG-1, LPCM, PCM, MP3) is transmitted to the next stage (the circular frame buffer). The parsing stage operates in two phases: the packet parser unpacks the stream, the audio parser checks the syntax of the bit stream.
Main decoding	An elementary stream is input and decoded samples are output from this stage. Registers AUD_AC3_DOWNMIX , AUD_MP_DOWNMIX (one to six channels) define the number of output channels. Dolby® Digital, MPEG-1 layers I and II, MPEG-2 layer II, LPCM and MP3 decoding formats are supported. The appropriate stream format must be set by registers AUD_STREAMSEL and AUD_DECODESEL before running the decoder.
Post decoding	Postdecoding includes specific PCM processing: DC filter, de-emphasis filter and downsampling filter. The DC and de-emphasis filters can be independently enabled or disabled by register AUD_PDEC , the downsampling filter by register AUD_DWSMODE . Postdecoding also provides a Pro Logic® decoder, described in Section 47.6.5: Pro Logic® decoding modes on page 470 . A specific Pro Logic® compatible downmix of the main six channels can be applied to VCR outputs and data from second input can be mixed to main left/right output.
Volume control	The volume is controlled independently for each channel in steps of 1 dB, by registers AUD_CHAN_IDX , AUD_VOLUME0 and AUD_VOLUME1 .

Confidential

47.3 Operation

47.3.1 Reset

Software resets the audio decoder using the following procedure.

1. Soft reset: 1 is written to register [AUD_SOFTRESET](#) to reset the audio decoder.
2. The interrupt related registers ([AUD_INTE](#), [AUD_INT](#), [AUD_ERROR](#)) and command registers ([AUD_SOFTRESET](#), [AUD_RUN](#), [AUD_PLAY](#), [AUD_MUTE](#), [AUD_SKIP_MUTE_CMD](#), [AUD_SKIP_MUTE_VALUE](#)) are reset to zero.
3. The volume registers are reset to 0, no other decoding configurations are changed. The DSP returns to idle mode.

47.3.2 Clocks

The following clocks are used by the audio decoder:

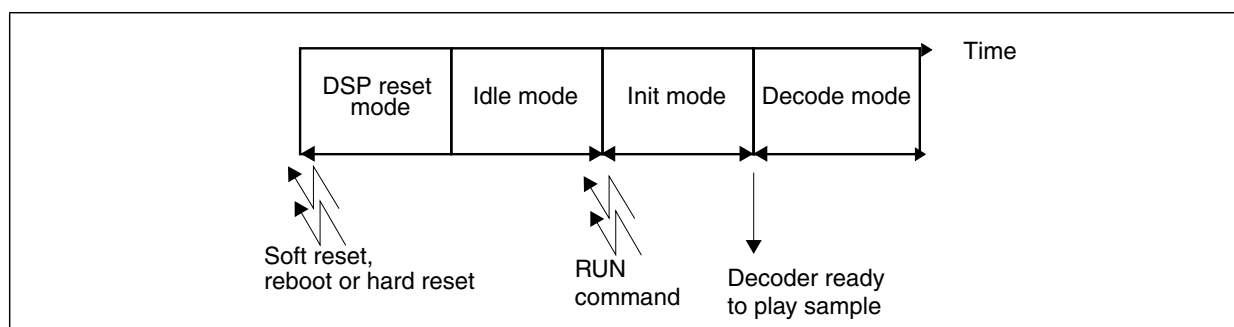
Table 164: Audio decoder clocks

The PCM clock	(PCMCLK signal) used by the external DACs to convert PCMOUT1. An embedded PLL from the 27 MHz clock input usually generates the signal. If necessary an external PLL can also generate it. The internal frequency synthesizer can generate $256 \times F_s$ or $384 \times F_s$, where $F_s = 12$ (MP3 only), 32, 44.1, 48, 96, 16, 22.05 or 24 kHz.
Audio system PLL	The system PLL creates the audio system clock from the 27 MHz input clock.
Bit clock SCLK	The PCM serial clock is the bit clock. It provides clocks for each time slot (16 cycles for each channel in 16-bit mode, 32 cycles for each channel in 18-, 20-, 24-bit modes). The frequency of SCLK is, therefore, fixed to $2 \times N_b \text{ time slots} \times F_s$, where F_s is the sample frequency. The clock is derived from PCMCLK. Register AUD_PCMDIVIDER must be configured according to the selected output precision and the frequency of PCMCLK, so that the device can construct SCLK: $FSCLK = FPCMCLK / \{2 \times (\text{AUD\_PCMDIVIDER} + 1)\}$ giving: $\text{AUD\_PCMDIVIDER} = FPCMCLK / (2 \times FSCLK) - 1$
Word clock LRCLK	The frequency of LRCLK is given by: FLRCLK = FSCLK/32; for 16-bit PCM output, FLRCLK = FSCLK/64; for 18-, 20- or 24-bit PCM output. No special configuration is required. Bit INV in register AUD_PCMCONF changes the polarity (see Section 47.7: PCM output on page 472).

47.4 Decoding states

There are two decoding modes: idle and decode (see Figure 165). Register `AUD_RUN` changes the mode.

Figure 165: Decoding states



47.4.1 Reset mode

After a hardware or software reset, the DSP software resets itself then enters the idle mode.

47.4.2 Idle mode

In this mode, the embedded DSP does not decode, that is, no data is processed; the chip is waiting for the **run** command. All configuration registers must be initialized during this mode. In idle mode, even if the chip is not processing data, the DAC clocks can be output, enabling the setup of the external DACs. Once the PCMCLK, SCLK and LRCLK clocks are configured, they can be output by setting register `AUD_MUTE`.

Table 165: Idle mode, play and mute command effects

PLAY	MUTE	Clock (SCLK, LRCLK) state	PCM output
X	0	Not running if PLAY is set to 0 after reset. ^a	0
X	1	Running	0

a. The **play** command has no effect in this state, as the decoder is not running. It can, however, be issued and may be executed as soon as the decoder enters the decode mode.

47.4.3 Decode mode

This state is entered after the **run** command has been sent, that is, when register `AUD_RUN` = 1. In this mode, data is processed; the decoder can play sound, or mute the outputs by using registers `AUD_PLAY` and `AUD_MUTE`.

To decode and output streams, register `AUD_PLAY` must be set. If register `AUD_MUTE` is reset, sound is sent to outputs; if register `AUD_MUTE` is set, outputs are muted.

Table 166: Decode mode, play and mute command effects^a

AUD_PLAY	AUD_MUTE	Clock state	PCM output	Decoding
0	0	Not running	0	No
0	1	Running	0	No
1	0	Running	Decoded samples	Yes
1	1	Running	0	Yes

- a. Configuration registers cannot be changed in this state, the chip must be soft reset beforehand. Only the following registers can be changed on the fly: [AUD_CHAN_IDX](#), [AUD_VOLUME0](#), [AUD_VOLUME1](#), [AUD_OCFG](#), [AUD_AC3_DOWNMIX](#), [AUD_MP_DOWNMIX](#).

47.5 Stream parsers

The synchronization status of both parsers is provided in register [AUD_SYNC_STATUS](#). Each time the synchronization status of one of the two parsers changes, the interrupt SYN is generated (if enabled) and the status can be read in [AUD_SYNC_STATUS](#).

47.5.1 Packet parser

The packet parser unpacks the stream, sorts packets and transmits data to the audio parser. Before unpacking packets and transmitting data, the packet parser must detect the packet start by recognizing the packet synchronization word.

The parser can be set to search for two packet synchronization words before starting to unpack and transmit, by setting register [AUD_PACKET_LOCK](#) to 1. Otherwise, the packet parser starts handling the stream once it has detected information matching the packet synchronization word.

The packet parser is also able to perform selective decoding. It can decode audio packets that match a specified ID. This ID is specified in registers [AUD_ID](#) and [AUD_ID_EXT](#); the function is enabled by setting register [AUD_ID_EN](#).

47.5.2 Audio parser

The audio parser verifies the stream syntax, and separates audio from nonaudio data, sending audio data to the frame buffer. The parser must detect the audio synchronization word corresponding to the type of stream to be decoded.

The parser can be set to detect more than one synchronization word before parsing, by setting register [AUD_SYNC_LOCK](#) to a value between 1 and 3. This number represents the number of supplementary sync words to detect before being synchronized.

47.6 Decoding modes

47.6.1 Dolby® Digital decoding modes

The decoder must be programmed to decode a Dolby® Digital encoded audio bit stream by setting register [AUD_DECODESEL](#) to 0.

The following modes refer to different implementations of the dialog normalization and dynamic range control features. The mode is selected by programming register [AUD_AC3_COMP_MOD](#).

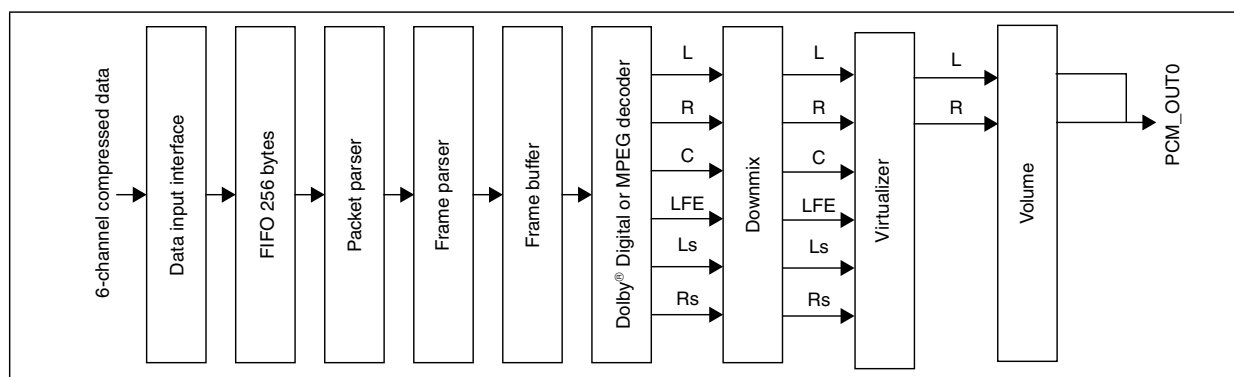
Table 167: Decoding modes

Line	In line mode (AUD_AC3_COMP_MOD = 2), dialog normalization is always enabled. This is done by the decoder itself and dialog is reproduced at a constant level. The two scaling registers AUD_AC3_HDR (for high level cut compression) and AUD_AC3_LDR (for low level boost compression) use and scale the dynamic range control variable encoded in the bit stream. For 2 front/0 rear downmix, the high level cut compression is not scalable.
RF	In RF mode (AUD_AC3_COMP_MOD = 3), the decoder performs dialog normalization. Dialog is reproduced at a constant level. The dynamic range control and heavy compression variables encoded in the bit stream are used, but compression scaling is not allowed. This means that registers AUD_AC3_HDR and AUD_AC3_LDR cannot be used in this mode. An 11 dB gain shift is applied on the output channels.
Custom 0	In custom 0 mode (AUD_AC3_COMP_MOD = 0), the decoder does not perform dialog normalization; this must be done by external circuitry. The two scaling registers AUD_AC3_HDR (for high level cut compression) and AUD_AC3_LDR (for low level boost compression) use and scale the dynamic range control variable encoded in the bit stream.
Custom 1	In custom 1 mode (AUD_AC3_COMP_MOD = 1), the decoder performs dialog normalization. The two scaling registers AUD_AC3_HDR (for high level cut compression) and AUD_AC3_LDR (for low level boost compression) use and scale the dynamic range control variable encoded in the bit stream.

47.6.2 MPEG decoding modes

MPEG-1 layer I and layer II encoded data are decoded, as well as MPEG-2 layer II data with or without extension (that is, 6-channel streams). The MPEG input format must be specified in register [AUD_DECODESEL](#), where [AUD_DECODESEL](#) equals 1 for MPEG-1 and equals 2 for MPEG-2. The dataflow is show in [Figure 166](#).

Figure 166: 6-channel compressed data decoding flow



47.6.3 Dual mode decoding modes

In dual mode, two completely independent mono program channels (for example, bilingual) are encoded in the bit stream, referred to as channel 1 and channel 2. Register [AUD_MP_DUALMODE](#) in MPEG format, and register [AUD_AC3_DUALMODE](#) in Dolby® Digital format set the left/right output to the following options:

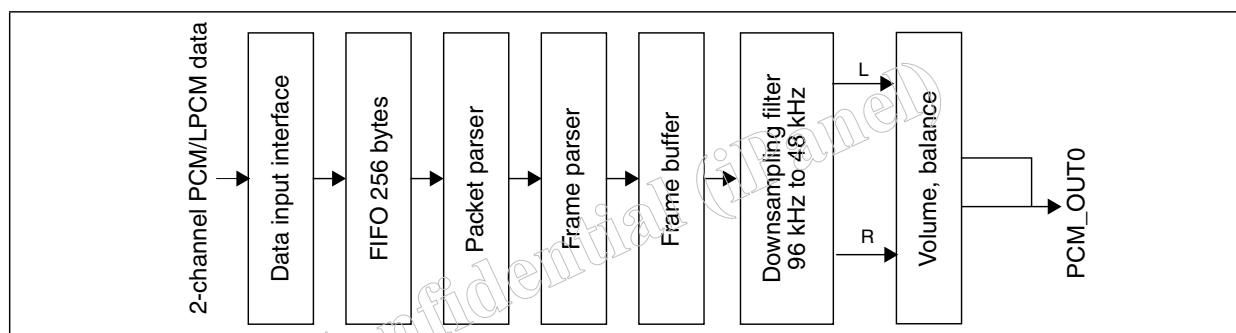
- output channel 1 on both L/R outputs,
- output channel 2 on both L/R outputs,
- mix channels 1 and 2 to monophonic and output on both L/R,
- output channel 1 on left output, and channel 2 on right output.

47.6.4 PCM decoding modes

The decoder supports PCM when register [AUD_DECODESEL](#) equals 3.

When decoding PCM streams encoded at 96 kHz, register [AUD_DWSMODE](#) configures the filter that downsamples the stream from 96 to 48 kHz.

Figure 167: PCM decoding flow



47.6.5 Pro Logic® decoding modes

Pro Logic® compatible downmix: a multichannel bit stream can be decoded and downmixed to provide a 2-channel output compatible with Pro Logic® (Lt, Rt). Registers [AUD_AC3_DOWNMIX](#) and [AUD_MP_DOWNMIX](#) select this downmix. The two channels can be used as the input of a Pro Logic decoder and player (for example, home theatre).

Pro Logic® decoding: a 2-channel Pro Logic® bit stream can be decoded. The two channels could come from a Dolby® Digital 2-channel bit stream, a LPCM or an MPEG-1 bit stream. The 2-channel bit stream can be converted into a 4-channel output (L, R, C, S). The surround (S) is simultaneously sent on Ls and Rs channels. A Pro Logic® downmix enables the channels to output on PCM data to be configured. This is done through register [AUD_PL_DWNX](#).

Figure 168: Dolby® Digital and Pro Logic® decoding flow

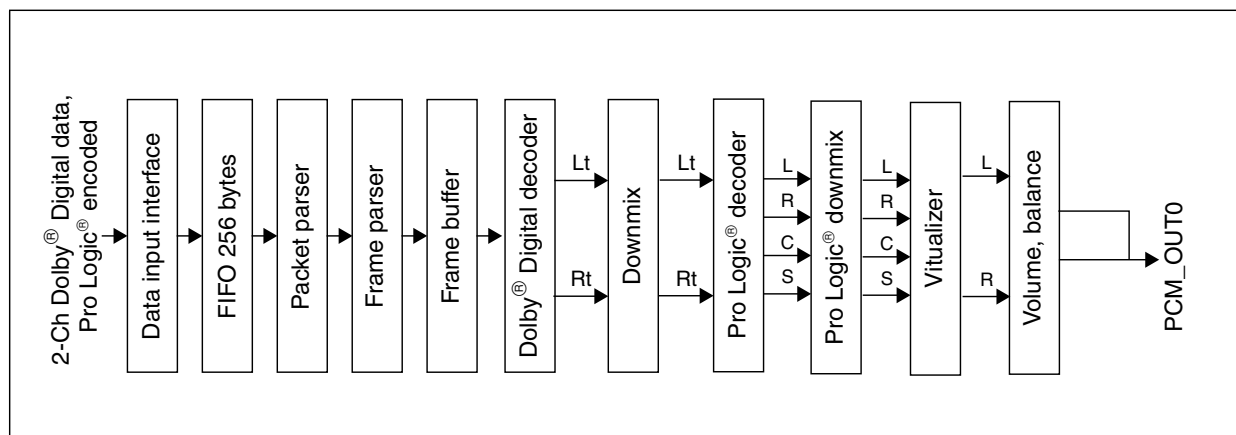


Figure 169: MPEG and Pro Logic® decoding flow

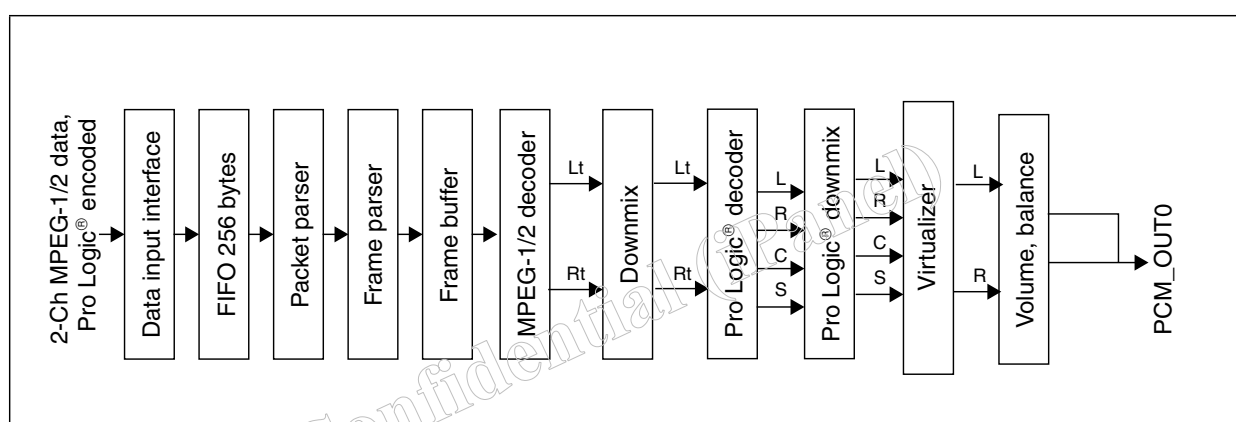
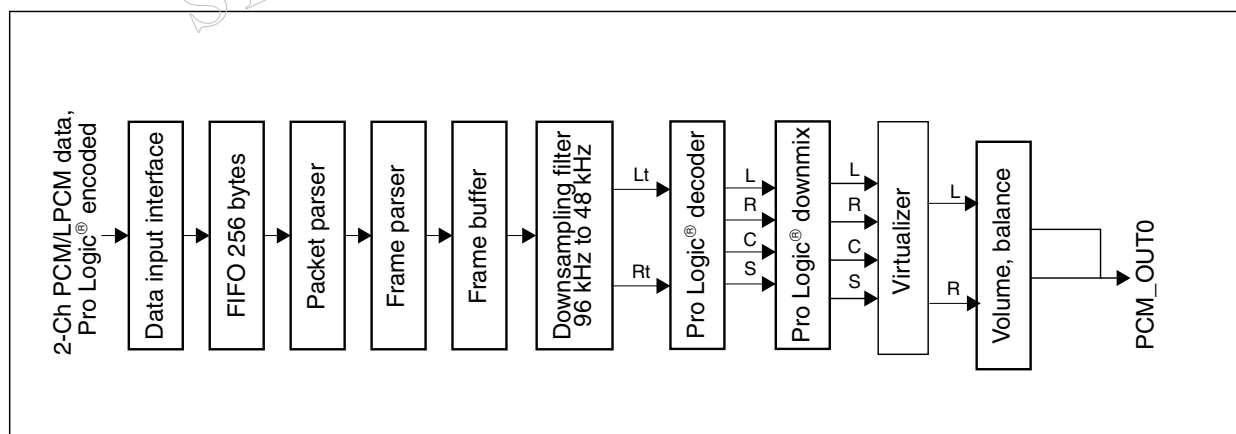


Figure 170: PCM and Pro Logic® decoding flow



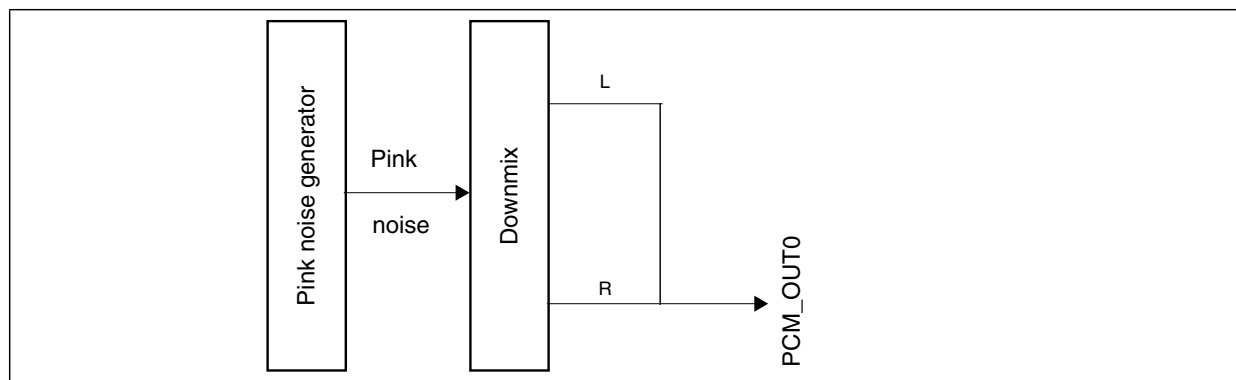
47.6.6 Pink noise decoding modes

The pink noise generator is used to position the speakers in the listening room for optimum sound quality.

The decoder is programmed to generate pink noise by writing the value 4 in register [AUD_DECODESEL](#) and 3 in register [AUD_STREAMSEL](#). Register [AUD_PN_CHANNELCONF](#) selects the pink noise output channels.

For pink noise generation, the register configuration should be: [AUD_OCFG](#) = 0 and volume control set to 0 dB.

Figure 171: Pink noise decoding flow



Note: The appropriate pink noise level is obtained by attenuating all outputs by 10 dB through volume registers.

47.6.7 MP3 decoding mode

MP3 supports the following frequencies: 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, and 48 kHz.

Downmix, postprocessing and PCM delay are not supported in MP3 mode, also fast forward and automatic upsampling are not available. When in MP3 mode, the VCR output cannot be used.

Volume control is possible in this mode if register [AUD_OCFC](#) is set to zero.

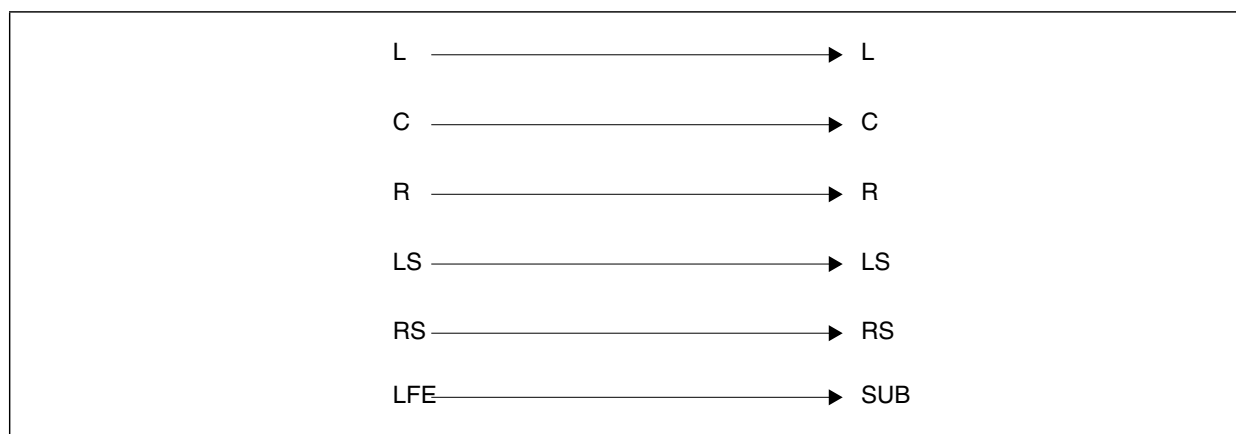
47.7 PCM output

47.7.1 Output configuration

[Figure 172](#) shows the configuration at the PCM output stage. Outputs are scaled and rounded (see [Section 47.7.2](#)).

Note: Register [AUD_OCFC](#) must always be set to 0.

Figure 172: PCM output configurations



47.7.2 PCM scaling

PCM scaling is required for each decoding mode. It is applied at the end of the filtering steps, before PCM output, allowing maximum effective word width for most of the prior signal processing.

Each channel has independent volume control for PCM scaling (registers [AUD_CHAN_IDX](#), [AUD_VOLUME0](#), [AUD_VOLUME1](#)).

47.7.3 Output quantization

For 16-, 18- and 20-bit DACs, quantization with rounding is applied together with the PCM scaling. The sample value is multiplied by a rounding factor and rounded to 24 bits. The result is then left shifted (4/6/8) for PCM output. The output precision is selectable from 16 to 24 bits/word by configuring PREC (register [AUD_PCMCONF](#)).

47.7.4 Interface and output formats

The decoded audio data is output in serial PCM format. The interface signals are detailed in [Table 168](#).

Table 168: Interface formats

PCMOUT[0:3]	PCM data output
SCLK	Bit clock (or serial clock) output
LRCLK	Word clock (or left/right channel select clock) output
PCMCLK	PCM clock input or output

47.7.5 Output precision and format selection

The PCM output is set in register [AUD_PCMCONF](#), bit fields ORD, DIF, FOR and PREC.

- PREC sets the output precision from 16 to 24 bits/word.
- In 16-bit mode, data can be output either with the MSB or LSB first, by setting ORD.
- When PREC is set for more than 16 bits, 32 bits are output for each channel.
- In this configuration, bit FOR selects either I²S or Sony formats; DIF positions the 18, 20 or 24 bits either at the beginning or the end of each 32-bit frame.

[Figure 173](#) and [Table 169](#) describe the different output formats, followed by two configuration examples

The diagram illustrates the timing relationship between the LRCLK clock signal and the PCMOUT[2:0] data signal for two different modes: 16 SCLK cycles and 32 SCLK cycles.

16 SCLK cycles mode:

- LRCLK:** A square wave pulse representing 16 SCLK cycles.
- PCMOUT[2:0]:** A 3-bit data signal. For ORD = 0, it shows a sequence of bits: M, L, M, L, S, S, S, S. For ORD = 1, it shows: L, M, L, M, S, S, S, S.

32 SCLK cycles mode:

- LRCLK:** A square wave pulse representing 32 SCLK cycles.
- PCMOUT[2:0]:** A 3-bit data signal. For FOR = 1 and DIF = 1, it shows: M, S, 18, 20 or 24 bits, L, S, 0, M, S, 18, 20 or 24 bits, L, S, 0, FOR = 1, DIF = 1. For FOR = 0 and DIF = 0, it shows: 0, M, S, 18, 20 or 24 bits, L, S, 0, M, S, 18, 20 or 24 bits, L, S, 0, FOR = 0, DIF = 0. For FOR = 0 and DIF = 1, it shows: 0, M, S, 18, 20 or 24 bits, L, S, 0, 0, M, S, 18, 20 or 24 bits, L, S, 0, FOR = 0, DIF = 1. For FOR = 1 and DIF = 0, it shows: MSB, M, S, 18, 20 or 24 bits, L, S, MSB, M, S, 18, 20 or 24 bits, L, S, FOR = 1, DIF = 0.

Note: Only PCMOUT0 or PCMOUT3 can be output on the STi5516. The top level configuration bits 1 and 0 in register CONFIG_CONTROL_C (AUDIOPCMOUT1_SEL_VCR_CANDSUB and AUDIODAC_SEL_VCR_NOT_MAINLR) allow MUXing of these internal signals to the PCMOUT1 pin.

Table 169: PCM output formats

AUD_PCMCONF register settings				Data in sample Memory data [23:0] ^a	Data sent on the PCM serial output (left bit first)	
PREC	ORD	FOR	DIF			
0:16-bit mode	1	-	-	{d23 - d8} - {8 x 0}	{d8 - d23}	16 bits
	0	-	-		{d23 - d8}	
1:18-bit mode	-	0	0	{d23 - d6} - {6 x 0}	{13 x 0} {0} {d23 - d6}	32 bits
			1		{0} {d23 - d6} {13 x 0}	
		1	0		{14 x d23} {d26 - d6}	
			1		{d23 - d6} {14 x 0}	
2:20-bit mode	-	0	0	{d23 - d4} - {4 x 0}	{11 x 0} {0} {d23 - d4}	
			1		{0} {d23 - d4} {11 x 0}	
		1	0		{12 x d23} {d23 - d4}	
			1		{d23 - d4} {12 x 0}	
3:24-bit mode	-	0	0	{d23 - d0}	{7 x 0} {0} {d23 - d0}	
			1		{0} {d23 - d0} {7 x 0}	
		1	0		{8 x d23} {d23 - d0}	
			1		{d23 - d0} {8 x 0}	

a. The internal 24-bit decoded, scaled and rounded audio samples are listed as they are stored in memory. These 24 bits are referred to as d23, d22,..., d0, where MSB = d23, LSB = d0.

Configuration example 1

In 16-bit mode, with ORD = 1: in memory, 24 bits are stored, where only the 16 MSB (d23 to d8) are significant and the eight remaining bits are 0. This is noted: {d23 - d8} {8 x 0}. The data is sent LSB first, that is, d8 is sent first and d23 is sent last. This is noted {d8 - d23}. 16 bits only are transmitted per channel.

Configuration example 2

In 20-bit mode (ORD is meaningless in this mode), with FOR = 1 and DIF = 0: in memory, 24 bits are stored, where only the 20 MSB (d23 to d4) are significant and the remaining four LSB are 0. This is noted: {d23 - d4} {4 x 0}. 32 bits are transmitted per channel on the PCM outputs: the first 12 transmitted bits are d23, the last bits are d23 to d4, where d23 is transmitted first. This is noted: {12 x d23} {d23 - d4}.

Only one PCM configuration can be supported at a time. If the internal audio DAC and external audio DACs are to be used simultaneously, the external DAC must support the internal format, based on the I²S standard. This is shown in [Figure 174](#), [Figure 175](#) and [Table 170](#). In this format, the left and right data channels are time-multiplexed. The serial data is sampled in the DAC at the rising edge of S_{CLK}.

Figure 174: Reset timing

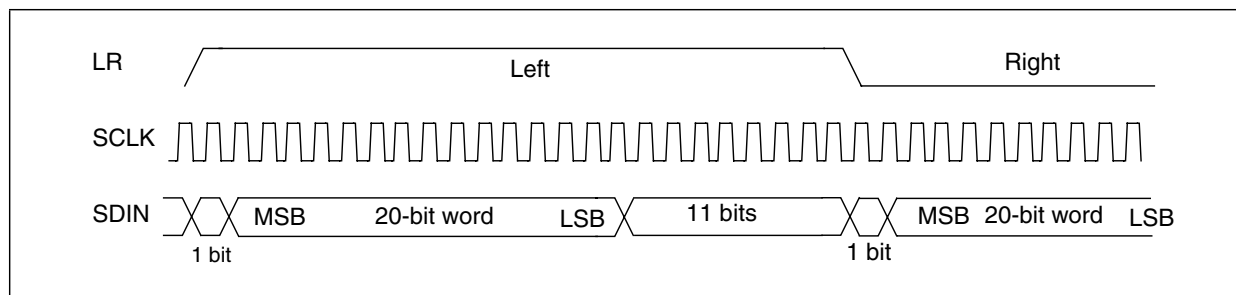


Figure 175: Serial input timing

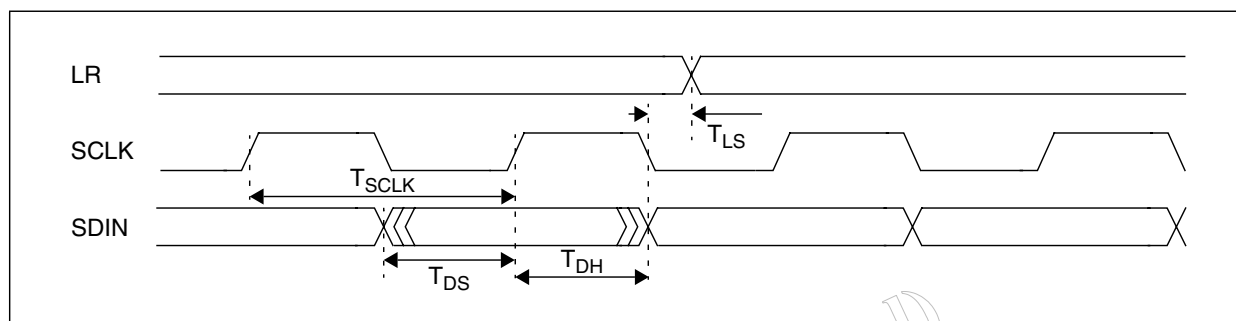


Table 170: Serial input timings

Symbol	Parameter	Min	Typ	Max	Units
T_{SCLK}	Bit clock pulse cycle time	Normal	320		ns
		X2 mode	160		ns
T_{LS}	LR edge to SCLK falling edge	-25		25	ns
T_{DS}	SDIN set up time	50			ns
T_{DH}	SDIN hold time	50			ns
T_{SETUP}	PDIN set up time	0.25			T_{LR}

Double mode is selected by setting a high level on X2.

47.7.6 Clock polarity

The fields SCL and INV of register [AUD_PCMCONF](#) select the polarity of the PCM serial output clock (SCLK) and the PCM word clock (LRCLK) respectively.

[Figure 176](#) shows the polarities of SCLK and LRCLK.

When SCL is set to 0, the DAC samples LRCLK and PCMOUT on the rising edge of SCLK, and when SCL is set to 1, on the falling edge.

Figure 176: SCLK and LRCLK polarity selection

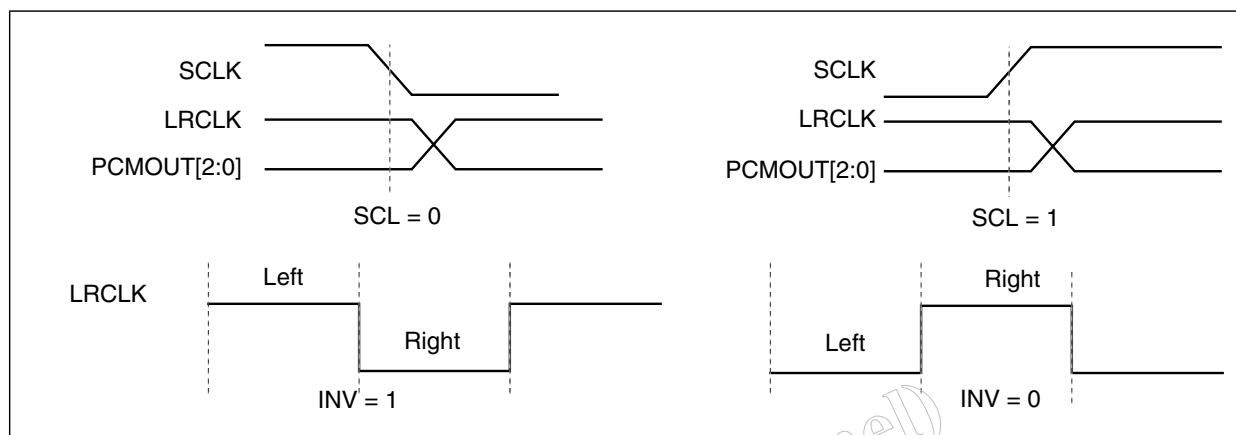


Table 171: PCM configuration for I²S and Sony compatible outputs

Register configuration		Output format compatibility	
		I ² S	Sony
AUD_PCMCONF	DIF	1: Not right padded	
	FOR	0: I ² S	1: Sony
	INV	0: Do not invert LRCLK	1: Invert LRCLK
	SCL	0: Do not invert SCLK	0: Do not invert SCLK

47.8 S/PDIF output

47.8.1 Overview of the S/PDIF output

The S/PDIF output pad is a TTL output pad with slew rate control. The DC output capability is 4 mA and the voltage drop is 3 V. This output must be connected to a TTL driver before being connected to a transformer.

The S/PDIF output supports IEC-60958 and IEC-61937 standards. The following registers must be initialized to configure the S/PDIF output.

- The category code must be entered in register [AUD_SPDIF_CAT](#). This code is related to the type of application, and is specified in the digital output interface standard.
- The status bits to be transmitted on the S/PDIF output must be programmed in register [AUD_SPDIF_STATUS](#).
- IEC clock setting must be specified in register [AUD_SPDIF_CONF](#).
- Data type dependent information can be specified in register [AUD_SPDIF_DTDI](#).
- The S/PDIF type is selected through register [AUD_SPDIF_CMD](#). The IEC unit can output decoded data (PCM mode), encoded data, null data or pause bursts.

When configured in IEC-60958 mode, the S/PDIF output is used to transmit the decoded left and right channels. This is selected by choosing the PCM mode in register [AUD_SPDIF_CMD](#) and resetting the COM status bit in register [AUD_SPDIF_STATUS](#).

When configured in IEC-61937 mode, the S/PDIF output is used to transmit encoded data taken directly from the frame buffer. This is selected by choosing the encoded mode (bit CMD) in register [AUD_SPDIF_CMD](#) and setting the bit COM in register [AUD_SPDIF_STATUS](#). The decompressed data is output simultaneously on the PCM outputs.

When encoded S/PDIF data is output, a latency is inserted between the S/PDIF and PCM outputs. The PCM outputs are delayed with regard to the S/PDIF output. The latency is automatically set when bit LAT is 0 (register [AUD_SPDIF_CONF](#)). Standards define the value in autolatency mode. To control the latency manually, set LAT to 1 and define the latency in seconds by use of the register [AUD_SPDIF_LATENCY](#).

When configured in muted mode (register [AUD_SPDIF_CMD](#)), the outputs are PCM null data. This can be used to synchronize the external IEC receiver. Register [AUD_SKIP_MUTE_CMD](#) bit MUTE is used to transmit bursts of pause frames in IEC-61937 format.

47.9 Interrupts

47.9.1 Interrupt register

The audio decoder contains a 16-bit interrupt register [AUD_INT](#) associated with a 16-bit enable register [AUD_INTE](#). A bit set in [AUD_INTE](#) enables the corresponding interrupt. The interrupt associated with each bit is given in the [AUD_INT](#) description.

According to the type of interrupt, other information such as stream header, type of error detected or PTS value can be obtained by reading associated registers.

47.9.2 Error concealment

The audio core signals errors as interrupts. The core automatically handles most errors, but some require software action. Error categories are defined in register [AUD_ERROR](#).

Dolby® Digital decoding errors are signaled in [AUD_ERROR](#) but handled directly by the core. Software cannot change these errors. Dolby® Digital decoding errors signal that something went wrong during decoding. The core soft mutes the frame and continues to decode.

MPEG decoding errors are signaled in [AUD_ERROR](#) but are handled directly by the core. Nothing can be done by the software. They signal that something went wrong during the decoding. The core soft mutes the frame and continues decoding. Only one error in this category indicates a programming error. If triggering MPEG_EXT_CRC_ERROR, bit MC (register [AUD_MP_MC_OFF](#)) must be set. This indicates that the decoder tried to decode more than two channels whereas the incoming stream contains only two channels.

Packet and audio synchronization errors are handled internally, and usually indicate that the incoming bit stream is incorrect or that it has been incorrectly input to the chip. In these cases, the decoder resets the corresponding parsing stage (packet or audio parser), then searches for the next correct frame.

The miscellaneous error LATENCY_TOO_BIG indicates that the latency has been programmed greater than the maximum authorized value. The latency value should be changed or a switch made to autolatency mode. Other miscellaneous errors are handled internally.

47.10 Audio/video synchronization

47.10.1 Presentation time stamp detection

When enabled through register [AUD_INTE](#), the interrupt PTS is generated when a PTS is present in the frame that is being output on PCMOUT (the interrupt is fired when the first decoded samples of the first block of the frame are output).

47.10.2 Pause frames capability

The number of audio blocks for the audio decoder to pause must be programmed in register [AUD_SKIP_MUTE_VALUE](#). Then bit BLK of [AUD_SKIP_MUTE_CMD](#) must be set. The audio decoder finishes decoding the current frame, soft mutes the next frame, and pauses for the number of blocks specified in [AUD_SKIP_MUTE_VALUE](#). When the pause is finished, decoding continues.

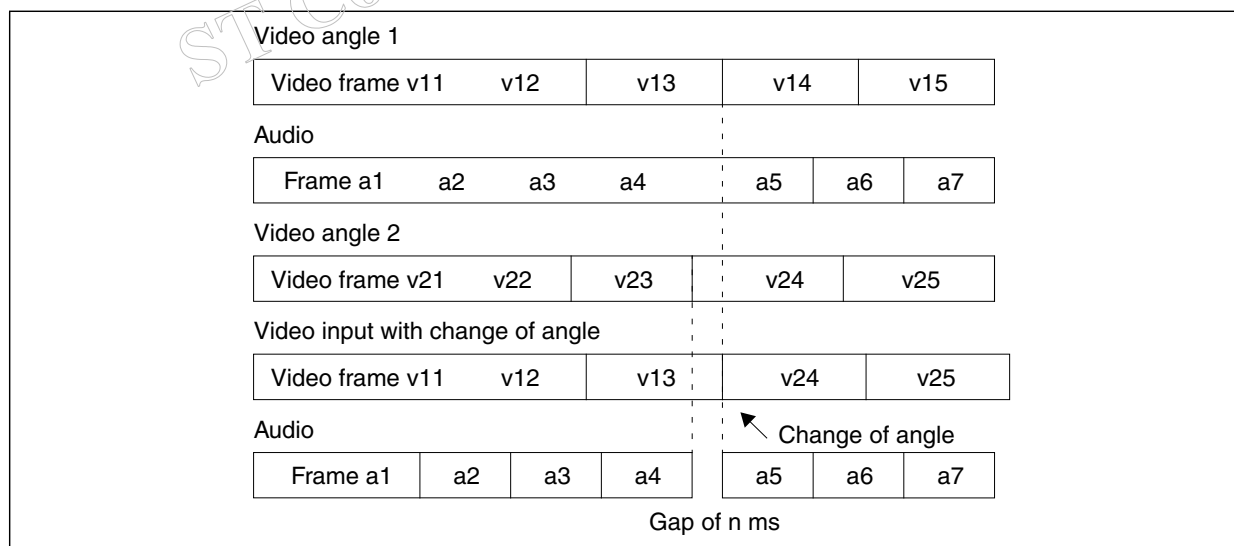
47.10.3 Skip frames capability

The number of frames to skip must be programmed in register [AUD_SKIP_MUTE_VALUE](#). Then bit SKP of [AUD_SKIP_MUTE_CMD](#) must be set. The audio decoder finishes decoding the current frame, soft mutes the next frame, and skips the number of frames specified in [AUD_SKIP_MUTE_VALUE](#). After skipping, it resumes decoding from the next incoming frame.

47.10.4 Pause burst capability

To synchronize video and audio outputs, the audio cell must be able to insert a pause on the output when required. This means that the audio decoder has to stop before decoding a new frame and the output of the audio has to be muted for a period of time as illustrated below.

Figure 177: Pause burst capability example illustration



Register [AUD_SKIP_MUTE_CMD](#) initiates a pause.

- If bit PAU is set, a pause is inserted until bit PAU is reset.
- If bit BLK is set, a pause burst is inserted for a duration set by [AUD_SKIP_MUTE_VALUE](#).

The granularity of the gap defined by this mechanism is:

- 256 sampling periods for AC-3 (5.3 ms at 48 kHz, 5.8 ms at 44.1 kHz),
- 96 sampling periods for MPEG (2 ms at 48 kHz).

47.10.5 Soft mute capability

The soft mute is effective from the next block boundary from the time when the application sets the MU bit of `AUD_SKIP_MUTE_CMD`. It is complete in less than 20 ms.

The time it takes to complete the soft mute is related to the decoder.

- In AC-3 the overlap and add technology ensures a 50% overlap between the previous block and the first muted block. This means that the muting is complete after one block, that is 256 samples.
- In MPEG the muting takes longer because the filter bank operates on a delay line of 512 samples. The muting is completed after 512 samples + 1 block (96 samples). This represents around two frames in MPEG-1 layer I (384 samples per frame) and only one frame for MPEG-1 layer II or MPEG-2 (1152 samples).
- For PCM, the signal mutes after 1 block (80 samples).

47.11 PCM beep tone

The PCM beep tone is a special mode used for set-top boxes. It generates a triangular signal of variable frequency and amplitude on the left and right channels. The intended use is for satellite dish alignment (22 kHz) and not for generating a beep tone through the television speaker. This is activated by using PCM mixing to play any type of sound effects mixed simultaneously with MPEG stream decoded audio. The amplitude of the PCM beep tone is 0 dB. The PCM beep tone can be sent to the S/PDIF output when the S/PDIF output is configured in PCM mode.

47.11.1 Activating PCM beep tone mode

1. Reset the DSP.
2. Set up the registers `AUD_DECODESEL` (0x4D) = 7 and `AUD_STREAMSEL` (0x4C) = 3.
3. Restart the DSP by asserting registers `AUD_RUN` and `AUD_PLAY`.

47.11.2 Changing the frequency

Set register `AUD_BEEP_FREQ` (0x68) according to the equation below:

$$\text{Beep\_tone frequency} = (F_s / 2) / (\text{register\_value} + 1)$$

47.11.3 Changing the amplitude

The amplitude of the PCM beep tone is 0 dB by default; to change the amplitude, set the registers below:

- `AUD_OCFG` (0x66) = 0,
- `AUD_CHAN_IDX` (0x67) = 0 (to select the channel pair [left and right]),
- `AUD_VOLUME0` (0x4E) = attenuation value (step of -1 dB) on left channel,
- `AUD_VOLUME1` (0x63) = attenuation value (step of -1 dB) on right channel.

The PCM beep tone can be sent to the S/PDIF output when the S/PDIF output is configured in PCM mode.

47.12 PCM mixing

A second serial input (I2S_IN2), which inputs PCM data can be used for PCM mixing with PCMOOUT0.

Note: It is not possible to mix a second PCM input when the main input is stopped.

I2S_IN2 accepts the same formats as the main I²S input I2S_IN1 (left/right aligned, 16-, 18-, 20- or 24-bit data word length). Two channel data (synchronized with LRCLK according to the specified format) is stored in a bank of four 16-bit registers. An interrupt is generated each time a stereo sample is received. The DSP moves the received data into a double buffer structure located in the 24-bit RAM. When the buffer is full, the DSP stops requesting new data.

AUDIO_REQ_2 is reset when the block sends an interrupt to the core. The software asserts AUDIO_REQ_2 at the end of the interrupt if the frame has not been fully received. When the DSP toggles the buffer, it sets AUDIO_REQ_2 to the next frame to be loaded.

The frame storage buffer is 1536 words wide. Size is defined from the largest frame size used in the decoder (256 for AC-3, 96 for MPEG, 80 or 160 for LPCM) taking into account the possible oversampling of the second input with respect to the main stream and the delta corresponds to a reservation in case of jitter or shift in sampling frequency between the two inputs.

To validate the PCM mixing configuration and **play** and **stop** commands, bit 0 of register [AUD_PCM MIX_UPDATE](#) must be set and the host must access (read or write) the generic [AUD_PLAY](#) register. This last action triggers an interrupt to the DSP core which then takes into account the new configuration and commands requested for the PCM mixing.

The polarity of AUDIO_REQ_2 is controlled through bit POL_2 of [AUD_SIN_SETUP](#) (when it is high, data must be input when AUDIO_REQ_2 = 0).

Double DMA injection

When mixing a second PCM audio stream from memory with an MPEG decoded stream from the PTI, use two PTI DMAs instead of one. The first one always writes the data at the real address (the useful DMA) and the second one always writes at the second address (the stuffing DMA). The stuffing DMA must always write at least one word between each write of the useful DMA.

Note: To simplify the implementation, put the stuffing DMA loop indefinitely in the same pattern, so that, once launched, additional CPU control is not required.

The procedure for implementing double DMA injection is listed below.

1. Set the PCMO request and attach it to PTI DMA1 and DMA3 (DMA2 has been used for legacy audio injection) by configuring CONFIG_CONTROL_A.

```
*(int *)0x20010000=0x05150000;
```
2. Disable the op code converter to allow configuration of the PCM interface registers by setting bit 14 of CONFIG_CONTROL_E.

```
*(int*) 0x20010028=0x4000;
```
3. Program the PCM configuration registers. For instance a 16 +16 bits unsigned.

```
*(int*)0x20500000=0x13;  
*(int*)0x20500004=0x201;  
*(int*)0x20500008=0xC01;  
*(int*)0x2050000C=0xC01;
```
4. Switch back on the op code converter, so the stream can start to be injected.

```
*(int*) 0x20010028=0x4000;
```
5. Inject an audio stream on compressed data input 1. This can be a live stream or a playback from a file, like in the standard case.

6. Prepare the mixing on the audio decoder.

For example a standard configuration for 16 +16 bits stereo.

```
*(char *)0x8ad=0;
*(char *)0x8ba=0x7f;
*(char *)0x8b2=0x58;
*(char *)0x8b1=0x16;
*(char *)0x895=0;
*(char *)0x8ab=0xf2;
*(char *)0x8ac=0x4;
*(char *)0x888=0x20;
*(char *)0x889=0x20;
*(char *)0x8fc=0x0;
*(char *)0x8ad=0x40; / start mixing */
```

7. Prepare the PTI DMAs. PTI DMA1 (the stuffing DMA) sends a constant pattern from the internal SRAM for instance, whereas PTI DMA3 (the useful DMA) sends the PCM stream. The mapping for the DMA is the TC3/PTI3 mode map.

```
*(int*)0x20021020=0x80000300; /* DMA1BASE */
*(int*)0x20021024=0x80000310; /* DMA1TOP */
*(int*)0x2002102c=0x80000300; /* DMA1READ */
*(int*)0x20021028=0x80000320; /* DMA1WRITE */
*(int*)0x20021030=0x00000000; /* DMA1SETUP */
*(int*)0x20021034=0x00040000; /* DMA1HOLDOFF */
*(int*)0x20021038=0x205000c4; /* DMA1CDADDR */
```

```
*(int*)0x20021060=PCMBase; /* DMA3BASE */
*(int*)0x20021064=PCMBase+PCMSize; /* DMA3TOP */
*(int*)0x2002106c=PCMBase; /* DMA3READ */
*(int*)0x20021068=PCMBase+PCMSize+16; /* DMA3WRITE */
*(int*)0x20021070=0x00000000; /* DMA3SETUP */
*(int*)0x20021074=0x00040001; /* DMA3HOLDOFF */
*(int*)0x20021078=0x205000c0; /* DMA3CDADDR */
```

8. Now launch the PTI DMA1 and DMA3 together.

Audio mixing should process correctly. In the example, the programming is done in order that both DMA's are looping back, so there is no task to setup and PCM should play indefinitely.

```
*(int*) 0x2002101C |= 0xA;
```

47.13 VCR output

PCMOUT3 outputs a Pro Logic® compatible downmix of the six channels of PCMOUT0 to PCMOUT2 allowing the recording of the multichannel audio with a stereo VCR.

In PCM mode, PCMOUT3 can be output through the S/PDIF by setting bit IECSELECT of [AUD_PCMCONF](#).

48 Audio decoder registers

The audio DSP device contains 256 8-bit registers. There are two register types:

- From address 0x00 to 0x3F, **real registers** that can be initialized after reset.
- From address 0x40 to 0xFF, **memory locations**. These can have different meanings and usage according to the mode in which the device operates, and cannot be hardware reset. At reset, they contain undefined values and must be re-initialized.

For example, if operating in AC-3 format, the register located at address 0x69 is [AUD_AC3_COMP_MOD](#). However, this address corresponds to register [AUD_MP_PROG_NUMBER](#) when decoding MPEG streams.

Read-only registers must never be written.

[Table 31: Audio decoder registers on page 49](#) lists the register map by address and function; each audio decoder register is then described individually.

Addresses are provided as the *AudioBaseAddress* + offset.

The *AudioBaseAddress* is:

0x0000 0800.

A register summary is given in [Table 31: Audio decoder registers on page 49](#).

48.1 Audio DSP start up registers

AUD_BREAKPOINT

Set breakpoints

7 6 5 4 3 2 1 0

AUD_BREAKPOINT

Address: *AudioBaseAddress*: + 0x2B

Type: Write only

Description: Set to 0x08 on start up.

AUD_CLOCKCMD

Start DSP clock

7 6 5 4 3 2 1 0

AUD_CLOCKCMD

Address: *AudioBaseAddress*: + 0x3A

Type: Write only

Description: Set to 0x00 to start the DSP clock. This allows the CPU clocks to be used by the audio cell.

AUD_INT_RAM**Status of hardware registers**

7 6 5 4 3 2 1 0

AUD_INT_RAM

Address: *AudioBaseAddress* + 0xFF

Type: Read only

Description: When set to 1, indicates the hardware registers are initialized and the audio DSP is ready.

48.2 Audio DSP version registers**AUD_VERSION****Version**

7 6 5 4 3 2 1 0

AUD_VERSION

Address: *AudioBaseAddress* + 0x0000

Type: Read only

Reset: 0x20 (soft)
0x20 (hard)

Description: This register gives the audio hardware version (binary decimal encoded).

AUD_IDENT**Identify**

7 6 5 4 3 2 1 0

AUD_IDENT

Address: *AudioBaseAddress* + 0x0001

Type: Read only

Reset: Not applicable

Description: This is a read-only register and is used to identify the IC on an application board.
AUD_IDENT always has the value 0x5A.

AUD_SOFTVER**Software version**

7 6 5 4 3 2 1 0

AUD_SOFTVER

Address: *AudioBaseAddress* + 0x0071

Type: Read/write

Reset: 0x0B (soft)
0x0B (hard)

Description: This register gives the version of the microcode running on the device. The register is updated after a soft reset of the device.

Confidential

48.3 RS232 activation registers

AUD_RS232_INTERF_ECHO

Enable RS232 input

7	6	5	4	3	2	1	0
Reserved						RS232_ON	Reserved

Address: *AudioBaseAddress* + 0x00EF

Type: Read/write

Reset: No change (soft)
0 (hard)

Description: The bit RS232_ON must be set to 1 to enable RS232 input.

48.4 Audio DSP setup and input registers

AUD_SIN_SETUP

Input data setup

7	6	5	4	3	2	1	0
Reserved				POL_2	POL_1	I_MODE	

Address: *AudioBaseAddress* + 0x000C

Type: Read/write

Reset: No change (soft)
0 (hard)

Description:

[7:4] **Reserved**

[3] **POL_2**: Polarity of AUDIO_REQ_2.

0: Signal AUDIO_REQ_2 is active high. Data must be input when AUDIO_REQ_2 is high.

1: Signal AUDIO_REQ_2 is active low. Data must be input when AUDIO_REQ_2 is low.

[2] **POL_1**: Polarity of AUDIO_REQ_1.

0: Signal AUDIO_REQ_1 is active high. Data must be input when AUDIO_REQ_1 is high.

1: Signal AUDIO_REQ_1 is active low. Data must be input when AUDIO_REQ_1 is low.

[1:0] **I_MODE**

00: Parallel input

01: Serial input (AUDIO_SCLK_1 + AUDIO_DATA_1 + AUDIO_REQ_1)

10: Reserved

11: A/D input (AUDIO_SCLK_1 + AUDIO_DATA_1 + AUDIO_REQ_1); S/PDIF input or PCM input.

AUD_CAN_SETUP**A/D converter setup**

7	6	5	4	3	2	1	0
Reserved				S16	SAM	FIR	PAD

Address: *AudioBaseAddress* + 0x000D

Type: Read/write

Reset: No change (soft)
0 (hard)

Description:

[7:4] **Reserved**

[3] **S16**

0: The slot count is 32 but only the first 16 bits are extracted.
1: The slot count is 16.

[2] **SAM**

0: The data is sampled on the rising edge of AUDIO_SCLK_1
1: Data is sampled on the falling edge of the AUDIO_SCLK_1

[1] **FIR**

0: The first channel is input when AUDIO_LRCLK_1 = 0
1: The first channel (left) is input when AUDIO_LRCLK_1 = 1

[0] **PAD**: When 1, AUDIO_LRCLK_1 is delayed by one cycle (padding mode).

48.5 PCM configuration registers**AUD_PCMDIVIDER****Divider for PCM clock**

7	6	5	4	3	2	1	0
AUD_PCMDIVIDER							

Address: *AudioBaseAddress* + 0x0054

Type: Read/write

Reset: Undefined

Description: The PCM divider must be set according to the formula below, where SCLK is the bit clock for the DAC. When Div is set to 0, SCLK is equal to PCMCLK:

$$\text{Div} = \text{PCMCLK} / (2 \times \text{SCLK}) - 1$$

When the internal PLL is used, $\text{PCMCLK} = 384 \times F_s$ or $256 \times F_s$.

If $\text{PCMCLK} = 384 \times F_s$, the formula becomes:

$$\text{Div} = (192 \times F_s / \text{SCLK}) - 1$$

If SCLK is $32 \times F_s$ (common case with the 16-bit DAC), Div must be set to 5.

[7:0] **AUD_PCMDIVIDER**

0000 0001: $\text{PCMCLK} = 256 F_s$, DAC is 32-bit mode

0000 0010: $\text{PCMCLK} = 384 F_s$, DAC is 32-bit mode

0000 0011: $\text{PCMCLK} = 256 F_s$, DAC is 16-bit mode

0000 0101: $\text{PCMCLK} = 384 F_s$, DAC is 16-bit mode

AUD_PCMCONF**PCM configuration**

7	6	5	4	3	2	1	0
IECSELECT	ORD	DIF	INV	FOR	SCL	PREC	

Address: *AudioBaseAddress* + 0x0055

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

- [7] **IECSELECT**: 0: PCMOUT0 is output through S/PDIF.
1: In PCM mode, PCMOUT3 is output through S/PDIF.
- [6] **ORD**: PCM order, only significant in 16-bit mode.
0: MSB sent first. 1: LSB sent first.
- [5] **DIF**: When 0, left padded.
- [4] **INV**
0: Left channel is output when LRCLK is low (I²S format).
1: Left channel is output when LRCLK is high (Sony format).
- [3] **FOR**: 0: I²S format. 1: Sony format.
- [2] **SCL**: 0: PCM outputs and LRCLK are stable on the rising edge of SCLK.
1: Polarity of SCLK is inverted, the PCM outputs and LRCLK are stable for the DACs on the falling edge of SCLK.
- [1:0] **PREC**: PCM precision.
00: 16-bit mode (16 slots) 01: 18-bit mode (32 slots)
10: 20-bit mode (32 slots) 11: 24-bit mode (32 slots)

AUD_PMCROSS**Cross PCM channels**

7	6	5	4	3	2	1	0
VCR		CLR		Reserved			

Address: *AudioBaseAddress* + 0x0056

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register only acts if bit PFC of register [AUD_SPDIF_DTDI](#) is set.

- [7:6] **VCR**: Cross left and right VCR channels.
00: Left channel is mapped on the left output, right channel is mapped on the right output.
01: Left channel is duplicated on both outputs.
10: Right channel is duplicated on both outputs.
11: Right and left channels are toggled.
- [5:4] **CLR**: Cross left and right channels.
- [3:0] **Reserved**

AUD_SFREQ**Sampling frequency**

7 6 5 4 3 2 1 0

AUD_SFREQ							

Address: *AudioBaseAddress* + 0x0005

Type: Read/write (Specific mode)

Reset: No change (soft)
0 (hard)

Description: This status register holds the code of the current output sampling frequency. If the audio stream is encoded (Dolby® Digital, MPEG), or packetized (DVD LPCM), the sampling frequency is automatically read in the audio stream and written to this register by the audio DSP.

For a PCM stream, the application writes to this register. The value in [AUD_SFREQ](#) corresponds to the following frequencies:

Table 172: AUD_SFREQ register values and sampling frequencies

Value	Frequency (kHz)	Value	Frequency (kHz)
0x0	48	0x8	24
0x1	44.1	0x9	22.05
0x2	32	0xA	16
0x3	-	0xB	-
0x4	96	0xC	12
0x5	88.2	0xD	11.025
0x6	64	0xE	8
0x7	-	0xF	-

48.6 ADC input/second input registers

AUD_ADCIN_PLAY**Switch second input processing**

7 6 5 4 3 2 1 0

Reserved	PLAY	Reserved					

Address: *AudioBaseAddress* + 0x00AD

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7] **Reserved**

[6] **PLAY**

0: Switch off can input two processing, even if data is coming in.
1: Switch on can input two processing.

[5:0] **Reserved**

AUD_SFREQ2**PCM sampling frequency for the second input**

7 6 5 4 3 2 1 0

AUD_SFREQ2

Address: *AudioBaseAddress* + 0x0094

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: The value in this register gives the sampling frequency of the incoming PCM stream, according to the table below. **AUD_SFREQ2** is supported from 8 kHz to 48 kHz.

Table 173: AUD_SFREQ2 register values and sampling frequencies

Value	Frequency (kHz)	Value	Frequency (kHz)
0x0	48	0x8	24
0x1	44.1	0x9	22.05
0x2	32	0xA	16
0x3	-	0xB	-
0x4	96	0xC	12
0x5	88.2	0xD	11.025
0x6	64	0xE	8
0x7	-	0xF	-

AUD_ADCIN_MODE**Configure the input mode of PCM data**

7 6 5 4 3 2 1 0

CHAN_SWAP	PCM_INPUT_MODE						

Address: *AudioBaseAddress* + 0x0095

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: PCM_INPUT_MODE in decimal.

[7] **CHAN_SWAP**

0: Do not swap channels

1: Swap left/right channels

[6:0] **PCM_INPUT_MODE**

0000: 16 slots

0001: 16 slots, LSB first

0010: 32 slots, left aligned

0011: 32 slots, right aligned

0100: 32 slots I²S mode

0101: 32 slots, sign extended

0110: 32 slots, 8 bits of data

0111: 32 slots, 16 bits of data

1000: 32 slots, 8 bits of data, mono

1001: User setup mode.^a

a. See registers **AUD_ADCIN_USERSETUP** and **AUD_ADCIN_USERSETUP2**.

AUD_ADCIN_CFG**Configure the second input hardware processing**

7	6	5	4	3	2	1	0
PREC2		CHK_SYNC2	LRCLK2_VAL	INV_LRCLK2	INV_SCLK2	MSBFIRST	I2SNOTSONY

Address: *AudioBaseAddress* + 0x00B1

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7:6] **PREC2**

00: 16 bits

10: 20 bits

01: 18 bits

11: 24 bits

[5] **CHK_SYNC2**

0: Begin capture on first LRCLK2 cycle

1: Capture of incoming data begins on second LRCLK2 cycle. This ensures good synchronization.

[4] **LRCLK2_VAL**: LRCLK2 start value

0: First sample input is taken during LRCLKIN2 = 0 1: First sample input is taken during LRCLKIN2 = 1

[3] **INV_LRCLK2**

0: DATA2 is a right sample when LRCLKIN2 = 1

1: DATA2 is a left sample when LRCLKIN2 = 1

[2] **INV_SCLK2**

0: SCLKIN2 strobes DATA2 on falling edge

1: SCLKIN2 strobes DATA2 on rising edge

[1] **MSBFIRST**

0: LSB arrives first

1: MSB arrives first

[0] **I2SNOTSONY**: Input standard

0: Sony

1: I²S

AUD_ADCIN_USERSETUP**Size of second input data**

7	6	5	4	3	2	1	0
Reserved			DATA_SIZE				

Address: *AudioBaseAddress* + 0x00AB

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register contains the size of the second input data minus one. For example, writing 0x17 in this register means that one LRCLK input edge contains 24 bits of data.
This register is enabled by writing 0x9 to PCM_INPUT_MODE ([AUD_ADCIN_MODE](#)).

Confidential

AUD_ADCIN_USERSETUP2**Second input data position**

7	6	5	4	3	2	1	0
Reserved				DATA_POSITION			

Address: *AudioBaseAddress* + 0x00AC

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register contains the position of the second input data, that is the number of bits to shift to find the LSB of each data.
This register is enabled by writing 0x9 to PCM_INPUT_MODE ([AUD_ADCIN_MODE](#)).

AUD_ADCIN_LEFT_VOL**Second input left attenuation**

7	6	5	4	3	2	1	0
AUD_ADCIN_LEFT_VOL							

Address: *AudioBaseAddress* + 0x0088

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register contains the scaling factor applied to the left channel of the second input. Its value varies from 0 to 1.0. See also [AUD_ADCIN_SHIFT](#) and bit ADCIN_PRE_VOLUME ([AUD_PCMMIX_UPDATE](#)).

AUD_ADCIN_RIGHT_VOL**Second input right attenuation**

7	6	5	4	3	2	1	0
AUD_ADCIN_RIGHT_VOL							

Address: *AudioBaseAddress* + 0x0089

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register contains the scaling factor applied to the right channel of the second input. Its value varies from 0 to 1.0. See also [AUD_ADCIN_SHIFT](#) and bit ADCIN_PRE_VOLUME ([AUD_PCMMIX_UPDATE](#)).

AUD_ADCIN_SHIFT**Second input level shift**

7 6 5 4 3 2 1 0

AUD_ADCIN_SHIFT

Address: *AudioBaseAddress* + 0x00FC

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register contains the shift value applied on the two channels of the second input.
The final attenuation on the second input is:

```
if AUD_ADCIN_SHIFT > 0,
    Left level = (original left level * AUD_ADCIN_LEFT_VOL/256) <<
AUD_ADCIN_SHIFT
    Right level = (original right level * AUD_ADCIN_RIGHT_VOL/256)
<< AUD_ADCIN_SHIFT
```

```
if AUD_ADCIN_SHIFT < 0
    Left level = (original left level * AUD_ADCIN_LEFT_VOL/256) >>
abs(AUD_ADCIN_SHIFT)
    Right level = (original right level * AUD_ADCIN_RIGHT_VOL/256)
>> abs(AUD_ADCIN_SHIFT)
```

See also [AUD_ADCIN_LEFT_VOL](#), [AUD_ADCIN_RIGHT_VOL](#), and bit 6 of [AUD_PCMMIX_UPDATE](#).

Confidential

48.7 PCM mixing registers

AUD_PCMMIX_UPDATE PCM mixing configuration

7	6	5	4	3	2	1	0
MAIN_PRE_VOLUME	ADCIN_PRE_VOLUME	LR_COPY_ON_VCR	MIX	2ND_INPUT_ON_VCR	USE_SRCT	LOAD_SRCT	Reserved

Address: *AudioBaseAddress* + 0x00B2

Type: Read/write

Reset: No change (soft)
Undefined (hard)

[7] **MAIN_PRE_VOLUME**

0: No pre-volume on first input.

1: Pre-volume applied on first input with the coefficient set in [AUD_PCMMIX_FIRSTINPUT_VOLUME](#).

[6] **ADCIN_PRE_VOLUME**

0: No pre-volume on second input.

1: Pre-volume applied on second input with the coefficients set in [AUD_ADCIN_LEFT_VOL](#), [AUD_ADCIN_RIGHT_VOL](#), and [AUD_ADCIN_SHIFT](#).

[5] **LR_COPY_ON_VCR**

0: No copy on VCR

1: Copy L/R channels (PCMOUT0) on VCR channels (PCMOUT3). This may be performed after mixing between second input and PCMOUT0 (see MIX bit).

[4] **MIX**: When 1, mix second input with L/R (PCMOUT0) using [AUD_PCMMIX_MIX_COEFFICIENT](#) and output on PCMOUT0.

[3] **2ND_INPUT_ON_VCR**: When 1, copy of second input on VCR channels (PCMOUT3) after SRC

[2] **USE_SRCT**: When 1, force DSP to use downloaded table for SRC instead of ROMed table

[1] **LOAD_SRCT**: When 1, initiate the download of a new user coefficient table for SRC

[0] **Reserved**

If bits 5 and 3 are simultaneously set, LR_COPY_ON_VCR has priority over 2ND_INPUT_ON_VCR.

Confidential

AUD_PCM MIX_FIRSTINPUT_VOLUME

Volume can be applied on main channels

7	6	5	4	3	2	1	0
NB_CHANS		VOLUME					

Address: *AudioBaseAddress* + 0x00B3

Type: Read/write

Reset: No change (soft)
Undefined (hard)**[7] NB_CHANS**

0: Volume is applied on L/R only

1: Volume is applied on the six main channels

[6:0] VOLUME: Volume attenuation by 1 dB steps from +12 to -64 dB.

0x00: +12 dB

0x0C: 0 dB

0x4C: -64 dB

AUD_PCM MIX_MIX_COEFFICIENT

Mixing level of second input

7	6	5	4	3	2	1	0
MIX_LEV							

Address: *AudioBaseAddress* + 0x00BA

Type: Read/write

Reset: No change (soft)
Undefined (hard)Description: This register stores the level of second input relative to L/R outputs (PCMOUT0) when activating bit MIX of [AUD_PCM MIX_UPDATE](#).

The mixer outputs:

$$\text{PCMOUT0} = (1 - \text{mix\_level}) \times \text{PCMOUT1} + \text{mix\_level} \times \text{second input}$$

where $\text{mix\_level} = \text{MIX\_LEV}/255$.**AUD_PCM MIX_SRC_MSB**

Input of the MSB of a new SRC coefficient

7	6	5	4	3	2	1	0
SRC_COEFF_MSB							

Address: *AudioBaseAddress* + 0x00B8

Type: Read/write

Reset: No change (soft)
Undefined (hard)Description: See [AUD_PCM MIX_ACK](#) and [AUD_PCM MIX_SRC_HANDSHAKE](#).**AUD_PCM MIX_SRC_LSB**

Input LSB of a new SRC coefficient

7	6	5	4	3	2	1	0
SRC_COEFF_LSB							

Address: *AudioBaseAddress* + 0x00B9

Type: Read/write

Reset: No change (soft)
Undefined (hard)Description: See [AUD_PCM MIX_ACK](#) and [AUD_PCM MIX_SRC_HANDSHAKE](#).

AUD_PCM MIX_SRC_HANDSHAKE Allow input of a new SRC coefficient

7	6	5	4	3	2	1	0
HSHAKE							

Address: *AudioBaseAddress* + 0x00B7

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: See also [AUD_PCM MIX_ACK](#), [AUD_PCM MIX_SRC_MSB](#),
[AUD_PCM MIX_SRC_LSB](#).

[7:0] **HSHAKE**: Write 0x01 when coefficients in [AUD_PCM MIX_SRC_MSB](#) and [AUD_PCM MIX_SRC_LSB](#) are ready for DSP. The DSP writes 0x00 when these coefficients have been read.

AUD_PCM MIX_ACK Activate DSP acquisition of a new SRC coefficient

7	6	5	4	3	2	1	0
Reserved							AUD_ACK

Address: *AudioBaseAddress* + 0x0016

Type: Read/write

Reset: 0 (soft)
0 (hard)

Description: When [AUD_PCM MIX_SRC_MSB](#) and [AUD_PCM MIX_SRC_LSB](#) are loaded with a new SRC coefficient and [AUD_PCM MIX_SRC_HANDSHAKE](#) is set to 1, write 1 to initiate the capture of this coefficient by the DSP. This bit generates an interrupt to the DSP core. The DSP then resets [AUD_PCM MIX_ACK](#) and [AUD_PCM MIX_SRC_HANDSHAKE](#).

Confidential

48.8 VCR configuration registers**AUD_VCR_OUTPUT Possible configurations for the VCR output**

7	6	5	4	3	2	1	0
Reserved			STEREO	PRL	Reserved	COPY	Reserved

Address: *AudioBaseAddress* + 0x00AE

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7:5] **Reserved**

[4] **STEREO**: When 1, stereo 2 front/0 rear downmix output on the VCR.

[3] **PRL**: When 1, Pro Logic® downmix output on the VCR.

[2] **Reserved**

[1] **COPY**: When 1, enables a hardware copy of left/right channels to the VCR channels.

[0] **Reserved**

48.9 S/PDIF output setup registers

AUD_SPDIF_CMD

S/PDIF control

7	6	5	4	3	2	1	0
AUX	Reserved					CMD	

Address: *AudioBaseAddress* + 0x005E

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register is the S/PDIF control register. Several modes are available, the mode is selected by value.

[7] **AUX**

0: PCMOUT0 (L/R) is transmitted to the S/PDIF out when S/PDIF is in PCM mode (see CMD bit).

1: PCMOUT3 (VCR) is transmitted to the S/PDIF out when S/PDIF is in PCM mode (see CMD bit).

[6:2] **Reserved**

[1:0] **CMD**

00: Off mode. The S/PDIF is not working; the output line is idle.

01: Muted mode. The outputs are PCM null data.

10: PCM mode (outputs are PCM data). Only the first two decoded channels (left and right) are transmitted.

11: Encoded. The compressed bit stream is transmitted (see IEC61937 standard).

AUD_SPDIF_CAT

Category code

7	6	5	4	3	2	1	0
CAT_CODE							

Address: *AudioBaseAddress* + 0x005F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Refer to IEC60958 and its latest amendments for Category code programming. IEC60958 bits 15 to 8 correspond to [AUD_SPDIF_CAT](#)[7:0]. In particular, [AUD_SPDIF_CAT](#)[7] is the L-bit. IEC60958 bit 2 (Cp-bit or copyright) is [AUD_SPDIF_STATUS](#) bit 1 (COP).

AUD_SPDIF_CONF**S/PDIF PCMCLK divider**

7	6	5	4	3	2	1	0
LAT	SM	RND	DIV				

Address: *AudioBaseAddress* + 0x0060

Type: Read/write

Reset: No change (soft)
Undefined (hard)

- [7] **LAT**: Configures the latency mode between the S/PDIF output (in mode compressed) and audio output.
0: Autolateness (latency is the transmission time for 2/3 of the payload, plus the time to decode an audio block). For MPEG autolateness, the latency is given here depending on the sampling frequency in the incoming bit stream: MPEG 48 kHz: 20.90 ms, MPEG 44.1 kHz: 22.95 ms, MPEG 32 kHz: 32.53ms.
1: User programmable latency register [AUD_SPDIF_LATENCY](#) is used.
- [6] **SM**: Sync mute mode, must be set to zero.
- [5] **RND**: Used to obtain a 16-bit rounding on the S/PDIF (when in PCM mode).
0: No rounding
1: Rounding
This bit has no effect on the precision of PCMOOUT[0:3] data.
- [4:0] **DIV**: This field is the PCMCLK divider. It must be set according to the formulae:
16-bit mode: $IECDIV = (1 + PCMDIV)/2 - 1$
32-bit mode: $IECDIV = PCMDIV$

The table below shows the relationship between the value of the IEC divider and the value of the PCM divider.

PCM divider value	Mode description	IEC divider value
5	PCMCLK = 384 F _s , DAC is 16-bit mode	2
3	PCMCLK = 256 F _s , DAC is 16-bit mode	1
2	PCMCLK = 384 F _s , DAC is 32-bit mode	2
1	PCMCLK = 256 F _s , DAC is 32-bit mode	1

AUD_SPDIF_STATUS S/PDIF status bit

7	6	5	4	3	2	1	0
Reserved	SFR				PRE	COP	COM

Address: *AudioBaseAddress* + 0x0061

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register is used to set the value of the status bit in the IEC958 data stream.

[7] **Reserved**

[6:3] **SFR**: Sampling frequency:

0000: 44.1 kHz

0010: 48 kHz

0011: 32 kHz

1010: 96 kHz

[2] **PRE**

0: Output does not have pre-emphasis.

1: Output has pre-emphasis.

[1] **COP**

0: Copy not allowed.

1: Copy allowed.

[0] **COM**: Compress data bit.

0: Noncompressed mode.

1: Compressed mode.

AUD_SPDIF_REP_TIME S/PDIF repetition time of a pause frame

7	6	5	4	3	2	1	0
AUD_SPDIF_REF_TIME							

Address: *AudioBaseAddress* + 0x0075

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: In compressed mode, a burst of pause frames is sent when there is no more data to transmit, due to an error or a gap in the incoming bit stream, for example. This register sets the size of a pause frame in IEC frames: Dolby® Digital = 4, MPEG = 32.

AUD_SPDIF_LATENCY Latency value

7	6	5	4	3	2	1	0
VALUE							

Address: *AudioBaseAddress* + 0x007E

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: If bit LAT of register [AUD_SPDIF_CONF](#) is set, a delay can be configured between the output of IEC61937 in compressed mode and the audio decoder output. To configure a latency period, set the register according the following formula:

$$\text{VALUE} = L \times F_s / 8$$
 where L = latency (seconds) and F_s = sampling frequency (Hz).
 The minimum latency delay is 0; the maximum is the time taken to decode a frame.

[7:0] **VALUE**: For Dolby® Digital, L = 1536 samples/sampling frequency.
 For MPEG, L = 1152 samples/sampling frequency

AUD_SPDIF_DTDI**S/PDIF data type information**

7	6	5	4	3	2	1	0
PFC	DTD	Reserved	DTDI				

Address: *AudioBaseAddress* + 0x007F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7] **PFC**: When 1, PCMCROSS function enabled

[6] **DTD**

0: Transmitted DTDI are extracted from the stream.

1: Data type dependent information used for the S/PDIF in compressed mode, can be set.
Refer to IEC958 standard for more information.

[5] **Reserved**

[4:0] **DTDI**: In Dolby® Digital mode: DTDI[4:3] = {0,0}; DTDI[2:0] = BSMOD[2:0]
In MPEG mode: DTDI[4:2] = {0,0,0}; DTDI[1] = DR; DTDI[0] = K

48.10 Audio command registers**AUD_SOFTRESET****Soft reset**

7	6	5	4	3	2	1	0
AUD_SOFTRESET							

Address: *AudioBaseAddress* + 0x0010

Type: Write only

Reset: Not applicable

Description: When bit 0 of this register is set, a soft reset occurs. The command registers and the interrupt registers listed below are cleared. The decoder goes into idle mode and the volumes are cleared.

- Command registers: [AUD_MUTE](#), [AUD_RUN](#), [AUD_ADCIN_PLAY](#), [AUD_SKIP_MUTE_CMD](#), [AUD_SKIP_MUTE_VALUE](#).
- Interrupt registers: [AUD_INTE](#), [AUD_INT](#), [AUD_ERROR](#).

Confidential

AUD_PLAY**Play**

7	6	5	4	3	2	1	0
Reserved							PLAY

Address: *AudioBaseAddress* + 0x0013

Type: Read/write

Reset: 0

Description: The **play** command is treated according to the state of the decoder.

- When in idle or initialize mode, the decoder ignores the **play** command.
- When in decode mode, **play** enables the decoding, see the table below:

PLAY	MUTE	SCLK, LRCLK state	PCMOUT	Decoding
0	0	Not running	0	No
0	1	Running	0	No
1	0	Running	Decoded samples	Yes
1	1	Running	0	Yes

AUD_MUTE**Mute**

7	6	5	4	3	2	1	0
Reserved							MUTE

Address: *AudioBaseAddress* + 0x0014

Type: Read/write

Reset: 0

Description: The **mute** command is handled differently according to the state of the decoder.

- When in idle mode after hardware reset, setting MUTE to 1 automatically runs the SCLK and LRCLK clocks and outputs them to the DACs.
- When playing, setting MUTE to 1 mutes the PCM outputs.
- Register **AUD_MUTE** has no effect on the S/PDIF output.

AUD_RUN**Run decoding**

7	6	5	4	3	2	1	0
Reserved							RUN

Address: *AudioBaseAddress* + 0x0072

Type: Read/write

Reset: 0

Description: This register enables the decoder to exit from idle mode. After a soft or hard reset, the decoder is in idle mode. It stays in this mode until run is set. In run mode, the decoder takes into account the state of all the configuration registers and begins to decode.

Only the **soft reset** or **reboot** commands can reset register **AUD_RUN**

AUD_SKIP_MUTE_CMD Skip or mute commands

7	6	5	4	3	2	1	0
Reserved				PAU	BLK	SKP	MU

Address: *AudioBaseAddress* + 0x0073

Type: Read/write

Reset: 0

Description: This register cannot be used in MP3 decoding mode. It is taken in to account at the beginning of decoding a frame.

[7:4] **Reserved**

[3] **PAU**: Pause command

1: Decoder pauses while decoding one block, then tests if the host has reset the bit. If the bit is still set, it pauses during decoding a block again and so on. If reset, it plays the incoming stream.

In compressed mode, the IEC61937 transmits a pause burst for the duration of the time taken to decode a frame with `GAP_LENGTH_PARAMETER_1`. If the bit is still set, it transmits a pause frame for the duration of the time taken to decode a block frame with `GAP_LENGTH_PARAMETER_2` and so on. If reset, it transmits the next frame.

In noncompressed mode, the IEC60958 transmits the same data as the decoder.

[2] **BLK**: Pause blocks of frames

1: Decoder pauses at the number of blocks set in the `AUD_SKIP_MUTE_VALUE` register. This command is useful for audio video synchronization when the video decoder is in late. Once the pause of blocks is finished, the decoder clears this bit.

In compressed mode, the IEC61937 transmits pause bursts. Repetition times and the gap length parameter are given in the following table.

In noncompressed mode, the IEC60958 is paused as the decoder (zeroes are transmitted)

[1] **SKP**: Skip frame

1: Decoder skips the number of frames set in the `AUD_SKIP_MUTE_VALUE` register. This command is useful for audio video synchronization when the audio decoder is in late. Once the frame skipped, the decoder clears this bit. This command is available for all decoders.

In compressed mode, the IEC61937 does not transmit the skipped frames.

In noncompressed mode, the IEC60958 does not transmit the skipped frame as the decoder.

[0] **MU**: Global mute command

1: PCM and S/PDIF outputs are muted.

In compressed mode, the S/PDIF transmits bursts of pause frames

In noncompressed mode, the IEC60958 transmits the same data as the decoder.

0: Normal operation.

AUD_SKIP_MUTE_VALUE Skip frames or mutes blocks of frame

7 6 5 4 3 2 1 0

AUD_SKIP_MUTE_VALUE							

Address: *AudioBaseAddress* + 0x0074

Type: Read/write

Reset: 0 (soft)
0 (hard)

Description: This register works according to soft mute block of frames (bit BLK) or skip frame (bit SKP) in register [AUD_SKIP_MUTE_CMD](#).
If the command **skip** is selected, the decoder skips n frames, and then clears the register.
If the command **aud_soft_mute_block** is selected, the decoder mutes n blocks, and then clears the register, where n is the value in this register. The following table gives the number of samples in a block or a frame.

	Samples per frames	Samples per block
Dolby® Digital	1536	256
MPEG layer II	1152	96
MPEG layer I	384	96
MP3	1152	288

48.11 Audio interrupt registers**AUD_INTE Interrupt enable**

7 6 5 4 3 2 1 0

INTEH							
INTEL							

Address: *AudioBaseAddress* + 0x0007 (INTEL)/0x0008(INTEH)

Type: Read/write

Reset: 0

Description: The audio decoder contains a 16-bit interrupt register ([AUD_INT](#)) associated with this 16-bit enable register. A bit set in [AUD_INTE](#) enables the corresponding interrupt. The interrupt associated with each bit is given in the [AUD_INT](#) description.

AUD_INT**Interrupts (L and H)**

	7	6	5	4	3	2	1	0
0x000A	Reserved					FBE	FBF	PCM
0x0009	ANC	PTS	BOF	DEM	SFR	ERR	HDR	SYN

Address: *AudioBaseAddress* + 0x0009 (INTL)/0x000A (INTH)

Type: Read only

Reset: 0

Description: Each interrupt can only occur if the corresponding bit is set in [AUD_INTE](#). The table below shows the condition indicated by each bit.

INTH

0x000A: [7:3] **Reserved**

0x000A: [2] **FBE**: The frame buffer memory contains one frame which is beginning to be decoded. The next frame begins to be parsed.

0x000A: [1] **FBF**: Frame buffer full. The frame buffer memory contains two frames: one decoded, and one parsed for next decoding.

0x000A: [0] **PCM**: PCM output underflow. Cleared after reset or when the MSB of the interrupt register is read.

INTL

0x0009: [7] **ANC**: Ancillary data registered. Cleared after reset or when the MSB of [AUD_ANCCOUNT](#) is read.

0x0009: [6] **PTS**: First bit of new frame with PTS at output stage. Cleared after reset or when the MSB of [AUD_PTS](#) is read.

0x0009: [5] **BOF**: First bit of new frame at output stage. Cleared after reset or when the MSB of the interrupt register is read.

0x0009: [4] **DEM**: De-emphasis changed. Cleared after reset or when the MSB of the interrupt register is read.

0x0009: [3] **SFR**: Sampling frequency changed. Cleared after reset or when the MSB of the interrupt register is read.

0x0009: [2] **ERR**: Error detected. Cleared after reset or when the MSB of [AUD_ERROR](#) is read.

0x0009: [1] **HDR**: Valid header registered. Cleared after reset or when the MSB of [AUD_HEAD4](#) is read.

0x0009: [0] **SYN**: Change in synchronization status. Cleared at reset or when the MSB of [AUD_SYNC_STATUS](#) is read.

AUD_SYNC_STATUS**Synchronization status**

7	6	5	4	3	2	1	0
Reserved				PAC		FRA	

Address: *AudioBaseAddress* + 0x0040

Type: Read only

Reset: 0

Description: This register indicates the status of the audio parser for synchronization. It is used in conjunction with registers [AUD_PACKET_LOCK](#) and [AUD_SYNC_LOCK](#). On read, the synchronization status interrupts.

[7:4] **Reserved**

[3:2] **PAC**: Packet status

00: Search packet synchronization word

01: Wait for confirmation. Sync word detected but parser not yet detected [PACKET_LOCK](#) + 1 sync words

10: Synchronized. [PACKET_LOCK](#) + 1 sync words detected

11: Reserved

[1:0] **FRA**: Frame status

00: Search audio synchronization

01: Wait for confirmation. Sync word detected but parser not yet detected [SYNC_LOCK](#) + 1 sync words

10: Synchronized. [SYNC_LOCK](#) + 1 sync words detected

11: Reserved

AUD_ANCCOUNT**Ancillary data**

7	6	5	4	3	2	1	0
AUD_ANCCOUNT							

Address: *AudioBaseAddress* + 0x0041

Type: Read only

Reset: 0

Description: This value gives the number of ancillary data in the stream. The ancillary data interrupt bit ANC of the [AUD_INT](#) register is cleared by a read.

AUD_HEAD4**Header 4**

	7	6	5	4	3	2	1	0
AC_3	Reserved					BSMOD		
MPEG-2	Reserved					0	DR	K
Other	Reserved					0	0	0

Address: *AudioBaseAddress* + 0x0042

Type: Read only

Reset: Undefined (soft)
Undefined (hard)

Description: This register contains header data HEAD[31:24]. The contents depend on the type of the frame.

When the host reads this register, the corresponding interrupt bit (HDR) is cleared.

[7:3] Reserved, must be set to 0.

[2:0] **Dolby® Digital: BSMOD** if a Dolby® Digital frame

MPEG-2:

[2]: Must be set to 0

[1]: DR: When set to 1, dynamic range exists

[0]: K: 0: Normal mode

1: Karaoke mode.

Other:

In all other types of frame HEAD4[2:0] = 000.

AUD_HEAD3**Header 3**

	7	6	5	4	3	2	1	0
	Reserved			DTYPE				

Address: *AudioBaseAddress* + 0x0043

Type: Read only

Reset: Undefined (soft)
Undefined (hard)

Description: This register contains header data HEAD[23:16]. HEAD3[7:5] = 000, in all cases
HEAD3[4:0] = DTYPE

DTYPE is the data type and is defined as follows:

[7:5] **Reserved**, must be set to 0.

[4:0] **DTYPE**

0000: Null data or linear PCM

0001: Dolby® Digital

0100: MPEG-1 Layer I

0101: MPEG-1 Layer II or MPEG-2 word extension

0110: MPEG-2 Layer II with extension

1001: MPEG-2 Layer II low sample rate

This register cannot detect the data type of data in a stream.

AUD_HEADLEN**Frame length**

7	6	5	4	3	2	1	0
HEADLEN[15:8]							
HEADLEN[7:0]							

Address: *AudioBaseAddress* + 0x0044 (AUD_HEADLEN[15:8])/0x0045 (AUD_HEADLEN[7:0])

Type: Read only

Reset: 0

Description: This register contains the bit length of the compressed data frame HEADLEN[15:0]. Header registers are all updated as soon as the decoder begins to decode a frame.

AUD_PTS**PTS**

	7	6	5	4	3	2	1	0
0x0046	Reserved							PTS[32]
0x0047	PTS[31:24]							
0x0048	PTS[23:16]							
0x0049	PTS[15:8]							
0x004A	PTS[7:0]							

Address: *AudioBaseAddress* + register offset

Type: Read/write

Reset: 0

Description: When the PTS interrupt is activated, a new PTS value is stored in this register. Once the PTS[32] value is read bit PTS of register **AUD_INT** is cleared.

AUD_ERROR**Error code**

7	6	5	4	3	2	1	0
AUD_ERROR							

Address: *AudioBaseAddress* + 0x000F

Type: Read only

Reset: 0

Description: This is a status register; when the ST20 reads this register, it is cleared along with the corresponding interrupt register. This 7-bit register is ANDed with 0x7F to get the correct value. The value in this register indicates the type of error that has occurred. These errors are defined in the table below.

The warnings and errors for the AC-3 and MPEG-1, -2, -3 decoder are overlapped from value 1 to value 21. There are no warnings in AC-3 mode, only critical errors.

An error interrupt occurs only for a critical error and not for a warning. If an error interrupt occurs, the error returned by register AUD_ERROR is the first invoked whilst frame decoding. Further errors or warnings may result from the first.

Confidential

Value	MPEG-1, 2	MP3	AC-3
-	Warnings		Errors
1	MPEG2_MC_CRC_ERROR	MP3_CRC_ERROR	EXPAND_DELTA_PAST_END_ARRAY
2	MPEG2_EXT_CRC_ERROR	MP3_CUTOFF_ERROR	XDCALL_TRY_TO_REUSE_REMAT_FLG
3	MPEG1_CRC_ERROR	MP3_BIG_VALUE_ERROR	XDCALL_TRY_TO_REUSE_COUPLING_STRA
4	MPEG_MC_LAYER1_NOT_SUPPORTED	MP3_HUFFTABLE_ERROR	XDCALL_CANT_COUPLE_IN_DUAL_MODE
5	MPEG_MC_EXT_BAD_SYNCWORD_ERROR	-	XDCALL_TRY_TO_REUSE_CPL_LEAK
6	-	-	XDCALL_TRY_TO_REUSE_SNR
7	-	-	XDCALL_TRY_TO_REUSE_BIT_ALLOC
8	-	-	XDCALL_TRY_TO_REUSE_COUPLING_EXPONENT_STRA
9	MPEG2_ILLEGAL_HEADER	MP3_MOD_BUF_SIZE_ERROR	XDCALL_TRY_TO_REUSE_EXPONENT_STRA
-	Errors		-
10	MPEG_BITRATE_ERROR	MP3_HUFFMAN_DECODE_ERROR	XDCALL_TRY_TO_REUSE_LFE_EXPONENT_STRA
11	MPEG_LAYER1_NOT_SUPPORTED	MP3_DYNPART_EXCHANGE_ERROR	XDCALL_CHBWCOD_IS_TOO_HIGH
12	MPEG_LAYER2_NOT_SUPPORTED	MP3_GR_LENGTH_ERROR	BSI_ERR_REV
13	MPEG_ILLEGAL_LAYER	MP3_CH_LENGTH_ERROR	BSI_ERR_CHANS
14	MPEG2_ILLEGAL_CHN_CONFIG	MP3_INPUT_BIT_AVAILABLE_ERROR	CRC_NOT_VALID
15	MPEG_MC_NOT_SUPPORTED	MP3_HEAD_FRAMELENGTH_ERROR	-
16	-	MP3_DYNPART_LENGTH_ERROR	-
17	-	MP3_BLOCK_TYPE_ERROR	-
18	-	MP3_HEAD_EMPHASIS_ERROR	-
19	-	MP3_HEAD_SAMP_FREQ_ERROR	-
20	-	MP3_HEAD_LAYER_ERROR	-
21	-	MP3_ILLEGAL_MODE	-

Confidential

Table 174: Audio errors

Number	Name
	Dolby® Digital decoding
0	NO-ERROR
	Packet synchronization
22	BAD_SUB_STREAM_ID
23	PACK_HEADER_FIELD_PRESENCE_NOT_SUPPORTED
24	BAD_INFO_IN_DVD_PACKET_HEADER
25	BAD_INFO_IN_PES_PACKET_HEADER
26	BAD_INFO_IN_DVDA_PACKET_HEADER
27	BAD_INFO_IN_PACKMP1_PACKET_HEADER
28	BAD_INFO_IN_SPDIF_PC
29	BIT_ERROR_IN_BURST_PAYLOAD
30	BAD_NULL_DATA_BURST
31	BAD_STREAM_ID
32	SYNCHRO_PACKET_NOT_FOUND
	Audio synchronization
33	BAD_CRC_AC3
34	BAD_LPCM_QUANTIZATION_WORDLENGTH
35	BAD_AUDIO_SAMPLING_FREQUENCY
36	BAD_MPEG_LAYER
37	MPEG_BITRATE_FREE_FORMAT
38	NOT_SUPPORTED_AC-3_FRMSIZECOD
39	BAD_CRC_MPEG_FRONT_END
40	BAD_MPEG_EXTENDED_RESERVED_BIT
41	MPEG_EXTENDED_SYNC_NOT_FOUND
42	MPEG_EXTENDED_LENGTH_TOO_SMALL
64	SYNCHRO_AUDIO_NOT_FOUND
	Other errors
67	LATENCY_TOO_BIG
68	REPEAT_BLOCK_ERROR
69	UNKNOW_SFREQ_FOR_LATENCY
70	LATENCY_TOO_SMALL
75	BAD_SFREQ
86	SRC_ERROR
87	ADCIN_MODE_INCORRECT
88	ADCIN_OVERFLOW_ERROR
128	BAD_HOST_CONFIG

Confidential

48.12 Audio DSP decoding algorithm registers

The table below shows how registers `AUD_STREAMSEL` and `AUD_DECODESEL` should be programmed for different types of bit stream.

Table 175: `AUD_STREAMSEL` and `AUD_DECODESEL` programming definitions

<code>AUD_STREAMSEL</code>	<code>AUD_DECODESEL</code>	Mode
0	0	MPEG-2 PES carrying Dolby® Digital (ATSC)
0	1	MPEG-2 PES carrying MPEG-1 frames
0	2	MPEG-2 PES carrying MPEG-2 frames
2	1	MPEG-1 packet carrying MPEG-1 audio
3	0	Dolby® Digital frames elementary streams
3	1	MPEG-1 frame elementary streams
3	2	MPEG-2 frame elementary stream
3	3	Stereo PCM (16-bit samples)
3	4	Pink noise generator
3	7	Activate PCM beep tone
3	9	MP3 elementary streams
5	0	IEC61937 input with Dolby® Digital frames
5	1	IEC61937 input with MPEG-1 frames
5	2	IEC61937 input with MPEG-2 frames

`AUD_DECODESEL`

Decoding algorithm

7 6 5 4 3 2 1 0

Reserved	DEC

Address: *AudioBaseAddress* + 0x004D

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register identifies the audio data type.

[7:4] **Reserved**

[3:0] **DEC**

0000: Dolby® Digital decoding

0010: MPEG-2

0100: Pink noise generator

0111: PCM beep tone generator

0001: MPEG-1

0011: PCM

0110: Reserved

1001: MP3 (product variant option)

AUD_STREAMSEL**Stream selection**

7	6	5	4	3	2	1	0
Reserved					STRSEL		

Address: *AudioBaseAddress* + 0x004C

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7:3] **Reserved**

[2:0] **STRSEL**

000: PES

010: Packet MPEG-1

100: Reserved

001: Reserved

011: Elementary stream/IEC60958

101: S/PDIF IEC61937

48.13 Audio DSP system synchronization registers**AUD_PACKET_LOCK****Packet lock**

7	6	5	4	3	2	1	0
Reserved							PACKET_LOCK

Address: *AudioBaseAddress* + 0x004F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register specifies the number of supplementary packet sync words that the packet parser must detect before it is considered as synchronized, and can send data to the audio parser (maximum = 1, minimum = 0). In this way, stream data cannot be sent to the audio parser instead of packet sync words.

[7:1] **Reserved**

[0] **PACKET_LOCK**

0: Packet parser is synchronized when it has detected one packet sync word

1: Packet parser is synchronized when it has detected two packet sync words

AUD_ID_EN**Enable audio ID**

7	6	5	4	3	2	1	0
Reserved							AUD_ID_EN

Address: *AudioBaseAddress* + 0x0050

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: If set to 1, the audio decoder decodes only the stream corresponding to the stream ID or substream ID of the packet layer. This is selected through registers [AUD_ID](#) or [AUD_ID_EXT](#). If set to 0, the decoder decodes all audio packets.

Confidential

AUD_ID**Audio ID**

7	6	5	4	3	2	1	0
Reserved				AUD_ID			

Address: *AudioBaseAddress* + 0x0051

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: When decoding packets, an identifier may be specified for a selected program. This register must be written with the packet ID. This feature is enabled when the register [AUD_ID_EN](#) is set, and only packets with matching ID are decoded. For MPEG-1 packets or PES MPEG-2 packets carrying MPEG streams, the five LSB are significant. For PES MPEG-2 packets carrying AC-3 streams, the audio ID choice is not available in the standard. For DVD packets, the three LSB of this register are significant. These bits correspond to the stream number defined in the STREAM_ID field of the audio packet header.

AUD_ID_EXT**Audio extension**

7	6	5	4	3	2	1	0
Reserved				AUD_ID_EXT			

Address: *AudioBaseAddress* + 0x0052

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: The three LSB of this register are significant. In the case of DVD MPEG-2 audio with extension bit stream, this register is used to select the stream defined in the STREAM_ID of the packets containing MPEG-2 extension bit stream data.

AUD_SYNC_LOCK**Sync lock**

7	6	5	4	3	2	1	0
Reserved						SYNC_LOCK	

Address: *AudioBaseAddress* + 0x0053

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register specifies the number of supplementary audio sync words that the audio parser must detect before it is considered as synchronized, and can send data to the decoder. In this way, stream data cannot be sent to the decoder instead of audio sync words. Maximum value = 3; minimum value = 0.

[7:2] **Reserved**

[1:0] **SYNC_LOCK**

00: Audio parser synchronized when one audio sync word detected

01, 10, 11: When the audio parser has detected SYNC_LOCK + 1 audio sync words, it sends the data to the decoder

48.14 Postdecoding and Pro Logic® registers

AUD_PDEC**Postdecoder control**

7	6	5	4	3	2	1	0
Reserved		DEM	DCF	DB	Reserved		PL

Address: *AudioBaseAddress* + 0x0062

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register controls the postdecoder operations.

[7:6] **Reserved**

[5] **DEM**: When 1, the de-emphasis filter is activated.

[4] **DCF**: When 1, the DC filter is activated.

[3] **DB**: When 1, the double stereo procedure is called. It consists in a copy of PCMOUT1 on to PCMOUT3 channels in order to have a pseudo 5-channels decoder effect.

[2:1] **Reserved**

[0] **PL**

0: The PL decoder is activated only if the output of the previous decoding stage is Pro Logic® encoded.

1: Pro Logic® decoding is activated.

AUD_PL_AB**Pro Logic® autobalance**

7	6	5	4	3	2	1	0
Reserved						PL_WS	PL_AB

Address: *AudioBaseAddress* + 0x0064

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register enables the auto balance function of the Pro Logic® decoder. The default value is zero (auto balance off).

[7:2] **Reserved**

[1] **PL_WS**: Select wide surround mode.

0: Disabled

1: Enabled

[0] **PL_AB**: Select the autobalance function (used to track out gain between Lt and Rt).

0: Disabled

1: Enabled

AUD_PL_DWNX**Pro Logic® decoder downmix**

7	6	5	4	3	2	1	0
Reserved					AUD_PL_DWNX		

Address: *AudioBaseAddress* + 0x0065

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: The value in this register controls the function of the Pro Logic® downmix.

[7:3] **Reserved**

[2:0] **AUD_PL_DWNX**

000, 001, 010: Pro Logic® is disabled.

011: 3 front/0 rear (L, C, R) three stereo.

100: 2 front/1 rear (L, R, S) phantom^a.

101: 3 front/1 rear (L, C, R, S).

110: 2 front/2 rear (L, R, S, S) phantom.

111: 3 front/2 rear (L, C, R, S, S).

a. Phantom mode means that the center is not used.

AUD_DWSMODE**Downsampling filter**

7	6	5	4	3	2	1	0
Reserved					AUD_DWSMODE		

Address: *AudioBaseAddress* + 0x0070

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register controls the downsampling filter. When decoding a 96 kHz PCM stream, it might be necessary to downsample the stream to 48 kHz.

[7:2] **Reserved**

[1:0] **AUD_DWSMODE:**

00, 01: Automatic (according to input bit stream frequency).

10: Suppress downsampling.

Confidential

48.15 Bass redirection registers

AUD_VOLUME0

Volume of first channel

7 6 5 4 3 2 1 0

AUD_VOLUME0

Address: *AudioBaseAddress* + 0x004E

Type: Read/write (Specific mode)

Reset: 0 (soft)
Undefined (hard)

Description: Attenuation applied to the channel selected by CHAN_IDX (register [AUD_CHAN_IDX](#)). The attenuation is -n dB where n is the contents of the register.

[7:0] **AUD_VOLUME0** is written with the attenuation applied to channel selected with the CHAN_IDX value 000: Left

Reading AUD_VOLUME0 provides the attenuation applied to channel selected with the CHAN_IDX value 101: Left

Other values of CHAN_IDX are meaningless.

AUD_VOLUME1

Volume of second channel

7 6 5 4 3 2 1 0

AUD_VOLUME1

Address: *AudioBaseAddress* + 0x0063

Type: Read/write (Specific mode)

Reset: 0 (soft)
Undefined (hard)

Description: Attenuation applied to the channel selected by CHAN_IDX (register [AUD_CHAN_IDX](#)). The attenuation is -n dB where n is the contents of the register

[7:0] **AUD_VOLUME1** is written with the attenuation applied to channel selected with the CHAN_IDX value 000: Right

Reading AUD_VOLUME1 provides the attenuation applied to channel selected with the CHAN_IDX value 101: Right

Other values of CHAN_IDX are meaningless.

Confidential

AUD_OCFG**Output configuration**

7	6	5	4	3	2	1	0
Reserved					OCFG_NUM		

Address: *AudioBaseAddress* + 0x0066

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7:3] **Reserved**

[2:0] **OCFG_NUM**

Always set to 000

AUD_CHAN_IDX**Channel Index**

7	6	5	4	3	2	1	0
Reserved					CHAN_IDX		

Address: *AudioBaseAddress* + 0x0067

Type: Read/write

Reset: 4 (soft)
Undefined (hard)

Description: This register identifies the pair of channels and the type of access.
Once read/write volume is finished, this register takes the value 4 again (waiting for read/write access). The volume is automatically updated at each block during decoding.

[7:3] **Reserved**

[2:0] **CHAN_IDX**

000: Left and right write access

001, 010, 011: Reserved

100: No pair selected. Indicates that volume can be read or written.

101: Left and right read access

110, 111: Reserved

To read a volume, [AUD_CHAN_IDX](#) must be set to the appropriate value. The DSP indicates that the attenuation is readable through registers [AUD_VOLUME0](#) and [AUD_VOLUME1](#) by automatically changing [AUD_CHAN_IDX](#) to value 4.

To write a volume, the attenuation of the pair of channels should be written in [AUD_VOLUME0](#) and [AUD_VOLUME1](#). Then [AUD_CHAN_IDX](#) is written to the appropriate value. The attenuation is updated on the next audio block and [AUD_CHAN_IDX](#) is automatically changed to 4.

48.16 Dolby® Digital configuration registers

AUD_AC3_DECODE_LFE Decode LFE

7	6	5	4	3	2	1	0
Reserved							DEC

Address: *AudioBaseAddress* + 0x0068

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: If DEC bit is set to 1, the device decodes LFE channel (if present).

AUD_AC3_COMP_MOD Compression mode

7	6	5	4	3	2	1	0
AUD_AC3_COMP_MOD							

Address: *AudioBaseAddress* + 0x0069

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: The value of this register defines the compression mode. In custom A mode, an external analog part performs the dialog normalization function, not the audio decoder. In all other modes, the audio decoder performs normalization.

[7:0] **AUD_AC3_COMP_MOD**

00: Custom A (analog)
10: Line out

01: Custom D (digital)
11: RF mode

AUD_AC3_HDR High dynamic range

7	6	5	4	3	2	1	0
AUD_AC3_HDR							

Address: *AudioBaseAddress* + 0x006A

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register corresponds to the dynamic range scale factor for high level signals, also called cut factor in the Dolby® specifications.
HDR = 255 x cut factor (in decimal), where the cut factor is a fractional number between 0 and 1. It is used to scale the dynamic range control word for high level signals that would otherwise tend to be reduced.
When HDR = 0xFF (cut factor = 1.0), high level signal reduction is the one given in the stream. A value of zero disables the high level compression. This word is ignored if the compression mode is set to RF mode.

AUD_AC3_LDR**Low dynamic range**

7 6 5 4 3 2 1 0

AUD_AC3_LDR

Address: *AudioBaseAddress* + 0x006B

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register corresponds to the dynamic range scale factor for low level signals, also called boost factor in the Dolby® specifications.
LDR = 255 x boost factor (in decimals), where the boost factor is a fractional number between 0 and 1.0.
The boost factor scales the dynamic range control word for low level signals that would otherwise tend to be amplified. When LDR = 0xFF (boost factor = 1.0), and the low level signals amplification is maximum. A value of zero disables the low level amplification. This word is ignored if compression is set to RF mode.

AUD_AC3_RPC**Repeat count**

7 6 5 4 3 2 1 0

AUD_AC3_RPC

Address: *AudioBaseAddress* + 0x006C

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: When a CRC error is detected, previous blocks can be repeated or muted. This register specifies the number of audio blocks to repeat before muting. If this is zero, then blocks are muted until the next frame is decoded.

Confidential

AUD_AC3_KARAMODE **Karaoke downmix**

7 6 5 4 3 2 1 0

AUD_AC3_KARAMODE

Address: *AudioBaseAddress* + 0x006D

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Downmix mode when a karaoke bit stream is received. A karaoke bit stream can be composed of five channels: L (left), R (right), M (music), V1 (vocal 1), V2 (vocal 2). There are two major modes when receiving a karaoke bit stream: aware and capable.

- When in aware mode ([AUD_AC3_KARAMODE](#) = 0), a predefined downmix is applied on all incoming channels.
- When in capable mode ([AUD_AC3_KARAMODE](#) = 4, 5, 6, 7), select different combinations of the two incoming vocal channels, V1 and V2.

An additional mode is added ([AUD_AC3_KARAMODE](#) = 3) to allow multichannel reproduction. In this case, the downmix specified by registers [AUD_AC3_DOWNMIX](#) and [AUD_AC3_DUALMODE](#) is applied.

[7:0] **AUD_AC3_KARAMODE**

0x00: Karaoke aware mode

Left = L + CLEV x M + SLEV x V1

Right = R + CLEV x M + SLEV x V2

0x01, 0x02: Reserved

0x03: Multichannel mode. Consider bit stream as multichannel. Perform downmix according to registers [AUD_AC3_DOWNMIX](#) and [AUD_AC3_DUALMODE](#)

0x04: Karaoke capable mode. Do not reproduce V1, V2.

Left = L + CLEV x M

Right = R + CLEV x M

0x05: Karaoke capable mode. Reproduction V1 only.

Left = L + CLEV x M + 0.707 x V1

Right = R + CLEV x M + 0.707 x V1

0x06: Karaoke capable mode. Reproduction V2 only.

Left = L + CLEV x M + 0.707 x V2

Right = R + CLEV x M + 0.707 x V2

0x07: Karaoke capable mode. Reproduction V1, V2.

Left = L + CLEV x M + V1

Right = R + CLEV x M + V2

Left: output channel. Right: output channel.

L, R, M, V1, V2: input channels (coded in Dolby® Digital karaoke bit stream).

CLEV: center mix level (value provided in the bit stream).

SLEV: surround mix level (value provided in the bit stream).

For further information, refer to annex C of ATSC standard: Digital Audio Compression (AC-3).

Confidential

AUD_AC3_DUALMODE Dual downmix

7 6 5 4 3 2 1 0

AUD_AC3_DUALMODE

Address: *AudioBaseAddress* + 0x006E

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register allows additional downmix to be set when in 2 front/0 rear output mode or when receiving a dual mode incoming bit stream, for example, a disk with two different languages on channel 1 and channel 2. In the following table, channel 1 and 2 represent the output channels after downmix performed with [AUD_AC3_DOWNMIX](#). This register enables mono downmix when [AUD_AC3_DOWNMIX](#) = 010 and [AUD_AC3_DUALMODE](#) = 11.

[7:0] **AUD_AC3_DUALMODE**

00: Output as stereo

01: Output channel 1 on both output L/R

10: Output channel 2 on both output L/R

11: Mix channel 1 and 2 to monophonic and output on both L/R

AUD_AC3_DOWNMIX Downmix

7 6 5 4 3 2 1 0

AUD_AC3_DOWNMIX

Address: *AudioBaseAddress* + 0x006F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7:0] **AUD_AC3_DOWNMIX**

000: 2 front/0 rear Dolby® Surround (LT, RT)

010: 2 front/0 rear (L, R)

100: 2 front/1 rear (L, R, S)

110: 2 front/2 rear (L, R, L_S, R_S Dolby® Phantom)

001: 1 front/0 rear (C)

011: 3 front/0 rear (L, C, R)

101: 3 front/1 rear (L, C, R, S)

111: 3 front/2 rear (L, C, R, L_S, R_S)

AUD_AC3_STATUS0 Dolby® Digital status 0

7 6 5 4 3 2 1 0

Reserved

FS_CODE

BITRATE_CODE

Address: *AudioBaseAddress* + 0x0076

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains bit stream information extracted from the stream.

[7] **Reserved**

[6:5] **FS_CODE**: Code identifying the sampling frequency.

[4:0] **BITRATE_CODE**: Code identifying the bit rate. BITRATE_CODE = FRMSIZECOD[5:1].

Confidential

AUD_AC3_STATUS1 **Dolby® Digital status 1**

7	6	5	4	3	2	1	0
Reserved				LFE	ACMOD		

Address: *AudioBaseAddress* + 0x0077

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains bit stream information extracted from the stream.

[7:4] **Reserved**

[3] **LFE**: Indicates if LFE channel is present in the stream.

[2:0] **ACMOD**: Audio coding mode. Indicates which channels are in use.

AUD_AC3_STATUS2 **Dolby® Digital status 2**

7	6	5	4	3	2	1	0
BSMOD				BSID			

Address: *AudioBaseAddress* + 0x0078

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains bit stream information extracted from the stream.

[7:5] **BSMOD**: Bit stream mode, indicates the type of service.

[4:0] **BSID**: Bit stream identification, indicates the version of the standard.

AUD_AC3_STATUS3 **Dolby® Digital status 3**

7	6	5	4	3	2	1	0
Reserved				CMIXLEVEL		SURMIXLEVEL	

Address: *AudioBaseAddress* + 0x0079

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains bit stream information extracted from the stream.

[7:4] **Reserved**

[3:2] **CMIXLEVEL**: Downmix level of center channel.

[1:0] **SURMIXLEVEL**: Downmix level of surround channel.

AUD_AC3_STATUS4**Dolby® Digital status 4**

7	6	5	4	3	2	1	0
Reserved			DSURMOD		COPYRIGHT	ORIGBS	LANCODE

Address: *AudioBaseAddress* + 0x007A

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains bit stream information extracted from the stream.

[7:5] **Reserved**

[4:3] **DSURMOD**: In 2 front/0 rear mode, indicates if the stream is Dolby® Surround encoded.

[2] **COPYRIGHT**: When set to 1, the stream is protected by copyright.

[1] **ORIGBS**: When set to 1, the stream is an original.

[0] **LANCODE**: When set to 1, a language code is provided in the stream.

AUD_AC3_STATUS5**Dolby® Digital status 5**

7	6	5	4	3	2	1	0
LANCODE							

Address: *AudioBaseAddress* + 0x007B

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains the code of the language of the audio service extracted from the stream.

AUD_AC3_STATUS6**Dolby® Digital status 6**

7	6	5	4	3	2	1	0
Reserved			DIALOG_NORMALIZATION				

Address: *AudioBaseAddress* + 0x007C

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains the code indicating the dialog normalization level extracted from the stream (see Dolby® specifications).

Confidential

AUD_AC3_STATUS7**Dolby® Digital status 7**

7	6	5	4	3	2	1	0
ROOM_TYPE			MIX_LEVEL				AUDPRODIE

Address: *AudioBaseAddress* + 0x007D

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description: This register contains bit stream information extracted from the stream.

[7:6] **ROOM_TYPE**: If AUDPRODIE is set, indicates room type.

[5:1] **MIX_LEVEL**: If AUDPRODIE is set, indicates the sound level.

[0] **AUDPRODIE**: When set to 1, room type and mix level are provided.

48.17 MPEG configuration registers**AUD_MP_SKIP_LFE****Channel skip**

7	6	5	4	3	2	1	0
Reserved							SKP

Address: *AudioBaseAddress* + 0x0068

Type: Read/write

Reset: 0x00 (soft)
Undefined (hard)

Description: If SKP is set to 1, the LFE channel is skipped. If SKP is set to 0, the LFE channel is decoded if present.

AUD_MP_PROG_NUMBER**Program number**

7	6	5	4	3	2	1	0
Reserved							PROG

Address: *AudioBaseAddress* + 0x0069

Type: Read/write

Reset: 0x00 (soft)
Undefined (hard)

Description: When the stream is in second stereo mode, this register specifies which program is played.

[7:1] **Reserved**

[0] **PROG**: Select program #0 or #1
0: L0, R0 in front channels

1: L2, R2 in front channels

Confidential

AUD_MP_DUALMODE **MPEG setup dual mode**

7 6 5 4 3 2 1 0

AUD_MP_DUALMODE							

Address: *AudioBaseAddress* + 0x006E

Type: Read/write

Reset: 0x00 (soft)
Undefined (hard)

Description:

[7:0] **AUD_MP_DUALMODE**

00: Output as stereo

10: Output channel 2 on both outputs L/R

01: Output channel 1 on both outputs L/R

11: Mix channel 1 and 2 to monophonic, and output on both L/R

AUD_MP_DRC **Dynamic range control**

7 6 5 4 3 2 1 0

Reserved							DRC

Address: *AudioBaseAddress* + 0x006A

Type: Read/write

Reset: 0x00 (soft)
Undefined (hard)

Description: When DRC is set to 1, dynamic range control is enabled. The dynamic range is set according to the data transmitted in the DVD MPEG stream.

AUD_MP_CRC_OFF **CRC check off**

7 6 5 4 3 2 1 0

Reserved							OFF

Address: *AudioBaseAddress* + 0x006C

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: When OFF is set to 1, the CRC in MPEG frame is not checked. When OFF is set to 0, the CRC in MPEG frame is checked if it exists. If a CRC error occurs, the decoder software mutes the frame (but does not stop).

AUD_MP_MC_OFF **Multichannel**

7 6 5 4 3 2 1 0

Reserved							MC

Address: *AudioBaseAddress* + 0x006D

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: When MC is set to 1, the multichannel part of the bit stream is not decoded, only the MPEG-1 compatible bit stream is decoded. Must be set to 1 for an MPEG-1 bit stream.

Confidential

AUD_MP_DOWNMIX

MPEG downmix

7 6 5 4 3 2 1 0

AUD_MP_DOWNMIX

Address: *AudioBaseAddress* + 0x006F

Type: Read/write

Reset: 0x08 (soft)
Undefined (hard)

Description: In the table below, LO, RO, CO, LsO, RsO represent the output channels after downmix, and L, R, C, LS, RS are the audio channels. The coefficients K_j, K_C, K_r, K_S, depend on the number of input channels. In the table, the equations are given for a 5-channel input bit stream. If the input bit stream does not contain five channels (L, C, R, LS and RS), the coefficient K_j, corresponding to the channel not present, is equal to 0. If the MPEG bit stream contains only one surround channel (S), replace (K_S x (LS + RS)), (K_S x LS) and (K_S x RS) by (K_S x S) in the equations.

Value	Output mode	Comment
0x00	2 front/0 rear (Dolby® surround L _T , R _T)	$LT = (L + 0.707C - 0.707 \times 0.5 (LS + RS)) / 2.414$, $RT = (R + 0.707C + 0.707 \times 0.5 (LS + RS)) / 2.414$
0x01	1 front/0 rear (C) = mono	$CO = K_j \times L + C + K_r \times R + K_S (LS + RS)$
0x02	2 front/0 rear (L, R) = stereo	$LO = (L + K_C \times C + K_S \times LS) / (1 + K_C + K_S)$, $RO = (R + K_C \times C + K_S \times RS) / (1 + K_C + K_S)$
0x03	3 front/0 rear (L, C, R)	$LO = L + K_S \times LS$, $RO = R + K_S \times RS$, $CO = C$
0x04	2 front/1 rear (L, R, S)	$LO = L + K_C \times C$, $RO = R + K_C \times C$, $LsO = RsO = K_S \times (LS + RS)$
0x05	3 front/1 rear (L, C, R, S)	$LO = L$, $RO = R$, $CO = C$, $LsO = RsO = K_S \times (LS + RS)$
0x06	2 front/2 rear (L, R, L _S , R _S)	$LO = L + K_C \times C$, $RO = R + K_C \times C$, $LsO = LS$, $RsO = RS$
0x07	3 front/2 rear (L, C, R, L _S , R _S)	$LO = L$, $RO = R$, $CO = C$, $LsO = LS$, $RsO = RS$
0x10	2 front/0 rear karaoke capable: V1 OFF, V2 OFF	$Lk = L + 0.707 G$, $Rk = R + 0.707 G$
0x11	2 front/0 rear karaoke capable: V1 ON, V2 OFF (Dolby® Digital like)	$Lk = L + 0.707 A1 + 0.707 G$, $Rk = R + 0.707 A1 + 0.707 G$
0x12	2 front/0 rear karaoke capable: V1 OFF, V2 ON (Dolby® Digital like)	$Lk = L + 0.707 A2 + 0.707 G$, $Rk = R + 0.707 A2 + 0.707 G$
0x13	2 front/0 rear karaoke capable: V1 ON, V2 ON	$Lk = L + 0.707 A1 + 0.707 G$, $Rk = R + 0.707 A2 + 0.707 G$
0x14	2 front/0 rear karaoke capable: V1 ON, V2 OFF	$Lk = L + 0.707 A1 + 0.707 G$, $Rk = R + 0.707 G$
0x15	2 front/0 rear karaoke capable: V1 OFF, V2 ON	$Lk = L + 0.707 G$, $Rk = R + 0.707 A2 + 0.707 G$
0x20	3 front/0 rear karaoke capable: V1 OFF, V2 OFF	$Lk = L$, $Ck = G$, $Rk = R$
0x21	3 front/0 rear karaoke capable: V1 ON, V2 OFF (Dolby® Digital like)	$Lk = L$, $Ck = G + A1$, $Rk = R$
0x22	3 front/0 rear karaoke capable: V1 OFF, V2 ON (Dolby® Digital like)	$Lk = L$, $Ck = G + A2$, $Rk = R$
0x23	3 front/0 rear karaoke capable: V1 ON, V2 ON	$Lk = L + 0.707 A1$, $Ck = G$, $Rk = R + 0.707 A2$
0x24	3 front/0 rear karaoke capable: V1 ON, V2 OFF	$Lk = L + 0.707 A1$, $Ck = G$, $Rk = R$
0x25	3 front/0 rear karaoke capable: V1 OFF, V2 ON	$Lk = L$, $Ck = G$, $Rk = R + 0.707 A2$

Confidential

AUD_MP_STATUS0**MPEG status 0**

7	6	5	4	3	2	1	0
ID	LAY	P	BRI				

Address: *AudioBaseAddress* + 0x0076

Type: Read only

Reset: Undefined

Description:

[7] **ID**: Identifier[6:5] **LAY**: Layer[4] **P**: Protection bit[3:0] **BRI**: Bit rate index**AUD_MP_STATUS1****MPEG status 1**

7	6	5	4	3	2	1	0
SFR	PAD	PRI	MOD	MEX			

Address: *AudioBaseAddress* + 0x0077

Type: Read only

Reset: Undefined

Description:

[7:6] **SFR**: Sampling frequency[5] **PAD**: Padding bit[4] **PRI**: Private bit[3:2] **MOD**: Mode[1:0] **MEX**: Mode extension**AUD_MP_STATUS2****MPEG status 2**

7	6	5	4	3	2	1	0
Reserved				C	OCB	EMP	

Address: *AudioBaseAddress* + 0x0078

Type: Read only

Reset: Undefined

Description:

[7:4] **Reserved**[3] **C**: Copyright[2] **OCB**: Original or copy bit[1:0] **EMP**: Emphasis rate index

AUD_MP_STATUS3**MPEG status 3**

7	6	5	4	3	2	1	0
CEN		SUR		LFE	AMX	DEM	

Address: *AudioBaseAddress* + 0x0079

Type: Read only

Reset: Undefined

Description:

[7:6] **CEN**: Center

[5:4] **SUR**: Surround

[3] **LFE**: Low-frequency effect

[2] **AMX**: Audio mix

[1:0] **DEM**: Dematrix procedure

AUD_MP_STATUS4**MPEG status 4**

7	6	5	4	3	2	1	0
EXT	NML			MFS	MLY	CIB	CIS

Address: *AudioBaseAddress* + 0x007A

Type: Read only

Reset: Undefined

Description:

[7] **EXT**: Extension bit stream present

[6:4] **NML**: Number of multilingual channels

[3] **MFS**: Multilingual F_S

[2] **MLY**: Multilingual layer

[1] **CIB**: Copyright ID bit

[0] **CIS**: Copyright ID start

AUD_MP_STATUS5**MPEG status 5**

7	6	5	4	3	2	1	0
AUD_MP_STATUS5							

Address: *AudioBaseAddress* + 0x007B

Type: Read only

Reset: Undefined

Description: The number of extended ancillary data bytes is contained in this register.

Note: *VCR processing is not allowed while decoding MP3 streams. The only way to have an output on the VCR is to set bit 1 of [AUD_VCR_OUTPUT](#) (hardware copy).*

In MP3 mode, only the downmix value 1 (mono) is available. The other values have no effect because the decoder output can only be 1 or 2 channels.

If the MP3 bit stream is mono encoded, the mono output is automatically copied on L and R channels.

AUD_CRC_OFF**CRC checking**

7	6	5	4	3	2	1	0
Reserved							CRC_OFF

Address: *AudioBaseAddress* + 0x006C

Type: Read only

Reset: No change (soft)
Undefined (hard)

Description:

[7:1] **Reserved**

[0] **CRC_OFF:**

0: The CRC is checked and if a CRC error occurs, the corresponding frame is muted.

1: The CRC in MP3 frame is not checked.

48.18 LPCM registers**AUD_DOWNSAMPLING****Audio downsampling**

7	6	5	4	3	2	1	0
Reserved					DISABLE	AUTO	

Address: *AudioBaseAddress* + 0x0070

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description:

[7:3] **Reserved**

[2] **DISABLE**

[1:0] **AUTO:** If 00 or 01, downsampling is automatically applied if $F_s = 96$ kHz.

If 10, no downsampling. The [AUD_SFREQ](#) register is automatically set to the output frequency value and not to the original input frequency value.

AUD_CHANNEL_ASSIGNMENT**Channel assignment**

7	6	5	4	3	2	1	0

Address: *AudioBaseAddress* + 0x00A8

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register describes the output configuration of the audio channels. The value must be comprised between 0 (mono) and 23 (8 channels). See the specification for read-only Disc/Part4. Version 1.0 table C1.2.

Confidential

AUD_MULTI_CHANNEL Multichannel structure

7	6	5	4	3	2	1	0
Reserved							MC

Address: *AudioBaseAddress* + 0x00A9

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register describes the multichannel structure for output channels (1 for multichannels and 0 for stereo).

AUD_LPCM_DOWNMIX LPCM downmix

7	6	5	4	3	2	1	0

Address: *AudioBaseAddress* + 0x006F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: 0x8: Downmix not applied
0x2: Stereo downmix (using coefficients from host 0x96 to 0xA7)

48.19 LPCM downmix coefficients

LPCM downmix coefficients are set by fourteen registers.

HOST_DM_COEFT_0 LPCM downmix coefficient 0

7	6	5	4	3	2	1	0
0	PH1L	PH2L	PH3L	PH4L	PH5L	Reserved	

Address: *AudioBaseAddress* + 0x0096

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Phase coefficients for channels mixing to Lmix
PHxL: 1 (invert input signal)
PHxL: 0 (input signal is not inverted)

HOST_DM_COEFT_1 LPCM downmix coefficient 1

7	6	5	4	3	2	1	0
PH1R	0	PH2R	PH3R	PH4R	PH5R	Reserved	

Address: *AudioBaseAddress* + 0x0097

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Phase coefficients for channels mixing to Rmix.

HOST_DM_COEFT_2 LPCM downmix coefficient 2

Address: *AudioBaseAddress* + 0x0098

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Lf to Lmix.

HOST_DM_COEFT_3 LPCM downmix coefficient 3

7	6	5	4	3	2	1	0
COEF0R							

Address: *AudioBaseAddress* + 0x0099

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Lf to Rmix.

HOST_DM_COEFT_4 LPCM downmix coefficient 4

7	6	5	4	3	2	1	0
COEF1L							

Address: *AudioBaseAddress* + 0x009A

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Rf to Lmix.

HOST_DM_COEFT_5**LPCM downmix coefficient 5**

7 6 5 4 3 2 1 0

COEF1R

Address: *AudioBaseAddress* + 0x009B

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Rf to Rmix.

HOST_DM_COEFT_6**LPCM downmix coefficient 6**

7 6 5 4 3 2 1 0

COEF2L

Address: *AudioBaseAddress* + 0x009C

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for C to Lmix.

HOST_DM_COEFT_7**LPCM downmix coefficient 7**

7 6 5 4 3 2 1 0

COEF2R

Address: *AudioBaseAddress* + 0x009D

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for C to Rmix.

HOST_DM_COEFT_8**LPCM downmix coefficient 8**

7 6 5 4 3 2 1 0

COEF3L

Address: *AudioBaseAddress* + 0x009E

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Ls/S to Lmix.

Confidential

HOST_DM_COEFT_9**LPCM downmix coefficient 9**

7 6 5 4 3 2 1 0

COEF3R

Address: *AudioBaseAddress* + 0x009F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Ls/S to Rmix.

HOST_DM_COEFT_10**LPCM downmix coefficient 10**

7 6 5 4 3 2 1 0

COEF4L

Address: *AudioBaseAddress* + 0x00A0

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Rs to Lmix.

HOST_DM_COEFT_11**LPCM downmix coefficient 11**

7 6 5 4 3 2 1 0

COEF4R

Address: *AudioBaseAddress* + 0x00A1

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for Rs to Rmix.

HOST_DM_COEFT_12**LPCM downmix coefficient 12**

7 6 5 4 3 2 1 0

COEF5L

Address: *AudioBaseAddress* + 0x00A2

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for LFE to Lmix.

Confidential

HOST_DM_COEFT_13 LPCM downmix coefficient 13

7 6 5 4 3 2 1 0

COEF5R

Address: *AudioBaseAddress* + 0x00A3

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: Gain mixing for LFE to Rmix. The coefficient value $\alpha[x]$, applied for channel x, is calculated with the following formula:

$$\alpha[x] = 2^{-[x + (y/30)]}$$

where $y = [0..29]$, $x = [0..7]$

COEF = [b7, b6, b5, b4, b3, b2, 1, b0]

$x = [b7, b6, b5]$ $y = [b4, b3, b2, b1, b0]$.

48.20 PCM beep tone registers

PCM beep-tone mode is programmed in registers [AUD_DECODESEL](#) (set to 0x7) and [AUD_STREAMSEL](#) (set to 0x3). In PCM beep tone mode, register [AUD_DOWNMIX](#) must be set to 0x8.

AUD_BEEP_FREQ PCM beep tone frequency

7 6 5 4 3 2 1 0

AUD_BEEP_FREQ

Address: *AudioBaseAddress* + 0x0068

Type: Read/write

Reset: 0 (soft)
Undefined (hard)

Description: The value in this register sets the PCM beep tone frequency according to the formula:

$$\text{Beep tone frequency} = (F_s/2)/(\text{Register\_value})$$

with Register_value in the range 8 to 254.

Confidential

AUD_BT_CHANNELCONF PCM beep tone channel configuration

7	6	5	4	3	2	1	0
Reserved	RS	LS	LFE	C	R	L	

Address: *AudioBaseAddress* + 0x0069

Type: Read/write

Reset: 0 (soft)
Undefined (hard)

Description: Register **AUD_DOWNMIX** must be set to 8.

[7:6] **Reserved**

[5] **RS**

0: Right surround channel is forced to 0

1: Right surround channel contains beep tone

[4] **LS**

0: Left surround channel is forced to 0

1: Left surround channel contains beep tone

[3] **LFE**

0: LFE channel is forced to 0

1: LFE channel contains beep tone

[2] **C**

0: Center channel is forced to 0

1: Center channel contains beep tone

[1] **R**

0: Right channel is forced to 0

1: Right channel contains beep tone

[0] **L**

0: Left channel is forced to 0

1: Left channel contains beep tone

Confidential

48.21 Pink noise register

Pink-noise mode is programmed in registers [AUD_DECODESEL](#) (set to 0x4) and [AUD_STREAMSEL](#) (set to 0x3). In pink noise mode, the [AUD_DOWNMIX](#) register must be set to 8, [AUD_OCFG](#) register must be set to 0x00, and attenuation on all channels must be set to -10 dB (registers [AUD_VOLUME0](#) and [AUD_VOLUME1](#)).

AUD_PN_CHANNELCONF PCM pink noise channel configuration

7	6	5	4	3	2	1	0
Reserved	RS	LS	LFE	C	R	L	

Address: *AudioBaseAddress* + 0x0069

Type: Read/write

Reset: 0 (soft)
Undefined (hard)

Description:

[7:6] **Reserved**

[5] **RS**

0: Right surround channel is forced to 0

1: Right surround channel contains pink noise

[4] **LS**

0: Left surround channel is forced to 0

1: Left surround channel contains pink noise

[3] **LFE**

0: LFE channel is forced to 0

1: LFE channel contains pink noise

[2] **C**

0: Center channel is forced to 0

1: Center channel contains pink noise

[1] **R**

0: Right channel is forced to 0

1: Right channel contains pink noise

[0] **L**

0: Left channel is forced to 0

1: Left channel contains pink noise

Confidential

48.22 General mode configuration registers

AUD_DEEMPH

De-emphasis

7	6	5	4	3	2	1	0
FORCE	Reserved					E	

Address: *AudioBaseAddress* + 0x00B5

Type: Read/write (Specific mode)

Reset: No change (soft)
Undefined (hard)

Description: The de-emphasis filter specified here is applied only if bit DEM of register [AUD_PDEC](#) is set. Whenever the de-emphasis status changes on the low pass filter, an interrupt is generated.

This register can be used in each decoder, but only MPEG and DVD_LPCM standards have an emphasis information in the bit stream header.

[7] **FORCE**: In MPEG and DVD_LPCM modes:

0: The emphasis information is extracted from the bit stream and register [AUD_DEEMPH](#) is written with the right corresponding value.

1: The value extracted from the bit stream is ignored. However if this value is different from the one of [AUD_DEEMPH](#) register an interrupt is generated at each frame.

In all other modes the bit FORCE must be set to 1 to force the de-emphasis.

[6:2] **Reserved**

[1:0] **E**

00: None

10: Reserved

01: 50/15s

11: CCITT J.17

AUD_DOWNMIX**Downmix values**

7 6 5 4 3 2 1 0

DOWNMIX_VALUE							

Address: *AudioBaseAddress* + 0x006F

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: The downmix table is identical for all decoders but the downmix coefficients may differ (mix coefficients different between AC-3 and MPEG).

DOWNMIX_VALUE	Output mode
0x00	2 front/0 rear (Dolby® Surround LT, RT)
0x01	1 front/0 rear (C) = mono
0x02	2 front/0 rear (L, R) = stereo
0x03	3 front/0 rear (L, C, R)
0x04	2 front/1 rear (L, R, S)
0x05	3 front/1 rear (L, C, R, S)
0x06	2 front/2 rear (L, R, LS, RS)
0x07	3 front/2 rear (L, C, R, LS, RS)
0x08	No downmix

AUD_DUALMODE**Dual mode**

7 6 5 4 3 2 1 0

Reserved						DUAL_VALUE	

Address: *AudioBaseAddress* + 0x006E

Type: Read/write

Reset: No change (soft)
Undefined (hard)

Description: This register allows additional downmix to be set when in 2 front/0 rear output mode or when receiving a dual mode incoming bit stream (for example, a disk with two different languages on channel 1 and channel 2). In the following table, channel 1 and 2 represent the output channels after downmix performed with [AUD_DOWNMIX](#). This register enables mono downmix when [AUD_DOWNMIX](#) = 2 and [AUD_DUALMODE](#) = 3.

[7:2] **Reserved**

[1:0] **DUAL_VALUE**

00: Output as stereo.

01: Output channel 1 on both output L/R.

10: Output channel 2 on both output L/R.

11: Mix channel 1 and 2 to monophonic and output on both L/R.

49 Audio decoder interface (AUDIF)

49.1 Overview

This module provides an interface to the audio DSP core, which is designed to accept two serial PCM input streams. A full crossbar multiplexor arrangement allows each DSP input channel to be selected from one of two sources.

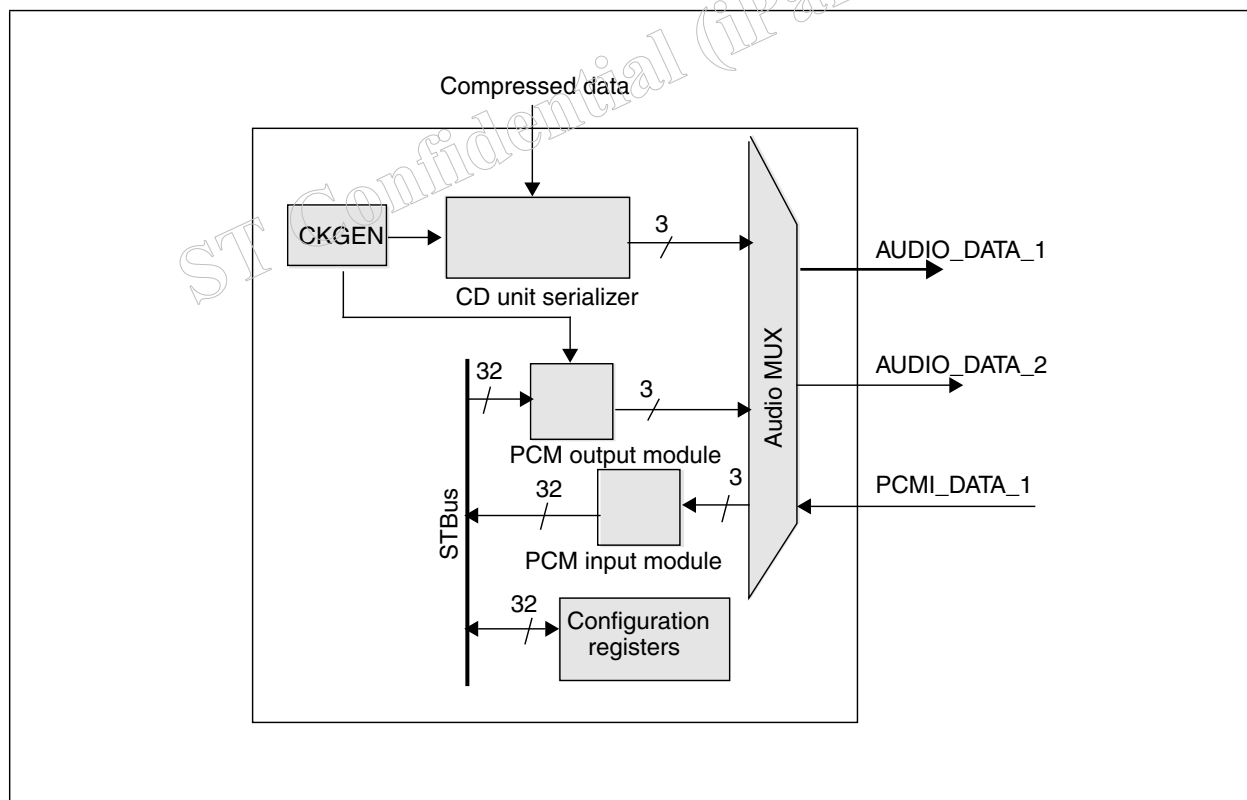
The PCM output block (PCMO) acts as an STBus (SuperHyway) target/buffer and is capable of constructing a serialized data stream in a number of standard audio formats. It is intended to facilitate the playback of PCM file data. Output buffer management (circular, linear, one-shot) is established using the PTI DMAs and is controlled by the CPU, a PCM buffer may be placed anywhere in external memory.

The PCM input block (PCMI) accepts a standard 3-wire serial audio input comprising LRCLOCK, BIT_CLOCK and serial data. This data is buffered, parallelized and formatted for the STBus interface. The DMA manages the transfer of data to and from the local memory using request lines. See [Figure 180: PCM formats on page 540](#).

The CD unit serializer provides a route for compressed audio data arriving from a PTI.

Note: Audio sampling frequencies are supported up to 48 kHz.

Figure 178: Audio interface block diagram



49.2 PCM input module (PCMI)

The PCMI module gets audio data from either PCM (from ADC in formats 1 to 7) or compressed data. It then parallelizes and stores the data in external memory in formats A to D (see [Table 176](#)).

49.2.1 LRCLK

The first sample captured after reset is captured on either LRCLK high or low (configurable).

49.2.2 PCMI formatter

PCMI gets data from memory in formats 1, 2, 3, 4, 6 and 7 and outputs them in formats A, B, or C (see [Table 176](#) for format definitions).

Table 176: Possible formats associations

Function description	Input format AUDIF_ PCMIF	Output format (to mem) AUDIF_IMF	PCMI_LRCLK hardware input pin	First sample capture AUDIF_ILR	Byte swap AUDIF_ISW	Sample precision AUDIF_IPREC
16, 18, 20, 24 bit PCM on 32 slots LRCLK	2, 3, 4, 5, 7	A	32-bit	High or low	Not used	Used
16 bits PCM on 16 LRCLK	0, 1	B	16-bit			
Byte file	6	C	a	High or low	Used	Not used
Compressed data	Not checked	D	a	High or low	Not used	
PCMO as PCMI ^b	Not checked	D	a	High		

a. The PCMI_LRCLK input pin must be either 32-bit slots or hardwired stuck at the AUDIF_ILR value.

b. In PCMO as PCMI mode, capture must begin on a high LRCLK sample as PCMO outputs the first sample on high LRCLK.

49.3 PCM output module (PCMO)

Audio data may be sampled on 32-, 16- or 8-bit long in the external memory buffer. However, audio data may also be continuous (compressed data). PCMO can also directly serialize the 32-bit words when they are compressed.

The scanning order for a 32-bit word is MSB first that is, bit[31], bit[30], bit[29], ... , bit[0].

Note: The default position when the FIFO is empty is that the serializer must send out zeros.

The input FIFO is an STBus target and is capable of storing two LR samples. It generates a pacing request to the DMA when it is half full.

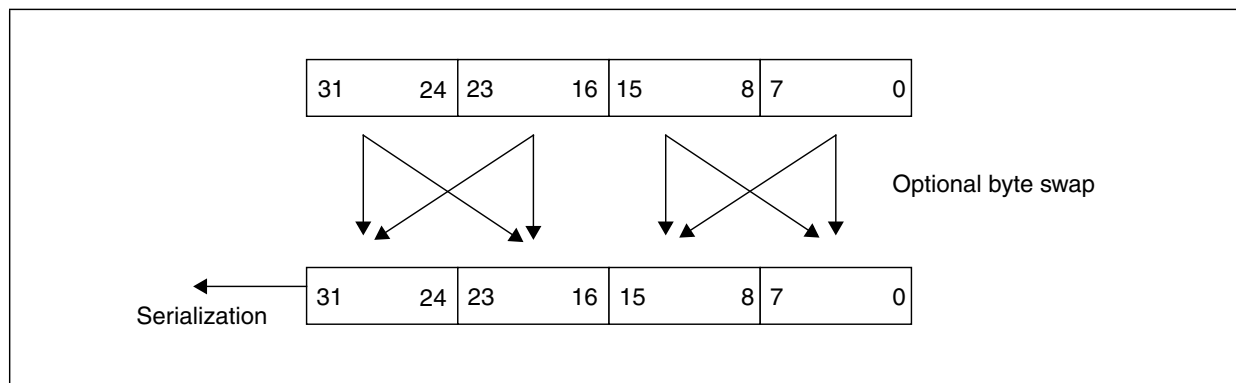
49.3.1 LRCLK

The LRCLK is used to associate the outgoing serial data with the left or right stereo channel. By convention, the first sample sent after reset is set LRCLK high, thereafter the controls within the audio DSP are setup to correctly process the data.

49.3.2 Byte swap

The byte swapping function shown in [Figure 179](#) is implemented before serialization. The main application of this optional mode arises from WinCE. The swap between the two 16-bit words is less costly and can be done by the audio DSP.

Figure 179: Byte swapping



49.3.3 Formatter

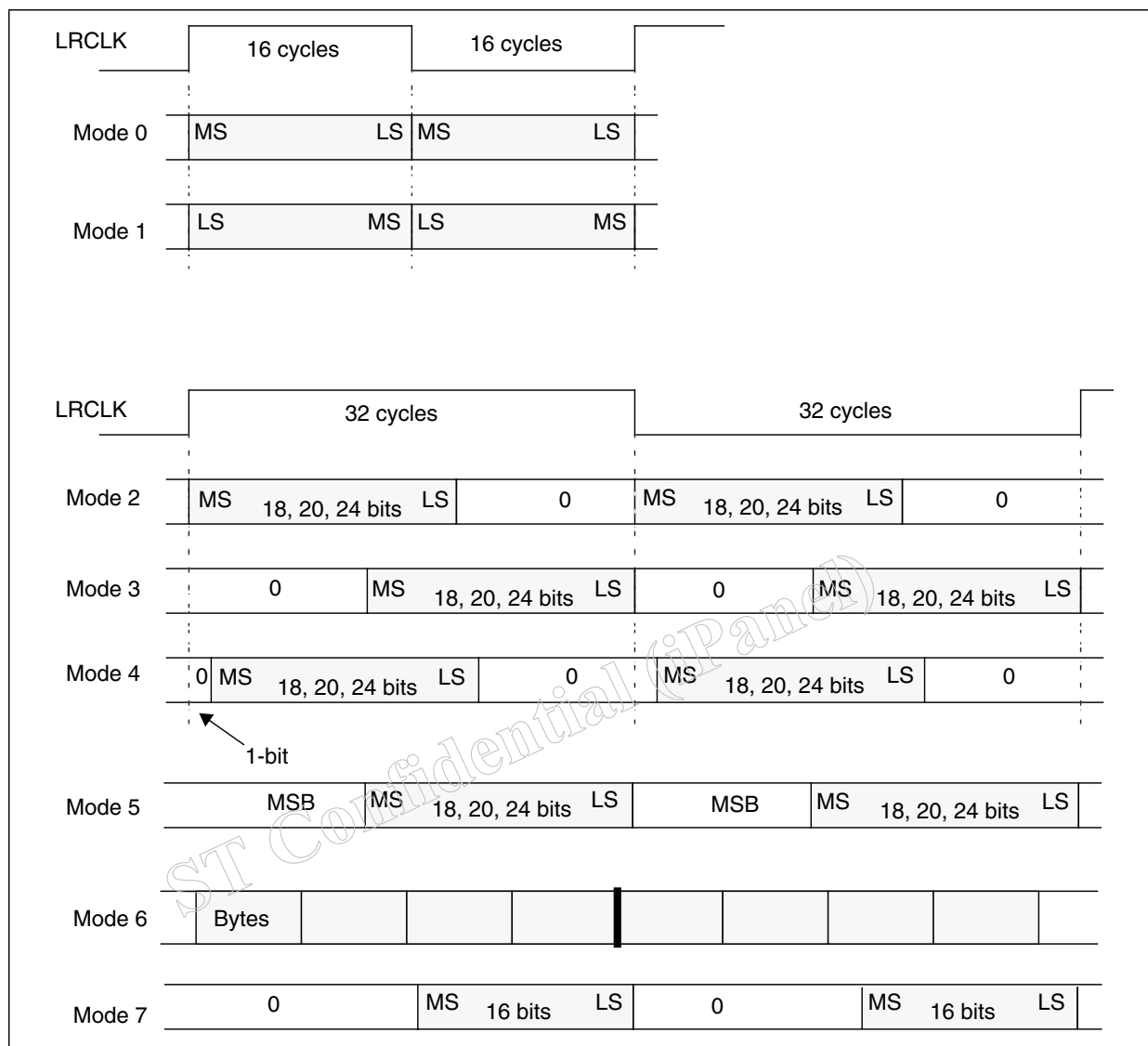
The PCMO gets data from the memory in formats A, B, C or D ([Table 176](#)) and outputs them in modes 1, 2, 3, 4, 6 or 7 (see [Section 49.3.4 on page 540](#) for format definitions).

Table 177: Possible formats associations

Function description	Input format (from mem) AUDIF_ OMF	Output format AUDIF_ PCMOF	Sample precision AUDIF_ OPREC	LRCLK hardware output pin	Byte swap AUDIF_ OSW	First sample polarity: AUDIF_ OLR
16,18,20,24 PCM on 32 slots	A	2, 3, 4, 5, 7	Not used Used Not used Used Must be 16 bit	32 slots	0	Usable
16 bit PCM	B	0, 1	Not used	16 slots		
Byte file	C	6	Not used	32 slots	Usable	
Compressed	D	As it is MSB first				

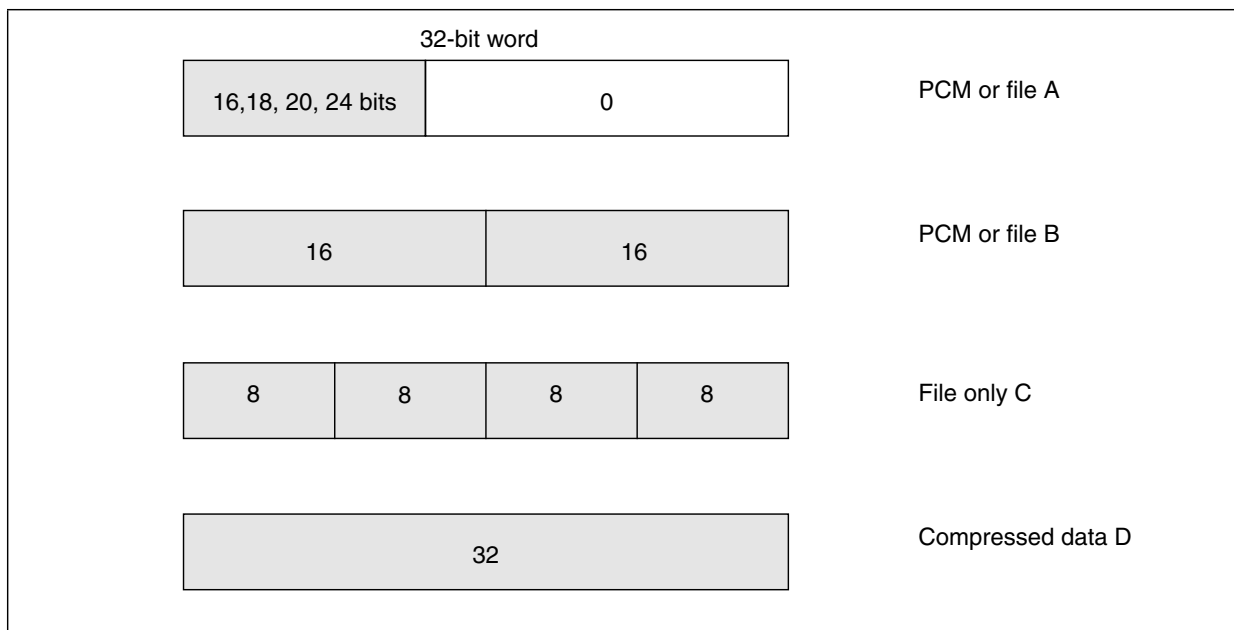
49.3.4 Example formats

Figure 180: PCM formats



Confidential

Figure 181: Memory storage formats



ST Confidential (iPanel)

Confidential

AUDIF_MUX**Audio interface MUX configuration**

	7	6	5	4	3	2	1	0
0x04	Reserved				AUDIF_R1P	AUDIF_OUT1[2:0]		
0x05	Reserved				AUDIF_R2P	AUDIF_OUT2[2:0]		
0x06	Reserved				AUDIF_R3P	AUDIF_OUT3[2:0]		
0x07	Reserved							

Address: *AudioIFBaseAddress* + 0x04

Type: Read/write

Reset: 0x00000000 (Hard)
No effect (nOrst)

Description:

[7:4] **Reserved**

[3] **AUDIF_RnP^a**: Polarity bit

The null input selection deactivates the output (asserts DMA_REQUEST at 0 in case of PCMO module, and asserts AUDIO_DATA, AUDIO_SCLK and AUDIO_LRCLK at 0 in case of PCM output.

For each input of AUDIOIF MUX, one bit codes the active polarity of the corresponding request

0: If input n is not used, then corresponding request is high (not requesting data)

1: If input n is not used, then corresponding request is low (not requesting data)

[2:0] **AUDIF_OUTn**

AUDIF_OUT1 configures output 1 of AUDIOIF, AUDIF_OUT2 configures output 2, AUDIF_OUT3 configures input 1 and should be set to 101.

For each output of the AUDIOIF MUX the 3 bit code selects the input associated.

000: Null input

001: CDUNIT_SERIALIZER

010: PCMO

011: PCMI formatter

100 to 111: Reserved

a. The polarity bit is only used when the input is left unconnected.

AUDIF_PCMICFG

PCMI formatter

	7	6	5	4	3	2	1	0
0x08	Reserved		AUDIF_PCMIF[2:0]			AUDIF_IMF[2:0]		
0x09	Reserved		AUDIF_ISP	AUDIF_IRP	AUDIF_ILR	AUDIF_ISWP	AUDIF_IPREC[1:0]	
0x0A	Reserved							
0x0B	Reserved							

Address: *AudioIFBaseAddress* + 0x08

Type: Read/write

Reset: 0x0000 (Hard)
No effect (nOrst)
No effect (nIrst)

Description:

0x08: [7:6] **Reserved**

0x08: [5:3] **AUDIF_PCMIF**: Input format

000: Mode 0

010: Mode 2

100: Mode 4

110: Mode 6

001: Mode 1

011: Mode 3

101: Mode 5

111: Mode 7

0x08: [2:0] **AUDIF_IMF**: PCM memory storage format

000: A

010: C

111: Unused

001: B

011: D

0x09: [7:6] **Reserved**

0x09: [5] **AUDIF_ISP**

0: Input SDATA is valid on SCLK rising edge (generated on falling edge)

1: Input SDATA is valid on SCLK on falling edge

0x09: [4] **AUDIF_IRP**

0: Request is active low

1: Request is active high

0x09: [3] **AUDIF_ILR**

0: First sample output on LRCLK low

1: First sample output on LRCLK high

0x09: [2] **AUDIF_ISWP**

0: No byte swap

1: Byte swap

0x09: [1:0] **AUDIF_IPREC**: Sample precision

00: 16 bits

10: 20 bits

01: 18 bits

11: 24 bits

0x0A: [7:0] **Reserved**

0x0B: [7:0]

Confidential

AUDIF_PCMOCFG**PCMO formatter**

	7	6	5	4	3	2	1	0
0x0C	Reserved		AUDIF_PCMOF[2:0]			AUDIF_OMF[2:0]		
0x0D	Reserved		AUDIF_OSP	AUDIF_ORP	AUDIF_OLR	AUDIF_OSWP	AUDIF_OPREC[1:0]	
0x0E	Reserved							
0x0F	Reserved							

Address: *AudioIFBaseAddress* + 0x0C

Type: Read/write

Reset: 0x0000 (Hard)
No effect (nOrst)

Description:

0x0C: [7:6] **Reserved**

0x0C: [5:3] **AUDIF_PCMOF[2:0]**: Output format

000: Mode 0

010: Mode 2

100: Mode 4

110: Mode 6

001: Mode 1

011: Mode 3

101: Mode 5

111: Mode 7

0x0C: [2:0] **AUDIF_OMF[2:0]**: PCM memory storage format

000: A

010: C

111: Unused

001: B

011: D

0x0D: [7] **Reserved**

0x0D: [5] **AUDIF_OSP**

When 0 output SDATA is valid on SCLK rising edge (generated on falling edge), when 1 on falling

0x0D: [4] **AUDIF_ORP**

When 0, request is active low, when 1, high

0x0D: [3] **AUDIF_OLR**

When 0, first sample output on LRCLK low, when 1, high

0x0D: [2] **AUDIF_OSWP**

When 0, no byte swap, when 1 byte swap

0x0D: [1:0] **AUDIF_OPREC[1:0]**: Sample precision

00: 16 bits

10: 20 bits

01: 18 bits

11: 24 bits

0x0E: [7:0] **Reserved**

0x0F: [7:0]

Confidential

AUDIF_PCMO

PCMO data from STBus

	7	6	5	4	3	2	1	0
0x30	Reserved							
0x31	Reserved							
0x32	Reserved							
0x33	Reserved							
0x34	Reserved							
0x35	Reserved							
0x36	Reserved							
0x37	Reserved							

Address: *AudiIFBaseAddress* + 0x30

Type: Write only

Reset: 0x0000 0000 0000 0000 (Hard)
 0x0000 0000 0000 0000 (nOrst)

Description: This register is not used. See *Double DMA injection* [on page 481](#).

Confidential

ST Confidential (iPanel)

51 Audio DAC

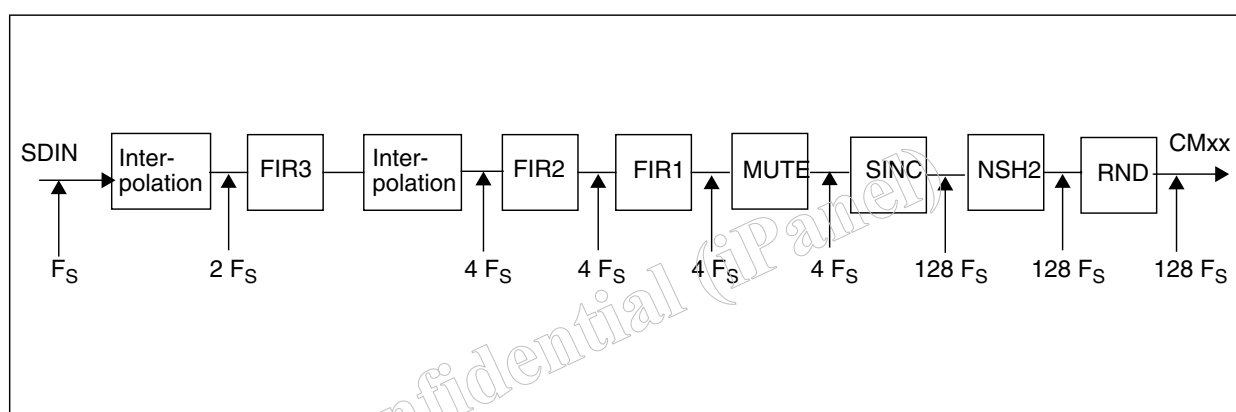
51.1 Description

The audio digital to analog converter is a high performance stereo audio converter which accepts a 24-bit serial data stream from the audio decoder macro block and converts it into a current source analog output signal. This signal is then filtered and transformed into a $1.1 V_{RMS}$ 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 182: Digital flow



The input stream **SDIN** derived from the audio decoder macro cell, sampled at F_S , is first interpolated by two and then filtered by a 75th order FIR filter, **FIR3**. This signal, at $2F_S$, is interpolated by two and filtered by a 20th order FIR filter **FIR2** and followed by a 6th order **FIR1** to compensate the attenuation near f_c . The signal, at $4F_S$, enters the **SINC** filter which interpolates by 32. The noise shaper then transforms this 23-bit signal to 5 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 $512F_S$ which is divided down internally to produce all other clocks.

51.2 Input signals and output pins

51.2.1 Supplies

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

- GNDD/VDDD: internal ground and 1.8 V digital power supply for the control logic, the digital filters and the digital outputs. They are routed inside the chip to the 1.8 V core supply.
- VSSAADAC, VDDAADAC, VDDASADAC: ground and 3.3 V analog supplies for the switches (control of the current sources) are supplied to the chip externally.
- GNDAADAC, VCCAADAC, VCCASADAC: grounds and 3.3 V analog supplies for the polarisation block and the current sources are supplied to the chip externally.

51.2.2 Input/output signals

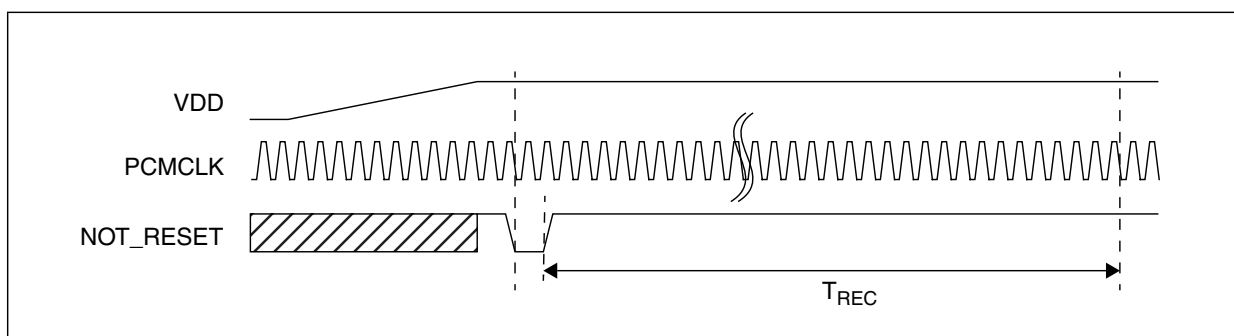
Table 178: Audio DAC output signals

Input/output	Pin name	Description
O	OUTPRIGHT	Right channel, differential positive analog output. The signal is then filtered.
	OUTMRIGHT	Right channel, differential negative analog output. The signal is then filtered.
	OUTPLEFT	Left channel, differential positive analog output. The signal is then filtered.
	OUTMRIGHT	Left channel, differential negative analog output. The signal is then filtered.
I/O	IREF	DAC reference current. This pin should be connected to an external resistor.
	VBGFIL	DAC filtered reference voltage This pin should be connected to an external capacitor.

51.2.3 Reset

A low level on NOT_RESET input puts the system in reset mode by initializing internal counters and controls registers. [Figure 183](#) shows the timing description of this mode.

Figure 183: Reset timing



After the NOT_RESET signal and the delay T_{REC} the data at output is valid (usually 43 samples of data).

51.3 Soft mute

The mute function is controlled by the soft mute input. The current output signal (OUTPRIGHT, OUTPLEFT, OUTMRIGHT, OUTMLEFT) is first attenuated to 96 dB. When the output current reaches the common mode current LCOM, the current sources are switched off one after the other in order to decrease the output current on OUTPRIGHT or OUTPLEFT. Once this sequence is complete, the analog part can be powered down.

Figure 184: Soft mute and digital power down

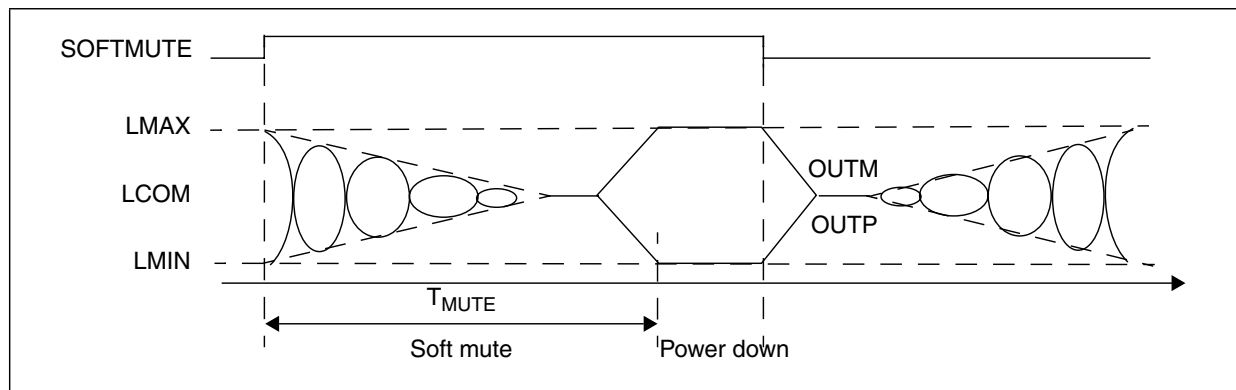


Table 179: Soft mute and digital power down timings

Symbol	Parameter	Min	Typ	Max	Unit
T_{MUTE}	Total time for mute/unmute sequence		$T_{\phi1} + T_{\phi2}$		T_{LR}^a
$T_{\phi1}$	Time for decrease/decrease the gain		1024		T_{LR}^a
$T_{\phi2}$	Time for switch on/off all of the current sources	512		1024	T_{LR}^a

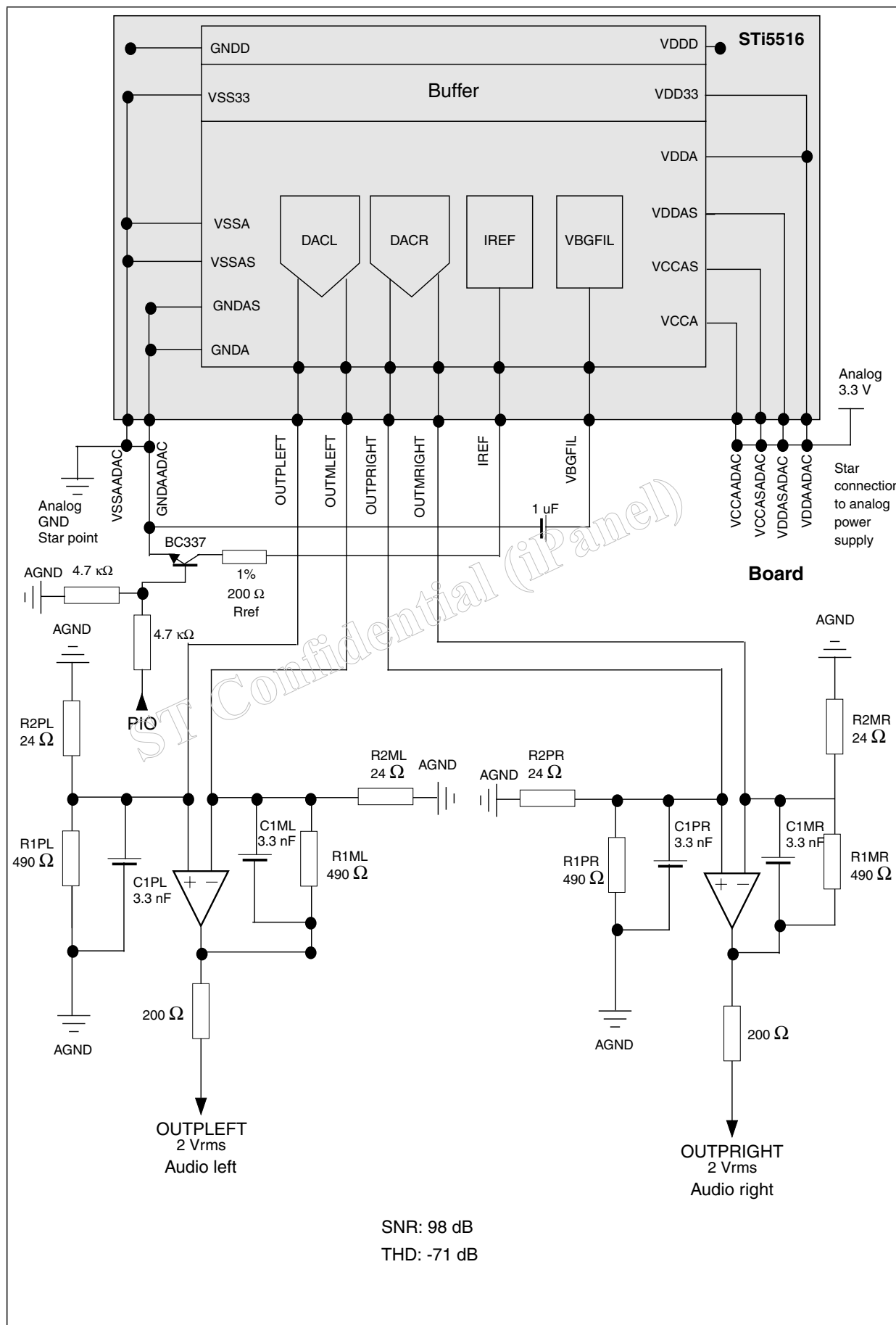
a. $T_{LR} = 1/F_S$ (Period of F_S)

51.4 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 the OUTMRIGHT and OUTMLEFT pins, 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 the IREF pin of the DAC. A typical value for R_{REF} is 200 Ω . R_{REF} should always be higher than 175 Ω to get proper band gap functionality.

In [Figure 185](#) the circuit using the transistor BC337 is part of the pop noise suppressing strategy. It is not needed if using a SWITCH matrix. For more details, see the *STi5516 bug list* (reference ADCS 7428678) and the *STBN pop noise suppressing strategy application note* (reference ADCS 7462125).

Figure 185: High end application (± 9 V symmetrical op-amp power supply)

52 Clock generator

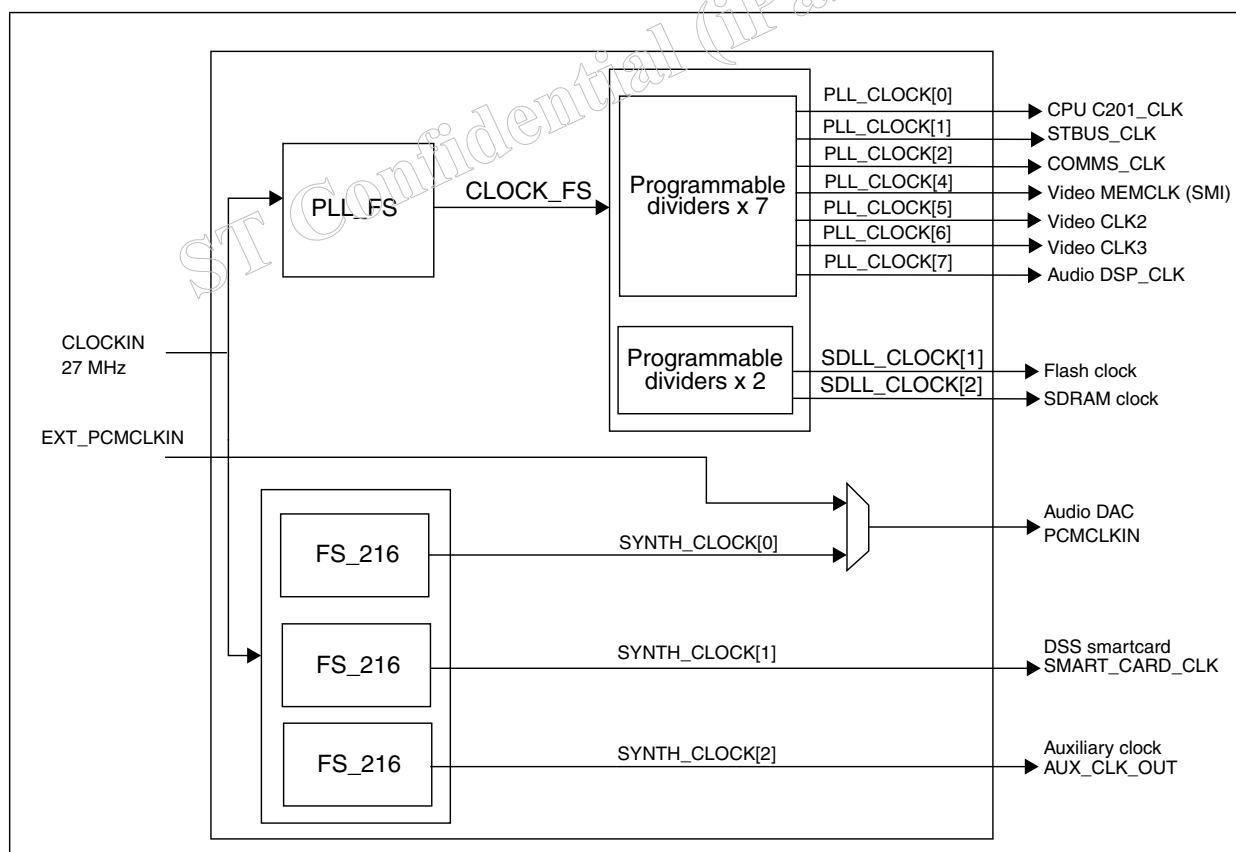
52.1 Overview

All of the STi5516 clocks are generated from a clock generator block, and can be defined in the following groups:

1. The system clocks are based on frequency synthesis PLL_FS. The system PLL multiplies the 27 MHz input clock to generate a common frequency which is divided by programmable dividers to provide various subsystem clocks, as shown in [Figure 186](#).
2. The STi5516 is configured as clock master and the padlogic and external clocks (SDRAM/flash) are phase aligned to the EMI padlogic clock using synchronous delay locked loop technology (SDLL) to ensure maximum performance on the EMI external bus
3. The PCM clock is based on a digital frequency synthesizer included in the clock generator.
4. The smartcard clock is based on a digital frequency synthesizer included in the clock generator.
5. The auxiliary clock is provided by a digital frequency synthesizer included in the clock generator.

[Figure 186](#) illustrates the device clock distribution.

Figure 186: Clocks generation



The clocking module utilizes an analogue PLL block, optimized for stable frequency synthesis (PLL_FS). In addition the three highly stable digital frequency synthesizer blocks provide low jitter clocks for audio DACs, smartcard and auxiliary clocks.

Most clocks are derived from the PLL_FS block which operates at a nominal frequency of up to 600 MHz. This is divided down to produce a series of phase-related clock channels. Each clock channel is created using shift registers to drive a clocked multiplexor. This is fully programmable through software so any waveform can be reproduced to a resolution of half a 600 MHz cycle (0.8 ns). This makes it possible to move clock edges.

Two channels are devoted to the EMI/padlogic external clocks (SDRAM/flash) but have added steering logic. This allows the generated waveforms to be shifted up or down by half a 600 MHz cycle relative to the EMI clock. Phase comparison logic in the padlogic generates these steering signals and allows the two external clocks to be aligned with the internal EMI clock. This functionality is identical to that of a synchronous delay locked loop (SDLL). The net effect gives the appearance of zero delay clock output pads (+/- 0.8 ns).

The guaranteed phase relationship between these channels simplifies interconnect bridging between different subsystem modules and gives lower latency compared to a fully asynchronous clocking scheme.

52.2 Maximum clock frequencies and restrictions

There are three main types of clock channels available at the clock generator output, these are labelled:

- PLL_CLOCK[8:0]: derived from high speed PLL (channels 3 and 8 are not connected or used),
- SDLL_CLOCK[2:0]: derived from high speed PLL with added (SDLL) phase steering. Channel 0 is not connected or used,
- SYNTH_CLOCK[2:0]: derived from low jitter digital frequency synthesiser blocks.

These are mapped to the internal subsystem modules as described in [Table 180](#). The table lists the clock domain settings for the CPU and subsystems with a recommended optimal configuration and maximum frequencies for each clock.

Note: *The recommended optimum configuration column represents the best overall configuration for maximum system performance, given the maximum limits for clock domains and other system constraints.*

Confidential

Table 180: Recommended configuration and maximum clock frequencies

Subsystem	Clock name	Clock	Recommended optimum (MHz) ^a	Maximum frequency (MHz)	MS ratio	Jitter
CPU	C201_CLK	PLL_CLOCK[0]	162 ^b	180 ^b	50:50	< 1 ns
PTI	STBUS_CLK	PLL_CLOCK[1]	81 ^b	90 ^b	50:50	< 1 ns
STBus	STBUS_CLK	PLL_CLOCK[1]	81 ^b	90 ^b	50:50	< 1 ns
TSSUB	STBUS_CLK	PLL_CLOCK[1]	81 ^b	90 ^b	50:50	< 1 ns
Comms	COMMS_CLK	PLL_CLOCK[2]	60.75	60.75 ^c	50:50	< 1 ns
SMI	MEMCLK	PLL_CLOCK[4]	121.5 ^{b,d}	135 ^{b,d}	50:50	< 1 ns
	CLK2	PLL_CLOCK[5]	60.75 ^{b,e}	67.5 ^{b,e}	50:50	< 1 ns
	CLK3	PLL_CLOCK[6]	30.375 ^{b,f}	33.75 ^{b,f}	50:50	< 1 ns
Audio decoder	DSP_CLK	PLL_CLOCK[7]	60.75 ^b	60.8 ^b	50:50	< 1 ns
EMI	STBUS_CLK	PLL_CLOCK[1]	81 ^b	90 ^b	50:50	< 1 ns
Flash	FLASHCLOCK	SDLL_CLOCK[1]	81	90	-	< 1 ns
SDRAM	SDRAMCLOCK	SDLL_CLOCK[2]	81	90	-	< 1 ns
PCM	PCMCLK	SYNTH_CLOCK[0]	Variable	20 ^g	50:50	< 1.3 ns
Smartcard	SMART_CARD_CLK	SYNTH_CLOCK[1]	Variable	20 ^g	50:50	< 1.3 ns
External	AUX_CLK_OUT	SYNTH_CLOCK[2]	Variable	20	50:50	< 1.3 ns

- a. The asynchronous bridges must also be programmed for optimal performance. See the *Programming asynchronous bridges application note* (ADCS 7464707).
- b. Must have an integer divider value (no half values)
- c. Ideally the clock should be a multiple of 10 MHz, deviations from this will cause an error to the nominal 1 μ s IR blaster timer register units so the clock should be as close as possible to this. At 60.75 MHz this gives a timer frequency of 1.0125 MHz (the period is 0.9877 μ s and the deviation is 1.23%).
- d. Must be 180° out of phase with video CLK3
- e. Must be half the speed of MEMCLK
- f. Must be a quarter of the speed of MEMCLK, less than the STBus frequency when multiplied by 1.5 and 180° out of phase with video MEMCLK
- g. Frequency external to the device. The internal frequency can be higher

52.3 Modes of operation

The STi5516 has on-chip phase lock loops to provide the following high frequency subsystem and external clocks:

- processor clock,
- interconnect clock,
- video clocks,
- audio clock,
- flash and SDRAM clocks (external clocks).

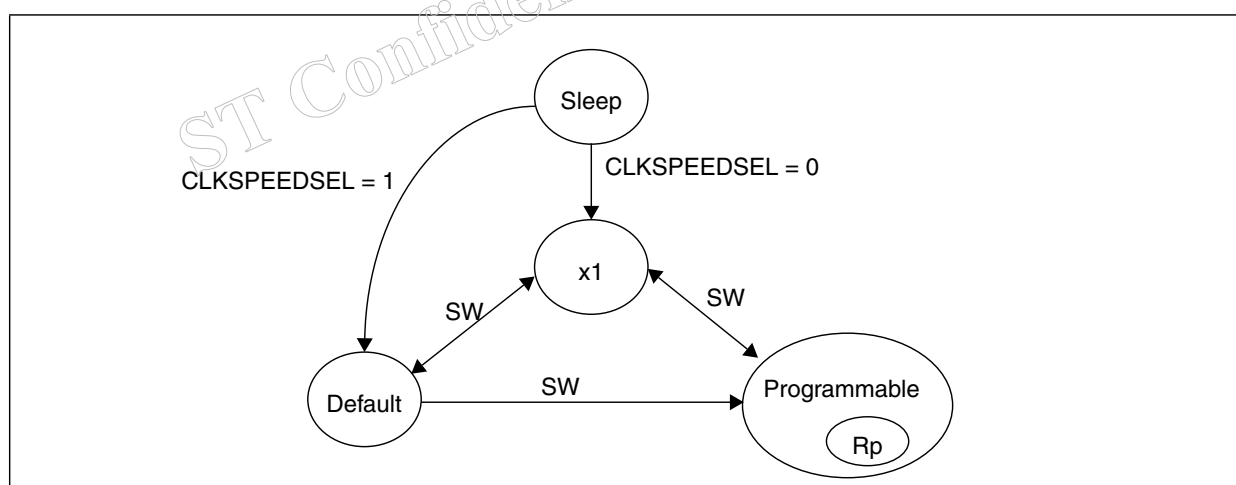
The clock generation architecture is arranged in several layers of increasing complexity allowing the system to be booted easily at low speed and then reconfigured later through software as necessary. A static pin (CLKSPEEDSEL) selects one of two start up modes and software is later able to program individual system clocks.

These clocks are derived from CLOCKIN (27 MHz). STi5516 operates in master configuration only. Within this configuration the clock generator operates in one of three states.

- X1 mode, where the subsystem and external clocks are directly clocked by the input clock.
- default mode, where the subsystem and external clock rates are selected from a table of defaults, see [Table 182 on page 557](#).
- programmable mode, where the subsystem and external clocks rates may be programmed by the user. These values must be programmed in X1 mode. A software induced transition to the programmable state then causes the clocks to be updated.

This is illustrated in [Figure 187](#).

Figure 187: Clock generator operating states



The transition between states is determined by the CLKSPEEDSEL pin or a processor WRITE_REQUEST. The label SW in the above diagram ([Figure 187](#)) indicates a software induced transition, in other words the CPU writes to the clock generator state register.

The clock generator has a reduced power mode which is accessed in programmable mode. This sources the programmable dividers with the 27 MHz clock and not the PLL clocks. The dividers are configured with divide ratios and the subsystem and external clocks are sourced with divided 27 MHz clocks.

52.3.1 Low power mode

The low power clock is derived from an external 32 kHz low power clock.

The clock generator enters low power mode when the LP_MODE input from the low power controller is asserted.

In low power mode all other clocks are set to zero and only the low power controller is clocked. An internal timer or external interrupt can cause LP_MODE to be deasserted, the clock generator then restarts all subsystem clocks.

52.3.2 Start up configurations

Table 181: Default subsystem clock rates

Subsystem	X1 mode		Default mode PLL freq = 450 MHz	
Subsystem	Frequency	Source	Frequency	Source
C201	27 MHz	Direct	90 MHz	PLL_FS
STBus, EMI	27 MHz	Direct	90 MHz	PLL_FS
TSSub	27 MHz	Direct	90 MHz	PLL_FS
PTI	27 MHz	Direct	90 MHz	PLL_FS
Comms	27 MHz	Direct	60 MHz	PLL_FS
AV/MEMCLK	27 MHz	Direct	112.5 MHz	PLL_FS
AV/CLK2	27 MHz	Direct	56.25 MHz	PLL_FS
AV/CLK3	27 MHz	Direct	28.125 MHz	PLL_FS
Audio	27 MHz	Direct	56.25 MHz	PLL_FS

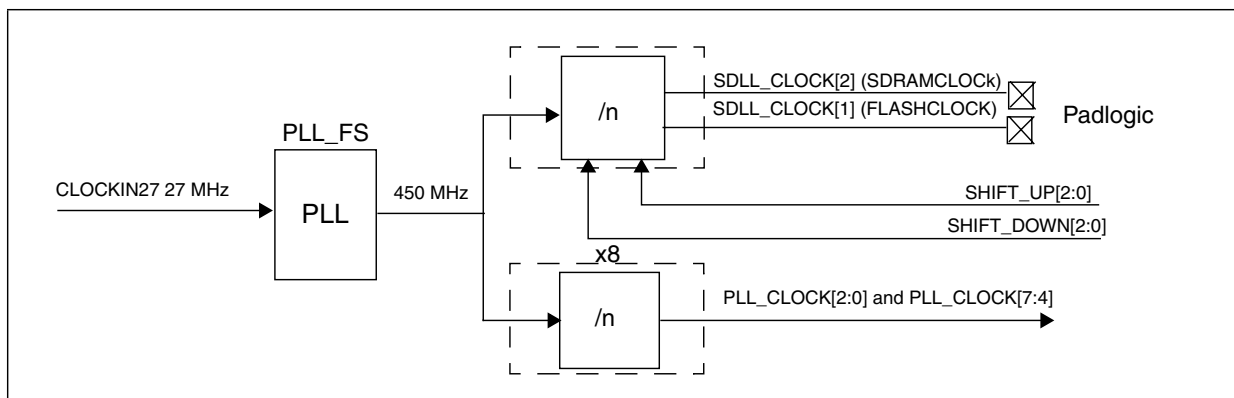
52.3.3 Clock master default mode

Figure 188 shows the clock generator set up for master default mode. In this mode all major subsystem clocks are derived from the PLL_FS. The phase difference between subsystem and padlogic clocks is measured in the EMI padlogic and passed to the clock generator (SHIFT_UP and SHIFT_DOWN signals). Alignment is achieved through an SDLL circuit on each of the three clock outputs. Figure 188 only shows the PLL circuit. Outside this block is a phase comparator which, when using the hardware method (SHIFT_EN of the SHIFT_CONFIG register = 1), compares the SDLL_CLOCK at the pad with the EMI clock. The difference between the two clocks is reflected by the value on the SHIFT_UP and SHIFT_DOWN signals, which in turn changes the SDLL_CLOCK phase with respect to the EMI clock. This is repeated until the phases are aligned $\pm 1/2$ PLL clock cycle.

In software mode (SHIFT_EN = 0), the shift is visible to software through the SHIFT_UP and SHIFT_DN bits in the SHIFT_CONFIG register, otherwise shifting is fully automated. When using the hardware method the register can still be read by software so that a decision can be made to stop shifting or set SHIFT_EN to 0.

- Note: 1 Only the EMI clock is required when no external synchronous devices such as flash or SDRAM are used*
- 2 The use of hardware shift enable mode is recommended when in default or programmable mode. This must be enabled by writing to the SHIFT_CONFIG[2:0] register before using the EMI.*

Figure 188: Clock master default mode



The up/down steering signals to the SDLL may be masked using a software register and the output of the phase comparators accessed directly by software from the padlogic. This enables software to break the loop and perform the edge adjustment by reprogramming the channels, testing and reprogramming until a static operating point is found.

As all PLL derived output channels are generated in the same way, there is very little skew between channel outputs. Very low skew must be maintained between leaf nodes of all clock trees originating from these channels by using careful clock balancing. CTGEN performs clock balancing between all PLL derived clock trees by describing them as the same source.

52.4 System clocks

All of the system clocks are generated from the system PLLs and programmable dividers. The reference input frequency is the 27 MHz clock. This reference is multiplied by programmable integrated PLLs, of which the outputs are steered to a bank of 12 dividers. [Table 182](#) summarizes the system clocks for X1 and default mode.

Each system clock can be bypassed with CLOCKIN and enabled (the output clock is turned off). The system clocks can be enabled independently but, during programmable mode, all system clocks are bypassed with the designated clock.

Two clocks are required at the device pins in order to support SDRAM and synchronous flash:

- SDRAMCLOCK,
- FLASHCLOCK.

These clocks are produced in the padlogic using a macro that mirrors the configuration of the EMI, in addition the EMI is able to switch clock speeds dynamically between accesses. The source and sampling clock in the padlogic tracks this change using the same macro. Each clock may be set to an integer division N of the EMI clock frequency where $N = 1, 2$ or 3 .

Clock deskewing and alignment in the system is accomplished using two SDLL channels within the clock generator block. Phase detectors in the padlogic steer the SDLL channels to align the two external device clocks with the internal clock when in master mode. This creates a single virtual bus clock that is aligned through a multichip system to within 0.8 ns.

The PLL lock state is readable, and the PLL reset is programmable. The processor makes a decision to switch to the PLL frequency based on the lock state of PLL_FS. The lock sensitivity of the PLL is software programmable, but under some circumstances the lock signal never goes high, for example with a jittery reference. To circumvent this, software can interpret the lock signal. The clock generator must first start in bypass mode for the CPU to boot and run software.

Table 182: PLL_FS output frequency according to CLKSPEEDSEL

Subsystem	Channel	X1 mode CLKSPEEDSEL = 0		Default mode CLKSPEEDSEL = 1 PLL freq = 450 MHz	
		Frequency	Source	Frequency	Source
C201	PLL_CLOCK[0]	27 MHz	Direct	90 MHz	PLL_FS
Interconnect	PLL_CLOCK[1]	27 MHz	Direct	90 MHz	PLL_FS
TSsub	PLL_CLOCK[1]	27 MHz	Direct	90 MHz	PLL_FS
PTIs	PLL_CLOCK[1]	27 MHz	Direct	90 MHz	PLL_FS
Comms	PLL_CLOCK[2]	27 MHz	Direct	90 MHz	PLL_FS
Video MEMCLKIN	PLL_CLOCK[4]	27 MHz	Direct	112.5 MHz	PLL_FS
Video CLK2	PLL_CLOCK[5]	27 MHz	Direct	56.25 MHz	PLL_FS
Video CLK3	PLL_CLOCK[6]	27 MHz	Direct	28.125 MHz	PLL_FS
DSP	PLL_CLOCK[7]	27 MHz	Direct	56.25 MHz	PLL_FS
EMI	PLL_CLOCK[1]	27 MHz	Direct	90 MHz	PLL_FS
Flash SDRAM	SDLL_CLOCK[1] SDLL_CLOCK[2]	27 MHz	Direct	90 MHz	PLL_FS
PCMCLKIN	SYNTH_CLOCK[0]	27 MHz	Direct	Configurable	Subsystem clock
SMART_CARD_CLK	SYNTH_CLOCK[1]	27 MHz	Direct	Configurable	Subsystem clock
AUX_CLK_OUT	SYNTH_CLOCK[2]	27 MHz	Direct	Configurable	Subsystem clock

The system PLL_FS multiplies the 27 MHz input clock, the output frequency is calculated as below, where N = 150, M = 18, P = 0 for clock_fs = 450 MHz for default states. For optimum operating frequency, clock_fs = 600 MHz with setting of N = 200, M = 18, P = 0.

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

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

$$1 \leq M \leq 255, 1 \leq N \leq 255, 0 \leq P \leq 5$$

$$1 \text{ MHz} \leq \frac{F_{(\text{clockin})}}{M} \leq 2 \text{ MHz}$$

$$200 \text{ MHz} \leq \left(\frac{2 \times N}{M} \right) \times F_{(\text{clockin})} \leq 622 \text{ MHz}$$

$$F_{(\text{clockin})} \leq 200 \text{ MHz}$$

52.5 Programmable dividers

The programmable dividers are sourced by the PLL output clocks during normal operation and by the 27 MHz clock when in reduced power mode. The lowest programmable frequency for each clock domain in this mode is 1.35 MHz

The dividers are programmable with a maximum divide ratio of 4:20, with half divide ratios up to a maximum of 9.5. This provides all the subsystem and padlogic clock requirements.

The divider outputs are disabled for low power mode. In reduced power mode the dividers may be configured to clock the subsystems at frequency divisions of 27 MHz, 10.8 MHz down to a minimum of 1.35 MHz.

The dividers can switch between X1 and programmable modes without adding glitches to the subsystem and padlogic clocks.

Example frequencies are shown in [Table 183](#) and [Table 184](#).

Table 183: Example output frequencies in MHz

Divide ratio	Output frequency (MHz)		
	N = 148, M = 24, P = 0, CLOCKIN = 27 MHz, PLL = 333 MHz	N = 150, M = 18, P = 0, CLOCKIN = 27 MHz, PLL = 450 MHz	N = 222, M = 24, P = 0, CLOCKIN = 27 MHz, PLL = 499.5 MHz
4	83.25	112.5	124.88
4.5	74	100	111
5	66.6	90	99.9
5.5	60.55	81.81	90.82
6	55.5	75	83.25
6.5	51.23	69.23	76.85
7	47.57	64.28	71.36
7.5	44.4	60	66.6
8	41.63	56.25	62.44
8.5	39.18	52.94	58.76
9	37	50	55.5
9.5	35.05	47.36	52.58
10	33.3	45	49.95
10.5	31.71	42.86	47.57
11	30.27	40.9	45.41
12	27.75	37.5	41.63
13	25.62	34.62	38.42
14	23.79	32.14	35.68
15	22.2	30	33.3

Table 184: Example output frequencies in MHz

Divide ratio	Output frequency (MHz)		
	N = 200, M = 18, P = 0, CLOCKIN = 27 MHz, PLL = 600 MHz	N = 207, M = 18, P = 0, CLOCKIN = 27 MHz, PLL = 612 MHz	N = 233, M = 18, P = 0, CLOCKIN = 27 MHz, PLL = 629 MHz
4	150	155.25	174.75
4.5	133.33	138	155.33
5	120	124.2	139.8
5.5	109.09	112.91	127.09
6	100	103.5	116.5
6.5	92.31	95.54	107.54
7	85.71	88.7	99.86
7.5	80	82.8	93.2
8	75	77.63	87.38
8.5	70.59	73.06	82.24
9	66.67	69	77.67
9.5	63.16	65.37	73.58
10	60	62.1	69.9
10.5	57.14	59.14	66.57
11	54.55	56.46	63.55
12	50	51.75	58.25
13	46.15	47.77	53.77
14	42.86	44.36	49.93
15	40	41.4	46.6

Confidential

52.6 PCM clock

A frequency synthesizer generates the DAC clocks for the audio decoder. After hard reset, the PCM clock pins are input to the device. When the SYNTH0_CONFIG0 and SYNTH0_CONFIG1 registers are set and the clock generator transitioned into programmable mode, the PCMCLK clock becomes an output.

Table 185: Audio frequency values

Audio frequency	SYNTH0_CONFIG0				SYNTH0_CONFIG1
	PLL_DIV ^a	NDIV	SDIV	MD	PE
384 x 32 kHz	0x0001	0x0001	0x0004	0x0011	0x3600
384 x 44.1 kHz	0x0001	0x0001	0x0003	0x0019	0x3EB2
384 x 48 kHz	0x0001	0x0001	0x0003	0x0017	0x4800
256 x 32 kHz	0x0001	0x0001	0x0004	0x001A	0x5100
256 x 44.1 kHz	0x0001	0x0001	0x0004	0x0013	0x6F05
256 x 48 kHz = 384 x 32 kHz	0x0001	0x0001	0x0004	0x0011	0x3600
256 x 96 kHz	0x0001	0x0001	0x0003	0x0011	0x3600
384 x 96 kHz	0x0001	0x0001	0x0002	0x0017	0x4800
256 x 192 kHz	0x0001	0x0001	0x0002	0x0011	0x3600
192 x 192 kHz = 384 x 96 kHz	0x0001	0x0001	0x0002	0x0017	0x4800
512 x 32 kHz ^b	0x0001	0x0001	0x0003	0x001A	0x50FF
512 x 44.1 kHz ^a	0x0001	0x0001	0x0003	0x0013	0x6F05
512 x 48 kHz ^a	0x0001	0x0001	0x0003	0x0011	0x3600
768 x 32 kHz	0x0001	0x0001	0x0003	0x0011	0x3600
768 x 44.1 kHz	0x0001	0x0001	0x0002	0x0019	0x3EB1
768 x 48 kHz	0x0001	0x0001	0x0002	0x0017	0x47FF

a. Normal operating mode

b. If the frequency synthesizer generates the PCMCLK at 512 x Fs, a divide by 2 in the audio decoder produces 256 x Fs. This allows the internal DAC and an external DAC working with a PCMCLK of 256 x Fs to be used.

Table 186 shows the combinations which are used for audio PCM clock frequencies with the audio decoder, the internal audio DAC and the PCMCLK output.

Table 186: PCM clock frequencies for the audio decoder and internal audio DAC

SYNTH_CLOCK[0]	Audio DSP	Internal DAC	PCMCLK output
512 x Fs	256 x Fs	512 x Fs	256 x Fs
768 x Fs	384 x Fs	Not available	384 x Fs

The SYNTH0_CONFIGn registers select the PCMCLK output frequency from one of the above values.

52.7 Smartcard clocks

The smartcard clock operates from 843 kHz to 20 MHz. The common smartcard frequencies are detailed in [Table 187](#). These register values must be used to achieve this frequency.

Table 187: Smartcard frequencies

Frequency	SYNTH1_CONFIG0				SYNTH1_CONFIG1
	PLL_DIV ^a	NDIV	SDIV	MD	PE
1 MHz	0x0001	0x0001	0x0007	0x001A	0x0000
2 MHz	0x0001	0x0001	0x0006	0x001A	0x0000
3 MHz	0x0001	0x0001	0x0006	0x0012	0x8000
4 MHz	0x0001	0x0001	0x0005	0x001A	0x0000
4.5 MHz	0x0001	0x0001	0x0005	0x0017	0x0000
18.436 MHz	0x0001	0x0001	0x0003	0x0017	0x48A7
27 MHz	0x0001	0x0001	0x0003	0x000F	0x0000

a. Normal operating mode

52.8 Auxiliary clock

The auxiliary clock operates from 843 kHz to 20 MHz. [Table 188](#) gives example values for programming the auxiliary clock.

Table 188: Auxiliary clock programming values

Frequency	SYNTH2_CONFIG0				SYNTH2_CONFIG1
	PLL_DIV ^a	NDIV	SDIV	MD	PE
1 MHz	0x0001	0x0001	0x0007	0x001A	0x0000
2 MHz	0x0001	0x0001	0x0006	0x001A	0x0000
3 MHz	0x0001	0x0001	0x0006	0x0012	0x7FFF
4 MHz	0x0001	0x0001	0x0005	0x001A	0x0000
5 MHz	0x0001	0x0001	0x0005	0x0015	0x3333
10 MHz	0x0001	0x0001	0x0004	0x0015	0x3333
15 MHz	0x0001	0x0001	0x0003	0x001C	0x1999
20 MHz	0x0001	0x0001	0x0003	0x0015	0x3333

a. Normal operating mode

53 Clock generator registers

Addresses are provided as the *ClockgenBaseAddress* + offset.

The *ClockgenBaseAddress* is:

0x2001 3000.

A register summary is given in [Table 33: Clock generator registers on page 54](#).

REGISTER_LOCK

Lock register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LOCK	KEY						

Address: *ClockgenBaseAddress* + 0x300

Type: Read/write

Reset: 0

Description: After reset, all configuration registers are locked. Configuration registers can only be updated when the lock bit is 0.

[15:9] **Reserved**

[8] **LOCK**: LOCK bit, default configuration = 0x1

0: Configuration registers unlocked for write access. 1: Configuration 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

Programming involves the following sequence.

- Unlock the registers by writing the keyword followed by the bit wise inverse.
- Configure the registers as required.
- Lock the registers by writing 1 to the LOCK bit (bit 8 of REGISTER_LOCK).

DIVIDER_MODE

Mode transitions register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														MODE1	MODE0

Address: *ClockgenBaseAddress* + 0x0F8

Type: Read/write

Reset: 0

Description: The DIVIDER_MODE register controls the state transitions of the clock generator. To program configuration registers PLLFSDIVn, CLKDIVn_CONFIGn and SDLLDIVn_CONFIGn, the clock generator must transition into the programmable state (DIVIDER_MODE, 0x02) after the new values have been written in x1 mode (DIVIDER_MODE, 0x00). The other configuration register can be written to after the register are unlocked but it is recommended to transition into x1 mode before initiating any writes and then transition into programmable mode upon completion.

[15:2] **Reserved**

[1:0] **MODE1, MODE0**

Used to transition between x1, default and programmable modes.

MODE1 = 0, MODE0 = 0: Transition into x1 mode

MODE1 = 0, MODE0 = 1: Transition into default mode

MODE1 = 1, MODE0 = 0: Transition into programmable mode

MODE1 = 1, MODE0 = 1: Transition into x1

REDUCED_POWER**Reduced power mode**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			REDUCED_DIVIDER_SDLL2	REDUCED_DIVIDER_SDLL1	Reserved		REDUCED_DIVIDER_PLL7	REDUCED_DIVIDER_PLL6	REDUCED_DIVIDER_PLL5	REDUCED_DIVIDER_PLL4	Reserved	REDUCED_DIVIDER_PLL2	REDUCED_DIVIDER_PLL1	REDUCED_DIVIDER_PLL0	PLL_BYPASS

Address: *ClockgenBaseAddress* + 0x0FC

Type: Read/write

Reset: 0

Description:

[15:13] **Reserved**

[12] **REDUCED_DIVIDER_SDLL2**

!: Selects reduced power mode for SDLL_CLOCK[2] divider, default configuration 0x00

[11] **REDUCED_DIVIDER_SDLL1**

!: Selects reduced power mode for SDLL_CLOCK[1] divider, default configuration 0x00

[10:9] **Reserved**

[8] **REDUCED_DIVIDER_PLL7**

!: Selects reduced power mode for PLL_CLOCK[7] divider, default configuration 0x00

[7] **REDUCED_DIVIDER_PLL6**

!: Selects reduced power mode for PLL_CLOCK[6] divider, default configuration 0x00

[6] **REDUCED_DIVIDER_PLL5**

!: Selects reduced power mode for PLL_CLOCK[5] divider, default configuration 0x00

[5] **REDUCED_DIVIDER_PLL4**

!: Selects reduced power mode for PLL_CLOCK[4] divider, default configuration 0x00

[4] **Reserved**

[3] **REDUCED_DIVIDER_PLL2**

!: Selects reduced power mode for PLL_CLOCK[2] divider, default configuration 0x00

[2] **REDUCED_DIVIDER_PLL1**

!: Selects reduced power mode for PLL_CLOCK[1] divider, default configuration 0x00

[1] **REDUCED_DIVIDER_PLL0**

!: Selects reduced power mode for PLL_CLOCK[0] divider, default configuration 0x00

[0] **PLL_BYPASS**

!: Selects reduced power mode for all dividers, default configuration 0x0

Confidential

PLLFSDIV0

FS_PLLDIV0 register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N								M							

Address: *ClockgenBaseAddress* + 0x000

Type: Read/write

Reset: 0

Description: The values needed for N and M can be obtained using the formulae in [Section 52.4: System clocks](#). [Section 52.5: Programmable dividers](#) shows some typical values for M and N, and the clock frequencies that are obtained for the divide ratios available.

[15:8] **N[7:0]**

Predivider value, default configuration = 0x96

[7:0] **M**

Feedback divider, default configuration = 0x12

PLLFSDIV1

FS_PLLDIV1 register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	LOK	EN	SETUP										P		

Address: *ClockgenBaseAddress* + 0x004

Type: Read/write

Reset: 0

Description: PLL_FS postdivider and electrical setup configuration

[15] **Reserved**

[14] **LOK**

Lock signal indicates state of PLL_FS phase alignment, LOK is high when PLL_FS is locked

0: PLL_FS unlocked

1: PLL_FS locked

[13] **EN**

Global enable, when low PLL_FS is in IDDQ mode for test, default configuration = 0x01

0: PLL disabled

1: PLL enabled

[12] **SETUP[9]**

PLL_BYPASS, default configuration = 0x0

[11:9] **SETUP[8:6]**

Lock detector threshold, default configuration = 0x4

[8:7] **SETUP[5:4]**

Start up control, default configuration = 0x2

[6:3] **SETUP[3:0]**

Charge pump control, default configuration = 0x07

[2:0] **P[2:0]**

Postdivider value, default configuration = 0x00

CLOCK_SEL1**PLL clock MUX selection register**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCMCLKIN_SEL	SOFT_BYPASS_27												PLL_SEL2	PLL_SEL1	PLL_SELO

Address: *ClockgenBaseAddress + 0x200*

Type: Read/write

Reset: 0

Description:

[15] **PCMCLKIN_SEL**

Select SYNTH[0] output or external clock, default configuration = 0

0: SYNTH_CLOCK[0] derived from frequency synthesizer SYNTH[0]

1: SYNTH_CLOCK[0] derived from external input EXT_PCMCLKIN.

[14:3] **SOFT_BYPASS_27**

Controls PLL_FS bypass switching frequency. Always set to 0.

0: Bypass mode, divider outputs are sourced from CLOCKIN27

1: Reserved

Each bit of SOFT_BYPASS_27 represents a divider

SOFT_BYPASS_27[0] = Bypass switch for PLL_CLOCK[0]

SOFT_BYPASS_27[8] = Bypass switch for PLL_CLOCK[8]

SOFT_BYPASS_27[9] = Bypass switch for SDLL_CLOCK[0]

SOFT_BYPASS_27[11] = Bypass switch for SDLL_CLOCK[2]

[2] **PLL_SEL2**

Selects PLLFS_CLOCK as divider input clock to SDLL_CLOCK[2:0]. Always set to 0.

0: PLLFS_CLOCK input to divider

1: Reserved

[1] **PLL_SEL1**

Selects PLLFS_CLOCK as divider input clock to PLL_CLOCK[8]. Always set to 0.

0: PLLFS_CLOCK input to divider

1: Reserved

[0] **PLL_SELO**

Selects PLLFS_CLOCK as divider input clock to PLL_CLPCK[7:0]. Always set to 0.

0: PLLFS_CLOCK input to divider.

1: Reserved

Confidential

53.1 Programmable dividers registers

The programmable dividers registers are identical for each of the PLL and SDLL clocks. The address base and clocks for the registers are shown in [Table 33: Clock generator registers on page 54](#). To program the divider to particular divide ratios use [Table 189](#). The SDLL_CLOCK channels are the only programmable dividers with phase shifting capabilities.

Table 189: Programmable divider register values

Divide ratio (Dec)	SDLL/CLKDIVn_CONFIG1 X[3:0] (Hex)	SDLL/CLKDIVn_CONFIG0 X[15:0] (Hex)	SDLL/CLKDIVn_DEPTH (Hex)	CLKDIVn_SEQUENCE X[19:0] (Hex)	Even (BIN)	HALF_NO T_ODD (BIN)
4	0x0	0xCCCC	0x05	0x0 CCCC	1	1
4.5	0x3	0x399C	0x07	0X3 399C	0	1
5	0x0	0x739C	0x04	0X0 739C	0	0
5.5	0x0	0x071C	0x00	0X0 071C	0	1
6	0x0	0x0E38	0x01	0X0 0E38	1	1
6.5	0x0	0x1C78	0x02	0X0 1C78	0	1
7	0x0	0x3C78	0x03	0X0 3C78	0	0
7.5	0x0	0x7878	0x04	0X0 7878	0	1
8	0x0	0xF0F0	0x05	0X0 F0F0	1	1
8.5	0x1	0xE1F0	0x06	0X1 E1F0	0	1
9	0x3	0xE1F0	0x07	0X3 E1F0	0	0
9.5	0x7	0xC1F0	0x08	0X7 C1F0	0	1
10	0xF	0x83E0	0x09	0XF 83E0	1	1
11	0x0	0x07E0	0x00	0X0 07E0	0	0
12	0x0	0x0FC0	0x01	0X0 0FC0	1	1
13	0x0	0x1FC0	0x02	0X0 1FC0	0	0
14	0x0	0x3F80	0x03	0X0 3F80	1	1
15	0x0	0x7F80	0x04	0X0 7F80	0	0
16	0x0	0xFF00	0x05	0X0 FF00	1	1
17	0x1	0xFF00	0x06	0X1 FF00	0	0
18	0x3	0xFE00	0x07	0X3 FE00	1	1
19	0x7	0xFE00	0x08	0X7 FE00	0	0
20	0xF	0xFC00	0x09	0XF FC00	1	1

SDLL/CLKDIVn_CONFIG0 CLOCK[n]/SDLL_CLOCK[n] configuration register 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X15	X14	X13	X12	X11	X10	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0

Address: *ClockgenBaseAddress* + 0x010, 0x020, 0x030, 0x040, 0x050, 0x060, 0x070, 0x080, 0x090, 0x0C0, 0x0D0, 0x0E0

Type: Read/write

Description: See [Chapter 52: Clock generator](#), [Table 189: Programmable divider register values](#).

SDLL/CLKDIVn_CONFIG1 CLOCK[n]/SDLL_CLOCK[n] configuration register 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												X19	X18	X17	X16

Address: *ClockgenBaseAddress* + 0x014, 0x024, 0x034, 0x044, 0x054, 0x064, 0x074, 0x084, 0x094, 0x0C4, 0x0D4, 0x0E4

Description: See [Chapter 52: Clock generator](#), [Table 189: Programmable divider register values](#).

SDLL/CLKDIVn_CONFIG2 CLOCK[n]/SDLL_CLOCK[n] configuration register 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									HALF_NOT_ODD	EVEN	SDLL/CLKDIVn_EN	SDLL/CLKDIVn_DEPTH			

Address: *ClockgenBaseAddress* + 0x018, 0x028, 0x038, 0x048, 0x058, 0x068, 0x078, 0x088, 0x098, 0x0C8, 0x0D8, 0x0E8

Description:

[15:7] **Reserved**

[6:5] **HALF_NOT_ODD, EVEN**

Used to set required divide ratio

0 EVEN 0 HALF_NOT_ODD: Divide by odd whole number, for example, 5

1 EVEN 1 HALF_NOT_ODD: Divide by even whole number, for example, 6

0 EVEN 1 HALF_NOT_ODD: Divide by half ratio, for example, 4.5

[4] **SDLL/CLKDIVn_EN**

Clock output enable, default = 0x01

0: Clock output disabled, at logic 0

1: Clock output enabled

[3:0] **SDLL/CLKDIVn_DEPTH**

The clock depth starts at 10 (depth bit pattern of 0000 = 10), so divider ratios below 10 have to be multiplied by 2, 3, 4 or 5 to get the depth >= 10

SDLL/CLKDIVn_SEQUENCE Sequence data

Sequence data, binary coded = SDLL/CLKDIVn_CONFIG1[3:0] + CLKDIVn_CONFIG0.

See [Table 189: Programmable divider register values](#) on page 566.

53.2 Shift register for SDLL_CLOCK[2:0]

The SDLL_CLOCK channels are the only programmable dividers with phase shifting capabilities. The following register bits SHIFT_ENn, SHIFT_UPn, SHIFT_DNn relate to the SDLL_CLOCK where n is the channel number.

SHIFT_CONFIG

Shift configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SHIFT_DN2	SHIFT_DN1	SHIFT_DN0	SHIFT_UP2	SHIFT_UP1	SHIFT_UP0	SHIFT_EN2	SHIFT_EN1	SHIFT_EN0

Address: *ClockgenBaseAddress + 0x0F0*

Type: Read/write

Reset: 0

Description: Shifting of the SDLL_CLOCK channels are controlled through software. The SDLL_CLOCK channels are as follows:

0: Not used,

1: Flash,

2: SDRAM

[15:9] **Reserved**

[8:6] **SHIFT_DN**

Read-only bit for software phase alignment of programmable divider

0: No shift right required

1: Shift right required

[5:3] **SHIFT_UP**

Read-only bit for software phase alignment of programmable divider

0: No shift left required

1: Shift left required

[2:0] **SHIFT_EN**

Software configurable bit to enable shifting of programmable divider

0: Software shifting enabled

1: Reserved. (Note: Hardware shifting is not longer supported)

53.3 Audio DAC, DSS smartcard, auxiliary clock registers

The registers are identical for PCMCLKIN, SMART_CARD_CLK and AUX_CLK_OUT. The address base for the registers are given at the beginning of this chapter.

SYNTHn_CONFIG0

Digital frequency synthesizer configuration register 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLL_DIV	PLL_SEL	OP_EN0	DISABLE0	MD0					SDIV0			NDIV0		

Address: *ClockgenBaseAddress* + 0x120, 0x130, 0x140

Type: Read/write

Reset: 0

Description: In register SYNTHnCONFIG0 the clock channels are as follows:
 0: PCMCLKIN,
 1: SMART_CARD_CLK,
 2: AUX_CLK_OUT.

[15:14] **Reserved**

[13] **PLL_DIV**

Loop divider for internal PLL, default configuration = 1

0: Multiply by 8

1: Multiply by 16 (normal operating mode)

SYNTH disable at logic 1 in test mode (assertion of TST_MODE pin)

[12] **PLL_SEL**

Set to 1 if no external PLL is used, default configuration = 1

[11] **OP_EN0**

Setting to 1 enables the output, otherwise output at logic 0

0: Output at logic 0

1: Out enabled

[10] **DISABLE0**

Setting the DISABLE to logic 1 shuts down the PLL

0: Normal mode

1: PLL shutdown

[9:5] **MD0[4:0]**

Coarse selector bus for phase taps selection

[4:2] **SDIV0[2:0]**

Output divider

[1:0] **NDIV0[1:0]**

Input divider

SYNTHn_CONFIG1**Digital frequency synthesizer configuration register 1**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE[15:0]															

Address: *ClockgenBaseAddress* + 0x124, 0x134, 0x144

Type: Read/write

Reset: 0

Description: In register SYNTHnCONFIG1 the clock channels are as follows:
 0: PCMCLKIN,
 1: SMART_CARD_CLK,
 2: AUX_CLK_OUT.

[15:0] **PE[15:0]**

Fine selector bus for phase taps selection

53.4 Low power mode registers**LPMODE0****LP_CLK configuration register 0**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPMARKCOUNT[7:0]								LPSPACECOUNT[7:0]							

Address: *ClockgenBaseAddress* + 0x20C

Type: Read/write

Reset: 0

Description:

[15:8] **LPMARKCOUNT[7:0]**

Count of clock periods during mark phase of LP_CLK
 Default configuration = 0x3F (63 dec)

[7:0] **LPSPACECOUNT[7:0]**

Count of clock periods during space phase of LP_CLK
 Default configuration = 0x3F (63 dec)

Confidential

LPMODE1

LP_CLK configuration register 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLFS_DIS	CLKOUT27_OP_DIS		SYNTH_CLOCK_OP_DIS						PLL_CLOCK_OP_DIS					INT_SEL

Address: *ClockgenBaseAddress* + 0x210

Type: Read/write

Reset: 0

Description:

[15] **Reserved**

[14] **PLLFS_DIS**

To disable the PLL_FS when low power mode is active, default value 1

0: PLL_FS is enabled during low power mode

1: PLL_FS is disabled during low power mode

[13] **CLKOUT27_OP_DIS**

Set to disable CLKOUT27 when low power mode is active, default value 1

0: CLKOUT27 is enabled during low power mode

1: CLKOUT27 is disabled during low power mode

[12:10] **SYNTH_CLOCK_OP_DIS[2:0]**

Set to disable an output clock when low power mode is active, default configuration = 0x7, that is, all clock outputs disabled when low power mode is active

[9:1] **PLL_CLOCK_OP_DIS[8:0]**

Set to disable an output clock when low power mode is active.

Default configuration = 0xFFF, (all clock outputs disabled when low power mode is active)

For example, PLL_CLOCK_OP_DIS[0] = 1, then PLL_CLOCK[0] output is disabled (at logic 0) when low power mode is active

PLL_CLOCK_OP_DIS[8] = 0, then PLL_CLOCK[8] output is enabled (at programmed frequency) when low power mode is active

[0] **INT_SEL**

Select internally generated low power clock (divide down of 27 MHz).

Default value = 0

0: EXT_LP_CLK used to source LP_CLK

1: Internally derived clock used to source LP_CLK

Confidential

LPMODE2

LP_CLK configuration register 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SDLL_CLOCK_OP_DIS		PLLCS_DIS	

Address: *ClockgenBaseAddress* + 0x214

Type: Read/write

Reset: 0

Description:

[15:4] **Reserved**

[3:1] **SDLL_CLOCK_OP_DIS[2:0]**

Set to disable an output clock when low power mode is active, default configuration = 0x7, that is, all clock outputs disabled when low power mode is active

[0] **PLLCS_DIS**

Always set to 1. Default value 1

0: Reserved

1: Always set to 1

53.5 CPU tick timer register

The **cpu_tick** function is generated using two timers which count the number of clock cycles for the mark and space phase of the **cpu_tick** period. The timers are programmable.

TICKTIMER

cpu_tick configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TICK_EN	TICKMARKCOUNT							TICKSPACECOUNT						

Address: *ClockgenBaseAddress* + 0x208

Type: Read/write

Reset: 0

Description:

[15] **Reserved**

[14] **TICK_EN**

cpu_tick output enable:

0: **cpu_tick** output disabled, at logic 0

1: **cpu_tick** enable

[13:7] **TICKMARKCOUNT[6:0]**

Count of clock periods during mark phase of **cpu_tick**

Default configuration = 0x0C (12 dec)

[6:0] **TICKSPACECOUNT[6:0]**

Count of clock periods during space phase of **cpu_tick**

Default configuration = 0x0D (13 dec)

54 Low power module (LPM) and power-down mode

54.1 Power-down mode

In power-down mode the internal clocks are turned off, the processor and all of the peripherals including the external memory controller are stopped. Effectively, the internal clock is stopped and functional operation is stalled.

On restart, the clock is restarted and the chip resumes normal operation. The PLL_FS is turned on and off using the LPMODE1 register and the PLLCS is turned on and off using the LPMODE2 register.

Provided that there are no active external interrupts, power-down is entered when low power LP_MODE signal is asserted.

Power-down is exited when an enabled external interrupt becomes active, when the LP_MODE signal is zero or when a wake up interrupt is triggered by the IR blaster.

Note: Please see [Chapter 53: Clock generator registers on page 562](#) for all LPM_MODE register details.

54.1.1 Entering and leaving power-down mode

The STi5516 enters power-down when the low power alarm counter is programmed and started, providing there are no external interrupts active.

The STi5516 exits power-down when:

- an enabled external interrupt becomes active,
- the low power alarm counter reaches zero.

Low power alarm

The low power alarm counter is a 40-bit counter which, when started, triggers off power-down mode. A write to the LPM_ALARMSTART register starts the low power alarm counter and the STi5516 enters low power mode. When the counter has counted down to zero and assuming no other valid wake up sources occur first, the STi5516 exits low power mode and the global clocks are turned back on.

54.2 Real-time counter

This timer keeps track of real time, even when the internal clocks are stopped. The timer is a 64-bit counter, powered from a separate VDD (RTCVDD) allowing it to operate even when the rest of the chip is not powered.

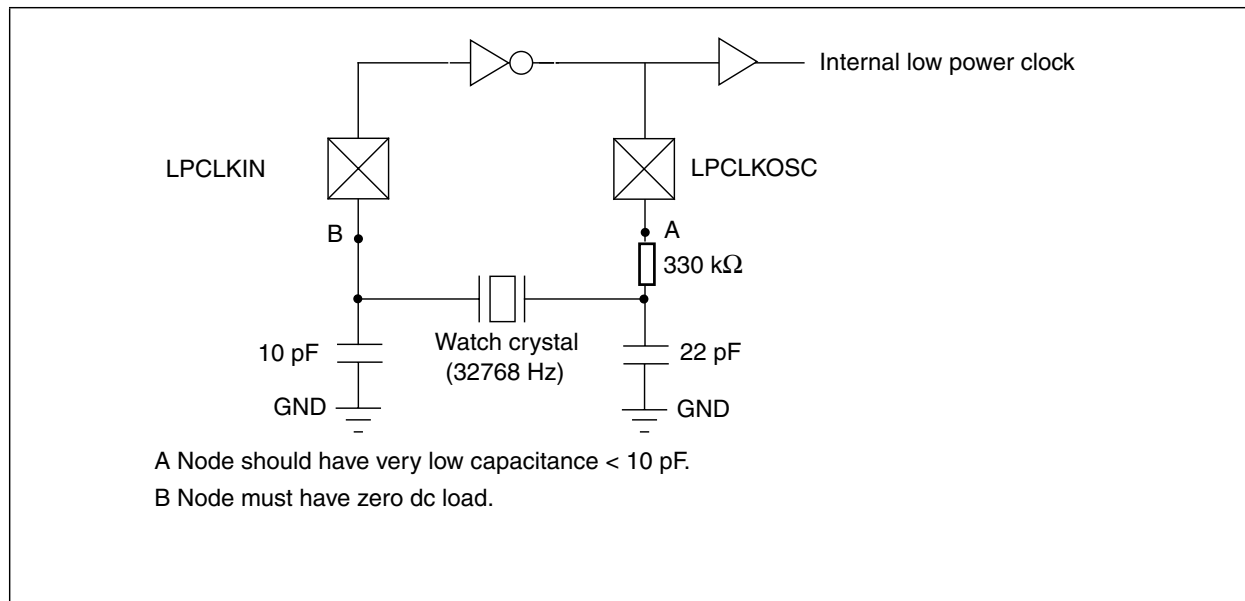
The timer must be clocked at all times by one of the following clocking sources even if the real-time counter or other low power features are not being used, otherwise the CPU might fail to boot.

- An external clock input (LPCLKIN): the specification for this clock should be as follows:
 - minimum frequency of 200 Hz
 - maximum frequency of 50 kHz,
 - maximum rise and fall times of 100 ns,
 - minimum mark/space of 40/60,
 - maximum mark/space of 60/40.

In this case the LPCLKOSC pin should not be connected on the board.

- A watch crystal, in the circuit shown in [Figure 189](#).

Figure 189: Watch crystal clocking source



54.3 Watchdog counter

The low power alarm counter can be used as a watchdog timer if bit 0 of the register LPM_WDENABLE is set. Setting bit 0 of LPM_WDENABLE disables the entering of low power mode when starting the low power alarm counter.

To trigger off watchdog functionality, the low power alarm is programmed and started as normal. When the low power alarm counts down to 1, the circuit resets. The LPM_WDFLAG register is set when a watchdog reset occurs.

55 Low power module (LPM) registers

Addresses are provided as the *LPCBaseAddress* + offset.

The *LPCBaseAddress* is:

0x2010 0000.

A register summary is given in [Table 43: Low power module \(LPM\) registers on page 69](#).

LPM_TIMER

Low power timer

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x400	LPTIMER[31:0]																															
0x404	LPTIMER[63:32]																															

Address: *LPCBaseAddress* + 0x400 and 0x404

Type: Read/write

Reset: Undefined

Description: The LPM_TIMER registers are the least significant and most significant words of the low power timer register. These enable the least significant or most significant word to be written independently without affecting other words. When either word of the register is written, the low power timer is stopped and the new value in LPM_TIMER is available to be written to the low power timer.

LPM_TIMERSTART

Low power timer start

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

Address: *LPCBaseAddress* + 0x408

Type: Write only

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

LPM_ALARM

Low power alarm

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x410	LPALARM[31:0]																															
0x414	Reserved																								LPALARM[39:32]							

Address: *LPCBaseAddress* + 0x410 and 0x414

Type: Read/write

Reset: Undefined

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

LPM_ALARMSTART**Low power alarm start**

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

Reserved																ALARMSTART	

Address: *LPCBaseAddress* + 0x418

Type: Write only

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

LPM_WDENABLE**Watchdog 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																ENABLE	

Address: *LPCBaseAddress* + 0x510

Type: Read/write

Reset: Undefined

Description: Setting the LPM_WDENABLE register enables the low power alarm counter to be used as a watchdog timer.

[31:1] **Reserved**[0] **ENABLE**

0: Alarm

1: Watchdog

LPM_WDFLAG**Watchdog flag**

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

Reserved																WDFLAG	

Address: *LPCBaseAddress* + 0x514

Type: Read only

Reset: Undefined

Description: The LPM_WDFLAG register is set when a watchdog reset occurs.

[31:1] **Reserved**[0] **WDFLAG**

0: No watchdog reset has occurred.

1: A watchdog reset has occurred.

Confidential

56 PWM and counter module

This module provides four PWM decoder (capture) inputs and four programmable timers. Each capture input can be programmed to detect rising edge, falling edge, both edges or neither edge (disabled). These facilities are clocked by two independent clocks, one for PWM outputs and one for capture inputs or timers.

In the STi5516, not all the facilities in the module can be used, since some of the interface pins of the module are not available as external pins of this device. In particular, the following interface pins are not available:

- PWM_OUT3
- PWM_CAPTURE1 and PWM_CAPTURE3
- PWM_COMPARE1 and PWM_COMPARE3

However the corresponding COMPAREOUT facilities can be used to generate interrupts at programmable time intervals. Otherwise the corresponding registers should not be used.

The module is programmed by means of registers described in the individual sections.

The module generates a single interrupt signal. The exact event which caused an interrupt can be determined by reading the status bits in a register, which can then be cleared.

56.1 External interface

Table 190: PWM and counter pins

Name	Function name	In/out	Function
PIO2[7]	PWM_OUT0	Out	PWM outputs
PIO3[7]	PWM_OUT1		
PIO4[7]	PWM_OUT2		
PIO2[5]	PWM_CAPTURE0	In	Capture trigger inputs
PIO4[5]	PWM_CAPTURE2		
PIO2[6]	PWM_COMPARE0	Out	Compare output
PIO4[6]	PWM_COMPARE2		

56.2 PWM outputs

There are four PWM outputs which share a common counter. The relative width (in counts) of the output pulse on pin PWM_OUTn is set between 1 and 256 by loading a value from 0 to 255 into the register PWM_nVAL. The width cannot be less than 1, and if it is 256 the pin is continuously high. Pulses occur every 256 counts.

The counter is clocked by the 27 MHz clock CLOCKIN divided by a prescaler. The prescaling factor, and therefore the period represented by one count, is determined by the value of field PWMCLKVALUE in register PWM_CONTROL. The factor can be from 1 to 16.

The counter (in register PWM_COUNT) is enabled by setting bit PWMENABLE of register PWM_CONTROL to 1. When it is disabled (PWMENABLE is 0), PWMOUTn is forced low. PWM_COUNT is writable at any time but can have a synchronization latency.

When the PWM counter overflows, an interrupt is generated if bit INTEN of register PWM_INTENABLE is set to 1. Bit INT of register PWM_INTSTATUS becomes 1, and can be reset by writing 1 to bit INTACK of register PWM_INTACK.

56.3 Capture inputs

There are four capture inputs which share a common counter with four compare facilities.

What constitutes an event on input PWM_CAPTUREn is defined by the code in register PWM_nCAPTUREEDGE. Possible events are rising edge, falling edge, both or neither (in other words, disabled).

When an input event occurs on input PWM_CAPTUREn, the value of the counter (in register PWM_CAPTURECOUNT) at that time is captured in register PWM_nCAPTUREVAL. The value can be 0x0000 0000 to 0xFFFF FFFF.

When an input event occurs, an interrupt is generated provided the CPTIEn bit of the PWM_INTENABLE register is set to 1. Bit CPTn of register PWM_INTSTATUS becomes 1, and can be reset by writing 1 to bit CPTIAN of register PWM_INTACK.

The counter is not stopped nor reset by any of these events. See [Section 56.5](#) for details.

56.4 Compare (programmable timer) facilities

There are four programmable timer facilities which share a common counter with four capture inputs. Each of the four compare registers PWM_nCOMPAREVAL in the module can be set to a value 0x0000 0000 to 0xFFFF FFFF.

When the counter in register PWM_CAPTURECOUNT reaches the value of register PWM_nCOMPAREVAL, two things happen:

- An interrupt is generated provided the CMPIEn bit of the PWM_INTENABLE register is set to 1. Bit CMPn of register PWM_INTSTATUS becomes 1, and can be reset by writing 1 to bit CMPIAn of register PWM_INTACK.
- Pin PWM_COMPAREn takes on the value set in register PWM_nCOMPAREOUTVAL.

The counter is not stopped nor reset by any of these events. See [Section 56.5](#) for details of the counter.

56.5 Capture/compare counter, prescaling and clocking

The capture/compare counter is clocked from the prescaled comms clock, and is common to all capture and compare functions. The prescaling factor, and therefore the period represented by one count, is determined by the value of field CAPTURECLKVALUE in register PWM_CONTROL. The factor can be from 1 to 32.

The counter (in register PWM_CAPTURECOUNT) is enabled by setting the CAPTUREENABLE bit of the PWM_CONTROL register to 1. When it is disabled (CAPTUREENABLE is 0), none of the capture or compare functions work. PWM_CAPTURECOUNT, like PWM_COUNT, can be read or written at any time.

When the capture/compare counter reaches its maximum count of 0xFFFF FFFF, it wraps round to count up from zero again.

57 PWM and counter module registers

Addresses are provided as the *PWMBaseAddress* + offset.

The *PWMBaseAddress* is:

0x2010 B000.

A register summary is given in [Table 50: PWM and counter module registers on page 77](#).

PWM_nCAPTUREEDGE

PWM n Capture event definition

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	Reserved																PWM_0CAPTUREEDGE															
0x34	Reserved																PWM_1CAPTUREEDGE															
0x38	Reserved																PWM_2CAPTUREEDGE															
0x3C	Reserved																PWM_3CAPTUREEDGE															

Address: *PWMBaseAddress* + 0x30 (PWM_0CAPTUREEDGE) to 0x3C (PWM_3CAPTUREEDGE)

Type: Read/write

Reset: Undefined

Description: The code in register PWM_nCAPTUREEDGE defines what constitutes an event on input pin PWM_CAPTUREn. Possible events are rising edge, falling edge, both or neither (in other words, disabled). Pin PWM_CAPTURE3 does not exist.

[31:2] **Reserved**

[1:0] **PWM_nCAPTUREEDGE**

01: Capture on rising edge

11: Capture on rising or falling edge

10: Capture on falling edge

00: Capture disabled

PWM_nCAPTUREVAL**PWM n capture value**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	PWM0CAPTUREVAL																															
0x14	PWM1CAPTUREVAL																															
0x18	PWM2CAPTUREVAL																															
0x1C	PWM3CAPTUREVAL																															

Address: *PWMBaseAddress* + 0x10 (PWM_0CAPTUREVAL) to 0x1C (PWM_3CAPTUREVAL)

Type: Read only

Reset: Undefined

Description: Each of the four capture value registers holds the 32-bit counter value at the time of the last event occurring at the corresponding PWM_CAPTUREn pin. When an input event occurs on one of the input pins PWM_CAPTUREn, the value of the counter in register PWM_CAPTURECOUNT at that time is captured in register PWM_nCAPTUREVAL. The value can be any 32-bit value.

The pin PWM_CAPTURE3, corresponding to register PWM_3CAPTUREVAL, does not exist.

When an input event occurs, an interrupt is generated if the register bit CPTIEn (PWM_INTENABLE) is set to 1. Register bit CPTn (PWM_INTSTATUS) becomes 1, and can be reset by writing 1 to register CPTIA n (PWM_INTACK).

The counter is not stopped or reset by any of these events.

Confidential

PWM_nCOMPAREOUTVAL**PWM n compare output value**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	Reserved																COMPAREOUTVAL0															
0x44	Reserved																COMPAREOUTVAL1															
0x48	Reserved																Reserved															
0x4C	Reserved																Reserved															

Address: *PWMBaseAddress* + 0x40 to 0x4C

Type: Read/write

Reset: Undefined

Description: This register holds the value which is written to the PWM_COMPAREn pin when the compare value in register PWM_nCOMPAREVAL matches the counter value PWM_CAPTURECOUNT. Since there are only two PWM_COMPAREn pins (0 and 2), they are reserved for the bit 0 of the registers 0 and 1 (COMPAREOUTVAL0 and COMPAREOUTVAL1) only.

PWM_nCOMPAREVAL**PWM n compare value**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	PWM0COMPAREVAL																															
0x24	PWM1COMPAREVAL																															
0x28	PWM2COMPAREVAL																															
0x2C	PWM3COMPAREVAL																															

Address: *PWMBaseAddress* + 0x20 (PWM0COMPAREVAL) to 0x2C (PWM3COMPAREVAL)

Type: Read/write

Reset: Undefined

Description: Each of the four compare registers PWM_nCOMPAREVAL in the module can be set to any 32-bit value. When the counter in register PWM_CAPTURECOUNT reaches the value of register PWM_nCOMPAREVAL, two things happen.

- An interrupt is generated if the register bit CMPIEn (PWM_INTENABLE) is set to 1. Register bit CMPn (PWM_INTSTATUS) becomes 1, and can be reset by writing 1 to register bit CMPIAn (PWM_INTACK).
- Pin PWM_COMPAREn takes on the value set in register PWM_nCOMPAREOUTVAL.

The counter is not stopped or reset by any of these events.

PWM_nVAL**PWM n pulse 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
0x00	Reserved																							PWM0VAL								
0x04	Reserved																							PWM1VAL								
0x08	Reserved																							PWM2VAL								
0x0C	Reserved																							PWM3VAL								

Address: *PWMBaseAddress* + 0x00 (PWM_0VAL) to 0x0C (PWM_3VAL)

Type: Read/write

Reset: Undefined

Description: These registers hold the counter values, which are used to determine the width of the pulse generated on the output pins PWM_OUTn. PWMn pulse width = (PWMnVAL + 1) x prescaled clock period.
If PWMnVAL is 255 then PWMn is always 1 (that is, it does not go low). PWM_3VAL has no associated output pin.

PWM_CAPTURECOUNT**PWM capture/compare counter**

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

Address: *PWMBaseAddress* + 0x64

Type: Read/write

Reset: Undefined

Description: This register holds the shared capture/compare counter used by all the capture and compare functions.

The capture/compare counter is clocked from the prescaled comms clock. The prescaling factor, and therefore the period represented by one count, is determined by the value of field CAPTURECLKVALUE in register PWM_CONTROL. The factor can be from 1 to 32.

The counter is enabled by setting register bit CAPTUREENABLE (PWM_CONTROL) to 1. When it is disabled (CAPTUREENABLE = 0), none of the capture or compare functions work. PWM_CAPTURECOUNT can be read or written at any time.

When the capture/compare counter reaches its maximum count of 0xFFFF FFFF, it wraps round to count up from zero again.

PWM_CONTROL**PWM 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

Reserved											CAPTUREENABLE	PWMENABLE	CAPTURECLKVALUE			PWMCLKVALUE			

Address: *PWMBaseAddress* + 0x50

Type: Read/write

Reset: Undefined

Description:

[31:11] **Reserved**[10] **CAPTUREENABLE**

Enables capture/compare counter when 1

[9] **PWMENABLE**

Enables PWM counter when 1

[8:4] **CAPTURECLKVALUE**

Capture/compare clock prescale factor 0 to 31 (divide clock by value + 1)

[3:0] **PWMCLKVALUE**

PWM clock prescale factor 0 to 15 (divide clock by value + 1)

PWM_COUNT**PWM output counter**

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

Reserved											COUNT									

Address: *PWMBaseAddress* + 0x60

Type: Read/write (but see text)

Reset: Undefined

Description: PWM output counter. The counter (in register PWM_COUNT) is enabled by setting register bit PWMENABLE (PWM_CONTROL) to 1. When it is disabled (PWMENABLE = 0), pins PWM_OUT[2:0] are forced low. PWM_COUNT is writable at any time but can have a synchronization latency.

PWM_INTACK**PWM interrupt acknowledge**

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

Reserved																														
																CMPIA3	CMPIA2	CMPIA1	CMPIA0	CPTIA3	CPTIA2	CPTIA1	CPTIA0	INTACK						

Address: *PWMBaseAddress* + 0x5C

Type: Write only

Description:

[31:9] **Reserved**

[8] **CMPIA3**

Compare 3 interrupt acknowledge: Write 1 to reset status bit

[7] **CMPIA2**

Compare 2 interrupt acknowledge: Write 1 to reset status bit

[6] **CMPIA1**

Compare 1 interrupt acknowledge: Write 1 to reset status bit

[5] **CMPIA0**

Compare 0 interrupt acknowledge: Write 1 to reset status bit

[4] **CPTIA3**

Capture 3 interrupt acknowledge: Write 1 to reset status bit

[3] **CPTIA2**

Capture 2 interrupt acknowledge: Write 1 to reset status bit

[2] **CPTIA1**

Capture 1 interrupt acknowledge: Write 1 to reset status bit

[1] **CPTIA0**

Capture 0 interrupt acknowledge: Write 1 to reset status bit.

[0] **INTACK**

Interrupt acknowledge: Write 1 to reset PWMINT to 0

PWM_INTENABLE

PWM interrupt enable

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

Reserved	CMPIE3
	CMPIE2
	CMPIE1
	CMPIE0
	CPTIE3
	CPTIE2
	CPTIE1
	CPTIE0
	INTEN

Address: *PWMBaseAddress* + 0x54

Type: Read/write

Reset: Undefined

Description:

[31:9] **Reserved**

[8] **CMPIE3**

Compare 3 interrupt enable

1: Enabled

[7] **CMPIE2**

Compare 2 interrupt enable

1: Enabled

[6] **CMPIE1**

Compare 1 interrupt enable

1: Enabled

[5] **CMPIE0**

Compare 0 interrupt enable

1: Enabled

[4] **CPTIE3**

Capture 3 interrupt enable

1: Enabled

[3] **CPTIE2**

Capture 2 interrupt enable

1: Enabled

[2] **CPTIE1**

Capture 1 interrupt enable

1: Enabled

[1] **CPTIE0**

Capture 0 interrupt enable

1: Enabled

[0] **INTEN**

PWM counter overflow interrupt enable

Confidential

PWM_INTSTATUS

PWM interrupt status

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

Reserved																CMP3		CMP2	CMP1	CMP0	CPT3	CPT2	CPT1	CPT0	INT

Address: *PWMBaseAddress* + 0x58

Type: Read only

Reset: Undefined

Description:

[31:9] **Reserved**

[8] **CMP3**

Compare 3 interrupt
1: Interrupt

[7] **CMP2**

Compare 2 interrupt
1: Interrupt

[6] **CMP1**

Compare 1 interrupt
1: Interrupt

[5] **CMP0**

Compare 0 interrupt
1: Interrupt

[4] **CPT3**

Capture 3 interrupt
1: Interrupt

[3] **CPT2**

Capture 2 interrupt
1: Interrupt

[2] **CPT1**

Capture 1 interrupt
1: Interrupt

[1] **CPT0**

Capture 0 interrupt
1: Interrupt

[0] **INT**

1: PWM counter overflow

58 Modem analog front-end interface (MAFEIF)

58.1 Overview

The modem analog front end interface (MAFEIF) is an integrated interface to an analog front end (AFE) for a modem such as the STLC7550.

In this chapter, the term sample is a 16-bit data object that is transferred to or from the modem through the MAFEIF, and the term sample period is the time from the start of one sample to the start of the next.

The MAFEIF simultaneously transmits samples into and out of the AFE. It typically operates at a rate of 9600 samples/second, giving a typical sample period of 100 μ s. That is, every 100 μ s, one sample is transmitted and another received through the MAFEIF.

The MAFEIF receives its system clock signal (SCLK) from the AFE. The SCLK frequency is typically 256 ticks/sample period, or 2.56 MHz. The first 16 ticks of the 256 tick sample period are used to exchange a 16-bit sample pair (1 bit per tick).

The MAFEIF uses one DMA to transfer samples from a transmit memory buffer to the AFE, and simultaneously uses a second DMA to receive samples from the AFE and write them into the receive memory buffer. The software driver is woken up every time a simultaneous transfer is completed, that is, every time a transmit memory buffer has been emptied and a receive memory buffer has been filled. For example, if each memory buffer contains 100 samples, the software is woken up every 10 ms (100 x 100 μ s). This is more stringent for handshake signals, where the buffer size could be as low as a few samples, for example, 4.

The software modem has two pairs of pointers (that is, four pointers) that point to two pairs of transmit/receive buffers. The modem and the MAFEIF alternately switch between the two pairs of pointers. While the MAFEIF transmits and receives using one pair of buffers, the software modem processes the information in the other pair. Using the above example for a buffer containing 100 samples, the software has 10 ms to wake up and then process one pair of transmit/receive buffers before they are required again by the MAFEIF.

58.2 Using the MAFEIF to connect to a modem

The following table lists the pins that are used by the MAFEIF to connect a modem:

Table 191: MAFEIF pins

Name	Type	MAFEIF function name (alternate)	MAFEIF function description
PIO2[0]	O	MAFE_HC1	Indicates to the AFE that a control/status exchange is to take place.
PIO2[1]	O	MAFE_DOUT	Line for serially transmitting samples to the AFE.
PIO2[2]	I	MAFE_DIN	Line for serially receiving samples from the AFE.
PIO2[3]	I	MAFE_FS	Signal from the AFE indicating the start of a sampling period. This is latched on falling edges of SCLK. For normal operation it should not remain high for more than 16 SCLK cycles, and there should be at least 20 SCLK ticks between consecutive rising edges of Fs.
PIO2[4]	I	MAFE_SCLK	Modem system clock. The frequency should be less than half of the device system clock.

58.3 Software

The MAFEIF software manages the data exchange between the software modem and MAFEIF, and handles the control/status exchange.

58.3.1 Data exchange

When the MAFEIF exchanges data, the software:

1. disables all interrupts,
2. sets the buffer size, for example, 100 samples (for handshake response times, the buffer size could be as low as a few samples, for example 4),
3. sets up both pairs of memory pointers in the MAFEIF (this is probably not changed again),
4. enables status (complete) interrupt,
5. sets the control (run) bit,
6. deschedules.

The MAFEIF then processes a buffer load of samples (that is, it transmits 100 samples and receives 100 samples). When this is complete, the MAFEIF sets the status (complete) bit, causing the software to be woken up. The software then:

7. processes the receive memory buffer and fills the next transmit memory,
8. confirms that there has been no overflow (that is, failure to finish the software processing of a buffer before that buffer has started to be overwritten again),
9. confirms that there have been no memory latency problems during the exchange of the previous buffer, by reading the status (missed) bit,
10. if there are no problems, the software writes to the MOD_ACK register and deschedules.

58.3.2 Control/status exchange

For a control/status exchange, the software writes to the MOD_CONTROL register to enable the status interrupt (CTRL_EMPTY), and then deschedules.

When the software wakes up, it reads the modem status and disables the status interrupt (CTRL_EMPTY) again.

59 Modem analog front-end interface (MAFEIF) registers

Addresses are provided as the *ModemBaseAddress* + offset.

The *ModemBaseAddress* is:

0x2011 3000.

A register summary is given in [Table 45: Modem analog front-end interface \(MAFEIF\) registers on page 70](#).

MOD_CONTROL

MAFEIF control

7	6	5	4	3	2	1	0
Reserved						START	RUN

Address: *ModemBaseAddress* + 0x00

Type: Read/write

Reset: Undefined

Description:

[7:2] **Reserved**

[1] **START**

Indicates which of the two pairs of memory buffer pointers it should start off using:

0: Indicates RECEIVE0_POINTER and TRANSMIT0_POINTER

1: Indicates RECEIVE1_POINTER and TRANSMIT1_POINTER

[0] **RUN**

1: The MAFEIF is to start exchanging data with the AFE.

0: The MAFEIF stops after completing the exchange of the current buffer load of samples.

MOD_STATUS

MAFEIF status

7	6	5	4	3	2	1	0
Reserved		MISSED	OVERFLOW	LAST	CTRL_EMPTY	COMPLETE	IDLE

Address: *ModemBaseAddress* + 0x04

Type: Read only

Reset: Undefined

Description:

[7:6] **Reserved**

[5] **MISSED**: 1: Memory latency is too high, causing a sample to be missed. The MAFEIF is exchanging samples faster than being read from/written to the memory buffers. Cleared by writing to MOD_ACK.

[4] **OVERFLOW**: 1: Overflow has occurred. The MAFEIF has completed the exchange of a buffer load of samples before software has acknowledged the previous buffer load. Cleared by writing to MOD_ACK.

[3] **LAST**: Indicates the last pair of buffer pointers used by the DMA

[2] **CTRL_EMPTY**

Set to 0 by writing to MOD_MAFE_CTRL. Set to 1 when the MAFEIF completes control/status exchange.

[1] **COMPLETE**

Set to 1 when a buffer load of samples has been exchanged. Cleared by writing to MOD_ACK.

[0] **IDLE**

0: MAFEIF is exchanging data with the AFE.

1: RUN bit low. MAFEIF is not exchanging data.

After software clears RUN, MAFEIF goes idle after exchanging the current buffer load of samples.

MOD_INT_ENABLE **Interrupt enable**

7	6	5	4	3	2	1	0
Reserved					INTENABLE2	INTENABLE1	INTENABLE0

Address: *ModemBaseAddress* + 0x08

Type: Read/write

Reset: Undefined

Description:

[7:3] **Reserved**[2:0] **INTENABLE[2:0]**

Enables interrupts connected to MOD_MAFE_STATUS[2:0].

1: Indicates that the corresponding interrupt is enabled.

MOD_ACK **Acknowledge**

7	6	5	4	3	2	1	0
ACK							

Address: *ModemBaseAddress* + 0x0C

Type: Write only

Reset: Undefined

Description:

[7:0] **ACK**

Clears the overflow, missed and complete flags in the MOD_MAFE_STATUS register.

MOD_BUFFER_SIZE **Buffer size**

7	6	5	4	3	2	1	0
SIZE							Reserved

Address: *ModemBaseAddress* + 0x10

Type: Read/write

Reset: Undefined

Description:

[7:1] **SIZE**

This value must be a multiple of two, it sets the buffer size (the number of 16-bit samples in a buffer).

[0] **Reserved****MOD_MAFE_CTRL** **MAFEIF control**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTVAL															

Address: *ModemBaseAddress* + 0x14

Type: Write only

Reset: Undefined

Description:

[15:0] **CNTVAL**

This number is the control value to send out to the MAFEIF.

MOD_MAFE_STATUS**MAFEIF status**

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

STATUS

Address: *ModemBaseAddress* + 0x18

Type: Read only

Reset: Undefined

Description:

[15:0] **STATUS**

This number is the status value received from the MAFEIF.

MOD_RECEIVE0_POINTER**receive_memory_buffer_0 start 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

ADDR

Res

Address: *ModemBaseAddress* + 0x20

Type: Read/write

Reset: Undefined

Description:

[31:2] **ADDR**

This number is the start address of the receive_memory_buffer_0.

[1:0] **Reserved**

MOD_RECEIVE1_POINTER**receive_memory_buffer_1 start 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

ADDR

Res

Address: *ModemBaseAddress* + 0x24

Type: Read/write

Reset: Undefined

Description:

[31:2] **ADDR**

This number is the start address of the receive_memory_buffer_1.

[1:0] **Reserved**

MOD_TRANSMIT0_POINTER**transmit_memory_buffer_0 start 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

ADDR

Res

Address: *ModemBaseAddress* + 0x28

Type: Read/write

Reset: Undefined

Description:

[31:2] **ADDR**

This number is the start address of the transmit_memory_buffer_0.

[1:0] **Reserved**

Confidential

MOD_TRANSMIT1_POINTER transmit_memory_buffer_1 start 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
ADDR																															Res

Address: *ModemBaseAddress* + 0x2C

Type: Read/write

Reset: Undefined

Description:

- [31:2] **ADDR**
This number is the start address of the transmit_memory_buffer_1.
- [1:0] **Reserved**

Confidential

ST Confidential (iPanel)



60 Infrared transmitter/receiver

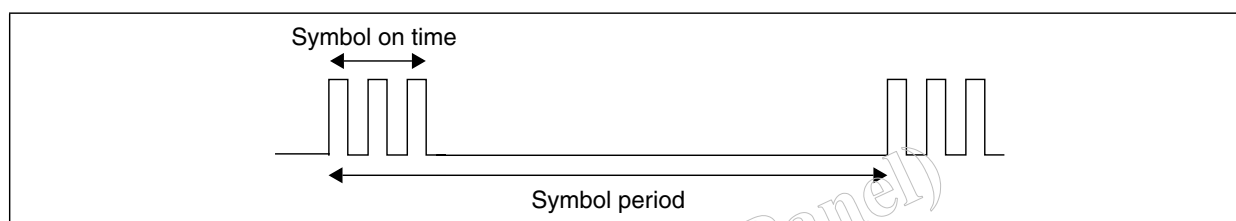
60.1 Overview

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 4-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 190](#).

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 190: IR transmitter/receiver symbol



60.2 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 191](#) shows the IR transmitter/receiver block diagram in a typical circuit configuration with input demodulating and output buffering (open drain).

In the transmitter there are two programmable dividers to generate the prescaled clock and the subcarrier clock. The subcarrier clock sets the resolution for the transmitted data. Both receivers contain a sampling rate clock, which samples the incoming data, and is programmed to 10 MHz.

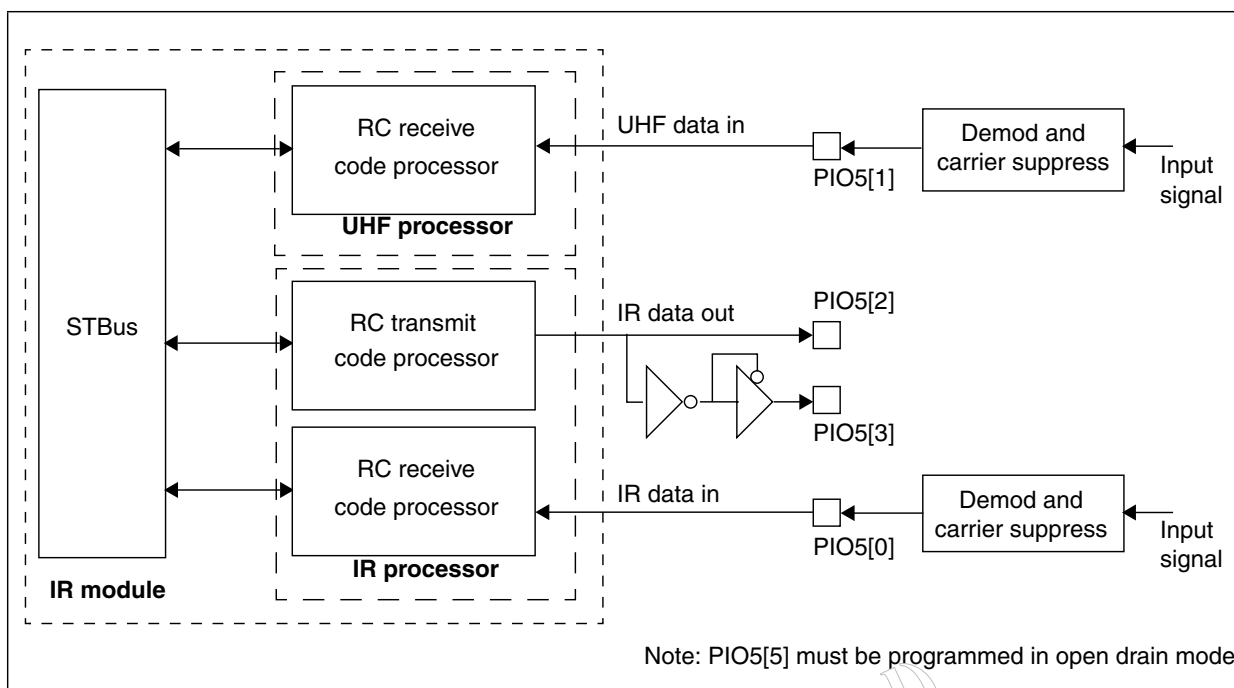
FIFOs buffer both the transmitter output and the receivers' inputs to avoid timing problems with the CPU. Interrupts can be set on the FIFOs' levels to prevent input data overrun and output data underrun.

The two receivers each have one input pin, and the transmitter has two output pins (one driven directly and the other inverted as open drain).

There are two 4-word FIFOs in the RC transmitter and two in each RC receiver. The fourth element in each 4-word FIFO is used internally and is not accessible to the STBus. Therefore, the 4-word FIFO is empty when there are three empty words and full when it contains three words. At all times, the fullness level of the 4-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 191: IR transmitter/receiver block diagram and implementation



60.2.1 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 three words are empty (buffer is empty),
- when two words are empty (buffer is half full),
- when at least one word is empty.

The transmit interrupt is cleared automatically when new data is written to the registers `IRB_TX_SYM_PERIOD_IR` and `IRB_TX_ON_TIME_IR`. Register bits `IRB_TX_INT_STATUS_IR[5:4]` give the FIFO's fullness status.

The frequency of the subcarrier is set by programming the registers `IRB_TX_PRE_SCALER_IR` and `IRB_TX_SUB_CARRIER_IR`.

The symbol period, in subcarrier cycles, is programmed in the register `IRB_TX_SYM_PERIOD_IR` and the on time of the IR pulse is written to the register `IRB_TX_ON_TIME_IR`. These two registers are four-word FIFOs. They must be programmed sequentially as a pair to increment the write pointer and be ready for the next data. Transmission is enabled by setting register `IRB_TX_EN_IR` bit 0 to 1. If new data is not written before the last symbol in the buffer is transmitted, no RC codes are generated. The output is driven to logic 0 and the register `IRB_TX_INT_STATUS_IR` bit 1 is set.

Before data can be transmitted, the underrun condition must be cleared as in the procedure below.

1. Disable the transmission by writing 0 to register `IRB_TX_EN_IR`.
2. Load at least one block of data into `IRB_TX_SYM_PERIOD_IR` and `IRB_TX_ON_TIME_IR`.
3. Clear the `TX_UNDERRUN` status bit by writing 1 to register `IRB_TX_CLR_UNDERRUN_IR`.

Transmission is resumed by writing 1 to register `IRB_TX_EN_IR`.

60.2.2 RC receive code processor

This section describes the UHF data and the IR data receivers. They are independent and identical except that the noise suppression filter is programmable in the UHF receiver, and is not used in the IR receiver. The 10 MHz sampling clock is common to both receivers and is set by register IRB_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 a four-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,
- two or more words are available to be read (FIFO half full),
- three words are available to be read (FIFO full).

The interrupt is cleared when the registers IRB_RX_SYM_PERIOD and IRB_RX_ON_TIME have been read. They must be read consecutively, as a pair, to increment the FIFO read pointer. Bits 4 and 5 of the register IRB_RX_INT_STATUS give the fullness level of the FIFO.

If the FIFO is full and has not been read before the arrival of new data, then this data is lost and a receive overrun flag is set in the status register IRB_RX_INT_STATUS. No new data is written to the FIFO while this condition exists. To reset the overrun flag the operations below must be performed.

1. Read at least one word from each of the receive FIFO registers, IRB_RX_SYM_PERIOD and IRB_RX_ON_TIME.
2. Clear the RXOVERRUNSTATUS bit by writing 0x01 to register IRB_RX_CLR_OVERRUN.

The last symbol is detected using a time out condition whose value is stored in microseconds in register IRB_RX_MAX_SYM_PERIOD. If no pulse has been received during this time then the last word in the FIFO IRB_RX_SYM_PERIOD has a value 0xFFFF. If the value of register IRB_RX_INT_EN bit 1 (LASTSYMBOLIRQENABLE bit) is 1, then an interrupt is triggered and the status register IRB_RX_INT_STATUS bit 1 is set. The interrupt and its status bit are cleared automatically when the last value in the FIFO has been read.

When register IRB_RX_INT_EN bit 0 is set to 0 then both the FIFO level interrupt and the last symbol interrupt are inhibited.

RC data reception can be disabled by setting register IRB_RX_EN bit 0 to 0. However, both receivers are normally always enabled.

60.2.3 Noise suppression filter

This filter is turned off in the IR receiver and is programmable in the UHF receiver using register IRB_RX_NOISE_SUPPRESS_WIDTH_UHF. Any pulses, either high or low, having a value in microseconds of less than the programmed width, are assumed to be noise and, therefore, suppressed.

The noise suppression filter can be disabled by writing 0x00 to register IRB_RX_NOISE_SUPPRESS_WIDTH_UHF.

61 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:

0x2011 5000.

A register summary is given in [Table 41: Infrared transmitter/receiver registers on page 66](#).

61.1 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 quadruple buffered.

IRB_TX_ON_TIME_IR Symbol ON time programming

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

TXONTIMEVAL

Address: *IRBBaseAddress + 0x0C*

Type: 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 quadruple buffered.

Note: *The registers IRB_TX_SYM_PERIOD_IR and IRB_TX_ON_TIME_IR act as a single register set. They must be programmed sequentially as a pair to latch in the data.*

IRB_TX_INT_EN_IR Transmit interrupt enable register

7 6 5 4 3 2 1 0

Reserved	TXFIFOIRQ	Reserved	TXIRQENABLE

Address: *IRBBaseAddress + 0x10*

Type: Read/write

Reset: 0x00

Description:

[7:6] **Reserved**

Set to logic 0

[5:4] **TXFIFOIRQ[1:0]**

Select the transmit FIFO fullness interrupt:

00: Invalid

01: One word available for read

10: Two words available for read (half full)

11: Three words available for read (FIFO full)

[3:1] **Reserved**

Set to logic 0

[0] **TXIRQENABLE**

Select the transmit interrupt enable/ disable:

0: Interrupt disable

1: Interrupt enable

Confidential

IRB_TX_INT_STATUS_IR Transmit Interrupt status register

7	6	5	4	3	2	1	0
Reserved		TXFIFOSTATUS		Reserved		TXUNDERRUN_STATUS	TXIRQSTATUS

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.

[7:6] **Reserved**

Set to logic 0

[5:4] **TXFIFOSTATUS[1:0]**

Transmit FIFO fullness status:

00: FIFO empty

01: One block full

10: Two blocks full

11: FIFO full

[3:2] **Reserved**

Set to logic 0

[1] **TXUNDERRUN_STATUS**

Transmit underrun status:

0: No under run

1: Under run occurred

[0] **TXIRQSTATUS**

Transmit interrupt status:

1: Interrupt enabled

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_CLR_UNDERRUN_IR Clears the underrun status

7	6	5	4	3	2	1	0
Reserved							CLRUNDERRUN

Address: *IRBBaseAddress* + 0x1C

Type: Write only

Reset: 0x00

Description: This register must be set to 1 as part of the procedure for clearing the underrun flag in the register IRB_TX_INT_STATUS_IR. No data is transmitted until this flag has been cleared.

IRB_TX_SUB_CARRIER_WIDTH_IR Subcarrier frequency programming

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 x 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.

61.2 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 quadruple buffered.

IRB_RX_SYM_PERIOD Received symbol period capture

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSYMBOLTIMEVAL															

Address: *IRBBaseAddress* + 0x44 (IR) *IRBBaseAddress* + 0x84 (UHF)

Type: 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 quadruple buffered.

Note: *The registers IRB_RX_SYM_PERIOD and IRB_RX_IR_ON_TIME act as a register set. A new value can only be read after reading both registers sequentially.*

IRB_RX_INT_EN Receive interrupt enable register

7	6	5	4	3	2	1	0
Reserved		RXFIFOIRQ[1:0]		Reserved		LASTSYMBOLIRQENABLE	RXIRQENABLE

Address: *IRBBaseAddress* + 0x48 (IR) *IRBBaseAddress* + 0x88 (UHF)

Type: Read/write

Reset: 0x00

Description: To inhibit all these interrupts the RXIRQENABLE bit (register bit 0) must be set to 0.

[7:6] **Reserved:** Set to logic 0

[5:4] **RXFIFOIRQ[1:0]:** Select the receive FIFO fullness interrupt

00: Invalid

01: One word available for read

10: Two words available for read (half full)

11: Three words available for read (FIFO full)

[3:2] **Reserved:** Set to logic 0

[1] **LASTSYMBOLIRQENABLE:** Select interrupt enable/disable on last symbol

1: Generate interrupt on last symbol received

[0] **RXIRQENABLE:** Select the receive interrupt enable/disable

0: Interrupt disable

1: Interrupt enable

IRB_RX_INT_STATUS**Receive Interrupt status register**

7	6	5	4	3	2	1	0
Reserved	RXFIFOSTATUS			Reserved	RXOVERRUNSTATUS	LASTSYMBOLIRQSTATUS	RXIRQSTATUS

Address: *IRBBaseAddress* + 0x4C (IR) *IRBBaseAddress* + 0x8C (UHF)

Type: Read only

Reset: 0x00

Description:

[7:6] **Reserved:** Set to logic 0

[5:4] **RXFIFOSTATUS[1:0]:** Receive FIFO fullness status

00: FIFO empty

01: One word in FIFO

10: Two words in FIFO

11: Three words in FIFO

[3] **Reserved:** Set to logic 0

[2] **RXOVERRUNSTATUS:** Receive overrun status

0: No overrun

1: Overrun occurred

[1] **LASTSYMBOLIRQSTATUS:** Last symbol interrupt status

1: Interrupt active

[0] **RXIRQSTATUS:** Receive interrupt status

1: Interrupt active

IRB_RX_EN**RC receive enable 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.

Confidential

IRB_RX_CLR_OVERRUN Clears the overrun status

7	6	5	4	3	2	1	0
Reserved							CLROVERRUN

Address: *IRBBaseAddress* + 0x58 (IR) *IRBBaseAddress* + 0x98 (UHF)

Type: Write only

Reset: 0x00

Description: This register must be set to 1 as part of the procedure for clearing the overrun flag in the register IRB_RX_INT_STATUS. No new data is written into the receive FIFO while this flag is set.

61.3 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.

61.4 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.

61.5 Reverse polarity registers

The two infrared transmitter/receiver 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 infrared 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 infrared 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

Confidential

62 Asynchronous serial controller (ASC)

62.1 Overview

The asynchronous serial controller, also referred to as the UART interface, provides serial communication between the STi5516 and other microcontrollers, microprocessors or external peripherals. The STi5516 provides four ASCs, two of which are generally used by the smartcard controllers.

Parity generation, 8- or 9-bit data transfer and the number of stop bits is programmable. Parity, framing, and overrun error detection is provided to increase the reliability of data transfers. The transmission and reception of data can simply be double-buffered, or 16-deep FIFOs may be used. Handshaking is supported on both transmission and reception. For multiprocessor communication, a mechanism to distinguish the address from the data bytes is included. Testing is supported by a loop back option. A dual mode 16-bit baudrate generator provides the ASC with a separate serial clock signal.

Each ASC supports full duplex, asynchronous communication, where both the transmitter and the receiver use the same data frame format and the same baudrate. Data is transmitted on the transmit data output pins ASCn_TXD and received on the receive data input pin ASCn_RXD.

Each ASC can be set to operate in smartcard mode for use when interfacing to a smartcard.

62.2 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, ASCn_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.

62.2.1 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.

62.2.2 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.

62.3 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).

62.3.1 8-bit data frames

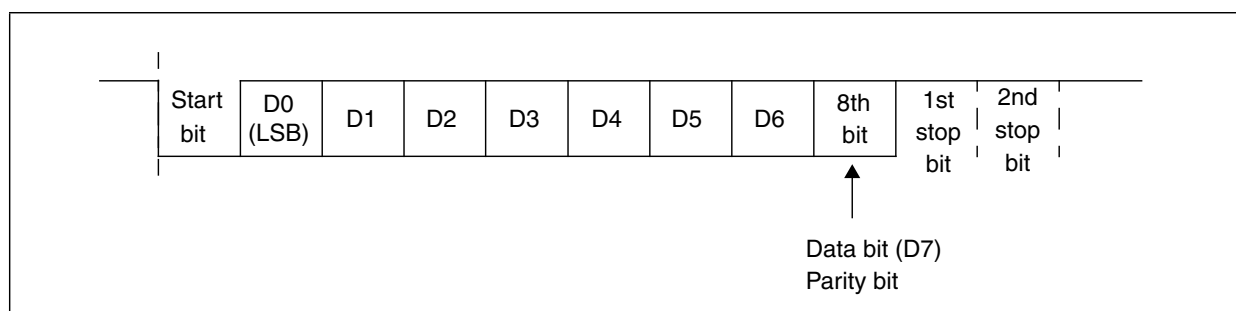
Figure 192 illustrates an 8-bit transmitted data frame. 8-bit frames may use one of the following formats:

- 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 192: 8-bit Tx data frame format

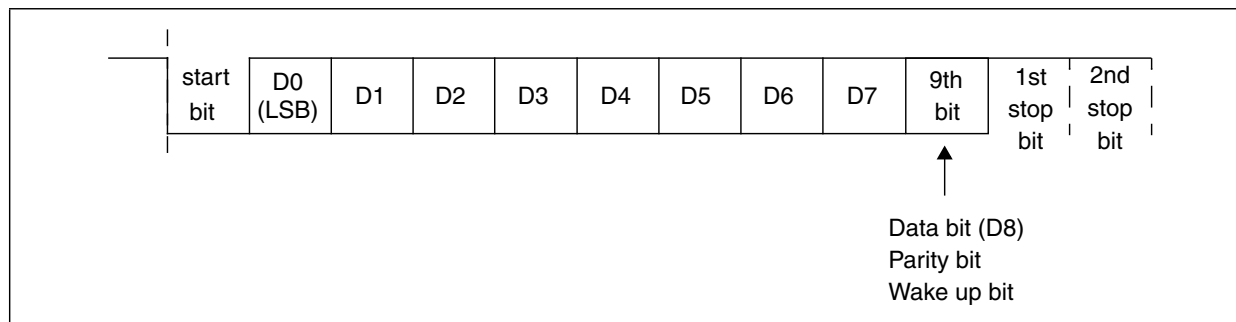


62.3.2 9-bit data frames

Figure 193 illustrates a 9-bit transmitted data frame. 9-bit data frames use 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 193: 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.

62.4 Transmission

Transmission begins at the next baudrate clock tick, provided that the RUN bit is set and data has been loaded into the ASC_n_TXBUFFER. If the CTSENABLE bit is set in the ASC_n_CONTROL register then transmission only occurs when NOT_ASCn_CTS is low.

The transmitter empty flag (TXEMPTY) indicates whether the output shift register is empty. It is set at the beginning of the last data frame bit that is transmitted, that is, during the first comms clock cycle of the first stop bit shifted out of the transmit shift register.

The loop back option (selected by the LOOPBACK bit of the ASC_n_CONTROL register) internally connects the output of the transmitter shift register to the input of the receiver shift register. This may be used to test serial communication routines at an early stage without having to provide an external network.

A transmission ends with stop bits (1 is output on ASCn_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.

62.4.1 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.

62.4.2 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.

62.5 Reception

Reception is initiated by a falling edge on the data input pin ASC_n_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 ASC_n_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 ASC_n_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 ASC_n_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.

62.5.1 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 ASC_n_RXD pin is not 1 during the programmed number of stop bit times (see [Section 62.5](#)).
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 62.3.1 on page 605](#)), the ninth bit of each input entry (bit 8 of 0 to 9) is undefined.

62.5.2 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 RXBUFFFULL bit of the ASC_n_STATUS register is set to 0. If one or more FIFO entries are present, the RXBUFFFULL 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 ASCn_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 (RXBUFFFULL) 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 ASCn_RXD pin.

62.5.3 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.

62.6 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.

62.6.1 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:

$$\text{BaudRate} = \frac{f_{\text{comms}}}{16 \times \text{ASCBaudRate}}$$

$$\text{ASCBaudRate} = \frac{f_{\text{comms}}}{16 \times \text{BaudRate}}$$

where:

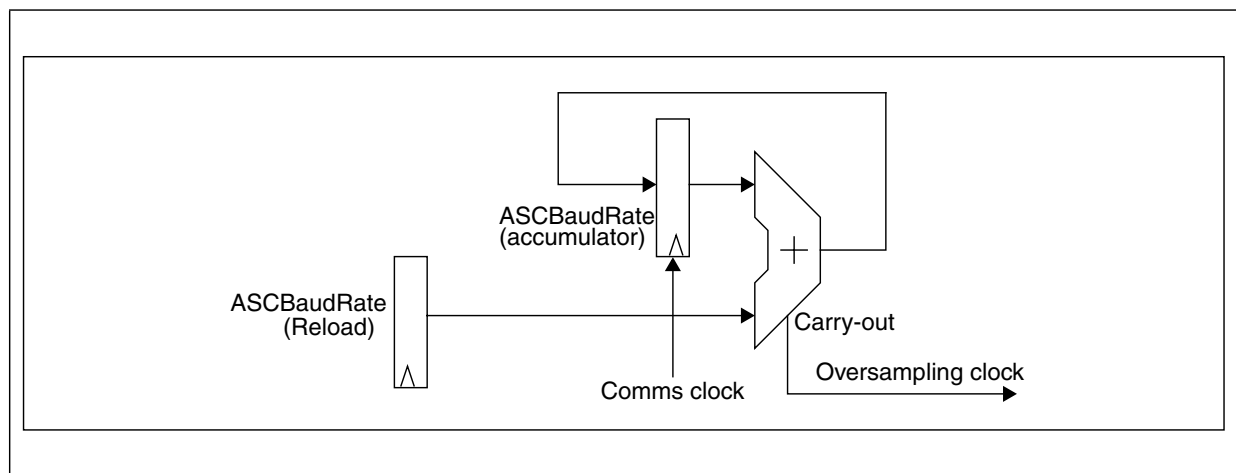
- *ASCBaudRate* represents the content of the ASC_n_BAUDRATE reload value register, taken as an unsigned 16-bit integer,
- f_{comms} is the frequency of the comms clock (clock channel PLL_CLOCK[2], see [Chapter 52: Clock generator on page 551](#)).

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 194](#).

Figure 194: Baudrate in mode 1

The CPU writes *ASCBAudRate* 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.

62.7 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 195.

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:

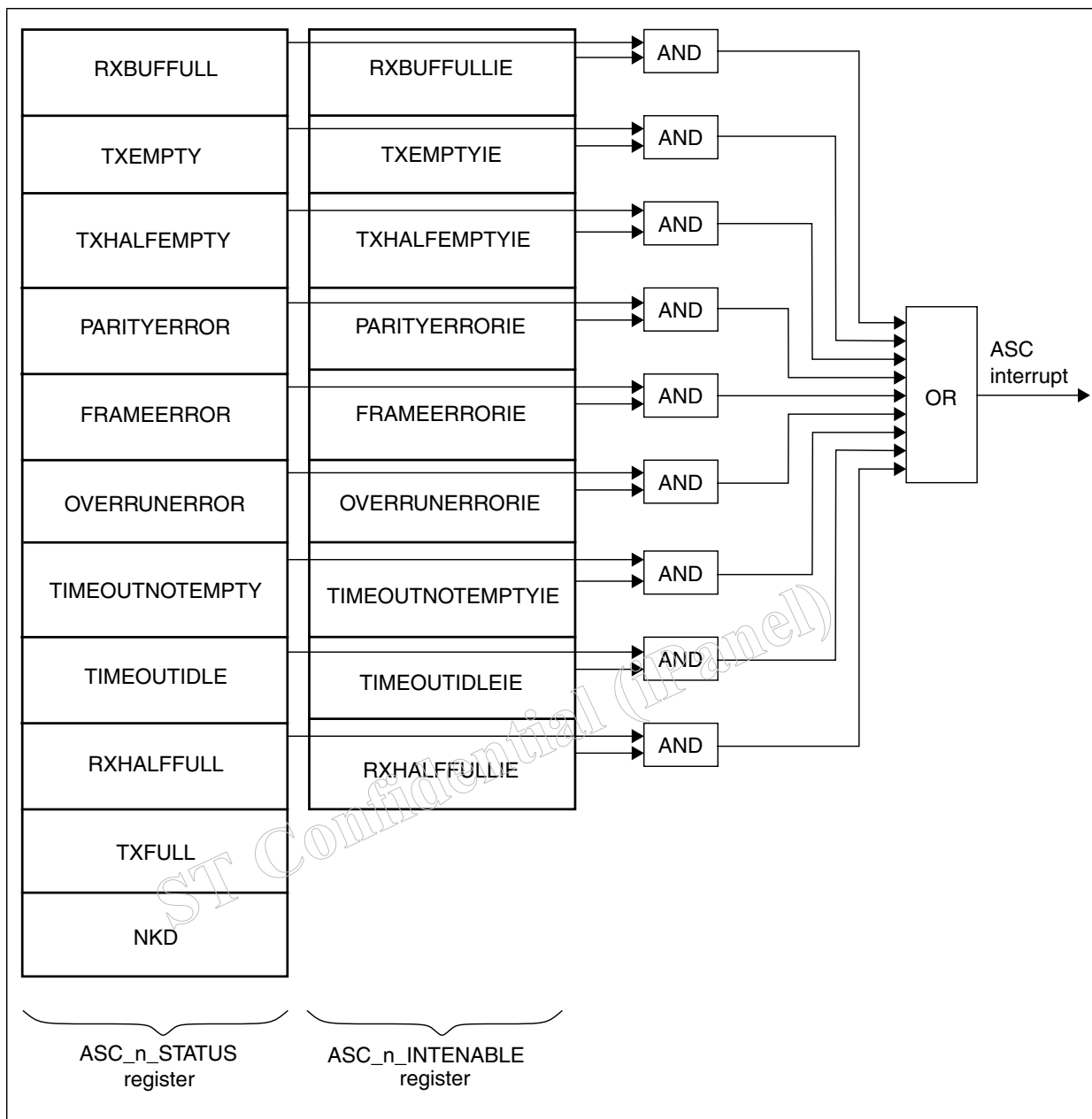
- transmitter interrupt status bits (TXEMPTY and TXHALFEMPTY) are reset when a character is written to the transmitter buffer,
- receiver interrupt status bit (RXBUFFFULL) is reset when a character is read from the receive buffer,
- PARITYERROR and FRAMEERROR status bits are reset when all characters 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.

62.7.1 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,
- RXBUFFFULL is activated when the received frame is moved to ASC_n_RXBUFFER.

Figure 195: ASC status and interrupt registers



As shown in [Figure 195](#), 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 handshakes should rely on TXEMPTY at the end of a data block to make sure that all data has really been transmitted.

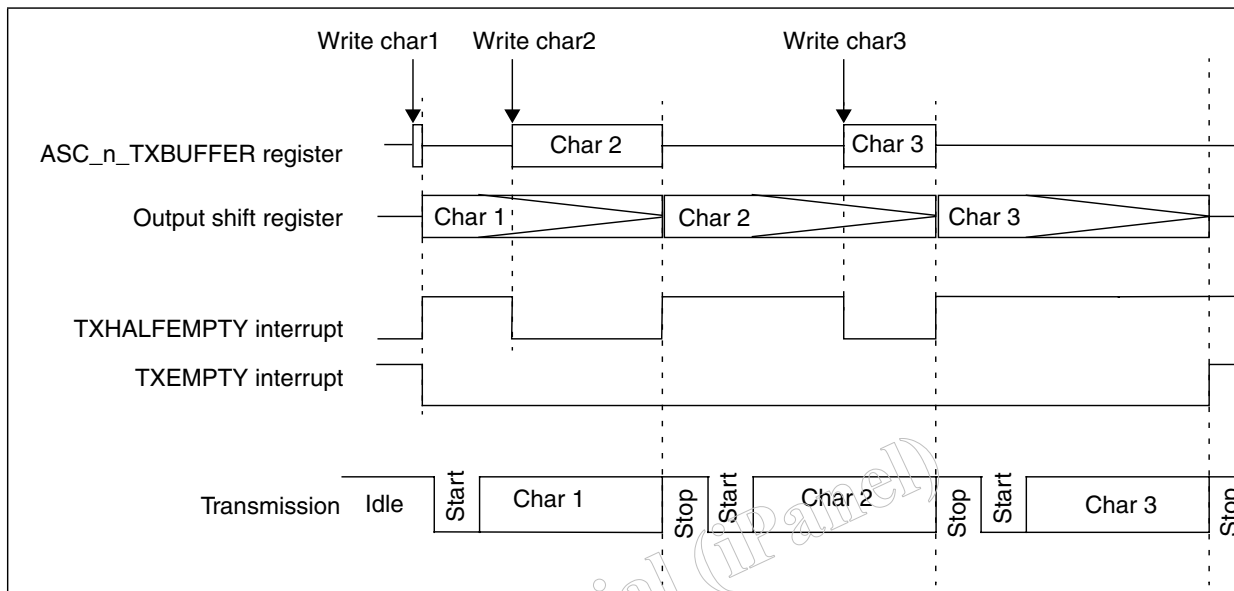
For single transfers it is sufficient to use the transmitter interrupt (TXEMPTY), which indicates that the previously loaded data has been transmitted, except for the last bit of a frame.

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.

62.7.2 Using the ASC interrupts when FIFOs are enabled

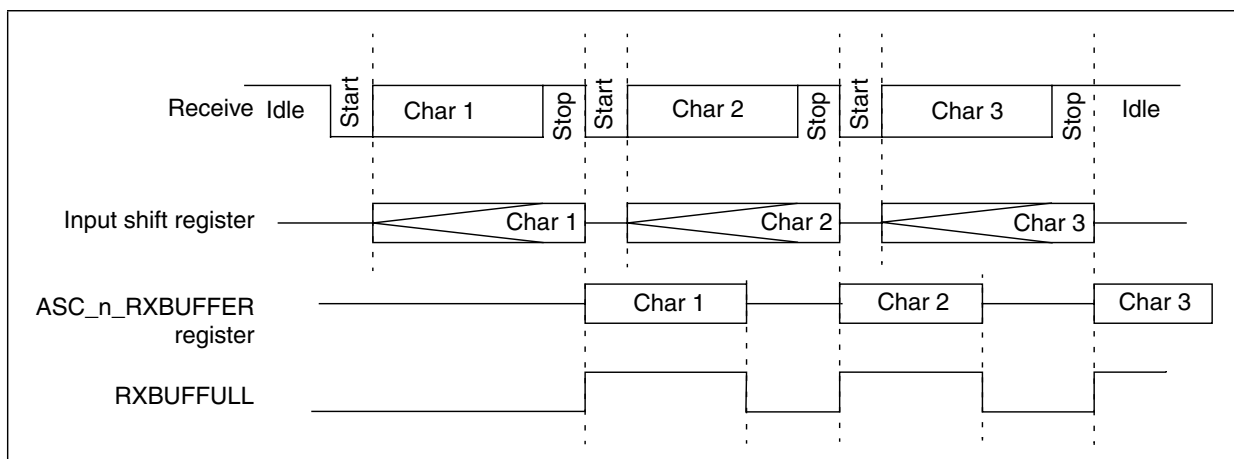
To transmit a large number of characters back to back, the driver routine initially writes 16 characters to ASC_n_TXBUFFER. Then every time a TXHALFEMPTY interrupt fires, it writes eight more. When there is nothing more to send, a TXEMPTY interrupt tells the driver that everything has been transmitted.

Figure 196: ASC transmission



When receiving, the driver can use RXBUFFFULL 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 197: ASC reception



62.8 Smartcard operation

Smartcard mode is selected by setting the SCENABLE bit in the ASC_n_CONTROL register to 1. In smartcard mode the RXD and TXD ports of the UART are both connected externally via a single bidirectional line to a smartcard I/O port. Characters are transferred to and from the smart card as 8-bit data frames with parity (see [Section 62.3 on page 605](#)). Handshaking between the UART and the smartcard ensures secure data transfer.

The UART supports both T=0 and T=1 protocol. In T=0 protocol, the reception of parity errors by either the UART or the smartcard is signalled by the automatic transmission of a NACK, where the receiver pulls the data line low, 0.5 baud clock periods after the end of the parity bit. The UART supports the reception and transmission of such NACKs. In T=1 protocol, this NACK behavior is not required, and any such behavior on the part of the UART can be disabled by setting the ASC_n_CONTROL bit NACKDISABLE.

When the SCENABLE bit in the ASC_n_CONTROL register is set to 0, normal UART operation occurs.

Smartcard operation complies with the ISO smartcard specification except where noted (see [Section 62.8.4](#)).

62.8.1 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](#) below). 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:

$$\text{Guardtime} = \text{guardtime} + 2 \pmod{256}$$

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.

62.8.2 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.

62.8.3 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 ASCn_TXD for one baud clock period from the end of the received stop bit. ASCn_RXD is masked when transmitting a NACK, since ASCn_TXD is tied to ASCn_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.

62.8.4 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.

63 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 $ASCnBaseAddresses$ are:

UART0: 0x2010 3000,

UART1: 0x2010 4000,

UART2: 0x2010 5000,

UART3: 0x2010 6000,

UART4: 0x2011 4000.

There is also one enable register located in the infrared blaster block. This is provided as $IRBBaseAddress$ + offset. The $IRBBaseAddress$ is 0x2011 5000.

A register summary is given in [Table 30: Asynchronous serial controller \(ASC\) registers on page 48](#).

ASC_ENABLE

Asynchronous I/O enable 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																ASC_EN															

Address: $IRBBaseAddress + 0x0D0$

Type: Read/write

Reset: 0

Description: If bit 0 in this register is set to 1, asynchronous data is available on the ASC2_TXD and ASC2_RXD pins.

Note: This register must be set to 1 to enable the ASC block.

Confidential

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 192 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 192: 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 193: 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, ASCn_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 27 26 25 24 23 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

Confidential

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.

64 Smartcard interface

64.1 Overview

The smartcard interface supports asynchronous protocol smartcards as defined in the ISO7816-3 standard. Limited support for synchronous smartcards can be provided in software by using the PIO bits to provide the clock, reset, and I/O functions on the interface to the card. Two smartcard interfaces are supported on the STi5516.

The UART function of the smartcard interface is provided by a UART (ASC). UART ASC0 can be used by smartcard0 and ASC1 can be used by smartcard1.

Each ASC used by a smartcard interface must be configured as eight data bits plus parity, 0.5 or 1.5 stop bits, with smartcard mode enabled. A 16-bit counter, the smartcard clock generator, divides down either the comms clock, or an external clock connected to a pin shared with a PIO bit, to provide the clock to the smartcard. PIO bits in conjunction with software are used to provide the rest of the functions required to interface to the smartcard. The inverse signalling convention, as defined in ISO7816-3, is handled in software, inverted data and most significant bit first. See [Chapter 62: Asynchronous serial controller \(ASC\) on page 604](#) for details of the ASC and [Chapter 68: Parallel I/O port on page 651](#) for details of the PIO ports.

64.2 External interface

The smartcard pin functions are described in the table below:

Table 194: Smartcard interface pins

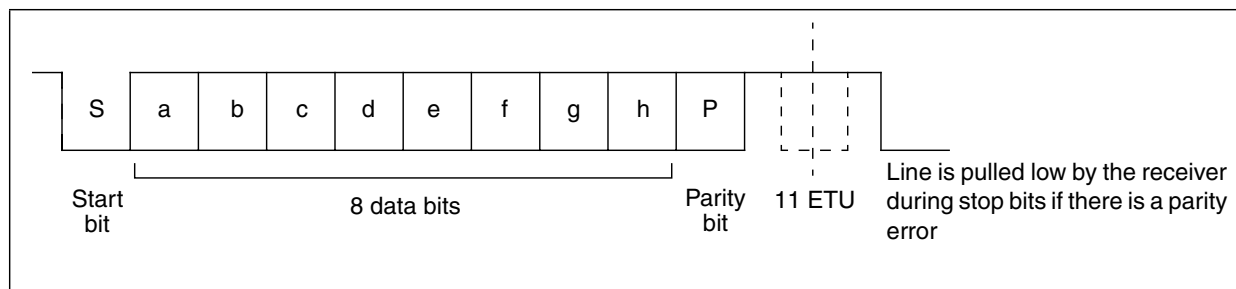
Pin	In/Out	Function
SCn_CLK	Out, open drain for 5V cards	Clock for smartcard
SC_EXTERNAL_CLOCK	In	External clock input to smartcard clock divider
SCn_DATA	Out, open drain driver	Serial data output. Open drain drive
SCn_DATA	In	Serial data input
SCn_RST	Out, open drain	Reset to card
SCn_CMD_VCC	Out	Supply voltage enable/disable
SCn_DETECT	In	Smartcard detection
SCn_DATA_DIR	Out	Indicates if the smartcard is operating in serial data output (open drain drive) mode or serial data input mode.

The SCn_RST, SCn_CMD_VCC, and SCn_DETECT signals are provided by alternate functions of the PIO pins. The UARTn_TXD data signal is connected to the SCn_DATA pin with the correct driver type and the clock generator is connected to the SCn_CLK pin.

The ISO standard defines the bit times for the asynchronous protocol in ETUs, which are related to the clock frequency received by the card. One bit time = one ETU.

The ASC transmitter output and receiver input must be connected together externally. For the transmission of data from the STi5516 to the smartcard, the ASC must be set up in smartcard mode.

Figure 198: ISO 7816-3 asynchronous protocol



64.3 Smartcard clock generator

The smartcard clock generator provides a clock signal to the smartcard. The smartcard uses this clock to derive the baudrate clock for the serial I/O between the smartcard and another UART. The clock is also used for the CPU in the card, if there is one present.

Operation of the smartcard interface requires that the clock rate to the card is adjusted while the CPU in the card is running code, so that the baudrate can be changed or the performance of the card can be increased. The protocols that govern the negotiation of these clock rates and the altering of the clock rate are detailed in the ISO7816-3 standard. The clock is used as the comms clock for the smartcard, so updates to the clock rate must be synchronized with the clock to the smartcard. This means the clock high or low pulse widths must not be shorter than either the old or new programmed value.

The clock generator clock source can be set to the system clock or an external pin. Two registers control the period of the clock and the running of the clock.

- The SCI_n_CLKVAL register determines the smartcard clock frequency. The value given in the register is multiplied by 2 to give the division factor of the input clock frequency. The divider is updated with the new value for the divider ratio on the next rising or falling edge of the output clock.
- The SCI_n_CLKCON register controls the source of the clock and determines whether the smartcard clock output is enabled. The programmable divider and the output are reset when the ENABLE bit is set to 0.

65 Smartcard interface registers

Addresses are provided as the *SmartcardnBaseAddress* + offset.

The *SmartcardnBaseAddresses* are:

SCG0: 0x2010 7000,

SCG1: 0x2010 8000.

A register summary is given in [Table 51: Smartcard interface registers on page 77](#).

SCI_n_CLKCON

Smartcard n clock 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																ENABLE		SOURCE													

Address: *SmartcardnBaseAddress* + 0x04

Type: Write only

Reset: 0

Description: This register controls the source of the clock and determines whether the smartcard clock output is enabled. The programmable divider and the output are reset when bit ENABLE is set to 0.

[31:2] **Reserved**

Write 0.

[1] **ENABLE:** Smartcard clock generator enable bit

0: Stop clock, set output low and reset divider

1: Enable clock generator

[0] **SOURCE:** Selects source of smartcard clock

0: Selects global clock

1: Selects external pin

SCI_n_CLKVAL

Smartcard n clock

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

Address: *SmartcardnBaseAddress* + 0x00

Type: Write only

Reset: 0

Description: This register determines the smartcard clock frequency. The 5-bit value given in the register is multiplied by 2 to give the division factor of the input clock frequency. For example, if SCnCLKVAL=8 then the input clock frequency is divided by 16. The value “zero” must not be written into this register.
The divider is updated with the new value for the divider ratio on the next rising or falling edge of the output clock.

66 Synchronous serial controller (SSC)

66.1 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 STi5516. The SSC gives full support for the I²C bus.

The SSC shares pins with the parallel input/output (PIO) ports. It supports half-duplex synchronous communication when used in conjunction with the PIO configuration.

The SSC uses three signals:

- serial clock SCLK,
- serial data in/out MRST,
- serial data out/in MTSR.

On the STi5516, the two serial data in/out signals are multiplexed onto a single pin (full-duplex mode is not supported).

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.

MRST and MTSR are connected together internally. These signals are connected to the SSC clock and data interface pins in a configuration which allows their direction to be changed when in master or slave mode (see [Section 66.2.1: Pin connection and control on page 630](#)). The serial clock signal is either generated by the SSC (in master mode) or received from an external master (in slave mode). The input and output data are synchronized to the serial clock.

The following features are programmable: baudrate, data width, shift direction, clock polarity, clock phase.

The SSC fully supports the I²C bus standard. The extra I²C 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.

66.2 Basic operation

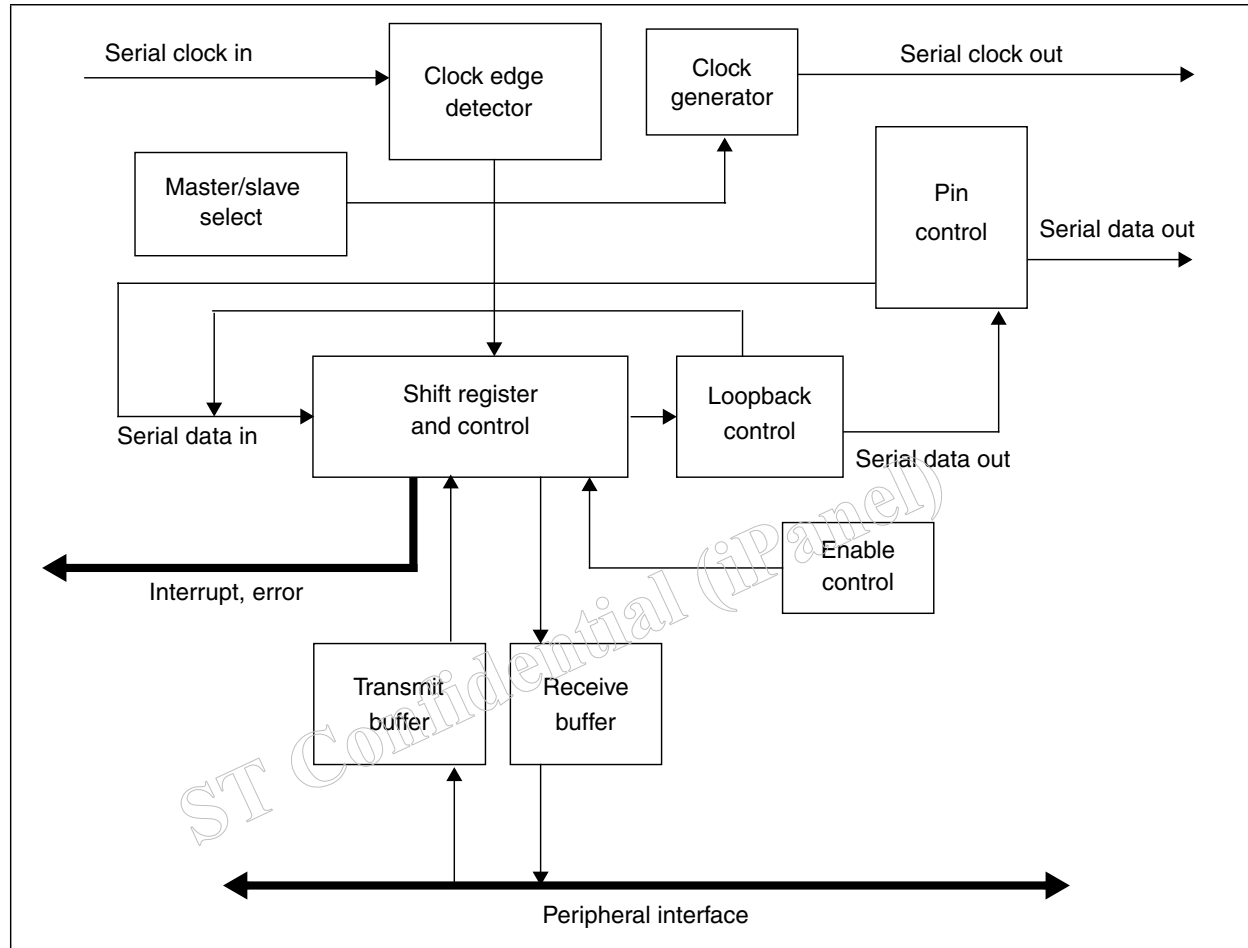
Control of the direction, either as input, output or bidirectional, of the SCLK, MTSR and MRST pins¹ is performed in software by configuring the PIO.

The serial clock output signal is programmable in master mode for baudrate, polarity and phase. This is described in [Section 66.2.2: Clock generation on page 630](#).

1. On the STi5516, the two serial data in/out signals are multiplexed on to a single pin (full-duplex mode is not supported).

The SSC works by taking the data frame (two to sixteen bits) from a transmission buffer and placing it into a shift register. It then shifts the data at the serial clock frequency out of the output pin and synchronously shifts in data coming from the input pin. The number of bits and the direction of shifting (MSB or LSB first) are programmable. This is described in [Section 66.2.4: Shift register on page 632](#).

Figure 199: SSC architecture



After the data frame has been completely shifted out of the shift register, it transfers the received data frame into the receive buffer. The transmit and receive buffers are described in [Section 66.2.7: Transmit and receive buffers on page 634](#). The SSC is therefore double buffered. This allows back-to-back transmission and reception of data frames up to the speed that interrupts can be serviced.

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 66.2.8: Loopback mode on page 634](#).

The SSC can be turned on and off by setting the enable control. This is described in [Section 66.2.9: Enabling operation on page 634](#). It can also be set to operate as a bus master or as a bus slave device. This is described in [Section 66.2.10: Master/slave operation on page 634](#).

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 66.2.11: Error detection on page 635](#).

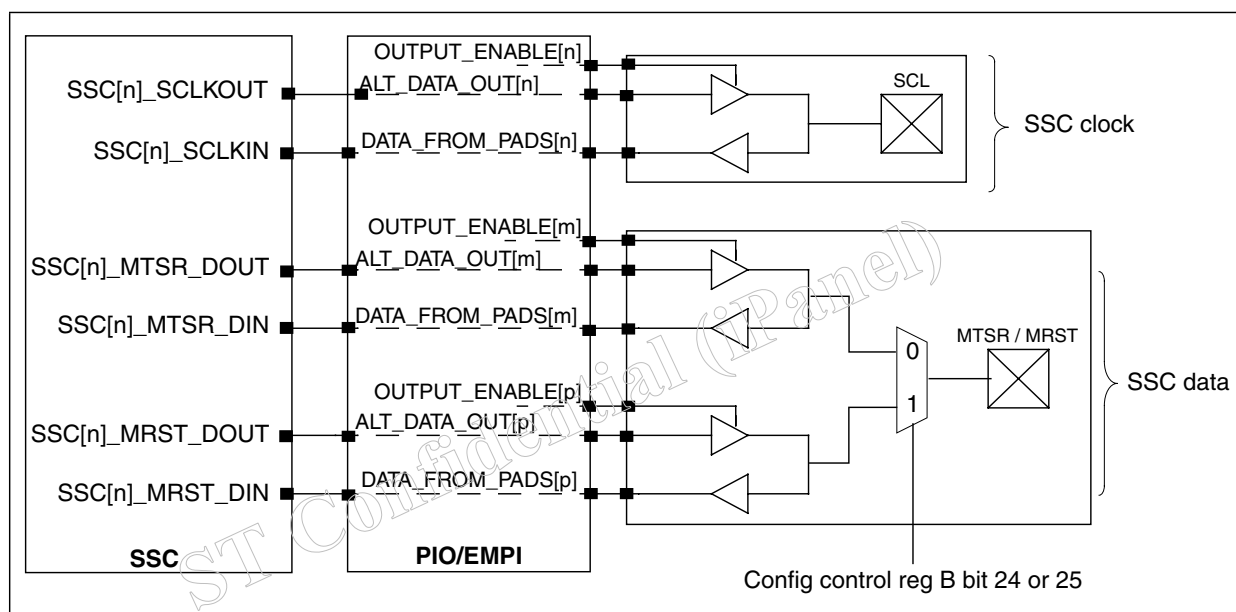
There are additional hardware features which can be independently enabled in order to fully support the I²C bus standard when used in conjunction with a suitable software driver. The additional I²C hardware is described in [Section 66.3: I²C operation on page 636](#).

66.2.1 Pin connection and control

The SCLK, MTSR and MRST signals are provided by four 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 200](#). Pins are shared with PIO.

Figure 200: SSC to PIO and EMPI connections



The pad control block inside the SSC determines which of the serial data input ports is used to read data from (depending on the master or slave mode). It also determines which of the serial data output ports to write data to (depending on the master or slave mode).

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.

66.2.2 Clock generation

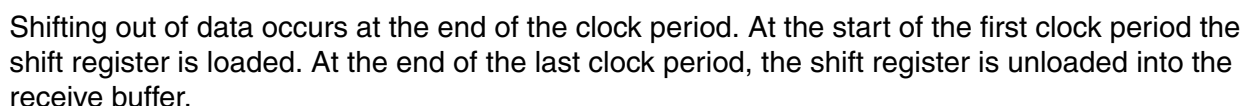
If the SSC is configured to be the bus master, then it generates a serial clock signal on the serial clock output port.

The clock signal can be controlled for polarity and phase and its period (baudrate) can be set to a variety of frequencies.

For I²C operation there are a number of additional clocking features. These are described in [Section 66.3: I²C operation on page 636](#).

The different combinations of polarity and phase are shown in [Figure 201](#).

Figure 201: Polarity and phase combinations



66.2.3 Baudrate generation

The SSC can generate a range of different baudrate clocks in master mode. These are set up by programming the baudrate generator register SSCnBRG.

In write mode this register is set up to program the baudrate as defined by the following formulae:

$$\text{Baudrate} = \frac{f_{\text{comms}}}{2 \times \text{SSCnBRG}} \quad \text{SSCnBRG} = \frac{f_{\text{comms}}}{2 \times \text{Baudrate}}$$

where SSCnBRG 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 195](#).

Table 195: Baudrates and bit times for different SSCnBRG 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.

66.2.4 Shift register

The shift register is loaded with the data in the transmit buffer at the start of a data frame. It then shifts data out of the serial output port and data in from the serial input port.

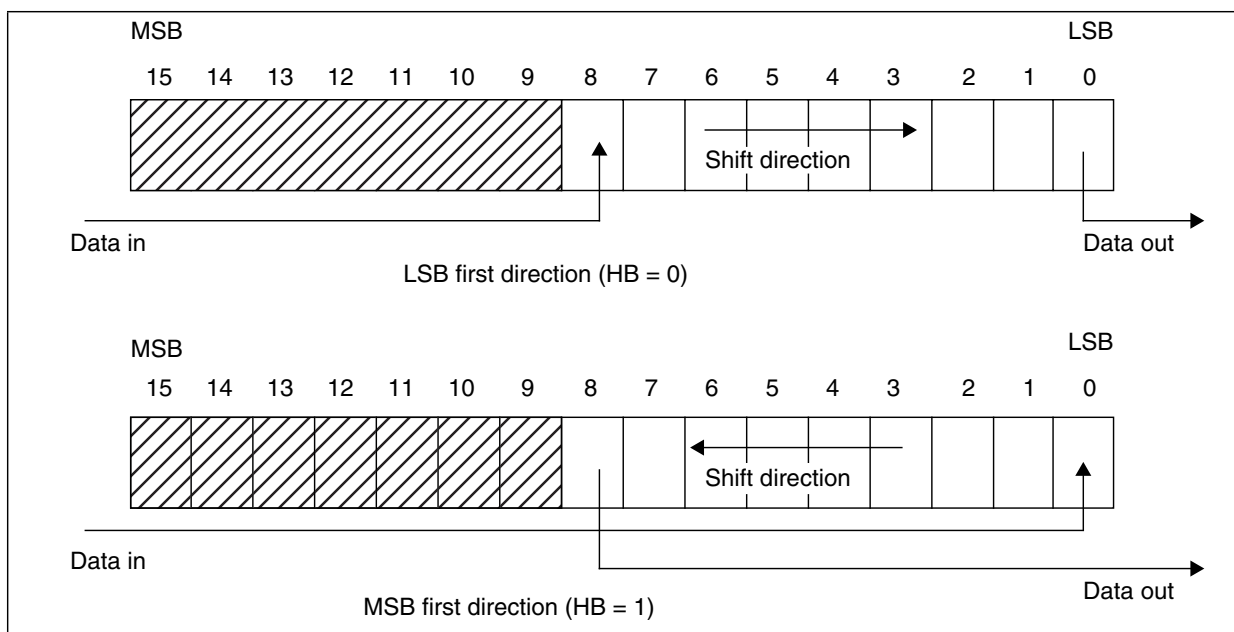
The shift register can shift out LSB first or MSB first. This is programmed by the heading control bit HB in the control register SSCnCON. A logic 1 indicates that the MSB is shifted out first and a logic 0 that the LSB shifts first.

The width of a data frame is also programmable from 2 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 I²C 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 202](#).

Figure 202: 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 66.2.2: Clock generation on page 630](#)). 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 I²C bus standard. These are described in [Section 66.3: I²C operation on page 636](#).

66.2.5 Receive data sampling

The data received by the SSC is sampled after the latching edge of the input clock, the latching edge being determined by the programming of the polarity and phase bits.

The data value which is finally latched is determined by taking three data samples at the third, fourth and fifth comms clock periods after the latching data edge. The data value is determined from the predominant data value in the three samples. This gives an element of spike suppression.

66.2.6 Antiglitich filter

The antiglitich filter suppresses any pulses which have a value in microseconds of less than a programmed width. Such signals may be either high or low. The filter has two registers, NOISE_SUPPRESS_WIDTH_SSC and PRE_SCALER_SSC.

NOISE_SUPPRESS_WIDTH_SSC holds the value of maximum glitch width. To suppress glitches of n microseconds and below, the value $n + 1$ is written into the register. Writing 0x00 into this register bypasses the antiglitich filter.

The comms clock is divided by a prescaler factor equivalent to 10 MHz, before being fed to the antiglitich filter. For example, if the comms clock is 50 MHz the prescaler division factor is 5.

66.2.7 Transmit and receive buffers

The transmit and receive buffers are used to allow the SSC to do back-to-back transfers; that is, continuous clock and data transmission.

The transmit buffer SSCnTBUF is written with the data to be sent out of the SSC. This is loaded into the shift register for transmission. Once this has been performed, the SSCnTBUF is available to be loaded again with a new data frame. This is indicated by the assertion of the transmit interrupt request status bit SSCTIR, which indicates that the transmit buffer is empty. This causes an interrupt if the transmit buffer empty interrupt is enabled, by setting the TIEN bit in the interrupt enable register SSCnIEN.

A transmission is started in master mode by a write to the transmit buffer. This starts the clock generation circuit and loads the shift register with the new data.

Continuous transfers of data are therefore possible by reloading the transmit buffer whenever the interrupt is received. The software interrupt routine has the length of time for a complete data frame in order to refill the buffer before it is next emptied. If the transmit buffer is not reloaded in time when in slave mode, a transmit error condition TE (see [Section 66.2.11: Error detection](#)) is generated.

The number of bits to be loaded into the transmit buffer is determined by the frame data width selected in the control register bit BM. The unused bits are ignored.

The receive buffer SSCnRBUF is loaded from the shift register when a complete data frame has been shifted in. This is indicated by the assertion of the receive interrupt request status bit RIR, which indicates that the receive buffer is full. This causes an interrupt if the receive buffer full interrupt is enabled, by setting the RIEN in the interrupt enable register.

The CPU should then read out the contents of this register before the next data frame has been received otherwise the buffer is reloaded from the shift register over the top of the previous data. This is indicated as a receive error condition RE. See [Section 66.2.11: Error detection](#).

The number of bits which is loaded into the receive buffer is determined by the frame data width selected in the control register BM. The unused bits are not valid and should be ignored.

66.2.8 Loopback mode

A loopback mode is provided which connects the SERIAL_DATA_OUT to SERIAL_DATA_IN. This allows software testing to be performed without the need for an external bus device. This mode is enabled by setting the LPB bit in the control register, SSCnCON. A setting of logic 1 enables loopback, logic 0 puts the SSC into normal operation.

66.2.9 Enabling operation

The transmission and reception of data by the SSC block can be enabled or disabled by setting the EN bit in the control register, SSCnCON. A setting of logic 1 turns on the SSC block for transmission and reception. Logic 0 prevents the block from reading or writing data to the serial data input and output ports.

66.2.10 Master/slave operation

The control of a number of the features of the SSC depends on whether the block is in master or slave mode. For example, in master mode the SSC generates the serial clock signal according to the setting of baudrate, polarity and phase. In slave mode, no clock is generated and instead the assumption is made that an external device is generating the serial clock.

Master or slave mode is set by the MS bit in the control register, SSCnCON. A setting of logic 0 means the SSC is in slave mode, a setting of logic 1 puts the device into master mode.

66.2.11 Error detection

A number of different error conditions can be detected by the SSC. These are related to the mode of operation (master or slave, or both).

On detection of any of these error conditions a status flag is set in the status register, SSCnSTAT. Also, if the relevant enable bit is set in the interrupt enables register SSCnIEN, then an error interrupt is generated from the SSC.

The different error conditions are described as follows.

Transmit error

A transmit error can be generated both in master and slave mode. It indicates that a transfer has been initiated by a remote master device before a new transmit data buffer value has been written in to the SSC.

In other words, the error occurs when old transmit data is going to be transmitted. This could cause data corruption in the half-duplex open drain configuration.

The error condition is indicated by the setting of the TE bit in the status register. An interrupt is generated if the TEEN bit is set in the interrupt enables register.

The transmit error status bit (and the interrupt, if enabled) is cleared by the next write to the transmit buffer.

Receive error

A receive error can be generated in both master and slave modes. It indicates that a new data frame has been completely received into the shift register and has been loaded into the receive buffer before the existing receive buffer contents have been read out. Consequently, the receive buffer has been overwritten with new data and the old data is lost.

The error condition is indicated by the setting of the RE bit in the status register SSCnSTAT. An interrupt is generated if the REEN bit is set in the interrupt enables register.

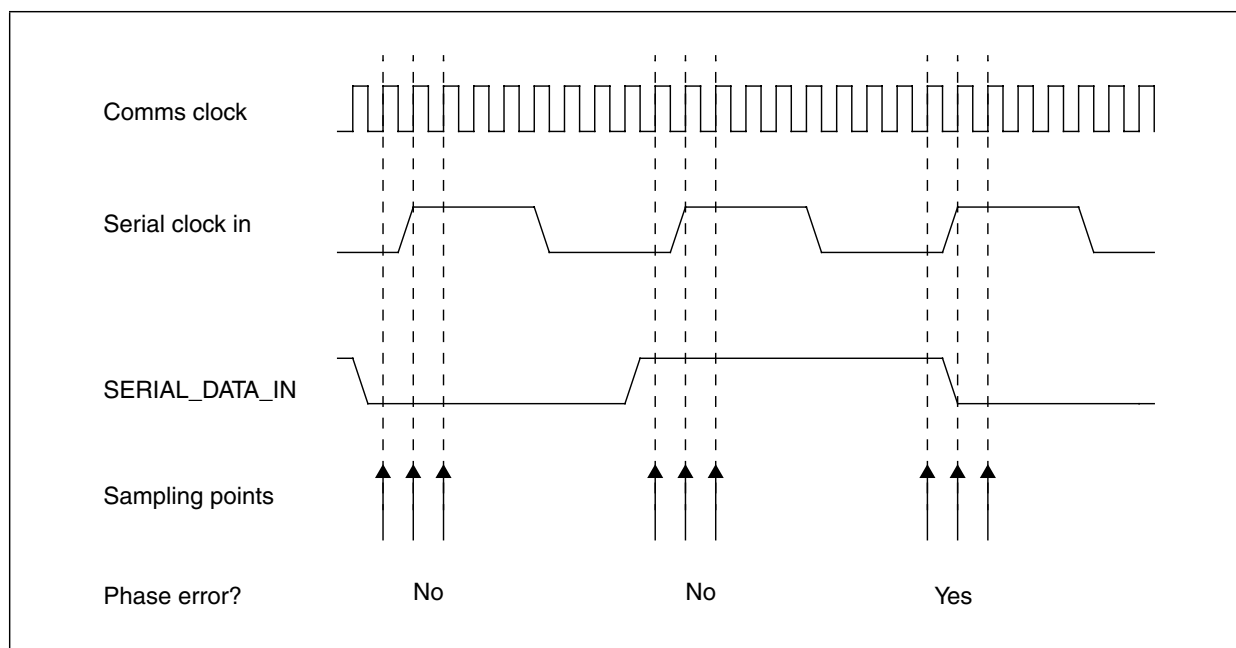
The receive error status bit (and the interrupt, if enabled) is cleared by the next read from the receive buffer.

Phase error

A phase error can be generated in master and slave modes. This indicates that the data received at the incoming data pin (MRST in master mode or MTSR in slave mode) has changed during the time from one sample before the latching clock edge and two samples after the edge.

The data at the incoming data pin is supposed to be stable around the time of the latching clock edge, hence the error condition. Each sample occurs at the comms clock frequency. The sampling scheme is shown in [Figure 203](#).

Figure 203: 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.

66.2.12 Interrupt mechanism

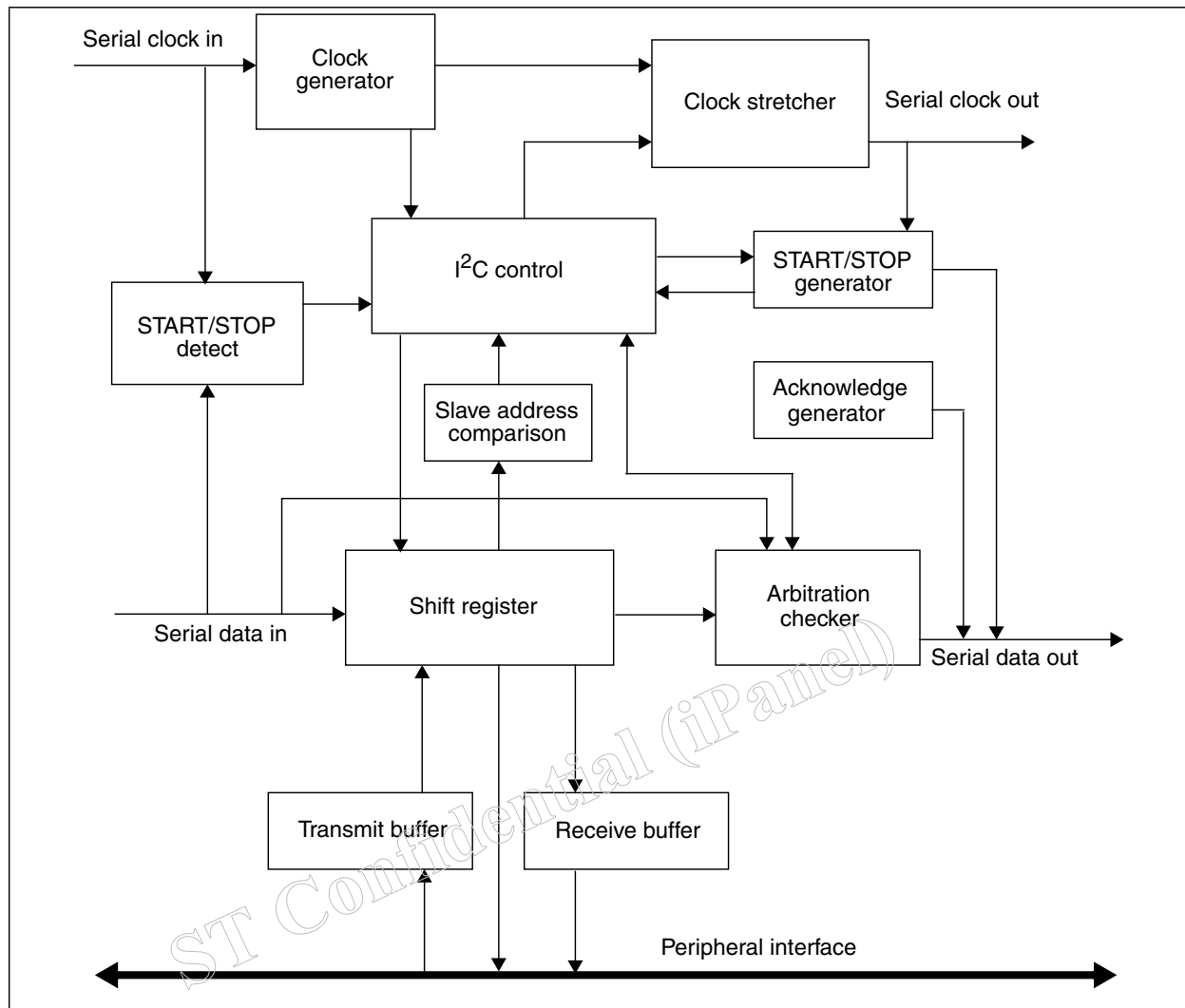
The SSC can generate a variety of different interrupts. They can all be enabled or disabled independently of each other. All the enabled interrupt conditions are ORed together to generate a global interrupt signal.

To determine which interrupt condition has occurred, a status register SSCnSTAT is provided which includes a bit for each condition. This is independent of the interrupt enables register SSCnIEN, and determines whether the condition asserts one or more of the interrupt signals.

66.3 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 204](#).

Figure 204: I²C architecture

66.3.1 I²C control

There are a number of features of the I²C-bus protocol which require special control.

- To allow slow slave devices to be accessed and to allow multiple master devices to generate a consistent clock signal, a clock synchronization mechanism is specified.
- START and STOP conditions must be recognized when in slave mode or multi-master mode. A START condition initiates the address comparison phase. A STOP condition indicates that a master has completed transmission and that the bus is now free.
- In slave mode (and in multi-master configurations), it is necessary to determine if the first byte received after a START condition is the address of the SSC. If it is, then an acknowledge must be generated in the ninth bit position.

Subsequently, an interrupt must be generated to inform the software that the SSC has been addressed as a slave device and therefore that it needs to either send data to the addressing master or to receive data from it.

In addition to normal 7-bit addressing, there is an extended 10-bit addressing mode where the address is spread over two bytes. In this mode, the SSC must compare two consecutive bytes with the incoming data after a START condition. It must also generate acknowledge bits for the first and second bytes automatically if the address matches.

The 10-bit addressing mode is further complicated by the fact that if the slave has been previously addressed for writing with the full 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 I²C control block which interacts with various other modules to perform the protocols.

In order to program for I²C mode, a separate control register SSCnI2C is provided. To perform any of the I²C hardware features, the I²C control bit I2CM, must be set in this register. When the I²C control bit is set, the clock synchronization mechanism is always enabled (see [Section 66.3.2: Clock synchronization on page 639](#)). When the I²C control bit is set, the START and STOP condition detection is performed. Fast mode is supported by bit 12 (I2CFSMODE) of the SSCnI2C register. In addition bits PH and PO of register SSC_nCON must be set to 1.

To program the slave address of the SSC the slave address register, SSCnSLAD must be written to with the address value. In the case of 7-bit addresses, only 7 bits should be written. For 10-bit addressing, the full 10 bits are written. The SSC then uses this register to compare the slave address transmitted after a START condition (see [Section 66.3.4: Slave address comparison on page 641](#)). 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 66.3.4: Slave address comparison](#)).

The clock stretching mechanism is enabled for various interrupt conditions when the I²C control enable bit I2CM in register SSCnI2C is set (see [Section 66.3.5: Clock stretching on page 642](#)).

To generate a START condition, the I²C START condition generate bit STRTG in register SSCnI2C, must be set (see [Section 66.3.6: START/STOP condition generation on page 642](#)). To generate a STOP condition, the I²C STOP condition generate bit STOPG, must be set (see [Section 66.3.6: 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 66.3.7: Acknowledge bit generation on page 643](#)).

To indicate to the software that various situations have arisen on the I²C bus, a number of status bits are provided in the status register SSCnSTAT. In addition, some of these bits can generate interrupts if corresponding bits are set in the interrupt enable register SSCnIEN.

To indicate that the SSC has been accessed as a slave device, the addressed as slave bit AAS in register SSCnSTAT, is set. This also causes an interrupt if the AASEN bit is set in register SSCnIEN.

The interrupt occurs after the SSC has generated the address acknowledge bit. In 10-bit addressing mode; 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 66.3.5: Clock stretching on page 642](#)). This forces the master device to wait until the software has processed the interrupt.

The status bit and the interrupt are reset by reading from the receive buffer SSCnRBUF, when the slave is being sent data, and by writing to the transmit buffer SSCnTBUF, when the SSC needs to send data.

To indicate that a STOP condition has been received, when in slave mode, the STOP condition detected bit STOP is set. This also causes an interrupt if the STOPEN bit is set in the interrupt enable register. The STOP interrupt and status bit is reset by a read of the status register SSCnSTAT.

To indicate that the SSC has lost the arbitration process, when in a multi-master configuration, the arbitration lost bit ARBL in register SSCnSTAT, is set. This also results in an interrupt if the ARBLEN bit is set in the interrupt enable register. The interrupt occurs immediately after the arbitration is lost.

Until the status bit is reset, the SSC holds the clock line low at the end of the current data frame, (see [Section 66.3.5: Clock stretching](#)). This forces the winning master device to wait until the software has processed the interrupt.

The interrupt and status bit is reset by a read of the status register SSCnSTAT.

To indicate that the I²C-bus is busy (that is, between a START and a STOP condition), the I²C bus busy bit BUSY in register SSCnSTAT is set. This does not generate an interrupt.

66.3.2 Clock synchronization

The I²C standard defines how the serial clock signal can be stretched by slow slave devices and how a single synchronized clock is generated in a multi-master environment. The clock synchronization of all the devices is performed as follows.

All master devices start generating their low clock pulse when the external clock line goes low (this may or may not correspond with their own generated high to low transition).

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.

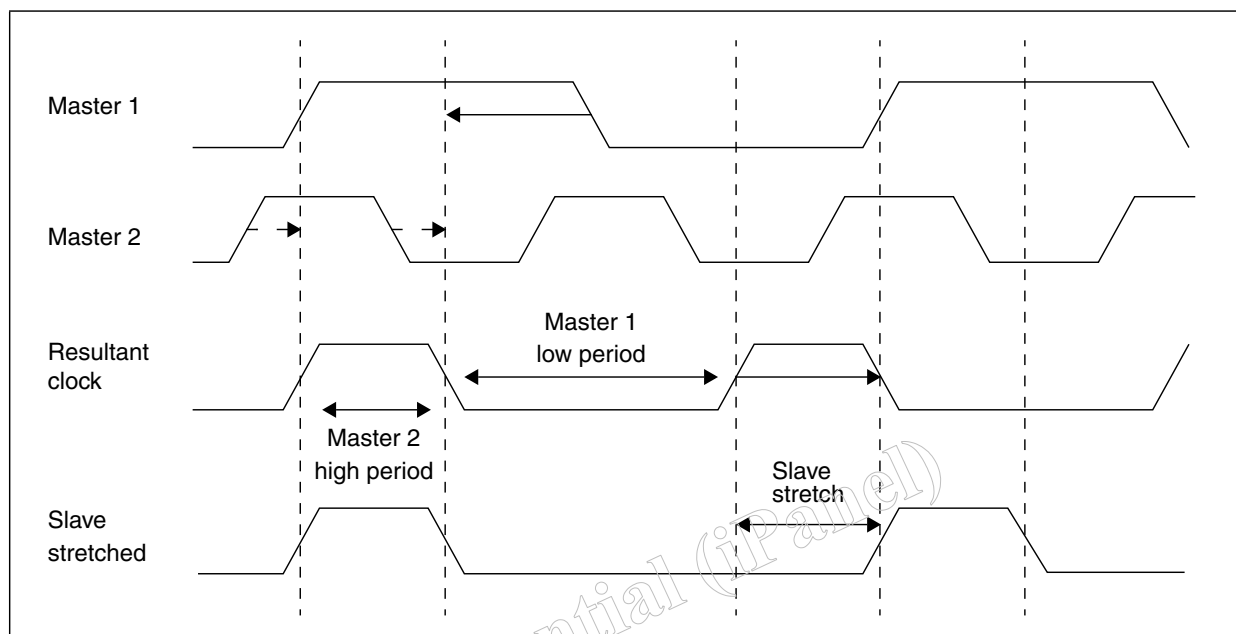
Once the external clock signal goes high, all the master devices begin counting off their high clock pulse. In this case the first master to finish counting attempts to pull the external clock line low and wins (because of the open drain line). The other master devices detect this and abort their high 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 205](#).

Figure 205: Synchronization and stretching



The SSC implements this clock synchronization mechanism when the I²C control bit I2CM, is enabled.

66.3.3 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 66.3.8: Arbitration checking on page 643](#)).

A STOP condition occurs when the transmit/receive data line changes from low to high during the high period of the clock line. It indicates that a master device has relinquished control of the bus (the bus is made free a specified time after the stop condition).

An additional piece of hardware is provided on the SSC to detect START and STOP conditions. This is necessary in slave mode as detection cannot be performed in time merely by programming the PIO pads. This is because there is not sufficient time for a software interrupt between the end of the START condition and the beginning of the data transmitted by a remote master.

START and STOP conditions are detected by sampling the data line continuously when the clock line is high. Minimum set up and hold times are measured by the counters.

The START condition is detected when data goes low (and the clock is high) and remains low for the minimum time specified by the I²C standard.

The STOP condition is detected when data goes high (and the clock is high) and remains high for the minimum time specified by the I²C standard.

START and STOP condition detection is enabled when the I²C control bit I2CM is set in the I²C control register.

When a START condition is triggered, the SSC informs the I²C control block which then initiates the address comparison phase.

When a STOP condition is triggered, the SSC sets the STOP bit in the status register. It also generates an interrupt if the STOPDEN bit is set in the interrupt enable register.

The interrupt and the status bit are cleared when the status register is read.

66.3.4 Slave address comparison

After a START condition has been detected, the SSC goes into the address comparison phase.

It receives the first eight bits of the next byte transmitted and compares the first seven bits against the address stored in the slave address register SSCnSLAD. If they match, the address comparison block indicates this to the I²C control block.

This generates an acknowledge bit in the next bit position and set the addressed as slave bit AAS in the status register. An interrupt is then generated after the acknowledge bit if the addressed as slave enable bit AASEN is set in the interrupt enables register.

The eighth bit of the first byte indicates whether the SSC is written to (low) or read from (high). This is used by the control block to determine if it needs to acknowledge the following data bytes (that is, when receiving data).

When 10-bit addressing mode is selected by setting the 10-bit addressing bit AD10 in register SSCnI2C, the first seven bits of the first data byte is compared against 11110nn, where nn is the two most significant bits of the 10-bit address stored in the slave address register.

The read/write bit then determines what to do next.

If the read/write bit is low, indicating a write, an acknowledge must be generated for the byte. The addressed as slave status bit and interrupt however are not yet asserted so, instead, the address comparator waits for the next data byte and compares this against the eight least significant bits of the slave address register.

If this matches, then the SSC is being addressed, so the second byte is acknowledged and the addressed as slave bit is set. An interrupt also occurs after the acknowledge bit if the addressed as slave interrupt enable is set.

On the other hand if the first byte sent has the read/write bit high, then the SSC only acknowledges it if it has previously been addressed and a STOP condition has not yet occurred (that is, the master has generated a repeated START condition). In this case the addressed as slave bit is set after the first byte plus acknowledge and an interrupt is generated if the interrupt enable is set. The second byte in this case is sent by the SSC as this is a read operation.

In all cases if the address does not match, then the SSC ignores further data until a STOP condition is detected.

66.3.5 Clock stretching

The I²C standard allows slave devices to hold the clock line low if they need more time to process the data being received (see [Section 66.3.2: Clock synchronization on page 639](#)). The SSC takes advantage of this by inserting extended clock low periods. This is done to allow a software device driver to process the interrupt conditions when in slave mode.

The clock stretching mechanism is used in the situations listed below.

- When the SSC has been addressed as a slave device and the interrupt has been enabled. The clock stretch occurs immediately after the first byte with acknowledge, after a START condition has occurred (or in the case of 1-bit addressing this might occur after the second byte plus acknowledge). This gives the software interrupt routine time to initialize for transmission or reception of data. The clock stretch is cleared by writing 0x1FF to the transmit buffer register.
- When the SSC is in slave mode and is transmitting or receiving. The clock stretch occurs immediately after each data byte plus acknowledge. When transmitting, this allows the software interrupt routine to check that the master has acknowledged before writing the next data byte into the transmit buffer. If no acknowledge is received, then the software must stop transmitting bytes. When receiving, it allows the software to read the next data byte before the master starts to send the next one. The clock stretch is cleared by a write to the transmit buffer when transmitting and by a read from the receive buffer when receiving.
- When the SSC loses arbitration. The clock stretch occurs immediately after the current data byte and acknowledge have been performed only if the master which has lost arbitration has been addressed. This gives the software time to abort its current transmission and prepare to retry after the next STOP condition. The clock stretch is not performed if the master which has lost arbitration has not been addressed.

If a clock stretching event occurs but no relevant interrupt is enabled then the clock is stretched indefinitely. Hence it is important that the correct interrupts are always enabled.

66.3.6 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 I²C control block determines this and instructs the START/STOP generator to hold the lines high between transactions.

The START/STOP generator is controlled by the START condition generate bit STRTG and the STOP condition generate bit STOPG in register SSCnI2C.

The generator pulls the SERIAL_DATA_OUT line low during the high period of the clock to produce a START condition. In the case of a STOP condition it pulls the data line high.

However, a START condition is only generated if the bus is currently free (that is, the BUSY bit in the status register is low). This is to prevent the SSC from generating a START condition when another master has just generated one.

If a START condition cannot be generated because the bus is busy, then the generator forces the arbitration checker to generate an arbitration lost interrupt and prevent data from being transmitted for the next byte. The software interrupt handler is therefore informed of the aborted transmission when servicing the interrupt. Bit 11 (REPSTRT) of register SSCnSTAT shows that a repeated start condition has occurred.

To properly generate the timing waveforms of the START and STOP conditions, the SSC contains a timing counter. This ensures the minimum setup and hold times are met with some additional margin.

66.3.7 Acknowledge bit generation

For I²C operation, it is required to both detect acknowledge bits when transmitting data, and to generate them when receiving data.

An acknowledge bit must be transmitted by the receiver at the end of every 8-bit data frame. The transmitter must verify that an acknowledge bit has been received before continuing.

An acknowledge bit is not generated by a master receiver for the last byte it wishes to receive. This “not acknowledge” is used by the slave device to determine when to stop transmission.

The acknowledge bit is generated by the receiver after the eight data bits have been transferred to it. In the ninth clock pulse, the transmitter holds the data line high and the receiver must pull the line low to acknowledge receipt. If the receiver is unable to acknowledge receipt, then the master generates a stop condition to abort the transfer.

Acknowledge bits are generated by the SSC when the acknowledge generation bit, ACKG, is set in the I²C 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.

66.3.8 Arbitration checking

This situation only arises when two or more master devices generate a START condition within the minimum hold time of the bus standard. This generates a valid start condition on the bus with more than one master valid.

However, a master device cannot determine if two or more masters have generated a START condition, so arbitration is always enabled. The arbitration for which device wins control of the bus is determined by which master is the first to transmit a low data bit on the data line when the other master wants to send a high bit. This master wins control of the bus. Therefore a master which detects a different data bit on its input to that which it transmitted must switch off its output stage for the rest of the eight bit data byte, as it has lost the arbitration.

The arbitration scheme does not affect the data transmitted by the winning master. Consequently, arbitration proceeds concurrently with data transmission and the data received by the selected slave during the arbitration process. It is valid that the winning master is actually addressing the losing master and hence this device must respond as if it were a slave device.

Arbitration is implemented in hardware by comparing the transmitted and received data bits every cycle. Loss of arbitration is indicated by the setting of the ARBL arbitration lost error flag in the status register. An interrupt also occurs if the ARBLN 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.

67 Synchronous serial controller (SSC) registers

Addresses are provided as the *SSCnBaseAddress* + offset.

The *SSCnBaseAddresses* are:

SSC0: 0x2010 9000,

SSC1: 0x2010 A000.

A register summary is given in [Table 53: Synchronous serial controller \(SSC\) registers on page 79](#).

SSCnBRG

SSC n baudrate generation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRG															

Address: *SSCnBaseAddress* + 0x00

Type: Read/write

Reset: 1

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					LPB	EN	MS	SR	PO	PH	HB	BM			

Address: *SSCnBaseAddress* + 0x0C

Type: Read/write

Reset: 0

Description:

[15:11] **Reserved**

[10] **LPB**: SSC loopback bit

0: Disabled, 1: Shift register output is connected to shift register input

[9] **EN**: SSC enable bit

0: Transmission and reception disabled, 1: Transmission and reception enabled

[8] **MS**: SSC master select bit

0: Slave mode, 1: Master mode

[7] **SR**: SSC software reset

0: Device is not reset, 1: All functions are reset while this bit is set

[6] **PO**: SSC clock polarity control bit

0: Clock idles at logic 0, 1: Clock idles at logic 1

Must be set in I²C mode.

[5] **PH**: SSC clock phase control bit

0: Pulse in second half cycle, 1: Pulse in first half cycle

Must be set in I²C 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

SSCnI2C

SSC n I²C control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			I2CFSMODE	REPSTRTG	Reserved					TXENB	AD10	ACKG	STOPG	STRTG	I2CM

Address: *SSCnBaseAddress* + 0x18

Type: Read/write

Reset: 0

Description: To suit I²C specifications, bits PH and PO of register SSC_nCON must also be set to 1.

[15:13] **Reserved**

[12] **I2CFSMODE**: Configures standard or fast mode for I²C operation
 0: Standard mode 1: Fast mode

[11] **REPSTRTG**: SSC I²C generate repeated START condition
 0: Disabled 1: Enabled

[10:6] **Reserved**

[5] **TXENB**: SSC I²C transaction enable control
 0: Disabled 1: Enabled

[4] **AD10**: SSC I²C 10-bit addressing control
 0: Disabled 1: Use 10 bit addressing

[3] **ACKG**: SSC I²C generate acknowledge bits
 0: Disabled 1: Generate acknowledge bits when receiving

[2] **STOPG**: SSC I²C generate STOP condition
 0: Disabled 1: Generate a STOP condition

[1] **STRTG**: SSC I²C generate START condition
 0: Disabled 1: Generate a START condition

[0] **I2CM**: SSC I²C control bit
 0: Disabled 1: Enable I²C features

SSCnIEN**SSCn interrupt enable**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				REPSTRTEN	NACKEN	Reserved	ARBLEN	STOPEN	AASEN	Reserved	PEEN	REEN	TEEN	TIEN	RIEN

Address: $SSCnBaseAddress + 0x10$

Type: Read/write

Reset: 0

Description: This register holds the interrupt enable bits, which can be used to mask the interrupts.

[15:12] **Reserved**

[11] **REPSTRTEN**: I²C repeated start condition interrupt enable
1: Repeated condition interrupt enabled

[10] **NACKEN**: I²C NACK condition interrupt enable
1: NACK condition interrupt enabled

[9] **Reserved**

[8] **ARBLEN**: I²C arbitration lost interrupt enable
1: Arbitration lost interrupt enabled

[7] **STOPEN**: I²C stop condition interrupt enable
1: Stop condition interrupt enabled

[6] **AASEN**: I²C 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

SSCnRBUF**SSC n receive buffer**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD[15:0]															

Address: $SSCnBaseAddress + 0x08$

Type: Read only

Reset: 0

Description: Receive buffer data RD15 to RD0.

SSCnSLAD**SSC n slave address**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						SL[9:7]			SL[6:0]						

Address: $SSCnBaseAddress + 0x1C$

Type: 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].

SSCnSTAT**SSC n status**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				REPSTRT	NACK	BUSY	ARBL	STOP	AAS	CLST	PE	RE	TE	TIR	RIR

Address: $SSCnBaseAddress + 0x14$

Type: Read only

Reset: 2, that is, all active bits clear except TIR.

Description:

[15:12] **Reserved**

[11] **REPSTRT**: I²C repeated start flag

1: I²C repeated start condition detected

[10] **NACK**: I²C NACK flag

1: NACK received

[9] **BUSY**: I²C bus busy flag

1: I²C bus busy

[8] **ARBL**: I²C arbitration lost flag

1: Arbitration lost

[7] **STOP**: I²C stop condition flag

1: Stop condition detected

[6] **AAS**: I²C addressed as slave flag

1: Addressed as slave device

[5] **CLST**: I²C 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

SSCnTBUF**SSC n transmit buffer**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															

Address: *SSCnBaseAddress* + 0x04

Type: Write only

Reset: 0

Description: Transmit buffer data TD15 to TD0.

CLEAR_STATUS_SSC**SSC clear bit operation**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CIR_REPSTRT	CIR_NACK	Reserved	CIR_SSCARBL	CIR_SSCSTOP	CIR_SSCAAS	Reserved					

Address: *SSCnBaseAddress* + 0x80

Type: Read/write

Reset: 0

Description:

[15:12] **Reserved**

[11] **CIR_REPSTRT**

1: To clear the SSCREPSTRT in SSCnSTAT

[10] **CIR_NACK**

1: To clear the SCCNACK in SSCnSTAT

[9] **Reserved**

[8] **CIR_SSCARBL**

1: To clear SSCARBL

[7] **CIR_SSCSTOP**

1: To clear SSCSTOP

[6] **CIR_SSCAAS**

1: To clear SSCAAS

[5:0] **Reserved**

NOISE_SUPPRESS_WIDTH_SSC Noise suppression width

7	6	5	4	3	2	1	0
NOISESUPPRESSWIDTH							

Address: *SSCnBaseAddress* + 0x100

Type: Read/write

Reset: 0x00

Description: The value, in microseconds, in this register determines the maximum width of noise pulses which the filter suppresses. To suppress glitches of n width, load n+1 in this register. All signal transitions whose width is less than the value in NOISE_SUPPRESS_WIDTH_SSC are suppressed. Writing 0x00 into this register bypasses the antiglitch filter.

PRE_SCALER_SSC**Clock prescaler**

7	6	5	4	3	2	1	0
Reserved				PRESCALEVAL			

Address: *SSCnBaseAddress* + 0x104

Type: Read/write

Reset: 0x00

Description: This register holds the prescaler division factor for glitch suppression, equivalent to 10 MHz. For example if the comms clock is 50 MHz the prescaler division factor should be 5.

Confidential

ST Confidential (iPanel)

68 Parallel I/O port

There are 48 bits of parallel I/O configured in six ports. Each bit is programmable as output or input. The output can be configured as a totem-pole or open-drain driver. The input compare logic can generate an interrupt on any change of any input bit. Many parallel I/O have alternate functions and can be connected to an internal peripheral signal such as a UART or SSC.

The PIO ports can be controlled by registers, mapped into the device address space. The registers for each port are grouped in a 4 Kbyte block, with the base of the block for port *n* at the address *PIO_nBaseAddress*. During reset all of the registers are reset to zero.

Each 8-bit PIO port has a set of eight-bit registers. Each of the eight bits of each register refers to the corresponding pin in the corresponding port. These registers hold:

- the output data for the port (PIO_PnOUT),
- the input data read from the pin (PIO_PnIN),
- PIO bit configuration registers (PIO_PnC[2:0]),
- the two input compare function registers (PIO_PnCOMP and PIO_PnMASK).

Each of the registers, except PIO_PnIN, is mapped on to two additional addresses so that bits can be set or cleared individually.

- The PIO_SET_x registers set bits individually. Writing 1 in these registers sets a corresponding bit in the associated register x; 0 leaves the bit unchanged.
- The PIO_CLEAR_x registers clear bits individually. Writing 1 in these registers resets a corresponding bit in the associated register x; 0 leaves the bit unchanged.

By default PIO5[3] and PIO4[4] use their alternate functions, even when the PIO is not in alternate function mode. To pass PIO data instead of the alternate function data, bits 4 and 5 in register CONFIG_CONTROL_C must be set to 1 after reset.

Confidential

69 Parallel 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 *PIOBaseAddress* + offset.

The *PIOBaseAddress* are:

PIO0: 0x2010 C000,
 PIO1: 0x2010 D000,
 PIO2: 0x2010 E000,
 PIO3: 0x2010 F000,
 PIO4: 0x2011 0000,
 PIO5: 0x2011 2000.

A register summary is given in [Table 48: Parallel I/O port registers on page 73](#).

PIO_CLEAR_PnC[2:0] Clear bits of PnC[2:0]

	7	6	5	4	3	2	1	0
0x28	CLEAR_PC0[7:0]							
0x38	CLEAR_PC1[7:0]							
0x48	CLEAR_PC2[7:0]							

Address: *PIOBaseAddress* + 0x28 (PIO_CLEAR_PnC0), 0x38 (PIO_CLEAR_PnC1), 0x48 (PIO_CLEAR_PnC2)

Type: Write only

Description: PIO_CLEAR_PnC[2:0] allows the bits of registers PIO_PnC[2:0] to be cleared individually. Writing 1 in one of these register clears the corresponding bit in the corresponding PIO_PnC[2:0] register, while 0 leaves the bit unchanged.

PIO_CLEAR_PnCOMP Clear bits of PnCOMP

	7	6	5	4	3	2	1	0
	CLEAR_PCOMP[7:0]							

Address: *PIOBaseAddress* + 0x58

Type: Write only

Description: PIO_CLEAR_PnCOMP allows bits of PIO_PnCOMP to be cleared individually. Writing 1 in this register clears the corresponding bit in the PIO_PnCOMP register, while 0 leaves the bit unchanged.

Confidential

PIO_CLEAR_PnMASK Clear bits of PnMASK

7 6 5 4 3 2 1 0

CLEAR_PMASK[7:0]

Address: *PIOBaseAddress* + 0x68

Type: Write only

Description: PIO_CLEAR_PnMASK allows bits of PIO_PnMASK to be cleared individually. Writing 1 in this register clears the corresponding bit in the PIO_PnMASK register, while 0 leaves the bit unchanged.

PIO_CLEAR_PnOUT Clear bits of PnOUT

7 6 5 4 3 2 1 0

CLEAR_POUT[7:0]

Address: *PIOBaseAddress* + 0x08

Type: Write only

Description: PIO_CLEAR_PnOUT allows bits of PIO_PnOUT to be cleared individually. Writing 1 in this register clears the corresponding bit in the PIO_PnOUT register, while 0 leaves the bit unchanged.

Confidential

ST Confidential (iPanel)

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: *PIOBaseAddress* + 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 196](#).

Table 196: 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: *PIOBaseAddress* + 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_CLEAR_PnCOMP, so that bits can be set or cleared individually.

PIO_PnIN**PIO input**

7 6 5 4 3 2 1 0

PIN[7:0]

Address: *PIOBaseAddress* + 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: *PIOBaseAddress* + 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_CLEAR_PnMASK, so that bits can be set or cleared individually.

PIO_PnOUT**PIO output**

7 6 5 4 3 2 1 0

POUT[7:0]

Address: *PIOBaseAddress* + 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_CLEAR_PnOUT, so that bits can be set or cleared individually.

PIO_SET_PnC[2:0]**Set bits of PnC[2:0]**

7 6 5 4 3 2 1 0

0x24 SET_PC0[7:0]

0x34 SET_PC1[7:0]

0x44 SET_PC2[7:0]

Address: *PIOBaseAddress* + 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: *PIOBaseAddress* + 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: *PIOBaseAddress* + 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.

Confidential

PIO_SET_PnOUT**Set bits of PnOUT**

7

6

5

4

3

2

1

0

SET_POUT[7:0]

Address: *PIOBaseAddress* + 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.

Confidential

ST Confidential (iPanel)

70 IEEE 1284 port (PC parallel port)

70.1 Overview

The IEEE 1284 port 8-bit wide parallel interface supports a high speed data input/output to and from the set top receiver. It is capable of interfacing to a PC using the IEEE 1284 standard. The interface has a dedicated DMA controller to transfer data between memory and the port with little CPU overhead.

The IEEE 1284 specification¹ defines a standard for asynchronous, interlocked, bidirectional parallel communications between a host and a peripheral. The IEEE 1284 port supports all IEEE 1284 modes of communication (except EPP mode) with appropriate software control. The port has three additional nonIEEE 1284 compliant modes to support transport stream output modes and allows software control of the port.

Data may be accessed and sourced either from internal registers or by a DMA transfer. DMA transfers are used where appropriate to increase throughput and decrease system load. DMA transfers are not word aligned and may transfer between 1 and 65535 bytes. The DMA may only operate in one direction at any one time.

An interrupt mechanism is used to indicate that the port has completed a transfer or has an event which needs servicing.

The IEEE 1284 port supports the following main modes of operation:

- IEEE 1284 mode,
- transport stream mode,
- software control mode.

Each of these modes has associated modes. The modes are described in later sections of this chapter.

Note: *The port only functions as a 1284 slave mode device in IEEE 1284 mode. The port then expects the other device to be a master mode device (that is, a PC host), rather than another slave mode device, such as a printer. It is possible, however to configure the port to be a master mode device, but this can only be used in software control mode (see [Section 70.4.7: Software mode on page 664](#)).*

70.1.1 Powering on, initialization and termination

The interface may be re-initialized at any time by the host; this produces an interrupt for the peripheral to respond to. The slave may request to terminate a communication, or request to interrupt the master, but waits for acknowledgment when operating in IEEE 1284 mode.

1. IEEE Standard 1284-1994: IEEE Standard Signalling method for a Bidirectional Parallel Peripheral Interface for Personal Computers.

70.2 IEEE 1284 port pins

The port pins meet IEEE 1284 level 2 device requirements and are designed to directly drive an IEEE 1284 compliant cable with external matching resistors. [Figure 206](#) shows how to interface the module to the cable.

Figure 206: Interface to a cable

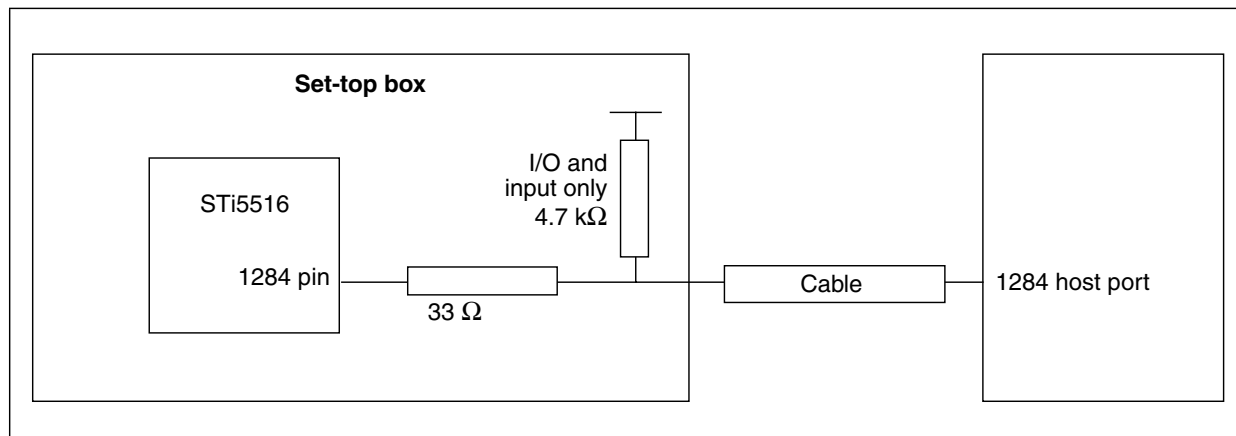


Table 197: IEEE 1284 port pins

Pin	In/Out	Function
1284DATA[7:0]	In/out	IEEE 1284 serial data.
1284NOTSELECTIN	In	The function of these control pins is dependent on the mode of operation of the IEEE 1284 port, as shown in Table 198 .
1284NOTINIT	In	
1284NOTFAULT	Out	
1284NOTAUTOFD	In	
1284SELECT	Out	
1284PERROR/TSPYTECLKVALID	Out	
1284BUSY/TSPACKETCLK	Out	
1284NOTACK/TSPYTECLK	Out	
1284NOTSTROBE	In	
1284INNOTOUT (PIO3[7])	Out ^a	IEEE 1284 data output enable for an external buffer.
1284PERIPHLOGICH (PIO4[3])	Out ^a	Peripheral logic high.
1284HOSTLOGICH (PIO4[4])	In ^a	Host logic high.

a. These pins have the PIO pin electrical characteristics and timings.

The nine control pins have different functions depending on the mode of operation of the port interface. The mapping of the IEEE 1284 port pins to the function of the pin in a specific mode is given in [Table 198](#). For full details of the IEEE 1284 signal functions in each mode refer to the *IEEE Standard 1284-1994*.

Table 198: IEEE 1284 port control pin functions

Pin	IEEE 1284 modes				Transport stream mode
	Compatible mode	Nibble mode	Byte mode	ECP mode	
1284NOTSTROBE	NSTROBE	HOSTCLK	HOSTCLK	HOSTCLK	
1284NOTACK	NACK	PTRCLK	PTRCLK	PERIPHCLK	TSBYTECLK
1284BUSY	BUSY	PTRBUSY	PTRBUSY	PERIPHACK	TSPACKET CLK
1284PERROR	PERROR	ACKDATAREQ	ACKDATAREQ	NACKREVERSE	TSBYTECLK VALID
1284SELECT	SELECT	XFLAG	XFLAG	XFLAG	
1284NOTAUTOFD	NAUTOFD	HOSTBUSY	HOSTBUSY	HOSTACK	
1284NOTINIT	NINIT	HIGH	HIGH	NREVERSE REQUEST	
1284NOTFAULT	NFAULT	NATAAVAIL	NATAAVAIL	NPERIPH REQUEST	
1284NOTSELECTIN	NSELECTIN	ACTIVE	ACTIVE	ACTIVE	

70.3 IEEE 1284 mode

The IEEE 1284 port supports IEEE 1284 modes of communication, as defined below, with appropriate software control and use of DMA transfers where appropriate to increase throughput and decrease system load. For full details of the IEEE 1284 protocols and signal functions in each mode refer to the *IEEE Standard 1284-1994*.

Forward transfer means a transfer from the host to the peripheral; reverse transfer means a transfer from the peripheral to the host.

70.3.1 IEEE 1284 mode initialization

The 1284MODEENABLE, 1284PULSEWIDTH and 1284PINOUT registers must be set before entering any IEEE 1284 mode. The 1284PERIPHLOGICH pin is forced high in all IEEE 1284 modes.

On entering the IEEE 1284 modes, the peripheral always completes an initialization sequence before starting in compatibility mode. If the OVERRIDEHOSTLOGICH bit in the 1284CONTROL register is not set then the part remains in this mode until the 1284HOSTLOGICH pin goes high.

It is the responsibility of the software driver to ensure that the 1284PERIPHLOGICH pin setting is correct before entering the IEEE 1284 modes.

The status of the peripheral is indicated to the host using the values in the 1284PINOUT register. If the BUSY bit is high then the peripheral is busy on entering compatible mode and the values of the 1284PERROR, 1284SELECT and 1284NOTFAULT pins reflect the values in the 1284PINOUT register. The value of the 1284BUSY and 1284NOTACK pins are not under user control.

70.3.2 Compatibility mode

Compatibility mode supports forward transfers only. This mode is comparable to the centronics parallel port (CPP).

Compatibility mode is always enabled when IEEE 1284 mode is enabled. Following initialization, or reset by either the host or the peripheral, the port operates in this mode until the host negotiation allows the port to move to another mode. Following any protocol exceptions or termination requests the module returns to this mode.

The busy status of the peripheral in this mode is controlled by the BUSY bit of the 1284PINOUT register. The peripheral becomes busy when a transfer occurs, or when the BUSY bit of the 1284PINOUT register is set. If the BUSY bit is high, the 1284PERERROR, 1284SELECT and 1284NOTFAULT pins are driven to the value given in the 1284PINOUT register. The value of the 1284NOTACK pin is not under user control.

70.3.3 Negotiation

The host may request that the IEEE 1284 compliant device change communication mode. This is done by placing an extensibility request on the data bus during negotiation mode. Negotiation may only be entered from compatibility mode. A negative response to a request stalls the port until the host terminates the transaction, and returns the port to compatibility mode.

The modes to which the module responds positively depends on the specific implementation and the 1284MODEENABLE register.

On entering this mode the 1284BUSY pin assumes the value in the 1284PINOUT register. The control of the other pins is dependent on the mode being entered, and whether data is available to be transferred.

70.3.4 Nibble mode

Nibble mode supports reverse transfers only. This is the most basic reverse transfer mode and is used as the reverse channel in conjunction with compatible mode. Nibble mode is always enabled if IEEE 1284 mode is enabled.

The data is transferred as 4-bit values on four of the IEEE 1284 control pins: 1284PERERROR, 1284BUSY, 1284NOTFAULT, 1284SELECT. The 1284BUSY pin reflects the value in the 1284PINOUT register or the data value depending on the point in the transfer. The other pins are not under user control.

70.3.5 Byte mode

Byte mode supports reverse transfers only. It uses a similar protocol to nibble mode, but transfers the data as 8-bit values on the data bus (1284DATA[7:0]).

The 1284BUSY pin reflects the value in the 1284PINOUT register. The other pins are not under user control.

70.3.6 ECP mode

ECP mode supports both forward and reverse transfers. The port supports run length encoding (RLE). The hardware allows access to channel and RLE data, and software support is provided. Expansion of incoming data using RLE encoding is supported in hardware and enabled using the 1284CONTROL register. All output RLE encoded data must be pre-encoded.

If channel or RLE information is passed to the DMA engines, a DMA error occurs.

The 1284NOTFAULT pin reflects the value in the 1284PINOUT register and is expected to be used to trigger a host interrupt.

For the case when the IEEE 1284 port is not busy and a forward transfer is occurring, then the peripheral should ensure that when a token becomes available it is accepted from the IEEE 1284 port within 35 ms. Failure to do so may cause the host to signal a time-out error. If hardware RLE decode is enabled, the application should ensure that a complete decoded RLE sequence is accepted within 35 ms.

The maximum RLE sequence length allowed by the IEEE 1284 standard is 128 bytes. The tokens may be accepted by either the DMA or register transfers.

70.3.7 Device identification

The peripheral asserts an interrupt to indicate a device identification request has occurred. Software handles this request and return the device identification data stream. The protocol used to return the identification stream depends on the 1284MODEENABLE register.

70.3.8 Host reset

The interface may be re-initialized at any time by the host. This produces an interrupt for the peripheral to respond to. The slave may request to terminate a communication, or request to interrupt the master, but waits for acknowledgment when operating in IEEE 1284 mode.

70.3.9 Termination

Following termination of a mode by the host, the peripheral always returns to compatible mode. The behavior of the 1284PERROW, 1284NOTFAULT and 1284SELECT pins is dependent on the value in the 1284PINOUT register, and reflects the value in this register if the BUSY bit is set. If the BUSY bit of the 1284PINOUT register is set, the peripheral is busy on entering compatible mode. The peripheral sets the value of the 1284BUSY pin.

An IEEE termination device is required for boards that use the STi5516, so that correct termination is provided when the STi5516 is configured as either a master or slave.

70.3.10 Data transfer rates

The data transfer rate in these modes is dependent on the host, operating mode and memory speed, and is expected to be limited by the host response time.

The DMA engine implements eight bytes of buffering for outgoing data, and four for incoming data.

70.4 Transport stream mode

The transport stream interface produces a byte wide output data stream compatible with the Link-IC protocol. The two alternate implementations of this output stream are defined below. The number of null byte transfers must be controlled by the driver software.

The following sections describe the pin and register functionality of the IEEE 1284 port in transport stream mode.

The value of the 1284PULSEWIDTH, 1284PINOUT and 1284PACKETSIZE registers must be set before entering transport mode.

70.4.1 TSBYTECLK

The data (1284DATA[7:0]), TSPACKETCLK and TSBYTECLKVALID are valid on the rising edge of this signal. The data, TSPACKETCLK and TSBYTECLKVALID change on the falling edge of this clock. The clock is active when a valid data token is available on the data bus.

The minimum frequency of the byte clock depends on the value held in the 1284PULSEWIDTH register. This gives the delay in clock cycles between byte clock edge transitions. At 40 MHz, a value of 2 in this register produces a byte clock with a nominal 100 ns period.

In transport stream mode A, the byte clock is free running and TSBYTECLKVALID going high indicates that the clock is active. The frequency of the clock is fixed, and in the case of memory stalls, TSBYTECLKVALID going low indicates there is no data packet to transmit.

In transport stream mode B, a rising transition only occurs on this clock when valid information is available to transmit. The frequency of the clock may change in the event of a memory stall. At the end of a packet transfer the clock becomes free running until the next packet transfer is started.

70.4.2 TSBYTECLKVALID

This validates the byte clock and indicates that the TSBYTECLK transition is valid.

70.4.3 TSPACKETCLK

This is high during a packet transfer. The length of a packet is defined by the 1284PACKETSIZE register. A packet transfer commences when valid data has been read from memory and is available on the data bus. It completes when the number of bytes defined by the 1284PACKETSIZE register have been transferred.

70.4.4 Packet size register

A write to the 1284PACKETSIZE register defines the number of bytes within a packet.

The IEEE 1284 packet size count is restarted after the required number of bytes have been transferred, and if a DMA transfer of greater than one packet is started, the second packet is transferred with a single null byte between packets. If a DMA transfer transfers an incomplete packet, the module stalls until more bytes become available.

The count may be restarted by writing to the RESET bit in the 1284CONTROL register or by writing to the 1284PACKETSIZE register.

70.4.5 Transfer stream mode A and B examples

Figure 207 and Figure 208 give examples of a single packet transfer in transport stream modes A and B respectively. The number of null bytes depends on the time taken to start a second DMA transfer.

Figure 207: Packet transfer in transport stream mode A

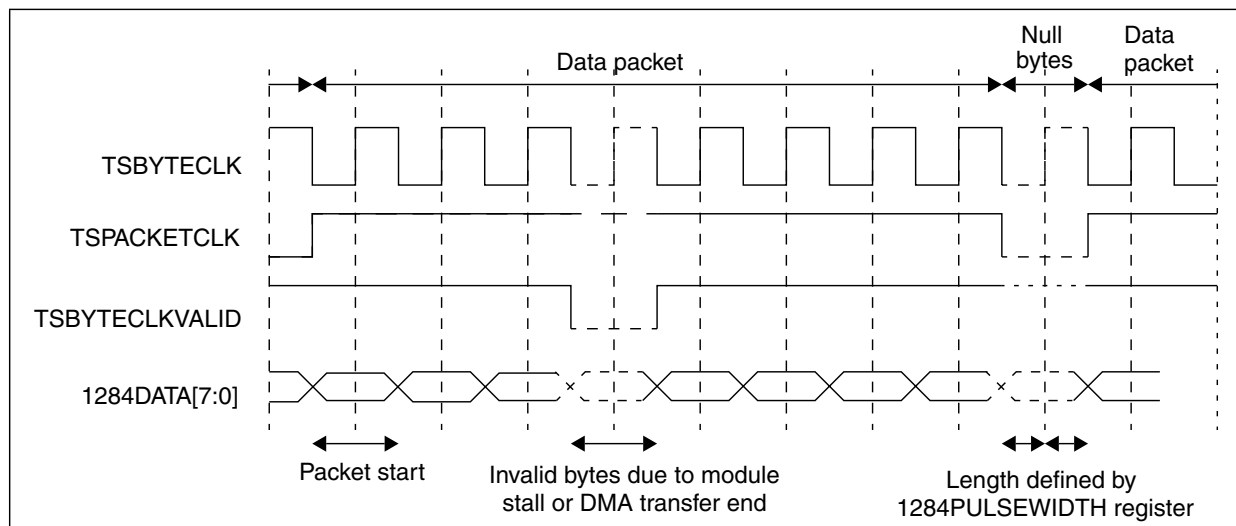
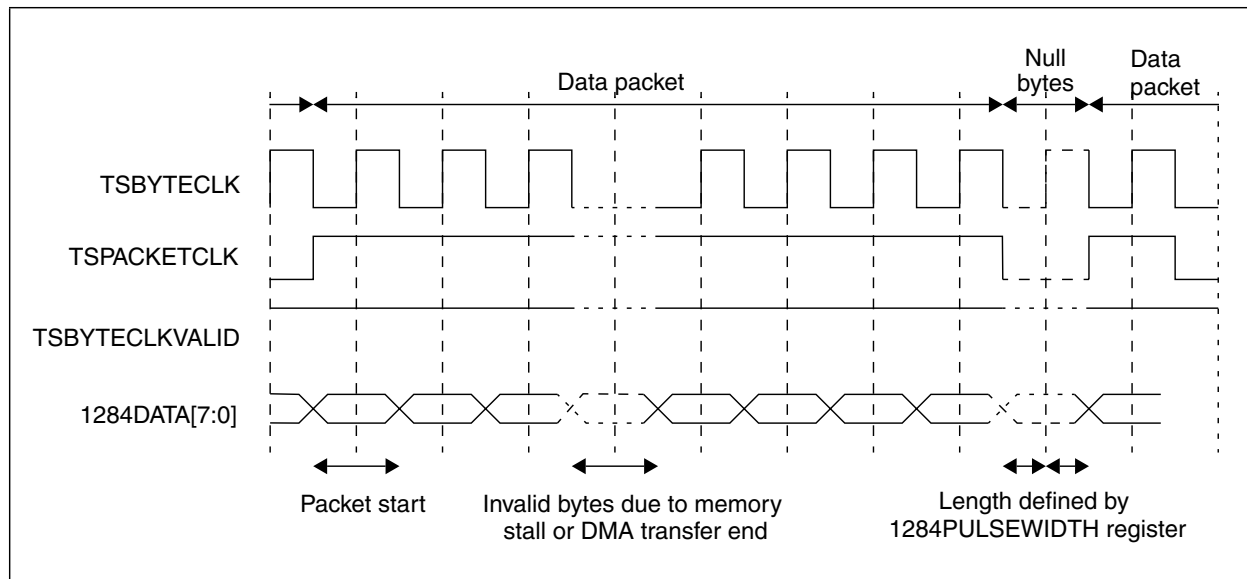


Figure 208: Packet transfer in transport stream mode B



70.4.6 Data rates

In transport mode the data throughput is a function of the memory speed, the byte clock rate and the packet size. Assuming an average memory speed of twelve cycles, a sustained data rate of 8 Mbytes/s can be maintained for word-aligned accesses for large packets.

70.4.7 Software mode

Software mode supports direct software control of the IEEE 1284 port, via the relevant control registers.

The peripheral may set the value of the output pins, control the value and direction of the data bus, and examine the input pins.

Interrupts may be set to occur if the input pins fail to match a pattern.

Data tokens may be transferred to and from the DMA engines.

In software mode the IEEE 1284 port can be configured to be a 1284 master mode device.

70.4.8 1284 master mode

This mode can only be used in software mode, as in [Section 70.4.7](#). The port can be programmed to be in master mode by writing 1 to bit 15 of the EMI_CONFIGPADLOGIC register. In this mode the function and direction of the IEEE 1284 port control signals are changed to support master mode. This is shown in [Table 199](#). The control signal pins must be directly controlled by software. The input control signals are input by reading the 1284PININ register and the output control signals are output by writing to the 1284PINOUT register.

Table 199: 1284 master mode control signals

Slave mode (default) pin name and function	Pin direction	Master mode pin name and function	Pin direction
1284NOTSELECTIN	I	1284NOTFAULT	O
1284NOTFAULT	O	1284NOTSELECTIN	I
1284NOTINIT	I	1284SELECT	O
1284SELECT	O	1284NOTINIT	I
1284NOTAUTOFD	I	1284PERROW	O
1284PERROW	O	1284NOTAUTOFD	I
1284HOSTLOGICH	I/O (PIO pin input)	1284BUSY	O
1284BUSY	O	1284HOSTLOGICH	I
1284NOTSTROBE	I	1284NOTACK	O
1284NOTACK	O	1284NOTSTROBE	I

70.4.9 Signal filtering

All IEEE 1284 control inputs (that is, all inputs except the data bus) have a digital filter to remove signal glitches less than the comms clock period.

71 IEEE 1284 port (PC parallel port) registers

All enables are active high unless otherwise stated.

Addresses are provided as the *1284BaseAddress* + offset.

The *1284BaseAddresses* are:

0x2012 5000.

A register summary is given in [Table 39: IEEE 1284 port registers on page 65](#).

1284CHECKSUM

1284 check sum

7 6 5 4 3 2 1 0

CHECKSUM

Address: *1284BaseAddress* + 0x28

Type: Read only

Reset: 0

Description: The 1284CHECKSUM register contains a checksum of all bytes transmitted or received by the IEEE 1284. A read from this register causes it to be reset. The checksum is calculated by the accumulative bitwise XOR of each bit in the byte passing through the IEEE 1284 with the previous checksum value.
This register is not defined in transport modes for a single cycle pulse width.
This function is not part of the IEEE 1284 Standard, and is an addition to allow rapid checksum calculation in a specific application.

Confidential

1284CONTROL**1284 control**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							VERRIDEHOSTLOGICH	RESET	EXTBUSDIRECTION	HWINPUTRLEEXPAN	Reserved			MODE	

Address: $1284BaseAddress + 0x08$

Type: Write only

Reset: 0

Description: The 1284CONTROL register controls the operating mode of the IEEE 1284 port. Setting the RESET bit forces all machines back to the idle status, discarding any stored data. This may cause protocol errors and loss of data. This is a functional synchronous reset, and returns the module to the initialization state in the enabled mode. This only resets the IEEE 1284 module, and not the DMA engines. If any IEEE 1284 mode or the transport stream mode is disabled during a transaction, the mode is disabled next time the controlling state machine reaches idle, after completing any ongoing transactions. In transport stream modes, the current package is completed before returning to idle. In IEEE 1284 mode it waits until returning to compatible mode. If hardware enables RLE expansion mode during a transaction (bit 5), the change takes effect next time the action associated with that transaction occurs.

Note: *When operating in IEEE 1284 mode, the point at which the IEEE 1284 port returns to idle is controlled by the host and therefore may be an unbounded period of time.*

[15:9] **Reserved**

[8] **VERRIDEHOSTLOGICH**

When set to 1, the 1284HOSTLOGICH input signal is forced high, so when operating in any IEEE 1284 mode the IEEE 1284 module always assumes that the input signals from the host are valid.

[7] **RESET**

When set to 1, the IEEE 1284 port is reset, and any stored data is discarded.

[6] **EXTBUSDIRECTION**

Enable external bus direction control.

[5] **HWINPUTRLEEXPAN**

Enable hardware input RLE expansion.

[4:3] **Reserved**

Write 0.

[2:0] **MODE**

IEEE 1284 port operating mode:

000: Software mode

001: IEEE 1284 mode

010: Transport stream mode A

011: Transport stream mode B

1284DATAIN**1284 data input**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CONTROL/ADDRESS	CONTROL/DATA[7]	DATA[6:0]						

Address: $1284BaseAddress + 0x20$

Type: Read only

Reset: 0

Description: When in any IEEE 1284 mode, a read from the 1284DATAIN register reads the input data token stored in the IEEE 1284 module. In other modes it reflects the data currently on the input data pins (1284DATA[7:0]).

This data is valid and available only when the INPUTDATAREADY bit is set high in the 1284STATUS register, see *1284STATUS* on page 678. If data is available, then reading this register removes the data token from the IEEE 1284 port and allows the next access sequence to proceed.

Reading from the register during a DMA transfer interrupts that transfer sequence, and may invalidate the DMA transfer.

Bit 8 is valid only in ECP mode and indicates byte packet type, in all other modes it is undefined. The type of an incoming data package is mode dependent.

[15:9] **Reserved**

[8] **CONTROL/ADDRESS**

Byte packet type

0: Data

1: Control packet in ECP mode

[7] **CONTROL/DATA[7]:**

ECP mode - Control packet type

1: Channel number packet

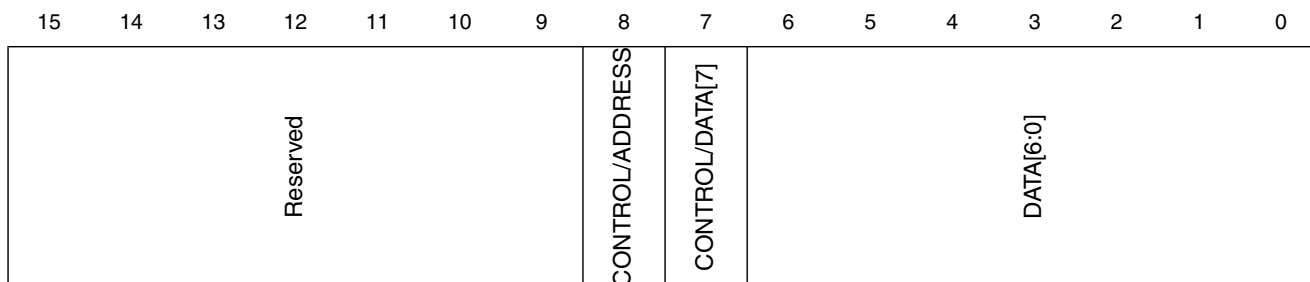
0: RLE count packet

Any other mode - Data currently on the data pin (1284DATA7)

[6:0] **DATA[6:0]:**

Any IEEE 1284 mode - Input data token stored in the IEEE 1284 module

Any other mode - Data currently on the data pins (1284DATA[6:0])

1284DATAOUT**1284 data output**

Address: $1284BaseAddress + 0x24$

Type: Write only

Reset: 0

Description: A write to the 1284DATAOUT register writes a data token to the IEEE 1284 module. If a write to this register occurs in IEEE 1284 mode or transport stream mode, and the OUTPUTDATAREADY bit of the 1284STATUS register is set high, then a data token is transferred to the IEEE 1284 port and the access sequence started. In other modes it reflects the value on the data pins if the data bus is being driven. Bit 8 is valid in ECP mode only and controls the type of access triggered. Writing to this register during a DMA access interrupts the access sequence and may invalidate the DMA transfer.

[15:9] **Reserved**

[8] **CONTROL/ADDRESS**

Byte packet type

0: Data

1: Control packet in ECP mode

[7] **CONTROL/DATA[7]**

ECP mode - Control packet type:

1: Channel number packet

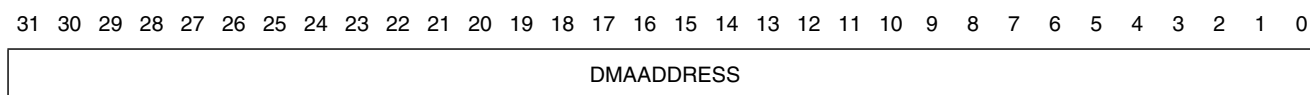
0: RLE count packet.

Any other mode - Data currently on the data pin (1284DATA7)

[6:0] **DATA[6:0]**

Any IEEE 1284 mode or transport mode - Output data token stored in the IEEE 1284 module

Any other mode - Data currently on the data pins (1284DATA[6:0])

1284DMAADDRESS**1284 DMA address**

Address: $1284BaseAddress + 0x40$

Type: Write only

Reset: 0

Description: This defines the byte address from which the DMA starts. Following the completion of a DMA access, this register points to the next location in memory. This register is undefined following system reset.

1284DMACONTROL**1284 DMA control**

7	6	5	4	3	2	1	0
Reserved					DMARESET	DMASTALL	DMADIRECTION

Address: $1284BaseAddress + 0x48$

Type: Read/write

Reset: 0

Description: The 1284DMACONTROL register controls the DMA transfer. The direction of the DMA access, either from memory to the IEEE 1284 port or vice versa is specified by the DMADIRECTION bit. Setting the DMARESET bit terminates the DMA transfer. Buffered incoming data is written to memory. Stored outgoing data is lost. The 1284DMACOUNT register shows the number of bytes successfully transferred before reset occurred. When reset is complete, the DMARESET bit is set to zero. If the DMA engine is reset while a DMA output is occurring and the byte transfer is in progress on the IEEE 1284 port, the byte transfer may be corrupted, or the host left in a position of expecting data to be transferred. This byte is not included in the DMA count. DMA reset is only expected to be used to clear the DMA engines in exceptional conditions (such as errors), at which point the interface is stalled, or by stalling the DMA engines for long enough for all buffered tokens to be removed. Setting the DMASTALL bit stops the DMA transfer. The 1284DMACOUNT register shows the total number of bytes remaining to be transferred. Resetting the DMASTALL bit allows the DMA transfer to continue.

[7:3] **Reserved**

[2] **DMARESET**: Terminates the DMA transfer

[1] **DMASTALL**: Stalls the DMA transfer

[0] **DMADIRECTION**: Direction of the DMA access:

0: From IEEE 1284 port to memory

1: From memory to IEEE 1284 port

1284DMACOUNT**1284 DMA count**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMACOUNT															

Address: $1284BaseAddress + 0x44$

Type: Read/write

Reset: Undefined

Description: In the event of a DMA error, or other exception, the 1284DMACOUNT register contains the number of bytes which remain to be transferred.

Writing to this register starts a new DMA sequence, starting from the address given in the 1284DMAADDRESS register, for the number of bytes written to this location.

Reading from this register gives the number of bytes remaining to be transferred. This value is constant only when the DMA engine has been stalled or reset. A value of zero in this register indicates that the DMA transfer has completed transfers to or from memory. If a zero is written to this register, a memory access may occur, but no data is transferred.

This register is undefined following system reset.

1284DMATOKEN**1284 DMA token**

7	6	5	4	3	2	1	0
Reserved						TOKENFROMDMA	TOKENTODMA

Address: *1284BaseAddress* + 0x30

Type: Read/write

Reset: 0

Description: This register allows the DMA engines to be used when driving the IEEE 1284 port directly in software mode. The register is valid only when software mode is enabled, writes to this register in other modes are undefined.

A read from this register indicates whether the token has been successfully transferred to the DMA engine. If the bit is high then the memory system has not yet accepted the token, if the bit is zero it indicates that it has accepted the token.

The token transfer only occurs if the direction of the DMA engine corresponds to the token transfer direction.

The DMA engine may assemble each byte token in word packets before writing the token to memory.

Writing 1 to bit 0 transfers a data token to the DMA engine from the data pins. Writing 1 to bit 1 transfers a data token to the data pins from the DMA engine.

[7:2] **Reserved**

[1] **TOKENFROMDMA**: Data token transfer from the DMA engine to the data pins

[0] **TOKENTODMA**: Data token transfer from the data pins to the DMA engine

1284INTACK**1284 interrupt acknowledge**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RESETACK	ERRORACK	REQUESTACK	DIRECCHANGEACK	MODECHANGEACK	PININTACK	DMAERRORACK	DMACOMLETEACK	INPUTAVAILACK	OUTPUTAVAILACK

Address: $1284BaseAddress + 0x54$

Type: Write only

Reset: 0

Description: Writing 1 to a bit in this register explicitly clears the associated active interrupt. The bits with their functions marked as not applicable in the table below reference interrupts which are implicitly cleared by completing the action associated with the interrupt. An explicit reset clears these bits, but the interrupt is immediately re-asserted if the triggering condition is still true.

The IEEE 1284 input and output interrupts are cleared when the associated data token is transferred. The DMA error and DMA complete interrupts can be cleared by resetting/restarting the DMA engine. A IEEE 1284 request is cleared by outputting a data token.

[15:10] **Reserved**

[9] **RESETACK**

1: The associated interrupt is cleared.

[8] **ERRORACK**

1: The associated interrupt is cleared.

[7] **REQUESTACK**

Not applicable

[6] **DIRECCHANGEACK**

1: The associated interrupt is cleared.

[5] **MODECHANGEACK**

1: The associated interrupt is cleared.

[4] **PININTACK**

Not applicable

[3] **DMAERRORACK**

Not applicable

[2] **DMACOMLETEACK**

Not applicable

[1] **INPUTAVAILACK**

Not applicable

[0] **OUTPUTAVAILACK**

Not applicable

1284INTENABLE**1284 interrupt enable**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RESETEN	ERROREN	REQUESTEN	DIRECCHANGEEN	MODECHANGEEN	PININTEN	DMAERROREN	DMACOMLETEEN	INPUTAVAILEN	OUTPUTAVAILEN

Address: $1284BaseAddress + 0x4C$

Type: Read/write

Reset: 0

Description: The 1284INTENABLE register determines whether an interrupt is enabled. If the bit relating to the interrupt is set, then if that event occurs, an interrupt is generated.

A DMA error occurs if a nondata packet (an RLE count, channel number or an address value) is passed during a DMA transfer. The DMA sequence stalls at this point. The DMA engine must then be reset to flush valid buffered incoming bytes to memory. The erroneous data token can be accessed directly and removed from the IEEE 1284 module by reading the 1284DATAIN register (see [1284DATAIN on page 668](#)). Outgoing data tokens are not checked. The DMA access can also be stalled by a number of events such as mode and direction changes, protocol errors and IEEE 1284 requests. These events can be monitored and treated as a DMA error if the events are seen, by explicitly resetting the DMA engines, which flushes buffered valid bytes to memory, leaving the engines in the same state as a DMA error following a reset.

[15:10] **Reserved**

[9] **RESETEN**

1: An interrupt is generated if the host system resets the IEEE 1284 port.

[8] **ERROREN**

1: An interrupt is generated if a protocol error is detected by the IEEE 1284 port.

[7] **REQUESTEN**

1: An interrupt is generated if a device id request is made.

[6] **DIRECCHANGEEN**

1: An interrupt is generated if the transfer direction of the IEEE 1284 port changes. The current transfer direction can be read from the 1284STATUS register.

[5] **MODECHANGEEN**

1: An interrupt is generated if the IEEE 1284 port changes mode. The operational modes are specified in the 1284STATUS register. Additional information on the direction of the bidirectional modes is also available in the 1284STATUS register or can be interpreted from bits 1:0 of the 1284INTSTATUS register.

[4] **PININTEN**

1: An interrupt is generated when the enabled (1284PININENABLE register) IEEE 1284 input pins fail to match the pattern in the 1284PININVALUE register. The value of the input pins can be read from the 1284PININ register, see [1284PININ on page 676](#).

[3] **DMAERROREN**

1: An interrupt is generated if a nondata packet (an RLE count, channel number or address value) is passed by the IEEE 1284 port during a DMA transfer.

[2] **DMACOMLETEEN**

1: An interrupt is generated when a DMA transfer is completed and all tokens have been transferred to and from the IEEE 1284 port.

[1] **INPUTAVAILEN**

1: An interrupt is generated when a IEEE 1284 input byte is available.

[0] **OUTPUTAVAILEN**

1: An interrupt is generated when the IEEE 1284 output is clear and available.

1284INTSTATUS**1284 interrupt status**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RESET	ERROR	REQUEST	DIRECCHANGE	MODECHANGE	PININT	DMAERROR	DMACOMPLETE	INPUTAVAIL	OUTPUTAVAIL

Address: $1284BaseAddress + 0x50$

Type: Read only

Reset: 0

Description: This register gives the identity of the event which caused the interrupt. It may also be read to monitor the status of masked interrupts.

[15:10] **Reserved**

[9] **RESET**

1: Indicates reset interrupt was generated.

[8] **ERROR**

1: Indicates error interrupt was generated.

[7] **REQUEST**

1: Indicates request interrupt was generated.

[6] **DIRECCHANGE**

1: Indicates direction change interrupt was generated.

[5] **MODECHANGE**

1: Indicates mode change interrupt was generated.

[4] **PININT**

1: Indicates input pin interrupt was generated.

[3] **DMAERROR**

1: Indicates DMA error interrupt was generated.

[2] **DMACOMPLETE**

1: Indicates DMA complete interrupt was generated.

[1] **INPUTAVAIL**

1: Indicates IEEE 1284 input byte available interrupt was generated.

[0] **OUTPUTAVAIL**

1: Indicates IEEE 1284 output clear and available interrupt was generated.

1284MODEENABLE**1284 mode enable**

7	6	5	4	3	2	1	0
Reserved	ENRLE	ENECP	Reserved	ENDEVID	Reserved	ENBYTE	

Address: $1284BaseAddress + 0x00$

Type: Write only

Reset: 0

Description: The 1284MODEENABLE register bits are a direct mask of the IEEE 1284 extensibility request values. If a bit corresponding to the mode is set low, the peripheral is refused access to enter the mode when operating in IEEE 1284 mode. If all the bits are disabled the device operates in the nibble and compatible modes.

If the register is modified, the change takes effect at the next IEEE 1284 negotiation transaction.

This register is only valid when IEEE 1284 mode is enabled.

[7:6] **Reserved**

Write 0

[5] **ENRLE**: Enable RLE

[4] **ENECP**: Enable ECP

[3] **Reserved**

Write 0

[2] **ENDEVID**: Enable device identification

[1] **Reserved**

Write 0

[0] **ENBYTE**: Enable byte

1284PACKETSIZE**1284 packet size**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PACKETSIZE														

Address: $1284BaseAddress + 0x2C$

Type: Write only

Reset: 0

Description: This register contains the packet size during a transport stream transfer. It is valid only when transport mode is enabled.

Confidential

1284PININ**1284 pin input**

7	6	5	4	3	2	1	0
Reserved			HOSTLOGICH	NOTSELECTIN	NOTINIT	NOTAUTOFD	NOTSTROBE

Address: $1284BaseAddress + 0x10$

Type: Read only

Reset: 0

Description: This register reflects the current status of the input pins in all modes. The value read is the value on the pins when the request is granted.

[7:5] **Reserved**

[4] **HOSTLOGICH**: 1284HOSTLOGICH pin status

[3] **NOTSELECTIN**: 1284NOTSELECTIN pin status

[2] **NOTINIT**: 1284NOTINIT pin status

[1] **NOTAUTOFD**: 1284NOTAUTOFD pin status

[0] **NOTSTROBE**: 1284NOTSTROBE pin status

1284PININENABLE**1284 pin input enable**

7	6	5	4	3	2	1	0
Reserved			HOSTLOGICHINTEN	NOTSELECTININTEN	NOTINITINTEN	NOTAUTOFDINTEN	NOTSTROBEINTEN

Address: $1284BaseAddress + 0x14$

Type: Read/write

Reset: 0

Description: This register enables generation of an interrupt based on the values contained in the 1284PININVALUE register.

[7:5] **Reserved**

[4] **HOSTLOGICHINTEN**

1: Enables generation of an interrupt if the associated value given in the 1284PININVALUE register does not match the 1284HOSTLOGICH input pin setting.

[3] **NOTSELECTININTEN**

1: Enables generation of an interrupt if the associated value given in the 1284PININVALUE register does not match the 1284NOTSELECT input pin setting.

[2] **NOTINITINTEN**

1: Enables generation of an interrupt if the associated value given in the 1284PININVALUE register does not match the 1284NOTINIT input pin setting.

[1] **NOTAUTOFDINTEN**

1: Enables generation of an interrupt if the associated value given in the 1284PININVALUE register does not match the 1284NOTAUTOFD input pin setting.

[0] **NOTSTROBEINTEN**

1: Enables generation of an interrupt if the associated value given in the 1284PININVALUE register does not match the 1284NOTSTROBE input pin setting.

1284PININVALUE**1284 pin input value**

7	6	5	4	3	2	1	0
Reserved			HOSTLOGICHINTVAL	NOTSELECTININTVAL	NOTINITINTVAL	NOTAUTOFDINTVAL	NOTSTROBEINTVAL

Address: $1284BaseAddress + 0x18$

Type: Read/write

Reset: 0

Description: This register holds the value against which the input pins are compared. An interrupt is generated for any differences found, if the corresponding bit in register 1284PININENABLE is set. The compare function is level sensitive, and any change in input must be held for longer than the interrupt response time to be seen as a valid interrupt. Following system reset this register is undefined.

[7:5] **Reserved**

[4] **HOSTLOGICHINTVAL**: Value to which the 1284HOSTLOGICH input pin setting is compared.

[3] **NOTSELECTININTVAL**: Value to which the 1284NOTSELECT input pin setting is compared.

[2] **NOTINITINTVAL**: Value to which the 1284NOTINIT input pin setting is compared.

[1] **NOTAUTOFDINTVAL**: Value to which the 1284NOTAUTOFD input pin setting is compared.

[0] **NOTSTROBEINTVAL**: Value to which the 1284NOTSTROBE input pin setting is compared.

1284PINOUT**1284 pin output**

7	6	5	4	3	2	1	0
Reserved	DATABUSENABLE	PERIPHLOGICH	BUSY	NOTACK	PERROR	SELECT	NOTFAULT

Address: $1284BaseAddress + 0x1C$

Type: Read/write

Reset: 0

Description: The bus direction pin signals are only under the control of this register when not controlled by the IEEE 1284 or transport mode state machine. All pins are under user control when the device is operating in software mode. A read from this register gives the current value of the output pin/bus direction. If the pin is not under user control this may not be the value written to this register, but reflects the current value on the output pins. If DATABUSENABLE is set low, the data bus is high impedance and may be driven by the host.

[7] **Reserved**

[6] **DATABUSENABLE**: 1284OUT output pin setting.

[5] **PERIPHLOGICH**: 1284PERIPHLOGICH output pin setting.

[4] **BUSY**: 1284BUSY output pin setting.

[3] **NOTACK**: 1284NOTACK output pin setting.

[2] **PERROR**: 1284PERROR output pin setting.

[1] **SELECT**: 1284SELECT output pin setting.

[0] **NOTFAULT**: 1284NOTFAULT output pin setting.

1284PULSEWIDTH**1284 pulse width**

7	6	5	4	3	2	1	0
CLOCKCYCLES							

Address: $1284BaseAddress + 0x04$

Type: Write only

Reset: Undefined

Description: In IEEE 1284 mode, the 1284PULSEWIDTH register specifies the time period (Tp) in number of comms clock cycles. Tp is defined in the IEEE 1284 spec as the minimum setup or pulse width for IEEE 1284 handshake.s. For the STi5516 comms clock running at 40 MHz this value must be 20 comms clock cycles minimum in order to comply with the IEEE 1284 minimum time period of 500 ns. This register must only be written when transport and IEEE 1284 modes are disabled.
In transport stream mode, this register specifies the minimum period between byte clock (TSBYTECLK) edge transitions. For details see [Section 70.4: Transport stream mode on page 662](#).
A write to this register takes effect at the next byte transfer.

1284STATUS**1284 status**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DATATRANSFERDIR	OPMODE				ENABLERLEEXT	1284REQUEST	INPUTDATAREADY	OUTPUTDATAREADY

Address: $1284BaseAddress + 0x0C$

Type: Read only

Reset: 0

Description: The 1284STATUS register gives the current status of the IEEE 1284 module. Bit 3 is valid in ECP mode. It indicates that RLE expansion has been enabled. If the hardware expansion is not enabled then software is required to expand the byte stream. Following system reset the IEEE 1284 module starts in software mode. A 1284REQUEST is cleared when the next data token is transferred to the IEEE 1284 module.

[15:9] **Reserved**

[8] **DATATRANSFERDIR:** Data transfer direction

0: From host to IEEE 1284 module

1: From IEEE 1284 module to host

[7:4] **OPMODE:** Operational mode. Valid values are as follows:

0000: IEEE 1284 mode: initialization

0001: IEEE 1284 mode: compatible

0010: IEEE 1284 mode: negotiation

0011: IEEE 1284 mode: nibble

0100: IEEE 1284 mode: byte

0101: IEEE 1284 mode: ECP

0111: IEEE 1284 mode: terminate

1000: software mode (peripheral control of IEEE 1284 port)

1001: Transport stream mode A

1010: Transport stream mode B

[3] **ENABLERLEEXT:** RLE extensions enabled - in ECP mode

[2] **1284REQUEST:** Device id request.

[1] **INPUTDATAREADY:** Input byte available.

Data from the host is available in the IEEE 1284 module input buffer.

[0] **OUTPUTDATAREADY:** Output clear and available.

The IEEE 1284 module is ready to output data to the host.

Confidential

72 Electrical specifications

72.1 Absolute maximum ratings

Table 200: Absolute maximum ratings

Symbol	Parameter	Min	Max	Units
VDD3 _{MAX}	VDD3 3.3 V DC supply voltage		4.6	V
VDD18 _{MAX}	VDD18 1.8 V DC supply voltage		2.5	V
VI18 _{MAX}	DC Voltage on input and bi-directional 1.8V tolerant pins	GND - 0.5	VDD18 + 0.5	V
VI5 _{MAX}	DC Voltage on input, output and bi-directional 5 V tolerant pins	GND - 0.5 ^a	5.5 ^b	V
VO3 _{MAX}	DC Voltage on input, output and bidirectional 3.3 V tolerant pins	GND - 0.5 ^a	VDD3 + 0.5 ^c	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

- AC transient undershoot of 0.7 V below GND - 0.5 V (i.e. - 1.2 V) is allowed for a maximum period of 2 ns.
- AC transient overshoot of 0.7 V above 5.5 V is allowed for a maximum period of 2 ns
- AC transient overshoot of 0.7 V above VDD3 + 0.5 V is allowed for a maximum period of 2 ns.

Note: Stresses greater than those listed in Table 200 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.

Confidential

72.2 Operating conditions

Table 201: Operating conditions

Symbol	Parameter	Min	Max	Typical	Units	Notes
VI	Input voltage (3.3 V I/O only)	0	3.6		V	a
VI18	Input voltage (1.8 V I/O only)	0	1.95		V	b
VI35	Input voltage (3.3 V/5 V tolerant I/O)	0	5.5		V	c
C _L	Load capacitance per pin		75		pF	
C _{LD}	Load capacitance per data pin		75		pF	
C _{LA}	Load capacitance per address/strobe pin		75		pF	
C _{LP}	Load capacitance per PIO pin		100		pF	
C _{LPI2C}	Load capacitance per I ² C PIO pin		400		pF	
TA	Operating temperature (ambient)	0	70		°C	
PD	Typical power dissipation			1.5	W	
	Maximum power dissipation at VDD3 = 3.6 V and VDD = 1.95 V		2.0	1.8	W	d
PD _{LP}	Power dissipation (low power mode)		400	250	mW	e
PD _{PD}	Power dissipation (power down mode)		60		mW	f

- Excursions beyond the supplies are permitted but not recommended. AC transient overshoot and undershoot are permitted up to the limits in [Table 200: Absolute maximum ratings on page 679](#).
- Excursions beyond the supplies are permitted but not recommended.
- AC transient overshoot and undershoot is permitted up to the limits in [Table 200: Absolute maximum ratings on page 679](#).
- Maximum dissipation of 2 W is allowed for the package.
- Power supplied to both RTCVdd and Vdd supplies and PLL still running, but internal clocks stopped.
- Power removed from Vdd but power remaining on RTCVdd to allow the real time clock to continue operating. The expected typical value was 0.008 mW, but due to a bug (GNBvd14846) on STi5516, 30 mA is consumed from the RTCVdd supply.

72.3 DC specifications

Table 202: DC specifications

Symbol	Parameter	Min	Typical	Max	Units	Notes
VDD18	Positive core supply voltage	1.65	1.8	1.95	V	
VDD3	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		VDD3 + 0.3	V	
V _{IH18}	Input logic 1 voltage (1.8 V input only)	1.3		RTC _{VDD} + 0.2	V	a
V _{IH5}	Input logic 1 voltage (5 V tolerant I/O only)	2.0		5.5	V	
V _{IHL}	Input logic 1 voltage (LP clocks only)	90% of VDD18			V	b
V _{IL}	Input logic 0 voltage			0.8	V	
V _{IL18}	Input logic 0 voltage (1.8 V input only)			0.5	V	a
V _{ILL}	Input logic 0 voltage (LP clocks only)			10% of VDD18	V	b
I _{IN}	Input current (input pin)			±5	μA	c
I _{OZ}	Off state digital output current			±50	μA	
I _{OZPIO}	Peak off state PIO input/output current			±200	μA	c
I _{OZEMI}	Peak off state EMI input/output current			1	mA	c
LWP _U	Input weak pull-up current			110	μA	
LWP _D	Input weak pull-down current			110	μA	
LWP _{PIO}	Input weak pull-up current on PIO pins		60	108	μA	d
V _{OH}	Output logic 1 voltage	2.4			V	e
V _{OL}	Output logic 0 voltage			0.4	V	d
C _{IN}	Input capacitance (input pins)			10	pF	c
C _{IO}	Input capacitance (bi-directional pins)			15	pF	c
C _{OUT}	Output capacitance			15	pF	c
V _{HYS}	Hysteresis voltage (3 V/5 V tolerant I/O)	0.4			V	
V _{HYS18}	Hysteresis voltage (1.8 V tolerant inputs)	0.2			V	a

a. NOT_RESET pin only

b. Low power clock must be monotonic

c. $0 \leq V_I \leq VDD3$

d. $V_I = VDD3$

e. $I_{load} = 4 \text{ mA}$ for IEEE1284, 8 mA for EMI/SMI, 4 mA for PIO, 4 mA for all other outputs

Note: For VDDAUDIOFSYN and VDDGENFSYN the supply tolerance is +/- 0.15 at synthesizer input, that is 1.65 V minimum to 1.95 V maximum.

72.4 Audio DAC specifications

Table 203: Audio DAC specifications: high end (± 9 V)

Symbol	Parameter	Min	Typ	Max	Units	Notes
VDDAADAC VDDASADAC VCCAADAC VCCASADAC	DC analog supply voltage	3.0	3.3	3.6	V	
SNR	Signal to noise ratio	98	-	-	dB	a
THD	Total harmonic distortion	-	-	-71	dB	a
R _{ref}	External reference resistor	175	200	-	Ω	
IOUT	Output current on OUTP	-	-	7.17	mA	

a. Measured using the ± 9 V op-amp power supply as shown in [Figure 185: High end application \(\$\pm 9\$ V symmetrical op-amp power supply\)](#) on page 550.

72.5 Video DAC specifications

Table 204: Video DAC test conditions

Test	Value
Nominal voltage	3.3 V
RL	162 Ω
R _{Iref}	10 k Ω
DLE/ILE tests	1 K point ramp (10 bits) is acquired. 10 points per step are captured.

Table 205: Video DAC specifications

Symbol	Parameter	Min	Typ	Max	Units
R _{Iref}	Reference current resistor	8	10	12	k Ω
V _{out}	DAC output voltage from code 0 to code maximum	1.20		1.4	V
	DAC to DAC V _{out} mismatch (per tri-DAC)	-2		2	%
ILE	LF integral nonlinearity Peak to Peak		1.5	2.5	LSB
DLE	LF differential nonlinearity		± 0.3	± 0.9	LSB

Confidential

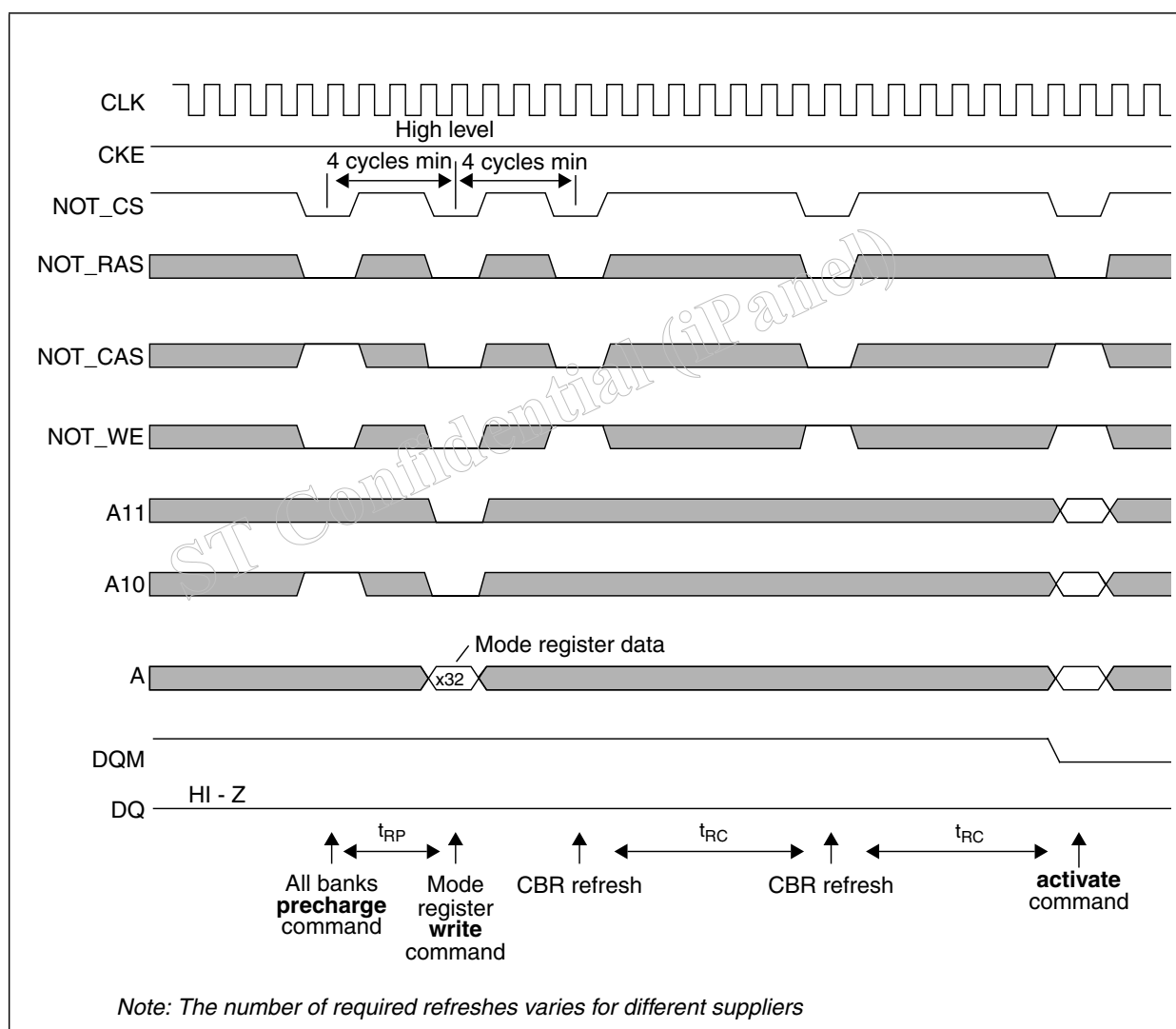
73 Timing specification

Timings are based on the following conditions unless otherwise stated:

- input rise and fall times of 1 ns (10% to 90%),
- output load = 75 pF,
- output threshold = 1.5 V.

73.1 SMI SDRAM

Figure 209: Synchronous DRAM power-on sequence



Confidential

Figure 210: AC parameters for read (synchronous DRAM)

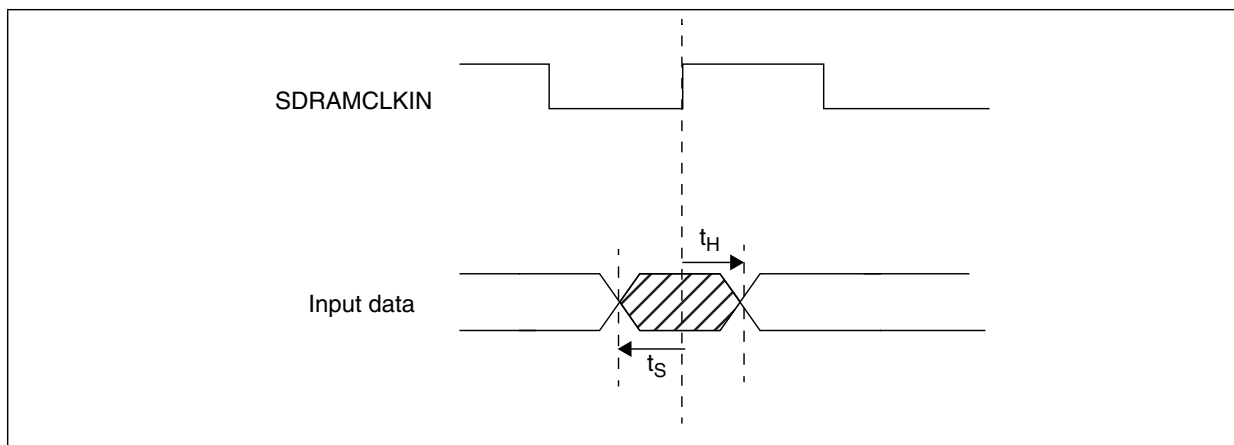


Figure 211: AC parameters for write (synchronous DRAM)

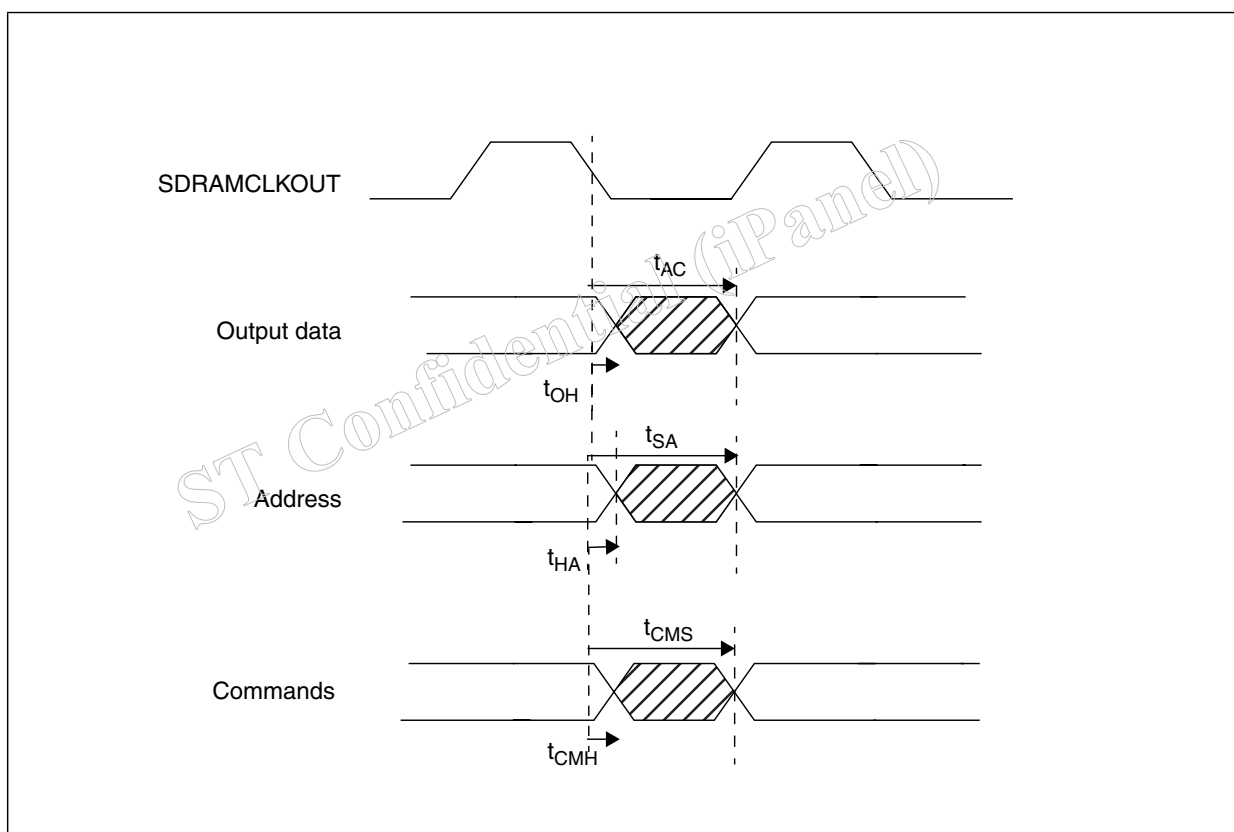
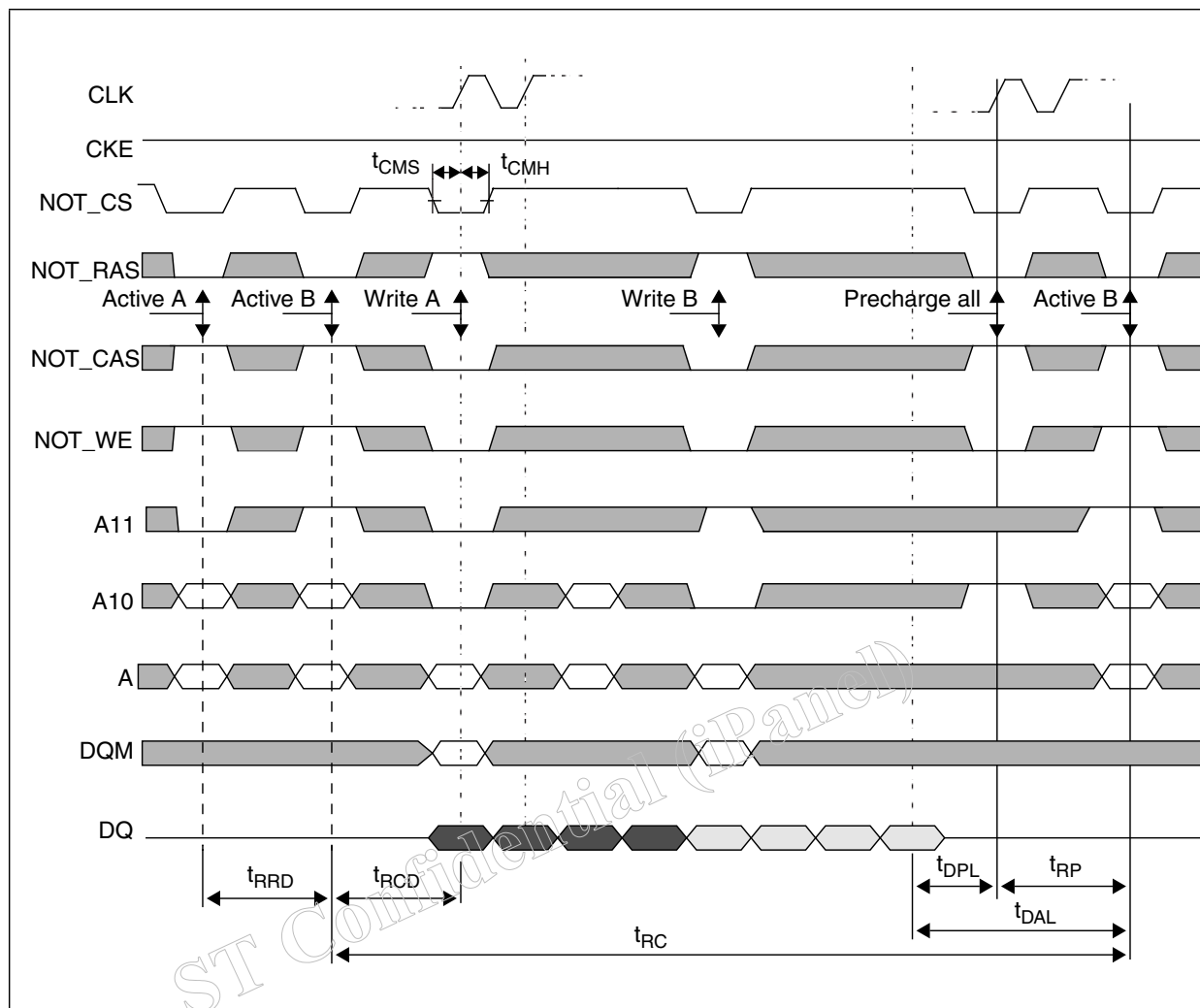


Figure 212: Synchronous DRAM write (burst length = 4, CAS latency = 3)



In Table 206, the unit T is the period of the memory subsystem clock (typically 8.23 ns / 121.5 MHz).

Table 206: 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	a
t_{CH}	Clock high level width			ns	b
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	c
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	

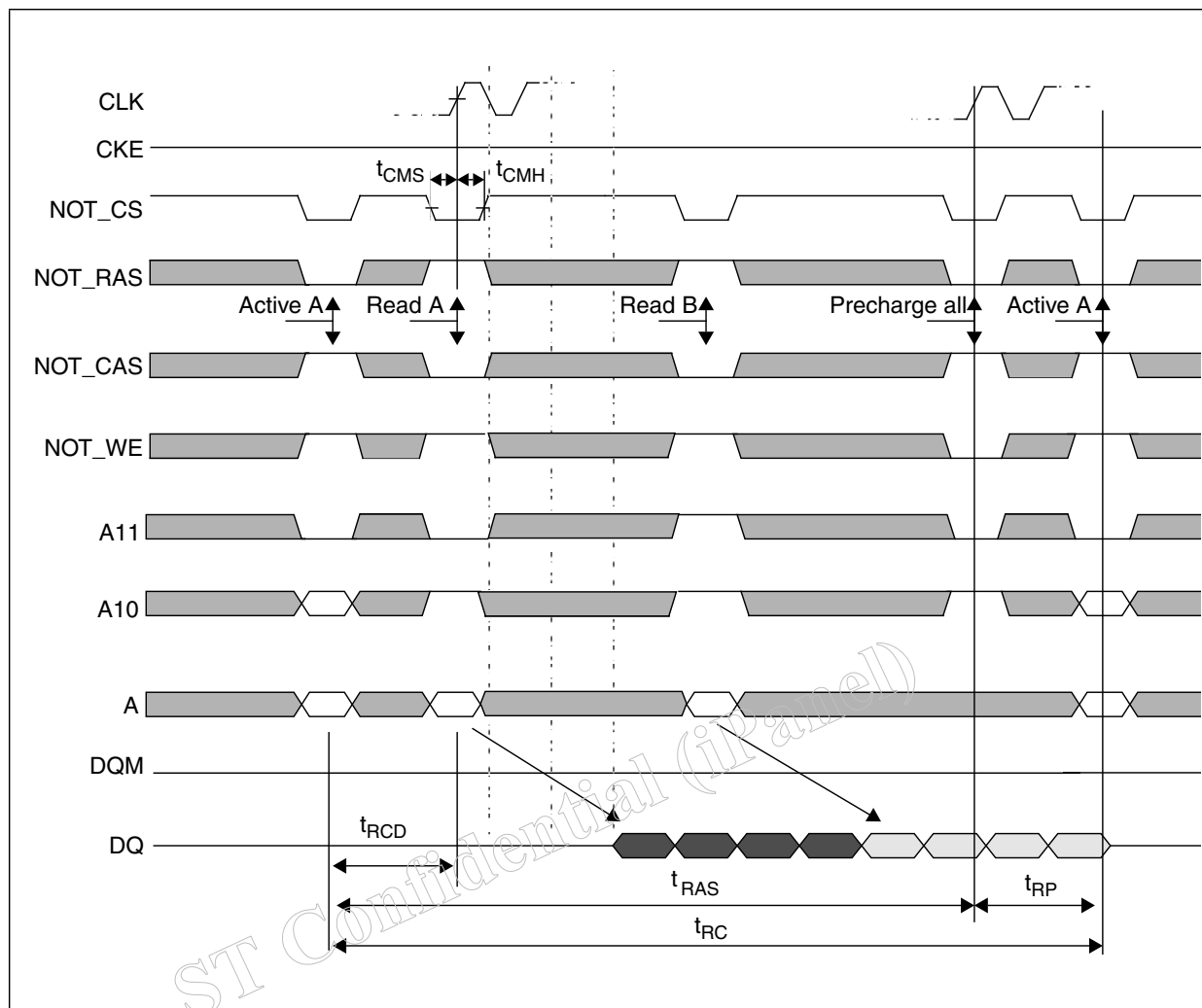
a. Maximum clock rate is 135 MHz.

b. A 50% duty cycle is recommended.

c. Negative values mean that the measured event is before the falling edge of MEMCLKOUT.

Confidential

Figure 213: Synchronous DRAM read (burst length = 4, CAS latency = 3)



Confidential

73.2 Audio PCM interface timings

Figure 214: PCM data output

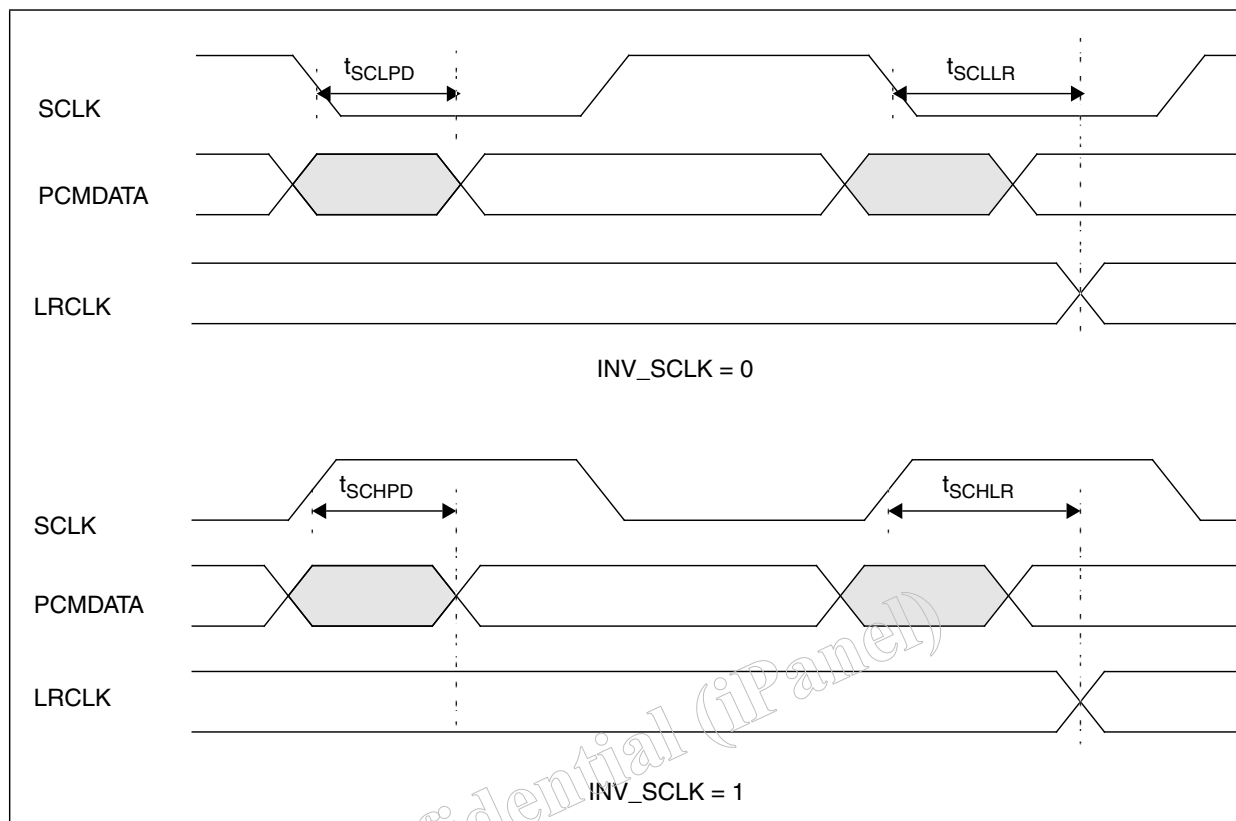


Table 207: PCM data output

Symbol	Parameter	Min	Max	Units
t_{SCLPD}	SCLK low to PCMDA valid		10	ns
t_{SCLLR}	SCLK low to LRCLK		10	ns
t_{SCHPD}	SCLK high to PCMDA valid		50	ns
t_{SCHLR}	SCLK high to LRCLK		50	ns

73.3 EMI timings

The EMI supports synchronous accesses to SDRAM and SFlash™ and also access to asynchronous peripherals.

73.3.1 Synchronous devices

All synchronous transactions originate and terminate at flip flops within the padlogic. Outputs are generated with respect to the rising edge of the bus clock, and inputs are sampled with respect to the same edge. The static timing characteristics for all EMI pins are shown in [Figure 215](#) and [Table 208](#).

Note: All outputs have been measured using the 15 pF drive strength selection.

Figure 215: EMI interface timings for synchronous transactions

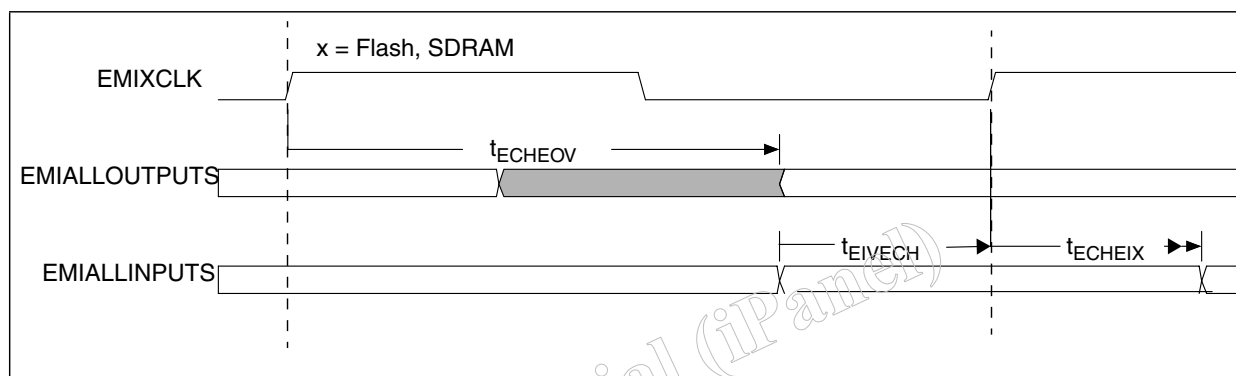


Table 208: EMI synchronous interface parameters (uncompensated, unshifted timing)

Output values assume a 75 pF external load			
Symbol	Min (ns)	Max (ns)	Description
t_{ECHEOV}	1.0	3.0	Bus clock rising edge to valid output data
t_{EIVECH}	4.0		Input valid to rising clock edge (input setup time)
t_{ECHEIX}	-1.0		Rising clock edge to input invalid (input hold time)

EMIALLOUTPUTS refers to all EMI address/data/strobe outputs, DMARAK/ACK outputs and NOT_EMIREQGNT output.

EMIALLINPUTS refers to all EMI data inputs, NOT_EMIAKREQ input and EMIWAITNOTTREADY input.

These values are static offsets within a bus clock cycle, they should be read in conjunction with the waveforms in [Chapter 16: External memory interface \(EMI\)](#) on page 141, which are cycle accurate only.

STMicroelectronics strongly recommends using the shift value of 1 for the clock because this value gives optimum set up for most memory devices. Hence the parametrics in [Table 209](#) are the limits to use for synchronous memory devices. However a manual shift offset may be applied, for a different shift value, using the formula:

$$\text{shifting setup} = t_{CHEOV} + \text{deskew\_value} \times \text{PLL\_period}$$

where:

- deskew_value is the number of bits that the value in CLKDIVn_SEQUENCE register is rotated to the left (see [Table 189: Programmable divider register values on page 566](#)),
- PLL_period is 2.06 ns (486 MHz).

Table 209: Recommended configuration

PLL frequency 540 MHz, external load of 75 pF, shift value of 1			
Symbol	Min (ns)	Max (ns)	Description
t_{ECHEOV}	3.0	5.1	Bus clock rising edge to valid output data
t_{EIVECH}	2.0	-	Input valid to rising clock edge (input setup time)
t_{ECHEIX}	1.1	-	Rising clock edge to input invalid (input hold time)

Bus cycle time is programmed as a division (1, 2 or 3) of the EMI subsystem clock frequency on a device by device basis. When configured as a clock master, the STi5516 EMI may be programmed over a range of frequencies up to 90 MHz. Refer to [Chapter 52: Clock generator on page 551](#) for details of this. A typical configuration for master mode is shown in [Table 210](#).

Table 210: Typical configuration, STBus clock = 90 MHz

Device type	Divide ratio	Clock name	Speed (MHz)
SDRAM	1	EMISDRAMCLK	90
SFlash™	2	EMIFLASHCLK	45

73.3.2 Asynchronous memory/peripherals

The EMI strobes are programmed in terms of internal clock phases, that is to say with half cycle resolution. This is illustrated in [Chapter 16: External memory interface \(EMI\) on page 141](#). The clock to output delay for all outputs (address, data and strobes) are closely matched, with a skew tolerance of ± 3 ns.

The input latch point for a read access is determined by the number of programmed EMI subsystem clock cycles for the latch point. The correction in [Figure 216](#) allows the latch point to be measured from the edge of an active strobe, that has been programmed to rise at the programmed read latch point.

For example if all signals are equally loaded with 75 pF, and with the same drive strength selection, the time between the address bus switching and a strobe or data bus output switching is only n programmed phases ± 3 ns. That is: worst case the strobe or data is a maximum of 3 ns after the address, or worst case the strobe or data is 3 ns before the address.

For a read cycle the data is latched by the STi5516 at the programmed number of EMI subsystem clock cycles from the end of the access plus a latch point correction time, which is effectively the read setup time. The latch point correction time (read setup time) is a minimum of 5 ns + skew tolerance correction of the output signal used as a reference. This is 5 ± 3 ns, thus the minimum read setup time relative to a strobe is 8 ns. This ensures that the read hold time is always a minimum of 0 ns, guaranteed by design.

Figure 216: Asynchronous access: READ

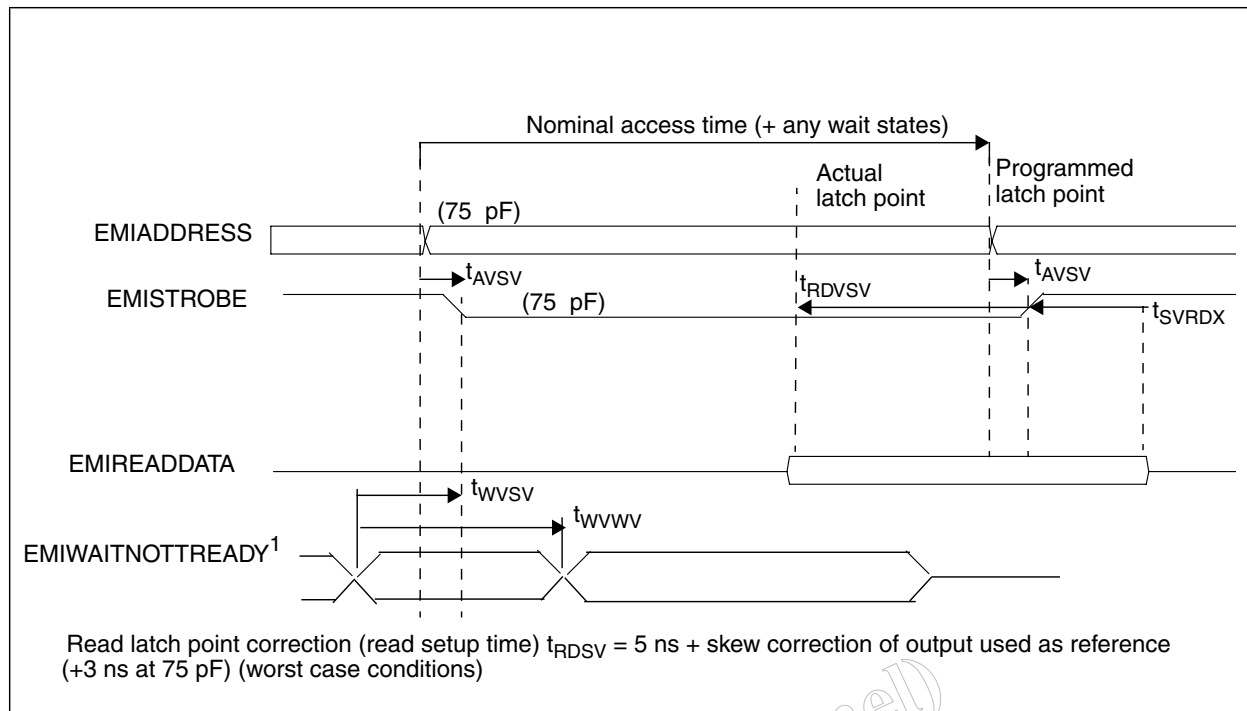


Figure 217: Asynchronous access: WRITE

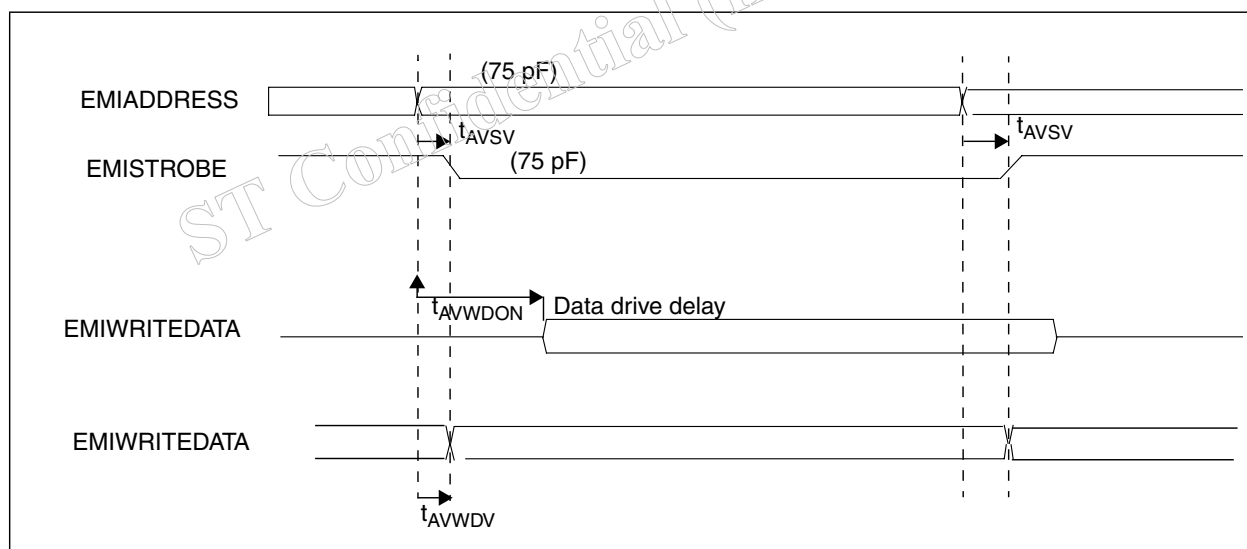


Table 211: EMI asynchronous access mode timings

Values assume a 75 pF external load				
Symbol	Min (ns)	Max (ns)	Description	Notes
t_{AVSV}	-3.0	3.0	Address valid to output strobe valid	a
t_{RDVSV}	8.0		Read data valid to strobe valid (read setup time)	b
t_{SVRDX}	0		Read data hold after strobe valid (read hold time)	c
t_{WVSV}	8.0		Wait valid to strobe valid (wait setup time)	d,e
t_{WVWV}	>1T		Minimum wait valid time (before state change)	e
t_{AVWDON}		3.0	Address valid to write data valid from tri-state	f
t_{AVWDV}	-3.0	3.0	Address valid to write data valid from on state	g

- a. Skew plus nominal N programmed EMI subsystem clock cycles of strobe delay.
- b. Skew from nominal programmed read latch point.
- c. Minimum values are guaranteed by design.
- d. Use an output strobe programmed to fall on the rising edge of the EMI subsystem clock, as a reference. EMIWAITNOTTREADY is sampled at the end of the first and subsequent clock cycles of the access, except for the penultimate cycle.
- e. EMIWAITNOTTREADY is synchronized by the EMI pad logic using the EMI subsystem rising clock edge, so violating setup time just means that if the signal was sampled low, no wait state is inserted. If sampled high a wait state is inserted. The signal is resampled on the next EMI subsystem clock cycle, where T is the EMI subsystem clock period.
- f. Skew from nominal programmed phases of data drive delay.
- g. No data drive delay.

Confidential

73.4 PIO timings

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

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

Table 212: PIO timings: SSC (I²C bus)

Symbol	Parameter	Min	Max	Units	Note
t_{PCHPOV}	PIO_REFCLOCK high to PIO output valid	-20.0	0.0	ns	a
t_{PCHWDZ}	PIO tristate after PIO_REFCLOCK high	-20.0	5.0	ns	a
t_{PIOr}	Output rise time	7.0	30.0	ns	
t_{PIOF}	Output fall time	7.0	30.0	ns	

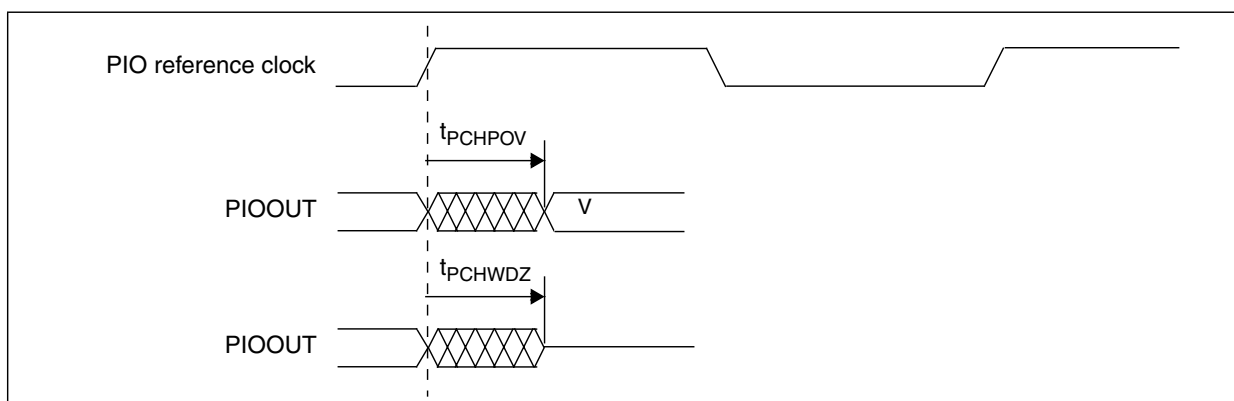
a. These timings only apply when a PIO pin is an output for the two SSC ports (PIO3 bits 0 to 3), since these outputs use a slew rate limited driver, to reduce noise on an I²C bus.

Table 213: PIO timings: general

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

a. These timings apply to all PIO outputs, other than the two SSC ports (PIO3 bits 0 to 3).

Figure 218: PIO timings



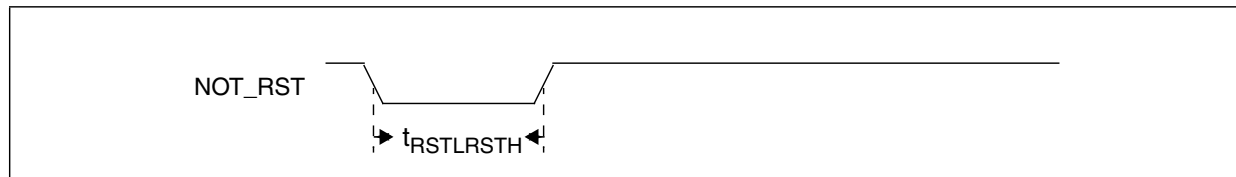
73.5 Reset timings

The $t_{RSTHRSTL}$ value in the following figure and table is for reset when the device is warm. A warm reset is defined as occurring when the clock and power supply are stable.

Table 214: Reset and analyze timings

Symbol	Parameter	Min	Nom	Max	Units
$t_{RSTLRSTH}$	NOT_RST pulse width low	5			CLK27MA

Figure 219: Reset and analyze timings

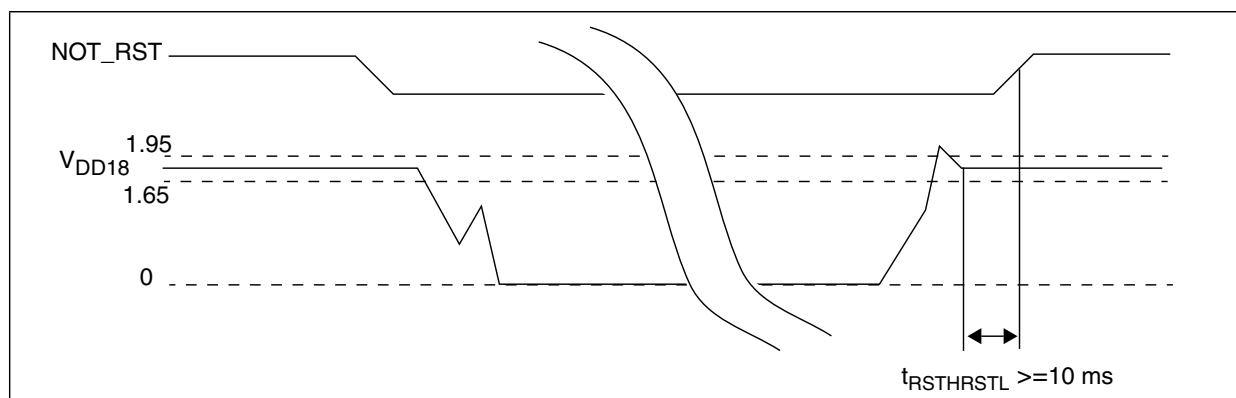


The $t_{RSTHRSTL}$ value specified in the following table and figure is for power-on reset when the device is cold. Warm reset parameters (when the clock and power supply are stable) fall within these limits; the $t_{RSTHRSTL}$ value should be used for both cases.

Table 215: Power-on reset timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
$t_{RSTHRSTL}$	NOT_RST pulse width low with a stable VDD	10			ms	Takes temp, voltage and process variations into account, and provides guardband. Crystal oscillator start-up times can be long and may dominate the time from power-up to the end of reset.

Figure 220: Reset timings



73.6 Clock timings

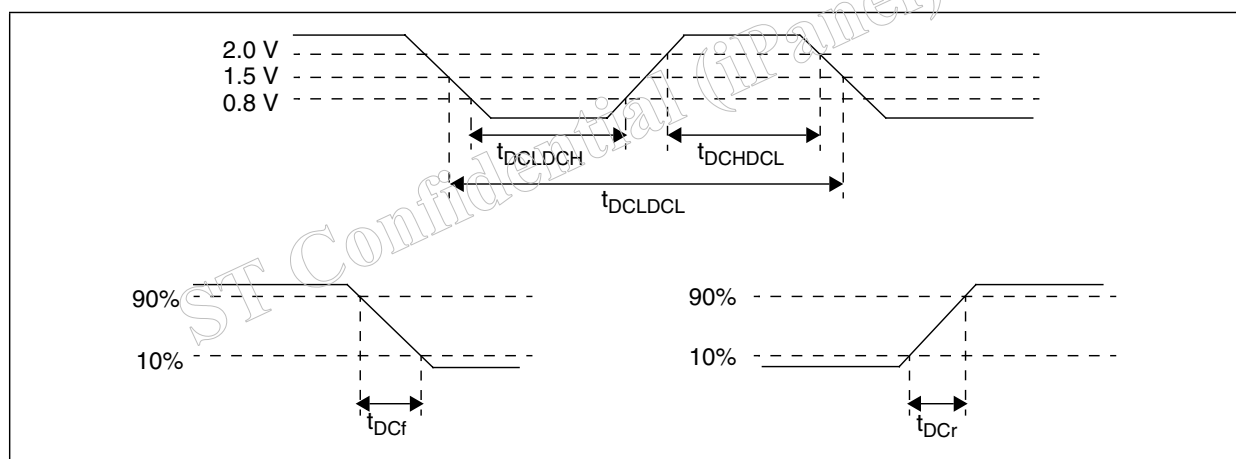
73.6.1 CLOCKIN timings

Table 216: CLOCKIN timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
t_{DCLDCH}	CLOCKIN pulse width low	6			ns	
t_{DCHDCL}	CLOCKIN pulse width high	10			ns	
t_{DCLDCL}	CLOCKIN period		37		ns	a, b
t_{DCr}	CLOCKIN rise time			10	ns	c
t_{DCf}	CLOCKIN fall time			10	ns	c

- a. Measured between corresponding points on consecutive falling edges.
- b. Variation of individual falling edges from their nominal times.
- c. Clock transitions must be monotonic within the range V_{IH} to V_{IL} .

Figure 221: CLOCKIN timings



73.6.2 LPCLKIN timings

Table 217: LPCLKIN timings

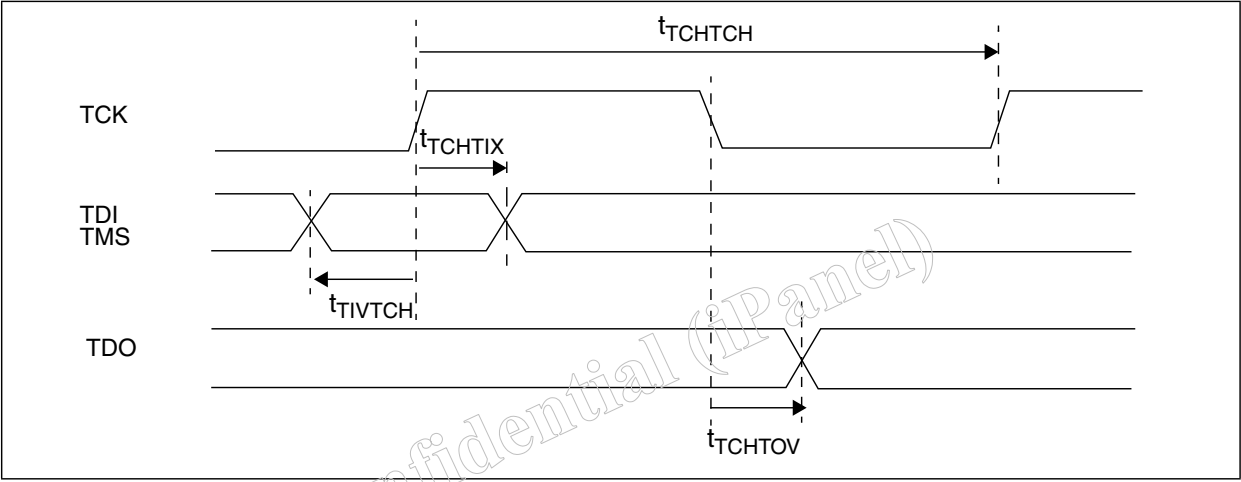
Symbol	Parameter	Min	Nom	Max	Units	Notes
t_{DCrLP}	LPCLKIN rise time			100	ns	
t_{DCfLP}	LPCLKIN fall time			100	ns	
t_{FLP}	LPCLKIN frequency	200			Hz	
				50	kHz	
t_{MSLP}	Mark/space	40/60		60/40		

73.7 TAP timings

Table 218: TAP timings

Symbol	Parameter	Min	Nom	Max	Units
t_{TCHTCH}	TCK period			6.25	MHz
t_{TIVTCH}	TAP inputs valid to TCK high	4			ns
t_{TCHTIX}	TAP input hold after TCK high	4			ns
t_{TCHTOV}	TCK low to TAP output valid			15	ns

Figure 222: TAP timings



Confidential

73.8 Transport stream timings

73.8.1 TSIN interface

Table 219: TSIN timings: parallel input mode

Symbol	Parameter	Min	Nom	Max	Units
t_{LCLLCL}	TSINBYTECLK period	66			ns
t_{LCHLCL}	TSINBYTECLK pulse width high	10			ns
t_{LCLLCH}	TSINBYTECLK pulse width low	10			ns
t_{LDVLCH}	TSIN signals valid to TSINBYTECLK high	4			ns
t_{LCHLDX}	TSIN signals hold after TSINBYTECLK high	2			ns

Figure 223: Parallel transport stream input timings (equivalent to 120 Mbit/s input rate)

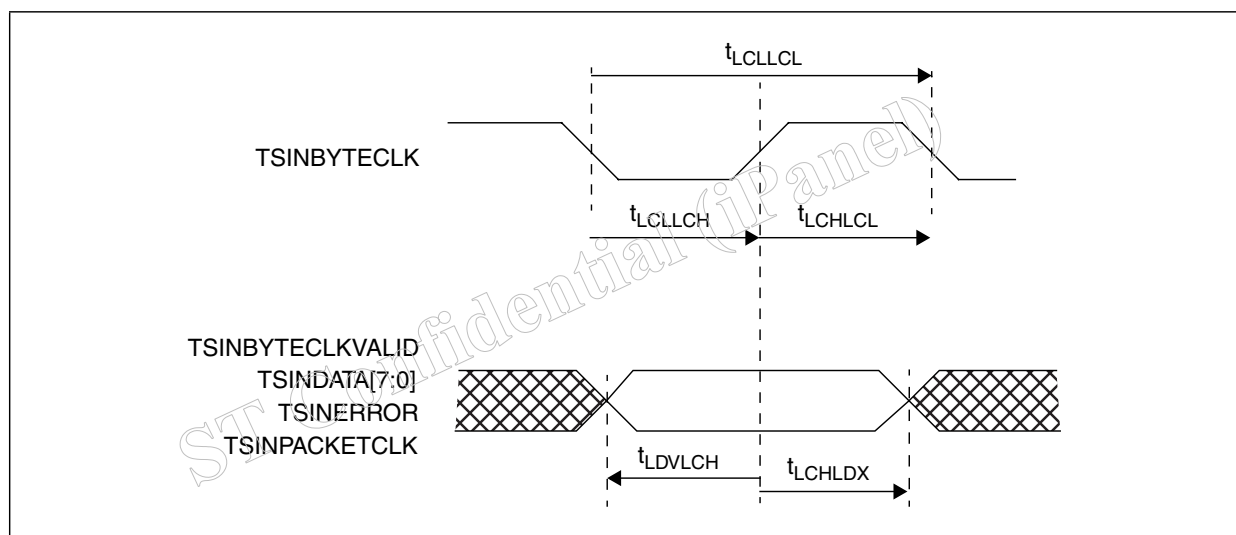
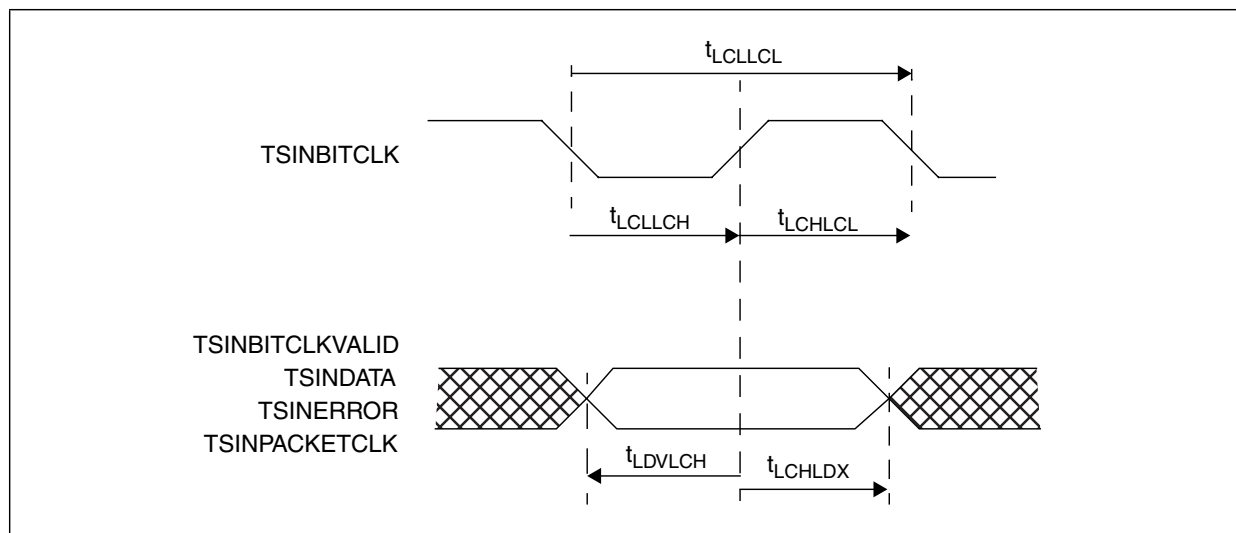


Table 220: TSIN timings: serial input mode

Symbol	Parameter	Min	Nom	Max	Units	Notes
t_{LCLLCL}	TSINBITCLK period	10			ns	a, b, c
t_{LCHLCL}	TSINBITCLK pulse width high	3			ns	
t_{LCLLCH}	TSINBITCLK pulse width low	3			ns	
t_{LDVLCH}	TSIN signals valid to TSINBITCLK high	3			ns	c
t_{LCHLDX}	TSIN signals hold after TSINBITCLK high	2			ns	

- The maximum frequency of the bit clock is 1/3 of the programmed STBus clock frequency for DSS format streams. See the STi5516 bug list, bug GNBvd09735, for further details.
- PTI hardware sync lock and drop method must be used for DVB format streams, otherwise the maximum bit clock frequency is limited to 1/3 of the STBus clock frequency.
- TSIN2L port is slower than TSIN1 in serial mode, with t_{LDVLCH} minimum of 5.0 ns.

Figure 224: Serial transport stream input timings



This applies to TSIN1 and TSIN2 ports in serial mode.

Equates to 100 Mbit/s transport rate.

For TSIN1 and TSIN2 in serial mode, data is bit 7 of TSINDATA.

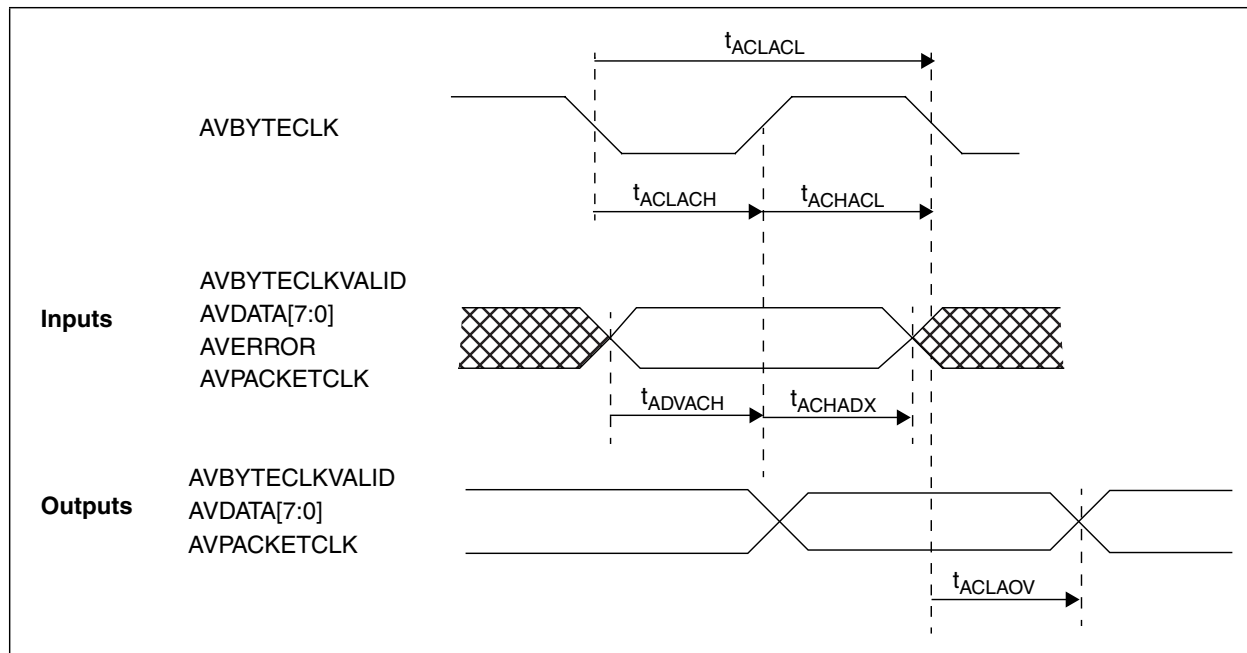
AVSTREAM interface on the LLI/1284 pins in AV mode

Table 221: AV timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
t_{ACLACL}	AVBYTECLK period	100			ns	
t_{ACHACL}	AVBYTECLK pulse width high	40			ns	
t_{ACLLAH}	AVBYTECLK pulse width low	40			ns	
t_{ADVACH}	LINKIC inputs valid to AVBYTECLK or TSINBYTECLK high	12			ns	a
t_{ACHADX}	LINKIC input signal hold after AVBYTECLK high	2			ns	a
t_{ACLADV}	AVBYTECLK low to output data valid	0		20	ns	

a. The inputs refer to the clock selected by the LLI_CONTROL and LLI_BYTECLOCKSELECT registers.

Figure 225: AV stream I/O timings



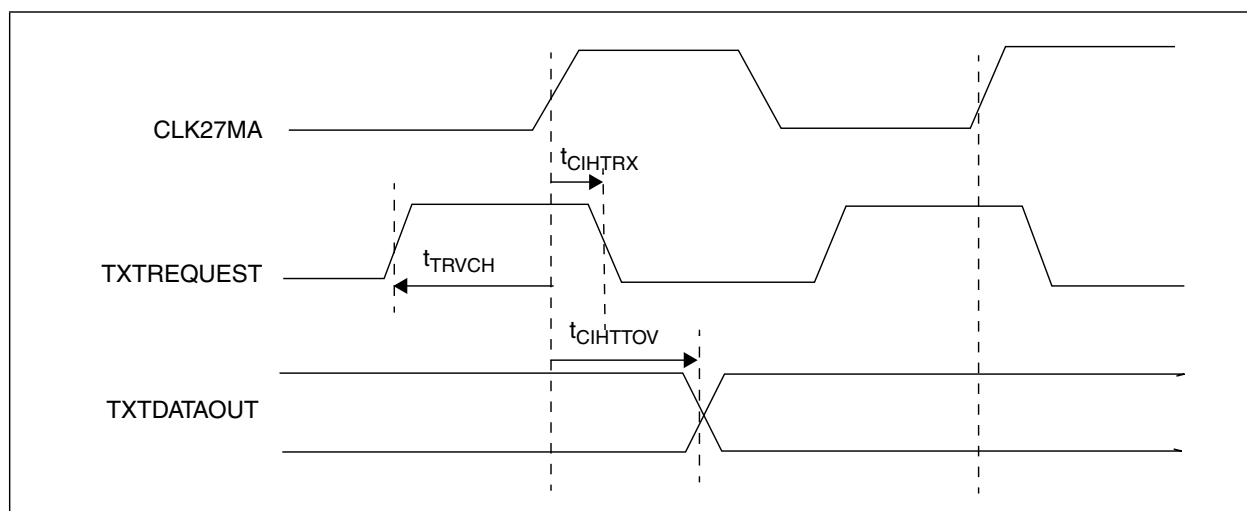
73.9 Teletext timings

73.9.1 Teletext data out

Table 222: Teletext out timings

Symbol	Parameter	Min	Nom	Max	Units
t_{TRVCIH}	TELETEXTREQUEST setup time to CLK27MA high	10			ns
t_{CIHTRX}	TELETEXTREQUEST hold time from CLK27MA high	5			ns
$t_{CIHTTOV}$	CLOCKIN high to teletext data output valid	8		27	ns

Figure 226: Teletext out



73.10 IEEE 1284 timings

The reference clock shown below is a virtual clock and is defined as the point at which all output strobes are valid.

Table 223: IEEE 1284 timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
$t_{IRCHIOV}$	I1284_REFCLOCK high to 1284 data outputs valid	-6.0		0	ns	a
t_{IIVIX}	I1284 inputs valid time	120			ns	b, c
t_{IOLIOV}	I1284INNOTOUT low to data enable	0		-4	ns	
t_{IOLIOZ}	I1284INNOTOUT low to data tristate			15	ns	c

- a. The IEEE 1284/1394 LLI pins do not use the 14mA drivers required by the IEEE 1284 standard, therefore an external buffer device must be used to implement an IEEE 1284 port.
- b. Equivalent to 3 clock cycles ($3 \times 27 \text{ MHz} = 111 \text{ ns}$) + guard band
- c. EXTBUSDIRECTION control bit set to 0

Figure 227: I1284 output skews

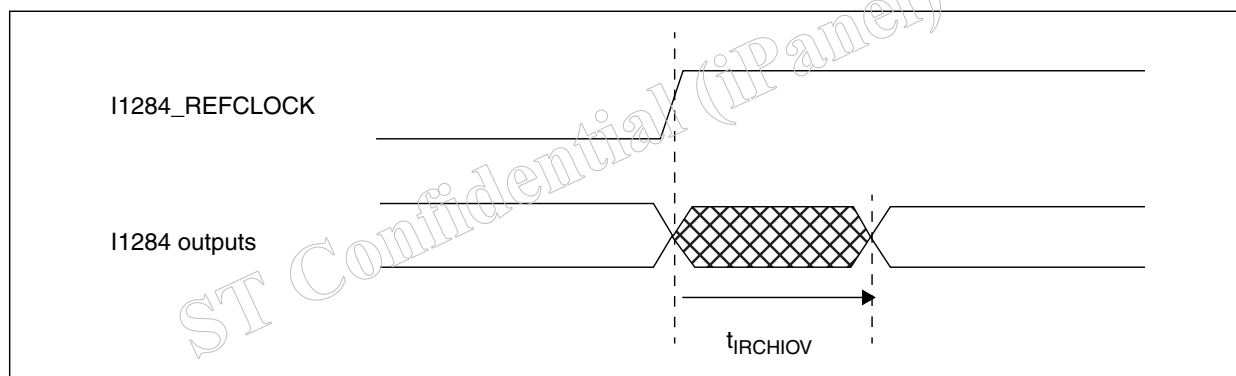
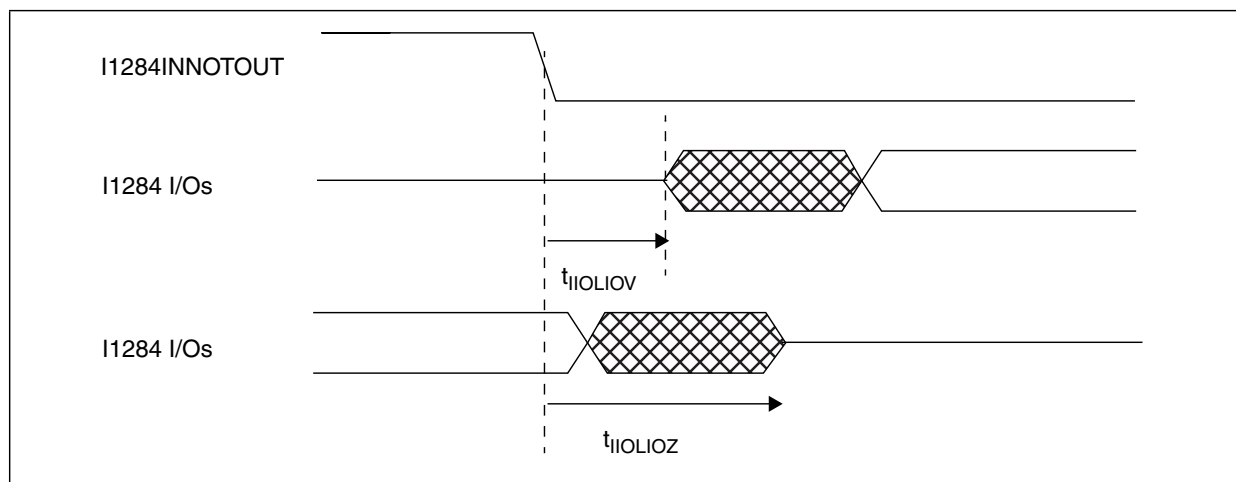


Figure 228: I1284INNOTOUT to I1284 I/O enable and disable



74 Revision history

Version E	
	Device name changed to from STi5516B to STi5516
29 Programmable transport interface (PTI) registers	PTI3 register map updated PTI_DMAENABLE uses bits 0 to 3 rather than 5 to 8.
38 Display planes	Double header mode removed
47 Audio decoder Section 47.2: Decoding process on page 465	The volume is controlled independently for each channel in steps of 1 dB
72 Electrical specifications	LWP _{PIO} max is 108 µA, typical is 60 µA
Version D	
Cover	CPU speed is 180 MHz
Introduction	CPU speed is 180 MHz
4 Pin list Section 4.2: STi5516 pin list on page 28 Section 4.4: Reset states on page 40	IEEE 1284/1394 pad types are C4 Pull-down on NOT_RESET pin removed
17 EMI registers	EMI_CLKENABLE address corrected to 0x68
18 EMI padlogic Section 18.2.6: Bus control input strobe on page 184 Section 18.2.9: Bus control output strobe generation on page 186	Removed retiming Removed retiming
19 EMI padlogic registers Section 19.3: Configuration registers on page 194	CONFIG_CONTROL_E, bit 10 description updated CONFIG_CONTROL_E, programmable pad strength bits updated
21 EMI buffer registers	Register descriptions have changed
24 Diagnostic controller (DCU) registers	DCU_CAPTUREn_VALUE, address calculation corrected to <i>DCUBaseAddress</i> + (0x404 + 0x08 x n)
26 Test access port registers	JTAG identification code is 0xnD4 1D041
32 IEEE1394 Link layer interface (LLI) Section 32.4: Pin function on page 280	Figure 77: Byte clock selection corrected
34 MPEG video decoder Section 34.8.5: Memory segments for 128-Mbit SDRAM on page 296	Frame buffer alignment must be on an even bank in the SDRAM
35 MPEG video decoder registers	Recommended values for the MPEGCONTROL register

39 Display planes registers	TDL_TDW is defined in units of 16 pixels.
47 Audio decoder	Additional explanation of PCM mixing in <i>Double DMA injection</i> on page 481
48 Audio decoder registers	AUD_ADCIN_CFG bit 2 = if 1, SCLKIN2 strobes DATA2 on rising edge, if 0, on falling edge
49 Audio interface (AUDIF)	Removed references to GPDMA
50 Audio interface (AUDIF) registers	Removed AUDIF_PCMI AUDIF_PCMO no longer used AUDIF_MUX register bits 2:0, 011 sets the PCMI formatter (GNBvd19566)
52 Clock generator Section 52.2: Maximum clock frequencies and restrictions on page 552	Maximum and recommended optimum clock frequencies revised
62 Asynchronous serial controller (ASC) Section 62.4: Transmission on page 607	Transmission only occurs when NOT_ASCnCTS is low
63 Asynchronous serial controller (ASC) registers	Base address for UART4 corrected to 0x2011 4000 New register ASC_ENABLE
73 Timing specification	T _{RC} changed from 8 to 12
Version C	
4 Pin list Section 4.4: Reset states on page 40	New section
12 Interrupt system registers	Register map changed
17 External memory interface registers Section 17.1: Configuration register format for peripherals on page 173	In EMI_CONFIGDATA2 BEE1TIMEWRITE applies to falling edge of BE and BEE2TIMEWRITE to the rising edge
38 Display planes Section 38.4.3: Block-to-row converter on page 353	Table 140: Vertical filter modes on page 357 modified
44 Digital encoder registers	Register map changed
66 Synchronous serial controller Section 66.1: Overview on page 628 Section 66.2.1: Pin connection and control on page 630	Description of SPI functionality removed Pin connection described more clearly
67 Synchronous serial controller registers	Register SSCnBRG description changed for slave mode
72 Electrical specifications	Full characterization data
73 Timing specifications	Full characterization data

Version B	
General	Fast-I and Irdeto no longer supported
4 Pin list Section 4.3: PIO pins and alternative functions on page 36	Alternate signals ASC4_RTS, ASC4_CTS, ASC3CTS and ASC3RTS renamed to NOT_ASCxxxxx
12 Interrupt system registers Section 12.1: Interrupt level controller registers on page 123	Register map changed
17 External memory interface (EMI) registers	Address offsets changed for configuration registers
29 Programmable transport interface (PTI) registers Section 29.3: PTI configuration registers on page 261	The timebase for the PTI is 90 kHz.
35 MPEG video decoder registers Section 35.1: Configuration and control (CFG) register information on page 328	Register map changed VID_RS_WRG, VID_RS_RMWG, VID_RS_RDG and VID_TP_SCDLIMIT added
37 Subpicture decoder registers	SPD_PR_CFG, SPD_LR_EVEN and SPD_LR_ODD added
39 Display planes registers	Register map changed Register VID_656B added
41 2-D block move registers	Register map changed
44 Digital encoder registers Section 44.4.4: Sync in data based synchronization on page 414	Note about use of autotest mode Register map changed
48 Audio decoder registers Section 48.11: Audio interrupt registers on page 502	Updated description for AUD_INT register
51 Audio DAC	High-end application schematic changed
53 Clock generator registers	Automatic mode is no longer supported (see SHIFT_CONFIG)
56 PWM and counter module Section 56.4: Compare (programmable timer) facilities on page 578	Pins PWM_COMPAREn referenced rather than PWM_COMPAREOUTn
57 PWM and counter module registers	Register PWM_COMPAREOUTVAL2 now used rather than PWM_COMPAREOUTVAL1
61 Infrared transmitter/receiver registers Section 61.2: RC receiver registers on page 599	Register IRB_TX_SUB_CARRIER_WIDTH_IR added
73 Timing specification	Added preliminary characterization data

75 Index of registers

1284CHECKSUM	666	AUD_CRC_OFF	527
1284CONTROL	667	AUD_DECODESEL	509
1284DATAIN	668	AUD_DEEMPH	535
1284DATAOUT	669	AUD_DOWNMIX	536
1284DMAADDRESS	669	AUD_DOWNSAMPLING	527
1284DMACONTROL	670	AUD_DUALMODE	536
1284DMACOUNT	670	AUD_DWSMODE	513
1284DMATOKEN	671	AUD_ERROR	506
1284INTACK	672	AUD_HEAD3	505
1284INTENABLE	673	AUD_HEAD4	505
1284INTSTATUS	674	AUD_HEADLEN	506
1284MODEENABLE	675	AUD_ID	511
1284PACKETSIZE	675	AUD_ID_EN	510
1284PININ	676	AUD_ID_EXT	511
1284PININENABLE	676	AUD_IDENT	484
1284PININVALUE	677	AUD_INT	503
1284PINOUT	677	AUD_INT_RAM	484
1284PULSEWIDTH	678	AUD_INTE	502
1284STATUS	678	AUD_LPCM_DOWNMIX	528
ASC_ENABLE	617	AUD_MP_CRC_OFF	523
ASC_n_BAUDRATE	618	AUD_MP_DOWNMIX	524
ASC_n_CONTROL	620	AUD_MP_DRC	523
ASC_n_GUARDTIME	621	AUD_MP_DUALMODE	523
ASC_n_INTENABLE	621	AUD_MP_MC_OFF	523
ASC_n_RETRIES	622	AUD_MP_PROG_NUMBER	522
ASC_n_RXBUFFER	622	AUD_MP_SKIP_LFE	522
ASC_n_RXRESET	622	AUD_MP_STATUS0	525
ASC_n_STATUS	623	AUD_MP_STATUS1	525
ASC_n_TIMEOUT	624	AUD_MP_STATUS2	525
ASC_n_TXBUFFER	624	AUD_MP_STATUS3	526
ASC_n_TXRESET	624	AUD_MP_STATUS4	526
AUD_AC3_COMP_MOD	516	AUD_MP_STATUS5	526
AUD_AC3_DECODE_LFE	516	AUD_MULTI_CHANNEL	528
AUD_AC3_DOWNMIX	519	AUD_MUTE	500
AUD_AC3_DUALMODE	519	AUD_OCFG	515
AUD_AC3_HDR	516	AUD_PACKET_LOCK	510
AUD_AC3_KARAMODE	518	AUD_PCMCONF	487
AUD_AC3_LDR	517	AUD_PCMCROSS	487
AUD_AC3_RPC	517	AUD_PCMDIVIDER	486
AUD_AC3_STATUS0	519	AUD_PCMMIX_ACK	495
AUD_AC3_STATUS1	520	AUD_PCMMIX_FIRSTINPUT_VOLUME	494
AUD_AC3_STATUS2	520	AUD_PCMMIX_MIX_COEFFICIENT	494
AUD_AC3_STATUS3	520	AUD_PCMMIX_SRC_HANDSHAKE	495
AUD_AC3_STATUS4	521	AUD_PCMMIX_SRC_LSB	494
AUD_AC3_STATUS5	521	AUD_PCMMIX_SRC_MSB	494
AUD_AC3_STATUS6	521	AUD_PCMMIX_UPDATE	493
AUD_AC3_STATUS7	522	AUD_PDEC	512
AUD_ADCIN_CFG	490	AUD_PL_AB	512
AUD_ADCIN_LEFT_VOL	491	AUD_PL_DWNX	513
AUD_ADCIN_MODE	489	AUD_PLAY	500
AUD_ADCIN_PLAY	488	AUD_PN_CHANNELCONF	534
AUD_ADCIN_RIGHT_VOL	491	AUD_PTS	506
AUD_ADCIN_SHIFT	492	AUD_RS232_INTERF_ECHO	485
AUD_ADCIN_USERSETUP	490	AUD_RUN	500
AUD_ADCIN_USERSETUP2	491	AUD_SFREQ	488
AUD_ANCCOUNT	504	AUD_SFREQ2	489
AUD_BEEP_FREQ	532	AUD_SIN_SETUP	485
AUD_BREAKPOINT	483	AUD_SKIP_MUTE_CMD	501
AUD_BT_CHANNELCONF	533	AUD_SKIP_MUTE_VALUE	502
AUD_CAN_SETUP	486	AUD_SOFTRESET	499
AUD_CHAN_IDX	515	AUD_SOFTVER	484
AUD_CHANNEL_ASSIGNMENT	527	AUD_SPDIF_CAT	496
AUD_CLOCKCMD	483	AUD_SPDIF_CMD	496
		AUD_SPDIF_CONF	497
		AUD_SPDIF_DTDI	499
		AUD_SPDIF_LATENCY	498
		AUD_SPDIF_REP_TIME	498
		AUD_SPDIF_STATUS	498

AUD_STREAMSEL	510	DEN_CFCOE[0:8]	455
AUD_SYNC_LOCK	511	DEN_CFG0	432
AUD_SYNC_STATUS	504	DEN_CFG1	433
AUD_VCR_OUTPUT	495	DEN_CFG2	434
AUD_VERSION	484	DEN_CFG3	435
AUD_VOLUME0	514	DEN_CFG4	436
AUD_VOLUME1	514	DEN_CFG5	437
AUDIF_GCF	542	DEN_CFG6	437
AUDIF_MUX	543	DEN_CFG7	438
AUDIF_PCMICFG	544	DEN_CFG8	439
AUDIF_PCMO	546	DEN_CGMS[1:3]	448
AUDIF_PCMOCFG	545	DEN_CID	446
BANK_0_BASE_ADDRESS	200	DEN_CLF1	451
BANK_1_BASE_ADDRESS	200	DEN_CLF2	452
BANK_2_BASE_ADDRESS	201	DEN_DAC13	445
BANK_3_BASE_ADDRESS	201	DEN_DAC45	445
BANK_4_BASE_ADDRESS	202	DEN_DAC6C	446
BANK_5_BASE_ADDRESS	202	DEN_IDFS[1:3]	441
BANKS_ENABLED	203	DEN_LFCOE[0:9]	457
BCK_U	376	DEN_PDFS[1:2]	442
BCK_V	376	DEN_REG_64	453
BCK_Y	376	DEN_REG_65	453
CACHEING_ENABLE	135	DEN_REG_69	454
CFG_CCF	328	DEN_REG_70	454
CFG_CDR	328	DEN_REG_71	454
CFG_DRC	329	DEN_STA	440
CFG_GCF	329	DEN_TTX[1:5]	448
CFG_MCF	330	DEN_VPS1	447
CLEAR_STATUS_SSC	649	DEN_WSS[1:2]	444
CLOCK_SEL1	565	DIVIDER_MODE	562
CONFIG_CONTROL_A	194	EMI_CLKENABLE	172
CONFIG_CONTROL_B	195	EMI_CONFIGDATA0	173
CONFIG_CONTROL_C	196	EMI_CONFIGDATA0	177
CONFIG_CONTROL_D	197	EMI_CONFIGDATA1	174
CONFIG_CONTROL_E	198	EMI_CONFIGDATA1	178
CONFIG_DEVICEID_REG	194	EMI_CONFIGDATA2	175
DCU_CAPABILITY	211	EMI_CONFIGDATA2	178
DCU_CAPTUREn_PROPERTIES	223	EMI_CONFIGDATA3	176
DCU_CAPTUREn_VALUE	223	EMI_CONFIGDATA3	179
DCU_COMPARE_STATUS	215	EMI_FLASHCLKSEL	171
DCU_COMPAREn_PROPERTIES	221	EMI_GENCFG	170
DCU_COMPAREn_VALUE1	222	EMI_GENCFG	193
DCU_COMPAREn_VALUE2	222	EMI_LOCK	169
DCU_CONTROL	212	EMI_REFRESHINIT	171
DCU_JUMPTRACE_ADDRESS	221	EMI_SDRAMCLKSEL	172
DCU_JUMPTRACE_BYTES	220	EMI_SDRAMINIT	171
DCU_JUMPTRACE_END_ADDRESS	220	EMI_SDRAMMODEREG	170
DCU_JUMPTRACE_FROM_LPTR	219	EMI_SDRAMNOPGEN	170
DCU_JUMPTRACE_LAST_CAPTURE0	220	EMI_STATUSCFG	169
DCU_JUMPTRACE_LAST_LPTR	219	EMI_STATUSLOCK	169
DCU_JUMPTRACE_PROPERTIES	218	ENABLES	94
DCU_JUMPTRACE_START_ADDRESS	220	FLUSH	136
DCU_JUMPTRACE_TO_LPTR	219	HOST_DM_COEFT_0	528
DCU_SEQUENCING_CONFIGURATION	216	HOST_DM_COEFT_1	529
DCU_SEQUENCING_DISABLE	225	HOST_DM_COEFT_10	531
DCU_SEQUENCING_ENABLE	224	HOST_DM_COEFT_11	531
DCU_SIGNALLING	213	HOST_DM_COEFT_12	531
DCU_STATUS	214	HOST_DM_COEFT_13	532
DCU_TRIGGER_IN	215	HOST_DM_COEFT_2	529
DCU_TRIGGER_IN_PROPERTIES	217	HOST_DM_COEFT_3	529
DCU_WRANGE_ENABLE_ONLY_IN_RANGE	225	HOST_DM_COEFT_4	529
DCU_WRANGE_ENABLE_ONLY_OUT_OF_RANGE	226	HOST_DM_COEFT_5	530
DCU_WRANGE_LOWER	226	HOST_DM_COEFT_6	530
DCU_WRANGE_UPPER	226	HOST_DM_COEFT_7	530
DEN_CCF1	450	HOST_DM_COEFT_8	530
DEN_CCF2	450	HOST_DM_COEFT_9	531
DEN_CDEL_LFC	456	ILC_CLEAR_ENABLE	125
		ILC_CLEAR_STATUS	124

REGION_0_ENABLE	136	TTXT_DMACCOUNT	405
REGION_1_BLOCK_ENABLE	137	TTXT_INTENABLE	405
REGION_1_TOP_ENABLE	138	TTXT_INTSTATUS	406
REGION_2_ENABLE	138	TTXT_MODE	406
REGION_3_BANK_ENABLE	140	TTXT_OUTDELAY	406
REGION_3_BLOCK_ENABLE	139	USD_BMC	399
REGISTER_LOCK	562	USD_BMH	400
SCI_n_CLKCON	627	USD_BMW	400
SCI_n_CLKVAL	627	USD_BRP	400
SDLL/CLKDIVn_CONFIG0	567	USD_BSK	401
SDLL/CLKDIVn_CONFIG1	567	USD_BWP	401
SDLL/CLKDIVn_CONFIG2	567	USD_PAT	401
SDLL/CLKDIVn_SEQUENCE	567	VID_656	385
SHIFT_CONFIG	568	VID_656B	385
SPD_CTL1	338	VID_ABG	307
SPD_CTL2	339	VID_ABL	307
SPD_HCN	339	VID_ABS	307
SPD_HCOL	339	VID_ABT	308
SPD_HLEX	340	VID_BFC	308
SPD_HLEY	340	VID_BFP	308
SPD_HLSX	340	VID_CDCOUNT	309
SPD_HLSY	340	VID_CSO	386
SPD_LUT	341	VID_CTL	309
SPD_SPR	341	VID_CWL	309
SPD_SXD0	341	VID_DCF	386
SPD_SXD1	342	VID_DFC	387
SPD_SYD0	342	VID_DFP	387
SPD_SYD1	342	VID_DFS	310
SPD_XD0	343	VID_DFW	310
SPD_YD0	343	VID_DIS	388
SSCnBRG	644	VID_FFC	310
SSCnCON	645	VID_FFP	311
SSCnI2C	646	VID_HDF	311
SSCnIEN	647	VID_HDS	312
SSCnRBUF	647	VID_ITM	312
SSCnSLAD	648	VID_ITS	313
SSCnSTAT	648	VID_LCK	317
SSCnTBUF	649	VID_LDP	313
STATUS	136	VID_LSO	389
STATUS	93	VID_LSR	389
SYNTHn_CONFIG0	569	VID_MWS	390
SYNTHn_CONFIG1	570	VID_MWSV	390
TDL_CSO	377	VID_MWV	390
TDL_DCF	377	VID_OUT	391
TDL_LSO	378	VID_PAN	391
TDL_LSR	378	VID_PFH	314
TDL_SCN	379	VID_PFV	314
TDL_SWT	379	VID_PPR1	315
TDL_TDW	379	VID_PPR2	316
TDL_TEP	380	VID_PTH	316
TDL_TEP2	380	VID_QMW	317
TDL_TOP	380	VID_RFC	317
TDL_TOP2	381	VID_RFP	318
TDL_XDO	381	VID_SCDCOUNT	318
TDL_XDS	381	VID_SCN	391
TDL_YDO	382	VID_SPB	319
TDL_YDS	382	VID_SPE	319
TICKTIMER	572	VID_SPREAD	319
TSMUX_CLOCKGEN	277	VID_SPWRITE	320
TSMUX_PTIA_SOURCE	274	VID_SRA	320
TSMUX_SWTS	275	VID_SRV	320
TSMUX_SWTS_CONFIG	275	VID_STA	321
TSMUX_TSiSnMODE	274	VID_STL	322
TSMUX_TSOUTCLKSOURCE	276	VID_TIS	323
TSMUX_TSOUTSOURCE	276	VID_TP_CD	323
TTXT_ABORT	404	VID_TP_CD_RD	324
TTXT_ACKODDEVEN	404	VID_TP_CDLIMIT	324
TTXT_DMAADDRESS	404	VID_TP_SCD	324

VID_TP_SCD_CURRENT	325	VID_VFLMODE	393
VID_TP_SCD_RD	325	VID_XDO	394
VID_TP_SCDLIMIT	318	VID_XDO_656	394
VID_TP_VLD	325	VID_XDS	394
VID_TP_VLD_RD	325	VID_XDS_656	395
VID_TRF	326	VID_XFW	395
VID_VBG	326	VID_YDO	395
VID_VBL	327	VID_YDO_656	396
VID_VBS	327	VID_YDS	396
VID_VBT	327	VID_YDS_656	396
VID_VFCMODE	392		

Confidential

ST Confidential (iPanel)

ST Confidential (iPanel)

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without the express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

TruSurround, SRS, and the SRS symbol are registered trademarks of SRS Labs, Inc. in the United States and selected foreign countries. SRS and TruSurround are incorporated under license from SRS Labs, Inc. and are protected under U.S. Patent Nos. 4,748,669 and 4,841,572 and corresponding patents worldwide.

Dolby is a registered trademark of Dolby Laboratories.

Pro Logic is a registered trademark of Dolby Laboratories.

DIRECTV and the Cyclone Design logo are registered trademarks of DIRECTV, Inc., a unit of Hughes Electronics Corp., and are used with permission.

All other trademarks are the property of their respective companies.

© 2003 STMicroelectronics - All Rights Reserved

STMicroelectronics Group of Companies

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan
Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - U.S.A.

www.st.com