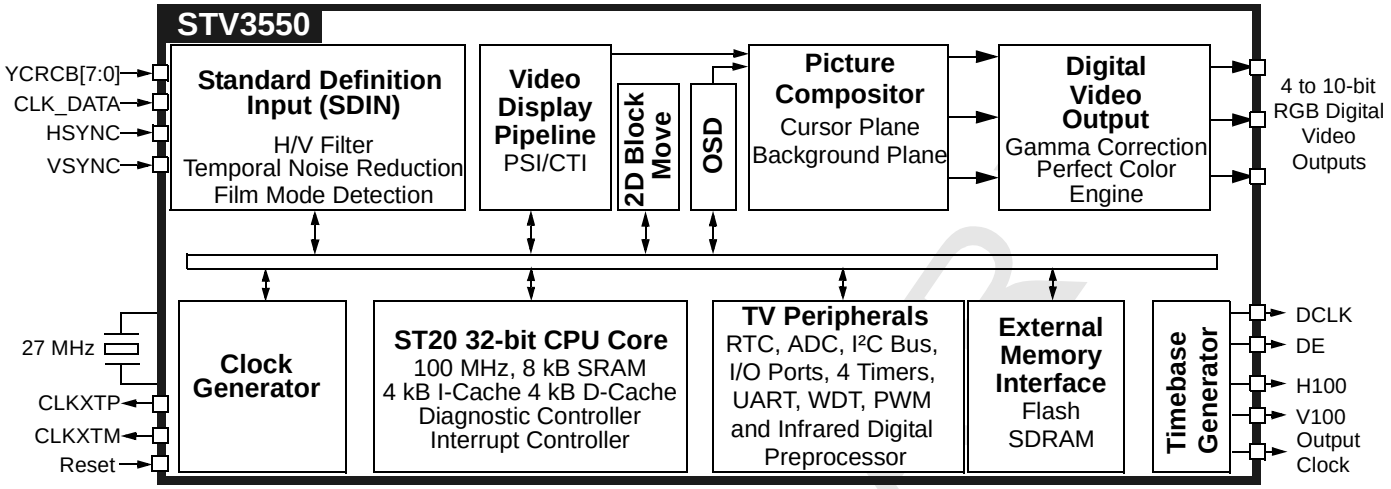
STV3550

LCD and Matrix Display TV Processor

DATASHEET



Main Features

- Fully-programmable Digital Video Output Stage for direct RGB interface to Flat Display Panel with 4- to 10-bit color resolution and pixel resolution from VGA to WXGA including HDTV2.
- **Versatile Integrated Up-Converter**
  - 50/60-Hz Progressive output with Line-Interpolation (A + A*), Field-Merging (A + B) or with Motion-adaptive De-interlacing based on median f(A, B)
  - Advanced Still Picture modes: AA*AA* and ABAB interlaced or AAAA non-interlaced
  - Automatic Movie mode detection and scanning
- **Standard Definition Input**
  - ITU-R BT.656/601 Video Input
  - Separate H/V inputs synchronous with input clock
  - 3D Temporal Noise Reduction with Comet-effect Correction
  - Movie Mode Detection with Motion Phase Recovery
  - Scene-change Detector for Contrast Enhancer and Up-conversion Control
  - Letterbox Format Detection and Auto-Format Correction
- **High-Quality Video Display**
  - Picture Structure Improvement including Color Transition Improvement, Luma Peaking/Coring and Luma Contrast Enhancer
  - H/V format conversion with Zoom In/Out (4x to 1/8x) with H/V decimation
  - Letterbox and 4:3 to 16:9 format conversion with programmable 5-segment Panoramic mode
  - Very flexible Sync Generator for Master and Slave modes by Vsync and Hsync signals generation
  - Progressive Display mode (60 Hz, 50 Hz) for full-screen graphic planes
  - Mosaic mode with up to 16 pictures displayed
- Picture Compositor to provide Transparency mode between Video and Graphic planes
- **High-Performance 8-bit Bitmap OSD Generator**
  - Pixel-based resolution with 10-bit RGB outputs
  - Programmable Resolution up to WXGA, all standard displays are supported:
    - Teletext 1.5 (480x520) and 2.5 (672x520)
    - Double-page Teletext (960x520) with Picture-and-Text
    - TeleWeb (640x480)
  - 4 graphic planes with full alpha-blending capabilities:
    - 24-bit Background Plane
    - 10-bit RGB Video Plane
    - Bitmap OSD Plane with Color Map
    - Up to 128 x 128 pixel Cursor Plane
  - 2D Graphics Accelerator
- **Embedded 32-bit ST20 CPU Core**
- **Peripherals and I/Os for TV Chassis Control:**
  - 30 fully-programmable I/Os (5V tolerant)
  - 4 external interrupts
  - 8-bit programmable PWM with 4 inputs/outputs
  - Infrared Digital Preprocessor
  - Real Time Clock and Watchdog Timer
  - 4 16-bit standard timers
  - 10-bit ADC with 6 inputs and wake-up capability
  - 2 Master/Slave I²C Bus Interfaces
  - UART and support for IrDA interfaces
- **Teletext 1.5 and 2.5, Closed-Caption, VPS and WSS VBI Data Decoding, TeleWeb Compliant**
- **Embedded Emulation Resources with In-Situ Flash Programming Capabilities**
- **1.8V and 3.3V Power supplies**
- **Eco Standby mode**
- **27-MHz Crystal Oscillator**
- **PC input compatible**

1 General Information

1.1 Introduction

This integrated circuit (IC) is dedicated to flat panel display TV chassis. Combined with a digital multi-standard video decoder (STV2310) delivering an ITU-R BT.601/656 video stream, it provides a cost-effective, high-performance solution for plasma or LCD TV applications. The STV3550 includes an up-converter, a 32-bit ST20 CPU core with all peripherals required for controlling the TV chassis. Teletext data is extracted from the incoming stream and decoded by the CPU. An embedded On-Screen Display (OSD) generator delivers the text and graphics. The Video Display Pipeline performs feature box image processing such as picture improvement, horizontal and vertical rescaling and Temporal Noise Reduction.

The chip operates with an external SDRAM that is used for the field-rate up-conversion and text and graphic generations.The external SDRAM can be configured as a single bank of 16/64/128 Mb (16-bit configuration) or a dual bank of 16/64 or 128 Mb (32-bit configuration). Application program codes are stored in an external Flash memory.

The STV3550 is designed using an 0.18 micron CMOS process and delivered in a 208-pin PQFP (0.5 mm pitch) package.

The STV3550 completes the Digital IC Core family (STi5xxx, STi7xxx) which offers common CPU and software platforms based on STMicroelectronics' 32-bit ST20 CPU core. This device, which is specifically designed for plasma or LCD TV applications, is completely compatible with the architecture and software of the STV3500 CTV100-CRT platform that targets CRT-based TV chassis.

Figure 1: Top Level Diagram

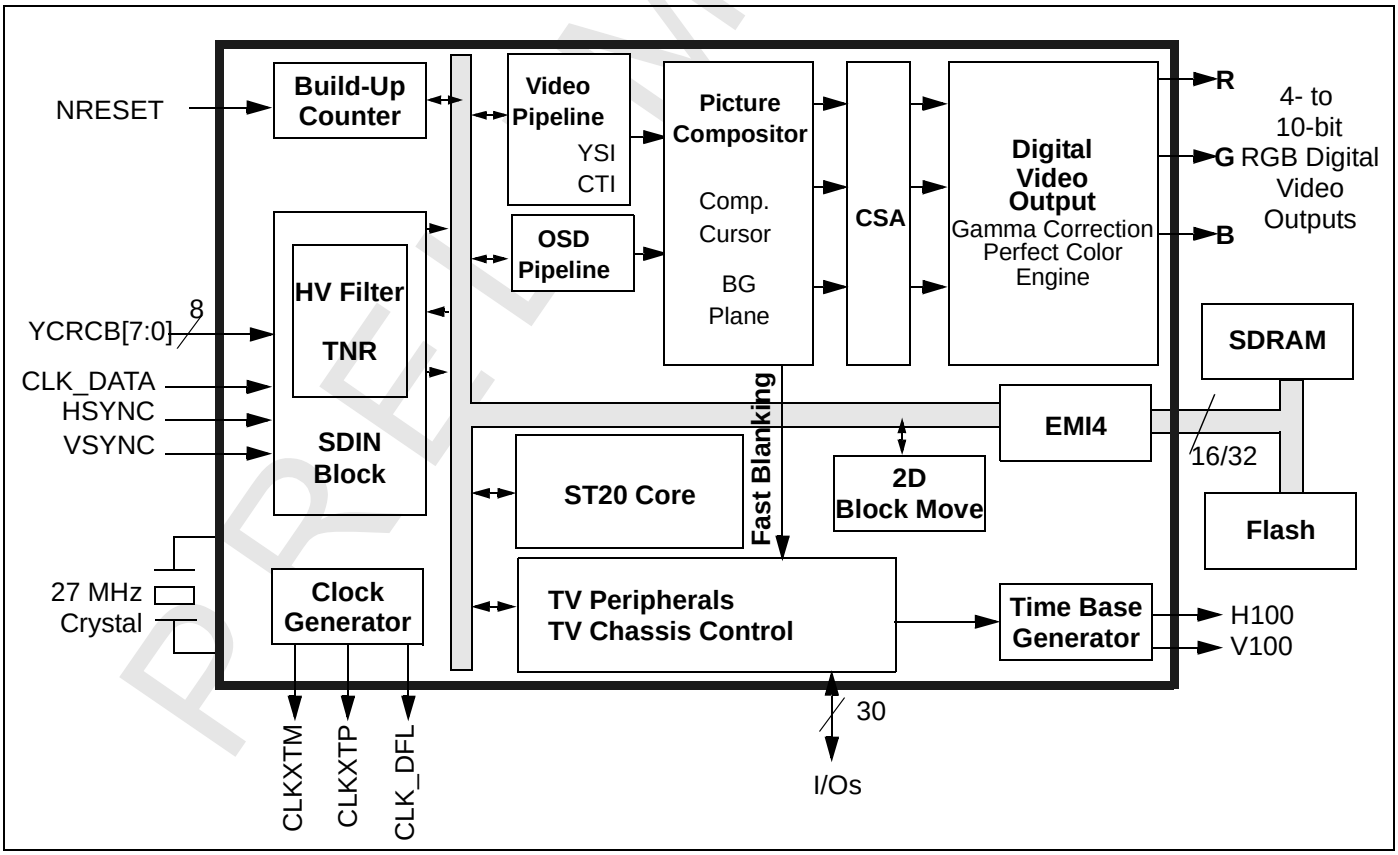
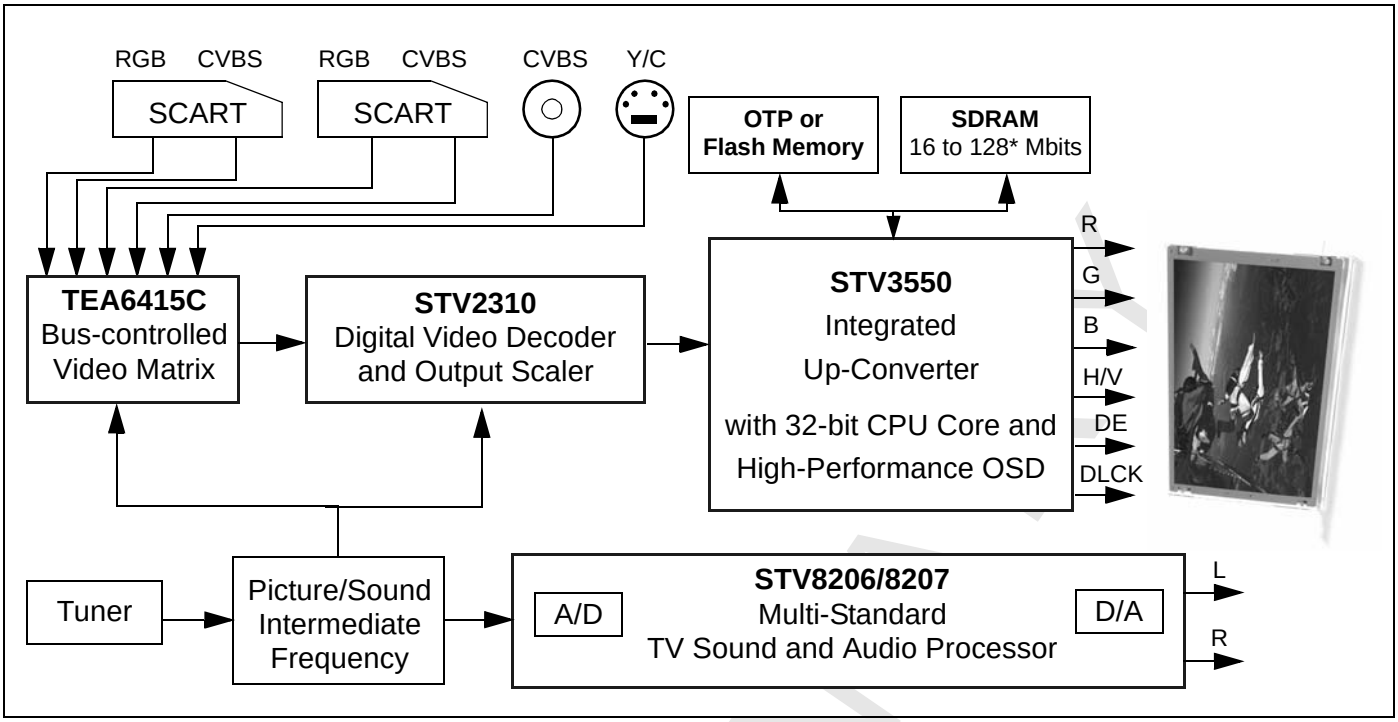


Figure 2: CTV100-LCD Platform Diagram Example



* *one or two banks*

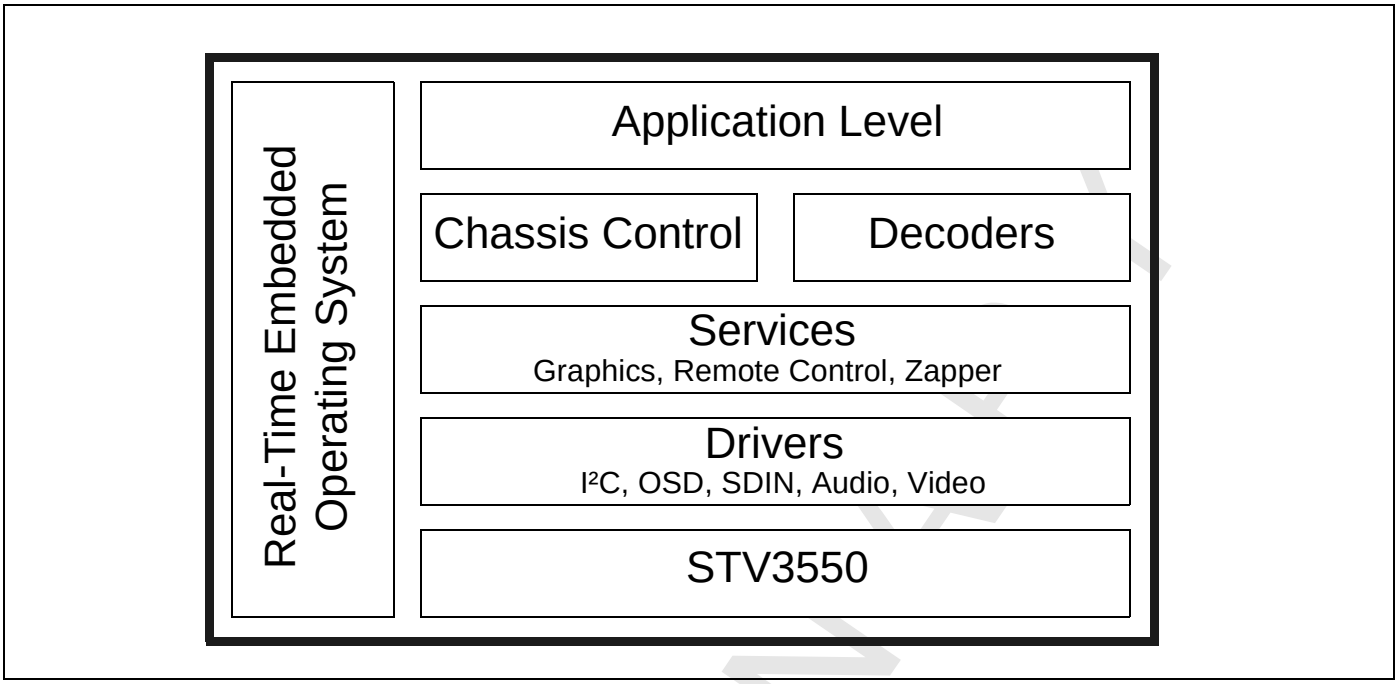
1.2 Software

The layering model adopted for the CTV100-LCD Software Stack is based on certain non-functional requirements:

- Re-usable
- Portable
- Modular
- Reliable and robust

- Readable and maintainable

Figure 3: Global Software Architecture

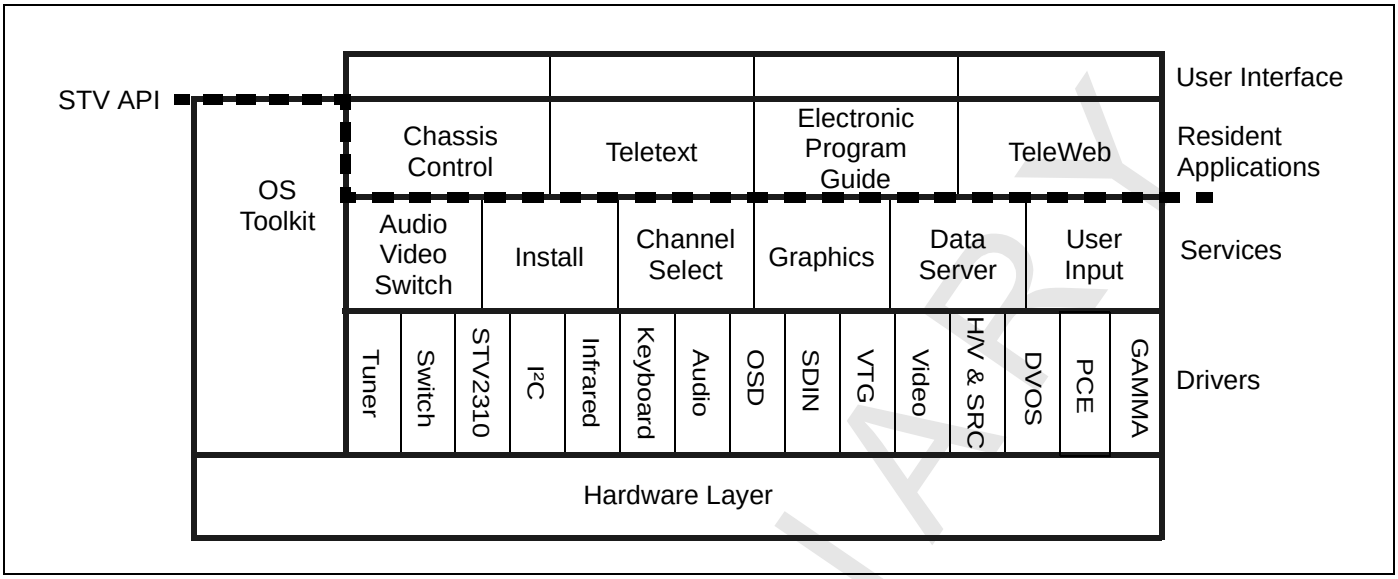


The CTV100-LCD application software consists of 4 main layers based on these non-functional requirements:

- System Layer provides certain general-purpose components such as Handle or Link List Managers. This layer also contains the Operating System Abstraction Layer (OSAL) components which enable other layers to be OS-independent.
- Driver Layer provides a hardware abstraction to the upper layers making them hardware-independent.
- Service Layer contains the components that provide the Application layer with high level interfaces in order to manage the TV set. The set of Service components included in the demonstration application are very useful for developing applications.

- Application Layer which contains the software that defines the “Look & Feel” of the TV set. This layer, for instance, contains the components that are responsible for the following features: interpretation of user inputs, display and navigation functions and Teletext applications.

Figure 4: Software Architecture Example



1.3 Related Documentation

There are several documents available that explain the various software capabilities. There are three different types of documents:

1. General Introduction Manual
2. User Guides
3. Reference Guide

1.3.1 General Introduction Manual

This document describes the general architecture of the features listed in Section 1.2: Software.

It also describes the relationships between the various service, driver, system and application modules. It lists all the available modules and their version number. A brief description of each function is also provided.

1.3.2 User Guides

A user guide is provided for each set of selected features. These documents will enable the user to get started with the relevant features. An example code is also included as well as a troubleshooting section.

A list (non-exhaustive) of existing user guides is given below.

1. CTV100-LCD: STV3550 User Controls User Guide
   - ➢ The Remote Control Service is responsible for the byte assembly of infrared frames coming from a Remote Control device. It provides the client with a high-level interface to manage Remote Control inputs.
   - ➢ The User Interface Service provides the client with a high-level interface to manage both Remote Control and Keyboard inputs. The client provides its own code, with which it will be notified.
2. CTV100-LCD: STV3550 Video and Sync User Guide

- ➢ The Video Service provides the client with a high-level interface for managing the Video.

  This service is responsible for the complete management of the video buffers and the video display, according to client configuration, in terms of the video mode (Up-conversion, Proscan...).

  It also offers video-related features, such as zoom or freeze frame, among others.

- ➢ The Synchronization Service is responsible for programming the field polarity. It provides the client with a high level interface for managing the Field Polarity sequences.

3. CTV100-LCD: STV3550 Source Selection User Guide

   - ➢ The Audio/Video Switch service is responsible for managing the connections between audio and video inputs and outputs by using drivers that provide audio/video switching functions.

     The Audio/Video Switch component is also responsible for detecting changes in the Slow Blanking Level (available on SCART connections) by using the ADC driver.

   - ➢ The Front End service is responsible for the tuning and the scanning operations. It provides the client with a high level interface working in the frequency domain.

4. CTV100-LCD: STV3550 Graphics User Guide

   - ➢ It provides the necessary software for creating cursors and graphic planes using respectively the Cursor Driver (DV_GAM) and the On-Screen Display Driver (STOSD).

   - ➢ It also includes the Two Dimensional Block Move Driver (DV_BME) in order to improve graphical applications which often require moving blocks of data.

     This document will describe the CTV100-LCD Basic Graphics Stack and also provide a guide for the implementation and configuration of a graphical user interface (GUI) stack based on the OSD.

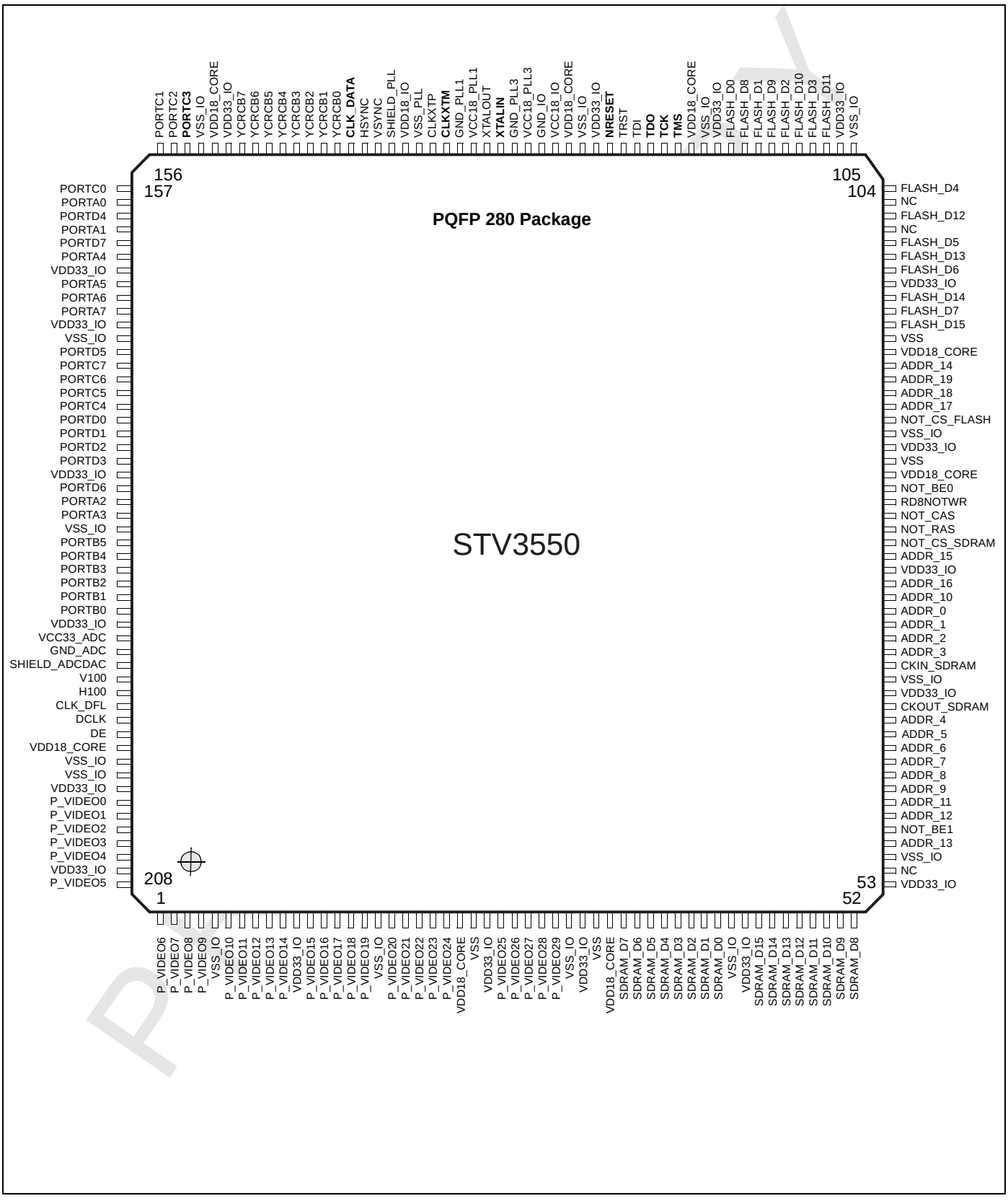
1.3.3 Reference Guide

There is one reference guide for each component (service, driver, etc.). This document includes the API as well as an Example of Use. This document specifically targets design engineers.

2 STV3550 Pin List

2.1 Pinout

Figure 5: Pinout Diagram



2.2 Pin Description

Table 1: Digital Video Input Stage

Pin No.	Pin Name	Pin Description
140	VSYNC	Vertical Sync Input
141	HSYNC	Horizontal Sync Input
142	CLK_DATA	Video Input Clock from STV2310
143	YCRCB0	4:2:2 Data Stream Input 0 from STV2310
144	YCRCB1	4:2:2 Data Stream Input 1 from STV2310
145	YCRCB2	4:2:2 Data Stream Input 2 from STV2310
146	YCRCB3	4:2:2 Data Stream Input 3 from STV2310
147	YCRCB4	4:2:2 Data Stream Input 4 from STV2310
148	YCRCB5	4:2:2 Data Stream Input 5 from STV2310
149	YCRCB6	4:2:2 Data Stream Input 6 from STV2310
150	YCRCB7	4:2:2 Data Stream Input 7 from STV2310

Table 2: Digital Video Output Stage

Pin No.	Pin Name	Pin Description
Digital		
193	V100	Vertical Sync Output
194	H100	Horizontal Sync Output
195	CLK_DFL	Clock Output
202	P_VIDEO0	Digital Video Output 0
203	P_VIDEO1	Digital Video Output 1
204	P_VIDEO2	Digital Video Output 2
205	P_VIDEO3	Digital Video Output 3
206	P_VIDEO4	Digital Video Output 4
208	P_VIDEO5	Digital Video Output 5
1	P_VIDEO6	Digital Video Output 6
2	P_VIDEO7	Digital Video Output 7
3	P_VIDEO8	Digital Video Output 8
4	P_VIDEO9	Digital Video Output 9
6	P_VIDEO10	Digital Video Output 10
7	P_VIDEO11	Digital Video Output 11
8	P_VIDEO12	Digital Video Output 12
9	P_VIDEO13	Digital Video Output 13
10	P_VIDEO14	Digital Video Output 14
12	P_VIDEO15	Digital Video Output 15
13	P_VIDEO16	Digital Video Output 16
14	P_VIDEO17	Digital Video Output 17
15	P_VIDEO18	Digital Video Output 18

Table 2: Digital Video Output Stage (continued)

Pin No.	Pin Name	Pin Description
16	P_VIDEO19	Digital Video Output 19
18	P_VIDEO20	Digital Video Output 20
19	P_VIDEO21	Digital Video Output 21
20	P_VIDEO22	Digital Video Output 22
21	P_VIDEO23	Digital Video Output 23
22	P_VIDEO24	Digital Video Output 24
26	P_VIDEO25	Digital Video Output 25
27	P_VIDEO26	Digital Video Output 26
28	P_VIDEO27	Digital Video Output 27
29	P_VIDEO28	Digital Video Output 28
30	P_VIDEO29	Digital Video Output 29
196	DCLK	Digital CMOS Clock
197	DE	Digital CMOS Data Enable

Table 3: Parallel Input/Output Pins

Pin No.	Pin Name	Main Function (after Reset)	Alternate Function
158	PORTA0	Port A0	PWMCapture0/PWM0
160	PORTA1	Port A1	PWMCapture1/INT2/PWM1
180	PORTA2	Port A2	PWMCapture2/INT3/extreg/PWM2
181	PORTA3	Port A3	PWMCapture3/PWM3
162	PORTA4	Port A4	UARTF TXD
164	PORTA5	Port A5	UARTF RXD/UARTF TXD Smartcard
165	PORTA6	Port A6	UARTF CTS
166	PORTA7	Port A7	UARTF RTS
188	PORTB0	Port B0	AD_0
187	PORTB1	Port B1	AD_1
186	PORTB2	Port B2	AD_2
185	PORTB3	Port B3	AD_3
184	PORTB4	Port B4	AD_4 / Timer Output 0
183	PORTB5	Port B5	AD_5/ Timer Output 1
157	PORTC0	Port C0	SDA_0
156	PORTC1	Port C1	SCL_0
155	PORTC2	Port C2	SDA_1
154	PORTC3	Port C3	SCL_1
173	PORTC4	Port C4	SDA_2
172	PORTC5	Port C5	SCL_2
171	PORTC6	Port C6	SDA_3
170	PORTC7	Port C7	SCL_3
174	PORTD0	Port D0	Timer Input 0

Table 3: Parallel Input/Output Pins (continued)

Pin No.	Pin Name	Main Function (after Reset)	Alternate Function
175	PORTD1	Port D1	Timer Input 1
176	PORTD2	Port D2	Timer Input 2 / Timer Output 2
177	PORTD3	Port D3	Timer Input 3/ Timer Output 3
159	PORTD4	Port D4	IR_in
169	PORTD5	Port D5	GFX_ACTIVE
179	PORTD6	Port D6	INT0
161	PORTD7	Port D7	INT1/16 x UART Clock

Table 4: External Memory Interface Pins

Pin No.	Pin Name	Pin Description
Flash Data Bus		
114	FLASH_D0	Flash Data Bus 0
112	FLASH_D1	Flash Data Bus 1
110	FLASH_D2	Flash Data Bus 2
108	FLASH_D3	Flash Data Bus 3
104	FLASH_D4	Flash Data Bus 4
100	FLASH_D5	Flash Data Bus 5
98	FLASH_D6	Flash Data Bus 6
95	FLASH_D7	Flash Data Bus 7
113	FLASH_D8	Flash Data Bus 8
111	FLASH_D9	Flash Data Bus 9
109	FLASH_D10	Flash Data Bus 10
107	FLASH_D11	Flash Data Bus 11
102	FLASH_D12	Flash Data Bus 12
99	FLASH_D13	Flash Data Bus 13
96	FLASH_D14	Flash Data Bus 14
94	FLASH_D15	Flash Data Bus 15
SDRAM Data Bus		
42	SDRAM_D0	SDRAM Data Bus 0
41	SDRAM_D1	SDRAM Data Bus 1
40	SDRAM_D2	SDRAM Data Bus 2
39	SDRAM_D3	SDRAM Data Bus 3
38	SDRAM_D4	SDRAM Data Bus 4
37	SDRAM_D5	SDRAM Data Bus 5
36	SDRAM_D6	SDRAM Data Bus 6
35	SDRAM_D7	SDRAM Data Bus 7
52	SDRAM_D8	SDRAM Data Bus 8
51	SDRAM_D9	SDRAM Data Bus 9
50	SDRAM_D10	SDRAM Data Bus 10

Table 4: External Memory Interface Pins (continued)

Pin No.	Pin Name	Pin Description
49	SDRAM_D11	SDRAM Data Bus 11
48	SDRAM_D12	SDRAM Data Bus 12
47	SDRAM_D13	SDRAM Data Bus 13
46	SDRAM_D14	SDRAM Data Bus 14
45	SDRAM_D15	SDRAM Data Bus 15
Address Bus		
73	ADDR_0	Address Bus 0
72	ADDR_1	Address Bus 1
71	ADDR_2	Address Bus 2
70	ADDR_3	Address Bus 3
65	ADDR_4	Address Bus 4
64	ADDR_5	Address Bus 5
63	ADDR_6	Address Bus 6
62	ADDR_7	Address Bus 7
61	ADDR_8	Address Bus 8
60	ADDR_9	Address Bus 9
74	ADDR_10	Address Bus 10
59	ADDR_11	Address Bus 11
58	ADDR_12	Address Bus 12
56	ADDR_13	Address Bus 13
91	ADDR_14	Address Bus 14
77	ADDR_15	Address Bus 15 / BA0
75	ADDR_16	Address Bus 16 / BA1
88	ADDR_17	Address Bus 17
89	ADDR_18	Address Bus 18 / Not_BE2
90	ADDR_19	Address Bus 19 / Not_BE3
Controls		
81	RD_NOTWR	SDRAM Write Enable / Flash Write Enable*
87	NOT_CS_FLASH	Flash Chip Select
78	NOT_CS_SDRAM	SDRAM Chip Select
79	NOT_RAS	SDRAM Row Address Strobe
80	NOT_CAS	SDRAM Column Address Strobe / Flash Output Enable*
82	NOT_BE0	SDRAM Byte Enable 0
57	NOT_BE1	SDRAM Byte Enable 1
66	CKOUT_SDRAM	Clock Output for SDRAM
69	CKIN_SDRAM	SDRAM Clock Feedback

Table 5: System Controls

Pin No.	Pin Name	Pin Description
132	XTALOUT	27 MHz Crystal Output
131	XTALIN	27 MHz Crystal Input
135	CLKXTM	27 MHz Differential Clock for STV2310
136	CLKXTP	27 MHz Differential Clock for STV2310
119	TCK	Test Clock Input
121	TDI	Test Data Input
120	TDO	Test Data Output
118	TMS	Test Mode Select Input
122	TRST	Test Reset Input
123	NRESET	STV3500 Reset Input (Active Low)

Table 6: Power Supplies

Pin No.	Pin Name	Pin Description
Analog		
192	SHIELD_ADCDAC	
139	SHIELD_PLL	To connect to Analog Ground Supply for PLL
190	VCC33_ADC	3.3 V Analog Voltage Supply for A/D Converter
191	GND_ADC	Analog Ground Supply for ADC
133	VCC18_PLL1	1.8 V Analog Voltage Supply for PLL
134	GND_PLL1	Analog Ground Supply for PLL
129	VCC18_PLL3	1.8 V Analog Voltage Supply for PLL
130	GND_PLL3	Analog Ground Supply for PLL
127	VCC18_IO	1.8 V Analog Voltage Supply for PLL I/Os
128	GND_IO	Analog Ground Supply for PLL I/Os
138	VDD18_PLL	1.8 V Analog Voltage Supply for PLL
137	VSS_PLL	Analog Ground Supply for PLL
Digital		
11	VDD33_IO	3.3 V Digital Voltage Supply
25	VDD33_IO	3.3 V Digital Voltage Supply
32	VDD33_IO	3.3 V Digital Voltage Supply
44	VDD33_IO	3.3 V Digital Voltage Supply
53	VDD33_IO	3.3 V Digital Voltage Supply
67	VDD33_IO	3.3 V Digital Voltage Supply
76	VDD33_IO	3.3 V Digital Voltage Supply
85	VDD33_IO	3.3 V Digital Voltage Supply
97	VDD33_IO	3.3 V Digital Voltage Supply
106	VDD33_IO	3.3 V Digital Voltage Supply
115	VDD33_IO	3.3 V Digital Voltage Supply
124	VDD33_IO	3.3 V Digital Voltage Supply

Table 6: Power Supplies (continued)

Pin No.	Pin Name	Pin Description
151	VDD33_IO	3.3 V Digital Voltage Supply
163	VDD33_IO	3.3 V Digital Voltage Supply
167	VDD33_IO	3.3 V Digital Voltage Supply
178	VDD33_IO	3.3 V Digital Voltage Supply
189	VDD33_IO	3.3 V Digital Voltage Supply
201	VDD33_IO	3.3 V Digital Voltage Supply
207	VDD33_IO	3.3 V Digital Voltage Supply
5	VSS_IO	Digital Ground Supply
17	VSS_IO	Digital Ground Supply
31	VSS_IO	Digital Ground Supply
43	VSS_IO	Digital Ground Supply
55	VSS_IO	Digital Ground Supply
68	VSS_IO	Digital Ground Supply
86	VSS_IO	Digital Ground Supply
105	VSS_IO	Digital Ground Supply
116	VSS_IO	Digital Ground Supply
125	VSS_IO	Digital Ground Supply
153	VSS_IO	Digital Ground Supply
168	VSS_IO	Digital Ground Supply
182	VSS_IO	Digital Ground Supply
199	VSS_IO	Digital Ground Supply
200	VSS_IO	Digital Ground Supply
23	VDD18_CORE	1.8 V Digital Voltage Supply
34	VDD18_CORE	1.8 V Digital Voltage Supply
83	VDD18_CORE	1.8 V Digital Voltage Supply
92	VDD18_CORE	1.8 V Digital Voltage Supply
117	VDD18_CORE	1.8 V Digital Voltage Supply
126	VDD18_CORE	1.8 V Digital Voltage Supply
152	VDD18_CORE	1.8 V Digital Voltage Supply
198	VDD18_CORE	1.8 V Digital Voltage Supply
24	VSS	Digital Ground Supply
33	VSS	Digital Ground Supply
84	VSS	Digital Ground Supply
93	VSS	Digital Ground Supply

Table 7: Not Connected

Pin No.	Pin Name	Pin Description
54	NC	
101	NC	
103	NC	

2.3 Parallel I/O Pins and Alternate Functions

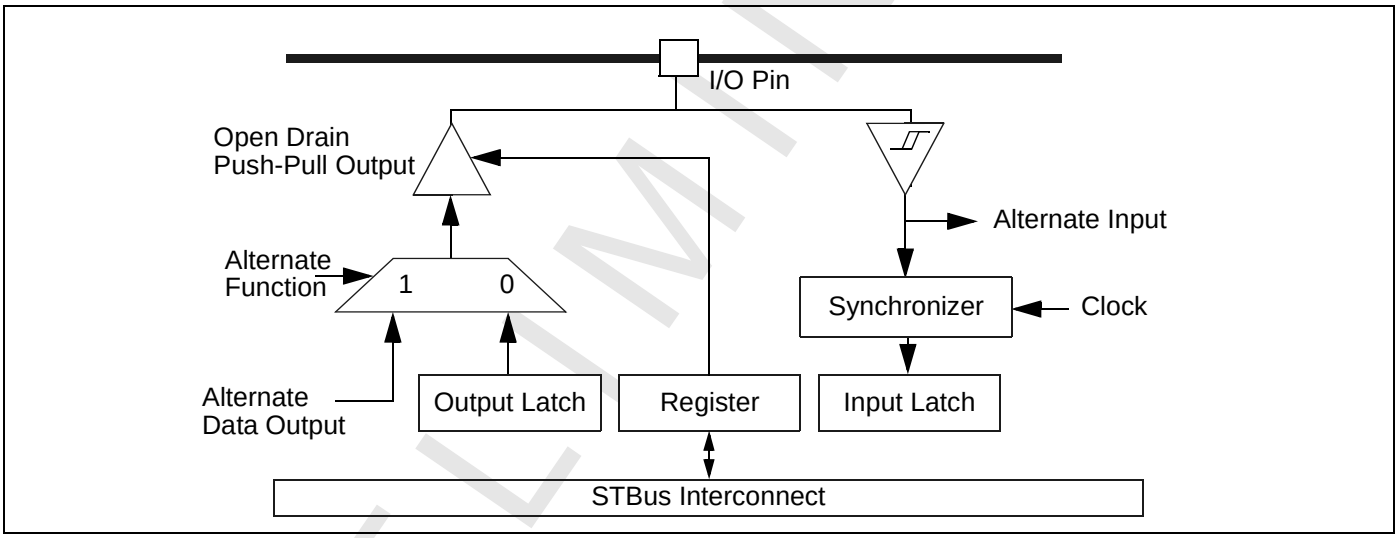
The STV3550 includes 30 Parallel I/O pins arranged in 4 banks of 6 to 8-bit ports. The pins can be programmable in input, output, bi-directional or as alternate function pins. The ports are listed in Table 3 on page 14.

When configured as an output, the pin can be either open drain or with weak pull-up.

The input to any pin can be compared to a stored value, and if not equal, can produce an interrupt.

Any pin can be configured to output an alternate functions rather than programmed value. The input channel can be directly connected to a peripheral device to make the pin as a primary input of the device. The following table shows the assignments of the alternate functions for the 30 I/O pins.

Figure 6: Pin Connections



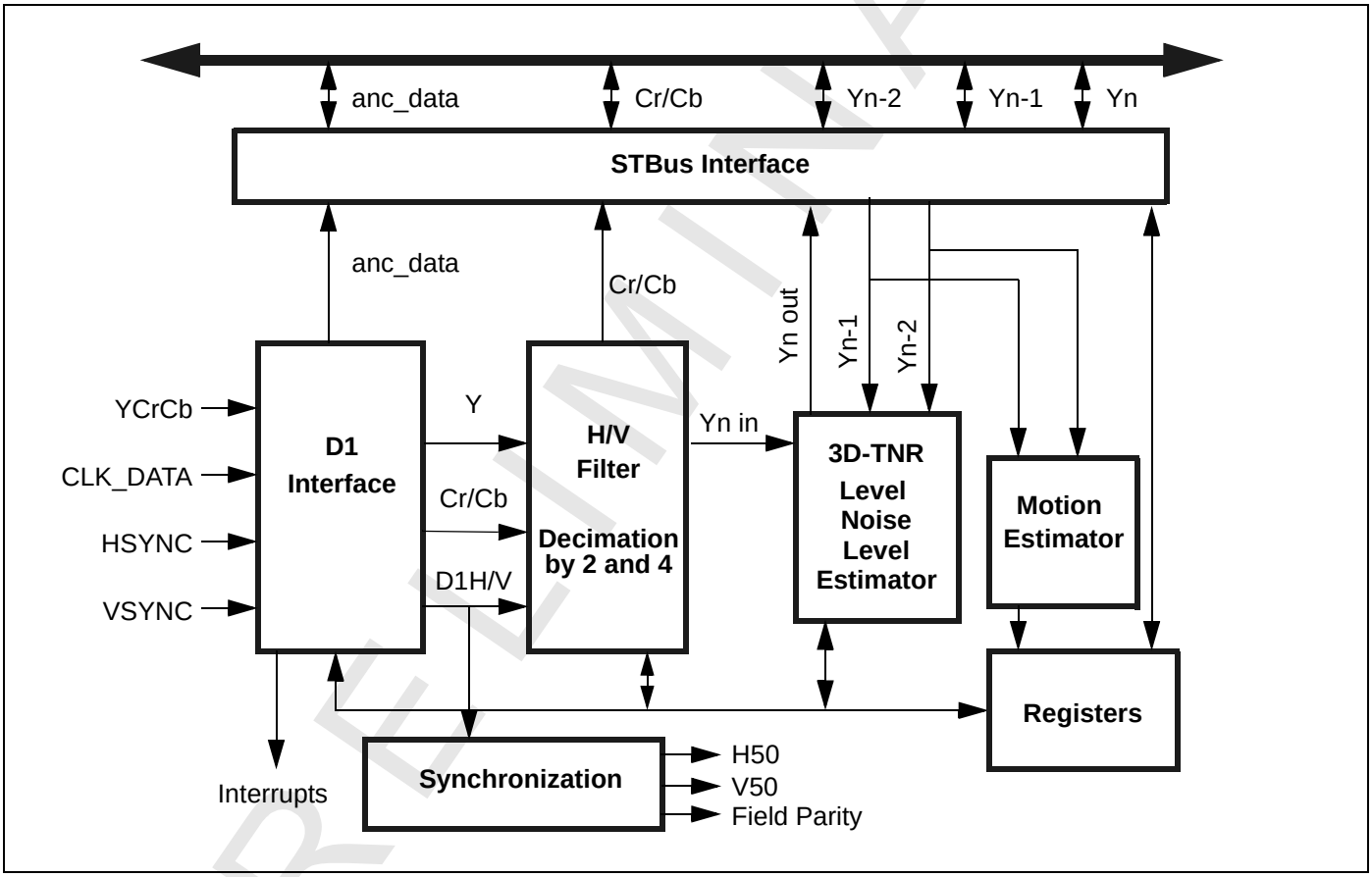
3 Video Functional Description

3.1 Standard Definition Input (SDIN)

The Standard Definition Input (SDIN) processes video signals before their storage in the external SDRAM. The SDIN includes the following features and functions:

- 4:2:2 D1 Input Stream De-multiplexing including ancillary data extraction
- Horizontal and Vertical Decimation by 2 and 4 with horizontal and vertical anti-aliasing filters (with programmable coefficients)
- 3D-Temporal Noise Reduction based on a proprietary ST algorithm
- Noise Level Estimator used to control the 3D-TNR through a software loop
- Motion Estimator to perform film mode detection or scene change detection through a software loop

Figure 7: SDIN Block Diagram



3.1.1 System Description

3.1.1.1 D1 Interface

The D1 interface extracts the luma and chroma pixels from the 8-bit data stream as well as the ancillary data embedded in the horizontal and vertical blanking interval (VBI). This data includes VBI data such as Teletext, Closed Caption, WSS, VPS, etc... and sync pulses for horizontal and vertical synchronization.

The ancillary data is stored aligned in 32-bit packets. If necessary, the packets are completed by a filler with programmable data (060h, by default).

The parity of each field is extracted from the embedded F pulse or from the external H/V pulse. The field can be determined as top first or bottom first by programming.

The data stream sequence for the Y, Cr or Cb samples is programmable.

3.1.1.2 D1 Formats

The typical input format supported by the SDIN is 720 pixels per line and 480 lines per frame for the 60-Hz source or 576 lines per frame for 50-Hz source (ITU-R BT.656) with a 27-MHz pixel clock.

The D1 interface can also support other formats with a pixel clock up to 30 MHz. The input format is given by the configuration of the interface that is programmable up to 960 pixels per line and 1024 lines per frame.

The 3D-TNR and the Motion Estimator can be used only with standardized line length of 720 pixels per line.

3.1.1.3 Synchronization Pulse Extraction

The SDIN interface is able to extract either the sync pulses embedded in the data stream as described in specification ITU-R BT.656 or external H/V pulses sent on the H/V inputs.

After extraction, the sync pulses are used to capture the data which is then sent to the Video Timebase Generator (VTG) to synchronize the chip. Synchronization modes are described in Section 3.2: Video Timebase Generator (VTG).

The following sync pulses can be extracted and are selected using the registers:

- Embedded F and H sync pulses (for the vertical sweep, and EAV or SAV pulse for the horizontal sweep),
- Embedded V and H sync pulses (rising or falling edge of the V sync pulse, for the vertical sweep, and EAV or SAV pulse for the horizontal sweep),
- External H/Vsync pulses (rising or falling edge of the external V sync pulse, for the vertical sweep, and rising or falling edge of the external H pulse for the horizontal sweep),
- External H/Fsync pulses (rising and falling edge of the external F sync pulse, for the vertical sweep, and rising or falling edge of the external H pulse for the horizontal sweep).

3.1.1.4 Horizontal and Vertical Filter

To perform high-factor Zoom-out on a picture (from 1/4 to 1/16 of the screen) with no extra load on the memory bandwidth, the SDIN can perform a horizontal and vertical decimation by 2 or 4 on the input signal. This means that unused pixels are not stored in the SDRAM and the memory bandwidth is not penalized.

Before decimation, the input signal passes through a horizontal and vertical filter to prevent aliasing effects. The coefficient of this filter is programmable and then can be optimized for each decimation factor. This filter and the decimator can be by-passed. The decimation factor is selectable by programming. When the decimator is active, the 3D-TNR is by-passed.

The vertical filter line-memories are limited to 720 pixels. When a line exceeds 720 pixels, the input line must be truncated to 720 pixels by the D1 interface (configured at 720 pixels per line).

3.1.1.5 3D Temporal Noise Reduction (TNR)

When the D1 interface is set in a standard configuration (720 pixels per line and 288 or 240 lines per field) and the decimation filter is by-passed, the luma signal can be filtered in the spatial and temporal domain in order to reduce in-band noise. The noise reduction is based on a proprietary ST

algorithm that includes a comet-effect canceller. The strength of the noise reduction can be adjusted in 32 steps depending on the noise level.

The SDIN includes a Noise Estimator that delivers a 32-step data item on the noise level for each field. This data can be processed by the CPU to control the 3D-Temporal Noise Reductor and then performs an adaptive noise reduction. This function is available in standard configurations only.

3.1.1.6 Motion Estimator

When the D1 interface is set in a standard configuration (720 pixels per line and 288 or 240 lines per field), the SDIN provides scene change and video picture format data based on 3 consecutive fields.

This information can be processed by the CPU to determine if the signal is from a video or a film source, and then the CPU can select the optimized up-conversion mode (Proscan field merging or interpolation).

This data is also used to determine the polarity of the field: Top First (generally used) or Bottom First.

3.2 Video Timebase Generator (VTG)

This block generates the up-converted horizontal and vertical sync pulses required to drive the video output stages. It receives the extracted H/V sync pulses from the SDIN block, in the 1H domain, and, depending on the selected synchronization mode, provides H/V sync pulses for the 2H domain.

The VTG is able to generate all output synchronization pulses from 1H to 3H for horizontal sweep and 50 Hz to 120 Hz for Vertical sweep, in Interlaced or Progressive mode.

The VTG clock is derived from the on-board crystal oscillator to reduce jitter. Vertical and horizontal sync pulses are generated by the division of the VTG_CLK signal frequency.

The CPU manages the division factors and processes the incoming 50-Hz H and V pulses to generate the required 100-Hz scanning sequences.

3.2.1 Synchronization Modes

- Slave by H and V signals (embedded or external)
- Slave by V signals (embedded or external)
- Master mode.
- Pass-through (for VGA application) with generator-locking of the OSD clock.

3.2.2 Deinterlacing Modes and Progressive Scan Output

The de-interlacing can be done in the following modes:

- Spatial line interpolation, using the high resolution vertical polyphase filter
- Motion adaptive spatial-temporal interpolation, using the median filter
- Field merging for film sources

Note: The scrolling mode is done by programming the Video Timebase Generator (VTG) that defines the output number of lines/field, and by programming the Video Display Pipeline that performs the number of pixel/line interpolations.

The following table shows typical de-interlacing modes, according to the input format:

Table 8: De-interlacing and Scaling Modes

		INPUT						
	Panels & Output modes	**50 Hz interlaced Video**	**50 Hz interlaced movie**	**50 Hz n-interlaced game**	**60 Hz interlaced Video**	**60 Hz interlaced movie (3:2)**	**60 Hz interlaced movie (2:2)**	**60 Hz n-interlaced game**
OUTPUT	**VGA**	A+A*B*+B Or A+Med (A,B) Z out H 720>640 Z out V 576>480	A+B Z out H 720>640 Z out V 576>480	A+A*B*+B Or A+Med (A,B) Z out H 720>640 Z out V 576>480	A+A*B*+B Or A+Med (A,B) Z out H 720>640 Z out V 576>480	3:2pd Z out H 720>640 Z out V 576>480	A+B Z out H 720>640 Z out V 576>480	A+A*B*+B Or A+Med (A,B) Z out H 720>640 Z out V 576>480
	XGA	A+A*B*+B Or A+Med (A,B) Z in H 720>1024 Z in V 576>768	A+B Z in H 720>1024 Z in V 576>768	A+A*B*+B Or A+Med (A,B) Zin H 720>1024 Z in V 576>768	A+A*B*+B Or A+Med (A,B) Zin H 720>1024 Z in V 576>768	3:2pd Zin H 720>1024 Z in V 576>768	A+B Zin H 720>1024 Z in V 576>768	A+A*B*+B Or A+Med (A,B) Zin H720>1024 Z in V 576>768

Note: 720 pixels/line refers to ITU-R BT 601/656 video input standard resolution.

3.2.3 Regulation Modes

The regulation modes of the input and output dataflows are based on the configurations of several blocks such as the SDIN, VTG, Display, and also depends on the selected synchronization modes.

These regulation modes are then managed by a software layer that controls the various registers and includes algorithms to maintain the balance between the input and output dataflows. This service software layer allows various regulation modes as described below, selectable at the application level.

When the VTG block is set in Slave mode, the PLL that line-locks the clock pixel to the Hsync signal is used to regulate the input and output dataflows.

When the VTG block is set in Master mode, the following regulation modes are possible:

1. Frame Skip & Repeat Mode (used for Proscan mode)

   In this mode, depending on the speed variations between the input flow and the output flow, a frame is repeated or skipped from time to time (1 to 2 per second).

2. Field Skip & Repeat Mode (used for field rate up-conversion as 100 Hz)

   In this mode, depending on the speed variations between the input flow and the output flow, a field is repeated or skipped from time to time (1 to 2 per second).

3. Line Skip & Repeat Mode

   In this mode, certain lines are repeated or skipped during the VBI, depending on the delay between the input and output Vsync signals.

4. Pixel Skip & Repeat Mode

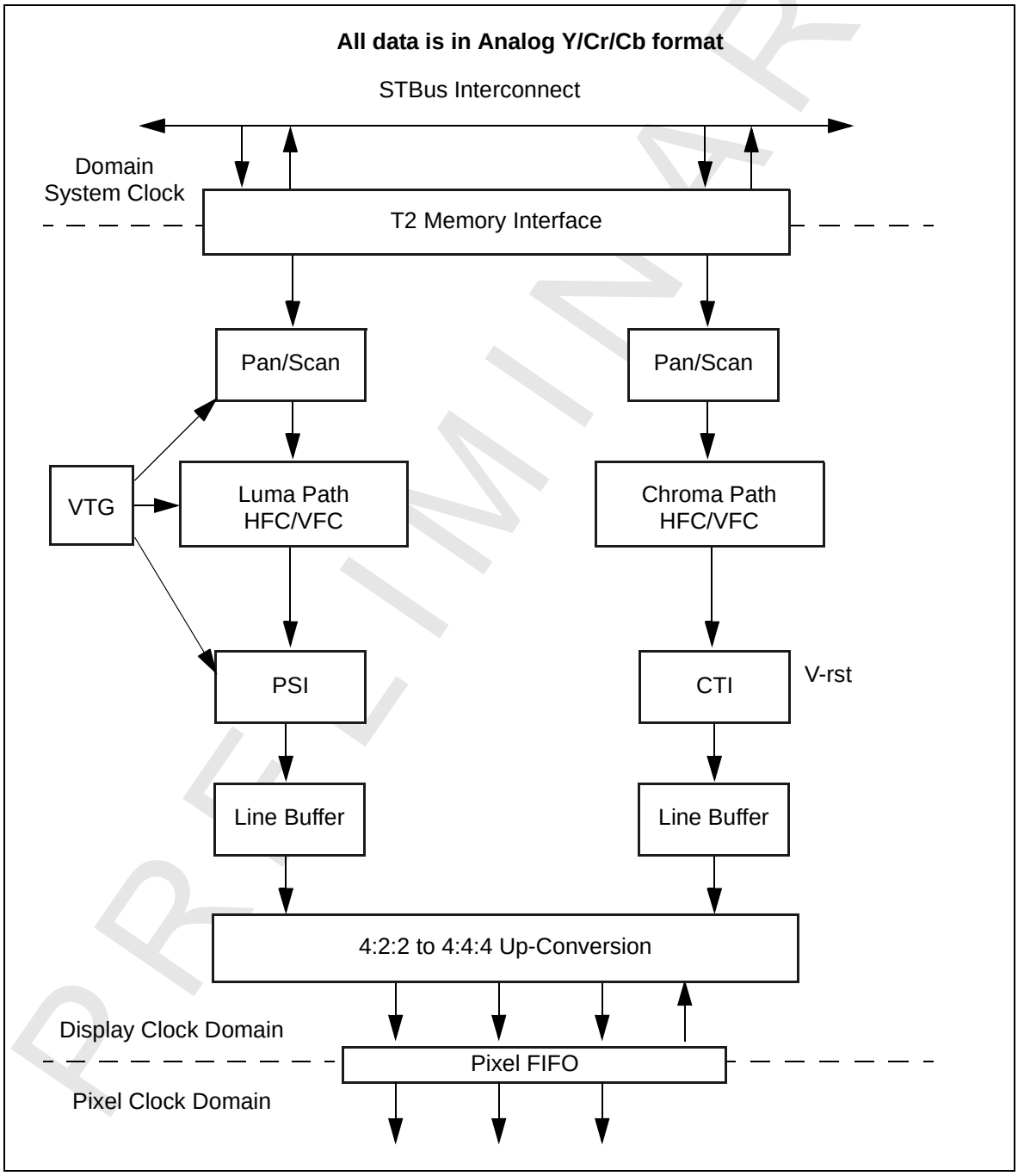
   In this mode; pixel periods are repeated or skipped during the horizontal or vertical blanking interval, depending on the delay between each Vsync signal.

3.3 Video Display Pipeline

The Video Display block includes all the processing required for adapting the incoming video stream to the selected display format. Depending on the up-conversion mode, this block generates the correct video sequence with the selected number of pixels per line, lines per field or fields per second. It uses high-resolution horizontal and vertical polyphase filters to interpolate the additional pixels and lines required for the selected display format. These filters are also used for zoom-in and zoom-out rescaling.

It also includes picture improvement algorithms (PSI/CTI) to provide the picture with a more “high-resolution” aspect.

Figure 8: Video Display Flowchart

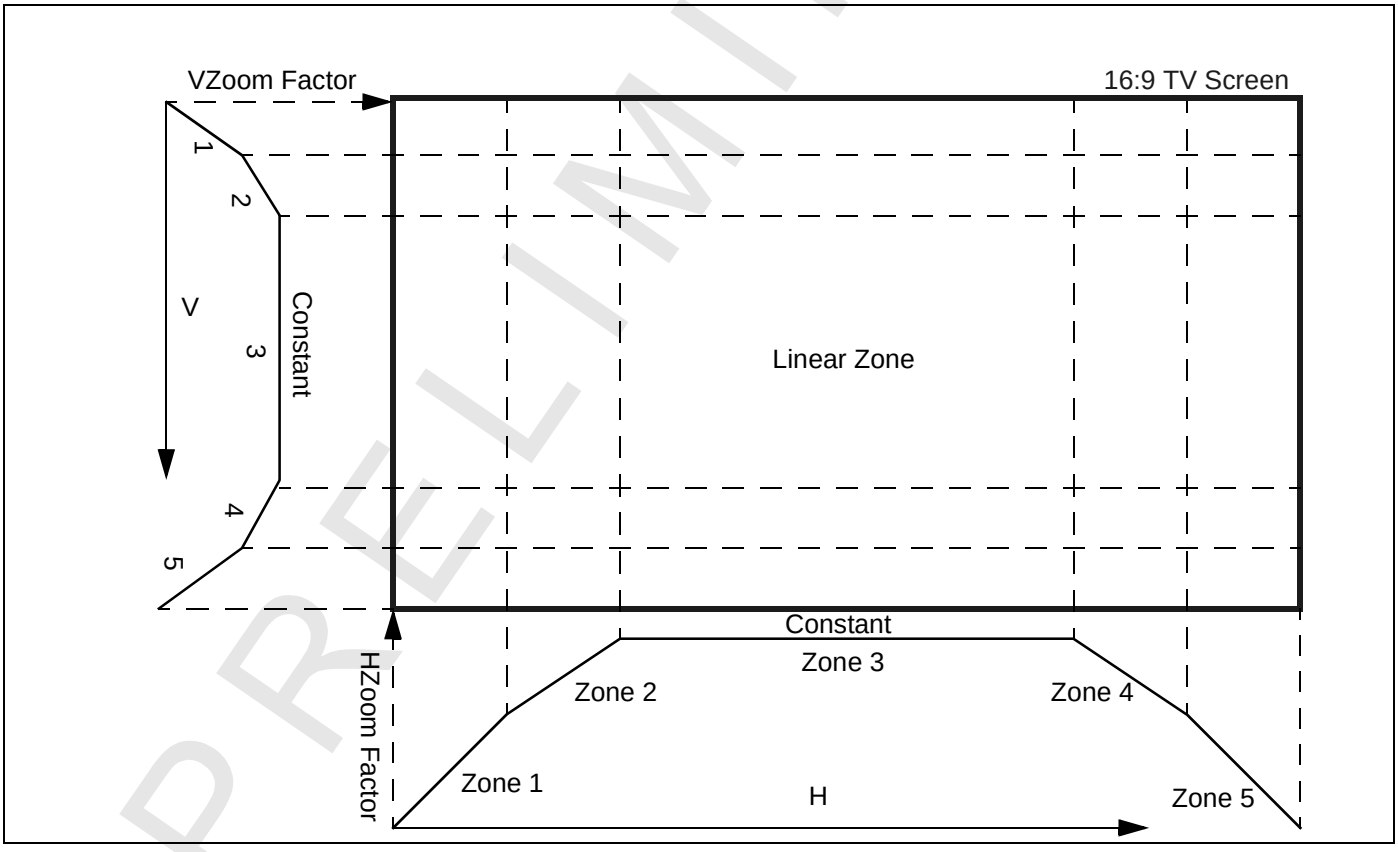


3.3.1 Main Features

- Horizontal resizing using a Sample Rate Converter based on an 8-tap, 32-phase polyphase filter with a programmable factor from 0.25 to 4.
- Vertical resizing using a Sample Rate Converter based on a 4-tap, 32-phase polyphase filter with a programmable factor from 0.5 to 4.
- Non-linear horizontal and vertical format conversion with up to 5 programmable segments.
- Horizontal video output resolution adjustable between 360 and 1920 active pixels per line by using a programmable pixel clock frequency.
- Vertical video output resolution adjustable between 240 and 1080 active lines per field.
- Field parity interpolation by controlling the phase offset in the vertical polyphase filter.
- Programmable Pan and Scan video positioning.
- Color Edge Replacement for Color Transient Improvement (CTI).
- Horizontal and Vertical Luma Peaking.
- Contrast Enhancement: Black, Grey, and White Stretch.
- 4:2:2 to 4:4:4 Chroma Up-sampling
- 10-bit RGB output

3.3.2 Horizontal and Vertical Rescaling

Figure 9: Zoom Rescaling Diagram



The rescaling function is performed by two 32-phase polyphase filters (Horizontal and Vertical), of which all coefficients are stored in a programmable table. In this case, if needed, the transfer function of the filter can be optimized by referring to a specific up-conversion process. This scaling

factor is programmable from 0.25 to 4 for horizontal scaling and 0.5 to 4 for vertical scaling with an accuracy of 1/8192. This allows for continuous linear zoom variations.

3.3.2.1 Non-Linear Zoom

The rescaling block performs the horizontal and vertical non-linear zoom as shown in Figure 9. The picture can be divided into 5 different segments in all directions.

3.3.2.2 Letter-box Format Detection

Letter-box format detection is included in order to automatically detect the horizontal black bars used with letter-box input video (Cinemascope,...) signals. This detection system is based on a proprietary algorithm that measures a certain level of data on a field basis.

This function can also detect video effects such as logos or subtitles inserted in the black parts of the video. The programmer can use this data to create an automatic format conversion for all the various cinema formats.

3.3.2.3 Format Conversion

The horizontal and vertical polyphase filters can be also used for the format conversions required for adapting the input video to the display format. Various examples are shown in the following figures. All formats are software-controlled and the letter-box format detector can also be used to automatically select the best-suited display format to prevent the horizontal black bars from being displayed.

Figure 10: Display Formats for 4:3 Input, 4:3 Aspect Ratio and 16:9 TV Screen

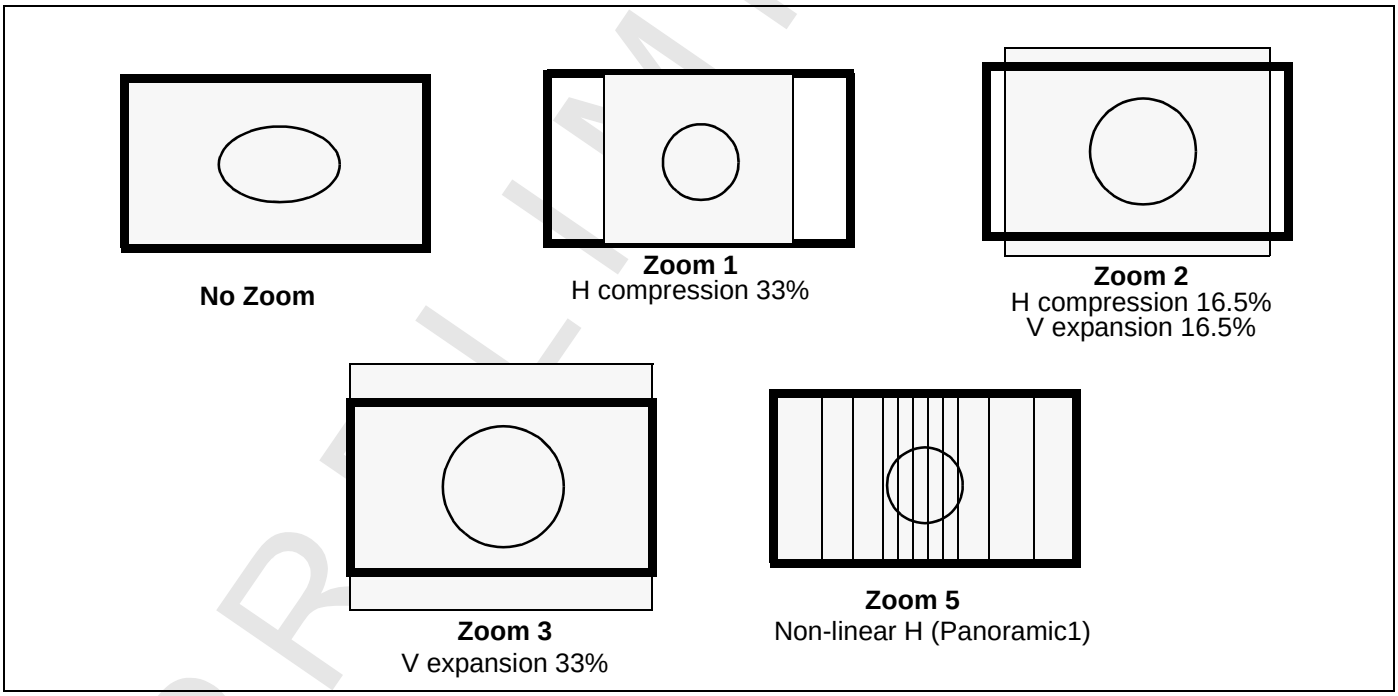


Figure 11: Display Formats for 4:3 Input, 16:9 Aspect Ratio (Letter-box) and 16:9 TV Screen

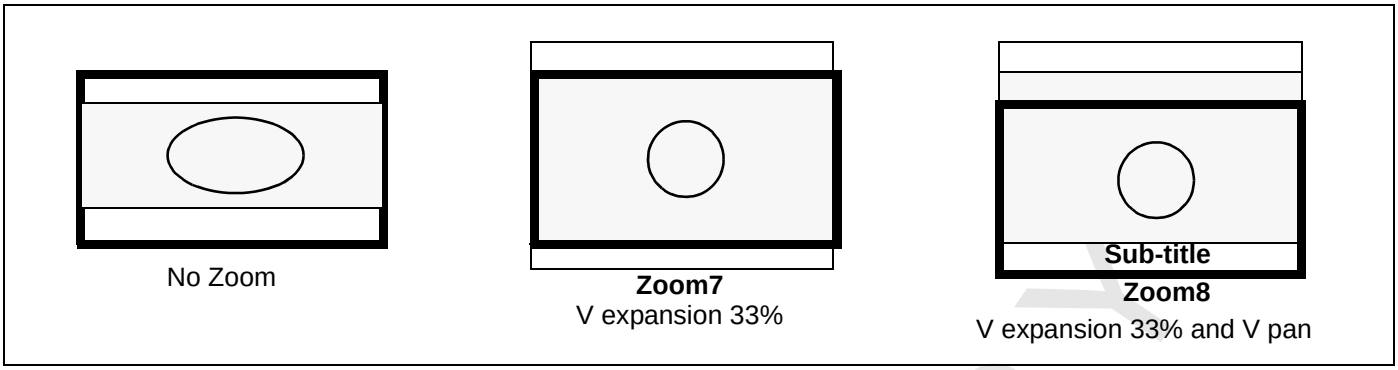


Figure 12: Display Formats for 4:3 Input, 2.35 Aspect Ratio (Wide Film) and 16:9 TV Screen

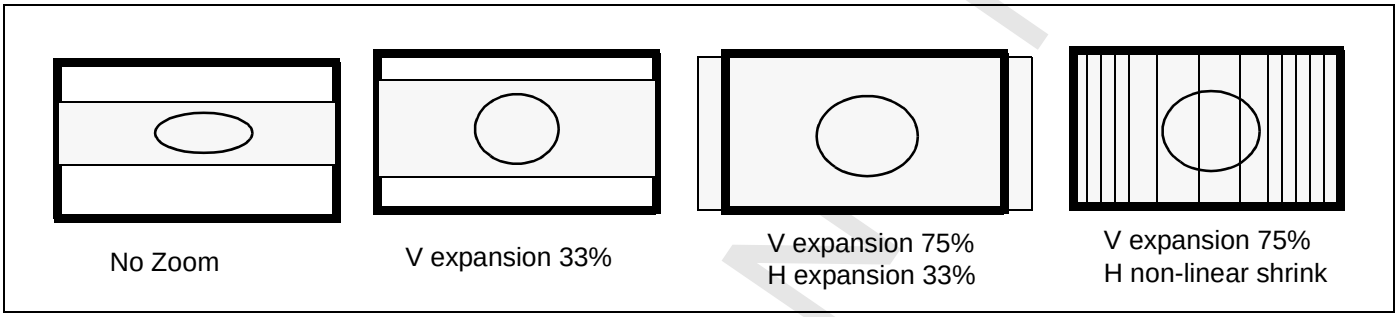
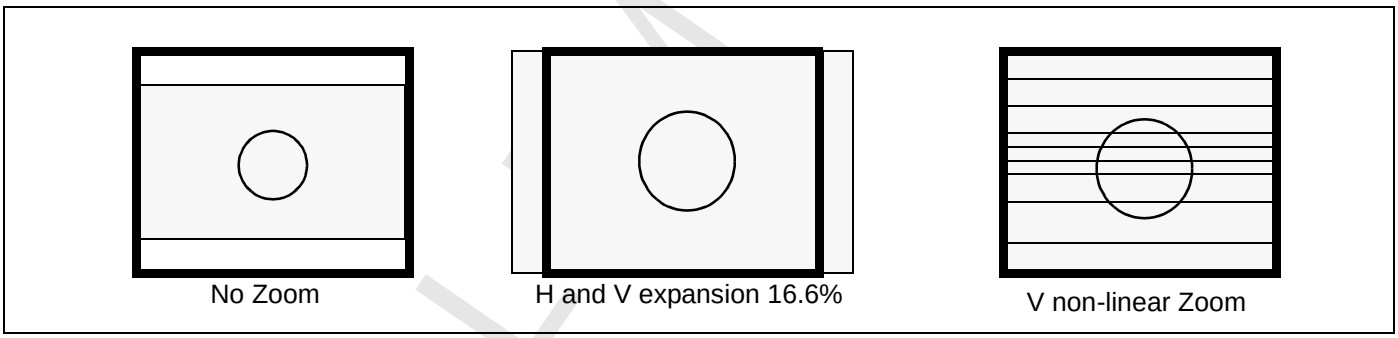


Figure 13: Display Formats for 4:3 Input, 16:9 Aspect Ratio and 4:3 TV Screen

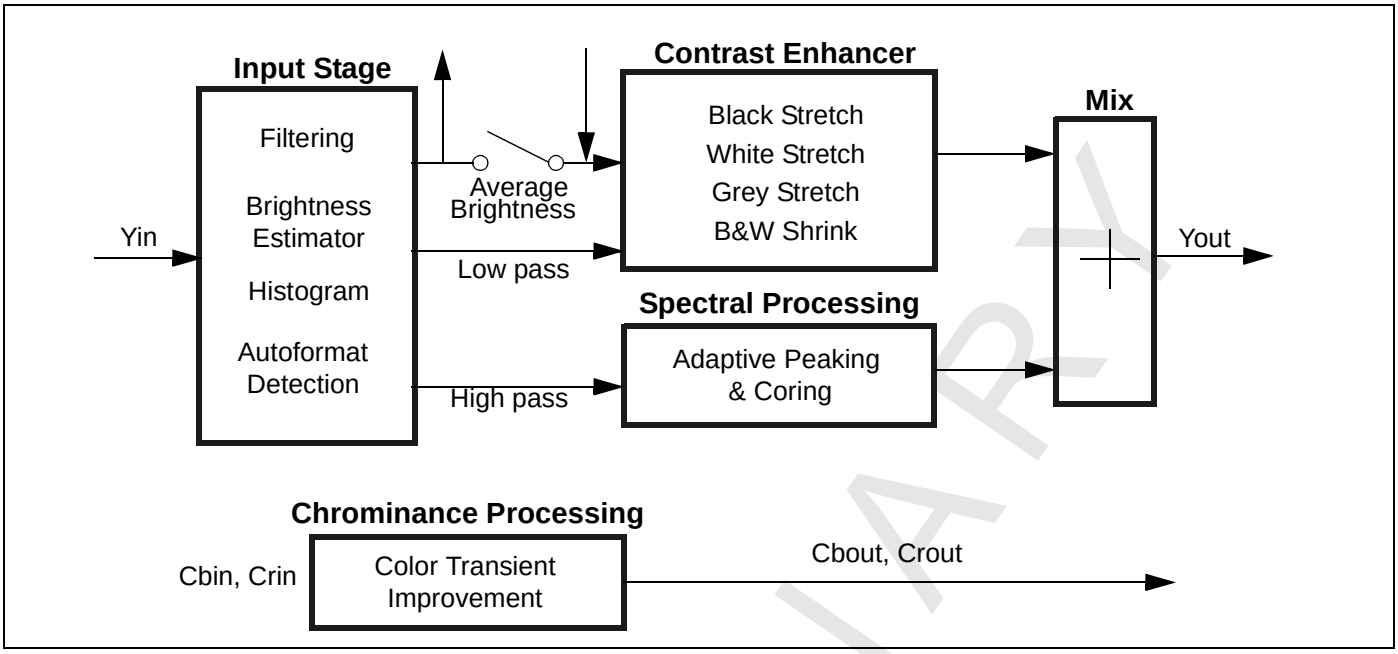


3.3.3 Image Improvement

3.3.3.1 Main Features

- Contrast Enhancement with 4 different modes: Black Stretch, White Stretch, Grey Stretch and Black & White Shrink
- Vertical and Horizontal peaking,
- Luma Coring
- Color Edge Replacement for Color Transition Improvement (CTI)
- Brightness Estimator
- Histogram

Figure 14: Image Improvement Block Diagram



3.3.4 Brightness Estimator

The Brightness Estimator provides the value of the average field brightness of the picture. It can be used to hardware tune the two brightness correction functions (spectral processing and contrast enhancer).

Otherwise, a manual mode enables the CPU to force the average brightness value of the image.

3.3.5 Histogram

A high-resolution, non-linear histogram that uses 16 levels to characterize the image luminance is included.

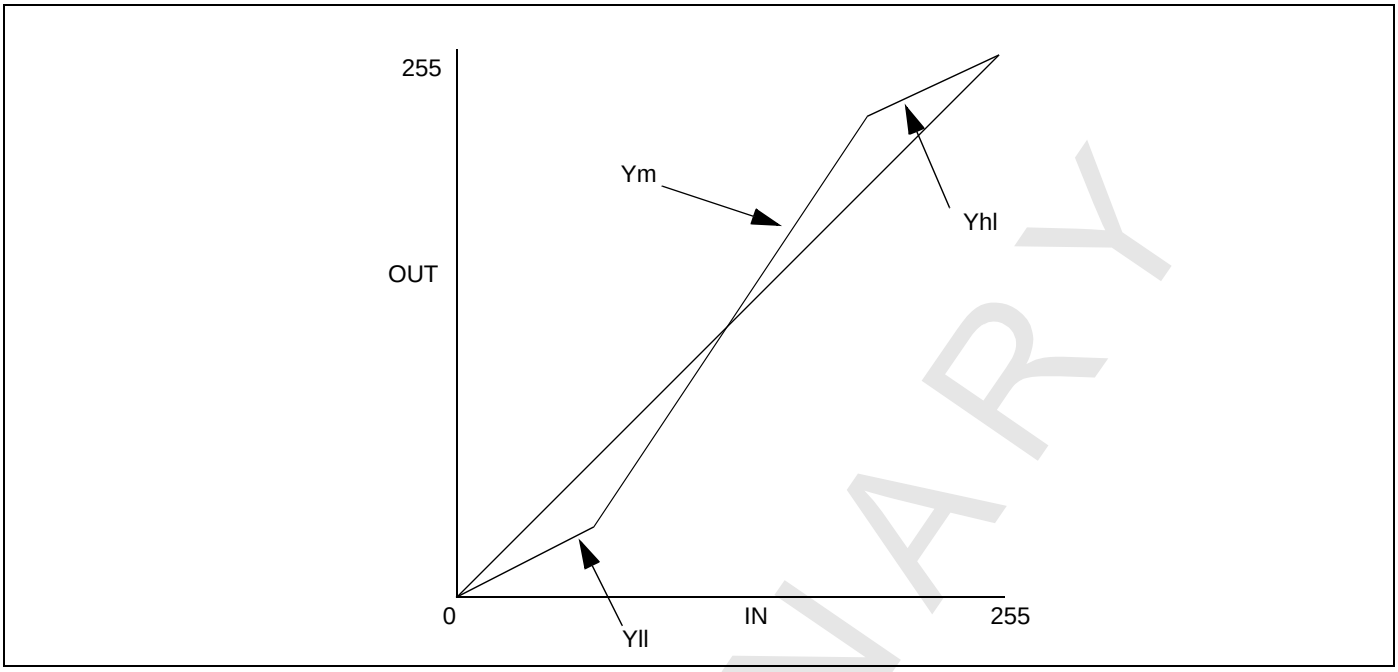
The accumulation of the histogram is done through an entire field and is ready to be used at the next V sync signal. A window for histogram calculation can be selected for reducing the image size which will be used for histogram processing.

3.3.6 Contrast Enhancer

The luminance processing block works only on the low pass spectrum of the input signal.

All these algorithms are based on the same model (Figure 15) that uses three different segments for contrast modification: the high limit (Yhl), the low limit (Yll) and the middle segment (Ym).

Figure 15: Yce = Ylp + Yg



- The Inflexion point is dependant either on the average brightness (when using brightness estimator values) or the input value (when using manual mode).
- The Range parameter modifies the slope of the Ym segment. Slope values are algorithm-dependant.

The Gain parameter modifies the overall curve gain in relation to the transfer curve (Yout = Yin).

Figure 16: Original Curve

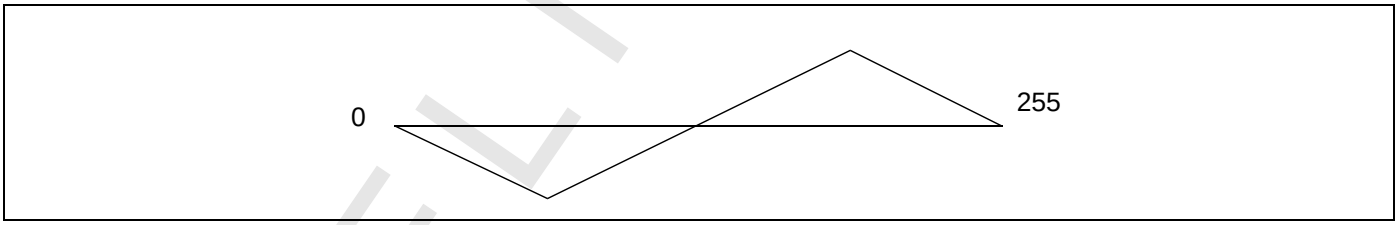


Figure 17: Example Curve after applying a Gain Factor of 2

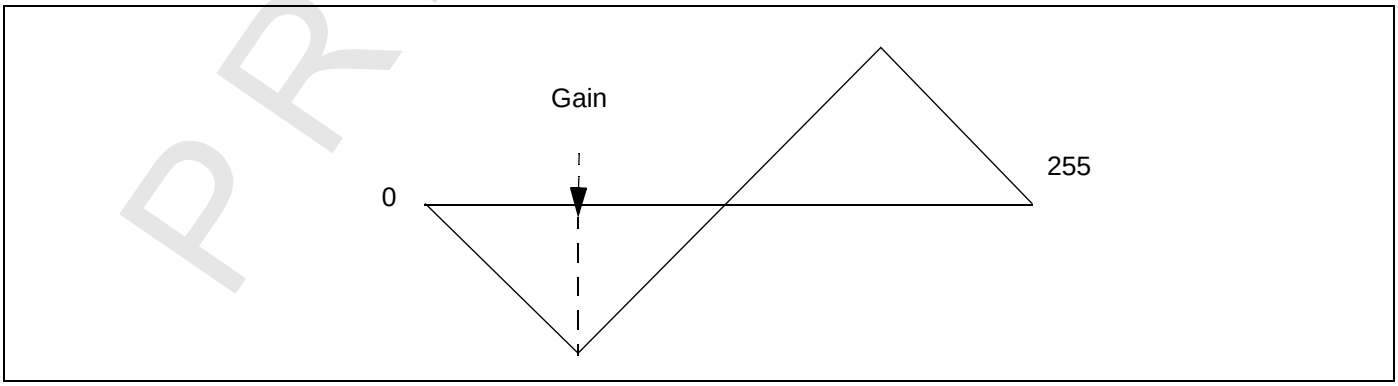


Table 9 shows the relation between the gain parameter and the gain factor values.

Table 9: Gain Factor

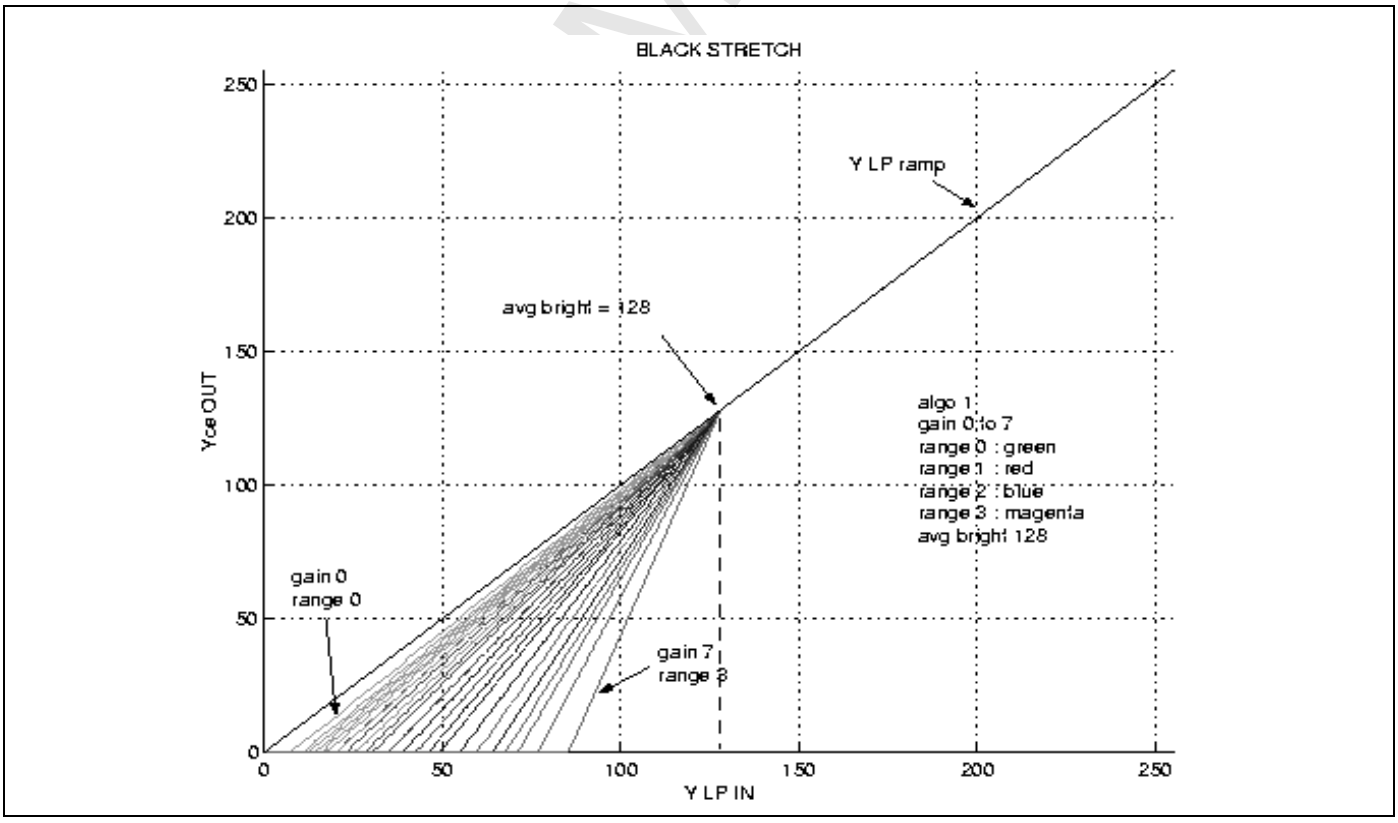
Gain Parameter	Gain Factor
0	0.500
1	0.750
2	0.825
3	1.000
4	1.125
5	1.500
6	1.750
7	2.000

3.3.6.1 Black Stretch

The Black Stretch algorithm is used to decrease the black part under the average brightness point. The inflection point is based on the average brightness, but another value can be set by user. The range and gain parameters are used to set the correction slope value.

This filter can be used when the overall picture is bright in order to improve the overall contrast. Figure 18 shows samples of the black stretch algorithm with an average brightness of 128 with various gain and range values.

Figure 18: Black Stretch



3.3.6.2 White Stretch

The White Stretch algorithm is used on a dark sequence in order to increase the contrast on the entire image. By default, the inflexion point is based on the average brightness, but another value can be set by user. Two modes are available by setting the range parameter:

1. When Range = 0, the inflection point is based on the value the average brightness.
2. When Range = 1, 2 or 3, the inflexion point IS NOT BASED on the value of the average brightness, but is used as parameter for an extended set of possibilities.

The following figures shows various examples of the white stretch algorithm with the following values: Gain = 7, Range = 0 and an Average Brightness of 64.

Figure 19: White Stretch 1

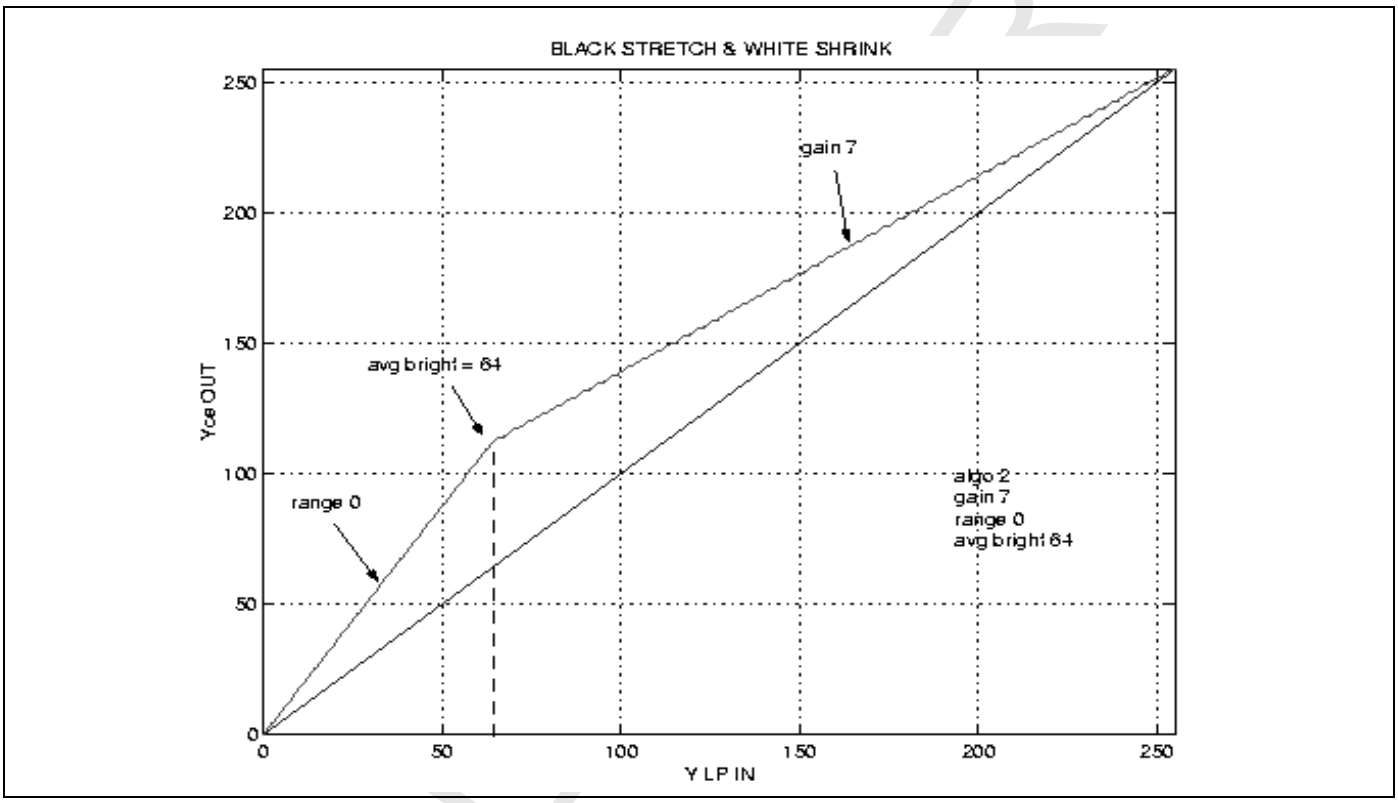


Figure 20: White Stretch 2

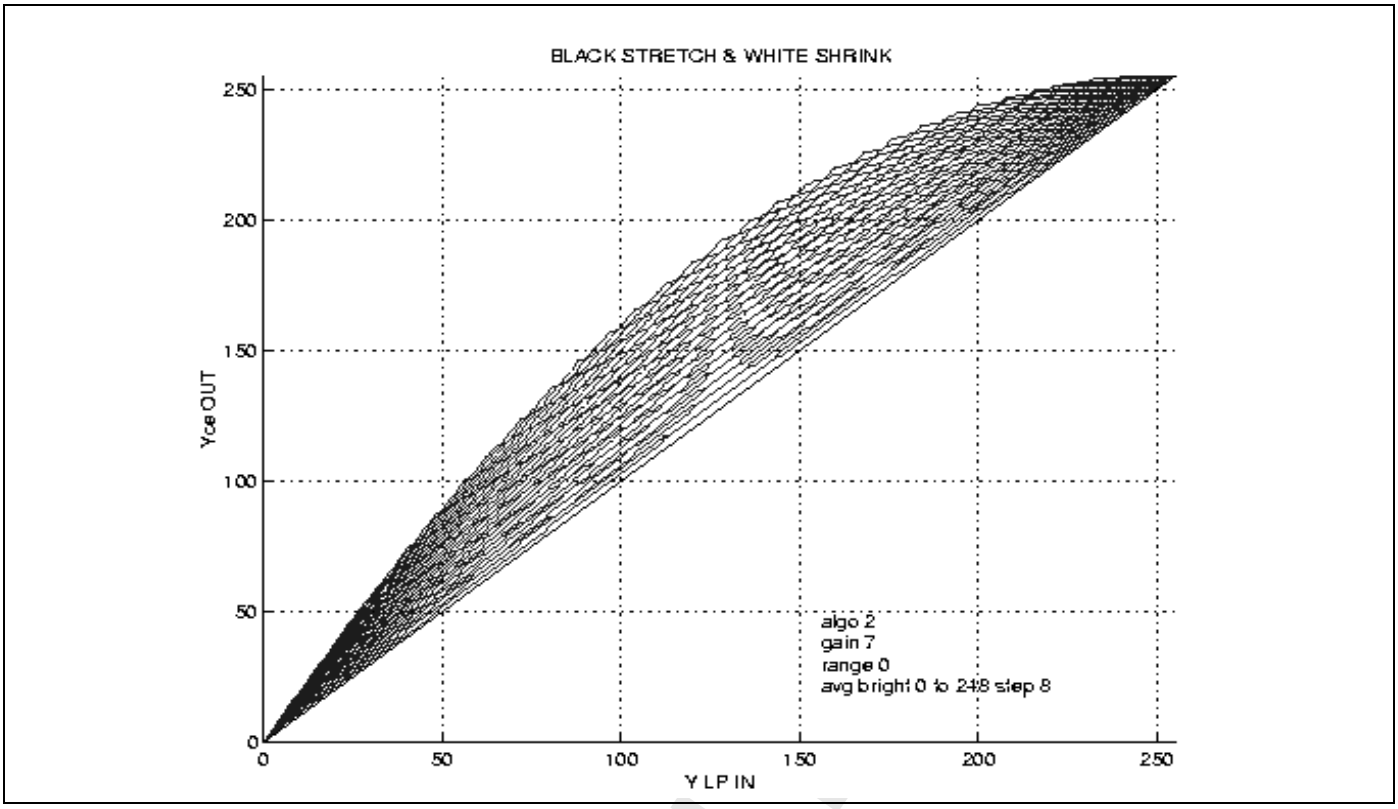
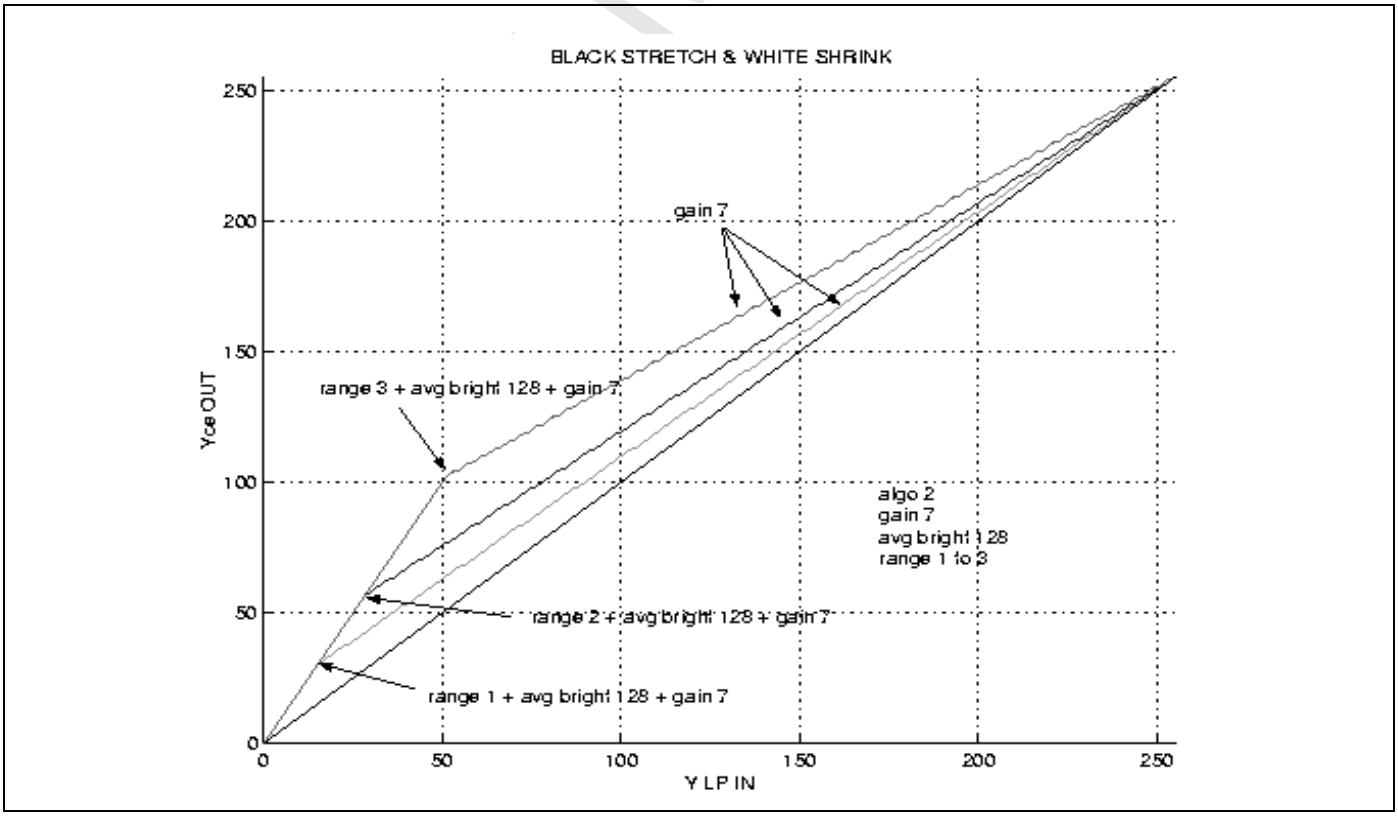


Figure 21 shows the special case of range values of 1, 2 and 3 in which the inflexion point is set by the user and is not based on the value of the average brightness.

Figure 21: White Stretch 3



3.3.6.3 Grey Stretch

In Grey Stretch mode, the Black and White Stretch values are combined to increase the contrast in both the black and white parts of the image. This function decreases the black level in the dark part of the image and increases the white level in the bright part. It can be used to increase the global contrast on low-contrast sequences that are centered on the average brightness. By default, the inflection point is based on the value of the average brightness, but another value can be set by the user. In this case, the range parameter is used to modify the slope of the central segment.

Table 10: Range and Slope Values

Range	Slope
0	0.25
1	0.50
2	0.75
3	1.00

The gain parameter magnifies the correction factor and modifies the global curves in relation to the standard linear curve.

Figure 22 shows the grey stretch algorithm with various range values.

Figure 22: Grey Stretch

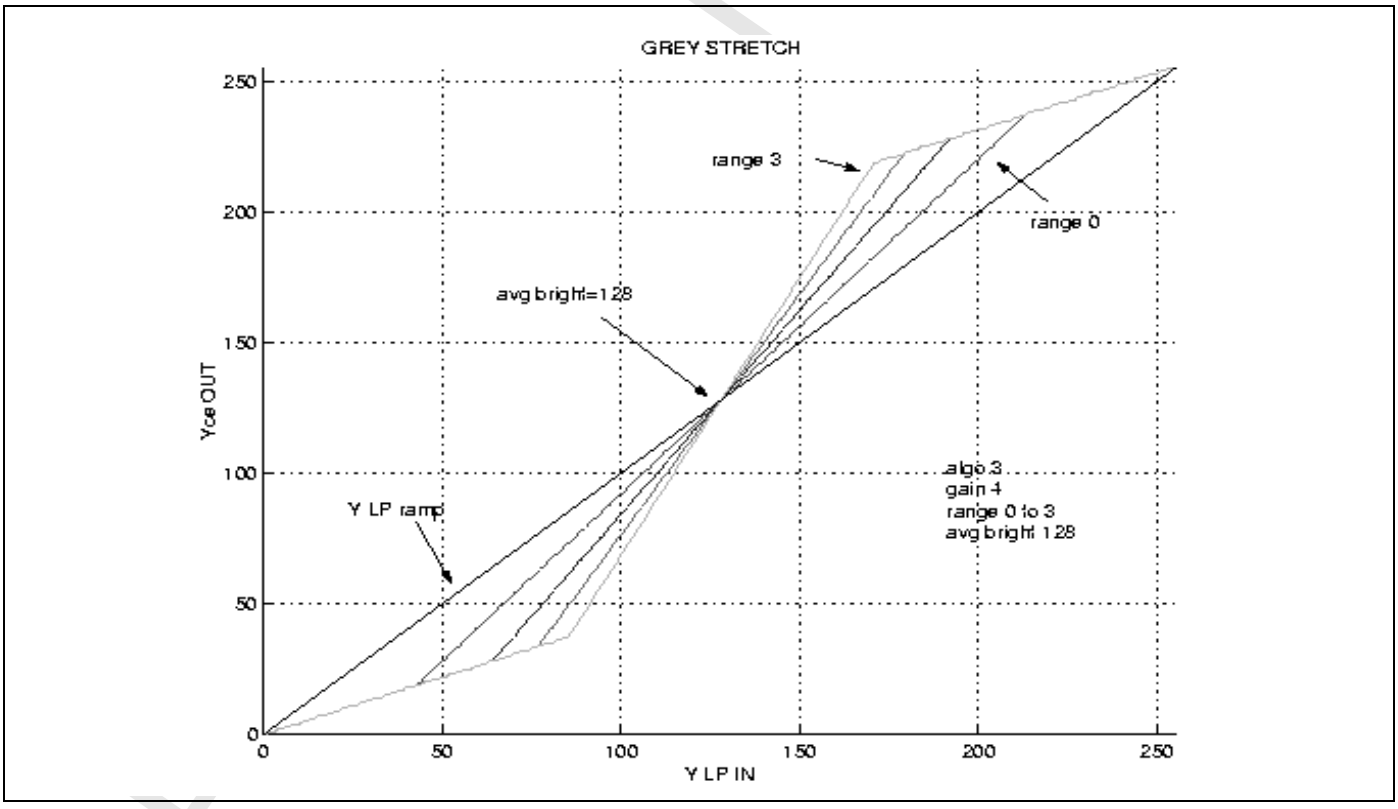
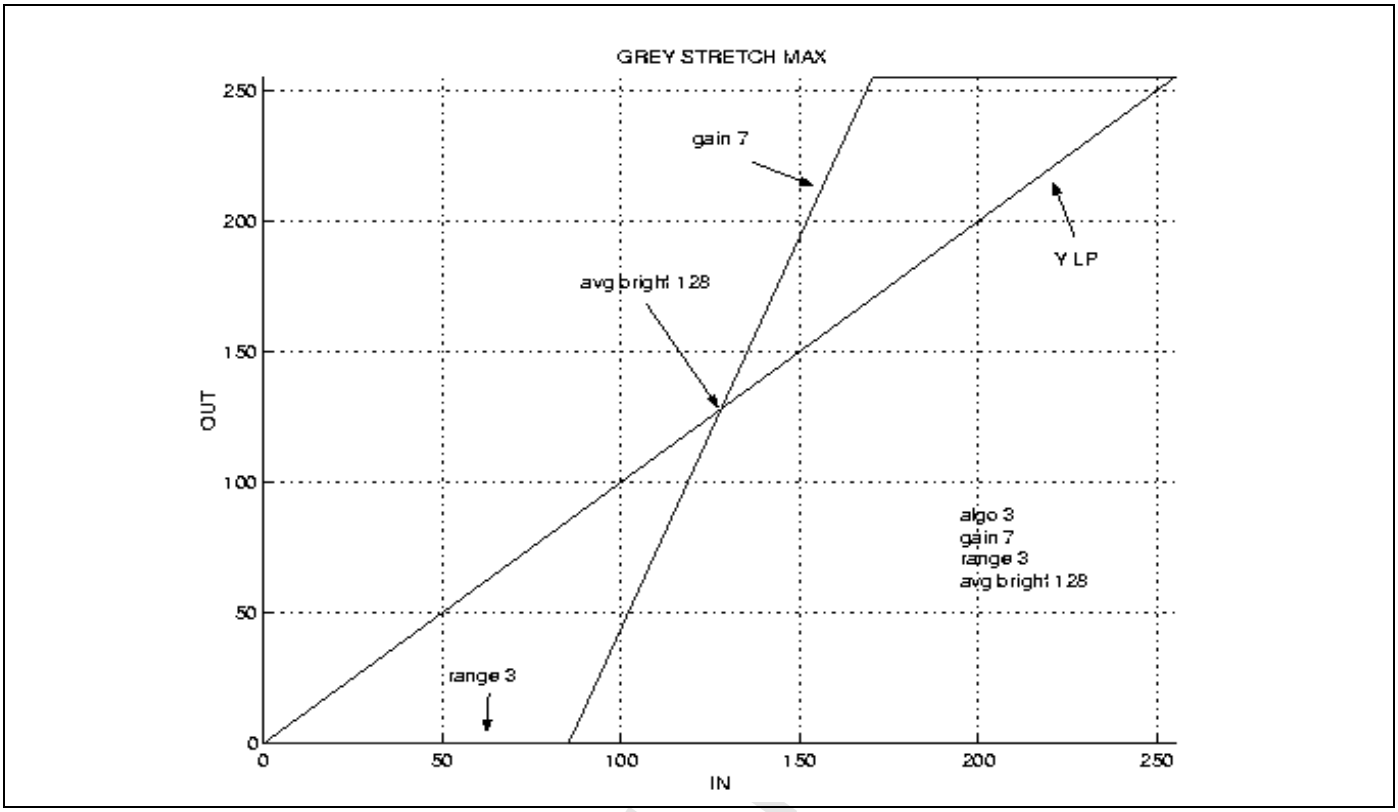


Figure 23 shows the grey stretch algorithm with the maximum gain value.

Figure 23: Grey Stretch Algorithm with Maximum Gain

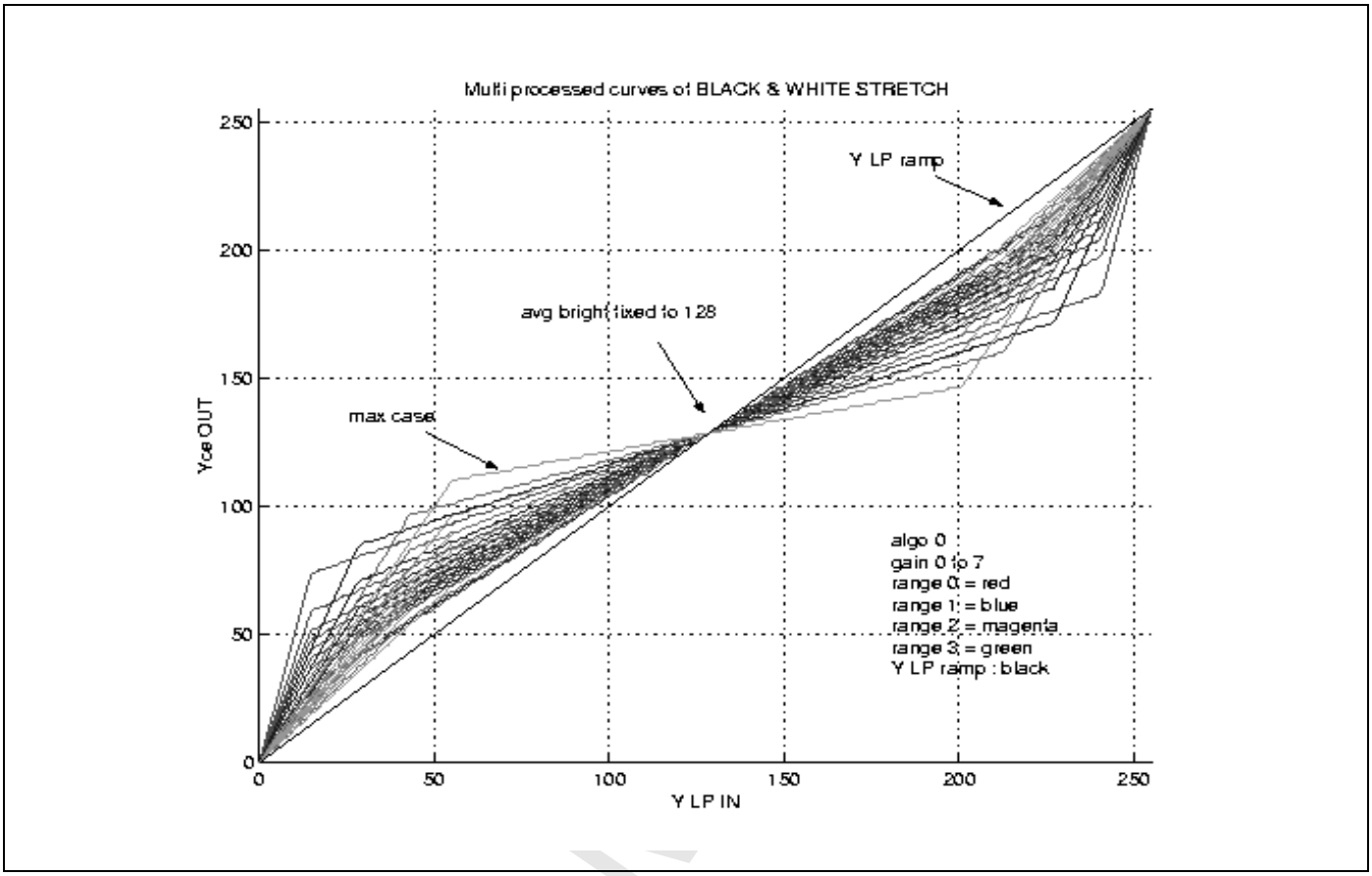


3.3.6.4 Black & White Shrink

The Black and White Shrink function increases the black level in the dark part of the picture and decreases the white level in the bright part while maintaining the average brightness of the image. The inflection point is set at a value of 128 and cannot be seen.

This function is used to reduce the global contrast on an over-contrasted sequence or to reveal details in the dark and bright areas. The Black and White Shrink function acts in the opposite manner of the Grey Stretch function.

Figure 24: Black and White Shrink Diagram



3.3.7 Spectral Processing

3.3.7.1 Adaptive Peaking

The luminance bandwidth must be improved when processing decoded composite signals and especially when a notch filter from the external digital video decoder is used. Luminance is improved in the STV3550 by adaptive peaking which increases the signal amplitude of a high-frequency spectrum.

This improves the outline of objects in a picture and increases the sharpness of the image.

Peaking effect is auto-adaptive and depends on image brightness.

In addition to the typical horizontal peaking, a vertical peaking mode can be selected in which vertical luminance components are used in the peaking algorithm. In this mode, vertical edges are also amplified. In this case, the vertical sharpness is much more sensitive to input noise and should be disabled.

3.3.7.2 Coring

If certain noisy sources are input (RF broadcast or VCR, for example), noise may be present in the signal. To prevent peaking due to noise amplification, certain coring thresholds must be used to suppress any increase of small-amplitude high-frequency signals (noise). The coring effect is programmable from 0 to 16 LSB.

As an example, the strength of the peaking is adjustable up to 64 steps from 0 to +10 dB. The frequency peaking is adjustable from 1.5 MHz to 3.5 MHz by software (1H domain frequencies).

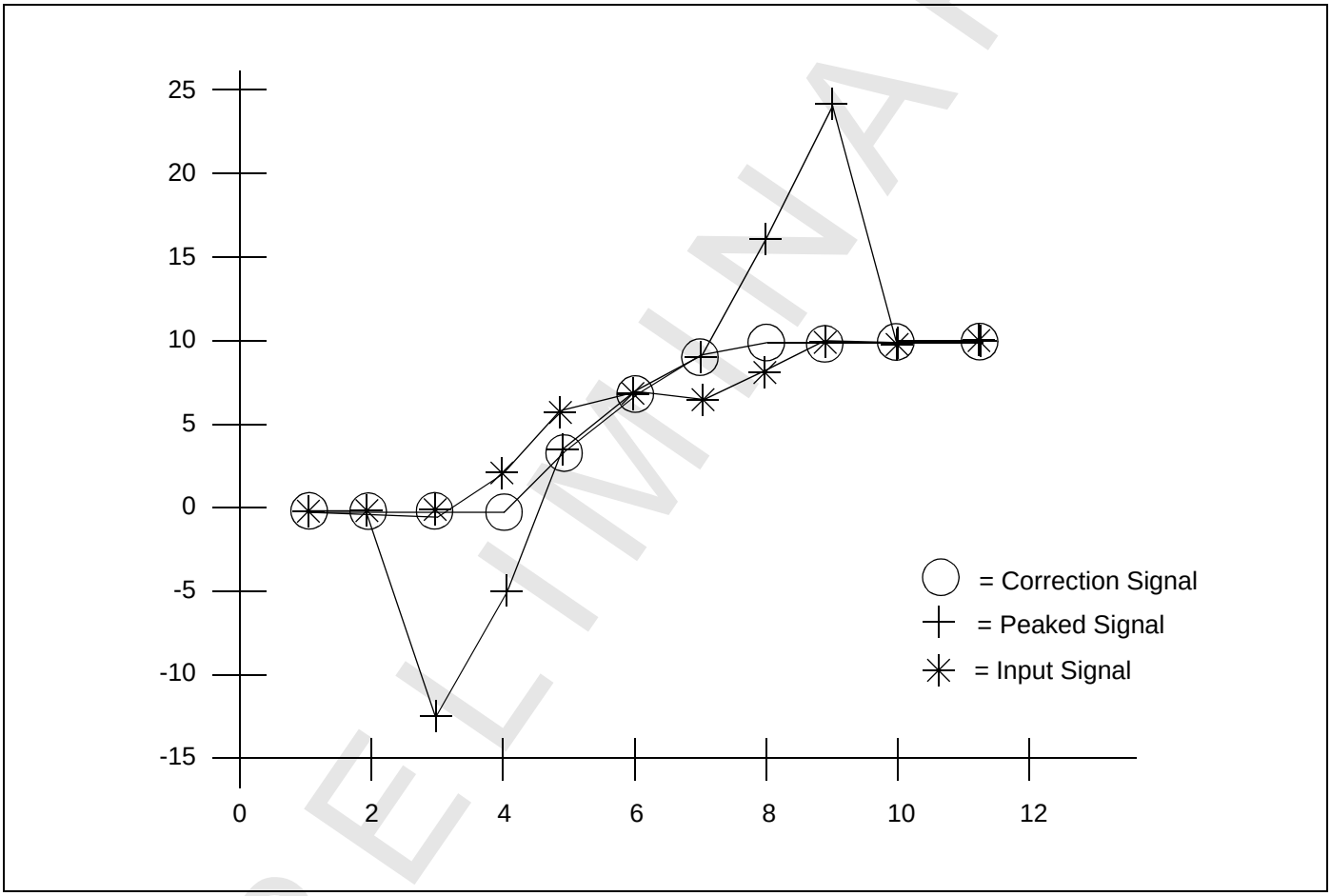
Coring is set using the coring level parameter which gives the strength of the noise suppression in 32 steps.

3.3.8 Color Transient Improvement

Color Transient Improvement (CTI) is used to improve color transitions and chroma resolution. It uses a custom filter to enhance chroma transitions. Overshoots and undershoots of correction are suppressed to eliminate incorrect colors at the output of the algorithm. Also small amplitudes are not amplified and are further reduced to suppress noise amplification.

Figure 25 shows an example of step enhancement and noise pass through.

Figure 25: Step Enhancement and Noise Pass Through



4 Graphics Functional Description

Note: Some descriptions in this section are not applicable for LCD applications.

4.1 On-Screen Display Generator (OSD)

4.1.1 General Information

The On-Screen Display (OSD) unit is used to overlay the video image with graphics or Teletext information generated by software. The OSD pipeline uses color look-up tables (CLUTs) also called palettes, with an 8-bit input. The CLUTs can then generate 256 predefined colors that are stored in the external SDRAM. The CLUT also includes 8 bits for Transparency Mode control. The output from the CLUT is in ARGB 32-bit/pixel format (8-A, 8-R, 8-G, 8-B). The OSD plane can consist of several display regions. For each region, a different CLUT can be defined.

Before being mixed with the video signal, the OSD output can be filtered by Anti-Flicker and Anti-Flutter filters. These filters are used to build an OSD plane in Progressive mode and store it in a field-based format so that it may be displayed in Interlaced mode. Other display modes can be used for a full-page graphic plane: Progressive mode or Non-Interlaced mode.

4.1.2 Main Features

- 8 bit per pixel Bitmap OSD with 10-bit RGB output resolution through a 32-bit CLUT
- Up to 1920 pixels per line and 1024 lines per field
- Capable of displaying Teletext 1.5 to 2.5, Teleweb and EPG applications
- 4 display planes (Video, Background, Graphics and Cursor) with mixing capabilities:
  - ➢ Global, pixel-based and proportional mixing
  - ➢ Programmable 8-bit color-based mixing factor for anti-aliasing effect on character edges
  - ➢ Programmable 8-bit region-based mixing factor for transparency effect
- Interlaced or Progressive Display Mode
- Anti-Flicker and Anti-Flutter filters
- 2-D Graphics Accelerator
- Programmable RGB gain and offset
- Field- or frame-based storage
- Embedded display special effects controlled by software
  - ➢ Flash, Underline, Italics, Fringe, Rounding and Shadow
  - ➢ Vertical Scrolling
  - ➢ Horizontal Scrolling
  - ➢ Rolling Scrolling
  - ➢ Opening Window

4.1.3 Functional Description

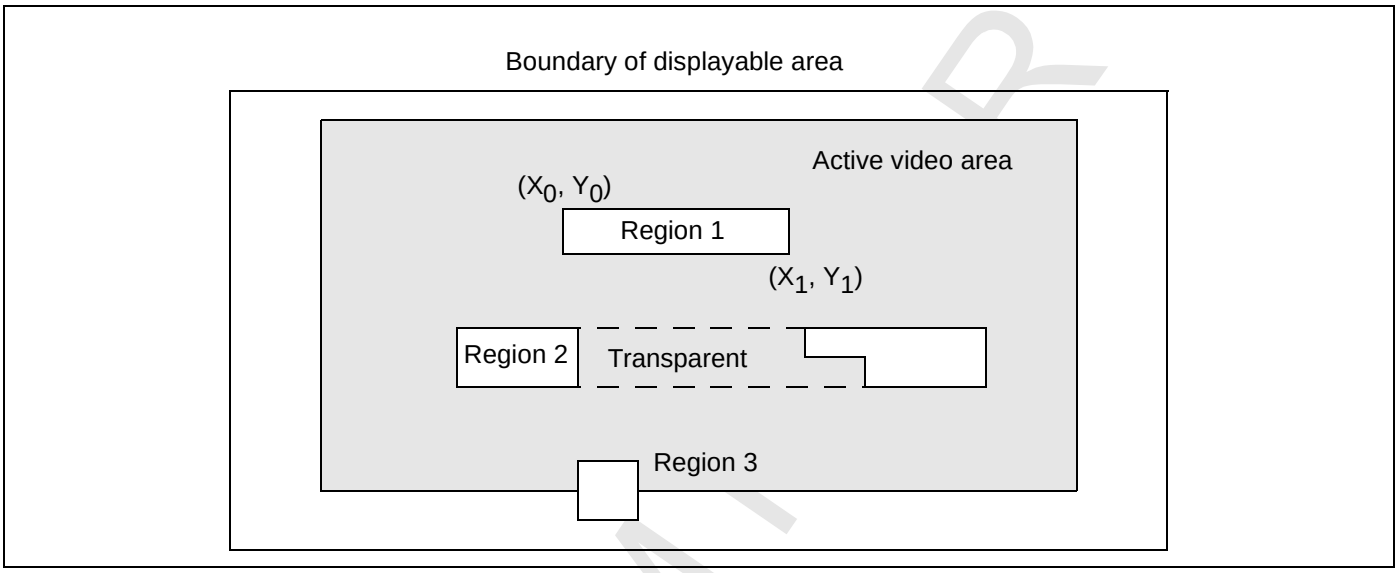
The OSD function displays a user-defined bitmap over any part of the displayable (i.e. non-blanked) screen, independent of the size and location of the active video area. This bitmap can be defined independently for each field. The OSD is enabled by setting the OSDON bit in the OSD_CONFIG register. The OSD bitmap is defined in relation to the display region and is independent of the decoded picture size and any pan/scan offset.

4.1.3.1 OSD Display Regions

The OSD consists of one or more display regions. Each display region is rectangular and can have its own color palette and other properties. Three examples of OSD regions are shown in Figure 26. In this figure, Region 3 shows that an OSD region can be displayed outside the active video area.

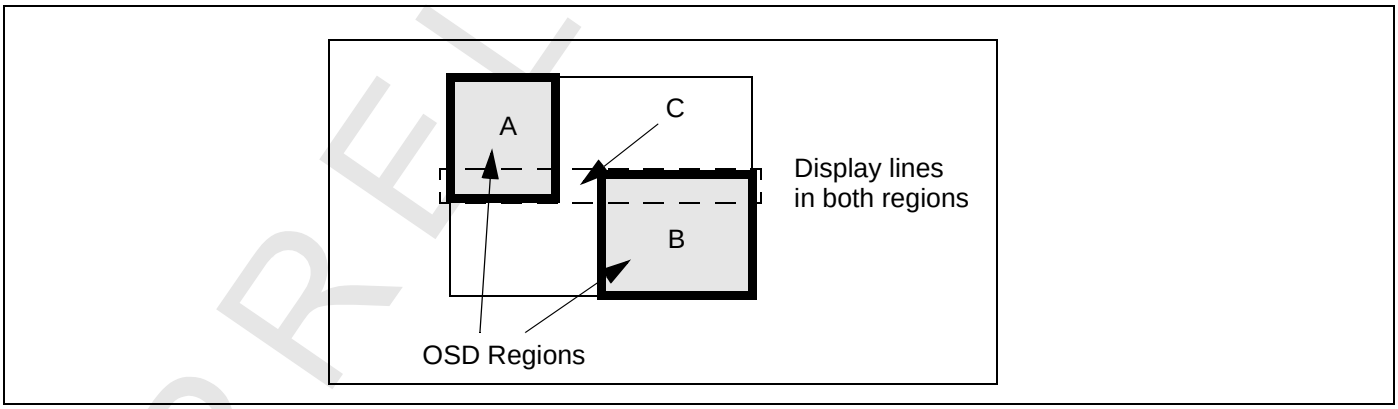
A single display line cannot be included in more than one active OSD region, i.e. only one OSD region can be active on a single line. If two OSD regions are required to be displayed on the same display line, a new OSD region must be defined that includes both regions.

Figure 26: OSD Regions



For example, in Figure 27, two different OSD regions (A and B) use some of the same display lines. If both OSD regions are to be active at the same time, a new region (C) must be defined including both regions A and B. The area of region C that is outside regions A and B can be defined as transparent.

Figure 27: Two Display Regions using the same Display Lines



4.1.4 Programming OSD Display Regions

The characteristics of each OSD region are contained in an OSD specification which is stored in the 64-Mbit SDRAM. The following data is contained in the OSD specification:

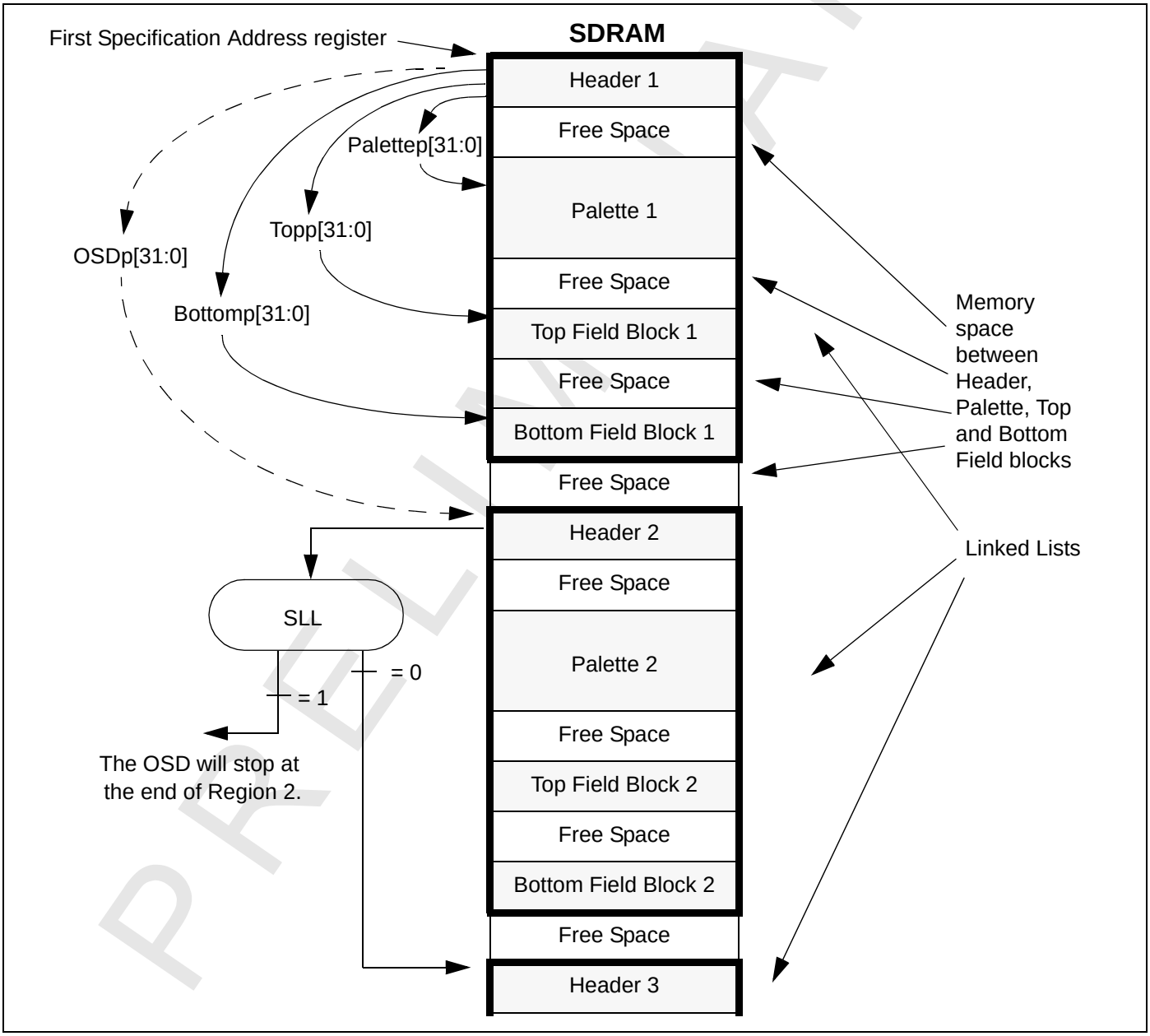
- A **header** which defines the boundaries of the region and contains 4 pointers that are used to identify the SDRAM location of the current palette, the current top block, the current bottom block, the next OSD region and other control information.

- A **color palette** which defines the colors in the SDRAM. Up to 256 colors can be used by the bitmap. If required, one of these colors can be “transparent” allowing the background to show through. These colors are loaded in the OSD CLUT.
- A **bitmap** composed of 2 blocks of 32-bit words.

CAUTION: When 2 regions are displayed one after the other, there must be at least one TV scan line between them with nothing displayed. This enables the OSD to reload the 256 colors in the CLUT.

OSD specifications are stored as linked lists, as shown in Figure 28 and Figure 29. The order that the specifications are stored in SDRAM is the order that the regions are displayed on the screen. This self-chained mode is managed by software by programming the address of the next OSD region to be displayed in the Next OSD Specification pointer (OSDp[31:0]). The Stop Linked List (SLL) bit indicates the last specification in the list.

Figure 28: OSD Specification



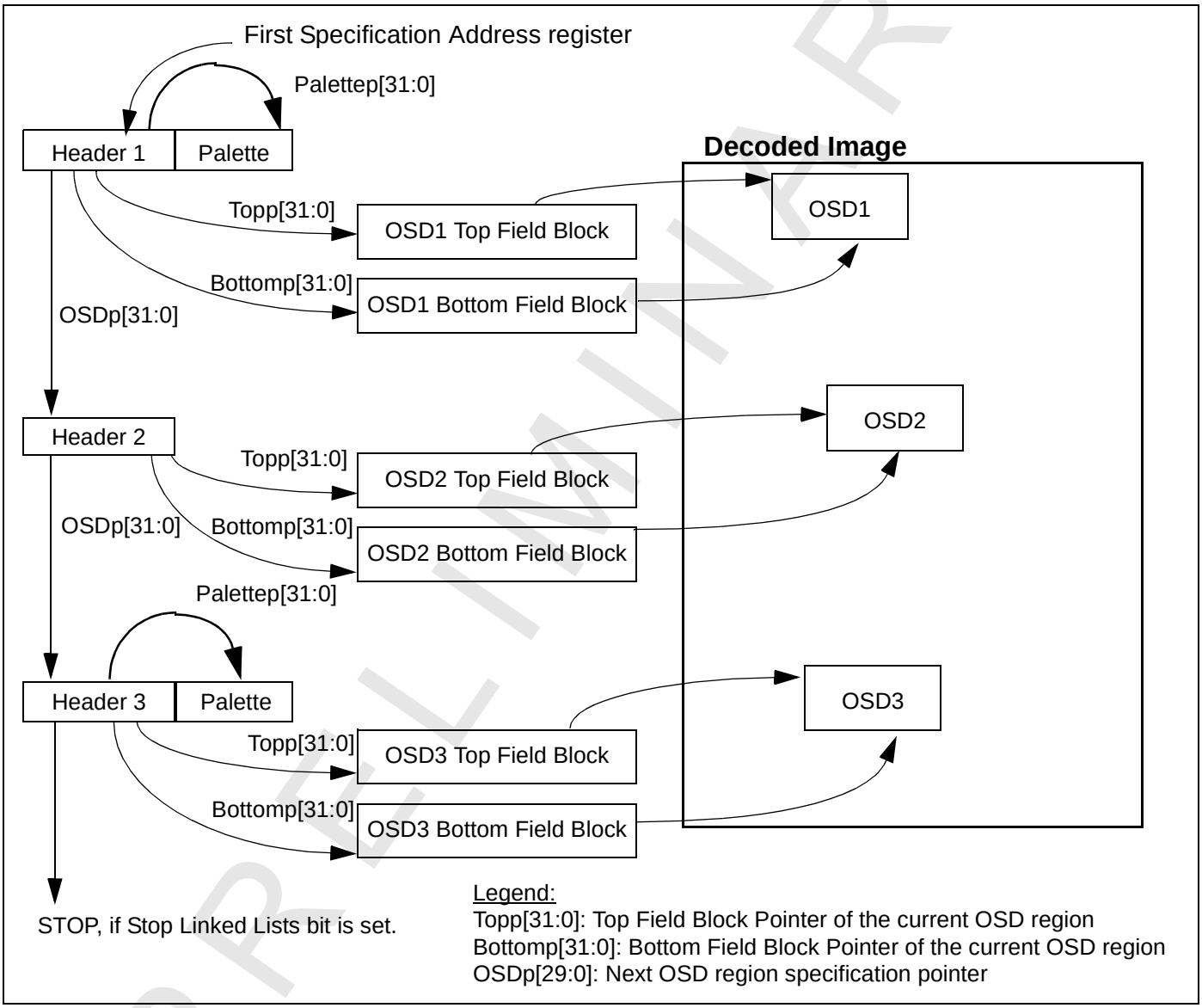
An OSD specification can be placed anywhere within the entire 64-Mbit SDRAM. Each OSD region displayed on the screen is the object of an OSD specification. The position of the first OSD region

(or OSD specification) stored in the SDRAM is defined in bits FSA[31:0] in the First Specification Address register.

Each block (Header, Palette and Bitmap Data) defines one field of one OSD region and must be contiguous. Each block and its first bitmap pixel address must be aligned on a 32-bit boundary in the SDRAM.

The line numbers used to define the top and bottom lines of an OSD region are the internal (field) line numbers illustrated in Figure 29. The same OSD specification may be used for both fields in a single frame.

Figure 29: Linked List Structure for OSD Data

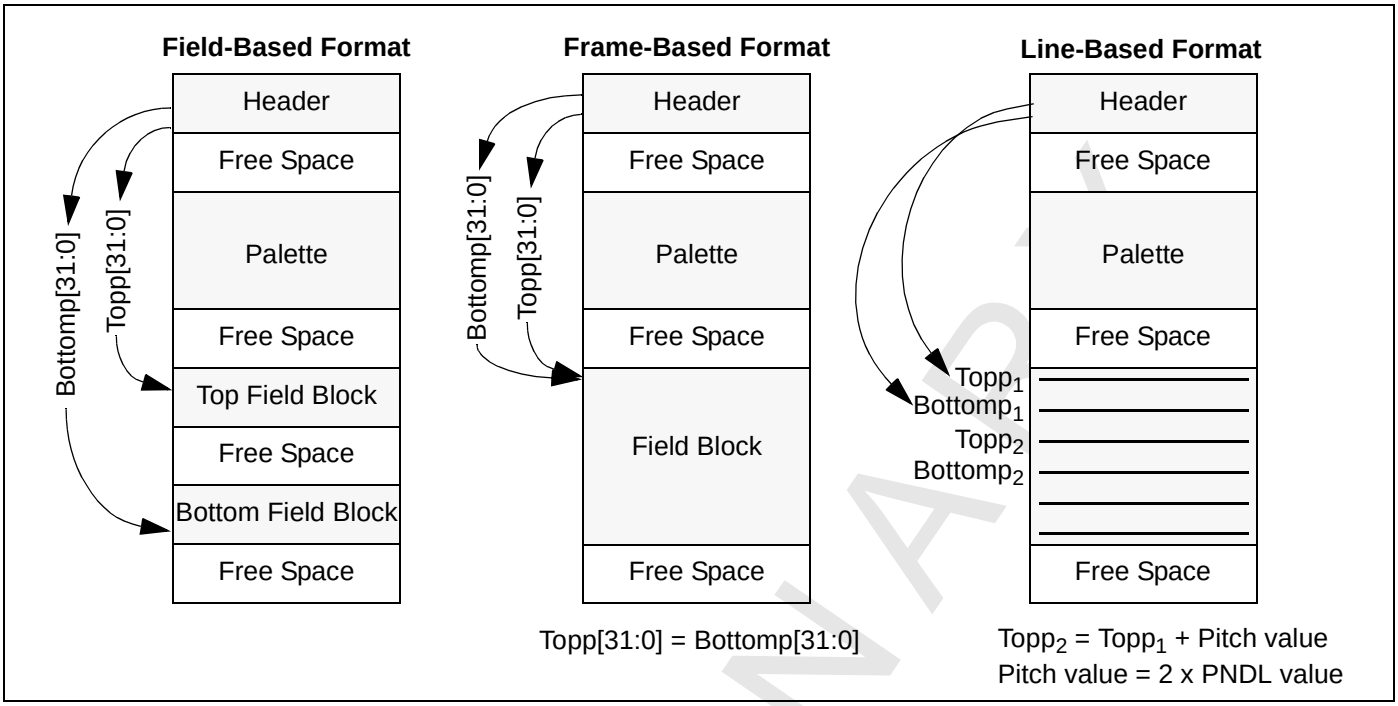


Field Storage Formats

OSD planes can be stored in a field-based format with an OSD top field and an OSD bottom field, or using a frame-based format.

If the bitmap is stored in frame-based format, it can be read as top and bottom fields using the Pitch function. Using pointers Topp[31:0] and Bottomp[31:0], the top field block and the bottom field block can be the same (e.g. it can be used for Simple Character mode). In this case, the data in the Top Field is the same as in the Bottom Field.

Figure 30: Field Storage Formats



4.1.4.1 Header Format

The header is a block of nine 32-bit words dedicated to a single specification (i.e. one OSD region on the screen). Figure 31 shows the header mapping, with each line representing a 32-bit word.

Figure 31: OSD Header Mapping

Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Reserved												PNDL[11:0]										Gain[5:0]						GaEn	MiMo [1:0]		FM [1:0]		SLL	NP
2	Reserved												Pitch[12:0]												NCL[7:0]								CLUTAdd[7:0]							
3	Offset[9:0]												TPND[21:0]																			
4	Rd		MiW[7:0]										Y_bottom[10:0]												X_left[11:0]							
5	MiGa[8:0]												Y_top[10:0]												X_right[11:0]							
6	Bottomp[31:0]																															
7	Topp[31:0]																															
8	Palettep[31:0]																															
9	OSDp[31:0]																															

Address Pointers

The address of the first byte (in the SDRAM) of the first color which will be loaded from the **palette** to the OSD CLUT is defined in bits Palettep[31:0]. If a **new palette** (CLUT programmed in the SDRAM) must be loaded when the OSD displays a new OSD region, the New Palette (NP) bit must be set.

Note: The Palettep[31:0] value is not the address of the first byte of the first color in the palette. It is the address of the first byte of the first color loaded in the OSD CLUT.

The address of the first byte (first pixel) in the **Top Field** block which will be displayed in the OSD region is defined in bits Topp[31:0].

The address of the first byte (first pixel) in the **Bottom Field** block which will be displayed in the OSD region is defined in bits Bottomp[31:0].

Note: Using pointers Topp[31:0] and Bottomp[31:0], the top field block and the bottom field block can be the same (e.g. it can be used for Simple Character mode).

The address of the **next OSD specification** in the SDRAM is defined in bits OSDp[31:0]. If the current OSD specification is the last one displayed on the TV screen, the **Stop Linked List** (SLL) bit must be set.

OSD Region Positions

The **top position** of the current OSD region is specified in bits Y_top[10:0] which define the vertical start of the OSD region on the screen. This value is given as a number of TV lines. The top line specified in the first word of an OSD region specification must be greater than or equal to 3.

The **bottom position** of the current OSD region is specified in bits Y_bottom[10:0] which define the vertical end of the OSD region on the screen. This value is given as a number of TV lines.

The **left position** of the current OSD region is specified in bits X_left[11:0] which define the horizontal start of the OSD region on the screen. This value is given as a number of pixel clock periods.

The **right position** of the current OSD region is specified in bits X_right[11:0] which define the horizontal end of the OSD region on the screen. This value is given as a number of pixel clock periods.

*Note: An OSD region must be at the same position (same TV line) in both Interlaced and Progressive mode. In Progressive or Non-Interlaced mode, the Y_top and Y_bottom positions are programmed the same way. This means that the number of TV lines displayed for an OSD region in Progressive mode is never an odd number. An OSD region will always have (Y_bottom -Y_top)*2 TV lines displayed.*

In Interlaced mode, the same number of TV lines is displayed in both fields. Value "Y_bottom - Y_top" is field-based.

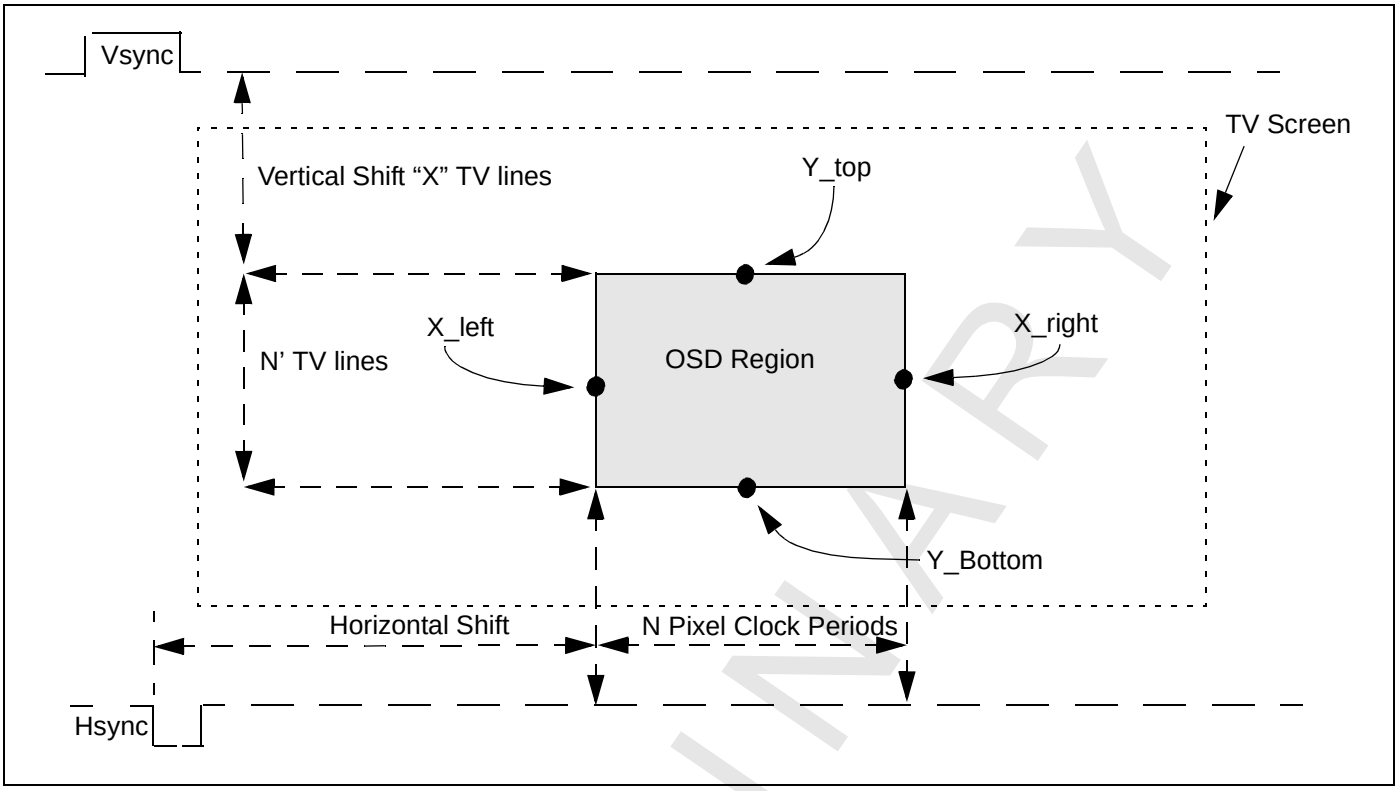
OSD Region Characteristics

The **total number of pixels to be displayed** in the top or bottom field for the current OSD region is given in bits TPND[21:0]. The number of pixels in the Top Field block is the same as in the Bottom Field block. This is not the number of pixels programmed in the Top and Bottom Field blocks.

The **pitch** value which will be added to the Topp[31:0] or Bottomp[31:0] pointers in order to load the next pixels is defined in bits Pitch[12:0]. This value represents the maximum number of pixels which will be displayed in a TV line.

The **number of pixels displayed in a single TV scan line** is given in bits PNDL[11:0]. This value is the same for all TV scan lines in a given OSD region.

Figure 32: OSD Region Positions



Color Characteristics

The address in the **OSD Color Look-Up-Table** (CLUT) (not in the SDRAM), where the color defined in the palette of the current specification and pointed to by Palettep[31:0] will be loaded, is defined in bits CLUTAdd[7:0]. If the New Palette bit is reset, these bits are ignored.

The **number of colors** which will be loaded in the OSD CLUT is defined in bits NCL[7:0]. The OSD CLUT can contain up to 256 colors. For more information, refer to Section 4.1.4.2: Color Data on page 46.

Note: When NCL[7:0] = 00h, one color will be loaded.

Each region may have its own palette, with its own number of colors (these colors will be loaded in the OSD CLUT before displaying the bitmap). If a sequence of regions uses the same palette, the palette need only be defined in the first region of the sequence. If a region only uses a part of the colors defined in the palette of the previous region, the unused colors from the previous palette can be reprogrammed for the current region in the OSD CLUT.

Mixing Characteristics

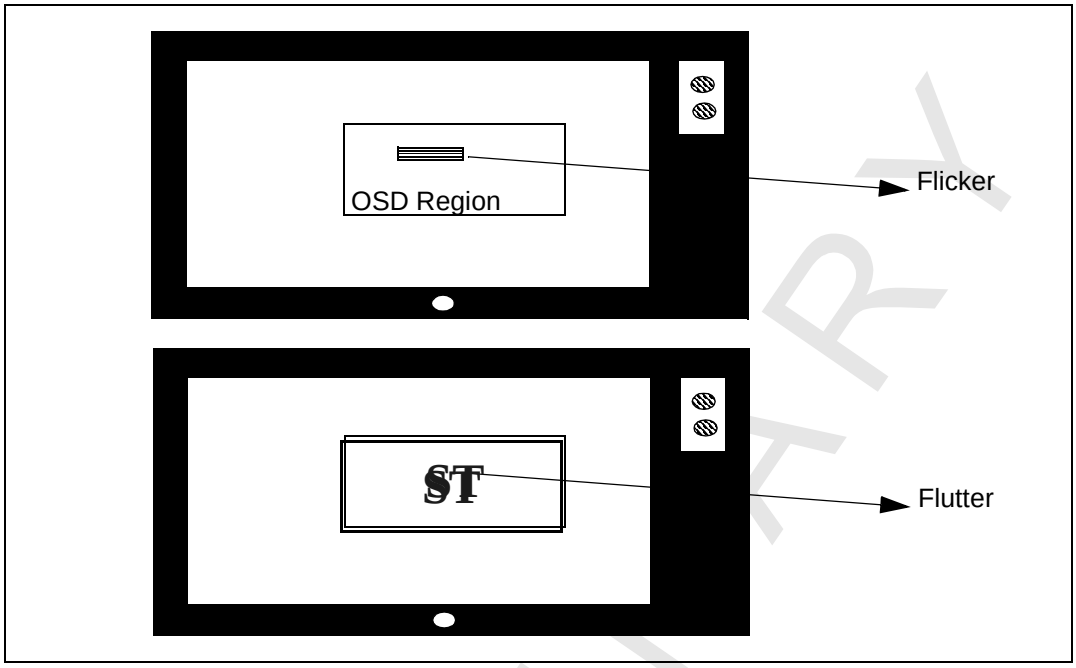
Four different modes may be applied to RGB components for mixing OSD and video signals: **Pixel Mixing mode**, **Global Mixing mode** (mixing factor defined in **Mix-Weight bits**), **Proportional Mixing mode** (mixing factor defined in **Mix-Gain bits**) and **No Mixing mode** (full OSD). The mixing mode is specified in bits MiMo[1:0]. For more information, refer to Section 4.1.5: Mixing OSD and Video Signals on page 48

Filter Mode

Flicker problems may occur inside an OSD when there is a significant difference in the color between pixels in a top line and those of its surrounding bottom lines (e.g. if one object is visible in one line and not in its surrounding lines).

In this case, the OSD consists of 2 different images which are displayed one after the other at a rate of 50 images per second. The human eye is able to detect the appearance and disappearance of both these images in the OSD. This effect is known as “flicker”.

Figure 33: Filter Modes



Flutter problems occur when the same image is used in both the top and bottom fields of a single frame. The top field is displayed in the same position as the bottom field.

In this case, the OSD consists of top field lines and bottom field lines which are displayed one after the other at a rate of 50 images per second. The OSD image seems to move up and down. This effect is called “flutter”.

Anti-Flicker and Anti-Flutter filter modes may be applied to Alpha and RGB color values stored in the CLUT. The filter mode is selected by bits FM[1:0].

Note: The Anti-Flicker and Anti-Flutter filters are exclusive and cannot be used simultaneously.

Anti-Aliasing Mode

Anti-Aliasing mode is used to smooth and remove the effects caused by insufficient sampling or poor filtering of the video signal. To enable Anti-Aliasing mode, reset the BDWOFF bit in the OSD_CONFIG register. The Anti-Aliasing value is stored in the BDW[7:0] bits in the OSD_CONFIG register. This value is only applied to the first and last lines of the OSD region. The same value is applied to all OSD regions displayed in the entire frame.

Gain and Offset Operations

A **gain value** may be applied to RGB outputs after the filter operations for the current OSD region by setting bit GaEn. This value is always associated with an offset value. 64 gain values are available by setting bits Gain[5:0]. The gain is never applied to Alpha or 1-Alpha values. The **offset** added to the RGB outputs after the gain operation for the current OSD region is defined in bits Offset[9:0].

R[9:0] = Gain[5:0]/32 x R[9:0] + Offset[9:0]

G[9:0] = Gain[5:0]/32 x G[9:0] + Offset[9:0]

B[9:0] = Gain[5:0]/32 x B[9:0] + Offset[9:0]

This can be used to independently control the contrast of the OSD plane in relation to the Video display plane.

Table 11: OSD Header Characteristics

Bitfield	Bits	Description
PNDL[11:0]	12	The Number of Pixels displayed per TV line. 000h: FFFh:
Gain[5:0]	6	Gain Factor applied to RGB outputs. 00h: 0/32 Gain factor 3Fh: 63/32 Gain factor
GaEn	1	Gain Enable 0: Gain and offset operations are disabled. 1: Gain and offset operations are enabled.
MiMo[1:0]	2	Mixing Mode 00: Pixel Mixing mode 01: Global Mixing mode 10: Proportional Mixing mode 11: No Mixing mode (full OSD)
FM[1:0]	2	Filter Mode 00: No Filter mode 01: Anti-Flicker mode 10: Anti-Flutter mode 11: Reserved
SLL	1	Stop Linked List 0: The next OSD specification will be displayed. 1: The current OSD specification is the last one displayed on the TV screen. After this specification is displayed, the OSD will switch off.
NP	1	New Palette 0: The CLUT from the previous OSD region will be used. 1: A new palette must be loaded for the current OSD region.
Offset[9:0]	10	Offset value added to RGB outputs. 000h: 3FFh:
CLUTAdd[7:0]	8	Address in the OSD CLUT where the first color (pointed by Palettep[31:0]) will be (re) loaded. 00h: FFh:
NCL[7:0]	8	Number of Colors Loaded in the OSD CLUT 00h: FFh:
MiW[7:0]	8	Mixing factor applied to RGB outputs when Global mixing mode is selected. Values greater than 80h are clamped to 80h. 00h: 80h:
MiGa[8:0]	9	Mixing factor which multiply or divide the Alpha parameter when Proportional mixing mode is selected. 000h: 1FFh:
TPND[21:0]	22	Total Number of Pixels Displayed. 000000h: 3FFFFFh:

Table 11: OSD Header Characteristics

Bitfield	Bits	Description
Pitch[12:0]	13	Pitch Value (Max. number of pixels which will be displayed on a TV line) 0000h: 1FFFh:
Y_top[10:0]	11	Top Position of the OSD Region (in number of TV scan lines). 000h: 7FFh:
Y_bottom[10:0]	11	Bottom Position of the OSD Region (in number of TV scan lines). 000h: 7FFh:
X_left[11:0]	12	Left Position of the OSD Region (in number of pixel clock periods). 000h: FFFh:
X_right[11:0]	12	Right Position of the OSD Region (in number of pixel clock periods). 000h: FFFh:
Palettep[31:0]	32	Palette Pointer 00000000h: FFFFFFFFh:
Topp[31:0]	32	Top Field Block Pointer 00000000h: FFFFFFFFh:
Bottomp[31:0]	32	Bottom Field Block Pointer 00000000h: FFFFFFFFh:
OSDp[31:0]	32	Next OSD Region Specification Pointer. 00000000h: FFFFFFFFh:

4.1.4.2 Color Data

Each color in the CLUT is defined using the RGB color elements and a mixing factor which determines the overlaying effect of the OSD.

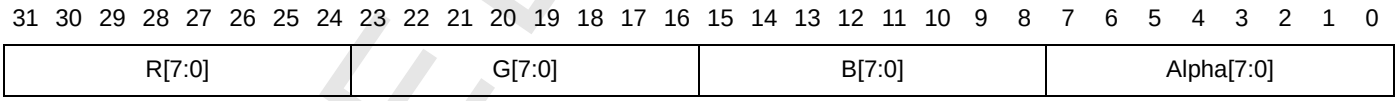


Table 12: : Palette Line Format in 32-bit Color

Bitfield	Bits	Description
R[7:0]	8	Red Color value 00h: FFh:
G[7:0]	8	Green Color value 00h: FFh:
B[7:0]	8	Blue Color value 00h: FFh:

Table 12: : Palette Line Format in 32-bit Color

Bitfield	Bits	Description
Alpha[7:0]	8	Alpha Pixel Mixing factor Values greater than 80h are clamped to 80h. 00h: 80h:

Note: Due to the GAMMA specification, the Alpha[7:0] range is from 0 to 128 in decimal (00h is transparent, 80h is full OSD). All values greater than 80h (from 81h to FFh) are clamped to 80h.

The user may define a color palette with “n” number of colors (“n” words of 32 bits) for a single OSD region. Before this region is displayed, these colors are loaded into a Color Look Up Table (CLUT) in SDRAM where the RGB and Alpha bits are written.

The user defines the number of colors that will be loaded into the CLUT (NCL[7:0]) and the address in the CLUT where the first color will be loaded (CLUTAdd[7:0]). These two parameters may be used to reprogram or modify certain colors in the CLUT when several regions are displayed simultaneously.

Note: Palettep[31:0] *points to the address in SDRAM of the first byte of the first color loaded.*

CLUTAdd[7:0] is only used to load the colors in the CLUT and not to read the CLUT during display.

The number of colors in the palette (programmed in the SDRAM) can be greater or less than 256 (a color is a word of 32 bits). But, a only maximum of 256 colors will be loaded in the OSD CLUT.

The NCL[7:0] parameter refers to a contiguous number of colors programmed in the palette. If NCL = 0, one color will be loaded.

Each OSD specification after the first one can either use the same color palette as the previous OSD region, or a new one can be defined. If a new palette is defined in the SDRAM, the following bits must be modified in the header:

- NP must be set.
- Palettep[31:0] will indicate the new address of the first byte to be loaded.
- NCL[7:0] gives the number of colors to be loaded in the OSD CLUT.
- CLUTAdd[7:0] gives the OSD CLUT address where the first color will be loaded.

CLUT outputs are 32-bits; 8-bits for R, 8-bits for G, 8-bits for B, plus 8 bits for transparency control (or mixing factor).

4.1.4.3 OSD Bitmap

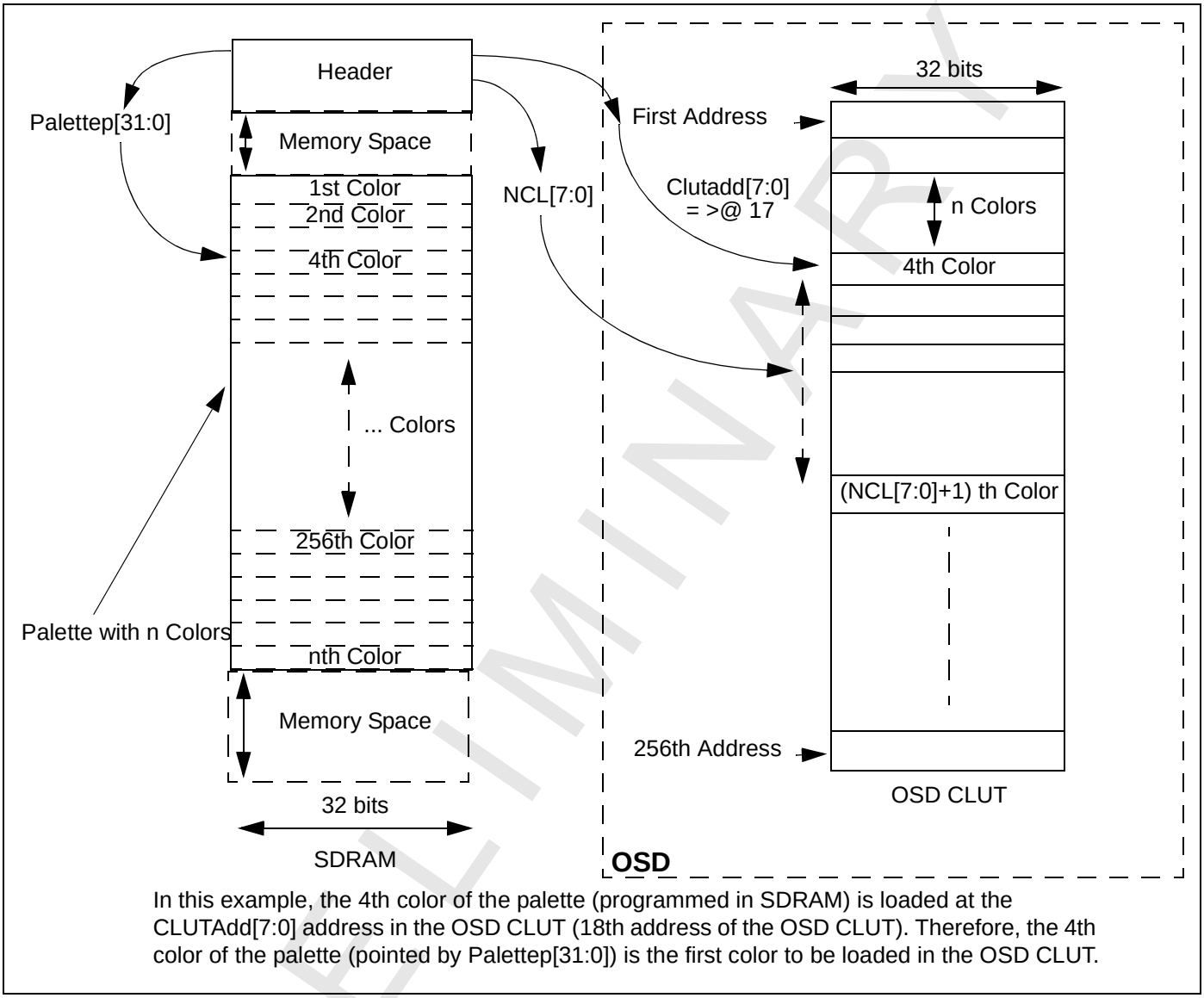
The bitmap defines the OSD pixels, from left to right, in order and within lines, and follows the lines from top to bottom. The first line of a bitmap is always a top line, and the last line is always a bottom line. The number of bits per pixel is 8. The value for each pixel gives the line of the palette or the address in the CLUT, which defines the color for the pixel.

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

The bitmap block is subdivided in two field blocks, the top field block and the bottom field block. These two blocks always contain the same number of pixels (in both Interlaced and Progressive mode). Each field block (top or bottom) defines one field of one region and each block must be contiguous, aligned on a 32-bit boundary in the SDRAM.

Two pointers, defined in the header, give the first bitmap pixel address in each field block. The bottom field block can be merged with the top field block (line-based format).

Figure 34: OSD CLUT Programming Flowchart



In this example, the 4th color of the palette (programmed in SDRAM) is loaded at the CLUTAdd[7:0] address in the OSD CLUT (18th address of the OSD CLUT). Therefore, the 4th color of the palette (pointed by Palettep[31:0]) is the first color to be loaded in the OSD CLUT.

4.1.5 Mixing OSD and Video Signals

The mixing function is used to blend each OSD pixel with the corresponding pixel generated by the video planes behind the OSD. The mix weight is a programmable parameter and can be set for each OSD region, or for each color. The Mixing Mode bits (MiMo[1:0]) are used to select one of four possible mixing modes.

Table 13: Mixing Modes

MiMo[1:0]	Description
00	Pixel Mixing mode
01	Global Mixing mode

Table 13: Mixing Modes

MiMo[1:0]	Description
10	Proportional Mixing mode
11	No Mixing mode

4.1.5.1 Pixel Mixing Mode

In Pixel Mixing mode, the only mixing factor applied is the Alpha mixing factor parameter (Alpha[7:0]) programmed in the CLUT.

4.1.5.2 Global Mixing Mode

In Global Mixing mode, the mixing factor applied is the Mix-Weight parameter (MiW[7:0]) programmed in the header. These bits define the mixing factor (transparent level) for each OSD region used for blending the OSD and video signals. This mixing value is applied to the entire OSD region and the Alpha mixing factor in the CLUT is ignored.

4.1.5.3 Proportional Mixing Mode

In Proportional Mixing mode, the mixing factor is the result of the multiplication or division of the Alpha mixing factor (Alpha[7:0]) programmed in the CLUT by the Mix-Gain parameter (MiGa[8:0]) programmed in the header. The Mix-Gain bits define the offset applied to the mixing factor (the transparent level) for each pixel of one OSD region. In this mode, if the MSB bit (MiGa[8]) is set, the operation is multiplication. If reset, the operation is division.

MiGa[8]	MiGa[7:0]	Description
0	XXh	Alpha[7:0] is divided by MiGa[7:0]/128
1	XXh	Alpha[7:0] is multiplied by MiGa[7:0]

Note: Due to the GAMMA specification, the Alpha[7:0] range is from 0 to 128 in decimal (00h is transparent, 80h is full OSD). All values greater than 80h (from 81h to FFh) are clamped to 80h.

4.1.5.4 No Mixing Mode

In No Mixing mode, the mixing factor is the full OSD. The Alpha mixing factor (Alpha[7:0]) programmed in the CLUT is ignored.

4.2 2D Graphics Accelerator

A 2D Graphics Accelerator can be used to speed-up the re-location and display of graphic images. This function will automatically generate the desired link list required to move an image block from one position to another in order to accelerate data transfers in memory.

This module reads a block from one area of the memory, processes this block (a process could be a simple copy), and writes the result in an other area. Memory addresses can be byte-aligned.

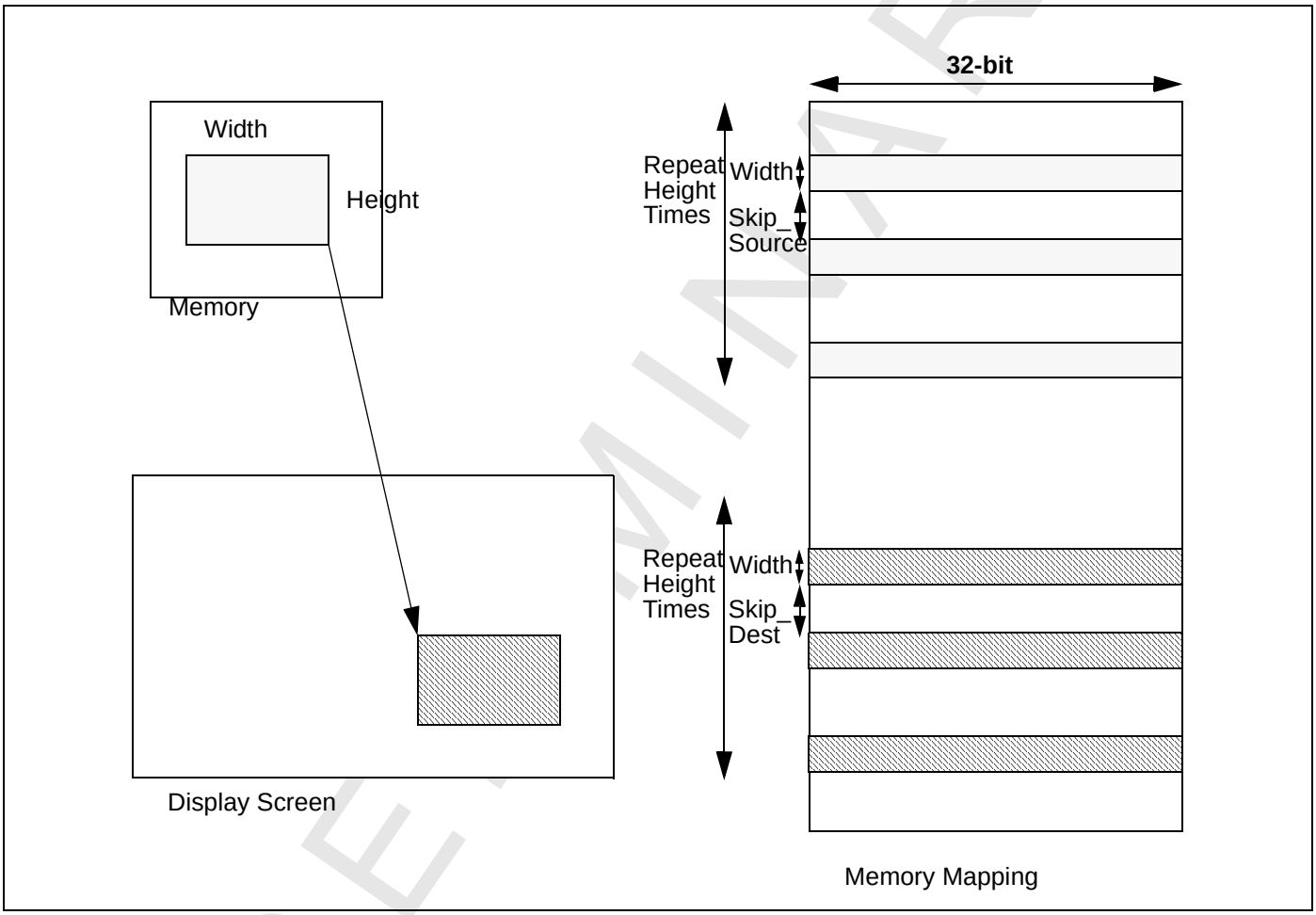
The copying is managed by a DMA engine which alleviates the CPU load.

To perform a 2D block move (with or without processing), the module must be initialized with the source and destination start addresses, the width and the height of the block and the source and destination skip value. (The skip value is equal to the number of columns of the region minus the width.) SKIP_SOURCE and SKIP_DEST data can be specified and their values can be different.

There are different types of processes that must also be selected:

- Read/Write (with or without filtered Data compression)
- Parity Checksum
- Simple transfer
- Color key based transfer: data transfered if different from a specified color (i.e. background masking)
- Filling
- Transfer with filtering (nibble to byte, parity check, ...)

Figure 35: 2D Block Move Flowchart



4.3 Graphic Application Examples

In addition to the menus required for the user interface, certain graphic applications are supported by the STV3550.

4.3.1 Teletext 1.5

- 40-character width by 26-character (12 x 10) rows

Note: Specific font formats can be required to fit with the panel resolution.

- 8 fully-defined colors for both foreground and background planes
- Mix mode (background color is fully-transparent to video)

- Subtitle mode (full-quality picture with subtitle windows)
- Newsflash mode / Box attribute (Text with solid background, other areas with standard video)
- Zoom mode: Characters and semi-graphic symbols are zoomed in by a vertical factor of 2. The zoom is done by software. Half a page is then displayable. A key enables the user to alternatively display the upper half of the page, the bottom half of the page or the complete page without zoom.

Figure 36: Teletext Level 1.5

4.3.2 Teletext Level 2.5

In Teletext 2.5, additional graphics objects are decoded to enhance Teletext page content. Up to 32 re-definable colors can be used by graphic objects. Moreover, up to 16 columns can be added to either side of the page. Horizontal resolution is then increased to 672 pixels per line.

4.3.3 TeleWeb

- Full support of authored content in a 640 x 480 pixel area
- Minimum colour resolution of 12 bit (RGB = 444), 24 bit recommended
- Support for a minimum of 196 colours (the default CLUT)
- Software Dithering to achieve best color matching
- Support for full (100%) and partial (30%) transparency
- Support for GIF, JPEG and PNG image formats, including animation

- Background plane

Figure 37: TeleWeb

4.4 Picture Compositor

The Picture Compositor is used to mix the video plane improved by the Picture Structure Improvement (YSI/CTI) module in the Video Display pipeline, with the graphics plane from the OSD pipeline. The Picture Compositor also includes a cursor plane and background color plane. A 256-level Transparency mode can be used for mixing the various planes. The following figure shows a typical configuration of the video/graphics datastream flow through the Picture Compositor.

Figure 38: Typical Configuration of Picture Compositor

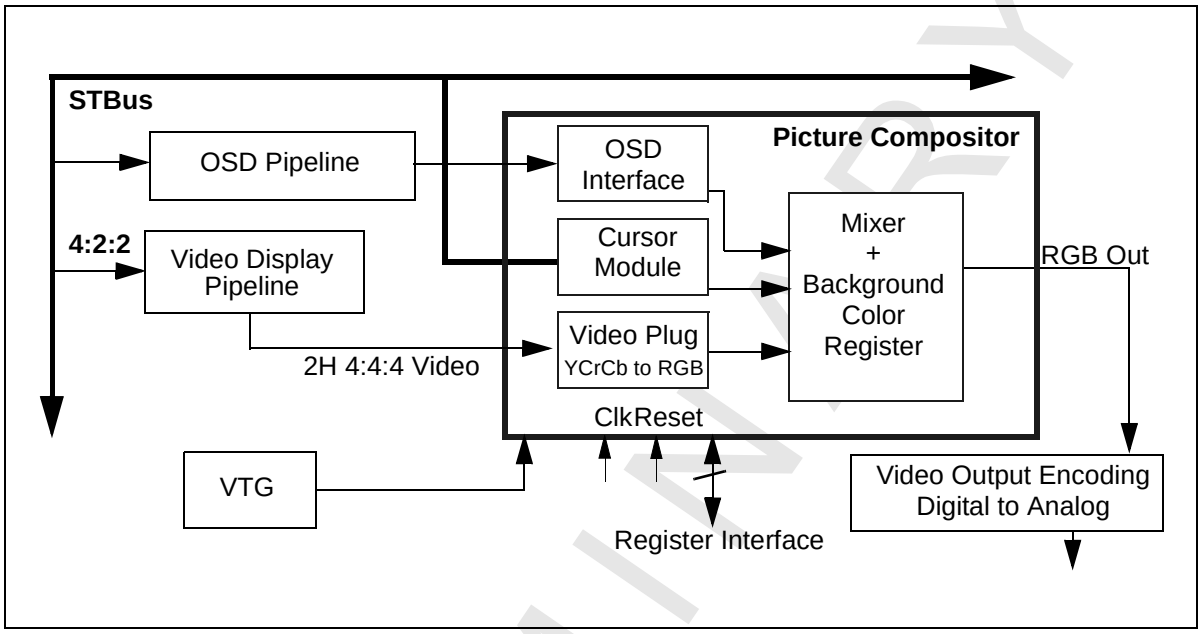
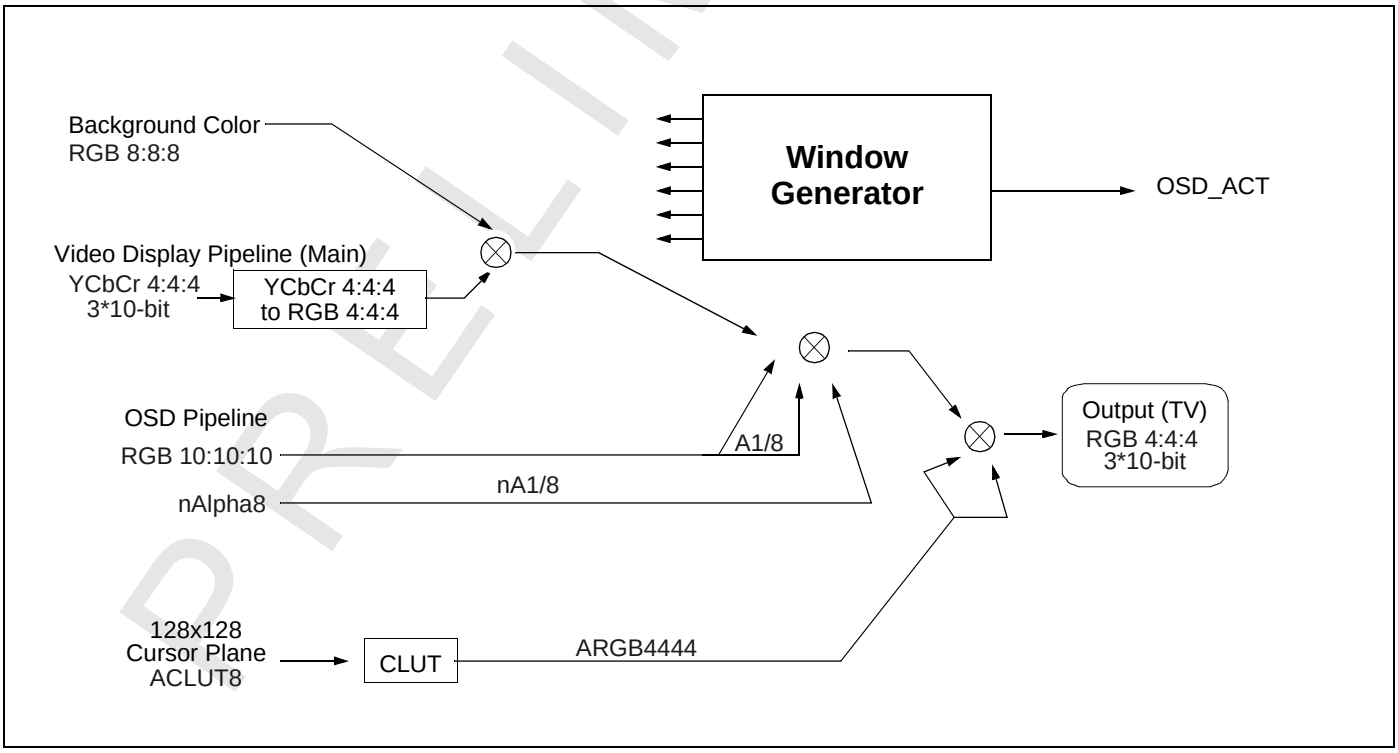


Figure 39: Picture Compositor Block Diagram



The Picture Compositor is a real-time multilayer (or multiplane) digital mixer. It can compose a single output stream from up to four sources:

- Background color plane (programmable register)
- Video plane (D1 input, *once captured into the local memory)*
- Cursor plane
- On-Screen Display (OSD) and Graphics (GFX) plane

4.4.1 Background Color Plane

The background color is stored in a 24-bit RGB register that acts as a full-screen plane filled with a solid color. This plane is always opaque with no alpha value associated. It will be seen on each screen location, according to the resulting transparency of all the foreground planes.

This plane, associated with the video plane and its mixing factor, can perform a contrast function used to adjust the contrast of the video plane, independently of the OSD plane.

4.4.2 Video Plane

The Video Display Pipeline provides a video stream that can be resized and pre-processed on the video input port. The video plane has the following characteristics:

- YCbCr 4:4:4 format with 10-components,
- SD format compatible,
- Single Viewport mechanism with a global alpha value,
- Color Space Conversion matrix (YCbCr 601 to RGB).

4.4.3 Cursor Plane

The curser plane is defined as a 128 x 128 pixel area stored in an external memory chip. Each cursor entry is an ARGB4444 pixel. The alpha factor of 4-bits is for antialiasing the cursor pattern on top of the composited output picture.

List of features :

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

4.4.4 Graphics Plane

List of features :

- On-Screen Display and Graphics data arrives in ARGB 8:10:10:10 format,
- Gain and offset adjustment.

The table below summarizes the various source formats for each GAMMA Picture Compositor plane.

Table 14: List of Accepted Picture Compositor Formats

Format	Description	Use	Cursor Plane	Graphics Plane	Video Plane	Background Color Plane
RGB888 (RGB24)	24-bit RGB	Only use for the background color plane with a field-based resolution (1 color only for a complete field).				X
ACLUT8	Bitmapped graphics with 8-bit per pixel (256 colors). For Graphics, the CLUT output is 8-bit per color component (RGB) and 8 bit for transparency (alpha). For the Cursor, the CLUT output is 4-bit per color component (RGB) and 4-bit for transparency (alpha)	Simple graphics with reduced memory needs. 8-bits per pixel can use the byte manipulation modes.	X	X		
YCbCr 4:2:2	Raster-based 4:2:2 picture from the D1 input. 4:2:2 to 4:4:4 up-conversion is performed in the video pipeline output stage before sending data to the Picture Compositor.	Format used for the video display pipeline output			X	

5 Output Stage

Note: Some descriptions in this section are not applicable for LCD applications.

5.1 Color Space Adaptor (CSA) and Interpolator

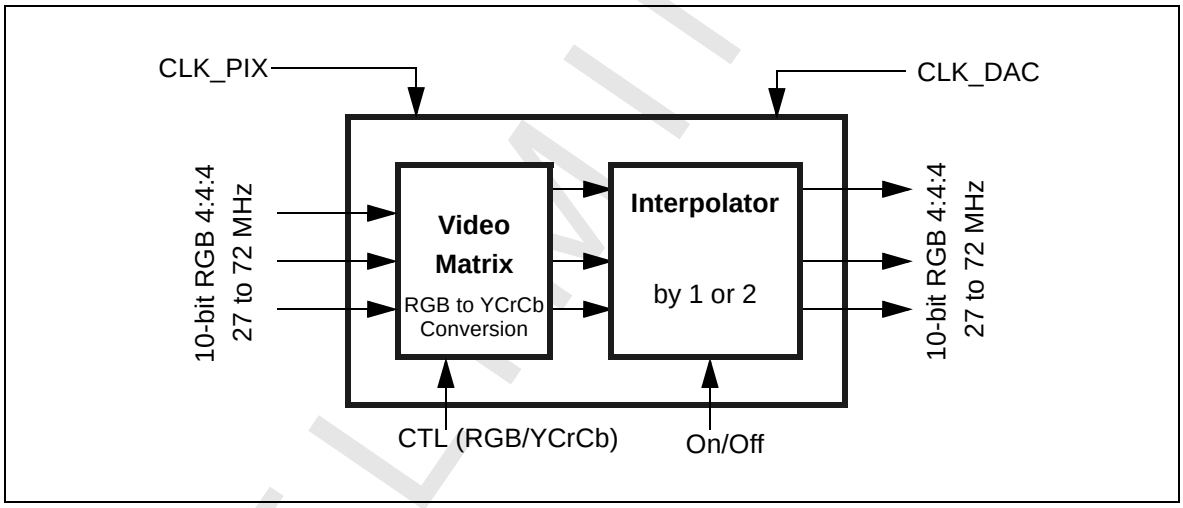
5.1.1 Main Features

- RGB to YUV Convertion
- Up-sampling by two
- GFX_ACTIVE Programmable Delay

5.1.2 General Description

At the output of the Picture Compositor, the video format is 3x10-bit RGB 4:4:4. The video signal is first sent to a Color Space Adaptor that outputs RGB signals. The bit stream is then interpolated in order to double the bit rate. The interpolation function can be used in the event of a double-rate OSD, or mixed mode (OSD and video).

Figure 40: CSA and Interpolation Diagram



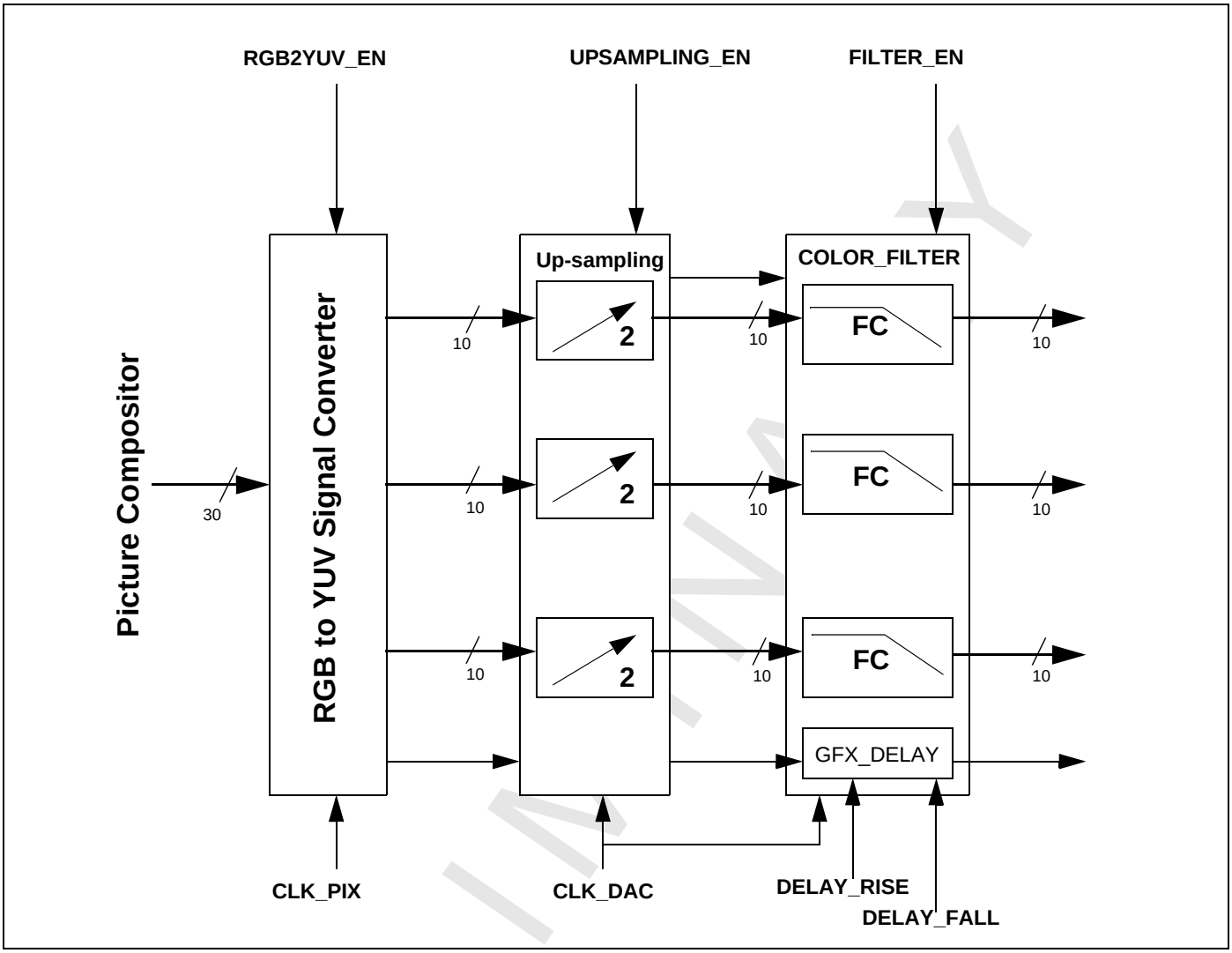
5.1.3 Up-sampling

In this sub-block, incoming RGB data are up-converted using the CLK_DAC clock. There are two possible cases:

1 the CLK_PIX and CLK_DAC clock frequency is the same: only the clock domain is changed,

2 or the CLK_DAC clock frequency is twice the CLK_PIX clock frequency: the clock domain is changed and the outputs signals are padded with a zero level every two samples.

Figure 41: Color Space Adaptor Block Diagram



Note: RGB to YUV conversion and color filter functions are not used in LCD applications

5.1.4 GFX_ACTIVE Signal

The chip delivers a GFX_ACTIVE signal that is high during the graphic insertion period and low during the video period. This signal can be used to control an external switch to insert the OSD when the video is generated externally (typically with a VGA source). Each edge of this signal can be adjusted in timing when compared with the Hout sync pulse.

The delay is adjustable from -2 DAC_clock cycles to +5 DAC_clock cycles for the rising edge , and from -3 DAC_clock cycles to +4 DAC_clock cycles for the falling edge, as shown in the table below.

Table 15: Delay on Rising/Falling Edges of GFX_ACTIVE Signal

DELAY_RISE	Delay Value	DELAY_FALL	Delay Value
0000	CSA Stage Number - 7 clock cycles	0000	CSA Stage Number - 7 clock cycles
0001	CSA Stage Number - 6 clock cycles	0001	CSA Stage Number - 6 clock cycles
0010	CSA Stage Number - 5 clock cycles	0010	CSA Stage Number - 5 clock cycles

Table 15: Delay on Rising/Falling Edges of GFX_ACTIVE Signal

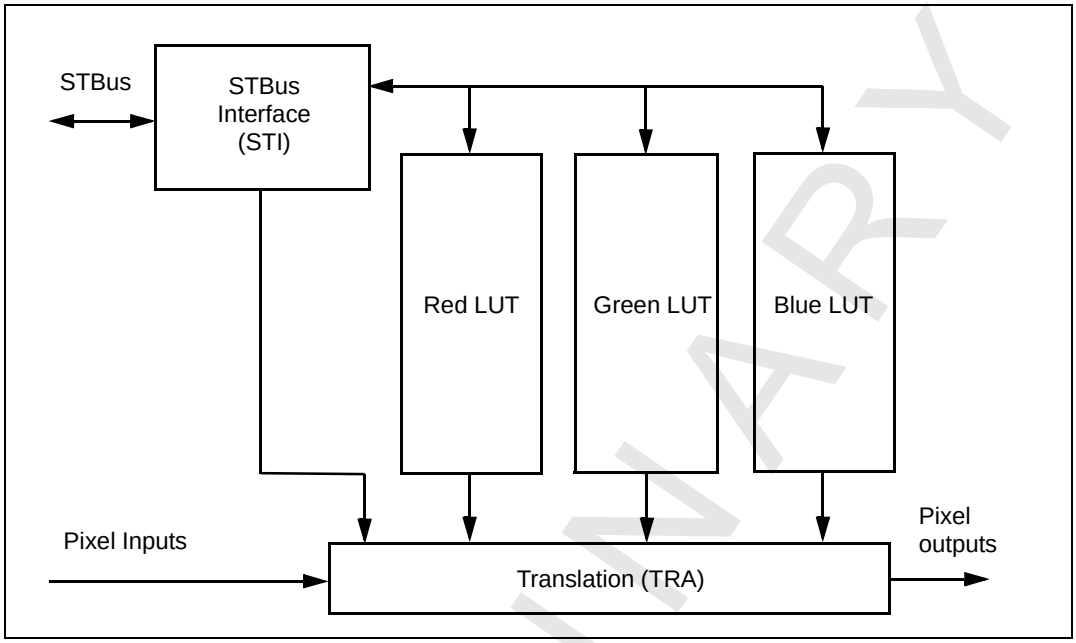
DELAY_RISE	Delay Value	DELAY_FALL	Delay Value
0011	CSA Stage Number - 4 clock cycles	0011	CSA Stage Number - 4 clock cycles
0100	CSA Stage Number - 3 clock cycles	0100	CSA Stage Number - 3 clock cycles
0101	CSA Stage Number - 2 clock cycles	0101	CSA Stage Number - 2 clock cycles
0110	CSA Stage Number - 1 clock cycle	0110	CSA Stage Number - 1 clock cycle
0111 (Standard)	CSA Stage Number	0111 (Standard)	CSA Stage Number
1000	CSA Stage Number + 1 clock cycle	1000	CSA Stage Number + 1 clock cycle
1001	CSA Stage Number + 2 clock cycles	1001	CSA Stage Number + 2 clock cycles
1010	CSA Stage Number + 3 clock cycles	1010	CSA Stage Number + 3 clock cycles
1011	CSA Stage Number + 4 clock cycles	1011	CSA Stage Number + 4 clock cycles
1100	CSA Stage Number + 5 clock cycles	1100	CSA Stage Number + 5 clock cycles
1101	CSA Stage Number + 6 clock cycles	1101	CSA Stage Number + 6 clock cycles
1110	CSA Stage Number + 7 clock cycles	1110	CSA Stage Number + 7 clock cycles
1111	CSA Stage Number + 8 clock cycles	1111	CSA Stage Number + 8 clock cycles

Obviously, the value of the DELAY_RISE and DELAY_FALL values must be compatible with the GFX_ACTIVE signal waveform. For example, if DELAY_RISE = 0111 and DELAY_FALL = 0000, the ninimum duration of the high level of the GFX_ACTIVE signal must be 10 CLK_DAC clock cycles.

5.2 Gamma Correction

The Gamma correction is a completely programmable block, which is used to correct the input pixels. The CPU supplies the gamma correction curves by writing the data into the Look Up Tables located in this block. This block can be bypassed, in which case the flow of pixels is left unchanged.

Figure 42: Gamma Correction Block Diagram



5.3 Perfect Color Engine

PCE (Perfect Color Engine) dithers an input 10 bit video stream down to 4-6-8-10 output bits.The dithering is done in space and time in such a way that the eye does not perceive objectionable artifacts such as:

- Fixed dither patterns
- Contours
- Flickering pixels
- Phase correlated flickering, which creates wave patterns known as swimming.

Note: The dithering efffect tends to average neighboring pixels and smooth the color tonal transition and contours of the image. It works by spreading the quantization error over neighboring pixels (spatially and temporally).

5.4 Digital Video Output Stage

5.4.1 Introduction

This block performs the digital video output stage. This block generates the Data Enable (DE) signal on account of the register configuration, it also resynchronises and recalibrates the Vsync and Hsync signals according to the register configuration. This block delivers the Data Clock (Dclk) with a delay selected by the register and fixes the default output state of all the digital video interface. This block also provide a reduce interface with a data valid on each edge of the clock.

The output frequency range is 22MHz up to 72MHz according to the user configuration.

5.4.2 Vsync Output Capability

In both cases we can select the output signal polarity, with a dedicated register bit.

```
case i_vsync = vs_pad
```

The VSYNC output is in the ‘clk_vtg’ domain. The pulse size is adjusted by the VTG block.

```
case i_vsync = vs_rst_100_dac (clk_dac domain - basic mode)
```

The VSYNC output pad is in the ‘clk_dac’ domain. We could delay the first edge (2^{14}-1 clock period) and we could adjust the pulse width (2^{16}-1 clock period).

5.4.3 Hsync Output Capability

The Hsync source choice is fully independent of the Vsync choice. The user must take care to keep the two choices in a consistent state.

In both cases we could select the output signal polarity, with a dedicated register bit.

```
case i_hsync = hs_pad
```

In the case of master mode with the Digital Video Scan, we can choose the H_ref signal as the STV3550 HSYNC output. This signal is in the ‘clk_vtg’ domain. The Pulse size is controlled by the VTG block.

In the case of slave mode with the Digital Video Scan, we can choose the H_gen signal as the STV3550 HSYNC output. This signal is in the ‘clk_dac’ domain. The Pulse size is controlled by the VTG block.

```
case i_hsync = hs_pix
```

The HSYNC output pad is in the ‘clk_dac’ domain. We could delay the first edge (2^{12}-1 clock period) and we could adjust the pulse width (2^{8} clock period).

5.4.4 Csync Output Capability

In the Csync mode the Hsync output PAD is replaced by the Csync output signal.

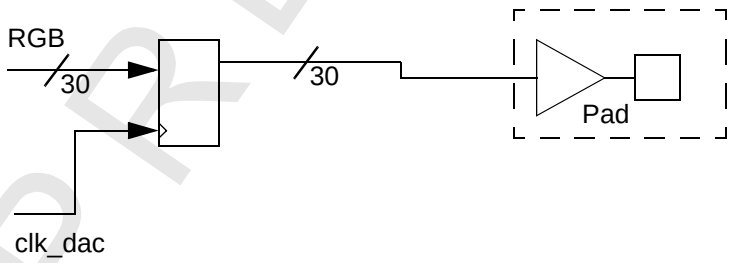
The Csync signal is the Hsync signal with a polarity inverted when the vsync signal is active.

We could select the Csync polarity with the Hsync polarity register.

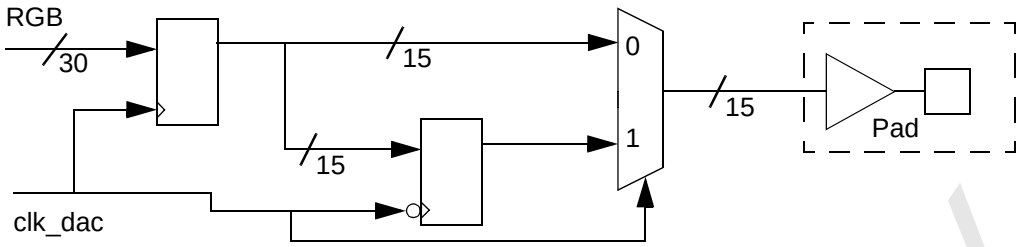
5.4.5 RGB Output

We could select the Digital Video Output signal polarity. We have to take two cases into account:

Output all of the signal with a dedicated pin:



Multiplex the output signal to reduce the number of pins:



In the case of 3 x 10 bits ouput or 3 x 8 bits output we could swap the MSB and LSB signal.

When we change the output mode we change the pinout according to Table 16.

Table 16: Pinout

STV3550 Pin Number	STV3550 Pad	Basic Output 3x10 bits	Output 3x9 bits	Output 3x8 bits	Output 3x7 bits	Output 3x6 bits	Output 3x5 bits	Output 3x4 bits
202	P0	R0	R0	R0	R0	R0	R0	R0
203	P1	R1	R1	R1	R1	R1	R1	R1
204	P2	R2	R2	R2	R2	R2	R2	R2
205	P3	R3	R3	R3	R3	R3	R3	R3
206	P4	R4	R4	R4	R4	R4	R4	G0
208	P5	R5	R5	R5	R5	R5	G0	G1
1	P6	R6	R6	R6	R6	G0	G1	G2
2	P7	R7	R7	R7	G0	G1	G2	G3
3	P8	R8	R8	G0	G1	G2	G3	B0
4	P9	R9	G0	G1	G2	G3	G4	B1
6	P10	G0	G1	G2	G3	G4	B0	B2
7	P11	G1	G2	G3	G4	G5	B1	B3
8	P12	G2	G3	G4	G5	B0	B2	0
9	P13	G3	G4	G5	G6	B1	B3	0
10	P14	G4	G5	G6	B0	B2	B4	0
12	P15	G5	G6	G7	B1	B3	0	0
13	P16	G6	G7	B0	B2	B4	0	0
14	P17	G7	G8	B1	B3	B5	0	0
15	P18	G8	B0	B2	B4	0	0	0
16	P19	G9	B1	B3	B5	0	0	0
18	P20	B0	B2	B4	B6	0	0	0
19	P21	B1	B3	B5	0	0	0	0
20	P22	B2	B4	B6	0	0	0	0
21	P23	B3	B5	B7	0	0	0	0
22	P24	B4	B6	0	0	0	0	0

Table 16: Pinout

STV3550 Pin Number	STV3550 Pad	Basic Output 3x10 bits	Output 3x9 bits	Output 3x8 bits	Output 3x7 bits	Output 3x6 bits	Output 3x5 bits	Output 3x4 bits
26	P25	B5	B7	0	0	0	0	0
27	P26	B6	B8	0	0	0	0	0
28	P27	B7	0	0	0	0	0	0
29	P28	B8	0	0	0	0	0	0
30	P29	B9	0	0	0	0	0	0

Table 17 shows the SWAP output:

Table 17: SWAP Output

STV3550 Pin Number	STV3550 Pad	Basic Output 3x10 bits	Output 3x8 bits
202	P0	R9	R7
203	P1	R8	R6
204	P2	R7	R5
205	P3	R6	R4
206	P4	R5	R3
208	P5	R4	R2
1	P6	R3	R1
2	P7	R2	R0
3	P8	R1	G7
4	P9	R0	G6
6	P10	G9	G5
7	P11	G8	G4
8	P12	G7	G3
9	P13	G6	G2
10	P14	G5	G1
12	P15	G4	G0
13	P16	G3	B7
14	P17	G2	B6
15	P18	G1	B5
16	P19	G0	B4
18	P20	B9	B3
19	P21	B8	B2
20	P22	B7	B1
21	P23	B6	B0
22	P24	B5	0

Table 17: SWAP Output

STV3550 Pin Number	STV3550 Pad	Basic Output 3x10 bits	Output 3x8 bits
26	P25	B4	0
27	P26	B3	0
28	P27	B2	0
29	P28	B1	0
30	P29	B0	0

Table 18 shows the multiplex output:

Table 18: Multiplex Output

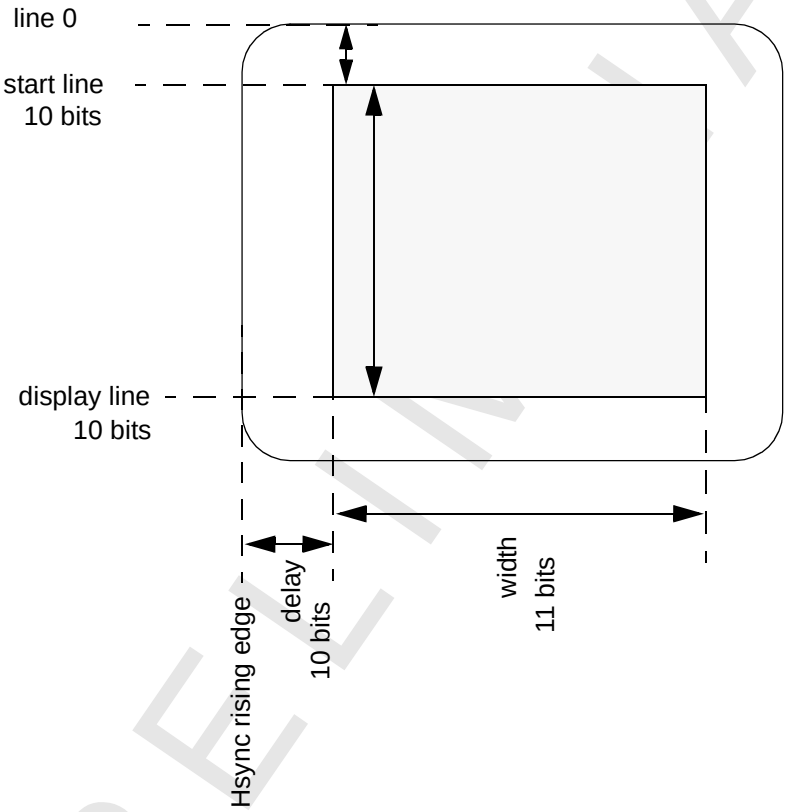
STV3550 Pin Number	STV3550 Pad	Multiplex Output 3x10bits Phase 0	Multiplex Output 3x10 bits Phase 1	Multiplex Output 3x8 bits Phase 0	Multiplex Output 3x8 bits Phase 1
202	P0	B0	G4	B0	G4
203	P1	B1	G5	B1	G5
204	P2	B2	G6	B2	G6
205	P3	B3	G7	B3	G7
206	P4	B4	R0	B4	R0
208	P5	B5	R1	B5	R1
1	P6	B6	R2	B6	R2
2	P7	B7	R3	B7	R3
3	P8	B8	G9	G0	R4
4	P9	B9	R9	G1	R5
6	P10	G0	R4	G2	R6
7	P11	G1	R5	G3	R7
8	P12	G2	R6	0	0
9	P13	G3	R7	0	0
10	P14	G8	R8	0	0
12	P15	0	0	0	0
13	P16	0	0	0	0
14	P17	0	0	0	0
15	P18	0	0	0	0
16	P19	0	0	0	0
18	P20	0	0	0	0
19	P21	0	0	0	0
20	P22	0	0	0	0
21	P23	0	0	0	0

Table 18: Multiplex Output

STV3550 Pin Number	STV3550 Pad	Multiplex Output 3x10bits Phase 0	Multiplex Output 3x10 bits Phase 1	Multiplex Output 3x8 bits Phase 0	Multiplex Output 3x8 bits Phase 1
22	P24	0	0	0	0
26	P25	0	0	0	0
27	P26	0	0	0	0
28	P27	0	0	0	0
29	P28	0	0	0	0
30	P29	0	0	0	0

5.4.6 Data Enable Output

This signal is fully generated according to the control register value. The user must assume the video window is correct according to the display data (video + OSD + background).

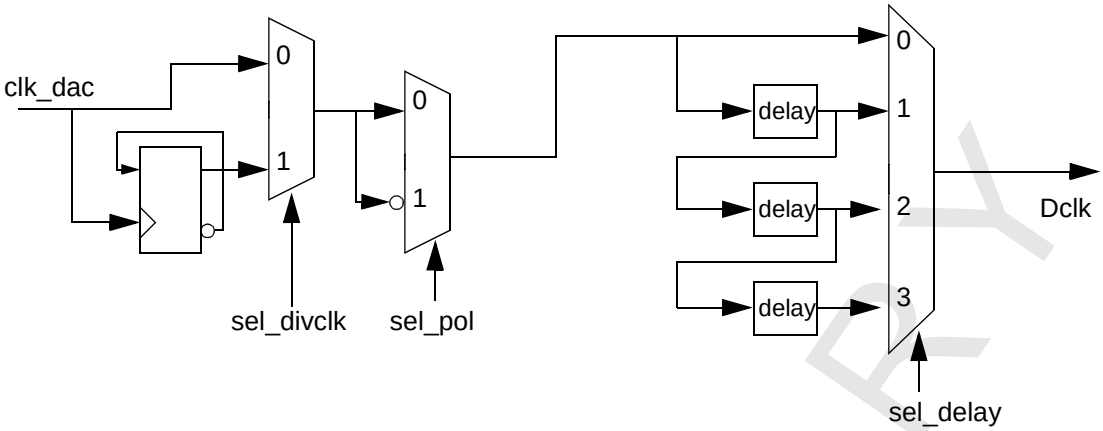


The Data Enable signal is in the clk_dac domain. This signal generates a jitter free signal only if the Hsync and Vsync signal are output in the same clock domain (‘clk_dac’). The reference signals to generate this signal are hs_pix, vs_rst_100_pix and lineupcnt_pix.

We could select the DE polarity with a dedicated register.

5.4.7 Data Clock Output

We could output the block clock with a choice of polarity and four different delays according to the configuration register.



We could also select an output of the original clock frequency or the clock frequency divided by 2.

5.4.8 Pad Control

If a PAD is not used by the Video application a way to control its state is provided using software.

One bit register control if the PAD is in video mode or in software mode, another bit register controls the software level for the PAD.

The PAD stays in push pull output in both cases. The software state is available only on the next edge of the clk_dac clock. The clk_dac clock must be enabled to change the PAD state.

We are able to reconfigure the following PAD:

`P_video<12:29>`

`dvo_de`

`Vsync`

`dvo_clk_2162`

The user must take care of the consistency choice of the Digital Video Output Interface.

5.4.9 Register

We have a dedicated register interface (STbus T1).

Register	Address	Description
	0x18	Pad output
	0x14	Pad Control
	0x10	DE Vertical control
	0x0C	DE Horizontal control
	0x08	Vsync control
	0x04	Hsync control
	0x00	block control

6 CPU and System Management Functional Description

6.1 ST20 C2C200 CPU Core

6.1.1 General Information

This is the 32-bit ST20 C2C200 CPU core from the ST20 CPU family with a programmable operating frequency between 4 MHz and 100 MHz, depending on the operating mode (full speed or low power). The cell includes a 4-kB instruction cache memory, a 4-kB data cache memory, a 8-kB SRAM, a Diagnostic Controller Unit (DCU) and an Interrupt Controller (ITC).

6.1.2 Main Features

- 32-bit VL-RISC Processor
- 60 MIPS (Dhrystone 2.1) at 100 MHz clock
- 16 cache registers of 32-bits providing fast access to local variables
- Fast-context switch time (less than 1 µs)
- Up to 16 interrupt inputs
- Includes 4-kB instruction cache memory, 4-kB data cache memory, and 8-kB SRAM for DMA transactions or cache increase
- Includes a Diagnostic Controller Unit for Real Time Debugging via a JTAG port and ST20 emulator

6.2 ST Bus Interconnect Overview

The STV3550 Interconnect is based on the STBus architecture. Its main function is to allow data to flow between different blocks called “initiators” and “targets”. Each initiator can make a transaction with any target. The priority of the transaction depends on the arbitration made by each interconnect module or arbiter.

6.3 STV3550 Memory Interface

In the STV3550, the memory interface is in four parts:

1. The **External Memory Interface** (EMI) manages all memory accesses from a protocol point of view.
2. The **Pad Glue Logic** (PGL) is used for multiplexing and sharing functions on the memory pads in order to optimize the number of pins. The Pad Glue Logic mainly implements combinational logic as well as some control logic.
3. The **Pad Logic** (PL) solves timing issues encountered when connecting high speed devices on a printed circuit board (PCB). The Pad Logic mainly implements the bi-directional pad and its associated control logic to accommodate the setup and hold time of memory devices.
4. The **Delay Locked Line** (DLL)

6.3.1 Memory Devices

The STV3550 requires a ROM device that contains the application software and a RAM device for video and software data. If the ROM bandwidth is not sufficient, the application software may be downloaded from ROM to RAM when booting the STV3550. (The download operation is implemented in software.)

The ROM device may be easily replaced by a Flash or a Synchronous Flash (SFlash) device, but the STV3550 is not able to support bursted accesses because there is no dedicated clock pad for a SFlash device. Therefore, a Synchronous Flash device may be connected, but it will be used in standard asynchronous mode by the STV3550.

The maximum ROM size is 16 MBits (or 2 MBytes), as the device is connected on 20 address lines and 16 data lines.

Typical Flash devices, such as STM-29W160DT (STMicroelectronics) or AM29BL162C (AMD) are compliant with the STV3550 Memory Interface.

The RAM device should be a Synchronous DRAM (SDRAM). A dedicated clock pad is available on the STV3550. The maximum memory size is 256 MBits (or 32 MBytes), connected on 14 address lines and 16 or 32 data lines. The SDRAM clock frequency may be up to 100 MHz.

Typical SDRAM devices, such as µPD4564163 (NEC) or KM432S2030C (SAMSUNG) are compliant with the STV3550 Memory Interface.

Memory devices can be connected in different ways to accommodate a large number of applications.

6.3.2 Configuring the STV3550 Memory Interface during Boot

The STV3550 always boots on the ROM device, regardless of the SDRAM configuration. After reset, the STV3550 Memory Interface accesses the ROM at a very low speed in order to accommodate the slowest ROM devices available on the memory market.

Note: The STV3550 boot address is 0x7FFFFFFF in the Flash memory.

Since the STV3550 is able to accommodate several memory configurations, the STV3550 Memory Interface should be configured accordingly during the boot. Software is used to configure the memory (i.e. SDRAM size, data bus size, SDRAM and ROM latencies, etc.) and does not require the use of external pins.

STV3550 embeds an internal RAM that can be used during the boot until the external SDRAM is ready.

Once the STV3550 Memory Interface has been programmed properly, it is strongly recommended not to apply any further modifications.

6.3.3 Address Format

When interfacing a memory device through the STV3550 Memory Interface, the STV3550 uses a word address. Depending on the data bus size of the device, and knowing that the minimum data bus size is 16 bits:

- When using a 16-bit data word device (1 word = 2 bytes), bit 0 is the least significant address bit,
- When using a 32-bit data word device (1 word = 4 bytes), bit 1 is the least significant address bit.

Byte Enable lines (NOT_BEx signals) are provided for processing only parts of data words when required.

Note: For SDRAM, the address bus is multiplexed on 14 address lines (row, column and bank addresses), while Flash devices do not use multiplexed addresses.

6.3.4 Control Registers

The STV3550 Memory Interface is able to support various memory configurations by programming control registers.

All the control registers are implemented in the STV3550 Memory Interface and are accessible by the ST20.

6.3.5 Memory Configurations

The STV3550 Memory Interface is able to support 2 hardware configurations depending on bandwidth requirements and application constraints:

- two separate 16-bit data word buses for Flash and SDRAM devices (low-cost STV3550 applications)
- a single merged 32-bit data word bus shared by all memory devices for high bandwidth requirements (high-end STV3550 applications)

The STV3550 Memory Interface has an address translator that is helpful when designing a single board embedding both low-cost and high-end configurations without modifying the SDRAM address bus connection on the printed circuit board.

6.3.6 Clock Management and Timing Issues

The STV3550 Memory Interface provides clock management features to:

- control the polarity of the clock signal sent to the SDRAM
- choose the SDRAM clock between the internal clock or a locked clock signal generated by a Delay Locked Line (DLL)
- provide low power control on SDRAM devices

The STV3550 Memory Interface and SDRAM operate at the same frequency (100 MHz at Full Speed).

When operating at frequencies greater than 70 MHz, the DLL must be used to ensure the safe data exchange between the STV3550 Memory Interface and the SDRAM device. The SDRAM clock is included within a loop in order to better take into account the propagation time in the wires between the STV3550 and SDRAM devices.

In order to accommodate various wire configurations, the SDRAM clock loop can be internally or externally closed to the STV3550. When externally closed, a dedicated Clock Input pad is used.

When interfacing the STV3550 with high-speed memory devices such as SDRAM, the user may face timing issues such as setup- and hold-time violations.The STV3550 Memory Interface is also able to eliminate these timing problems by providing written data, strobes and addresses either on the falling or rising edge of SDRAM clock signals.

6.3.7 STV3550 Memory Interfaces

The memory interface of the STV3550 is built around 61 lines. Certain lines have single functions, while others support several functions depending on the memory configuration. By sharing functions, the number of required pins is reduced, since the STV3550 is not able to access both Flash and SDRAM devices at the same time.

Pins have been named according to their function when used in a low-cost application configuration. There are no application-specific pins for a given memory configuration. Applications

are configured by software via control registers since the STV3550 is always able to boot, regardless of the memory configuration.

Table 19: Memory Interface Pins

Signal Names	I/O	Size	Comment
FLASH_D[15:0]	I/O	16	**Flash Data bus** The Flash device is always connected to this bus. When a 32-bit data word SDRAM is used, the Flash Data bus is the first half (LSBs) of the 32-bit bus (the second half being the SDRAM Data bus (MSBs)).
ADDR[19:0]	O	20	**Address bus** The Flash device uses this entire bus as a non-multiplexed bus. The SDRAM devices uses lines 15 and 16 as Bank Activate (BA0 and BA1, respectively) and lines 0 to 11 for Row or Column Selection (refer to data word size and addresses dependencies). When a 32-bit data word SDRAM is connected, lines 18 and 19 supply the Byte Enable data (NOT_BE[1:0] active low).
NOT_CS_FLASH	O	1	**Flash Chip Select** - Active Low
SDRAM_D[15:0]	I/O	16	**SDRAM Data bus** When a 16-bit data word SDRAM is used, it is connected to this bus. When a 32-bit data word SDRAM is used, this bus becomes the second half (MSBs) of the 32-bit bus (the first half (LSBs) being the Flash Data bus).
NOT_CS_SDRAM	O	1	**SDRAM Chip Select** - Active Low
RD_NOT_WR	O	1	**Read / Write** By default, this is the SDRAM Read/Write signal. When the Flash Write Feature is activated, this line is also the Flash Read/Write signal.
NOT_BE[1:0]	O	2	**SDRAM Byte Enable** - Active Low These lines become Byte Enable 3 and 2 (active low) when a 32-bit data word SDRAM is used.
NOT_RAS	O	1	**SDRAM Row Address Strobe** - Active Low
NOT_CAS	O	1	**SDRAM Column Address Strobe** - Active Low This line may be used as a Flash Output Enable if the Flash Write Feature is used.
CKOUT_SDRAM	O	1	**SDRAM Clock** Clock generated by STV3550 for the SDRAM.
CKIN_SDRAM	I	1	**SDRAM Clock Feedback** This input is used to correctly synchronize the data sent by the SDRAM using the internal clock of STV3550. This input is also used by the DLL embedded in the STV3550.
Total Number of Pins		**61**	

6.3.8 STV3550 Memory Interface Capabilities Regarding Flash Device

6.3.8.1 Supported Flash Commands

The STV3550 is able to read the Flash memory like an asynchronous memory device.

Write operations in Flash require specific sequences on the data and address buses. Since the STV3550 can drive any data and any addresses on the buses, write operations into Flash can be performed if the Flash device is used as an asynchronous memory.

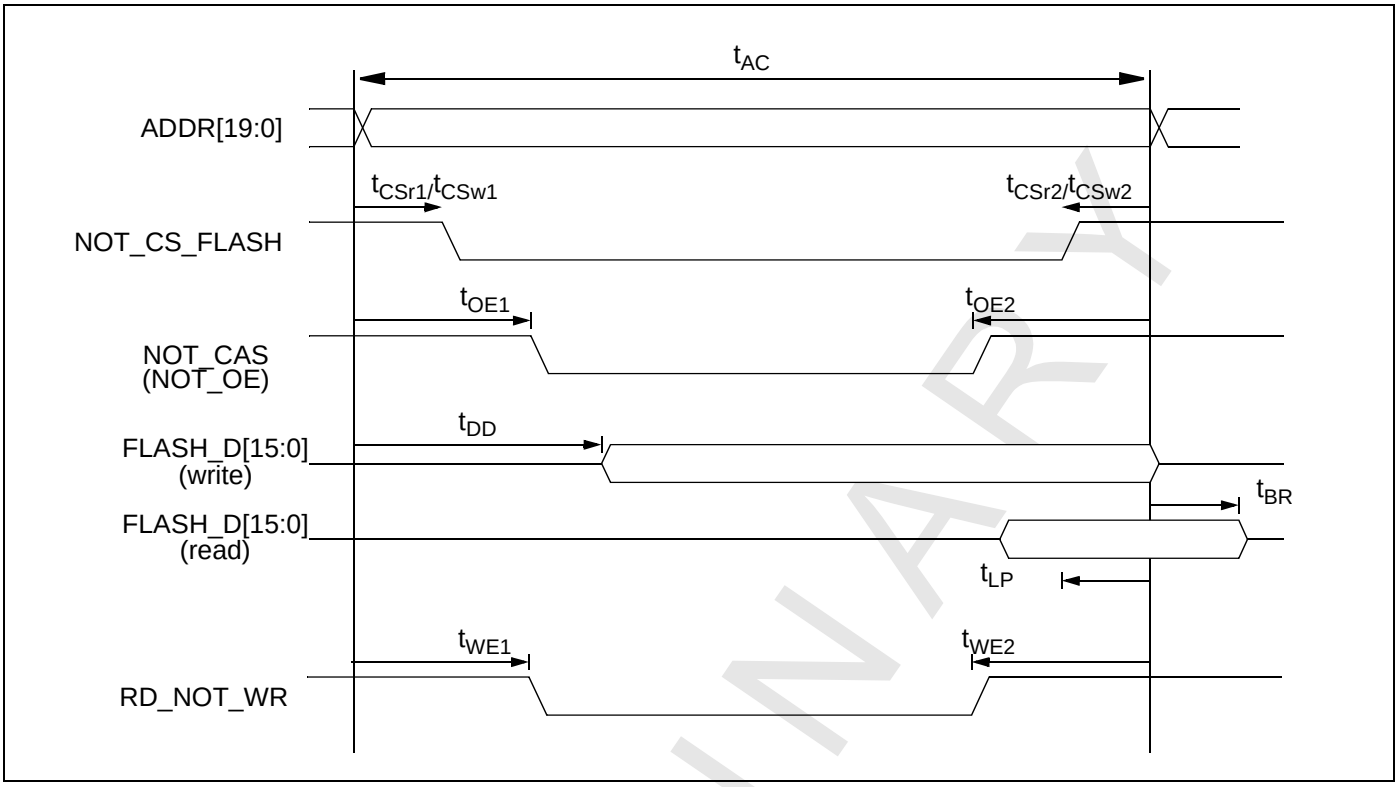
Control signals such as chip select, write enable and output enable (active low) can be configured by software, so that they are asserted or not by the STV3550 Memory Interface during each operation.

6.3.8.2 Flash Requirements

Time is expressed in number of STV3550 system clock cycles. Phase refers to half a cycle.

- Flash size up to 16 Mbits
- Access Cycle Time from 2 to 127 cycles (t_{AC})
- Bus Release Time from 0 to 15 cycles (t_{BR})
- Data Drive Delay from 0 to 31 phases after start of Access Cycle (t_{DD})
- Falling edge of NOT_CS_FLASH from 0 to 15 phases or cycles after start of Read Access Cycle (t_{CSr1})
- Rising edge of NOT_CS_FLASH from 0 to 15 phases or cycles before end of Read Access Cycle (t_{CSr2})
- Falling edge of NOT_CS_FLASH from 0 to 15 phases or cycles after start of Write Access Cycle (t_{CSw1})
- Rising edge of NOT_CS_FLASH from 0 to 15 phases or cycles before end of Write Access Cycle (t_{CSw2})
- Falling edge of NOT_CAS (NOT_OE) from 0 to 15 phases or cycles after start of Read Access Cycle (t_{OE1})
- Rising edge of NOT_CAS (NOT_OE) from 0 to 15 phases or cycles before end of Read Access Cycle (t_{OE2})
- Falling edge of RD_NOT_WR from 0 to 15 phases or cycles after start of Write Access Cycle (t_{WE1})
- Rising edge of RD_NOT_WR from 0 to 15 phases or cycles before end of Write Access Cycle (t_{WE2})
- Latch Point from 0 to 15 cycles before end of Access Cycle (t_{LP})

Figure 43: Access Cycle Timings



6.3.8.3 Default Settings For Boot Operation

The STV3550 system clock frequency is 27 MHz when the boot sequence is executed. Below are the default settings of the STV3550 Memory Interface:

- Access Cycle Time (t_{AC}) is 20 cycles (740 ns)
- Bus Release Time (t_{BR}) is 4 cycles
- Data Drive Delay (t_{DD}) is 10 phases
- Falling edge of NOT_CS_FLASH after start of Read Access Cycle ($t_{CSr1)}$ is 0
- Rising edge of NOT_CS_FLASH after start of Read Access Cycle ($t_{CSr2)}$ is 0
- Falling edge of NOT_CAS (NOT_OE) after start of Read Access Cycle ($t_{OE1)}$ is 0
- Rising edge of NOT_CAS (NOT_OE) after start of Read Access Cycle ($t_{OE2)}$ is 0
- Latch Point (t_{LP}) before end of Access Cycle is 0
- NOT_CS_FLASH is active during read only
- NOT_CAS (NOT_OE) is active during read only
- RD_NOT_WR is never active (only read operations authorized)

6.3.9 STV3550 Memory Interface Capabilities Regarding SDRAM Device

6.3.9.1 Supported SDRAM Commands

The STV3550 Memory Interface supports the following SDRAM commands:

- Mode Register Set
- Row Activate
- Precharge All

- Read and Write
- CBR Refresh
- No Operation (NOP)

The STV3550 Memory Interface always accesses SDRAM in Page mode regardless of the burst length programmed in the Mode Register Set command. It is recommended to use a burst length of 1 to prevent any problems when signal interfaces are shared by the SDRAM and Flash memories. (For more information, refer to Section 6.3.5: Memory Configurations.)

The STV3550 can be fully parameterized depending on the SDRAM device specification. Latencies, refresh interval, row/column addresses split, number of SDRAM sub-banks, etc., can be configured through the control registers.

The STV3550 Memory Interface is compliant with JEDEC and PC100 specifications which recommend applying a NOP signal for a minimum of 200 µs after the power-on sequence. Once this period has elapsed, the application software will ask the STV3550 Memory Interface to Precharge All the banks, to perform eight Refresh commands and to send the Mode Register Set command. Then the SDRAM device is ready to be used by STV3550 for standard data exchange.

Figure 44: State Table in the STV3550 Memory Interface

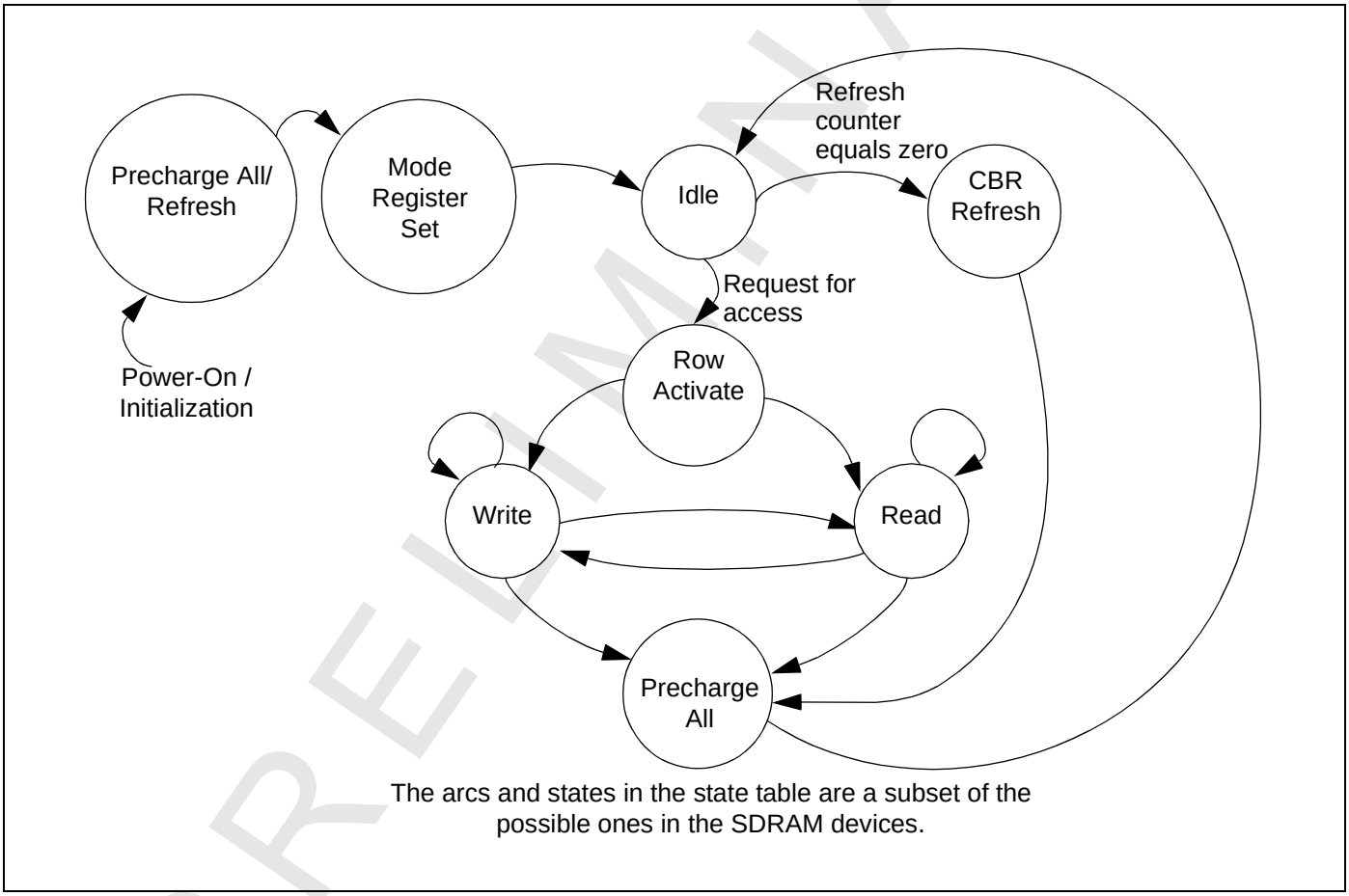


Table 20: SDRAM Commands

Command	Strobes state			Mode bits		Address	Notes
	NOT_RAS	NOT_CAS	RD_NOT_WR	Bank Select (ADDR[16], ADDR[15])	ADDR[10] AP[1]	ADDR [11:0]	
Precharge all	0	1	0	X	1	X	Used to close pages after a burst in simple page mode.
Row Activate	0	1	1	1/0	1/0	Row addresses	Opens a page (Row active)
Write	1	0	0	1/0	0	Column addresses	Writes data words (uses NOT_BE to mask data bytes)
Read	1	0	1	1/0	0	Col addr	Reads data words
Mode Register Set	0	0	0	1/0	1/0	Mode data	Done once only at start -up
CBR Refresh	0	0	1	X	X	X	CBR style refresh (Auto Refresh)
No Operation	1	1	1	X	X	X	NOP is performed between commands such as Row Activate and Read

1. Address bit 10 is also known as the Auto Precharge bit.

6.3.9.2 SDRAM Requirements

Time is expressed in number of STV3550 system clock cycles, with the SDRAM clock being the same as the STV3550 system clock.

- SDRAM size between 16Mbit and 256 Mbit
- 2 or 4 DRAM banks per SDRAM device
- up to 4096 pages per DRAM bank
- 128 to 4096 data word per page
- 16 bits or 32 bits data word size
- RAS to CAS between 1 and 8 cycles
- CAS latency between 1 and 8 cycles
- Write Recovery Time between 1 and 7 clock cycles
- Precharge Time between 1 and 16 cycles
- Refresh Period between 8 and 4096 cycles
- Refresh Time between 1 and 32 cycles
- Bus Release Time between 1 and 4 cycles
- Burst Length of 1 and Sequential Burst Capability
- SDRAM access cycle frequency up to 100 MHz

Figure 45: Typical Write Access Cycle

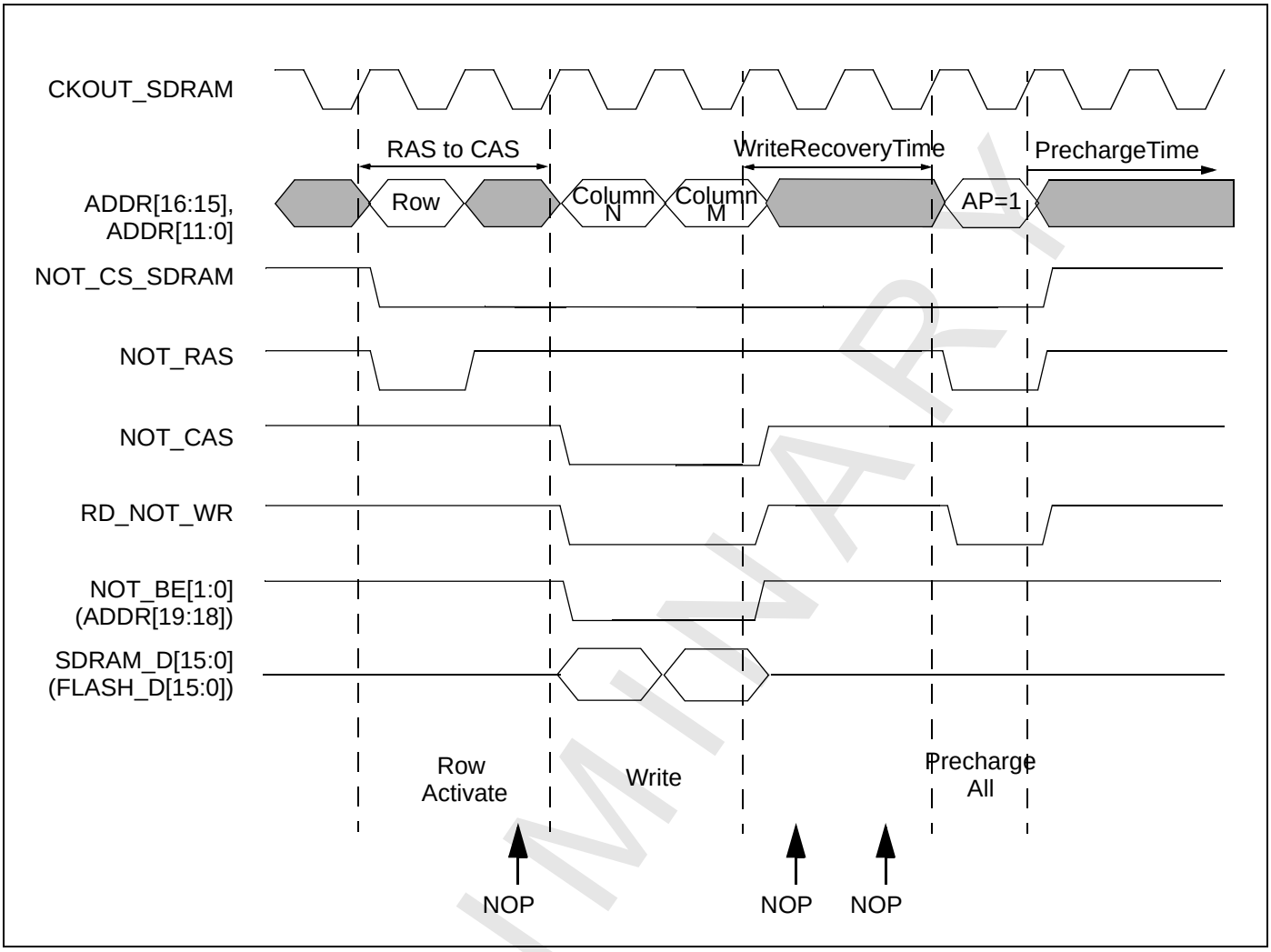


Figure 46: Typical Read Access Cycle (CAS Latency = 2)

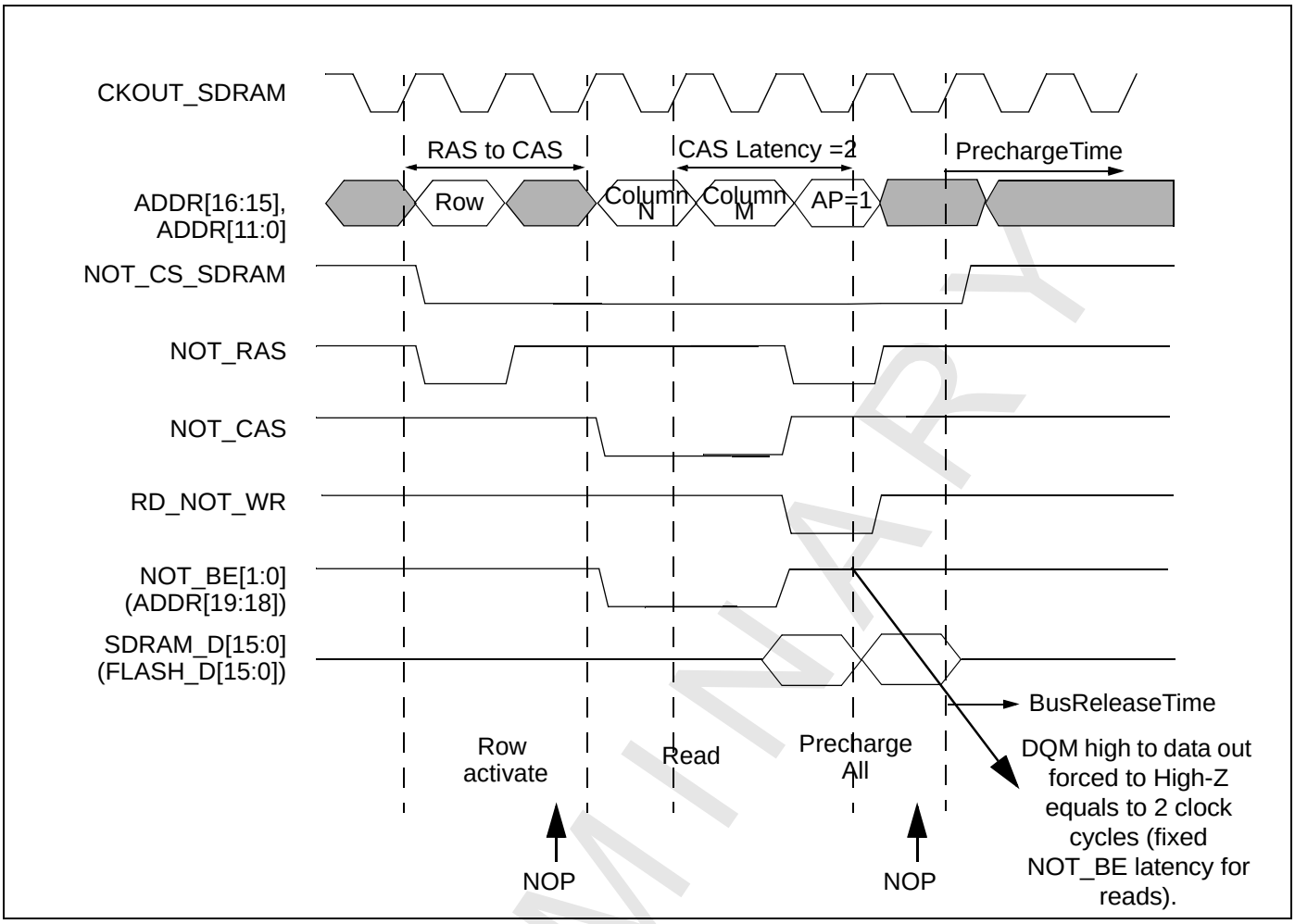
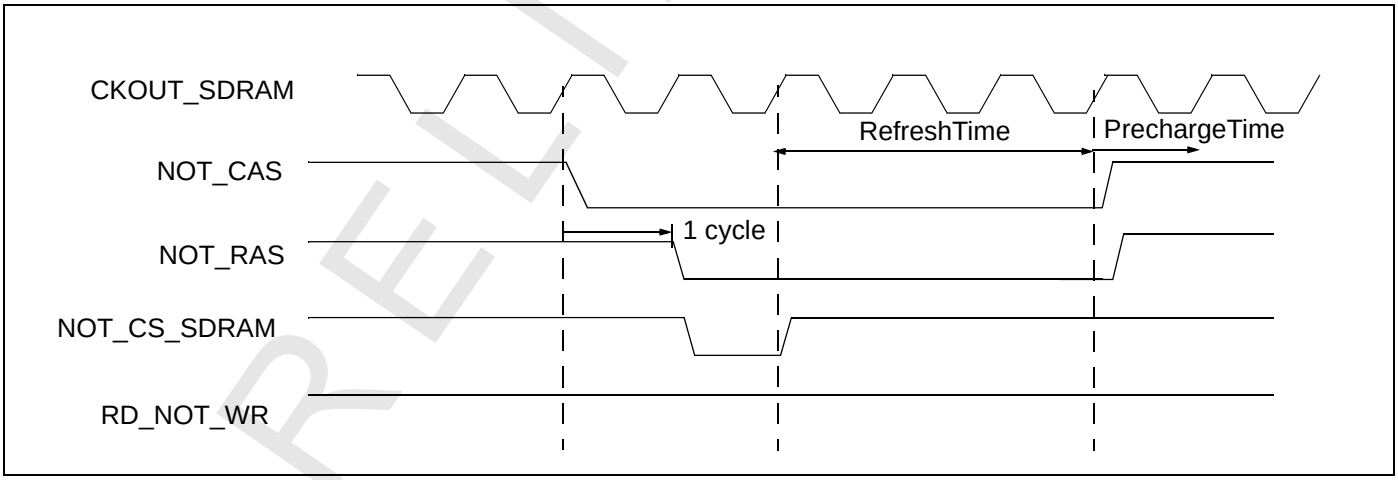


Figure 47: Typical Refresh Access Cycle



6.3.10 SDRAM Low Power Mode

The STV3550 is able to set the SDRAM into Low Power mode. Most SDRAM devices have a Clock Enable input pin that internally gates the SDRAM clock supplied by the STV3550.

The STV3550 does not have the corresponding output pin included in its memory interface, but this feature can be emulated using any of the standard STV3550 I/Os.

However, the STV3550 Memory Interface provides control over the SDRAM clock in order to switch the I/O without any timing violations according to Low Power Mode entry and exit specified for the SDRAM devices.

6.3.11 Memory Configurations

In all the following configurations, the Flash device has always 16 data lines and no more than 20 address lines.

The Flash device does not receive any clock signals from the STV3550, and the STV3550 considers the Flash memory as an asynchronous device.

The SDRAM device always receives clock signals from the STV3550.

The Address bus is always shared between the SDRAM and Flash memories. Additional memory parameters can be adjusted through software (SDRAM size and organization, Flash latency, etc.).

6.3.11.1 Low-Cost Configuration

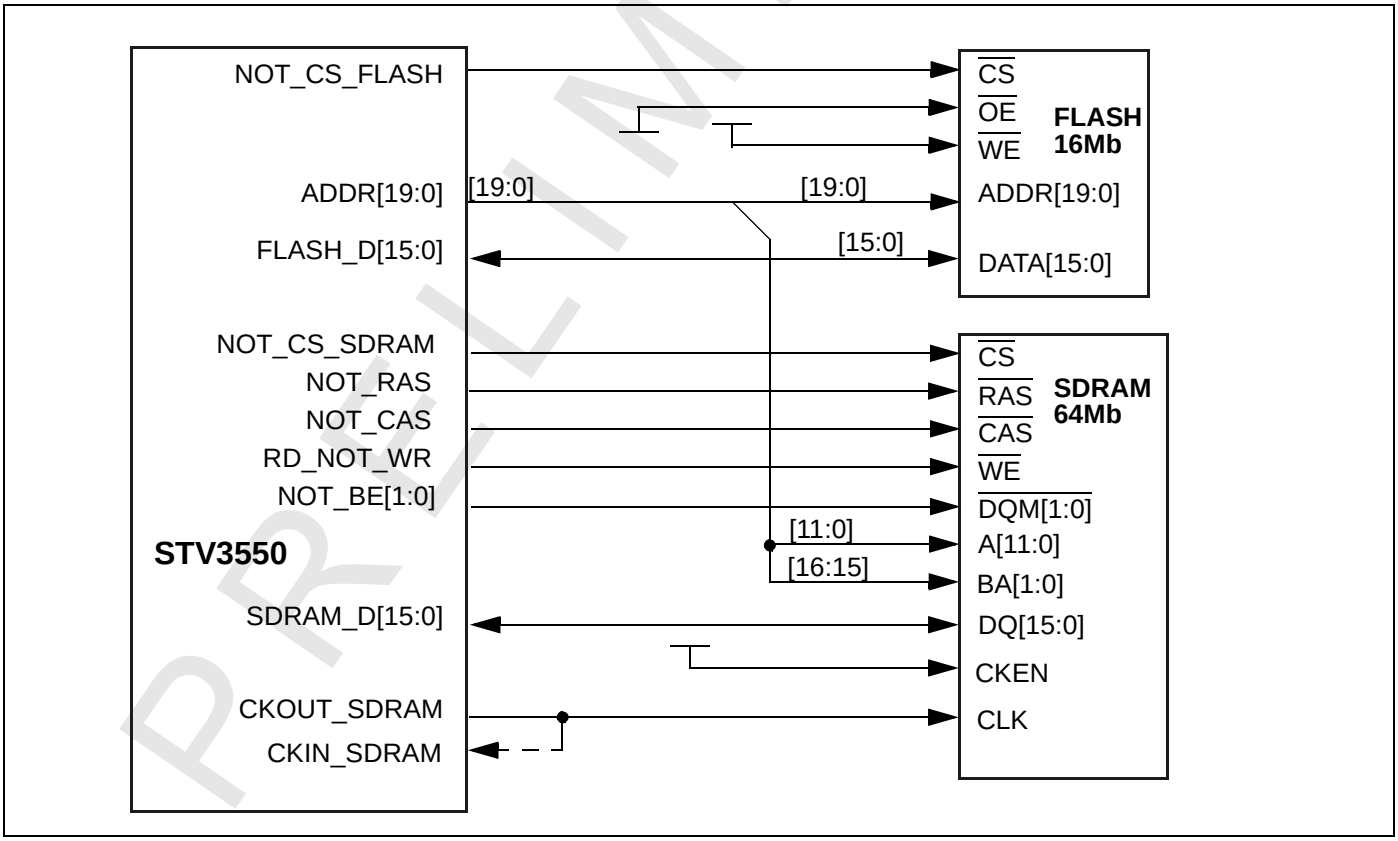
This configuration requires a SDRAM device with 16 data lines. A typical SDRAM size may be 64 Mbits (8 MBytes).

The Flash data bus is separate from the SDRAM data bus. Therefore, constraints when placing the devices are reduced, i.e. wires are short and wire loads are small.

Since 16-bit data words are used, address outputs ADDR[11:0] are connected to the SDRAM address bus and address outputs ADDR[16:15] are used as Bank Activate signals.

Flash is a Read-only device.

Figure 48: Low-Cost Configuration



6.3.11.2 High-End Configuration with one SDRAM Device

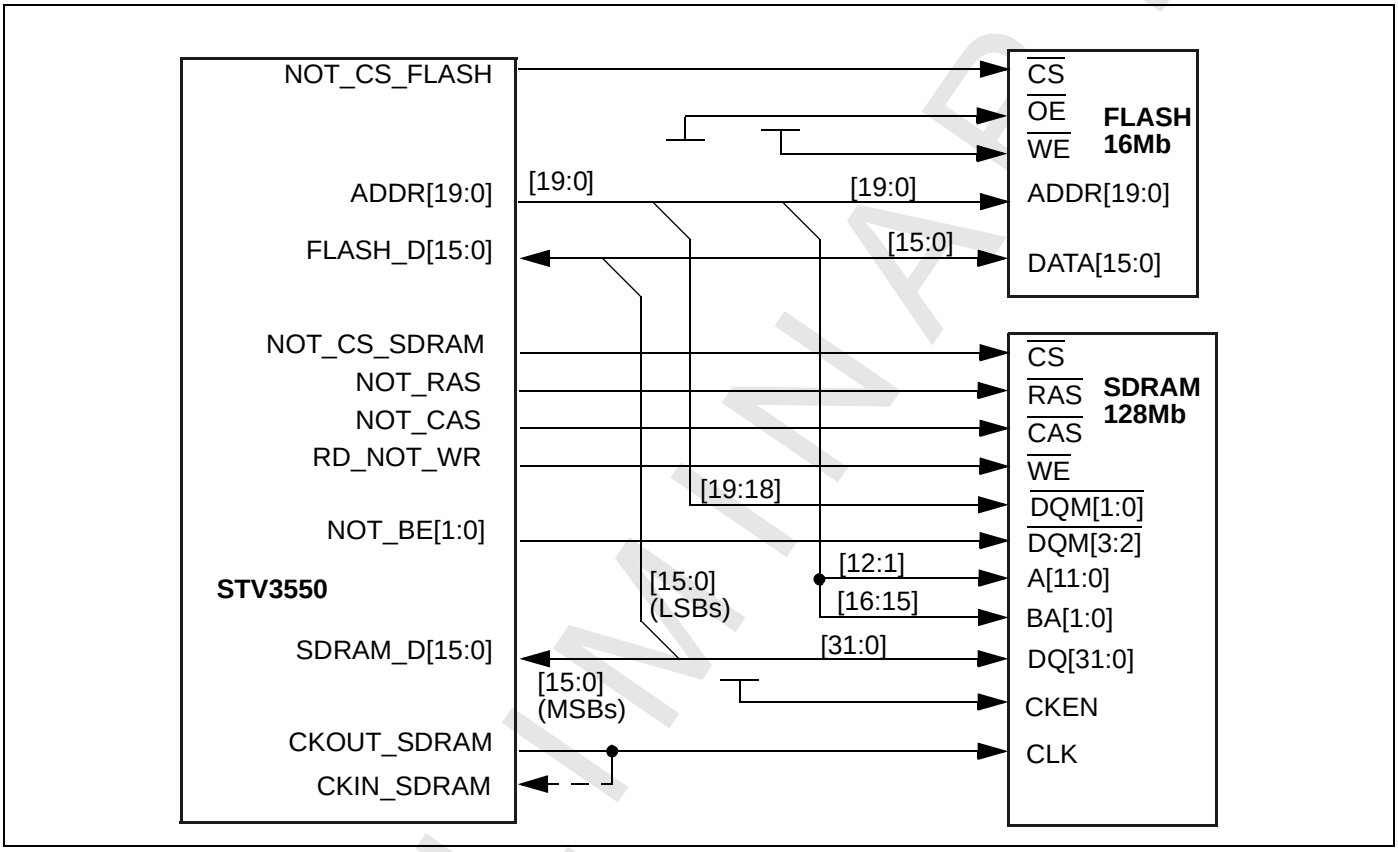
This configuration requires a SDRAM with 32 data lines. A typical size of the SDRAM may be 128 Mbits (16 MBytes).

Flash and SDRAM devices share the lower half of the data bus. As a result, placing memory devices on the board requires additional restrictions compared to the low-cost configuration.

Since the data word size is 32 bits, address outputs ADDR[12:1] are connected to the SDRAM address bus, and address outputs ADDR[16:15] are used as Bank Activate signals.

Note that outputs NOT_BE[1:0] are connected to $\overline{\text{DQM[3:2]}}$, while $\overline{\text{DQM[1:0]}}$ are provided through address outputs ADDR[19:18]. This means that internal Byte Enable 0 and 1 are provided through ADDR[19:18] and Byte Enable 2 and 3 are provided through NOT_BE[1:0]. The Flash is a Read-only device.

Figure 49: High-End Configuration using 1 SDRAM Device



6.3.11.3 High-End Configuration with two SDRAM Devices

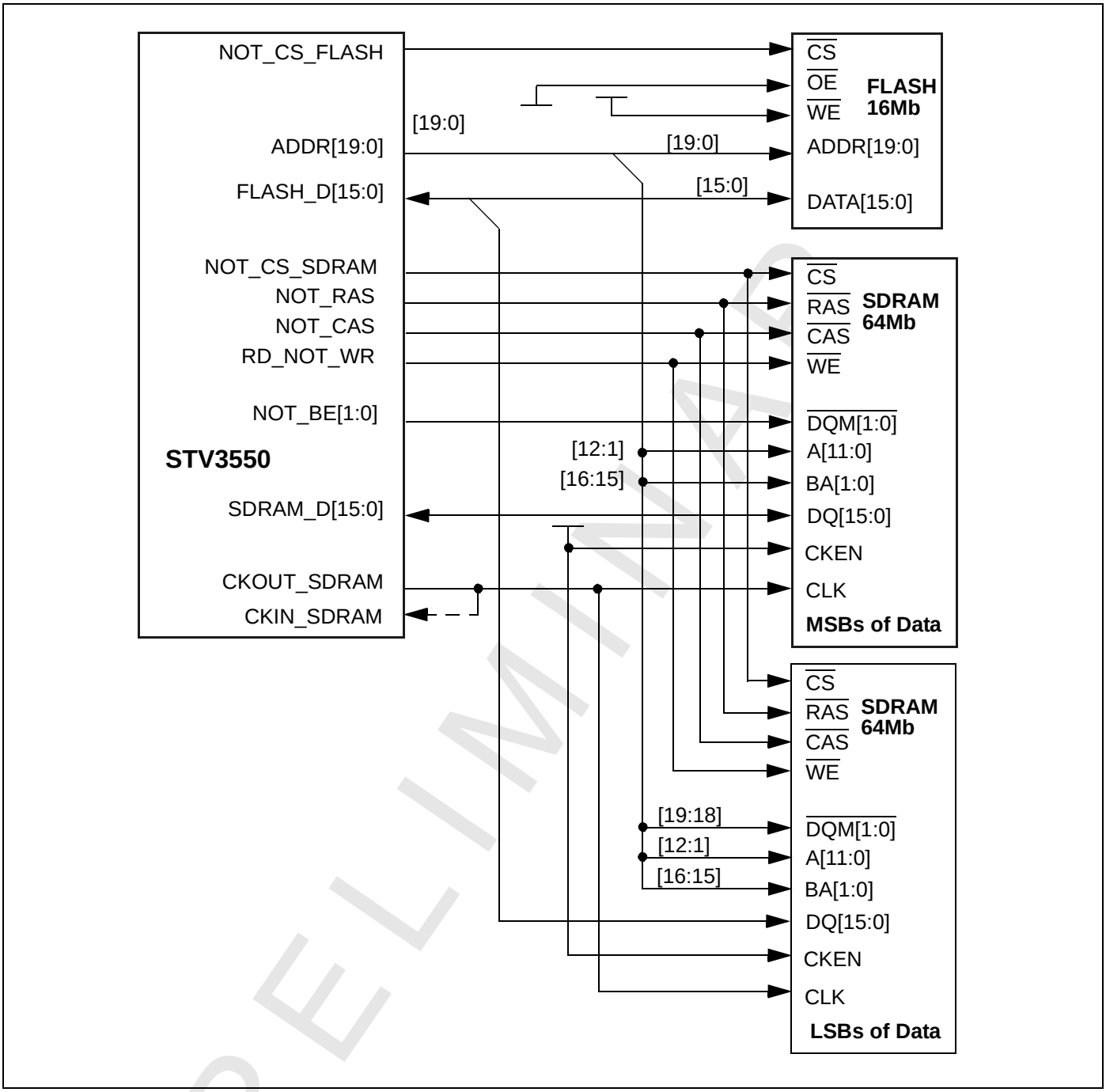
This configuration requires two SDRAM devices with 16 data lines each. A typical size of the SDRAMs may be 64 Mbits (8 MBytes).

One SDRAM device is connected to the upper half of the data bus, while the other device is connected to the lower half. The Flash memory shares the lower half of the data bus with one of the SDRAM devices.

From STV3550 point of view, since the data word size is 32 bits, address outputs ADDR[12:1] are connected to the SDRAM address buses, and address outputs ADDR[16:15] are used as Bank Activate signals.

Note that outputs NOT_BE[1:0] are connected to $\overline{\text{DQM[1:0]}}$ of the SDRAM connected to the upper half of the data bus, while address outputs ADDR[19:18] are driving $\overline{\text{DQM[1:0]}}$ of the other SDRAM. This means that internal Byte Enable 0 and 1 are provided through ADDR[19:18] and Byte Enable 2 and 3 are provided through NOT_BE[1:0]. The Flash is a Read-only device.

Figure 50: High-End Configuration using 2 SDRAM Devices



6.3.11.4 Mixed Configuration with two SDRAM Devices

This configuration is able to implement low-cost and high-end solutions on the same printed circuit board as described in previous sections.

This configuration is dedicated for the development of universal TV chassis boards that suit several types of hardware and software applications. It can also be used for a development board.

This configuration requires two SDRAM footprints with 16 data lines each.

One SDRAM device is connected to the upper half of the data bus, while the other device is connected to the lower half. The Flash memory shares the lower half of the data bus with one of the SDRAM devices.

When the board is dedicated to low-cost solutions, only one SDRAM (SDRAM 2) is soldered and the STV3550 Memory Interface is configured by software to only use one SDRAM.

When the board is dedicated to high-end solutions, both SDRAMs are soldered and the STV3550 Memory Interface is configured by software to use both SDRAMs.

Byte Enable signals are provided as described in previous sections, depending on the memory configuration.

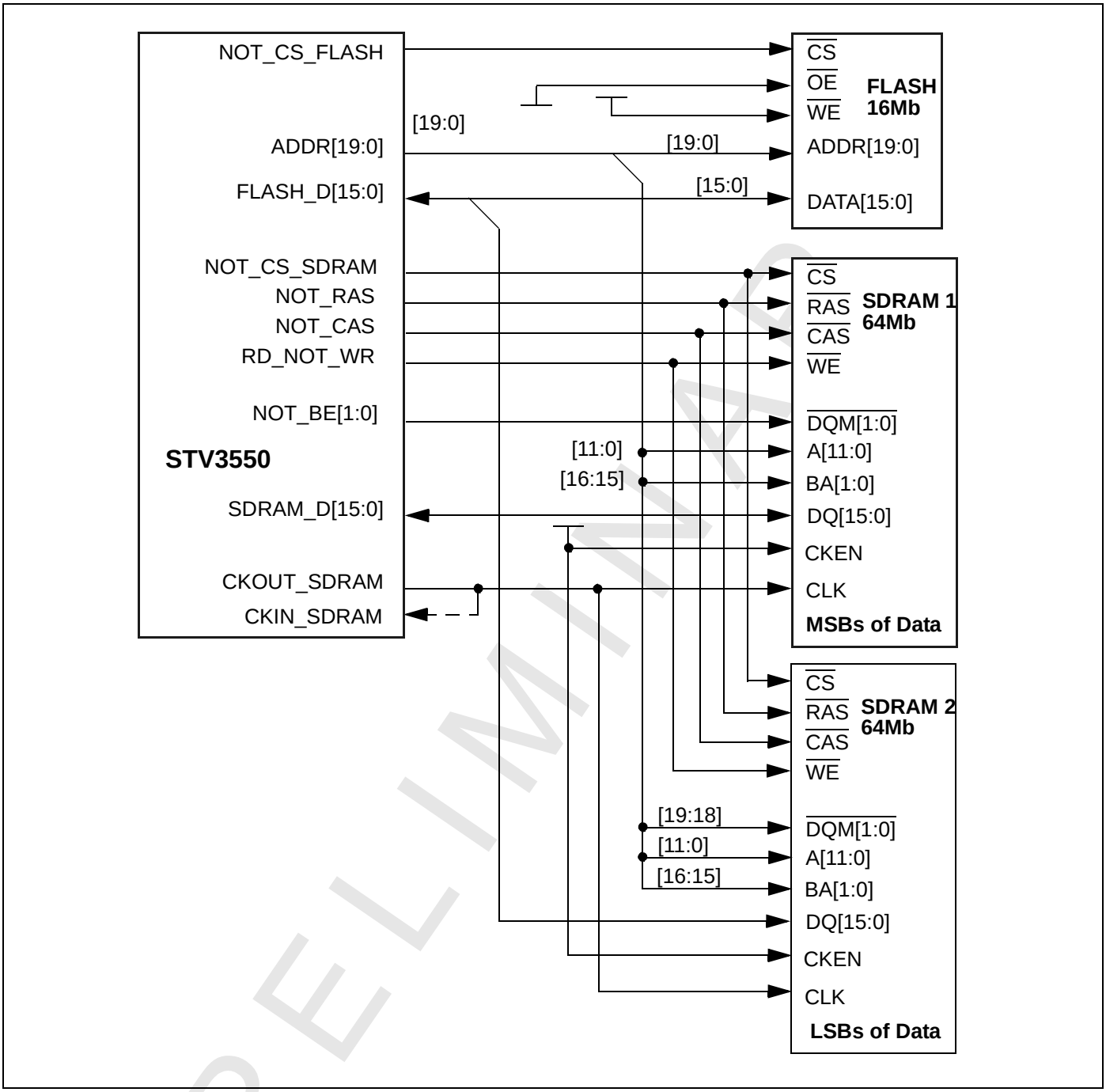
Note: It is recommended to maintain coherency between hardware and software configurations. It is especially recommended NOT to use a board equipped with two SDRAMs while the STV3550 Memory Interface is configured in a low-cost configuration as this may create bus contention.

The STV3550 Memory Interface includes an address translator which preventsany external modification the address bus towards the SDRAMs. Since the data word size differs from one configuration to the other one, *addresses ADDR[11:0]* are used in low-cost configurations, while addresses *ADDR[12:1]* are used in high-end configurations.

The Address bus is connected as in a low-cost configuration, while the address translator translates internal addresses ADDR[12:1] on external addresses ADDR[11:0] when a high-end configuration is required.

Note: Address translation only concerns addresses ADDR[11:0] and is only *performed when the SDRAM is accessed. Addresses are not translated when the Flash memory is accessed.*

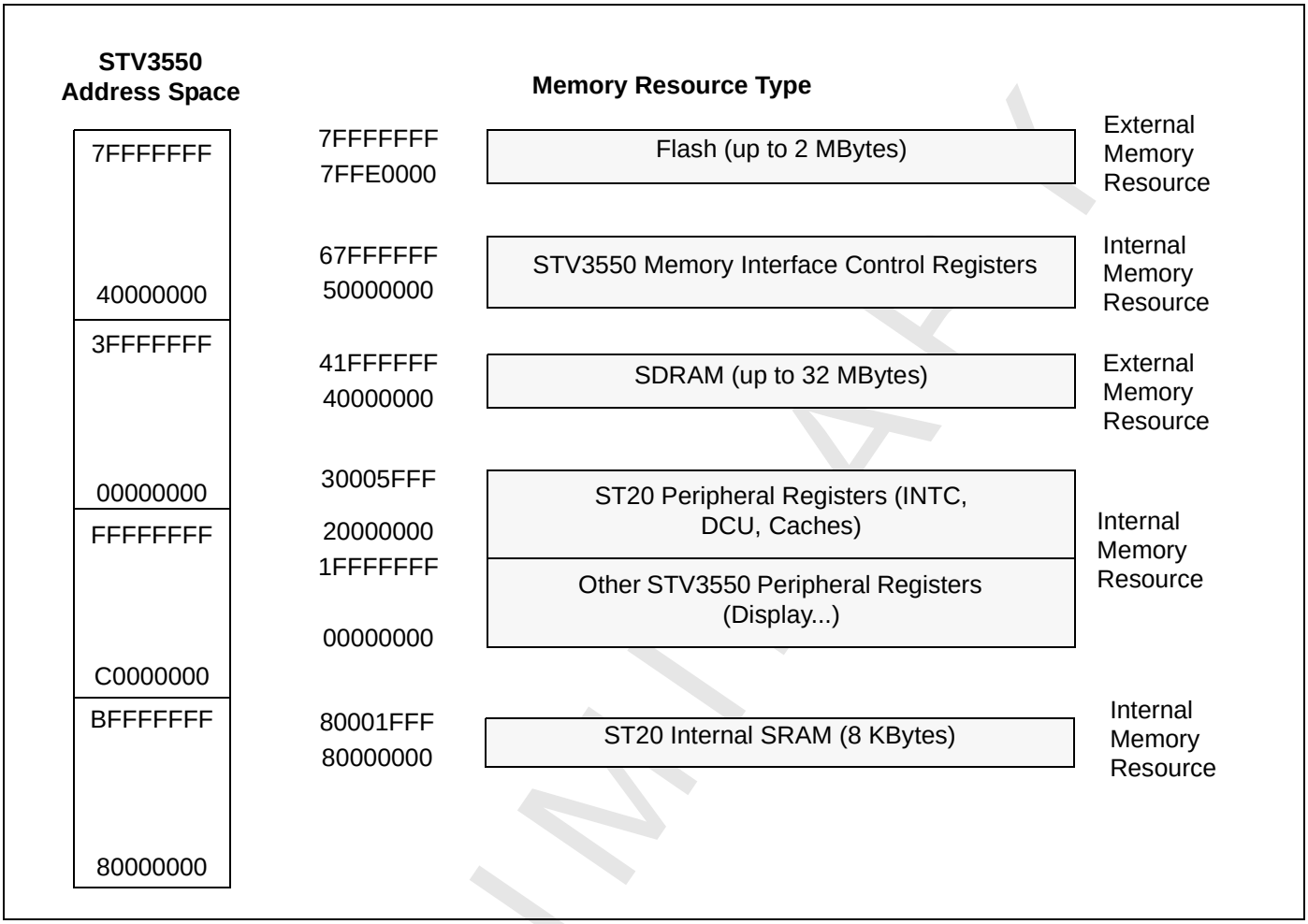
Figure 51: Mixed Configuration using 2 SDRAM Devices



6.3.12 STV3550 External and Internal Memory Mapping

The internal and external memory maps in the STV3550 is described in Figure 52.

Figure 52: Internal and External Memory Map



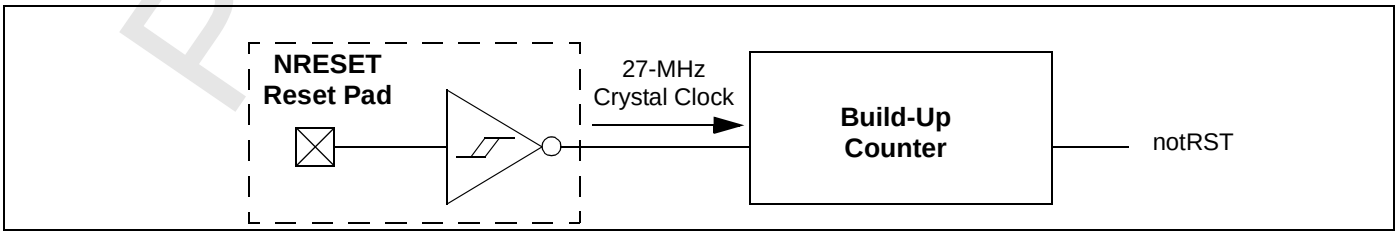
6.4 Reset Strategy

The chip includes 3 different types of reset:

1. An external Hard Reset controlled by the NRESET pin and usually used for the Power On Reset (POR)
2. An internal reset generated by the Watchdog
3. A Soft Reset controlled by the ST20 C2C200 CPU core

6.4.1 External Hard Reset

Figure 53: External Hard Reset



An external Hard Reset is triggered by asserting a low state on the NRESET pin of the STV3550. Assertion and de-assertion are considered as **asynchronous events**.

On a Power-On Reset (POR), the entire STV3550 is reset and the application is rebooted. All internal states are lost.

The reset is internally delayed by the Build-Up Counter (BUC) in order to wait for oscillator stabilization as well as to filter any glitches on the NRESET pin. The BUC then generates an asynchronous internal reset that is propagated to all blocks.

The asynchronous internal reset stops all the clocks that are not needed to boot the STV3550, while the system clock (for ST20 C2C200 CPU core, STV3550 Memory Interface, STBus Interconnect, etc.) is switched on the 27 MHz oscillator.

6.4.2 Internal Reset Generated by the Watchdog

When the Watchdog down-counter underflows, the Watchdog triggers a reset. During this reset, the Real Time Clock is partially reset (compare values are lost), but the time counter is not affected. Then the reset pulse acts as a reset caused by a POR for all other blocks.

Watchdog resets are flagged in the system in order to apply specific recovery algorithms.

6.4.3 Internal Soft Reset

Each block embeds an internal reset that is controlled by software. This reset behaves as a local hardware reset and is dedicated for software development step only.

6.5 Booting the STV3550

6.5.1 Typical Boot Sequence

A typical boot sequence aims at configuring all the hardware resources of the STV3550 prior to executing the main application program that manages the TV chassis.

The STV3550 always boots on the Flash device at address 0x7FFFFFFF. The STV3550 Memory Interface default configuration complies with most of the devices available on the Flash market, so the ST20 C2C200 CPU core is able to execute the first application tasks, which are:

- to speed-up the system clock from 27 MHz to the standard operating frequency by configuring the STV3550 Clock Generator,
- to configure the STV3550 Memory Interface to speed-up the Flash accesses and speed-up the software execution,
- to configure the STV3550 Memory Interface according to the specifications of the SDRAM device connected to the STV3550,
- to activate the instruction and data-cache capabilities of the ST20 C2C200 CPU.

Most applications require the activation of Watchdog capabilities as early as possible, so that this task can be part of the boot sequence as well.

The STV3550 Memory Interface must be configured in order to be able to access the SDRAM and execute the main application program. Otherwise, the internal 8-kB SRAM of the ST20 C2C200 CPU is large enough in order to execute the boot sequence which has a low complexity.

Note: The STV3550 Memory Interface should be configured only once, and it is the responsibility of the user to keep the STV3550 Memory Interface configuration unchanged in the main application program.

Depending on application requirements, tasks such as increasing the operating frequency may be delayed. Other high priority tasks such as TV chassis control can also be included in the boot sequence.

6.5.2 Starting The Main Application Program

Once hardware resources are properly configured, the main application program can be downloaded from Flash to SDRAM so that it can be executed into SDRAM with the best performances.

Main application programs include the OS/20 kernel, STV3550 video and peripheral resource management, etc., and is discussed in Software Application Notes.

6.6 Standby Mode

The chip includes a Standby mode used to put the TV chassis in standby. During this mode, the chip is supplied normally but only the sections used to restart the chip are clocked. The other clocks are switched off. The clocks used in Standby mode are generated from the 27-MHz crystal oscillator and divided by the Clock Generator block.

After a wake-up request from the IR receiver or from another interrupt source, the STV3550 restarts the CPU clock. Then, depending on software controls, the CPU continues the task where it has stopped and decodes the interrupt source.

Wake-up Interrupt sources can be:

- a command from the AD input level change
- a command from the IR detector
- a command from the Real Time Clock (RTC)
- 4 external interrupts

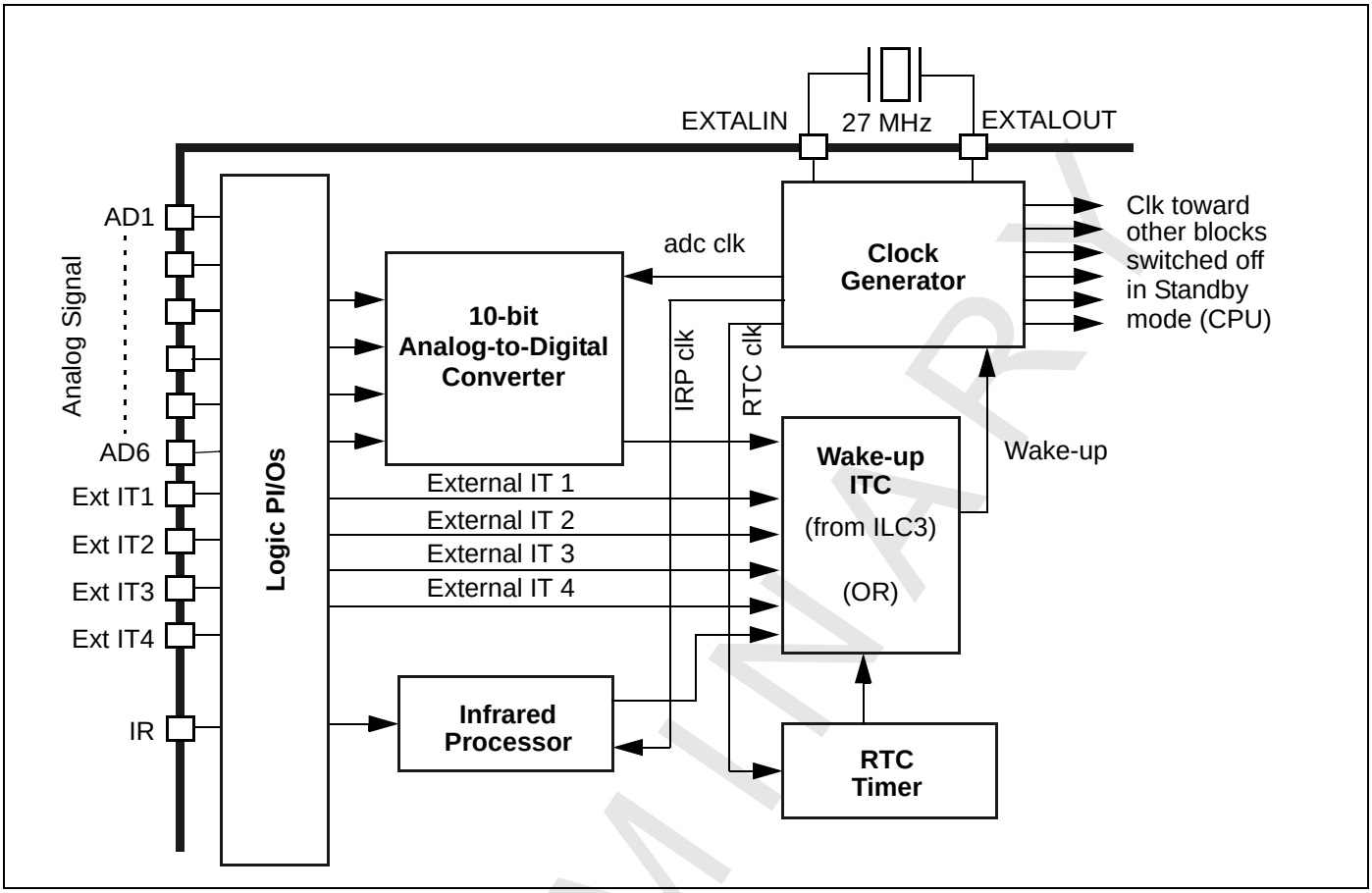
In Standby mode, only the following blocks are clocked:

- Infrared Preprocessor
- Real Time Clock (RTC)
- Analog-to-Digital Converter (ADC)

In addition to Standby mode, the STV3550 includes various “Low Power” modes.

These modes are performed by reducing the clock frequency for each block, or by switching off the blocks that are not in use (see Power Off bit or RST_soft reset bit for each block).

Figure 54: Example of Wake-up Interrupt Management



6.7 Interrupt Management

The ST20 core includes an Interrupt Controller (ITC) able to manage up to 16 interrupt levels. On top of this block, the STV3550 has an Interrupt Level Controller (ILC) which connects the 57 interrupt sources.

The interrupt sources can be on-chip or off-chip.They can be synchronous or asynchronous (the ILC re-synchronizes the asynchronous sources). Certain interrupt sources can also behave as wake-up sources used to exit Standby mode.

Table 21: Interrupt Sources (page 1 of 2)

Interrupt Source Name	Number of Interrupts	Number of Synchronous Interrupts	Number of Asynchronous Interrupts	Number of Wake-up Interrupts	Off-chip Interrupts
OSD	4	4			
Video Display Pipeline	5	5			
SDIN	19	10	9		
VTG	5		5		
Peripherals and I/Os	20	4	16	7	2
2DBM	2	2			

Table 21: Interrupt Sources (page 2 of 2)

Interrupt Source Name	Number of Interrupts	Number of Synchronous Interrupts	Number of Asynchronous Interrupts	Number of Wake-up Interrupts	Off-chip Interrupts
Line-locked PLL	1		1		
TOTAL	56	25	31	7	2

6.8 Clock Generator

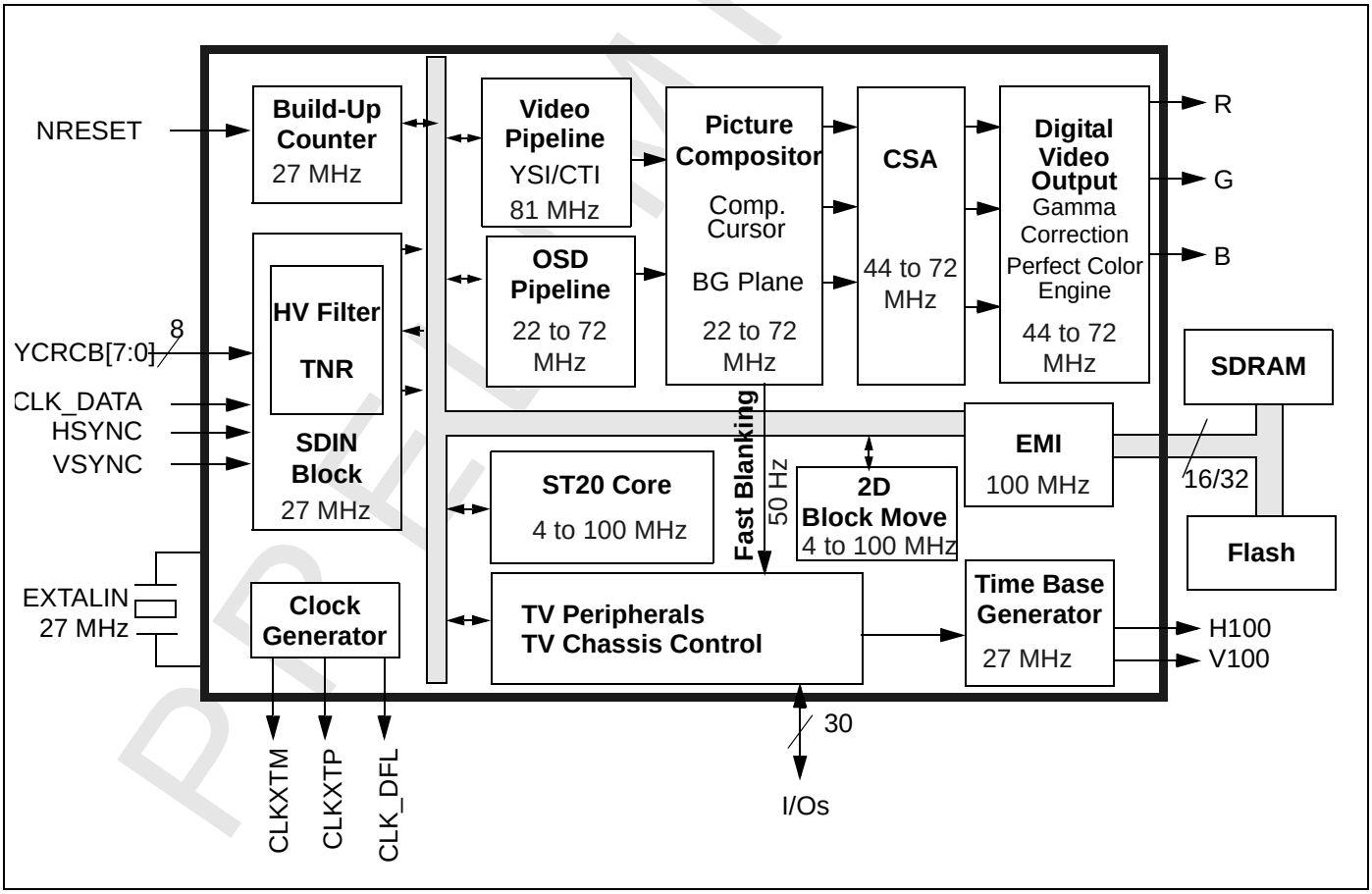
The chip includes different asynchronous clock domains. In general, each block connected to the Interconnect has two clock domains separated by a FIFO: the input or output clock domain and the Interconnect clock domain. The different clocks are generated by an on-chip clock synthesizer based on a 27 MHz reference clock. This can be the incoming D1 clock or a fixed external clock. All clocks are then referenced to the master clock. See Figure 55.

Note: 1 The pixel clock can be a free-running 22 to 72 MHz clock issued from a synthesizer. The pixel clock is obtained by a division of the DAC clock signal.

2 A second possibility is to have the pixel clock locked on the front-end clock (D1), through the synthesizer, used as PLL. The time constant is adjustable by programming.

3 All the other clocks are generated by a high frequency P/Q PLL, locked to the crystal oscillator.

Figure 55: Clock Frequency Domains



7 TV Chassis Control

The STV3550 includes the following peripherals for TV chassis control:

- 30 fully-programmable I/Os
- 4 external interrupts
- 8-bit programmable PWM with 4 inputs/outputs
- Infrared Digital Preprocessor
- Real Time Clock and Watchdog Timer
- 4 16-bit basic timers
- 10-bit ADC with 6 inputs and wake-up capability
- 2 Master/Slave I²C Bus Interface
- UART and support for IrDA interfaces

7.1 PWM and Counter Module

This module contains two separate functions:

- Pulse Width Modulation (PWM) decoder linked to 4 PWM outputs,
- a 32-bit counter with 4 input capture and compare capabilities.

7.1.1 External Interface

Table 22: Parallel Input/Output Pins

Pin No.	Pin Name	Main Function (after Reset)	Alternate Function
131	PORTA0	Port A0	PWMCAPTURE0/PWM0
133	PORTA1	Port A1	PWMCAPTURE1/PWM1
10	PORTA2	Port A2	PWMCAPTURE2/PWM2
11	PORTA3	Port A3	PWMCAPTURE3/PWM3

7.1.2 PWM Functions

The pulse width signal generator is used for the digital generation of up to 4 analog outputs when used with an external filtering network.

The unit, based on a unique 8-bit free-running counter clocked by the 27 MHz clock, can be prescaled by a 4-bit prescaler and provide frequencies from 6591.8 Hz up to 105468.75 Hz. Four 8-bit comparators are used to select 4 aspect ratios for the same PWM frequency. The period of the four 8-bit PWMs is therefore identical and synchronized.

The four PWM outputs must be configured as Alternate function output push-push or open drain.

An interrupt can be generated at the counter overflow.

When the PWM counter starts, the output is set to 1. When the counter matches the PWM compare value, the output is reset to 0. The output will return to 1 when the counter overflows.

7.1.3 Counter Functions

This cell is based on a 32-bit up-counter associated with a 5-bit prescaler that is clocked at 27 MHz divided by 4. There are:

- four 32-bit input capture registers linked to four capture inputs,
- four 32-bit compare registers.

The up-counter can be programmed to start at a specific value. At overflow, this counter is reloaded with the initial value.

When an external event is detected on the external PWM capture pin, the counter value is copied into the capture register. This event can be triggered by a rising edge, a falling edge or both. When this occurs, an interrupt is generated and the capture register value can read and its value reset.

The four 32-bit compare registers can be programmed to trigger an interrupt when the counter value is equal to one the compare register values. This function is used to create four additional timers.

Interrupts can be generated at any of the following events:

- at input capture event,
- at counter overflow,
- when the compare value matches the counter value.

Note: 1 As only a single interrupt is generated, the interrupt source must be identified by software.

2 The counter can be read anytime, but can only be written to when it is stopped. Compare registers can be read/written anytime, but capture registers are read only.

7.2 Infrared Receiver Preprocessor

The Infrared Receiver (IR) Preprocessor measures the interval between adjacent edges of the demodulated output signal from the IR amplifier/detector. Control bits determine whether the intervals of interest involve consecutive positive edges, negative edges, or any two edges. The measurement is represented in terms of 64.133 kHz clock cycles.

Whenever an edge of specified polarity is detected, the number of clock cycles counted since the previously detected edge is latched into the first of two 12-bit registers (IRP0 and IRP1) and an interrupt request is generated. When the interrupt is detected, the value of register IRP0 is read and this value used to calculate the interval between the two specified consecutive edges.

At the counter overflow, the counter value is latched immediately and an interrupt is generated.

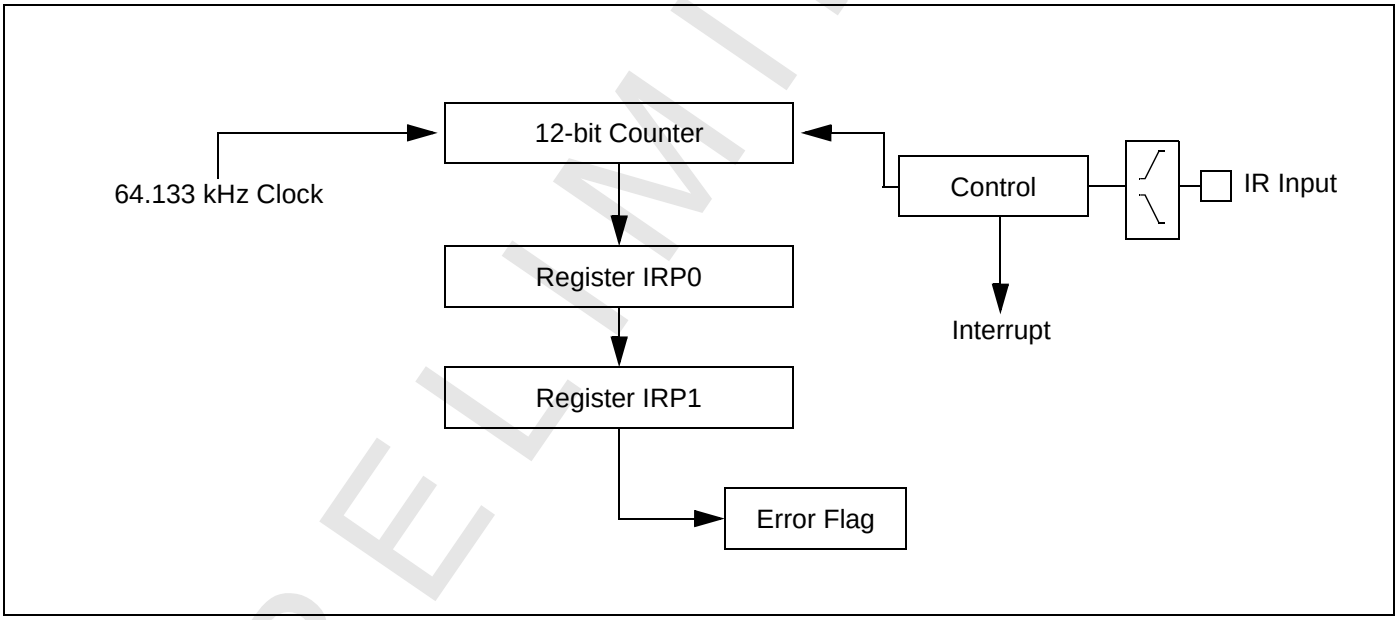
If a new edge of the specified polarity is detected or if the counter overflows and register IRP0 does not contain its maximum value before it is read, the value of register IRP0 is latched into register IRP1 if this register is empty. If register IRP1 is not empty, the new value is stored in register IRP0.

If a new edge is detected and if both registers (IRP0 and IRP1) are not empty, the new edge cannot be taken into account until both registers are reset. (The interval is not measured.)

An internal algorithm is used to reset the flags and register values.

The IR input signal is preprocessed by a spike filter that determines whether all pulses narrower than either 2 µs or 160 µs are filtered out before the signal is applied to the edge detector.

Figure 56: Infrared Receiver Diagram



7.3 Watchdog Timer (WDT)

The Watchdog Timer (WDT) provides a fail-safe mechanism used to reset the chip if the software fails to clear a counter within a given period.

The Watchdog Timer (WDT) has a counter which is clocked to provide up to a 2-second delay (11 bits). This counter is periodically cleared by software as described below. If the software fails to clear the counter within the timeout period, a “watchdog reset” signal is generated to reset the chip. The counter is in the CLK_RTC clock domain.

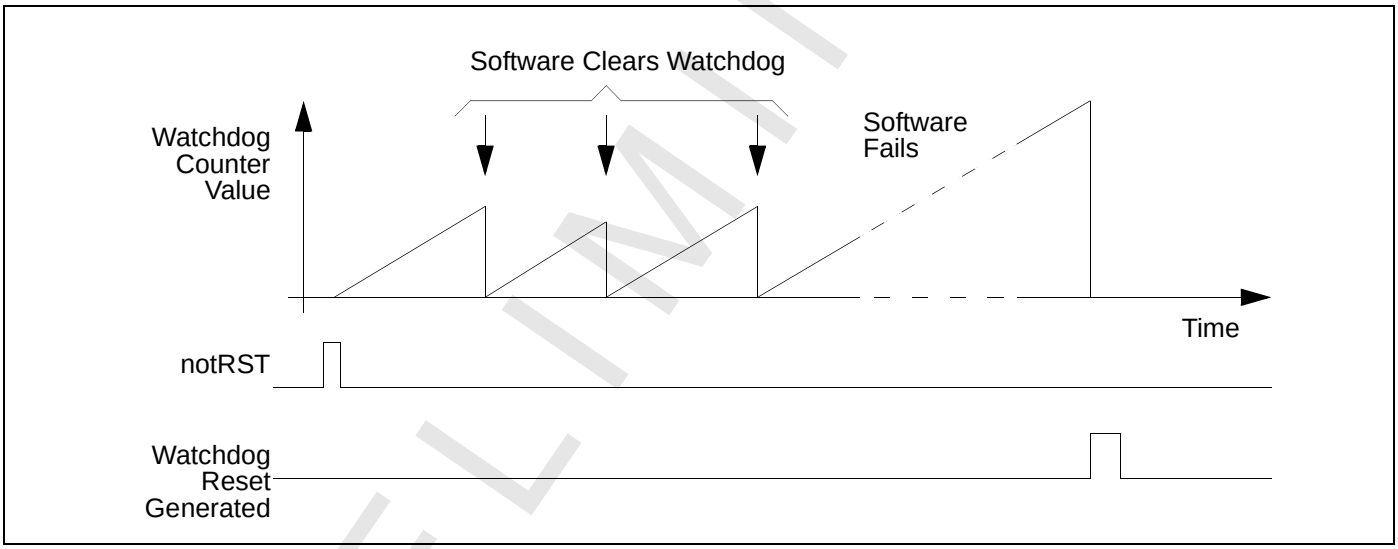
A timeout register is used to configure the watchdog delay. The transfer occurs at the next clear of the timer.

A status flag is set by a watchdog reset. This can be used to indicate to application code that the system was reset by the watchdog timer. This status bit is reset only by the external reset input to the chip.

The watchdog timer function is enabled by a control bit. If this bit is reset, a watchdog reset must never occur.

The watchdog counter can be also stopped by an internal signal. When this signal is high, the watchdog counter is stopped. This signal may be use by an emulator or a dedicated mode to stop the watchdog activity.

Figure 57: Watchdog Timer Behavior



Note: The counter is described as a counter that counts from zero up to the maximum allowed number.

7.3.1 Clearing the Counter

The watchdog counter is cleared under software control. To minimize the possibility of this happening erroneously, there are two registers which have to be written to in order to clear the counter. Each of these must be have the correct values (0xA and 0x5, respectively) written to them to clear the counter.

The command registers themselves are cleared by:

- a successful software clear of the counter,
- writing the incorrect value to either of the registers,
- triggering a watchdog reset.

7.3.2 Generation of Internal Watchdog Reset Signal

When the Watchdog Timer counter reaches the value of the timeout register, the Internal Watchdog Reset signal is asserted. This signal is also asserted asynchronously when the external reset input to the chip is low. The Internal Watchdog Reset signal is used to generate a reset signal to the rest of the chip.

When the counter is reset, its timeout is reset to 2s. A clear must be performed to set a new timeout.

7.4 Real Time Clock (RTC)

The Real Time Clock (RTC) provides a set of continuously running counters which can be used, with suitable software, to provide a clock/calendar function. The counter values can be written to set the current time/date data. The RTC is clocked by a 32.768 kHz crystal oscillator.

7.4.1 Real Time Clock (RTC)

The RTC contains two counters:

- 30-bit “milliseconds” counter,
- 16-bit “weeks” counter.

This allows large time values to be represented with high accuracy, without the complication of handling long (greater than 32 bit) integers in C.

Both counters increment at 1.024 kHz.

Because the RTC must run continuously, the counters are not reset by the software or Watchdog reset. Only a global external chip reset will reset the counters.

7.4.1.1 RTC Counters

Because the “milliseconds” counter increments at 1.024 kHz, the value does not actually represent milliseconds and must be taken into account by any concerned software applications. The milliseconds counter is modulo 1 week, or 619,315,200; i.e. 1024 (one second) x 60 (one minute) x 60 (one hour) x 24 (one day) x 7 (one week).

The “weeks” counter is incremented when the milliseconds counter wraps from 619,315,199 to 0. This is a 16-bit counter. The current value of both counters can be read at any time by the CPU.

7.4.1.2 RTC Alarm

The RTC can provide an interrupt to the CPU when the counter reaches a programmed value and the function is enabled.

Two dedicated registers, a 30-bit “milliseconds” register and a 16-bit “weeks” register can be programmed by software.

If the Alarm Enable bit is set and the RTC counter reaches (equals) the reference value, an interrupt is generated.

The alarm fuction is capable of waking up the MCU even in Eco Standby mode.

Application examples could be the automatic TV switch-off or wake-up functions.

7.5 Basic Timer

The chip embeds four Basic Timers. Each timer includes a programmable 16-bit down counter and an associated 16-bit prescaler with Single and Continuous counting modes capability. The input clock to the prescaler can be driven either by an internal clock equal (27 MHz) divided by 4 or by an external clock connected to the Timer Input pin. The maximum input clock frequency cannot exceed the internal clock frequency divided by 8. The input and output pins, when available, may be connected as Alternate Functions of an I/O port bit.

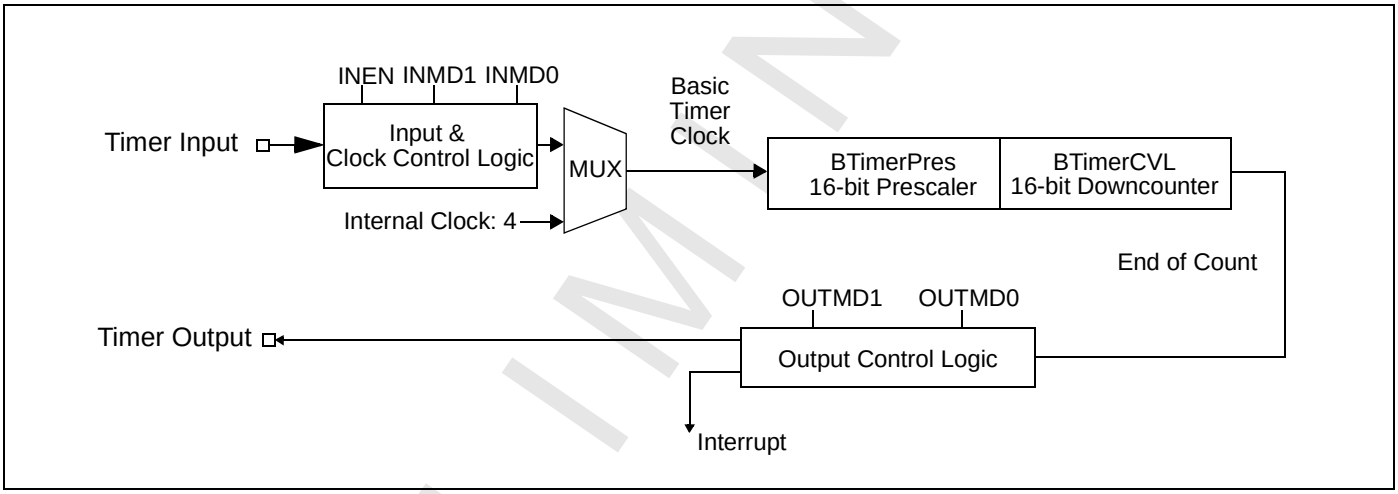
The Basic Timer can be used in one of four programmable input modes:

- event counter,
- gated external input mode,
- triggerable input mode,
- retriggerable input mode.

The Timer Output pin can be used to generate a Square Wave or Pulse Width Modulated signal.

The Basic Timer End Of Count condition is able to generate an interrupt. The End of Count condition is defined as the Counter Underflow, whenever 00h is reached.

Figure 58: Basic Timer Block Diagram



7.5.1 Functional Description

7.5.1.1 Timer/Counter Control

The timer can run in Continuous or Single mode. An initial pre-scale or counter value can be loaded.

IMPORTANT: In order to prevent incorrect counting of the Basic Timer, the prescaler and counter registers must be initialized before the timer is started. If this is not done, the counting will start with the reset values.

Single/Continuous Mode

In Single mode, at the End of Count, the Basic Timer stops, reloads the constant and resets the Start/Stop bit. (The user programmer can inspect the current timer status by reading this bit). Setting the Start/Stop bit will restart the counter.

In Continuous mode, at the End of the Count, the counter automatically reloads the constant and restarts. It is only stopped by resetting the Start/Stop bit.

7.5.1.2 Basic Timer Input Modes

Several inputs modes are available for the basic timers.

- **Event Counter Mode**

  The Basic Timer is driven by the signal applied to the input pin (Input Timer) which acts as an external clock. In this case, the unit works as an event counter. The event is a high to low transition on Input Timer. Spacing between trailing edges should be at least one internal clock cycle multiplied by 8 (i.e. the maximum Basic Timer input frequency is 3.38 MHz when the internal clock frequency is 27 MHz).

- **Gated Input Mode**

  The Basic Timer uses the internal clock and starts and stops the Timer according to the state of Input Timer pin. When the status of the Input Timer pin is High, the Basic Timer count operation proceeds. When the status of the Input Timer pin is Low, the counting is stopped.

- **Triggerable Input Mode**

  The Basic Timer is triggered only once by:

  ➢ enabling the timer,

  ➢ a high to low transition on the Input Timer pin.

  The timer must be reactivated once the counter reaches 0.

- **Retriggerable Input Mode**

  In this mode, each high to low transition on the Input Timer pin causes the counter to restart from the last constant loaded in the prescaler and load registers.

  As with Triggerable Input Mode, the Basic Timer is started by:

  ➢ enabling the timer,

  ➢ a high to low transition on the Input Timer pin.

7.5.1.3 Basic Timer Output Modes

Several output modes are available for the basic timers.

- **No Output Mode**

  The output is disabled and the corresponding pin is set high (reset state).

- **Square Wave Output Mode**

  The Basic Timer toggles the state of the Output Timer pin on every End Of Count condition. When the internal clock frequency is 27 MHz, square waves can be generated within a period ranging from 296 ns to 1271 seconds.

- **PWM Output Mode**

  The output value can be forced to high or low when the timer reaches End Of Count. This enables the user to generate PWM signals by modifying the value previously forced between End of Count events, based on software counters decremented on Basic Timer interrupts.

7.5.2 Interrupt Selection

The Basic Timer can generate an interrupt request at every End of Count event.

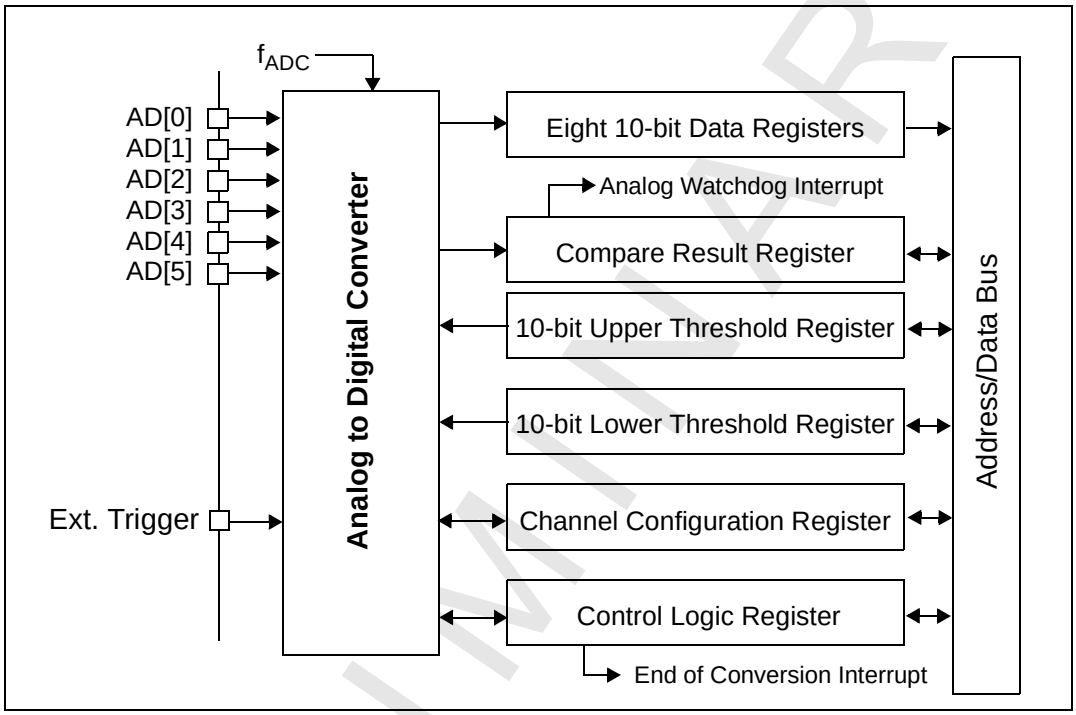
Note: If the counter is loaded with the value 0000h, the counting is stopped but the interrupt is continuously generated.

7.6 Analog to Digital Converter

7.6.1 Introduction

The on-chip Analog-to-Digital Converter (ADC) peripheral is a successive approximation converter used to convert analog voltage levels from up to 6 different sources at low frequency. When the requested conversion is completed, the result is stored in one of the eight 10-bit data registers and an End of Conversion interrupt is generated, if enabled by software. Another interrupt (Analog Watchdog) may be generated if the result of the conversion is higher or lower than a programmable reference level.

Figure 59: A/D Converter Diagram



7.6.2 Main Features

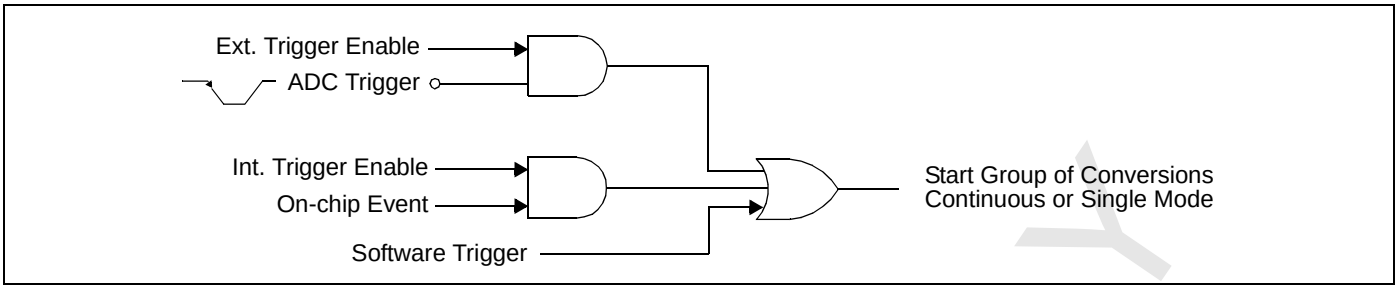
- 10-bit resolution with a guaranteed accuracy of 6 bits
- Up to 6 selectable analog inputs
- Single conversion time: 4.75 µs @ f_{ADC} = 3.4 MHz (27/8 MHz).
- Single / Continuous / Scan conversion modes
- External source trigger
- Analog watchdog on 2 threshold levels
- One interrupt for End of Conversion event or Analog Watchdog event.

7.6.3 General Description

The ADC can be used to convert the analog signals from a single input channel in 1-Channel mode or from up to six input channels in Scan mode.

Conversions can be started by software or by a falling edge on the external trigger pin.

Figure 60: A/D Converter Trigger Source



End of Conversion Interrupt

An interrupt can be generated at the end of each conversion.

Analog Watchdog Interrupt

Moreover, another interrupt can be generated based on the upper or lower threshold limits of the watchdog. See Section 7.6.4.

7.6.3.1 1-Channel mode

Only the selected analog input is converted. At the end of this conversion, the overflow status and result are stored in the control registers of the selected channel.

7.6.3.2 Scan Mode

Up to 6 channels can be sequentially converted starting with Channel 0 and ending with Channel 7. At the end of the conversion of each channel, the overflow status and result are stored in the control register of the corresponding channel.

7.6.3.3 Single / Continuous Mode

Each main mode can run in either Single or Continuous mode.

In Single mode, the conversion sequence for the analog channel input is performed one single time.

In Continuous mode, the conversion sequence is performed continuously until reset by software.

7.6.4 Analog Watchdog

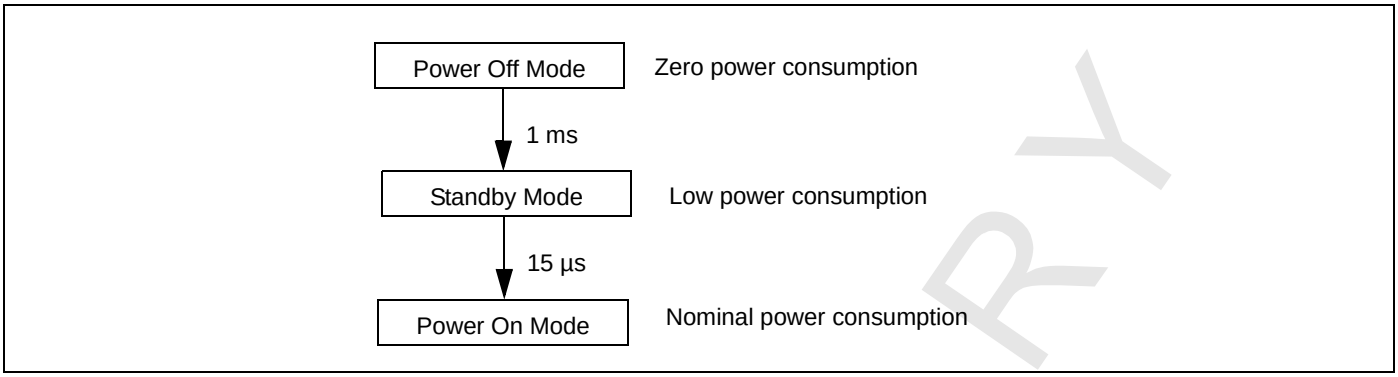
The Analog Watchdog is used to generate an interrupt based on two programmable threshold levels.

For each channel, when the level of the analog input exceeds upper and lower threshold values, an interrupt is generated.

7.6.5 Low Power Modes

There are two Low Power modes available in order to reduce power consumption. In Power Off mode there is no power consumption.

Figure 61: Low Power Modes



7.6.5.1 Power Off Mode

At reset, the ADC is in Power Off mode. In Power Off mode, the ADC consumes zero power. To switch into Power On mode, you must first pass through Standby mode and wait for a stabilization time-out of 1 ms.

7.6.5.2 Standby Mode

A Standby mode is available to reduce total chip power consumption while maintaining a rapid response time; i.e. to switch from Standby to Power On mode, a time-out of only 15 µs is required.

7.7 Inter Integrated Circuit Bus (I²C)

The SSC2 is a high-speed synchronous serial interface which can be used to interface to a wide variety of serial memories, remote control receivers, and other microcontrollers.

There are a number of serial interface standards for these. The SSC2 supports the I²C bus. The general programmable features should also allow it to interface to other serial bus standards.

The SSC2 supports full-duplex and half-duplex synchronous communication when used in conjunction with the PIO for output and input pad control.

It uses a 2 PIO pad interface: serial clock “SCLK”, serial data in/out “SDA”. These PIO pads are connected to the SSC2 clock/data interface pins in a configuration which allows their direction to be changed when in master or slave mode (see Section 7.7.3: PIO Pad Connection & Control).

The serial clock signal is either generated by the SSC2 (in master mode) or received from an external master (in slave mode). The output and input data is synchronized to the serial clock.

The following features are programmable: baud rate, data width, shift direction (heading control), clock polarity and clock phase.

The I²C features include: multi-master arbitration, acknowledge generation, start/stop condition generation/detection and clock stretching. These allow software to fully implement all aspects of the standard e.g. master and slave mode, multi-master mode, 10 bit addressing and fast mode.

7.7.1 Basic Features

Each STV3550 I2C cell has the following features:

- Possibility of swapping SCL and SDA lines,
- Selection one SCL, SDA couple between two choices,
- Each STV3550 I²C cell can be connected to four I²C buses.

7.7.2 Functional Description

The basic architecture of the SSC2 is shown in Figure 62.

Control of the direction, as an input, output or bi-directional, of the SCL and SDA pins is done in software by configuring the PIO pads.

The SSC2 itself has clock in/out and data in/out pins for connection to all 2 pads and a pin control block which selects the relevant data input and output according to the master or slave mode set (see Section 7.7.3: PIO Pad Connection & Control).

The serial_clock_out signal is programmable in master mode for baud rate, polarity and phase. This is described in Section 7.7.4: Clock Generation.

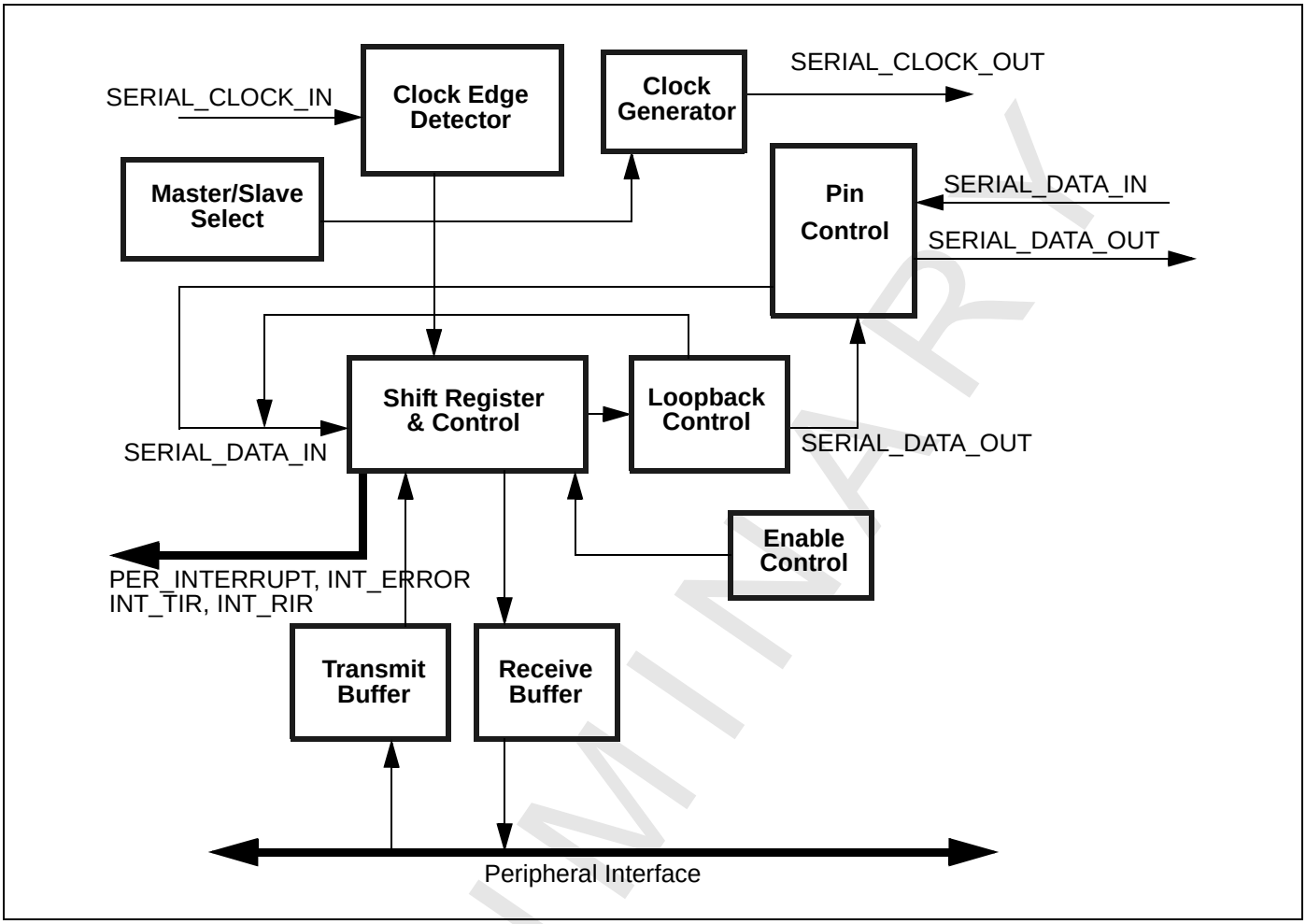
The SSC2 works by taking the data frame (2 to 16 bits) from a transmission buffer and placing it into a shift register. It then shifts the data at the serial_clock frequency out of the serial_data_out pin and synchronously shifts in data coming from the serial_data_in pin. The number of bits and the direction of shifting (MSB or LSB first) are programmable. This is described in Section 7.7.6: Shift Register.

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 7.7.9: Transmit & Receive Buffers.

The SSC2 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 SSC2 can also be configured to loop the serial_data_out back to serial_data_in in order to test the device without any external connections. This is described in Section 7.7.10: Loopback Mode.

Figure 62: Basic SSC2 Architecture



The SSC2 can be turned on and off by setting the enable control. This is described in Section 7.7.11: Enabling Operation.

The SSC2 can be set to operate as a bus master or as a bus slave device. This is described in Section 7.7.12: Master/Slave Operation.

The SSC2 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 7.7.13: Error Detection.

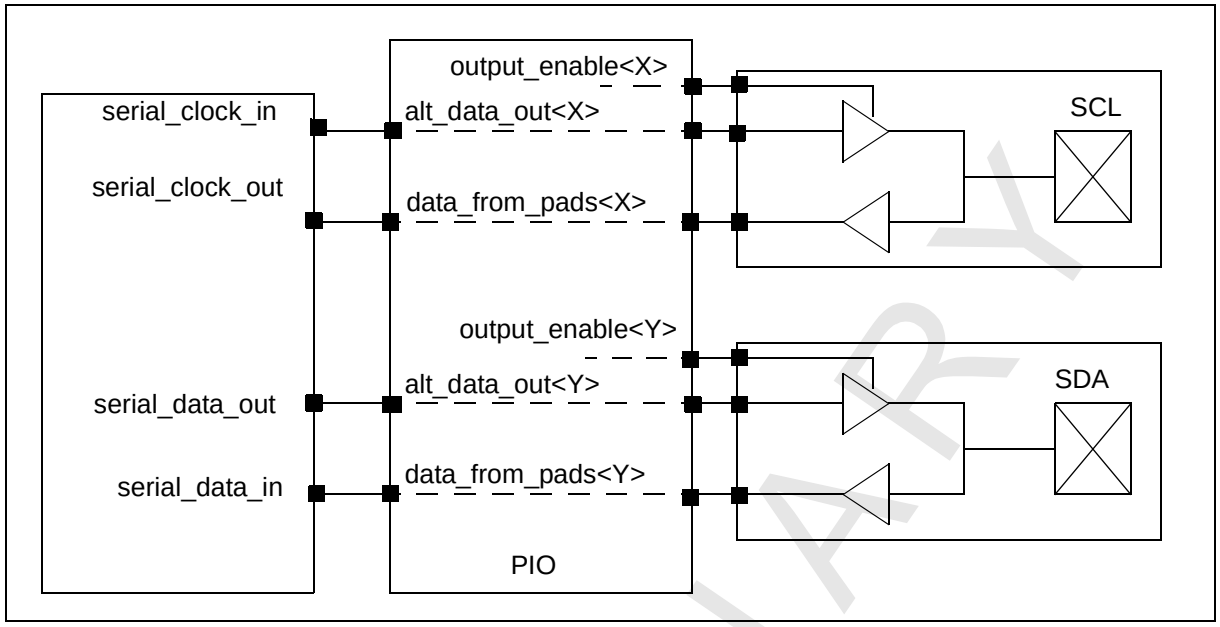
7.7.3 PIO Pad Connection & Control

In I²C mode, only the SDA pad will be used as an input and output.

The SDA and SCL pads are provided by 2 bits of a standard ST20 PIO block. Their directions (input, output or bidirectional) can therefore be configured in software using the appropriate PIO settings. Consequently the SSC2 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 SSC2 ports and the relevant PIO pads are shown in Figure 63.

Figure 63: PIO Pad Connections To SSC2



The pad control block inside the SSC2 determines which of the serial_data_in ports is used to read data from (depending on the master or slave mode). It also determines which of the serial_data_out ports to write data to (depending on the master or slave mode).

The deselected serial_data_out port is driven to ground (except in I²C mode when it is driven high). Therefore the user must ensure that the relevant PIO pad output enable is turned off depending on the master/slave status of the SSC.

It is up to the user to ensure that the PIO pads are configured correctly for direction and output driver type (e.g. push/pull or open drain).

Throughout the rest of this document the data in and out ports will be referred to as “serial_data_out” and “serial_data_in”, where this is assumed to be the correct pair of pins dependent on the master or slave mode of the SSC2.

7.7.4 Clock Generation

If the SSC2 is configured to be the bus master, then it will generate a serial clock signal on the serial_clock_out port.

The clock signal can be controlled for polarity and phase and its period (baud rate) can be set to a variety of frequencies.

For I²C operation there are also a number of additional clocking features. These are described in Section 7.7.16: I²C Operation.

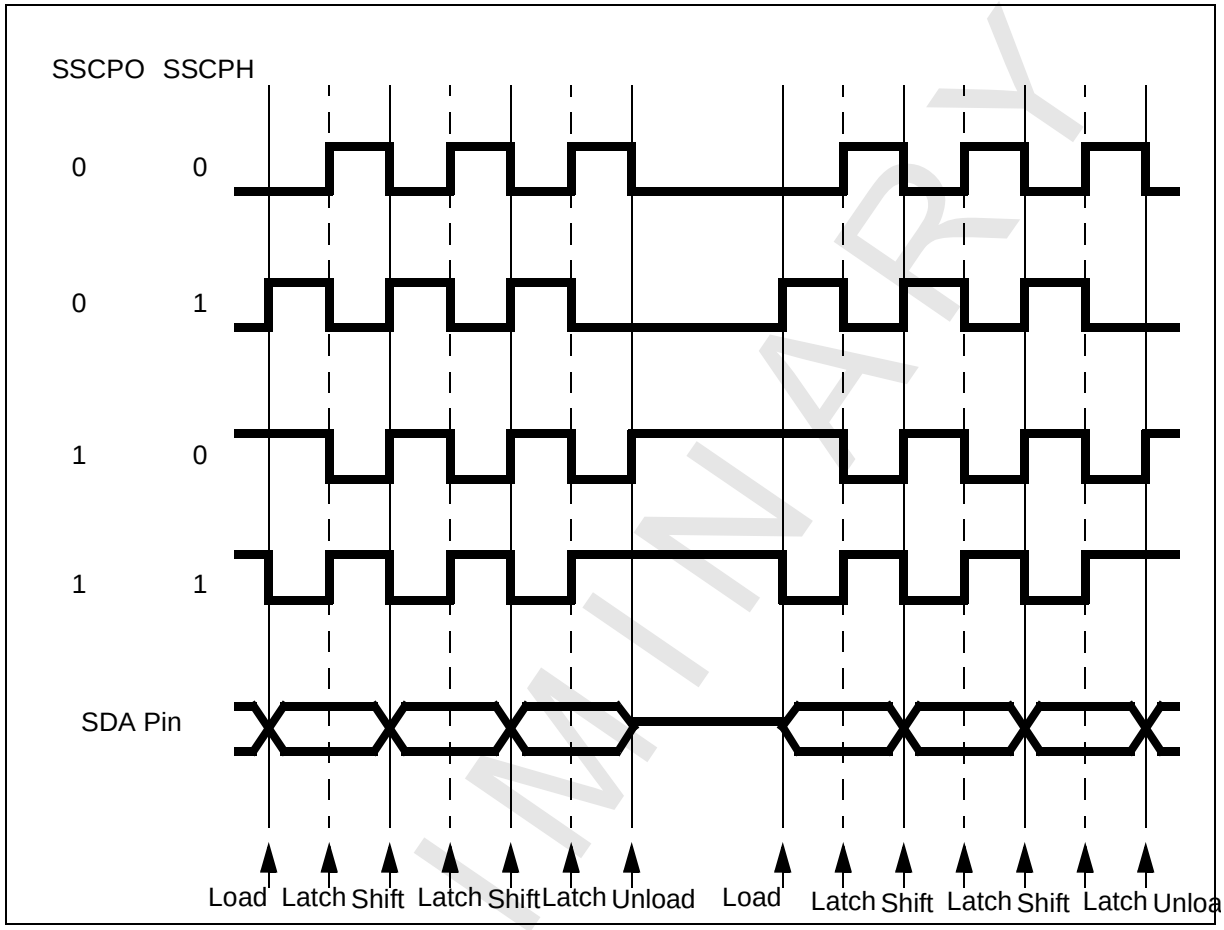
7.7.5 Clock Control

In master mode, the serial clock, SCL, is generated by the SSC2 according to the setting of the phase, SSCPH, and polarity, SSCPO bits in the control register.

The polarity bit, SSCPO, defines the logic level the clock idles at i.e. when the SSC2 is in master mode, but is between transactions. A polarity bit of 1 indicates an idle level of logic 1, 0 indicates idle of logic 0.

The phase bit, SSCPH, indicates whether a pulse is generated in the first or second half of the cycle. Note that this is a pulse relative to the idle state of the clock line i.e. if polarity is 0 then the pulse is positive going, if polarity is 1 then the pulse will be negative going. A phase setting of 0 causes the pulse to be in the second half of the cycle, a setting of 1 causes the pulse to occur in the first half of the cycle. The different combinations of polarity and phase are shown in Figure 64.

Figure 64: Clock Polarity And Phase Control



The SSC2 will always latch incoming data in the middle of the clock period at the point shown in the diagram. With the different combinations of polarity and phase it is possible to generate or not generate a clock pulse before the first data bit is latched.

Shifting out of data occurs at the end of the clock period. At the start of the first clock period the shift register is loaded. At the end of the last clock period, the shift register is unloaded into the receive buffer.

Baud Rate Generation

The SSC2 can generate a range of different baud rate clocks in master mode. These are setup by programming the baud rate generator register, SSCBRG.

Writing this register programs the baud rate as defined by the following formulas.

$$\text{Baudrate} = \frac{f_{CPU}}{2 \times \text{SSCBRG}} \qquad \text{SSCBRG} = \frac{f_{CPU}}{2 \times \text{Baudrate}}$$

Where SSCBRG represents the content of the reload register, as an unsigned 16 bit integer and f_{CPU} represents the CPU clock frequency.

At a CPU clock frequency of 40 MHz, the following baud rates are generated as shown in Table 23.

Table 23: Baud Rates and Bit Times for Various SSCBRG Reload Values

Baud Rate	Bit Time	Reload Value
Reserved. Use a reload value > 0	-	0000H
5 MBaud	200 ns	0004H
3.3 MBaud	300 ns	0006H
2.5 MBaud	400 ns	0008H
2.0 MBaud	500 ns	000AH
1.0 MBaud	1 us	0014H
100 KBaud	10 us	00C8H
10 KBaud	100 us	07D0H
1.0 KBaud	1 ms	4E20H

Note: The reload value must be greater than zero and it will also have a minimum value (i.e. maximum frequency) which is related to the CPU clock frequency. In normal mode this should be a minimum of 0x0008. In I²C mode it is the sum of the settings of the data_hold constant and the data_setup constant. i.e. 21 + 18 = 0x0027 at 60 MHz or 7 + 6 = 0x000D at 20 MHz (see Table 25*).*

The value in SSCBRG 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 SSCBRG number of CPU clock cycles.

Reading the SSCBRG register will return the current count value. This can be used to determine how far into each half cycle the counter is.

7.7.6 Shift Register

The shift register is loaded at the start of a data frame with the data in the transmit buffer. It then shifts data out of the serial_data_out port and data in from the serial_data_in port.

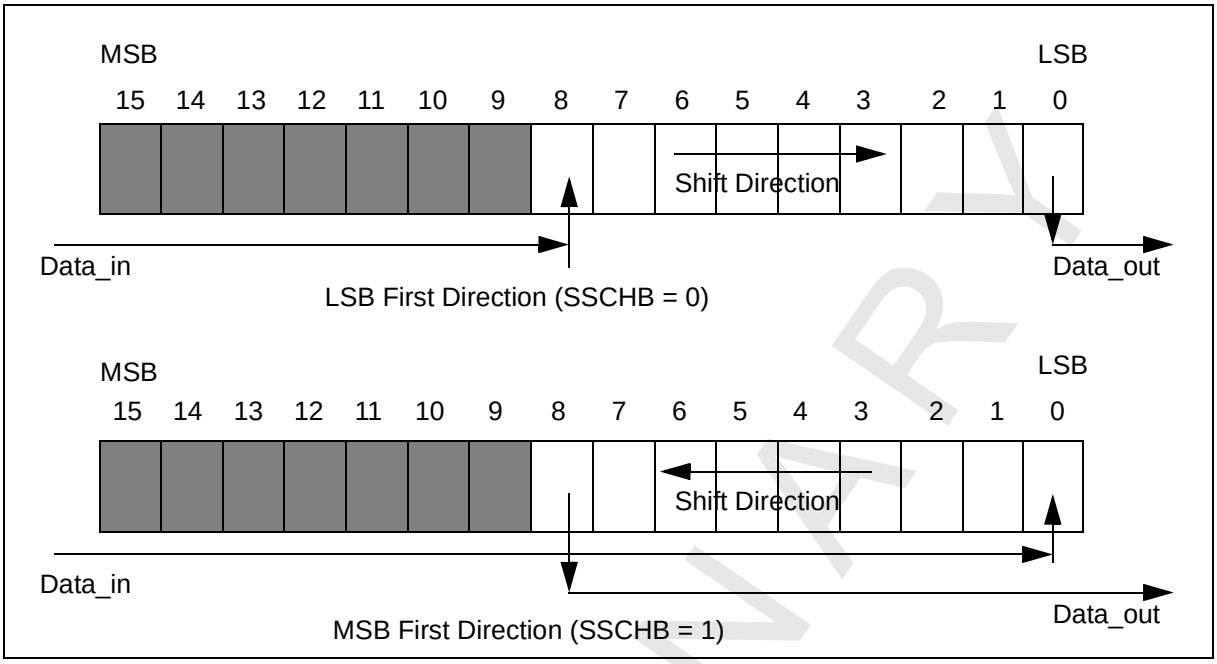
The shift register can shift out LSB first or MSB first. This is programmed by the heading control bit, SSCHB in the control register. A logic 1 indicates that the MSB will be shifted out first and a logic 0 that the LSB will shift first.

The width of a data frame is also programmable from 2 bits to 16 bits. This is set by the SSCBM bit field of the control register. A value of “0000” is not allowed, subsequent values set the bit width to the value plus one e.g. “0001” sets the frame width to 2 bits and “1111” sets it to 16 bits.

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 65.

Figure 65: Shift Register Operation For 9 Bit Data Frames



The shift register shifts at the end of each clock cycle. The clock pulse for shifting is presented to it from the serial clock input.

When a complete data frame has been shifted, the contents of the shift register (i.e. 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 7.7.16: I²C Operation.

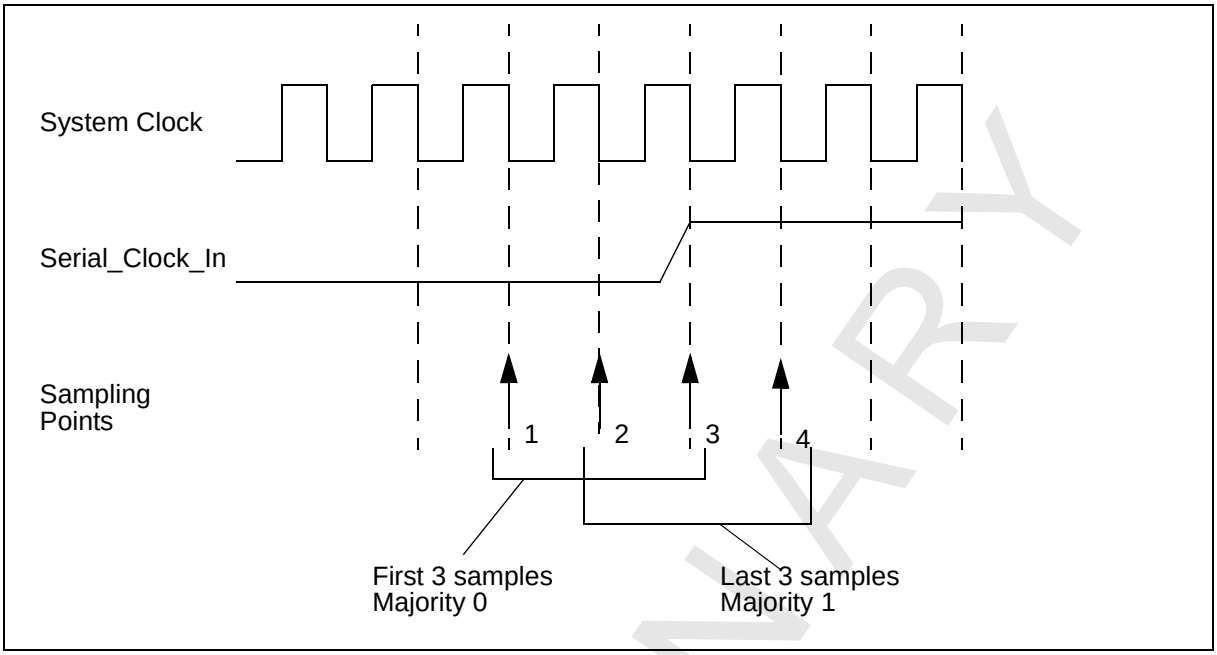
7.7.7 Clock Edge Detection

In order to generate the correct shifting and latching signals for the parts of the SSC2, correct detection of the input serial clock edges must occur. It is possible that the input clock edge rates are very slow (of the order of several hundred nano-seconds), hence it is essential that the edge detection can cope with noise spikes.

To achieve this, the SSC2 uses a majority sampling approach to detect the clock edges. Four consecutive samples are taken. The majority value of the first 3 samples is then taken to provide a previous signal value. The majority of the last 3 samples is used to provide a current signal value. If

the value has changed between the previous and current values, then an edge is detected. This is illustrated in Figure 66.

Figure 66: Clock Edge Detection (Rising Edge)



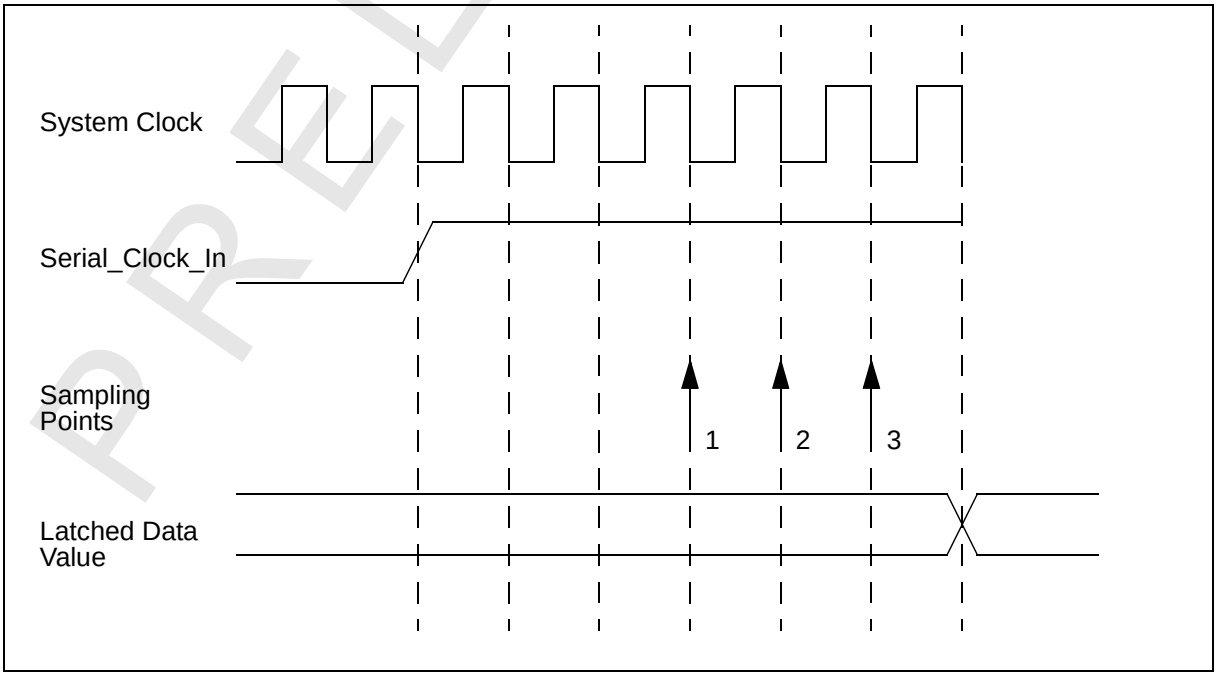
7.7.8 Receive Data Sampling

The data received by the SSC2 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 3 data samples at the 3rd, 4th and 5th system clock periods after the latching data edge. The data value is determined from the predominant data value in the 3 samples. This therefore gives an element of spike suppression.

The sampling process is shown in Figure 67.

Figure 67: Receive Data Sampling



7.7.9 Transmit & Receive Buffers

The transmit and receive buffers are used to allow the SSC2 to do back-to-back transfers (i.e. continuous clock and data transmission).

The transmit buffer (SSCTBUF), is written with the data to be sent out of the SSC2. This is loaded into the shift register for transmission. Once this has been done, the SSCTBUF is then 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 will cause an interrupt if the transmit buffer empty interrupt is enabled by setting the SSCTIEN bit in the interrupt enable register.

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, SSCTE (see Section 7.7.13: 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, SSCBM. The unused bits are ignored.

The receive buffer (SSCRBUF) 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, SSCRIR, which indicates that the receive buffer is full. This will cause an interrupt if the receive buffer full interrupt is enabled by setting the SSCRIEN 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 will be reloaded from the shift register over the top of the previous data. This will be indicated as a receive error condition, SSCRE, (see Section 7.7.13: Error Detection).

The number of bits which will be loaded into the receive buffer is determined by the frame data width selected in the control register, SSCBM. The unused bits are not valid and should be ignored.

7.7.10 Loopback Mode

A loopback mode is provided which connects the serial_data_out to serial_data_in. This allows software and testing to be done without the need for an external bus device. This mode is enabled by setting the SSCLPB bit in the control register. A setting of logic 1 enables loopback, logic 0 puts the SSC2 into normal operation.

7.7.11 Enabling Operation

The transmission and reception of data by the SSC2 block can be enabled or disabled by setting the SSCEN bit in the control register. A setting of logic 1 turns on the SSC2 block for transmission and reception. Logic 0 prevents the block from reading or writing data to the serial_data in and out ports.

7.7.12 Master/Slave Operation

A number of features of the SSC2 are controlled by whether the block is in master or slave mode. For example, in master mode the SSC2 will generate the serial_clock signal according to the setting of baud rate, polarity and phase. In slave mode, no clock is generated and instead the assumption is that an external device is generating the serial_clock.

Master or slave mode is set by the SSCMS bit in the control register. A setting of logic 0 means the SSC2 is in slave mode, a setting of logic 1 puts the device into master mode.

Note: The SSCMS bit is automatically cleared to 0 (slave mode) if the device loses arbitration in a multi-master environment.

7.7.13 Error Detection

A number of different error conditions can be detected by the SSC2. These are related to the mode of operation (master or slave, or both).

On detection of any of these error conditions a status flag will be set in the status register, SSCSTAT. Also, if the relevant enable bit is set in the interrupt enables register, SSCIEN, then an error interrupt will be generated from the SSC2.

The different error conditions are described below.

Transmit Error

A transmit error can only be generated in 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 SSC2.

In other words the error occurs when old transmit data is going to be transmitted from the slave. This could cause data corruption in the half-duplex open drain configuration.

The error condition is indicated by the setting of the SSCTE bit in the status register. An interrupt will be generated if the SSCTEEN 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 SSCRE bit in the status register. An interrupt will be generated if the SSCREEN 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.

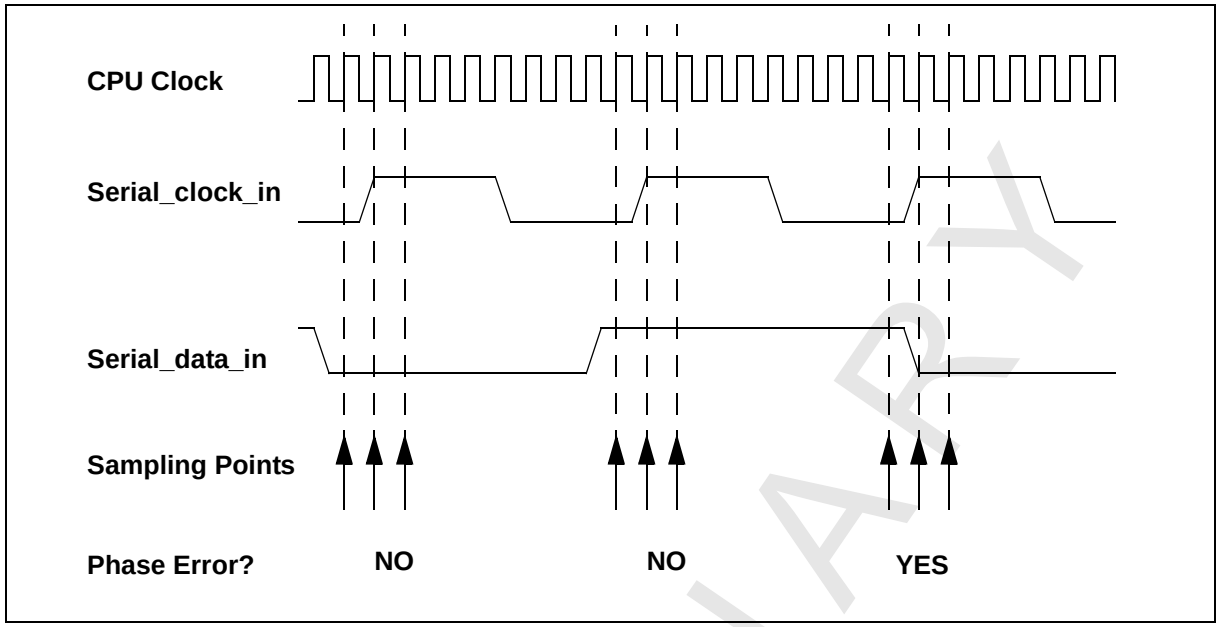
Phase Error

A phase error can be generated in master and slave modes. It indicates that the data received at the incoming data pin 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 CPU clock frequency. The sampling scheme is shown in Figure 68.

Figure 68: Phase Error Detection



The error condition is indicated by the setting of the SSCPE bit in the status register. An interrupt will be generated if the SSCPEEN 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.

7.7.14 Interrupt Mechanism

The SSC2 can generate a variety of different interrupts. They can all be enabled or disabled independently of each other. All the enabled interrupt conditions are OR-ed together to generate a global interrupt signal. In addition a dedicated interrupt signal is generated for transmit buffer empty and receive buffer full. An additional combined interrupt signal is generated for the error conditions:- receive error, transmit error, phase error and baud rate error.

To determine which interrupt condition occurred, a status register, SSCSTAT is provided which includes a bit for each condition. This is independent of the interrupt enables register, SSCIEN, which determines whether the condition asserts one or more of the interrupt signals.

7.7.15 Software Reset

A software reset facility is included in order to clear the device if various unexpected conditions occur. This is useful during software or hardware development as it allows the SSC2 to be reset without affecting the rest of the chip.

The reset is performed by setting the SSCSR bit in the SSCCON register. While the bit is set the entire SSC2 functionality will be held in the reset state. That is ALL functionality, except for the current settings of the SSCCON, SSCI2C, SSCTBUF, SSCSLAD, SSCBRG and SSCIEN registers. If these are required to be cleared, this can be done by programming these registers while the SSCSR bit is held high.

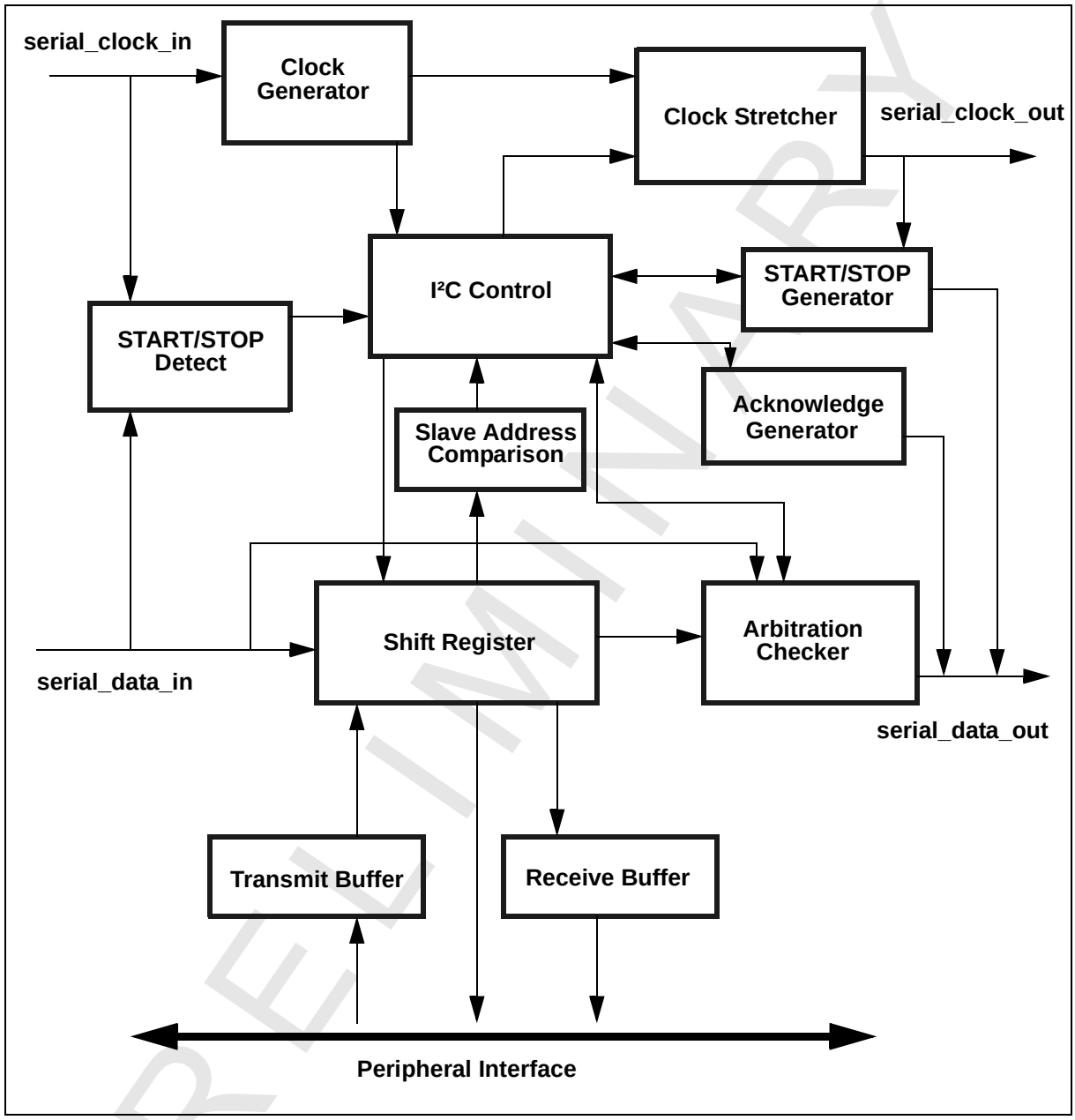
When the SSCSR bit is cleared the device will proceed according to the settings of the SSCCON and SSCI2C registers.

7.7.16 I²C Operation

This section describes the 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 69.

Figure 69: SSC2 Architecture with I²C Hardware



7.7.16.1 I²C Control

There are a number of features of the I²C-bus protocol which require special control:

1. 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.
2. 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.

3. 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 SSC2. If it is, then an acknowledge must be generated in the 9th bit position.

   Subsequently, an interrupt must be generated to inform the software that the SSC2 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 2 bytes. So in this mode, the SSC2 must compare 2 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, then 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.
4. In order for the software interrupt handler to have time to service interrupts, the SSC2 can hold the clock line LOW until the software releases it. This is called clock stretching.
5. In master mode, the SSC2 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 (i.e. a START condition has already been received).
6. When the SSC2 is receiving data from another device, it must generate acknowledge bits in the 9th 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 SSC2 should always acknowledge a matching address byte (i.e. the first byte after a START condition).
7. 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 (i.e. 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 SSC2 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, we are being addressed as a slave device. So the SSC2 must then go into slave mode and compare the address in the normal fashion (and generate an acknowledge if we are addressed).

   After the byte plus acknowledge the SSC2 must indicate to the software that we have lost arbitration by setting a flag.

All of these features are provided in the SSC2 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 is provided, SSCI2C.

To perform any of the I²C hardware features, the I²C control enable bit, SSCI2CM, must be set.

When the I²C control bit is set, the clock synchronization mechanism is always enabled (see Section 7.7.18: Clock Synchronization).

When the I²C control bit is set the START and STOP condition detection is performed.

To program the slave address of the SSC2 the slave address register, SSCSLAD 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 to. The SSC2 then uses this register to compare the slave address transmitted after a START condition (see Section 7.7.20: Slave Address Comparison).

To perform 10 bit address comparison and address acknowledge generation, the 10 bit addressing mode bit, SSCAD10 must be set (see Section 7.7.20: Slave Address Comparison).

The clock stretching mechanism is enabled for various interrupt conditions when the I²C control enable bit, SSCI2CM is set (see Section 7.7.21: Clock Stretching).

To generate a START condition, the I²C START condition generate bit, SSCSTRTG, must be set (see Section 7.7.22: START/STOP Condition Generation).

To generate a STOP condition, the I²C STOP condition generate bit, SSCSTOPG, must be set (see Section 7.7.22: START/STOP Condition Generation).

To generate acknowledge bits (i.e. a LOW data bit), after each 8 bit data byte when receiving data, the acknowledge generation bit, SSCACKG, 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 7.7.23: Acknowledge Bit Generation).

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, SSCSTAT. In addition, some of these bits can generate interrupts if corresponding bits are set in the interrupt enable register, SSCIEN.

To indicate that the SSC2 has been accessed as a slave device, the addressed as slave bit, SSCAAS, is set. This will also cause an interrupt if the SSCAASEN bit is set in the interrupt enable register.

The interrupt will occur after the SSC2 has generated the address acknowledge bit. In 10 bit addressing mode, the interrupt will occur after the second byte acknowledge bit, in the situations where 2 bytes of address are sent, or it will occur after the first byte acknowledge in the situation where only one byte is required.

Until the status bit is reset, the SSC2 will hold the clock line LOW (see Section 7.7.21: Clock Stretching). This forces the master device to wait until the software has processed the interrupt.

The status bit and the interrupt are reset by writing to the transmit buffer, SSCTBUF.

To indicate that a STOP condition has been received the STOP condition detected bit, SSCSTOP is set. This will also cause an interrupt if the SSCSTOPEN bit is set in the interrupt enable register.

The SSCSTOP interrupt and status bit will be reset by a read of the status register, SSCSTAT.

To indicate that the SSC2 has lost the arbitration process, when in a multi-master configuration, the arbitration lost bit, SSCARBL, is set. This will also result in an interrupt if the SSCARBLEN bit is set in the interrupt enable register. The interrupt will occur immediately after the arbitration is lost.

Until the status bit is reset, the SSC2 will hold the clock line LOW at the end of the current data frame, (see Section 7.7.21: Clock Stretching). This forces the winning master device to wait until the software has processed the interrupt.

The interrupt and status bit will be reset by a read of the status register, SSCSTAT.

To indicate that the I²C-bus is busy (i.e. between a START and a STOP condition), the I²C bus busy bit, SSCBUSY, is set. This does not generate an interrupt.

7.7.17 Clock Period In I²C Mode

The I²C standard requires that at 400KBits/s the clock low period is longer than the clock high period. This is performed by adding a constant (based in the system clock frequency) to the clock low period count and subtracting the count from the clock high period count. This mechanism is enabled for all frequencies when the SSCI2CM bit is set.

7.7.18 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 among all the devices is performed as follows.

All master devices start generating their low clock pulse when the external clock line goes low (this may or may not correspond with their own generated high to low transition).

They count out their low clock period and when finished they attempt to pull the clock line to high. However, if another master device is attempting to use a slower clock frequency, then it will be holding the clock line low, or if a slave device wants to, it can extend the clock period by deliberately holding the clock low.

Due to the open-drain output drive, the slower clock will win and the external clock line will remain low until this device has finished counting its slow clock pulse, or until the slave device is ready to proceed.

In the meantime, the faster master device will have detected a contradiction and will go into a wait state until the clock signal goes high again.

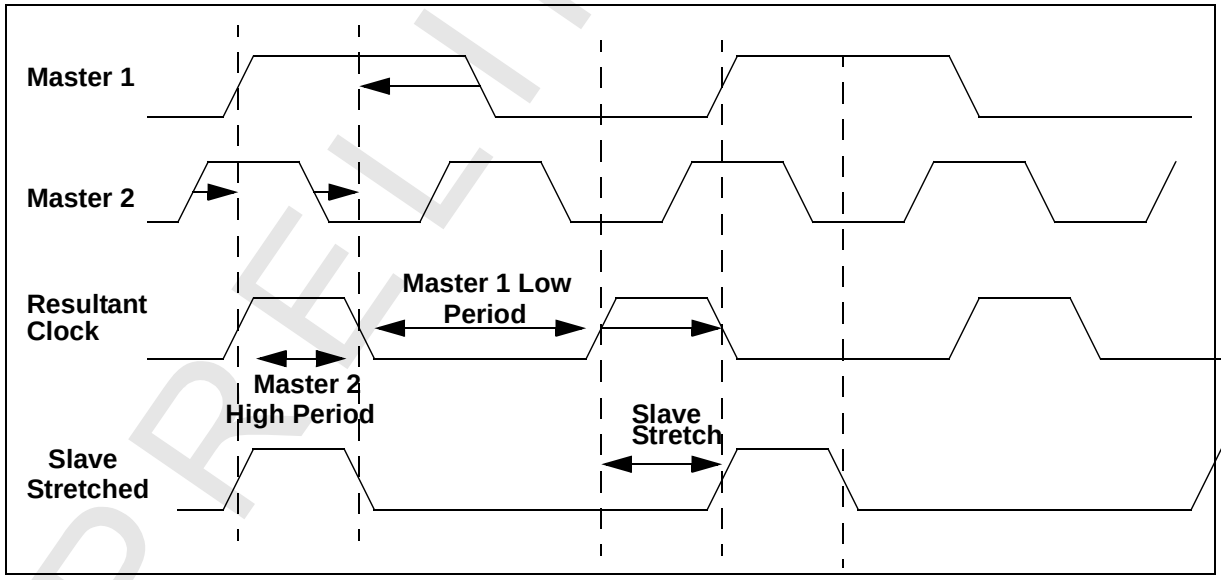
Once the external clock signal goes high, all the master devices will begin counting off their high clock pulse. In this case the first master to finish counting will attempt to pull the external clock line low and will win (because of the open drain line). The other master devices will detect this and will abort their high pulse count and switch to counting out their low clock pulse.

Consequently, the faster master device will determine the length of the high clock pulse, and the slowest master or slave device will determine 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 70.

Figure 70: Clock Synchronization Mechanism



The SSC2 implements this clock synchronization mechanism when the I²C control bit, SSCI2CM, is enabled.

7.7.19 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 detect when the bus is free.

A START condition is when the transmit/receive data line goes 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 will automatically get 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 7.7.24: Arbitration Checking).

A STOP condition is when the transmit/receive data line goes 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 will be free a specified time after the stop condition).

An additional piece of hardware is provided on the SSC2 to detect START and STOP conditions. This is necessary in slave mode because it cannot be done in time by programming the PIO pads. 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. The same edge detection mechanism as for the serial clock is used to determine if the data line changes. If a falling edge is seen then a START condition is detected. If a rising edge is seen then a STOP condition is detected. The sampling approach allows an element of spike suppression on the data line.

The START and STOP condition detection is enabled when the I²C control bit, SSCI2CM, is set in the I²C control register.

When a START condition is triggered, the SSC2 will inform the I²C control block which will then initiate the address comparison phase.

When a STOP condition is triggered the SSC2 will set the SSCSTOP bit in the status register. It will also generate an interrupt if the SSCSTOPEN bit is set in the interrupt enable register.

The interrupt and the status bit will be cleared when the status register is read.

7.7.20 Slave Address Comparison

After a START condition has been detected, the SSC2 goes into the address comparison phase.

It receives the first 8 bits of the next byte transmitted and compares the first 7 bits against the address stored in the slave address register, SSCSLAD. If they match, then the address comparison block indicates this to the I²C control block.

This will then generate an acknowledge bit in the next bit position and will set the addressed as slave bit, SSCAAS, in the status register. An interrupt will then be generated after the acknowledge bit if the addressed as slave enable bit, SSCAASEN, it set in the interrupt enables register. This interrupt and status bit will be cleared by a write to the transmit buffer.

The 8th bit of the first byte indicates whether the SSC2 will be written to (LOW) or read from (HIGH). This will be used by the control block to determine if it needs to acknowledge the following data bytes (i.e. when receiving data).

When 10 bit addressing mode is selected by setting the 10 bit addressing bit, SSCAD10, the first 7 bits of the first data byte will be compared against “11110XX”, where XX are the 2 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, then an acknowledge must be generated for the byte, but the addressed as slave status bit and interrupt are not yet asserted. Instead, the address comparator will wait for the next data byte and will compare this against the 8 least significant bits of the slave address register.

If this also matches, then the SSC2 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 had the read/write bit HIGH, then the SSC2 should only acknowledge if it has previously been addressed and a STOP condition has not yet occurred (i.e. the master has generated a repeated START condition). In this case the addressed as slave bit is set here, after the first byte plus acknowledge and an interrupt is generated if the interrupt enable is set. The second byte in this case will be sent by the SSC2 as this is a read operation.

In all cases if the address does not match, then the SSC2 ignores further data until a STOP condition is detected.

7.7.21 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 7.7.18: Clock Synchronization). The SSC2 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 following situations:-

1. When the SSC2 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 10 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.
2. When the SSC2 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.
3. When the SSC2 loses arbitration. The clock stretch occurs immediately after the current data byte and acknowledge have been performed. This gives the software time to abort its current transmission and prepare to retry after the next STOP condition.

Clock stretching is always cleared by a write to the transmit buffer. However, the clock stretch is only cleared after the required data setup time has expired, otherwise the new transmit buffer data may affect the value on the data line close to the clock edge.

If a clock stretching event occurs but no relevant interrupt is enabled then the clock will be stretched indefinitely until the next transmit buffer write. Hence it is important that the correct interrupts are always enabled. There is a clock stretch status bit, SSCCLST, which indicates when clock stretching is in operation. This will be cleared after the data setup time after the transmit buffer is written to.

7.7.22 START/STOP Condition Generation

As a master device the SSC2 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 holds the lines high between transactions.

The START/STOP generator is controlled by the START condition generate bit, SSCSTRTG and the STOP condition generate bit, SSCSTOPG in the I²C control register.

The generator will pull 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 will pull the data line HIGH.

However, a START condition will only be generated if the bus is currently free (i.e. the SSCBUSY bit in the status register is LOW). This is to prevent the SSC2 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 will force 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.

To properly generate the timing waveforms of the START and STOP conditions, the SSC2 contains a timing counter. This ensures the minimum setup and hold times are met with some additional margin.

7.7.23 Acknowledge Bit Generation

For I²C operation, it is required to both detect acknowledge bits when transmitting data, and 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 will not be 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 8 data bits have been transferred to it. In the 9th 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 will generate a stop condition to abort the transfer.

Acknowledge bits are generated by the SSC2 when the acknowledge generation bit, SSCACKG, is set in the I²C control register. They are only generated when receiving data.

When in master mode and receiving data the SSCACKG bit should be set to 0 before the last byte to be received.

The SSC2 will automatically generate acknowledge bits when addressed as a slave device.

7.7.24 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 who 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 8 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 valid data will have been received by the selected slave during the arbitration process.

It is possible 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 SSCARBL, arbitration lost error flag in the status register. An interrupt will also occur if the SSCARBLEN bit is set in the interrupt enables register.

Loss of arbitration also causes a clock stretch to be inserted. The interrupt and the clock stretch will occur immediately after the 8 bits plus acknowledge.

The arbitration lost status and interrupt will be cleared by a read of the status register.

The clock stretch is cleared when the software writes to the transmit buffer.

7.7.25 I²C Timing Specification

The I²C specification defines a number of timing constraints which must be met on the output clock and data pins. The key values which must be met are described in Table 24.

Table 24: Key I²C Timing Requirements

Parameter	MIN	MAX
SCL Clock Frequency		100KHZ or 400KHz[1]
Hold Time of SCL after SDA START or REPSTART Condition. After this time the first clock low pulse is created	1.0us or 0.6us[1]	
LOW Period of the SCL clock	4.7us or 1.3us[2]	
HIGH period of the SCL clock	4.0us or 0.6us[1]	
SDA Set-up time relative to SCL for a repeated START condition	4.6us or 0.6us[1]	
SDA Data Hold Time relative to SCL	300ns[2]	
SDA Data Setup Time relative to SCL	250ns or 100ns[3]	
Rise time of SDA and SCL lines	$20+0.1C_b$[4]	100ns or 300ns[1]
Fall time of SDA and SCL lines	$20+0.1C_b$[3]	300ns
SDA Setup time relative to SCL for a STOP condition	4.0us or 0.6us[1]	
Capacitive load of each bus line (C_b)		400pF

1. In FAST Mode
2. Actual requirement is 0ns but hold time must be at least 300ns on SDA line to comply with uncertain period of max fall time of 300ns on the SCL line.

3. In FAST mode, but when used with normal mode devices must still meet the 250ns minimum
4. Where C_b is the total capacitance on one bus line in pF

In order to meet these various timing requirements of the I²C standard a number of counters are included in the design. These are programmed to take a number of constants which will provide the times required.

The constants are defined for a particular system clock frequency to be sufficient to meet the minimum timing requirements (there are no maximum times defined for these timings). If the system clock frequency is higher or very much slower than the one used to define the constants then the values need to be re-defined.

The constants used for the current implementation are shown in Table 23.

Table 25: I²C Timing Constants

I²C Timing Name	I²C Standard Minimum Time	Constant Value (cycles)	Time Produced At 60 MHz
Data Hold Time	300ns	21	350ns
Setup Time For STOP Condition $T_{SU.STO}$	4.0us	300	5.0us
Setup Time For START (repeated) Condition $T_{SU.ST}$	4.7us	300	5.0us
Hold Time (repeated) START Condition $T_{HD:STA}$	4.0us	300	5.0us
Data Setup Time $T_{SU:DAT}$	250ns	21	300ns

7.8 Asynchronous Serial Controller (ASC)

The Asynchronous Serial Controller, also referred to as the UART interface, provides serial communication between the STV3550 and other microcontrollers, microprocessors or external peripherals.

Eight or nine bit data transfer, parity generation, and the number of stop bits are programmable. Parity, framing, and overrun error detection is provided to increase the reliability of data transfers. 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 baud rate generator provides the ASC with a separate serial clock signal.

The ASC supports full-duplex asynchronous communication, where both the transmitter and the receiver use the same data frame format and the same baud rate. Data is transmitted on the transmit data output pin **TxD** and received on the receive data input pin **RxD**.

The registers for the ASC are grouped in a 4 Kbyte block, with the base of the block for ASC number *n* at the address *ASCnBaseAddress*. The value of each *ASCnBaseAddress* is given in the register information at the back of this chapter.

7.8.1 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 baud rate generator run bit (**Run**) is set to 1. When the **Run** bit is set to 0, **TxD** will be 1. Setting the **Run** bit to 0 will immediately freeze 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 may result in unpredictable behavior.

The ASC can be set to use either double-buffering or a 16-deep FIFO on transmission and reception.

7.8.1.1 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.

7.8.1.2 Transmission and Reception

Serial data transmission or reception is only possible when the baud rate 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**. Because data transmission is double-buffered or uses a FIFO (selectable in the **ASC**_*n*_**Control** register), 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 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** will be set when the receive buffer register has not been read by the time reception

of a second character is complete. 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) 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.

7.8.2 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 the setting of the **Mode** bit field in the control register.

The transmitted data frame consists of three basic elements:

- The start bit
- The data field (8 or 9 bits, least significant bit (LSB) first, including a parity bit or wake-up bit, if selected)
- The stop bits (0.5, 1, 1.5 or 2 stop bits)

7.8.2.1 8-bit Data Frames

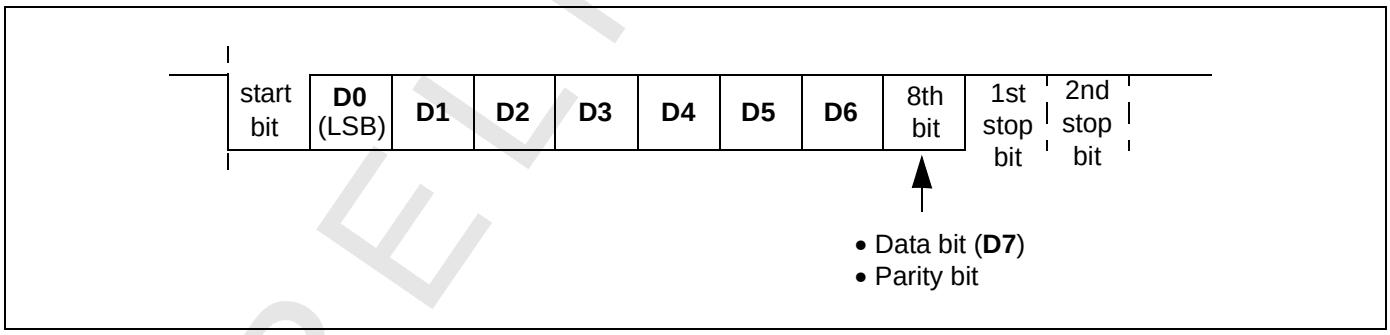
Figure 71 illustrates a 8-bit transmitted data frame. 8-bit frames may use of one of the following formats:

- Eight data bits **D0-7** (**Mode** set to 001)
- Seven data bits **D0-6** 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 will be set and the odd parity bit will be cleared.

In receive mode the parity error flag (**ParityError**) will be 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 71: 8-bit Tx Data Frame Format

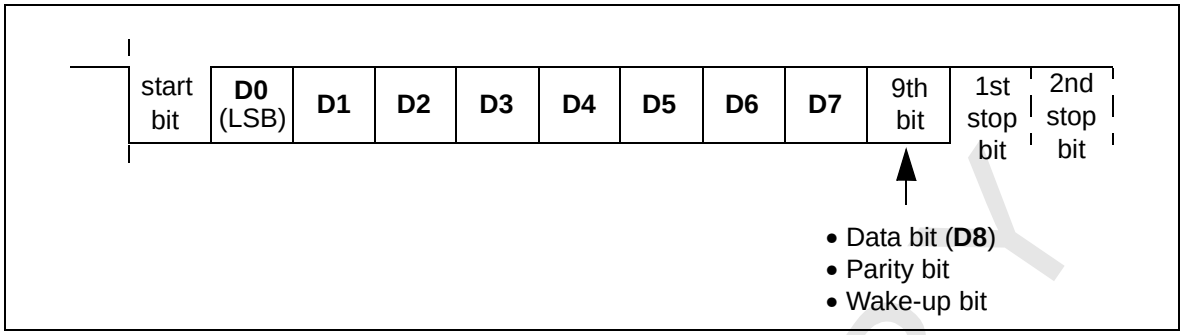


7.8.2.2 9-bit Data Frames

Figure 72 illustrates a 9-bit transmitted data frame. 9-bit data frames use of one of the following formats:

- Nine data bits **D0-8** (**Mode** set to 100)
- Eight data bits **D0-7** plus an automatically generated parity bit (**Mode** set to 111)
- Eight data bits **D0-7** plus a wake-up bit (**Mode** set to 101)

Figure 72: 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 will be set and the odd parity bit will be cleared. The parity error flag (**ParityError**) will be 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 request will be activated and no data will be transferred.

This feature may be used to control communication in multi-processor 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 a 1 for an address byte and a 0 for a data byte, so no slave will be interrupted by a data byte. An address byte will interrupt all slaves (operating in *8-bit data plus wake-up bit* mode), so each slave can examine the 8 least significant bits (LSBs) of the received character, which is the address. The addressed slave will switch to 9-bit data mode, which enables it to receive the data bytes that will be 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.

7.8.3 Transmission

Transmission begins at the next baud rate clock tick, provided that the **Run** bit is set and data has been loaded into the **ASC**_*n*_**TxBuffer**. If the **CTSEnable** bit is set in the **ASC**_*n*_**Control** register then transmission only occurs when **CTS** is high.

The transmitter empty flag (**TxEmpty**) indicates whether the output shift register is empty. It will be set at the beginning of the last data frame bit that is transmitted, i.e. during the first system 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.

7.8.3.1 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, i.e. when it contains 16 characters. Further writes to **ASC**_*n*_**TxBuffer** will 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 8 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 (i.e. the output shift register is empty) and something is written to the **ASC**_*n*_**TxBuffer** so that the output FIFO becomes non-empty, 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 baud rate tick.

When the transmitter is just about to transmit the stop bits, and if the output FIFO is non-empty, the output shift register will be immediately loaded from the output FIFO, and the transmission of this new data will begin as soon as the current stop bit period is over (i.e. the next start bit will be transmitted immediately following the current stop bit period). If the output FIFO is empty at this point, the output shift register will become empty. Thus back-to-back transmission of data can take place. If the output FIFO is empty at this point, the output shift register will become empty. 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.

7.8.3.2 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 will overwrite the contents. The **TxHalfEmpty** bit of the **ASC**_*n*_**Status** register is set when the output buffer is empty.

7.8.4 Reception

Reception is initiated by a falling edge on the data input pin **RxD**, provided that the **Run** and **RxEnable** bits of the **ASC**_*n*_**Control** register are set.

Controlled data transfer can be achieved using the **RTS** handshaking signal provided by the UART. The sender checks the **RTS** to ensure the UART is ready to receive data. In double-buffered reception **RTS** goes high when **ASC**_*n*_**RxBuffer** is empty, in FIFO-controlled operation it goes high when **RxHalfFull** is zero.

The **RxD** pin is sampled at 16 times the rate of the selected baud rate. 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 a 0, then the receive circuit is reset and waits for the next falling edge transition at the **RxD** pin. If the start bit is valid, i.e. is 0, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register. For subsequent data and parity bits, the majority decision of the seventh, eighth and ninth samples in each bit time is used to determine the effective bit value. The effective values received on **RxD** are shifted into a 10-bit input shift register.

For 0.5 stop bits, the majority decision of the third, fourth, and fifth samples during the stop bit is used to determine the effective stop bit value. For 1 and 2 stop bits, the majority decision of the seventh, eighth, and ninth samples during the stop bits is used to determine the effective stop bit values. For 1.5 stop bits, the majority decision of the fifteenth, sixteenth, and seventeenth 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 will not be recognized.

7.8.4.1 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 OR-ed to yield a single parity error bit.
- The framing error bit (**FrameError**) in the **ASC**_*n*_**Status** register is set when the **RxD** pin is not a 1 during the programmed number of stop bit times (see section 7.8.4)
  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 0-9) of each input entry records whether or not there was a frame error when that entry was received (i.e. 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 0-9) 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 0-9) and the ninth (bit 8 of 0-9) 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 7.8.2.1), the ninth bit of each input entry (bit 8 of 0-9) is undefined.

7.8.4.2 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 i.e. no entries are present, the **RxBufFull** bit of the **ASC**_*n*_**Status** register is set to ‘0’. If one or more FIFO entries are present, the **RxBufFull** bit of the **ASC**_*n*_**Status** register is set to 1. If the input FIFO is not empty, a read from **ASC**_*n*_**RxBuffer** will get 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 8 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 a ‘1’). The receive circuit then waits for the next falling edge transition at the **RxD** pin.

The **OverrunError** bit of the **ASC**_*n*_**Status** register is set when the input FIFO is full and a character is loaded from the input shift register into the input FIFO. It is cleared when the **ASC**_*n*_**RxBuffer** register is read.

After changing the **FifoEnable** bit, it is important to reset the FIFO to empty by writing to the **ASC**_*n*_**RxReset** register; otherwise the state of the FIFO pointers may be garbage.

Double-buffered Reception

Double-buffered operation is enabled and the FIFOs disabled by writing 0 to the **FifoEnable** bit of the **ASC**_*n*_**Control** register. This mode can be seen as equivalent to a FIFO-controlled operation with a FIFO of length 1 (the first FIFO vector is in fact used as the buffer). When the last stop bit has been received (at the end of the last programmed stop bit period) the content of the receive shift register is transferred to the receive data buffer register (**ASC**_*n*_**RxBuffer**). The receive buffer full flag (**RxBufFull**) is set, and the parity (**ParityError**) and framing error (**FrameError**) flags are updated at the same time, after the last stop bit has been received, i.e. at the end of the last stop bit programmed period. The flags are updated even if no valid stop bits have been received. The receive circuit then waits for the next falling edge transition at the **RxD** pin.

7.8.4.3 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 baud rate 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 will decrement 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 will be set.

When the software has emptied the input FIFO, the time-out counter will reset and start decrementing. If no more characters arrive, when the counter reaches zero the **TimeoutIdle** bit of the **ASC**_*n*_**Status** register will be set.

7.8.5 Baud Rate Generation

Each ASC has its own dedicated 16-bit baud rate generator with 16-bit reload capability. The baud rate generator has two possible modes of operation.

The **ASC**_*n*_**BaudRate** register is the dual-function baud rate 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 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 will not be reloaded until the first subsystem clock cycle after the **Run** bit is 1.

The baud rate 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 subsystem clock whereas Mode 1 uses a loop-back accumulator. Mode 0 is recommended for low baud rates (below 19.2K baud), where its error deviation is low, and Mode 1 is recommended for baud rates above 19.2 K.

7.8.5.1 Baud Rates

The baud rate generator provides an internal oversampling clock at 16 times the external baud rate. This clock only ticks if the **Run** bit of the **ASC**_*n*_**Control** register is set to 1. Setting this bit to 0 will immediately freeze the state of the ASCs transmitter and receiver.

Mode 0

When the **BaudMode** bit in the **ASC**_*n*_**Control** register is set to 0, the baud rate and the required reload value for a given baud rate can be determined by the following formula:

$$BaudRate = \frac{f_{CPU}}{16 \times ASCBaudRate} \qquad ASCBaudRate = \frac{f_{CPU}}{16 \times BaudRate}$$

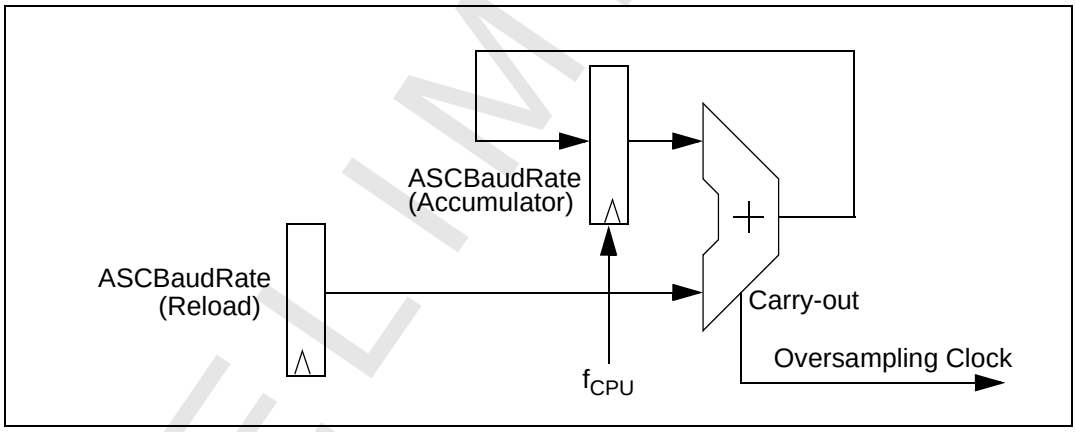
where: *ASCBaudRate* represents the content of the **ASC**_*n*_**BaudRate** reload value register, taken as an unsigned 16-bit integer and f_{CPU} is the frequency of the CPU.

The baud rate counter is clocked by the subsystem 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 baud rate 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 baud rate is controlled by the following circuit:



Writes to **ASC**_*n*_**BaudRate** go to the reload register. Reads from **ASC**_*n*_**BaudRate** return the value in the accumulator register. Both registers are 16 bits wide and are clocked by the subsystem clock.

If the system clock frequency is f_{CPU}, writing a value of *ASCBaudRate* to the **ASC**_*n*_**BaudRate** register results in an average oversampling clock frequency of:

$$\frac{ASCBaudRate \times f_{CPU}}{2^{16}}$$

so the baud rate is given by:

$$BaudRate = \frac{ASCBaudRate \times f_{CPU}}{16 \times 2^{16}}$$

This gives good granularity, and hence low baud rate deviation errors, at high baud rate frequencies.

7.8.6 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 will 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 73.

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:

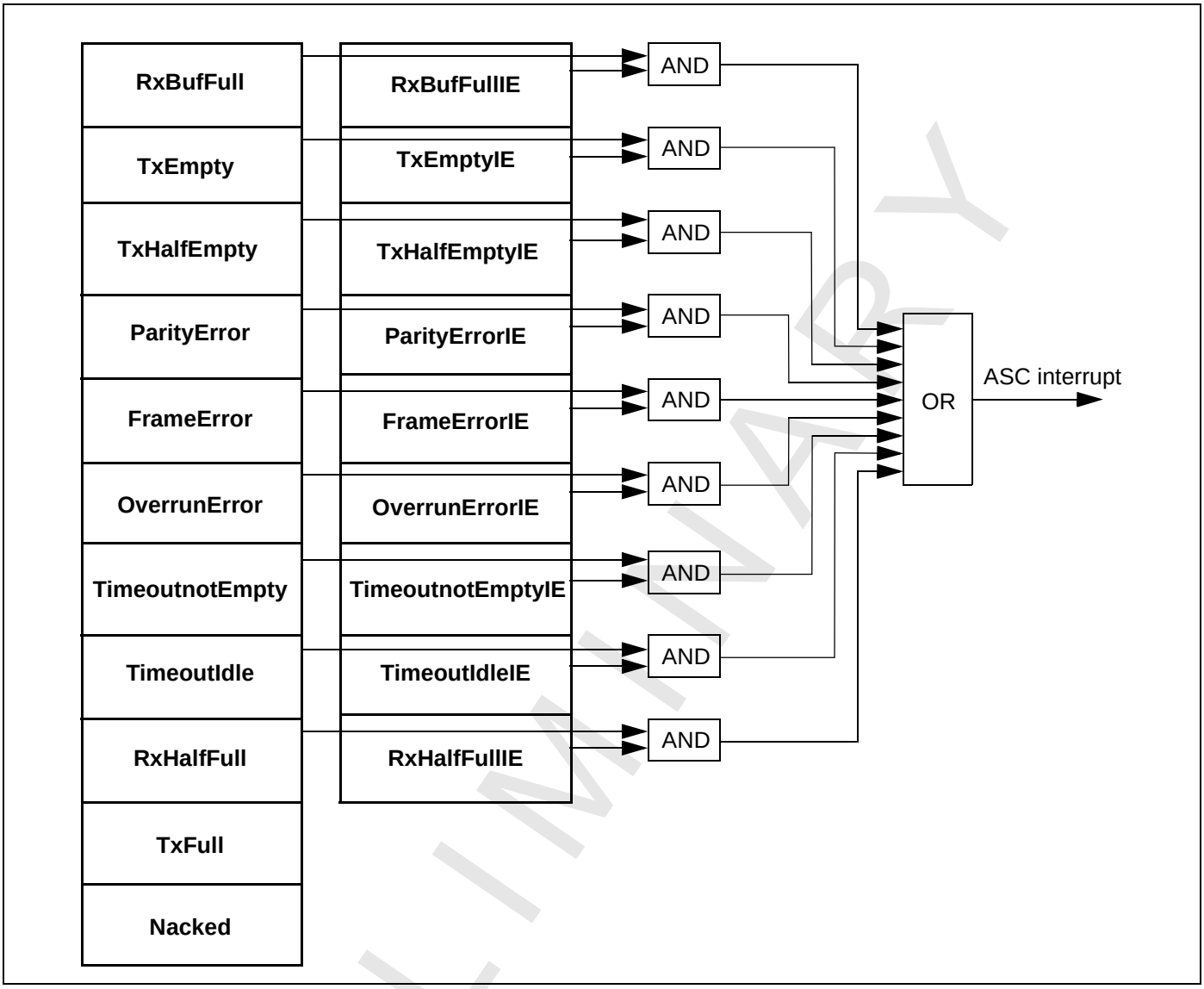
- Transmitter interrupt status bits (**TxEmpty**, **TxHalfEmpty**) are reset when a character is written to the transmitter buffer
- Receiver interrupt status bit (**RxBufFull**) is reset when a character is read from the receive buffer
- **ParityError** and **FrameError** status bits are reset when all characters containing errors have been read from the receive input buffer
- The **OverrunError** status bit is reset when a character is read from **ASC**_*n*_**RxBuffer**

7.8.6.1 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 (i.e. other than the error interrupt) when FIFOs are disabled the ASC provides three interrupt requests to control data exchange via the serial channel:

- **TxHalfEmpty** is activated when data is moved from **ASC**_*n*_**TxBuffer** to the transmit shift register
- **TxEmpty** is activated before the last bit of a frame is transmitted
- **RxBufFull** is activated when the received frame is moved to **ASC**_*n*_**RxBuffer**

Figure 73: ASC Status and Interrupt Registers



As shown in Figure 74,**TxHalfEmpty** is an early trigger for the reload routine, while **TxEmpty** indicates the completed transmission of the data field of the frame. Therefore, software using handshake should rely on **TxEmpty** at the end of a data block to make sure that all data has really been transmitted.

For single transfers it is sufficient to use the transmitter interrupt (**TxEmpty**), which indicates that the previously loaded data has been transmitted, except for the last bit of a frame.

For multiple back-to-back transfers it is necessary to load the next data before the last bit of the previous frame has been transmitted. The use of **TxEmpty** alone would leave just one stop bit time for the handler to respond to the interrupt and initiate another transmission. Using the output buffer interrupt (**TxHalfEmpty**) to signal for more data allows the service routine to load a complete frame, as **ASC**_*n*_**TxBuffer** may be reloaded while the previous data is still being transmitted.

7.8.6.2 Using the ASC Interrupts when FIFOs are Enabled

To transmit a large number of characters back to back, the driver routine would initially write 16 characters to **ASC**_*n*_**TxBuffer**. Then every time a **TxHalfEmpty** interrupt fired, it would write 8

more. When there is nothing more to send, a **TxEmpty** interrupt would tell the driver that everything has been transmitted.

When receiving, the driver could use **RxBufFull** to interrupt every time a character arrived. Alternatively, if data is coming in back-to-back, it could use **RxHalfFull** to interrupt it when there was more than 8 characters in the input FIFO to read. It would have as long as it takes to receive 8 characters to respond to this interrupt before data could overrun. If less than 8 characters streamed in, and no more were received for at least a time-out period, the driver could be woken up by one of the two time-out interrupts, **TimeoutNotEmpty** or **TimeoutIdle**.

Figure 74: ASC Transmission

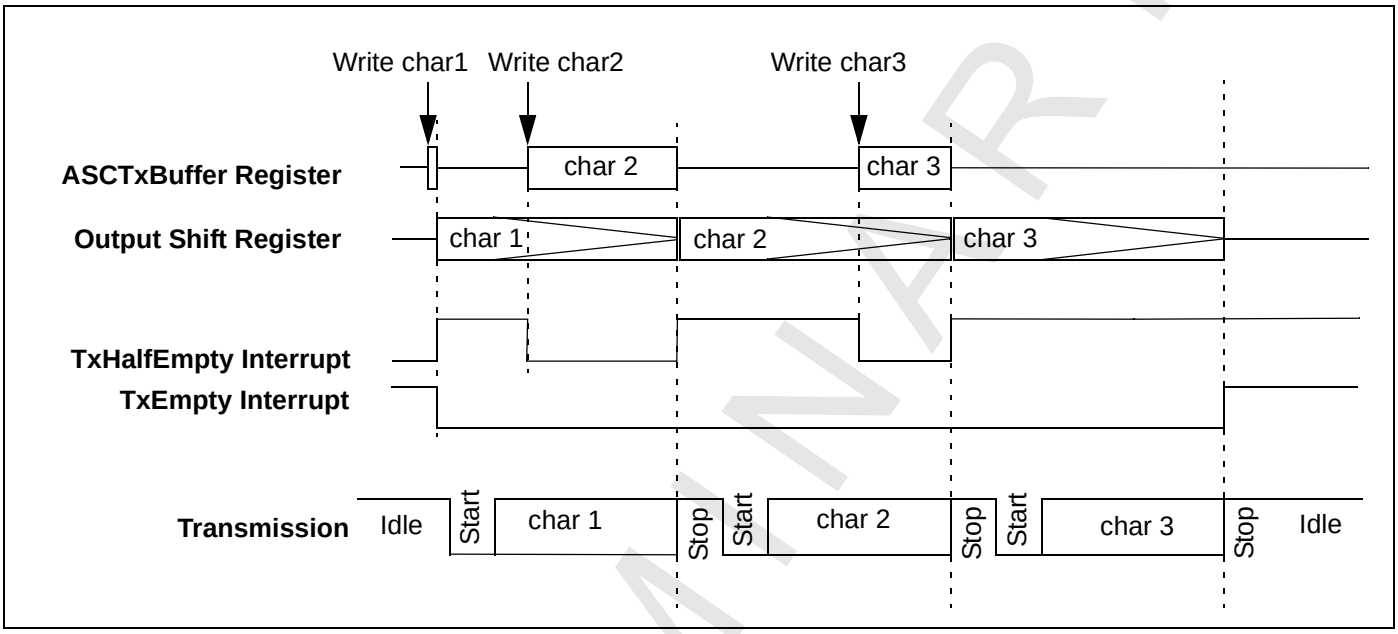
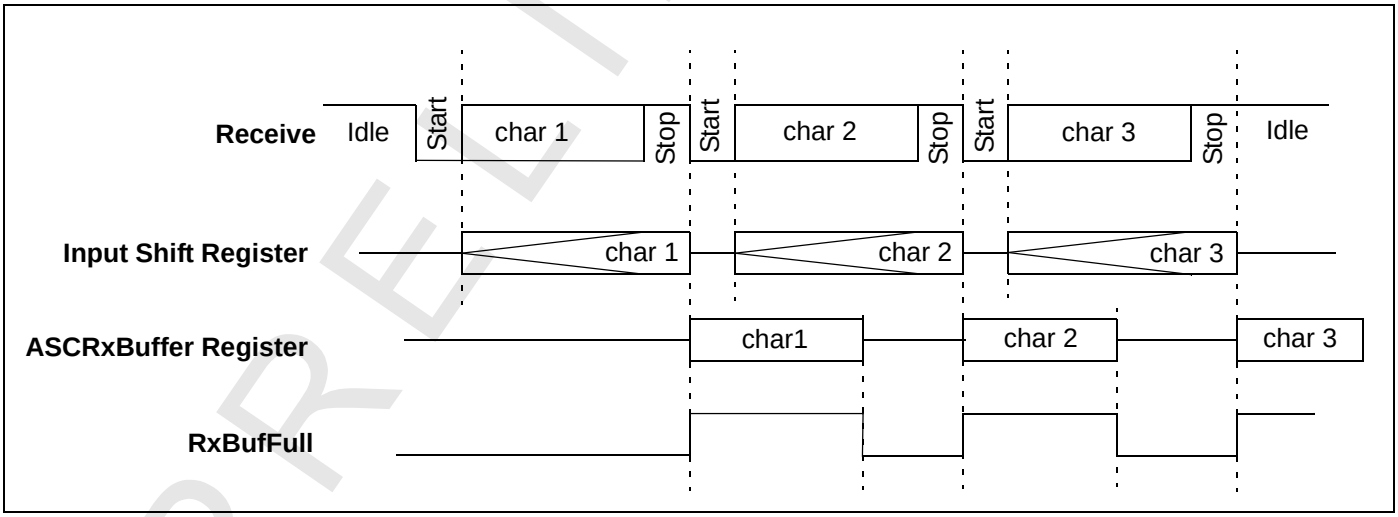


Figure 75: ASC Reception



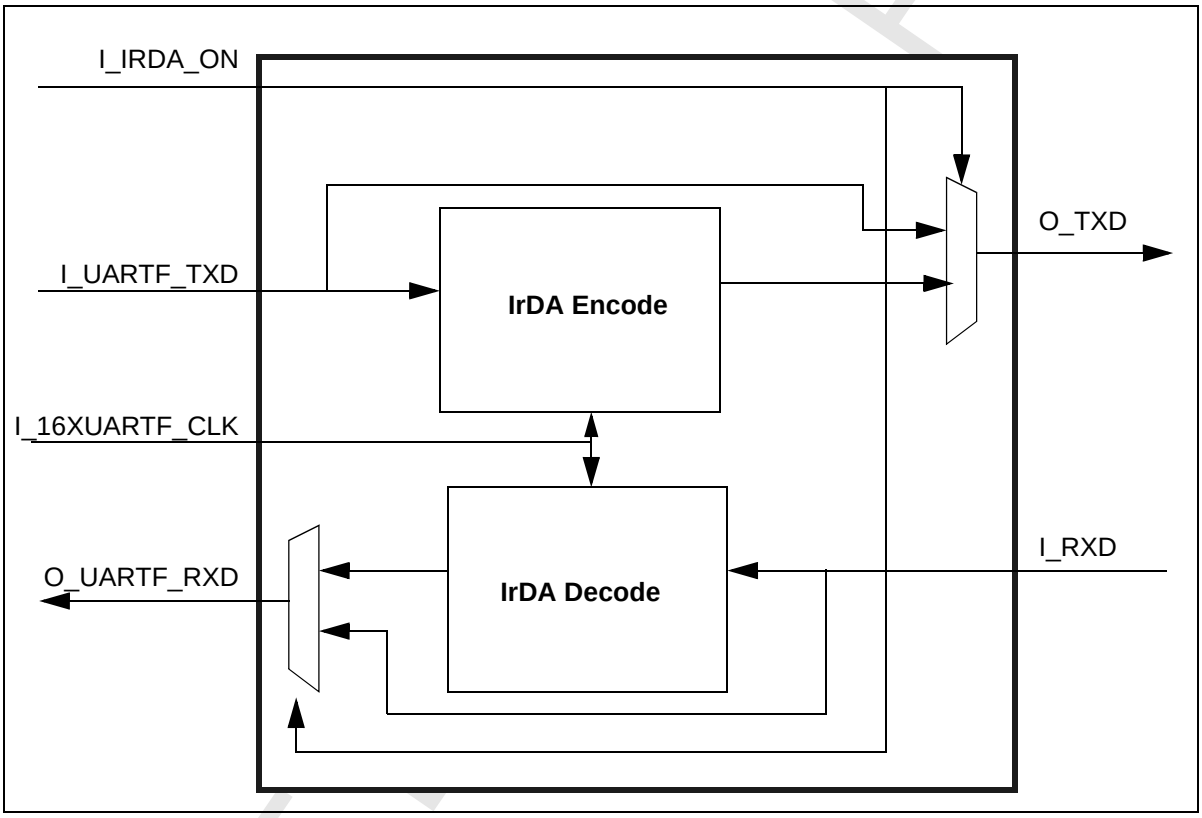
7.9 IrDA Encoder/Decoder

The IrDA Encoder/Decoder performs the modulation/demodulation used to both encode and decode the electrical pulses sent between the IR Transceiver and the UART. The block contains the following machines which derive their timing from the system clock and the enable signal (16X_UART_CLK):

- Serial IR Encoder
- Serial IR Decoder

The Encode block is driven by the negative edge triggered I_UARTF_TXD signal from the UART. This initiates the modulation state machine resulting in the 3/16 modulated O_TXD signal which drives the IR Transceiver module.

Figure 76: IrDA Block Diagram



7.9.1 Encoding Scheme

The Encoder sends a pulse for every space or ‘0’ that is sent on the I_UARTF_TXD line. On a high to low transition of the I_UARTF_TXD line, the generation of the pulse is delayed for 7 cycles of the I_16XUARTF_CLK line before the pulse is set high for 3 cycles of the I_16XUARTF_CLK line (or 3/16 of a bit time) and subsequently pulled low. The Encoding scheme is described in Figure 77.

7.9.2 Decoding Scheme

A high to low transition of the I_RXD line from the IR Transceiver signifies a 3/16 pulse. This pulse is stretched to accommodate 1 bit time (16 I_16XUARTF_CLK clock cycles). Every pulse that is received is translated into a ‘0’ or space on the O_UARTF_RXD line equal to 1 bit time. The maximum value is 80% of the minimum bit time. The Decoding scheme is described in Figure 78.

Figure 77: IrDA Encoding Scheme

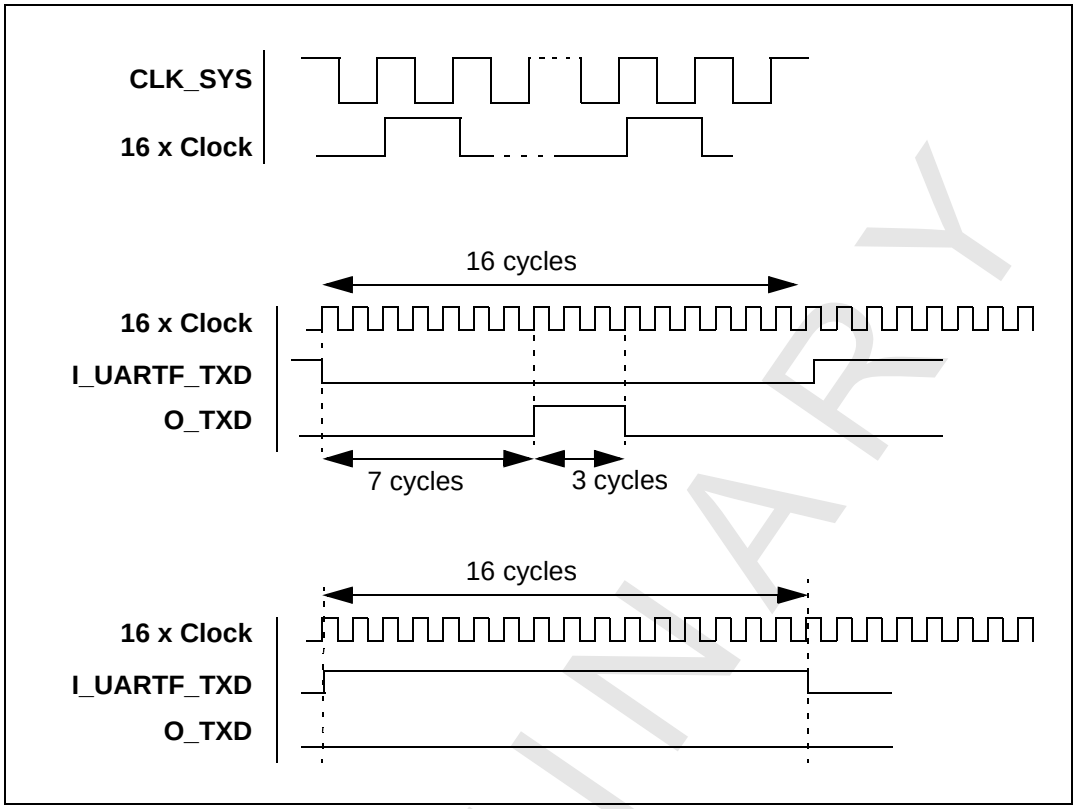
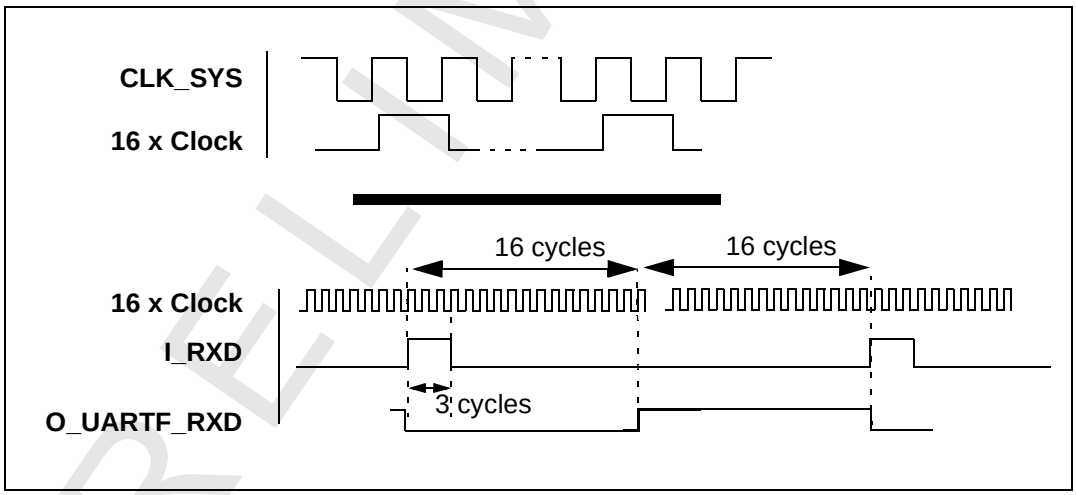


Figure 78: IrDA Decoding Scheme



7.9.3 Register

There is not a dedicated register interface (STBus T1), only a single 'I_IRDA_ON' primary input with a reset value of '0'.

When the IrDA block is 'off' (bit equal to '0'), the block is bypassed.

- O_TXD = '0'
- O_UARTF_RXD = '1'

Note: The dedicated register bit is in the VTG block.

8 General Package Information

8.1 Package Mechanical Data

Figure 79: 208-pin Plastic Quad Flat Package

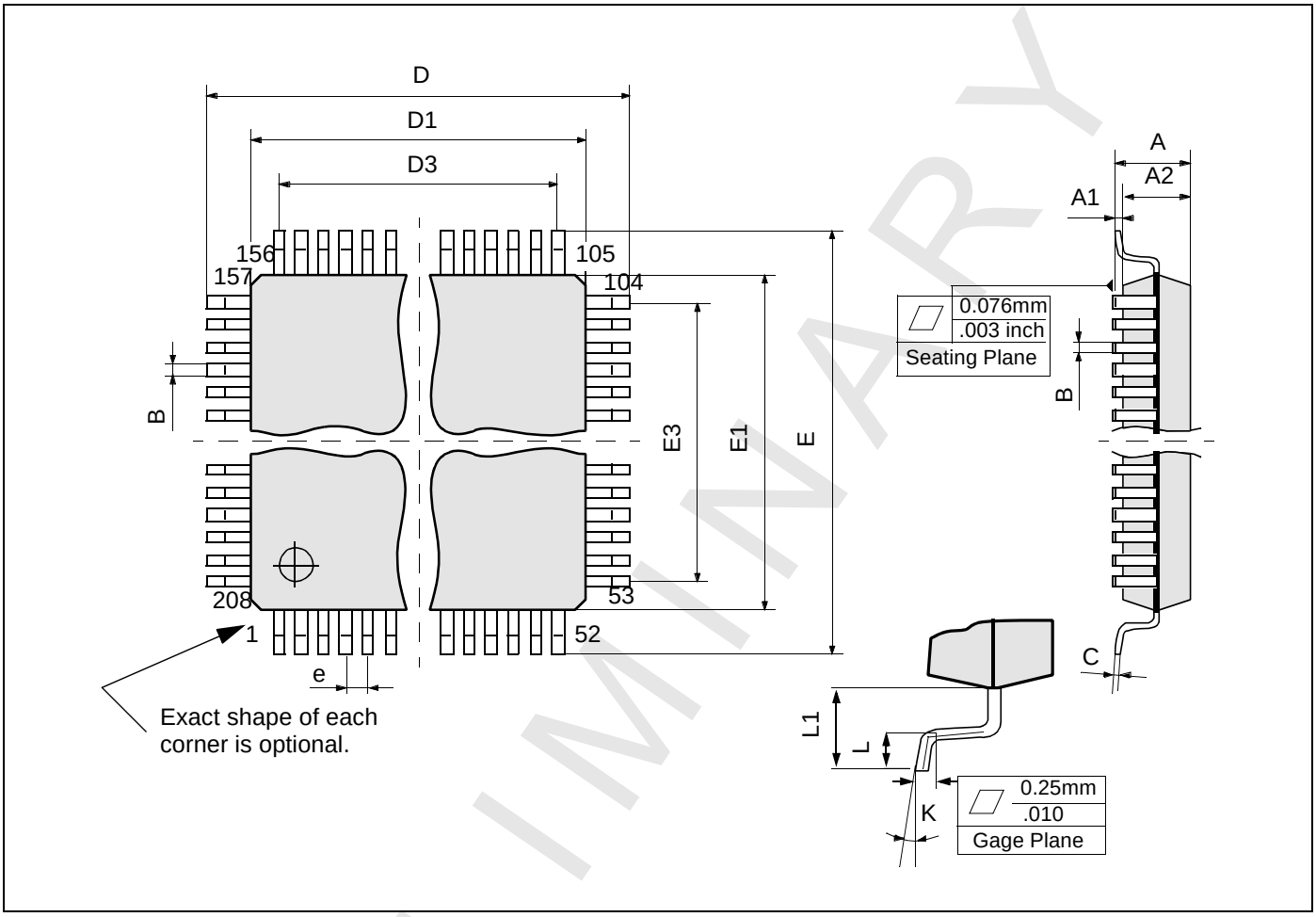


Table 26: Package Mechanical Dimensions

Dim.	mm			inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			4.10			0.161
A1	0.25			0.010		
A2	3.40	3.20	3.60	0.134	0.126	0.142
B	0.17		0.27	0.007		0.011
C	0.09		0.20	0.003		0.008
D		31.60			1.205	
D1		28.00			1.102	
D3		25.50			1.004	
e		0.50			0.020	
E		30.60			1.205	

Table 26: Package Mechanical Dimensions

Dim.	mm			inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
E1		28.00			1.102	
E3		25.50			1.004	
L	0.45	0.60	0.75	0.018	0.024	0.029
L1		1.30			0.051	
K	0°	3.5°	7°	0°	3.5°	7°

9 Electrical Characteristics

The STV3550 contains circuitry used to protect inputs against damage due to high static voltage or electric fields. Nevertheless, it is recommended that normal precautions be observed in order to avoid subjecting this high-impedance circuit to voltages above those quoted in the Absolute Maximum Ratings. For proper operation, it is recommended that the input voltage V_{IN} be constrained within the range: $(V_{SS} - 0.3\ V) \leq V_{IN} \leq (V_{DD33\_IO} + 0.3\ V)$

To enhance reliability of operation, it is recommended to configure unused I/Os as inputs and to connect them to an appropriate logic voltage level such as V_{SS} or V_{DD}. It is also recommended to connect V_{DDA} and V_{DD} together on applications. (Same remark for V_{SSA} and V_{SS}).

All the voltages in the following tables are referenced to V_{SS}.

Stresses greater than those listed as “Absolute Maximum Ratings” may cause permanent damage to the device. Functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

9.1 Absolute Maximum Ratings

Symbol (Note 1)	**Ratings**	**Value**	**Unit**
VCC33_ADC	3.3 V Analog Voltage Supply for A/D Converter	3.8	V
VCC18_IO	1.8 V Analog Voltage Supply for PLL IOs	2.2	V
VCC18_PLL3	1.8 V Analog Voltage Supply for PLL 3	2.2	V
VCC18_PLL	1.8 V Analog Voltage Supply for PLL	2.2	V
VDD18_PLL1	1.8 V Analog Voltage Supply for PLL1	2.2	V
VDD33_IO	3.3 V Digital Voltage Supply	3.8	V
VDD18_CORE	1.8 V Digital Voltage Supply	2.2	V
VI_{MAX}	Voltage on Input and Bi-directional pins (See Note 2)	V_{SS}-0.3 to VDD33_IO+0.3	V
VO_{MAX}	Voltage on Output pins (See Note 3)	V_{SS}-0.3 to VDD33_IO+0.3	V
IO_{MAX}	Output Current	20	mA
ESD	ESD Susceptibility (Human Body Model: 100 pF capacitor discharged through a 1.5 kΩ serial resistor) (See Note 4)	2	kV

Note: 1 All voltages listed are referenced to ground (0 V, GND, V_{SS}).

2 Voltage at input pins are 5 V tolerant, except for analog PIO pins.

3 Voltage at output pin configured in open-drain are 5 V tolerant.

4 Mininum guaranteed value.

9.2 Thermal Data

Symbol	Ratings	Value	Unit
T_{AMB}	Ambient Temperature Range	0 to + 70	°C
T_{STG}	Storage Temperature Range	-65 to +150	°C
T_J	Junction Temperature	150	°C
R_{thJA}	Maximum Thermal Resistance (Junction-to-Ambient)	45	°C/W

9.3 DC Electrical Characteristics

(T_{AMB} = 0 to 70°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
VCC33_ADC	3.3 V Analog Voltage Supply for A/D Converter		3.0	3.3	3.6	V
VCC18_IO	1.8 V Analog Voltage Supply for PLL IOs		1.6	1.80	2.0	V
VCC18_PLL3	1.8 V Analog Voltage Supply for PLL 3		1.6	1.80	2.0	V
VCC18_PLL	1.8 V Analog Voltage Supply for PLL		1.6	1.80	2.0	V
VDD18_PLL1	1.8 V Analog Voltage Supply for PLL1		1.6	1.80	2.0	V
VDD33_IO	3.3 V Digital Voltage Supply		3.0	3.3	3.6	V
VDD18_CORE	1.8 V Digital Voltage Supply		1.6	1.80	2.0	V
V_{IH}	Input Logic 1 Voltage (except XTALIN)	CMOS	2.0	3.3	VDD33_IO+0.3	V
V_{IH}	Input Logic 1 Voltage (XTALIN only)	CMOS	1.5	1.8	VDD18_IO+0.3	V
V_{IL}	Input Logic 0 Voltage	CMOS			0.8	V
I_L	Input Leakage Current on Input and Bi-directional pins[1]		-1		+1	µA
V_{OH}	Output High Level	Push-pull I_{LOAD} = -0.8 mA	V_{DD}-0.8			V
V_{OL}	Output Low Level	Push-pull I_{LOAD} = +1.6 mA			0.4	V
V_{IHRS}	Reset in High Level		0.7 V_{DD}			V
V_{ILRS}	Reset in Low Level				0.3 V_{DD}	V
V_{HYRS}	Reset in Hysteresis		0.3			V
V_{IHVH}	HSYNC/VSYNC Input High Level		0.7 V_{DD}			V
V_{ILVH}	HSYNC/VSYNC Input Low Level				0.3 V_{DD}	V
V_{HYHV}	HSYNC/VSYNC Input Hysteresis		0.5			V
I_{LKRS}	Reset Pin Input	0 < V_{IN} < VDD33_IO	-1		+1	µA
I_{LKAD}	A/D Pin Input Leakage Current	Input	-1		+1	µA
I_{LKOS}	XTALIN Pin Input Leakage Current	0 < V_{IN} < VCC18_IO	-1		+1	µA

1. $0 \le V_I \le$ VDD33_IO (for XTALIN only: $0 \le V_I \le$ VCC18_IO)

9.4 Supply Current Characteristics

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
VCC33_ADC	3.3 V Analog Voltage Supply (for A/D Converter) Current	Typical Standby Mode		0.6 0.25		mA
VCC18_PLL1	1.8 V Analog Voltage Supply (for PLL) Current	Typical Standby Mode		20 11		mA
VCC18_PLL3	1.8 V Analog Voltage Supply (for PLL) Current	Typical Standby Mode		7 6		mA
VCC18_IO	1.8 V Analog Voltage Supply (for PLL I/Os) Current	Typical Standby Mode		15 13		mA
VDD18_PLL1	1.8 V Analog Voltage Supply (for PLL) Current	Typical Standby Mode		17 3		mA
VDD33_IO	3.3 V Digital Voltage Supply Current	Typical Standby Mode		30 0.4		mA
VDD18_CORE	1.8 V Digital Voltage Supply Current	Typical Standby Mode		400 1.5		mA

9.5 H/V Synchronization Characteristics

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
	Output Luma/Sync Non-alignment				± 2.5	nS
	Output Chroma/Sync Non-alignment				± 20	nS
	Line-PLL Capture Range			± 2		kHz
	Time Constant to recover 40 µs of phase shift			TBD		
	HSYNC Uncertainty in relation to RGB/YCrCb outputs				5	ns
	VSYNC Uncertainty in relation to RGB/YCrCb outputs				5	ns

9.6 Clock Characteristics

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
	Differential Clock			27		MHz
CLKXTM and CLKXTP						
V_{OL}	Output Voltage Low Level	I_{LOAD} = 1.6 mA			0.6	

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
V_{OH}	Output Voltage High Level	I_{LOAD} = -0.8 mA	1.2			
Internal Clock						
f_{XTAL}	Crystal Frequency			27		MHz
R_{BIAS}	Internal Bias Resistance					
	Accuracy (including Temp. shift)				±50	ppm
	Peak-to-Peak Jitter				±2	nS

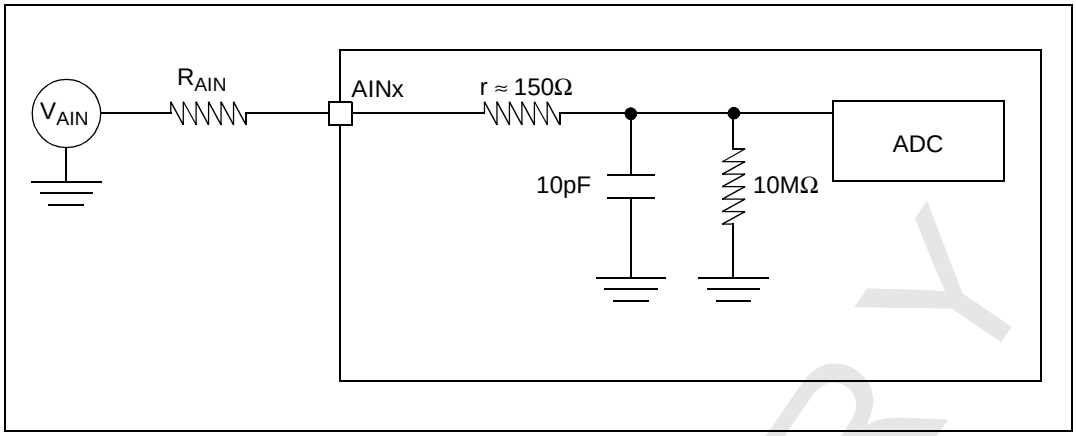
9.7 ADC Characteristics

Analog Parameters Table
(VCC33_ADC = 3.3 V ±10%; CLK_ADC (Typ.) = 3.4 MHz; T_{AMB} = 0 to 70°C; unless otherwise specified).

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
	Analog Input Range		GND_ ADC		VCC33_ ADC	V
	Conversion Time Fast/Slow		16			CLK_AD C
	Power-up Time			5		ms
	Resolution			7		bits
	Differential Non Linearity	DNL Error= Max {[V(i) -V(i-1)] / LSB-1}			5	LSBs[1]
	Integral Non Linearity	INL Error= Max {[V(i) -V(0)] / LSB-i}			7	LSBs
	Absolute Accuracy	Absolute Accuracy = Overall Max. Conversion Error			7	LSBs
	Input Resistance	Must be considered as the on-chip serial resistor before the sampling capacitor.			1.5	kOhm
	Hold Capacitance				1.92	pF
	Clock Frequency			3.4		MHz

1. LSB = VCC33_ADC/1024

Figure 80: Typical Application with A/D Converter



9.8 I²C Bus Characteristics

Table 27: Key I²C Timing Requirements

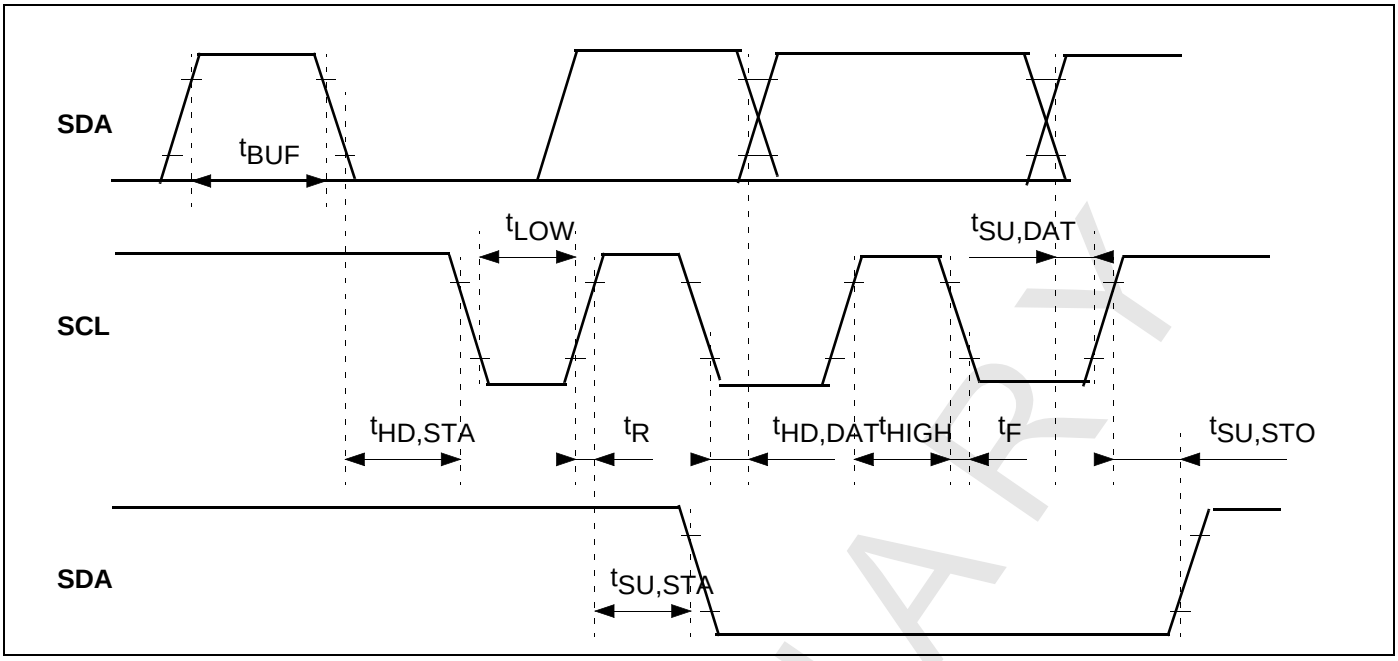
Parameter	Min.	Max.
SCL Clock Frequency		100 kHZ or 400 kHz[1]
Hold Time of SCL after SDA START or REPSTART Condition. After this time the first clock low pulse is created	1.0us or 0.6us[1]	
LOW Period of the SCL clock	4.7us or 1.3us[2]	
HIGH period of the SCL clock	4.0us or 0.6us[1]	
SDA Set-up time relative to SCL for a repeated START condition	4.6us or 0.6us[1]	
SDA Data Hold Time relative to SCL	300ns[2]	
SDA Data Setup Time relative to SCL	250ns or 100ns[3]	
Rise time of SDA and SCL lines	$20+0.1C_b$[4]	100ns or 300ns[1]
Fall time of SDA and SCL lines	$20+0.1C_b$[3]	300ns
SDA Setup time relative to SCL for a STOP condition	4.0us or 0.6us[1]	
Capacitive load of each bus line (C_b)		400pF

1. In FAST Mode
2. Actual requirement is 0ns but hold time must be at least 300ns on SDA line to comply with uncertain period of max fall time of 300ns on the SCL line.
3. In FAST mode, but when used with normal mode devices must still meet the 250ns minimum
4. Where C_b is the total capacitance on one bus line in pF

Table 28: I²C Timing Constants

I²C Timing Name	I²C Standard Minimum Time	Constant Value (cycles)	Typical at 60 MHz
Data Hold Time	300ns	21	350ns
Setup Time For STOP Condition $T_{SU.STO}$	4.0us	300	5.0us
Setup Time For START (repeated) Condition $T_{SU.ST}$	4.7us	300	5.0us
Hold Time (repeated) START Condition $T_{HD:STA}$	4.0us	300	5.0us
Data Setup Time $T_{SU:DAT}$	250ns	21	300ns

Figure 81: I²C Bus Timing



10 Timing Specifications

The timings are based on the following conditions unless otherwise stated:

1. Input rise and fall times of 3 ns (between 10% and 90%).
2. Output load = 30 pF
3. Output threshold = 1.5 V

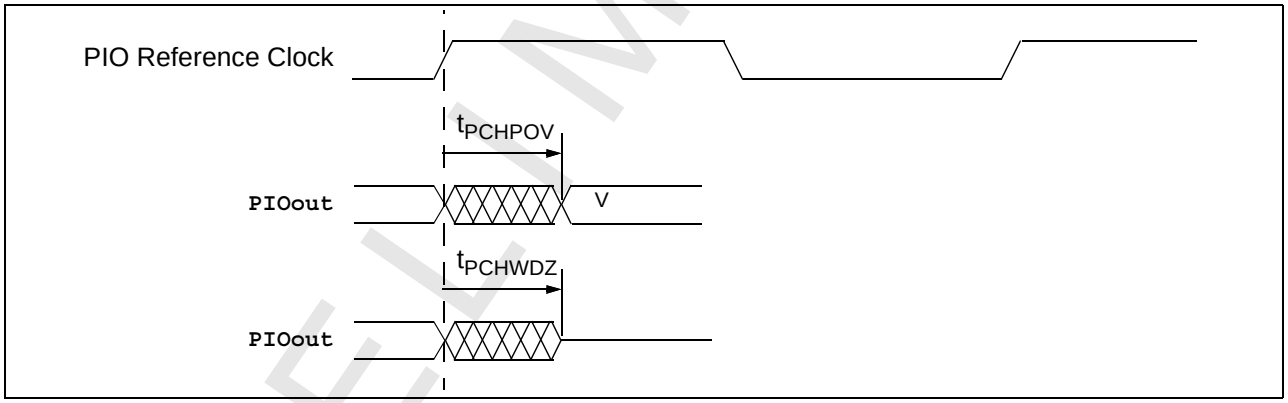
10.1 PIO Timings

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

Symbol	Parameter	Min	Max	Units	Note
t_{PCHPOV}	PIO_refclock high to PIO output valid		20	ns	1
t_{PCHWDZ}	PIO tristate after PIO_refclock high		20	ns	1
t_{PIOr}	Output Rise Time (C_{LOAD} = 50 pF)		20	ns	
t_{PIOf}	Output Fall Time (C_{LOAD} = 50 pF)		20	ns	

1. These timings do not apply when a PIO pin is an output for the PWM outputs, since these are generated using a different clock.

Figure 82: PIO Timings

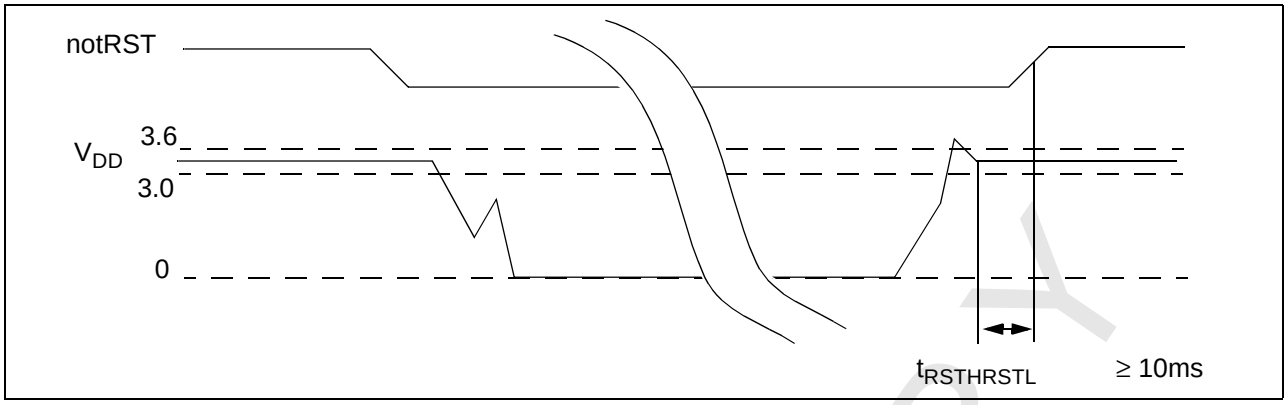


10.2 Reset Timings

The $\mathbf{t}_{RSTHRSTL}$ 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) will fall within these limits and the $\mathbf{t}_{RSTHRSTL}$ value should be used for both cases.

Symbol	Parameter	Min	Nom	Max	Units	Notes
$t_{RSTHRSTL}$	**notRST**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 83: Reset Timings



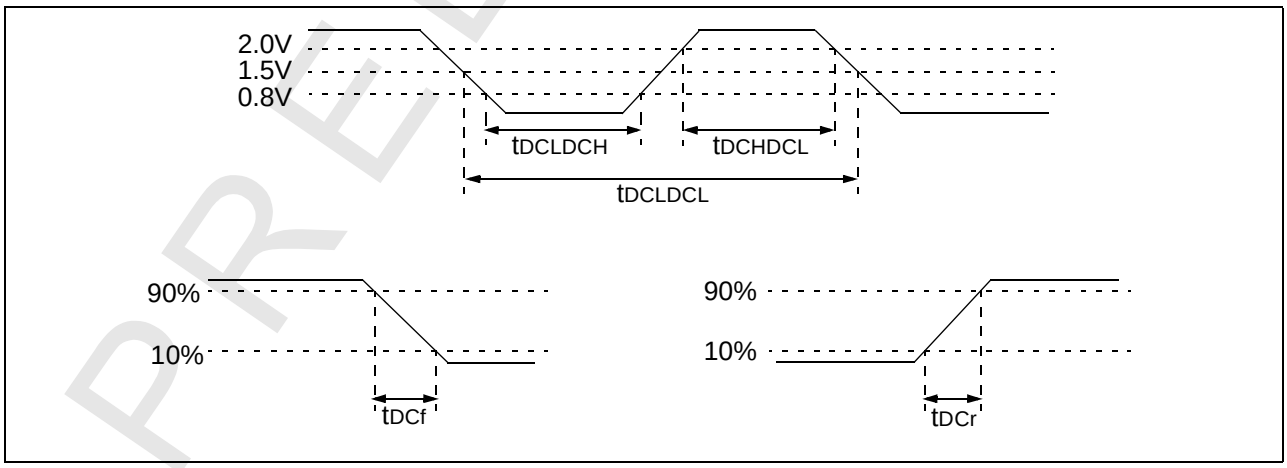
10.3 Clock Timings

10.3.1 XTALIN Timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
tDCLDCH	EXTALIN Pulse Width Low	6			ns	
tDCHDCL	EXTALIN Pulse Width High	10			ns	
tDCLDCL	EXTALIN Period		37		ns	1, 2
tDCr	CLKXTP Rise Time			10	ns	3
tDCf	CLKXTM Fall Time			10	ns	10.3.1

1. Measured between corresponding points on consecutive falling edges.
2. Variation of individual falling edges from their nominal times.
3. Clock transitions must be monotonic within the range VIH to VIL.

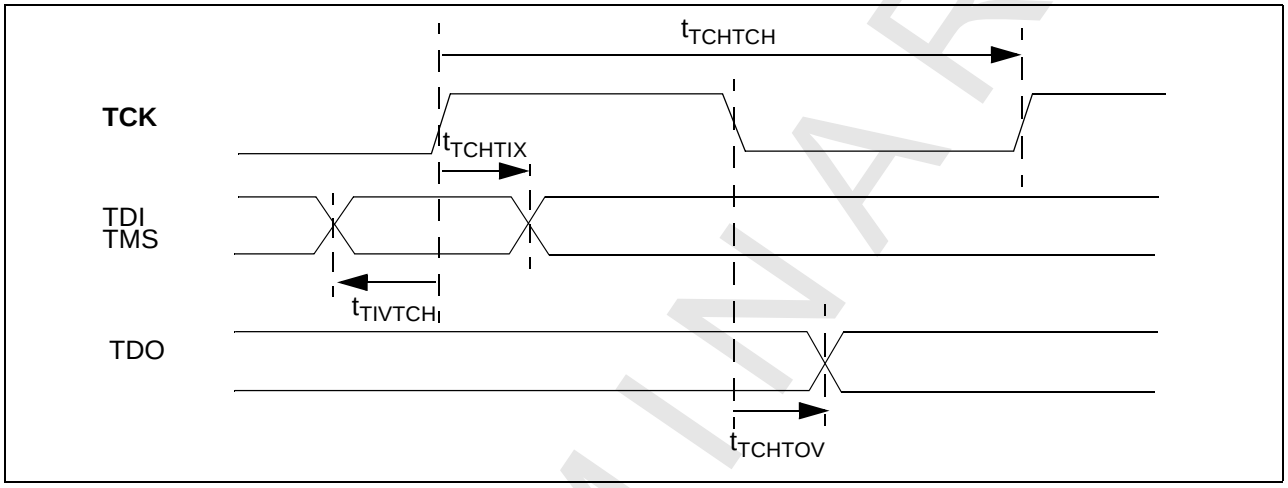
Figure 84: XTALIN Timings



10.4 TAP Timings

Symbol	Parameter	Min	Nom	Max	Units
t_{TCHTCH}	TCK period	50			ns
t_{TIVTCH}	TAP inputs valid to TCK high	10			ns
t_{TCHTIX}	TAP input hold after TCK high	10			ns
t_{TCHTOV}	TCK low to TAP output valid			50	ns

Figure 85: TAP Timings

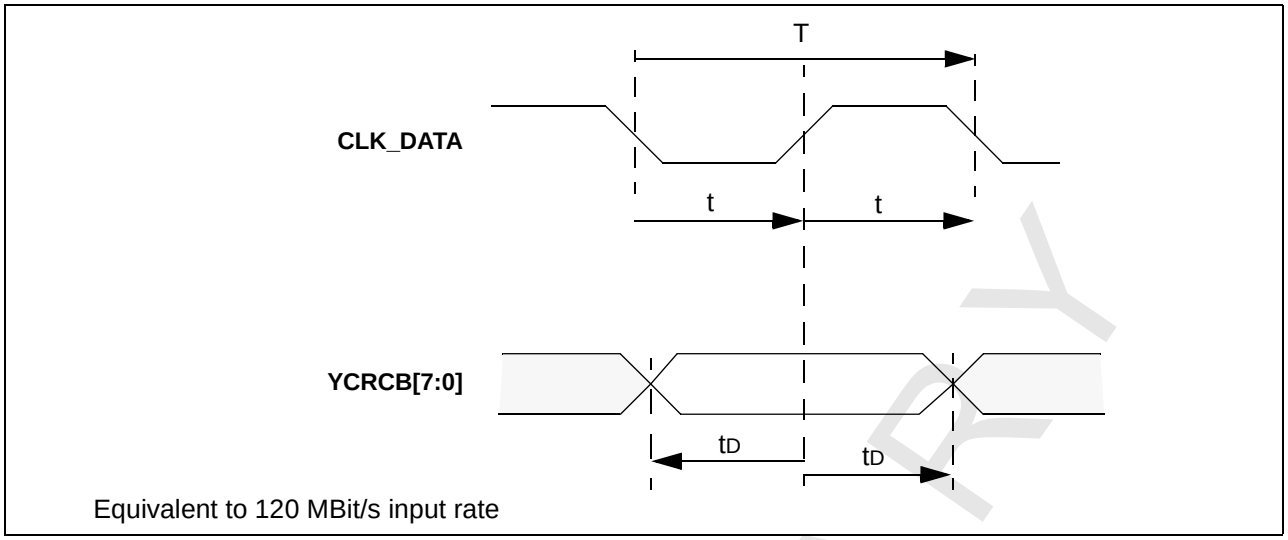


10.5 Input Video Stream Timings

10.5.1 SDIN Interface

Symbol	Parameter	Min	Nom	Max	Units
T	Clock Width (CLK_DATA)	37			ns
t	Clock Pulse Width (CLK_DATA)	15.5	18.5	21.5	ns
tD	Clock Pulse Width (YCRCB[7:0])	15.5	18.5	21.5	ns

Figure 86: SDIN Parallel Input Video Stream Input Timings



10.6 Output Video Port Interface (AC Electrical Characteristics)

10.6.1 Normal Mode (single edge clock output)

Symbol	Parameter	Min	Typ	Max	Units
fDCLK	Output operating frequency for: D_{CLK} @ C_{LOAD} = 50 pF			72	MHz
tDCLK rise				5	ns
tDCLK fall				5	ns

10.6.2 Multiplexed Mode (dual edge clock output)

Symbol	Parameter	Min	Typ	Max	Units
fDCLK	Output operating frequency for: D_{CLK} @ C_{LOAD} = 50 pF			36	MHz
tDCLK rise				5	ns
tDCLK fall				5	ns

Index

R

S

T

V

W

XYZ

11 Revision History

Table 29: Summary of Revisions

Revision	Modification	Date
1.0	First Draft	23 June 2003
1.1	Correction to XTALOUT and XTALIN pin number assignments	October 2003
1.2	Various corrections, modifications and updates.	January 2004
1.3	Inclusion of values in Chapter 10: Timing Specifications. Other minor corrections.	February 2004