PNX8526 User Manual UM10104_1

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

Revision History:

Version	Date	Document Status	Author(s)
01	8 October 2003	First release	Stephen Hibberd

PHILIPS

Contents

Chapter 1: Functional Specification

Chapter 2: System Memory Map and Protection

Chapter 3: Boot

Chapter 4: Endian Mode

Chapter 5: Clock Reset and Power Management

Chapter 6: Priority Interrupt Controller

Chapter 7: Pixel Formats

Chapter 8: PCI-XIO

Chapter 9: Main Memory Interface (MMI)

Chapter 10: GPIO/IR

Chapter 11: UART Ports

Chapter 12: SSI Port

Chapter 13: USB Port

Chapter 14: IEEE 1394

Chapter 15: I2C Ports

Chapter 16: Smartcard UART

Chapter 17: DV Input Ports

Chapter 18: Transport Stream Output Port

Chapter 19: DV Output Ports

Chapter 20: Audio Input Ports

Chapter 21: Audio Output Ports

Chapter 22: SPDIF Input Ports

Chapter 23: SPDIF Output Port

Chapter 24: MIPS RISC Processor Core

Chapter 25: Internal TM32 CPU Core Processor

Chapter 26: Video Input Processor (VIP)

Chapter 27: Memory Based Scaler (MBS)

Chapter 28: Advanced Image Composition Processor (AICP)

Chapter 29: 2D Drawing Engine

Chapter 30: Transport Stream Network

Chapter 31: Transport Stream Direct Memory Access

Chapter 32: DMA Controller and CRC Checker

Chapter 33: MPEG System Processor (MSP)

Chapter 34: MPEG Video Decoder

Chapter 35: PI-Bus Architecture

Chapter 36: MMI Arbiter

Chapter 37: Boundary-Scan Hardware

Chapter 38: MIPS EJTAG Software Debug Port

Chapter 39: TM32 JTAG Software Debug Port

Chapter 1: Functional Specification

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

1.1 Introduction

The PNX8526 is a highly integrated media processor intended for deployment in Advanced Set Top Box (ASTB) and Digital Television (DTV) systems. The PNX8526 is targeted at the mid to high-end ASTB/DTV systems, focusing on decoding “all format” HD and SD MPEG2 source material with standard definition (SD), or double line-rate SD display capabilities. This assumption implies that although the PNX8526 can process high level input formats, its display capabilities are primarily targeted at the hundreds of millions of NTSC, PAL, and SECAM television sets in use. It is also targeted at lower cost DTVs: those DTVs that are not considered high definition. Progressive output is also available for double line-rate television displays or for high resolution graphic content to be displayed on a computer monitor. The PNX8526 is designed in a high performance 0.12 micron process.

The PNX8526 is responsible for the source decode functions in ASTB/DTV systems. These functions include conditional access, MPEG2 transport stream de-mux, MPEG2 video decode, audio decode and processing, graphics generation, video processing, and image composition and display. A 32-bit 200 MHz VLIW processor, referred to as the TriMedia 3200 CPU core (TM32G CPU), carries out the majority of media processing operations performed by the PNX8526. Fixed function hardware will perform some operations that are not handled by the TM32 CPU. Additionally, the PNX8526 supports a number of peripheral interfaces such as I^2C, USB, IDE and UART. Additional interfaces such as IEEE-1284 and Ethernet are supported via Super I/O devices that reside on a PCI expansion bus. This bus also provides for glueless interface to 8-bit wide slave devices, such as Flash/ROM, DOCSIS modem, UARTs, etc.

An embedded MIPS processor (PR3940) running at 150 MHz is intended to run the OS. (There is no direct support for an external processor; however, a CPU of any type may be connected to the PNX8526 via the PCI interface.) This implies a complete CPU subsystem consisting of the CPU itself, local memory, and an interface to PCI. The MIPS processor is primarily responsible for control functions and graphics-intensive operating systems, while the TM32 CPU is responsible for running all real-time media processing functions. All resources supported within the PNX8526 are accessible by both the MIPS processor and the TM32 CPU. The software documentation of the PNX8526 provides more details on the interaction between the MIPS and the TM32 CPU.

The PNX8526 is intended to be used with a small companion IC, the PNX8510. This analog companion chip provides the majority of analog video and audio support for the output of the PNX8526. The PNX8510 companion is capable of simultaneously driving two video channels (6 DACs) and two stereo audio channels (4 DACs).

1.2 PNX8526 Feature Summary

- 200 MHz, 5 instruction/clock cycle 32-bit VLIW processing core (TM32G CPU)
- 150 MHz, MIPS PR3940 processing core
- External CPU support via PCI
- Support for multiple digital video (D1) input streams
- Support for multiple MPEG2 or DIRECTV transport streams (parallel format)
- On-chip conditional access for DVB, DES, MULTI2, ICAM, DIRECTV
- On-chip copy protection support for OpenCable and ATSC (NRSS-B)
- Simultaneous decode of two SD streams (MPEG2) or one HD MPEG Stream (AFD style HD-SD decode)
- Simultaneous decode of two AC-3 or equivalent audio streams
- High performance 2D rendering and DMA capability
- Dual image composition/screen refresh engines: four layer primary output, two layer secondary output
- Multiple channel output to support watch/record and multi-room modes
- Embedded 1394 link layer with 5C copy protection
- Soft modem support via SSI interface
- 16, 32, and 64 MB Unified Memory Architecture implemented with high speed SDRAM (166 MHz)
- System expansion capability via industry standard PCI bus
- Core peripherals (I^2C, UART, USB, etc.) on the chip, other peripherals supported via third-party SuperIO chip

1.3 PNX8526 Functional Overview

The functionality achieved within the PNX8526 can be divided into three major categories: *decode*, *processing,* and *display*. Decode functions take input data streams and convert those streams into memory based structures that the PNX8526 may further process. Decode functions may be simple, as in the case of storing D1 video into memory, or substantially more complex, as in the case of MPEG2. Processing functions are those that modify an existing data structure and prepare that structure for display functions. Display functions take the processed data structures from memory and generate the appropriate output stream. As in the case of the decode functions, display functions can be relatively simple, such as an I^2S audio output, or very complex, as in the case of multi-channel video display.

All decoded data structures are stored in memory, even when further processing is not required. This mechanism implies that there is no direct path between input and output data streams. The memory serves as the buffer to de-couple input and output data streams. Based on the mode of operation, there may be multiple data structures in memory for a given input stream. The PNX8526 uses the TM32 CPU and a timestamping mechanism to determine when a specific memory data structure is to be displayed.

The PNX8526 implements the required decode, processing, and display functions with a combination of fixed function hardware and TM32 CPU software modules. The MIPS processor is not intended to be involved with the three primary function types, other than to control them. The PNX8526 provides a good balance between those functions that are implemented in fixed hardware and those that are programmed to run on the TM32 CPU. The following tables illustrate how the major tasks are implemented under each of the three main functional areas.

Table 1: Decode Function Responsibilities

Decode Functions	TM32 CPU	Hardware	Comments
D1 digital video input		X	Video Input Processor (VIP)
Section filtering		X	Programmable hardware, with transparent mode to escape to software if a new provider uses an unforeseen method
PID filtering		X	
Transport de-mux		X	Flexible hardware, with option to perform transparent mode and do de-mux in software for ATSC/ other
MPEG2 decode		X	Complete MPEG2 algorithm below slice level
MPEG2 decode	X		Above slice level
Conditional access		X	DVB, MULTI2, ICAM, DES, DIRECTV internal to the PNX8526, POD, CI external
Audio decode	X		Two AC-3 streams or equivalent

Table 2: Processing Function Responsibilities

Processing Functions	TM32CPU	Hardware	Comments
Audio processing	X		Processing and mixing of various PCM or AC-3 streams
2D Rendering and DMA		X	
Primary video processing		X	Scaling, anti-flicker, and de-interlacing
Display CSC Matrix, CLUT, BCS, gamma correction		X	
Advanced video processing	X		Algorithms not in hardware may run on TM32 CPU (as bandwidth allows)

Table 2: Processing Function Responsibilities *…Continued*

Processing Functions	TM32CPU	Hardware	Comments
V.34, V.90 modem	X		
Transport stream creation		X	Partial transport stream for transmission across 1394 (simple derivative of input stream)
Transport stream creation	X		Transport stream assembly for transmission across 1394 (intelligently assembled stream)

Table 3: Display Function Responsibilities

Display Features	TM32CPU	Hardware	Comments
Video stream composition		X	Advanced Image Composition Processor (AICP)
Audio stream composition	X		

1.4 Display Modes

Table shows the supported display resolutions that provide acceptable performance, features and quality. The PNX8526 is primarily intended to drive standard definition analog television sets and low cost digital televisions. The PNX8526 is not intended to support full high definition (HD) output nor does it support on-chip high end image enhancement. However, the PNX8526 display capabilities are programmable, and a wide variety of display systems can be supported.

The PNX8526 supports two simultaneous output video channels: primary and secondary. In a watch/record scenario, the primary channel is intended to be the viewable channel. It will display a fully composited video image consisting of PIPs, menus, and other graphical information. The secondary channel is intended to be the recordable channel.

Other uses are possible. For example; external high-end image enhancement can be applied to the secondary display output channel, and brought back in to the PNX8526 through the third video input, and blended with graphics for emission through the primary display channel.

The PNX8526 hosts two independent Advanced Image Composition Processor (AICP) modules. The primary AICP contains four image layers and is mainly intended to be connected to a TV or monitor. The second AICP contains two layers and is intended to connect to a VCR. The layers have their own pixel clock and frame timing generator for fully independent display modes. Because of the independence of the layers, a wide variety of scenarios is possible.

To support scenarios other than watch/record, one or both AICPs can be set to divide their layers over two pixel/timing synchronous multiplexed outputs. This allows refresh of up to four screens, but with more limited compositing capabilities.

Table 4: Display Resolutions

Display Modes	Interface Mode	Sync	Interface Speed
4:2:2 CVBS or Y/C PAL/NTSC/SECAM resolution 4:2:2 Example PAL: 864 pixel/line x 312.5 lines/field x 50 Hz = 13.5 MHz/Y samples 6.75 MHz/U samples 6.75 MHz/V samples	4:2:2 Muxed Components	embedded SAV/ EAV D1 style	27 MHz
4:4:4 RGB or YUV PAL/NTSC/SECAM resolution 4:4:4 Example PAL: 864 pixel/line x 312.5 lines/field x 50 Hz = 13.5 MHz/component	4:4:4 Muxed Components	embedded SAV/ EAV D1 style or external H/V/ Blank	40.5 MHz
4:4:4 RGB or YUV 2FH (double line frequency -> double refresh rate) Example PAL: 864 pixel/line x 312.5 lines/field x 50 Hz x 2 = 27 MHz/component	4:4:4 Muxed Components	embedded SAV/ EAV D1 style or external H/V/ Blank	81 MHz
4:4:4 RGB or YUV 480P (PAL/NTSC resolution, progressive) Example PAL: 864 pixel/line x 625 lines/field x 50 Hz = 27 MHz/ component	4:4:4 Muxed Components	embedded SAV/ EAV D1 style external H/V/ Blank	81 MHz

1.5 System Functional Overview

Figure 1 shows a block diagram of a typical PNX8526-based system. The system shown is a “standalone system,” which uses the internal MIPS host.

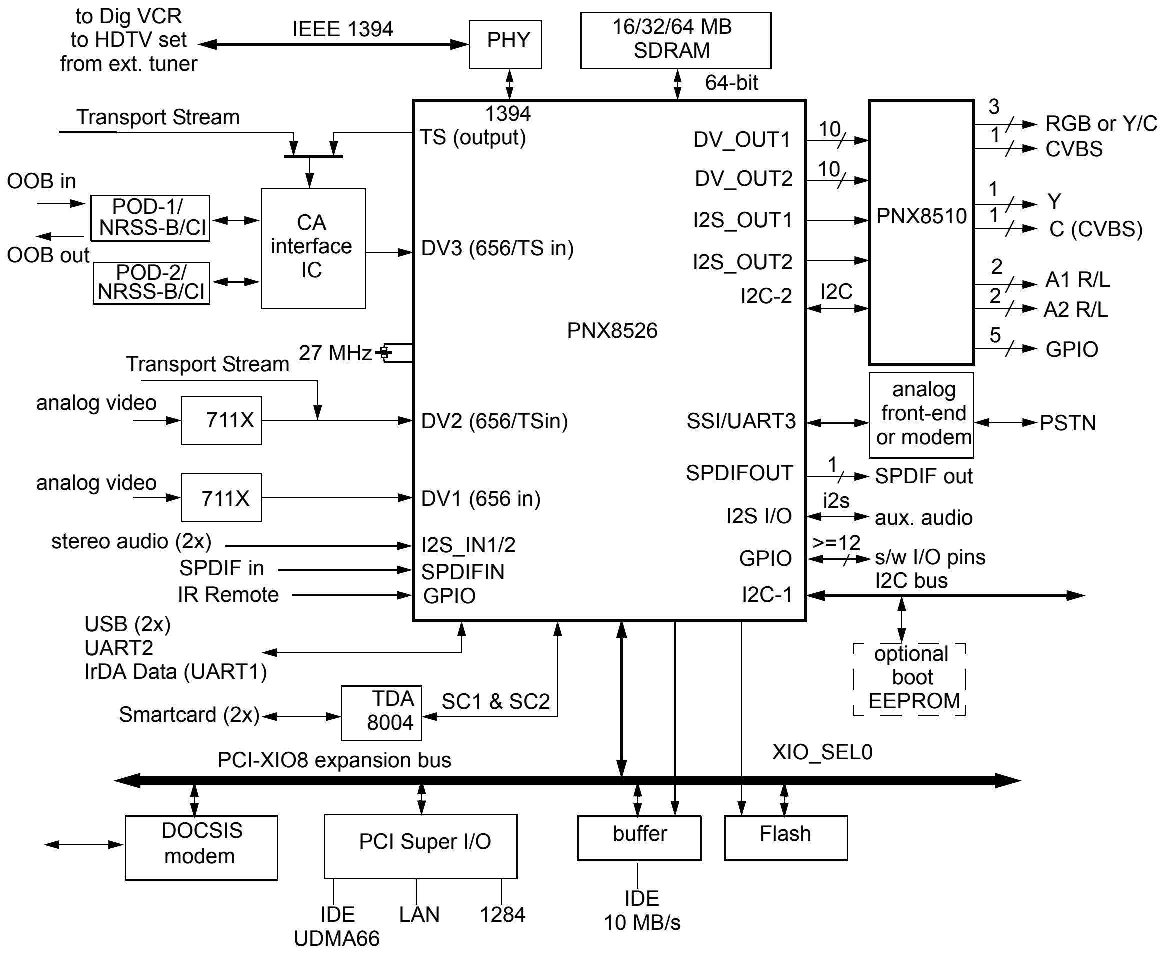


Figure 1: PNX8526-Based System Block Diagram

The PNX8526 runs on a single 27 MHz xtal from which all internal and external clocks are derived by on-chip synthesizers. (See Section 1.6.2 for details.) The PNX8526 boots directly from attached Flash memory or ROM. If desired, custom boot methods can be programmed using the optional I^{2}C boot EEPROM.

The PNX8526 has three Digital Video inputs that accept digitized analog video (ITU-656), although only two ITU-656 streams can be processed simultaneously. (See Section 1.6.10) Two of these inputs, DV2 and DV3, can also accept scrambled transport streams.

The DV inputs support parallel transport stream formats. In addition, a single incoming 1394 transport stream is supported. Two selected transport streams can undergo internal de-scrambling and decoding.

Based on the system implementation, one or both transport streams may pass through Point of Deployment (POD) or Common Interface (CI) conditional access modules before transfer into the PNX8526. Either a single companion IC, such as the SCM Microsystems CIMaX, or two CIMaX chips can be used. In the latter case, it is possible to handle dual decoding no matter which conditional access system is used.

The PNX8526 contains on-chip DVB, MULTI2 and DES hardware de-scramblers, as well as an ICAM verifier. The entitlement system for these de-scramblers is provided via two Smartcard interfaces.

The TM32 CPU does further processing on the result of the transport stream de-mux.

For MPEG2 video, a slice level HL MPEG2 video decoder performs the majority of the MPEG2 algorithm. The TM32 CPU does all MPEG2 processing above the slice level. Two simultaneous SD streams or one HD stream may be processed.

All audio processing is done by the TM32 CPU. Compressed audio will be present in memory from either the transport stream de-multiplex or from the SPDIF input port. The SPDIF input port is intended primarily for DTV applications where a SPDIF source is available from an external source device, such as a DVD player. PCM (stereo sample) audio is present in memory from the I^2S input ports or SPDIF input. Two AC-3 (or equivalent) compressed audio streams may be decoded simultaneously. The TM32 CPU may also process effects, enhancements and mix the audio data. Multi-channel compressed audio or down-mixed stereo PCM audio is transmitted over the SPDIF output interface. Multi-channel audio samples are Dolby Pro Logic down-mixed into the two stereo I^2S interfaces to the PNX8510 companion IC. In addition to the two I^2S inputs and two I^2S outputs, a bi-directional I^2S interface is provided. This allows connection of other audio inputs or outputs—headphones, for example.

Remark: There is not enough computing power to support encoding of multi-channel compressed audio simultaneous with video processing. So the multi-channel compressed audio transmitted over SPDIF must be from one of the original compressed sources.

Graphics rendering may be accomplished with the MIPS or the TM32 CPU by utilizing the 2D Drawing and DMA engine. This engine can perform fast area fills, 3-operand bitblts, monochrome data expansion, and lines. It can also be used as a generic DMA engine to transfer data between memory locations on a byte-aligned basis. An alpha bitblt capability is also provided to allow for anti-aliased text and lines as well as source/destination blending operations.

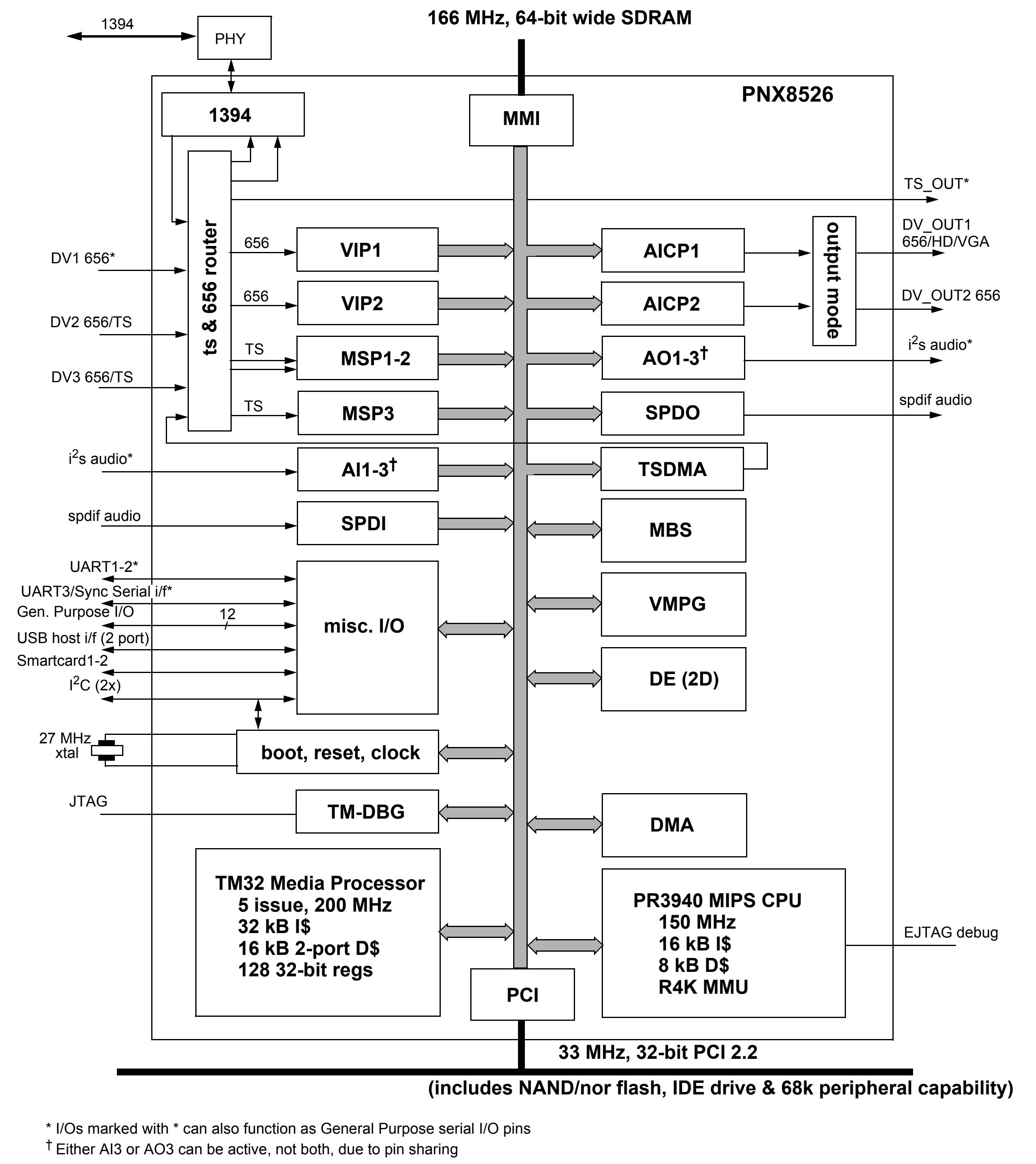
Once all video and graphics data for specific fields or frames has been generated in memory, the video display pipeline starts processing those images for display. The video processing functions include 6-tap horizontal/vertical scaling, anti-flicker filtering, and de-interlacing (when progressive output is required). The processed images are then combined for each output. Up to four surfaces of any supported format may be combined to produce the primary display output. Up to two surfaces are combined to produce the secondary output. Compositing of more surfaces for future video algorithms is possible by using the TM32 CPU and/or the memory based scaler prior to invoking the compositing/display engine. This is subject to CPU and memory bandwidth availability.

The PNX8526 contains a 1394 interface with 5C copy protection. The PNX8526 1394 can simultaneously transmit two transport streams while receiving one transport stream. The transmitted streams can be partial transport streams (created by PID filtering of an input) or one of the two streams can be software generated. In the case of receiving a scrambled 1394 transport stream input, the stream can either use the on-chip de-scramblers, or may be routed to the external companion CA IC for de-scrambling by the POD/CI CA module(s).

The PNX8526 contains a variety of peripheral interfaces to support both ASTB and DTV requirements. There are two Smartcard interfaces, two USB ports, two I^2C ports, one IrDA Data UART and two general purpose UARTs, one of which (UART3) is multiplexed with an SSI interface for soft modem support. The PNX8526 also contains an integrated IDE controller, which only requires an external isolation buffer to implement a full disk interface with sustained speeds up to 10 MB/s. A third-party PCI Super I/O chip may be utilized to provide peripheral functionality not contained on the PNX8526. Functions such as IEEE-1284, 10/100 Ethernet, floppy drive support, UDMA66 IDE controllers and others are currently available in low-cost, commercially available parts.

1.6 PNX8526 Detailed Functional Description

1.6.1 Functional Block Diagram



* I/Os marked with * can also function as General Purpose serial I/O pins
† Either AI3 or AO3 can be active, not both, due to pin sharing

Figure 2: PNX8526 Functional Block Diagram

1.6.2 System Boot

The PNX8526 uses a scripted boot. The hardware Boot module reads a script consisting of simple commands (“write a given value at a given address, delay xxx cycles...”). The choice of script is determined by pullup/pulldown resistors on the three BOOTMODE pins (these pins double as GPIO[2:0] after boot)..

Table 5: BOOTMODE

BootMode[2:0]	Boot Method
000	Use I^2C EEPROM script, fully customized boot
001	Use built-in script 1
010	Use built-in script 2
011	External EEPROM boot script with fast IIC clock, testmode
100	Use built-in script 3: fast boot script used for manufacturing
Others	Reserved

Built-in script 1 initializes the system and starts the MIPS executing from a direct addressable memory (ROM, nor-Flash) attached to PCI-XIO. The initial clocks are chosen conservatively, and the MIPS is responsible for completing the full-speed initialization.

Built-in script 2 initializes the system, copies part of an attached NAND-Flash to SDRAM, and starts MIPS execution in SDRAM.

The I^2C EEPROM allows a fully customized boot method, and is typically used for:

- External host configurations - the boot script initializes the system and sets personality data in the PCI module (subsystem vendor ID, etc.). This prepares the PNX8526 for the host enumeration of all PCI devices. The PNX8526 then awaits external host configuration, TM32 program download and start.
- Custom boot, for example over an attached LAN, or over another device capable of providing the initial SDRAM boot image
- Boot with special SDRAM configurations not supported by script1 and 2

The PNX8526 on-chip MIPS is capable of direct ROM or standard Flash execution to allow for booting.

Remark: Direct execution from NAND-Flash is not supported.

Direct execution, however, has limited performance. Therefore, the first thing the MIPS typically does is to copy a file contained in Flash to system SDRAM and execute it at high speed. That Flash file contains the self-decompressing initial system software application. This multi-stage boot process minimizes system memory cost.

The scripted boot, in combination with an appropriately programmed I^2C EEPROM, allows booting in many ways, including direct TM32 CPU boot from SDRAM.

A standalone PNX8526 system is able to reliably update its own Flash boot image, whether the Flash is standard or NAND-Flash. In most systems this is done by extra Flash storage capacity that is used by the Flash update software to guarantee atomicity of a boot image update under power failure. The update either succeeds or the old boot image is retained. In some systems, it may be cost attractive to use a medium sized boot I^2C EEPROM instead. This boot EEPROM holds the code to recover a corrupted Flash from some system resource, such as a cable feed.

Boot is fully explained in Chapter 3 Boot.

1.6.3 TM32 CPU Core Processor

The TM32 CPU is a 0.12 micron core version of the TriMedia 3200 media processor CPU. This CPU core is a 200 MHz, 5 instructions per clock cycle, Very Long Instruction Word (VLIW) processor, with an extensive set of multimedia instructions. It implements the TriMedia 1300 instruction set. The TM32 CPU supports 32-bit integer and IEEE-compatible 32-bit floating point data formats. It also provides a Single Instruction Multiple Data (SIMD) style operation set for operating on dual 16-bit or quad 8-bit packed data. It has a peak floating point compute capacity of 1 G operations/s, and has 800 M multiply-add/s capability on 16-bit data. Its on-chip dual access 16 kB 8-way set-associative data cache provides a CPU local data bandwidth of 1.6 GB/s. Its on-chip 32 kB 8-way set-associative instruction cache provides 224 bits of instructions every clock cycle, for an instruction rate of 5.6 GB/s.

In the PNX8526 target operating mode, the TM32 CPU is responsible for one transport stream creation, dual audio decode and control of the HL MPEG2 decoder. The TM32 CPU has sufficient compute performance to deal with a variety of future operating modes. It can decode most SD video compression standard and associated audio at full frame rate, such as decoding a DV camcorder image stream arriving over 1394. It can do intelligent conversion from NTSC video to 60 Hz progressive. It is also capable of doing all audio and video compression, decompression and processing necessary for bi-directional video conferencing.

The TM32 CPU is responsible for all media processing and real-time processing functions within the PNX8526. It runs a small real-time kernel operating system, which allows it to respond efficiently and predictably to real-time events. In some cases, the TM32 CPU will handle a media processing function in conjunction with fixed hardware, such as the HL MPEG2 decoder. Fixed hardware that is associated with media processing exists on a segment of the PI-Bus local to the TM32 CPU. This ensures low latency from the TM32 CPU to fixed hardware devices.

The TM32 CPU executes code from the unified system SDRAM memory.

The TM32 CPU is capable of operating in little or big-endian mode. The mode is chosen shortly after CPU startup by setting the CPU PCSW.

Debug of software running on the TriMedia CPU is performed using an interactive source debugger on a PC with a JTAG plugin board. The PC talks to the TM32 CPU through the PNX8526 JTAG pins and the TM_DBG module (Figure 7) . The TMN_DBG provides an improved version of the TM1x00 JTAG debug port.

1.6.4 PR3940 MIPS RISC Core

The MIPS PR3940 core is a 150 MHz, 195 MIPS (Dhrystone 2.1) class processor that implements the MIPS-II and MIPS-16 ISA. It has a built-in R4000 TLB-based MMU, and splits 16 kB Instruction and 8 kB Data caches with 2-way and 4-way set associativity. The PR3940 fully implements kernel/user mode style system protection and memory address translation and access control. This allows the PNX8526 to run mature operating systems with inter-process protection. The PR3940 is an integer-only core that uses traps and emulation to support floating point instructions.

This processor is responsible for running the operating system and control functions within the system. It also serves to execute Conditional Access software modules that are only available as MIPS ISA binaries.

Devices associated with control functions exist on a segment of the PI-Bus close to the PR3940. This ensures low latency from the PR3940.

The PR3940 core executes code from the unified system SDRAM memory. It can also execute directly from standard Flash or ROM, but with reduced performance. Direct execution from NAND-Flash is not supported.

The PR3940 core is capable of operating in little or big-endian mode. The mode is chosen at MIPS reset time. The PNX8526 has provisions to deploy Flash updates that change the endianness of operation of the software stack. This is implemented by a MIPS self-reset during the application boot process.

Debug of software running on the MIPS core uses an industry standard EJTAG port with a dedicated set of pins.

1.6.5 OS Security

By default, both CPUs in the PNX8526 have access to all of system memory and to all on-chip devices. By utilizing the MIPS TLB, it is possible to protect the OS kernel from corruption by user-mode processes, and to run user-mode device driver processes that are protected from errant or malignant user-mode processes. In addition, the PNX8526 has special hardware provisions that allow setting aside a private range of memory for the TM32 CPU, and also allows selectively disabling access to devices for the TM32 CPU. This can be used to allow the TM32 CPU software the same protection and access rights as MIPS user-mode processes.

This advanced capability allows application of the PNX8526 in open systems, where user-mode tasks can be downloaded to either CPU without compromising the security of the system. An errant or malignant user-mode task on either CPU is hardware protected from overwriting kernel data structures, other user-mode task data or gaining access to private devices.

1.6.6 2D Drawing and DMA Engine

The PNX8526 includes a 2D rendering and DMA engine. The 2D Draw engine performs high speed graphics operations, including solid fills, three-operand bitblt, lines, and monochrome data expansion. Supported drawing formats include 8, 16, and 32 bits per pixel. Monochrome data can be color expanded to any supported pixel format. Anti-aliased lines and fonts are supported via a 16-level alpha blend bitblt. A full 256-level alpha bitblt is available to blend source and destination images together. Drawing is supported to any naturally aligned memory location and at any naturally aligned image stride, i.e. 16 and 32-bit pixels should be allocated at byte addresses that are a multiple of 2 or 4, respectively.

1.6.7 HL MPEG2 Decoder (VMPG)

The high-level MPEG2 video decoder does all portions of the MPEG2 algorithm below slice level. The TM32 CPU processes above the slice level. All 18 ATSC formats may be decoded. Additionally, multiple SD streams may be processed simultaneously.

1.6.8 Video Processing and Display (MBS and AICP)

Images residing in the system DRAM may require further processing after being decoded (MPEG), captured (analog video), or rendered (graphics). There is dedicated hardware to perform common functions, such as scaling, anti-flicker filtering, format conversions and de-interlacing. The TM32 CPU may also be utilized for video processing algorithms that are not implemented in hardware. The memory based scaler (MBS) module performs video processing operations with memory images as source and destination. The two advanced image composition processors (AICPs) then perform the final processing, including color-space conversion, and combine the images to produce the final display output on the primary and secondary channels.

The on-chip hardware image processing modules all use the same “native” pixel formats, as shown in Table 6. This ensures that image data produced by one module can be read by another.

Remark: The MBS supports all formats on input, so it can be used for format conversion coincident with a scaling operation.

The MBS contains a universal pixel format converter, such that software written to paint pixels in a specific packed RGB or YUV image format can be ported without modification.

Table 6: Native Pixel Format Summary

Name	Note	VIP Out	MPG Out	MBS In	MBS Out	2D Draw*	AICP In
1 bpp indexed	CLUT entry = 24-bit color + 8-bit alpha			x			x
2 bpp indexed				x			x
4 bpp indexed				x			x
8 bpp indexed				x		x	x
RGBa 4444	16-bit unit, containing one pixel with alpha	**		x	x	x	x
RGBa 4534		**		x	x	x	x
RGB 565	16-bit unit, containing one pixel, no alpha	**		x	x	x	x
RGBa 8888	32-bit unit, containing one pixel with alpha	**		x	x	x	x
packed YUVa 4:4:4	32-bit unit containing one pixel with alpha	x		x	x	x	x
packed YUV 4:2:2 (UYVY)	16-bit unit, two successive units contain two horizontally adjacent pixels, no alpha	x		x	x		x
packed YUV 4:2:2 (YUY2, 2vuy)		x		x	x		x
planar YUV 4:2:2	Three arrays, one for each component	x		x	x		
semi-planar YUV 4:2:2	Two arrays, one with all Ys, one with U and Vs	x		x	x		

Table 6: Native Pixel Format Summary *…Continued*

Name	Note	VIP Out	MPG Out	MBS In	MBS Out	2D Draw*	AICP In
planar YUV 4:2:0	Three arrays, one for each component			x	x		
semi-planar YUV 4:2:0	Two arrays, one with all Ys, one with U and Vs		x	x	x		

* Shown are the 2D Engine frame buffer formats where drawing, raster ops and alpha-blending of surfaces can be accelerated. Additionally, the 2D Drawing Engine host port supports 1 bpp monochrome font/pattern data, and 4 and 8-bit alpha only data for host-initiated, anti-aliased drawing.
** The VIP is capable of producing RGB formats, but not when performing horizontal scaling.

1.6.8.1 Memory Based Scaler (MBS)

Processing of images in the main memory may be performed with the use of a Memory Based Scaler (MBS). As the name implies, the primary function of the MBS is to read images from memory, perform a scaling operation, and write the resultant image back into memory. This method has a number of advantages over an “on-the-fly” or real-time approach. Perhaps the biggest advantage is that the scaling operation is de-coupled from the display operation. In the case of a PIP, a real-time approach implies that a huge amount of memory bandwidth must be concentrated in a very small amount of time while the rest of the time the scaler remains idle. The MBS has the ability to process many images in the time it takes to display a single frame. Because the MBS is not a real-time device, it utilizes any available memory bandwidth that is not claimed by higher priority PNX8526 resources. This helps with averaging out memory access bandwidth.

In addition to its basic scaling capabilities, the MBS performs linear and non-linear aspect ratio conversion (including panorama and amaronap modes), anti-flicker filtering, de-interlacing and pixel format conversion functions.

The video processing functions are based on 4 and 6-tap polyphase filters with up to 64 phases. Three 6-tap filter units are used for horizontal scaling/filtering while three 4-tap filter units are assigned to vertical scaling/filtering. For some video formats (e.g. YUV 4:2:x) the three 4-tap filters can be combined to work as two 6-tap filters.

The MBS contains simple median filters to support de-interlacing in combination with other operations. Alternately, a special 2x32 phase mode allows higher quality inter-field filtering style de-interlacing by using different polyphase coefficient sets for every other output line.

Although a video processing task is constrained to 1024 pixels horizontally on either input or output, any arbitrary input to output size conversion can be achieved by processing a video source within multiple scale tasks. The MBS can continue processing a new task where a previous task left off by programming the start phase of the new task according to the phase where the previous task ended.

The processing speed of the MBS depends on available memory bandwidth, but it has a theoretical upper limit of 120 million pixels per second. Depending on compression or expansion ratio, this rate can only be achieved on either input or output.

The MBS supports a wide variety of input and output pixel data formats. On input, it can handle any of the native PNX8526 pixel formats. Input also has a universal pixel converter, where any non-standard (but packed) RGB or YUV format can be converted prior to operation. Input formats include indexed pixels with color depths of 1, 2, 4, and 8 bits. An internal LUT is used to convert the index 1, 2, 4, and 8-bit formats into true color, plus an optional alpha value. The output format is independent of the input format. Output formats include any of the native PNX8526 pixel formats, excluding the indexed color formats.

1.6.8.2 Advanced Image Composition Processors (AICP)

The two AICPs combine images from main memory into composited images and refresh the primary and secondary display channel. All compositing is done either in RGB or YUV color space, depending on the AICP output mode. Each layer performs color space conversion if the memory format it receives differs from the output format. The overall architecture of the AICP is shown in Figure 3

The Primary AICP composites up to four image layers. The secondary AICP supports two layers. The AICPs can allocate layers in a general way to each of the display channels.

A layer is responsible for producing a valid pixel for every display coordinate. All of the layers are identical, so there are no restrictions as to how each layer may be utilized. For example, the primary display can have four video layers, four graphics layers, three graphics + one video layer, etc. A wide variety of RGB, YUV, and alpha formats are supported as input to a layer module. The layer structure is described in more detail in Section 1.6.8.3.

Remark: “Region based graphics” are not supported at the AICP hardware level; software must generate one uniform color depth surface if an application requires region based graphics.

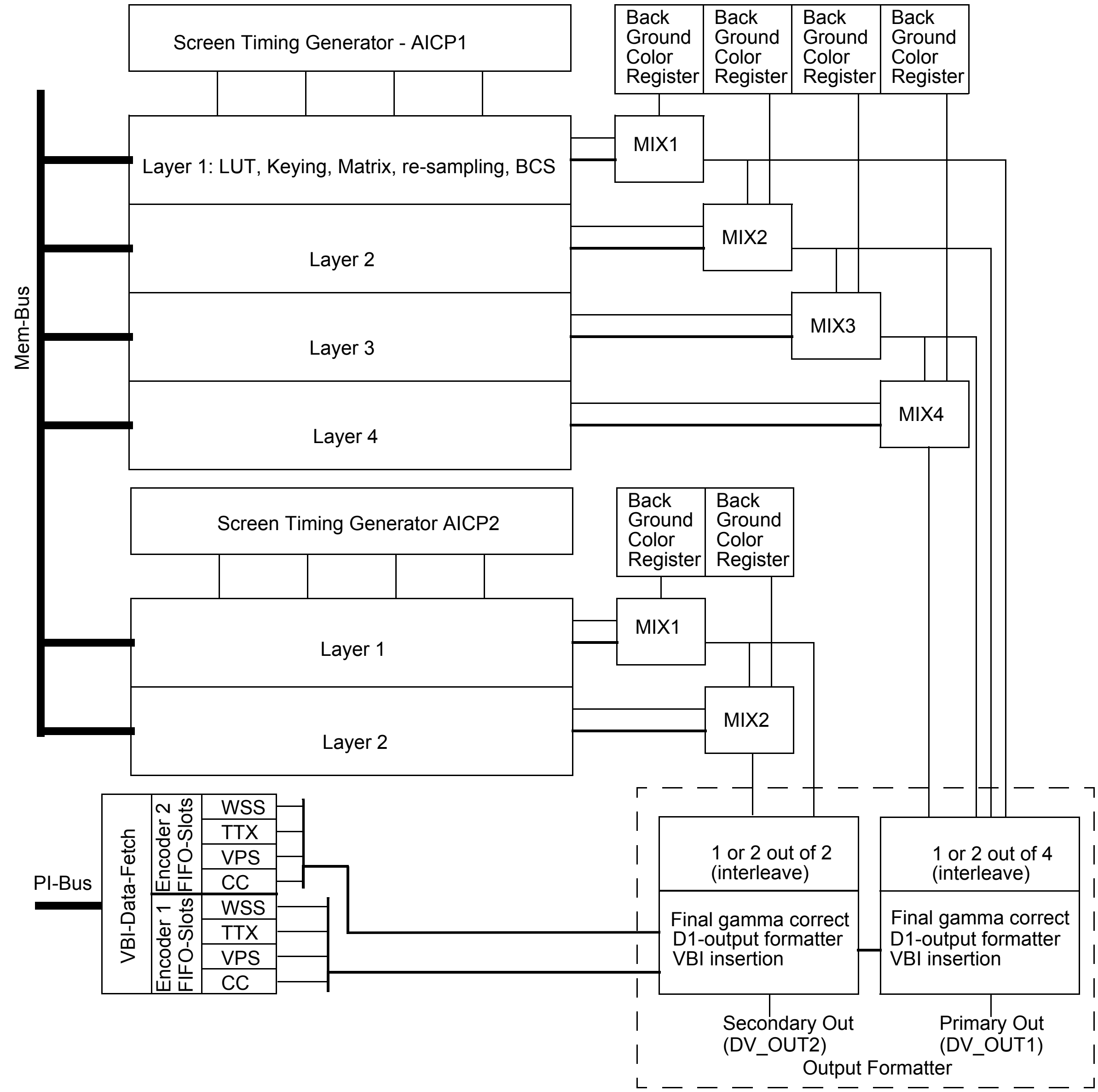

Screen Timing Generator - AICP1
Back Ground Color Register
Back Ground Color Register
Back Ground Color Register
Back Ground Color Register
Layer 1: LUT, Keying, Matrix, re-sampling, BCS
MIX1
Layer 2
MIX2
Layer 3
MIX3
Layer 4
MIX4
Mem-Bus
Screen Timing Generator AICP2
Back Ground Color Register
Back Ground Color Register
Layer 1
MIX1
Layer 2
MIX2
PI-Bus
VBI-Data-Fetch
Encoder 2 FIFO-Slots
WSS
TTX
VPS
CC
Encoder 1 FIFO-Slots
WSS
TTX
VPS
CC
1 or 2 out of 2 (interleave)
Final gamma correct D1-output formatter VBI insertion
1 or 2 out of 4 (interleave)
Final gamma correct D1-output formatter VBI insertion
Secondary Out (DV_OUT2)
Primary Out (DV_OUT1)
Output Formatter

Figure 3: AICP1 and AICP2 Architecture Overview

The properly formatted pixels from each layer are combined back-to-front in a cascaded series of mixer units. There is one mixer associated with each layer of the AICP. Compositing functionality is highly programmable and includes chroma, alpha or color key based alpha-blending, and pixel selection from the previous layer, the current layer or the background color. Alpha-blending can either use a per-pixel alpha or global surface alpha. A mode such as PIP is simple to implement by setting layer_N for full screen video and layer_N+1 to the PIP. PIP size and position may be changed on a frame-by-frame basis. Effects such as blending a PIP in and out of the full screen video are easy to achieve using the 256 level alpha blend capability of each mixer. Details of the AICP mixer are described in Section 1.6.8.4

A special output formatter makes it possible to take any combination of two output streams allocated from the image layers and interleave them into a data stream running at twice the frequency. With the two AICPs in the PNX8526 a maximum of four different image streams can be generated. It should be noted that two streams originating from the same AICP are fully pixel/line synchronous and must hence have the same resolution and refresh rate. Any resolution/mode can be run in this multiplexed fashion, subject to pin speed limitations. (The 2Fh or 480P 4:4:4 modes can not be interleaved.) Each interleaved AICP channel needs a private PNX8510 companion chip (see Section 1.6.9).

Each AICP layer can read images in any of the native pixel formats, excluding the planar or semi-planar formats, which require MBS preprocessing.

Video arithmetic precision is as follows:

- 8 bits per component for memory image data
- 8 bits per component after layer CLUT
- 9 bits per component after layer color space conversion
- 9 bits per component compositing
- 10 bits per component after final gamma look-up

The AICP operates in master mode. In master mode, the AICP generates all video clocks and synchronization timing.

The primary and secondary AICP each have separate synthesizers for pixel-clock generation. Software uses these synthesizers to achieve perfect lock to the transmission source of the digital video that is being displayed by the AICP. In a view/record scenario, where the viewed channel differs from the recorded channel, neither will ever frame-skip. If a channel carries a PIP that is not transmission synchronous, it may occasionally frame-skip to correct for transmission drift.

1.6.8.3 AICP Layer Structure

An AICP layer contains all the necessary functions to process and handle a single image surface. It is the layers responsibility to produce a constant pixel stream for an image stored in the frame buffer based on the programmed pixel format, the layer position within the overall screen and the required pixel manipulation operations. The layer has a generic, highly programmable structure that allows it to process a wide variety of different images such as live video material and computer generated graphic contents. The operation mode of a layer mostly depends on the incoming pixel data format and the selected output mode of the AICP. The target should always be to perform as few conversions as possible to a specific image in order to preserve picture quality. The layer control unit coordinates the size and location of a particular surface. It signals the pixel formatter block when to start and to stop the generation of an active pixel stream on a line-by-line basis. The layer control unit also provides a very simple error recovery mechanism in case of a FIFO underflow, which ensures correct restart of the next display field.

The layer register array contains all necessary, layer specific programming registers. It is accessible as part of the AICP memory mapped register space.

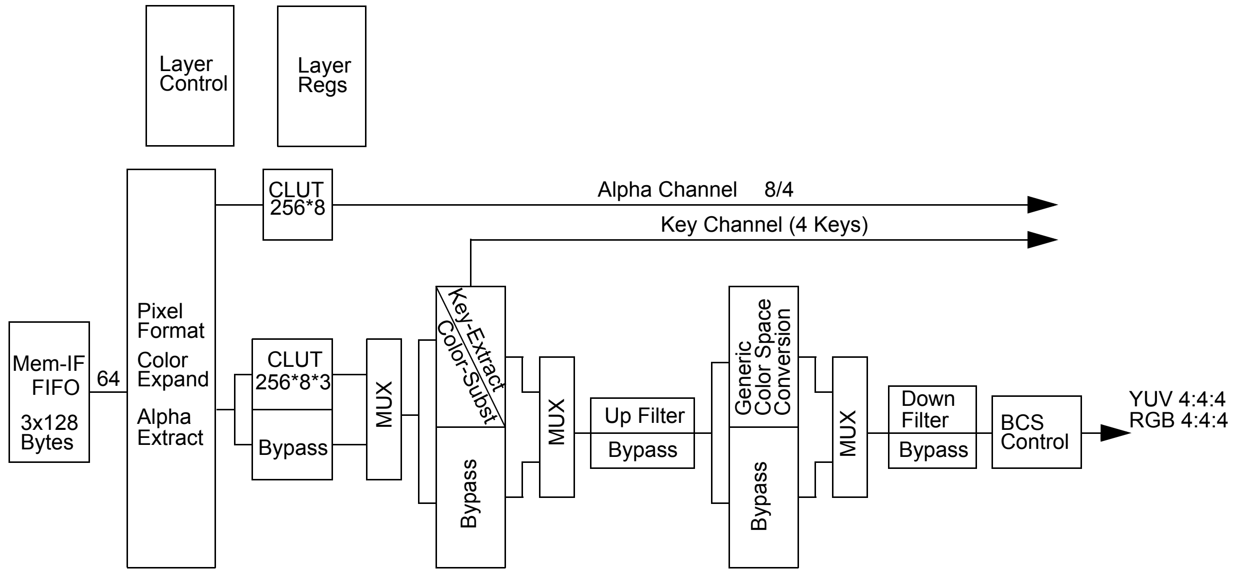


Figure 4: AICP Layer Block Diagram

The pixel formatter fetches the raw pixel data from the layer FIFO and splits it up into the color and potential alpha components. The pixel formatter follows a generic approach which makes it possible to adopt any future pixel formats as long as the total number of bits per pixel does not exceed 32. The pixel formatter is also responsible for color expansion and “4:2:2 to pseudo 4:4:4” conversion to simplify the processing of the data stream.

Following the pixel formatter, a set of 256x8-bit look-up tables is placed in the pixel data path. These tables are independent from each other, they do not share the address lines. With this approach a pre-gamma correction on component basis as well as the expansion of indexed colors is possible.

The color key block is placed after the look-up tables because it has to operate on expanded but unfiltered data. The block generates a set of up to four color keys used to indicate a match to a set of up to four pre-specified colors or color ranges. The color matching logic is highly programmable, resulting in a flexible scheme which uses masks and raster operations on a color component basis. Furthermore a key color exchange mode is implemented to reduce visible artifacts introduced through filtering of images containing a color key.

Connected to the color key unit, an up-sampling filter converts co-sited or interspersed 4:2:2 data formats into 4:4:4 formats.

The layer also contains a universal color space matrix with a set of programmable coefficients to assure that all possible YUV and RGB data format conversions can be accomplished. This approach supports conversion between any flavor of RGB to YUV, YUV to RGB as well as YUV to YUV.

The color space conversion results in a data path expansion to 9 bits. All processing after this uses 9 bits per component.

The down filter unit prepares the pixel stream to be encoded with a standard digital video encoder requiring 4:2:2 YUV data formats. Due to the reduced chrominance bandwidth available in 4:2:2 formats, the signal needs to be band-limited to avoid artifacts.

The layer data path is concluded with a block that controls the brightness, contrast and saturation of an image layer. The block can be used to equalize layer-specific image properties before mixing.

1.6.8.4 AICP Mixer Structure

Each AICP layer provides the pixel stream corresponding to its image. The AICP mixers, shown in Figure 5, compose all layers back-to-front to produce an output picture. In general, a mixer takes two pixel streams, one from the previous layer, another from the current layer. Both pixel streams have specific mixing information associated with each input pixel. That means, in detail, every pixel comes with an alpha, four color keys, a previous key, a new pixel key1 and a new pixel key2 information. By programming multiple raster operations (ROPs), software can control whether a pixel is:

- Shown from the current or the previous layer
- Alpha blended (result is a blend of the current and previous layers)
- And/or inverted (inversion of the previous layer)

The inputs to those raster operations are mask/key information and pixel key information bits. Whenever an input key specification matches a key pattern of a pixel, the output of the ROP is active. All pixel modification features are controlled by those active signals. Other ROPs are implemented to decide which pixel information (color keys, alpha, key1/key2, previous key) is passed out with the newly assembled/ blended/modified pixel. By using the output of one mixer as input to the next mixer, software can generate a hierarchically composited picture.

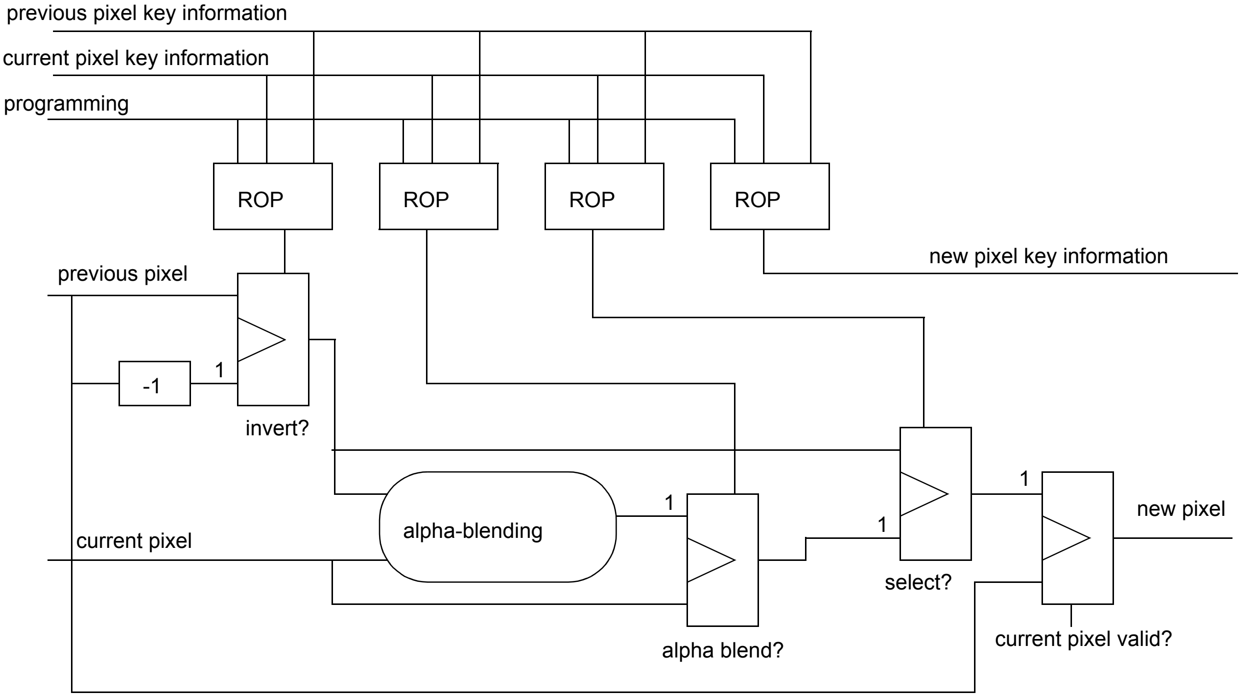


Figure 5: AICP Mixer Block Diagram

1.6.9 PNX8510 Analog Companion Chip

The PNX8510 is a companion IC that provides the final, analog stage in the video and audio pipelines.

The PNX8510 has two major interfaces: video and audio. The video interface consists of two 10-bit D1 ports. There are two integrated digital encoders (DENCs) that support NTSC, PAL, or SECAM output. Six 10-bit video DACs support two output channels. The first channel can support SCART output with four DACs. The second channel supports only composite or S-Video using two DACs. The D1 ports support interleaved modes so that the PNX8526 controller can use one PNX8510 companion to drive two display channels.

The PNX8510 supports audio via two I^2S interfaces. There are four audio DACs to support two stereo channel outputs. Control of the PNX8510 is achieved via an I^2C or the D1-VBI interface. Full details of the PNX8510 are provided in the PNX8510 analog Companion Chip Data Sheet.

1.6.10 Video Input Processor (VIP)

The PNX8526 accepts up to two simultaneous video input streams from analog video sources digitized by Philips 711x series video decoders. A video stream may be stored anywhere in unified memory. It is possible to store the video data as separate fields or as a single frame. The video data can be scaled horizontally (using a high quality 6-tap polyphase or transposed polyphase filter). The VIP supports a variety of output pixel formats, including RGB and YUV4:2:2 formats, as shown in Figure 6. The

VIP performs error feedback rounding if programmed to emit a YUV memory format with less than 8 bits per component. Contrast, Brightness and Saturation control must be performed in the 711x decoder chip.

The primary use of each VIP is for 13.5 Mpix/s (27 MHz byte rate) digital video. Higher data rates, up to a 40-MHz pixel rate are supported only for specific scenarios that do not exceed the system bandwidth budget.

To reduce the amount of video data written to memory the VIP can compress video lines horizontally using a 6-tap polyphase scaler. This scaler can run in both normal and transposed polyphase modes. Vertical scaling, mirroring or de-interlacing requires use of the MBS.

VBI data may be captured from the incoming stream and placed in unified memory for re-insertion by the video DENC(s) and/or software processing. VBI data extraction is done via definition of either a separate VBI capture window or by processing ANC header codes.

The VIP supports a raw data-streaming input mode similar to the TM1x00 line of media processors. This mode is suitable for handling future 8 to 10-bit data formats.

1.6.11 Transport Stream Routing and MSP

Three MPEG System Processors (MSPs) are the primary point of entry for transport streams. A transport stream routing network is associated with the three MSP modules. See Figure 6 for a diagram of all transport stream routing capabilities.

The PNX8526 supports decoding of up to two simultaneous input transport streams, selected from a variety of sources, including one received over 1394, and up to two from the DV2 and DV3 inputs. Parallel only transport stream formats are supported on the DV inputs. The DV2 and DV3 inputs each can receive a single parallel format transport stream.

Internal de-scramblers are provided for both selected transport streams. Transport streams may be passed through external Point of Deployment (POD) or Common Interface (CI) conditional access modules before delivery to the PNX8526. This requires an external POD/CI interface chip, such as a CIMaX or similar. Refer to block diagram Figure 1.

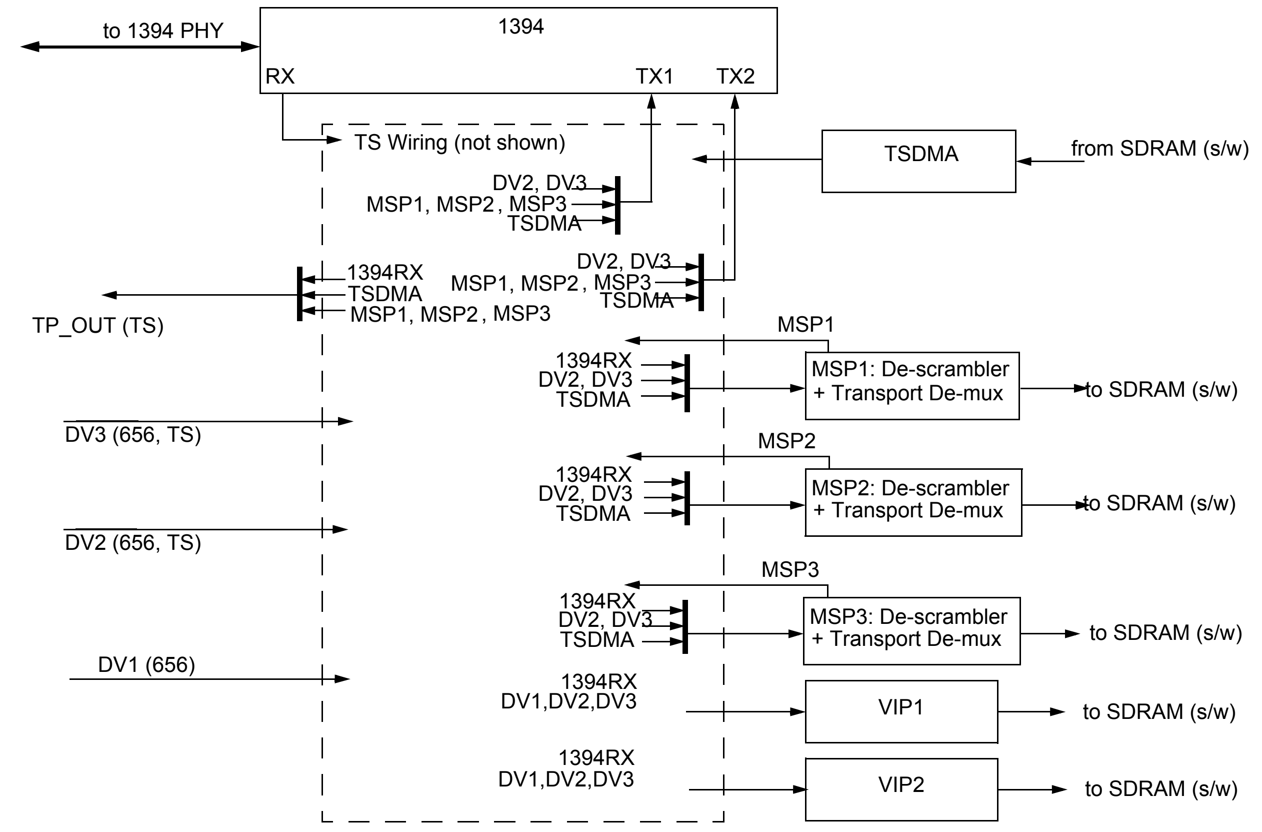


Figure 6: Transport Stream, Digital Video Input and 1394 System Connections

All MSPs are capable of parsing MPEG2 and DIRECTV transport streams, including de-scrambling, de-multiplexing and routing of appropriate data to memory. They are compliant with DVB, DES, ICAM and MULTI2 de-scrambler standards. The MSP block is also capable of doing the DES ECB copy protection decoding used by ATSC or OpenCable™ external conditional access modules. These functions are mutually exclusive—each MSP can only perform one de-scramble operation at a time.

Each MSP performs PID filtering and timestamping, de-scrambling, de-multiplexing and section-filtering on a transport stream, and DMAs the resulting data to system memory. Once the transport streams have been transferred to unified memory, the TM32 CPU processes the audio, video and other streaming data types. The TM32 CPU also performs clock recovery based on the transport stream timestamps.

Remark: It is possible to selectively bypass the hard-wired MSP processing such as built-in section filter and transport de-multiplexer and perform these functions in TM32 CPU software.

For use as a transport stream transmitted across 1394, the MSP blocks are also capable of providing a partial transport stream derived from PID filtering on the input. Simple software program map table insertion capability is provided.

For a more complicated transport stream, TM32 CPU software can generate one arbitrary transport stream through the TSDMA block. This stream has programmable packet length, clock and inter-packet gap. The source is dual DMA buffers used in a ping-pong fashion.

The TM32 software or MSP hardware-generated transport streams can be transmitted across the 1394 or the TS out pins. The software-generated transport stream can also be sent into each MSP for diagnostic purposes.

1.6.12 Main Memory Interface (MMI)

A 64-bit, 166 MHz SDRAM interface is provided. This interface can support memory footprints of 16, 32, or 64 MB. The following table illustrates the supported memory configurations:

Table 7: Memory Configurations

Total Amount of SDRAM	Implemented	Number of Memory Chips
16 MB	2M X 32 (64 Mbit)	2
32 MB	4M X 16 (64 Mbit)	4
	4M X 32 (128 Mbit)	2
64 MB	8M X 16 (128 Mbit)	4
	8M X 32 (256 Mbit)	2

The memory interface generates the required SDRAM protocol, but isolates internal PNX8526 resources from this protocol by using the 64-bit DVP memory protocol.

The memory interface also arbitrates on the memory highway, guaranteeing adequate bandwidth and latency to internal resources. The PNX8526 memory arbiter uses a programmable cyclical list based arbiter, with the ability to give all unused list slots to the TM32 CPU and/or PR3940 MIPS CPU. This arbiter guarantees required latency to all real-time agents, while optimizing CPU performance.

1.6.13 Conditional Access Interfaces

The PNX8526 supports both proprietary internal de-scrambling algorithms as well as external replaceable conditional access modules. Internal de-scramblers are provided for DVB, MULTI2 and DES. An ICAM hardware verifier is included. The external conditional access modules that are supported include:

- DVB's Common Interface
- OpenCable's Point of Deployment (POD) interface
- National Renewable Security System (NRSS-B) interface

These standards are all based on a variant of the 16-bit PC card interface. It is assumed that one or more separate companion ICs, such as the SCM Microsystems CIMaX will be used to interface between the POD/CI modules and the PNX8526 IC. The companion ICs will connect as peripherals to the PNX8526 expansion bus for CPU access. These companion ICs will deliver a cleartext or copy-protected transport stream to one or both of the PNX8526 transport stream inputs. Copy protection is undone by the on-chip MSP modules, which support the Opencable and ATSC (NRSS-B) DES ECB copy protection decoding. Copy protection decoding for DVB is currently not supported.

Authentication for copy protection of conditional access modules is performed in software.

1.6.14 Audio Interfaces (AI, AO, SPDI and SPDO)

The PNX8526 contains three Audio Input and three Audio Output modules. They are connected to provide the following functionality:

- Two I^2S stereo input ports (I2S_IN1/2). These ports are used in association with the two analog video input streams. The I^2S data is transferred via DMA to the unified memory array. The stereo audio inputs support up to 16-bit samples at rates up to 96 kHz.
- Two dedicated stereo I^2S outputs that interface with the companion analog IC (PNX8510) for two independent stereo audio programs. The stereo audio outputs support up to 32-bit samples at rates up to 96 kHz.
- A third I^2S port that can be configured as either stereo input or as an 8-channel output. This port outputs up to eight sample-rate synchronous audio channels over four I^2S data wires with common control wires. This port has 16-bit input sample precision and up to 32-bit output sample precision at rates up to 96 kHz.

The SPDI/O modules provide SPDIF (Sony Philips Digital Interface) input and output capability as follows:

- A SPDIF output with IEC-1937 capabilities. Transmitted data is generated by TM32 CPU software. This output port can carry either stereo PCM samples from an internal audio mix, or one of the originally received compressed audio programs (5.1 channel AC-3, multi-channel MPEG audio). No transcoding of audio is provided in the normal system operation compute budget. The sample rate of transmitted audio is controlled by software, allowing perfect synchronization to the broadcast audio source.
- A SPDIF input to connect to external sources, such as a DVD player. The incoming data is timestamped and written to unified system memory. Data interpretation and sample rate recovery is by software on the TM32 CPU. The audio data received can be in a variety of formats, such as stereo PCM data, 5.1 channel AC-3 data per IEC-1937 or other. Software-decoded audio can be used for mixing with other audio for output along one of the audio outputs.

1.6.15 1394 Interfaces

The PNX8526 includes a 1394 link layer controller and provides a PHY-LINK interface for interfacing with an external 1394 or 1394.a compliant physical layer device running at 400, 200 or 100 Mbits/s. A typical use of the 1394 interface will be to pass through HD tuner programs to an external HDTV set, to send a single program transport stream to a Digital VCR and to receive time-shifted programs from an external Digital VCR. The PNX8526 supports the following enhancements to the 1394 link layer required for Set Top Box applications:

- IEC61883 International Standard of Digital Interface
- Copy Protection based on 5C

The 1394 on the PNX8526 can simultaneously send two transport streams across the 1394, while receiving one transport stream. The transport stream routing to/from 1394 is shown in Figure 6. Transmitted transport streams can be any of the system input transport streams, streams generated by MSP hardware (by PID filtering), or a transport stream created by TM32 software and emitted by TSDMA (one only).

The transport stream received across the 1394 can be used as one of the two transport streams de-scrambled and decoded by the PNX8526. Alternately, the PNX8526 TSOUT pins allow the 1394-received transport stream to run through external conditional access hardware and re-enter the PNX8526 as a cleartext transport stream.

The authentication needed for 1394 5C copy protection is performed in software on the TM32 CPU core.

1.6.16 Utility DMA Controller and CRC Checker

The PNX8526 provides a four-channel scatter/gather type DMA controller, which can move data anywhere in system physical address space—to/from DRAM, to/from XIO or PCI (areas mapped in the PCI controller). It uses a linked-list DMA control data structure. It can also be used to compute a Cyclical Redundancy Check (CRC) over a range of memory.

1.6.17 PCI Bus Interface Unit

The PNX8526 contains an expansion bus interface unit ‘PCI-XIO8’ that allows easy connection of a variety of board-level memory components and peripherals. The bus interface is a single set of pins that allows simultaneous connection of 32-bit PCI master/slave devices as well as 8-bit microprocessor slave peripherals and standard or disk-type (NAND) Flash memory. In addition, with external isolation buffers, a full PIO mode IDE interface with sustained speeds of 10 MB/s is provided.

The bus interface unit contains a built-in DMA unit that can move blocks of data from a peripheral to or from the PNX8526 SDRAM. The DMA unit can access PCI as well as 8-bit wide XIO devices. The DMA unit packs XIO 8-bit device data to/from 32-bit words, so that no CPU involvement is required to pre or post-process the data. The IDE interface has its own independent DMA controller to write to/read from SDRAM.

1.6.17.1 PCI Capabilities

The PNX8526 complies with Revision 2.2 of the PCI Bus Specification, and operates as a 32-bit PCI master/target at 33 MHz.

The PNX8526 as PCI master allows either of its CPUs to generate all single cycle PCI transaction types, including memory cycles, I/O cycles, configuration cycles and interrupt acknowledge cycles. As a PCI target, the PNX8526 responds to memory transactions and configuration type cycles, not to I/O cycles.

PCI clock is an input to the PNX8526, but the PNX8526 also generates a PCI drive-compliant 33 MHz clock, which can be wired to be used as the PCI clock for the entire system.

1.6.17.2 Simple Peripheral Capabilities (XIO8)

The 8-bit microprocessor peripheral interface is a master-only interface, and provides non-multiplexed address and data lines. A total of 26 address bits are provided, as well as a bi-directional 8-bit data bus. Up to 64 MB of peripheral address space is allowed. The interface control signals are compatible with a Motorola® 68360 bus interface, and support both fixed wait-state or dynamic completion acknowledgement. The PNX8526 as bus master only transfers 8-bit quantities, however, using a valid MC68360 subset.

Three pre-decoded chip select pins are available to accommodate typical outside slave configurations with minimal or no external glue logic. Each chip select pin has an associated programmable address range within the 64-MB peripheral space. Each chip select pin can also obey external DTACK completion signaling, or be set to have known wait cycles.

The peripheral interface derives 24 of the 26 address wires and all 8 data wires from the PCI AD[31:0] pins. An XIO access appears as a valid PCI transaction to PCI master/targets on the same wires.

Standard random-access Flash memory connects as an 8-bit peripheral. NAND-Flash, sometimes called “disk flash,” has private pins to act as control signals and uses the same eight data wires.

The table below summarizes extension capabilities of the bus interface unit.

Table 8: PCI-XIO8 Bus Interface Unit Capabilities

External Device	Device Type	Capabilities
External PCI Master	32-Bit, 33 MHz PCI Masters	Glueless connection supported for up to three external PCI masters. Additional external masters can be supported with external arbitration. External PCI bus masters can perform high bandwidth, fairly low latency DMA into and out of PNX8526 SDRAM. Large block transfer capable devices can sustain up to 100 MB/s into SDRAM.
External PCI Slave	32-Bit, 33 MHz PCI Targets	Glueless connection supported for multiple devices subject only to capacitive loading constraints. MIPS CPU can perform low-latency up to 32-bit writes and reads to/from PCI targets. DMA masters on the MIPS PI-Bus can do DMA to/from PCI memory targets.
External 8 Bit Slave	8-Bit Wide, De-muxed Address/Data Devices on XIO Bus	Up to three devices supported gluelessly, or unlimited number subject to capacitive loading rules with external address decode logic. MIPS CPU and TriMedia CPU can perform 32-bit or smaller reads and writes to these XIO devices, which are mapped to 8-bit wide transfers by the bus interface unit. DMA masters on PI-Bus can DMA to/from memory connected to XIO.
Standard Flash	8-Bit Wide	PNX8526 provides 3 chip selects, one of which is available for a Flash device. Address range and wait states are programmable. The MIPS CPU can execute or read from Flash. Execution is low performance and only recommended for boot usage. The TM32 CPU can read Flash, if XIO is enabled for TriMedia. The MIPS CPU can reprogram Flash using special software. Flash can be a DMA source for PI-Bus DMA masters. Flash cannot be the target of a PI bus master write—writes require a software flash programming protocol. Using the built-in XIO DMA unit, a bandwidth of 11 MB/s can be obtained from a 70 ns byte-wide Flash device on a 33 MHz PCI bus. Flash is mostly active during system booting, or with low bandwidth, during system operation in order to implement a paging as well as a small non-volatile file system.

Table 8: PCI-XIO8 Bus Interface Unit Capabilities *…Continued*

External Device	Device Type	Capabilities
NAND-Flash	8-Bit Wide	Direct execution, direct PI-Bus read or direct PI-Bus write from this Flash type are not supported. Explicit programmed I/O through special NAND-Flash PCI-XIO8 control/status registers is used to implement a file system on this disk-like Flash type. A minimal file system loader must be resident in I^{2}C boot EEPROM if this is the only non-volatile memory in the system. Using the NAND-Flash XIO provisions, a peak bandwidth of 13 MB/s, and a sustained bandwidth of 11 MB/s can be obtained from a AM30LV0064D 8Mx8 UltraNAND or equivalent Flash device.
CIMaX Device	8-Bit Data, 26-Bit Address	The external logic for conditional access consists of a CIMaX device, with two PCMCIA slot devices and glue logic (373, 245). This entire subsystem behaves like an 8-bit wide slave with up to 26-bit address space. This subsystem interfaces gluelessly to the XIO bus, except for the possible logic needed to combine the DTACK signaling of multiple devices. There is medium bandwidth of communication between CIMaX and the PNX8526.
DOCSIS Devices		Future DOCSIS devices are expected to be PCI bus mastering devices. They connect gluelessly.
External SRAM, ROM, EEPROM	8-Bit Wide	Counts as generic XIO slave device.
External SDRAM	Not Supported	Not supported on PCI-XIO8
External Motorola-Style Masters	Not Supported	The PNX8526 PCI-XIO8 does not support external Motorola-style masters. The PNX8526 assumes that it always is the master over the XIO bus.
External 8-bit DMA devices	Not Supported	Not supported. Use one of the streaming DV inputs or outputs instead.

1.6.18 Peripherals

The PNX8526 supports the required peripherals to build a baseline ASTB or DTV system. Peripheral functionality beyond what is supported in the PNX8526 may be achieved by using commercially available Super IO devices that reside on the PCI bus. The PNX8526 supports the following on-chip peripherals:

- One USB Root Host with two ports, each with individual overcurrent detect and powerdown control. The Root Host complies with revision 1.1 of the USB Specification (Universal Host Controller).
- Two Smartcard UART Interfaces (requires external TDA8004 or similar)
- Two I^{2}C interfaces (multi-master, 100/400 kHz)
- One 2-wire IrDA Data UART (UART1)
- Two 4-wire UART (UART2/3, general purpose, one is muxed with SSI)
- One SSI: a Synchronous Serial Interface to implement soft modem. It is a bus-mastering DMA version. The IOM-2 mode for Siemens® ISDN is *not* supported. The SSI requires an external bit clock source.
- GPIO (General Purpose Software I/O) Pins with IR remote receive capability.

1.6.19 GPIO Pins

The PNX8526 has 12 dedicated GPIO pins (three of these twelve double as boot mode pins with resistor straps, but may be used as GPIO after boot). In addition, 49 other pins (with a high likelihood of not being used in certain systems), are designated as optional GPIO pins. These can operate either in regular mode or in GPIO mode.

For any GPIO pin, the PNX8526 provides the following functionality:

- Individual or group-wise software value and drive on/off assignment (group chosen by mask)
- Fast software read capability
- Signal event sequence timestamping capability
- Signal generation capability

Up to 16 GPIO pins can be designated as monitored. There are 12 “single event” monitors that generate a precise timestamp and interrupt on a single change of a single bit (rising, falling or either). There are four “multi-event” monitors that DMA an event list with timestamps or a sampled data list into memory. Each monitor can sample 1, 2 or 4 GPIO pins at a time. Edge events are timestamped with better than 75 ns resolution. Sampling can be based on a programmable divider of 108 MHz, or on a clock provided by another GPIO pin.

Timestamped event lists can be used to interpret signals with frequencies up to 100-200 kHz, depending on system bandwidth. Sampled lists can be used to interpret signals up to several MHz in frequency.

The event sequence timestamping mechanism can be used for many functions and is particularly useful for interpreting remote control commands, as described below.

Alternatively, the four multi-event monitors can also individually be set as event sequence generators, driven from either a sampled list in memory or a timestamp/ event list in memory. Either 1, 2 or 4 designated GPIO pins can be driven from each generator.

The sequence generation capability is well suited to IR blaster applications.

1.6.20 Remote Control

The PNX8526 uses the GPIO pin event sequence timestamping mechanism or sampling mechanism and software to interpret remote control commands. Event sequence timestamping can resolve events on signal edges with 75 ns accuracy. A sequence of events followed by a period of inactivity generates an interrupt. Software then interprets the “character” by looking at the event list. The sampling method instead allows constant frequency sampling and software interpretation of the sampled patterns.

These mechanisms allow interpretation of a wide variety of remote control protocols. The Philips RC-5, RC-6 and RC-MM as well as the IrDA Control 1.0 remote control protocols are all decoded with this mechanism, provided that the RF demodulation is performed externally. Most other consumer electronic vendor remote control protocols can be supported by appropriate software.

The GPIO DMA of event lists or samples into memory assures the ability to interpret complex, fast edge-rate sequences with precise resolution and a minimum of CPU interrupts.

1.6.21 IR Blaster

The GPIO DMA-based signal sequence generation capability is suitable towards creating IR control signals of any type, including RC-5, RC-6, RC-MM and IrDA Control. The GPIO block provides the capability to software generate (sampling or timed event based) signals, which are then optionally modulated upon a digital carrier wave with 33%, 50% or 66% duty cycle. The carrier is a divider of 108 MHz.

This creates a flexible IR Blaster capability, where software can implement any current or future protocol.

1.6.22 Endianness

The PNX8526 fully supports little and big-endian software stacks. Endian mode is set at boot or shortly after, and not changed.

The PNX8526 always starts its on-chip MIPS device in a fixed endianness, determined by the boot script. There is a provision for MIPS software to change endianness dynamically, such that a field software Flash upgrade can contain an “opposite endianness” operating system from the one originally released.

The PNX8526 on-chip peripherals and coprocessors observe the system global endianness flag, as does the MIPS CPU. TM32 endianness is set by the TM32 program module itself and should always be identical to system endianness.

When selecting PCI peripherals for a dual-endianness product, care must be taken to ensure that they can operate without “CPU fixup” in either endianness. Typically, PowerPC-compatible PCI devices support both endian modes.

1.6.23 Interrupts

The PNX8526 on-chip devices and off-chip interrupt sources connect to an interrupt request line on both processors. Software uses an “interrupt enable mask” to decide which device is handled by which processor. At any moment, only one CPU is responsible for a given device, but different software applications might have different CPU assignments for devices.

The TM32 CPU itself has a built-in interrupt controller that provides efficient auto-vectoring interrupts for up to 32 separate requests. Additional interrupts are provided for by the TPIC block, which contains one or more PICs (DVP-standard Programmable Interrupt Controllers). The MIPS CPU has a single interrupt request line and uses one or more PIC blocks to mask and prioritize interrupts.

The PR3940 and TM32 CPU have the capability to assert mutual interrupts. This is done by one CPU writing to the MMIO location of the interrupt controller on the other CPU. (This would also apply to any off-chip CPU.)

In case of an external PCI host CPU, the PR3940 is not used. In that case, the PICs associated with the PR3940 are used to provide the INTA signal, which then functions as interrupt request to the PCI host CPU.

1.6.24 Special Display Scenarios

The PNX8526 supports a few non-standard display operating modes that can be used in specific applications.

The AICP has a mode where the two 10-bit outputs are used to emit a high-bandwidth YUV 4:2:2 pixel structure by using the DV1 (primary) output for Y and DV2 (secondary) output for multiplexed UV. Using this mode, a single image stream with pixel rates up to 81 Mpix/s can be emitted across the DV1 and DV2 output pins. Due to the high memory bandwidth consumed in this mode, other PNX8526 features are limited. There is no secondary AICP in this mode. Scaling by the MBS and use of additional AICP layers may be limited by memory bandwidth availability.

The primary AICP supports YUV 4:2:2 refresh of a 960x1080I display, i.e. half horizontal resolution HD. In this mode, the secondary AICP can still be used. Because of the high memory bandwidth usage by the AICP, not all PNX8526 features are available in this scenario.

The PNX8526 is not able to support a primary AICP operation at 1440x960I with a simultaneous secondary AICP operation. The DV output clock rate limitation of approximately 80 MHz requires that both the DV1 and DV2 pins are used for this resolution. No 4:1:1 mode display output is supported.

1.7 PNX8526 Internal Architecture

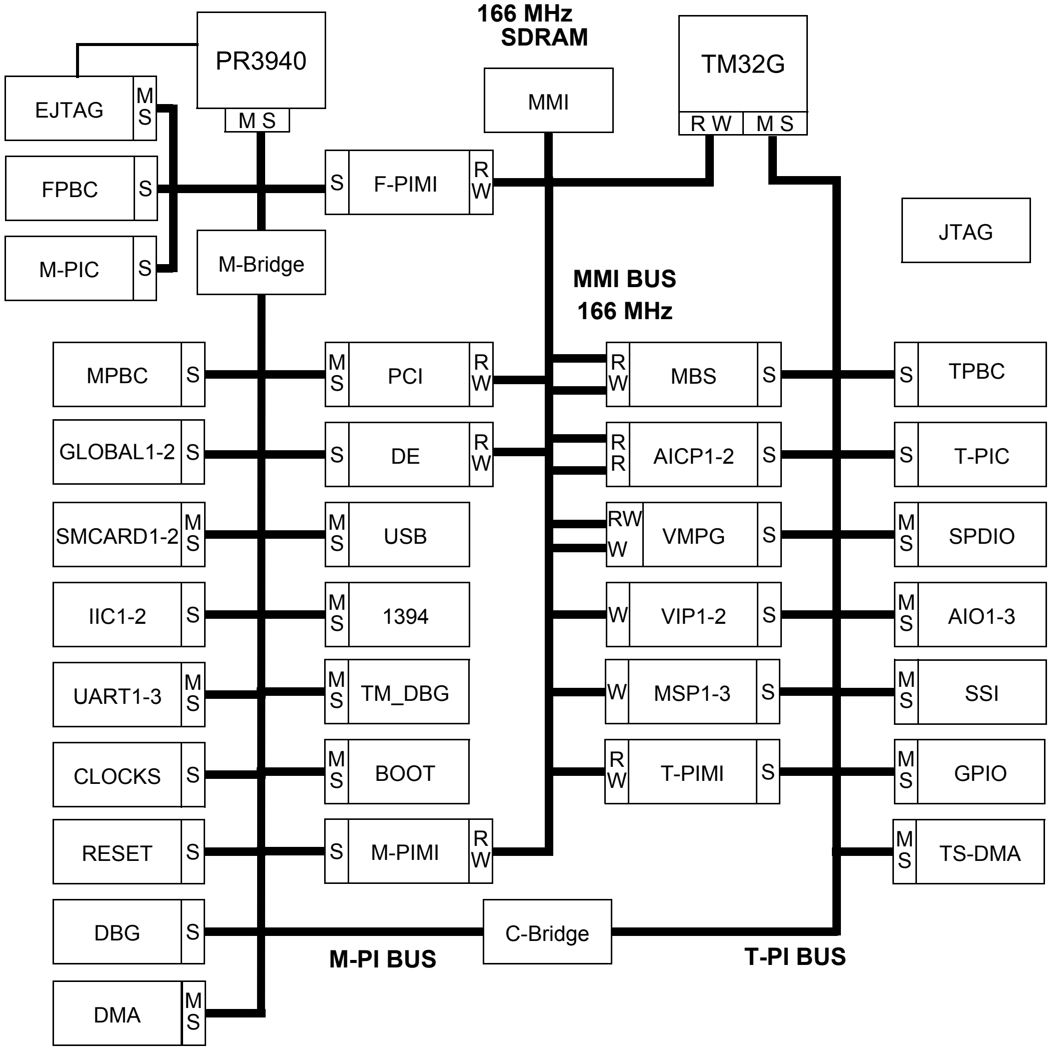


Figure 7: PNX8526 Bus Topology

The PNX8526 IC is a Philips Digital Video Platform (DVP) compliant design. A high-speed, 64-bit wide memory interface, the MMI bus provides high bandwidth and low latency memory access to those modules requiring high performance memory access. Two PI-Buses and a crossover PI-to-PI MMIO bridge provide memory mapped I/O access from each processor to control or observe status of all peripheral modules. Modules may also use PI segments as a medium bandwidth memory access bus via the T-PIMI and M-PIMI DMA gateways. Modules processing audio or video streams or requiring real-time interrupt response attach to the TM32 CPU PI segment.

Both the MIPS PR3940 and TM32 CPU processors have low latency, high bandwidth connections to the 64-bit memory interface. Additionally, both processors interface to PI segments for MMIO access. Both processors can access all modules, however, there may be some additional latency if MIPS tries to access media processing modules or when the TM32 CPU accesses control functions.

1.7.1 Miscellaneous Modules

The following modules support the system, but don’t provide user-level functionality:

- FPBC - Fast PI-Bus controller
- MPIC - MIPS Programmable Interrupt Controller
- EJTAG - the module that implements the MIPS debug port
- FPIMI - Fast PI-Bus to MMI. This unit provides main memory access to the MIPS.
- M-Bridge - Makes transactions between the MIPS fast PI-Bus and MIPS peripheral PI-Bus.
- MPBC - MIPS side PI-Bus Controller
- GLOBAL - Global register file, which contains software readable and writable registers not associated with a particular peripheral.
- CLOCKS - Global clock, reset and powerdown logic. Manufactures all clocks and resets for all modules. All clocks are derived from a single 27 MHz xtal connected to the PNX8526. Some clocks are made by a simple PLL with feedback divider and pre-divider. Where needed (audio, video clocks), TriMedia-style Direct Digital Synthesizers are used to provide for dynamically (software) programmable clocks.
- PIMI - These two modules provide DMA access to Main Memory for the peripherals on the MIPS and TriMedia side PI-Buses, respectively.
- C-Bridge - Crossover bridge. It allows the MIPS to do MMIO (programmed I/O) transactions to devices on the TriMedia side, and the TM32 CPU to do MMIO transactions to devices on the MIPS side. In normal operation, a device is always managed by one CPU only.
- TPBC - TriMedia side PI-Bus Controller.
- TPIC - TriMedia Programmable Interrupt Controller. The TM32 CPU has 32 auto-vectoring direct request inputs. The TPIC is an interrupt expander to provide for additional interrupt lines.
- JTAG - This module contains the chip boundary-scan logic. The pins of this module are also used for the TM32 CPU debug port.

1.8 Power Requirements

The PNX8526 requires a 1.26 Volt core supply voltage and a 3.3 Volt I/O supply voltage.

In full performance operating mode, with the TM32 CPU at 200 MHz, MIPS at 150 MHz and 64 bit SDRAM at 166 MHz, total power is 2.0 watts; roughly divided as 1.5 watts for the core and 0.5 watt for I/O.

Power dissipation for each CPU is linear in response to frequency. Reduced dissipation modes are possible where compute budget allows. (The CPUs have a limited set of programmable clock rates.) The TM32 CPU core supports full dynamic powerdown/wakeup with 0 dissipation. Power can be reduced further by powering down individual on-chip modules (software powerdown) and by halting the MIPS CPU. Provisions exist for automatic wakeup (both CPUs) on interrupts and I/O events.

1.9 Package

The PNX8526 is packaged in a 456-pin PBGA package, with 36 center thermal/ground balls and 35 x 35 mm body size. This allows for up to 352 active signal pins.

1.10 Delta between PNX8525 and PNX8526

The following lists the differences between the PNX8525 with limitation as defined in PNX8525 Limiation Document V1.5, or later and the PNX8526.

Device Identification

- The JTAG Device ID has changed
- The MIPS Module ID has changed
- THE TM32 Module ID has changed

Table 9: Functional Deltas PNX8526

	PNX8525	PNX8526
JTAG Device ID	0x1_25A0_02B	tbd
MIPS Module ID	0x0010_2B22	0x0010_2B23
TM32 Module ID	0x2B80_0001	0x2B80_3001

Physical Characteristics

- Reduced Core supply voltage
- Reduced Core Power Dissipation

Table 10: Physical Deltas PNX8526

	PNX8525	PNX8526
Core Supply Voltage	1.8 V	1.26 V
Core Power Dissipation	4.5 W	2.0 W

Chapter 2: System Memory Map and Protection

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

2.1 Introduction

All on/off-chip CPUs and devices in the PNX8526 system see objects in the system at the same physical address. To implement protection, not all masters in the system are allowed to access all objects. In particular, it is possible to set registers such that processes on the TM32 CPU core can not access devices and memory ranges that are mission critical.

This chapter describes the following:

- Standard PNX8526 system memory map (internal MIPS host CPU)
- Object visibility rules and protection mechanisms
- Registers controlling the system memory map and protection mechanisms
- Alternate system memory map (external PCI host CPU).
- Memory map view from the MIPS, TM32 CPU, PI-Bus masters and external PCI devices

Remark: Alternate PNX8526 system configurations exist, but are not supported.

2.2 Standard System Memory Map

The standard system memory map applies to PNX8526 configurations without an external PCI host processor. In such systems, the PNX8526 hardware reset and boot block script initialize on-chip registers and start the on-chip MIPS CPU. Initialization software on the MIPS completes system initialization.

The resulting standard system memory map is shown in Figure 1

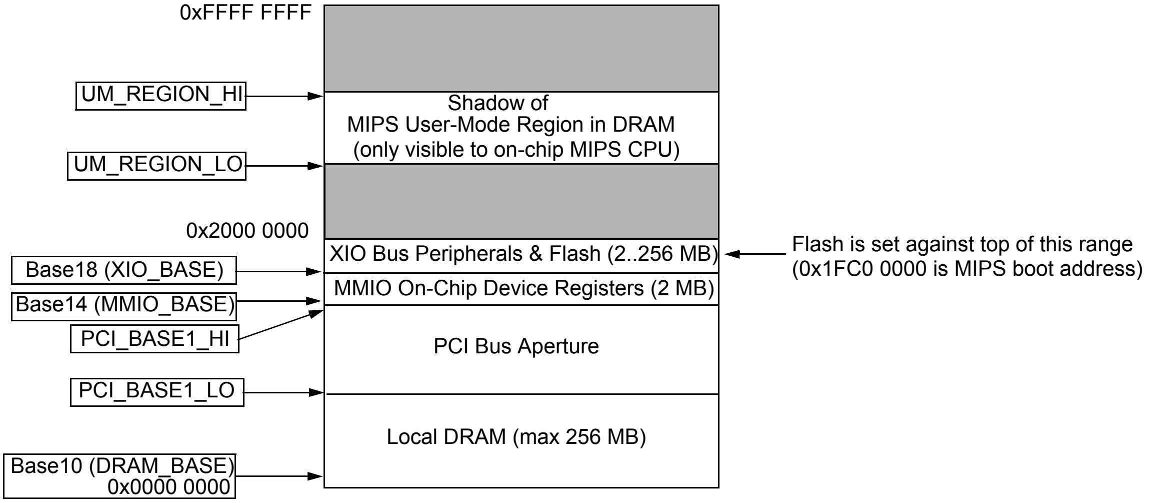

0xFFFF FFFF
UM_REGION_HI
Shadow of
MIPS User-Mode Region in DRAM
(only visible to on-chip MIPS CPU)
UM_REGION_LO
0x2000 0000
XIO Bus Peripherals & Flash (2..256 MB)
Flash is set against top of this range
(0x1FC0 0000 is MIPS boot address)
Base18 (XIO_BASE)
MMIO On-Chip Device Registers (2 MB)
Base14 (MMIO_BASE)
PCI_BASE1_HI
PCI Bus Aperture
PCI_BASE1_LO
Local DRAM (max 256 MB)
Base10 (DRAM_BASE)
0x0000 0000

Figure 1: PNX8526 Standard System Memory Map (Internal MIPS Host)

2.2.1 Apertures in the Standard System Memory Map

The PNX8526 local DRAM aperture is 16, 32 or 64 MB, and it is used for accessing the SDRAM.

The XIO bus aperture is not used in all systems. If used, it can be set from 2..256 MB in size. An access to this aperture goes to external XIO peripherals attached to the PNX8526 PCI-XIO bus, such as an IDE disk drive, ROM, Flash, SRAM memory, or 8-bit peripheral devices. Due to the sharing of wires between PCI and XIO devices, the XIO aperture is not accessible to PCI bus masters. It is however claimed as an aperture upon building the PCI bus memory map to prevent a local XIO peripheral from being assigned the same address as a PCI device.

The MMIO aperture is a fixed 2 MB in size. It contains the 32-bit wide control and status registers of all on-chip devices of the PNX8526.

The PCI bus aperture starts above DRAM and ends at the first MMIO aperture address. An access to this aperture goes to PCI bus target devices, which may be a range of memory or a PCI device control/status register. See Chapter 8 PCI-XIO for more information.

The “shadow of DRAM” aperture is only used in case of MIPS operation with the Translation Lookaside Buffer (TLB) off. In that case, it can be enabled to allow MIPS User-Mode access to a portion of system DRAM. It is described in more detail in Section 2.7.1.

An example of a typical system configuration and its standard memory map is shown in Figure 2. This is the configuration set by the PNX8526 built-in boot script #1.

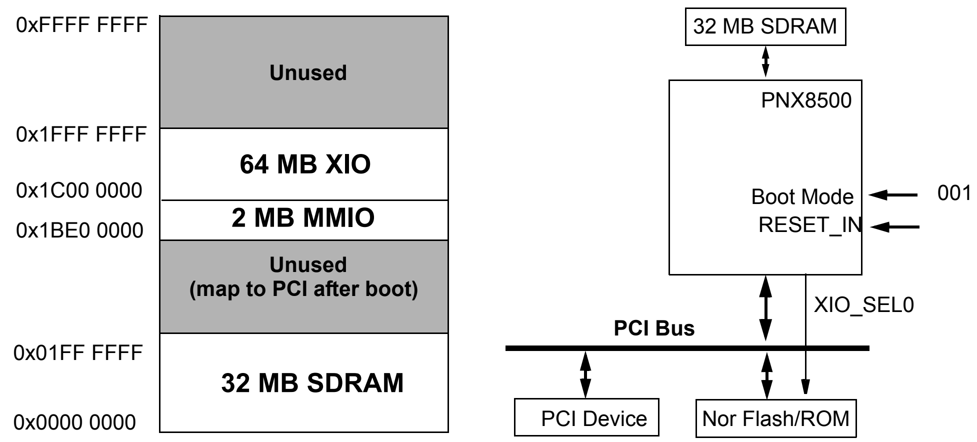


Figure 2: System Configuration and Std. Memory Map (Boot Script #1)

2.2.2 Building the Standard System Memory Map

The system memory map is built according to the following rules:

- DRAM is set to start at address 0, with a size equal to or greater than the actual DRAM size in the system. The aperture size *must be* a power of 2 between 2 MB and 256 MB.
- If used in the system, the XIO aperture is set against the top of the first 512 MB. It must have a size equal to or greater than the actual peripheral address range. The aperture size *must be* a power of 2 between 2 MB and 256 MB, and allocated on a natural boundary i.e., a 128-MB aperture must start on an address that is a multiple of 128 MB.
- The 2-MB MMIO aperture, with on-chip device control/status registers, is set against XIO. It *must be* allocated on a 2-MB boundary.
- The area between the top of DRAM and MMIO is designated as a bridge to the PCI bus.

The PNX8526 built-in boot scripts perform the first three of the above steps. MIPS software is responsible for the last step. If the size assumptions by the built-in boot scripts are inappropriate, a custom boot script can be used. Refer to Chapter 3 Boot.

2.2.3 Rationale for the Standard System Memory Map

The PNX8526 address decoder logic requires that all three apertures are powers of two in size and are “naturally aligned” i.e., a 32-MB aperture must start on an address that is a multiple of 32 MB.

It is required that MIPS kernel mode processes can access any object in the system. Due to the nature of the virtual to physical address translation in the on-chip MIPS, kernel mode access is limited to the first 512 MB of physical address space. Hence, the standard system memory map puts all objects in the lower 512 MB.

Furthermore, the MIPS exception vectors must reside in the lower part of the physical address space, necessitating DRAM at this address.

The MIPS starts execution from physical address 0x1FC0 0000, which is mapped to the XIO, allowing MIPS to start from external ROM or Flash.

Remark: The system has a special MIPS boot provision, described in Section 2.7.1.1, which can be used to boot MIPS without any direct addressable XIO to external ROM or Flash.

The choice of MMIO positioned below XIO is arbitrary.

The PCI aperture must be kept as large as possible to allow for multiple PCI devices with attached local memories. Hence, it fills all left over space.

2.3 Hardware Limitations to Object Visibility

The PNX8526 hardware imposes the following limits:

- On-chip device DMA is not allowed to access MMIO registers.
- T-PI Bus DMA devices are not allowed to target the PCI or XIO aperture.

In addition, the TM32 CPU core is set up such that it never maps the PCI or XIO aperture into its address space. This convention is highly recommended as certain PCI and XIO devices may have side effects from read operations, and the TM32 CPU generates speculative reads.

Table 1: PNX8526 Hardware Limitations to Object Visibility

Master Type	DRAM	MMIO	PCI	XIO
On-chip MIPS CPU	yes	yes	yes	yes
On-chip TM32 CPU core	yes	yes	Note A	Note A
On-chip M-PI DMA devices	yes	no	yes	yes
On-chip T-PI DMA devices	yes	no	no	no
Off-chip PCI bus masters	yes	yes	yes	no

[1-1] The TM32 CPU core is not normally set to see PCI or XIO directly in its address map. It can however use explicit MMIO transactions to the PCI-XIO block to perform any single cycle type PCI transaction, including memory, I/O and intack. Using the same method, it can also perform XIO bus transactions.

2.4 Protection Mechanisms

2.4.1 Introduction

MIPS architecture uses virtual-to-physical address translation and kernel mode or User Mode to accomplish the protection and encapsulation common in modern operating systems:

- The kernel can access all system resources.
- Kernel memory is protected from errant User-Mode processes.
- User-Mode processes are protected from each other.
- User-Mode processes can not control system resources (devices), except as described in the next bullet.
- The kernel can give a User-Mode driver task-specific privileges, such as managing a disk drive to implement a file system by mapping the device into the User-Mode task address space.

The on-chip PR3940 MIPS, when used with its Translation Lookaside Buffer (TLB) on, provides the mechanisms to implement the above protection and encapsulation.

2.4.2 MIPS with TLB Disabled

The PNX8526 provides protection if an application wants to run the MIPS with TLB off. This provision allows kernel mode access to all resources, per Table 1, but with User-Mode access limited to a range of DRAM.

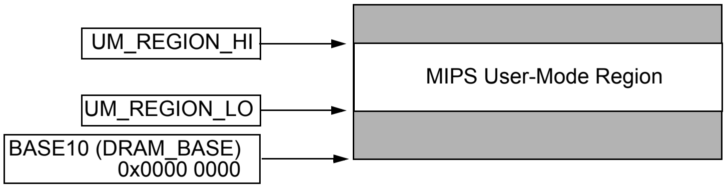


Figure 3: MIPS User-Mode DRAM Aperture Region

Refer to the UM_REGION_LO/HI register for instructions on how to enable and set the region. At reset, this mechanism is disabled. Refer to Section 2.7.1 for the MIPS Virtual Memory Map view of the system in this mode, which includes:

- The kernel is protected from errant user processes.
- The kernel can control/access all system resources.
- User processes can not access any system resources other than the designated memory region.
- User processes are not protected from one another.

- MIPS user processes can be protected from corruption by errant TM32 CPU core processes and vice versa. See Section 2.4.3.

2.4.3 TM32 Protection Mechanisms

TM32 CPU and MIPS User-Mode processes have similar privileges and protection that are available.

- The TM32 CPU core processes are protected from MIPS User-Mode processes (but not from one another).
- Devices can be given to the TM32 CPU core tasks to control.

  The TM32 CPU in the PNX8526 has no built-in supervisor/User-Mode protection or TLB. To implement the above protection, the following system provisions exist:
- It is possible to limit DRAM access by the TM32 CPU and its DMA devices to a specific range of DRAM.
- It is possible to disallow access by the TM32 CPU to individual device MMIO registers.

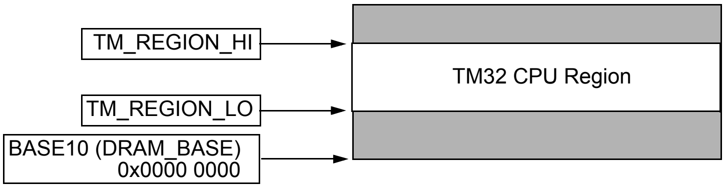


Figure 4: TM32 CPU and Device DRAM Aperture Region

The TM32 DRAM region protection is activated by putting a value in the TM_REGION_HI register that is greater than the value in TM_REGION_LO. On reset, the TM32 DRAM region is enabled, but ranges from 0..64 MB i.e., the TM32 can access all of DRAM.

The TM OWNED M-PI and TM OWNED T-PI registers contain a bit for each on-chip PNX8526 device. Setting this bit associates a device with the TM32 CPU. If this bit is set, the device can be programmed by the TM32 and should be considered to have no more rights than the TM32.

Remark: The TM_OWNED bits are part of the Global 2 unit.

By disabling TM32 CPU access to the Global 2 unit (TM_OWNED M-PI register, bit 13=0), the TM32 CPU is no longer able to change the TM_OWNED bits.

If a device’s TM_OWNED bit is set to ‘1’ (default):

- The device’s MMIO registers are readable/writable by the TM32 CPU.
- The device can only DMA to/from the TM32 CPU region in DRAM.

If set to ‘0’:

- The device’s MMIO registers are inaccessible by the TM32 CPU (causes ERROR ACK).
- The device can DMA from/to all of DRAM.

Remark: The bus location of a device (on the MIPS or TM32 PI-Bus) has no influence on a device’s accessibility by the TM32 core.

Different software applications may partition devices across the MIPS or TM32 CPU as required by setting the TM_OWNED bit. No provisions exist to protect a device from access by the MIPS. A kernel-mode process on MIPS is trusted and a User-Mode process can be TLB mapped to include/exclude a given device.

2.5 Registers Descriptions

Table 2: System Memory Map and Protection Register Summary

Offset	Name	Description
TM32 CPU Protection Registers (in GLOBAL2)		
0x04 D000	TM_OWNED_M_PI	M-PI attached devices associated with TM32 CPU core
0x04 D004	TM_OWNED_T_PI	T-PI attached devices associated with TM32 CPU core
MIPS TLB Off Protection Registers (in GLOBAL2)		
0x04 D010	UM_REGION_LO	MIPS User-Mode region low address register
0x04 D014	UM_REGION_HI	MIPS User-Mode region high address register
TM32 CPU Protection Registers (in GLOBAL2)		
0x04 D018	TM_REGION_LO	TM32 CPU Region low address register
0x04 D01C	TM_REGION_HI	TM32 CPU Region high address register
General Registers (in GLOBAL2)		
0x04 D200	PI_DRAM_LO	determines low address of DRAM seen by PI-bus masters
0x04 D204	PI_DRAM_HI	determines high address of DRAM seen by PI-bus masters
0x04 D600	IO_MUX_CTRL	controls I/O multiplexing for the Audio, SSI, RGB, ICAM and VIP functions

2.5.1 Global 2 Registers

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Devices Controlled or Accessible by TM32 CPU Core Registers				
Offset 0x04 D000			*TM OWNED M-PI*	
31:20		-	Unused	Ignore during writes and read as zeroes.
19	R/W	0x1	TM_OWNED_GLBREG_1	0 = TM32 does not have access to Global 1 registers. 1 = TM32 has access to Global 1 registers.
18	R/W	0x1	TM_OWNED_DMA	0 = TM32 does not have access to the DMA registers. 1 = TM32 has access to the DMA registers.
17	R/W	0x1	TM_OWNED_TMDBG	0 = TM32 does not have access to the Debug registers. 1 = TM32 has access to the Debug registers
16	R/W	0x1	TM_OWNED_RESET	0 = TM32 does not have access to the Reset registers. 1 = TM32 has access to the Reset registers.
15	R/W	0x1	TM_OWNED_2D	0 = TM32 does not have access to the 2D Draw Engine registers. 1 = TM32 has access to the 2D Draw Engine registers.
14	R/W	0x1	TM_OWNED_MBC	0 = TM32 does not have access to the M-PI Bus controller registers. 1 = TM32 has access to the M-PI Bus controller registers.
13	R/W	0x1	TM_OWNED_GLBREG_2	0 = TM32 does not have access to Global 2 registers. 1 = TM32 has access to Global 2 registers.
12	R/W	0x1	TM_OWNED_UART_3	0 = TM32 does not have access to the UART3 registers. 1 = TM32 has access to the UART 3 registers.
11	R/W	0x1	TM_OWNED_UART_2	0 = TM32 does not have access to the UART2 registers. 1 = TM32 has access to the UART 2 registers.
10	R/W	0x1	TM_OWNED_UART_1	0 = TM32 does not have access to the UART1 registers. 1 = TM32 has access to the UART1 registers.
9	R/W	0x1	TM_OWNED_1394	0 = TM32 does not have access to the IEEE 1394 registers. 1 = TM32 has access to the IEEE 1394 registers.
8	R/W	0x1	TM_OWNED_USB	0 = TM32 does not have access to the USB registers. 1 = TM32 has access to the USB registers.
7	R/W	0x1	TM_OWNED_CLOCKS	0 = TM32 does not have access to the Clock registers. 1 = TM32 has access to the Clock registers.
6	R/W	0x1	TM_OWNED_I2C_2	0 = TM32 does not have access to the I^2C2 registers. 1 = TM32 has access to the I^2C2 registers.
5	R/W	0x1	TM_OWNED_I2C_1	0 = TM32 does not have access to the I^2C1 registers. 1 = TM32 has access to the I^2C1 registers.
4	R/W	0x1	TM_OWNED_SMARTCARD_2	0 = TM32 does not have access to Smartcard2 registers. 1 = TM32 has access to the Smartcard2 registers.
3	R/W	0x1	TM_OWNED_SMARTCARD_1	0 = TM32 does not have access to Smartcard1 registers. 1 = TM32 has access to the Smartcard1 registers.
2	R/W	0x1	TM_OWNED_BOOT	0 = TM32 does not have access to the Boot registers. 1 = TM32 has access to the Boot registers.

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
1	R/W	0x1	TM_OWNED_DEBUG	0 = TM32 does not have access to the Debug registers. 1 = TM32 has access to the Debug registers.
0	R/W	0x1	TM_OWNED_PCI_XIO	0 = TM32 does not have access to the PCI-XIO block registers. 1 = TM32 has access to the PCI-XIO and its block registers.
Offset 0x04 D004			*TM OWNED T-PI*	
31:23		-	Unused	Ignore during writes and read as zeroes.
22	R/W	0x1	TM_OWNED_MSP2	0 = TM32 does not have access to the MSP2 registers. 1 = TM32 has access to the MSP2 registers.
21	R/W	0x1	TM_OWNED_MSP1	0 = TM32 does not have access to the MSP1 registers. 1 = TM32 has access to the MSP1 registers.
20	R/W	0x1	TM_OWNED_TSDMA	0 = TM32 does not have access to the TSDMA registers. 1 = TM32 has access to the TSDMA registers.
19	R/W	0x1	TM_OWNED_AIN3	0 = TM32 does not have access to the Audio In3 registers. 1 = TM32 has access to the Audio In3 registers.
18	R/W	0x1	TM_OWNED_AOUT3	0 = TM32 does not have access to the Audio Out3 registers. 1 = TM32 has access to the Audio Out3 registers.
17	R/W	0x1	TM_OWNED_AIN2	0 = TM32 does not have access to the Audio In2 registers. 1 = TM32 has access to the Audio In2 registers.
16	R/W	0x1	TM_OWNED_AOUT2	0 = TM32 does not have access to the Audio Out2 registers. 1 = TM32 has access to the Audio Out2 registers.
15	R/W	0x1	TM_OWNED_AIN1	0 = TM32 does not have access to the Audio In1 registers. 1 = TM32 has access to the Audio In1 registers.
14	R/W	0x1	TM_OWNED_AOUT1	0 = TM32 does not have access to the Audio Out1 registers. 1 = TM32 has access to the Audio Out1 registers.
13	R/W	0x1	TM_OWNED_AICP2	0 = TM32 does not have access to the AICP2 registers. 1 = TM32 has access to the AICP2 registers.
12	R/W	0x1	TM_OWNED_AICP1	0 = TM32 does not have access to the AICP1 registers. 1 = TM32 has access to the AICP1 registers.
11	R/W	0x1	TM_OWNED_MBS	0 = TM32 does not have access to the MBS registers. 1 = TM32 has access to the MBS registers.
10	R/W	0x1	TM_OWNED_D3D	0 = TM32 does not have access to the D3D registers. 1 = TM32 has access to the D3D registers.
9	R/W	0x1	TM_OWNED_SPDIFIN	0 = TM32 does not have access to the SPDIF In registers. 1 = TM32 has access to the SPDIF In registers.
8	R/W	0x1	TM_OWNED_SPDIFOUT	0 = TM32 does not have access to the SPDIF Out registers. 1 = TM32 has access to the SPDIF Out registers.
7	R/W	0x1	TM_OWNED_SSI	0 = TM32 does not have access to the SSI registers. 1 = TM32 has access to the SSI registers.
6	R/W	0x1	TM_OWNED_VIP_2	0 = TM32 does not have access to the VIP2 registers. 1 = TM32 has access to the VIP2 registers.
5	R/W	0x1	TM_OWNED_VIP_1	0 = TM32 does not have access to the VIP1 registers. 1 = TM32 has access to the VIP1 registers.

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R/W	0x1	TM_OWNED_MPG	0 = TM32 does not have access to the MPEG registers. 1 = TM32 has access to the MPEG registers.
3	R/W	0x1	TM_OWNED_GPIO	0 = TM32 does not have access to the GPIO registers. 1 = TM32 has access to the GPIO registers.
2	R/W	0x1	TM_OWNED_TBC	0 = TM32 does not have access to the T-PI Bus controller registers. 1 = TM32 has access to the T-PI Bus controller registers.
1	R/W	0x1	TM_OWNED_TPIC	0 = TM32 does not have access to the PIC registers. 1 = TM32 has access to the PIC registers.
0	R/W	0x1	TM_OWNED_TM32	0 = TM32 does not have access to the PI-Bus registers. 1 = TM32 has access to the PI-Bus registers.

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
MIPS User-Mode Region Registers				
Offset 0x04 D010			***UM_REGION_LO***	
31:16	R/W	0x0000	UM_REGION_LO	Lowest PI-Bus address mapped to MIPS User-Mode region, 64-kB granularity.
15:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x04 D014			***UM_REGION_HI***	
31:16	R/W	0x0000	UM_REGION_HI	First PI-Bus address outside the MIPS User-Mode region, 64-kB granularity. Reset to 0 to disable the MIPS User-Mode region.
15:0		-	Unused	Ignore during writes and read as zeroes.
TriMedia DRAM Region Registers				
Offset 0x04 D018			***TM_REGION_LO***	
31:16	R/W	0x0000	TM_REGION_LO	Lowest PI-Bus address that maps to the TM32 CPU core and its controlled devices in DRAM region, 64-kB granularity. TM aperture is reset to start at 0.
15:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x04 D01C			***TM_REGION_HI***	
31:16	R/W	0x0400	TM_REGION_HI	First PI-Bus address outside TM32 CPU core and its controlled devices in DRAM region, 64-kB granularity. TM aperture is reset to 64 MB.
15:0		-	Unused	Ignore during writes and read as zeroes.

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
DRAM Location Seen from PI-Bus Registers				
Offset 0x04 D200			***PI_DRAM_LO***	
31:16	R/W	0x0000	PI_DRAM_LO	Lowest PI-Bus address mapped to DRAM, 64-kB granularity. PI-Bus DRAM aperture is reset to start at 0.
15:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x04 D204			***PI_DRAM_HI***	
31:16	R/W	0x0400	PI_DRAM_HI	First address outside PI-Bus DRAM space, 64-kB granularity. PI-Bus DRAM aperture is reset to 64 MB.
15:0		-	Unused	Ignore during writes and read as zeroes.

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
I/O Multiplexer Control Register				
Offset 0x04 D600			***IO_MUX_CTRL***	
31:15		-	Unused	Ignore during writes and read as zeroes.
14	R/W	0x0	AIO_MUX_SEL	I2S_IO Audio mode: 0 = Select I2S_IO as Audio Out. 1 = Select I2S_IO as Audio In.
13	R/W	0x0	SSI_SEL	SSI or UART3 mode: 0 = Select UART3. 1 = Select SSI.
12	R/W	0x0	RGB24_SEL	Audio Out2 or RGB mode: 0 = Select Audio Out2. 1 = Select RGB (DV_OUT [23:20]).
11:10	R/W	0x0	SMCRD2_MUX_CTRL	ICAM or SmartCard2 mode: 00 = SmartCard1 module ports go to SmartCard2 pins. 01 = SmartCard2 module ports go to SmartCard2 pins. 10 = ICAM1 module ports go to SmartCard2 pins. 11 = ICAM2 module ports go to SmartCard2 pins.
9:8	R/W	0x0	SMCRD1_MUX_CTRL	ICAM or SmartCard2 mode: 00 = SmartCard1 module ports go to SmartCard1 pins. 01 = SmartCard2 module ports go to SmartCard1 pins. 10 = ICAM1 module ports go to SmartCard1 pins. 11 = ICAM2 module ports go to SmartCard1 pins.
7		-	Unused	Ignore during writes and read as zeroes.

Global 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
6:4	R/W	0x0	VIP2_MUX_CTRL[2:0]	VIP2 module selection: 000 = VIP data from DV1 port 001 = VIP data from DV2 port 010 = VIP data from DV3 port *011 = VIP data from DV_OUT1(AICP1) port *100 = 1394 data from link core * These functions are not available.
3		-	Unused	Ignore during writes and read as zeroes.
2:0	R/W	0x0	VIP1_MUX_CTRL[2:0]	VIP1 module selection: 000 = VIP data from DV1 port 001 = VIP data from DV2 port 010 = VIP data from DV3 port *011 = VIP data from DV_OUT2 (AICP2) port *100 = 1394 data from link core * These functions are not available.

2.5.2 PCI, TM32, and MIPS Aperture Control Registers

The following registers relate to the PNX8526 System Memory Map and Object Visibility. For more information, see Chapter 8 PCI-XIO, Chapter 25 Internal TM32 CPU Core Processor, and Chapter 35 PI-Bus Architecture.

Table 3: Registers - PCI, TM32 and MIPS

Offset	Name	Description
BASE Registers (PCI-XIO)		
0x04 0050	BASE10 (DRAM_BASE)	Shadow of PCI config register that determines DRAM base address in memory map
0x04 0054	BASE14 (MMIO_BASE)	Shadow of PCI config register that determines MMIO base address in memory map
0x04 0058	BASE18 (XIO_BASE)	Shadow of PCI config register that determines XIO base address in memory map
0x04 0018	PCI_base1_lo	Low address of region 1 of PI-bus that gets bridged to PCI
0x04 001C	PCI_base1_hi	High address of region 1 of PI-bus that gets bridged to PCI
0x04 0020	PCI_base2_lo	Low address of region 2 of PI-bus that gets bridged to PCI
0x04 0024	PCI_base2_hi	High address of region 2 of PI-bus that gets bridged to PCI
TM32 CPU Aperture Control Registers (TM32 CPU)		
00x10 0034	TM32_DRAM_LO	Low address of DRAM
00x10 0038	TM32_DRAM_HI	High address of DRAM
00x10 003C	TM32_DRAM_CLIMIT	Cacheable limit (addresses above this are not cached)
MIPS-Related Registers (FPBC)		

Table 3: Registers - PCI, TM32 and MIPS

Offset	Name	Description
0x03 F020	MIPS_BOOT_ADR0	Address of first of 4 registers to re-map the MIPS reset vector
0x03 F024	MIPS_BOOT_ADR4	Address of second of 4 registers to re-map the MIPS reset vector
0x03 F028	MIPS_BOOT_ADR8	Address of third of 4 registers to re-map the MIPS reset vector
0x03 F02C	MIPS_BOOT_ADRC	Address of fourth of 4 registers to re-map the MIPS reset vector
0x03 F034	EN_MIPS_REMAP_FPBC	Enable the 4 registers to re-map the MIPS reset vectors at 0x1fc0,0000..000c

2.6 Alternate System Memory Map with External Host CPU

2.6.1 PCI Standard Boot and Memory Map Assembly

If an external host CPU (MIPS or other) is present on the PCI bus, it has the responsibility to enumerate and configure all PCI resident devices, including each PNX8526. This configuration process builds an address map where apertures of all devices (including each PNX8526) are given unique PCI addresses. The standardized protocol that accomplishes this is described in the PCI Local Bus Specification, Revision 2.2.

The configuration process of the PNX8526 in this case is summarized below:

1. The boot block writes to the PCI_SETUP register to set the desired size for each of the DRAM, MMIO and XIO apertures—typically equal to attached DRAM size on a particular board; 2 MB for MMIO and up to 128 MB for XIO. The PCI-XIO block generates a “retry” on any attempt by the host to access it until the PCI_SETUP write is completed.
2. The host PCI BIOS reads each base address PCI configuration register of a PCI device, in particular the lsb to determine if the requested aperture is a PCI memory or I/O space aperture. The PNX8526 has three such base addresses, each requesting a PCI memory space type aperture. They are Base10 (DRAM), Base14 (MMIO) and Base18 (XIO).
3. The host writes an “all 1” value to each base address register and reads it back. The PCI block hardware returns 0s in all “don’t care bits” and 1s in all actually writable bits, from which the host deduces the size of the requested memory aperture.
4. The host writes a unique address value to each base address register to set the aperture base address of DRAM, MMIO and XIO.

Remark: The outcome of this host configuration process is three apertures (DRAM, MMIO, XIO) that will most likely be adjacent to each other in the PCI address space and anywhere between 0x0 to 0xFFFF FFFF.

2.6.2 Internal MIPS and External Host CPU

The address decoding logic for DRAM, XIO and on-chip MMIO devices is designed to decode addresses relative to the base address values established by the PCI memory map building protocol. The TM32 CPU core is able to execute code and load/store data at any physical address. However, the on-chip MIPS CPU is not able to execute in this unpredictable PCI host address map environment, since it expects a certain physical memory layout e.g., DRAM and MMIO must be within the first 512 MB.

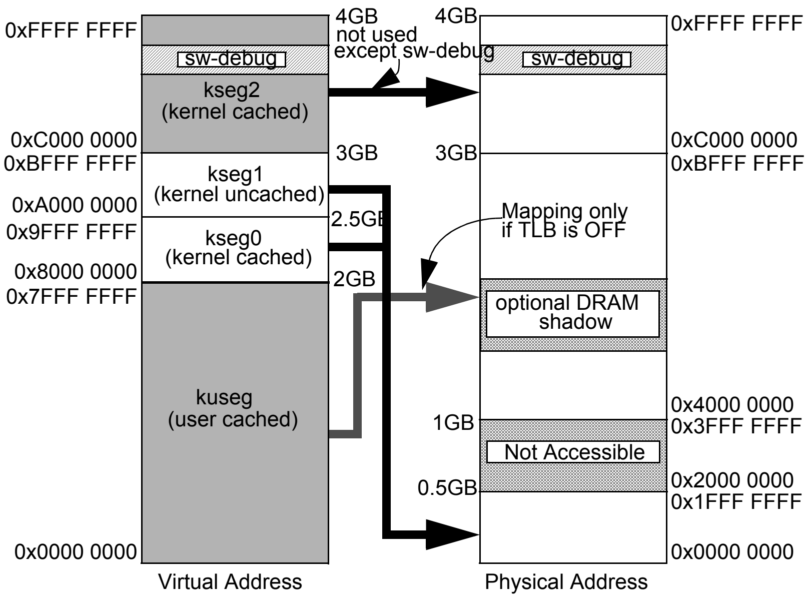
Hence, the internal MIPS CPU is not used when an external host is present. In certain instances e.g., if the code on the external host is developed specifically for the PNX8526, and it can guarantee “MIPS-friendly” base address values, the internal MIPS can still be used.

2.7 Memory Map Perspectives

2.7.1 View from MIPS

The on-chip MIPS is only active when no external host CPU is present. If MIPS is active, it is in the “standard system memory map” environment, where *all* system resources are in the first 512 MB of the physical address space.

The chosen virtual-to-physical addressing scheme for the internal PR3940 MIPS CPU is shown in the following diagram.



Note: Shaded areas are TLB mappable.

Figure 5: MIPS Address Map

Kseg0 and kseg1 are *not* mappable via the TLB. They map directly to the lower 0.5-GB where all system resources (DRAM, PCI, MMIO, XIO) reside.

The software debug area is the only part of kseg2 that is used in the PNX8526. This SW debug area is non-cacheable and can not be mapped with a TLB. The mapping of this area can be enabled/disabled in the PNX8526.

User-Mode processes will reside in kuseg. They will almost always be mapped via the MIPS TLB. The TLB allows per page translation to any physical address in the 4 GB physical memory space.

If the TLB is off, kuseg is mapped to 0x4000 0000 as shown in the diagram. This mode of operation can be supported by programming the PNX8526 UM_REGION_LO and UM_REGION_HI registers with the physical address range that is desired to shadow to DRAM. This shadow then maps to actual DRAM at UM_REGION_LO (0x4000 0000) to UM_REGION_HI (0x4000 0000).

2.7.1.1 MIPS Exception Vector Logic

The exception vectors for the MIPS reside in kseg0 or kseg1 depending on the value of the MIPS internal Boot Exception Vector (BEV) bit. These exception vectors and the corresponding virtual and physical address locations are shown in the following table.

Table 4: MIPS Exception Vector Addresses

Exception	BEV = 0		BEV = 1 (Reset Default)	
	Virtual Address	Physical Address	Virtual Address	Physical Address
Reset	0xbfc0 0000	0x1fc0 0000	0xbfc0 0000	0x1fc0 0000
Soft Reset				
UTLB Miss	0x8000 0000	0x0000 0000	0xbfc0 0100	0x1fc0 0100
TLB Miss	0x8000 0080	0x0000 0080	0xbfc0 0180	0x1fc0 0180
TLB Modification				
Bus Error				
Non-Maskable Interrupt	0xbfc0 0000	0x1fc0 0000	0xbfc0 0000	0x1fc0 0000
Address Error	0x8000 0080	0x0000 0080	0xbfc0 0180	0x1fc0 0180
Overflow				
System Call				
Break Point				
Reserved Instruction				
Coprocessor unusable				
Interrupt				
Debug	0xbfc0 0200	0x1fc0 0200 or 0xff20 0200	0xbfc0 0200 or 0xff20 0200	0x1fc0 0200 or 0xff20 0200

Remark: The BEV bit is set to 1 during reset and typically set to 0 after boot of the operating system.

The PNX8526 has a special “reset remap” system provision in the F-PI Bus controller that provides a way to have a programmable reset vector (or exception vectors) during boot:

- The bit ‘EN_MIPS_REMAP_FPBC’ in the FPBC bus controller enables four 32-bit registers that respond to MIPS accesses starting at address 0x1fc0 0000, 0x1fc0 0100 or 0x1fc0 0180. This mapping is by default enabled at reset, but can be disabled by boot if desired.
- These registers are reset to contain a ‘load register’, ‘jump to register’ and ‘NOP’ instruction. The address that this sequence jumps to can be changed in the boot script by writing a different ‘load register’ value to the first register (MIPS_BOOT_ADR0 in FPBC). The reset default is ‘lui t0, A010’, causing execution to start in kseg1, uncached, 0xA010 0000, which maps to 1 MB into the SDRAM.

The boot script can use the reset remap mechanism to start the MIPS at any desired address e.g., from an XIO non-volatile memory or from initialized SDRAM.

A similar mechanism exists to remap the debug exception vector. Refer to Chapter 35 PI-Bus Architecture for details.

2.7.2 View from the TM32 CPU Core

The memory map seen by the TM32 CPU core contains three apertures. Each aperture is independent. Any address is a legal base address, including 0x0000 0000. Apertures should not be set to overlap or extend across the 0xFFFF FFFF limit of 32-bit addressing conventions. The apertures are shown in the following block diagram.

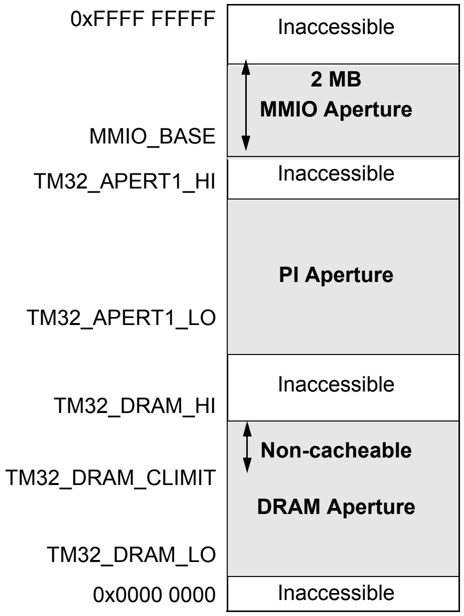


Figure 6: Memory Map for TM32 CPU Core

The TM32 CPU core requires all apertures to be a multiple of 64 kB and reside on a 64-kB boundary. They are programmed by writing to MMIO registers inside the TM32 CPU core, with the exception of the MMIO_BASE, which is directly taken from the PCI Base14 register content. In the PNX8526, the TM32_DRAM_LO/HI registers must be set to view the entire PNX8526 SDRAM aperture. Protection can be accomplished by using the TM_REGION_LO/HI registers, as described in Section 2.4.3.

The TM32 CPU can access the DRAM aperture with all load and store instructions. Loads and stores in the cacheable area use the data cache. Loads and stores in the non-cacheable area bypass the data cache and go directly to memory across the memory interface. Execution is supported from the entire DRAM aperture, which always uses the instruction cache.

Upon either reset, TM32 CPU puts all registers and cache control in its defined initial state. It does *not* start program execution. Program execution is started when the code for “start” is written to the TM32_CTL MMIO register. Execution starts at the address contained in the TM32_START_ADDR MMIO register and the boot address is therefore flexible. Also, the addresses for the exception vectors are programmable and do not put any constraints on the system design.

The PI aperture is not normally used in the PNX8526. It is enabled by writing a TM32_APERT1_HI greater than TM32_APERT1_LO. In that case, loads/stores to such addresses cause (non-cached) accesses across the T-PI bus. In special applications of the PNX8526, this could be used to map (part of) PCI and/or XIO directly into the TM32 address map.

Remark: Due to the TM32 CPU core architecture and compiler code generator, it does speculative loads i.e., it may perform loads to any location in its address map without an explicit request. The MMIO devices inside the PNX8526 are designed to cope with this, but not all PCI or XIO devices may be able to do so. For this reason, direct mapping of PCI or XIO is not generally recommended.

2.7.3 View from PCI Bus

There are two different cases to consider for PCI:

- The PNX8526 is PCI configuration manager. Its CPU allocates base addresses for all PCI components in the system.
- An external CPU is PCI configuration manager on the PCI bus and the PNX8526 is one of the components.

2.7.3.1 PNX8526 as PCI Configuration Manager

As PCI configuration manager, the PNX8526 builds the standard system memory map. External PCI bus masters see the memory map, but can only access the PNX8526 SDRAM and MMIO apertures and other PCI target devices, not the XIO aperture

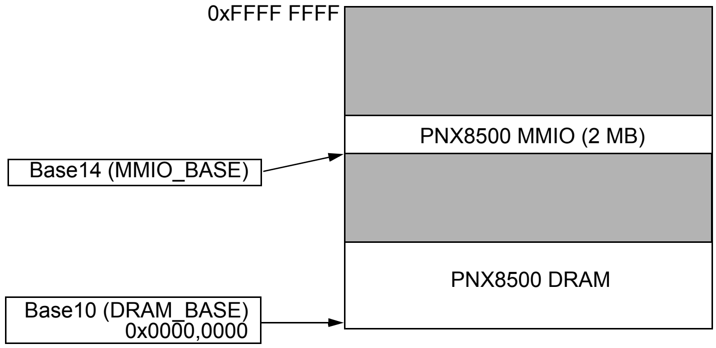


Figure 7: PNX8526 (as seen from a PCI Bus Master - #1)

2.7.3.2 An External Host CPU as PCI Configuration Manager

In this case, the host CPU PCI BIOS builds the system memory map. The host CPU assigns each PNX8526 in the system a unique Base10, Base14 and Base18 per the procedure described in Section 2.5.2.

Usually, apertures of a given device end up adjacent to one another. In a X86 PC, apertures typically end up near the high-end of the 4-GB address space. Address 0 is never used as the first 640 kB are used by the PC. Other host CPUs may have their own conventions. In general, no assumptions should be made about the resulting memory layout, other than that apertures don’t overlap.

Remark: PCI specification requires an aperture to be aligned at a boundary which is the same size as the aperture e.g., a 128-MB aperture will be located at a 128-MB boundary. External PCI bus masters see the memory map, but can only directly access the PNX8526 SDRAM and MMIO apertures, not the XIO aperture. See Figure 7.

It is possible for the external host CPU to do an indirect access to PNX8526 XIO by writing to the PCI XIO module MMIO registers to request a single cycle XIO access or a XIO DMA transaction.

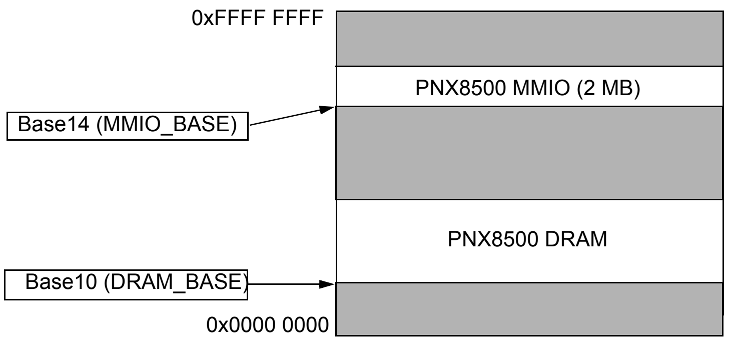


Figure 8: PNX8526 (as seen from a PCI Bus Master - #2)

2.7.4 View from Fast PI-Bus

Visibility from the MIPS and the EJTAG modules on the Fast PI-Bus is shown in Figure 8 below. All addresses are physical addresses. The MIPS User-Mode region will map to the SDRAM. The User-Mode region must be above address 0x4000 0000 as described in the MIPS documentation for MIPS address map mode 1.

To allow direct access to the PNX8526 apertures from kseg0 and kseg1 (cached and uncached kernel mode) inside the MIPS, PCI, XIO and MMIO apertures should all be within the lower 512 MB physical address space. The pi_dram_lo and pi_dram_hi apertures specify the MIPS view of SDRAM. This MIPS exception handler is located at address 0x0. Therefore pi_dram_lo must be specified at address 0x0.

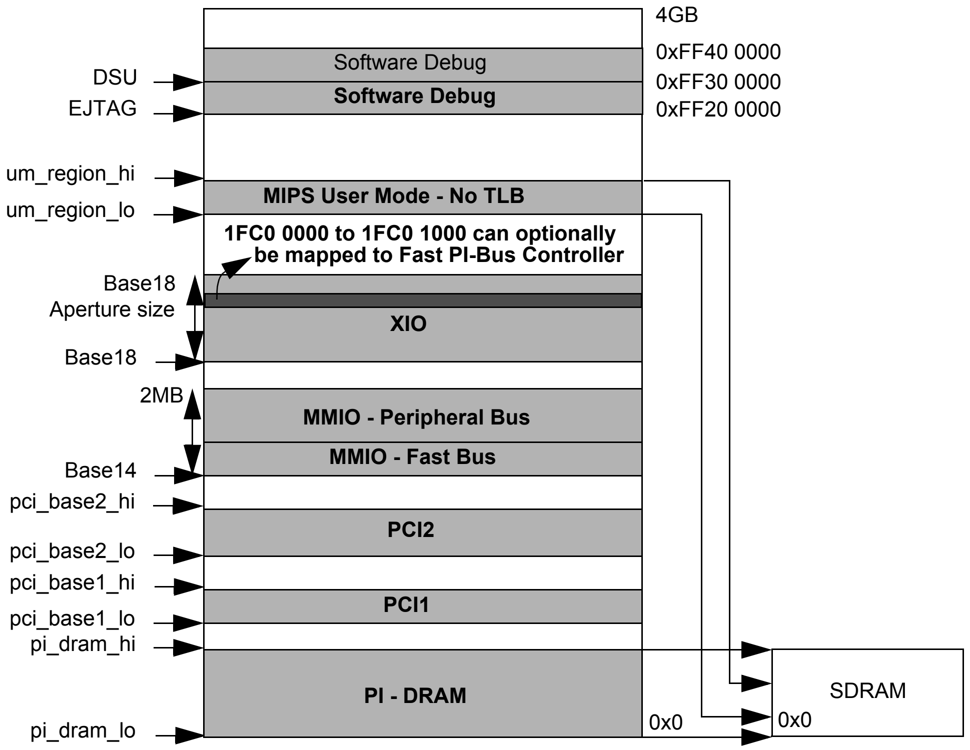


Figure 9: View of Physical Address Space from Fast PI-Bus

If the “EN_MIPS_REMAP_FPBC” bit in the Fast PI-Bus Controller (FPBC) bus controller is set, the FPBC will map the physical address space from 0x1FC0 0000 to 0x1FC0 1000 to internal registers. These registers contain code to relocate MIPS execution to SDRAM for exception handling e.g., of system reset or debug exception.

This process is exemplified in the reset vector as follows:

- The MIPS processor will boot from virtual address 0xBFC0 0000, which maps to physical address 0x1FC0 0000. The MIPS processor requires boot code at this location.
- As described in Section 2.7.1.1, inside the FPBC module there are four registers that are mapped from 0x1FC0 0000 to 0x1FC0 000C. This mapping is enabled at reset time and can be disabled during or after boot.
- When the Boot block releases the MIPS reset, the MIPS will start fetching code from address 0x1FC0 0000. It will execute a ’load register’ followed by a ’jump register’ and a NOP instruction stored in these registers. Following execution of ’jump register’ the execution will be transferred to the SDRAM. For bus accesses to non-defined areas, the bus controller responds with error acknowledge on reads and writes.

2.7.5 View from M-PI and T-PI Buses

Accessibility from the M-PI and T-PI buses is shown in Figure 9. SDRAM is accessible to all peripherals.

Peripherals on the M-PI bus can also DMA to the PCI or XIO apertures by enabling PCI-XIO block mapping of PI-Bus to the PCI1/PCI2/XIO apertures. On the T-PI bus, only the SDRAM aperture is visible to the peripherals. TM32 will typically be enabled to access the SDRAM and MMIO apertures only. However, the TM32 can be enabled to also access the PCI1/2 and XIO apertures. The XIO and PCI apertures are connected by the cross-over bridge. TM32 access to these apertures is not enabled as XIO and PCI peripherals may be sensitive to speculative reads done by the TM32.

Only the Boot, PCI, MIPS, TM32 and EJTAG modules are allowed to access MMIO space. Accesses from all other peripherals are blocked.

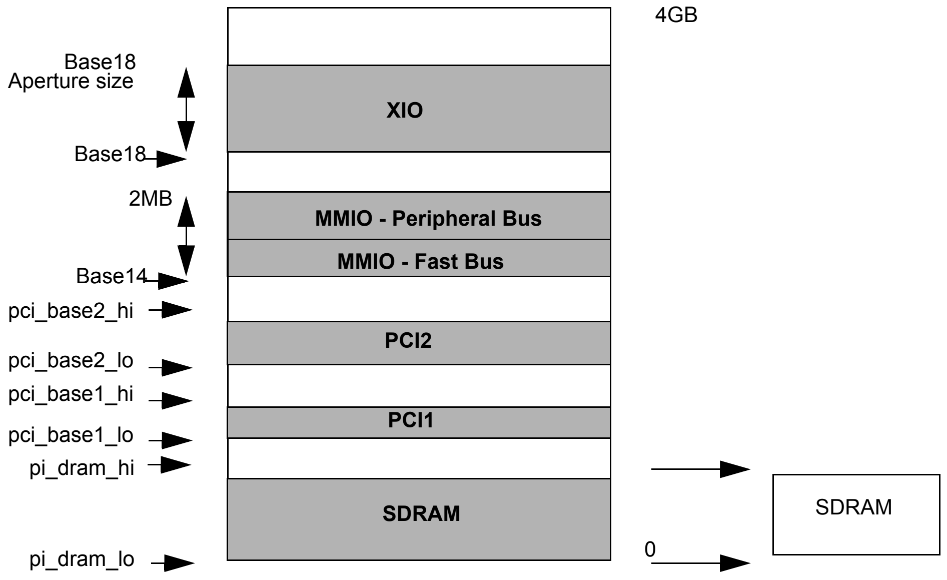


Figure 10: View from M-PI Bus and T-PI Bus

Chapter 3: Boot

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

3.1 Functional Specification

The Boot module is used to initialize some of the hardware by executing a boot script (see Section 3.4.2 and Section 3.4.3 for examples). Once the boot script has been executed, the PR3940 MIPS can begin to run the “boot software code” (not to be confused with the boot script)

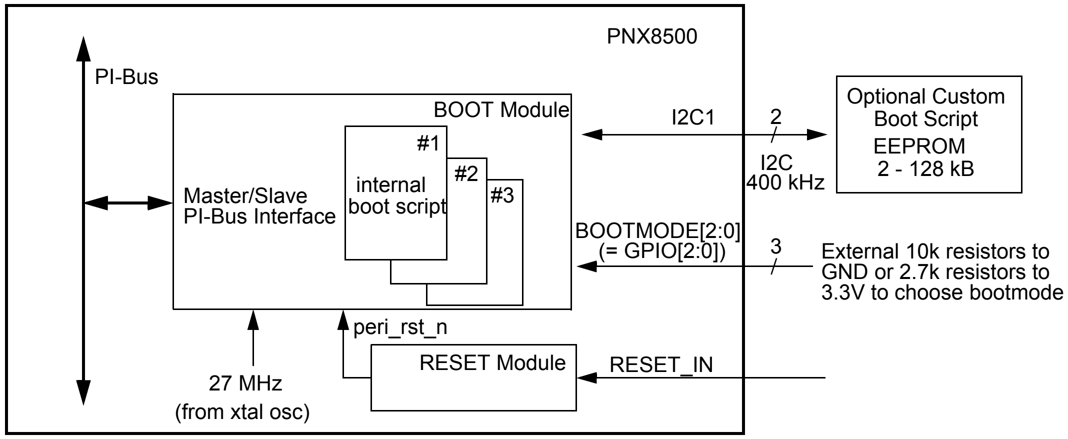


Figure 1: PNX8526 System Components Related to Boot

The Boot module is activated upon the following events:

- Hardware RESET (assertion, followed by de-assertion of RESET_IN)
- Software RESET (a write to the RST_CTL register asserting PNX8526 soft reset)
- PR3940 MIPS watchdog timeout

Once activated, the Boot module goes through the following event sequence:

- Delays a few cycles to allow GPIO[2:0] pins to tri-state and pullup/down to stabilize.
- Samples the logic value on the BOOTMODE[2:0] pins to determine which boot script to execute.

- Executes the chosen script, which may be one of two built-in scripts or an external I^2C EEPROM script. Script entries cause the boot block to perform writes across the PI-Bus to initialize MMIO registers and memories.
- Terminates Boot and sets the Boot_Status MMIO register to “Boot Over.”

3.2 Symbolic Boot Language

Boot scripts are written using a simple symbolic language as shown below.

```
// PNX8526 boot script example

mem1 write 00047250 00000003 // enable clk_pci in CLK_PCI_CTL
mem1 write 00047200 00000003 // enable clk_mem in CLK_MEM_CTL
// ... some words omitted ...
mem1 write 0004080c 000002f7 // start DMA from flash to DRAM
idle 324000              // wait 12 mSec for DMA to complete
```

In this simple language, “mem1 write <hex1> <hex2>” stands for a write relative to the current MMIO_BASE i.e. a write of value <hex2> to the device control register with MMIO offset <hex1>.

The command “mem0 write <hex1> <hex2>” stands for a write relative to SDRAM base i.e., a write that fills SDRAM at address <hex1> with a value <hex2>. The store operation follows endian mode rules, that is...

- for hex2 = 0xabcdefgh, and operating mode is little-endian (mode at start of boot), the value “gh” will end up at byte address <hex1>, “ef” at <hex1+2>, etc.
- for hex2 = 0xabcdefgh, and operating mode is big-endian, the value “ab” will end up at byte address <hex1>, “cd” at <hex1+2>, etc.

The command “idle <decimal1>” stands for delay <decimal1> 27 MHz clock ticks. A “//” starts a comment that lasts till the end of the line. Empty lines are allowed anywhere.

Other Issues

- A store to RST_CTL may change the value of endian mode during the boot process and hence, the interpretation of the next “mem0” command. Provide at least 10 μs delay between such a store and the next “mem0 write.”
- A store to PCI “base_address_10_image” MMIO register (only allowed if PCI configuration management is enabled), establishes the DRAM_BASE address and hence, should be performed before any “mem0” commands are issued.
- A store to PCI “base_address_14_image” MMIO register (only allowed if PCI configuration management is enabled) changes the MMIO_BASE address. Provides at least 10 μs delay between such a store and the next “mem1 write.”

A tool is available that translates the PNX8526 symbolic language to a binary image suitable for EEPROM programming. Contact a Philips PNX8526 sales representative. Read the tool documentation carefully to see how the latest release of the tool deals with the above issues.

3.3 Internal vs. External Host Boot

The following table describes the BOOTMODE[2:0] settings for selecting either an internal or external boot script.

Table 1: Boot Mode and Choice of Boot Script

BOOTMODE [2:0]	Script
000	External EEPROM boot script
001	Built-in script #1: 32 MB of SDRAM at 100 MHz. Start MIPS from random-access, non-volatile memory (Flash, ROM...) attached to XIO profile 0.
010	Built-in script #2: same as script #1, except start MIPS in SDRAM after transferring 8 kB of boot code from NAND-Flash to SDRAM.
011	External EEPROM boot script with fast IIC clock, testmode
100	Built-in script #3: fast boot script used for manufacturing
Other	Reserved for future extension.

In most cases, the PNX8526 is used with the internal PR3940 MIPS host, which means it can boot from a built-in boot script without the need for a boot EEPROM. This is shown on the right side of Figure 2. In this case, a Flash, ROM or other non-volatile memory device stores all boot software code for the PNX8526. This is the common method of booting that will be described first in Section 3.4.

In cases where built-in scripts are not applicable e.g., if a non-standard DRAM configuration is required or a different type of NAND- Flash is used, a boot EEPROM is required. This will be covered in Section 3.5.4.

In a configuration with an external host CPU, a serial boot EEPROM is always required. It contains board level personality data needed by the host BIOS. In such a case no Flash/ROM needs to be associated with the PNX8526. The host CPU loads the PNX8526 application code from its Flash or disk, fills SDRAM and starts the TM32 CPU. Refer to Section 3.5.5 for details on external host boot

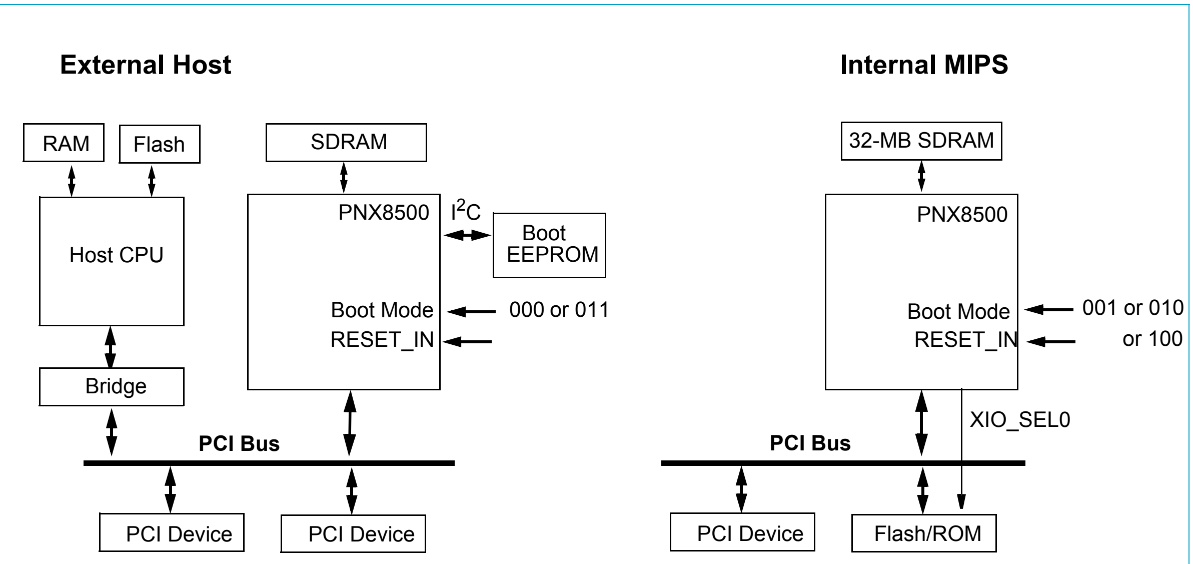


Figure 2: External Host vs. Internal MIPS Boot Environment

3.4 Built-In Boot Scripts

3.4.1 MIPS Reset Vector Re-Direction

The PNX8526 contains a special provision to allow re-directing of the MIPS reset vector. This special provision is explained in Section 2.7.1.1 of Chapter 2 System Memory Map and Protection as well as in Chapter 35 PI-Bus Architecture in the F_PI Bus Controller registers. The boot scripts below use this mechanism.

3.4.2 Built-In Boot Script #1 (Boot Mode 001

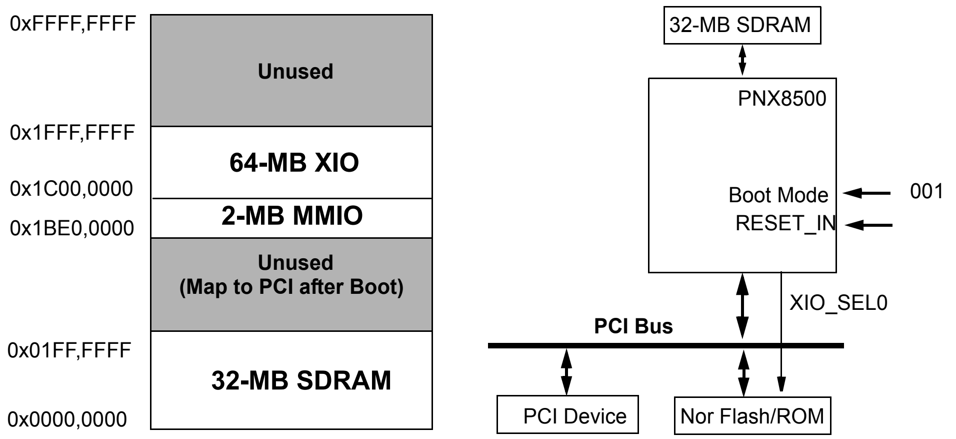


Figure 3: System Configuration and Physical Memory Map for Script #1

```
//    PNX8526 built-in script #1
//
//      - configure memory as 32 MByte, running at 100 MHz
// - set F-PI bus clock to 100 MHz
// - memory map (standard config for internal MIPS):
// SDRAM/Base10 at 0x0000.0000
// MMIO /Base14 at 0x1be0.0000 (2M - Set against XIO)
// XIO  /Base18 at 0x1c00.0000 (64M required by WinCe)
// - set MMI refresh corresponding to 100MHz SDRAM.
// - start internal MIPS direct execution from non-volatile
//   memory attached to XIO chip-select 0(NorFlash,ROM ...)
//
//
//  o Clock Module Setting
//     - clk_mem, clk_fpi running at 100MHz
//       only clk_fpi, clk_mips phase alignment required.
//     - clk_tpi, clk_mpi always running at 1/2 clock f_pi
//       and phase shifted 180 degrees
//
//    o enable clk_pci by writing to CLK_PCI_CTL   (0x047250)
mem1 write 00047250 00000003
//    o enable clk_mem by writing to CLK_MEM_CTL   (0x047200)
mem1 write 00047200 00000003
//    o enable clk_fpi by writing to CLK_FPI_CTL   (0x047204)
mem1 write 00047204 00000005
//    o enable clk_tpi by writing to CLK_TPI_CTL   (0x04720C)
mem1 write 0004720C 00000003
//    o enable clk_mpi by writing to CLK_MPI_CTL   (0x047210)
mem1 write 00047210 00000003
//
//  - All other clocks are running at reset frequency 27MHz
//
//  o MMI Enable
//    o Write to mm_enable to enable SDRAM interface.
mem1 write 0004d414 00000001
//
//  o PCI Settings
//    o Write to pci setup register
//      o enable pci system arbitration
//      o enable config mgmt
//      o expansion rom disabled
//      o base10 16/32/64MB, prefetchable, enabled
//      o base14  2MB, non-prefetchable, enabled
//      o base18 64MB, non-prefetchable, enabled
mem1 write 00040010 01d60e03   #If 32MB Base10 aperture
//      mem1 write 00040010 01d60e83   #If 64MB Base10 aperture
//      mem1 write 00040010 01d60d83   #If 16MB Base10 aperture
//
//    o Enable memory spaces and mastering on PCI
//      (Due to PCI specification - this can not be done per default)
mem1 write 00040044 00000006 0
//
//  o Set GPIO settings to allow XIO profile 0 accesses
//    (This could be done per default - But it is not desirable
//     since this would change the GPIO defaults and make the
//     implementation less generic and less flexible for if
//     the XIO profile 0 was actually being used as GPIOs
mem1 write 0010400c 01551500 0 // GPIO reg MC3
//
//  o Enable NOR flash or ROM on profile 0:
//    o Write to xio sel0 profile register
//      Don’t use ack, No. of wait states -> 6 (state machine adds 2
// more clocks, which will yield 240 ns, sel0_offset ->0,
// sel0_type -> NOR and enabled.
mem1 write 00040814 00000c09 0
//  o Setup mips reset vector relocation to execute from 0x1c00.0000
mem1 write 0003F020 3C08BC00 0 // move mips boot from
//   A010 > 1C00 + A000 = BC00
//
//
//  o Release MIPS reset, it starts running little-endian
mem1 write 00060000 00000008
//  o we’re done, end of script leads to generation of end of boot
```

3.4.3 Built-In Boot Script #2 (Boot Mode 010)

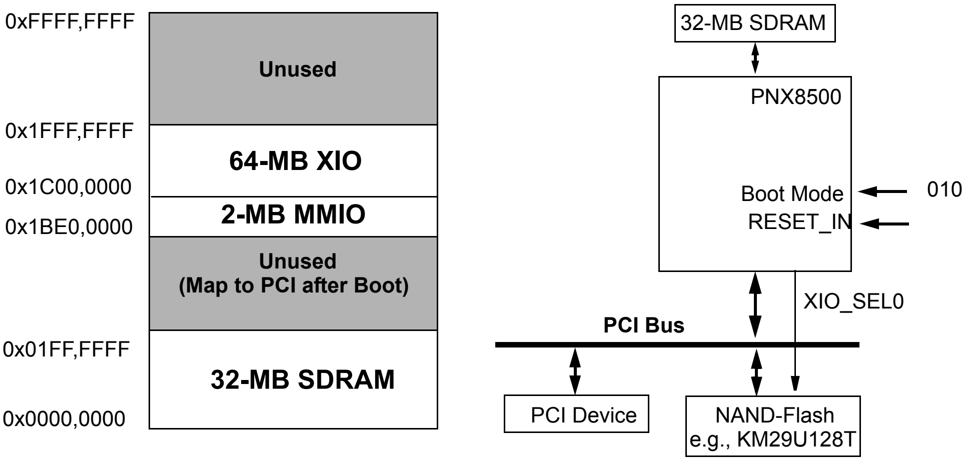


Figure 4: System Configuration and Physical Memory Map for Script #2

```
//    PNX8526 built-in script #2
//      - configure memory as 32 MByte, running at 100 MHz
// - set F-PI Bus clock to 100 MHz
// - memory map (standard internal MIPS config):
// SDRAM/Base10 at 0x0000.0000
// MMIO /Base14 at 0x1be0.0000 (2M - Set against XIO)
// XIO  /Base18 at 0x1c00.0000 (64M required by WinCe)
// - set MMI refresh corresponding to 100MHz SDRAM.
// - configure for a Samsung KM29U128T Nand Flash on XIO chip select 0
// - transfer the first 8 kByte of Flash to SDRAM at 0x0010,0000
// - start internal MIPS from SDRAM at 0x0010,0000
//
```

```
//
//  o Clock Module Setting
//     - clk_mem, clk_fpi running at 100MHz
//       only clk_fpi, clk_mips phase alignment required.
//     - clk_tpi, clk_mpi always running at 1/2 clock f_pi
//       and phase shifted 180 degrees
//
//    o enable clk_pci by writing to CLK_PCI_CTL   (0x047250)
mem1 write 00047250 00000003
//    o enable clk_mem by writing to CLK_MEM_CTL   (0x047200)
mem1 write 00047200 00000003
//    o enable clk_fpi by writing to CLK_FPI_CTL   (0x047204)
mem1 write 00047204 00000005
//    o enable clk_tpi by writing to CLK_TPI_CTL   (0x04720C)
mem1 write 0004720C 00000003
//    o enable clk_mpi by writing to CLK_MPI_CTL   (0x047210)
mem1 write 00047210 00000003
//
//     - All other clocks are running at reset frequency 27MHz
//
//  o MMI Enable
//    o Write to mm_enable to enable SDRAM interface.
mem1 write 0004d414 00000001
//
//
//  o PCI Settings
//    o Write to pci setup register
//      o enable pci system arbitration
//      o enable config mgmt
//      o expansion rom disabled
//      o base10 16/32/64MB, prefetchable, enabled
//      o base14  2MB, non-prefetchable, enabled
//      o base18 64MB, non-prefetchable, enabled
mem1 write 00040010 01d60e03   #If 32MB Base10 aperture
//  mem1 write 00040010 01d60e83   #If 64MB Base10 aperture
//  mem1 write 00040010 01d60d83   #If 16MB Base10 aperture
//
//    o Enable memory spaces and mastering on PCI
//      (Due to PCI specification - this can not be done per default)
mem1 write 00040044 00000006 0
//
//  o Set GPIO settings to allow XIO profile 0 accesses
//    (This could be done per default - But it is not desirable
//     since this would change the GPIO defaults and make the
//     implementation less generic and less flexible for if
//     the XIO profile 0 was actually being used as GPIOs
mem1 write 0010400c 01551500 0 // GPIO reg MC3
//
//  o Transfer Initial code for MIPS boot to
//    SDRAM by programming DMA utility device. Default DMA
//    start address is 0x1c00.0000, dma_lenght is 8kB and
//    the default SDRAM address is 0x0010.0000
//    o Write to xio sel0 profile register
//      o enable xio profile
//      o 8MB sel0_size
//      o Nand Flash
//      o 0 offset from xio aperture
//      o use_ack
//      o REN_lo=REN_hi=120ns
//      o WEN_lo=WEN_hi=120ns
mem1 write 00040814 006a8011 0
//
//    o Kick off DMA by writing to DMA controls register
//      o Command type write
//      o Initiate DMA transaction
//      o Max burst size 128 data phases
//      o linear increment of address
//      o send to xio
mem1 write 0004080c 00000296
//
//    o Wait until transfer is complete in 12ms
//      (Calculations by indicate this is over in 6 ms)
idle 324000
//
//  o Release MIPS reset, it starts running little-endian at0x0010,0000
mem1 write 00060000 00000008
// o we're done, end of script leads to generation of end of boot
```

3.4.4 Built-In Boot Script #3, for Boot Mode 100

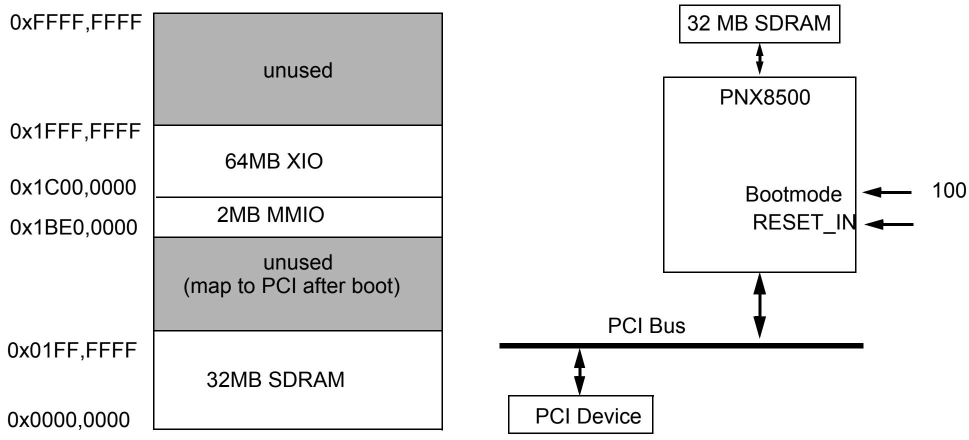


Figure 5: System Configuration and Physical Memory Map for Script #3

```
//    PNX8500 built-in script #3
//      - configure memory as 32 MByte, running fromexternal clock source
//- set F-PI bus clock running from external clock source
//- memory map (standard internal MIPS config):
// SDRAM/Base10 at 0x0000.0000
// MMIO /Base14 at 0x1be0.0000 (2M - Set against XIO)
// XIO  /Base18 at 0x1c00.0000 (64M required by WinCe)
//- The purpose of this script is to do fast boot for functional test.
//- Enabling required clocks to run from external source
//- Release sys_rst_out_n for external PCI master
//
//
//  o Clock Module Setting
//     - clk_mem, clk_fpi running from external source
//       only clk_fpi phase alignment required.
//     - clk_tpi, clk_mpi always running at 1/2 clock f_pi
//       and phase shifted 180 degrees
//
//    o enable clk_pci by writing to CLK_PCI_CTL   (0x047250)
mem1 write 00047250 00000005
//    o enable clk_mem by writing to CLK_MEM_CTL   (0x047200)
mem1 write 00047200 00000005
//    o enable clk_fpi by writing to CLK_FPI_CTL   (0x047204)
mem1 write 00047204 00000009
//    o enable clk_tpi by writing to CLK_TPI_CTL   (0x04720C)
mem1 write 0004720C 00000003
//    o enable clk_mpi by writing to CLK_MPI_CTL   (0x047210)
mem1 write 00047210 00000003
//
//     - All other clocks are running at reset frequency 27MHz
//
//  o MMI Enable
//    o Write to mm_enable to enable SDRAM interface.
mem1 write 0004d414 00000001
//
//
//  o PCI Settings
//    o Write to pci setup register
//      o enable pci system arbitration
//      o enable config mgmt
//      o expansion rom disabled
//      o base10 16/32/64MB, prefetchable, enabled
//      o base14  2MB, non-prefetchable, enabled
//      o base18 64MB, non-prefetchable, enabled
mem1 write 00040010 01d60e03   #If 32MB Base10 aperture
// mem1 write 00040010 01d60e83   #If 64MB Base10 aperture
// mem1 write 00040010 01d60d83   #If 16MB Base10 aperture
//
//    o Enable memory spaces and mastering on PCI
//      (Due to PCI specification - this can not be done per default)
mem1 write 00040044 00000006 0
//
//  o Set GPIO settings to allow XIO profile 0 accesses
//    (This could be done per default - But it is not desirable
//     since this would change the GPIO defaults and make the
//     implementation less generic and less flexible for if
//     the XIO profile 0 was actually being used as GPIOs
mem1 write 0010400c 01550000 0 // GPIO reg MC3
//
//  o PCI Settings
//    o Write to sub_system and vendor ID register
//      o enables pci interface
mem1 write 0004006c 12345678
//
// o Release SYS_OUT reset,
mem1 write 00060000 00000002
// o we're done, end of script leads to generation of end of boot
```

3.5 External Boot Scripts

3.5.1 External I^{2}C Boot EEPROM Types

The PNX8526 Boot module supports all I^{2}C EEPROMs (sometimes called ‘2 wire’ EEPROMs) that use a 1-byte or 2-byte address protocol and respond to an I^{2}C device code 1010_2. Subtle differences exist between devices, in particular:

- Avoid devices that have partial array write protection, since such devices could be overwritten by accidental or intentional I^{2}C writes, causing boot failure on next reset.
- Some devices may have additional functionality that is useful.
- Availability may vary.
- Programming protocols may vary.

Table 2 lists a variety of devices currently on the market. This list is by no means exhaustive, nor has operation of these devices been verified.

Table 2: Examples of I^{2}C EEPROM Devices

Size	Device	Write Protection Coverage	Address Protocol	Comments
128 Bytes	ATMEL® 24C01A	full array	1 byte	suitable
128 Bytes	Philips PCA8581C	none	1 byte	not recommended due to lack of write protection
256 Bytes	ATMEL 24C02	full array	1 byte	suitable
256 Bytes	Philips PCF85102C-2	none	1 byte	not recommended due to lack of write protection
512 Bytes	ATMEL 24C04	full array	1 byte	suitable
1 kB	ATMEL 24C08	full array	1 byte	suitable
2 kB	ATMEL 24C16	upper 1 kByte	1 byte	tested, but not recommended, first 1 kB can be overwritten
2 kB	Philips PCF85116-3	full array	1 byte	suitable
2 kB	Summit SMS8198	full array	1 byte	tested and suitable, includes power on reset circuit suitable for board level reset generation
4 kB	ATMEL 24C32	upper 1 kByte	2 bytes	not recommended, first part can be overwritten
8 kB	ATMEL 24C64	upper 2 kByte	2 bytes	not recommended, first part can be overwritten
16 kB	ATMEL 24C128	full array	2 bytes	suitable
32 kB	ATMEL 24C256	full array	2 bytes	tested and suitable
64 kB	ATMEL 24C512	full array	2 bytes	suitable

The Boot module has a limited intelligence I^2C interface:

- It does not arbitrate for the I^2C bus. No other masters are allowed to be active on this bus upon reset.
- It runs at a fixed 330-kHz frequency (derived from the 27-MHz PNX8526 xtal).
- It supports slave clock stretching.
- It can handle EEPROMs with both 1-byte and 2-byte addressing formats (see Section 3.5.3).

3.5.2 External EEPROM Boot Script Binary Format

The boot script consists of a sequence of 32-bit commands. Valid commands are:

- Write a given 32-bit value at a given address (useful for writing device control registers).
- Write an arbitrary length list of 32-bit values starting at a given address (useful for filling memory with a processor binary image).
- Delay by a given number of 27-MHz clock ticks (useful to wait for completion of an action, such as a PLL frequency change, or a device DMA operation).
- Terminate Boot.

For a 2048-byte or smaller EEPROM, the script must start at byte address 1 (not 0). For a 4096-byte or larger EEPROM, the boot script must start at byte address 0. For clarification, refer to Section 3.5.3.

Each set of four successive bytes is assembled into a 32-bit word value (the byte read first ends up as the least significantChapter 4 Endian Mode byte (lsbyte), etc.). The 32-bit words are then interpreted as commands, as shown in Table 3.

Table 3: Boot Script Binary Format

LSBytes of First Command Word	Command	Encoding (Bits)
....00	Write 32-bit value 'v' at 32-bit word address 'a'	word1: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa00 word2: vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
....01	Write an arbitrary length list of 32-bit values starting at address 'a'	word1: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa01 word2: length word3: last: $value_k$
....0010	Delay by 'n' 27-MHz ticks $0<=n<=2^{28}-1$	word1: nnnnnnnnnnnnnnnnnnnnnnnnnnnn0010
....0110	Terminate boot process, stop reading EEPROM	word1: xxxxxxxxxxxxxxxxxxxxxxxxxxxx0110
other	Reserved for future extension	

3.5.2.1 Binary Boot Script Endian Mode Issues

Remark: 32-bit words “v” are written across the PI-Bus exactly as they are assembled by the Boot module i.e., the msbits of the 32-bit word are transmitted as msbits on the PI-Bus, etc.

When writing to an MMIO register address, there is no endian mode issue. The msbits of the word “v” end up as the msbits of the MMIO register.

When writing to an SDRAM address there is an endian mode issue. Depending on the current endian mode (which can be changed by boot itself when it writes to the RST_CTL MMIO register), 32-bit PI-Bus words get written to memory in one of two ways:

- In little-endian mode, the msbits of “v” (or last read EEPROM byte of the word), end up in memory byte address “a+3” and lsbits (or first read EEPROM byte), end up at address “a.”
- In big-endian mode, the msbits of “v” (or last read EEPROM byte), end up at address “a” and the lsbits (or first read EEPROM byte), end up at address ‘a+3’. For clarification, refer to Chapter 4 Endian Mode .

3.5.3 Details on I^{2}C Operation

In order to retrieve the boot script, the Boot module performs the following across I^{2}C:

- START, 1010000, wait-for-ack, 00000000, wait-for-ack, STOP.
- START, 1010001, wait-for-ack, <accept data byte, send ACK or STOP if done>.

The interpretation of this sequence by 2048 bytes or smaller EEPROMs is:

- Write a byte value 0 to address 0 (setting the next address-pointer to byte address 1).
- Read, starting from address 1.

Hence, for a 2048-byte or smaller EEPROM, the boot image must start at byte 1.

The interpretation of this sequence by 4096 bytes or larger EEPROMs is:

- Write a 0 byte-long sequence to address 0 (setting next address pointer to byte address 0).
- Read, starting from address 0.

Hence, for a 4096-byte or larger EEPROM, the boot image must start at byte 0.

3.5.4 Internal Host Booting Using an External Script

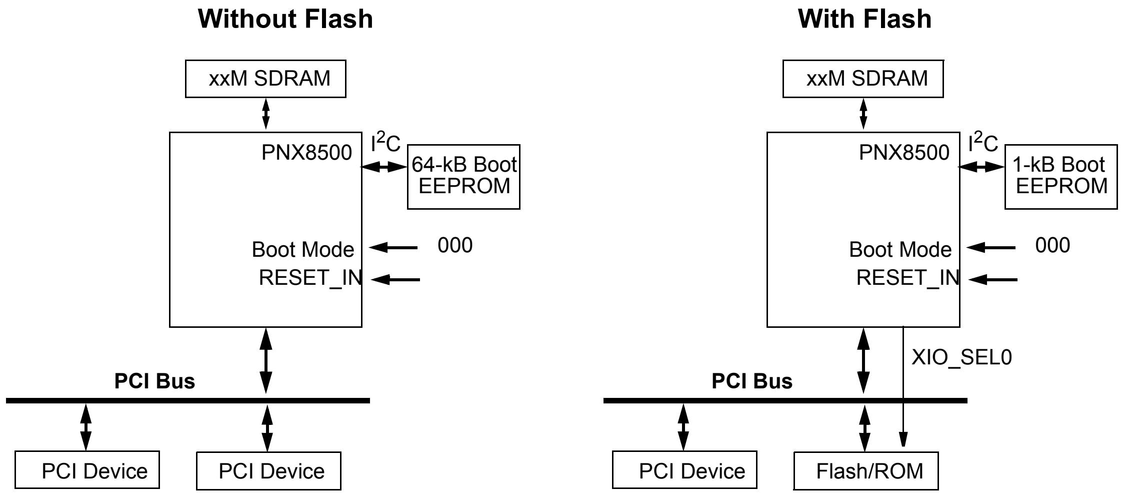


Figure 6: Custom Boot: Without Flash — With Flash

If none of the built-in scripts is suitable e.g., due to a different type of NAND-Flash or a different memory organization, an external serial boot EEPROM is required. Depending on the application characteristics, this can be a small (1 kB or less) EEPROM that contains a small boot script and starts MIPS from Flash/ROM. Alternately, a large serial EEPROM can be used that contains e.g., a complete disk file system or upload capability from another device, etc.

3.5.5 External Host Boot

It is possible to use the PNX8526 in a configuration where an external CPU, such as an external MIPS, X86, PowerPC, or SH4 is the host. In that case, the PNX8526 behaves as a plug-in PCI device. Most of the responsibility of booting is taken care of by the host PCI BIOS and by the PNX8526 driver on the host. However, there is still a requirement for a boot script in order to initialize the hardware and get it ready to act as a recognizable PCI device.

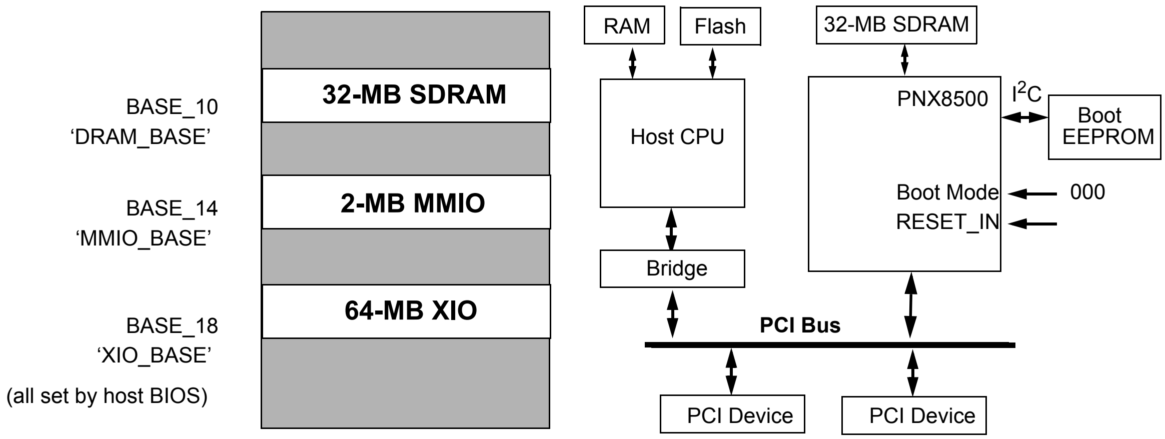


Figure 7: External Host Boot System Environment and System Memory Map

External host boot differs significantly from all other boot scripts previously described. It requires an external EEPROM. This EEPROM has the responsibility to bring the system into a good initial state and to personalize certain data:

- Subsystem Vendor ID, a 16-bit value that identifies the 'board' vendor. Philips has code 0x1131. Manufacturers of PCI plug-in cards for the open market must obtain and use their own ID (from the PCI Special Interest Group). See PCI Local Bus Specification, Rev2.2, Chap 6.
- Subsystem ID, a 16-bit value established by the board vendor to identify a particular board. Allocated by the vendor's PCI Special Interest Group representative. This value is used to identify a suitable driver for PC plug-and-play.
- Sizes for all apertures that must be given unique physical addresses at PCI BIOS device enumeration time (DRAM, MMIO and XIO for the PNX8526).

Remark: The aperture sizes are written by boot into PCI block registers. The host PCI BIOS software retrieves the values by a write, followed by a readback to the PCI Configuration space base address registers. It then assigns a suitable value to each base address. See PCI Local Bus Specification, Rev2.2, Section 6.2.1.5 for more details.

3.5.5.1 Example of External Host Boot Script

```
// Example PNX8526 external host boot script (untested)
//
//This script will:
//  - set size of all PCI apertures (SDRAM, MMIO and XIO)
//- initialize Subsystem Vendor ID and Subsystem ID
//- set up clocks to suitable values
//- stop (host will do the rest)
//
//Then the host BIOS will:
//- read size of all PNX8526 PCI apertures
```

```
// - assign a base to each aperture
//
//Then the PNX8526 driver will:
// - set all registers to establish TM32 and PI-Bus memory map
// - configure all other key registers
// - download code into SDRAM
// - point the TM32 at it, and start it
//
// set clk_mem, clk_fpi to 100 MHz, clk_tpi, clk_mpi to 71.5 MHz
mem1 write 00047200 00000003 // select PLL0 output as clk_mem
mem1 write 00047204 00000005 // set clk_fpi equal to clk_mem
mem1 write 0004720c 00000003 // set clk_tpi to functional clock
mem1 write 00047210 00000003 // set clk_mpi to functional clock
//
// enable SDRAM interface
mem1 write 0004d414 00000001
//
// PCI settings
// - enable target abort
// - enable PCI to memory highway interface
// - enable XIO (for this example)
// - 64 MByte XIO, non-prefetchable
// - 2 MByte MMIO, non-prefetchable
// - 32 MByte DRAM, non-prefetchable
// - disable config mgmt (host bridge is PCI config mgr)
mem1 write 00040010 01d64a00
// - enable memory spaces and mastering
mem1 write 00040044 00000006
//
// Define personality data (EXAMPLE C !!!!!!!!)
// - Philips: 0x1131
// - Philips board number 0x00aa (EX. ONLY, not a real code)
mem1 write 0004006c 00aa1131
// that’s it - rest done by host !
```

3.6 Boot Stages and Responsibilities

Stage 0 is the stage that starts MIPS execution in a mode that is conservative (slow SDRAM). Much more than stage 0 is involved to get an actual application running at full speed.

Remark: In case of boot script #1, the MIPS ends up in direct execution from non-volatile XIO memory. This is a slow mode of execution. It is much better to bring code into SDRAM and execute it there.

3.6.1 After Stage 0

Software on the MIPS is responsible for further system bringup:

- Optimize the operating speed of the system.
- Set system endian mode.

- Read the file containing the Flash application image into SDRAM.
- Execute it.

3.6.2 Checklist

To get to full-speed application execution, the different stages have to accomplish the following steps together:

- Set endian mode.
- If direct executing from XIO, copy (part of) code to SDRAM and execute from there.
- Change all clocks to optimum value (this assumes that the code image is built with knowledge of the actual MIPS, TM32 and SDRAM speed operating capabilities).
- Change MMI refresh setting to optimal.
- Configure the F-PIMI (posted writes, enable busy/ext_wb_empty generation in the F-PI Bus controller).
- Configure PI_DRAM_LO/HI to actual SDRAM memory size.
- Configure all bridges.
- Release sys_rst_out_n so that board level peripherals start operation.
- Configure PCI subsystem vendor ID register.
- Configure TM32.
- Configure the MMI arbiter.
- Configure powerdown/up configuration (en_mips_coma_sf in mm_self_refresh).
- Configure clocks for all on-chip peripherals.
- Configure io_mux register in global registers.

3.7 Register Descriptions

The base address for the PNX8526 Boot module is 0x04 2000.

BOOT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 2000			*Boot Status*	
31:1	R	31’b0	Reserved	
0	R	0	Boot Status	1 = Boot over 0 = Boot in progress
Offset 0x04 2FFC			*Module ID*	
31:16	R	0x010A	Boot Module ID	These bits represent the Boot Module ID.
15:12	R	1	Major Revision	Major Revision number of the module
11:8	R	0	Minor Revision	Minor Revision number of the module
7:0	R	0	Aperture Size	Aperture Size is 4 kB.

Chapter 4: Endian Mode

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

4.1 Introduction

Two addressing conventions exist in the computer industry: little-endian and big-endian.

- In the little-endian convention, a multi-byte number is stored in memory with the least significant byte at the lowest memory address, and subsequent bytes at increasing addresses.
- In the big-endian convention, the most significant byte is stored at the lowest memory address, and subsequent bytes are stored at increasing addresses.

The PNX8526 supports both big-endian mode and little-endian mode, allowing it to run either little-endian or big-endian software, as required by the particular application.

This chapter explains the concepts and programmer's view of data structures required for peripheral control and peripheral DMA. The chapter also contains a section that shows how hardware blocks, buses and bridges implement the programmer's view.

4.2 Feature Summary

- The PNX8526 supports big-endian and little-endian operation.
- The system as a whole (CPUs and on-chip DMA peripherals) must operate in the same endian mode.
- When used with an external CPU, the PNX8526 must operate in the same endian mode as the external CPU.
- The endian-mode choice is made at boot time. It can be changed by software, but this requires a partial system reset.

4.3 Endian Mode Theory

There are two basic laws of endian mode: one imposed by CPU history, and one by convention. Both must be met by any system architecture that implements dual-endian operation capability. In addition, there are some implementation choices for a system architecture. Section 4.6 explains the choices that were made for the PNX8526 on-chip buses. These choices are somewhat arbitrary, but they must be followed to ensure future compatibility.

4.3.1 Law 1: The “CPU Rule”

This section is intended to explain CPU endian modes in detail. For those familiar with CPUs and endian modes, it is optional reading.

The following summarizes how CPUs and byte-addressable memory operate:

- When storing a “n byte” size item from a CPU register to memory at address “A,” the bytes modified are always the bytes with byte address “A”..”A+n-1.”
- In little-endian mode, the byte at address “A” receives the least significant bits of the multi-byte item.
- In big-endian mode, the byte at address “A” receives the most significant bits.

Consider the following example a (hypothetical) C struct:

```
struct {
UInt8 C;// “command” byte
UInt8 F;// “flags” byte
UInt16 L;// “length” 16 bit value
UInt32 A;// “address” 32 bit value
} DMA_Descriptor;
```

Remark: This is based on an example in the Apple® publication,”Designing PCI Cards and Drivers for Power Macintosh Computers”.

A compiler would assign byte offsets as follows: C:0, F:1, L:2, A:4. This assignment is independent of system endian mode. Figure 1 and Figure 2 show the two layout views.

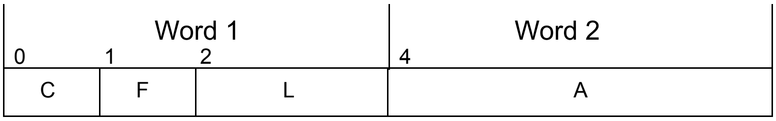


Figure 1: Big-Endian Layout of DMA_Descriptor

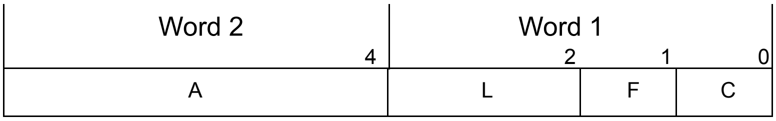


Figure 2: Little-Endian Layout of DMA_Descriptor

The PR3940 and TM32 CPUs on the PNX8526 both support 8, 16 and 32-bit data types and a memory system that is byte addressable. Both CPUs support a big-endian and little-endian mode of operation. The effect of a CPU store instruction on memory is defined in Table 1. As an example, a 16-bit store operation always stores the 16-bit quantity contained in the 16 lsbits of the CPU register. And the memory locations affected are “a” and “a+1.” But which byte goes where is dependent on endian mode.

Table 1: Memory Result of a Store to Address ‘a’ Instruction

Endian Mode	R13 Content	Data Size	Result of Store$_{size}$(R13, Address a)
little	0x04050607	8 bits	m[a] = 0x07
little	0x04050607	16 bits	m[a] = 0x07; m[a+1] = 0x06
little	0x04050607	32 bits	m[a] = 0x07; m[a+1] = 0x06; m[a+2] = 0x05; m[a+3] = 0x04
big	0x04050607	8 bits	m[a] = 0x07
big	0x04050607	16 bits	m[a] = 0x06; m[a+1] = 0x07
big	0x04050607	32 bits	m[a] = 0x04; m[a+1] = 0x05; m[a+2] = 0x06; m[a+3] = 0x07

The effect of a CPU load instruction on a register is defined for unsigned and signed loads in Table 2 and Table 3.

Remark: A load always sets all bits of the CPU register. In the case of an unsigned load, higher order bits are filled with zeroes. In the case of a signed load, higher order bits are filled with the sign bit of the data item loaded.

Table 2: Register Result of an (Unsigned) Load Instruction

Memory Content	Endian Mode	Data Size	Register Value Result of Load$_{size}$(Address a)
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	8 bits	0x000000AA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	16 bits	0x0000BBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	32 bits	0xDDCCBBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	8 bits	0x000000AA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	16 bits	0x0000AABB
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	32 bits	0xAABBCCDD

Table 3: Register Result of a (Signed) Load Instruction

Memory Content	Endian Mode	Data Size	Register Value Result of Load$_{size}$(address a)
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	8 bits	0xFFFFFFAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	16 bits	0xFFFFBBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	32 bits	0xDDCCBBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	8 bits	0xFFFFFFAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	16 bits	0xFFFFAABB
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	32 bits	0xAABBCCDD

An interesting example is the small C program below that determines whether the program runs on a big-endian or little-endian mode machine.

```
int w = 0x04050607;
char *a = (char *) &w;
if (*a == 0x04) printf(“big-endian”); else printf(“little-endian”);
```

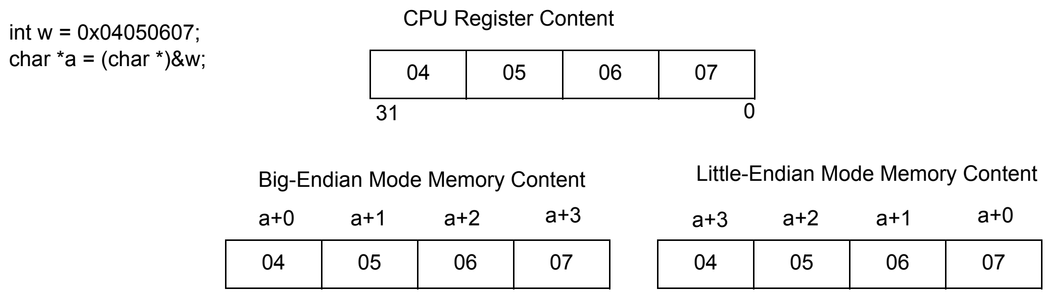


Figure 3: Memory Content Created by the C Program

4.3.2 Law 2: The “DMA Convention Rule”

The DMA convention rule says that “when a stream of items enters the system, items should be placed in memory such that an item that arrived later has a higher address value.” On output, a similar convention holds—items sent first are those with the lowest addresses.

A variant of this rule relates to the storage of images. Pixels from left to right have increasing addresses. Lines from top to bottom have increasing addresses.

This is a convention that keeps programmers sane. It may also be seen as arbitrary, but obviously the best choice between two alternatives.

A more precise version of this rule is:

If item ’0’ of a DMA item stream is placed at address “A,” item “i” of a DMA stream should be placed at byte address “A+i*s,” where “s” is the item size in bytes.

For an example of this rule, refer to Section 4.5.

4.4 PNX8526 Endian Mode Architecture Details

The programmer’s view of the PNX8526 endian architecture is as follows:

- The CPUs, peripherals and coprocessors on the PNX8526 store and retrieve audio samples, image pixels and data observing both the CPU rule and DMA rule.
- The system as a whole runs in either little-endian or big-endian mode.
- The mode is determined by the “sys_big_end” bit in the RST_CTL register of the Reset module, which is “exported” to other modules.
- The value of this bit is set during system initialization.

4.4.1 Global Endian Mode

Both CPUs and all peripherals always operate in a single endian mode. This endian mode is determined by the sys_big_end bit in the RST_CTL register of the PNX8526 Reset module. The value of this bit is set during system boot and normally not changed afterwards. It is possible for software to change this bit after boot, by writing to the RST_CTL register with a pattern that contains the new sys_big_end value and a ‘1’ in the pulse_mips_reset bit position of the same register. This causes a reset of the MIPS, which then comes back up in the designated new endian mode.

Table 4: RST_CTL.sys_big_end Bit

RST_CTL.sys_big_end	PNX8526 Endian Mode	PI-Bus Big-Endian Signal
0	little-endian	0
1	big-endian	1

Remark: The TM32 CPU core endian mode is determined by a bit in its PCSW. This is historically set by the “crt0.s” software module on the TM32 CPU core, which initializes the PCSW.

The PNX8526 version of this software module is responsible for reading the RST_CTL.sys_big_end bit value and establishing the same TM32 CPU core endian mode as the rest of the system.

4.4.2 Peripheral Control

All peripheral blocks have Control and Status registers, accessed by CPU Programmed I/O. In the PNX8526, all programmed I/O happens through Memory Mapped I/O registers. A CPU can access device Control and Status registers by using the correct MMIO address for a device register. In the PNX8526, all device registers are 32 bits wide and may only be accessed through 32-bit load/store operations.

A control/status register load/store always copies the 32 bits verbatim between a CPU register and the device register. The device’s left-most msb (bit 31) ends up in the CPU’s left-most msb (bit 31), and the device’s right-most lsb (bit 0) ends up in the right-most CPU register bit. This happens regardless of system endian mode settings.

MMIO load and store instructions always see the same bit layout of device MMIO registers, regardless of endian mode. The field and bit layout is precisely as specified in the device register table with bit 31 designating the msb and bit 0 the lsb.

Remark: The packing and ordering of packed bit structure fields in C compilers are not precisely defined. Typically, big-endian C compilers pack fields from left (msb) to right (lsb). Little-endian C compilers pack from right to left. Because of this and also because of inherent inefficient code when accessing structure fields, it is not recommended to use C structure declarations to access MMIO register fields.

4.4.3 Peripheral DMA

Every DMA capable peripheral in PNX8526 observes the PI-Bus big-endian signal, and therefore the global RST_CTL.sys_big_end value, to determine how to write each data item or unit to memory.

An example of a unit would be:

- 16-bit audio sample
- 32-bit audio sample
- 32-bit unit containing a RGBa 8888 true color pixel with alpha value.

The peripheral performs byte swapping within units as needed, and packs units as needed for transmission across on-chip buses. Byte swapping is done in such a fashion that 8, 16 and 32-bit units always end up in memory bytes in the form prescribed by the CPU rule. Successive item packing are placed in incrementing addresses, as designated by the DMA rule.

For more information on what constitutes a “unit,” and precise endian mode views of image and audio data, refer to Chapter 7 Pixel Formats, and to the audio peripheral documentation in Chapter 20 Audio Input Ports and Chapter 21 Audio Output Ports.

There are different classes of DMA devices. A device such as a UART always deals with a single unit size (a character). A device such as an Audio Out module deals with a single unit size *in a given operating mode*—16-bit/sample or 32-bit/sample mode, for example. The module is disabled and reset between mode changes. A device dealing with transport streams may have to deal with byte and 32-bit mixed data.

4.4.4 SIMD Programming Issues

The peripheral DMA hardware architecture ensures that software dealing with loads and stores of unit size data can be written in a way that is oblivious to the endian mode.

With the current TM32 CPU core, this is not possible for software that performs Single Instruction Multiple Data (SIMD) style programming using multimedia operations. Consider the case of a FIR filter, operating on 16-bit sample units, but using 2-at-a-time load/store/multiply operations. Which of the two 16-bit halfwords is the earlier sample?

- For big-endian mode, the msb halfword contains the earlier sample.
- For little-endian mode, the lsb halfword contains the earlier sample.

The current TM32 CPU core on PNX8526 requires that SIMD software be written aware of endian mode.

4.4.5 Optional Endian Mode Override

Some PNX8526 DMA peripherals have bits in a control register that allow override of the global endian mode. Refer to each module for details. This method is used only in modules that deal with DMA of data of a single, fixed size (in a given mode). Such modules implement a field that allows selection of the following modes:

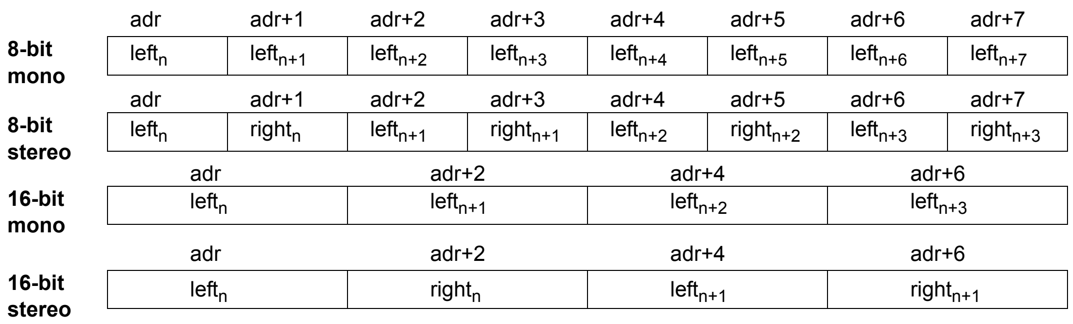
- Normal mode (reset default), obey global PNX8526 endian mode
- Explicit little-endian, unswapped
- Swap over 16 bits
- Swap over 32 bits

4.5 Example: Audio In—Programmer’s View

The PNX8526 Audio In block receives mono or stereo, 8 or 16-bit/sample audio data and fills memory with an 8 or 16-bit data structure. The data structure is put in memory according to both endian mode laws.

A programmer’s view of the Audio In function is sketched in Figure 4. The programmer sees the address of each item (according to the DMA rule), and expects that 16-bit values are correctly stored in memory according to the CPU law. The DMA logic of the Audio In block therefore needs to write data in memory (byte) locations *precisely,* as in Table 5.

Remark: The programmer also sees the Control and Status registers of the audio In block as in Figure 5. These registers are always seen with the same bit-layout in the CPU register, regardless of endian mode.



	adr	adr+1	adr+2	adr+3	adr+4	adr+5	adr+6	adr+7
8-bit mono	left$_{n}$	left$_{n+1}$	left$_{n+2}$	left$_{n+3}$	left$_{n+4}$	left$_{n+5}$	left$_{n+6}$	left$_{n+7}$

	adr	adr+1	adr+2	adr+3	adr+4	adr+5	adr+6	adr+7
8-bit stereo	left$_{n}$	right$_{n}$	left$_{n+1}$	right$_{n+1}$	left$_{n+2}$	right$_{n+2}$	left$_{n+3}$	right$_{n+3}$

	adr	adr+2	adr+4	adr+6
16-bit mono	left$_{n}$	left$_{n+1}$	left$_{n+2}$	left$_{n+3}$

	adr	adr+2	adr+4	adr+6
16-bit stereo	left$_{n}$	right$_{n}$	left$_{n+1}$	right$_{n+1}$

Figure 4: Audio In Memory Data Structure (Programmer’s View)

Table 5: Precise Mapping Audio In Sample Time and Bits to Memory Bytes

Operating Mode	m[adr]	m[adr+1]	m[adr+2]	m[adr+3]
8-bit mono—little-endian or big-endian	left$_{n}$[7:0]	left$_{n+1}$[7:0]	left$_{n+2}$[7:0]	left$_{n+3}$[7:0]
8-bit stereo—little-endian or big-endian	left$_{n}$[7:0]	right$_{n}$[7:0]	left$_{n+1}$[7:0]	right$_{n+1}$[7:0]
16-bit mono—little-endian	left$_{n}$[7:0]	left$_{n}$[15:8]	left$_{n+1}$[7:0]	left$_{n+1}$[15:8]

Table 5: Precise Mapping Audio In Sample Time and Bits to Memory Bytes *…Continued*

Operating Mode	m[adr]	m[adr+1]	m[adr+2]	m[adr+3]
16-bit mono—big-endian	left_n[15:8]	left_n[7:0]	left_{n+1}[15:8]	left_{n+1}[7:0]
16-bit stereo—little-endian	left_n[7:0]	left_n[15:8]	right_n[7:0]	right_n[15:8]
16-bit stereo—big-endian	left_n[15:8]	left_n[7:0]	right_n[15:8]	right_n[7:0]

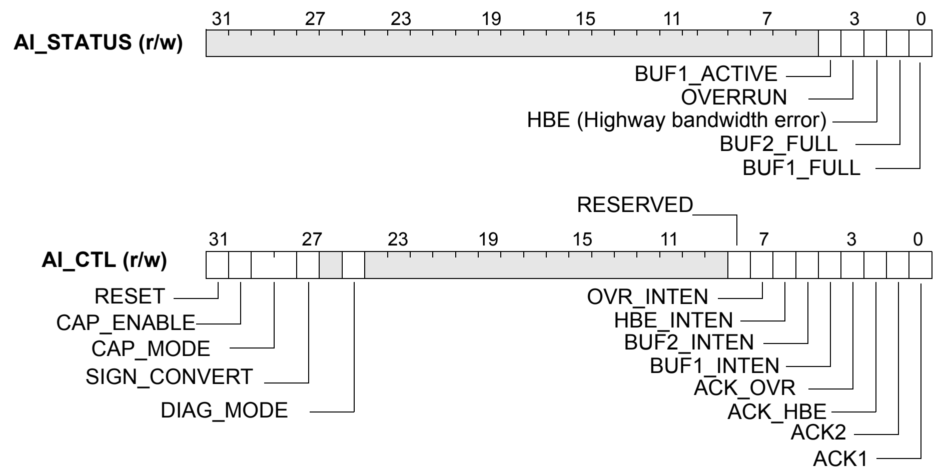


Figure 5: Audio In Control/Status MMIO Registers

4.6 Implementation Details

4.6.1 Endian System Block Diagram

Figure 6 shows a system block diagram with two example DMA peripherals (“IP-blocks”). Each DMA peripheral deals with a 16-bit unit size. IP block 1 transfers data across the PI-Bus, the other across the DVP Memory Bus. For the following, assume that both devices are input devices that DMA data to system memory. For DMA output devices that read data from memory, the operation is similar but in the opposite direction.

The system endian mode of operation is designated to each component by the PI-Bus big-endian signal - asserted for big-endian mode, negated for little-endian mode.

Remark: Both IP blocks are identical. They perform all DMA data transfers across the standard 32-bit “L2 interface.”

This interface uses a simple rule determining where bytes on the 32-bit bus end up, irrespective of endian mode. The 8 lsb wires end up at a memory byte address of the form “4n+0,” the 8 msb wires end up at an address of the form “4n+3,” etc. according to Table 6. Four byte-lane enable signals indicate which lanes carry valid data during this cycle.

The DMA peripheral, or IP-block, must deal with unit endian swapping and unit packing. Swapping is defined as “positioning each byte of a unit correctly with respect to the memory byte address that it is supposed to go to.” Swapping is what implements the CPU rule. Packing is defined as “the action that places consecutive units simultaneously on a wider bus in order to implement the DMA rule.”

As shown in IP-block 1 and 2, swapping ensures that the most significant 8 bits and least significant 8 bits of a unit are placed appropriately on the L2 interface according to the PI-Bus big-endian signal value. Once the unit is appropriately swapped, packing always puts the first unit on the lower bits of the L2 bus in order to ensure that the DMA rule is obeyed.

The DMA peripheral need not be aware of the details of either the PI-Bus, or the DVP Memory Bus. It just swaps and packs, based on its knowledge of unit size and system endian mode, and creates the valid L2 interface data.

Two standard solutions are provided to interface L2 to the DVP buses:

- The “PI-DMA Gizmo” connects the 32-bit L2 interface to the PI-Bus.
- The “packer Lego” and “DMA Gizmo” map the 32-bit L2 interface to the 64-bit DVP Memory Bus.

Since the PI-Bus uses a different endian convention than the L2 interface, the PI-DMA Gizmo must perform endian swapping based on the PI-Bus opcode and system endian mode. The reverse operation happens in the PIMI, which provides the DMA gateway to system memory. It must translate the PI-Bus convention back to the L2 interface convention. Refer to Section 4.6.6 for details.

Remark: The connection from the 32-bit L2 interface to the DVP Memory Bus is identical, both for the IP block as well as for the PIMI. No swapping is needed in this path because the L2 interface uses the same rule for byte lane addresses as the DVP Memory Bus.

The following subsections show how unit data of different lengths travels across the three key interfaces: the 32-bit L2 interface, the PI-Bus and the DVP Memory Bus.

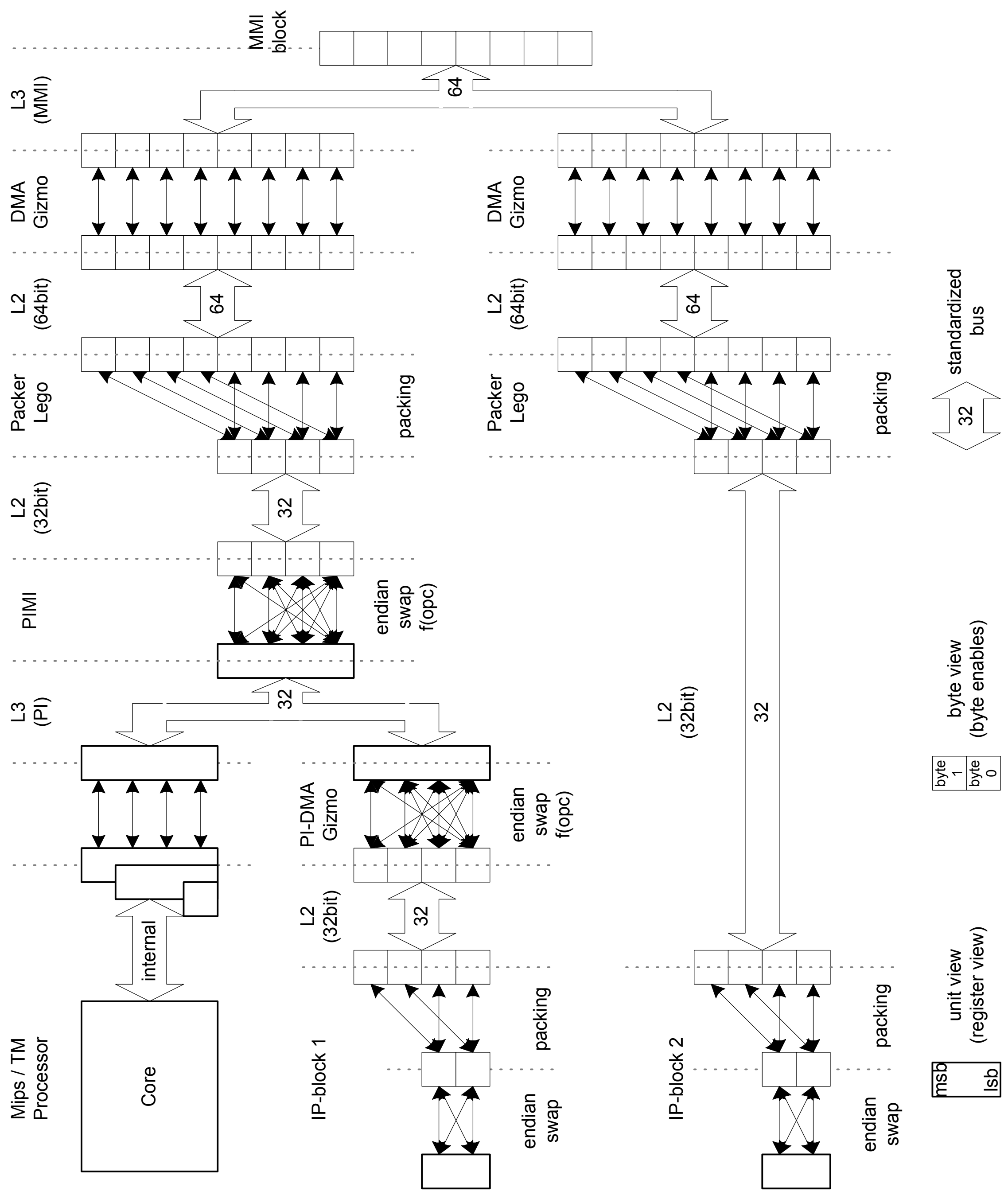


Figure 6: System Block Diagram: Endian-Related Blocks

4.6.2 DMA Across 32-Bit L2 Interface

The address invariance rule of the L2 interface is given in Table 6. A given byte lane implies the address, regardless of endian mode of the system.

Table 6: 32 Bit L2 Interface Byte Address

L2-D[31:24]	L2-D[23:16]	L2-D[15:8]	L2-D[7:0]
4n+3	4n+2	4n+1	4n+0

Devices dealing with 8, 16 or 32-bit units must place bytes on the L2 interface given in Table 7. Any device designed according to these rules can be connected to the PI-bus or DVP Memory Bus using standard building blocks.

Table 7: L2 Interface Rules

Device Item Unit Size	System Endian Mode	L2_D[31:24]	L2_D[23:16]	L2_D[15:8]	L2_D[7:0]
8 bits	either	item #4 with address a+3	item #3 with address a+2	item #2 with address a+1	item #1 with address a
16 bits	big	item #2 with address a+2		item #1 with address a	
		bits 7..0	bits 15..8	bits 7..0	bits 15..8
16 bits	little	item #2 with address a+2		item #1 with address a	
		bits 15..8	bits 7..0	bits 15..8	bits 7..0
32 bits	big	item with address a			
		bits 7..0	bits 15..8	bits 23..16	bits 31..24
32 bits	little	item with address a			
		bits 31..24	bits 23..16	bits 15..8	bits 7..0

4.6.3 DMA Across PI-Bus

The PI-Bus is said to be “endian neutral.” This is true as long as a device only transports one unit at a time across PI, so 8 bits at a time for a UART, 16 bits at a time for a 16-bit Audio device, etc. For performance, however, units need to be packed for transport. This, plus the basic DMA rule and CPU rule create the issue of endian mode on PI, as shown below.

The PI-Bus associates byte addresses with byte lanes as a function of bus opcode and system endian mode (PI-Bus big-endian signal) according to Table 8.

Table 8: PI-Bus Byte Address as a Function of OPCODE and Big-Endian

BIG-ENDIAN	PI OPCODE	PI-D[31:24]	PI-D[23:16]	PI-D[15:8]	PI-D[7:0]
0	WDx	4*A+3	4*A+2	4*A+1	4*A+0
0	HW0	----	----	4*A+1	4*A+0
0	HW1	----	----	4*A+3	4*A+2
0	BY0	----	----	----	4*A+0
0	BY1	----	----	----	4*A+1
0	BY2	----	----	----	4*A+2

Table 8: PI-Bus Byte Address as a Function of OPCODE and Big-Endian *…Continued*

BIG-ENDIAN	PI OPCODE	PI-D[31:24]	PI-D[23:16]	PI-D[15:8]	PI-D[7:0]
0	BY3	----	----	----	4*A+3
0	TB0	----	4*A+2	4*A+1	4*A+0
0	TB1	----	4*A+3	4*A+2	4*A+1
1	WDx	4*A+0	4*A+1	4*A+2	4*A+3
1	HW0	----	----	4*A+0	4*A+1
1	HW1	----	----	4*A+2	4*A+3
1	BY0	----	----	----	4*A+0
1	BY1	----	----	----	4*A+1
1	BY2	----	----	----	4*A+2
1	BY3	----	----	----	4*A+3
1	TB0	----	4*A+0	4*A+1	4*A+2
1	TB1	----	4*A+1	4*A+2	4*A+3
WD = 32-bit word ; HW = 16-bit halfword; BY = byte; TB = three byte					

Devices that directly place data in or retrieve data from memory are DMA devices. Low and medium bandwidth devices perform DMA over the PI-Bus—such devices can have a single PI-Bus connection for both MMIO and DMA.

The 32-bit PI-Bus used in the PNX8526 allows 8-bit at-a-time, 16-bit at-a-time or 32-bit at-a-time writes and reads. If a device uses the same transfer size across PI-Bus as the underlying unit size, the device need not be aware of system endian mode.

For devices with 8 and 16-bit data types, this is inefficient use. The architecture of the buses and arbitration in the PNX8526 recommends transferring data in larger chunks, preferably chunks of 16 bytes or more. This requires coalescing smaller items into 32-bit words.

Devices that don’t use the standard L2 bus and building blocks and group smaller data items into 32-bit PI words must observe the PI-Bus packing rules in Table 9.

Table 9: PI-Bus Unit DMA Rules (32 Bits at-a-time Transfer)

Device Item Unit Size	System Endian Mode	PI_D[31:24]	PI_D[23:16]	PI_D[15:8]	PI_D[7:0]
8 bits	big	item #1 with address a	item #2 with address a+1	item #3 with address a+2	item #4 with address a+3
8 bits	little	item #4 with address a+3	item #3 with address a+2	item #2 with address a+1	item #1 with address a

Table 9: PI-Bus Unit DMA Rules (32 Bits at-a-time Transfer) *…Continued*

Device Item Unit Size	System Endian Mode	PI_D[31:24]	PI_D[23:16]	PI_D[15:8]	PI_D[7:0]
16 bits	big	item #1 with address a bit15...................................bit0		item #2 with address a+2 bit15...................................bit0	
16 bits	little	item #2 with address a+2 bit15...................................bit0		item #1 with address a bit15...................................bit0	
32 bits	either	item (address a) bit31...bit0			

4.6.4 DMA Across DVP Memory Bus

The 64-bit DVP Memory Bus associates byte-addresses with byte lanes in a fixed manner, independent of system endian mode, according to Table 10.

Table 10: DVP Memory Bus Byte Address

Data[63:56]	Data[55:48]	Data[47:40]	Data[39:32]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
8n+7	8n+6	8n+5	8n+4	8n+3	8n+2	8n+1	8n+0

Devices that directly place data in or retrieve data from memory are DMA devices. High bandwidth devices, or devices that require low latency access to memory, perform DMA over the DVP Memory Bus.

DVP Memory Bus DMA devices use large (64-byte or larger) block transfers using the 8-byte lanes of the DVP Memory Bus on PNX8526.

Devices that deal with 8, 16 or 32-bit item data sizes and connect directly to the DVP Memory Bus must follow the rules in Table 11. Note in particular the bit numbers of the 16 and 32-bit data items in big-endian modes. Devices that use the L2 interface can rely on the standard packer and DMA Gizmo to accomplish the DVP Memory Bus rules.

Table 11: DVP Memory Bus Item DMA Rules

Device Item Unit Size	System Endian Mode	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
8 bits	either	item # 8 addr. a+7	item #7 addr. a+6	item #6 addr. a+5	item #5 addr. a+4	item #4 addr. a+3	item #3 addr. a+2	item #2 addr. a+1	item #1 addr. a
16 bits	big	item #4 address a+6 b7...b0 b15...b8		item #3 address a+4 b7...b0 b15...b8		item #2 address a+2 b7...b0 b15...b8		item #1 address a b7...b0 b15...b8	
16 bits	little	item #4 address a+6 b15...b8 b7...b0		item #3 address a+4 b15...b8 b...b0		item #2 address a+2 b15...b8 b7...b0		item #1 address a b15...b8 b7...b0	
32 bits	big	item #2 address a+4 b7...b0 b15...b8 b23...b16 b31...b24				item #1 address a b7..b0 b15...b8 b23...b16 b31...b24			
32 bits	little	item #2 address a+4 b31...b24 b23...b16 b15...b8 b7...b0				item #1 address a b31...b24 b23...b16 b15...b8 b7...b0			

4.6.5 PIO-Only Devices

It is possible that a device uses a PIO-only approach. A UART or simple I^{2}C interface device may not want to implement any DMA, but instead present four characters at a time in an MMIO register. This type of device presents a minor endian-mode issue. There are two possible implementations for the data registers in such a device:

- Present items in a fixed order, e.g. the first item in d[7:0], second in d[15:8], ... fourth in d[31:24].
- Present items in an order that corresponds to system endianness, i.e. for big-endian mode first item in d[31:24], for little-endian mode the first item in d[7:0].

The second method is preferred, but the first method is acceptable for low-speed devices where CPU software can easily do the endian-mode conversion.

4.6.6 PIMI Bridge

The PIMI bridge translates PI-Bus initiated read and write transactions to DVP Memory transactions. This translation is performed in a byte-address invariant way, i.e. the byte address associated with every 8-bit quantity must be equal on each side of the bridge.

Remark: This translation need not be aware of the unit size being transported.

The IP-block that was the originator of units has performed swapping and packing such that each byte has been given the correct byte address.

All the PIMI has to do is use the PI-Bus opcode to determine transaction size and derive from this the intended byte address for each byte of data. This places data appropriately on the DVP Memory Bus side. Another way to think about this is that the PIMI has to undo the translation from the L2 bus to the PI-Bus.

The translation occurs in two steps. When writing device data to memory, the first step converts 32-bit PI-Bus writes to writes across a 32-bit L2 interface. The second step performs packing from the 32-bit L2 bus to the 64-bit DVP Memory Bus. For devices that read memory data, the two steps occur in the reverse direction.

The PI-Bus associates addresses with each byte transferred, depending on the PI-Bus opcode and the setting of the big-endian signal. This byte address is given in Table 8, where the value “A” denotes the integer value on PI-Bus A[n:2] address wires.

The L2 interface always associates addresses in a fixed manner with a given byte lane, as indicated in Table 12.

The net result of this and the byte-address invariance rule for bridges results in the PIMI translation matrix of Table 13.

Table 12: 32-Bit L2 Interface Byte Address

L2-D[31:24]	L2-D[23:16]	L2-D[15:8]	L2-D[7:0]
4n+3	4n+2	4n+1	4n+0

Table 13: PI-Bus to L2 Bridging as a Function of PI OPCODE and Big-Endian

BIG-ENDIAN	PI OPCODE	PI-D[31:24]	PI-D[23:16]	PI-D[15:8]	PI-D[7:0]
0	WDx	L2-D[31:24]	L2-D[23:16]	L2-D[15:8]	L2-D[7:0]
0	HW0	----	----	L2-D[15:8]	L2-D[7:0]
0	HW1	----	----	L2-D[31:24]	L2-D[23:16]
0	BY0	----	----	----	L2-D[7:0]
0	BY1	----	----	----	L2-D[15:8]
0	BY2	----	----	----	L2-D[23:16]
0	BY3	----	----	----	L2-D[31:24]
0	TB0	----	L2-D[23:16]	L2-D[15:8]	L2-D[7:0]
0	TB1	----	L2-D[31:24]	L2-D[23:16]	L2-D[15:8]
1	WDx	L2-D[7:0]	L2-D[15:8]	L2-D[23:16]	L2-D[31:24]
1	HW0	----	----	L2-D[7:0]	L2-D15:8]
1	HW1	----	----	L2-D[23:16]	L2-D[31:24]
1	BY0	----	----	----	L2-D[7:0]
1	BY1	----	----	----	L2-D[15:8]
1	BY2	----	----	----	L2-D[23:16]
1	BY3	----	----	----	L2-D[31:24]
1	TB0	----	L2-D[7:0]	L2-D[15:8]	L2-D[23:16]
1	TB1	----	L2-D[15:8]	L2-D[23:16]	L2-D[31:24]

4.6.7 PCI Interface

The PCI interface on the PNX8526 connects to the off-chip PCI-bus, the PI-Bus and the DVP Memory Bus. As with any bridge, the PCI interface must maintain the byte address of any byte of a transaction on all sides of the bridge.

The PCI interface bridges the following transactions in the PNX8526:

- PCI master read/writes from/to PNX8526 MMIO registers using 32-bit transactions only
- PCI master read/writes from/to PNX8526 SDRAM
- PI-Bus master initiated read/writes from/to PCI targets
- PCI interface internal DMA transactions, where the source can be a PCI target or SDRAM, and the destination is a PCI target or SDRAM.

The 32-bit PCI bus uses byte address conventions identical to the L2 interface and DVP Memory Bus: Refer to Table 14.

Table 14: 32 Bit PCI Interface Byte Address

PCI-AD[31:24]	PCI-AD[23:16]	PCI-AD[15:8]	PCI-AD[7:0]
4n+3	4n+2	4n+1	4n+0

The PI-Bus uses byte address conventions given in Table 8.

The DVP Memory Bus uses the byte address conventions given in Table 10.

With the above byte address conventions on the three sides of the bridge and the byte address invariance rule for bridges, the swap modes can be derived. Since the convention on the PCI bus closely matches those on the DVP Memory Bus, the only system endian mode dependent swapping occurs for transactions involving the PI-Bus.

4.7 Detailed Example

This section describes all steps involved in how a big-endian mode external CPU (e.g. a Power Macintosh), paints an RGB565 pixel format frame buffer in the PNX8526 SDRAM and how this is displayed on the AICP. This example illustrates the following:

- The Power Macintosh PCI bridge and its address invariance rule based swapper
- The GIB-endian PCI pixel transfer
- How data arrives correct in SDRAM in native RGB565 pixel format
- How the AICP takes it and displays it
- How the TM32 CPU core sees the data

The Power Macintosh was the first platform that successfully demonstrated big-endian operations across the PCI bus. Details of how this works can be found in the Apple document “Designing PCI Cards and Drivers for Power MacIntosh Computers.”

Suppose that the big-endian CPU in the Power Macintosh uses a 32-bit store operation to create two RGB565 pixels. Pixel 1, the left-most pixel, has (byte) address “A” and pixel 2 has address “A+2.” Since these two pixels are transferred in a single 32-bit word, “A” is a multiple of 4.

The intermediate stages that the data goes through can be found in Figure 7

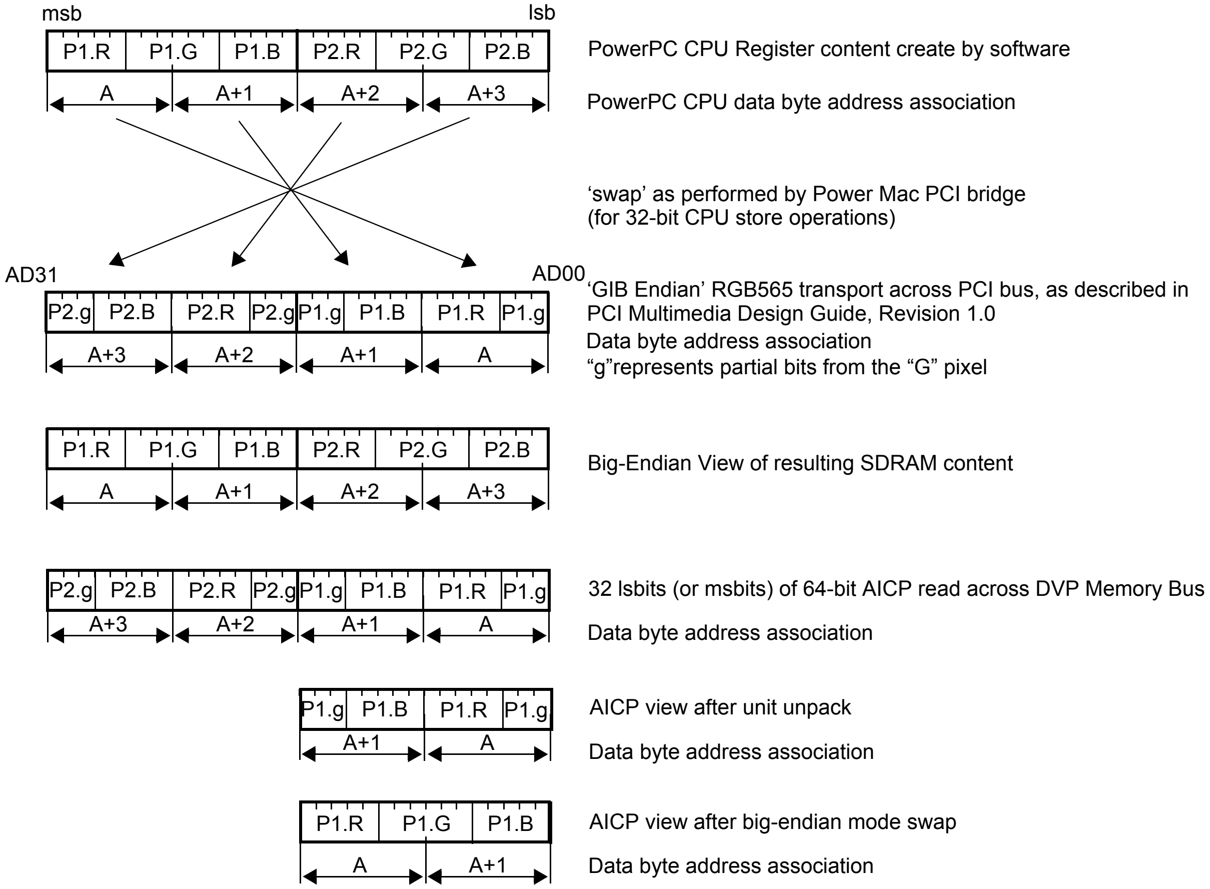

msb
lsb
P1.R P1.G P1.B P2.R P2.G P2.B
A A+1 A+2 A+3
PowerPC CPU Register content create by software
PowerPC CPU data byte address association
‘swap’ as performed by Power Mac PCI bridge
(for 32-bit CPU store operations)
AD31
AD00
P2.g P2.B P2.R P2.g P1.g P1.B P1.R P1.g
A+3 A+2 A+1 A
‘GIB Endian’ RGB565 transport across PCI bus, as described in
PCI Multimedia Design Guide, Revision 1.0
Data byte address association
“g”represents partial bits from the “G” pixel
P1.R P1.G P1.B P2.R P2.G P2.B
A A+1 A+2 A+3
Big-Endian View of resulting SDRAM content
P2.g P2.B P2.R P2.g P1.g P1.B P1.R P1.g
A+3 A+2 A+1 A
32 lsbits (or msbits) of 64-bit AICP read across DVP Memory Bus
Data byte address association
P1.g P1.B P1.R P1.g
A+1 A
AICP view after unit unpack
Data byte address association
P1.R P1.G P1.B
A A+1
AICP view after big-endian mode swap
Data byte address association

Figure 7: Big-Endian External CPU Drawing Two RGB565 Pixels

Remark: The Power Macintosh architecture contains a PCI bridge that maintains byte address invariance. Since all stages inside the PNX8526 maintain byte addresses, the end-to-end result of the complex sequence of actions is a successfully rendered pair of RGB565 pixels.

It is recommended to use only external big-endian CPU/PCI bridge combinations that implement the Power Macintosh style byte-invariant address model with the PNX8526. Some external CPU PCI bridges may only contain a static, transaction-size CPU unaware swapper. The use of such external components is not recommended and will require special care in software.

Chapter 5: Clock Reset and Power Management

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

5.1 Introduction

This chapter includes separate sections on the Clock and Reset modules (see Section 5.2 below and Section 5.3 on page 5-124, respectively). Power Management is also covered at the end of the chapter following the Reset Module registers (see Section 5.4 on page 5-132).

5.2 Clock Module Functional Description

The Clock module generates and controls all modules, buses and CPU clocks in the PNX8526. The features of the PNX8526 clock system are as follows:

- Generates all bus clocks.
- Generates module clocks that meet frequency and jitter requirements.
- Module clocks are under software control.
- Allows bypass of full-custom blocks to support system debug.
- Glitch-free clock control (frequency transitions and switching)
- Clock frequencies are programmable to match CPU speeds with memory subsystems.

5.2.1 Overview

The Clock module has two main interfaces (see Figure 1):

- An interface to the Custom Analog Block (CAB). The CAB module includes all PLLs, several high speed clock dividers, and several Direct Digital Synthesizer (DDS) blocks that provide the main source of clocks for the Clock module.
- An interface to the MIPS PI-Bus. Clock controls are programmed via the PI-Bus.

Clock control logic consists of the following:

- Programmable dividers controlled by configuration registers
- Clock-blocking circuitry to allow for safe, glitch-free switching of clocks. Clocks are typically switched when:
  - PLLs or dividers are reprogrammed.
  - Clocks are switched on/off for powerdown reasons.

  - Following reset and boot up of the chip when all clocks are switched from 27 MHz to their programmed functional frequencies.
- A line of delay cells to allow for clock matching

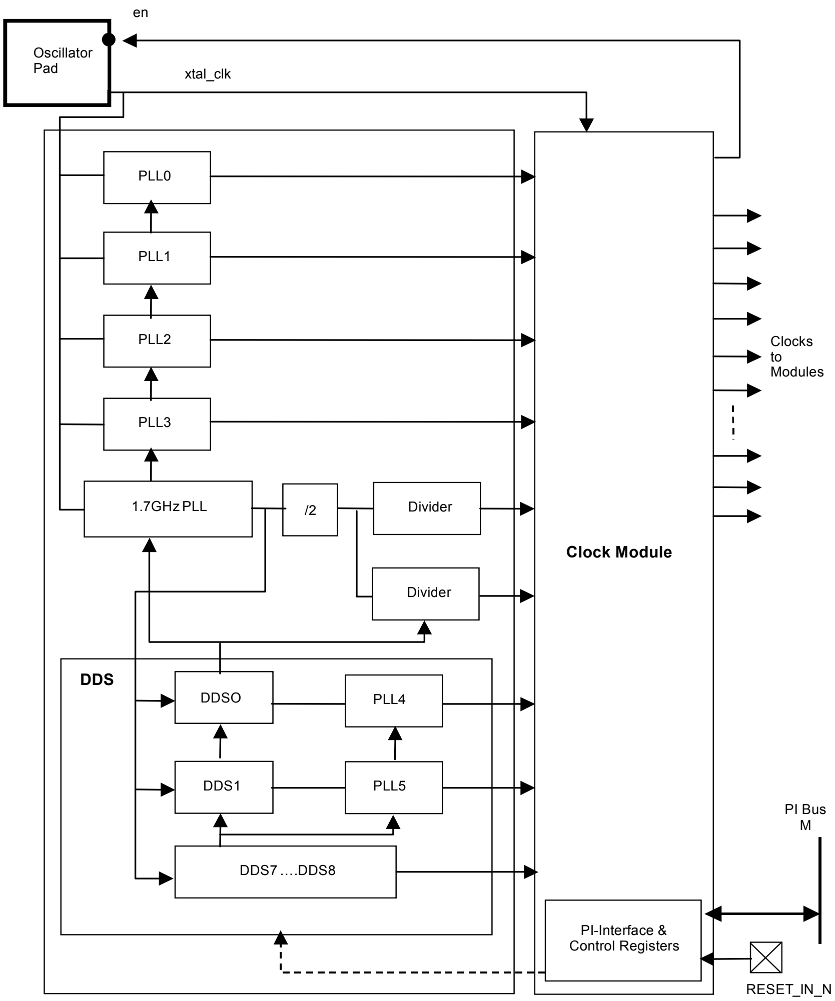


Figure 1: Clock Module Block Diagram

Remark: Not all clocks to the other modules are generated in the Clock module. There may be other clocks that come into the PNX8526 from external sources. Some of these clocks will be fed through the Clock module so that they may undergo the same controls required during reset and powerdown.

5.2.1.1 Custom Analog Block (CAB)

The Custom Analog Block (CAB) is designed to produce clocks from 40 to 200 MHz and a clock of 1.728 GHz.

A 27 MHz crystal provides the source clock for all PLLs in the CAB block. The PLLs are programmable from the Clock module registers to generate a range of possible frequencies. The oscillator pad has an enable controlled by the Clock module.

The DDS blocks are required to make slight adjustments to each video and audio clock to track transmission sources. Software controls this tracking by programming the DDS (8...0) blocks to adjust the clocks. These adjustments are very fine and a 1.7 GHz PLL generates a 1.728 GHz clock with only 0.6 ns jitter to the synthesizers. Some of the video clocks will not tolerate a short term jitter of 0.6 ns, so additional PLLs (PLL4 and PLL5) are required to improve short term jitter.

Table 1: Specification of HC-49U 27.00000 MHZ Crystal

Frequency	**27.00000 MHZ fundamental**
Package	HC-49U
Temperature range	10C to 60C
Capacitive load (CL)	18pF
Frequency accuracy (all included :temperature, aging , frequency at 25C)	+/-40 ppm
* series resonance resistor	40ohm max.
* shunt capacitance	7pF max.
* drive level	1mW max.
* motional capacitance	20fF maximum. (as low as possible).

Table 2: Operating Conditions of the 1.728 GHz PLL

		Limits		
Parameter	**Symbol**	**Min**	**Max**	**Units**
Reference Clock Frequency Range	clk_in	27	27	MHz
PLL Clock Output Frequency	clk_out	1728	1728	MHz
PLL Clock Feedback Frequency	clk_fdb	27	27	MHz

Table 3: AC Characteristics of the 1.728 GHz PLL

		Limits		
Parameter	**Min**	**Typ**	**Max**	**Units**
RMS long term Output Clock Jitter (10µs)		20		ps
Clk_out Duty Cycle	40		60	%
Clk_out Edge Rate Cload = 0.1pF, 10% - 90%			180	ps
Acquisition Time		10		µs
Conditions: Clk_in Duty Cycle is 40 to 60% and rise time max is 2 ns.				

Table 4: Operating Conditions of the DDS

		Limits		
Parameter	**Symbol**	**Min**	**Max**	**Units**
Reference Clock Frequency Range	clk9, clk3	-	1728	MHz
DDS Output Frequency	clk_out		864	MHz
Digital GND Voltage	gnd	0	0	Vdc
VIH (I/P=1)	VIH	1.08		Vdc
VIL (I/P=0)	VIL		0.2	Vdc
Clk_out Loading	CL		0.5	pf

Table 5: AC Characteristics of the DDS

			Limits		
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Units**
Clk_out Duty Cycle		40		60	%
Clock_out Edge Rate Cload = 0.1pF, 10% - 90%				0.3	ns
Clk9, Clk3 Rise/Fall Time	@ 1.728 GHz			180	ps
Clk9, Clk3 Duty Cycle	@ 1.728 GHz	40		60	%

Conditions:

- Clk_in Duty Cycle is 40 to 60% and rise/fall time can be longer (if clk9 and clk3 freq is low, max is 2 ns)

- Due to Clk3 and Clk9 very high frequencies, the setup and hold times are “don’t care” for freq <31:0> and shift_en.

5.2.2 Chip I/O

The following chip I/Os are related directly to the Clock module:

Table 6: Clock Module I/O

Chip Port	Type	Description
XTAL	OSC	27 MHz clock input from oscillator pad
PLL_OUT	OUT	Clock to off-chip PCI devices; note this signal is routed back into the PCI_CLK input pad if the PNX8526 is in PCI master mode.

The following chip I/Os are related to clocks generated or received by the PNX8526 modules independently of the Clock module:

Table 7: Other Clock Module I/O

Chip Port	Type	Description
CLK_1394	IN	External 1394 module clock from chip pin
PCI_CLK	IN	External PCI clock from chip pin (33 MHz, 66 MHz)
MM_CLK_OUT1	I/O	SDRAM clock output
DV2_CLK	IN	Clock to MSP1/2: muxed with clk_tsdma and clk_tsio
DV3_CLK	IN	Clock to MSP1/2: muxed with clk_tsdma and clk_tsio
DV2_DATA	IN	Data to MSP1/2: muxed with clk_tsdma and clk_tsio
DV3_DATA	IN	Data to MSP1/2: muxed with clk_tsdma and clk_tsio
TS_CLK	I/O	Digital video output clock
DV_CLK1	I/O	Clock output from ICP1
DV_CLK2	I/O	Clock output from ICP2
I2S_IN1_OSCLK	I/O	Audio oversample output clock
I2S_IN1_SCK	I/O	Audio input/output bit clock
I2S_IN2_OSCKL	I/O	Audio oversample output clock
I2S_IN2_SCK	I/O	Audio input/output bit clock
I2S_IO_OSCKL	I/O	Audio oversample output clock
I2S_IO_SCK	I/O	Audio input/output bit clock
I2S_OUT1_OSCKL	I/O	Audio oversample output clock
I2S_OUT1_SCK	I/O	Audio input/output bit clock
I2S_OUT2_OSCKL	I/O	Audio oversample output clock
I2S_OUT2_SCK	I/O	Audio input/output bit clock
AIO_OSCLK	I/O	GPIO used as external source for a module clock
AIO_SCK	I/O	GPIO used as external source for a module clock
AIO_SD[1:0]	I/O	GPIO used as external source for a module clock
SSI_SCKL_CTSN	IN	SSI module serial clock external input; GPIO used as external source for a module clock
GPIO[8:3]	I/O	GPIO used as external source for a module clock
UA1_TX	I/O	GPIO used as external source for a module clock
UA1_RX	I/O	GPIO used as external source for a module clock
UA2_TX	I/O	GPIO used as external source for a module clock
UA2_RX	I/O	GPIO used as external source for a module clock
UA2_RTSN	I/O	GPIO used as external source for a module clock
UA2_CTSN	I/O	GPIO used as external source for a module clock
SSI_TX	I/O	GPIO used as external source for a module clock
SSI_RXD	I/O	GPIO used as external source for a module clock

Table 7: Other Clock Module I/O *…Continued*

Chip Port	Type	Description
SSI_FS_RTSN	I/O	GPIO used as external source for a module clock
DBG_TCK	I	JTAG clock used for TMDBG
JTAG_TCK	I	E-JTAG clock used for MIPS E-JTAG

5.2.3 Operation

5.2.3.1 System Level Clocks

The clocks required for the PNX8526 modules are summarized in Table 8.

Table 8: System Level and Internal Clocks

Bus or Module	Signal Name	Description	Required Frequencies (see note)	Clock Requirements	Clock Module Control
Memory Bus	clk_mem	Clock to memory highway and MMI module	Low frequencies, **100 MHz**, 125 MHz, 133 MHz, 143 MHz, 166MHz	TM32 must be in powerdown mode when this clock is reprogrammed.	PLL0 and CLK_MEM_CTL register control
Fast PI-Bus	clk_fpi	Clock to Fast F-PI Bus	Low frequencies, **100 MHz**, 125 MHz, 133 MHz, 143 MHz, 150 MHz	Can also be clk_mem	PLL1and CLK_FPI_CTL register control
MIPS PI-Bus	clk_mpi	Clock to M-PI peripheral bus	Low frequencies..... **50 MHz**, 72 MHz, 75 MHz	Special alignment wrt clk_fpi and half the frequency of clk_fpi	PLL1 and CLK_MPI_CTL register control.
TriMedia PI-Bus	clk_tpi	Clock to TM32 T-PI Bus	Low frequencies.... **50 MHz**, 72 MHz, 75 MHz	Special alignment wrt clk_fpi and half the frequency of clk_fpi	PLL1 and CLK_TPI_CTL register control.
MIPS	clk_mips	Clock to PR3940 MIPS core	Same as clk_fpi	Aligned to clk_fpi	PLL1 and CLK_MIPS_CTL register control
PCI	clk_pci	Clock to PCI block	33 MHz	-	CLK_PCI_CTL register control. Always a chip input: source can be from off chip or if generated internally, clk_pci is sent off chip and routed back into the chip
USB	clk48	Clock to USB block	48 MHz	+/- 0.25% tolerance	CLK48_CTL register control
	clk12	Clock to USB block	12 MHz	-	Divided down version of clk48

Table 8: System Level and Internal Clocks *…Continued*

Bus or Module	Signal Name	Description	Required Frequencies (see note)	Clock Requirements	Clock Module Control
	clk_1x	Internal USB clock	12 MHz or 1.5 MHz	-	n/a
	clk_4x	Internal USB clock	48 MHz or 6 MHz	-	n/a
	clk_pll	Internal USB clock	12 MHz or 1.5 MHz	-	n/a
MPG	clk_vmpg	Video MPEG decoder clock	Low freq, up to 133 MHz	-	PLL3 and CLK_VMPG_CTL register control
IIC1	clk_iic1	IIC1 module clock	12 MHz	-	CLK_IIC1_CTL register control
	scl1_out	Output clock generated in IIC1	12 MHz/n n=30,40,60,80,120,160,240,480	-	n/a
IIC2	clk_iic2	IIC2 module clock	12 MHz	-	CLK_IIC2_CTL register control
	scl2_out	Output clock generated in IIC2	12 MHz/n n=30,40,60,80,120,160, 240,480	-	n/a
UART1	clk_uart1	UART1 module clock	3.6923 MHz	-	CLK_UART1_CTL register control
	baud_clk	Internal UART1 clock	Baud rate = 3.6923 MHz /n n=[Baud rate[15:0]+1]*16	-	n/a
UART2	clk_uart2	UART2 module clock	3.6923 MHz	-	CLK_UART2_CTL register control
	baud_clk	Internal UART2 clock	3.6923 MHz/n n=[Baud rate+1]*16	-	n/a
UART3	clk_uart3	UART3 module clock	3.6923 MHz	-	CLK_UART3_CTL register control
	baud_clk	Internal UART3 clock	3.6923 MHz/n n=[Baud rate+1]*16	-	n/a
D2D	clk_d2d	2D Drawing Engine clock	Low freq...120 MHz	-	CLK_D2D_CTL register control
1394	clk_1394_i	1394 module clock	49.152 MHz	-	external chip i/p
SMART1,2	clk_smart	Smartcard module clock	54 MHz	-	CLK_SMART_CTL register control
MBS	clk_mbs	MBS module clock	108 MHz .. 120 MHz	-	PLL2 and CLK_MBS_CTL register control
VIP1	clk_vip1	VIP1 module clock	Low freq .. 80 MHz	-	CLK_VIP1_CTL register control

Table 8: System Level and Internal Clocks *…Continued*

Bus or Module	Signal Name	Description	Required Frequencies (see note)	Clock Requirements	Clock Module Control
VIP2	clk_vip2	VIP2 module clock	Low freq .. 80 MHz	-	CLK_VIP2_CTL register control
MSP1	clk_vmsp1 clk_nds1 msp1_in_clk clk_1394tx1 mspout1_clk	MSP1 module clock	Low freq .. 108 MHz	-	CLK_MSP1_CTL CLK_MSP1_SRC_CTL CLK_VMSP1_CTL CLK_1394TX1_CTL CLK_1394TX1_SRC_CTL CLK_NDS1_CTL MSPOUT1_CLK_CTL MSPOUT1_CLK_SRC_CTL register controls
MSP2	clk_vmsp2 clk_nds2 msp2_in_clk clk_1394tx2 mspout2_clk	MSP2 module clock	Low freq ..108 MHz	-	CLK_MSP2_CTL CLK_MSP2_SRC_CTL CLK_VMSP2_CTL CLK_1394TX2_CTL CLK_1394TX2_SRC_CTL CLK_NDS2_CTL MSPOUT2_CLK MSPOUT2_CLK_SRC_CTL register controls
MSP3	clk_vmsp3 msp3_in_clk mspout3_clk	MSP3 module clock	Low freq ..108 MHz	-	CLK_MSP3_CTL CLK_MSP3_SRC_CTL CLK_VMSP3_CTL MSPOUT3_CLK MSPOUT3_CLK_SRC_CTL register controls
TMDBG	DBG_TCK	TMDBG JTAG clock	Up to 33MHz	-	Input clock
EJTAG	JTAG_TCK	EJTAG clock	Up to 33MHz	-	Input clock
MSP 1, 2	clk_tsdma	MSP1/2 modules	Low freq..108MHz	-	CLK -TSDMA_CTL CLK -TSDMA -SRC_CTL
MSP 1, 2	clk_1394rx	MSP1/2 modules	Low freq .. 108 MHz	-	CLK_1394RX_CTL CLK_1394RX_SRC_CTL
TSOUT	tsout_clk_out tsout_serial_clk tsout_parallel_clk	MSP1/2 modules	Low freq .. 108MHz	-	TSOUT_CLK_OUT_CTL TSOUT_CLK_OUT_SRC_CTL TSOUT_SERIAL_CTL TSOUT_PARALLEL_CTL TSOUT_PARALLEL_SRC_CTL register control
ICP1	clk_icp1_mux	ICP1 pixel clock from DDS	Up to 81 MHz Typical: 27 MHz, 54 MHz, 81 MHz	-	DDS0_ICP1_CTL

Table 8: System Level and Internal Clocks *…Continued*

Bus or Module	Signal Name	Description	Required Frequencies (see note)	Clock Requirements	Clock Module Control
	clk_icp1	ICP1 pixel clock	Up to 50 MHz	-	CLK_ICP1_CTL: Can be programmed to 1/2, 1/3, 1/4 or 1/6 of the clk_icp1_mux
ICP2	clk_icp2_mux	ICP2 pixel clock from DDS	Up to 81 MHz Typical: **27 MHz**, 54 MHz, 81 MHz	-	DDS1_ICP2_CTL
	clk_icp2	ICP2 pixel clock	Low freq .. 50 MHz	-	CLK_ICP2_CTL: Can be programmed to 1/2, 1/3, 1/4 or 1/6 of the clk_icp2_mux
AIN-1	ai1_osclk	Audio oversample clock form DDS block	Low freq ... 40 MHz	-	DDS2_AI1_CTL
AOUT-1	ao1_osclk	Audio oversample clock form DDS block	Low freq .... 40 MHz	-	DDS3_AO1_CTL
AIN-2	ai2_osclk	Audio oversample clock form DDS block	Low freq .... 40 MHz	-	DDS4_AI2_CTL
AOUT-2	ao2_osclk	Audio oversample clock form DDS block	Low freq .... 40 MHz	-	DDS5_AO2_CTL
AOUT-3	aio3_osclk	Audio oversample clock form DDS block	Low freq .... 40 MHz	-	DDS6_AIO3_CTL
SPDIO	clk_spdo	SPDIO module clock	Low freq .. **6.14 MHz** .. 12.29 MHz	-	DDS7_SPDO_CTL
	clk_spdi	SPDIO module clock	72 MHz, **144 MHz**	-	CLK_SPDI_CTL
Debug and GPIO	clk_tstamp	Timestamp clock	108 MHz	-	CLK_TSTAMP_CTL register control.
Debug and GPIO	clk_spy	Spy clock	13.5 MHz	-	CLK_SPY_CTL

[8-1] Nominal value is shown in bold.

5.2.3.2 Sources of PNX8526 Clocks

All clocks in the PNX8526 clock system are generated from four possible sources:

- Four identical “standard” PLLs
- High-frequency dividers from the 1.7 GHz PLL in the CAB
- Direct Digital Synthesizer (DDS) blocks in the CAB

- External clock inputs, or derived from input data streams.

The clock assignments for each of these four sources are summarized in Table 9, Table 10, Table 11, and Table 12, respectively.

Table 9: Standard PLL Clock Assignments

Source	Clocks	Expected N, M, P Parameters (see note)
PLL0	- clk_mem	143 MHz: (N,M,P) = (51,3,1) ...... 143.1 MHz **100 MHz**: (N,M,P) = (35,3,1) ...... 99.9 MHz ... low frequencies
PLL1	- clk_fpi	143 MHz: (N,M,P) = (51,3,1) ...... 143.1 MHz **100 MHz**: (N,M,P) = (35,3,1) ...... 99.9 MHz ... low frequencies
PLL2	- clk_mbs	120 MHz: (N,M,P) = (80,7,1) ...... 143.1 MHz **100 MHz**: (N,M,P) = (35,3,1) ...... 99.9 MHz ... low frequencies
PLL3	- clk_vmpg	133 MHz: (N,M,P) = (87,7,1) ...... 133.5 MHz **100 MHz**: (N,M,P) = (35,3,1) ...... 99.9 MHz ... low frequencies
Note: Parameters are in decimal.		

Table 10: 1.7 GHz PLL Divider Clocks

Clocks	Divider Values	Divider in CAB	Divider in Clock
- clk_pci	1.728 GHz/52 = 33.23 MHz	y	n
- clk48	1.728 GHz/36 = 48 MHz	y	n
- clk12 - clk_iic1, clk_iic2	clk48/4 = 12 MHz	n	y
- clk_uart1, clk_uart2, clk_uart3	clk48/13 = 3.69 MHz	n	y
- clk_d2d	1.728 GHz/n: n=16 gives 108 MHz n=20 gives 86.4 MHz n is programmable in the Clock module Note: clk_mbs (from PLL2) will also be selectable for clk_d2d)	y	n
- clk_vmsp1	1.728 GHz/16 = 108 MHz	y	n
- clk_vmsp2	1.728 GHz/16 = 108 MHz	y	n
- clk_vmsp3	1.728 GHz/16 = 108 MHz	y	n
- clk_nds1	108 MHz/2 = 54 MHz	n	y
- clk_nds2	108 MHz/2 = 54 MHz	n	y

Table 10: 1.7 GHz PLL Divider Clocks *...Continued*

Clocks	Divider Values	Divider in CAB	Divider in Clock
- clk_spdi	1.728 GHz/12 = 144 MHz	y	n
	144 MHz/2 = 72 MHz	n	y
- clk_tstamp	1.728 GHz/16 = 108 MHz	y	n
- clk_spy	108 MHz/8 = 13.5 MHz (Note: This division is done in the Clock Module)	n	y

Table 11: DDS Clock

Source	Clocks	Notes
DDS0	- clk_icp1_mux	Typical: **27 MHz**, 54 MHz, 81 MHz
DDS1	- clk_icp2_mux	Typical: **27 MHz**, 54 MHz, 81 MHz
DDS2	- ai1_osclk	low freq .. 30 MHz
DDS3	- ao1_osclk	low freq .. 30 MHz
DDS4	- ai2_osclk	low freq .. 30 MHz
DDS5	- ao2_osclk	low freq .. 30 MHz
DDS6	- aio3_osclk	low freq .. 30 MHz
DDS7	- clk_spdo	low freq .... 12.29 MHz (default **6.14 MHz**)
DDS8	- clk_tsdma	low freq .... **24 MHz**
Note: Bold denotes the default frequency.		

DDS frequencies are determined by the following equation:

$$\mathrm{freq} = (1.728\ \mathrm{GHz} * \mathrm{N})/2^{32}$$

N= value programmed into DDS frequency control registers - DDSn_CTL[31:0].

Table 12: External Clocks

Clock	Frequency	IN/OUT	I/O Name	Description
xtal_clk	27 MHz	OSC	XTAL	27 MHz clock input from oscillator pad
clk_pci	33 MHz	OUT	PLL_OUT	Clock to off-chip PCI devices. Note this signal is routed back into the PCI_CLK input pad if the PNX8526 is in PCI master mode.
clk_1394_i		IN	CLK_1394	External 1394 module clock from chip pin
clk_pci_i	33 MHz	IN	PCI_CLK	External PCI clock from chip pin (33 MHz)
mm_clk_out	143 MHz, 133MHz ... low freq.	OUT	MM_CLK_OUT1	SDRAM clock output
DV1_CLK		IN	DV1_CLK	Source clock for VIP1
DV2_CLK		IN	DV2_CLK	Source clock for MSP1/2
DV3_CLK		IN	DV3_CLK	Source clock for MSP1/2
DV2_DATA		IN	DV2_DATA	Source clock for MSP1/2
DV3_DATA		IN	DV3_DATA	Source clock for MSP1/2

Table 12: External Clocks *…Continued*

Clock	Frequency	IN/OUT	I/O Name	Description
tsout_clk_out		OUT	ts_clk	Digital video o/p clock
dv_clk1		OUT	dv_clk1	Clock output from ICP1
dv_clk2		OUT	dv_clk2	Clock output from ICP2
i2s_in1_osclk		OUT	i2s_in1_osclk	Audio oversample output clock
i2s_in1_sck		OUT	i2s_in1_sck	Audio input/output bit clock
i2s_in2_osclk		OUT	i2s_in2_osclk	Audio oversample output clock
i2s_in2_sck		OUT	i2s_in2_sck	Audio input/output bit clock
i2s_io_osclk		OUT	i2s_io_osclk	Audio oversample output clock
i2s_io_sck		OUT	i2s_io_sck	Audio input/output bit clock
i2s_out1_osclk		OUT	i2s_out1_osclk	Audio oversample output clock
i2s_out1_sck		OUT	i2s_out1_sck	Audio input/output bit clock
i2s_out2_osclk		OUT	i2s_out2_osclk	Audio oversample output clock
i2s_out2_sck		OUT	i2s_out2_sck	Audio input/output bit clock
ssi_sclk_ctsn		IN	ssi_sclk_ctsN	SSI module serial clock external input
scl1_out	12 MHz/n n=30,40,60,80,120,160,240,480	OUT	I2C1_SCL	Clock output generated internally in IIC-1
scl2_out	12 MHz/n, n=30,40,60,80,120,160,240,480	OUT	I2C2_SCL	Clock output generated internally in IIC-2
sc1_scck	40.5 MHz/n, n=4,6,10,16	OUT	SC1_SCCK	Clock output generated internally in SmartCard-1
sc2_scck	40.5 MHz/n, n=4,6,10,16	OUT	SC2_SCCK	Clock output generated internally in SmartCard-2

5.2.3.3 PLL Programming

A PLL consists of a Voltage Controlled Oscillator (VCO) and a Post Divide (PD) circuit, as shown in Figure 2. Each PLL[0-5] has a control register for programming its parameters

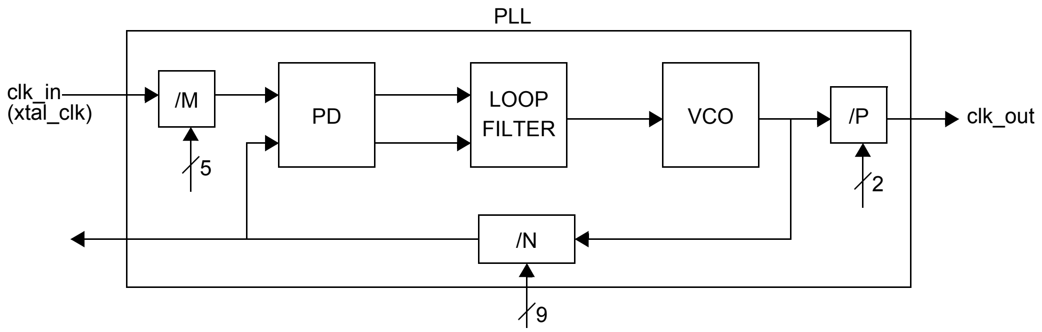


Figure 2: PLL Parameters

Remark: Custom PLLs are used for PLL0...5 in Figure 1. The frequency for this PLL is given by:

Frequency = 27 MHz * N/M *P

where N is 9-bit, M is 5-bit, and P is 2-bit. The N, M and P bits are programmable in the control register.

The frequency from the VCO can be determined as follows:

VCO = 27 MHz * N/M

The output of the VCO can be post-divided by 1, 2, 4 according to the following equation:

Frequency = VCO / P

The bit width of N, M is 9 and 5-bit, respectively. P is a two-bit divider that can divide by 1(P=00), 2(P=01), 4(P=10) or 8(P=11).

Constraints

The PLLs have certain characteristics that must be met according to the following equations:

$150\text{MHz} \geq \text{Fpd} \geq 1.6875\text{MHz}$ ……………………………………………….(1)

$270\text{MHz} \geq \text{Fvco} \geq 27\text{MHz}$………………………………………………...(2)

Fpd = (Fvco/N) i.e., (Fin/M) = (Fvco/N). Therefore, N/M = (Fvco/Fin) and for Fin = 27 MHz, we end up withN/M = (Fvco/Fin).………………........................(3)

Inequality (1) is needed to guarantee the loop's stability, which implies that $M \leq 16$.

To produce a frequency, e.g. of 42 MHz, Fvco must be 4x42 MHz = 168 MHz in order to satisfy the inequality (2).

Since Fin = 27 MHz, then equation (3) gives N/M = (168/27) i.e., N/M = (56/9). So choose N=59 and M=9.

Table 13 provides values for the PLL parameters to achieve certain output frequencies.

Table 13: Programmable Output Frequencies

M	Fpd	N	Fvco MHz	f(P)	Fout MHz
10	2.7	61	164.7	1	164.7
10	2.7	53	143.1	1	143.1
9	3	88	264	2	132
9	3	89	267	2	133.5
9	3	83	249	2	124.5
9	3	84	252	2	126
9	3	80	240	2	120
5	5.4	37	199.8	2	99.9
7	3.857	43	165.9	2	82.93
9		59	168	4	42

5.2.3.4 Bypass Clock Sources

In the event of a problem with any of the clock sources from the CAB, it is possible to switch these clocks to off-chip sources. The external clock sources will be routed through the GPIOs. Table 14 shows the GPIOs to be used in the Bypass mode.

Table 14: Bypass Clock Sources

Clocks from Clock Module	Bypass Control Register	GPIO Assignment	Frequency
clk_mem	CLK_MEM_CTL.sel_ext_clk.	GPIO[3]	low to 166 MHz
clk_fpi	CLK_FPI_CTL.sel_ext_clk.	GPIO[4]	low to 143 MHz
clk_mips			low to 150 MHz
clk_mbs	CLK_MBS_CTL.sel_ext_clk	GPIO[5]	low to 120 MHz
clk_vmpg	CLK_VMPG_CTL.sel_ext_clk	GPIO[6]	low to 133 MHz
clk_pci	CLK_PCI_CTL.sel_ext_clk	GPIO[7]	33 MHz
clk48	CLK48_CTL.sel_ext_clk	GPIO[8]	48 MHz
clk12	CLK12_CTL.sel_ext_clk	SSI_TX	12 MHz
clk_iic1	CLK_IIC1_CTL.sel_ext_clk		12 MHz
clk_iic2	CLK_IIC2_CTL.sel_ext_clk		12 MHz
clk_uart1	CLK_UART1_CTL.sel_ext_clk	SSI_RXD	3.69 MHz
clk_uart2	CLK_UART2_CTL.sel_ext_clk		3.69 MHz
clk_uart3	CLK_UART3_CTL.sel_ext_clk		3.69 MHz
clk_smart1	CLK_SMART1_CTL.sel_ext_clk		54 MHz
clk_smart2	CLK_SMART2_CTL.sel_ext_clk		54 MHz
clk_d2d	CLK_D2D_CTL.sel_ext_clk	UA1_TX	80 MHz .. 108 MHz
clk_vmsp1	CLK_VMSP1_CTL.sel_ext_clk	UA1_RX	108 MHz
clk_vmsp2	CLK_VMSP2_CTL.sel_ext_clk		108 MHz
clk_vmsp3	CLK_VMSP3_CTL.sel_ext_clk		108 MHz
clk_spdi	CLK_SPDI_CTL.sel_ext_clk	UA2_TX	72 MHz, 144 MHz
clk_tstamp	CLK_TSTAMP_CTL.sel_ext_clk	UA1_RX	108 MHz
clk_spy	CLK_SPY_CTL.sel_ext_clk	SSI_FS_RTSN	13.5 MHz
clk_icp1_mux	DDS0_ICP1_CTL.sel_ext_clk	UA2_RX	Up to 81 MHz
clk_icp2_mux	DDS1_ICP2_CTL.sel_ext_clk		Up to 81 MHz
clk_icp1	CLK_ICP1_CTL.sel_ext_clk	UA2_RTSN	Up to 40.5 MHz
clk_icp2	CLK_ICP2_CTL.sel_ext_clk		Up to 40.5 MHz
ai1_osclk	DDS2_AI1_CTL.sel_ext_clk	UA2_CTSN	low freq .... 33 MHz
ao1_osclk	DDS3_AO1_CTL.sel_ext_clk		low freq .... 33 MHz
ai2_osclk	DDS4_AI2_CTL.sel_ext_clk		low freq .... 33 MHz
ao2_osclk	DDS5_AO2_CTL.sel_ext_clk		low freq .... 33 MHz
aio3_osclk	DDS6_AIO3_CTL.sel_ext_clk		low freq .... 33 MHz
clk_spdo	DDS7_SPDO_CTL.sel_ext_clk	UA1_RX	108 MHz

Table 14: Bypass Clock Sources *…Continued*

Clocks from Clock Module	Bypass Control Register	GPIO Assignment	Frequency
msp1_in_clk	MSP1_IN_CTL.sel_ext_clk	SSI_SCLK_CTSN	0 to 108 MHz
msp2_in_clk	MSP2_IN_CTL.sel_ext_clk		0 to 108 MHz
clk_1394tx1	1394TX1_CLK_CTL.sel_ext_clk		0 to 108 MHz
clk_1394tx2	1394TX2_CLK_CTL.sel_ext_clk		0 to 108 MHz
clk_tsdma	CLK_TSDMA_CTL.sel_ext_clk		0 to 108 MHz
clk_nds1	CLK_NDS1_CTL.sel_ext_clk	AIO_OSCLK	54 MHz
clk_nds2	CLK_NDS2_CTL.sel_ext_clk		54 MHz
tsout_clk_out	TSOUT_CLK_OUT_CTL.sel_ext_clk	AIO_SCK	0 to 108 MHz
tsout_serial_clk	TSOUT_SERIAL_CLK_CTL.sel_ext_clk		0 to 108 MHz
tsout_parallel_clk	TSOUT_PARLLEL_CLK_CTL.sel_ext_clk	AIO_WS	0 to 13.5 MHz

5.2.3.5 Power Up and Reset Sequence

On power up, the Clock module outputs all clocks to modules, defaulting to the 27 MHz xtal_clk clock. Reset of all modules and the boot-up sequence executed by the Boot module runs on the 27 MHz clock.

At some point in the boot-up sequence, the Boot module programs all clk_mem and PI-Bus clock PLLs to their required frequencies for functional operation. After waiting for the 300 μs PLL settling time, the Boot block will set the “exit_reset” registers for these clocks and the clock module will switch these bus clocks from the 27 MHz xtal_clk to their individual programmed functional frequencies.

5.2.3.6 Powerdown

All clocks generated in the Clock module may be disabled by programming the relevant clock enable bit of each clock control register. It is possible to gate module clocks in individual modules rather than in the Clock module.

5.2.3.7 Clock Detection

Clock detection is required in the case of an external clock being removed or connected—e.g., if the video cable to the Set Top Box is suddenly removed (and an external video clock thereby stopped), this event will be detected by the Clock module. Also the Clock module detects when the cable is reconnected and a clock is present again.

These events are flagged by an interrupt that is routed to both the M-PIC and the T-PIC. The clock detection is done on the following clock inputs to the PNX8526:

- DV1_CLK
- DV2_CLK
- DV3_CLK
- CLK_1394
- I2S_IN1_SCK

- I2S_IN2_SCK
- I2S_IO_SCK
- DV1_DATA (via TSIO, called ts_s12_clk in the Clocks module)
- DV2_DATA (via TSIO, called ts_s22_clk in the Clocks module)

Clock detection is based on an 8-bit Gray counter running at the external clock frequency. This allows detection of clocks between 1 MHz and 200 MHz. It takes up to 2 μs from when a clock is removed until the interrupt condition is generated.

Interrupt Generation

Interrupts will be generated whenever “clock present” changes status. Therefore an interrupt will be generated if a clock changes from “present” to “non-present” or from “non-present” to “present.” The interrupt registers are implemented using the standard peripheral interrupt module and can thus be enabled/cleared/set by software.

5.2.4 Register Descriptions

The base address for the Clock module in the PNX8526 is 0x04 7000.

5.2.4.1 Register Address Map

Table 15: Clock Module Register Summary

Offset	Name	Description
0x04 7000	PLL0_CTL	PLL0 Control Register
0x04 7004	PLL1_CTL	PLL1 Control Register
0x04 7008	PLL2_CTL	PLL2 Control Register
0x04 700C	PLL3_CTL	PLL3 Control Register
0x04 7010	PLL4_CTL	PLL4 Control Register
0x04 7014	PLL5_CTL	PLL5 Control Register
0x04 7018	PLL1_7_CTL	PLL1_7 GHz Control Register
0x04 701C-709C	Reserved	-
0x04 7100	DDS0_ICP1_CTL	DDS0: clk_icp1_mux frequency control
0x04 7104	DDS1_ICP2_CTL	DDS1: clk_icp2_mux frequency control
0x04 7108	DDS2_AI1_CTL	DDS2: ai1_osclk frequency control
0x04 710C	DDS3_AO1_CTL	DDS3: ao1_osclk frequency control
0x04 7110	DDS4_AI2_CTL	DDS4: ai2_osclk frequency control
0x04 7114	DDS5_AO2_CTL	DDS5: ao2_osclk frequency control
0x04 7118	DDS6_AIO3_CTL	DDS6: aio3_osclk frequency control
0x04 711C	DDS7_SPDO_CTL	DDS7: clk_spdo frequency control
0x04 7120	DDS8_TSDMA_CTL	DDS8: clk_tsdma frequency control
0x04 7124-719C	Reserved	-
0x04 7200	CLK_MEM_CTL	clk_mem control

Table 15: Clock Module Register Summary *…Continued*

Offset	Name	Description
0x04 7204	CLK_FPI_CTL	clk_fpi control
0x04 7208	Reserved	-
0x04 720C	CLK_TPI_CTL	clk_tpi control
0x04 7210	CLK_MPI_CTL	clk_mpi control
0x04 7214	Reserved	-
0x04 7218	CLK_VMPG_CTL	clk_vmpg control
0x04 721C	CLK_D2D_CTL	clk_d2d control
0x04 7220	CLK_VIP1_CTL	clk_vip1 control
0x04 7224	CLK_VIP2_CTL	clk_vip2 control
0x04 7228	CLK_SMART_CTL	clk_smart control
0x04 722C	CLK48_CTL	clk48 control
0x04 7230	CLK12_CTL	clk12 control
0x04 7234	CLK_IIC1_CTL	clk_iic1 control
0x04 7238	CLK_IIC2_CTL	clk_iic2 control
0x04 723C	CLK_UART1_CTL	clk_uart1 control
0x04 7240	CLK_UART2_CTL	clk_uart2 control
0x04 7244	CLK_UART3_CTL	clk_uart3 control
0x04 7248	CLK_MSP1_SRC_CTL	clk_msp1 source control
0x04 724C	CLK_MSP1_CTL	clk_msp1 control
0x04 7250	CLK_PCI_CTL	clk_pci control
0x04 7254	CLK_MBS_CTL	clk_mbs control
0x04 7258	CLK_SPDI_CTL	clk_spdi control
0x04 725C	CLK_TSTAMP_CTL	clk_tstamp control
0x04 7260	CLK_SPY_CTL	clk_spy control
0x04 7264	CLK50_CTL	clk50 control
0x04 7268	CLK_VMSP1_CTL	clk_vmsp1 control
0x04 726C	CLK_VMSP2_CTL	clk_vmsp2 control
0x04 7270	CLK_NDS1_CTL	clk_nds1 control
0x04 7274	CLK_NDS2_CTL	clk_nds2 control
0x04 7278	CLK_1394TX1_SRC_CTL	clk_1394tx1 source control
0x04 727C	CLK_1394TX1_CTL	clk_1394tx1 control
0x04 7280	CLK_1394TX2_SRC_CTL	clk_1394tx2 source control
0x04 7284	CLK_1394TX2_CTL	clk_1394tx2 control
0x04 7288	CLK_TSDMA_SRC_CTL	clk_tsdma source control
0x04 728C	CLK_TSDMA_CTL	clk_tsdma control
0x04 7290	TSOUT_CLK_OUT_SRC_CTL	tsout_clk_out source control
0x04 7294	TSOUT_CLK_OUT_CTL	tsout_clk_out control
0x04 7298	TSOUT_SERIAL_CLK_CTL	tsout_serial_clk control

Table 15: Clock Module Register Summary *…Continued*

Offset	Name	Description
0x04 729C	Reserved	-
0x04 7300	CLK_ICP1_MUX_CTL	clk_icp1_mux control
0x04 7304	CLK_ICP1_CTL	clk_icp1 control
0x04 7308	CLK_ICP2_MUX_CTL	clk_icp2_mux control
0x04 730C	CLK_ICP2_CTL	clk_icp2 control
0x04 7310	AI1_OSCLK_CTL	ai1_osclk control
0x04 7314	AO1_OSCLK_CTL	ao1_osclk control
0x04 7318	AI2_OSCLK_CTL	ai2_osclk control
0x04 731C	AO2_OSCLK_CTL	ao2_osclk control
0x04 7320	AIO3_OSCLK_CTL	aio3_osclk control
0x04 7324	CLK_SPDO_CTL	clk_spdo control
0x04 7328-739C	Reserved	-
0x04 7400	GPIO_CLK_Q0_CTL	gpio_clkout_q0 control
0x04 7404	GPIO_CLK_Q1_CTL	gpio_clkout_q1 control
0x04 7408	GPIO_CLK_Q2_CTL	gpio_clkout_q2 control
0x04 740C	GPIO_CLK_Q3_CTL	gpio_clkout_q3 control
0x04 7410-741C	Reserved	-
0x04 7420	CLK_VIP1_SEL_CTL	clk_vip1 source select
0x04 7424	CLK_VIP2_SEL_CTL	clk_vip2 source select
0x04 7428	CLK_SMART2_CTL	clk_smart2 select
0x04 742C	CLK_AI1_SCK_O_CTL	clk_ai1_sck_o select
0x04 7430	CLK_AO1_SCK_O_CTL	clk_ao1_sck_o select
0x04 7434	CLK_AI2_SCK_O_CTL	clk_ai2_sck_o select
0x04 7438	CLK_AO2_SCK_O_CTL	clk_ao2_sck_o select
0x04 743C	CLK_AIO3_SCK_O_CTL	clk_aio3_sck_o select
0x04 7440-745C	Reserved	-
0x04 7460	CLK_OPTION_DIV_CTL	optional clock for msp control
0x04 7464	MSPOUT1_CLK_SRC_CTL	mspout1_clk source control
0x04 7468	MSPOUT1_CLK_CTL	mspout1_clk control
0x04 746C	MSPOUT2_CLK_SRC_CTL	mspout2_clk source control
0x04 7470	MSPOUT2_CLK_CTL	mspout2_clk control
0x04 7474	CLK_1394RX_SRC_CTL	clk_1394rx source control
0x04 7478	CLK_1394RX_CTL	clk_1394rx control
0x04 747C	MSP2_IN_CLK_SRC_CTL	msp2_in_clk source control
0x04 7480	MSP2_IN_CLK_CTL	msp2_in_clk control
0x04 7484	TSOUT_PARALLEL_CLK_SRC_CTL	tsout_parallel_clk source control
0x04 7488	TSOUT_PARALLEL_CLK_CTL	tsout_parallel_clk control
0x04 748C	CLK_VMSP3_CTL	msp3 control

Table 15: Clock Module Register Summary *…Continued*

Offset	Name	Description
0x04 7490	MSP3_IN_CLK_SRC_CTL	msp3_in_clk source control
0x04 7494	MSP3_IN_CLK_CTL	msp3_in_clk control
0x04 7498	MSPOUT3_CLK_SRC_CTL	mspout3_clk source control
0x04 749C	MSPOUT3_CLK_CTL	mspout3_clk control
0x04 74A0-7FDC	Reserved	-
0x04 7FE0	INTERRUPT STATUS	Status of Clock Detection interrupts
0x04 7FE4	INTERRUPT ENABLE	Enable Clock Detection interrupts
0x04 7FE8	INTERRUPT CLEAR	Clear Clock Detection interrupts
0x04 7FEC	INTERRUPT SET	Set Clock Detection interrupts
0x04 7FF0	Reserved	-
0x04 7FF4	POWERDOWN	Powerdown mode
0x04 7FF8	Reserved	-
0x04 7FFC	Module ID	Module Identification and revision information

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Clock Control				
Offset 0x04 7000			*PLL0_CTL*	
Reset values are set for expected frequencies for faster boot-up, shorter boot code. Note: When this register is written, PI_ACK must be held in “WAT” until blocking of the clock has started.				
31	R	n/a	pll0_blocked	1 = blocking of clock from the PLL is in progress
30:26	R/W	n/a	Reserved	Read as 0, write nothing
25	R/W	0	sel_pwrdwn_clk_mem	1 = select XTAL_CLK/16 as source clock for clk_mem - this is used in powerdown mode to slow down clk_mem. NOTE: to slow down clk_fpi, clk_mips, clk_mpi & clk_tpi, CLK_FPI_CTL must be set to hex00000005
24:16	R/W	23h	pll0_n	9-bit N parameter to PLL0
15:13	R/W	n/a	Reserved	Read as 0, write nothing
12:8	R/W	3h	pll0_m	5-bit M parameter to PLL0
7:4	R/W	n/a	Reserved	Read as 0, write nothing
3:2	R/W	1h	pll0_p	2-bit P parameter to PLL0. The programming values for P are: 00: P=1, 01: P=2, 10: P=4, 11: P=8
1	R/W	0	pll0_pd	1 = powerdown PLL0
0	R/W	n/a	Reserved	Read as 0, write nothing
Offset 0x04 7004			*PLL1_CTL*	
Reset values are set for expected frequencies for faster boot-up, shorter boot code. Note: When this register is written, PI_ACK must be held in “WAT” until blocking of the clock has started				
31	R	n/a	pll1_blocked	1 = blocking of clock from the PLL is in progress

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
30:25	R/W	n/a	Reserved	Read as 0, write nothing
24:16	R/W	23h	pll1_n	9-bit N parameter to PLL1
15:13	R/W	n/a	Reserved	Read as 0, write nothing
12:8	R/W	3h	pll1_m	5-bit M parameter to PLL1
7:4	R/W	n/a	Reserved	Read as 0, write nothing
3:2	R/W	1h	pll1_p	2-bit P parameter to PLL1. The programming values for P are: 00: P=1, 01: P=2, 10: P=4, 11: P=8
1	R/W	0	pll1_pd	1 = powerdown PLL1
0	R/W	n/a	Reserved	Read as 0, write nothing
Offset 0x04 7008			***PLL2_CTL***	
Reset values are set for expected frequencies for faster boot-up, shorter boot code. Note: When this register is written, PI_ACK must be held in “WAT” until blocking of the clock has started				
31	R	n/a	pll2_blocked	1 = blocking of clock from the PLL is in progress
30:25	R/W	n/a	Reserved	Read as 0, write nothing
24:16	R/W	23h	pll2_n	9-bit N parameter to PLL2
15:13	R/W	n/a	Reserved	Read as 0, write nothing
12:8	R/W	3h	pll2_m	5-bit M parameter to PLL2
7:4	R/W	n/a	Reserved	Read as 0, write nothing
3:2	R/W	1h	pll2_p	2-bit P parameter to PLL2. The programming values for P are: 00: P=1, 01: P=2, 10: P=4, 11: P=8
1	R/W	0	pll2_pd	1 = powerdown PLL2
0	R/W	n/a	Reserved	Read as 0, write nothing
Offset 0x04 700C			***PLL3_CTL***	
Reset values are set for expected frequencies for faster boot-up, shorter boot code. Note: When this register is written, PI_ACK must be held in “WAT” until blocking of the clock has started				
31	R	n/a	pll3_blocked	1 = blocking of clock from the PLL is in progress
30:25	R/W	n/a	Reserved	Read as 0, write nothing
24:16	R/W	23h	pll3_n	9-bit N parameter to PLL3
15:13	R/W	n/a	Reserved	Read as 0, write nothing
12:8	R/W	3h	pll3_m	5-bit M parameter to PLL3
7:4	R/W	n/a	Reserved	Read as 0, write nothing
3:2	R/W	1h	pll3_p	2-bit P parameter to PLL3. The programming values for P are: 00: P=1, 01: P=2, 10: P=4, 11: P=8.
1	R/W	0	pll3_pd	1 = powerdown PLL3
0	R/W	n/a	Reserved	Read as 0, write nothing
Offset 0x04 7010			***PLL4_CTL***	
31	R	n/a	pll4_blocked	1 = blocking of clock from the PLL is in progress
30:25	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
24:16	R/W	1Eh	pll4_n	9-bit N parameter to PLL4
15:13	R/W	n/a	Reserved	Read as 0, write nothing
12:8	R/W	2h	pll4_m	5-bit M parameter to PLL4
7:4	R/W	n/a	Reserved	Read as 0, write nothing
3:2	R/W	3h	pll4_p	2-bit P parameter to PLL4. The programming values for P are: 00: P=1, 01: P=2, 10: P=4, 11: P=8
1	R/W	0	pll4_pd	1 = powerdown PLL4
0	R/W	0	pll4_bp	1 = bypass PLL4
Offset 0x04 7014			*PLL5_CTL*	
31	R	n/a	pll5_blocked	1 = blocking of clock from the PLL is in progress
30:25	R/W	n/a	Reserved	Read as 0, write nothing
24:16	R/W	1Eh	pll5_n	9-bit N parameter to PLL5
15:13	R/W	n/a	Reserved	Read as 0, write nothing
12:8	R/W	2h	pll5_m	5-bit M parameter to PLL5
7:4	R/W	n/a	Reserved	Read as 0, write nothing
3:2	R/W	3h	pll5_p	2-bit P parameter to PLL5. The programming values for P are: 00: P=1, 01: P=2, 10: P=4, 11: P=8
1	R/W	0	pll5_pd	1 = powerdown PLL5
0	R/W	0	pll5_bp	1 = bypass PLL5
Offset 0x04 7018			*PLL1_7GHZ_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2	R/W	0	pll1_7ghz_pd	1 = powerdown PLL1_7GHZ
1:0	R/W	n/a	Reserved	Read as 0, write nothing
Offset 0x04 701C—709C			*RESERVED 0x04 7100DDS0_ICP1_CTL*	
31:0	R/W	0400-0000h	dds0_icp1_ctl[31:0]	32-bit DDS0 control (default = 27MHz)
Offset 0x04 7104			*DDS1_ICP2_CTL*	
31:0	R/W	0400-0000h	dds1_icp2_ctl[31:0]	32-bit DDS1 control (default = 27MHz)
Offset 0x04 7108			*DDS2_AI1_CTL*	
31:0	R/W	05ED-097Bh	dds2_ai1_ctl[31:0]	32-bit DDS2 control (default = 40MHz)
Offset 0x04 710C			*DD3_AO1_CTL*	
31:0	R/W	05ED-097Bh	dds3_ao1_ctl[31:0]	32-bit DDS3 control (default = 40MHz)
Offset 0x04 7110			*DDS4_AI2_CTL*	
31:0	R/W	05ED-097Bh	dds4_ai2_ctl[31:0]	32-bit DDS4 control (default = 40MHz)
Offset 0x04 7114			*DDS5_AO2_CTL*	

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:0	R/W	05ED-097Bh	dds5_ao2_ctl[31:0]	32-bit DDS5 control (default = 40MHz)
Offset 0x04 7118			***DDS6_AIO3_CTL***	
31:0	R/W	05ED-097Bh	dds6_aio3_ctl[31:0]	32-bit DDS6 control (default = 40MHz)
Offset 0x04 711C			***DDS7_SPDO_CTL***	
31:0	R/W	00E9-0452h	dds7_spdo_ctl[31:0]	32-bit DDS7 control (default = 128*48kHz = 6.14MHz)
Offset 0x04 7120			***DDS8_TSDMA_CTL***	
31:0	R/W	038E-38E3h	dds8_tsdma_ctl[31:0]	32-bit DDS8 control (default = 24MHz)
Offset 0x04 7124—719C			***RESERVED 0x04 7200CLK_MEM_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_mem	10 = sel_ext_clk_mem - clk_mem = GPIO[3] 01 = exit_rst_clk_mem - clk_mem = functional clock (PLL0) 00 = clk_mem = 27MHz xtal_clk
0	R/W	1	en_clk_mem	1 = enable clk_mem
Offset 0x04 7204			***CLK_FPI_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3:1	R/W	0h	sel_clk_fpi	100 = sel_ext_clk_fpi - clk_fpi = GPIO[4] 010 = sel_clk_mem_fpi - source of clk_fpi is clk_mem001 = exit_rst_clk_fpi - source of clk_fpi is PLL1 000 = clk_fpi = 27MHz xtal_clk
0	R/W	1	en_clk_fpi	1 = enable clk_fpi
Offset 0x04 7208			***Reserved 0x04 720CCLK_TPI_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_tpi	0 = clk_tpi = 27MHz xtal_clk 1 = clk_tpi = functional clock
0	R/W	1	en_clk_tpi	1 = enable clk_tpi
Offset 0x04 7210			***CLK_MPI_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_mpi	0 = clk_mpi = 27MHz xtal_clk 1 = clk_mpi = functional clock
0	R/W	1	en_clk_mpi	1 = enable clk_mpi
Offset 0x04 7214			***Reserved 0x04 7218CLK_VMPG_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_vmpg	10 = sel_ext_clk_vmpg - clk_vmpg = GPIO[6]01 = exit_rst_clk_vmpg - clk_vmpg = functional clock 00 = clk_vmpg = 27MHz xtal_clk
0	R/W	1	en_clk_vmpg	1 = enable clk_vmpg
Offset 0x04 721C			***CLK_D2D_CTL***	
31:8	R/W	n/a	Reserved	Read as 0, write nothing
7	R/W	1	div_clk_d2d_pd	1 = powerdown clk_d2d divider in the CAB
6:5	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R/W	1	div_clk_d2d	Divide clk_d2d:0 = 86.4MHz - divide 1.728GHz by 20 1 = 108MHz - divide 1.728GHz by 16
3:1	R/W	0h	sel_clk_d2d	100 = sel_ext_clk_d2d - clk_d2d = UA1_TX010 = sel_clk_mbs_d2d - select clk_mbs input 001 = exit_rst_clk_d2d - clk_d2d = functional clock 000 = clk_d2d = 27MHz xtal_clk
0	R/W	1	en_clk_d2d	1 = enable clk_d2d
Offset 0x04 7220			***CLK_VIP1_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2	R/W	0h	sel_invert_clk_vip1	1 = select inverted functional clock
1	R/W	0h	exit_rst_clk_vip1	0 = clk_vip1 = 27MHz xtal_clk 1 = clk_vip1 = functional clock
0	R/W	1	en_clk_vip1	1 = enable clk_vip1
Offset 0x04 7224			***CLK_VIP2_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2	R/W	0h	sel_invert_clk_vip2	1 = select inverted functional clock
1	R/W	0h	exit_rst_clk_vip2	0 = clk_vip2 = 27MHz xtal_clk 1 = clk_vip2 = functional clock
0	R/W	1	en_clk_vip2	1 = enable clk_vip2
Offset 0x04 7228			***CLK_SMART1_CTL***	
(N.B: CLK_TSTAMP_CTL.div_clk_tstamp_pd must be set to ‘0’ for clk_smart1 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_smart1	10 = sel_ext_clk_clk_smart: clk_smart = ext cclock 01 = exit_rst_clk_smart1: clk_smart = functional clock 00 = clk_smart = 27MHz xtal_clk
0	R/W	1	en_clk_smart	1 = enable clk_smart
Offset 0x04 722C			***CLK48_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	1	div_clk48_pd	1 = powerdown clk48 divider in the CAB
2:1	R/W	0h	sel_clk48	10 = sel_ext_clk48 - clk48 = GPIO[8] 01 = exit_rst_clk48 - clk48 = functional clock 00 = clk48 = 27MHz xtal_clk
0	R/W	1	en_clk48	1 = enable clk48
Offset 0x04 7230			***CLK12_CTL***	
(N.B: CLK48_CTL.div_clk48_pd must be set to ‘0’ for clk12 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk12	10 = sel_ext_clk12 - clk12 = SSI_TX01 = exit_rst_clk12 - clk12 = functional clock00 = clk12 = 27MHz xtal_clk
0	R/W	1	en_clk12	1 = enable clk12
Offset 0x04 7234			***CLK_IIC1_CTL***	
(N.B: CLK48_CTL.div_clk48_pd must be set to ‘0’ for clk_iic1 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_iic1	10 = sel_ext_clk_iic1 - clk_iic1 = SSI_TX01 = exit_rst_clk_iic1 - clk_iic1 = functional clock00 = clk_iic1 = 27MHz xtal_clk

CLOCK REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
0	R/W	1	en_clk_iic1	1 = enable clk_iic1
Offset 0x04 7238			***CLK_IIC2_CTL***	
(N.B: CLK48_CTL.div_clk48_pd must be set to ‘0’ for clk_iic2 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_iic2	10 = sel_ext_clk_iic2 - clk_iic2 = SSI_TX 01 = exit_rst_clk_iic2 - clk_iic2 = functional clock 00 = clk_iic2 = 27MHz xtal_clk
0	R/W	1	en_clk_iic2	1 = enable clk_iic2
Offset 0x04 723C			***CLK_UART1_CTL***	
(N.B: CLK48_CTL.div_clk48_pd must be set to ‘0’ for clk_uart1 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_uart1	10 = sel_ext_clk_uart1 - clk_uart1 = SSI_RXD 01 = exit_rst_clk_uart1 - clk_uart1 = functional clock 00 = clk_uart1 = 27MHz xtal_clk
0	R/W	1	en_clk_uart1	1 = enable clk_uart1
Offset 0x04 7240			***CLK_UART2_CTL***	
(N.B: CLK48_CTL.div_clk48_pd must be set to ‘0’ for clk_uart2 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_uart2	10 = sel_ext_clk_uart2 - clk_uart2 = SSI_RXD 01 = exit_rst_clk_uart2 - clk_uart2 = functional clock 00 = clk_ uart2 = 27MHz xtal_clk
0	R/W	1	en_clk_uart2	1 = enable clk_uart2
Offset 0x04 7244			***CLK_UART3_CTL***	
(N.B: CLK48_CTL.div_clk48_pd must be set to ‘0’ for clk_uart3 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_ uart3	10 = sel_ext_clk_ uart3 - clk_ uart3 = SSI_RXD 01 = exit_rst_clk_ uart3 - clk_ uart3 = functional clock 00 = clk_ uart3 = 27MHz xtal_clk
0	R/W	1	en_clk_uart3	1 = enable clk_uart3
Offset 0x04 7248			***MSP1_IN_CLK_SRC_CTL***	
31:20	R/W	n/a	Reserved	Read as 0, write nothing
19:16	R/W	0h	sel_msp1_in_clk_src2	1001 = select clock specified by bits [9:8] 0110 = select 1394rx_clk 0101 = select ts_p22_clk 0100 = select ts_p21_clk 0011 = select ts_p12_clk 0010 = select ts_p11_clk 0001 = select DV3_clk 0000 = select DV2_clk
15:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_msp1_in_clk_src	11 = select clock specified by bits [2:0] 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_msp1_in_clk_div8	101 = select dds8_clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 724C			***MSP1_IN_CLK_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:1	R/W	0h	sel_msp1_in_clk_4x1 mux	10 = sel_ext_clk_msp1 - clk_msp1 = SSI_SCLK_CTSN 01 = exit_rst_clk_msp1 - clk_msp1 = functional clock 00 = clk_msp1 = 27MHz xtal_clk
0	R/W	1	en_msp1_in_clk	1 = enable clk_msp1
Offset 0x04 7250			***CLK_PCI_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	div_clk_pci_pd	1 = powerdown clk_pci divider in the CAB
2:1	R/W	0h	sel_clk_pci	10 = sel_ext_clk_pci - clk_pci = GPIO[7] 01 = exit_rst_clk_pci - clk_pci = functional clock 00 = clk_pci = 27MHz xtal_clk
0	R/W	1	en_clk_pci	1 = enable clk_pci
Offset 0x04 7254			***CLK_MBS_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_mbs	10 = sel_ext_clk_mbs - clk_mbs = GPIO[5] 01 = exit_rst_clk_mbs - clk_mbs = functional clock 00 = clk_mbs = 27MHz xtal_clk
0	R/W	1	en_clk_mbs	1 = enable clk_mbs
Offset 0x04 7258			***CLK_SPDI_CTL***	
31:7	R/W	n/a	Reserved	Read as 0, write nothing
6	R/W	0h	div_clk_spdi_pd	1 = powerdown clk_spdi divider in the CAB
5:3	R/W	0h	div_clk_spdi	000 = clk_spdi = 144MHz, from CAB 001 = clk_spdi = 72MHZ, div by 2 in wsg_clock
2:1	R/W	0h	sel_clk_spdi	10 = sel_ext_clk_spdi - clk_spdi = UA2_TX 01 = exit_rst_clk_spdi - clk_spdi = functional clock 00 = clk_spdi = 27MHz xtal_clk
0	R/W	1	en_clk_spdi	1 = enable clk_spdi
Offset 0x04 725C			***CLK_TSTAMP_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0	div_clk_tstamp_pd	1 = powerdown clk_tstamp divider in the CAB
2:1	R/W	0h	sel_clk_tstamp	10 = sel_ext_clk_tstamp - clk_tstamp = UA1_RX 01 = exit_rst_clk_tstamp - clk_tstamp = functional clock 00 = clk_tstamp = 27MHz xtal_clk
0	R/W	1	en_clk_tstamp	1 = enable clk_tstamp
Offset 0x04 7260			***CLK_SPY_CTL***	
(N.B: CLK_TSTAMP_CTL.div_clk_tstamp_pd must be set to ‘0’ for clk_spy functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_spy	10 = sel_ext_clk_spy - clk_spy = SSI_FS_RTSN 01 = exit_rst_clk_spy - clk_spy = functional clock 00 = clk_spy = 27MHz xtal_clk
0	R/W	1	en_clk_spy	1 = enable clk_spy
Offset 0x04 7264			***CLK50_CTL***	
(also controls CLK25 since they must be switched simultaneously)				
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	1	exit_rst_clk50	0 = clk50 = 27MHz xtal_clk 1 = clk50 = functional clock

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	R/W	1	en_clk50	1 = enable clk50
Offset 0x04 7268			***CLK_VMSP1_CTL***	
(N.B: CLK_TSTAMP_CTL.div_clk_tstamp_pd must be set to ‘0’ for clk_vmsp1 functional clock to work)				
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_vmsp1	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_vmsp1	10 = sel_ext_clk_vmsp1 - clk_vmsp1 = UA1_RX 01 = exit_rst_clk_vmsp1 - clk_vmsp1 = functional clock 00 = clk_vmsp1 = 27MHz xtal_clk
0	R/W	1	en_clk_vmsp1	1 = enable clk_vmsp1
Offset 0x04 726C			***CLK_VMSP2_CTL***	
(N.B: CLK_TSTAMP_CTL.div_clk_tstamp_pd must be set to ‘0’ for clk_vmsp2 functional clock to work)				
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_vmsp2	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_vmsp2	10 = sel_ext_clk_vmsp2 - clk_vmsp2 = UA1_RX 01 = exit_rst_clk_vmsp2 - clk_vmsp2 = functional clock 00 = clk_vmsp2 = 27MHz xtal_clk
0	R/W	1	en_clk_vmsp2	1 = enable clk_vmsp2
Offset 0x04 7270			***CLK_NDS1_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_nds1	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_nds1	10 = sel_ext_clk_nds1 - clk_nds1 = AIO_OSCLK 01 = exit_rst_clk_nds1 - clk_nds1 = functional clock 00 = clk_nds1 = 27MHz xtal_clk
0	R/W	1	en_clk_nds1	1 = enable clk_nds1
Offset 0x04 7274			***CLK_NDS2_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_nds2	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_nds2	10 = sel_ext_clk_nds2 - clk_nds2 = AIO_OSCLK 01 = exit_rst_clk_nds2 - clk_nds2 = functional clock 00 = clk_nds2 = 27MHz xtal_clk
0	R/W	1	en_clk_nds2	1 = enable clk_nds2
Offset 0x04 7278			***CLK_1394TX1_SRC_CTL***	
31:20	R/W	n/a	Reserved	Read as 0, write nothing
19:16	R/W	0h	sel_clk_1394tx1_src2	1001 = select clock specified by bits [9:8] 0111 = select mspout2_clk 0110 = select clk_tsdma 0101 = select ts_p22_clk 0100 = select ts_p21_clk 0011 = select ts_p12_clk 0010 = select ts_p11_clk 0001 = select DV3_clk 0000 = select DV2_clk
15:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_clk_1394tx1_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:0	R/W	0h	sel_clk_1394tx1_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 727C			*CLK_1394TX1_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_1394tx1_4x1 mux	10 = sel_ext_clk_1394tx1 - clk_1394tx1 = SSI_SCLK_CTSN 01 = exit_rst_clk_1394tx1 - clk_1394tx1 = functional clock 00 = clk_1394tx1 = 27MHz xtal_clk
0	R/W	1	en_clk_1394tx1	1 = enable clk_1394tx1
Offset 0x04 7280			*CLK_1394TX2__SRC_CTL*	
31:20	R/W	n/a	Reserved	Read as 0, write nothing
19:16	R/W	0h	sel_clk_1394tx2_src2	1001 = select clock specified by bits [9:8] 0111 = select mspout2_clk 0110 = select clk_tsdma 0101 = select ts_p22_clk 0100 = select ts_p21_clk 0011 = select ts_p12_clk 0010 = select ts_p11_clk 0001 = select DV3_clk 0000 = select DV2_clk
15:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_clk_1394tx2_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_clk_1394tx2_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 7284			*CLK_1394TX2_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_1394tx2_4x1 mux	10 = sel_ext_clk_1394tx2 - clk_1394tx2 = SSI_SCLK_CTSN 01 = exit_rst_clk_1394tx2 - clk_1394tx2 = functional clock 00 = clk_1394tx2 = 27MHz xtal_clk
0	R/W	1	en_clk_1394tx2	1 = enable clk_1394tx2
Offset 0x04 7288			*CLK_TSDMA_SRC_CTL*	
31:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_clk_tsdma_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_clk_tsdma_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 728C			*CLK_TSDMA_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_tsdma	1 = select inverted functional clock

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:1	R/W	0h	sel_clk_tsdma_4x1mux	10 = sel_ext_clk_tsdma - clk_tsdma = SSI_SCLK_CTSN 01 = exit_rst_clk_tsdma - clk_tsdma = functional clock 00 = clk_tsdma = 27MHz xtal_clk
0	R/W	1	en_clk_tsdma	1 = enable clk_tsdma
Offset 0x04 7290			*TSOUT_CLK_OUT_SRC_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3:0	R/W	0h	sel_tsout_clk_out_src	1000 = select dds8_clk_tsdma 0111 = select clk_tsdma 0110 = select mspout1_clk 0101 = select mspout2_clk 0100 = select 1394rx_clk 0011 = select ts_s22_clk 0010 = select DV3_clk 0001 = select ts_s12_clk 0000 = select DV2_clk
Offset 0x04 7294			*TSOUT_CLK_OUT_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_tsout_clk_out	1 = select inverted functional clock
2:1	R/W	0h	sel_tsout_clk_out_4x1mux	10 = sel_ext_tsout_clk_out - tsout_clk_out = AIO_OSCLK 01 = exit_rst_tsout_clk_out - tsout_clk_out = functional clock 00 = tsout_clk_out = 27MHz xtal_clk
0	R/W	1	en_tsout_clk_out	1 = enable tsout_clk_out
Offset 0x04 7298			*TSOUT_SERIAL_CLK_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_tsout_seial_clk	1 = select inverted functional clock
2:1	R/W	0h	sel_tsout_serial_clk	10 = sel_ext_tsout_serial_clk - tsout_serial_clk = AIO_OSCLK 01 = exit_rst_tsout_serial_clk - tsout_serial_clk = functional clock 00 = tsout_serial_clk = 27MHz xtal_clk
0	R/W	1	en_tsout_serial_clk	1 = enable tsout_serial_clk
Offset 0x04 729C			*RESERVED 0x04 7300CLK_ICP1_MUX_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_icp1_mux	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_icp1_mux	10 = sel_ext_clk_icp1_mux - clk_icp1_mux = UA2_RX 01 = exit_rst_clk_icp1_mux - clk_icp1_mux = functional clock 00 = clk_icp1_mux = 27MHz xtal_clk
0	R/W	1	en_clk_icp1_mux	1 = enable clk_icp1_mux
Offset 0x04 7304			*CLK_ICP1_CTL*	
31:7	R/W	n/a	Reserved	Read as 0, write nothing
6	R/W	0	sel_invert_clk_icp1	1 = select inverted clock from divider
5:3	R/W	1	div_clk_icp1	Divide clk_icp1: 101 = divide clock by 8 100 = divide clock by 6 011 = divide clock by 4 010 = divide clock by 3 001 = divide clock by 2 000 = no divide
2:1	R/W	0h	sel_clk_icp1	10 = sel_ext_clk_icp1 - clk_icp1 = UA2_RTSN 01 = exit_rst_clk_icp1 - clk_icp1 = functional clock 00 = clk_icp1 = 27MHz xtal_clk

CLOCK REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
0	R/W	1	en_clk_icp1	1 = enable clk_icp1
Offset 0x04 7308			*CLK_ICP2_MUX_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_icp2_mux	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_icp2_mux	10 = sel_ext_clk_icp2_mux - clk_icp2_mux = UA2_RX 01 = exit_rst_clk_icp2_mux - clk_icp2_mux = functional clock 00 = clk_icp2_mux = 27MHz xtal_clk
0	R/W	1	en_clk_icp2_mux	1 = enable clk_icp2_mux
Offset 0x04 730C			*CLK_ICP2_CTL*	
31:7	R/W	n/a	Reserved	Read as 0, write nothing
6	R/W	0	sel_invert_clk_icp2	1 = select inverted clock from divider
5:3	R/W	1	div_clk_icp2	Divide clk_icp1: 101 = divide clock by 8 100 = divide clock by 6 011 = divide clock by 4 010 = divide clock by 3 001 = divide clock by 2 000 = no divide
2:1	R/W	0h	sel_clk_icp2	10 = sel_ext_clk_icp2 - clk_icp2 = UA2_RTSN 01 = exit_rst_clk_icp2 - clk_icp2 = functional clock 00 = clk_icp2 = 27MHz xtal_clk
0	R/W	1	en_clk_icp2	1 = enable clk_icp2
Offset 0x04 7310			*AI1_OSCLK_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_ai1_osclk	10 = sel_ext_ai1_osclk - ai1_osclk = UA2_CTSN 01 = exit_rst_ai1_osclk - ai1_osclk = functional clock 00 = ai1_osclk = 27MHz xtal_clk
0	R/W	1	en_ai1_osclk	1 = enable ai1_osclk
Offset 0x04 7314			*AO1_OSCLK_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_ao1_osclk	10 = sel_ext_ao1_osclk - ao1_osclk = UA2_CTSN 01 = exit_rst_ao1_osclk - ao1_osclk = functional clock 00 = ao1_osclk = 27MHz xtal_clk
0	R/W	1	en_ao1_osclk	1 = enable ao1_osclk
Offset 0x04 7318			*AI2_OSCLK_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_ai2_osclk	10 = sel_ext_ai2_osclk - ai2_osclk = UA2_CTSN 01 = exit_rst_ai2_osclk - ai2_osclk = functional clock 00 = ai2_osclk = 27MHz xtal_clk
0	R/W	1	en_ai2_osclk	1 = enable ai2_osclk
Offset 0x04 731C			*AO2_OSCLK_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_ao2_osclk	10 = sel_ext_ao2_osclk - ao2_osclk = UA2_CTSN 01 = exit_rst_ao2_osclk - ao2_osclk = functional clock 00 = ao2_osclk = 27MHz xtal_clk

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	R/W	1	en_ao2_osclk	1 = enable ao2_osclk
Offset 0x04 7320			***AIO3_OSCLK_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_aio3_osclk	10 = sel_ext_aio3_osclk - aio3_osclk = UA2_CTSN 01 = exit_rst_aio3_osclk - aio3_osclk = functional clock 00 = aio3_osclk = 27MHz xtal_clk
0	R/W	1	en_aio3_osclk	1 = enable aio3_osclk
Offset 0x04 7324			***CLK_SPDO_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_clk_spdo	10 = sel_ext_clk_spdo - clk_spdo = UA1_RX 01 = exit_rst_clk_spdo - clk_spdo = functional clock 00 = clk_spdo = 27MHz xtal_clk
0	R/W	1	en_clk_spdo	1 = enable clk_spdo
Offset 0x04 7328—739C			***RESERVED 0x04 7400GPIO_CLK_Q0_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_gpio_clkout_q0	0 = gpio_clkout_q0 = 27MHz xtal_clk 1 = gpio_clkout_q0 = functional clock
0	R/W	1	en_gpio_clkout_q0	1 = enable gpio_clkout_q0
Offset 0x04 7404			***GPIO_CLK_Q1_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_gpio_clkout_q1	0 = gpio_clkout_q1 = 27MHz xtal_clk 1 = gpio_clkout_q1 = functional clock
0	R/W	1	en_gpio_clkout_q1	1 = enable gpio_clkout_q1
Offset 0x04 7408			***GPIO_CLK_Q2_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_gpio_clkout_q2	0 = gpio_clkout_q2 = 27MHz xtal_clk 1 = gpio_clkout_q2 = functional clock
0	R/W	1	en_gpio_clkout_q2	1 = enable gpio_clkout_q2
Offset 0x04 740C			***GPIO_CLK_Q3_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_gpio_clkout_q3	0 = gpio_clkout_q3 = 27MHz xtal_clk 1 = gpio_clkout_q3 = functional clock
0	R/W	1	en_gpio_clkout_q3	1 = enable gpio_clkout_q3
Offset 0x04 7410—741C			***Reserved 0x04 7420CLK_VIP1_SEL_CTL***	
(NB: VIP1 data muxing must be programmed in the IO-MUX to match the programming of the VIP1 clock muxing)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_clk_vip1_src	100 = source is clk_1394rx 011 = source is clk_icp_out2 010 = source is DV3_CLK pad 001 = source is DV2_CLK pad 000 = source is DV1_CLK pad
Offset 0x04 7424			***CLK_VIP2_SEL_CTL***	
(NB: VIP2 data muxing must be programmed in the IO-MUX to match the programming of the VIP2 clock muxing)				

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_clk_vip2_src	100 = source is clk_1394rx 011 = source is clk_icp_out1 010 = source is DV3_CLK pad 001 = source is DV2_CLK pad 000 = source is DV1_CLK pad
Offset 0x04 7428			***CLK_SMART2_CTL***	
(N.B: CLK_TSTAMP_CTL.div_clk_tstamp_pd must be set to ‘0’ for clk_smart2 functional clock to work)				
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0	sel_clk_smart2	10 = sel_ext_clk_clk_smart2: clk_smart2 = ext clock 01 = exit_rst_clk_smart2: clk_smart2 = functional clock 00 = clk_smart2 = 27MHz xtal_clk
0	R/W	1	en_clk_smart2	1 = enable clk_smart2
Offset 0x04 742C			***CLK_AI1_SCK_O_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_ai1_sck_o	0 = clk_ai1_sck_o = 27MHz xtal_clk 1 = clk_ai1_sck_o = functional clock
0	R/W	1	en_clk_ai1_sck_o	1 = enable clk_ai1_sck_o
Offset 0x04 7430			***CLK_AO1_SCK_O_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_ao1_sck_o	0 = clk_ao1_sck_o = 27MHz xtal_clk 1 = clk_ao1_sck_o = functional clock
0	R/W	1	en_clk_ao1_sck_o	1 = enable clk_ao1_sck_o
Offset 0x04 7434			***CLK_AI2_SCK_O_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_ai2_sck_o	0 = clk_ai2_sck_o = 27MHz xtal_clk 1 = clk_ai2_sck_o = functional clock
0	R/W	1	en_clk_ai2_sck_o	1 = enable clk_ai2_sck_o
Offset 0x04 7438			***CLK_AO2_SCK_O_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_ao2_sck_o	0 = clk_ao2_sck_o = 27MHz xtal_clk 1 = clk_ao2_sck_o = functional clock
0	R/W	1	en_clk_ao2_sck_o	1 = enable clk_ao2_sck_o
Offset 0x04 743C			***CLK_AIO3_SCK_O_CTL***	
31:2	R/W	n/a	Reserved	Read as 0, write nothing
1	R/W	0	exit_rst_clk_aio3_sck_o	0 = clk_aio3_sck_o = 27MHz xtal_clk 1 = clk_aio3_sck_o = functional clock
0	R/W	1	en_clk_aio3_sck_o	1 = enable clk_aio3_sck_o
Offset 0x04 7440—745C			***RESERVED 0x04 7460CLK_OPTION_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:0	R/W	0	sel_clk_option_div	101 = divide clock by 8: 2.25MHz 100 = divide clock by 6: 3.0MHz 011 = divide clock by 4: 4.5MHz 010 = divide clock by 3: 6MHz 001 = divide clock by 2: 9MHz 000 = no divide: 18MHz
Offset 0x04 7464			***MSPOUT1_CLK_SRC_CTL***	
31:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_mspout1_clk_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_mspout1_clk_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 7468			***MSPOUT1_CLK_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_mspout1_clk	1 = select inverted functional clock
2:1	R/W	0h	sel_mspout1_clk_4x1 mux	10 = sel_ext_mspout1_clk - mspout1_clk = SSI_SCLK_CTSN 01 = exit_rst_mspout1_clk - mspout1_clk = functional clock 00 = mspout1_clk = 27MHz xtal_clk
0	R/W	1	en_mspout1_clk	1 = enable mspout1_clk
Offset 0x04 746C			***MSPOUT2_CLK_SRC_CTL***	
31:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_mspout2_clk_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_mspout2_clk_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 7470			***MSPOUT2_CLK_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_mspout2_clk	1 = select inverted functional clock
2:1	R/W	0h	sel_mspout2_clk_4x1 mux	10 = sel_ext_mspout2_clk - mspout2_clk = SSI_SCLK_CTSN 01 = exit_rst_mspout2_clk - mspout2_clk = functional clock 00 = mspout2_clk = 27MHz xtal_clk
0	R/W	1	en_mspout2_clk	1 = enable mspout2_clk
Offset 0x04 7474			***CLK_1394RX_SRC_CTL***	
31:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_clk_1394rx_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:0	R/W	0h	sel_clk_1394rx_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 7478			*CLK_1394RX_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_1394rx_clk	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_1394rx_4x1mux	10 = sel_ext_clk_1394rx - clk_1394rx = SSI_SCLK_CTSN 01 = exit_rst_clk_1394rx - clk_1394rx = functional clock 00 = clk_1394rx = 27MHz xtal_clk
0	R/W	1	en_clk_1394rx	1 = enable clk_1394rx
Offset 0x04 747C			*MSP2_IN_CLK_SRC_CTL*	
31:20	R/W	n/a	Reserved	Read as 0, write nothing
19:16	R/W	0h	sel_msp2_in_clk_src2	1001 = select clock specified by bits [9:8] 0110 = select 1394rx_clk 0101 = select ts_p22_clk 0100 = select ts_p21_clk 0011 = select ts_p12_clk 0010 = select ts_p11_clk 0001 = select DV3_clk 0000 = select DV2_clk
15:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_msp2_in_clk_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_msp2_in_clk_div8	101 = select dds8_clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 7480			*MSP2_IN_CLK_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_msp2_in_clk_4x1mux	10 = sel_ext_clk_msp2 - clk_msp2 = SSI_SCLK_CTSN 01 = exit_rst_clk_msp2 - clk_msp2 = functional clock 00 = clk_msp2 = 27MHz xtal_clk
0	R/W	1	en_msp2_in_clk	1 = enable clk_msp2
Offset 0x04 7484			*TSOUT_PARALLEL_CLK_SRC_CTL*	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_tsout_parallel_clk_src	110 = select parallel clock (divided by 8) 101 = select clk_tsdma 100 = select mspout1_clk 011 = select mspout2_clk 010 = select 1394rx_clk 001 = select DV3_clk 000 = select DV2_clk
Offset 0x04 7488			*TSOUT_PARALLEL_CLK_CTL*	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_tsout_parallel_clk	1 = select inverted functional clock
2:1	R/W	0h	sel_tsout_parallel_clk_4x1mux	10 = sel_ext_tsout_parallel_clk - tsout_parallel_clk = AIO_WS 01 = exit_rst_tsout_parallel_clk - tsout_parallel_clk = functional clock 00 = tsout_parallel_clk = 27MHz xtal_clk

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	R/W	0	en_tsout_parallel_clk	1 = enable tsout_parallel_clk
Offset 0x04 748C			***CLK_VMSP3_CTL***	
(N.B: CLK_TSTAMP_CTL.div_clk_tstamp_pd must be set to ‘0’ for clk_vmsp3 functional clock to work)				
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_clk_vmsp3	1 = select inverted functional clock
2:1	R/W	0h	sel_clk_vmsp3	10 = sel_ext_clk_vmsp3 - clk_vmsp1 = UA1_RX 01 = exit_rst_clk_vmsp3 - clk_vmsp1 = functional clock 00 = clk_vmsp3 = 27MHz xtal_clk
0	R/W	1	en_clk_vmsp3	1 = enable clk_vmsp3
Offset 0x04 7490			***MSP3_IN_CLK_SRC_CTL***	
31:20	R/W	n/a	Reserved	Read as 0, write nothing
19:16	R/W	0h	sel_msp3_in_clk_src2	1001 = select clock specified by bits [9:8] 0110 = select 1394rx_clk 0101 = select ts_p22_clk 0100 = select ts_p21_clk 0011 = select ts_p12_clk 0010 = select ts_p11_clk 0001 = select DV3_clk 0000 = select DV2_clk
15:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_msp3_in_clk_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_msp3_in_clk_div8	101 = select dds8_clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 7494			***MSP3_IN_CLK_CTL***	
31:3	R/W	n/a	Reserved	Read as 0, write nothing
2:1	R/W	0h	sel_msp3_in_clk_4x1 mux	10 = sel_ext_clk_msp3 - clk_msp3 = SSI_SCLK_CTSN 01 = exit_rst_clk_msp3 - clk_msp3 = functional clock 00 = clk_msp3 = 27MHz xtal_clk
0	R/W	1	en_msp3_in_clk	1 = enable clk_msp2
Offset 0x04 7498			***MSPOUT3_CLK_SRC_CTL***	
31:10	R/W	n/a	Reserved	Read as 0, write nothing
9:8	R/W	0h	sel_mspout3_clk_src	11 = select clock divided by 8 (parallelised) 10 = select dds8_clk_tsdma 01 = select DV3_clk 00 = select DV2_clk
7:3	R/W	n/a	Reserved	Read as 0, write nothing
2:0	R/W	0h	sel_mspout3_clk_div8	101 = select clk_option to be divided by 8 100 = select dds8_clk_tsdma to be divided by 8 011 = select ts_s22_clk to be divided by 8 010 = select DV3_clk to be divided by 8 001 = select ts_s12_clk to be divided by 8 000 = select DV2_clk to be divided by 8
Offset 0x04 749C			***MSPOUT3_CLK_CTL***	
31:4	R/W	n/a	Reserved	Read as 0, write nothing
3	R/W	0h	sel_invert_mspout3_clk	1 = select inverted functional clock

CLOCK REGISTERS

Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:1	R/W	0h	sel_mspout3_clk_4x1 mux	10 = sel_ext_mspout3_clk - mspout3_clk = SSI_SCLK_CTSN 01 = exit_rst_mspout3_clk - mspout3_clk = functional clock 00 = mspout3_clk = 27MHz xtal_clk
0	R/W	1	en_mspout3_clk	1 = enable mspout3_clk
Offset 0x04 74A0—7FF8			***RESERVED 0x04 7FE0INTERRUPT STATUS***	
31	R	0	ts_s22_clk_present	1 = Clock present 0 = Clock NOT present
30	R	0	ts_s12_clk_present	1 = Clock present 0 = Clock NOT present
29	R	0	aio3_sckin_present	1 = Clock present 0 = Clock NOT present
28	R	0	ai2_sckin_present	1 = Clock present 0 = Clock NOT present
27	R	0	ai1_sckin_present	1 = Clock present 0 = Clock NOT present
26	R	0	clk_1394_present	1 = Clock present 0 = Clock NOT present
25	R	0	dv3_clk_present	1 = Clock present 0 = Clock NOT present
24	R	0	dv2_clk_present	1 = Clock present 0 = Clock NOT present
23	R	0	dv1_clk_present	1 = Clock present 0 = Clock NOT present
22:9	R/W	n/a	Reserved	Read as 0, write nothing
8	R	0	ts_s22_clk_int	1 = Clock int
7	R	0	ts_s12_clk_int	1 = Clock int
6	R	0	aio3_sckin_int	1 = Clock int
5	R	0	ai2_sckin_int	1 = Clock int
4	R	0	ai1_sckin_int	1 = Clock int
3	R	0	clk_1394_int	1 = Clock int
2	R	0	dv3_clk_int	1 = Clock int
1	R	0	dv2_clk_int	1 = Clock int
0	R	0	dv1_clk_int	1 = Clock int
Offset 0x04 7FE4			***INTERRUPT ENABLE***	
31:9	R/W	n/a	Reserved	Read as 0, write nothing
8	R/W	0	ts_s22_clk_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
7	R/W	0	ts_s12_clk_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
6	R/W	0	aio3_sckin_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
5	R/W	0	ai2_sckin_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
4	R/W	0	ai1_sckin_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
3	R/W	0	clk_1394_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
2	R/W	0	dv3_clk_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
1	R/W	0	dv2_clk_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
0	R/W	0	dv1_clk_int enable	1 = Interrupt enabled 0 = Interrupt NOT enabled
Offset 0x04 7FE8			***INTERRUPT CLEAR***	
31:9	R/W	n/a	Reserved	Read as 0, write nothing

CLOCK REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
8	R/W	0	clear ts_s22_clk_int	1 = clear interrupt
7	R/W	0	clear ts_s12_clk_int	1 = clear interrupt
6	R/W	0	clear aio3_sckin_int	1 = clear interrupt
5	R/W	0	clear ai2_sckin_int	1 = clear interrupt
4	R/W	0	clear ai1_sckin_int	1 = clear interrupt
3	R/W	0	clear clk_1394_int	1 = clear interrupt
2	R/W	0	clear dv3_clk_int	1 = clear interrupt
1	R/W	0	clear dv2_clk_int	1 = clear interrupt
0	R/W	0	clear dv1_clk_int	1 = clear interrupt
Offset 0x04 7FEC			*SET INTERRUPT*	
31:9	R/W	n/a	Reserved	Read as 0, write nothing
8	R/W	0	set ts_s22_clk_int	1 = set interrupt
7	R/W	0	set ts_s12_clk_int	1 = set interrupt
6	R/W	0	set aio3_sckin_int	1 = set interrupt
5	R/W	0	set ai2_sckin_int	1 = set interrupt
4	R/W	0	set ai1_sckin_int	1 = set interrupt
3	R/W	0	set clk_1394_int	1 = set interrupt
2	R/W	0	set dv3_clk_int	1 = set interrupt
1	R/W	0	set dv2_clk_int	1 = set interrupt
0	R/W	0	set dv1_clk_int	1 = set interrupt
Offset 0x04 7FF0			*RESERVED 0x04 7FF4POWERDOWN*	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value.1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0	-	-	Unused	Ignore during writes and read as zeroes.
Offset 0x04 7FF8			*RESERVED 0x04 7FFCMODULE_ID*	
31:16	R	0x0108	module_id	Module ID
15:12	R	1	rev_major	Major revision
11:8	R	0	rev_minor	Minor revision
7:0	R	0	app_size	Aperture size is 0 = 4 kB.

5.3 Reset Module Functional Description

The Reset module generates all reset signals required for the PNX8526:

- Reset to all PI-Bus peripherals, which also initiate the Boot module.

- Separate MIPS reset signal.
- Reset for all external devices.

These resets are triggered or released in a number of ways:

- External reset input to the PNX8526
- Programmable "assert reset" and "release reset" registers
- Watchdog timeout issued by watchdog timer in MIPS CPU causes all resets to be asserted
- Programmable "do" software system reset, which also asserts all resets.

The Reset module also provides control over the endianness of the system.

5.3.1 Overview

Reset is a module on the M-PI Bus, as shown in Figure 3.

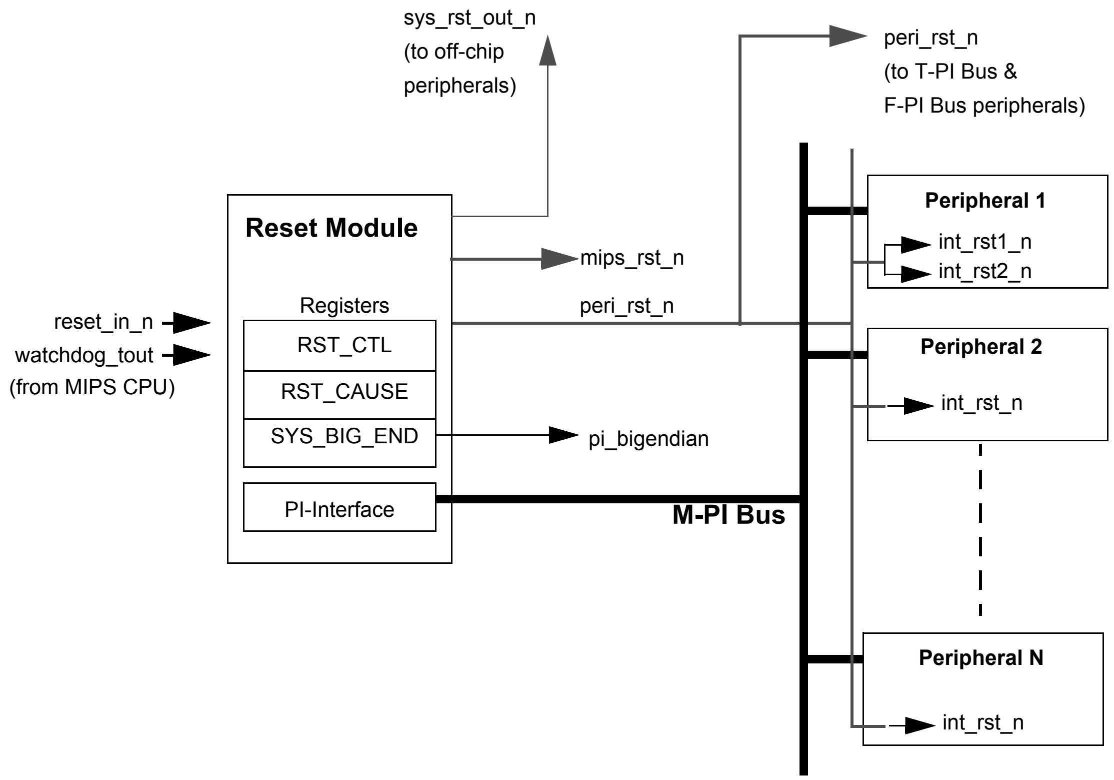


Figure 3: PNX8526 Resets

During normal boot of the system, the external reset input to the PNX8526, reset_in_n, will be asserted following power up of the system. This causes the assertion of all the reset signals as shown in Figure 4. When the Clock module receives the peri_rst_n, it ensures that all modules receive the 27 MHz crystal oscillator input. The 27 MHz clock to all modules will remain until the Boot module programs a register in the Clock module to switch from 27 MHz to the functional module clocks. This allows all modules to be reset synchronously with the 27 MHz clock and all modules will be designed to operate at a minimum of 27 MHz.

After de-asserting the reset_in_n signal, the peri_rst_n is also de-asserted and all peripherals release their internal resets synchronously. The Boot module will set up the system configuration registers and then release the MIPS reset by writing to the "release MIPS reset" bit (rel_mips_rst_n) of the RST_CTL register. The MIPS processor will later release the sys_rst_out_n by writing to the "release external reset" bit, rel_sys_rst_out, bit in the RST_CTL register.

After the Boot module has programmed all PLLs, it will set a "switch_to_modclks" register in the Clock module and the Clock module will safely switch from the 27 MHz clock to the separate module functional clocks. This is shown in Figure 4

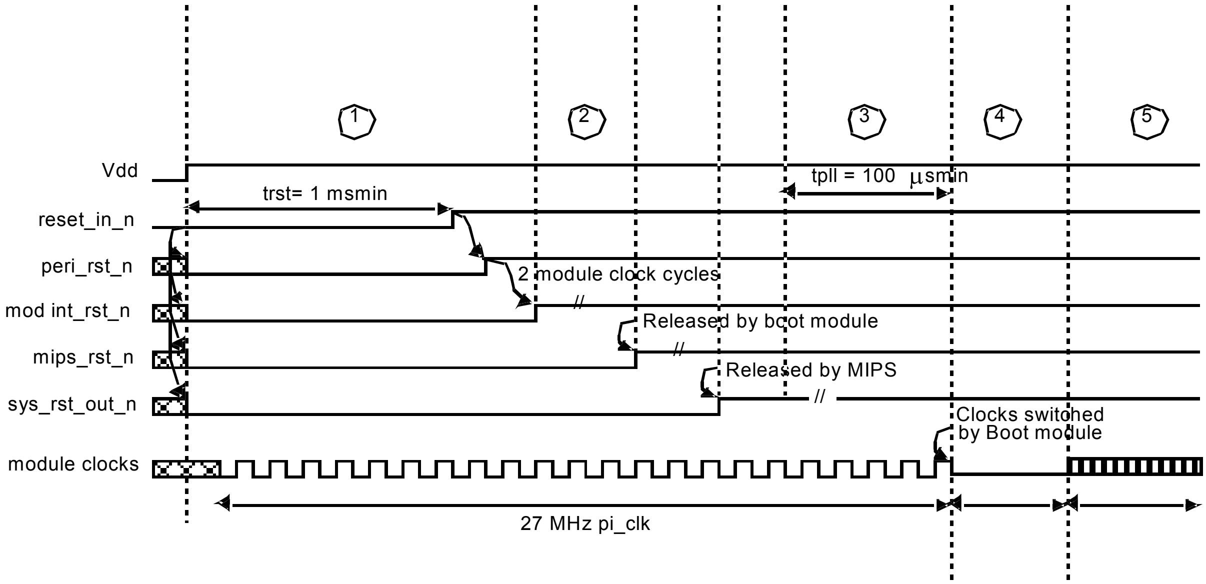


Figure 4: Reset Timing Diagram

Referring to above:

1. reset_in_n asserted for 1 msmin after power-good. peri_rst_n follows the release of reset_in_n. Clock module kicks off 27 MHz clock to all modules.
2. All module resets sync to 27 MHz and all modules are reset at the same time. The Boot script can now kick off.
3. Boot script programs all PLLs to desired frequencies, then waits for PLL settle time (100 µsmin) before programming "switch_to_modclks" reg in the Clock module.
4. All module clocks are blocked in the clock module to ensure safe, glitchless switch over from 27 MHz to module clocks.
5. All modules are now receiving their required clocks for normal function.

According to PCI Standards, during power up reset_in_n must remain asserted for a minimum of 1 ms (t_{rst}). At all other times reset_in_n must be asserted for a minimum of 100 μs.

Remark: The entire boot-up sequence shown in Figure 4 is repeated in the event of a software reset or a MIPS CPU watchdog timeout.

In both cases, the Clock module will switch all module clocks to 27 MHz; peri_rst_n will be asserted for 100 μs before being released, and the Boot module will run the boot-up script again.

5.3.1.1 Chip I/O

Table 16: Reset Module I/O

Chip Port	Type	Description
RESET_IN	I	External reset input to the PNX8526—activates all resets generated in the Reset module. The pad has hysterisis control to prevent spikes on RESET_IN.
SYS_RSTN_OUT	O	Reset output from the PNX8526 chip to reset external devices. Note: the pad is a PCI IO pad which will always be used as an output.

5.3.1.2 Major Interfaces

The PIO interfaces to the M-PI Bus and allows PI read/write access to the Reset module registers.

5.3.2 Operation

The Reset module has four major blocks:

- PIO interface which allows PI-Bus read/write access to the Reset module configuration registers.
- PERI_RST State Machine generates reset to peripherals, MMI, TM32 CPU core.
- MIPS_RST State Machine generates reset to MIPS CPU.
- SYS_RST_OUT State Machine generates reset to external devices.

The following sections describe the operation of the reset state machines. All reset signals are generated “glitch-free.”

5.3.2.1 PERI_RST State Machine

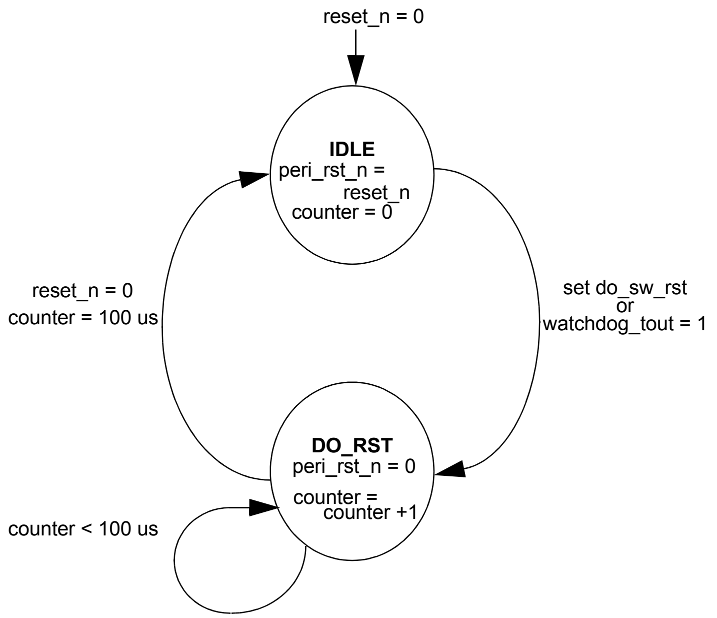


Figure 5: PERI State Machine

The peri_rst_n signal is asserted if a software reset is programmed or the MIPS watchdog timer issues a timeout signal or an external reset_in_n is asserted. Both software reset and watchdog timeout reset cause peri_rst_n to be asserted for 100 µs. A counter is triggered to count 100 µs and will run at the pi_clk frequency of 27 MHz during reset. The peri_rst_n signal is synchronized in each PI-Bus peripheral.

5.3.2.2 MIPS_RST State Machine

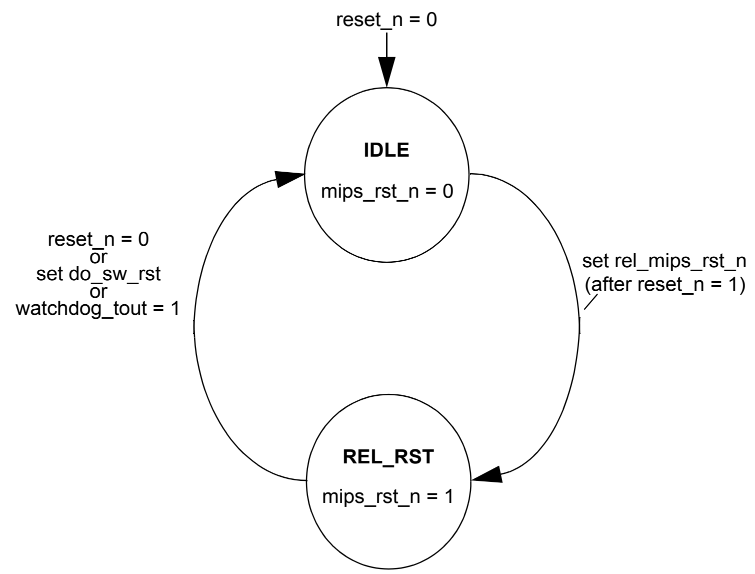


Figure 6: MIPS_RST State Machine

In the normal sequence of events, after power up of the PNX8526, the Boot module will set rel_mips_rst_n allowing the MIPS CPU to get out of reset and start fetching instructions (see Figure 4). The mips_rst_n may be asserted again if:

- A software reset is issued by setting the do_sw_rst bit of the RST_CTL register.
- A timeout is issued by the watchdog timer in the MIPS CPU.
- An external reset_in_n is asserted.

In the case of a software reset or watchdog timeout, a full system reboot sequence is conducted and mips_rst_n will remain asserted until released by the Boot module.

5.3.2.3 SYS_RST_OUT State Machine

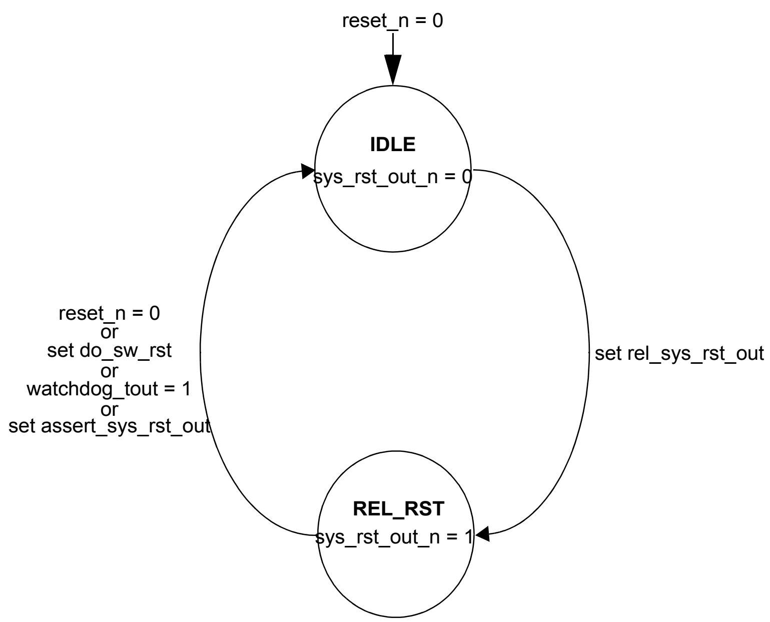


Figure 7: SYS_RST_OUT State Machine

In the normal sequence of events, the MIPS CPU will set rel_sys_rst_out once the MIPS CPU is out of reset, as shown in Figure 4. The sys_rst_out_n will be asserted again if:

- The assert_sys_rst_out bit of the RST_CTL register is set.
- A software reset is issued by setting the do_sw_rst bit of the RST_CTL register.
- A timeout is issued by the watchdog timer in the MIPS CPU.
- An external reset_in_n is asserted.

In the case of a software reset or watchdog timeout, a full system reboot sequence is conducted and sys_rst_out_n will remain asserted until released by the MIPS CPU.

5.3.3 Register Descriptions

The base address of the Reset module in the PNX8526 is 0x06 0000.

5.3.3.1 Register Address Map

Table 17: Reset Module Register Summary

Offset	Name	Description
0x06 0000	RST_CTL	Controls do/release of all resets, swap endianess
0x06 0004	RST_CAUSE	Read-only status of resets cause
0x06 0008	EN_WATCHDOG_RST	Enable watchdog_tout reset

Table 17: Reset Module Register Summary *…Continued*

Offset	Name	Description
0x06 000C-0FF0	Reserved	
0x06 0FF4	POWERDOWN	Powerdown mode
0x06 0FFC	Module ID	Module Identification and revision information

RESET REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x06 0000			***RST_CTL***	
31:6	R/W	-	Unused	
5	R/W	0	sys_big_end	0 = Little-Endian 1 = Big-Endian
4	W	-	Unused	
3	W	NI	rel_mips_rst_n	0 = No action 1 = Release MIPS Reset.
2	W	NI	do_sw_rst	0 = No action 1 = Do Software Reset.
1	W	NI	rel_sys_rst_out	0 = No action 1 = Release System Reset of External Peripherals.
0	W	NI	assert_sys_rst_out	0 = No action 1 = Do System Reset of External Peripherals.
Offset 0x06 0004			***RST_CAUSE***	
Note: This register is set on every write to RST_CTL register or watchdog timeout or reset_in_n.				
31:2		-	Unused	
1:0	R	NI	RST_CAUSE	Reset Cause register: 00 = Cause is external system reset, reset_in_n. 01 = Cause is software system reset. 11 = Cause is watchdog timeout.
Offset 0x06 0008			***EN_WATCHDOG_RST***	
31:1		-	Unused	
0	R/W	1	EN_WATCHDOG_RST	Enable Watchdog Reset register: 0 = Disable reset due to watchdog timeout. 1 = Enable reset upon watchdog timeout.
Offset 0x06 000C-0FF0			***Reserved***	
Offset 0x06 0FF4			***POWERDOWN***	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).

RESET REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
30:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x06 0FFC			*MODULE_ID*	
31:16	R	0x0123	Module_ID	Reset Module ID: 0x0123
15:12	R	0	rev_major	Major revision
11:8	R	0	rev_minor	Minor revision
7:0	R	0	app_size	Aperture size is 0 = 4 kB.

5.4 Power Management

The PNX8526 central Clock module creates the clocks for all device modules. A PI-only device in the PNX8526 typically has two clocks: the PI clock and the module clock. A DVP memory DMA device typically has three clocks: PI clock, MMI clock and module clock.

5.4.1 Module Control

Power management procedures allow the shutdown of unused portions of the PNX8526 to conserve power. The following sections detail the safe powerdown and wake up procedures.

5.4.1.1 Device Powerdown

Powerdown of a device in the PNX8526 is accomplished by the following sequence:

1. A CPU (MIPS or TM) disables the device by writing to the regular device's MMIO Control (CTL) register, 'ENABLE' bit.
2. The CPU waits for the device to acknowledge that it has successfully finished any pending transactions. Most devices are “instant down,” but some special devices may need a device-specific wait until the STATUS register indicates they have achieved a coherent state suitable for powerdown.

   Any pending device interrupts should be handled at this point by the CPU, and the device should not generate new interrupts or bus transactions when disabled. At this point, the module registers are still fully functional. Any device register can be read/written, and the device can be re-enabled if desired.
3. The CPU then writes to the device's POWERDOWN control bit. This bit does not gate the internal module clock or other clocks.

   At this point, except for the register that contains the POWERDOWN bit, none of the device's registers are accessible.

   Reads from any other register will return a known value (0xdeadabba) to indicate that the real content isn't available. Writes to any register, except the POWERDOWN bit register, will result in instant PI-Bus error ACK.
4. The CPU then performs MMIO transactions to the central Clock module to stop the module clock in a controlled way (i.e. with no glitches/illegal periods).

The register with the POWERDOWN bit is (still) the only accessible register, and the block is fully powered down. The PI and MMI clocks are still running.

Remark: The PI and MMI clock frequencies may be lowered in certain powerdown modes.

5.4.1.2 Device Wakeup

Wakeup of a device is under CPU control. The sequence is the reverse of powerdown:

1. Start the module clock
2. Disable the POWERDOWN bit
3. Setup the device and then enable it.

It is important to note that the PI-Bus must operate at full speed (at least 60MHz) when accessing peripheral registers. Exceptions to this are accesses to the clock module and interrupt module registers. These registers are guaranteed to work for any combination of bus clock and module clock frequency.

5.4.2 CPU Idle Power

Aside from the system power modes, the internal MIPS and the TM32 CPU core have provisions to reduce (but not shut down) power whenever the idle task is dispatched—i.e., whenever there is no CPU work to be done. Refer to the PR3940 MIPS RISC Core Data Book and the TM32 Media Core Processor Data Book for details.

Essentially, the MIPS enters powerdown by a special “write to coprocessor 0,” and wakes up whenever an interrupt occurs. The following powerdown modes are supported by the PR3940:

- Sleep: Not very low power since snooping is supported. Do not use this mode.
- Coma: In this mode, clocks to ALL internal PR3940 units, except for the interrupt, are off. The PR3940 will wake up from Coma when an interrupt is asserted.

The PR3940 clock should never be disabled in the Clock module. This would remove the possibility of wakeup from a hardware interrupt.

The TM32 CPU core enters powerdown by performing a “store” to a specific MMIO address (the POWERDOWN register). It wakes up when an interrupt occurs.

The TM32 CPU core also has an externally initiated full power shutdown mode. When this is requested (by writing to the global register file), the TM32 CPU core finishes any pending transactions and does a controlled shutdown of PLL and clocks, reducing power to zero. This powerdown state can be initiated for example, by the MIPS.

The bits in the global register file will be:

- tm32_pwrdwn_req: To request TM32 CPU core powerdown. This bit will reset to 0 (no pwrdwn request). The bit is R/W.

- tm32_pwrdwn_ack: Acknowledge powerdown from TM32 CPU core. This signal will be asserted when TM32 CPU core has entered powerdown mode. This bit is Read only.

TM32 CPU core will only exit this mode when tm32_pwrdwn_req is de-asserted. This means that power up will be controlled from the MIPS and this will therefore be slow. This could cause real-time interrupt events to be missed.

5.4.3 PNX8526 System Power Modes

The PNX8526 has a number of system powerdown modes. A mode determines what is on and what is off. The transition between power modes is performed by software on an internal or external host CPU.

Bus clocks (clk_fpi, clk_mpi, clk_tpi, clk_mem) will never be stopped. They can optionally be generated from a divided version of the XTAL clock (XTAL/16) thus eliminating the need for having any PLLs active. When changing the speed of the bus clocks the PI clocks should be lowered before clk_mem. This will require the F-PIMI to be set to asynchronous mode.

The frequency can be controlled in the Clock module. The bus clock frequencies can be programmed to be very low. This also means that the bridges (PIMIs, PIC, etc.) will always be clocked.

In each power mode, certain parts of the system are shut down and certain clocks may have been lowered.

Examples of power modes are given below.

Reduced Power Mode

- All video and audio associated devices are off.
- TM32 CPU core is fully shut down.
- PI-Bus, MIPS and DVP memory highways are running at normal speed.

Minimal Power Mode

- All devices are off (except devices capable of detecting wakeup events).
- TM32 CPU core is fully shutdown.
- MIPS CPU is in Coma mode.
- PI-Buses are running at reduced speed.
- DVP memory highway is running at reduced speed.
- SDRAMs are in power standby mode.

In order to enter the minimal power mode, the following sequence is required:

1. Shut down all DMA peripherals.
2. Set MMI refresh rate to match the new desired clock frequency of clk_mem. This is done by writing to global register file 1.

3. Set ALL PIMIs to asynchronous mode. This allows the clk_pi and the clk_mem to be asynchronous. This is done by writing to global register file 2.
4. Change clk_fpi, clk_tpi and clk_mpi (Done by one write to clock module.)
5. Disable the SDRAM interface.
6. Change clk_mem.

Remark: TM32 core must be in powerdown mode when clk_mem is changed.

7. Enable the SDRAM interface. This must be done to ensure no SDRAM access is in progress while changing clk-mem.
8. Set SDRAM to self-refresh mode and the MIPS CPU to Coma mode. This can be done in two different ways:
   - Via Internal CPU
     The MIPS CPU enters Coma mode. This causes the MIPS C0_COMA output to be asserted which ensures that mm_enable is de-asserted to the MMI. This causes the MMI to enter self-refresh mode.
   - Via External CPU
     The external CPU writes to the mm_self_refresh register in the global register file 2. This causes mm_enable to be de-asserted and the MMI enters self-refresh mode. The state of the MMI can be read by polling the mm_enable bit in the self-refresh register.

Wakeup

Wakeup is the inverse of powerdown.

1. The CPU is awakened by interrupt. The CPU enters running mode. This causes the mm_enable signal to be asserted, and the MMI will exit the self-refresh mode. If there is no internal processor wakeup, the exit from self-refresh mode must be done by writing to the mm_self_register in global register file 2.
2. The CPU accesses the interrupt vector location.
3. The bus clock frequency must be increased as follows:
   - Change clk_mem.
   - Change clk_pi.
4. Set PIMIs to desired operating mode.
5. Set MMI refresh rate to match new clk_mem frequency.
6. Enable/access peripherals.

Remark: Peripherals can only be accessed when the PI-Buses are running at full speed.

Software on both CPUs determines the nature of the wakeup event and either wakes up more devices to get into another power mode, or decides that the event can be ignored (key “4” was pressed on the IR remote, not the power key).

5.4.3.1 PCI Power Modes

Assignment to the PCI power mode bits causes a CPU interrupt. The CPU examines the value to determine the requested power mode, and (if appropriate) establishes this power mode and acknowledges success or failure to the PCI host CPU.

Chapter 6: Priority Interrupt Controller

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

6.1 Functional Description

The Priority Interrupt Controller (PIC) module is programmed to determine which peripheral interrupt request should raise an interrupt to the CPU. In the PNX8526, there are two CPUs, so there are two PICs—one for the MIPS processor, called M-PIC, and the other for the TriMedia processor, called T-PIC. M-PIC and T-PIC are identical modules but will be configured separately by software.

6.1.1 Overview

The PIC modules in the PNX8526 have the following functionality:

- **Global interrupts disable:** all interrupts to the CPU and optional power manager can be disabled.
- **Multi-CPU support:** each interrupt request line connected to a PIC can be disabled. Therefore the same line can be connected to multiple PICs.
- **Prioritization:** each interrupt request line generates a CPU or powerdown interrupt if the priority level of that interrupt is greater than the value of the PICs priority limiter.
- **Identification:** the CPU can efficiently read the identification of the highest priority interrupt that is causing the PIC to raise an interrupt.
- **Inter-processor communication:** one PIC generates an interrupt request to another PIC by an MMIO write. This enables one CPU to raise an interrupt to another CPU.
- **Set and Clearing Interrupts**: software can generate an interrupt to the CPU via an MMIO write to the PIC. This is mainly intended for testing purposes.

The overall interrupt architecture in the PNX8526 is shown in Figure 1.

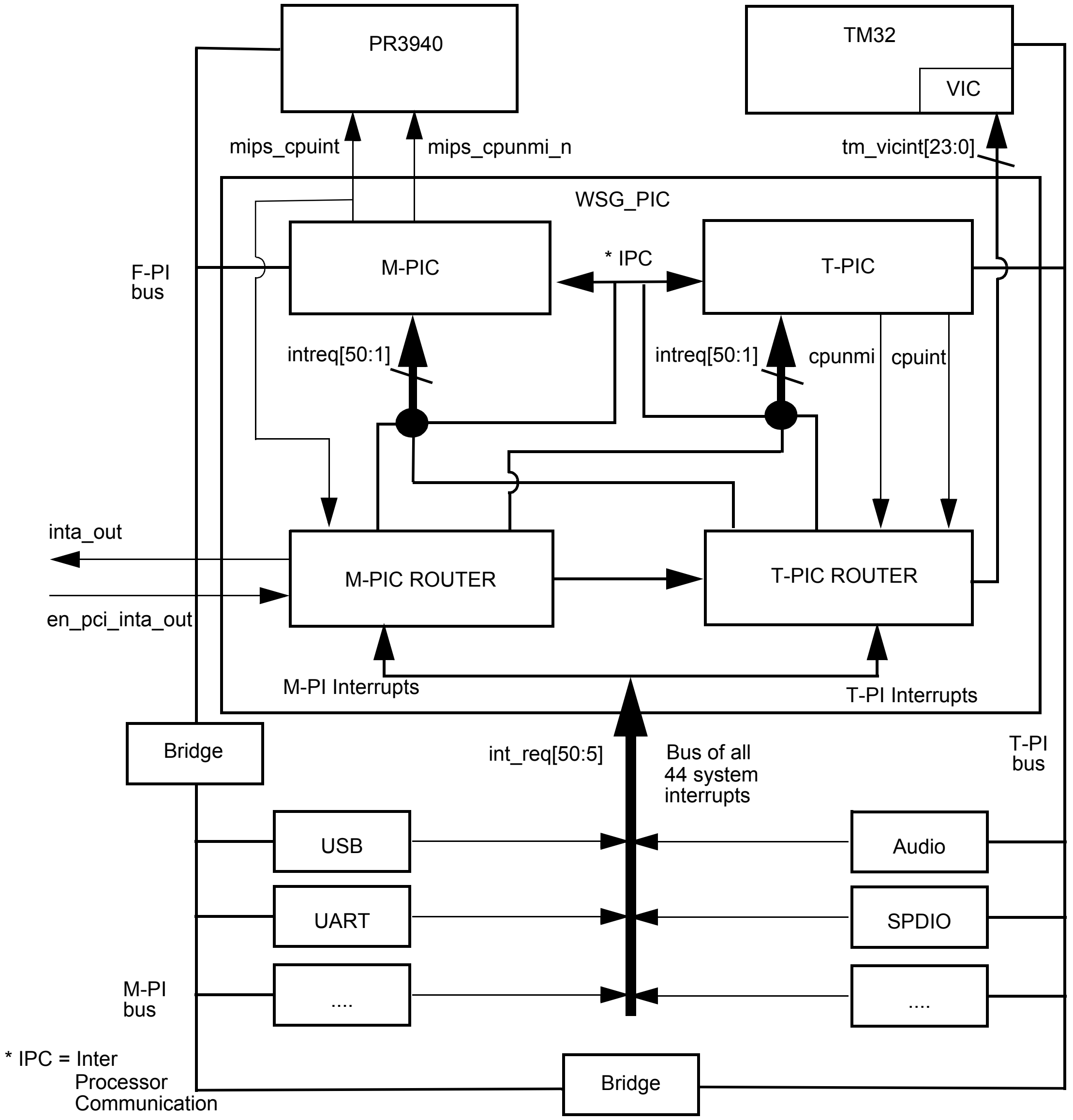


Figure 1: PNX8526 Interrupt Diagram

The M-PIC and T-PIC are identical. They both collect interrupts from devices on the F-PI, M-PI and T-PI buses and handle other tasks, like generating the PCI interrupt and tm_vicint vector for the TM32.

The PR3940 MIPS RISC processor requires a negative edge NMI. Therefore, the PICs generate a negative edge on cpunmi_n when an NMI has been detected.

Time-critical interrupts for the TM32 processor are sent straight to the TM32 processor via bits in the tm_vicint bus (see Section 6.2.2). They are also sent indirectly to the TM32 via the T-PIC's cpuint and cpunmi ports (see Section 6.1.2).

Figure 2 shows the PIC I/O diagram.

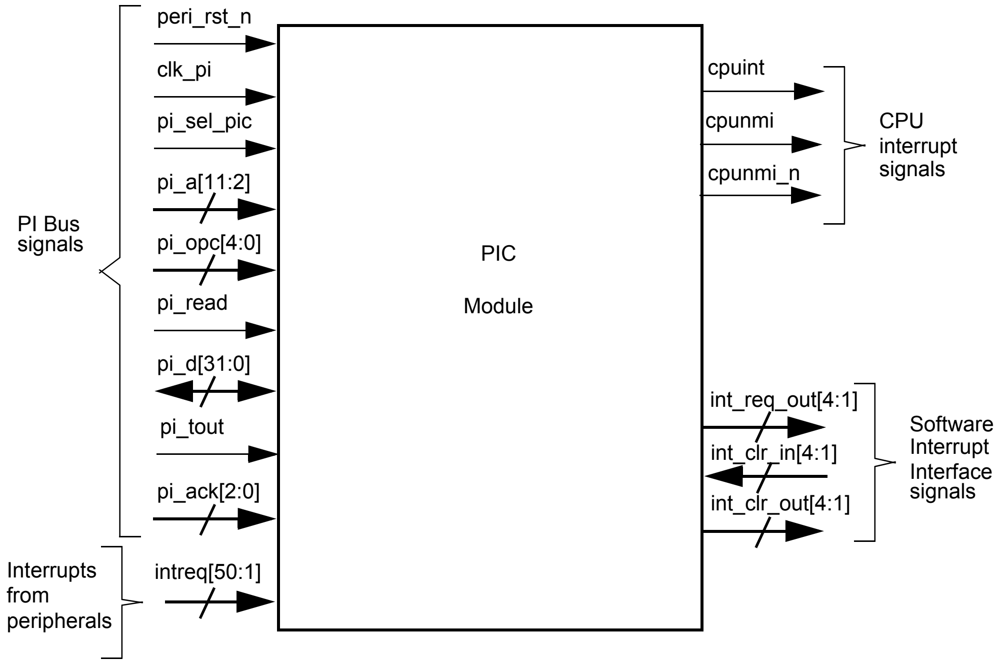


Figure 2: Priority Interrupt Controller IO Diagrams

The PIC is a PI slave. It handles 50 interrupt requests (intreq[50:1]). Its lowest four interrupt requests are assigned as software interrupts for inter-processor communication.

The PIC generates three interrupts for its CPU—cpuint, cpunmi and cpunmi_n.

6.1.2 PIC Interrupt Requests

Table 1: Table of Interrupt Requests to the PIC

Interrupt Request	Interrupt Description	Polarity	Connection to TriMedia VIC
1	Software Interrupt for Inter-Processor Communication	Positive	No
2	Software Interrupt for Inter-Processor Communication	Positive	No
3	Software Interrupt for Inter-Processor Communication	Positive	No
4	Software Interrupt for Inter-Processor Communication	Positive	No
5	Universal Serial Bus Interrupt	Positive	No
6	General Purpose IO Interrupt FIFO 0	Positive	No
7	General Purpose IO Interrupt FIFO 1	Positive	No
8	General Purpose IO Interrupt FIFO 2	Positive	No
9	General Purpose IO Interrupt FIFO 3	Positive	No
10	General Purpose IO Interrupt TSU	Positive	No
11	General Purpose IO TM-VIC Interrupt	Positive	vicint 20
12	1394 FireWire™ Interrupt	Positive	vicint 0
13	Advanced Image Composition Processor 1 Interrupt	Positive	vicint 10
14	Advanced Image Composition Processor 2 Interrupt	Positive	vicint 4
15	I^{2}C 1 Interrupt	Positive	No
16	I^{2}C 2 Interrupt	Positive	No
17	Smartcard 1 Interrupt	Positive	No
18	Smartcard 2Interrupt	Positive	No
19	UART 1 Interrupt	Positive	No
20	UART 2 Interrupt	Positive	No
21	UART 3 Interrupt	Positive	No
22	PCI interrupt	Positive	vicint 16
23	T-PI bus controller error interrupt	Positive	No
24	M-PI bus controller error interrupt	Positive	No
25	F-PI bus controller error interrupt	Positive	No
26	2D Drawing Engine Interrupt	Positive	No
27	MBS Interrupt	Positive	vicint13
28	MPEG Interrupt	Positive	vicint14
29	Video Input Processor 1	Positive	vicint 9
30	Video Input Processor 2	Positive	vicint 22
31	SPDIF Input Interrupt	Positive	vicint 24
32	SPDIF Output Interrupt	Positive	vicint 25
33	Audio Input 1 Interrupt	Positive	vicint 11
34	Audio Output 1 Interrupt	Positive	vicint 12
35	Audio Input 2 Interrupt	Positive	vicint 26
36	Audio Output 2 Interrupt	Positive	vicint 27

Table 1: Table of Interrupt Requests to the PIC *…Continued*

Interrupt Request	Interrupt Description	Polarity	Connection to TriMedia VIC
37	Audio Input & Output 3 Interrupt	Positive	vicint 23
38	Serial Sychronous Interface Interrupt	Positive	vicint 15
39	MPEG System Processor 1 MIPS Interrupt	Positive	No
40	MPEG System Processor 1 Trimedia Interrupt	Positive	vicint 19
41	MPEG System Processor 2 MIPS Interrupt	Positive	No
42	MPEG System Processor 2 Trimedia Interrupt	Positive	vicint 3
43	Transport Stream DMA interrupt	Positive	vicint 21
44	DMA interrupt	Positive	No
45	Unused		
46	TriMedia debug interrupt	Positive	vicint 18
47	PCI INTA interrupt	Negative	No
48	Clock module interrupt	Positive	No
49	MPEG System Processor 3 MIPS Interrupt	Positive	No
50	MPEG System Processor 3 Trimedia Interrupt	Positive	vicint 17
Total	50	48 Pos/ 1 Neg 1 Unused	22 vicints

6.2 Operation

6.2.1 PIC Interrupt Outputs

The priority interrupt controller receives 50 interrupt request lines (intreq_[50:1]) from interrupt-generating devices. All the interrupt requests must be glitch-free. Each of these interrupt request lines is a level-based signal that is individually prioritized to produce the following interrupt signals to the CPU.

cpuint is a level based signal that is asserted when an intreq[i] is pending and its priority is greater than the priority limiter. This PIC output is either sent to the internal processor or to an external processor via PCI. Refer to Figure 1.

cpunmi is a level-based signal that is asserted when an intreq[i] is pending and its priority is set to a maximum. This PIC output is sent to the CPU if the processor has a level-based NMI input.

cpunmi_n is an inverted version of cpunmi. This was created because the PR3940 MIPS CPU needs a negative edge on its NMI input to trigger a Non-Maskable Interrupt inside the processor.

6.2.2 TriMedia Interrupt Requests

The TriMedia VIC receives a normal cpuint and cpunmi from the T-PIC (Interrupts 1 to 2). Other than these, the TriMedia has time-critical interrupts connected directly to avoid the delay of going through the T-PIC (Interrupts 0,3,4 and 9 to 27). It also has four TM32 core internal timer interrupts (Interrupts 5 to 8) and four software interrupts for host communication, applications, debugger and RTOS (Interrupts 28 to 31).

Table 2: Table of Interrupt Request Signals to the VIC

No.	Interrupt Connected to TriMedia VIC
0	1394 FireWire Interrupt
1	PIC cpuint
2	PIC cpunmi
3	MPEG System Processor 2 TriMedia Interrupt
4	AICP 2 Interrupt
5	TM Core Internal timer
6	TM Core Internal timer
7	TM Core Internal timer
8	TM Core Internal timer
9	Video Input Processor 1
10	AICP 1
11	Audio Input 1
12	Audio Output 1
13	MBS
14	MPEG
15	SSI
16	PCI
17	MPEG System Processor 3 TriMedia
18	TM Debug
19	MPEG System Processor 1 TriMedia
20	GPIO VIC Interrupt
21	TSDMA
22	Video Input Processor 2
23	Audio Input & Output 3
24	SPDIF Input
25	SPDIF Output
26	Audio Input 2
27	Audio Output 2
28	Software host communication
29	Software application
30	Software debugger
31	Software RTOS

6.2.3 Inter-Processor Communication

In order to implement Inter-Processor Communication (IPC) between the PR3940 MIPS CPU and the TM3200 CPU, one CPU must be able to interrupt the other. This is accomplished through software interrupts (intreq[4:1]). The CPUs use set and clear bits to control these software interrupts.

The direction of the four software interrupts can be divided between the two processors in any way—e.g., two from the PR3940 MIPS and two from the TM3200 or three from the PR3940 MIPS and one from the TM3200, etc.

The following diagram shows how the M-PIC and T-PIC are connected for efficient IPC

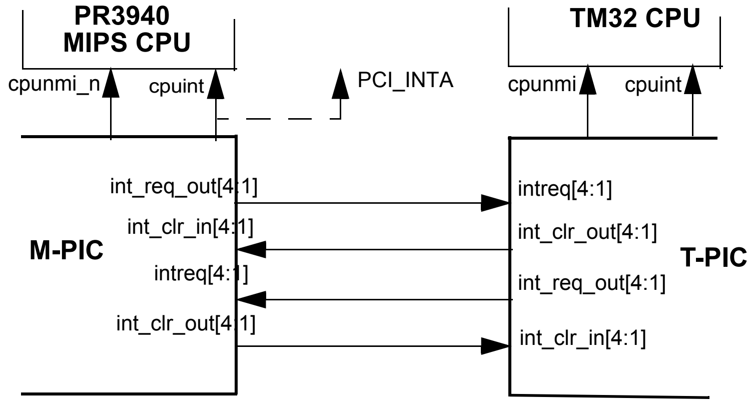


Figure 3: Interface between PICs for Inter-Processor Communication

.IPC functionality is described in the following example:

1. MIPS writes to M-PIC int_set1 register to set int_req_out1. This causes T-PIC intreq1 to go high and T-PIC cpuint will be asserted if intreq1's priority is higher than the priority limiter. (T-PIC CPU covers all normal and NMI interrupts generated by the PIC.)
2. TM3200 reads the interrupt, executes the ISR (Interrupt Service Routine), and then clears the interrupt. The clearing is done by writing to T-PIC int_clr1 register which causes int_clr_out1 to go high. The T-PIC cpuint is immediately de-asserted (done by blocking the T-PIC intreq1 inside T-PIC).
3. M-PIC int_clr_in1 goes high, and the int_set1 register inside the M-PIC is cleared. This causes M-PIC int_req_out1 to be de-asserted and thus T-PIC intreq1 is de-asserted. This releases the T-PIC int_clr_out1 and M-PIC int_clr_in1 signals.
4. Software should not write to the M-PIC's int_set1 register again until the M-PIC int_clr_in1 signal is cleared. This is required to ensure that clearing the previous interrupt has been completed. All transfers between the two PICs are asynchronous.

Figure 4 shows waveforms of the above example

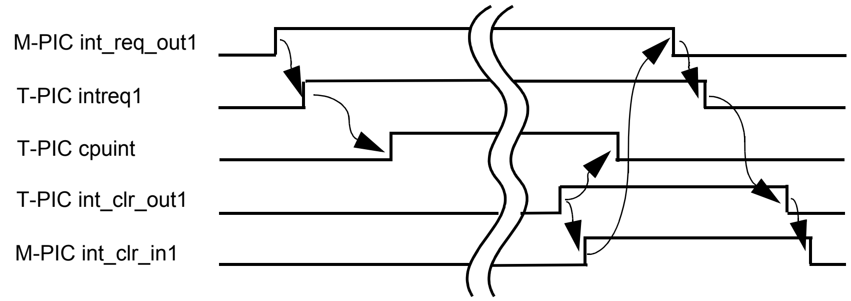


Figure 4: IPC Timing Diagram

6.2.3.1 IPC Registers

The registers PIC_INT_REG_[4:1] are reserved for Inter-Processor Communication. Each of these four registers can generate and receive software interrupts. Software will assign which of the four registers generate and which receive software interrupts.

- IPC registers assigned to generate a software interrupt: Each of these registers can be programmed to generate and clear a separate software interrupt. In this case the INT_PRIORITY for these registers should be set to zero so that no interrupt is generated to the PIC's own processor.
- IPC registers assigned to receive a software interrupt: Each of these registers can be programmed to clear and set priority on a separate software interrupt and read to detect if a software interrupt has been generated. The INT_SET bit for these registers should not be asserted while this register is assigned for receiving a software interrupt.

6.2.4 Interrupt Source Register

The PIC_INT_SRC register allows the CPU to identify the cause of the interrupt and vector to the right interrupt service routine. The CPU accomplishes this as follows:

- Create a 4096 byte-aligned Interrupt Service Routine (ISR) address table in memory.
- Program the base address of the table in the INT_TABLE_ADDR field of the PIC_INT_SRC register.
- After an interrupt has been detected, read the PIC_INT_SRC register. The INT_SRC field of this register provides the identity of the highest priority pending interrupt. The INT_TABLE_ADDR field provides the base address of the Interrupt Handlers address table.
- Read the ISR address from the table and jump to it.

An ISR address table is shown in Figure 5. Each entry in the table is 8 bytes wide: 4 bytes for the ISR address and 4 bytes reserved for the new value of the priority limiter.

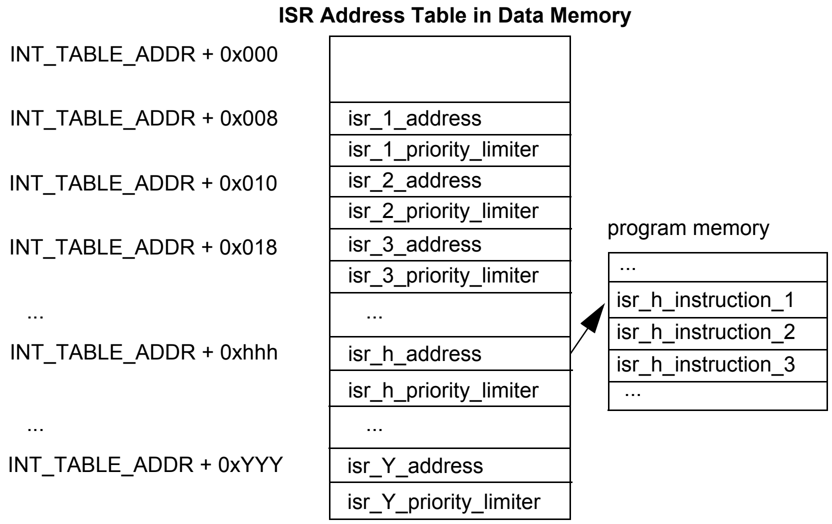


Figure 5: Vectoring Using ISR Table and the PIC_INT_SRC Register

6.2.4.1 Special Interrupt Service Cases

Change priority limiter value and then read PIC_INT_SRC register:

One PI clock cycle of delay is needed between these transactions to ensure that the CPU reads the correct PIC_INT_SRC. See Figure 6,below. Software should ensure these PI transactions are not performed back-to-back to avoid this race condition.

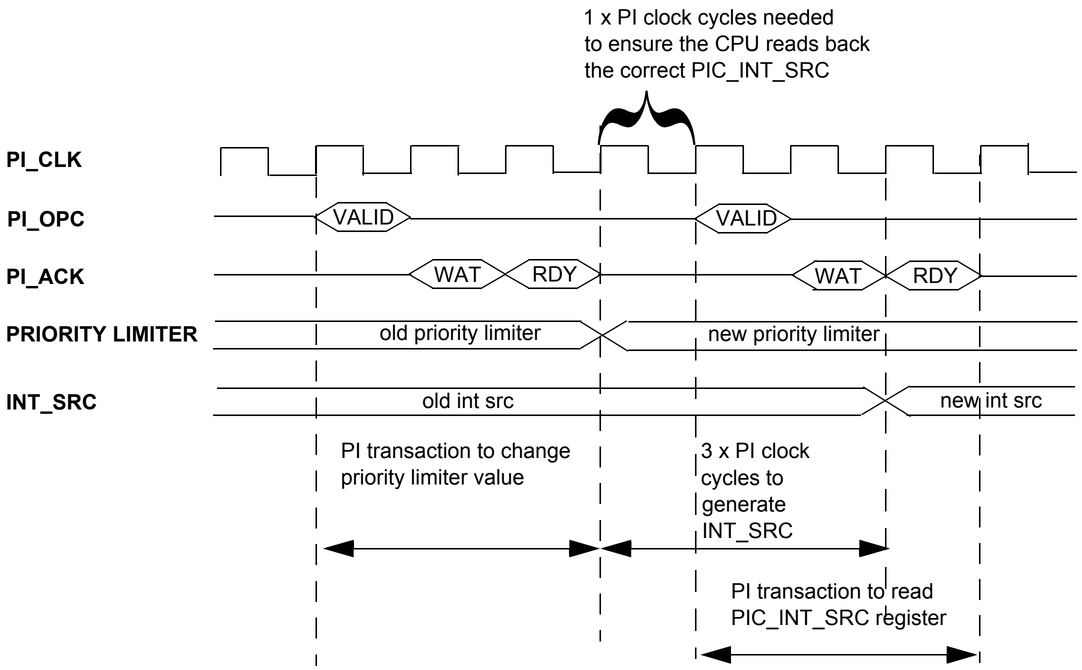


Figure 6: PI Clock Cycles Needed Between Transactions (Priority Limiter Change and Read of PIC_INT_SRC)

Interrupt was raised, but interrupt source is de-asserted:

It is possible that, by the time the CPU gets to read the PIC_INT_SRC register, there are no pending interrupts higher than the priority limiter. In this case the INT_SRC field equals zero. This can happen for various reasons. The interrupt service routine should be able to cope with the situation.

Multiple pending interrupts:

Multiple pending interrupt inputs may have the same priority level. The PIC chooses one of them as the source for interrupting the connected CPU by using the input line number as a parameter. An interrupt line connected to a lower input-line number has a higher priority than interrupts connected to higher input-line numbers (intreq[1] has a higher priority than intreq[2], and so forth).

Clear the interrupt in the module or PIC, then read the PIC_INT_SRC register:

Three PI clock cycles of delay are needed between these transactions to ensure that the CPU reads the correct PIC_INT_SRC. See Figure 7, below. Software should ensure these PI transactions are not performed back-to-back to avoid this race condition.

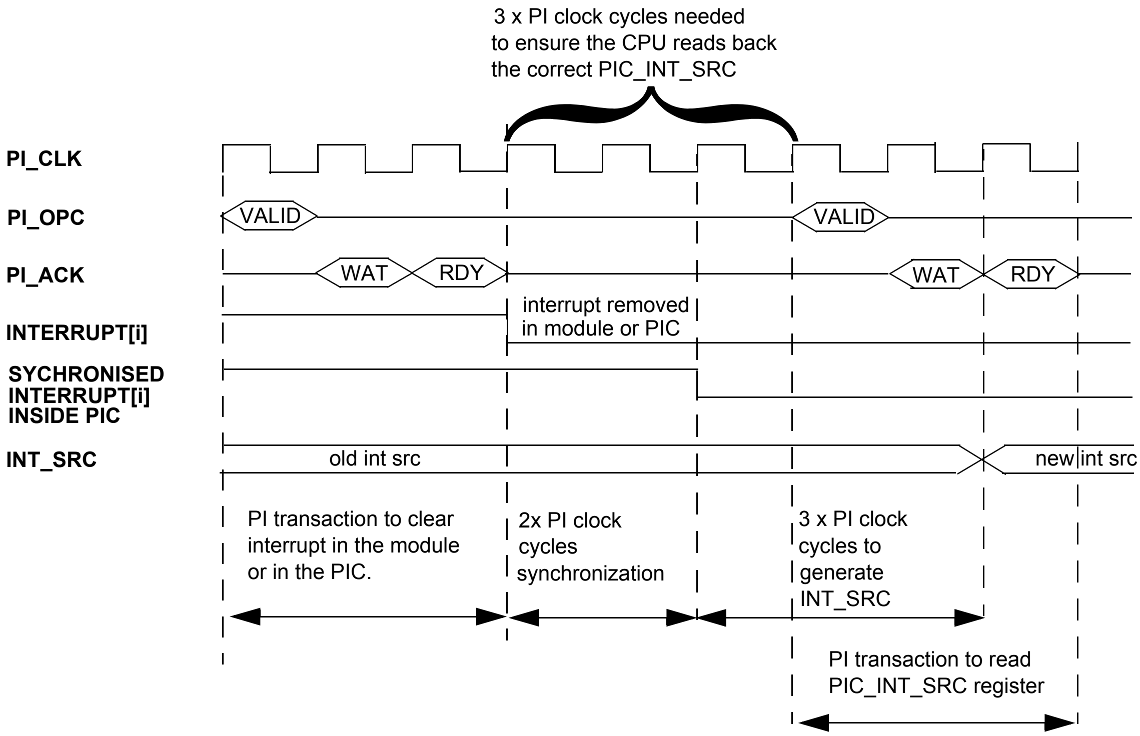


Figure 7: PI Clock Cycles Needed Between Transactions (Interrupt Clear and Read of PIC_INT_SRC)

6.2.5 Interrupt Priority

Each interrupt request line has a priority value programmed in its register. The PIC_INT_PRIORITY register has a programmable limiter value to control which interrupt request lines can assert cpuint. Pending Interrupts with priority greater than the PRIORITY_LIMITER raise an interrupt to the CPU.

The PIC supports 16 priority levels (0-15) for each interrupt. An intreq[i] can be disabled from raising an interrupt to the CPU by setting its INT_PRIORITY to zero. At reset, the INT_PRIORITY of all interrupts is set to zero, thereby disabling them.

The PRIORITY_LIMITER can be set (0-15) to control which devices may interrupt the CPU. At the highest level, 15, cpuint is disabled and only CPUNMI can be generated.

While servicing an interrupt, the priority limiter can be used to prevent interrupts with lower priority from interfering with the current execution. This is to support nesting of interrupts.

6.2.6 Device Interrupt Registers

The PNX8526 has 50 interrupt request lines, and the PIC has 50 matching interrupt registers. These registers serve several purposes:

- Provide an interrupt pending bit that indicates if intreq[i] is pending.
- Allow software to set and clear the interrupt pending bit.
- Allow software to set the priority for intreq[i].

Remark: There is no way to tell if INT_PENDING was set by intreq[i] or by software using INT_SET. Therefore it is up to software to track the use of INT_SET.

6.2.6.1 Special Cases

Writing INT_SET and INT_CLR to PIC_INT_REG_i

INT_SET	INT_CLR	Action
0	0	Update INT_PRIORITY
0	1	Clear INT_PENDING
1	0	Set INT_PENDING
1	1	Invalid, no operation performed

Clear the interrupt in the module or the PIC, then read the PIC_INT_REG_i:

One PI clock cycle of delay is needed between these transactions to ensure the CPU reads the correct value in the INT_PENDING field in the PIC_INT_REG_i register. Software should ensure these PI transactions are not performed back-to-back to avoid this race condition.

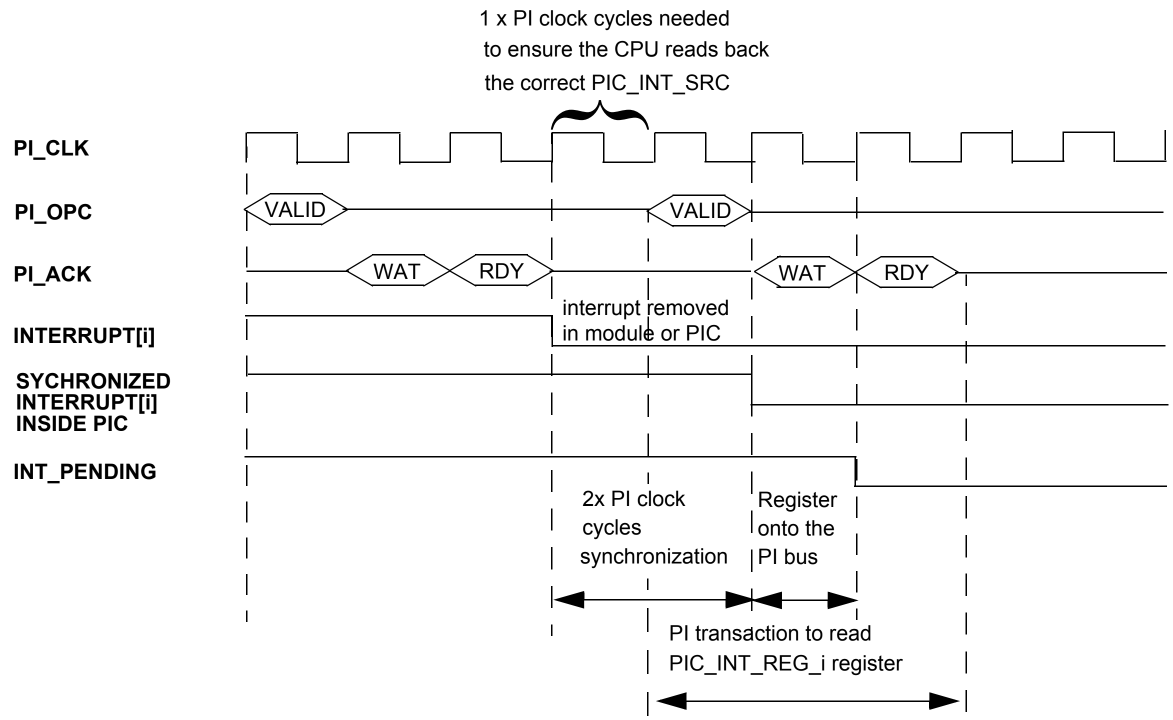


Figure 8: PI Clock Cycles Needed Between Transactions (Interrupt Clear and Read of PIC_INT_REG_i)

6.2.7 PCI Interrupt Enable Register

The PCI_INTA is the PNX8526 PCI interrupt pin. It can be enabled as an output pin and connected to pci_inta_out, or it can be enabled as an input pin and connected to one of the PIC's intreq lines. This control is done via the Global 2 register Enable_IntA_O
(see Section 6.3.1 on page 6-154).

6.3 Register Descriptions

The PNX8526 has two PIC modules: M-PIC services the MIPS PR3940 and T-PIC services the TriMedia TM32 processor. The register base address for M-PIC is 0x03 E000. The register base address for T-PIC is 0x10 2000.Register Address Map

Table 3: PIC Register Summary

Offset	Name	Description
M-PIC		
0x03 E000	PIC_INT_PRIORITY	PIC Interrupt Priority register
0x03 E004	PIC_INT_SRC	PIC Interrupt Source register
0x03 E008—E010	Reserved	
0x03 E014	PIC_INT_REG_1	Interrupt register 1 (reserved for software interrupt)
0x03 E018	PIC_INT_REG_2	Interrupt register 2 (reserved for software interrupt)
0x03 E01C	PIC_INT_REG_3	Interrupt register 3 (reserved for software interrupt)
0x03 E020	PIC_INT_REG_4	Interrupt register 4 (reserved for software interrupt)
0x03 E024	PIC_INT_REG_5	USB Interrupt
0x03 E028	PIC_INT_REG_6	General Purpose IO Interrupt FIFO 0
0x03 E02C	PIC_INT_REG_7	General Purpose IO Interrupt FIFO 1
0x03 E030	PIC_INT_REG_8	General Purpose IO Interrupt FIFO 2
0x03 E034	PIC_INT_REG_9	General Purpose IO Interrupt FIFO 3
0x03 E038	PIC_INT_REG_10	General Purpose IO Interrupt TSU
0x03 E03C	PIC_INT_REG_11	General Purpose IO TM-VIC Interrupt
0x03 E040	PIC_INT_REG_12	1394 FireWire Interrupt
0x03 E044	PIC_INT_REG_13	AICP 1 Interrupt
0x03 E048	PIC_INT_REG_14	AICP 2 Interrupt
0x03 E04C	PIC_INT_REG_15	I^{2}C 1 Interrupt
0x03 E050	PIC_INT_REG_16	I^{2}C 2 Interrupt
0x03 E054	PIC_INT_REG_17	Smartcard 1 Interrupt
0x03 E058	PIC_INT_REG_18	Smartcard 2 Interrupt
0x03 E05C	PIC_INT_REG_19	UART 1 Interrupt
0x03 E060	PIC_INT_REG_20	UART 2 Interrupt
0x03 E064	PIC_INT_REG_21	UART 3 Interrupt
0x03 E068	PIC_INT_REG_22	PCI Interrupt
0x03 E06C	PIC_INT_REG_23	T-PI bus controller error Interrupt
0x03 E070	PIC_INT_REG_24	M-PI bus controller error Interrupt
0x03 E074	PIC_INT_REG_25	F-PI bus controller error Interrupt
0x03 E078	PIC_INT_REG_26	2D Drawing Engine Interrupt
0x03 E07C	PIC_INT_REG_27	MBS Interrupt
0x03 E080	PIC_INT_REG_28	MPEG Interrupt

Table 3: PIC Register Summary *…Continued*

Offset	Name	Description
0x03 E084	PIC_INT_REG_29	VIP 1 Interrupt
0x03 E088	PIC_INT_REG_30	VIP 2 Interrupt
0x03 E08C	PIC_INT_REG_31	SPDIF Input Interrupt
0x03 E090	PIC_INT_REG_32	SPDIF Output Interrupt
0x03 E094	PIC_INT_REG_33	Audio Input 1 Interrupt
0x03 E098	PIC_INT_REG_34	Audio Output 1 Interrupt
0x03 E09C	PIC_INT_REG_35	Audio Input 2 Interrupt
0x03 E0A0	PIC_INT_REG_36	Audio Output 2 Interrupt
0x03 E0A4	PIC_INT_REG_37	Audio Input & Output 3 Interrupt
0x03 E0A8	PIC_INT_REG_38	SSI Interrupt
0x03 E0AC	PIC_INT_REG_39	MSP 1 MIPS Interrupt
0x03 E0B0	PIC_INT_REG_40	MSP 1 TriMedia Interrupt
0x03 E0B4	PIC_INT_REG_41	MSP 2 MIPS Interrupt
0x03 E0B8	PIC_INT_REG_42	MSP 2 TriMedia Interrupt
0x03 E0BC	PIC_INT_REG_43	Transport Stream DMA Interrupt
0x03 E0C0	PIC_INT_REG_44	DMA Interrupt
0x03 E0C4	PIC_INT_REG_45	Unused
0x03 E0C8	PIC_INT_REG_46	TriMedia debug interrupt
0x03 E0CC	PIC_INT_REG_47	PCI INTA Interrupt
0x03 E0D0	PIC_INT_REG_48	Clocks module interrupt
0x03 E0D4	PIC_INT_REG_49	MSP 3 MIPS Interrupt
0x03 E0D8	PIC_INT_REG_50	MSP 3 TriMedia Interrupt
0x03 E0DC—EFF0	Reserved	
0x03 EFF4	Powerdown	Powerdown mode
0x03 EFF8	Reserved	
0x03 EFFC	Module ID	Module Identification and revision information
T-PIC		
The T-PIC registers begin at 0x10 2000. They are identical to the M-PIC registers, described above.		

Table 4: PIC Global 2 Register Summary

Address	Name	Description
0x04 D050	PCI Inta Output Enable	Global 2 register

M-PIC REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 E000			*PIC_INT_PRIORITY*	
31:4		-	Unused	Read as zero
3:0	R/W	0xF	PRIORITY_LIMITER	Interrupt Priority Limiter: Pending Interrupts with priority greater than the PRIORITY_ LIMITER raise an interrupt to the CPU. Special case: PRIORITY_LIMITER = 15 No interrupts can be raised to the CPU via the CPUINT, only CPUNMI can be generated.
Offset 0x03 E004			*PIC_INT_SRC*	
31:12	R/W	0	INT_TABLE_ADDR	Base address for Interrupt Handlers address table. Interrupt Handler routines address table is 4096 byte-aligned in memory.
11:3	R	0	INT_SRC	Interrupt Source identifies the highest priority pending interrupt at the time this register is read. Allows room for 511 interrupt sources. Value 0 indicates no pending Interrupt.
2:0		-	Unused	Read as zero.
Offset 0x03 E008-E010			*Reserved*	
Offset 0x03 E014			*PIC_INT_REG_1*	
31	R	NI	INT_PENDING	Interrupt Pending Reads to this register indicate whether the interrupt is pending for service regardless of the INT_PRIORITY value and the priority limiter. 0 = intreq[i] is not asserted and INT_SET bit is not set. 1 = intreq[i] is asserted or INT_SET bit is set.
30	W	0	INT_SET	Interrupt Set. Writes to this register have the following effect: 0 = No effect 1 = Makes INT_PENDING =1 and INT_PRIORITY is not updated. Reads return zero. Reset: INT_SET is inactive after reset.
29	W	0	INT_CLR	Interrupt Clear. Writes to this register have the following effect: 0 = No effect 1 = Makes INT_PENDING = 0 if intreq[i] is not asserted and INT_PRIORITY is not updated. Reads return zero. Reset: INT_CLR is inactive after reset.
28:8		-	Unused	Read as zero.
7:4	R	0	Reserved	May be used for future increase in the number of priority levels.
3:0	R/W	0	INT_PRIORITY	The INT_PRIORITY field determines the priority level of the intreq[i] line. This field is only written if INT_SET and INT_CLR are 0.
Offset 0x03 E018			*PIC_INT_REG_2*	
Interrupt register 2. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E01C			*PIC_INT_REG_3*	
Interrupt register 3. This register is identical to PIC_INT_REG_1 (0x03 E014).				

M-PIC REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 E020			*PIC_INT_REG_4*	
Interrupt register 4. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E024			*PIC_INT_REG_5*	
USB Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E028			*PIC_INT_REG_6*	
General Purpose IO Interrupt FIFO 0. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E02C			*PIC_INT_REG_7*	
General Purpose IO Interrupt FIFO 1. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E030			*PIC_INT_REG_8*	
General Purpose IO Interrupt FIFO 2. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E034			*PIC_INT_REG_9*	
General Purpose IO Interrupt FIFO 3. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E038			*PIC_INT_REG_10*	
General Purpose IO Interrupt TSU. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E03C			*PIC_INT_REG_11*	
General Purpose IO TM-VIC Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E040			*PIC_INT_REG_12*	
1394 FireWire Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E044			*PIC_INT_REG_13*	
AICP 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E048			*PIC_INT_REG_14*	
AICP 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E04C			*PIC_INT_REG_15*	
I^{2}C 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E050			*PIC_INT_REG_16*	
I^{2}C 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E054			*PIC_INT_REG_17*	
Smartcard 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E058			*PIC_INT_REG_18*	
Smartcard 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E05C			*PIC_INT_REG_19*	
UART 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E060			*PIC_INT_REG_20*	
UART 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E064			*PIC_INT_REG_21*	
UART 3 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E068			*PIC_INT_REG_22*	

M-PIC REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
PCI Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E06C			*PIC_INT_REG_23*	
T-PI bus controller error Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E070			*PIC_INT_REG_24*	
M-PI bus controller error Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E074			*PIC_INT_REG_25*	
F-PI bus controller error Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E078			*PIC_INT_REG_26*	
2D Drawing Engine Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E07C			*PIC_INT_REG_27*	
MBS Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E080			*PIC_INT_REG_28*	
MPEG Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E084			*PIC_INT_REG_29*	
VIP 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E088			*PIC_INT_REG_30*	
VIP 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E08C			*PIC_INT_REG_31*	
SPDIF Input Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E090			*PIC_INT_REG_32*	
SPDIF Output Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E094			*PIC_INT_REG_33*	
Audio Input 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E098			*PIC_INT_REG_34*	
Audio Output 1 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E09C			*PIC_INT_REG_35*	
Audio Input 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0A0			*PIC_INT_REG_36*	
Audio Output 2 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0A4			*PIC_INT_REG_37*	
Audio Input & Output 3 Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0A8			*PIC_INT_REG_38*	
SSI Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0AC			*PIC_INT_REG_39*	
MSP 1 MIPS Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0B0			*PIC_INT_REG_40*	
MSP 1 TriMedia Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				

M-PIC REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 E0B4			***PIC_INT_REG_41***	
MSP 2 MIPS Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0B8			***PIC_INT_REG_42***	
MSP 2 TriMedia Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0BC			***PIC_INT_REG_43***	
Transport Stream DMA Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0C0			***PIC_INT_REG_44***	
DMA Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0C4			***PIC_INT_REG_45***	
Unused.				
Offset 0x03 E0C8			***PIC_INT_REG_46***	
TriMedia debug interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0CC			***PIC_INT_REG_47***	
PCI INTA Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0D0			***PIC_INT_REG_48***	
Clocks module interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0D4			***PIC_INT_REG_49***	
MSP 3 MIPS Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0D8			***PIC_INT_REG_50***	
MSP 3 TriMedia Interrupt. This register is identical to PIC_INT_REG_1 (0x03 E014).				
Offset 0x03 E0DC—EFF0 Reserved				
Offset 0x03 EFF4			***PIC_POWERDOWN***	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x03 EFF8			***Reserved***	
Offset 0x03 EFFC			***PIC_MOD_ ID***	
31:16	R	0x011D	MODULE ID	The PIC module ID is 0x011D.
15:12	R	1	MAJREV	Major Revision
11:8	R	0	MINREV	Minor Revision
7:0	R	0	MODULE APERTURE SIZE	Aperture size = 4 kB*(bit_value+1)

6.3.1 Global 2 Registers

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
PCI Inta Output Enable Register				
Offset 0x04 D050			*ENABLE_INTA_O*	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x0	ENABLE_INTA_O	Enables PCI INTA output. 0 = Disable PCI inta output. 1 = Enable PCI inta output. Reset is to be disabled.

Chapter 7: Pixel Formats

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

7.1 Introduction

Several hardware subsystems in the PNX8526 deal directly with images. Such hardware subsystems must follow the same memory representation and interpretation of images in order to successfully pass images between subsystems.

Software modules also constitute subsystems that can produce or consume images. Although most modules are designed with an abstraction layer from the underlying format, some legacy modules may exist that assume a particular pixel representation in memory.

In order to run a wide variety of software modules on the PNX8526, it would be ideal if the hardware subsystems all supported programmable pixel formats. This, however, is not feasible. Therefore, the PNX8526 uses the following pixel format strategy:

- A limited number of native pixel formats are supported by all image subsystems, as appropriate.
- The Memory Based Scaler supports conversion from arbitrary pixel formats to any native format during the anti-flicker filtering operation. (This operation is usually required on graphics images anyway, so no extra passes are introduced.)
- Hardware subsystems support all native pixel formats in both little-endian and big-endian system operation.
- Software always sees the same component layout for a native pixel format unit, whether it is running in little-endian or big-endian mode— i.e., for a given native format, RGB (or YUV) and alpha are always in the same place.
- Software dealing with multiple units at a time in Single Instruction Multiple Data (SIMD) style must be aware of system endian mode.
- The native formats of the PNX8526 include the most common indexed, packed RGB, packed YUV and planar YUV formats used by Microsoft® DirectX and Apple® QuickTime, with 100% bit layout compatibility in both little and big-endian modes.

7.2 Summary of Native Pixel Formats

Table 1 and Figure 1 summarize the native pixel formats and the image hardware subsystems that support them.

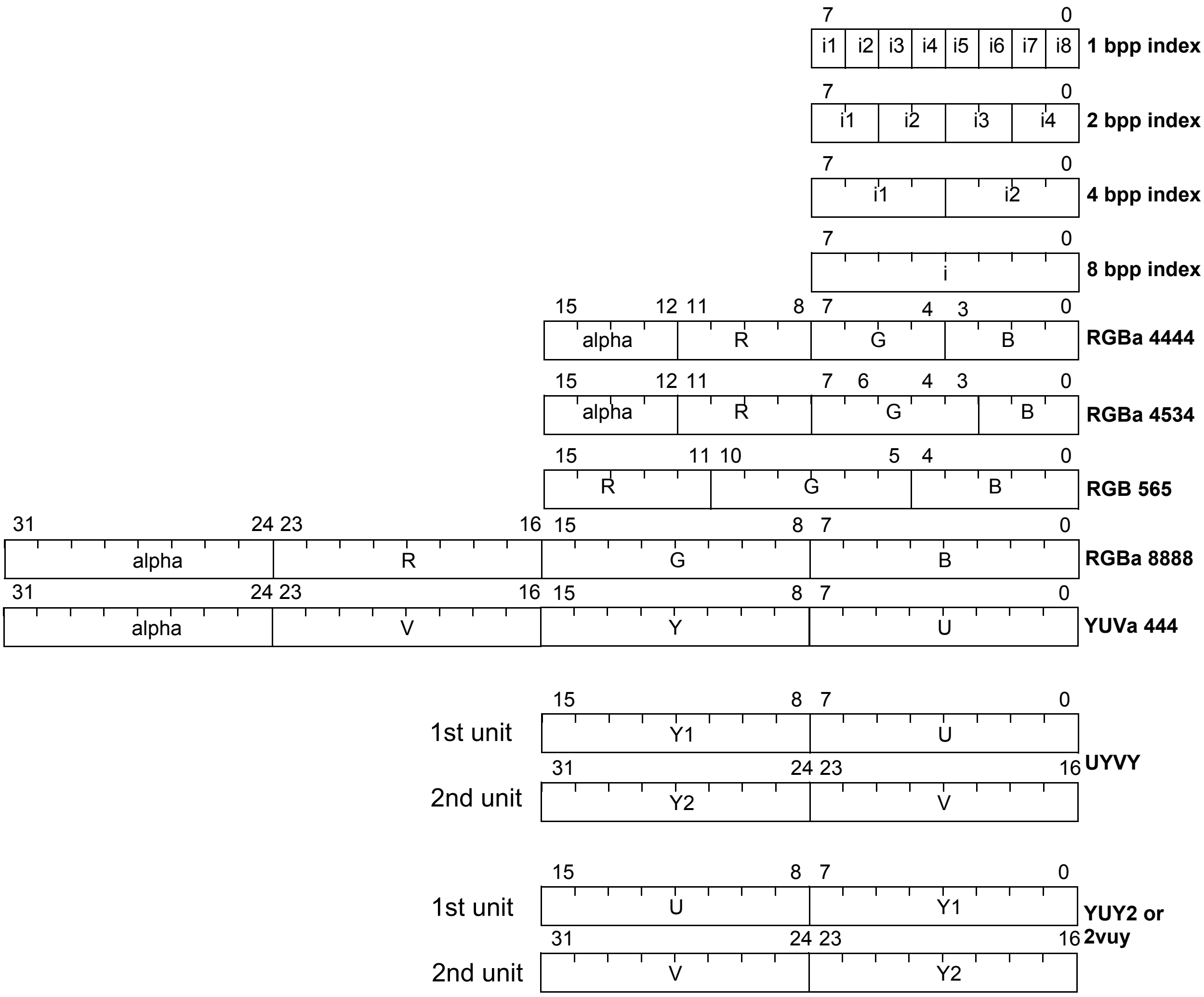
Table 1: Native Pixel Format Summary

Name	Note	VIP Out	MPG Out	MBS In	MBS Out	2D Draw Engine (2)	AICP In
1 bpp indexed	CLUT entry = 24 bit color + 8 bit alpha			x			x
2 bpp indexed				x			x
4 bpp indexed				x			x
8 bpp indexed				x		x	x
RGBa 4444	16 bit unit, containing 1 pixel with alpha	(1)		x	x	x	x
RGBa 4534		(1)		x	x	x	x
RGB 565	16 bit unit, containing 1 pixel, no alpha	(1)		x	x	x	x
RGBa 8888	32 bit unit, containing 1 pixel with alpha	(1)		x	x	x	x
packed YUVa 4:4:4	32 bit unit containing 1 pixel with alpha	x		x	x	x	x
packed YUV 4:2:2 (UYVY)	16 bit unit, 2 successive units contain 2 horizontally adjacent pixels, no alpha	x		x	x		x
packed YUV 4:2:2 (YUY2, 2vuy)		x		x	x		x
planar YUV 4:2:2	3 arrays, 1 for each component	x		x	x		
semi-planar YUV 4:2:2	2 arrays, 1 with all Ys, 1 with U and Vs	x		x	x		
planar YUV 4:2:0	3 arrays, 1 for each component			x	x		
semi-planar YUV 4:2:0	2 arrays, 1 with all Ys, 1 with U and Vs		x	x	x		

(1) The VIP is capable of producing RGB formats, but not when performing horizontal scaling.

(2) Shown are the 2D Drawing Engine frame buffer formats where drawing, RasterOps and alpha-blending of surfaces can be accelerated. The 2D Drawing Engine host port also supports 1 bpp monochrome font/pattern data, and 4 and 8-bit alpha only data for host initiated anti-aliased drawings.

The layout shown in Figure 1 is the way that a unit ends up in a CPU register given a unit size (8, 16 or 32 bits) load operation, regardless of the PNX8526 endian mode of operation.



(For planar YUV 4:2:0 formats, refer to Figure 9 and Figure 10)

Figure 1: Native Pixel Format Unit Layout

7.3 Native Pixel Format Representation

7.3.1 Indexed Formats

The indexed formats support a 1, 2, 4 or 8-bit pixel format. For each of the respective 2, 4, 16 or 256 code values, a full look-up for a 24-bit color and 8-bit alpha is performed, using a subsystem-specific, programmable color look-up table.

Figure 2 shows the “software view” of the four indexed formats. Packing of pixels within the byte always uses the “first, left-most pixel in most significant bit(s)” packing convention. Pixel groups from left to right have increasing memory byte addresses.

Indexed formats are accepted by the AICP and MBS. The 2D Drawing Engine supports 8 bpp indexed as a frame buffer format, but supports no other indexed variants.

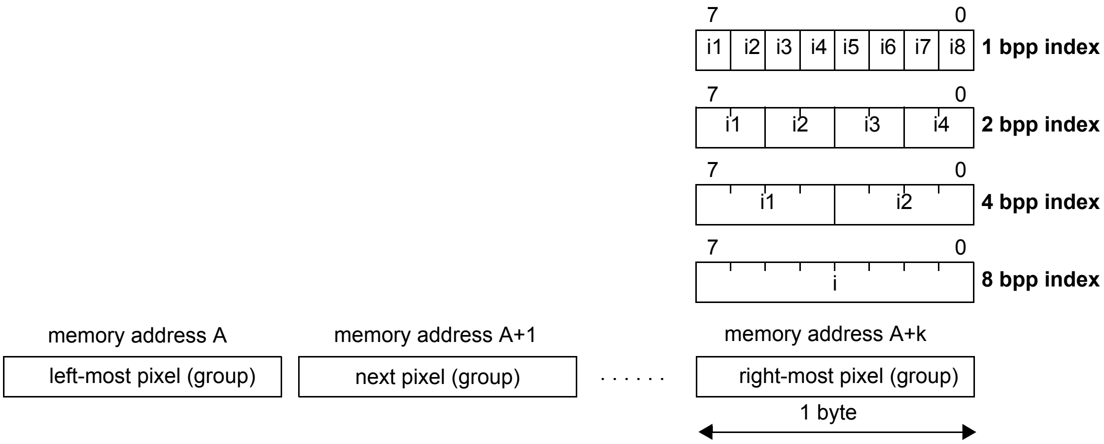


Figure 2: Indexed Formats

7.3.2 16-Bit/Pixel Packed Formats

The 16-bit native formats are RGBa 4444, RGBa 4534 and RGB 565.

Figure 3 shows the “software view” of these formats. The CPU register layout is always the same when performing 16-bit load/store instructions, regardless of system endian mode. Adjacent pixels have left-to-right increasing memory addresses.

16-bit formats are accepted and produced by the AICP, VIP, MBS and the 2D Drawing Engine

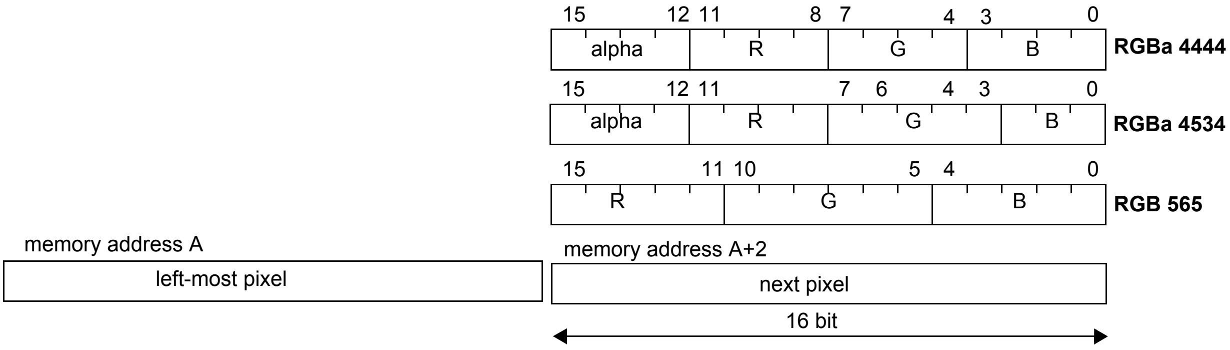


Figure 3: 16-Bit/Pixel Packed Formats

7.3.3 32-Bit/Pixel Packed Formats

The 32-bit formats include RGBa 8888 and YUVa 4:4:4 with an 8-bit per pixel alpha.

Figure 4 shows the “software view,” resulting from a 32-bit load into a CPU register. This view is independent of system endian mode. Left-to-right pixels have increasing memory addresses.

32-bit formats are accepted and produced by all units except the MPEG-2 decoder.

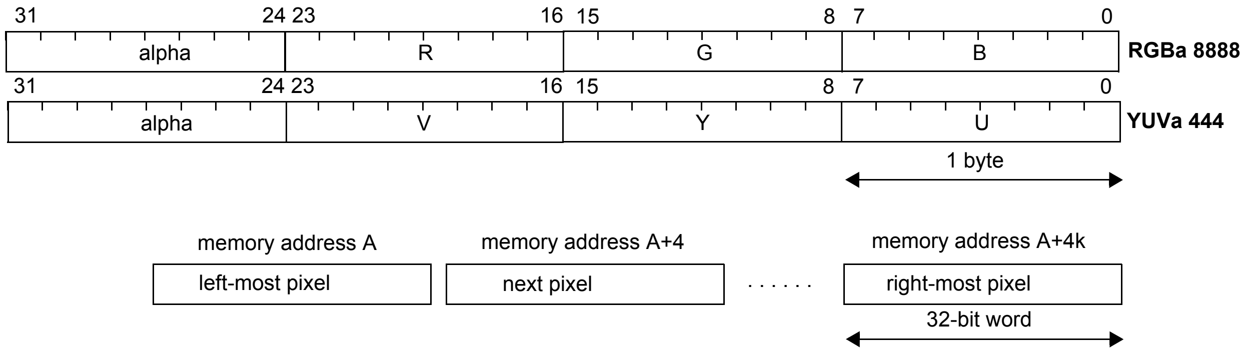


Figure 4: 32-Bit/Pixel Packed Formats

7.3.4 Packed YUV 4:2:2 Formats

Packed YUV 4:2:2 formats store two horizontally adjacent pixels into two 16-bit units. Each pixel has an individual luminance (Y1 for the left of the pair, Y2 for the right of the pair). There is a single U and V value associated with the pair. The U and V values are taken from the same spatial position as the Y1 sample (Figure 7).

There are two variants of packed YUV 4:2:2: the Microsoft ‘UYVY’ format and the Microsoft ‘YUY2’ format. The big-endian view of the YUY2 format is identical to the Power MacIntosh ‘2vuy’ format.

Figure 5 and Figure 6 show the software view of UYVY and YUY2/2vuy. Two successive 16-bit units contain a pair of pixels. This view is independent of system endian mode.

Packed YUV 4:2:2 formats are accepted and produced by the MBS, VIP and AICP.

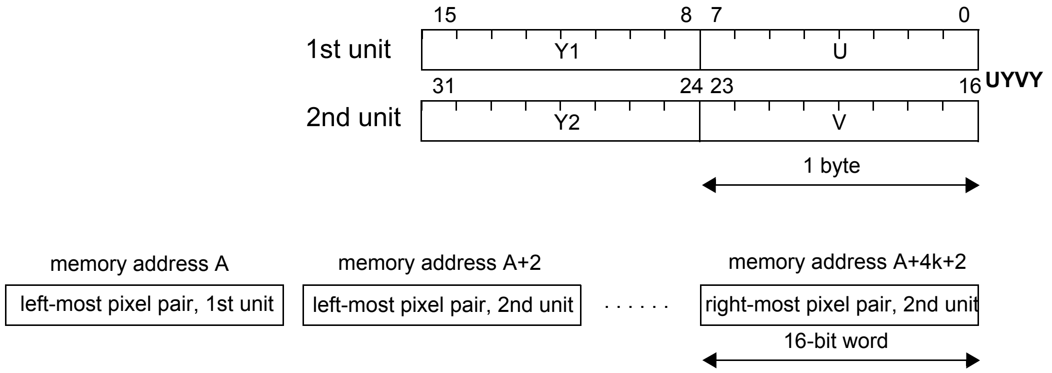


Figure 5: UYVY Packed YUV 4:2:2 Format

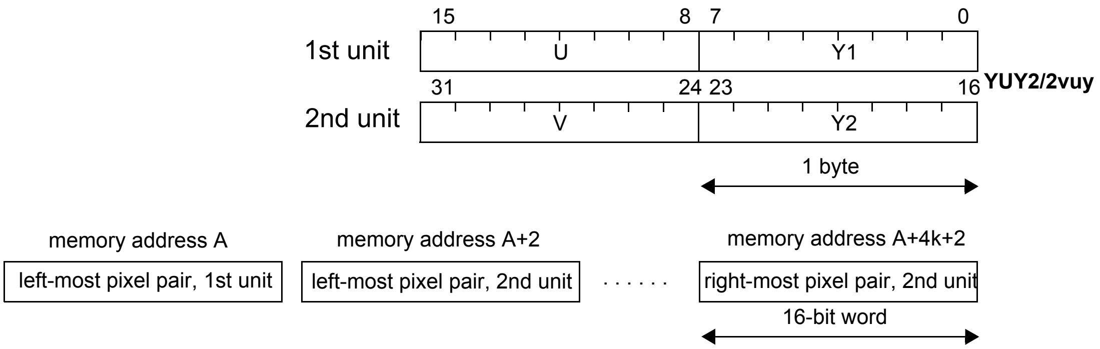


Figure 6: YUY2/2vuy Packed YUV 4:2:2 Format

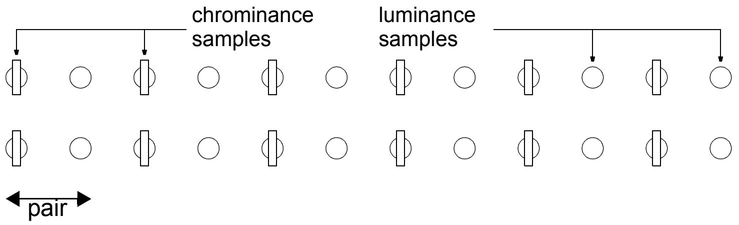


Figure 7: Spatial Sampling Structure of Packed and Planar YUV 4:2:2 Data

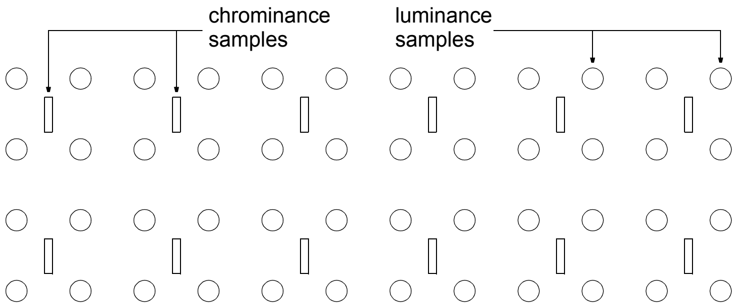


Figure 8: Spatial Sampling Structure of YUV 4:2:0 Data

7.3.5 Planar YUV 4:2:0 and YUV 4:2:2 Formats

The spatial sampling structure of planar YUV 4:2:0 data is shown in Figure 8. The planar YUV 4:2:2 data has the spatial sampling structure as in Figure 7.

There are two variants of planar YUV 4:2:0 and two of YUV 4:2:2:

- Planar, or ‘3 plane’ format. An image is described by 3 pointer values (Py, Pu, Pv). Each pointer points to a 2D array of Y, U and V values. Figure 9.

- Semi-planar, or ‘2 plane’ format. An image is described by 2 pointer values (Py, Puv). The Y pointer points to a 2D array of Y values. The Puv pointer points to a 2D array of UV pair values.

Remark: The U value of a UV pair always has the lower byte address. See Figure 10.

The MBS supports all planar formats on input and output. The VIP can produce the planar and semi-planar YUV 4:2:2 planar formats. The semi-planar YUV 4:2:0 format is the only format produced by the MPEG video decoder hardware.

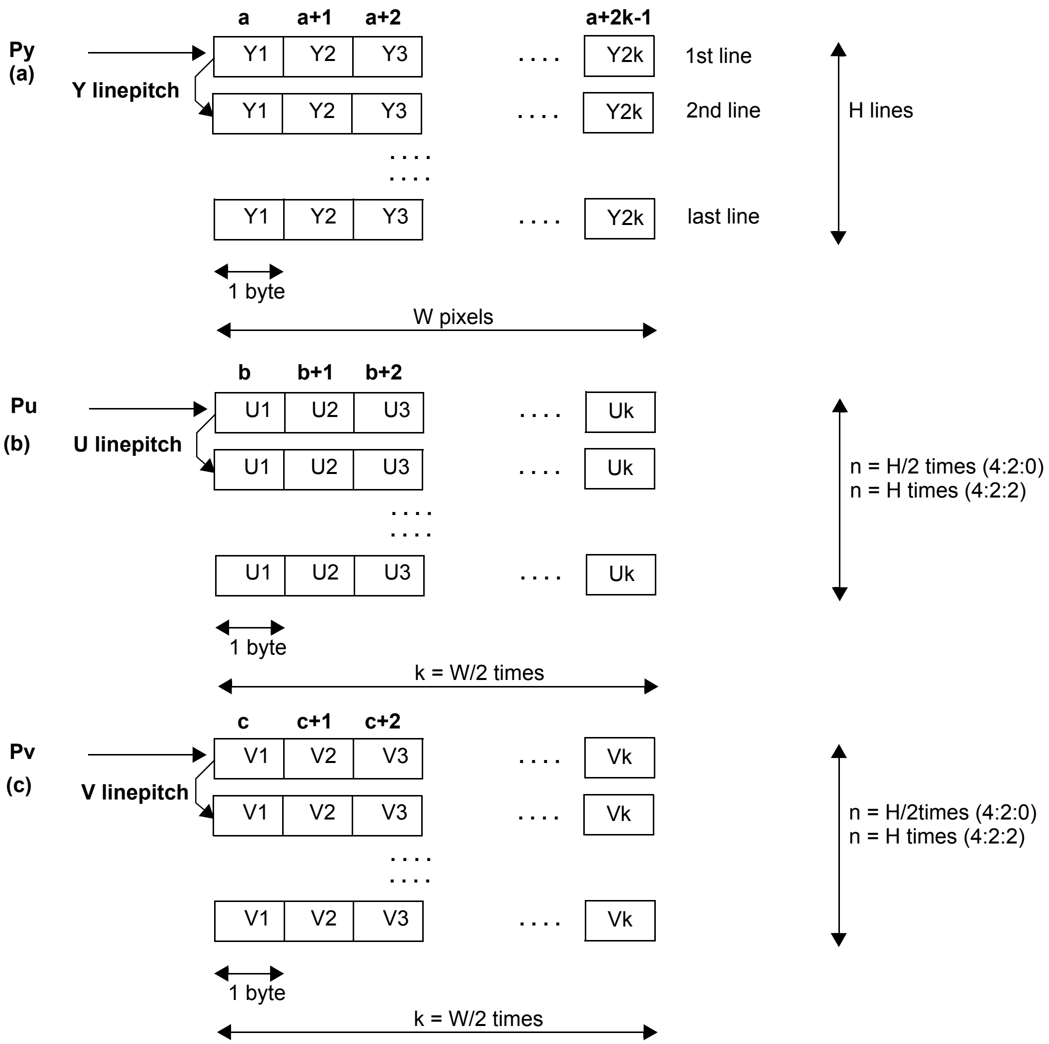


Figure 9: Planar YUV 4:2:0 and 4:2:2 Formats

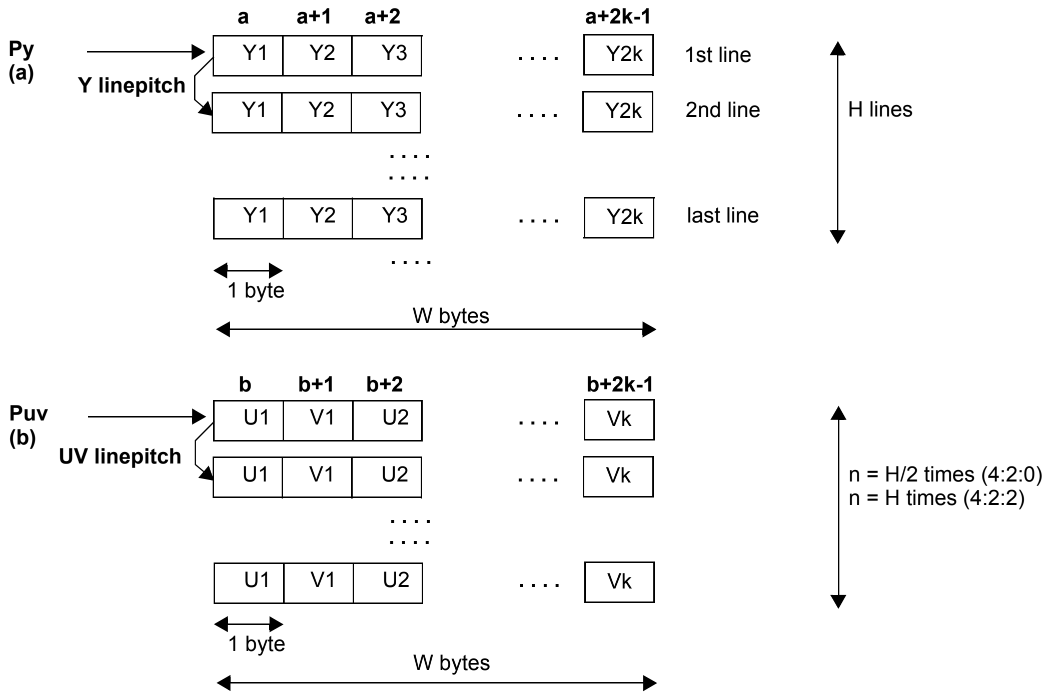


Figure 10: Semi-Planar YUV 4:2:0 and YUV 4:2:2 Formats

7.4 Universal Converter

The MBS input stage contains a universal pixel format converter that can convert any packed 16 or 32-bit pixel RGB format to an 8-bit alpha, R, G and B value for internal processing. This conversion can be done in combination with any MBS operation, particularly anti-flicker filtering.

The conversion is done by specifying the following:

- the width (16 or 32 bits) of a unit (this designates endian mode handling)
- the position (bit 31..0) within the unit for each of the alpha, R,G and B fields
- the width (1..8 bit) of each of the alpha, R,G and B fields

7.5 Alpha Value and Pixel Transparency

Many of the native pixel formats include a per-pixel alpha value. This alpha value is an inverse measure of the ‘transparency’ of a pixel. The AICP and 2D Drawing Engine are the only subsystems that interpret per-pixel alpha values when compositing surfaces. The native pixel format convention of a per-pixel alpha is shown in Table 2.

Table 2: Alpha Code Value and Pixel Transparency

Alpha Code	Transparency	Value
0	Fully transparent	0/256
1	Almost fully transparent	1/256
...		
254	Almost fully opaque	254/256
255	Fully opaque	256/256

The AICP input stage allows full alpha value table look-up using a 256 entry, 8-bit wide table. This can be used to translate a non-native format alpha convention to the native convention.

7.6 RGB and YUV Values

All RGB native formats use unsigned R,G and B values. The full range of values is allowed, i.e. 8-bit fields have a range [0,255].

All YUV native formats use the conventions from ITU-R BT. 601-4.

- Y ranges [16,235] with 16 designating a E_Y = 0 and 235 designating E_y = 1.0
- U, or Cb ranges [16,240] with 128 for a E_{Pb} = 0.0, 16 for E_{Pb} = -0.5, 240 for E_{Pb} = +0.5
- V, or Cr ranges [16,240] with 128 for a E_{Pr} = 0.0, 16 for E_{Pr} = -0.5, 240 for E_{Pr} =+0.5

7.7 Image Storage Format

With the exception of the planar formats, described in Section 7.3.5, the layout of an image in memory is determined by the following elements:

- pixel format, which implies the unit size(s)
- origin pointer—the (byte) address of the first unit of the image
- line pitch—the address difference between a pixel on a line and a pixel directly below it
- width W, in number of pixels
- height H, in number of lines

Remark: For indexed formats, each unit contains one or more pixels. For the packed formats, a unit is a pixel, with the exception of packed YUV 4:2:2 where two units are needed to describe a pixel pair.

Figure 11 shows how images are stored in memory

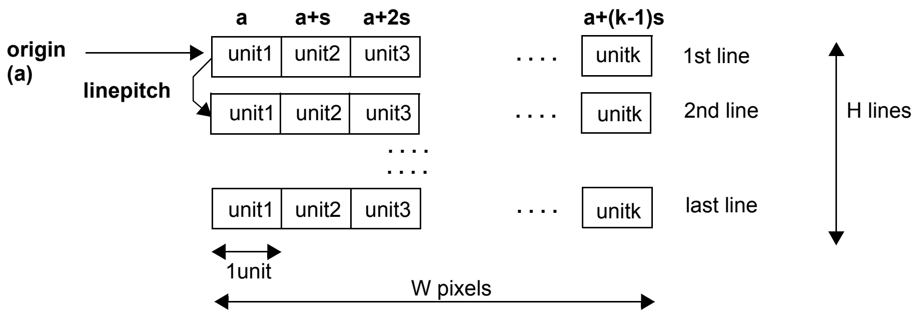


Figure 11: Image Storage Format

7.8 System Endian Mode

The PNX8526 is designed to run either little-endian or big-endian software. The entire system always operates in a single endian mode—i.e. both CPUs and all hardware subsystems run either little or big-endian. This is determined by a global endianness flag.

The endian mode determines how a multi-byte value is stored to/loaded from memory byte addresses.

For the native pixel formats, Section 7.3 always shows two elements in the figures: the layout of a ‘unit’, which is always 8, 16 or 32 bits, and the mapping of adjacent units to memory byte addresses. These two elements are always maintained, independent of system endian mode.

What this implies is that each hardware subsystem needs to map a unit to memory byte addresses in an endian mode-dependent manner. The rules are as follows:

- Storing a 16-bit unit to address ‘A’ results in modifying memory bytes ‘A’ and ‘A+1’
- Storing a 32-bit unit to memory address ‘A’ results in modifying memory bytes ‘A’ to ‘A+3’
- In little-endian mode, the least significant bits of a unit go to the lowest byte address
- In big-endian mode, the most significant bits of a unit go to the lowest byte address
- Going from left to right, adjacent units go to increasing memory addresses

The full details of how a hardware subsystem accomplishes this are not described here. In summary, a hardware block interfaces to a 32 or 64-bit DMA bus (the L2 interface) that has precise rules for which byte lane corresponds to which memory byte address. This allows a block to send or load the bytes belonging to successive units according to the above rules.

Chapter 8: PCI-XIO

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

8.1 Functional Description

The PCI-XIO module is a PCI-to-PI bridge with additional complex functionality that includes an independent DMA engine interface to the DVP memory bus and extended I/O (XIO) functionality. XIO supports IDE, NAND and NOR type Flash and 8-bit Motorola devices.

This module has the following interfaces:

- PCI Revision 2.2
- 32-Bit PI-Bus
- XIO (shared with PCI-AD and PCI-C/BE# lines)
- 64-Bit DVP Memory Bus

8.1.1 Overview

The PCI-XIO module is designed to optimize the throughput of PI-to-PCI and PCI-to-PI traffic. There is no frequency relationship assumed between PI and PCI clocks. The details of the different functions inside this module are described below.

8.1.1.1 PCI Interface

The PCI-XIO module supports 33 MHz PCI spec version 2.2. It can operate as a configuration manager when the internal processor (e.g., MIPS) is enabled. It can also act as a target to external configuration cycles when an external processor and north bridge are used in the system.

Features:

- Three base addresses are supported.
- Option to enable internal PCI arbiter which can support up to three external PCI masters. If IDE or Motorola XIO devices are enabled, the maximum number of external masters is two.
- As a PCI master, it can generate all types of single transaction PCI cycles: IO, memory, interrupt acknowledge and configuration cycle.
- Linear burst mode is supported on memory transactions. Other burst mode transfers are terminated after a single data transfer.
- A DMA engine provides high speed transfer to and from SDRAM and an external PCI device. The DMA can also be used to transfer data to and from XIO devices.

- The PCI clock and RST are generated externally and input to this module.
- In PCI terminology it is a single function device.

The following general PCI features are not implemented in the PCI-XIO module:

- As a PCI target, the device only responds to memory and configuration cycles.
- Subtractive decoding is not supported.
- There is no hard-coded legacy decoding of address space (such as VGA IO and memory).
- Burst-to-IO and configuration space is not supported.
- The INT function is provided by the PIC. Refer to Chapter 6 Priority Interrupt Controller for information on the Priority Interrupt Controller (PIC).

Supported commands on PCI are shown in the following table:

Table 1: Supported PCI Commands

Command: PCI Target Responds to:	**Command: PCI Master Generates**
	IO Read
	IO Write
Memory Read	Memory Read
Memory Write	Memory Write
Configuration Read	Configuration Read
Configuration Write	Configuration Write [1]
Memory Read Multiple	Memory Read Multiple
Memory Read Line	Memory Read Line
Memory Write and Invalidate	Memory Write and Invalidate
	Interrupt Acknowledge
Note : Configuration write can be initiated only when configuration management is enabled.	

8.1.1.2 PI Interface

The PI side of the bridge acts both as a master and slave on the 32-bit PI-Bus. It supports both big and little-endian systems.

Features:

- As a PI Master
  - Supports default grant
  - Does locked WDUs for burst-writes. This feature may be disabled, when it issues multiple unlocked WDUs.
  - Ping-pong 32 x 32-bit buffers to sustain continuous external PCI agent write-bursts
  - Supports all subword PI opcodes
  - Supports “PI retract/error-acknowledge” and “PI timeout”

- As a PI Slave
  - Supports all opcodes for both reads and writes
  - Supports PI timeout
  - Only contiguous addresses with the same opcode are supported in a locked transfer.
  - Three PI selects: one each for PCI space, XIO space and internal MMIO registers
  - Supports posted and non-posted writes. In non-posted writes, only WDU and subword accesses are allowed.
  - Bursting to internal MMIO register space is not allowed.
  - Retracts on PI-Bus when getting a “retry” on PCI for reads and non-posted single writes.

8.1.1.3 DVP Memory Bus Interface

To optimize PCI-to-system memory throughput in the DVP system, a direct path has been provided between PCI and the DVP memory bus using this interface. This interface is master on the 64-bit DVP memory bus. There is a switch to divert traffic from the external PCI masters to the system memory through this interface. By default that traffic goes via the PI interface.

Features:

- Separate read/write buffers of 64x32 bits
- For PCI burst reads, speculative read of 32 words (128 bytes) is done from the memory
- 166 MHz DVP memory bus
- Continuous PCI write/read bursts can be sustained (contingent on availability of the DVP memory bus).

8.1.1.4 XIO Interface

The XIO interface uses a part of the PCI interface and some additional signals to interface with external Flash (NAND and NOR types), NOR-ROM, IDE and 8-bit Motorola devices. This function “steals” a PCI cycle and runs an XIO transfer using part of the PCI bus before giving control back to PCI. The XIO port may be accessed at any time after the configuration registers have been initialized. Up to three profiles may be enabled at one time. Each one requires a chip select. When 64 MB addressing is required, an extra pin (XIO_A[25]) is required. When an IDE or Motorola profile is enabled, a REQ_B or GNT_B is used for the acknowledge or interrupt. Flash profiles have a dedicated ACK pin to allow PCI transactions to continue while the device is busy. Any pin that is not being used in one of these functions may be turned over for other uses by the GPIO module.

Motorola Interface

In this XIO mode, any 8-bit Motorola 68360 type external slave can be addressed. For details about connecting a Motorola device to a PCI interface, please refer to Table 1. Even though the Motorola interface is an asynchronous interface, internal timings are generated in multiples of PCI clock. For programming to do Motorola cycles, please refer to XIO Selx Profile registers. For writes, data-strobe (DS) assertion time is made programmable by using sel0_we_hi field. There is an option to use the acknowledge from the device DSACK, or to have a fixed wait time before probing for read-data and removing DS for write-data.

NAND-Flash Interface

A flexible interface is provided to interface to a NAND-Flash. There are two registers that define the type of cycle that will be performed. The read and write strobes can be programmed independently with a high timer from one to four pci clocks. A cycle may contain 0, 1, or 2 commands and 0, 1, 2, or 3 address phases with or without data. Refer to Section 8.2.1 for information on how to use this interface.

NOR Flash Interface

In this XIO mode, any 8-bit NOR flash can be addressed. Up to 64 MB may be addressed. The DS timing is programmable as is the WN timing. The user has the option of monitoring the R/BN signal from the flash or using a fixed response for the DS low timing.

IDE Interface

In this XIO mode, an IDE disk drive can be addressed. Only PIO mode is supported. The internal DMA engine can be programmed to perform data transfer to and from the IDE once the disk drive’s registers have been programmed. The DIOR and DIOW strobe high and low times are programmable. Refer to Section 8.2.4 for more details.

8.2 Operation

The PCI / XIO module supports the 33 MHz PCI spec version 2.2.

The PI interface supports all subword operations along with all block word operations. PI double-word operations are not supported because the bus is restricted to 32 bits. The PI section will also split a PCI transaction into multiple transactions on the PI-Bus when the “byte enables” on PCI do not match any PI opcode.

All word writes from PCI are mapped to WDU (single data phase) opcodes on PI. During a burst, these WDU will be locked together. Breaks in the sequence, such as a null data phase or “less than a word” will split the internal transfer into multiple transfers on PI. Subword data phases will always be transferred as isolated (non-locked) transfers on PI. Reads from PCI may be mapped to WDU or WD32 (32 data phase) PI transactions. All subword reads are mapped to WDU reads. If spec_rd_en bit is asserted for one of the PCI apertures and the address is aligned to 128 bytes, a WD32 read is executed on PI. This allows the PCI to have data ready instantly after the initial latency. This should only be used when there are no speculative read side effects. If the address is not “128-byte aligned” or the spec_rd_en bit is not asserted, the read will map to WDU. In this mode, data is retrieved only as requested.

Remark: Reads with high initial latency will be very slow as each data phase will have the equivalent of the initial latency.

Access from external masters may be restricted to word-only if desired. When this feature is enabled, attempted access with less than a word will result in the PCI slave terminating the transaction with a target abort or ignoring the write and returning 0 on read. This behavior is determined by a configuration switch.

When the PCI device can not return data on reads within 16 clocks, the transactions will terminate in a retry. The read will be completed internally and the PCI will hold the data exclusively for the initiating agent for the duration of the “read_lifetime” timer. No other read will be accepted while the timer is active. After the timer expires, any read request will be accepted. The saved data will be tossed if a different master requests a read.

Any XIO device(s) can be accessed any time after the PCI configuration space has been initialized. To be PCI compliant, it will monitor the internal address rather than the IO pads. At this time, an XIO cycle is run on the PCI pins. PCI pins AD[31:24] present the address to the device while AD[7:0] contain the data. The PCI CBE pins are used for XIO control. There are three dedicated pins to be used as chip selects to the device(s).

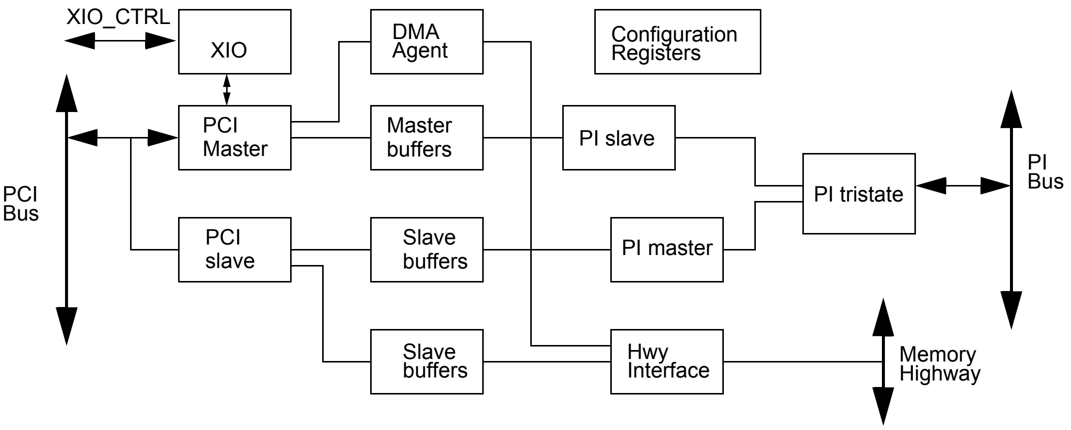


Figure 1: PCI Block Diagram

The following table shows how the XIO supports Motorola 68360 8-bit devices along with various flash types.

Table 2: XIO Pin Multiplexing

PCI Signal	I/O	68360 8-Bit	NOR Flash	NAND-Flash	IDE	Can Be GPIO
DEVSEL#	I/O	NA	NA	NA	NA	N
FRAME#	I/O	NA	NA	NA	NA	N
IRDY#	I/O	NA	NA	NA	NA	N
TRDY#	I/O	NA	NA	NA	NA	N
STOP#	I/O	NA	NA	NA	NA	N
IDSEL	I/O	NA	NA	NA	NA	N
PAR	I/O	NA	NA	NA	NA	N

Table 2: XIO Pin Multiplexing *…Continued*

PCI Signal	I/O	68360 8-Bit	NOR Flash	NAND- Flash	IDE	Can Be GPIO
PERR#	I/O	NA	NA	NA	NA	N
SERR#	I/O	NA	NA	NA	NA	N
REQ_A#	I	NA	NA	NA	NA	Y
REQ_B#	I	DSACK	NA	NA	NA	Y
REQ#	I/O	NA	NA	NA	NA	N
GNT_A#	O	NA	NA	NA	NA	Y
GNT_B#	O	NA	NA	NA	INTREQ	Y
GNT#	I/O	NA	NA	NA	NA	N
AD[31:24]	I/O	D[7:0]	D[7:0]	AD[7:0]	D[15:8]	N
AD[23:16]	I/O	A[23:16]	A[23:16]	NA	D[7:0]	N
AD[15]	I/O	A[15]	A[15]	NA	CS1	N
AD[14]	I/O	A[14]	A[14]	NA	CS0	N
AD[13:11]	I/O	A[13:11]	A[13:11]	NA	A[2:0]	N
AD[10]	I/O	A[10]	A[10]	NA	DIOW	N
AD[9]	I/O	A[9]	A[9]	NA	DIOR	N
AD[8]	I/O	A[8]	A[8]	NA	DATA_DIR	N
AD[7:2]	I/O	A[7:2]	A[7:2]	NA	NA	N
AD[1]	I/O	A[1]	A[1]	ALE	NA	N
AD[0]	I/O	A[0]	A[0]	CLE	IORDY	N
CBE3#	I/O	A[24]	A[24]	NA	NA	N
CBE2#	I/O	AS	OEN	REN	NA	N
CBE1#	I/O	R/WN	WN	WEN	NA	N
CBE0#	I/O	DS	NA	SA	NA	N
XIO_A[25]	O	A[25]	A[25]	NA	NA	Y
XIO_ACK	I	NA	R/BN	R/BN	NA	Y
XIO_SEL0,1,2	O	CS	CE	CE	CE	Y

8.2.1 NAND-Flash Interface

Interfacing to a NAND-Flash involves both hardware setup and software support. The hardware support is designed to be very flexible in supporting the standard devices plus extensions that may be provided by some flash vendors. Table 4 shows recommended settings for the hardware configured for various NAND-Flash operations.

A NAND transaction may consist of 0, 1, or 2 command phases and 0, 1, 2 or 3 address phases, and n data phases. The sequence is as follows: first command, low address (address bits [7:0]), middle address (address bits [16:8]), high address

(address bits [24:17]), data, second command. For transactions with fewer than three address phases, low address is first dropped, then middle address. Any transaction that includes an address phase must include at least one command phase.

Table 3: Recommended Settings for NAND

Description	Cmd No.	Addr No.	Include Data	Monitor ACK	Cmd A	Cmd B	Notes
Read Data	1	3	Y	Y	00h or 01h	NA	Recommended to use DMA. This may be set to more than one segment if restricting max_burst_size to 128.
Read ID	1	1	Y	N	90h	NA	Recommended to use direct (or indirect) access.
Read Status	1	0	Y	N	70h	NA	May read up to four bytes of status with direct access.
Write Data	2	3	Y	Y	80h	10h	Recommended to use DMA.
Block Erase	2	2	N	Y	60h	d0h	Recommended to use direct (or indirect) access.
Reset	1	0	N	N	ffh	NA	Recommended to use direct (or indirect) access.

With a direct access to the NAND, n is limited to 4 bytes. Using the DMA, n is limited to the segment length, 512 or 528 bytes with spare area. This is to allow time for the busy signal to become stable at segment boundaries. The DMA may be programmed to read much larger areas if the NAND does not assert its busy state or is allowed to pause at segment boundaries. Programmers should consult the vendor’s data sheet for the appropriate NAND-flash selection.

The WEN and REN timing information will also be found in the data sheets. The PNX8526 supports read profiles with low time from 1 to 4 PCI clock periods. Write profiles of 1 to 4 PCI clock periods is supported for command and address writes. Data writes must use a high time of at least 2 PCI clock periods. If data is not part of the transaction, the second command will follow the last address phase.

The ACK signal is monitored, when enabled, only at predetermined parts of the transaction. During read operations, it will monitor the ACK after the last address phase, before the read begins. The fixed delay must be programmed to a value sufficient to allow the ACK to become valid before sampling it. This should include time to double synchronize the ACK to the PCI clock. The ACK is also sampled before starting a NAND transaction (but after the PCI wrapper has started). This applies to all types of transactions. Even a status read will stall until the device is ready if monitor ACK is enabled when starting the NAND transaction.

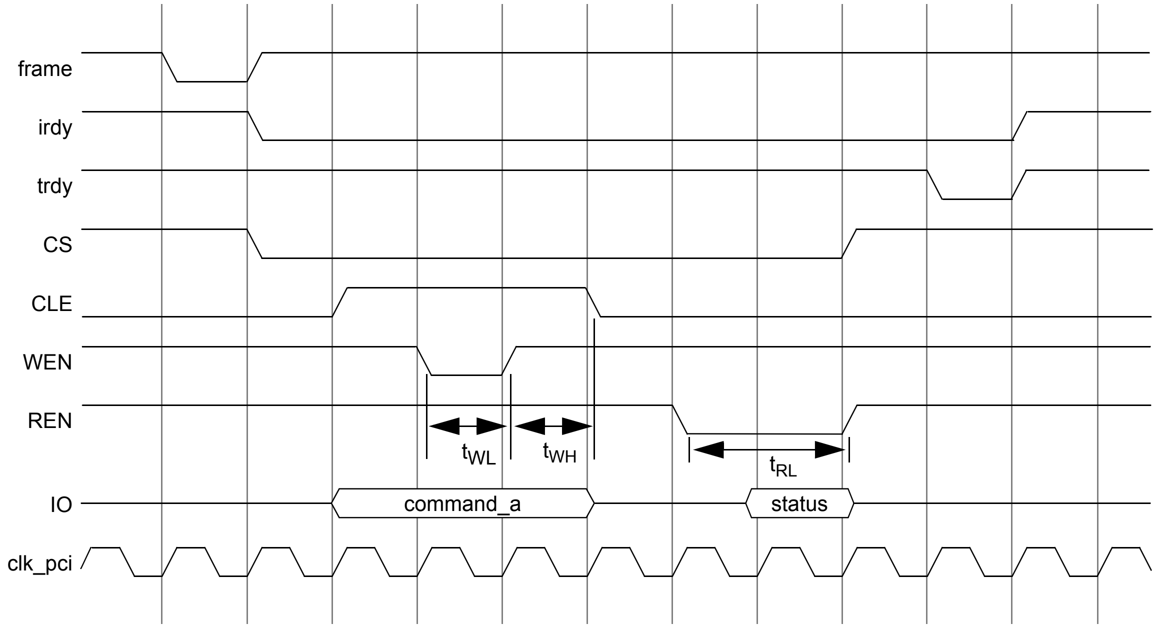
The read data operation may be done by blending DMA and direct access to minimize the time the PCI bus is blocked from other types of transactions. To do this, set the profile to issue 1 command, 3 address phase, and no data. Also load the appropriate command into the Command A register. Next do a write to the starting address of interest. Change the profile to 0 command, 0 address, include data. The DMA should be programmed to transfer the selected amount of data to SDRAM. If the DMA is started before the device is ready, it will stall until the device is ready.

The ACK may be monitored to determine when the device is ready prior to initiating the DMA. Once the device is ready, no further monitoring of the ACK takes place. If the amount of data to be transferred exceeds one segment, the max burst size should be set to 128 to allow for pause in the transfer that allows the ACK to be monitored between segments.

Remark: This approach will not pause at the correct location if the spare area is being accessed.

A pin is provided for accessing the spare area. This is multiplexed with CBE[0] of the PCI bus. It will be driven to the selected state during all command phases, address phases and data phases. This may not meet the timing requirements on some manufactured devices. For devices that need the pin to be stable outside of this area, the SA pin may be tied low if always used, tied high if never used, or tied to a GPIO and enabled as needed. The boot script that loads the initial code expects the spare area access to be disabled.

Examples of block erase, data read and write and status read are shown in the following timing diagrams.

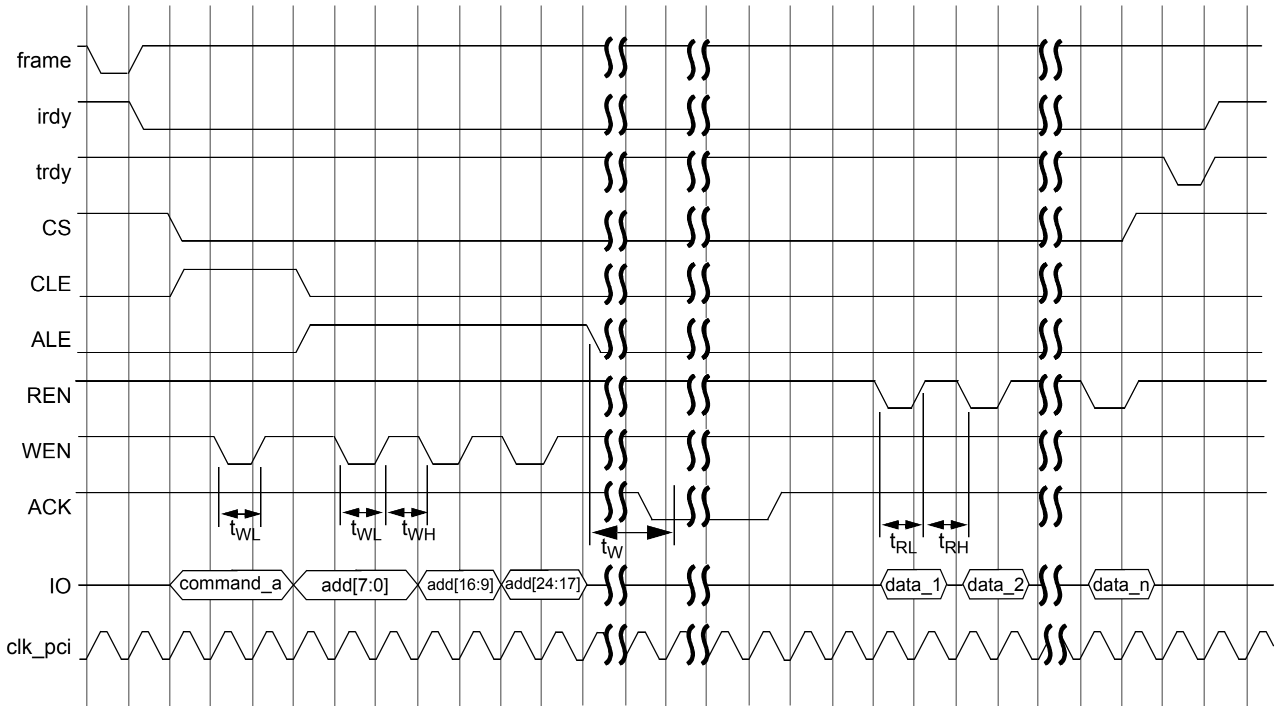


t_{WH}: (WEN_hi + 1) * clk_pci period Shown with WEN_hi = 0

t_{WL}: (WEN_lo + 1) * clk_pci periodShown with WEN_lo = 0

t_{RL}: (REN_lo + 1) * clk_pci periodShown with REN_lo = 1

Figure 2: Read Status



t_{WH}: (WEN_hi + 1) * clk_pci period Shown with WEN_hi = 0

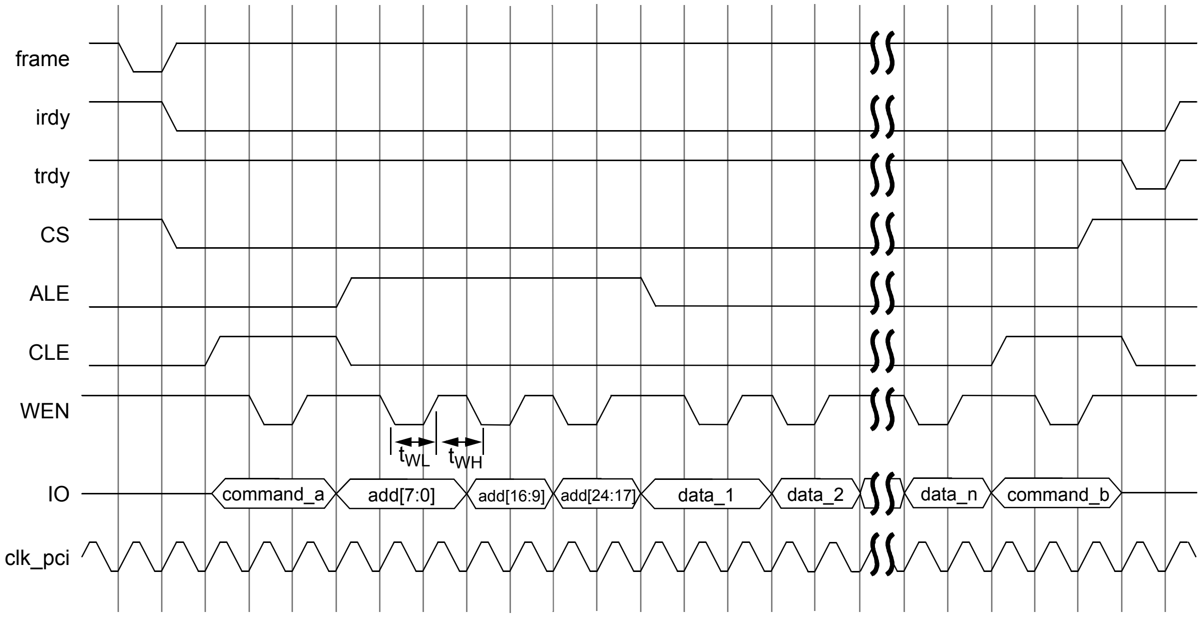
t_{WL}: (WEN_lo + 1) * clk_pci periodShown with WEN_lo = 0

t_{RH}: (REN_hi + 1) * clk_pci periodShown with REN_hi = 0

t_{RL}: (REN_lo + 1) * clk_pci periodShown with REN_lo = 0

t_{W}: (DLY + 1) * clk_pci periodWait time until ACK valid

Figure 3: Read Data

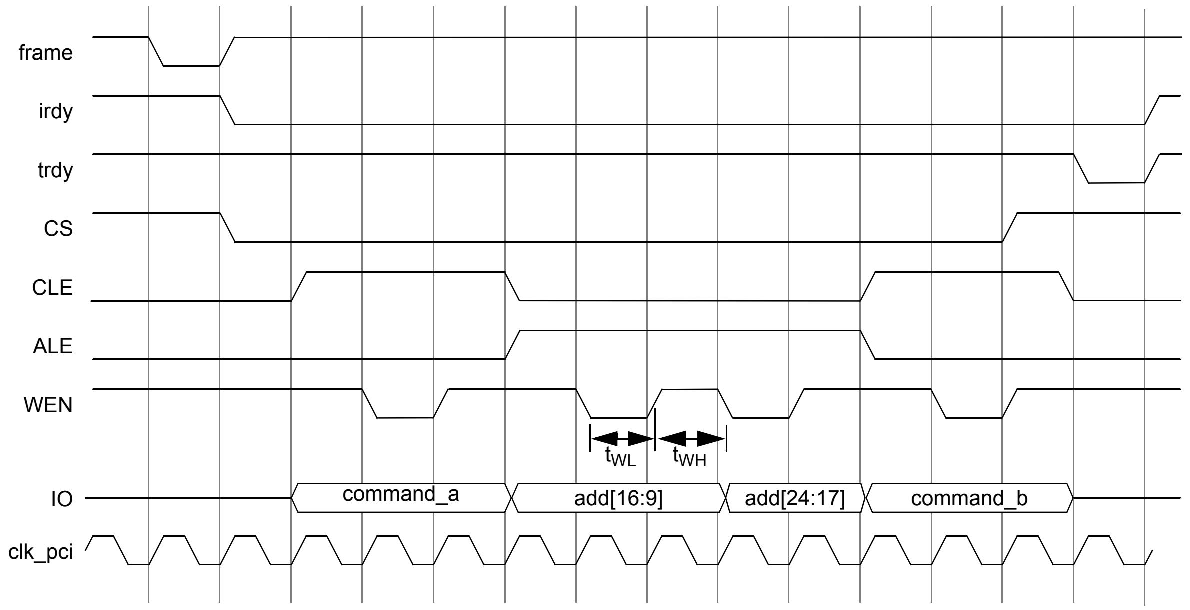


t_{WH}: (WEN_hi + 1) * clk_pci period Shown with WEN_hi = 0

t_{WL}: (WEN_lo + 1) * clk_pci periodShown with WEN_lo = 0

Figure 4: Write Data

Refer to Table 2 on page 8-170 for signal descriptions.



t_{WH}: (WEN_hi + 1) * clk_pci period Shown with WEN_hi = 0

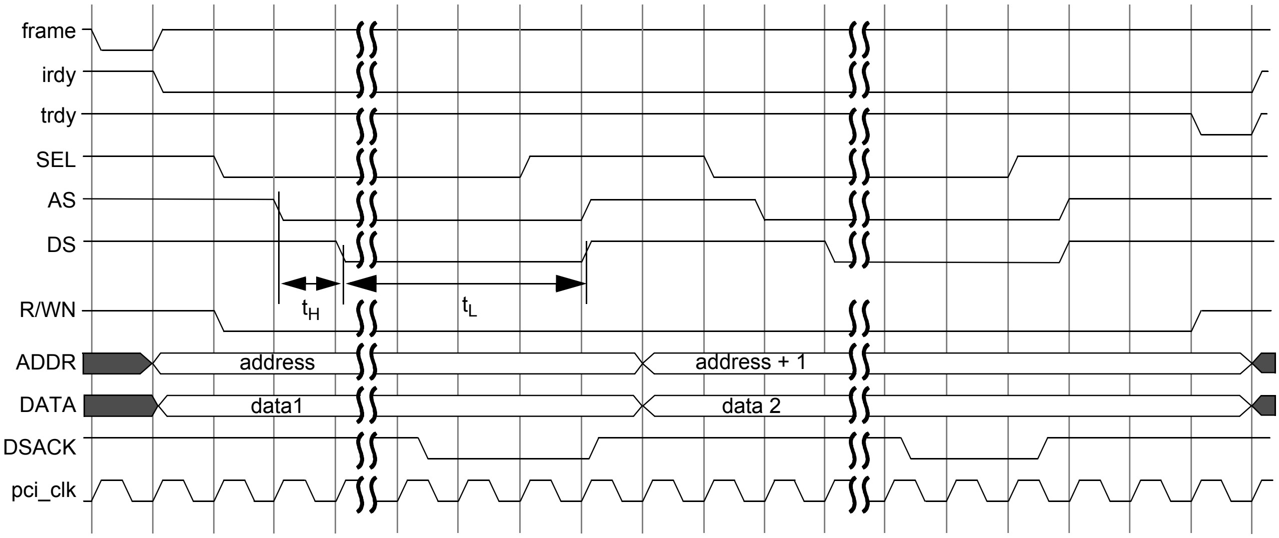
t_{WL}: (WEN_lo + 1) * clk_pci periodShown with WEN_lo = 0

Figure 5: Block Erase

8.2.2 Motorola Style Interface

The Motorola style interface supports 8-bit devices only. The following timing diagrams illustrate a 2-byte write and 2-byte read operation. The time between the falling edge of AS and DS is controlled by the DS time high field in the profile register. The time low is determined by the DS time low field of the profile register or by the external device if “wait for ACK” is enabled.

The t_H (write time high) and t_L (wait low) timing should be programmed to match the device according to the vendor’s specification. The resolution is a multiple of the PCI clock period. Refer to the descriptions for the XIO Select Profile registers.

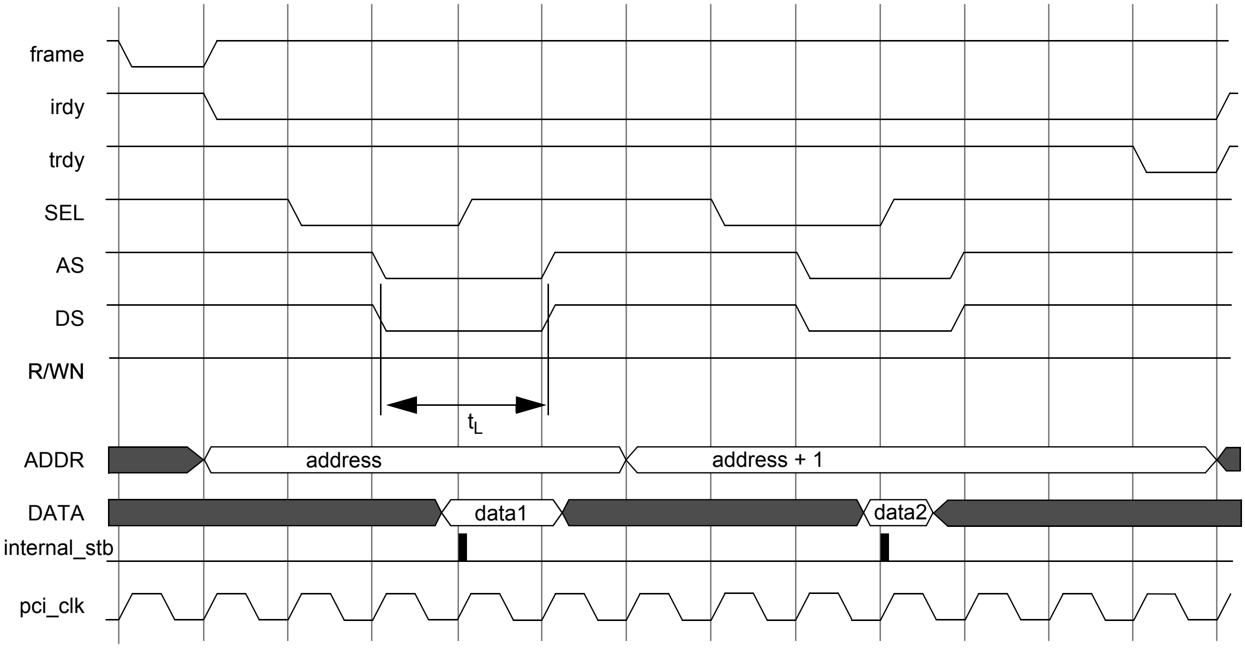


t_H: DS_high * clk_pci period Shown with DS_high = 1

t_L: DS_low * clk_pci periodShown with DSACK monitoring

Figure 6: Motorola Write

Refer to Table 2 for signal descriptions.



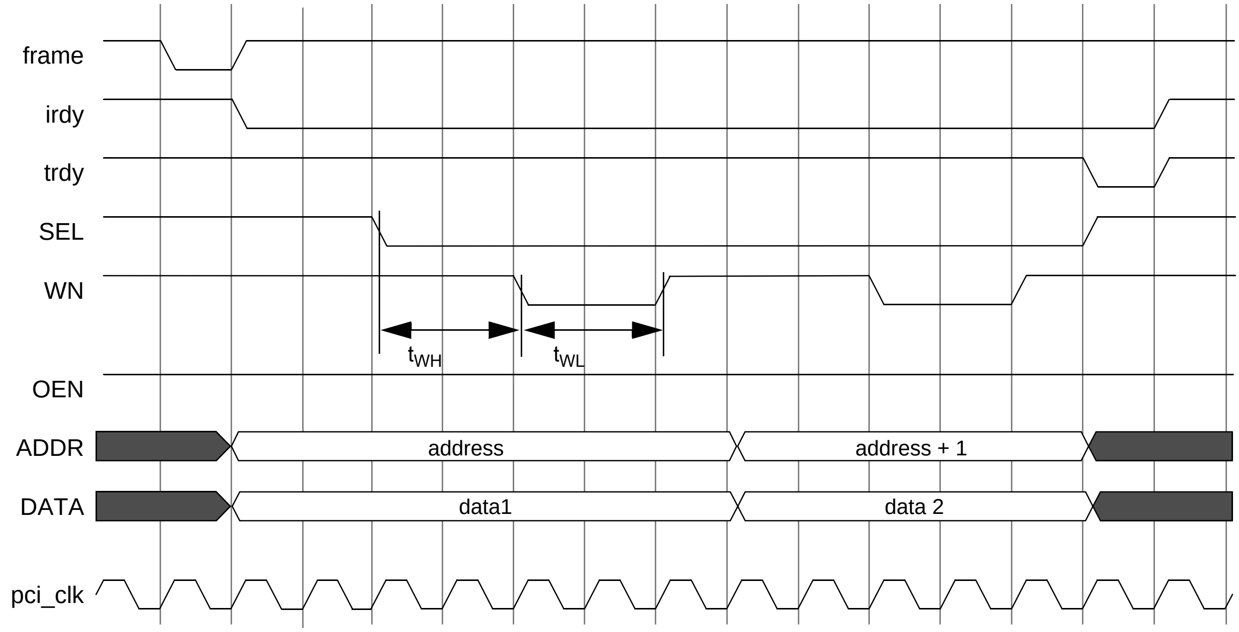
t_L: (DS_low +1) * clk_pci periodShown with DS_lo = 1

Figure 7: Motorola Read

Refer to Table 2 for signal descriptions.

8.2.3 NOR Flash Interface

The NOR flash interface supports 8-bit devices only. The following timing diagrams illustrate write and read timings for a typical NOR device. The busy signal is not shown; it should be inactive during these cycles. Typically, the busy signal will be monitored before starting a transaction to the NOR flash. The t_{WH} (write time high) and t_{WL} (write time low) timing should be programmed to match the device based on the flash vendor’s specification. Refer to the descriptions for the XIO Select Profile registers. The resolution is a multiple of the PCI clock period. The t_R (read time, or “wait for data”) is also programmed in the profile bits[13:9].

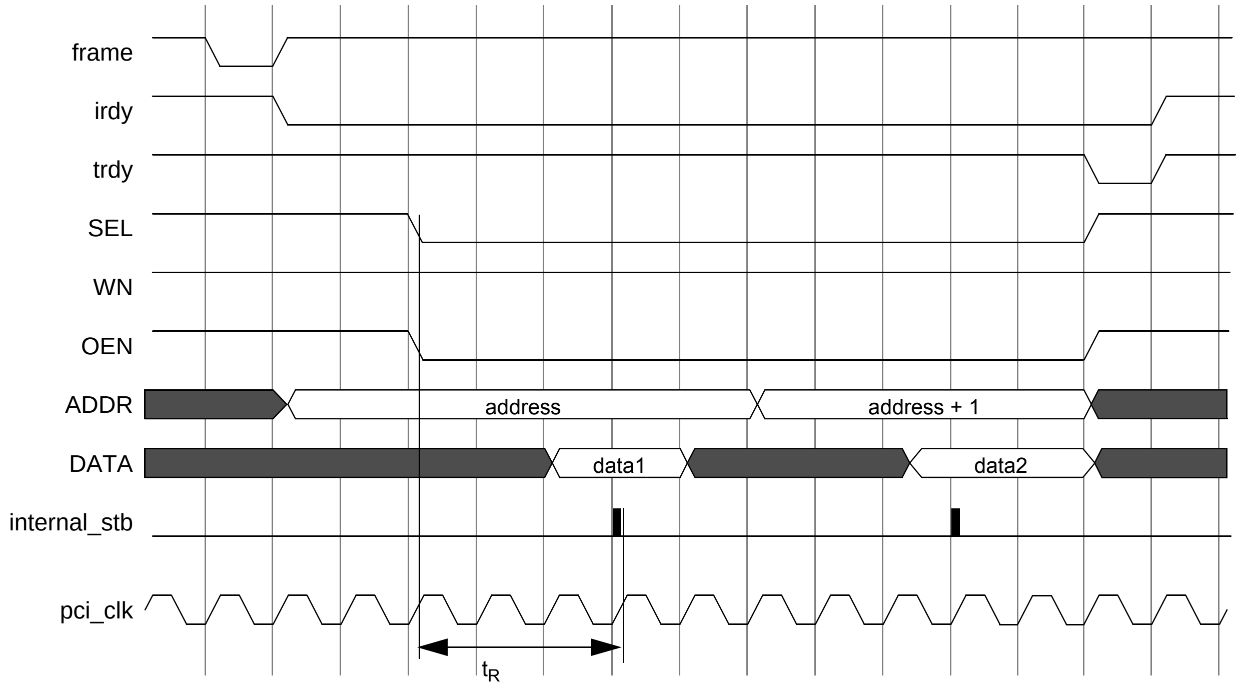


t_{WH}: (WN_high + 1) * clk_pci period Shown with WN_high = 1

t_{WL}: (WN_low + 1) * clk_pci periodShown with WN_low = 1

Figure 8: NOR Flash Write

Refer to Table 2 for signal descriptions.



t_R: OEN_lo * clk_pci period Shown with OEN_lo = 3

Figure 9: NOR Flash Read

Refer to Table 2 for signal descriptions.

8.2.4 IDE Description

The IDE (ATA) interface supports PIO mode transfer with a theoretical maximum transfer rate of 16.6 MB/s (PIO-4 mode). (The PNX8526’s bandwidth requirement for data transfer on IDE is under 10 MB/s.) The XIO module DMA is used for data transfer to and from the disk.

All IDE disk registers (eight command and one control) are accessible via PI. All IDE disk registers are indirectly accessed via GPXIO registers.

Figure 10 shows a block diagram of the IDE interface.

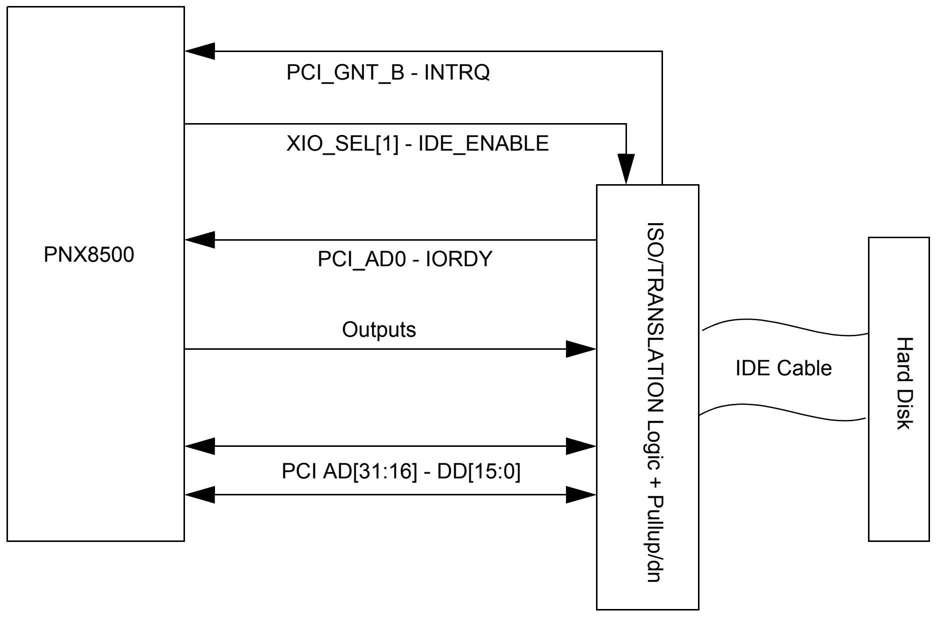
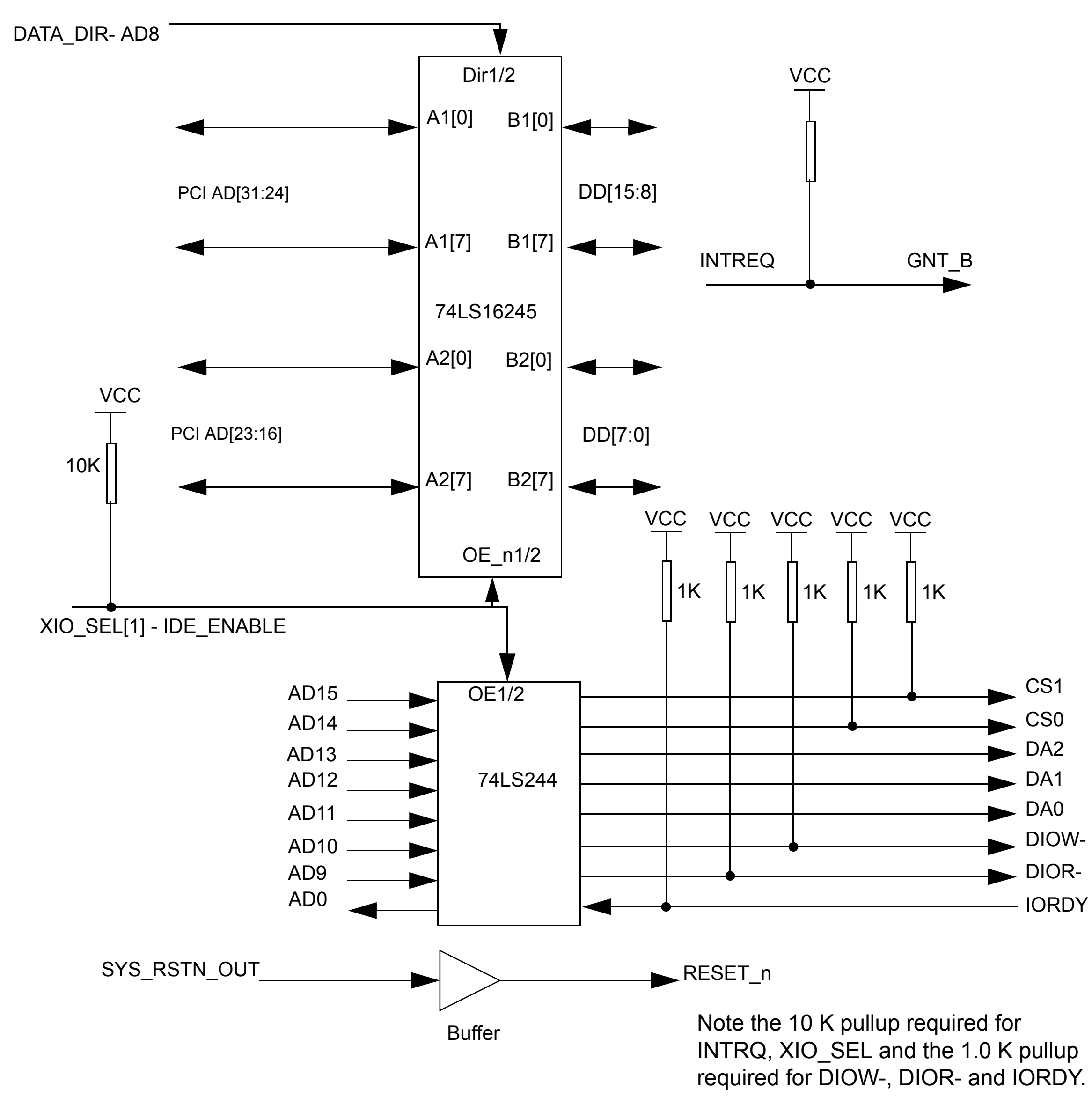


Figure 10: IDE Interface

The IDE port is multiplexed with PCI, FLASH and Motorola interface pins. There are two dedicated pins, IDE_ENABLE (XIO_SEL[1]) and INTRQ. The IDE Disk interrupt (INTRQ) is connected to PCI_GNT_B, which is routed to the PIC through GPIO.

The SYS_RSTN_OUT of PNX8526 is connected with the IDE interface reset. All outputs are driven on PCI-CLK. All inputs are registered on PCI_CLK. The Low and High periods of DIOR/DIOW are programmable (using sel profile register).

All physical signals need to be isolated from PCI on the board as shown in Figure 11



Note the 10 K pullup required for INTRQ, XIO_SEL and the 1.0 K pullup required for DIOW-, DIOR- and IORDY.

Figure 11: Isolation Translation Logic

8.2.4.1 Data Transfer Operation

In PIO mode, data transfer to/from disk is done using read/write operations of the command and control block registers. PI/PO protocol is explained in the ATA-2 specification.

All command block registers can be programmed using direct or indirect access in the XIO block. All disk registers are programmed. When the disk is ready to transfer data, DMA is enabled.

8.2.4.2 Registers

All IDE device registers are defined in the ATA-2 Specification. These registers can be accessed directly from PI or indirectly via GPXIO registers. The lower five bits of the GPXIO address register need to be configured as follows:

Table 4: GPXIO Address Configuration

Address to be Written	Register Name	Address on IDE				
		CS1	CS0	DA2	DA1	DA0
5’b40	Data register	1	0	0	0	0
5’b44	ERR/Feature	1	0	0	0	1
5’b48	Sector count	1	0	0	1	0
5’b4C	Sector number	1	0	0	1	1
5’b50	Cylinder Low	1	0	1	0	0
5’b54	Cylinder High	1	0	1	0	1
5’b58	Device/Head	1	0	1	1	0
5’b5C	Status/Command	1	0	1	1	1
5’b38	Alternate status/Device control	0	1	1	1	0

Programming IDE Registers

IDE is a submodule of PCI-XIO. It shares PCI pins with other XIO blocks. Three XIO SEL pins can be configured for use by any XIO device. Each SEL pin is associated with the profile register in the PCI block. The profile register determines the mode of the SEL pin, pulse width for control signals and memory apertures for each mode.

Before accessing any IDE register, the appropriate profile register needs to be programmed. For example, if XIO_SEL[1] has been used for IDE, the sel1_profile register needs to be programmed and IDE needs to be enabled.

- At power on, the IDE disk will respond in PIO-0 mode only.
- Program the appropriate register in PIO-0 mode to set PIO-4 mode.
- Using sel1_profile register, set lo and high period of DIOR/DIOW pulses for PIO-4 mode.
- High period in selx_profile register is used for the setup time of DA/CS lines with DIOR/DIOW.
- Low period in selx_profile register is used for the lo period of the DIOR/DIOW pulse.
- Hold of DA/CS with respect to DIOR/DIOW is always for one PCI clock.
- Recommended values for sel_we_hi and sel_we_lo for PIO-0 mode are 7 and 13, respectively (assuming a 33 MHz PCI clock).
- Recommended values for sel_we_hi and sel_we_lo for PIO-4 mode are 1 and 3 respectively.

- During DMA transactions the high period is used for the setup of the first transaction only.

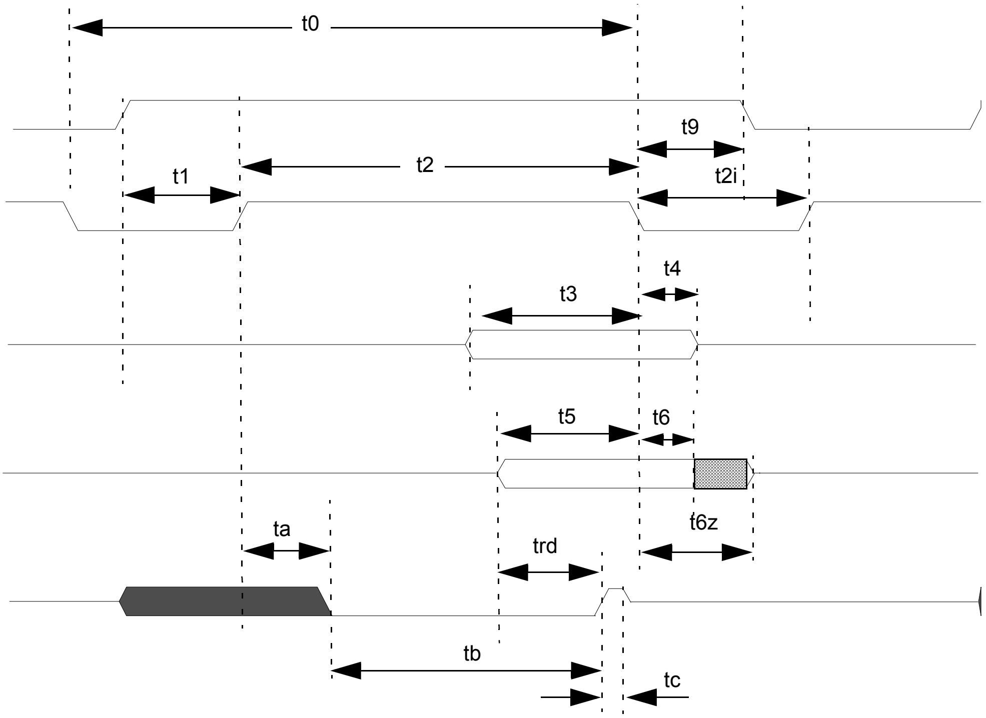
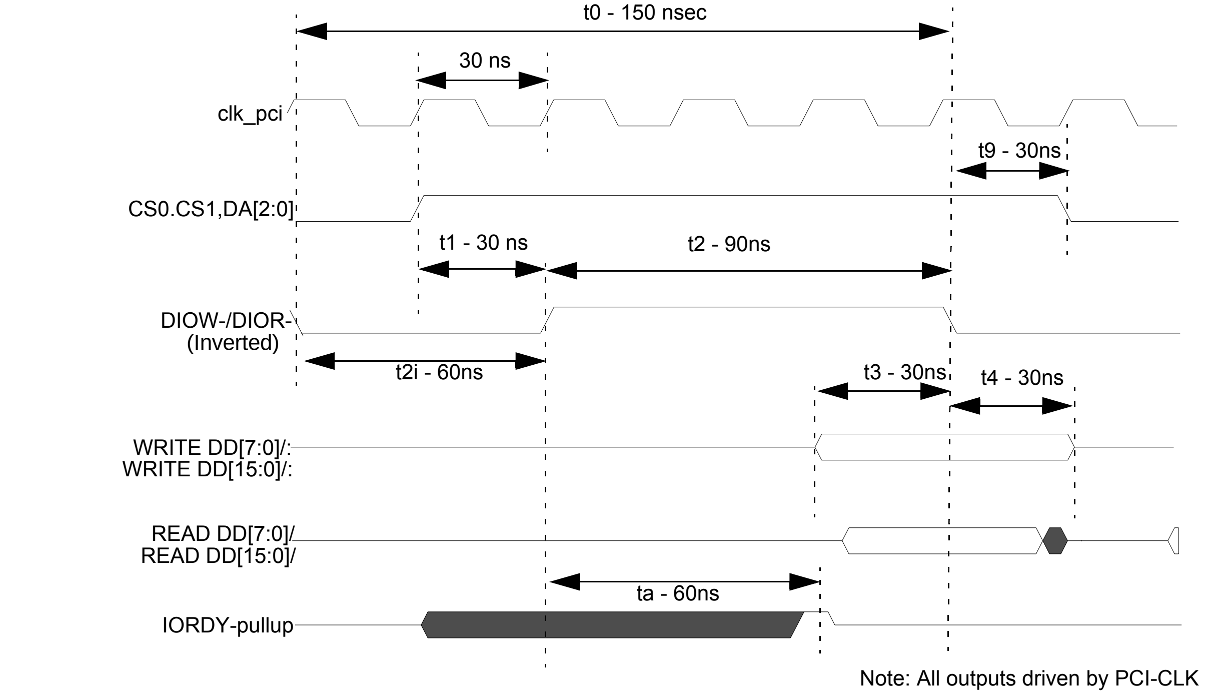


Figure 12: Register Transfer/PIO Data Transfer on IDE

Table 5: IDE Timing

PIO Timings (ATA-2 Spec)		Mode 0	Mode 4 (ns)
t0	Cycle time (min)	600	120
t1	ADD valid to DIOR/DIOW setup (min)	70	25
t2	DIOR/DIOW pulse width (min)	165	70
t2i	DIOR/DIOW Recovery time (min)	-	25
t3	DIOW data setup (min)	60	20
t4	DIOW data hold (min)	30	10
t5	DIOR data setup (min)	50	20
t6	DIOR data hold (min)	5	5
t6z	DIOR data tristate(max)	30	30
t9	DIOR/DIOW to add,cs hold (min)	20	10
trd	Read data valid to IORDY active (min)	0	0
ta	IORDY setup time (max)	35	35
tb	IORDY pulse width (max)	1250	1250
tc	IORDY assertion to release (max)	-	5



Note: All outputs driven by PCI-CLK

Figure 13: Timings on IDE Bus

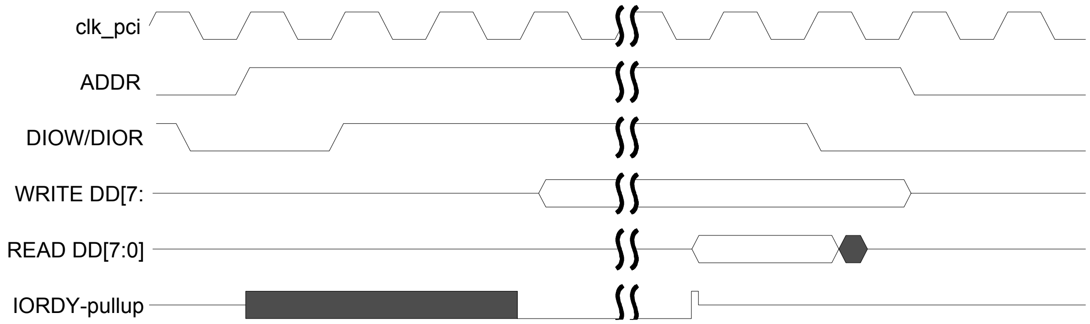


Figure 14: IDE Transaction, Flow Controlled by Device IORDY

8.2.5 PCI Interrupt Enable Register

PCI_INTA is the PNX8526 PCI interrupt pin. It can be enabled as an output pin and connected to pci_inta_out. Or it can be enabled as an input pin and connected to one of the PIC's intreq lines. This control is done via the Global 2 register Enable_IntA_O (see Section 8.3.2 on page 8-204).

8.3 Register Descriptions

The base address for the PNX8526 PCI-XIO module is 0x04 0000.

The following section describes the registers in the PCI-XIO block. The PCI configuration registers and the memory mapped IO registers are included.

8.3.1 Register Address Map

The following table is a summary of all the registers in this module.

Table 6: PCI-XIO Register Summary

Address	Name	Description
0x04 0000	pci_status	PCI Interrupt Status
0x04 0004	pci_int_mask	PCI Interrupt Enable
0x04 0008	pci_int_clr	PCI Interrupt Clear
0x04 000C	pci_int_set	PCI Interrupt Set
0x04 0010	pci_setup	PCI Setup register
0x04 0014	pci_control	PCI Control register
0x04 0018	pci_base1_lo	Internal view of external PCI bottom address, 1st aperture
0x04 001C	pci_base1_hi	Internal view of external PCI top address, 1st aperture
0x04 0020	pci_base2_lo	Internal view of external PCI bottom address, 2nd aperture
0x04 0024	pci_base2_hi	Internal view of external PCI top address, 2nd aperture
0x04 0028	read_lifetime	Length of time data is held exclusively for requesting agent.
0x04 002C	gppm_addr	General purpose PCI Master Cycle address register
0x04 0030	gppm_wdata	General purpose PCI Master Cycle write data register
0x04 0034	gppm_rdata	General purpose PCI Master Cycle read data register
0x04 0038	gppm_ctrl	General purpose PCI Master Cycle control register
0x04 003C	Reserved	
0x04 0040	device/vendorid	Image of device id and vendor id (configuration reg. 00)
0x04 0044	config_cmd_stat	Image of configuration command and status register (configuration reg 04)
0x04 0048	class code/rev id	Image of class code and revision id (configuration reg. 08)
0x04 004C	latency timer	Image of latency timer, cache line size (configuration reg. 0C)
0x04 0050	base10	Image of configuration base address 10 (configuration reg. 10)
0x04 0054	base14	Image of configuration base address 14 (configuration reg. 14)
0x04 0058	base18	Image of configuration base address 18 (configuration reg. 18)
0x04 005C—0068	Reserved	
0x04 006C	subsystem ids	Subsystem id, subsystem vendor id (configuration reg. 2C)
0x04 0070	Reserved	
0x04 0074	cap_pointer	Image of capabilities pointer (configuration reg. 34)
0x04 0078	Reserved	
0x04 007C	config_misc	Image of interrupt line, and interrupt line registers (configuration reg. 3C)
0x04 0080	pmc	Power management capabilities (configuration reg. 40)
0x04 0084	pwr_state	Power Management control (configuration reg. 44)
0x04 0088—4 07FC	Reserved	
0x04 0800	dma_eaddr	PCI address for DMA transaction.

Table 6: PCI-XIO Register Summary *…Continued*

Address	Name	Description
0x04 0804	dma_iaddr	Internal address for DMA transaction
0x04 0808	dma_length	DMA length in words
0x04 080C	dma_ctrl	DMA control
0x04 0810	xio_ctrl	XIO misc control
0x04 0814	xio_sel0_prof	XIO sel0 profile
0x04 0818	xio_sel1_prof	XIO sel1 profile
0x04 081C	xio_sel2_prof	XIO sel2 profile
0x04 0820	gpxio_addr	Indirect general purpose XIO address
0x04 0824	gpxio_wdata	Indirect general purpose XIO write data
0x04 0828	gpxio_rdata	Indirect general purpose XIO read data
0x04 082C	gpxio_ctrl	Indirect general purpose XIO control
0x04 0830	nand_ctrls	NAND-Flash profile controls
0x04 0834—0FF8	Reserved	
0x04 0FFC	module_id	DVP Module ID

…Continued

Table 7: PCI Configuration Register Summary

Address	Name	Description
0x0000	Device / Vendor ID	Device ID and Vendor ID
0x0004	Command / Status	Command and Status register
0x0008	Class Code/Rev ID	The Class code to be specified appropriate for the application. This will be implemented as a parameter. The Rev ID will initially be 0.
0x000C	Latency Timer/ Cache Line size	Latency Timer, Cache Line Size.
0x0010	Base Address 10	Base Address, memory
0x0014	Base Address 14	Base Address, memory — MMIO
0x0018	Base Address 18	Base Address, memory — XIO
0x001C—0028	Reserved	
0x002C	Subsystem ID	Subsystem ID and Subsystem Vendor ID
0x0030	Reserved	
0x0034	Capability Pointer	Capabilities Pointer Register
0x0038	Reserved	
0x003C	INTR	Interrupt Line, Interrupt Pin, Min_Gnt, MAX_Lat
0x0040	pmc	Power management Capability
0x0044	pwr_state	Power Management control

Table 8: PCI-XIO Global 2 Register Summary

Address	Name	Description
0x04 D050	PCI Inta Output Enable	Global 2 register

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
PCI Control Registers				
Offset 0x04 0000			*PCI Interrupt Status*	
31:21	R	0	Reserved	
20	R	0	pwrstate_chg	Power management register has been changed.
19	R	0	xio_acc_err	XIO access error. XIO or PCI master not enabled.
18	R	0	pim_rd_err	PI master read error detected.
17	R	0	pim_wr_err	PI master write error detected.
16	R	0	pis_rd_err	PI slave read error detected.
15	R	0	pis_wr_err	PI slave write error detected.
14	R	0	xio_ack_done	Rising edge of xio_ack has been observed.
13	R	0	gpxio_done	GPXIO transaction completed.
12	R	0	dma_done	DMA transaction completed.
11	R	0	serr_seen	SERR observed on PCI bus.
10	R	0	gppm_done	General purpose PCI command completed.
9	R	0	err_bad_cmd	Non-supported DMA command attempted or other error.
8	R	0	err_base10_subword	Subword attempt to base10 aperture when restrained to word only (not used on the PNX8526).
7	R	0	err_base14_subword	Subword attempt to base14 aperture when restrained to word only.
6	R	0	err_base18_subword	Subword attempt to base18 aperture when restrained to word only (not used on the PNX8526).
5	R	0	mstr_parity err	PCI master set or observed parity error (PERR).
4	R	0	err_parity	Detected parity error (PERR).
3	R	0	err_sys	Signaled system error (SERR)
2	R	0	err_r_mabort	Received Master Abort.
1	R	0	err_r_tabor	Received Target Abort.
0	R	0	err_s_tabort	Signaled Target Abort.
Offset 0x04 0004			*PCI Interrupt Enable*	
31:21	R	0	Reserved	
20	R/W	0	en_int_pwrstate_chg	Enable interrupt on change of power state register.
19	R/W	0	en_int_xio_acc_err	Enable interrupt on XIO access error.

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
18	R/W	0	en_int_pim_rd_err	Enable interrupt on PI master read error.
17	R/W	0	en_int_pim_wr_err	Enable interrupt on PI master write error.
16	R/W	0	en_int_pis_rd_err	Enable interrupt on PI slave read error.
15	R/W	0	en_int_pis_wr_err	Enable interrupt on PI slave write error.
14	R/W	0	en_int_xio_ack_done	Enable interrupt on rising edge of xio_ack done.
13	R/W	0	en_int_gpxio_done	Enable interrupt on gpxio_done.
12	R/W	0	en_int_dma_done	Enable interrupt on dma_done.
11	R/W	0	en_int_serr_seen	Enable interrupt on SERR observed on PCI bus.
10	R/W	0	en_int_gppm_done	Enable interrupt on general purpose PCI master cycle done.
9	R/W	0	en_int_bad_cmd	Enable interrupt on Bad-Command status.
8	R/W	0	en_int_base10_subword	Enable interrupt on subword attempt to Base10 error status.
7	R/W	0	en_int_base14_subword	Enable interrupt on subword attempt to Base14 error status.
6	R/W	0	en_int_base18_subword	Enable interrupt on subword attempt to Base18 error status.
5	R/W	0	en_int_mstr_parity err	Enable interrupt on Master Parity Error.
4	R/W	0	en_int_parity	Enable interrupt on Parity Error status.
3	R/W	0	en_int_err_sys	Enable interrupt on System Error status.
2	R/W	0	en_int_r_mabort	Enable interrupt on received Master Abort status.
1	R/W	0	en_int_r_tabort	Enable interrupt on received Target Abort status.
0	R/W	0	en_int_s_tabort	Enable interrupt on signaled Target Abort status.
Offset 0x04 0008			***PCI Interrupt Clear***	
31:21	R	0	Reserved	
20	W	0	clr_pwrstate_chg	Clear power state change register flag.
19	W	0	clr_xio_acc_err	Clear XIO access error.
18	W	0	clr_pim_rd_err	Clear PI master read error flag.
17	W	0	clr_pim_wr_err	Clear PI master write error flag.
16	W	0	clr_pis_rd_err	Clear PI slave read error flag.
15	W	0	clr_pis_wr_err	Clear PI slave write error flag.
14	W	0	clr_xio_ack_done	Clear xio_ack done flag.
13	W	0	clr_gpxio_done	Clear GPXIO done flag.
12	W	0	clr_dma_done	Clear dma_done flag.
11	W	0	clr_serr_seen	Clear serr_seen flag.
10	W	0	clr_gppm_done	Clear GPPM done flag.
9	W	0	clr_bad_cmd	Clear Bad-Command status.
8	W	0	clr_base10_subword	Clear subword attempt to Base10 error status.
7	W	0	clr_base14_subword	Clear subword attempt to Base14 error status.
6	W	0	clr_base18_subword	Clear subword attempt to Base18 error status.

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
5	W	0	clr_mstr_parity err	Clear Master Parity Error.
4	W	0	clr_parity	Clear Parity Error status.
3	W	0	clr_err_sys	Clear System Error status.
2	W	0	clr_r_mabort	Clear received Master Abort status.
1	W	0	clr_r_tabort	Clear received Target Abort status.
0	W	0	clr_s_tabort	Clear signaled Target Abort status.
Offset 0x04 000C			*PCI Interrupt Set*	
31:21	R	0	Reserved	
20	W	0	set_pwrstate_chg	Set change of power state register flag.
19	W	0	set_xio_acc_err	Set XIO access error.
18	W	0	set_pim_rd_err	Set PI master read error flag.
17	W	0	set_pim_wr_err	Set PI master write error flag.
16	W	0	set_pis_rd_err	Set PI slave read error flag.
15	W	0	set_pis_wr_err	Set PI slave write error flag.
14	W	0	set_xio_ack_done	Set xio_ack done flag.
13	W	0	set_gpxio_done	Set GPXIO done flag.
12	W	0	set_dma_done	Set dma_done flag.
11	W	0	set_serr_seer	Set serr_seen flag.
10	W	0	set_gppm_done	Set gppm_done flag.
9	W	0	set_bad_cmd	Set Bad-Command status.
8	W	0	set_base10_subword	Set subword attempt to Base10 error status.
7	W	0	set_base14_subword	Set subword attempt to Base14 error status.
6	W	0	set_base18_subword	Set subword attempt to Base18 error status.
5	W	0	set_mstr_parity err	Set master Parity Error.
4	W	0	set_parity	Set Parity Error status.
3	W	0	set_err_sys	Set System Error status.
2	W	0	set_r_mabort	Set received Master Abort status.
1	W	0	set_r_tabort	Set received Target Abort status.
0	W	0	set_s_tabort	Set signaled Target Abort status.
Offset 0x04 0010			*PCI Setup*	
This register must be initialized before any PCI cycles will be entertained. The boot loader is expected to load the values at boot time. Write once by boot loader, otherwise read only. Because this register is “written once” the bit fields are designated “R/W1.”				
31:28	R	0	Reserved	
27	R/W1	1	d2_support	Support for D2 power state
26	R/W1	1	d1_support	Support for D1 power state
25	R/W1	0	Reserved	

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
24	R/W1	0	en_ta	Terminate restricted access attempt with target abort (otherwise, ignore writes, return 0 on read).
23	R/W1	0	en_pci2mmi	Enable memory hwy interface.
22	R/W1	0	en_xio	Enable XIO functionality.
21	R/W1	0	base18_prefetchable	PCI base address 18 is a prefetchable memory aperture.
20:18	R/W1	011	base18_siz	The size of aperture located by PCI cfg base 18 is: 011 = 16 MB 100 = 32 MB 101 = 64 MB 110 = 128 MB This aperture is used as the XIO aperture in the PNX8526.
17	R/W1	1	en_base18	Enable 3rd aperture, PCI base address 18 (non-PNX8526 application). The PNX8526 will always use this aperture.
16	R/W1	0	base14_prefetchable	PCI Base address 14 is a prefetchable memory aperture.
15	R	0	Reserved	
14:12	R/W1	000	base14_siz	The size of aperture located by PCI cfg base 14 is: 000 = 2 MB or 32 if IO This aperture is used as the MMIO aperture in the PNX8526.
11	R/W1	1	en_base14	Enable 2nd aperture, PCI base address 14 (non-PNX8526 application). The PNX8526 will always use this aperture.
10	R/W1	1	base10_prefetchable	PCI Base address 10 is a prefetchable memory aperture.
9:7	R/W1	100	base10_siz	The size of aperture located by PCI cfg base 10 is: 011 = 16 MB 100 = 32 MB 101 = 64 MB This aperture is used as the DRAM aperture in the PNX8526.
6:2			Reserved	
1	R/W1	0	en_config_manag	Enable configuration management.
0	R/W1	0	en_pci_arb	Enable PCI system arbitration.
Offset 0x04 0014			*PCI Control*	
31:10	R	0	Reserved	
9	R/W	0	en_serr_seen	Enable monitoring of the SERR pin.
8	R/W	0	en_pi_wp	Enable write-posting from PI-Bus.
7	R/W	0	en_wrlock	Enables locked PI-write transactions originating from PCI. If this bit is ‘0’ the bridge breaks down a burst from the PCI side into single WDU writes on the PI side.
6	R/W	0	en_base10_spec_rd	Read ahead to optimize PCI read latency to base 10.
5	R/W	0	en_base14_spec_rd	Read ahead to optimize PCI read latency to base 14.
4	R/W	0	en_base18_spec_rd	Read ahead to optimize PCI read latency to base 18.
3	R/W	0	disable_subword2_10	Disable subword access to/from Base10 aperture.

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2	R/W	1	disable_subword2_14	Disable subword access to/from Base14 aperture.
1	R/W	1	disable_subword2_18	Disable subword access to/from Base18 aperture.
0	R/W	1	en_retry_timer	Enables timer for 16 tic rule enforcer. This bit does not affect access to the XIO aperture.
Offset 0x04 0018			*PCI_Base1_lo*	
31:21	R/W	0	pci_base1_lo	For internal address decoding: low bar of first aperture for external PCI access. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended that the PCI_Base1_lo be initialized before the PCI_Base1_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space.
20:0	R	0	Reserved	
Offset 0x04 001C			*PCI_Base1_hi*	
31:21	R/W	0	pci_base1_hi	For internal address decoding: high bar of first aperture for external PCI access (up to but not including). This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended the PCI_Base1_lo be initialized before the PCI_Base1_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space.
20:0	R	0	Reserved	
Offset 0x04 0020			*PCI_Base2_lo*	
31:21	R/W	0	pci_base2_lo	For internal address decoding: low bar of second aperture for external PCI access. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended the PCI_Base2_lo be initialized before the PCI_Base2_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space.
20:0	R	0	Reserved	
Offset 0x04 0024			*PCI_Base2_hi*	
31:21	R/W	0	pci_base2_hi	For internal address decoding: high bar of second aperture for external PCI access (up to but not including). This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended the PCI_Base2_lo be initialized before the PCI_Base2_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space.
20:0	R	0	Reserved	
Offset 0x04 0028			*Read Data Lifetime Timer*	
31:16		-	Unused	

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:0	R/W	8000	read_lifetime	This register is the amount of time (in PCI clocks) that the PCI will hold a piece of data exclusively for an external PCI master. The timer is initiated when the PCI can not complete the requested read in 16 clock cycles and issues a retry.
Offset 0x04 002C			***General Purpose PCI Master (GPPM) Address***	
31:0	R/W	0	gppm_addr	This register will be written with the address for the single data phase cycle to be issued on the PCI bus. It will accept only 32-bit writes. When issuing type 0 configuration transactions, the device number (bits [15:11]) is expanded to bits [31:11] on the PCI bus.
Offset 0x04 0030			***General Purpose PCI Master (GPPM) Write Data***	
31:0	R/W	0	gppm_wdata	This register will be written with the data for the single data phase cycle to be issued on the PCI bus. This register will accept any size write.
Offset 0x04 0034			***General Purpose PCI Master (GPPM) Read Data***	
31:0	R	0	gppm_rdata	This register will hold data from the selected target after completion of the read.
Offset 0x04 0038			***General Purpose PCI Master (GPPM) Control***	
31:11	R	0	Reserved	
10	R	0	gppm_done	1 = cycle has completed. This bit can also be viewed in the pci_status register. Write to register 0008 to clear.
9	R/W	0	init_pci_cycle	1 = initiate a PCI single data phase transaction on the PCI bus with address “gppm_addr” and data “gppm_data.”
8	R	0	Reserved	
7:4	R/W	0	gppm_cmd	Command to be used with PCI cycle. The acceptable commands to use in the command field include IO read, IO write, memory Read, memory Write, configuration read and interrupt acknowledge. If configuration management is enabled, configuration write may be used.
3:0	R/W	0	gppm_ben	Byte enables to be used with PCI cycle
Offset 0x04 0040			***Image of Device ID and Vendor ID***	
31:16	R	0x8500	device_id	PCI configuration device ID
15:0	R	0x1131	vendor_id	PCI configuration vendor ID
Offset 0x04 0044			***Image of Command/Status***	
31:16	R	0x0290	status	PCI configuration status register
15:0	R/W*	0x0000	command	PCI configuration command register. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only. Refer to configuration register 4 for details on which bit are implemented and are controllable.
Offset 0x04 0048			***Image of Class Code/Revision ID***	
31:8	R/W1*	040000	class code	PCI configuration class code. *Write-once/Read-only

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7:0	R	0	revision id	PCI configuration revision ID
Offset 0x04 004C			*Image of Latency Timer/Cache Line Size*	
31:24	R	0	BIST	PCI configuration BIST
23:16	R	0	Header Type	PCI configuration Header Type
15:8	R/W*	0	latency timer	PCI configuration latency timer. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
7:0	R/W*	0	cache line size	PCI configuration cache line size. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
Offset 0x04 0050			*Base Address 10 Image*	
31:4	R/W*	0	Base Address 10	PCI configuration Base address for DRAM. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
3	R	cfg	Prefetchable	Value is determined at boot time.
2:0	R	0	Type	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x04 0054			*Base Address 14 Image*	
31:4	R/W*	0	Base Address 14	PCI configuration Base address for MMIO. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
3	R	cfg*	Prefetchable	Value is determined at boot time. *Value determined by PCI Setup register.
2:0	R	0	Type	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x04 0058			*Base Address 18 Image*	
31:4	R/W*	0	Base Address 18	PCI configuration Base address for XIO. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
3	R	cfg	Prefetchable	Value is determined at boot time.
2:0	R	0	Type	Indicates PCI “type 0” memory space (locatable anywhere in 32-bit address space).
Offset 0x04 006C			*Subsystem ID/Subsystem Vendor ID Write Port*	
This register must be initialized before any PCI cycles will be entertained. The boot loader is expected to load the values at boot time. This register is a Write-once/Read-only register (R/W1).				

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:16	R/W1	0	subsystem ID	This is the write port for the Subsystem ID (PCI config 2C).
15:0	R/W1	0	subsystem vendor ID	This is the write port for the Subsystem Vendor ID (PCI config 2C).
Offset 0x04 0070			***Reserved***	
Offset 0x04 0074			***Image of Configuration Reg 34***	
31:8	R	0	Reserved	
7:0	R	40	CAP_PTR	Capabilities Pointer
Offset 0x04 007C			***Image of Configuration Reg 3C***	
31:24	R	0x18	max_lat	Max Latency
23:16	R	0x09	min_gnt	Minimum Grant
15:8	R	0x01	interrupt pin	Interrupt pin information
7:0	R/W*	0x00	Interrupt Line	This register conveys interrupt line routing information. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
Offset 0x04 0080			***Image of Configuration Reg 40***	
31:27	R	00000	Reserved	
26	R	cfg*	d2_support	1 = Device supports D2 power management state. *Value determined by PCI Setup register.
25	R	cfg*	d1_support	1 = Device supports D1 power management state. *Value determined by PCI Setup register.
24:19	R	0	Reserved	
18:16	R	010	version	Indicates compliance with version 1.1 of PM.
15:8	R	00	Next Item Pointer	There are no other extended capabilities.
7:0	R	01	Cap_ID	Indicates this is power management data structure.
Offset 0x04 0084			***Image of Configuration Reg 44***	
31:2	R	0	Reserved	
1:0	R/W*	0	pwr_state	Power State *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
Offset 0x04 0800			***DMA PCI Address***	
This register will accept only word writes.				
31:0	R/W	1c00_0000	dma_eaddr	This is the external starting address for the DMA engine. It is used for DMA transfers over PCI and XIO. Bit 0 and 1 are not used because all DMA transfers are word aligned.
Offset 0x04 0804			***DMA Internal Address***	
This register will accept only word writes.				
31:0	R/W	0010_0000	dma_iaddr	This is the internal read source/ write destination address in SDRAM.
Offset 0x04 0808			***DMA Transfer Size***	
This register will accept any size writes.				

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:16	R/W	0	Reserved	
15:0	R/W	800	dma_length	This is the length of the DMA transfer (number of 4-byte words).
Offset 0x04 080C			***DMA Controls***	
This register will accept any size writes.				
31:10	R	0	Reserved	
9	R/W	0	snd2xio	0 = DMA will target PCI. 1 = DMA will target XIO.
8	R/W	0	fix_addr	0 = DMA will use linear address. 1 = DMA will use a fixed address (for XIO).
7:5	R/W	0	max_burst_size	PCI transaction will be split into multiple transactions. Max size: 000 = 8 data phase 001 = 16 data phase 010 = 32 data phase 011 = 64 data phase 100 = 128 data phase 101 = 256 data phase 110 = 512 data phase 111 = No restriction in transfer length
4	R/W	0	init_dma	Initiate DMA transaction. This bit is cleared by the DMA engine when it begins its operation.
3:0	R/W	0	cmd_type	Command to be used for DMA. This field is restricted to memory type commands.
Offset 0x04 0810			***XIO Control Register***	
31:2	R	0	Reserved	
1	R		xio_ack	Live XIO_ACK status bit.
0	R/W	0	Reserved	
Offset 0x04 0814			***XIO Sel0 Profile***	
This register sets up the profile of the XIO select 0 line. All times are in reference to PCI clocks.				
31:23	R	0	Reserved	
22	R/W	0	sel0_use_ack	0 = Fixed wait state 1 = Wait for ACK Not used for IDE.
21:18	R/W	0	sel0_we_hi	68360: DS time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	R/W	0	sel0_we_lo	68360: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13:9	R/W	0	sel0_wait	68360: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	R/W	0	sel0_offset	Starting address offset from start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	R/W	0	sel0_type	Device type selected: 00 = 68360 type device 01 = NOR Flash 10 = NAND-Flash 11 = IDE
2:1	R/W	0	sel0_siz	Amount of address space allocated to Sel0: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	R/W	0	en_sel0	1 = Enable sel0 profile.
Offset 0x04 0818			***XIO Sel1 Profile***	
This register sets up the profile of the XIO select 1 line. All times are in reference to PCI clocks.				
31:23	R	0	Reserved	
22	R/W	0	sel1_use_ack	1 = Wait for ACK 0 = fixed wait state. Not used for IDE.
21:18	R/W	0	sel1_we_hi	63860: time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	R/W	0	sel1_we_lo	63860: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	R/W	0	sel1_wait	63860: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	R/W	0	sel1_offset	Address offset form start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	R/W	0	sel1_type	Sel1 is configured as: 00 = 68360 type device 01 = NOR Flash 10 = NAND-Flash 11 = IDE

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:1	R/W	0	sel1_siz	Amount of address space allocated to Sel1: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	R/W	0	en_sel1	Enable sel1 profile.
Offset 0x04 081C			*XIO Sel2 Profile*	
This register sets up the profile of the XIO select 2 line. All times are in reference to PCI clocks.				
31:23	R	0	Reserved	
22	R/W	0	sel2_use_ack	0 = Fixed wait state. 1 = Wait for ACK Not used for IDE
21:18	R/W	0	sel2_we_hi	68360: DS time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	R/W	0	sel2_we_lo	63860: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	R/W	0	sel2_wait	68360: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	R/W	0	sel2_offset	Address offset form start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	R/W		sel2_type	Sel2 is configured as: 00 = 68360 type device 01 = NOR Flash 10 = NAND-Flash 11 = IDE
2:1	R/W	0	sel2_siz	Amount of address space allocated to Sel2: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	R/W	0	en_sel2	Enable sel2 profile.
Offset 0x04 0820			*GPXIO_address*	
31:0	R/W	0	gpxio_addr	General Purpose XIO cycle address. This register sets the address for an indirect read or write to/from XIO address space. Only 4 byte writes are allowed in this register. The values programmed for bits 0 and 1 are not used by the XIO module. Refer to gpxio_ben.

PCI-XIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 0824			*GPXIO_write_data*	
31:0	R/W	0	gpxio_wdata	General Purpose XIO cycle data. This register is programmed with data for a write cycle.
Offset 0x04 0828			*GPXIO_read_data*	
31:0	R	0	gpxio_rdata	General Purpose XIO cycle data. This register contains the data of a read cycle after completion.
Offset 0x04 082C			*GPXIO_ctrl*	
This register controls the type of access to XIO and provides status.				
31:11	R	0	Reserved	
10	W	0	set_gpxio_done	Set gpxio_done.
9	R	0	gpxio_cyc_pending	1 = GPXIO transaction on XIO is pending.
8	R	0	gpxio_done	General Purpose XIO cycle complete. This bit may also be set by writing 1 to bit 10 (set_gpxio_done). This bit is cleared by writing 1 to bit 6 or 7.
7	R/W	0	clr_gpxio_done	1 = Clear “gpxio_done.”
6	R/W	0	gpxio_init	1 = Initiate a transaction on XIO. The type of transaction will match the profile of the selected aperture. This bit gets cleared if the cycle has been initiated. This bit clears bit 8 if set.
5	R/W	0	gpxio_int	Generate interrupt on completion of XIO cycle.
4	R/W	0	gpxio_rd	Read/not write command on XIO
3:0	R/W	0	gpxio_ben	Active low byte enables to be used on the indirect XIO cycle. These are used to determine how many bytes to access and the lower two address bits for use in “gpxio_addr”.
Offset 0x04 0830			*NAND-Flash controls*	
21:16	R/W	17	nand_ctrls	This field controls the type of NAND-Flash access cycle. The bits are defined as follows: [21]: 1 = Enable access to spare area; 0: access normal area [20]: 1 = Include data in access cycle; 0 access does not include data phase(s) [19:18] = Number of commands to be used in NAND-Flash access [17:16] = Number of address phases to be used in NAND-Flash access.
15:8	R/W	0	command_b	This is the second command for NAND-Flash when two commands are required to complete a cycle.
7:0	R/W	0	command_a	This is the command type to be used with NAND-Flash cycles when one or more commands are required to complete a cycle.
Offset 0x04 0FFC			*Module ID*	
31:16	R	0x0113	Module ID	PNX8526 Module ID
15:12	R	0	Major Revision number	Major Revision number
11:8	R	1	Minor revision number	Minor revision number
7:0	R	0	mod_size	Module size is 4 kB.

PCI CONFIGURATION REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x0000			*Device ID/Vendor ID*	
31:16	R	0x8500	Device ID	The ID assigned by the PCI SIG representative. The value will be hard coded. Reset value 8500 has been assigned to the PNX8526.
15:0	R	0x1131	Vendor ID	Value 0x1131 is the ID assigned to Philips Semiconductors by the PCI SIG representative.
Offset 0x0004			*Command/Status*	
31	R/W	0	Parity Error	This bit will be set whenever the device detects a parity error. Write 1 to clear.
30	R/W	0	Signaled System Error	This bit is set whenever the device asserts SERR. Write 1 to clear.
29	R/W	0	Received Master Abort	Set by the PCI master when its transaction is terminated with a master abort. Write 1 to clear.
28	R/W	0	Received Target Abort	Set by the PCI master when its transaction is terminated with a target abort. Write 1 to clear.
27	R/W	0	Signaled Target Abort	Set by the PCI target when it terminates a transaction with a target abort. Write 1 to clear.
26:25	R	01	Devsel Timing	The PCI target uses medium DEVSEL timing.
24	R/W	0	Master Data Parity Error	Set by the PCI master when PERR is observed.
23	R	1	Fast Back-to-Back Capable	The PCI supports fast back-to-back transactions.
22	R	0	Reserved	
21	R	cfg*	66 MHz Capable	0 = 33 MHz PCI (PNX8526 is 33 MHz). *Value determined by PCI Setup register.
20	R	1	Capabilities List	Indicates a new Capabilities linked list is available at offset 40h.
19:10	R	0000	Reserved	
9	R/W	0	Fast back-to-back enable	Enable fast back-to-back transactions for PCI master.
8	R/W	0	SERR enable	Enable SERR to report system errors.
7	R	0	Stepping Control	Address stepping is not supported.
6	R/W	0	Parity Error Response	0 = No parity error response 1 = Enable parity error response.
5	R	0	VGA Palette Snoop	VGA is not supported.
4	R/W	0	Memory Write & Invalidate	Enable use of memory write and invalidate.
3	R	0	Special Cycles	Special cycles are not supported.
2	R/W	0	Enable Bus Master	Enable the PCI bus master.
1	R/W	0	Enable Memory space	Enable all memory apertures.
0	R	0	Enable IO Space	The PNX8525 does not respond to IO transactions.

PCI CONFIGURATION REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x0008			*Class Code/Revision ID*	
31:8	R/W*	0x040000	Class Code	The PNX8526 is defined as a multimedia video device. *The boot loader may change the class code to an alternate value if done before writing to the pci_setup register.
7:0	R	1	Revision ID	Revision ID. Will initially be assigned to 0. Revision ID must not be synthesized. It will need to be changed with revised silicon, whether for bug fixes or enhancements.
Offset 0x000C			*Latency Timer/Cache Line Size*	
31:16	R	0x0000	Reserved	Note: BIST is not implemented. Header is 0.
15:8	R/W	0	Latency Timer	Latency Timer
7:0	R/W	0	Cache Line Size	Cache Line Size
Offset 0x0010			*Base10 Address Register*	
This aperture is for the SDRAM on the PNX8526. 64M, 32M or 16M memory sizes are supported on the PNX8526.				
31:28	R/W	0	Base10 Address	Upper 4 bits of base10 address of the first memory aperture
27:21	R/W*	0	Base10 Address	*The base 10 can be configured to various aperture sizes from 2 MB to 256 MB. (See PCI Setup register on page 8-190). Depending on aperture size selected, various bits will be R/W or Read Only. Bit: 27262524232221 256M:RORORORORORORO 128M:RWRORORORORORO 64M:RWRWRORORORORO 32M:RWRWRWRORORORO 16M:RWRWRWRWRORORO 8M:RWRWRWRWRWRORO 4M:RWRWRWRWRWRWRO 2M:RWRWRWRWRWRWRW
20:4	R	0	Reserved	
3	R	cfg	Prefetchable	Value is determined at boot time.
2:0	R	0	Type	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x0014			*Base14 Address Register*	
This aperture will be set to 2 MB for MMIO on the PNX8526.				
31:28	R/W	0001	Base14 Address	Upper 4 bits of base14 address of the first memory or IO aperture

PCI CONFIGURATION REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
27:21	R/W*	1011111	Base14 Address	*The base 14 can be configured to various aperture sizes from 2 MB to 256 MB. (See PCI Setup register on page 8-190). Depending on aperture size selected, various bits will be R/W or Read Only. **Bit: 27262524232221** 256M:ROROROROROROR O 128M:RWROROROROROR O 64M:RWRWROROROROR O 32M:RWRWRWROROROR O 16M:RWRWRWRWROROR O 8M:RWRWRWRWRWROR O 4M:RWRWRWRWRWRWRO 2M:RWRWRWRWRWRWRW
20:4	R	0	Reserved	
3	R	cfg	Prefetchable	Value is determined at boot time.
2:0	R	0	Type	Type of aperture. 0 = PCI type 0 1 = PCI type 1 Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x0018			***Base18 Address Register***	
This aperture is for the XIO on the PNX8526, which supports up to 128 MB of XIO memory space.				
31:28	R/W	0001	Base18 Address	Upper 18 bits of base address of the first memory or IO aperture
27:21	R/W*	1100000	Base18 Address	*The base 18 can be configured to various aperture sizes from 2 MB to 256 MB. (See PCI Setup register on page 8-190). Depending on aperture size selected, various bits will be R/W or Read Only. **Bit: 27262524232221** 256M:RORORORORORORO 128M:RWRORORORORORO 64M:RWRWRORORORORO 32M:RWRWRWRORORORO 16M:RWRWRWRWRORORO 8M:RWRWRWRWRWRORO 4M:RWRWRWRWRWRWRO 2M:RWRWRWRWRWRWRW
20:4	R	0	Reserved	
3	R	cfg*	Prefetchable	Prefetchable if configured as 1. *Value determined by PCI Setup register.
2:0	R	0	Memory	This bit indicates type 0 memory aperture.
Offset 0x002C			***Subsystem ID/Subsystem Vendor ID***	
The values used in this register will be loaded into the register before entertaining any transactions on the PCI bus. The boot loader will initialize control register address 0x006C with the correct values.				
31:16	R	0	Subsystem ID	Subsystem ID. The value for this field is provided by Philips PCI SIG representative for Philips internal customers. External customers will provide their own number.

PCI CONFIGURATION REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:0	R	0	Subsystem Vendor ID	Subsystem Vendor ID. The value for this field is 1131 for Philips internal customers. External customers need to apply to the PCI SIG to obtain a value if they do not have one already.
Offset 0x0030			***Reserved***	
Offset 0x0034			***Capabilities Pointer***	
31:8	R	0	Reserved	
7:0	R	0x40	cap_pointer	Indicates extended capabilities are present starting at 40.
Offset 0x003C			***Max_Lat, Min_Gnt, Interrupt pin, Interrupt Line***	
31:24	R	0x18	max_lat	Indicates the max latency tolerated in 1/4 microsecond for PCI master.
23:16	R	0x09	min_gnt	Indicates how long the PCI master will need to use the bus.
15:8	R	0x01	interrupt_pin	Indicates which interrupt pin is used.
7:0	R/W	0x00	interrupt_line	Interrupt routing information
Offset 0x0040			***Power Management Capabilities***	
31:27	R	0x0000	Reserved	
26	R	cfg*	d2_support	1 = Device supports D2 power management state *Value determined by PCI Setup register.
25	R	cfg*	d1_support	1 = Device supports D1 power management state *Value determined by PCI Setup register.
24:19	R	0	Reserved	
18:16	R	010	version	Indicates compliance with version 1.1 of PM.
15:8	R	00	Next Item Pointer	There are no other extended capabilities.
7:0	R	01	Cap_ID	Indicates this is power management data structure.
Offset 0x0044			***PMCSR***	
31:2	R		Reserved	
1:0	RW		pwr_state	power_state

8.3.2 Global 2 Registers

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
PCI Inta Output Enable Register				
Offset 0x04 D050			*ENABLE_INTA_O*	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x0	ENABLE_INTA_O	Enables PCI INTA output. 0 :Disable PCI inta output. 1 :Enable PCI inta output. Reset is to be disabled.

Chapter 9: Main Memory Interface (MMI)

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

9.1 Introduction

This chapter describes the SDRAM controller of the PNX8526. This controller is called the Main Memory Interface (MMI). The MMI is the interface between internal PNX8526 devices and off-chip Synchronous DRAM memory. The MMI Arbiter is described separately in Chapter 36 MMI Arbiter.

Throughout this chapter functional *timing* diagrams are used in a plain ascii-based format. A timing diagram may look like the following:

enable	___111
data	VVV
data2	012345
address	AAABBB

This timing diagram represents one signal per line, with the signal name listed on the left side. All signals are synchronous with a clock. The ASCII characters represent the value of the signal over time using one character per clock cycle to represent the value in that clock cycle that is sampled on the next clock edge. The characters mean the following:

low value	‘_’
high value	‘1’
don’t care or undefined	space
a valid value	‘V’
corresponding data beat of a burst data transfer	number
value belonging to transaction ‘A’ or ‘B’ or similar	‘A’ or ‘B’ or similar

The MMI connects the PNX8526 to SDRAM memory over a 64-bit bus operating at up to 166 MHz. This provides a peak bandwidth of 1.33 GB.

The memory controller uses efficient interleaving, pipelining and overlapping of memory transactions to achieve high-sustained bandwidth utilization. The latency of the SDRAM controller (excluding arbitration delay) is minimal by using a zero-cycle latency design. This helps optimize the performance of CPUs.

The memory controller has special support for MPEG video, allowing two-dimensional block transfers for MPEG motion compensation and image construction with high bus bandwidth utilization. There are two formats for storing data in memory; one optimized for MPEG video, the other optimized for linear transfers.

The controller includes various convenient features such as automatic initialization, refresh, power management, memory protection and support for a variety of memory configurations.

9.2 SDRAM Configurations

9.2.1 Supported Chips and Configurations

The MMI supports Jedec-standard SDRAM of the 64-Mbit and 128-Mbit generations. All SDRAM chips must have 4 banks per chip. The 4 banks are interleaved to obtain high bandwidth utilization. Therefore, 2 bank versions of SDRAM are not supported.

The PNX8526 uses 3.3V LVTTL I/O pins and the SDRAM chips must be compatible with this.

At most, 4 SDRAM chips can be connected to operate the system at full speed. No board level buffers or glue logic are necessary.

The SDRAM memory consists of a single rank of memory chips on a 64-bit wide data bus. Several memory chips must be connected in parallel to fill the 64-bit bus width.

There are three possible memory footprints supported by the PNX8526: 16 MB, 32 MB and 64 MB.

The following four memory configurations are possible:

Table 1: Supported Memory Configurations

Size (MB)	# Chips	Type of Chip
16	2	2M by 32 SDRAM (64 Mbits)
32	4	4M by 16 SDRAM (64 Mbits)
32	2	4M by 32 SDRAM (128 Mbits)
64	4	8M by 16 SDRAM (128 Mbits)

9.2.2 Detailed SDRAM Specifications

This section describes the logic and timing specs that SDRAM devices must comply with to be supported.

Logically, the SDRAM devices must follow the "PC100 SDRAM" specification published on the Intel® website.

SDRAM timing details differ between vendors, part types and speed grades. The SDRAM controller is intended to accommodate the timing specs of almost all parts/vendors. The following lists timing/functionality assumptions that the MMI relies on.

Remark: Timing numbers have been selected to emphasize the use of SDRAM devices with high clock speed, rather than SDRAM devices with low latency.

Table 2: Specs of Supported SDRAM

SDRAM Specs	Value
Burst length	Full page burst length is used for all transfers
CAS latency	3 cycles
tRC: RAS cycle time	10 cycles
tRAS: RAS to precharge delay	7 cycles
tRCD: RAS to CAS delay	3 cycles
tRRD: RAS to RAS delay	2 cycles
tRP: Precharge time	3 cycles
tCCD: CAS to CAS delay	1 cycle
tRFC: Refresh to RAS delay	12 cycles
tRDL: Last write data into row precharge	2 cycles

The precharge command may be used to terminate read and write bursts. In case of a read burst, the precharge command must be followed by 2 more valid data beats before the DQ bus goes to hi-Z, as follows:

command	P
DQ bus	VVV---

In case of a write burst, the precharge command is issued 2 cycles after the last write data beat and DQM is asserted after the end of the data beat as follows:

command	P
DQ bus	VVV---
DQM	___11

The SDRAM controller creates bus turnaround cycles on the SDRAM data bus. At least 1 Hi-Z cycle is created on the SDRAM DQ bus when read switches to write or write switches to read.

9.3 SDRAM I/O Pins

This section discusses the chip I/O pins that the MMI uses to connect to the SDRAM. All pins use 3.3V LVTTL signal levels. Table 3 shows the following 93 signal pins:

Table 3: I/O Pin List for the SDRAM Bus

Pin Name	Type	I/O Buffer	Purpose
MM_CLK[1:0]	O		SDRAM clock 2 identical clock pins that share the load
MM_CKE	O		SDRAM clock enable
MM_CS#	O		SDRAM chip select
MM_RAS#	O		SDRAM RAS control
MM_CAS#	O		SDRAM CAS control
MM_WE#	O		SDRAM write enable
MM_A[11:0]	O		row/column address
MM_BA[1:0]	O		bank address
MM_DQ[63:0]	I/O		read/write data
MM_DQM[7:0]	O		byte write enable

Remark: The MM_DQ pins are bi-directional off-chip. On-chip, this signal is split into two uni-directional signals (MM_DQ_IN and MM_DQ_OUT) between the MMI and I/O pads. There is also a signal MM_DQ_ENABLE that controls the tri-state enable of the MM_DQ tri-state driver pads.

9.3.1 Pin Connections

Each of the control and address pins in Table 3 are connected to pins on the SDRAM chips with corresponding names. The MM_DQ and MM_DQM pins are split into equal groups and connected to different SDRAM chips. For instance, in a 16-MB system with 2 chips of 2M by 32, MM_DQ[31:0] and MM_DQM[3:0] are connected to DQ[31:0] and DQM[3:0] of one SDRAM chip and MM_DQ[63:32] and MM_DQM[7:4] are connected to DQ[31:0] and DQM[3:0] of the other chip.

The SDRAM clock has 2 output pins: MM_CLK[0] and MM_CLK[1]. Each pin needs to be connected to the CLK input of half of the SDRAM chips.

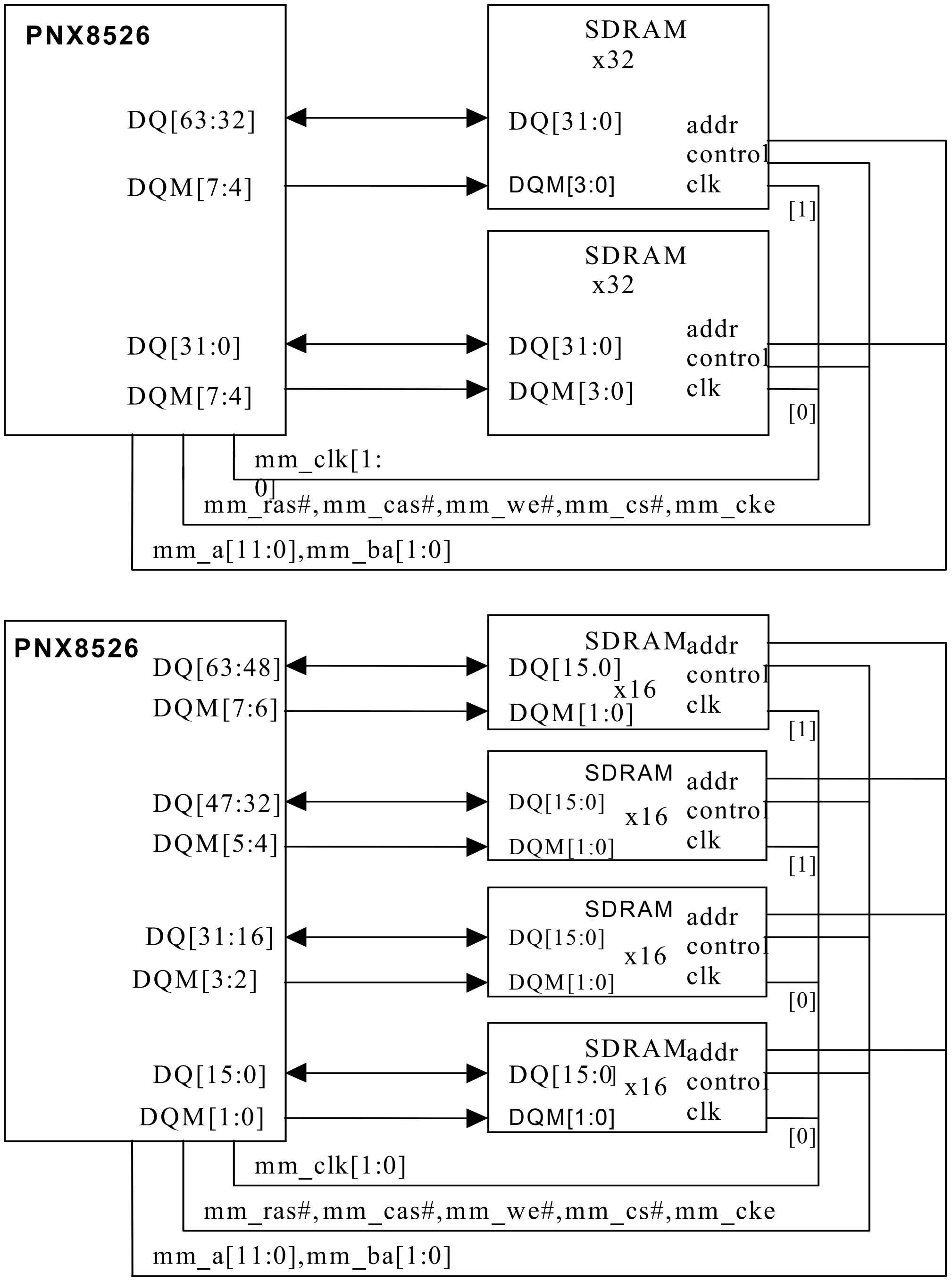


Figure 1: SDRAM Board Connections (PNX8526 Connection to 2 or 4 SDRAM Chips)

9.4 SDRAM Initialization

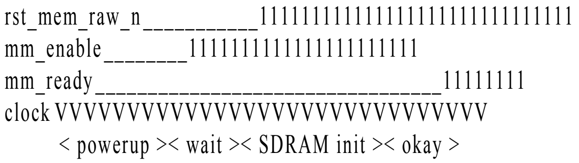
This section describes chip initialization as it relates to the SDRAM controller and SDRAM chips. The initialization sequence is as follows:

1. Power up.
   The system clock generators need to be turned on, and the clocks to the external SDRAM and on-chip SDRAM controller stabilized. The MMI reset signal *rst_mem_raw_n* must be asserted low and be asynchronously stable and glitch-free during power up i.e., when power starts to come up and clocks may or may not be running, the *rst_mem_raw_n* signal must be low.
2. Release SDRAM reset.
   The reset signal *rst_mem_raw_n* is low during power up and clock stabilization. After the power up is completed, the *rst_mem_raw_n* is asserted high. The MMI contains a synchronizer in the MMI clock domain for this signal.
3. Assert mm_enable
   After the reset has been de-asserted, the MMI waits for the signal mm_enable to be asserted. This triggers the SDRAM controller to start the SDRAM initialization procedure. Configuration signals (see below) and other input signals of the SDRAM controller must get their initial values before mm_enable is asserted. The clock to SDRAM MM_CLK needs to be stable for at least 200 μs before mm_enable is asserted.
4. SDRAM initialization.
   The SDRAM controller carries out the initialization of the SDRAM chips, which includes setting the SDRAM mode register to the value: {5'b0, 3'b011, 1'b0, 3'b111}.
   This means CAS latency=3, wrap type=linear, burst length=full page.
5. SDRAM ready.
   When the SDRAM initialization is completed, the mm_ready signal is asserted by the SDRAM controller. During reset, power up, and initialization, this signal is kept de-asserted.

When the mm_ready signal is asserted, the SDRAM controller starts listening to the memory bus request signals and any other control signal inputs. Memory transactions can now start.

The mm_enable signal must be asserted for initialization to start. Otherwise initialization will not occur.

Table 4: Timing of SDRAM Initialization



```
rst_mem_raw_n__________1111111111111111111111111111111
mm_enable_______11111111111111111111111
mm_ready______________________________11111111
clock VVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
     < powerup >< wait >< SDRAM init >< okay >
```

9.5 SDRAM Refresh

The SDRAM controller supports automatic refresh of the SDRAM chips using the SDRAM CBR auto refresh mechanism. The SDRAM controller contains a refresh counter that counts SDRAM clock cycles between refresh operations. The counter is preset with a programmable refresh value so that software can control the refresh rate. The programmable refresh value is presented to the SDRAM controller as an input signal mm_refresh.

The refresh works as follows. Whenever the refresh counter expires, the current memory transaction is completed, all SDRAM banks are precharged, and one CBR auto refresh command is issued to all SDRAM chips simultaneously. This refreshes one row in all banks in all SDRAM chips. After a required waiting time, new memory transactions can start. Typically, the whole refresh operation takes about 20 SDRAM clock cycles.

When the refresh operation is completed, the refresh counter is loaded again with the value of the mm_refresh signal and count down for the next refresh operation starts.

The refresh counter is a 16-bit counter. The refresh initialization value is a 10-bit value that is loaded into the 10 most significant bits (msbits) of the refresh counter; the 6 lsbits are set to 0 when the refresh counter is loaded. In other words, the value of mm_refresh is multiplied by 64 before it's loaded into the refresh counter.

A signal mm_refresh_enabled can be used to turn SDRAM refresh on or off altogether. Normally, refresh should be on, but it can be useful to turn it off when the chip is exercised on a tester with a program that contains loops.

A signal mm_short_refresh tells the SDRAM controller to do a special, fast refresh sequence. In this mode it is assumed that the clock frequency of the system is so low that the SDRAM refresh can be completed in 1 clock cycle, and the total refresh operation takes only a few clock cycles. When the system operates at a very low speed, the mm_refresh value may need to be set to a very low value, and refresh will take up an unusually large chunk of the available SDRAM bandwidth. Doing a fast SDRAM sequence will help to reduce this penalty.

The value that needs to be programmed into mm_refresh is:
mm_refresh = REFRESH_INTERVAL / CLOCK_PERIOD / SCALING

in which REFRESH_INTERVAL is the time between refreshes, CLOCK_PERIOD is the SDRAM clock period and SCALING is the scaling factor 64. A refresh value of 0 should not be programmed; it would result in having a refresh period of 64 kB cycles.

Typically, SDRAM requires 4096 refreshes every 64 milliseconds (ms) so the REFRESH_INTERVAL is 15 μs.

If a 100 MHz system is used, mm_refresh is 15 μs / 10ns / 64 = 23. This is a good default value.

For a 133 MHz, 143 MHz or 166MHz system, mm_refresh must be 30,32 or 38 respectively.

OEM customers need to check what is required for the specific SDRAM brand used in the system and adjust the value accordingly. Also, adjustment needs to be made for the specific clock period used.

9.6 Powerdown and Self-Refresh

The SDRAM controller supports a low-power mode for the SDRAM called self-refresh.

Two internal signals implement the powerdown control protocol of the SDRAM controller: mm_enable (input) and mm_ready (output). These signals are also used during initialization of the MMI. (The mm_enable signal is controlled in the global 2 register module.)

To enter SDRAM self-refresh mode, the system must de-assert the mm_enable signal. This signal must be asserted during normal operation. When the SDRAM controller notices that the mm_enable signal is de-asserted, it will start to put the SDRAM into self-refresh. When completed, the SDRAM controller de-asserts the mm_ready signal, which is asserted during normal operation. If the mm_enable signal is asserted quickly again, before the mm_ready signal has gone down, the SDRAM will still be put into self-refresh for a minimum amount of time and the mm_ready will go low for some time. During this time no memory traffic will occur.

When the SDRAM is in the self-refresh state i.e., when the mm_ready signal and the mm_enable signal are both low, the clock to the SDRAM chips may be stopped by the system.

The SDRAM controller itself can also be powered down simply by stopping the clock feeding into the SDRAM control module. Whenever this clock is stopped, the SDRAM controller does not respond to any changes in its input signals, such as Memory Bus request signals. Whenever this clock is operating, but the SDRAM is in self-refresh mode, the SDRAM controller only listens to the mm_enable signal.

When the system wants to wake up SDRAM from self-refresh, it must first resume the clock feeding into the SDRAM controller, and then assert the mm_enable signal (this may happen in the same clock cycle). This will trigger the SDRAM controller to bring the SDRAM chips back up from self-refresh. When that is completed, the SDRAM controller asserts the mm_ready signal and memory transactions can resume.

The timing of the powerdown protocol is illustrated below:

Table 5: Timing of SDRAM Powerdown Signals

```
mm_enable 1111______________________________1111111111111111111111
mm_ready 1111111111111111111______________________________11111111
clock VVVVVVVVVVVVVVVVVVV          VVVVVVVVVVVVVVVVVVVVVVV
<go into    ><self-refresh on><come out of ><traffic>
<self-refresh><clock may stop ><self-refresh><okay   >
```

9.7 Defaults

9.7.1 Values for Address Bits

During operation of the SDRAM controller at various times, the address bits MM_A have “don’t care” values. In such cases the controller will minimize the changes in value of the address bits in order to minimize power dissipation.

9.7.2 Tri-State Bus Driver

During idle bus cycle (including during powerdown, boot and reset), the SDRAM controller must drive the tri-state SDRAM data bus to a value to avoid the bus floating for a long time.

A default driver mechanism is included in the SDRAM controller that takes care of this. During ongoing read or write bursts, the default driver is inactive. When no read data stream is present for a few cycles, the default driver kicks into action. The bus can be floating for up to 7 clock cycles. The default value that is driven onto the bus is constant high

```
64'hFFFFFFFFFFFFFFFF
```

When the default driver is active, DQM is also asserted.

9.8 Memory Protection

9.8.1 Architecture

The Main Memory Interface (MMI) unit in the PNX8526 contains a memory protection mechanism that enables privileged access to a memory region under software control.

The privileged memory region is demarcated by a set of base and bound registers: TM_REGION_LO and TM_REGION_HI. A memory access to an address outside the region between TM_REGION_LO and TM_REGION_HI is disabled by the MMI if it comes from a device that participates in the memory protection scheme. A memory access to an address between TM_REGION_LO and TM_REGION_HI is always allowed.

There is a TM_OWNED bit for each of the peripherals connecting directly to the Memory Bus. These are: ICP1, ICP2, MBSR, MBSW, VIP1, VIP2, VMPGL, VMPGR, VMPGW, TM32, D2D, D3D, PCI.

Remark: Protection for peripherals that connect through a PI-Bus and PIMI to the memory bus is handled by the PIMI, not by the MMI.

Whenever the TM_OWNED bit is set for a peripheral, the access outside of the region between TM_REGION_LO and TM_REGION_HI is protected.

The memory protection is depicted in Figure 2.

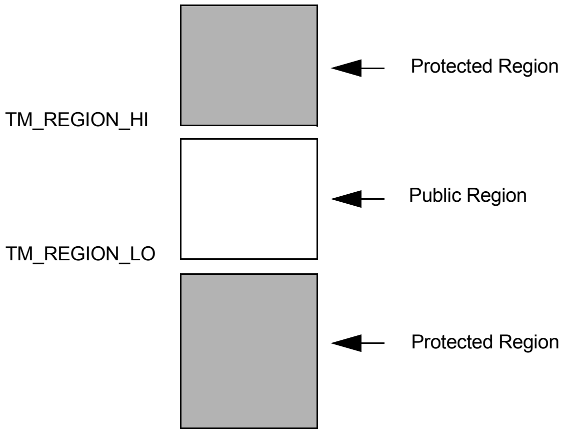


Figure 2: Memory Protection Regions in the Memory Address Space

9.8.2 Registers

The base and bound registers TM_REGION_LO and TM_REGION_HI control the size of the protection region. TM_REGION_LO and TM_REGION_HI must not cross a 64-MB boundary.

Values in TM_REGION_LO and TM_REGION_HI are 64-kB aligned. Both registers are 32-bit registers with the low order 16 bits fixed to 0 (i.e. ,writing to the 16 lower order bits has no effect, reading them returns the value 0).

The TM_REGION_LO and TM_REGION_HI register values are transferred to the MMI when the MMI is first enabled or when either register has been reprogrammed.

The value of TM_REGION_LO must be smaller than or equal to the value in TM_REGION_HI. If TM_REGION_LO > TM_REGION_HI, the operation of the protection mechanism is undefined.

When TM_REGION_LO == TM_REGION_HI then effectively the whole memory is protected.

TM_REGION_LO indicates the start of the region where access is always allowed; the region with address less than TM_REGION_LO is the lower protected region. TM_REGION_HI is the address where the upper protected region starts.

The protection regions are defined as follows:

```
if a device i does a memory access at address A[i], then
if (TM_OWNED[i] == 1)
 if (TM_REGION_LO <= A[i] < TM_REGION_HI)
 then access allowed
 else access denied
else access allowed
```

Unaligned memory transactions may be partially inside and partially outside the protected region. If that happens, the whole transaction is denied access, including the parts of the transaction that access an allowed address.

The protection algorithm for an unaligned transaction for device i is:

```
  if (TM_OWNED[i] == 1) {
  if (TM_REGION_LO <= A[i] < TM_REGION_HI)
  then
   if (TM_REGION_LO <= (A[i] + size[i] - size_of_word) <
TM_REGION_HI)
   then access allowed
   else access denied
  else access denied
  }
  else access allowed
```

9.8.3 Protection Behavior

When a protected region is accessed and protection activated, then the following happens:

- If the attempted memory operation is a memory write operation, then the write is suppressed and nothing is written to the memory.

- If the attempted memory operation is a memory read operation, then the value 0xdeadbeefdeadbeef is returned for each 64-bit word in the memory transaction.

9.9 Register Descriptions

In previous sections the MMI Control signals: mm_sdram_size, mm_refresh, mm_refresh_enabled, mm_short_refresh, tm_region_lo, tm_region_hi, tm_owned have been introduced.

These signals are associated with MMIO registers to allow software control of the configuration of the SDRAM controller. The MMIO registers are located in Global 2 registers. The base address for these registers is 0x04 D400.

The value that the configuration signals mm_sdram_size, mm_refresh, mm_refresh_enabled, mm_short_refresh have during boot are used by the SDRAM controller as the initial values during SDRAM initialization. These boot values must be fixed before the mm_enable signal is asserted and remain the same afterwards.

9.9.1 Register Address Map

Table 6: *SDRAM Controller* Register Summary (Global 2 Registers)

Offset	Name	Description
0x04 D400	MM_SDRAM_SIZE	Indicates SDRAM size.
0x04 D404	MM_REFRESH	Indicates the number of SDRAM clock cycles between refreshes scaled by 64.
0x04 D408	MM_SHORT_REFRESH	Turns on fast SDRAM refresh during emulation.
0x04 D410	MM_REFRESH_ENABLED	Turns SDRAM refresh on/off.
0x04 D410	MM_ENABLE_INTERLEAVE	Enables linear or interleave mode.
0x04 D414	MM_SELF_REFRESH	Controls SDRAM self-refresh.
0x04 D418	MM_CONTROL	Main control register for the MMI

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
MMI SDRAM Control Registers				
Offset 0x04 D400			*MM_SDRAM_SIZE*	
31:2		-	Unused	Ignore during writes and read as zeroes.
1:0	R/W	0x2	MM_SDRAM_SIZE[1:0]	Specifies SDRAM footprint: 00: Unused 01: 16 MB 10: 32 MB 11: 64 MB
Offset 0x04 D404			*MM_REFRESH*	
31:10		-	Unused	Ignore during writes and read as zeroes.

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
9:0	R/W	0x17	MM_REFRESH[9:0]	Refresh period in clock cycles, scaled by 64. Note: This should only be modified when mm_enable=0 and mm_ready=0, then the SDRAM has entered self-refresh and it is safe to change the refresh period.
Offset 0x04 D408			***MM_SHORT_REFRESH***	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x0	MM_SHORT_REFRESH	Short refresh for Quickturn: 0: Regular refresh 1: Short refresh
Offset 0x04 D40C			***MM_REFRESH_ENABLED***	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x1	MM_REFRESH_ENABLED	Refresh on or off: 0: Refresh is off. 1: Refresh is on.
Offset 0x04 D410			***MM_ENABLE_INTERLEAVE***	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x1	MM_ENABLE_INTERLEAVE	Memory format: 0: Linear memory format 1: Two-way interleave memory format
Offset 0x04 D414			***MM_CONTROL***	
31:3		-	Unused	Ignore during writes and read as zeroes.
2	R/W	0x0	EN_MIPS_COMA_SF	Enable MIPS coma putting SDRAM into self-refresh. 0: When MIPS goes into coma mode, SDRAM is not put into self-refresh. 1: When MIPS goes into coma mode, SDRAM is put into self-refresh by MMI (SDRAM memory controller).
1	R	0x1	MM_READY	SDRAM Intialization or powerdown status: 0: SDRAM is being initialized or in powerdown. 1: SDRAM intialization is complete and SDRAM is not in powerdown.
0	R/W	0x1	MM_ENABLE	Intialize or powerdown SDRAM: 0: Powerdown SDRAM. 1: Functional MMI SDRAM controller
Offset 0x04 D418			***MM_READY_ENABLE***	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x0	MM_READY_ENABLE	Enable write to mm_enable bit not to finish until mm_ready reacts: 0: Write to mm_enable finishes immediately. 1: Write to mm_enable bit finishes when mm_ready equals the new value of mm_enable.

9.10 Unified Memory Organization in PNX8526

9.10.1 Introduction

This section describes the organization of the unified SDRAM memory as used in the PNX8526 project. Information is provided on:

- The mapping of addresses to banks, columns and rows in SDRAM
- Two-way and four-way interleaving of SDRAM banks
- How MPEG frames are stored in the memory
- How to do motion compensation of MPEG frames
- A brief discussion of gaps between memory transactions

The PNX8526 has a unified memory organization i.e., a single SDRAM memory is used for many different purposes:

- Video frame processing by an MPEG decoder, including efficient retrieval of reference frames for motion compensation
- General purpose microprocessor memory traffic, in particular cache miss traffic
- Video I/O and graphics display
- 2D and 3D graphics generation
- Various other I/O traffic

These different types of memory traffic are combined on one memory bus that achieves high sustained effective bandwidth close to the peak bandwidth of the bus.

The memory is optimized for a 64-bit bus using 64-Mbit SDRAM devices. The minimum memory size and granularity are 16 MB.

9.10.2 Organization of Addresses

The memory organization is optimized for transfers of 128-byte blocks. A two-way interleaving scheme is used between adjacent regions of 64 bytes. This makes it possible to have multiple back-to-back transfers of 128 bytes without gaps and with 100% bus utilization. Only when transfer is switched from read to write or from write to read do gaps have to be inserted. A 64-bit bus and SDRAM CAS latency=3 are used.

Figure 3 depicts interleaving of 64-byte regions. Table 7 shows the SDRAM commands for two consecutive 128-byte accesses with two-way interleaving and 100% bandwidth utilization on a 64-bit bus.

Regions of 64 bytes are Interleaved Between Even and Odd Numbered SDRAM Banks as shown in Figure 3.

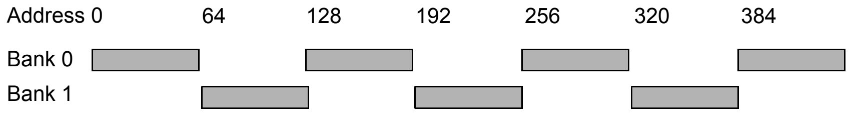


Figure 3: Two-Way Interleaving

Table 7: SDRAM Traffic for Interleaving of 128-byte Transactions

SDRAM command	R	C	R	CP	R	CP	R	CP	P
applied to bank	0SW	0	1	10	0	01	1	10	1
SDRAM data to bank0	01234567				01234567				
SDRAM data to bank1	89ABCDEF					89ABCDEF			
SDRAM data bus	0123546789ABCDEF0123456789ABCDEF								

[7-1] A 128-byte write operation is split into 2 transfers of 64 bytes; the first one from the even bank, the 2nd from the odd bank. While the 2nd transfer takes place, a precharge, a RAS, and a CAS command for the even SDRAM bank is given. A subsequent 128-byte write follows the first one without gaps.

Regions of 64 bytes in the memory address space alternate between SDRAM banks. A transfer of 128 bytes is split into a transfer of 64 bytes from the even numbered bank followed by a transfer of 64 bytes from the odd numbered bank. While the transfer from the odd numbered bank takes place, the even numbered bank is available to do precharge, RAS, and CAS for the next 128-byte transaction. If done, transfer of the 2nd transaction can follow the first transaction seamlessly.

Apart from the interleaving of 64-byte regions, the address space is mapped linearly to SDRAM rows. That makes it easy to have a variety of block transfer sizes in a transaction, while accessing a single SDRAM row. It also makes it possible to store the pixels of a video line in the same SDRAM bank.

It is assumed there are 4 SDRAM banks in a SDRAM chip. Those banks are alternated with each other, i.e., if an address region fills a row in bank 0 and bank 1 (using the two-way interleaving scheme), then the next consecutive addresses are in the same row in banks 2 and 3 of the SDRAM. This organization makes it possible to store consecutive video lines in different SDRAM banks, allowing interleaving access to the video lines by interleaving the SDRAM banks.

At least an 8-bit column address must be available. This makes the size of an SDRAM row at least 2 kB, big enough to store a full video line up to HD resolution.

In summary, byte addresses are mapped to SDRAM banks, columns, and rows as follows:

address = {remaining_rows_and_columns, bank[1], column[7:3], bank[0], column[2:0], byte_offset[2:0]}

The organization of the rows and remaining columns, if any, and the total address width depend on the particular SDRAM type and memory size used.

9.10.3 MPEG Memory Format

9.10.3.1 General Description

This section discusses the memory format and the location of pixels in memory for MPEG frames. With MPEG frames, the I-frames are used as reference frames for motion compensation. They are also the final output of the MPEG video decoder unit. The format discussed here is not used for other video frames, such as the output of the scaler unit or D1 video input units.

The video lines of an MPEG frame are stored in memory such that all pixels of a video line can fit in one SDRAM row of one SDRAM bank. The video lines do not need to be aligned with SDRAM row boundaries, so they can start in one row and continue to the next row. Consecutive video lines are stored adjacent to each other in the SDRAM rows, so there is no padding between video lines of a frame.

The SDRAM chips that are used have at least 4 banks. These 4 banks are interleaved with each other, such that when the video lines completely fill a row in one bank, the next row to fill will be in the next bank.

The purpose of this organization is to allow easy access to a multitude of video lines by interleaved access to the 4 SDRAM banks. This makes it possible to do efficient two-dimensional fetching of groups of pixels from adjacent video lines in a single two-dimensional memory transaction.

Since the memory uses the 64 bytes interleaved format in which groups of 64 bytes are alternately stored in odd or even numbered memory banks, the pixels of a video line can not be stored sequentially if they are to be in the same bank.

Therefore, to store a video line, the MMI skips the 64-byte portions that are not in the same SDRAM bank. When it gets to the end of an SDRAM row, it wraps back and uses the skipped portions of the memory range, which are then all positioned in the next SDRAM row. Skipping 64-byte portions is called the 64-byte stride format.

It is assumed that the width of the SDRAM bus is 64 bits and that SDRAM parts have an 8-bit column address or more. If that is the case, then an SDRAM row contains 2 kB of memory, or more, which is sufficient to store an entire video line of up to 2048 pixels.

Figure 4 depicts what storage of MPEG video lines looks like, taking 64 bytes interleaving into account.

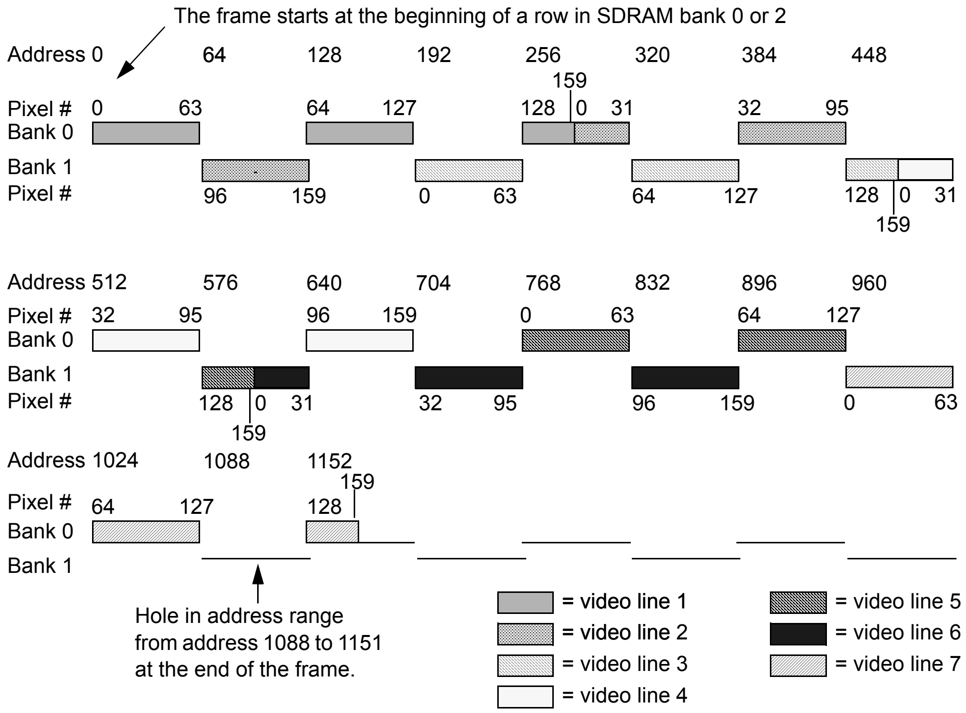


Figure 4: MPEG Video Lines in Memory (with 64-Byte Stride Format)

9.10.3.2 Padding

Some padding is necessary at the beginning and end of a frame to ensure that 64-byte portions of video lines using the 64-byte stride format are not mingled with 64-byte portions of other surrounding data that don’t use the 64-byte stride format. The video frame has to start on a 4-kB boundary, i.e. at the start of an odd and even pair of SDRAM rows. The end of the video frame has to be padded to fill any holes left over between 64-byte portions of the last video lines of the frame.

9.10.3.3 Frames, Fields and Pixels

The MPEG video frames are stored in YUV 4:2:0 format. The Y component of pixels is stored separately from the U and V components of pixels. The U and V components are stored together as UV pixels.

A frame consists of 2 fields; the even field contains the even-numbered video lines in the frame and the odd field contains the odd-numbered video lines in the frame.

The fields are stored separately to make it possible to do motion compensation on the fields separately. This is necessary because the MPEG standard allows different motion vectors for each field.

A video frame is stored in 4 different parts: Y components for the even field, UV components for the even field, Y components for the odd field, and UV components for the odd field.

9.10.3.4 Alternate Address Space for 64-Byte Stride

The 64-byte stride is best described by an alternate address space. Bytes that are consecutive in the alternate address space are observing the 64-byte stride format in the regular address space. There is a one-to-one mapping function f between an address AA in the alternate address space and an address A in the regular address space as follows:

AA[25:0] = f(A) = {A[25:12], A[6], A[11:7], A[5:0]}

and the inverse mapping function ff is:

A[25:0] = ff(AA) = {AA[25:12], AA[10:6], AA[11], AA[5:0]}

This assumes a 26-bit address space.

9.10.3.5 Y Pixels

The following specifies precisely how the Y component of pixels (or Y pixels in short) is organized in memory.

The 2 video fields of a video frame are in separate locations of memory. The video lines of one field are stored together in linear order in memory. There is no padding between the lines. The fields must be aligned on a 4-kB boundary.

The even numbered lines of a frame form the even field. The odd numbered lines of a frame form the odd field. We refer to lines in a field with their number in the frame (not in the field). Each Y pixel uses 1 byte.

Assume the start address of the even field is FIELD_EVEN_BASE with the alternate address

AFIELD_EVEN_BASE = f(FIELD_EVEN_BASE)

in the alternate address space. Assume the length of a video line is LINE_SIZE pixels (for instance 720).

Similarly, the start address of the odd field is FIELD_ODD_BASE with the alternate address

AFIELD_ODD_BASE = f(FIELD_ODD_BASE)

in the alternate address space.

The address PIXEL_ADDRESS_EVEN of Y-pixel number P on an even numbered line L of the frame in the even field is:

PIXEL_ADDRESS_EVEN = ff(AFIELD_EVEN_BASE + L/2 * LINE_SIZE + P)

The address PIXEL_ADDRESS_ODD of pixel number P on an odd numbered line L of the frame in the odd field is:

PIXEL_ADDRESS_ODD = ff(AFIELD_ODD_BASE + (L-1)/2 * LINE_SIZE + P)

A 16 by 16 macroblock in a frame, starting at line L and pixel P, has its pixels stored at addresses A[X,Y] as follows:

```
FOR (Y=L; Y<=L+15;Y=Y+1)
FOR (X=P; X<=P+15;X=X+1)
if (Y mod 2 == 0)
/* line in the even field */
 A[X,Y] = ff(AFIELD_EVEN_BASE + L/2 * LINE_SIZE + P);
else
```

```
/* line in the odd field */
A[X,Y] = ff(AFIELD_ODD_BASE + (L-1)/2 * LINE_SIZE + P);
```

9.10.3.6 UV Pixels

The U and V pixels are stored together in UV pixels.

A video frame of 720 by 480 pixels in YUV 4:2:0 video format consists of a Y frame of 720 by 480 bytes, a U frame of 360 by 240 bytes and a V frame of 360 by 240 bytes. Because of subsampling, the U and V frames each have 360 pixels per line and 240 lines. U and V pixels each use 1 byte per pixel.

The 16-bit UV pixels are formed as follows:
for every U pixel U[L,P] online L, pixel P in the U frame and every V pixel V[L,P] on the same line L and the same pixel P, the UV pixel UV[L,P] is defined as:

UV[L,P] = {V[L,P], U[L,P]}

The U pixel is the least significant byte in the UV pixel, the V pixel is the most significant byte in the UV pixel.

The U frame of 360 by 240 and the V frame of 360 by 240 are combined into one UV frame of 720 by 240 bytes with 2 bytes per UV pixel. The even lines of the frame form the even field of 720 by 120 bytes, the odd lines in the frame form the odd field of 720 by 120 bytes.

For the other possible video frame sizes, storage is similar.

The UV pixels are stored in memory as follows.

Assume the start address of the even UV field is FIELD_EVEN_BASE with the alternate address

AFIELD_EVEN_BASE = f(FIELD_EVEN_BASE)

in the alternate address space. Assume the length of a video line is LINE_SIZE bytes (for instance 720).

Similarly, the start address of the odd field is FIELD_ODD_BASE, with the alternate address

AFIELD_ODD_BASE = f(FIELD_ODD_BASE)

in the alternate address space.

Both fields have to be aligned on a 4-kB boundary, so

FIELD_EVEN_BASE mod 4096 = 0

and

FIELD_ODD_BASE mod 4096 = 0.

The address PIXEL_ADDRESS_EVEN of pixel number P on an even numbered line L in the even field of the UV frame is:

PIXEL_ADDRESS_EVEN = ff(AFIELD_EVEN_BASE + L/2 * LINE_SIZE + P*2)

The address PIXEL_ADDRESS_ODD of pixel number P on an odd numbered line L in the odd field of the UV frame is:

```
  PIXEL_ADDRESS_ODD = ff(AFIELD_ODD_BASE + (L-1)/2 * LINE_SIZE +
P*2)
```

A 16 by 8 UV macroblock in a frame, starting at line L and pixel P, consisting of 8 UV pixels of 2 bytes each on 8 lines, has its pixels stored at addresses A[X,Y] as follows:

```
FOR (Y=L; Y<=L+7;Y=Y+1)
FOR (X=P; X<=P+7;X=X+1)
if (Y mod 2 == 0)
/* line in the even field */
A[X,Y] = ff(AFIELD_EVEN_BASE + L/2 * LINE_SIZE + 2*P);
else
/* line in the odd field */
A[X,Y] = ff(AFIELD_ODD_BASE + (L-1)/2 * LINE_SIZE + 2*P);
```

9.10.4 16.4 Motion Compensation

This section illustrates how to do data transfers for MPEG motion compensation.

Motion compensation is done per macroblock and field. The Y macroblocks and UV macroblocks are each processed independently. In Figure 5 the organization of a macroblock (Y-component of video) is depicted.

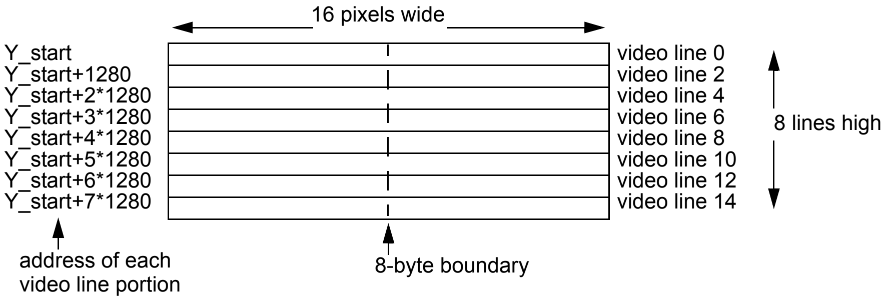


Figure 5: Storage of Y Macroblocks

Odd and even fields are stored separately and aligned on an 8-Bit boundary. The even field macroblock is 16 pixels wide and 8 lines high, containing the even numbered video lines. In this example, a pixel (Y-component) uses 1 byte and line_size is 1280.

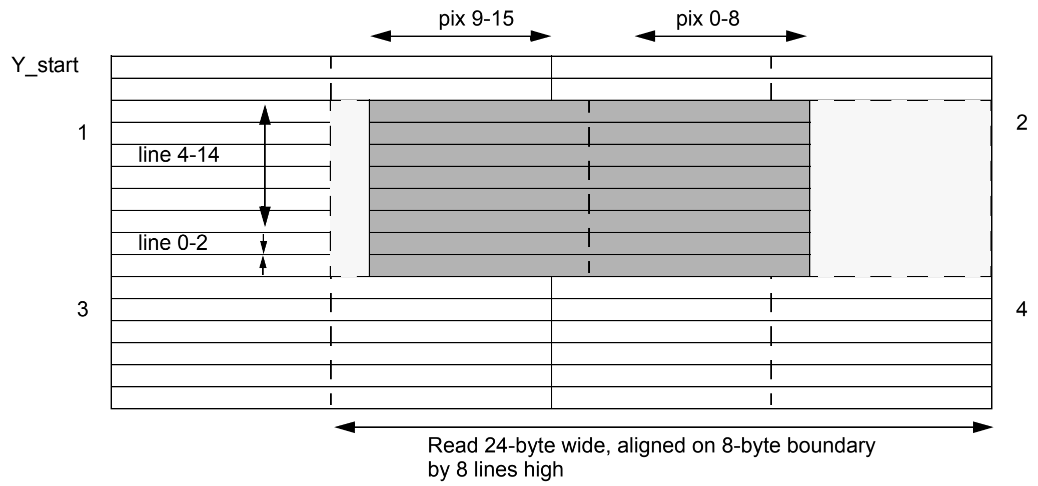


Figure 6: Reading a Block of 16 Pixels by 8 Lines

In Figure 6 above, the block is totally unaligned with respect to the macroblocks of the video frame. In this case, it is necessary to read the surrounding 24 pixels wide by 8 lines, aligned on an 8-byte boundary.

Frames are stored field-wise, i.e., the even-numbered lines of the frame are stored in the even field and the odd-numbered lines are stored in the odd frame. Macroblocks are 16 pixels wide and 8 video lines high per Y field. UV macroblocks are 8 pixels wide (16 bytes) and 4 lines high per field. The pixels of a video line are stored adjacent to each other in memory using the
64-byte stride format. There is a video line size stride in memory between each consecutive video line.

To keep the explanation simple, the examples in this section use addresses with pixels always adjacent to each other and disregarding the 64-byte stride format. In reality, the two-dimensional memory operations take the 64-byte stride into account.

To do motion compensation the MMI has to fetch a 16 by 8 Y macroblock out of its surrounding Y macroblocks. It's not aligned in any way, as depicted in Figure 6, so in this example it needs pixels 9-15 of line 4-14 of macroblock 1, pixels 0-8 of line 4-14 of macroblock 2, pixels 9-15 of line 0-2 of macroblock 3 and pixels 0-8 of line 0-2 of macroblock 4.

To read this unaligned block from memory, a two-dimensional read operation is used that reads the surrounding 24 byte-wide block, aligned on an 8-byte boundary by 8 lines high. Alignment of 8 bytes is the minimum alignment size supported by the memory system (that's 1 word on the 64-bit bus). Any unused pixels in the block being read are discarded or may be cached for possible later use.

The memory system supports a 24-byte by 4 line two-dimensional read operation, so the 24 by 8 read is split into two separate memory operations reading 24 by 4 each. In the example shown in Figure 6, what is needed is to do a 24 by 4 read at address (Y_start+2*LINE_SIZE+9), followed by a 24 by 4 read at address (Y_start+6*LINE_SIZE+9).

Since adjacent video lines are either in the same SDRAM row or in different SDRAM banks, the two-dimensional memory transactions can be carried out efficiently by interleaving access to SDRAM banks.

In the case of motion compensation with half pixel resolution, a block of 17 pixels wide and 9 lines high may need to be read. This needs to be split into two-dimensional read operations, one read of 24 bytes by 4 lines, plus one read of 24 bytes by 5 lines, aligned on an 8-bit boundary. The memory system supports a special transaction type for the 24-byte by 5-line two-dimensional read.

Where the Y-component of the field macroblock measures 16 bytes by 8 lines, the corresponding UV components of the UV field macroblock are stored in 16 bytes wide by 4 lines high (YUV 4:2:0 video format).

For motion compensation of the UV component, we need to read the 16 by 4 block, in a manner similar to the Y-block by using a 24-byte by 4-line two-dimensional memory read operation aligned on an 8-byte boundary.

In case of half-pixel resolution we may need to read 18 pixels by 5 lines for the UV component. This needs to be done with a 24 by 5 read, aligned on an 8-byte boundary.

9.10.5 Storing MPEG Frames Using 32 by 4 Writes

The result of MPEG motion compensation is calculated macroblock by macroblock. To store this into memory, two-dimensional write operations are supported by the memory system. The write operation is 32 bytes horizontally by 4 video lines vertically, aligned on a 16-byte boundary.

The two fields of a frame are stored separately. A field macroblock contains 16 by 8 Y pixels and 16 by 4 UV pixels. Two adjacent macroblocks need to be accumulated to fill the 32 pixels horizontally needed for the 32 by 4 write operation.

To write two adjacent field macroblocks to memory, two 32-byte by 4-line write operations are issued for Y and one 32 by 4 write operation for UV pixels.

The macroblocks are always nicely aligned making these write operations very efficient, unlike reading macroblocks for motion compensation that are unaligned and therefore less efficient.

If a video line contains an odd number of macroblocks, then at the end of the video line it may need to write only a single macroblock to memory. That can be done by masking out the unused portion of the 32 by 4 write operation using byte write enables.

9.10.6 Reading Video Lines of Calculated MPEG Frames

Since the MPEG frames are stored in memory using the 64-byte stride format, retrieving video lines of the calculated frames for further processing has to be special. MPEG video lines need to be fetched using 64-byte read bursts aligned on 64-byte boundaries (which is less efficient than reading frames produced by the scaler that can be fetched with128-byte bursts).

Since video lines are generally not aligned on 64-byte boundaries, a first 64-byte read needs to be done to read the first few pixels. The number of pixels in this first group is variable and depends on the alignment of the video line. After the initial group, groups of 64 pixels can be read aligned on 64-byte boundaries using 64-byte read bursts.

Remark: When calculating the address of the next group of 64 pixels, the address has to be
incremented by 128, not by 64 to take the 64-byte stride format into account.

The end of the video line may be unaligned to a 64-byte boundary, so that the last 64 bytes read burst returns some pixels of the next video line. Those pixels may need to be discarded, if the next video line is not needed immediately.

9.10.7 Storage of Other Video Frames

Video frames other than MPEG reference frames are not involved in motion compensation and it is not necessary to access their individual macroblocks. Therefore, they do not need to use the memory format that is optimized for two-dimensional memory transactions. Instead, these frames should use the regular linear memory format. Each video line is stored linearly in memory and only linear access to them is used.

It is preferred to use 128-byte read and write transactions. These will use two-way interleaving and optimal memory bandwidth utilization.

9.11 Performance Features

The memory controller includes features to aggressively optimize bandwidth utilization. Pipelining and interleaving are used to overlap transactions with each other using multiple SDRAM banks simultaneously. MPEG two-dimensional transactions are internally reordered to optimize bandwidth.

Latency is optimized in several ways. When a request enters the memory controller from the arbiter, an SDRAM row activate command is immediately generated and sent from the PNX8526 I/O pads to the SDRAM chips in the next cycle. The moment that an arbitration decision is made is also postponed as long as possible. The arbiter can still change its selection of which internal bus agent wins the arbitration in the same cycle as the first row activate command is generated. Phrased differently, the SDRAM controller has zero-cycle latency.

Chapter 10: GPIO/IR

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

10.1 Introduction

The PNX8526 has a number of General Purpose Input Output (GPIO) pins. There are a number of dedicated GPIO pins and certain others that can be configured as GPIO. These are pins that might not otherwise be required in some system configurations.

The functions of the PNX8526 GPIO module are as follows:

- Masked software group/signal value setting
- Software observation
- Signal sequence monitoring (using event timestamps or samples)
- Timed pattern generation capability (using event timestamps or samples)

10.2 Functional Description

10.2.1 GPIO Pin Assignment

The PNX8526 has 12 dedicated GPIO pins, GPIO[11:0] in the pin list. In addition, Table 1 lists the 49 other pins that can either operate with primary functionality or be set as GPIO pins. GPIO mode selection is on a pin-by-pin basis.

Table 1: GPIO Pin Assignments

Primary Function	GPIO Number[1]	Primary Function	GPIO Number[1]
PCI_GNT_B	60	TS_SOP	31
PCI_GNT_A	59	TS_CLK	30
PCI_REQ_B	58	TS_DATA[7:0]	29—22
PCI_REQ_A	57	SSI_SCLK_CTSN	21
XI0_A25	56	SSI_FS_RTSN	20
XI0_ACK	55	SSI_RXD	19
XI0_SEL[2:0]	54—52	SSI_TXD	18
AIO_SD[3:0]	51—48	UA2_CTSN	17
AIO_WS	47	UA2_RTSN	16
AIO_SCK	46	UA2_RX	15
AIO_OSCLK	45	UA2_TX	14

Table 1: GPIO Pin Assignments *...Continued*

Primary Function	GPIO Number[1]	Primary Function	GPIO Number[1]
DV1_VALID	44	UA1_RX	13
DV1_CLK	43	UA1_TX	12
DV1_DATA[9:0]	42—33	Dedicated GPIO	11—00
TS_VALID	32		

[1] This GPIO Number is used in the field names of the GPIO Control and Data registers.

The following internal signals are supplied to the GPIO module to enable monitoring using event timestamping:

- VIP1/2 timestamp: vip1/2_eow_aux, vip1/2_eow_vid
- ICP1/2 timestamp: icp1_tstamp, icp2_tstamp
- AI1/2, AO 1/2, AIO timestamp: ai1/2_tstamp, ao1/2_tstamp, aio_tstamp
- SPDIO: spdo_tstamp, spdi_tstamp1 (SPDI word select timestamp), spdi_tstamp2
- GPIO timestamps: LAST_WORD<0..3>
- TSDMA timestamps: tsdma_tstamp1/2

Any of the GPIO or internal signals listed above can be selected for signal monitoring. The GPIO pins can also be selected for pattern generation.

When a GPIO pin is programmed to GPIO mode, and the primary functional mode is an input to a module, the normal functional input signal is held inactive. Table 2 below shows the relevant signals and their inactive state. This is to avoid unpredictable behavior in a module when a pin is being used as a GPIO.

Table 2: Inactive States for Module Inputs

Functional Module Input	Destination Module	Inactive State
PCI_REQ_B	PCI	1
PCI_REQ_A	PCI	1
XIO_ACK	XIO	1
AIO_SD	AIO	0
AIO_WS	AIO	0
AIO_SCK	AIO	0
DV1_VALID	MSP/VIP	NA
DV1_CLK	MSP	NA
DV1_DATA[9:0]	MSP/VIP	NA
SSI_SCLK_CTSN	SSI	1
SSI_FS_RTSN	SSI	1

Table 2: Inactive States for Module Inputs *…Continued*

Functional Module Input	Destination Module	Inactive State
SSI_RXD	SSI	0
UA2_CTSN	UART2	1
UA2_RX	UART2	0
UA1_RX	UART1	0

The dedicated GPIO pins GPIO[11:9] are used as ICAM/Smartcard IO when not programmed in GPIO mode. Table 3 below describes the muxing. The ICAM/ Smartcard inputs are held inactive when GPIO[11:9] are programmed in GPIO mode.

Table 3: SmartCard GPIO Muxing

GPIO	Smartcard Signal	Inactive State
GPIO[11]	ICAM2_SC_C4	0
GPIO[10]	ICAM2_SC_C8	0
GPIO[9]	ICAM2_SC_SETVPP	N/A

10.2.2 GPIO Mode Settings

Each GPIO pin operates in one of three modes:

- primary function
- open drain output
- tri-state output.

10.2.2.1 GPIO mode select

Each pin has a 2-bit mode field in a 32-bit word. Values are as follows:

Table 4: GPIO Mode Select

GPIO Mode	Description
00	Retain pin mode of operation. This will not overwrite current mode.
01	Switch pin mode to primary operating mode. Refer to Table 1.
10	Switch pin mode to GPIO.
11	Switch pin mode to open-drain GPIO (this prevents active high drive)

10.2.2.2 GPIO Data Settings

Each GPIO pin can be used as a single output, or several can be grouped and used as a bus. The GPIO data setting can be programmed by a single MMIO write that provides a mask, drive and data value. The Mask and IO Data (IOD) make up a 2-bit value—the Mask bit is located in the upper 16 bits (31:16) and the IOD bit is located in the lower 16 bits (15:0) of the 32-bit MMIO register. For example, Mask bit[16] is paired with IOD bit[0] and [17]...[1], [18]...[2], etc. This pairing makes up the 2-bit value for programming the GPIO data setting, shown in Table .

Table 5: Settings for Mask[xx] and IOD[xx] Bits

MASK[xx] Bit	IOD[xx] Bit	Description
0	0	Retains current stored data (will not overwrite current data). Not readable.
0	1	Sets the corresponding GPIO pin in tri-state mode allowing the pin to be used as data input.
1	0	GPIO output mode. Drives a generated pattern (if enabled) or IOD ('0' in this case) onto the corresponding GPIO pin.
1	1	GPIO output mode. Drives a generated pattern (if enabled) or IOD ('1' in this case) onto the corresponding GPIO pin. Note: If open-drain mode is selected, drive to '1' is disabled.

[5-1] The number portion of MASKxx or IODxx identifies the GPIO number of the particular pin. Refer to Table 1.

With 32-bit MMIO registers, this creates update capability for up to 16 signals simultaneously.

To avoid glitches on GPIO lines, the following programming order is recommended:

Step 1) Program Mode Select register with '10' [GPIO mode]
Step 2) Program Mode Select register with '01' [primary operating mode]

Remark: This programming order is not required for GPIO[60:52].

10.2.2.3 Reading GPIO Pin Status

Each GPIO pin status can be read by software using MMIO read. In the 32-bit register, the lower 16 bits are the GPIO pin data values.

For "open drain" or "tri-state" output values, the input value read by software is the pad value, not the driven value.

Software reading of the GPIO input pad is always possible, even when the GPIO pin is operating in primary function mode.

10.2.2.4 Signal Monitoring and Pattern Generation

There are four FIFO queues available to perform signal monitoring or pattern generation. Each FIFO queue can be programmed to operate in either of these modes for a selected GPIO pin.

The FIFO is a DMA/memory based FIFO to allow efficient CPU access to large event lists.

A double buffer style FIFO is used.The base start addresses for both DMA buffers in every queue is programmable as is the size of the DMA buffers. The 'size' parameter allows DMA buffers to be up to 1 MB.

There is also 128 bytes of internal buffering for each FIFO queue to ensure efficient use of the DMA streaming. This internal buffering is split into two buffers of 64-byte capacity. This is implemented as one 32x32 register file divided into two 64-byte buffers.

Initially in pattern generation modes, the hardware requests 2 x 64 bytes from the DMA buffer(s) in order to fill its two 64-byte internal buffers. When one of these internal buffers becomes empty 64-byte DMA requests are generated to fill the empty internal buffer. The DMA read requests are always 64 bytes.

In signal monitoring, the 64-byte internal buffers are filled by the GPIO and flushed out to the DMA buffers. The DMA write requests are always 64 bytes.

In addition there are 12 32-bit (31-bit timestamp 1-bit direction) timestamp units which can be used for event timestamping for a selected GPIO pin. Therefore a total of 16 signals can be selected for monitoring and four for pattern generation.

A 34-bit time counter runs at 108 MHz. The 31 msbits of this counter are used for the master time count. This gives a resolution of 13.5MHz for master time (and timestamps). The master time counter is reset by peri_rst_n.

The master time counter is used to:

- Generate timestamps in event timestamping
- Compare timestamps in pattern generation
- Also readable by the CPU

Signal Monitoring

A maximum of 16 signals selected from the GPIO pins or the PNX8526 internal GPIO observable signals, can be simultaneously “monitored” using either the four FIFO queues or the 12 32-bit timestamp units.

Signals can be monitored in two different ways:

- Event Timestamping: Using event timestamps whenever a signal changes state
- Signal Sampling: Sampling the signal value at a programmable frequency or by a selected clock input

The 12 Timestamp Units can only monitor signals using event timestamping. The four FIFO queues can be programmed to monitor signals using either event timestamping or signal sampling.

In each of the 12 timestamp units, 31-bit timestamps plus a direction bit are written to 32-bit registers. A DATA_VALID interrupt is generated whenever data is written to the register. An overrun error interrupt is generated whenever new data is received before the DATA_VALID interrupt has been cleared. The old data is not overwritten: the new data is lost.

The TSU (timestamp unit) register is stable to be read by software while the relevant DATA_VALID flag is raised.

In each of the four FIFO queues, timestamps or samples are initially written to two internal 64 byte buffers. When one of these internal buffers is full, a DMA request for the contents of the internal buffer to be written to a DMA buffer in memory is generated. Writing to the internal buffers switches to the second buffer. An interrupt is generated if overrun occurs in the internal buffers (INT_OE).

GPIO MMIO Description for Signal Monitoring FIFO queues

Upon reset, signal monitoring is disabled (FIFO_MODE, EVENT_MODE= 00), and DMA buffer 1 is the active DMA buffer. Software initiates signal monitoring by providing two equal size empty DMA buffers and putting their base address and size in the BASE1, BASE2 and SIZE registers. Once two valid DMA buffers are assigned, monitoring can be enabled by programming FIFO_MODE and EVENT_MODE. The GPIO hardware will proceed to fill DMA buffer 1 with timestamps or samples. Once DMA buffer 1 fills up, BUF1_RDY is asserted, and monitoring continues without interruption in DMA buffer 2. If BUF1_RDY_EN is enabled, a level triggered interrupt request is generated to the chip level interrupt controller.

When BUF1_RDY is high, software is required to assign a new, empty buffer to BASE1 and then clear the BUF1_RDY flag (write a ‘1’ to BUF1_RDY_CLR) before buffer 2 fills up to avoid overrun.

Monitoring continues in DMA buffer 2 until it fills up. At that time, BUF2_RDY is asserted and monitoring continues in the new DMA buffer 1, etc.

If the software fails to read the full DMA buffers in time (i.e., BUFx_RDY is not cleared in time), the overrun (FIFO_OE) error flag is raised and data may be lost. The FIFO_OE error flag can only be cleared by an explicit write of logic ’1’ to FIFO_OE_CLR.

If enabled, an interval of silence can cause a BUFx_RDY flag to be asserted before all locations in the DMA buffer have been filled. Therefore, whenever BUFx_RDY is asserted, software is required to read the relevant INT_STATUS register to know exactly how many valid 32-bit words of data are in the DMA buffer. The INT_STATUS holds the VALID_PTR field which gives this information.

The number of valid 32-bit data words written to the DMA buffers is loaded by the GPIO hardware to the VALID_PTR field of the INT_STATUS register immediately before the GPIO sets the relevant BUFx_RDY flag. If a second BUFx_RDY is activated before the first flag was cleared VALID_PTR cannot be updated by the GPIO until the first activated BUFx_RDY flag is cleared by software. This clear will allow the GPIO to load the new VALID_PTR value for the second buffer.

If both BUFx_RDY flags are cleared at the same time, i.e., if the value of VALID_PTR is not needed, the VALID_PTR value points back to the first buffer whose BUFx_RDY flag was raised. If the VALID_PTR value is required to be read, each BUFx_RDY must be cleared individually and in the correct order.

VALID_PTR is stable to be read by software when a BUFx_RDY flag is raised. BASE1 should be stable to be loaded by the GPIO hardware when BUF1_RDY is cleared by software and BASE2 should be stable to be loaded by the GPIO hardware when BUF2_RDY is cleared by software.

Software activated upon a BUFx_RDY or DATA_VALID interrupt can scan the FIFO or read the timestamp unit register in order to interpret complex bit serial protocols.

Remark: A DMA buffer can “fill up” in two ways: all available locations are written to, or in the event of timestamping, an interval of silence can occur.
SIZE must be a multiple of 64 bytes to accommodate 64-byte DMA accesses. SIZE is a static configuration register and should not change during GPIO operation.

Event Timestamping

For each monitored signal, rising edges, falling edges or both edges can be monitored.

Any change (according to the monitored edge event) generates a 31-bit timestamp and a 1-bit edge direction in a 32-bit word. The 1-bit direction indicator is logic 1 if a rising edge has occurred and logic 0 if a falling edge has occurred. The direction bit is the msb of the 32-bit word generated.

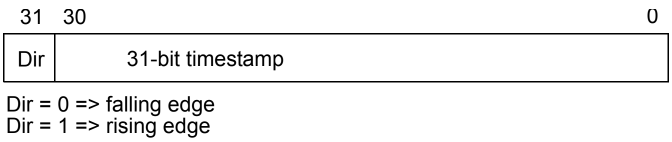


Figure 1: 32-Bit Timestamp Format

The event timestamps are written (per monitored signal) to a DMA buffer or a timestamp unit register, which is software readable.

When the DMA buffers are being used, if events occur on a monitored signal and an interval of silence follows, the relevant internal buffer contents are flushed to the DMA buffers.

When the contents of the internal buffer are flushed to the DMA buffer the relevant BUFx_RDY flag is set. The BUFx_RDY interrupt indicates that the DMA buffer is ready to be read by software and writing is switched to the second DMA buffer.

When an interval of silence occurs all 64 bytes of the internal buffer are flushed even though there may not be 64 bytes of valid data in the internal buffer. Software must then read the module status to read the address where the last valid 32-bit data word was written (VALID_PTR).

The length of the interval duration is programmed using the INTERVAL register.

Remark: If there is no internal buffer data to be flushed and no valid data in the DMA buffers, the interval of silence will not cause BUFx_RDY to be asserted.
Timestamping always works, even if the pin selected for monitoring is operating in regular mode.

Signal Sampling

In “signal sampling” a signal or a group of signals can be monitored at a programmed frequency or by a selected clock input.

The programmed sampling frequency is divided down from 108 MHz using a 16-bit divider. The sampling frequency is programmed in the DIVIDER register.

Instead of using the programmed sampling frequency it is possible to program any of the GPIO inputs as the sampling clock. This is enabled using registers EN_CLOCK_SEL and CLOCK_SEL. If this feature is used, it is important to know that these clocks need to be turned on by programming the clock module (see Clocks module specification). Also, if the GPIO being used as a sampling clock is toggling before CLOCK_SEL and EN_CLOCK_SEL have been programmed, it is important to block these clocks in the Clocks module until CLOCK_SEL and EN_CLOCK_SEL are programmed. This avoids the possibility of glitches on the clock.

GPIO inputs can be grouped together and sampled at once in the same FIFO queue. It is possible to sample one, two or four GPIO inputs in one FIFO queue. The one, two or four bits fill a 32-bit word full of 32, 16 or 8 samples (see Figure 3 below). The 32-bit word of the sampled signals is written to the DMA buffer. The numbers of GPIO inputs sampled per FIFO queue can be programmed in the EN_IO_SEL register. The GPIO inputs selected for sampling is programmed in the IO_SEL register.

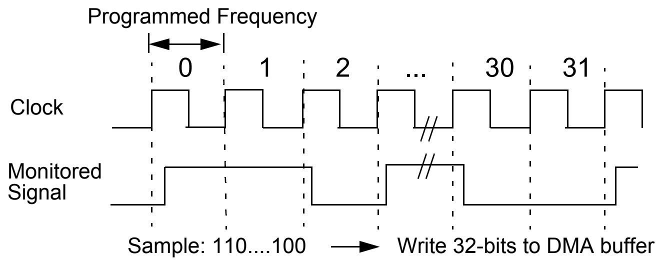


Figure 2: Signal Sampling

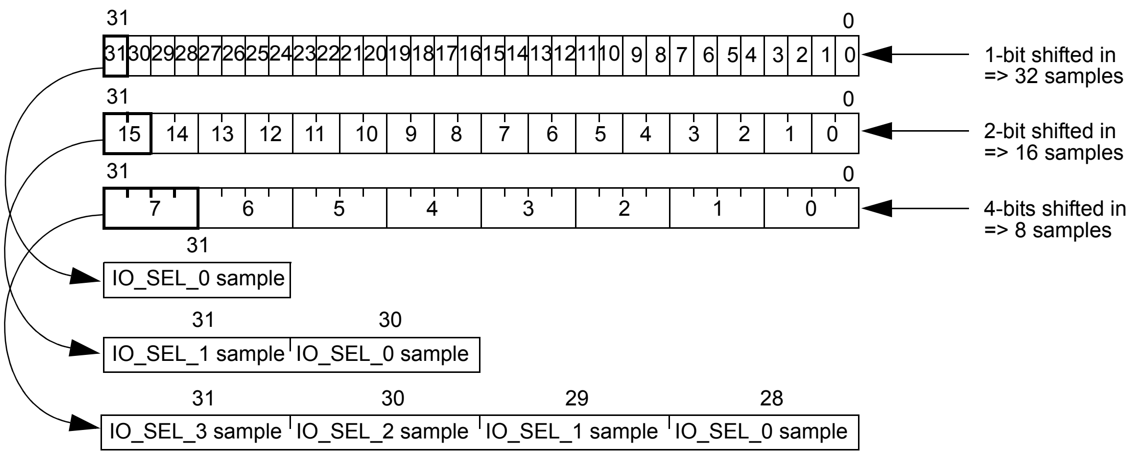


Figure 3: GPIO Input Merging in Signal Sampling Mode

Pattern Generation Capability

A maximum of four outputs, selected from the GPIO pins and the C1394_SYNC0/1 GPIO outputs, can be simultaneously “driven” using timed events.

Patterns can be generated in two different ways:

- Using timestamp information with a direction bit
- Using signal samples.

In each of the FIFO queues, when software has written values into the DMA buffers, pattern generation can be enabled.

GPIO MMIO Description for Pattern Generation FIFO Queues

Upon reset, transmission is disabled (FIFO_MODE, EVENT_MODE=00), and DMA buffer 1 is the active buffer. The system software initiates transmission by providing two DMA buffers containing valid data and by putting their base addresses in the two BASEx registers, their maximum size into the SIZE register and the number of valid words in DMA buffer 1 into BUF_LEN.

When the FIFO Queue is programmed into Pattern Generation (FIFO_MODE[1]=1) mode, BUF1_RDY and BUF2_RDY flags will get set, indicating that it is ready for a new DMA buffer containing valid data to be assigned.

Once two valid buffers are assigned and the FIFO queue has been enabled (FIFO_MODE[1]=1), the BUF1_RDY flag must be cleared by software so that the hardware can load the BASE1 and BUF_LEN values. After BUF1_RDY has been cleared the software can program the BUF_LEN value for DMA buffer 2. When the BUF2_RDY flag is cleared the BASE2 and BUF_LEN values for DMA buffer 2 are loaded by the hardware.

Remark: If the BUF_LEN values for DMA buffer1 and DMA buffer 2 are identical both BUF1_RDY and BUF2_RDY can be cleared at the same time.

The GPIO hardware now proceeds to empty DMA buffer 1 by requesting 64-byte DMA reads from the DMA buffers and transmitting the samples/timestamps on the GPIO pins. Once DMA buffer 1 empties, BUF1_RDY is asserted. If BUF2_RDY has been cleared, transmission continues without interruption from DMA buffer 2. If BUF1_RDY_EN is enabled, a level-triggered, system level interrupt request is generated.

While BUF1_RDY is high, the system software is required to assign a new buffer to BASE1, the number of valid words in the new buffer to BUF_LEN and then clear BUF1_RDY (write a ‘1’ to BUF1_RDY_CLR) before DMA buffer 2 fills up to avoid underrun. Transmission continues from buffer 2, until it is empty. At that time, BUF2_RDY is asserted, and transmission continues from the new buffer 1, etc.

Remark: The BASEx and BUF_LEN values for a DMA buffer are only loaded into the GPIO hardware when the relevant BUFx_RDY signal has been cleared. Since the BUF_LEN register is shared between both DMA buffers it important that the value in BUF_LEN when BUFx_RDY is being cleared is the correct value for that DMA buffer.

The BASEx and BUF_LEN values should be stable before software clears BUFx_RDY.

Remark: The DMA buffer sizes must be a multiple of 64 bytes to accommodate 64-byte DMA access and restrictions. SIZE is a static configuration register and must not be changed during GPIO operation.

Pattern Generation Using Timestamps

This form of pattern generation is the inverse of event timestamping. Software fills a (per signal) DMA buffer with timed events (31-bit timestamp + 1-bit direction). The hardware performs the scheduled event on a selected GPIO signal when the master timestamp clock reaches this value.

Pattern Generation Using Signal Samples

In this type of pattern generation, software fills a (per signal) DMA buffer with sampled values. The sampling frequency is divided down from 108 MHz using a 16-bit divider. The signal pattern is then generated using the 32-bit sample read from the DMA buffer and the programmed sampling frequency.

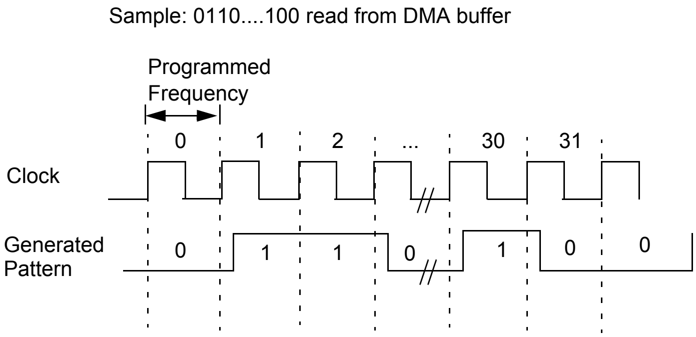


Figure 4: Pattern Generation Using Samples

In this mode, the EN_CLOCK_SEL and CLOCK_SEL can be programmed such that a signal running at the sampling frequency is output onto a GPIO output pin selected by CLOCK_SEL. The duty-cycle of this signal may not be 50:50. This depends on the sampling frequency programmed.

GPIO outputs can be grouped together in one FIFO queue. One FIFO queue can drive one, two or four outputs. The number of outputs that can be driven by each queue is selected by programming EN_IO_SEL. The driven GPIO output pins are selected by programming the IO_SEL register. The 32-bit sample read from the DMA buffer is either 32 1-bit samples if one output is being driven by the FIFO queue, 16 2-bit samples if two outputs are being driven by the FIFO queue, or eight 4-bit samples if four outputs are being driven by the FIFO queue.

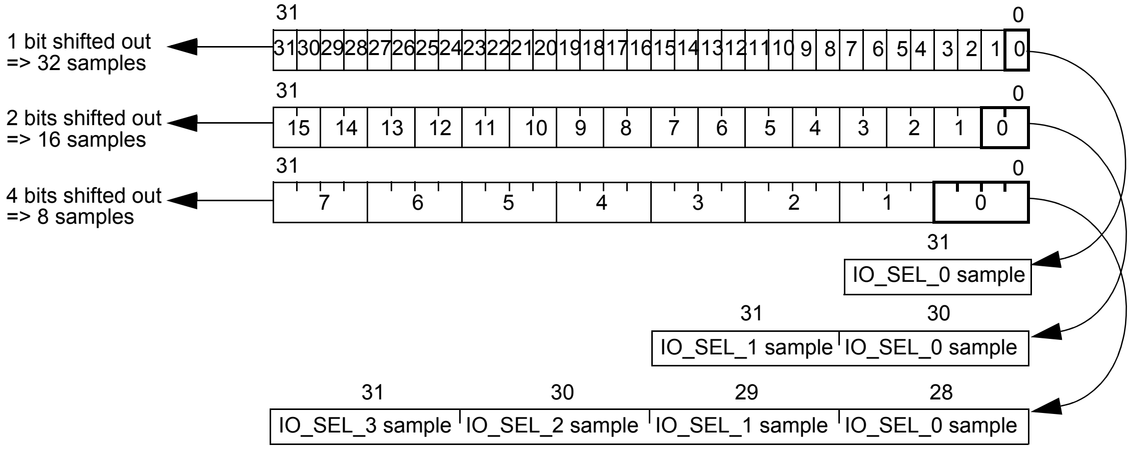


Figure 5: GPIO Output Merging in Pattern Generation Using Samples Mode

GPIO Timestamp Signal Generation

In pattern generation modes the GPIO can be programmed to generate events which signal the last 32-bit word read from a DMA buffer has arrived at a GPIO output pin.

The event will be a positive edge pulse with the duration of the event to be greater than or equal to 148 ns (2 x [1/13.5 MHz]). The specific event generated for each FIFO queue is LAST_WORD.

This event can then be selected as an input signal to an event timestamper (FIFO queue or TSU) to timestamp the occurrences of LAST_WORD. This is programmed using IO_SEL (see Table 8).

This mode is programmed using the EN_EV_TSTAMP field of the relevant GPIO_EV register.

Error Behavior

A DMA buffer overrun, FIFO_OE, occurs in signal monitoring modes if a new, empty DMA buffer is not supplied by software in time—i.e., if BUF1_RDY and BUF2_RDY are both active.

An internal buffer overrun, INT_OE, occurs when the contents of an internal GPIO buffer are ready to be transferred to a DMA buffer, but DMA buffers are full.

If either a FIFO_OE or INT_OE error occurs, signal monitoring is temporarily halted, and incoming timestamps/samples will be lost. In the case of FIFO_OE, sampling resumes as soon as the control software makes one or more new buffers available by clearing the relevant BUFx_RDY. In the case of INT_OE, sampling will resume as soon as the internal buffer can be written to memory. INT_OE and FIFO_OE are “sticky” error flags meaning they will remain set until an explicit software write of logic 1 to FIFO_OE_CLR or INT_OE_CLR is performed.

GPIO Frequency Restrictions

DMA requests must be able to tolerate waits of up to 40 µs and DMA requests can not be made more frequently than once every 40 µs.

This restricts the highest frequency that can be tolerated in signal monitoring and pattern generation for the FIFO queues.

In the FIFO queues the internal buffering allows DMA requests of 64 bytes of data (16 x 32 bits). The calculations below show the maximum frequencies allowed for signals to be monitored and patterns to be generated if one FIFO queue is requesting DMA at any one time.

Remark: Sampling calculations assume 1-bit sampling (EN_IO_SEL = 00/11).

Timestamping: *1* edge -> 32 bits

=> 16 edges = 64 bytes of data

=> 16 edges can occur every 40us

=> 1 edge can occur every 2.5us = <u>400 kHz maximum freq.</u>

Sampling: *1* edge -> 1 bit

=> 512 edges = 64 bytes of data

=> 512 edges can occur every 40us

=> 1 edge can occur every 78.125ns = <u>12.8 MHz maximum freq.</u>

There is one DMA read channel and one DMA write channel available for all four FIFO queues. Each FIFO queue will only make 64-byte DMA requests. The DMA has 128 bytes of buffering.

The total bandwidth for all four FIFO queues in DMA read or DMA write is 64 bytes per 40 µs but 128 bytes can be stored in the DMA gizmo. Therefore, in the worst case situation where all FIFO queues are requesting DMA in the same direction and at the same time, after 2x40 µs all data from all FIFO queues has been granted a DMA request or is in the DMA buffer. This means the FIFO queues can request another DMA request without any loss of data every 2x40 µs.

So, in this “worst case” situation the monitored/generated signal frequencies that can be tolerated are as follows:

Timestamping: 1 edge -> 32 bits

=> 16 edges = 64 bytes of data

=> 16 edges can occur every 2x40 µs

=> 1 edge can occur every 5 µs = *200 kHz maximum freq.*

Sampling: 1 edge -> 1 bit

=> 512 edges = 64 bytes of data

=> 512 edges can occur every 2x40 µs

=> 1 edge can occur every 156.25 ns = *6.4 MHz maximum freq.*

Remark: Sampling calculations assume 1-bit sampling (EN_IO_SEL = 00/11).

Similar calculations for frequency tolerances can be made for 2 or 3 queues requesting DMA in the same direction and at the same time, and for queues which use multi-bit sampling (EN_IO_SEL = 01/10).

IR Applications

For each FIFO queue programmed in signal monitoring or pattern generation modes, it is possible to divide the 108 MHz clock to obtain suitable frequencies for IR applications.

As well as the 16-bit divider to divide the 108 MHz clock, each FIFO queue has a 5-bit divider which can be enabled if sub-carrier frequencies are required for transmission. Therefore, in IR applications, a FIFO queue can produce IR signals at a required TX frequency and have the option to multiplex a sub-carrier frequency onto the TX frequency if required. See Figure 6 for an example.

Table 6: IR Protocol RX/TX Frequencies

PROTOCOL	MIN. PULSE	REQ. FREQ	FREQ_DIV[15:0]	CARRIER_DIV[4:0]	Error (%)
CIR (IrDA Control)	6.67 µs	150 kHz	0x2D0	1 or Disabled	0
CIR with Sub-Carrier (TX)	0.667 µs	1.5 MHz	0x24	14h	0
RC-MM	27.77 µs	36 kHz	0xBB8	1 or Disabled	0
RC-MM Sub-Carrier	9.26 µs*	108 kHz	0x1F4	6h	0

[6-1] Note: RF Sub-Carrier is 36 kHz, ONTIME should be between 25-50% of 27.77 µs period. (108 kHz = 33%)

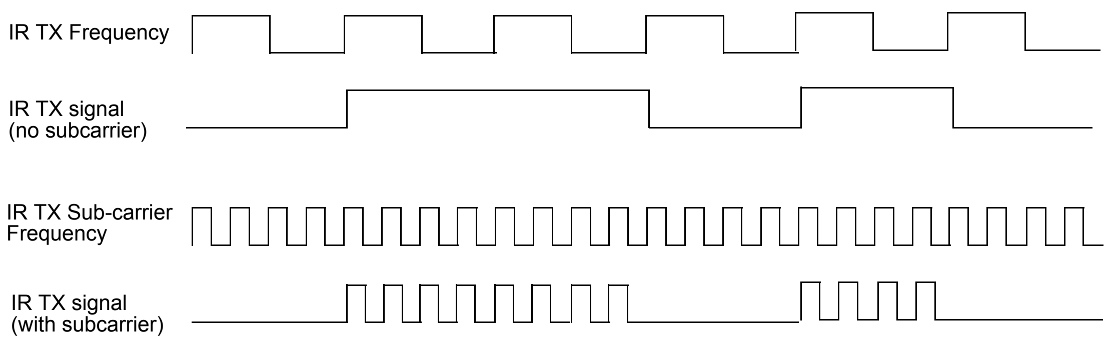

IR TX Frequency
IR TX signal (no subcarrier)
IR TX Sub-carrier Frequency
IR TX signal (with subcarrier)

Figure 6: Example of IR TX Signals with and without Sub-Carrier

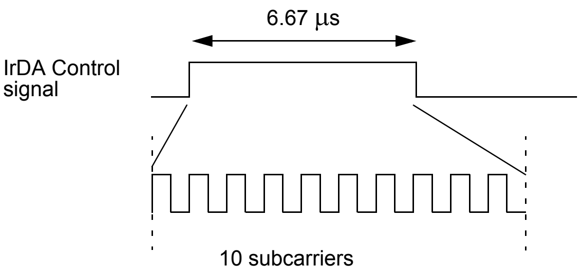


Figure 7: IrDA Control TX with Sub-Carrier Enabled

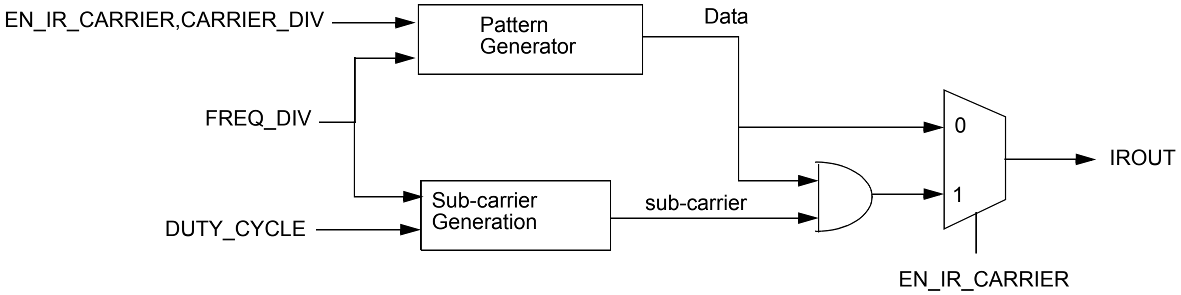


Figure 8: Sub-Carrier Muxing for TX

Duty-Cycle Programming

In the RC-MM IR protocol the duty-cycle of the sub-carriers must be between 25-50%. To accommodate this protocol and others it is possible to program the duty-cycle to be either 33%, 50%, or 66%.

In the 33% and 66% duty-cycle cases FREQ_DIV must be programmed such that the resulting frequency is three times the actual sub-carrier frequency—i.e., the minimum pulse width (high or low) is programmed. Similarly, in the 50% duty-cycle case FREQ_DIV must be programmed such that the resulting frequency is two times the actual sub-carrier frequency.

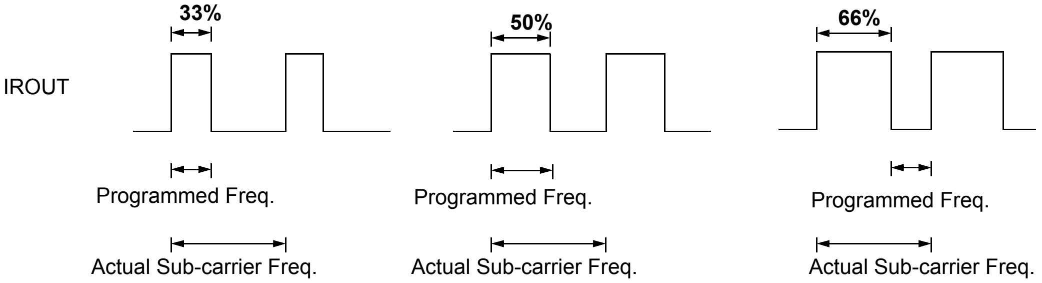


Figure 9: Example Duty Cycles for IR TX Signals

Spike Filtering

When signal sampling at a programmed frequency, a filtering feature is available in the GPIO module. This filter removes spikes that may occur on an IR RX signal. This feature is enabled by programming the EN_IR_FILTER and IR_FILTER registers. The IR_FILTER value represents the spike filter pulse width—i.e., all pulses less than the IR_FILTER pulse width are considered spikes and not passed through to the signal monitoring control.

Interrupts

Each FIFO queue has four types of interrupts:

- BUF1_READY: DMA buffer 1 ready for reading/writing
- BUF2_READY: DMA buffer 2 ready for reading/writing
- FIFO_OE: DMA buffer overrun error
- INT_OE: Internal buffering overrun error.

Each timestamp unit has two types of interrupts:

- DATA_VALID: TSU has data ready to be read
- INT_OE: Internal buffering overrun error

For each FIFO queue, the interrupts are ORed together (if enabled) and one PIC interrupt is produced. Therefore the four FIFO queues produce four separate PIC interrupts.

The 12 timestamp unit interrupts are ORed together (if enabled) to produce one interrupt. Therefore all 12 TSUs produce one PIC interrupt.

All the interrupt status bits are “sticky” bits and can only be cleared by writing a 1 to the relevant interrupt clear register.

Also, all five PIC interrupts are ORed together to generate one VIC interrupt. A GPIO status register, VIC_INT_STATUS, can be read to figure out whether a FIFO queue or a TSU caused the interrupt. This can be used to figure out which INT_STATUS register should be read.

TM3200 Media Core Processor Interval Timer

Any of the GPIO or internal inputs can be switched onto the GPIO outputs gpio_timer[1:0] by programming the TIMER_IO_SEL register.

This output can be selected and used in the TM3200 core interval timer.

10.3 Register Descriptions

The base address for the GPIO module in the PNX8526 is 0x10 4000.

Remark: ALL programmable fields related to FIFO Queue or TSU operation are assumed static when the relevant FIFO queue or TSU is enabled. This excludes the fields BASE1, BASE2, BUF_LEN and Interrupt control registers.

10.3.1 Register Address Map

Table 7: GPIO Module Register Summary

Offset	Name	Description
0x10 4000	Mode Control 0	The Mode Control bit pairs which control GPIO pins 15-0.
0x10 4004	Mode Control 1	The Mode Control bit pairs which control GPIO pins 31-16.
0x10 4008	Mode Control 2	The Mode Control bit pairs which control GPIO pins 47-32.
0x10 400C	Mode Control 3	The Mode Control bit pairs which control GPIO pins 60-48.
0x10 4010	Mask and IO Data 0	Mask and IO data for GPIO pins 15-0.
0x10 4014	Mask and IO Data 1	Mask and IO data for GPIO pins 31-16.
0x10 4018	Mask and IO Data 2	Mask and IO data for GPIO pins 47-32.
0x10 401C	Mask and IO Data 3	Mask and IO data for GPIO pins 60-48.
0x10 4020	Internal Signals	Internal signals to be timestamped, software readable.
0x10 4024	GPIO_EV0	GPIO signal monitoring OR pattern generation control register for FIFO queue 0.
0x10 4028	GPIO_EV1	GPIO signal monitoring OR pattern generation control register for FIFO queue 1.
0x10 402C	GPIO_EV2	GPIO signal monitoring OR pattern generation control register for FIFO queue 2.
0x10 4030	GPIO_EV3	GPIO signal monitoring OR pattern generation control register for FIFO queue 3.
0x10 4034	GPIO_EV4	GPIO signal monitoring control register for timestamp unit 0
0x10 4038	GPIO_EV5	GPIO signal monitoring control register for timestamp unit 1
0x10 403C	GPIO_EV6	GPIO signal monitoring control register for timestamp unit 2
0x10 4040	GPIO_EV7	GPIO signal monitoring control register for timestamp unit 3
0x10 4044	GPIO_EV8	GPIO signal monitoring control register for timestamp unit 4
0x10 4048	GPIO_EV9	GPIO signal monitoring control register for timestamp unit 5
0x10 404C	GPIO_EV10	GPIO signal monitoring control register for timestamp unit 6
0x10 4050	GPIO_EV11	GPIO signal monitoring control register for timestamp unit 7
0x10 4054	GPIO_EV12	GPIO signal monitoring control register for timestamp unit 8
0x10 4058	GPIO_EV13	GPIO signal monitoring control register for timestamp unit 9
0x10 405C	GPIO_EV14	GPIO signal monitoring control register for timestamp unit 10
0x10 4060	GPIO_EV15	GPIO signal monitoring control register for timestamp unit 11
0x10 4064	IO_SEL0	IO Select register for FIFO queue 0
0x10 4068	IO_SEL1	IO Select register for FIFO queue 1
0x10 406C	IO_SEL2	IO Select register for FIFO queue 2
0x10 4070	IO_SEL3	IO Select register for FIFO queue 3
0x10 4074	PG_BUF_CTRL0	Pattern Generation DMA buffer control register. for FIFO queue 0
0x10 4078	PG_BUF_CTRL1	Pattern Generation DMA buffer control register. for FIFO queue 1
0x10 407C	PG_BUF_CTRL2	Pattern Generation DMA buffer control register for FIFO queue 2.
0x10 4080	PG_BUF_CTRL3	Pattern Generation DMA buffer control register for FIFO queue 3.
0x10 4084	BASE1_PTR0	Base address for DMA buffer 1 of FIFO queue 0.
0x10 4088	BASE1_PTR1	Base address for DMA buffer 1 of FIFO queue 1.
0x10 408C	BASE1_PTR2	Base address for DMA buffer 1 of FIFO queue 2.

Table 7: GPIO Module Register Summary *…Continued*

Offset	Name	Description
0x10 4090	BASE1_PTR3	Base address for DMA buffer 1 of FIFO queue 3.
0x10 4094	BASE2_PTR0	Base address for DMA buffer 2 of FIFO queue 0.
0x10 4098	BASE2_PTR1	Base address for DMA buffer 2 of FIFO queue 1.
0x10 409C	BASE2_PTR2	Base address for DMA buffer 2 of FIFO queue 2.
0x10 40A0	BASE2_PTR3	Base address for DMA buffer 2 of FIFO queue 3.
0x10 40A4	SIZE0	Size of queue 0 in bytes.
0x10 40A8	SIZE1	Size of queue 1 in bytes.
0x10 40AC	SIZE2	Size of queue 2 in bytes.
0x10 40B0	SIZE3	Size of queue 3 in bytes.
0x10 40B4	DIVIDER_0	Frequency divider for FIFO queue 0
0x10 40B8	DIVIDER_1	Frequency divider for FIFO queue 1
0x10 40BC	DIVIDER_2	Frequency divider for FIFO queue 2
0x10 40C0	DIVIDER_3	Frequency divider for FIFO queue 3
0x10 40C4	TSU0	Timestamp Unit 0.
0x10 40C8	TSU1	Timestamp Unit 1
0x10 40CC	TSU2	Timestamp Unit 2
0x10 40D0	TSU3	Timestamp Unit 3
0x10 40D4	TSU4	Timestamp Unit 4
0x10 40D8	TSU5	Timestamp Unit 5
0x10 40DC	TSU6	Timestamp Unit 6
0x10 40E0	TSU7	Timestamp Unit 7
0x10 40E4	TSU8	Timestamp Unit 8
0x10 40E8	TSU9	Timestamp Unit 9
0x10 40EC	TSU10	Timestamp Unit 10.
0x10 40F0	TSU11	Timestamp Unit 11
0x10 40F4	TIME_CTR	31-bit timestamp master time counter. Runs at 13.5MHz (108Mhz/8).
0x10 40F8	TIMER_IO_SEL	Selects inputs to be output onto gpio_timer[1:0]
0x10 40FC	VIC_INT_STATUS	Combined Interrupt status register for the VIC interrupts
0x10 4100—4F9C	Reserved	-
0x10 4FA0	INT_STATUS0	Interrupt status register, combined with module status for FIFO queue 0
0x10 4FA4	INT_ENABLE0	Interrupt enable register for FIFO queue 0
0x10 4FA8	INT_CLEAR0	Interrupt clear register (by software) for FIFO queue 0
0x10 4FAC	INT_SET0	Interrupt set register (by software) for FIFO queue 0
0x10 4FB0	INT_STATUS1	Interrupt status register, combined with module status for FIFO queue 1
0x10 4FB4	INT_ENABLE1	Interrupt enable register for FIFO queue 1
0x10 4FB8	INT_CLEAR1	Interrupt clear register (by software) for FIFO queue 1
0x10 4FBC	INT_SET1	Interrupt set register (by software) for FIFO queue 1
0x10 4FC0	INT_STATUS2	Interrupt status register, combined with module status for FIFO queue 2

Table 7: GPIO Module Register Summary *…Continued*

Offset	Name	Description
0x10 4FC4	INT_ENABLE2	Interrupt enable register for FIFO queue 2
0x10 4FC8	INT_CLEAR2	Interrupt clear register (by software) for FIFO queue 2
0x10 4FCC	INT_SET2	Interrupt set register (by software) for FIFO queue 2
0x10 4FD0	INT_STATUS3	Interrupt status register, combined with module status for FIFO queue 3
0x10 4FD4	INT_ENABLE3	Interrupt enable register for FIFO queue 3
0x10 4FD8	INT_CLEAR3	Interrupt clear register (by software) for FIFO queue 3
0x10 4FDC	INT_SET3	Interrupt set register (by software) for FIFO queue 3
0x10 4FE0	INT_STATUS4	Interrupt status register, combined with module status for TSUs
0x10 4FE4	INT_ENABLE4	Interrupt enable register for TSUs
0x10 4FE8	INT_CLEAR4	Interrupt clear register (by software) for TSUs
0x10 4FEC	INT_SET4	Interrupt set register (by software) for TSUs
0x10 4FF0	Reserved	-
0x10 4FF4	POWERDOWN	Powerdown mode, module clock switched off.
0x10 4FF8	Reserved	-
0x10 4FFC	Module ID	Module Identification and revision information

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
GPIO Mode Control				
Offset 0x10 4000			*Mode Control for GPIOs 15—0*	
31:30	R/W	0b11	MC for GPIO number 15	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1. The following values apply to writing to all of the bit pairs: 00 = Retain current GPIO Mode of operation (will not overwrite current mode). Not readable 01 = Place pin in primary function mode (see MUX table). 10 = Place pin in GPIO function mode (see MUX table). 11 = Place pin in GPIO function with open-drain output mode.
29:28	R/W	0b11	MC for GPIO number 14	
27:26	R/W	0b11	MC for GPIO number 13	
25:24	R/W	0b11	MC for GPIO number 12	
23:22	R/W	0b11	MC for GPIO number 11	
21:20	R/W	0b11	MC for GPIO number 10	
19:18	R/W	0b11	MC for GPIO number 09	
17:16	R/W	0b11	MC for GPIO number 08	
15:14	R/W	0b11	MC for GPIO number 07	
13:12	R/W	0b11	MC for GPIO number 06	
11:10	R/W	0b11	MC for GPIO number 05	
9:8	R/W	0b11	MC for GPIO number 04	
7:6	R/W	0b11	MC for GPIO number 03	
5:4	R/W	0b11	MC for GPIO number 02	
3:2	R/W	0b11	MC for GPIO number 01	
1:0	R/W	0b11	MC for GPIO number 00	

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 4004			*Mode Control for GPIOs 31—16*	
31:30	R/W	0b11	MC for GPIO number 31	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1. The following values apply to writing to all of the bit pairs: 00 = Retain current GPIO Mode of operation (will not overwrite current mode). Not readable 01 = Place pin in primary function mode (see MUX table). 10 = Place pin in GPIO function mode (see MUX table). 11 = Place pin in GPIO function with open-drain output mode.
29:28	R/W	0b11	MC for GPIO number 30	
27:26	R/W	0b11	MC for GPIO number 29	
25:24	R/W	0b11	MC for GPIO number 28	
23:22	R/W	0b11	MC for GPIO number 27	
21:20	R/W	0b11	MC for GPIO number 26	
19:18	R/W	0b11	MC for GPIO number 25	
17:16	R/W	0b11	MC for GPIO number 24	
15:14	R/W	0b11	MC for GPIO number 23	
13:12	R/W	0b11	MC for GPIO number 22	
11:10	R/W	0b11	MC for GPIO number 21	
9:8	R/W	0b11	MC for GPIO number 20	
7:6	R/W	0b11	MC for GPIO number 19	
5:4	R/W	0b11	MC for GPIO number 18	
3:2	R/W	0b11	MC for GPIO number 17	
1:0	R/W	0b11	MC for GPIO number 16	
Offset 0x10 4008			*Mode Control for GPIOs 47—32*	
31:30	R/W	0b11	MC for GPIO number 47	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1. The following values apply to writing to all of the bit pairs: 00 = Retain current GPIO Mode of operation (will not overwrite current mode). Not readable 01 = Place pin in primary function mode (see MUX table). 10 = Place pin in GPIO function mode (see MUX table). 11 = Place pin in GPIO function with open-drain output mode.
29:28	R/W	0b11	MC for GPIO number 46	
27:26	R/W	0b11	MC for GPIO number 45	
25:24	R/W	0b11	MC for GPIO number 44	
23:22	R/W	0b11	MC for GPIO number 43	
21:20	R/W	0b11	MC for GPIO number 42	
19:18	R/W	0b11	MC for GPIO number 41	
17:16	R/W	0b11	MC for GPIO number 40	
15:14	R/W	0b11	MC for GPIO number 39	
13:12	R/W	0b11	MC for GPIO number 38	
11:10	R/W	0b11	MC for GPIO number 37	
9:8	R/W	0b11	MC for GPIO number 36	
7:6	R/W	0b11	MC for GPIO number 35	
5:4	R/W	0b11	MC for GPIO number 34	
3:2	R/W	0b11	MC for GPIO number 33	
1:0	R/W	0b11	MC for GPIO number 32	
Offset 0x10 400C			*Mode Control for GPIOs 60—48*	
31:26		-	Unused	

GPIO REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
25:24	R/W	0b11	MC for GPIO number 60	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1. The following values apply to writing to all of the bit pairs: 00 = Retain current GPIO Mode of operation (will not overwrite current mode). Not readable 01 = Place pin in primary function mode (see MUX table). 10 = Place pin in GPIO function mode (see MUX table). 11 = Place pin in GPIO function with open-drain output mode.
23:22	R/W	0b11	MC for GPIO number 59	
21:20	R/W	0b11	MC for GPIO number 58	
19:18	R/W	0b11	MC for GPIO number 57	
17:16	R/W	0b11	MC for GPIO number 56	
15:14	R/W	0b11	MC for GPIO number 55	
13:12	R/W	0b11	MC for GPIO number 54	
11:10	R/W	0b11	MC for GPIO number 53	
9:8	R/W	0b11	MC for GPIO number 52	
7:6	R/W	0b11	MC for GPIO number 51	
5:4	R/W	0b11	MC for GPIO number 50	
3:2	R/W	0b11	MC for GPIO number 49	
1:0	R/W	0b11	MC for GPIO number 48	

GPIO REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
GPIO Data Control				
Offset 0x10 4010			***Mask and IO Data for GPIOs 15—0***	
31:16	R/W	0x0000	MASK[15:00]	Mask IOD
15:0	R/W	0xFFFF	IOD[15:00]	00 Retains current stored data (will not overwrite current data). Not readable. 01 Sets the corresponding GPIO pin in tri-statemode, allowing the pin to be used as data input. 10GPIO output mode. Drives a generated pattern (if enabled) or IOD (‘0’ in this case) onto the corresponding GPIO pin. 11 GPIO output mode. Drives a generated pattern (if enabled) or IOD (‘1’ in this case) onto the corresponding GPIO pin. Note: If open-drain mode is selected, drive to ‘1’ is disabled.
Offset 0x10 4014			***Mask and IO Data for GPIOs 31—16***	
31:16	R/W	0x0000	MASK[31:16]	See Mask and IO Data for GPIOs 15-0 (0x10 4010) for bit descriptions.
15:0	R/W	0xFFFF	IOD[31:16]	
Offset 0x10 4018			***Mask and IO Data for GPIOs 47—32***	
31:16	R/W	0x0000	MASK[47:32]	See Mask and IO Data for GPIOs 15-0 (0x10 4010) for bit descriptions.
15:0	R/W	0xFFFF	IOD[47:32]	

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 401C			*Mask and IO Data for GPIOs 60—48*	
31:29		-	Unused	See Mask and IO Data for GPIOs 15-0 (0x10 4010) for bit descriptions.
28:16	R/W	0x0000	MASK[60:48]	
15:13		-	Unused	See Mask and IO Data for GPIOs 15-0 (0x10 4010) for bit descriptions.
12:0	R/W	0xFFFF	IOD[60:48]	

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Readable Internal PNX8526 Signals				
Offset 0x10 4020			*Internal Signals*	
31:15		-	Unused]	
19	R	0	last_word_q3	Reads value of GPIO’s last 32-bit word timestamp for Queue 3.
18	R	0	last_word_q2	Reads value of GPIO’s last 32-bit word timestamp for Queue 2.
17	R	0	last_word_q1	Reads value of GPIO’s last 32-bit word timestamp for Queue 1.
16	R	0	last_word_q0	Reads value of GPIO’s last 32-bit word timestamp for Queue 0.
15	R	0	tsdma_tstamp2	Reads value of TSDMA timestamp 2 signal input to the GPIO module.
14	R	0	tsdma_tstamp1	Reads value of TSDMA timestamp 1 signal input to the GPIO module.
13	R	0	vip2_eow_aux	Reads value of VIP2 end of aux window timestamp signal input to the GPIO module.
12	R	0	vip2_eow_vid	Reads value of VIP2 end of vid window timestamp signal input to the GPIO module.
11	R	0	vip1_eow_aux	Reads value of VIP 1 end of aux window timestamp signal input to the GPIO module.
10	R	0	vip1_eow_vid	Reads value of VIP 1 end of vid window timestamp signal input to the GPIO module.
9	R	0	icp2_tstamp	Reads value of ICP 2 timestamp signal input to the GPIO module.
8	R	0	icp1_tstamp	Reads value of ICP 1 timestamp signal input to the GPIO module.
7	R	0	spdi_tstamp2	Reads value of SPDIF IN timestamp 2 signal input to the GPIO module.
6	R	0	spdi_tstamp1	Reads value of SPDIF IN timestamp 1 (word select timestamp) signal input to the GPIO module.
5	R	0	spdo_tstamp	Reads value of SPDIF OUT timestamp signal input to the GPIO module.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R	0	ai2_tstamp	Reads value of Audio IN 2 timestamp signal input to the GPIO module.
3	R	0	ai1_tstamp	Reads value of Audio IN 1 timestamp signal input to the GPIO module.
2	R	0	ao2_tstamp	Reads value of Audio OUT 2 timestamp signal input to GPIO module.
1	R	0	ao1_tstamp	Reads value of Audio OUT 1 timestamp signal input to GPIO module.
0	R	0	aio_tstamp	Reads value of Audio IO IN timestamp signal input to the GPIO module.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 4024			*GPIO_EV0*	
31		-	Unused	
30	R/W	0	EN_EV_TSTAMP	Enables an event timestamp signal to be generated whenever the last 32-bit word from a DMA buffer reaches the GPIO output pins. This field is only valid in Pattern Generating modes (FIFO_MODE[1]=1).
29	R/W	0	EN_IR_CARRIER	Enables a subcarrier IR transmission. FREQ_DIV[15:0] is combined with CARRIER_DIV[4:0] to generated sub-carrier and TX frequencies. 0 = IR Carrier disabled, CARRIER_DIV[4:0] not used. 1 = IR Carrier enabled, CARRIER_DIV[4:0] used. Note: This field is only valid in Pattern Generation using sample mode (FIFO_MODE=11) with EN_CLOCK_SEL disabled.
28	R/W	0	EN_IR_FILTER	Enables a received IR signal to be filtered. No signal pulses less than the period programmed in IR_FILTER are passed through to the monitoring logic. Note: This field is only valid in Signal Sampling mode (FIFO_MODE=01) with EN_CLOCK_SEL disabled.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
27:26	R/W	0	EN_CLOCK_SEL	In Signal Sampling mode, this bit enables an input signal selected by CLOCK_SEL to be used to sample the input signal selected by IO_SEL. 00,10 = CLOCK_SEL disabled. 01 = CLOCK_SEL enabled, sample on positive edge. 11 = CLOCK_SEL enabled, sample on negative edge. In Pattern Generation using sample mode, this bit enables an output signal selected by CLOCK_SEL to be used to output the sampling frequency. This feature is valid if sampling frequency is less than 108 MHz. 00,10 = CLOCK_SEL disabled. 01 = CLOCK_SEL enabled, output clk at sampling freq. 11 = CLOCK_SEL enabled, output inverted clk at sampling freq. Note: This field is only valid in Signal Sampling mode (FIFO_MODE=01) and Pattern Generation using sample mode (FIFO_MODE=11).
25:18	R/W	0	CLOCK_SEL	In Signal Sampling mode, this field selects the GPIO input pin to be used to ‘sample’ the signal(s) selected by IO_SEL. Note: The GPIO input used for sampling must be slower than 108 MHz. In Pattern Generation using sample mode, this field selects which GPIO output pin to output the sampling frequency clock on. Note: This field is only valid in Signal Sampling mode (FIFO_MODE=01) and Pattern Generation using sample mode (FIFO_MODE=11). Refer to Table 8 for field values.
17:16	R/W	0	EN_IO_SEL	This field selects how many GPIO pins should be sampled in one FIFO queue: 00,11 = IO_SEL_0 enabled: 1-bit samples. 01 = IO_SEL_0 and IO_SEL_1 enabled: 2-bit samples. 10 = IO_SEL_0, IO_SEL_1, IO_SEL_2 and IO_SEL_3 enabled: 4-bit samples. Note: This field is only valid in Signal Sampling mode (FIFO_MODE=01) or Pattern Generation using sample mode (FIFO_MODE=11). In all other modes only IO_SEL_0 is enabled.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:4	R/W	0	INTERVAL	Interval of silence. If a change is monitored on a signal and no more signal activity is monitored for a time equal to the interval of silence, writing to the current buffer is halted and a BUFx_RDY interrupt is generated. Writing continues to the alternate buffer. This field is only valid if FIFO_MODE[1:0] = 00. 0x000 = Disabled. 0x001 = 1x128 13.5 MHz period, 9.48 µs 0x002 = 2x128 13.5 MHZ periods, 18.96 µs 0x003 = 3x128 13.5 MHZ periods, 28,44 µs .... 0x3FFh = 1023x128 13.5 MHz period, 9.69 ms .... 0xFFF = 4095x128 13.5 MHz period, 38.8 ms Note: Field in only valid in Event Timestamping mode (FIFO_MODE=00, EVENT_MODE>00).
3:2	R/W	0	EVENT_MODE	Timestamping event mode: 00 = Event detection disabled. 01 = Capture negative edge. 10 = Capture positive edge. 11 = Capture either edge. Note:F ield is valid in Event Timestamping mode(FIFO_MODE[1:0]=00).
1:0	R/W	0	FIFO_MODE	This bit selects what mode of operation the FIFO queue is in: 00 = Event Timestamping (or Disabled if EVENT_MODE[1:0] = 00) 01 = Signal Sampling 10 = Pattern Generation using timestamps. 11 = Pattern Generation using samples.
Offset 0x10 4028			***GPIO_EV1***	
Refer to GPIO_EV0 at offset 0x10 4024 for descriptions.				
Offset 0x10 402C			***GPIO_EV2***	
Refer to GPIO_EV0 at offset 0x10 4024 for descriptions.				
Offset 0x10 4030			***GPIO_EV3***	
Refer to GPIO_EV0 at offset 0x10 4024 for descriptions.				
Signal Monitoring Control Registers for the Timestamp Units				
Offset 0x10 4034			***GPIO_EV4***	
31:10		-	Unused	
9:2	R/W	0	IO_SEL	This field selects the GPIO pin or internal global signal to be monitored. Refer to Table 8 for field values. Note: IO_SEL cannot be written to unless timestamping is disabled (EVENT_MODE=00).

GPIO REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
1:0	R/W	0	EVENT_MODE	Timestamping event mode: 00 = Event detection disabled. 01 = Capture negative edge. 10 = Capture positive edge. 11 = Capture either edge.
Offset 0x10 4038			***GPIO_EV5***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 403C			***GPIO_EV6***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4040			***GPIO_EV7***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4044			***GPIO_EV8***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4048			***GPIO_EV9***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 404C			***GPIO_EV10***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4050			***GPIO_EV11***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4054			***GPIO_EV12***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4058			***GPIO_EV13***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 405C			***GPIO_EV14***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				
Offset 0x10 4060			***GPIO_EV15***	
Refer to GPIO_EV4 at offset 0x10 4034 for descriptions.				

GPIO REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Monitoring and Pattern Generation Control Registers				
Offset 0x10 4064			***IO_SEL0***	
Note: IO_SEL cannot be written to unless all signal generation and monitoring is disabled (FIFO_MODE=00;EVENT_MODE=00).				

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:24	R/W	0	IO_SEL_3	This field selects a GPIO pin which should be merged with the GPIO pin selected by IO_SEL_0, IO_SEL_1 and IO_SEL_2 to enable 4-bit samples in one FIFO queue. This field is only used in Signal Sampling mode and Pattern Generation using sample mode and is enabled by EN_IO_SEL
23:16	R/W	0	IO_SEL_2	This field selects a GPIO pin which should be merged with the GPIO pins selected by IO_SEL_0, IO_SEL_1 and IO_SEL_3 to enable 4-bit samples in one FIFO queue. This field is only used in Signal Sampling mode and Pattern Generation using sample mode and is enabled by EN_IO_SEL
15:8	R/W	0	IO_SEL_1	This field selects a GPIO pin which should be merged with the GPIO pin selected by IO_SEL_0 to enable 2-bit samples in one FIFO queue. This field is only used in Signal Sampling mode and Pattern Generation using sample mode and is enabled by EN_IO_SEL
7:0	R/W	0	IO_SEL_0	In Signal Monitoring modes (FIFO_MODE[1]=0) this field selects the GPIO pin or internal global signal to be observed. In Pattern Generation modes (FIFO_MODE[1]=1) this field selects the GPIO pin which is to be driven. Refer to Table 8 for field values.
Offset 0x10 4068			*IO_SEL1*	
Refer to IO_SEL0 at offset 0x10 4064 for descriptions.				
Offset 0x10 406C			*IO_SEL2*	
Refer to IO_SEL0 at offset 0x10 4064 for descriptions.				
Offset 0x10 4070			*IO_SEL3*	
Refer to IO_SEL0 at offset 0x10 4064 for descriptions.				
Offset 0x10 4074			*PG_BUF_CTRL0*	
31:18		-	Unused	
17:0	R/W	0	BUF_LEN	This field indicates how many valid 32-bit words software has written to a DMA buffer. When BUF1_RDY is cleared, the BUF_LEN value is loaded for DMA buffer 1. When BUF2_RDY is cleared, the BUF_LEN value is loaded for DMA buffer 2. The 18-bit field allows DMA buffer lengths as large as 1 MB. 0x00000 = One 32-bit word 0x00001 = Two 32-bit words ..... 0x3FFFF = 262143 32-bit words Note:This field is valid in Pattern Generation modes(FIFO_MODE[1]=1).
Offset 0x10 4078			*PG_BUF_CTRL1*	
Refer to PG_BUF_CTRL0 at offset 0x10 4074 for descriptions.				
Offset 0x10 407C			*PG_BUF_CTRL2*	
Refer to PG_BUF_CTRL0 at offset 0x10 4074 for descriptions.				

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 4080			***PG_BUF_CTRL3***	
Refer to PG_BUF_CTRL0 at offset 0x10 4074 for descriptions.				
Offset 0x10 4084			***BASE1_PTR0***	
31:2	R/W	0	BASE1_PTR	Start byte address for DMA buffer 1 of FIFO queue.
1:0		-	Unused	
Offset 0x10 4088			***BASE1_PTR1***	
Refer to BASE1_PTR0 at offset 0x10 4084 for descriptions.				
Offset 0x10 408C			***BASE1_PTR2***	
Refer to BASE1_PTR0 at offset 0x10 4084 for descriptions.				
Offset 0x10 4090			***BASE1_PTR3***	
Refer to BASE1_PTR0 at offset 0x10 4084 for descriptions.				
Offset 0x10 4094			***BASE2_PTR0***	
31:2	R/W	0	BASE2_PTR	Start byte address for DMA buffer 2 of FIFO queue.
1:0		-	Unused	
Offset 0x10 4098			***BASE2_PTR1***	
Refer to BASE2_PTR0 at offset 0x10 4094 for descriptions.				
Offset 0x10 409C			***BASE2_PTR2***	
Refer to BASE2_PTR0 at offset 0x10 4094 for descriptions.				
Offset 0x10 40A0			***BASE2_PTR3***	
Refer to BASE2_PTR0 at offset 0x10 4094 for descriptions.				
Offset 0x10 40A4			***SIZE0***	
31:20		-	Unused	
13:0	R/W	0	SIZE	Size, in 64-byte multiples, of each of the two DMA buffers. 0x0001 = 64 bytes 0x0002 = 128 bytes .... 0x3FFF = 1 MB
Offset 0x10 40A8			***SIZE1***	
Refer to SIZE0 at offset 0x10 40A4 for descriptions.				
Offset 0x10 40AC			***SIZE2***	
Refer to SIZE0 at offset 0x10 40A4 for descriptions.				
Offset 0x10 40B0			***SIZE3***	
Refer to SIZE0 at offset 0x10 40A4 for descriptions.				
Offset 0x10 40B4			***DIVIDER0***	
31:23		-	Unused	

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
22:21	R/W	0	DUTY_CYCLE	This field selects the duty cycle for subcarriers. 00 = 33% duty-cycle 01 = 50% duty-cycle 10 = 66% duty cycle 11 = Illegal This field is only valid in pattern generation modes and when EN_IR_CARRIER=1.
20:16	R/W	00	CARRIER_DIV (when FIFO_MODE[1] =1)	Used in IR TX applications if a subcarrier is required for transmission. To enable this divider EN_IR_CARRIER=1. If enabled, the IR subcarrier frequency is defined by programming FREQ_DIV. The ‘ONTIME’ is defined by FREQ_DIV x CARRIER_DIV. 0x00, 0x01 = Disabled. 0x02 = Sampling frequency is FREQ_DIV/2. ..... 0x1F = Sampling frequency is FREQ_DIV/31.
18:16		00	IR_FILTER (when FIFO_MODE[1] =0)	Used in IR RX applications to filter a received IR signal. To enable this divider EN_IR_FILTER=1. If enabled, IR pulses greater than IR_FILTER are passed through to the signal monitoring logic. 0x0 = 54/108 MHz, 0.5 µs 0x1 = 108/108 MHz, 1 µs 0x2 = 162/108 MHz, 1.5 µs 0x3 = 216/108 MHz, 2 µs 0x4 = 270/108 MHz, 2.5 µs 0x5 = 324/108 MHz, 3 µs 0x6 = 378/108 MHz, 3.5 µs 0x7 = 432/108 MHz, 4 µs Note: The filter operates on one input per queue. This bit is the input selected by IO_SEL[7:0]. If used in multi-bit sampling modes (IO_SEL_EN = 01/10) be aware that the filtered signal is delayed by the selected IR_FILTER value with respect to the other signals sampled in the queue.
15:0	R/W	0000	FREQ_DIV	16-bit Frequency Divider for signal sampling and pattern generation using samples. If EN_CARRIER_FREQ = 0 Sampling Freq. = 108 MHz/ FREQ_DIV 0x0000 = Disabled. 0x0001 = Sampling frequency is 108 MHz 0x0002 = Sampling/Carrier frequency is 54 MHz ..... 0xFFFF = Sampling/Carrier frequency is 1.648 kHz If EN_CARRIER_FREQ = 1 Carrier Freq. = 54 MHz/FREQ_DIV 0x0000 = Disabled. 0x0001 = Carrier frequency is 54 MHz 0x0002 = Carrier frequency is 26 MHz ..... 0xFFFF = Carrier frequency is 824 Hz

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 40B8			***DIVIDER1***	
Refer to DIVIDER0 at offset 0x10 40B4 for descriptions.				
Offset 0x10 40BC			***DIVIDER2***	
Refer to DIVIDER0 at offset 0x10 40B4 for descriptions.				
Offset 0x10 40C0			***DIVIDER3***	
Refer to DIVIDER0 at offset 0x10 40B4 for descriptions.				

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Timestamp Unit Registers				
Offset 0x10 40C4			***TSU0***	
31	R	0	Direction	This field indicates the direction of the event which occurred: 0 = A falling edge 1 = A rising edge
30:0	R	0	Timestamp	This field holds the 31-bit timestamp.
Offset 0x10 40C8			***TSU1***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40CC			***TSU2***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40D0			***TSU3***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40D4			***TSU4***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40D8			***TSU5***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40DC			***TSU6***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40E0			***TSU7***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40E4			***TSU8***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40E8			***TSU9***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
Offset 0x10 40EC			***TSU10***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 40F0			***TSU11***	
Refer to TSU0 at offset 0x10 40C4 for descriptions.				
GPIO Time Counter				
Offset 0x10 40F4			***TIME_CTR***	
31		-	Unused	
30:0	R	0	TIME_CTR	GPIO master time counter. It is incremented at a frequency of 13.5 MHz.
GPIO TM Timer Input Select				
Offset 0x10 40F8			***TIMER_IO_SEL***	
31:16		-	Unused	
15:8	R/W	0	TIMER_IO_SEL1	Selects GPIO or internal input to be output onto gpio_timer[1]. See Table 1 for valid values.
7:0	R/w	0	TIMER_IO_SEL0	Selects GPIO or internal input to be output onto gpio_timer[0]. See Table 1 for valid values.
VIC Interrupt Status				
Offset 0x10 40FC			***VIC_INT_STATUS***	
31:5		-	Unused	
4	R	0	TSU Status	TSU status bit for interrupts of the 12 TSUs (ORed together)
3	R	0	FIFO Queue 3 status	FIFO Queue 3 status bit for all interrupts (ORed together)
2	R	0	FIFO Queue 2 status	FIFO Queue 2 status bit for all interrupts (ORed together)
1	R	0	FIFO Queue 1 status	FIFO Queue 1 status bit for all interrupts (ORed together)
0	R	0	FIFO Queue 0 status	FIFO Queue 0 status bit for all interrupts (ORed together)
Offset 0x10 4100—4F9C			***Reserved***	

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
GPIO Interrupt Registers for the FIFO Queues				
Offset 0x10 4FA0			***INT_STATUS0***	
31:14	R	0	VALID_PTR	Indicates how many valid 32-bit words of data have been written by the GPIO to the current DMA buffer. 0x00000 = One 32-bit word 0x00001 = Two 32-bit words ..... 0x3FFFF = 262143 32-bit words When a BUFx_RDY signal occurs, the address to read from can be calculated using VALID_PTR. This field is only updated by the GPIO after the relevant BUFx_RDY flag is cleared by software. It is valid in SIgnal Monitoring modes only.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
5:4	R	0	Reserved	
3	R	0	INT_OE	Internal overrun error. Internal GPIO data buffer has overrun before data has been written to external DMA buffer. Data has been lost. ONLY USED in signal monitoring modes.
2	R	0	FIFO_OE	FIFO overrun error. A new, empty DMA buffer was not supplied in time. ONLY USED in signal monitoring modes.
1	R	0	BUF2_RDY	In Signal Monitoring mode: DMA buffer 2 is ready to be read. It is either full or an interval of silence has occurred. In Pattern Generation mode: Contents of DMA buffer 2 have been read.
0	R	0	BUF1_RDY	In Signal Monitoring mode: DMA buffer1 is ready to be read. It is either full or an interval of silence has occurred. In Pattern Generation mode: Contents of DMA buffer 1 have been read.
Offset 0x10 4FA4			***INT_ENABLE0***	
31:4		-	Unused	
3	R/W	0	INT_OE_EN	Active high Internal overrun error interrupt enable for queue 0
2	R/W	0	FIFO_OE_EN	Active high FIFO overrun error interrupt enable for queue 0
1	R/W	0	BUF2_RDY_EN	Active high Buffer 2 ready interrupt enable for queue 0
0	R/W	0	BUF1_RDY_EN	Active high Buffer 1 ready interrupt enable for queue 0
Offset 0x10 4FA8			***INT_CLEAR0***	
31:4		-	Unused	
3	W	0	INT_OE_CLR	Active high internal overrun error interrupt clear for queue 0
2	W	0	FIFO_OE_CLR	Active high FIFO overrun error interrupt clear for queue 0
1	W	0	BUF2_RDY_CLR	Active high Buffer 2 ready interrupt clear for queue 0
0	W	0	BUF1_RDY_CLR	Active high Buffer 1 ready interrupt clear for queue 0
Offset 0x10 4FAC			***INT_SET0***	
31:4		-	Unused	
3	W	0	INT_OE_SET	Active high internal overrun error interrupt set for queue 0
2	W	0	FIFO_OE_SET	Active high FIFO overrun error interrupt set for queue 0
1	W	0	BUF2_RDY_SET	Active high Buffer 2 ready interrupt set for queue 0
0	W	0	BUF1_RDY_SET	Active high Buffer 1 ready interrupt set for queue 0
Offset 0x10 4FB0			***INT_STATUS1***	
Refer to INT_STATUS0 at offset 0x10 4FA0 for descriptions.				
Offset 0x10 4FB4			***INT_ENABLE1***	
Refer to INT_ENABLED0 at offset 0x10 4FA4 for descriptions.				
Offset 0x10 4FB8			***INT_CLEAR1***	
Refer to INT_CLEAR0 at offset 0x10 4FA8 for descriptions.				

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 4FBC			*INT_SET1*	
Refer to INT_SET0 at offset 0x10 4FAC for descriptions.				
Offset 0x10 4FC0			*INT_STATUS2*	
Refer to INT_STATUS0 at offset 0x10 4FA0 for descriptions.				
Offset 0x10 4FC4			*INT_ENABLE2*	
Refer to INT_ENABLED0 at offset 0x10 4FA4 for descriptions.				
Offset 0x10 4FC8			*INT_CLEAR2*	
Refer to INT_CLEAR0 at offset 0x10 4FA8 for descriptions.				
Offset 0x10 4FCC			*INT_SET2*	
Refer to INT_SET0 at offset 0x10 4FAC for descriptions.				
Offset 0x10 4FD0			*INT_STATUS3*	
Refer to INT_STATUS0 at offset 0x10 4FA0 for descriptions.				
Offset 0x10 4FD4			*INT_ENABLE3*	
Refer to INT_ENABLED0 at offset 0x10 4FA4 for descriptions.				
Offset 0x10 4FD8			*INT_CLEAR3*	
Refer to INT_CLEAR0 at offset 0x10 4FA8 for descriptions.				
Offset 0x10 4FDC			*INT_SET3*	
Refer to INT_SET0 at offset 0x10 4FAC for descriptions.				

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
GPIO Module Status Register for the 12 Timestamp Units (TSU)				
Offset 0x10 4FE0			*INT_STATUS4*	
31:24		-	Unused	
23	R	0	INT_OE_11	Internal overrun error in TSU11. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
22	R	0	INT_OE_10	Internal overrun error in TSU10. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
21	R	0	INT_OE_9	Internal overrun error in TSU9. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
20	R	0	INT_OE_8	Internal overrun error in TSU8. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
19	R	0	INT_OE_7	Internal overrun error in TSU7. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
18	R	0	INT_OE_6	Internal overrun error in TSU6. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17	R	0	INT_OE_5	Internal overrun error in TSU5. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
16	R	0	INT_OE_4	Internal overrun error in TSU4. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
15	R	0	INT_OE_3	Internal overrun error in TSU3. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
14	R	0	INT_OE_2	Internal overrun error in TSU2. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
13	R	0	INT_OE_1	Internal overrun error in TSU1. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
12	R	0	INT_OE_0	Internal overrun error in TSU0. Data in TSU overwritten before read by CPU (i.e., before DATA_VALID interrupt was cleared).
11	R	0	DATA_VALID_11	Data in TSU11 is ready to be read.
10	R	0	DATA_VALID_10	Data in TSU10 is ready to be read.
9	R	0	DATA_VALID_9	Data in TSU9 is ready to be read.
8	R	0	DATA_VALID_8	Data in TSU8 is ready to be read.
7	R	0	DATA_VALID_7	Data in TSU7 is ready to be read.
6	R	0	DATA_VALID_6	Data in TSU6 is ready to be read.
5	R	0	DATA_VALID_5	Data in TSU5 is ready to be read.
4	R	0	DATA_VALID_4	Data in TSU4 is ready to be read.
3	R	0	DATA_VALID_3	Data in TSU3 is ready to be read.
2	R	0	DATA_VALID_2	Data in TSU2 is ready to be read.
1	R	0	DATA_VALID_1	Data in TSU1 is ready to be read.
0	R	0	DATA_VALID_0	Data in TSU0 is ready to be read.
Offset 0x10 4FE4			*INT_ENABLE4*	
31:24		-	Unused	
23	R/W	0	INT_OE_11_EN	Internal overrun interrupt enable register for TSU11: 0 = Interrupt disabled. 1 = Interrupt enabled.
22	R/W	0	INT_OE_10_EN	Internal overrun interrupt enable register for TSU10: 0 = Interrupt disabled. 1 = Interrupt enabled.
21	R/W	0	INT_OE_9_EN	Internal overrun interrupt enable register for TSU9: 0 = Interrupt disabled. 1 = Interrupt enabled.
20	R/W	0	INT_OE_8_EN	Internal overrun interrupt enable register for TSU8: 0 = Interrupt disabled. 1 = Interrupt enabled.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19	R/W	0	INT_OE_7_EN	Internal overrun interrupt enable register for TSU7: 0 = Interrupt disabled. 1 = Interrupt enabled.
18	R/W	0	INT_OE_6_EN	Internal overrun interrupt enable register for TSU6: 0 = Interrupt disabled. 1 = Interrupt enabled.
17	R/W	0	INT_OE_5_EN	Internal overrun interrupt enable register for TSU5: 0 = Interrupt disabled. 1 = Interrupt enabled.
16	R/W	0	INT_OE_4_EN	Internal overrun interrupt enable register for TSU4: 0 = Interrupt disabled. 1 = Interrupt enabled.
15	R/W	0	INT_OE_3_EN	Internal overrun interrupt enable register for TSU3: 0 = Interrupt disabled. 1 = Interrupt enabled.
14	R/W	0	INT_OE_2_EN	Internal overrun interrupt enable register for TSU2: 0 = Interrupt disabled. 1 = Interrupt enabled.
13	R/W	0	INT_OE_1_EN	Internal overrun interrupt enable register for TSU1: 0 = Interrupt disabled. 1 = Interrupt enabled.
12	R/W	0	INT_OE_0_EN	Internal overrun interrupt enable register for TSU0: 0 = Interrupt disabled. 1 = Interrupt enabled.
11	R/W	0	DATA_VALID_11_EN	Data valid interrupt enable register for TSU11: 0 = Interrupt disabled. 1 = Interrupt enabled.
10	R/W	0	DATA_VALID_10_EN	Data valid interrupt enable register for TSU10: 0 = Interrupt disabled. 1 = Interrupt enabled.
9	R/W	0	DATA_VALID_9_EN	Data valid interrupt enable register for TSU9”: 0 = Interrupt disabled. 1 = Interrupt enabled.
8	R/W	0	DATA_VALID_8_EN	Data valid interrupt enable register for TSU8: 0 = Interrupt disabled. 1 = Interrupt enabled.
7	R/W	0	DATA_VALID_7_EN	Data valid interrupt enable register for TSU7: 0 = Interrupt disabled. 1 = Interrupt enabled.
6	R/W	0	DATA_VALID_6_EN	Data valid interrupt enable register for TSU6: 0 = Interrupt disabled. 1 = Interrupt enabled.

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
5	R/W	0	DATA_VALID_5_EN	Data valid interrupt enable register for TSU5: 0 = Interrupt disabled. 1 = Interrupt enabled.
4	R/W	0	DATA_VALID_4_EN	Data valid interrupt enable register for TSU4: 0 = Interrupt disabled. 1 = Interrupt enabled.
3	R/W	0	DATA_VALID_3_EN	Data valid interrupt enable register for TSU3: 0 = Interrupt disabled. 1 = Interrupt enabled.
2	R/W	0	DATA_VALID_2_EN	Data valid interrupt enable register for TSU2: 0 = Interrupt disabled. 1 = Interrupt enabled.
1	R/W	0	DATA_VALID_1_EN	Data valid interrupt enable register for TSU1: 0 = Interrupt disabled. 1 = Interrupt enabled.
0	R/W	0	DATA_VALID_0_EN	Data valid interrupt enable register for TSU0: 0 = Interrupt disabled. 1 = Interrupt enabled.
Offset 0x10 4FE8			*INT_CLEAR4*	
31:24		-	Unused	
23	W	0	INT_OE_11_CLR	Active high clear for internal overrun interrupt for TSU11
22	W	0	INT_OE_10_CLR	Active high clear for internal overrun interrupt for TSU10
21	W	0	INT_OE_9_CLR	Active high clear for internal overrun interrupt for TSU9
20	W	0	INT_OE_8_CLR	Active high clear for internal overrun interrupt for TSU8
19	W	0	INT_OE_7_CLR	Active high clear for internal overrun interrupt for TSU7
18	W	0	INT_OE_6_CLR	Active high clear for internal overrun interrupt for TSU6
17	W	0	INT_OE_5_CLR	Active high clear for internal overrun interrupt for TSU5
16	W	0	INT_OE_4_CLR	Active high clear for internal overrun interrupt for TSU4
15	W	0	INT_OE_3_CLR	Active high clear for internal overrun interrupt for TSU3
14	W	0	INT_OE_2_CLR	Active high clear for internal overrun interrupt for TSU2
13	W	0	INT_OE_1_CLR	Active high clear for internal overrun interrupt for TSU1
12	W	0	INT_OE_0_CLR	Active high clear for internal overrun interrupt for TSU0
11	W	0	DATA_VALID_11_CLR	Active high clear for data valid interrupt for TSU11
10	W	0	DATA_VALID_10_CLR	Active high clear for data valid interrupt for TSU10
9	W	0	DATA_VALID_9_CLR	Active high clear for data valid interrupt for TSU9
8	W	0	DATA_VALID_8_CLR	Active high clear for data valid interrupt for TSU8
7	W	0	DATA_VALID_7_CLR	Active high clear for data valid interrupt for TSU7
6	W	0	DATA_VALID_6_CLR	Active high clear for data valid interrupt for TSU6
5	W	0	DATA_VALID_5_CLR	Active high clear for data valid interrupt for TSU5

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	W	0	DATA_VALID_4_CLR	Active high clear for data valid interrupt for TSU4
3	W	0	DATA_VALID_3_CLR	Active high clear for data valid interrupt for TSU3
2	W	0	DATA_VALID_2_CLR	Active high clear for data valid interrupt for TSU2
1	W	0	DATA_VALID_1_CLR	Active high clear for data valid interrupt for TSU1
0	W	0	DATA_VALID_0_CLR	Active high clear for data valid interrupt for TSU0
Offset 0x10 4FEC			*INT_SET4*	
31:24		-	Unused	
23	W	0	INT_OE_11_SET	Active high set for internal overrun interrupt for TSU11
22	W	0	INT_OE_10_SET	Active high set for internal overrun interrupt for TSU10
21	W	0	INT_OE_9_SET	Active high set for internal overrun interrupt for TSU9
20	W	0	INT_OE_8_SET	Active high set for internal overrun interrupt for TSU8
19	W	0	INT_OE_7_SET	Active high set for internal overrun interrupt for TSU7
18	W	0	INT_OE_6_SET	Active high set for internal overrun interrupt for TSU6
17	W	0	INT_OE_5_SET	Active high set for internal overrun interrupt for TSU5
16	W	0	INT_OE_4_SET	Active high set for internal overrun interrupt for TSU4
15	W	0	INT_OE_3_SET	Active high set for internal overrun interrupt for TSU3
14	W	0	INT_OE_2_SET	Active high set for internal overrun interrupt for TSU2
13	W	0	INT_OE_1_SET	Active high set for internal overrun interrupt for TSU1
12	W	0	INT_OE_0_SET	Active high set for internal overrun interrupt for TSU0
11	W	0	DATA_VALID_11_SET	Active high set for data valid interrupt for TSU11
10	W	0	DATA_VALID_10_SET	Active high set for data valid interrupt for TSU10
9	W	0	DATA_VALID_9_SET	Active high set for data valid interrupt for TSU9
8	W	0	DATA_VALID_8_SET	Active high set for data valid interrupt for TSU8
7	W	0	DATA_VALID_7_SET	Active high set for data valid interrupt for TSU7
6	W	0	DATA_VALID_6_SET	Active high set for data valid interrupt for TSU6
5	W	0	DATA_VALID_5_SET	Active high set for data valid interrupt for TSU5
4	W	0	DATA_VALID_4_SET	Active high set for data valid interrupt for TSU4
3	W	0	DATA_VALID_3_SET	Active high set for data valid interrupt for TSU3
2	W	0	DATA_VALID_2_SET	Active high set for data valid interrupt for TSU2
1	W	0	DATA_VALID_1_SET	Active high set for data valid interrupt for TSU1
0	W	0	DATA_VALID_0_SET	Active high set for data valid interrupt for TSU0
Offset 0x10 4FF0			*Reserved*	

GPIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
GPIO POWERDOWN				
Offset 0x10 4FF4			*POWERDOWN*	
31	R/W	0	POWERDOWN	0 = Normal operation of peripheral. This is the reset value. 1 = Module is in powerdown and module clock can be removed. Module must respond to all reads. Module must respond to all reads. Generate e.g. DEADABBA (exept for reads of the powerdown bit). Module should generate ERR ACK on writes (except for writes to the powerdown bit).
30:0		-	Unused	
Offset 0x10 4FF8			*Reserved*	
GPIO Module ID				
Offset 0x10 4FFC			*MODULE_ID*	
31:15	R	0x010F	Module ID	Unique 16-bit code. Module ID 0 and -1 are reserved for future use.
14:12	R	0	MajRev	Major Revision
11:8	R	1	MinRev	Minor Revision
7:0	R	0	Module Aperture Size	Aperture size = 4 kB*(bit_value+1), so 0 means 4 kB (the default).

Table 8 shows selectable pins for signal monitoring and pattern generation.

Table 8: GPIO IO_SEL Values

Signal	CLOCK_SEL/ IO_SEL (hex)	MODE
C1394_SYNC0	0x52	Pattern Generation
C1394_SYNC1	0x51	Pattern Generation
LAST_WORD3	0x50	Signal Monitoring
LAST_WORD2	0x4f	Signal Monitoring
LAST_WORD1	0x4e	Signal Monitoring
LAST_WORD0	0x4d	Signal Monitoring
tsdma_tstamp2	0x4c	Signal Monitoring
tsdma_tstamp1	0x4b	Signal Monitoring
vip2_eow_aux	0x4a	Signal Monitoring
vip2_eow_vid	0x49	Signal Monitoring
vip1_eow_aux	0x48	Signal Monitoring
vip1_eow_vid	0x47	Signal Monitoring
icp2_tstamp	0x46	Signal Monitoring
icp1_tstamp	0x45	Signal Monitoring
spdi_tstamp2	0x44	Signal Monitoring

Table 8: GPIO IO_SEL Values *…Continued*

Signal	CLOCK_SEL/ IO_SEL (hex)	MODE
spdi_tstamp1	0x43	Signal Monitoring
spdo_tstamp	0x42	Signal Monitoring
ai2_tstamp	0x41	Signal Monitoring
ai1_tstamp	0x40	Signal Monitoring
ao2_tstamp	0x3F	Signal Monitoring
ao1_tstamp	0x3E	Signal Monitoring
aio_tstamp	0x3D	Signal Monitoring
PCI_GNT_B	0x3C	Signal Monitoring; Pattern Generation
PCI_GNT_A	0x3B	Signal Monitoring; Pattern Generation
PCI_REQ_B	0x3A	Signal Monitoring; Pattern Generation
PCI_REQ_A	0x39	Signal Monitoring; Pattern Generation
XI0_A25	0x38	Signal Monitoring; Pattern Generation
XI0_ACK	0x37	Signal Monitoring; Pattern Generation
XI0_SEL[2]	0x36	Signal Monitoring; Pattern Generation
XI0_SEL[1]	0x35	Signal Monitoring; Pattern Generation
XI0_SEL[0]	0x34	Signal Monitoring; Pattern Generation
AIO_SD[3]	0x33	Signal Monitoring; Pattern Generation
AIO_SD[2]	0x32	Signal Monitoring; Pattern Generation
AIO_SD[1]	0x31	Signal Monitoring; Pattern Generation
AIO_SD[0]	0x30	Signal Monitoring; Pattern Generation
AIO_WS	0x2F	Signal Monitoring; Pattern Generation
AIO_SCK	0x2E	Signal Monitoring; Pattern Generation
AIO_OSCLK	0x2D	Signal Monitoring; Pattern Generation
DV1_VALID	0x2C	Signal Monitoring; Pattern Generation
DV1_CLK	0x2B	Signal Monitoring; Pattern Generation
DV1_DATA[9]	0x2A	Signal Monitoring; Pattern Generation
DV1_DATA[8]	0x29	Signal Monitoring; Pattern Generation
DV1_DATA[7]	0x28	Signal Monitoring; Pattern Generation
DV1_DATA[6]	0x27	Signal Monitoring; Pattern Generation
DV1_DATA[5]	0x26	Signal Monitoring; Pattern Generation
DV1_DATA[4]	0x25	Signal Monitoring; Pattern Generation
DV1_DATA[3]	0x24	Signal Monitoring; Pattern Generation
DV1_DATA[2]	0x23	Signal Monitoring; Pattern Generation
DV1_DATA[1]	0x22	Signal Monitoring; Pattern Generation
DV1_DATA[0]	0x21	Signal Monitoring; Pattern Generation
TS_VALID	0x20	Signal Monitoring; Pattern Generation
TS_SOP	0x1F	Signal Monitoring; Pattern Generation

Table 8: GPIO IO_SEL Values *…Continued*

Signal	CLOCK_SEL/ IO_SEL (hex)	MODE
TS_CLK	0x1E	Signal Monitoring; Pattern Generation
TS_DATA[7]	0x1D	Signal Monitoring; Pattern Generation
TS_DATA[6]	0x1C	Signal Monitoring; Pattern Generation
TS_DATA[5]	0x1B	Signal Monitoring; Pattern Generation
TS_DATA[4]	0x1A	Signal Monitoring; Pattern Generation
TS_DATA[3]	0x19	Signal Monitoring; Pattern Generation
TS_DATA[2]	0x18	Signal Monitoring; Pattern Generation
TS_DATA[1]	0x17	Signal Monitoring; Pattern Generation
TS_DATA[0]	0x16	Signal Monitoring; Pattern Generation
SSI_SCLK_CTSN	0x15	Signal Monitoring; Pattern Generation
SSI_FS_RTSN	0x14	Signal Monitoring; Pattern Generation
SSI_RXD	0x13	Signal Monitoring; Pattern Generation
SSI_TXD	0x12	Signal Monitoring; Pattern Generation
UA2_CTSN	0x11	Signal Monitoring; Pattern Generation
UA2_RTSN	0x10	Signal Monitoring; Pattern Generation
UA2_RX	0x0F	Signal Monitoring; Pattern Generation
UA2_TX	0x0E	Signal Monitoring; Pattern Generation
UA1_RX	0x0D	Signal Monitoring; Pattern Generation
UA1_TX	0x0C	Signal Monitoring; Pattern Generation
GPIO[11]	0x0B	Signal Monitoring; Pattern Generation
GPIO[10]	0x0A	Signal Monitoring; Pattern Generation
GPIO[9]	0x09	Signal Monitoring; Pattern Generation
GPIO[8]	0x08	Signal Monitoring; Pattern Generation
GPIO[7]	0x07	Signal Monitoring; Pattern Generation
GPIO[6]	0x06	Signal Monitoring; Pattern Generation
GPIO[5]	0x05	Signal Monitoring; Pattern Generation
GPIO[4]	0x04	Signal Monitoring; Pattern Generation
GPIO[3]	0x03	Signal Monitoring; Pattern Generation
GPIO[2]	0x02	Signal Monitoring; Pattern Generation
GPIO[1]	0x01	Signal Monitoring; Pattern Generation
GPIO[0]	0x00	Signal Monitoring; Pattern Generation

Chapter 11: UART Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

11.1 Introduction

The UART ports are responsible for transferring data between the serial interface and memory. The PNX8526 contains three Universal Asynchronous Receiver/Transmitter (UART) ports, identified as UART1, UART2 and UART3. UART1 has two pins, UART2 and UART3 have four pins. (UART3 is multiplexed on the SSI pins.)

Each of these ports is driven by a UART module. Each UART module contains 16x8-bit receiver and transmitter FIFOs and has independent, bi-directional DMA.

The UART module contains the following features:

- Adjustable baud rate counter from 230 kB/s down to 50 bps
- 16-byte FIFOs for transmit and receive
- Transmit (TXD) and Receive (RXD) data pins
- Hardware flow controls: RTSN, CTSN (not provided on UART1)
- Full duplex (independent DMA for transmit and receive channels)
- One start bit, one or two stop bits
- Even, odd or no parity
- 7 or 8 bits per character
- Programmable timeout on receive

Table 1: UART I/O Pins

Signal	Type	Description
TXD	OUTPUT	This pin is the UART transmit signal from the UART module.
RXD	INPUT	This pin is the UART receive signal to the UART module.
RTSN	OUTPUT	This pin is the UART Request to Send signal. It is used to indicate the UART is ready to send the next character. This pin is left open for the 2-pin version.
CTSN	INPUT	This pin is the UART Clear to Send signal. It is used to indicate that the modem or data set is ready to exchange data. This pin is tied low (active) for the 2-pin version.

11.2 Functional Description

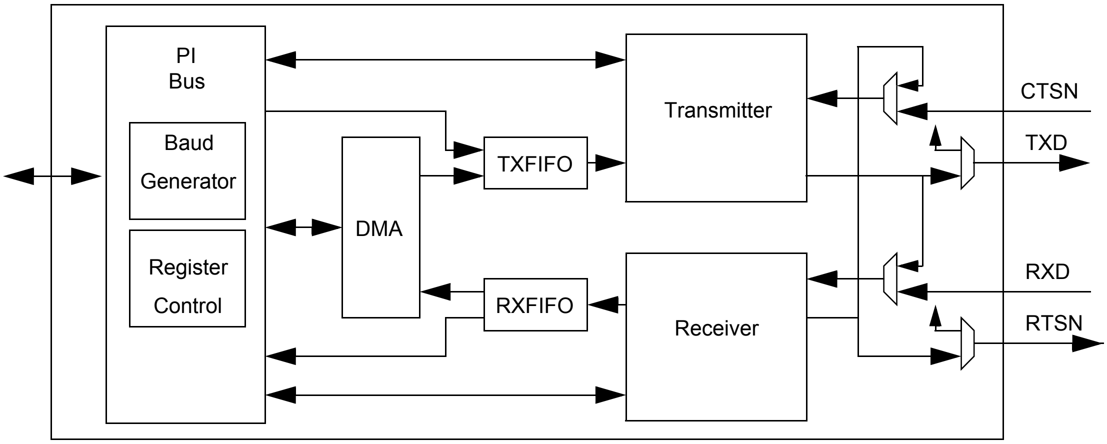


Figure 1: UART Block Diagram

11.2.1 Transmitter and TXFIFO

The transmitter holding interrupt goes high whenever the FIFO reaches the empty state and the TX holding register is emptied as well. The transmitter holding interrupt indicates that as many as 16 8-bit characters may be written to the FIFO. Every write of a character beyond these 16 is ignored without any warning or interrupt flagging.

The UART starts reading this data as soon as the FIFO is not empty. The transmitter takes the data from the FIFO, formats it according to the register settings, and emits the pulses on the output line.

Remark: In the case where only one character is written to the FIFO, and the FIFO is immediately emptied to the transmitter holding register, the transmitter holding interrupt is delayed by one character transmission time. This ensures that the empty status is reached before the interrupt routine has finished.

11.2.2 Receiver and RXFIFO

The receiver takes the incoming bits, decodes them, and shifts the data to the FIFO.

The receiver FIFO receives 7 or 8-bit characters (together with the error flags for parity, frame and break) until it is full. A Receiver Data Interrupt is issued when the programmed threshold in the Receiver FIFO is reached. It is reset by clearing the Receive Data Interrupt when it is below this threshold.

If the FIFO reaches the full state, an Overrun Error Flag is set and an interrupt is issued. New characters are not accepted. The Interrupt Service Routine must empty the FIFO in time to avoid overruns.

Remaining data in the receiver FIFO are indicated with a timeout interrupt. The timeout interrupt is set after a programmable number of character transmission times (the default is 4) when no data is written to or read from the FIFO and the FIFO is not empty.

11.2.3 Clock Settings and Baud Generator

The clock for the UART is set in the Clocks module. (Chapter 5 Clock Reset and Power Management for more information.) The default clock is 3.6923 MHz. The UART is capable of running at higher frequencies (up to 16 MHz) with baud rates up to 1 Mbit/s. At higher frequencies, it must be ensured that the PI-Bus clock is running at greater than twice the UART frequency.

The baud generator provides the baud clock (determined by the baud rate register) to the transmitter and receiver.

11.3 Operation

The UART module has two operation modes: Interrupt and DMA.

- The Interrupt mode uses Interrupt Service Routines of the CPU to do the data transfers. The transfers are buffered with two FIFOs. This allows higher latency times for serving this interrupt. The transfer is 8 bits per data.
- The DMA mode autonomously handles bi-directional transfers. A transfer is 8 bits per data.

Transmit

In both modes, for every data written to the internal transmitter holding register or transmitter FIFO, the UART starts a serial transmission of this character via the serial output pin (TXD).

Receive

The UART receives serial data via the serial input pin (RXD) and stores this as a complete byte in the receiver register or the receiver FIFO.

- In the interrupt mode the UART issues an interrupt to the CPU when data are available.
- In DMA mode, it starts the transfer to memory.

11.3.1 Interrupts

Interrupts are indicated through the interrupt line to the Interrupt Controller. Whenever an interrupt is set, the interrupt line goes active. Reading the Interrupt Status Register does not clear the interrupts. Interrupts are cleared by writing a “1" to the corresponding bit in the Interrupt Clear Register.

At the end, the Interrupt Service Routine must clear the served interrupt. This is to ensure that the read pointer of the Receiver FIFO is updated after a read.

11.4 Register Descriptions

The PNX8526 contains three Universal Asynchronous Receiver/Transmitter (UART) ports. Refer to Section 11.1 on page 11-266 for the feature distinctions between these ports. The modules and their registers are identical.

The base addresses for the UART1, UART2 and UART3 are at 0x04 A000, 0x04 B000 and 0x04 C000, respectively.

11.4.1 Register Address Map

Table 2: UART Register Summary

Offset	Name	Description
	UART1	
0x04 A000	DCR	Data Control Register
0x04 A004	MCSR	Modem Control and Status Register
0x04 A008	BAUDRATE	Baud Rate
0x04 A00C	CFG	DMA and Configuration Register
0x04 A010	TXSTART	TXDMA Buffer Start Address
0x04 A014	TXLENGTH	TXDMA Length Register
0x04 A018	TXCOUNTER	TXDMA Count Register
0x04 A01C	RXSTART	RXDMA Buffer Start Address
0x04 A020	RXLENGTH	RXDMA Length Register
0x04 A024	RXCOUNTER	RXDMA Count Register
0x04 A028	FIFOS/RBR/THR	FIFO Status/RX/TX Buffer Register
0x04 A02C—04 AFDC	Reserved	
0x04 AFE0	INT_ST	Interrupt Status Register
0x04 AFE4	INT_EN	Interrupt Enable Register
0x04 AFE8	INT_CLR	Interrupt Clear Register
0x04 AFEC	INT_SET	Interrupt Set Register
0x04 AFF0	Reserved	
0x04 AFF4	PD	Powerdown Register
0x04 AFF8	Reserved	
0x04 AFFC	Module ID	Module Identification Register
	UART2	
0x04 B000	DCR	Data Control Register
0x04 B004	MCSR	Modem Control and Status Register
0x04 B008	BAUDRATE	Baud Rate
0x04 B00C	CFG	DMA and Configuration Register
0x04 B010	TXSTART	TXDMA Buffer Start Address
0x04 B014	TXLENGTH	TXDMA Length Register

Table 2: UART Register Summary

Offset	Name	Description
0x04 B018	TXCOUNTER	TXDMA Count Register
0x04 B01C	RXSTART	RXDMA Buffer Start Address
0x04 B020	RXLENGTH	RXDMA Length Register
0x04 B024	RXCOUNTER	RXDMA Count Register
0x04 B028	FIFOS/RBR/THR	FIFO Status/RX/TX Buffer Register
0x04 B02C—04 BFDC	Reserved	
0x04 BFE0	INT_ST	Interrupt Status Register
0x04 BFE4	INT_EN	Interrupt Enable Register
0x04 BFE8	INT_CLR	Interrupt Clear Register
0x04 BFEC	INT_SET	Interrupt Set Register
0x04 BFF0	Reserved	
0x04 BFF4	PD	Powerdown Register
0x04 BFF8	Reserved	
0x04 BFFC	Module ID	Module Identification Register
	UART3	
0x04 C000	DCR	Data Control Register
0x04 C004	MCSR	Modem Control and Status Register
0x04 C008	BAUDRATE	Baud Rate
0x04 C00C	CFG	DMA and Configuration Register
0x04 C010	TXSTART	TXDMA Buffer Start Address
0x04 C014	TXLENGTH	TXDMA Length Register
0x04 C018	TXCOUNTER	TXDMA Count Register
0x04 C01C	RXSTART	RXDMA Buffer Start Address
0x04 C020	RXLENGTH	RXDMA Length Register
0x04 C024	RXCOUNTER	RXDMA Count Register
0x04 C028	FIFOS/RBR/THR	FIFO Status/RX/TX Buffer Register
0x04 C02C—04 CFDC	Reserved	
0x04 CFE0	INT_ST	Interrupt Status Register
0x04 CFE4	INT_EN	Interrupt Enable Register
0x04 CFE8	INT_CLR	Interrupt Clear Register
0x04 CFEC	INT_SET	Interrupt Set Register
0x04 CFF0	Reserved	
0x04 CFF4	PD	Powerdown Register
0x04 CFF8	Reserved	
0x04 CFFC	Module ID	Module Identification Register

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 A000			*Data Control Register*	
31		-	Unused	
30	R/W	0	TXBREAK	Transmission Break initiation. This is the Break Control bit. It causes a break condition to be transmitted to the receiving UART. The break is disabled by setting this bit to a logic 0. The Break Control bit acts only on TXD and has no effect on the transmitter logic. Note: This feature enables the CPU to alert a terminal in a computer communications system. 1. Set break when transmitter is empty. 2. Load all 0s into transmitter. 3. Wait for the transmitter to be idle, and clear break when normal transmission has to be restored.
29		-	Unused	
28	R/W	0	EVENPARITY	Even Parity Select. Setting this bit selects even parity instead of odd parity, if the ENPARITY bit is set.
27	R/W	0	ENPARITY	Parity Enable Bit. Setting this bit will cause parity to be generated and received.
26	R/W	0	TWOSTOP	Number of stop bits. Setting this bit will cause the transmitter to transmit two stop bits instead of one stop bit.
25		-	Unused	
24	R/W	0	BIT_8	Word select bit 0 = 7 bits per character mode 1 = 8 bits per character mode
23:22	R/W	0	FIFO_THRES	Set the trigger level for RCVR FIFO Interrupt (RX_INT) 23:22Trigger Level (in decimal value) 00 01 01 04 10 08 11 14
21:19		-	Unused	
18	W	0	TX_FIFO_RST	Writing a “1” to this bit clears all the transmitter FIFO counters (pointer and status) to zero. This bit is self-clearing.
17	W	0	RX_FIFO_RST	Writing a “1” to this bit clears all the receiver FIFO counters (pointer and status) to zero. This bit is self-clearing.
16	W	0	RX_FIFO_PT_ADV	Writing a “1” to this bit advances the RX FIFO read pointer. This bit is self-clearing.
15:0		-	Unused	
Offset 0x04 A004			*Modem Control and Status Register*	
31:24	R/W	0	SCR	Scratch register - for software use
23	R	0	DCD	Data Carrier Detect is the complement to the DCDN input pin.
22:21		-	Unused	

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
20	R	0	CTS	This bit is the complement to the CTSN input pin. Software has to monitor this bit to know when the data carrier (e.g. modem, etc.) has room to receive data. If the Loop bit (reg 0x004 bit [4]) is set, CTS is equivalent to RTS. IF CTSENABLE (reg 0x00C bit [5]) is set, the automatic hardware flow control is enabled and the transmitter stops automatically if CTS is inactive.
19:5		-	Unused	
4	R/W	0	LOOP	Loopback mechanism. Setting this bit will cause the transmitted data to internally loopback to the receive data and the request_to_send signal to the clear_to send signal. The external UA_TXD output pin is held high and the external UA_DTRN output pin is held low when this bit is set
3:2		-	Unused	
1	R/W	0	RTS	Request to Send. This bit controls the Request To Send (RTSN) output. When this bit is set to a logic 1, the RTSN output is forced to a logic 0 (UART has characters to transmit). When this bit is reset to a logic 0, the RTSN output is forced to a logic 1. (There are no characters to transmit.) Note: RTSENABLE (reg 0x00C bit [4]) set the hardware flow control of the RTSN output. In this case, writing to this bit will have no effect on the RTSN output.
0	R/W	0	DTR	Data Terminal Ready. This bit is the complement to the DTRN output pin. Software set and reset. No hardware support required.
Offset 0x04 A008			***Baud Rate Register***	
31:16		-	Unused	

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:0	R/W	0000	BAUDRATE	The standard baud rates are determined as follows: Baud Rate = UART_CLOCK (default = 3. 6923 MHz) / ((BAUDRATE(15:0) + 1) * 16). A change of this value during operation will become effective after one baud cycle (1/baud rate) with the current baud rate. [15:0]Baud ratePercent Error 02304000.002 11152000.002 3576000.002 5384000.002 11192000.002 2396000.002 4748000.002 9524000.002 19112000.002 3836000.002 7673000.002 15351500.002 1715134.50 20951100.001 3071750.002 4609 500.001
Offset 0x04 A00C			*Configuration Register*	
31:16	R/W	0004	RXTOCNTR	RX Timeout Control bits RX TimeOut = (RXTOCNTR[15:0]) * Character transmission time
15		-	Unused	
14	R/W	0	TXDMALOOPEN	The DMA channel TX controller supports two modes depending on the state of this bit. When DMALOOPEN is “0”, the DMA channel TX controller will stop executing when it reaches the end of the DMA buffer. When DMALOOPEN is “1”, the DMA controller will loopback to the start of the DMA buffer when the end of the DMA buffer is reached and will continue operating.
13	R/W	0	TXDMAPAUSE	Setting this bit will pause the current DMA transfer but it still allows the ongoing byte transfer(s) to be finished.The TX/RX register (0x028) and TX DMA Counter register (0x018) are accessible by software. When the bit is cleared the DMA operation will resume from where it stopped.
12	R/W	0	TXDMAEN	Enables the DMA channel TX transmit function. It should not be set until the START and LENGTH registers are set, module is enabled, and empty flag is set. When DMA operation is finished, software has to clear this bit and enable it for another DMA read operation. The DMA operation will abort if this bit is disabled prematurely. All the read bytes that were received (equal the TX DMA Length minus the content of the TX DMA Counter register) will be transmitted.

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
11		-	Unused	
10	R/W	0	RXDMALOOPEN	The DMA channel RX controller supports two modes depending on the state of this bit. When DMALOOPEN is “0”, the DMA channel RX controller will stop executing when it reaches the end of the DMA buffer. When DMALOOPEN is “1”’, the DMA controller will loopback to the start of the DMA buffer when the end of the DMA buffer is reached and will continue operating.
9	R/W	0	RXDMAPAUSE	Setting this bit will pause the current DMA transfer but it still allows the ongoing byte transfer to be finished. The RX/TX register (0x028) and RX DMA Counter register (0x024) are accessible by software. When the bit is cleared the DMA operation will resume from where it stopped.
8	R/W	0	RXDMAEN	Enables the DMA channel RX receive function. It should not be set until the START and LENGTH registers are set, module is enabled. and empty flag is set. When the DMA operation is finished, software has to clear this bit and enable it for another DMA write operation. The DMA operation will abort if this bit is disabled prematurely. All the written bytes that were sent (equal the RX DMA LENGTH minus the content of the RX DMA Counter register) will be flushed to the memory.
7:6		-	Unused	
5	R/W	0	CTSENABLE	Automatic CTSN processing (hardware flow control) is enabled when this bit is ‘1.’ When the computer (terminal) wishes to send data it activates the Request to Send line (set register 0x0004 bit [1], RTS, and disables RTSENABLE, register 0x000c bit [4]). If the carrier has room for data, the carrier will reply by activating the Clear to Send line and the computer will begin sending data. When this bit is set, the CTSN pin is evaluated automatically and the UART starts the transfer.
4	R/W	0	RTSENABLE	This bit enables the automatic generation of the RTSN signal. RTSN is active when the UART can receive characters (e.g., FIFO is not full). This option should be used if the UART is acting as a data carrier and is asked by the terminal if it is able to receive data.
3	R/W	0	RXBREAKHALT	Setting this bit will cause the receiver to halt (no more data is allowed to get into the receiver FIFO) after receiving a break, until the receiver FIFO is emptied and the UARTRXD signal goes to the marking state (’1’). Otherwise, the receiver will hold until the UARTRXD signal goes to the marking state without regard for the state of the Receiver FIFO.
2	R/W	0	DTINVERT	Setting this bit will cause the UARTTXD and UARTRXD signals to be inverted.
1	R/W	0	TXDDIS	Setting this bit will cause the UARTTXD signal to go low.
0		-	Unused	
Offset 0x04 A010			***TX DMA TX Start Address Register***	

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:0	R/W	0000	TXSTART	These bits define the start address for the DMA buffer for DMA transfers from the receiver FIFO. The value has to be the physical address.
Offset 0x04 A014			*TX DMA Length Register*	
31:12		-	Unused	
11:0	R/W	0000	TXLENGTH - 1	Length of DMA buffer in bytes (number of bytes to be transferred) minus 1. The last address in the DMA buffer is given by TXSTART + (TXLENGTH - 1).
Offset 0x04 A018			*TX DMA Counter Register*	
31:13		-	Unused	
12:0	R	0000	TXCOUNTER	TXDMA countdown counter value in bytes decrements from TXLENGTH to zero. This register will be reloaded with new “TXLENGTH” value when a new DMA operation gets started.
Offset 0x04 A01C			*RX DMA Start Address Register*	
31:0	R/W	0000	RXSTART	These bits define the start address for the transmitter DMA buffer. The value has to be the physical address.
Offset 0x04 A020			*RX DMA Length Register*	
31:16		-	Unused	
15:0	R/W	0000	RXLENGTH - 1	Length of DMA buffer in bytes (number of bytes to be transferred) minus 1. The last address in the DMA buffer is given by RXSTART + (RXLENGTH - 1).
Offset 0x04 A024			*RX DMA Counter Register*	
31:17		-	Unused	
16:0	R	0000	RXCOUNTER	RXDMA countdown counter value in bytes decrements from RXLENGTH to zero. This register will be reloaded with new “RXLENGTH” value when a new DMA operation gets started.
Offset 0x04 A028			*RBR/THR/FIFOS Receive/Transmit and FIFO Status Register*	
31:21		-	Unused	
20:16	R	0	TXFIFO_STA	TXFIFO Status: 5’h10 indicates 16 bytes remaining in the TX FIFO. 5’h0 indicates the TX FIFO is empty. These bits are reset by writing a “1” to bit [18] of register 0x000.
15:13		-	Unused	
12:8	R	0	RXFIFO_STA	RXFIFO Status: 5’h10 indicates 16 bytes remaining in the RX FIFO (5’h0 indicates the RX FIFO is empty). These bits are reset by writing a “1” to bit [17] of register 0x000.
7:0	R	00	RBR	Receiver Buffer register. Reading this register will not clear the RX_INT interrupt. Software should write a “1” to bit[16] of register 0x000 (RX_FIFO_PT_ADV bit) in order to advance the RX FIFO read pointer for the next read.
7:0	W		THR	Transmit Holding register
Offset 0x04 AFE0			*Interrupt Status Register*	
31:14		-	Unused	

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13	R	0	TXDMAF_INT	Issues an interrupt when the TX DMA controller reaches the end of specified buffer (TXCOUNTER=0).
12	R	0	TXDMAH_INT	Issues an interrupt when the TX DMA controller reaches the halfway point in specified buffer (TXCOUNTER = TXLENGTH/2 if TXLENGTH is even and TXCOUNTER = TXLENGTH/2 - 1 if TXLENGTH is odd).
11	R	0	RXDMAF_INT	Issues an interrupt when the RX DMA controller reaches the end of specified buffer (RXCOUNTER=0).
10	R	0	RXDMAH_INT	Issues an interrupt when RX DMA controller reaches the halfway point in the specified buffer (RXCOUNTER = RXLENGTH/2 if RXLENGTH is even and RXCOUNTER = RXLENGTH/2 - 1 if RXLENGTH is odd).
9	R	0	DDCD_INT	This bit is high if the value of the DCDN input pin has changed.
8	R	0	DCTS_INT	This bit is high if the value of the CTSN input pin has changed.
7	R	0	TX_INT	Issues an interrupt when transmitter holding FIFO becomes empty.
6	R	0	EMPTY_INT	Issues an interrupt when the transmitter shift register becomes empty and the transmitter holding FIFO is also empty.
5	R	0	RCVTO_INT	Issues an interrupt when the receiver times out (timeout is equal to (RXTOCNTR[15:0]) *character transmission times) and no data is written to or read from the RX FIFO, and the RX FIFO is not empty.
4	R	0	RX_INT	Issues an interrupt when the threshold level is reached (the threshold level is set by writing to bits [23:22] of register 0x000).
3	R	0	RXOVRN_INT	Receiver Overflow issues an interrupt when the receiver holding FIFO is full and a new receive data is coming.
2	R	0	FRERR_INT	Frame Error Issues an interrupt when receiver does not detect stop bit following the character and received character is not 0x00.
1	R	0	BREAK_INT	Issues an interrupt when receiver detects a BREAK condition i.e., received character is 0x00 and no stop bit.
0	R	0	PARITY_INT	Issues an interrupt when receiver detects parity bit error.
Offset 0x04 AFE4			*Interrupt Enable Register*	
31:14		-	Unused	
13:0	R/W	0000	INTR_EN	A logic “1” written to a specific bit location will enable the corresponding interrupt in the Interrupt Status register.
Offset 0x04 AFE8			*Interrupt Clear Register*	
31:14		-	Unused	
13:0	W	0000	INTR_CLR	A logic “1” written to a specific bit location will clear the corresponding interrupt in the Interrupt Status register.This bit is self-clearing.
Offset 0x04 AFEC			*Interrupt Set Register*	
31:14		-	Unused	

UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13:0	W	0000	INTR_SET	A logic "1" written to a specific bit location will set the corresponding interrupt in the Interrupt Status register.
Offset 0x04 AFF4			*Powerdown Register*	
31	R/W	0	PD	UART Powerdown indicator 1 = Powerdown 0 = Power up When this bit equals 1, no other registers are accessible.
30:0		-	Unused	
Offset 0x04 AFFC			*Module ID Register*	
31:16	R	0x0107	MOD_ID	UART Module ID Number
15:12	R	0	REV_MAJOR	Major revision
11:8	R	0	REV_MINOR	Minor revision
7:0	R	0	APP_SIZE	Aperture size is 0 = 4 kB.

UART 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
UART 2 registers begin at offset 0x04 B000. In all other respects, they are identical to the UART 1 registers.				

UART 3 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
UART 3 registers begin at offset 0x04 C000. In all other respects, they are identical to the UART 1 registers.				

Chapter 12: SSI Port

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

12.1 Introduction

The Synchronous Serial Interface (SSI) connects the PNX8526 to modem Analog Front End (AFE) devices from various manufacturers. Supported AFEs include the ST Microelectronics STLC7550 and Siemens ALIS chip sets.

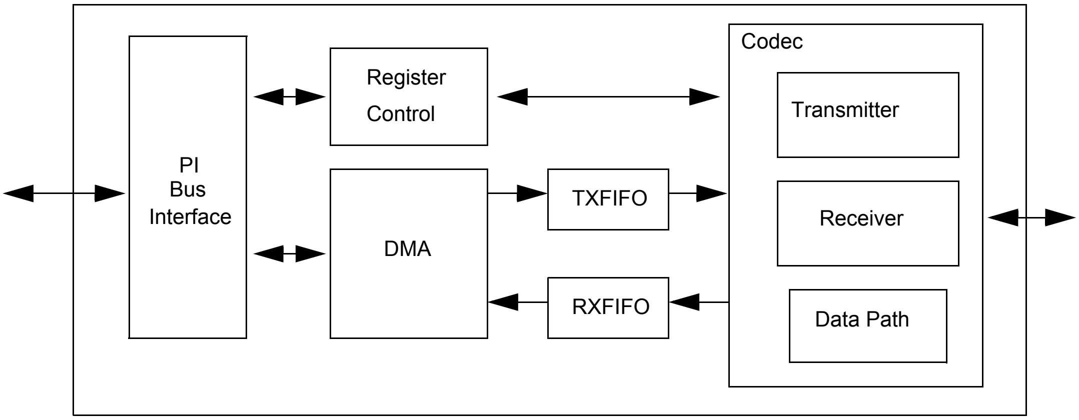


Figure 1: SSI Block Diagram

12.2 Functional Description

The SSI module has two main interfaces. The on-chip PI-Bus interface is used for communication with the CPU and system memory. Data samples are transferred to and from system memory using DMA, while control and status information are sent and retrieved using IO access.

The off-chip SSI interface is used for communication with the AFE. The SSI in the PNX8526 is always the slave when interfacing to an external device i.e., the shift clock SCLK and the frame synchronization signal FS are inputs only. A positive, 1-SCLK wide, pulse on FS (whose edges are synchronous to the rising edge of SCLK) signifies the beginning of a new frame.

12.2.1 PI-Bus Interface

The SSI module can be configured for either a big-endian or little-endian operation. As a slave, the SSI accepts only single-word access. As a master in DMA Write, data can be written to memory by single 32-bit Word Data, Unlocked (WDU) or chained transactions if there is more than one word of data in the receive FIFO.

DMA Read supports WD4, WD2 and WDU transactions. No bus cycle will be issued if the SSI encounters a bus error or timeout. It's then up to software to handle the error. Modem data samples are represented by 16-bit integers in two's complement format. For transmit data, sample N is to be transmitted first, followed by samples N+1, then N+2, etc. This means in the little-endian setting, the two halfwords are to be swapped before being serialized.

12.2.2 DMA Channels

The SSI module contains two physical DMA channels, one for fetching transmit data from memory and another for putting received data to memory. Each has its own FIFO for storing temporary data

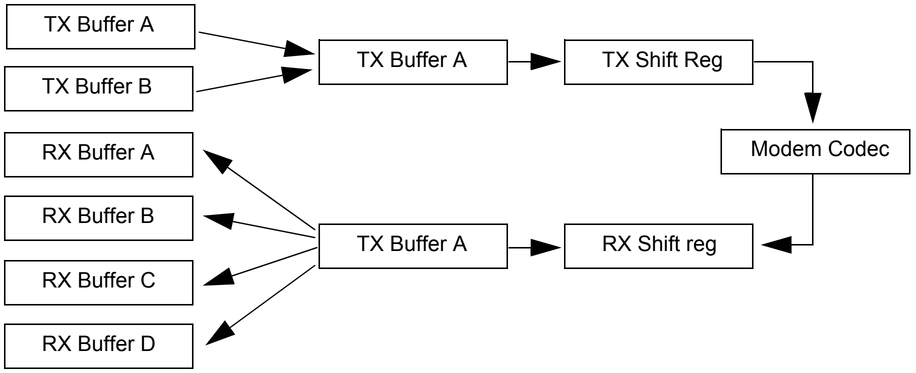


Figure 2: DMA Channels

If a DMA channel is defined as a logical connection between a source and destination, there are up to six channels. This means there are six pairs of DMA address and length registers. Software can specify up to two areas in memory from which data are to be fetched and up to four areas to which they are written. However, at any given time, only three DMA processes are possible, one for reading data from memory and two for writing data to memory.

12.2.2.1 DMA Modes

Basically there are two modes of DMA transfer: normal mode and auto-initialization. When two buffers are used together in a DMA process, a third mode, ping-pong, is also available.

- In normal mode, after the last data has been transferred, the DMA hardware stops. To initiate another DMA process, software has to program the DMA length register again. Reprogramming the address register is optional. As soon as the length register is written to, a new DMA process is begun (provided the length is not zero).
- In the auto-initialization mode, when the end of the buffer is reached or when the word count reaches 0, the hardware automatically reloads the initial address and transfer length and executes the same DMA process again. Thus once set up, the DMA engine continuously transfers data to and from memory until it's reset by software or an error condition is encountered.

- The ping-pong mode is simply auto-initialization when applied to a pair of buffers. After the software has set up addresses and lengths for both buffers (e.g., A and B) the hardware would start with the buffer whose registers are programmed first (e.g., Buffer A). When the end of Buffer A is reached, the hardware switches over to Buffer B, i.e., it loads Buffer B’s address and length into the DMA state machine and resumes the transfer. If auto-initialization is also enabled, the hardware goes back to Buffer A once Buffer B is done.

12.2.2.2 DMA Buffers

Separate DMA channels are used for each data pipe between the memory and the external device. Two buffers, Tx A and B, can be used for transmitting data to the codec. Four buffers, Rx A, B, C, and D, are available for receiving data. Used individually, each buffer is equivalent to any other of the same type (Tx or Rx). When used in pairs in double-buffering or ping-pong fashion, the only restriction is that the buffers must be paired up correctly. The allowable combinations are Tx A and B, Rx A and B, and Rx C and D.

There is one restriction on the use of DMA buffers, the buffer length has to be equal to or greater than 2.

12.2.3 FIFOs

There are two FIFOs in the SSI module:

- One 8-word deep FIFO holds data to be transmitted. Data are transferred from memory with DMA.
- One 8-word deep FIFO holds received data. Data are transferred to memory with DMA.

12.2.4 Codec Interface

The codec interface is composed of modem codec circuitry. It contains control and status registers. Two telephony codecs are explicitly supported by the SSI module. They are the Siemens PSB-4596 and the ST Microelectronics STLC-7550. The frame structure and timing of the latter are for terminal applications.

12.3 Operation

Sampled data and control/status data, if present, are serially transferred in the same frame, the exact format and sequence being dependent on the codec used. The msb is transmitted first. Sampled data are transferred in 16-bit packets while control/status data are carried in 8 or 16-bit packets (8-bit control/status packet for Siemens AFE, 16-bit control/status packets for ST Microelectronics AFE). Data samples are represented in two’s complement format. The transmitter clocks data out on SCLK’s positive edge; the receiver samples them on its negative edge.

Sampled data and control data are handled separately. Inside the SSI module, sampled data are stored in 8-deep, 32-bit wide FIFOs. Control data are stored in a 16-bit control register for ST Microelectronics STLC-7550 and in an 8-bit control register for the Siemens PSB-4595. Status is stored in a 16-bit status register for ST Microelectronics STLC-7550 and in an 8-bit status register for the Siemens PSB-4596.

Transmit and receive data have separate FIFOs. During transmission, the codec interface logic is responsible for multiplexing data from the various FIFOs/registers and sending them out in the right sequence. When receiving, the codec interface is responsible for extracting valid sampled data and status data storing them to the appropriate FIFO/registers.

Software will not issue a new command to the codec until:

- the previous write command has been completely transmitted or
- data returned from the codec (in response to a previous read command has been read by the CPU).

Both polling and interrupt can be used to check for completion of command transmission to the codec.

To provide flexibility, the packet length, number of packets, the order of sending control and sampled data, and the number of FS pulses in a frame are programmable.

12.3.1 Interrupts

One interrupt output is generated from the SSI module. It is connected to the PNX8526 Interrupt controller.

12.3.1.1 Interrupt Sources

The SSI generates an interrupt when:

- Data transmit buffer is empty.
- Data transmit buffer underruns.
- Data receive buffer is full.
- Data receive buffer overruns.
- Data receive buffer has reached a pre-programmed mark.
- A block of data of pre-programmed size has been transferred to/from memory.
- A command has been sent to the codec.
- Status data in response to a read request is available for read.
- Frame detection error: no frame detected or more frame sync pulses than expected
- Bus error: DMA transfers on PI-Bus terminated by ERR.

Transmit Buffer Empty and Receive Buffer Full occur when the end of the Transmit or Receive buffer is reached. These conditions coincide with their DMA Length values being decremented to 0.

Transmit Buffer Underrun occurs when the buffer's last word has been fetched and the next buffer has not been set up. For the telephony codec, this condition indicates an error. In both cases, the interrupt remains asserted unless and until it's cleared by software. If auto-initialize mode is enabled, the buffer underrun interrupts would never be generated.

Receive Buffer Overrun occurs when a receive buffer is filled up and there are more data coming in from the codec.

An interrupt is generated when a command has been sent to the codec. This means the content of the Codec Control register has to be transferred to the transmit shift register and is available for another write. The command may or may not be completely sent out to the codec.

12.3.1.2 Interrupt Usage

Control Data Transfer

It is recommended that polling mode be used. After a command is sent, the driver can check the Codec Status Register Command Transmission Completion Bit (Bit [17]) before another command is sent. Similarly, Status Data Valid Bit (Bit [16]) can be polled when waiting for codec status data. With SCLK frequency to be at least 256 kHz, the application has to determine whether it can handle the interrupt rate, which is expected to be high during call setup.

Sampled Data Transfer

In normal operation, the CPU is to be interrupted only when a block of data of a certain size has been transmitted or received. The data pump operates on 48 sample blocks. The DMA transfer length could be set to this size and the end-of-buffer interrupt be enabled. Alternatively, the DMA channel can be programmed for auto-initialization mode, the transfer length set to an integer multiple of 48 samples, and the Operating Block Size register set to 48. Once the DMA is started, the host is interrupted only when a new block of data is available for processing (or when a processed data block has been transmitted.) As long as the processing time is less than the transmission time on the serial bus, the only overhead is one interrupt service per block.

12.4 Major Interfaces

The SSI module connects the PNX8526 to either ST Microelectronics STLC-7550 or Siemens PSB-4596 Analog Front End chip

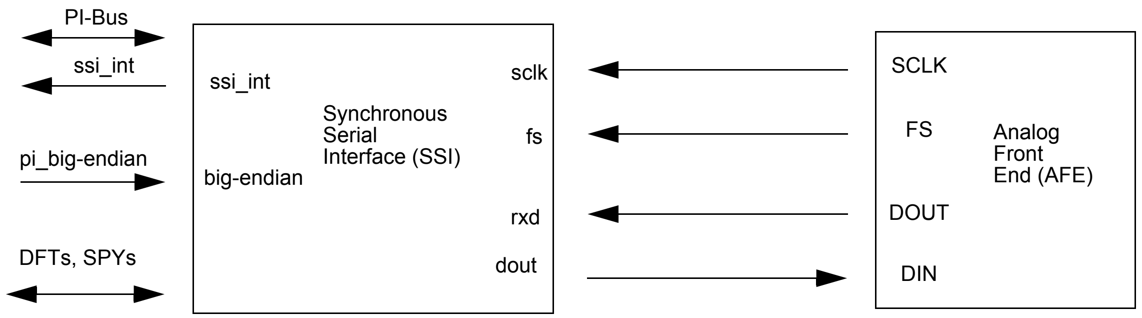


Figure 3: SSI Interface

12.4.1 Siemens ALIS_D PSB-4596

Clock: Provided by a crystal or oscillator. Any frequency from 16.384 MHz to 50 MHz. We will be using 24.576 MHz.

Frame rate varies between 7.2 kHz and 32 kHz. SCLK (256 kHz to 2048 kHz) should be programmed to have frequency at least 32 times the sampling frequency Fs. Settings to configure the codec for master, multiplexed mode: Bus Master = ’1’, SCI_CLK/MODE =’1’ during RESET, SCI_CS/SWAP = ’0’. The PSB 4596 drives DAT_CLK and FSC out. In every frame, 16-bit modem data are followed by 16-bit control data.

12.4.2 ST Microelectronics STLC-7550

Clock: Provided by a crystal or oscillator. Use 36.864 MHz crystal and connect MCM to logic 1. Frequencies of 18.432 MHz and 9.216 Mhz can be used, but they cannot generate sampling frequencies 13.96 kHz, 11.82 kHz, and 10.97 kHz.

Fs = XTAL_IN / (M*Q*OVER)

SCLK = XTAL_IN / (M*Q)

SCLK ranges from 460.8 kHz (= 7200 * 64) to 3.072 MHz (= 16000 * 192).

Settings to configure the codec for master, multiplexed mode: M/S# = 1, HC[1:0] = 1x, TS = ’0’: Again, SCLK and FS are driven out from the codec, and 16-bit modem data are followed by 16-bit control data. The frame synchronization FS is pulsed twice in this case. The first one, the Primary Frame Synchronization Pulse (PFSP), is generated at the beginning of a frame and is followed by 16 bits of sampled data. The Secondary Frame Synchronization Pulse is generated at the midpoint of the sampling period and is followed by 16 bits of control data.

The codec’s PFSP is synchronized with SS as follows: The codec is assumed to be free running after power up. The SSI is also enabled regardless of whether it’s in sync with the codec.

By writing a control halfword to the Codec Control Register and comparing with that coming back from the codec, the driver can determine the correct data and control sequence. If necessary, it then may set the SWAP bit in the STLC Frame Width Register to instruct the SSI to reverse the (internal) time slots for data and control to match that of the codec. Finally, the SYNCDONE bit in the same register is set to complete the synchronization.

12.5 Register Descriptions

The base address for the SSI module in the PNX8526 is 0x10 8000.

12.5.1 Register Address Map

Table 1: SSI Module Register Summary

Offset	Name	Description
0x10 8000	SSI_CNTR1	SSI Control Register 1
0x10 8004	SSI_CNTR2	SSI Control Register 2
0x10 8008	SSI_STAT	SSI Status Register
0x10 800C	COD_CNTR	Codec Control Register
0x10 8010	COD_STAT	Codec Status Register
0x10 8014	TXDMA_ADDR_A	TXDMA Base Address Register A
0x10 8018	TXDMA_LEN_A	TXDMA Length Register A
0x10 801C	TXDMA_ADDR_B	TXDMA Base Address Register B
0x10 8020	TXDMA_LEN_B	TXDMA Length Register B
0x10 8024	TXDMA_CNT_AB	TXDMA Current Word Count for A or B
0x10 8028	RXDMA_ADDR_A	RXDMA Base Address Register A
0x10 802C	RXDMA_LEN_A	RXDMA Length Register A
0x10 8030	RXDMA_ADDR_B	RXDMA Base Address Register B
0x10 8034	RXDMA_LEN_B	RXDMA Length Register B
0x10 8038	RXDMA_CNT_AB	RXDMA Current Word Count for A or B
0x10 803C	RXDMA_ADDR_C	RXDMA Base Address Register C
0x10 8040	RXDMA_LEN_C	RXDMA Length Register C
0x10 8044	RXDMA_ADDR_D	RXDMA Base Address Register D
0x10 8048	RXDMA_LEN_D	RXDMA Length Register D
0x10 804C	RXDMA_CNT_CD	RXDMA Current Word Count for C or D
0x10 8050	RXDMA_WM_AB	RXDMA Buffer Water Mark Register for A or B
0x10 8054	RXDMA_WM_CD	RXDMA Buffer Water Mark Register for C or D
0x10 8058	BLOCK_SIZE	Operating Block Size Register
0x10 805C—8070	Reserved	-
0x10 8074	STLC_FR_WIDTH	STLC Frame Width Register
0x10 8078—8FF0	Reserved	-

Table 1: SSI Module Register Summary *…Continued*

Offset	Name	Description
0x10 8FF4	PD	Powerdown Register
0x10 8FF8	Reserved	-
0x10 8FFC	Module ID	Module Identification Register

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 8000			*SSI Control Register 1*	
31:11		-	Unused	
10	R/W	0	STORE_MP_LF	Set to store memory pointer to the beginning of a packet when that packet's last frame is received. Otherwise, the memory pointer is stored when the first frame of a packet is received.
9	R/W	0	EN_FR_ERR_INT	Set to enable the Frame-Error Interrupt.
8	R/W	0	EN_PI_ERR_INT	Set to enable the interrupt due to PI-Bus ERR during DMA.
7	R/W	0	EN_RX	Set to enable the receiver state machine. When cleared, inputs (FS, DIN) are ignored after the current frame is over.
6	R/W	0	EN_TX	Set to enable the transmit state machine. When cleared, DOUT is tri-stated at the end of the frame.
5		-	Unused	
4	R/W	0	RX_CH_RST	Receive channel reset. Same as a hard reset except that register settings are not changed. Inputs are ignored.
3	R/W	0	TX_CH_RST	Transmit channel reset. Same as a hard reset except that register settings are not changed. Outputs are tri-stated.
2	R/W	0	EN_GLO_INT	SSI Global Interrupt Enable. Set to 1 to enable interrupt from the SSI module.
1:0	R/W	00	EXT_DEVICE	Specify the external device. 00 = PSB-4596 01 = STLC-7550 10 = Reserved 11 = Reserved
Offset 0x10 8004			*SSI Control Register 2*	
31	R/W	0	EN_RXDMA_OVN_INT_D	Set to enable Receive DMA Buffer 'D' Overrun Interrupt.
30	R/W	0	EN_RXDMA_OVN_INT_C	Set to enable Receive DMA Buffer 'C' Overrun Interrupt.
29	R/W	0	EN_RXDMA_OVN_INT_B	Set to enable Receive DMA Buffer 'B' Overrun Interrupt.
28	R/W	0	EN_RXDMA_OVN_INT_A	Set to enable Receive DMA Buffer 'A' Overrun Interrupt.
27:24		-	Unused	
23	R/W	0	EN_RXBLK_INT	Set to enable the generation of interrupt when a data block of the size as specified in the Operating Block Size register has been received and stored in memory.
22	R/W	0	EN_RXDMA_FULL_INT_D	Set to enable Receive DMA Buffer 'D' Full Interrupt.
21	R/W	0	EN_RXDMA_FULL_INT_C	Set to enable Receive DMA Buffer 'C' Full Interrupt.
20	R/W	0	EN_RXDMA_FULL_INT_B	Set to enable Receive DMA Buffer 'B' Full Interrupt.

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19	R/W	0	EN_RXDMA_FULL_INT_A	Set to enable Receive DMA Buffer 'A' Full Interrupt.
18	R/W	0	EN_RXDMA_PPONG	Receive DMA Mode. Logic '1' to enable ping-pong DMA.
17	R/W	0	EN_RXDMA_AUTOINIT	Receive DMA Mode: Logic '0' for normal mode Logic '1' for auto-initialize mode
16	R/W	0	EN_RXDMA	Receive DMA enable.
15	R/W	0	EN_TXBLK_INT	Set to enable the generation of interrupt when a data block of the size as specified in the Operating Block Size register has been fetched from memory.
14	R/W	0	EN_TXDMA_UNR_INT_B	Set to enable Transmit DMA Buffer 'B' Underrun Interrupt.
13	R/W	0	EN_TXDMA_UNR_INT_A	Set to enable Transmit DMA Buffer 'A' Underrun Interrupt.
12	R/W	0	EN_TXDMA_EMP_INT_B	Set to enable Transmit DMA Buffer 'B' Empty Interrupt.
11	R/W	0	EN_TXDMA_EMP_INT_A	Set to enable Transmit DMA Buffer 'A' Empty Interrupt.
10	R/W	0	EN_TXDMA_PPONG	Transmit DMA Mode. Logic '1' to enable ping-pong DMA.
9	R/W	0	EN_TXDMA_AUTOINIT	Transmit DMA Mode: Logic '0' for normal mode Logic '1' for auto-initialize mode
8	R/W	0	EN_TXDMA	Transmit DMA enable.
7:6		-	Unused	
5	R/W	0	EN_RXWTM_INT_D	Enables the interrupt generated when Receive Buffer D reaches the watermark specified in register 0x054.
4	R/W	0	EN_RXWTM_INT_C	Enables the interrupt generated when Receive Buffer C reaches the watermark specified in register 0x054.
3	R/W	0	EN_RXWTM_INT_B	Enables the interrupt generated when Receive Buffer B reaches the watermark specified in register 0x050.
2	R/W	0	EN_RXWTM_INT_A	Enables the interrupt generated when Receive Buffer A reaches the watermark specified in register 0x050.
1	R/W	0	EN_CMD_SENT_COD_INT	Set to enable the interrupt generated when a command has been sent to the modem codec.
0	R/W	0	EN_RDATA_AVAIL_COD_INT	Interrupt Set to enable the interrupt generated when read data from the external modem codec is available.
Offset 0x10 8008			*SSI Status Register* [1-1]	
31	R/W	0	PI_ERR	Bus error : DMA transfers on PI-Bus terminated by ERR.
30	R/W	0	FRAME_ERR	No Frame Dectected Error / Unexpected Frame Error.
29	R/W	0	RXDMA_OVRUN_ERR_D	Receive DMA Buffer 'D' overrun.
28	R/W	0	RXDMA_OVRUN_ERR_C	Receive DMA Buffer 'C' overrun.
27	R/W	0	RXDMA_OVRUN_ERR_B	Receive DMA Buffer 'B' overrun.
26	R/W	0	RXDMA_OVRUN_ERR_A	Receive DMA Buffer 'A' overrun.
25	R/W	0	TXDMA_OVRUN_ERR_B	Transmit DMA Buffer 'B' underrun.
24	R/W	0	TXDMA_OVRUN_ERR_A	Transmit DMA Buffer 'A' underrun.

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
23	R/W	0	CMD_SENT	Set when a command has been sent to the external modem codec.
22	R/W	0	CODEC_RDATA_AVAIL	Set when read data from external modem codec is available for read.
21:16	R/W	0	Unused	
15	R/W	0	RXBLK_DONE	Set when a block of data as specified in Operating Block Size Reg has been put in memory.
14:12		0	Unused	
11	R/W	0	RXDMA_FULL_D	Receive DMA Buffer 'D' Full. Set when the end of Rx D is reached.
10	R/W	0	RXDMA_FULL_C	Receive DMA Buffer 'C' Full. Set when the end of Rx C is reached.
9	R/W	0	RXDMA_FULL_B	Receive DMA Buffer 'B' Full. Set when the end of Rx B is reached.
8	R/W	0	RXDMA_FULL_A	Receive DMA Buffer 'A' Full. Set when the end of Rx A is reached.
7	R/W	0	TXBLK_DONE	Set when a block of data as specified in Operating Block Size Reg has been transmitted.
6		0	Unused	
5	R/W	0	RX_WMARK_D	Set when the Receive Buffer D reaches its watermark.
4	R/W	0	RX_WMARK_C	Set when the Receive Buffer C reaches its watermark.
3	R/W	0	RX_WMARK_B	Set when the Receive Buffer B reaches its watermrak.
2	R/W	0	RX_WMARK_A	Set when the Receive Buffer A reaches its watermark.
1	R/W	0	TXDMA_EMPTY_B	Transmit DMA Buffer 'B' Empty. Set after the last data in Tx B is fetched.
0	R/W	0	TXDMA_EMPTY_A	Transmit DMA Buffer 'A' Empty. Set after the last data in Tx A is fetched.
[1-1]	Note: Bits in this register (0x008) are set to logic "1" when their associated event occurs. Writing "1" to a bit clears it. Writing "0" to a bit has no affect.			
Offset 0x10 800C			***Codec Control Register***	
31	R/W	0	EN_RXCLK_DIV2	When set, the internal receive clock is divided by 2. The clock edge that samples the frame synchronization pulse asserted will resync the receive clock divider to be the data capture edge. Data samples will be captured every other clock thereafter until the end of the valid slots in the frame.
30:27	R/W	0	FRAM_RATE	Control the divide ratio for the programmable frame rate divider used to generate the frame sync pulses. Valid values are 1 to 16 with 16 corresponding to "0000".
26:23	R/W	0	VALID_LOT_SIZE	Control the valid slot size starting from Slot 1. A value of 0 corresponds to 16 slots.
22	R/W	0	EN_SYNC_PULSE	Set if each packet requires a sync pulse (applied only to modem codec).

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
21	R/W	0	FRAM_SYNC_POLARITY	Frame Sync Polarity controls which edge of the frame synchronization signal is active. Logic ’0’ indicates active rising edge. Logic ’1’ means falling edge.
20	R/W	0	TX_CLK_POLARITY	Transmit Clock Polarity. Data are transmitted on the rising edge of SCLK if TCP = ’0’; on the falling edge of TCP = ’1’.
19	R/W	0	RX_CLK_POLARITY	Receive Clock Polarity determines the sample clock’s edge. Data are sampled on the falling edge of SCLK if RCP = ’0’; on the rising edge if RCP = ’1’.
18		-	Unused	
17	R/W	0	RX_SHIFT_LSB	Receive Shift Direction bit controls the shift direction of the receive shift register. Data is shifted in lsb first when RSD = ’1’ and msb first when RSD = ’0’.
16		-	Unused	
15:0	R/W	0	STLC7550 control bits[15:0] or PSB4596 control bits[7:0]	Refer to the data sheet of these ICs . In the case of the PSB4596, bits [15:8] are unused.
Offset 0x10 8010			*Codec Status Register*	
31:19		-	Unused	
18	R	0	SERIALIZER_EMPTY	Set when the last data bit has been shifted out of the serializer. It is cleared on writes to the Codec Status register.
17	R	0	CCR_SENT_TO_TX	Similar to bit 23 of the SSI Status register but not dependent on the interrupt enable bit. It’s intended for polling i.e., it’s set as long as the condition remains true. Set when the content of the Codec Control register has been transferred to the transmitter (serializer) and the former is ready to accept new write data. It is reset on writes to the Codec Control or Codec Status registers.
16	R	0	CODEC_RDATA_AVAIL	Similar to bit 22 of the SSI Status register but not dependent on the interrupt enable bit. It’s intended for polling i.e., it’s set as long as the condition remains true. Set when read data from external modem codec is available for read. It is cleared on writes to the Codec Status register.
15:0	R	0	CARRY_CNTRL_DATA	Bits [15:0] carry control data. On reads, they reflect the values in the STLC-7550’s Control register or the PSB 4596’s last valid status byte (bits [7:0] only). Writes to this register are ignored.
Offset 0x10 8014			*Transmit DMA Address Register A*	
31:2	R/W	0	TXDMA_ADDR_A	Physical Base Address of the data to be transmitted. A DMA page (base address + transfer length) must not cross the 16-MB page boundary. When only one DMA buffer is needed, either register set, A or B, could be used.
1:0		-	Unused	
Offset 0x10 8018			*Transmit DMA Transfer Length Register A*	
31:14		-	Unused	
13:0	R/W	0	TXDMA_LENGTH_A	Transfer Length in 32-bit words

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 801C			*Transmit DMA Address Register B*	
31:2	R/W	0	TXDMA_ADDR_B	Physical Base Address of the data to be transmitted. A DMA page (base address + transfer length) must not cross the 16-MB page boundary. When only one DMA buffer is needed, either register set, A or B, could be used.
1:0		-	Unused	
Offset 0x10 8020			*Transmit DMA Transfer Length Register B*	
31:14		-	Unused	
13:0	R/W	0	TXDMA_LENGTH_B	Transfer Length in 32-bit words
Offset 0x10 8024			*Transmit DMA Current Word Count*	
31:16		-	Unused	
15	R	0	TXDMA_ACTIVE_B	Set if the Transmit DMA Channel B is active
14	R	0	TXDMA_ACTIVE_A	Set if the Transmit DMA Channel A is active
13:0	R	0	NUM_WORDS_FETCHED	Represents the number of data words that have been fetched from memory.
Offset 0x10 8028			*Receive DMA Address Register A*	
31:2	R/W	0	RXDMA_ADDR_A	Physical Base Address of the memory region where receive data are to be stored. A DMA page (base address + transfer length) must not cross the 16-MB page boundary. When only one DMA buffer is needed, either register set A, B, C, or D can be used. When used in pairs as ping-pong buffers, A goes with B, C with D.
1:0		-	Unused	
Offset 0x10 802C			*Receive DMA Transfer Length Register A*	
31:14		-	Unused	
13:0	R/W	0	RXDMA_LENGTH_A	Transfer Length in Words
Offset 0x10 8030			*Receive DMA Address Register B*	
31:2	R/W	0	RXDMA_ADDR_B	Physical Base Address of the memory region where receive data are to be stored. A DMA page (base address + transfer length) must not cross the 16-MB page boundary. When only one DMA buffer is needed, either register set A, B, C, or D can be used. When used in pairs as ping-pong buffers, A goes with B, C with D.
1:0		-	Unused	
Offset 0x10 8034			*Receive DMA Transfer Length Register B*	
31:14		-	Unused	
13:0	R/W	0	RXDMA_LENGTH_B	Transfer Length in Words
Offset 0x10 8038			*Receive DMA Current Word Count for Channel A or B,*	
31:16		-	Unused	
15	R	0	RXDMA_ACTIVE_B	Set if the Receive DMA Channel B is active
14	R	0	RXDMA_ACTIVE_A	Set if the Receive DMA Channel A is active

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13:0	R	0	NUM_WORDS_WRITTEN	Represents the number of data words that have moved to memory.
Offset 0x10 803C			*Receive DMA Address Register C*	
31:2	R/W	0	RXDMA_ADDR_C	Physical Base Address of the memory region where receive data are to be stored. A DMA page (base address + transfer length) must not cross the 16-MB page boundary. When only one DMA buffer is needed, either register set A, B, C, or D can be used. When used in pairs as ping-pong buffers, A goes with B, C with D.
1:0		-	Unused	
Offset 0x10 8040			*Receive DMA Transfer Length Register C*	
31:14		-	Unused	
13:0	R/W	0	RXDMA_LENGTH_C	Transfer Length in Words
Offset 0x10 8044			*Receive DMA Address Register D*	
31:2	R/W	0	RXDMA_ADDR_D	Physical Base Address of the memory region where receive data are to be stored. A DMA page (base address + transfer length) must not cross the 16-MB page boundary. When only one DMA buffer is needed, either register set A, B, C, or D can be used. When used in pairs as ping-pong buffers, A goes with B, C with D.
1:0		-	Unused	
Offset 0x10 8048			*Receive DMA Transfer Length Register D*	
31:14		-	Unused	
13:0	R/W	0	RXDMA_LENGTH_D	Transfer Length in Words
Offset 0x10 804C			*Receive DMA Current Word Count for Channel C or D*	
31:16		-	Unused	
15	R	0	RXDMA_ACTIVE_D	Set if the Receive DMA Channel D is active.
14	R	0	RXDMA_ACTIVE_C	Set if the Receive DMA Channel C is active.
13:0	R	0	NUM_WORDS_WRITTEN	Represents the number of data words that have moved to memory.
Offset 0x10 8050			*Receive Buffer Watermark Register for Buffer A and/or B*	
31:16		-	Unused	
15:14	R/W	0	WATER_MARK_AB_SEL	00 = Watermark not applied. 01 = Watermark applied to Buffer A only. 10 = Watermark applied to Buffer B only. 11 = Watermark applied to both Buffers A and B.
13:0	R/W	0	WATER_MARK_AB_VALUE	Watermark Values. When the number of data words in buffers A and/or B are equal to the number specified by Bits [13:0], an interrupt is generated. The Buffer Indicator, bits [15:14], determines to which buffer(s) this watermark is applied.
Offset 0x10 8054			*Receive Buffer Watermark Register for Buffer C and/or D,*	
31:16		-	Unused	

SSI PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:14	R/W	0	WATER_MARK_CD_SEL	00 = Watermark not applied. 01 = Watermark applied to Buffer C only. 10 = Watermark applied to Buffer D only. 11 = Watermark applied to both Buffers A and B.
13:0	R/W	0	WATER_MARK_CD_VALU E	Watermark Values. When the number of data words in buffers A and/or B are equal to the number specified by Bits [13:0], an interrupt is generated. The Buffer Indicator, bits [15:14], determines to which buffer(s) this watermark is applied.
Offset 0x10 8058			***Operating Block Size Register***	
31:8		-	Unused	
7:0	R/W	0	OPER_BLK_SIZE	Operating Block Size specifies the number of data samples (in words) to be transferred to/from memory before an interrupt is generated. It’s expected that the software DSP data pump processes transmit and receive data in blocks of the same sizes.
Offset 0x10 805C-8070			***Reserved***	
Offset 0x10 8074			***STLC Frame Width Register***	
31	R/W	0	SYNCDONE	SYNCDONE bit is set after software has figured out the correct sequence of sampled data and control data sent from the codec.
30	R/W	0	SWAP	SWAP bit is set to instruct the SSI hardware to reverse the current sequence of sampled data and control data being sent and received. This bit must be programmed to a proper value before or at the same time the SYNCDONE bit is set.
29:10		-	Unused	
9:0	R/W	0	TIME_INTERVAL_FS	Specify the time between frame synchronization pulses from the STLC-7550 in units of SCLKs. Used to detect the missing-frame error.
Offset 0x10 8FF4			***Powerdown Register***	
31	R/W	0	POWER_DOWN	PD, SSI Powerdown indicator 1 = Powerdown (in this mode, no other registers are accessible). 0 = Power up.
30:0		-	Unused	
Offset 0x10 8FFC			***Module ID Register***	
31:16	R	0x010C	MODULE_ID	SSI Module ID number
15:12	R	0	REV_MAJOR	Major revision
11:8	R	0	REV_MINOR	Minor revision
7:0	R	0	APP_SIZE	Aperture size is 0 = 4 kB.

Chapter 13: USB Port

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

13.1 Introduction

The USB module supports communication with peripherals following Open Host Controller Interface (OHCI) Standards, revision 1.0 and USB Standards, revision 1.1.

The USB module has two main interfaces:

- The USB Interface, USB_IF, consists of two down-ports capable of full speed or low speed communication. Each port is designed with a standard 2-pin interface for connecting USB devices (printers, keyboards, mouse, etc.)
- The Host Controller Interface (HCI_IF) communicates with the internal system of the PNX8526. To allow for this communication, the USB_PI is connected to the HCI_IF of the USB core. The USB_PI translates all HCI transactions to equivalent PI transactions so that the USB core is integrated with PNX8526.

Remark: The USB specification is available through www.usb.org.

13.2 Functional Description

Figure 1 shows the various components and how the USB module integrates with the PNX8526. The USB module connects to the PI-Bus. The USB_PI block of the USB module interfaces to the PI-Bus. The USB_PI also interfaces to the uhostc block, the core of the USB module. The USB_PI and the uhostc blocks communicate using the OHCI standard protocol

.

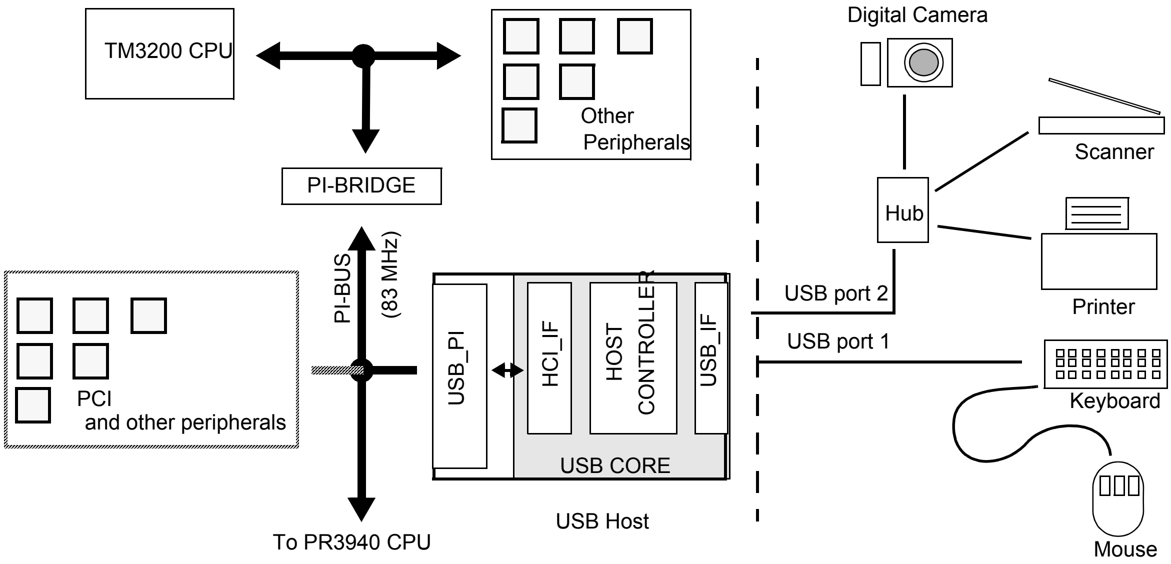


Figure 1: USB Module and Other Components

The USB devices shown in Figure 1 are merely suggestive. It is important to note that logically, communication is always USB host—device, regardless of cabling.

13.2.1 Operation

The USB Host has a master and slave interface to the PI-Bus. The USB_PI also incorporates the USB Host interrupts to be compliant with the PNX8526 Interrupt Architecture (see Figure 2).

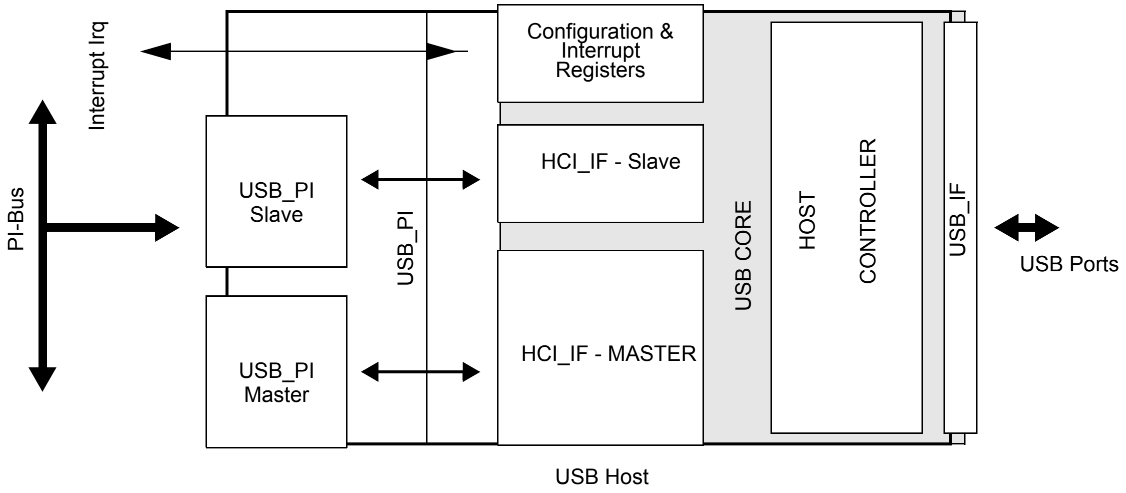


Figure 2: USB Host: USB Core and USB_PI Blocks

Before a USB device can be used functionally, the system must initiate the device via a process called device enumeration. This involves the system extracting all available information on the device in order to set up a suitable data structure in memory. Only when the data structure is in place can data transactions take place between the device and the system.

13.2.1.1 USB Data Transactions

The USB Host initiates transactions to move data between the Host System memory and the USB devices on the USB Bus. Four types of data transfer are possible, which require appropriate support in the software stack in the platform/middleware. The four types are:

- Control: Typically used to set up a device, when connected, but may also be used to change device settings.
- Bulk Data: Used for large blocks of data.
- Interrupt Data: Used for user input or when a device needs attention.
- Isochronous Data: Used for a continuous data pipe at a relatively constant data rate.

The transactions are controlled by the Host Controller Driver (HCD) which defines data structures to describe every transaction. These data structures are called TRANSFER DESCRIPTORS.

Typically the descriptor contains all information needed to generate a transaction:

- USB Device address: to identify which device is being processed
- Type of transfer: Isochronous (e.g. audio to speaker); Bulk (e.g. printers); Interrupt (control)
- Direction of transfer, in or out: (write to or read from system memory)
- Address of device memory buffer in the system.

13.2.2 Parameters

The PNX8526 USB module uses the parameters listed in Table 1. These are set up in the HcRHDescriptorA and HcRHDescriptorB registers.

Table 1: USB Module Parameters

Parameter	Description
NDP = 4’b0010	Number of downstream ports PNX8526 has 2 ports.
POTPGT = 8’h02	Power on to Power good time in ms 2 ms for PNX8526
NPS = 1’b0	No power switching 0 = Power switching done.
NOCP = 1’b0	No overcurrent protection 0 = Overcurrent protection exists.

Table 1: USB Module Parameters *…Continued*

Parameter	Description
PSM = 1’b0	Power switching Mode 0 = Global power switching.
OCPM = 1’b0	Overcurrent protection Mode 0 = Global.
PPCM = 2’b00	Per port Control mask Don’t care.
DR = 2’b00	Device Removable 0 = Yes.

13.2.3 Powerdown Feature

The OHCI configuration block provides a register that allows software to put the system into suspend mode. This occurs when software sets HCFS bits of HcControl (offset 0x04 8004) to USB SUSPEND state. On seeing the state switched to suspend, Host Control (HC) waits for 5 ms and asserts RCFG_GlobalSuspend.

Once asserted, clock start/stop logic, implemented in the design, sets a Clk_Stop signal that instructs the application to stop clk_48 and clk_12. HC stays in a suspend state until software switches the state to USB RESET/USB RESUME or RemoteWakeUp event is detected on the downstream ports (D+, D-). In case of software initiated reset/resume, the clocks must be powerd on before writing to the OHCI register. This is possible by monitoring the ClockControl register (offset 0x04 807C) status signals.

In the case of RemoteWakeUp (the Connect/Disconnect event detected on the downstream ports is considered RemoteWakeUp if the RWE bit is set in HcControl register), clock start/stop logic asserts the Clk_Start signal, which instructs the application to start the clocks. At the same time, the configuration block generates the WakeUp signal to the HC and enters the resume state.

13.3 Register Descriptions

The base address for the USB module in the PNX8526 is 0x04 8000.

Table 2: USB Module Register Summary

Address	Name	Description
0x04 8000	HcRevision	BCD value of HCI specification revision this device follow.
0x04 8004	HcControl	Defines the operating modes for the host controller.
0x04 8008	HcCommandStatus	The command register for the host controller driver.
0x04 800C	HcInterruptStatus	This register provide Status on various events that cause hardware interrupts.
0x04 8010	HcInterruptEnable	Each of this register bit correspond to an associated bit in the Hc Interrupt Status register.
0x04 8014	HcInterruptDisable	Each of this register bit correspond to an associated bit in the Hc InterruptStatus register.
0x04 8018	HcHCCA	Contains the physical address of the host controller communication area.

Table 2: USB Module Register Summary *…Continued*

Address	Name	Description
0x04 801C	HcPeriodCurentED	Contains the physical address of the current Isochronous or Interrupt Endpoint descriptor.
0x04 8020	HcControlHeadED	Contains the physical address of the first Endpoint descriptor of Control List.
0x04 8024	HcControlCurrentED	Contains the physical address of the current Endpoint descriptor of Control List.
0x04 8028	HcBulkHeadED	Contains the physical address of the first Endpoint descriptor of Bulk List.
0x04 802C	HcBulkCurrentED	Contains the physical address of the current Endpoint descriptor of Bulk List.
0x04 8030	HcDoneHead	Contains the physical address of the last completed transfer descriptor that was added to the done queue.
0x04 8034	HcFMInterval	Contain a 14-bit value which indicates the bit time interval in a Frame and a 15-bit value indicating the Full speed max packet size.
0x04 8038	HcFMRemaining	Is a 14-bit counter showing the bit time remaining in the current frame.
0x04 803C	HcFMNumber	16-bit counter which provide timing reference among events.
0x04 8040	HcPeriodicStart	A 14-bit programmable value which determine when is the earliest time HC should start processing the periodic list.
0x04 8044	HcLSThreshold	A 11-bit-values used by the HC to determine transfer size of LS packet.
0x04 8048	HcRHDescriptorA	The characteristics of root hub are described using this register and next register.
0x04 804C	HcRHDescriptorB	The characteristics of root hub are described using this register and previous register.
0x04 8050	HcRHStatus	Tells the status of the root hub.
0x04 8054	HcRHPortStatus[1]	The port status of USB port[0] is reported here.
0x04 8058	HcRHPortStatus[2]	The port status of USB port[1] is reported here.
0x04 807C	ClockControl	USB Clock start/stop status and associated set & clear bits.
0x04 8FF4	Powerdown	Powerdown mode
0x04 8FFC	Module ID	This is the module ID for the USB module.

USB REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 8000			***HcRevision***	
31:8			Reserved	
7:0	R	10	Rev	Contains BCD value of the HCI specification implemented in the PNX8526.
Offset 0x04 8004			***HcControl***	
31:11			Reserved	
10	R/W	0b	RWE	Remote Wakeup Enable
9	R/W	0b	RWC	Remote Wakeup Connected. This bit is set during POST by system firmware to indicate the HC supports remote wakeup signaling. Bit is reset after a h/w reset but doesn’t alter its value after a s/w reset.

USB REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
8	R/W	0b	IR	Interrupt Routing 0 = All interrupts are routed to normal host bus interrupt mechanism. 1 = Interrupts are sent to SMI. (not used in Viper)
7:6	R/W	00b	HCFS	Host Controller Functional State for USB 00b = USB RESET 01b = USB RESUME 10b = USB OPERATIONAL 11b = USB SUSPEND The hardware may automatically transition from USB SUSPEND state to USB RESUME state. The hardware enters USB SUSPEND state after a software reset, where as it enters USB RESET after a hardware reset.
5	R/W	0b	BLE	Bulk List Enable
4	R/W	0b	CLE	Control List Enable
3	R/W	0b	IE	Isochronous Enable
2	R/W	0b	PLE	Periodic List Enable
1:0	R/W	00b	CBSR	Control Bulk Service Ratio
Offset 0x04 8008			***HcCommandStatus***	
31:18			Reserved	
17:16	R	00b	SOC	Scheduling Overrun Count These bits are incremented on each scheduling overrun error.
15:4			Reserved	
3	W	0b	OCR	Ownership Change Request. This bit is auto cleared after setting 0x000C[30].
2	R/W	0b	BLF	Bulk List Filled This bit is set by s/w to indicate that there is a pending TD in the bulk list. This bit is cleared by h/w after servicing the TD.
1	R/W	0b	CLF	Control List Filled This bit is set by s/w to indicate that there is a pending TD in the control list. This bit is cleared by h/w after servicing the TD.
0	R/w	0b	HCR	Host Controller Reset This bit is set to initiate a s/w reset. The h/w moves into the USB SUSPEND state and resets all registers unless otherwise noted. This bit gets self cleared after the reset operation is over.
Offset 0x04 800C			***HcInterruptStatus****	
*These bits are automatically set by H/W and can be only cleared by writing a 1 to the corresponding bit.				
31			Reserved	
30	R/W	0b	OC	Ownership Change.
29:7		0b	Reserved	
6	R/W	0b	RHSC	Root Hub Status Change

USB REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
5	R/W	0b	FNO	Frame Number Overflow
4	R/W	0b	UE	Unrecoverable Error
3	R/W	0b	RD	Resume Detected
2	R/W	0b	SF	Start of Frame
1	R/W	0b	WDH	Writeback Done Head
0	R/W	0b	SO	Scheduling Overrun
Offset 0x04 8010			***HcInterruptEnable****	
*These bits may be set by writing a 1. Writing 0 has no effect. Bits are cleared by writing a 1 to corresponding bit in register 0x04 8014.				
31	R/W	0b	MIE	0 = Ignore by HC 1 = Enable interrupt generation due to events specified in the other bits of this register. Used by HCD as a Master Interrupt Enable.
30	R/W	0b	OC	0 = Ignore 1 = Enable interrupt generation due to Ownership Change.
29:7		0b	Reserved	
6	R/W	0b	RHSC	0 = Ignore 1 = Enable interrupt generation due to Root Hub Status Change.
5	R/W	0b	FNO	0 = Ignore 1 = Enable interrupt generation due to Frame Number Overflow.
4	R/W	0b	UE	0 = Ignore 1 = Enable interrupt generation due to Unrecoverable Error.
3	R/W	0b	RD	0 = Ignore 1 = Enable interrupt generation due to Resume Detect.
2	R/W	0b	SF	0 = Ignore 1 = Enable interrupt generation due to Start of Frame.
1	R/W	0b	WDH	0 = Ignore 1 = Enable interrupt generation due to Hc Done Head Writeback.
0	R/W	0b	SO	0 = Ignore 1 = Enable interrupt generation due to Scheduling Overrun.
Offset 0x04 8014			***HcInterruptDisable****	
*This register reads back the contents of register 0x04 8010. Writing a 1 disables the corresponding interrupt bit by clearing the bit in register 0x04 8010.				
31	W	0b	MIE	0 = Ignore by HC 1 = Disable interrupt generation due to events specified in the other bits of this register. Used by HCD as a Master Interrupt Enable.
30	W	0b	OC	0 = Ignore 1 = Disable interrupt generation due to Ownership Change.
29:7		0b	Reserved	
6	W	0b	RHSC	0 = Ignore 1 = Disable interrupt generation due to Root Hub Status Change.

USB REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
5	W	0b	FNO	0 = Ignore 1 = Disable interrupt generation due to Frame Number Overflow.
4	W	0b	UE	0 = Ignore 1 = Disable interrupt generation due to Unrecoverable Error.
3	W	0b	RD	0 = Ignore 1 = Disable interrupt generation due to Resume Detect.
2	W	0b	SF	0 = Ignore 1 = Disable interrupt generation due to Start of Frame.
1	W	0b	WDH	0 = Ignore 1 = Disable interrupt generation due to Hc Done Head Writeback.
0	W	0b	SO	0 = Ignore 1 = Disable interrupt generation due to Scheduling Overrun.
Offset 0x04 8018			***HcHCCA***	
31:8	R/W	0	HcHCCA	Base address of the Host Controller Communication Area.
7:0		-	Unused	
Offset 0x04 801C			***HcPCED***	
31:4	R	0	HcPCED	Period Current ED
3:0		-	Unused	
Offset 0x04 8020			***HcCHED***	
31:4	R/W	0	HcCHED	Control Head ED
3:0		-	Unused	
Offset 0x04 8024			***HcCCED***	
31:4	R/W	0	HcCCED	Control Current ED
3:0		-	Unused	
Offset 0x04 8028			***HcBHED***	
31:4	R/W	0	HcBHED	Bulk Head ED
3:0		-	Unused	
Offset 0x04 802C			***HcBCED***	
31:4	R/W	0	HcBCED	Bulk Current ED
3:0		-	Unused	
Offset 0x04 8030			***HcDH***	
31:4	R	0	HcDH	Done Head
3:0		-	Unused	
Offset 0x04 8034			***HcFMInterval***	
31	R/W	0b	FIT	Frame Interval Toggle
30:16	R/W	0	FSMPS	FS Largest Data Packet
15:14			Reserved	
13:0	R/W	2EDF	FI	Frame Interval

USB REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 8038			*HcFMRemaining*	
31	R	0b	FRT	Frame Remaining Toggle
30:14			Reserved	
13:0	R	0	FR	Frame Remaining
Offset 0x04 803C			*HcFMNumber*	
31:16			Reserved	
15:0	R	0	FN	Frame Number
Offset 0x04 8040			*HcPeriodic Start*	
31:14			Reserved	
13:0	R/W	0	PS	Periodic Start
Offset 0x04 8044			*HcLSThreshold*	
31:12			Reserved	
11:0	R/W	0628	LST	LS Threshold
Offset 0x04 8048			*HcRHDescriptorA*	
31:24	R/W	02	POTPGT	Power On To Power Good Time
23:13		000	Reserved	
12	R/W	0	NOCP	No Over Current Protection
11	R/W	0	OCPM	Over Current Protection Mode
10	R	0	DT	DeviceType
9	R/W	0	NPS	No Power Switching
8	R/W	0	PSM	Power Switching Mode
7:0	R	02	NDP	Number Downstream Ports
Offset 0x04 804C			*HcRHDescriptorB*	
31:19	R	0	Reserved	Reserved
18:17	R/W	0	PPCM	Port Power Control Mask for port #2 & port #1
16:3	R	0	Reserved	Reserved
2:1	R/W	0		
0	R	0	Reserved	Reserved
Offset 0x04 8050			*HcRHStatus*	
31	W	0	CRWE	(W) ClearRemote Wakeup Enable. Writing a 1 clears the DRWE bit.
30:18			Reserved	
17	R/W	0b	CCIC	Over Current Indicator Change. Automatically set by h/w whenever there is a change in OCI bit. Cleared by writing a 1. Write 0 has no effect.
16	W	0b	LPSC	(R) Local Power Status Change. Always Read 0. (W) Set Global Power.

USB REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15	R/W	0b	DRWE	(R) Device Remote Wakeup Enable. 0 = Disabled 1 = Enabled (W) Set Remote Wakeup Enable by writing a 1. Write 0 has no effect.
14:2			Reserved	
1	R	0b	OCI	Over Current Indicator
0	W	0b	LPS	(R) Local Power Status. Always read 0 (W) 1 = Clear Global Power. 0 = No effect.
Offset 0x04 8054			***HcRHPortStatus[1]***	
31:21		0	Reserved	
20	R/W	0b	PRSC	Port Reset Status Change This bit is set at the end of the 10-ms port reset signal. Writing a 1 clears this bit. Write 0 has no effect. 0 = Port reset is not complete 1 = Port reset is complete
19	R/W	0b	OCIC	Port Over Current Indicator Change This bit is valid only if the per port overcurrent condition bit is set. This bit is set when the root hub changes the Port Over Current Indicator bit. Writing a 1 clears this bit. Write 0 has no effect. 0 = No change in Port Over Current Indicator 1 = Port Over Current Indicator has changed
18	R/W	0b	PSSC	Port Suspend Status Change This bit is set when the full resume sequence has been completed. The HCD writes 1 to clear this bit. Write 0 has no effect. This bit is also cleared when the Reset Status Change is set. 0 = Resume is not completed 1 = Resume completed
17	R/W	0b	PESC	Port Enable Status Change This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from Software writes do not set this bit. This bit is cleared by writing a 1. Write 0 has no effect. 0 = No change in Port Enable Status 1 = Change in Port Enable Status
16	R/W	0b	CSC	(R) Connect Status Change This bit is set whenever a connect or disconnect event occurs. 0 = No change in Current Connect Status 1 = Change in Current Connect Status (W) Clear Connect Status Change Writing a 1 clears this status bit. Write 0 has no effect.
15:10			Reserved	

USB REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
9	R/W	NI	LSDA	(R) Low Speed Device Attached 0 = Full speed device is attached. 1 = Low speed device is attached. (W) Clear Port Power Writing a 1 clears the PPS bit. Write 0 has no effect.
8	R/W	0b	PPS	(R) Port Power Status 0 = Port power is off 1 = Port power is on. (W) Set Port Power Write 1 sets the bit. Write 0 has no effect.
7:5			Reserved	
4	R/W	0b	PRS	(R) Port Reset Status 0 = Port reset signal is not active. 1 = Port reset signal is active. This bit self clears after the PortResetStatusChange PRSC bit is set. (W) Set Port Reset Set the PRS bit by writing a 1 when CCS is 1. If CCS is a 0 when writing a 1, CSC bit will be set. Write 0 has no effect.
3	R/W	0b	POCI	(R) Port Over Current Indicator 0 = No overcurrent condition 1 = Overcurrent condition is detected. (W) Clear Suspend Status The HCD writes a 1 to resume and clear PSS. Write 0 has no effect.
2	R/W	0b	PSS	(R) Port Suspend Status. 0 = Port is not suspended. 1 = Port is suspended. This bit is cleared when Port Reset Status Change is set at the end of the port reset or when the HC is placed in the USB RESUME state. (W) Set Port Suspend When CCS is a 1, writing a 1 sets this bit. Write 0 has no effect. If CCS is 0 when writing a 1, CSC bit will be set instead.
1	R/W	0b	PES	(R) Port Enable Status 0 = Port is disabled. 1 = Port is enabled. (W) SetPortEnable. Writing a 1 sets this bit. CCS should be 1 to set this bit. If CCS is 0 while writing a 1 to this bit, only CSC bit will be set. This bit is also set if not already, at the completion of a port reset when Reset Status Change is set or port suspend when Suspend Status Change is set. Write 0 has no effect.

USB REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
0	R/W	0b	CCS	(R) Current Connect Status 0 = No device connected. 1 = Device Connected. (W) Clear Port Enable. Reset PES bit by writing a 1. Write 0 has no effect.
Offset 0x04 8058			***HcRHPortStatus[2]***	
Note: All registers are identical to HcRHPortStatus[1]. See Address 0x04 8054 for register descriptions.				
Offset 0x04 807C			***Clock Control***	
31:12	R	0000h	Reserved	
11	W	0h	ClearStpClk	Writing 1 to this bit clears the USB Stop Clock bit. Writing a 0 has no effect.
10	W	0b	SetStpClk	Writing 1 to this bit sets the USB Stop Clock bit. Writing a 0 has no effect. Intended for software debug.
9	W	0b	ClearStrtClk	Write 1 to this bit, to clear the USB Start Clock bit. Writing a 0 has no effect
8	W	0b	SetStrtClk	Write 1 to this bit, to set the USB Start Clock bit. Writing a 0 has no effect. Intended for software debug.
7:2	R	00h	Reserved	
1	R	0b	StpClk	Indicates USB clocks may be stopped. This bit is reset by writing a 1 to bit 11.
0	R	0b	StrtClk	Indicates USB clocks should be started within 50 uS. This bit is cleared by writing a 1 to bit 9 of this register.
Offset 0x04 8FF4			***POWERDOWN***	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x04 8FFC			***Module ID***	
31:16	R	0x0109	Module ID	Module ID assigned for USB
15:12	R	0	MajRev	Current major revision of this module
11:8	R	0	MinReV	Current minor revision of this module
7:0	R	0	ApertureSz	Aperture size of this module is 4kB.

Chapter 14: IEEE 1394

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

14.1 Introduction

The IEEE 1394_AVLINK_CORE supports transport of audio and video streams over an IEEE 1394 bus. Encryption and decryption according to the 5C content protection protocol is embedded to support encrypted MPEG2-TS streams.

The standard PHY-LINK interface allows this core to be connected to the Philips PDI1394P2x physical layer interface chip, or any other external PHY layer chip.

Control of the IEEE 1394_AVLINK_CORE takes place via the host interface. The host interface is a PI-Slave interface. This interface is used to access the control and status registers of the core and also those of the PHY chip. In addition this interface can be used to access the asynchronous FIFOs.

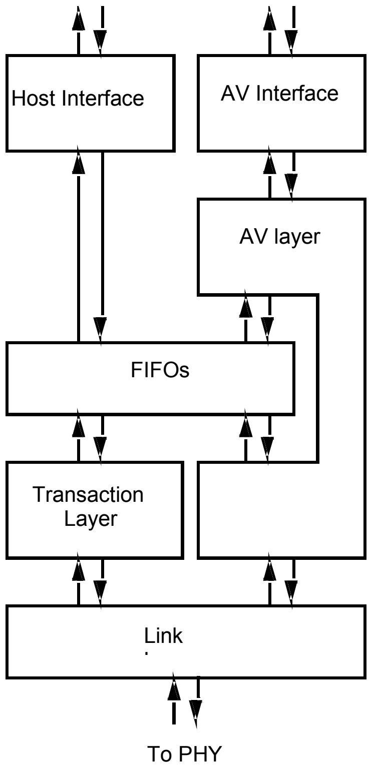


Figure 1: Top Level Block Diagram of IEEE 1394_AVLINK_CORE

The isochronous interfaces (one per stream) accept or supply a real-time 8-bit wide AV-stream. The AV module will process AV streams according to the IEC61883 standard. It takes care of timestamping, CIP header generation and isochronous packet assembly. The system is configurable and allows up to four simultaneous isochronous streams.

Closely related to the AV layer is the 5C content protection module. This is an M6 stream cipher which is able to encrypt and decrypt the AV source packet stream. The system assumes that the 5C defined authentication and key exchange is taken care of by software.

The embedded FIFOs serve as buffers between the application and the IEEE 1394 bus. The size of each FIFO is software programmable.

The bandwidth of the system is limited to an average of 90Mb/s per AV stream.

14.1.1 Features

Transport Streams

- Simultaneously sends and receives transport streams: sends out two while receiving one

Link Layer

- Interfaces to IEEE 1394-1995 and IEEE 1394.a PHY chip at speeds of 100, 200 or
400 Mbps.
- IEEE 1394.a compliant
- Indication of error condition on IEEE 1394 bus using interrupts
- Transmits and receives IEEE 1394 serial bus packets
- Isochronous resource manager capable
- Generates and checks 32-bit CRC
- Detects lost cycle start messages

Transaction Layer

- Automatic generation of acknowledge message
- Single/Dual Phase retry mechanism
- Multiple outstanding requests
- Flexible HW/SW support for split transaction timer

FIFO

- Four FIFOs for asynchronous packets
- Software programmable FIFO size

- Up to 8K x 32 total FIFO size

AV-Layer Module

- Configurable for transmitting and/or receiving multiple streams of up to 90 Mb/s
- ISO/IEC 61883 compliant (un)packing, supporting 188 bytes MPEG2-TS packets, 140 bytes DIRECTV packets, DV and mLAN.
- Optional timestamping
- CIP Header generation
- Standard MPEG streaming interface, 8-bit wide data, clock, sync and valid signal

5C Content Protection Module

- M6 stream cipher
- Supports full and restricted authentication

Performance

- 49.152 MHz system clock provided by Physical Layer chip.
- AV streaming clock up to 24 MHz
- Up to 90 Mb/s bandwidth per isochronous channel

14.2 Functional Description

14.2.1 PHY-LINK Interface

The PHY interface communicates with the PHY chip following the protocol described in the P1394.a specification.

Table 1: PHY Interface

Name	Type	Description
		Physical Interface
PHY_D[7:0]	I/O	PHY data IO. Data on pins 1:0 for 100Mbps packets, 3:0 for 200 Mbps packets, and 7:0 for 400 Mbps packets.
PHY_LREQ	O	Link Request. Used by the link to make bus requests and access PHY registers.
PHY_CTL[1:0]	I/O	PHY control. Indicates the mode for data on the DIn port.
PHY_ISO_N	I	Signals which type of isolation mode at the PHY-LINK interface is used: 0 = 1394-1995 Annex J type isolation, enable differentiator circuitry. 1 = direct connection or single capacitor isolation mode. This disables the differentiator circuitry.
CLK_L1394	I	PHY system clock. 49.152 MHz.
		PHY-Related Signals
C1394_PHY		Signals which type of PHY chip has been connected: 0 = 1394.a PHY 1 = 1394-1995 Annex J PHY This is 1 by default and is set/reset from Global2 register C1394_PHY[0] at offset 0x04D060.

14.2.1.1 PHY Interface Timing

This is a standard interface described in IEEE 1394-1995 Annex-J.

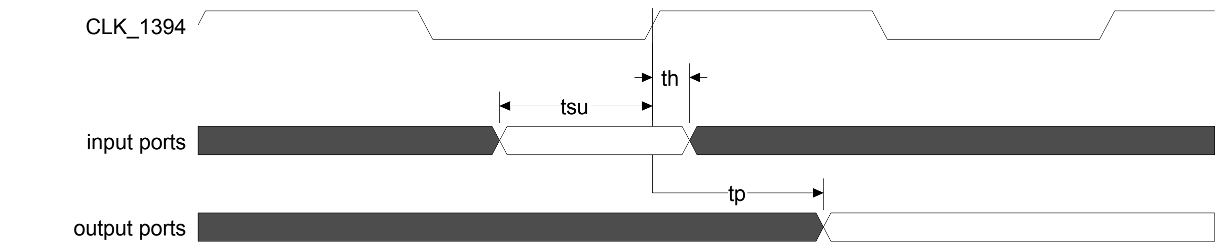


Figure 2: AC Timing for PHY-Link Interface Signals

Table 2: Phy-Link Interface Signals

		Limits		
Symbol	Parameter	Min	Max	Unit
f	CLK_1394 frequency	49.1 47	49.1 57	MHz
tsu	input setup time for PHY_Din, PHY_CTLin	2	-	ns
th	input hold time	0	-	ns
tp	output propagation delay for PHY_Dout, PHY_Douten_n, PHY_LREQ, PHY_LREQen_n, PHY_CTLout, PHY_CTLouten_n	-	2	ns

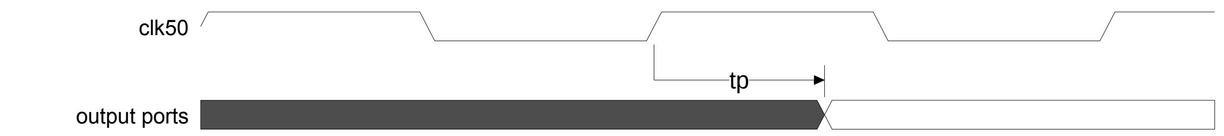


Figure 3: AC Timing for itx_clkout Related Signals

Table 3: itx_clkout Related Signals

		Limits		
Symbol	Parameter	Min	Max	Unit
tp	Output propagation delay for itx_clkout, itx_clken	0	8	ns

14.2.2 Host Interface

Reading Data from the Asynchronous Receive Queues

The protocol for reading asynchronous receive queues can be done such that it is insensitive to speculative read actions from the host system. A state diagram explains the mechanism.

The state machine below shows how quadlets can be read from the asynchronous receive queues. Asynchronous packets are stored in the queue. That makes the queue not empty. When there is data in the queue the transfer register will load that data. At this point the interrupt bit ASYINTACK.*QAV is asserted. Upon reading this bit, the CPU can read the data from the transfer register. Depending on the setting of the ASYCTL.EN_LDTR, the following will happen:

- When EN_LDTR is ‘0’, the transfer register will go to the TR_EMPTY state from which it can automatically load the next quadlet from the queue, as long as it is available.
- When EN_LDTR is ‘1’ the CPU has to clear *QAV bit to cause the state machine to reach the TR_EMPTY state. Only then can the transfer register be updated, if data are available.

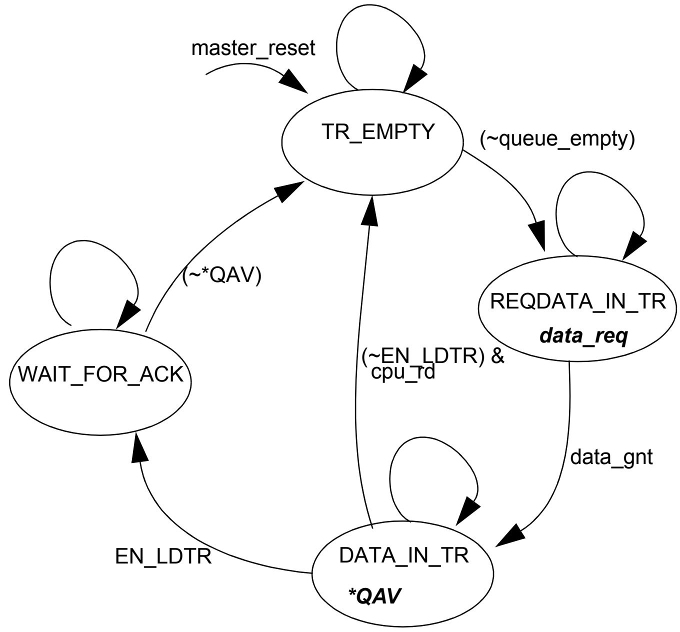


Figure 4: Asynchronous Receive Queue State Machine

14.2.3 Timer

A timer is available to aid the implementation of higher level protocols such as AV/C and HAVi. The basic timing unit of the timer is 81.4 ns (1 period of 12.288MHz). As TIMER.PRELOAD is 24 bits wide, the maximum timer timeout is 1.37s.

The timer counts down from the value specified in TIMER.PRELOAD and sets the LNKPHYINTACK.TIMER interrupt flag upon reaching 0. In normal timer mode (TIMER.TMCONT = 1) the timer will automatically reload with the TIMER.PRELOAD value every time it expires and continue counting, otherwise it will simply interrupt and stop.

Setting the TIMER.TMGOSTOP bit to logic 1 will cause the timer to start counting. When TIMER.TMBRE is set to logic 1, the timer will automatically start upon a IEEE 1394 bus reset, the TMGOSTOP function is disabled in this case.

When TIMER.TMCONT =1, failing to acknowledge the LNKPHYINTACK.TIMER interrupt has no effect on the timer operation.

14.2.4 Link Layer

The Link Layer is based on a module licensed from Apple Computer, Inc.

The transmitter block takes data from either the asynchronous transmit FIFO or the isochronous transmit FIFO and creates correctly formatted Serial Bus packets to be transmitted through the PHY interface. The transmitter also sends out the cycle start packets when the chip is cycle master.

The receiver block takes incoming data from the PHY interface, determines if the incoming data are addressed to this chip, detects cycle start messages, and passes valid packets to the receive FIFO.

The cycle timer module contains the cycle counter and cycle offset timer. The offset timer can be free running by reloading upon a low to high transition on the CycleIn pin, or take a reload value from the receiver based on the state of the CycleMaster and CycleSource pins. The timer can also be disabled using the CycleTimerEnable pin.

The cycle monitor watches chip activity and handles scheduling of isochronous activity. It detects missing cycle starts and instructs the transmitter to send a cycle start message when the CycleMaster pin is asserted.

The crc module generates a 32-bit auto-DIN crc for error detection. See the P1394 specification for details on the generation of the crc.

The PHY interface provides PHY-level services to the transmitter and receiver. This includes gaining access to the serial bus, sending packets, receiving packets, and sending and receiving ACKs.

14.2.5 FIFO

The FIFOs in the IEEE 1394_AVLINK_CORE are taking care of collecting the asynchronous and isochronous packets before transmission or after reception.

All the FIFOs are mapped onto a single SRAM. The address space that a FIFO occupies is flexible. For each FIFO a software programmable base register points to the beginning of the FIFO memory space, an end register points to the last address of the FIFO area. Additional pointers have been implemented in the logic to keep track of the address where to write data to or read data from.

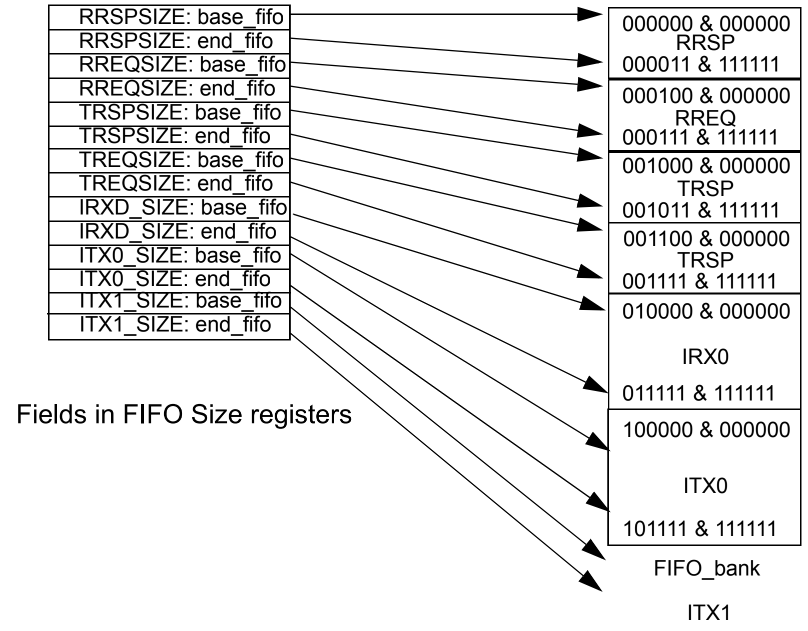


Figure 5: Default Setting of Size Programmable FIFOs

Each FIFO can be programmed to a certain size with a granularity of 64 Quadlets. The size is determined by the values of the base_fifo and end_fifo fields of the FIFO Size registers. The following formula applies:

fifo_size = (end_fifo - base_fifo +1) x 64 quadlets.

The FIFOs have been implemented on a single memory. The base_fifo and end_fifo fields are used to determine the physical start and end address of each FIFO inside the memory.

Remark: The start address of a FIFO is {fifo_addr[11:6] = base_fifo, fifo_addr[5:0] = 000000}. The end address of a FIFO is {fifo_addr[11:6] = end_fifo, fifo_addr[5:0] = 111111}. End_fifo should be larger than or equal to base_fifo. The hardware is not checking for any invalid values.

14.2.6 AV Interface and AV Layer

The AV layer allows AV packets to be transmitted from one node to another. The isochronous AV transmitter(s) and receiver(s) within the AV layer perform all the functions required to pack and unpack IEC61883 packet data for transfer over an IEEE 1394 network. Once the AV layer is properly configured for operation, no further host controller service should be required.

The number of isochronous AV transmitters and AV receivers is flexible. Multiple transmitters and/or receivers can be instantiated thereby supporting multiple isochronous channels. All these isochronous channels can be operated simultaneously.

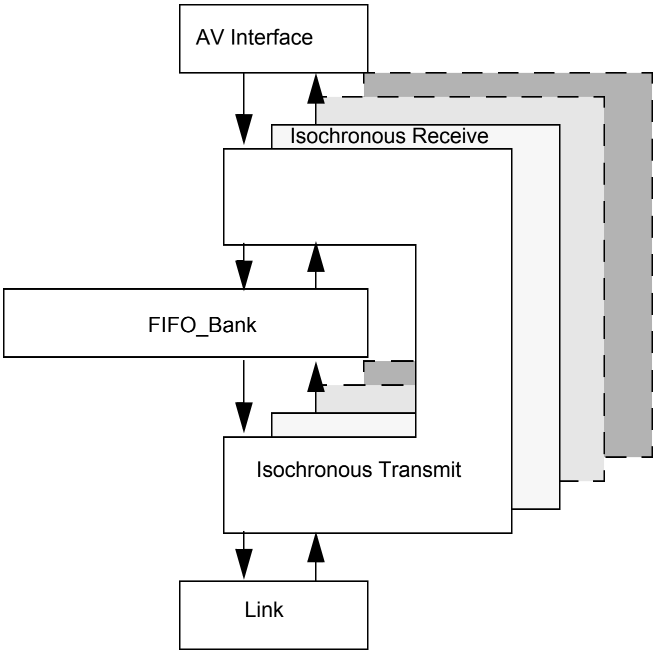


Figure 6: Isochronous Transmit and Receive Blocks

The IEEE 1394_AVLINK_CORE is specifically designed to support the IEC61883 International Standard of Digital Interface for Consumer Electronic Audio/Video Equipment. The IEC specification defines a scheme for mapping various types of AV data streams onto IEEE 1394 isochronous data packets. The standard also defines a software protocol for managing isochronous connections in an IEEE 1394 bus called Connection Management Protocol (CMP). It also provides a framework for transfer of functional commands, called Function Control Protocol (FCP).

A feature of the IEC61883 International Standard is the definition of Common Isochronous Packet (CIP) headers. These CIP headers contain information about the source and type of data stream mapped onto the isochronous packets.

The AV Layer supports the use of CIP headers. CIP headers are added to transmitted isochronous data packets at the AV data source. When receiving isochronous data packets, the AV layer automatically analyzes their CIP headers. The analysis of the CIP headers determines the method the AV layer uses to unpack the AV data from the isochronous data packets.

The information contained in the CIP headers is accessible via registers in the host interface.

(See IEC61883 International Standard of Digital Interface for Consumer Electronic Audio/Video Equipment for more details on CIP headers).

14.2.6.1 Isochronous IEC 61883 Transmitter

After the AV packet has been written into the AV layer, the AV layer creates an isochronous bus packet with the appropriate CIP header. The bus packet along with the CIP header is transferred over the appropriate isochronous channel/packet. The size and configuration of isochronous data packet payload transmitted is determined by the AV layer's configuration registers accessible through the host interface.

The data format from the isochronous transmitter is defined in Table 4. Basically each packet payload is preceded by a special quadlet (isoctl) containing control information for the link layer, such as length in bytes (16-bit) channel number, speed of transmission and tag and sync bits for the isochronous packet header.

The packing mode *pm* defines how bus packets are constructed from data blocks.

- In packing mode 0 any number of data blocks up to the specified maximum value *maxbl* can be put in a bus packet without any alignment restrictions.
- In packing mode 1 bus packets will always contain a fixed number of data blocks (specified by *maxbl*).
- Packing mode 2 is the MPEG mode: *maxbl* must either be a power of 2 less than 2^{FN} or an integer multiple of 2^{FN}. In the first case, the bus packet size is fixed to *maxbl* data blocks, and in the latter case the bus packet size is a variable (up to *maxbl* data blocks), but in this case bus packets will always contain entire source packets only. In both cases for packing mode 2, an additional alignment of bus packets and source packets is enforced such that changes in *maxbl* will take effect only after an appropriate boundary in a source packet has been reached. The alignment requirement is such that if the payload contains N data blocks (N = 1, 2, 4, 8 and $N \le 2^{FN}$), then the first data block in the payload must be aligned to an N/8 boundary in the source packet.
- Packing mode 3 will generate payloads without data blocks (just the CIP header).

Depending on the number of available data blocks (data delivered by the application and/or added by the level 1 packing), the value of *maxbl* and the packing mode (*pm*) the transmitter may or be not be able to generate a non-empty payload.

The contents of the output stream quadlets (isoctl + isochronous payload) is as follows:

- The first quadlet is a control value for the link layer which is used to construct the IEEE isochronous packet header. It is not part of the isochronous payload. The remaining quadlets form the payload.
- The first two payload quadlets are always transmitted and are known as the CIP header.
- The E flag is 1 in the last header quadlet (2nd CIP quadlet in this case) and 0 in the other CIP quadlets (1st in this case).
- The F flag is always 0 to indicate that this format is used.

- The SID field contains the Source ID of this node on the IEEE 1394 bus. This 6-bit value is obtained from the link layer which in turn reads it from the physical layer after each bus reset.
- DBS, FN, QPC, SPH, FMT and FDF are copied directly from the corresponding control registers.

Table 4: AV Isochronous Transmit Packet Format

<table>
<tr><th></th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>n</td><td colspan="16">length</td><td colspan="2">tag</td><td colspan="6">channel</td><td colspan="2"><r></td><td colspan="2">spd</td><td colspan="4">sy</td></tr>
<tr><td>n+1</td><td>e</td><td>f</td><td colspan="6">sid</td><td colspan="8">dbs</td><td colspan="2">fn</td><td colspan="3">qpc
sph</td><td></td><td colspan="2"><r
></td><td colspan="8">dbc</td></tr>
<tr><td>n+2</td><td>e</td><td>f</td><td colspan="6">fmt</td><td colspan="8">fdf</td><td colspan="16">fdf/syt</td></tr>
<tr><td>...
n+m</td><td colspan="32">N datablocks of DBS quadlets each (N=>20)</td></tr>
</table>

DBC is a data block continuity counter. It specifies the data block sequence number of the first data block that follows the CIP header. If N data blocks will follow, then the value of DBC in the next AV payload will be N higher than the current value (counting modulo 256). When an isochronous AV transmitter is initialized this value is reset to 0. The numbering shall be aligned with the source packets such that if a source packet is divided in k (1, 2, 4 or 8) data blocks then the first data block of each source packet will have a data block number *dbc* with *dbc* modulo k = 0. If N = 0 in an AV payload then DBC will contain the same value as it would have when N > 0, i.e. the sequence number of the next expected data block.

The SYT field is actually part of the FDF field which is 24 bits wide. For certain data formats the lower 16 bits of FDF can be replaced by a so called synchronization timestamp (SYT). This SYT insertion is enabled when the EN_FS control bit has been set. When enabled the SYT field contains either a valid stamp (*syt* signal) or a value of all 1’s, regardless of the FDF control register value. When disabled the SYT field contains the lower 16 bits from the FDF control register value.

Packing order of bits in bytes and bytes in quadlets is always such that the most significant bit of each byte is transmitted first on the IEEE 1394 bus and that bytes are transmitted in the same order as they were delivered by the application (most significant byte of a quadlet first). The most significant byte of each quadlet is transmitted first.

14.2.6.2 Isochronous IEC 61883 Receiver

The isochronous AV receiver acts as a stream converter. Both its main input and output consists of a data stream each with certain properties and timing characteristics. Specific format conversion parameters are extracted from the input stream. Timing is based on control signals for both input and output side and data extracted from the input stream. Because of different timing characteristics and bus width of input and output it will be necessary to buffer data during the conversion.

14.2.6.3 Format of Stream Received from Link Layer

The input stream format is nearly identical to the output stream format of the isochronous AV transmitter. The format of the isochronous transmitter is defined in Table 5.

Table 5: AV Isochronous Receiver Packet Format

<table>
<tr><th></th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th></th></tr>
<tr><td>n</td><td colspan="16">length</td><td colspan="2">tag</td><td colspan="6">channel</td><td colspan="4">tcode</td><td colspan="4">sy</td><td>isohdr</td></tr>
<tr><td>n+1</td><td>e</td><td>f</td><td colspan="6">sid</td><td colspan="8">dbs</td><td colspan="2">fn</td><td colspan="3">qpc</td><td>sph</td><td colspan="2">&lt;r&gt;</td><td colspan="8">dbc</td><td rowspan="2">CIP header</td></tr>
<tr><td>n+2</td><td>e</td><td>f</td><td colspan="6">fmt</td><td colspan="8">fdf</td><td colspan="16">fdf/syt</td></tr>
<tr><td>...</td><td colspan="32">N datablocks of DBS quadlets each (N>= 0)</td><td>AV payload</td></tr>
<tr><td>n+m</td><td colspan="16">&lt;reserved&gt;</td><td colspan="2">spd</td><td colspan="10">&lt;reserved&gt;</td><td colspan="4">errcode</td><td>isostat</td></tr>
</table>

The length field indicates the number of bytes in the payload (CIP header + AV payload). The tag and channel number are compared with a programmable value and the receiver will only accept the remainder of the packet if these values match. The SY code is available through a status register. Based on the DBS, FN, QPC and DBC fields the receiver must first attempt to reconstruct the source packets and check for sequence errors. As soon as all sequence errors and crc errors for all data blocks of any specific source packet have been computed, the source packet shall be marked with a 2-bit error field indicating whether or not that source packet contains a sequence error or a crc error. At that time the source packet becomes available for extraction and delivery of application packets.

The SID, FMT and FDF fields are not interpreted by the receiver.

During the final quadlet (isostat) the spd and errcode fields are stored for retrieval by the host CPU through a status register. A crc error in the bus packet data field results in errcode 13.

14.2.6.4 Synchronization Timestamps

If control bit *en_fs* = ‘0’ then the lower 16 bits of the 2nd CIP header quadlet are not interpreted as SYT stamps and not processed at all, and the *fsync* output remains unchanged. If *en_fs* = ‘1’ then the SYT field is checked for validity become equal to the reference time the SYT stamp is discarded and the *fsync* output is activated for a duration of 244ns (12 cycles of the 49.152MHz clock). Up to six SYT stamps can be queued while the previous one is still being processed. When the SYT queue in the isochronous receiver is full then the most recently received SYT timestamp is overwritten with the next arriving SYT timestamp.

14.2.6.5 mLAN

The 61883 isochronous transmitter has some features to support mLAN. These can be turned on by setting ITX_n_PKCTL.mLAN to logic 1.

The oldest SYT timestamp in the SYT queue will be sent first but it will not be transmitted with an empty bus packet. Any pending SYT timestamp will be held until the next non-empty bus packet is sent.

At the moment of transmission the SYT timestamp should at least point one cycle in the future. If it points to a time that is less than one cycle in the future, it will be discarded.

The SYT queue in the isochronous transmitter can store four entries, the SYT queue in the isochronous receiver can store six entries.This supports the case where an 8KHz signal is applied to itx_fsync and mLAN=1 and SYT_Delay = 10b. Assuming there is data on every cycle, the receiver will receive an SYT timestamp each cycle with the first SYT timestamp pointing just less than six cycles in the future.

When the SYT queue in the isochronous receiver is full then the most recently received SYT timestamp is overwritten with the next arriving SYT timestamp.

When the SYT queue in the isochronous transmitter is full, then any next arriving SYT will be ignored until a location in the queue gets opened up.

14.2.6.6 Digital Camera

One feature has been implemented specifically to support the IEEE 1394 digital camera specification. This spec requires that the SY field be set to 0001b at the first isochronous data block of a frame, and be set to 0000b on all other isochronous data blocks. The itx_sysync port has been dedicated to this function.

The isochronous transmitter will detect for which data byte the itx_sysync port was asserted. The isochronous bus packet that carries this byte in its payload will have bit 0 of the SY field in the isochronous packet header being set to logic 1, see Figure 7.

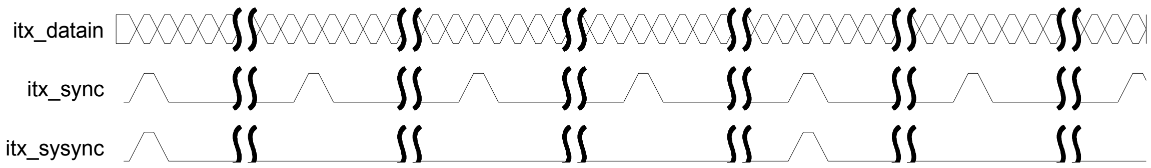


Figure 7: Behavior of Sysync at the Isochronous Transmitter

The isochronous receiver will assert the irx_sysync pin for the compete length of the packet for which bit 0 of the SY field in the packet header was set to logic 1, see Figure 8.

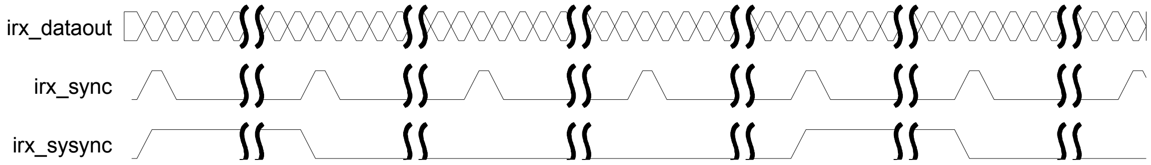


Figure 8: Behavior of Sysync at the Isochronous Receiver

Remark: It is assumed that an application source packet coincides with an asynchronous bus packet.

14.2.6.7 AV Receiver Interface

Delivery to the host application is based on an application clock (*aclk*) which must be equal or higher in frequency (on average) to the corresponding *aclk* at the transmitter node (otherwise the receiver buffer will overflow). For timestamped data this is more critical because packet delivery is based on this clock and negative deviations from the correct frequency can cause the next timestamp to be ‘late’, i.e. point so far into the past that it is interpreted as a future value (25 bits represent a window of 1 second).

Data are delivered on an 8-bit bus (*adata_a*) during N consecutive clock cycles of the *aclk* clock for an N byte application packet. The cycle during which the first data byte of the application packet is present on the *adata_a* output bus is indicated by the *async_a* output. Both the *aerr_a* status output bits remain constant during the entire delivery of each application packet. They are set to the correct value before or at the same time as the first data byte appears on the *adata_a* bus.

14.2.7 Content Protection

The IEEE 1394_AVLINK_CORE provides content protection according to the 5C protocol. Two modules are available. An accelerator for authentication and key exchange such that full authentication can be supported. The second is the M6 stream cipher for encryption and decryption of an isochronous stream.

14.2.8 Asynchronous Packet Transmitter

The asynchronous transmitter takes care of the transmission of packets present in the request and response memory queue. It will retransmit packets a programmable number of times in case the destination node is busy. If a transmitted request packet results in an acknowledge pending returned (split transaction), the transmitter will wait a programmable amount of time for the expected response to be returned, before transmitting the next available request packet. After a transmission has been completed, it assembles a confirmation which is passed on to the receiver module and saved in one of the receiver queues, depending on the packet type that was transmitted.

14.2.8.1 Asynchronous Transmit Packet Formats

These sections describe the formats in which packets need to be delivered to the queues (FIFOs) for transmission. There are four basic formats as shown in Table 6.

These packet formats are described in the following sections. For more information, see the IEEE 1394 Specification. Gray fields are reserved and should be set to zero values.

Table 6: Asynchronous Transmit Packet Formats

Format	Usage	Transaction Codes
No-data packet	Quadlet read requests	4
	Quadlet/block write responses	2
Quadlet packet	Qaudlet write requests	0
	Quadlet read responses	6
Block packet	Block read requests	5
	Block write requests	1
	Block read responses	7
	Lock requests	9
	Lock responses	B_{hex}
Unformatted transmit	Concatenated self-ID / PHY packets	E_{hex}

No-Data Transmit

The no-data transmit formats are shown in the following Tables. The first quadlet contains packet control information. The second and third quadlets contain 16-bit destination ID and either the 48-bit, quadlet aligned destination offset (for requests) or the response code (for responses).

Table 7: Quadlet Read Request Transmit Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													0	spd		tLabel						rt		tCode				0000			
destinationID																destinationOffsetHigh															
destinationOffsetLow																															

Table 8: Quadlet/Block Write Response Packet Transmit Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													0	spd		tLabel						rt		tCode				0000			
destinationID																rCode															

Table 9: No-Data Transmit Format

Field Name	Description
spd	Indicates the speed at which this packet is to be sent: 00=100 MB, 01=200 MB, 10=400 MB, and 11=undefined.
tLabel	This field is the transaction label, which is used to pair up a response packet with its corresponding request packet.
rt	The retry code for this packet. Supported values are: 00=retry1, and 01=retryX.
tCode	The transaction code for this packet.
DestinationID	Contains a node ID value.
DestinationOffsetHigh DestinationOffsetLow	The concatenation of these two field addresses a quadlet in the destination node's address space.
rCode	Response code for write response packet. rCodeMeaning 1-3Reserved 4Resource conflict detected by responding agent. Request may be retried 5Hardware error. Data not available 6Field within request packet header contains unsupported or invalid value 7Address location within specified node not accessible. 8Reserved 9Node successfully completed requested action A-FReserved

Quadlet Transmit

Three quadlet transmit formats are shown below. The first quadlet contains packet control information. The second and third quadlets contain 16-bit destination ID and either the 48-bit quadlet-aligned destination offset (for requests) or the response code (for responses). The fourth quadlet contains the quadlet data for read response and write quadlet request formats or the upper 16 bits contain the data length for the block read request format.

Table 10: Quadlet Write Request Transmit Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													0	spd		tLabel						rt		tCode				0000			
destinationID																destinationOffsetHigh															
destinationOffsetLow																															
quadlet data																															

Table 11: Quadlet Read Response Transmit Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													0	spd		tLabel						rt		tCode				0000			
destinationID																rCode															
quadlet data																															

Table 12: Block Read Request Transmit Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													0	spd		tLabel						rt		tCode				0000			
destinationID																destinationOffsetHigh															
destinationOffsetLow																															
data length																															

Table 13: Quadlet Transmit Fields

Field Name	Description
spd, tLabel, rt, tCode, destinationID, destinationOffsetHigh, destinationOffsetLow, rCode	See Table 9 for more information.
Quadlet data	For quadlet write requests and quadlet read responses, this field holds the data to be transferred
Data length	The number of bytes requested in a block read request

Block Transmit

The block transmit format is shown below. This is the generic format for reads and writes. The first quadlet contains packet control information. The second and third quadlets contain the 16-bit destination node ID and either the 48-bit destination offset (for requests) or the response code and reserved data (for responses). The fourth quadlet contains the length of the data field and the extended transaction code (all zeros except for lock transaction). The block data, if any, follows the extended transaction code.

Table 14: Block Packet Transmit Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n														0	spd		tLabel						rt		tCode				0000			
n+1	destinationID																destinationOffsetHigh															
n+2	destinationOffsetLow																															
n+3	data length																extendedTcode															
n+4	Block data																															
...																																
n+m									padding (if needed)																							

Table 15: Block Read or Lock Response Transmit Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n														0	spd		tLabel						rt		tCode				0000			
n+1	destinationID																rCode															
n+2																																
n+3	data length																extendedTcode															
...	Block data																															
n+m									padding (if needed)																							

Table 16: Block Transmit Field

Field Name	Description
spd, tLabel, rt, tCode, destinationID, destinationOffsetHigh, destinationOffsetLow, rCode	See Table 9 for more information.
data length	The number of bytes of data to be transmitted in this packet
extended Tcode	The tCode indicates a lock transaction, which specifies the actual lock action to be performed with the data in this packet.
Block data	The data to be sent. If dataLength=0, no data should be written into the FIFO for this field. Regardless of the destination or source alignment of the data, the first byte of the block must appear in the high order byte of the first quadlet.
padding	If the dataLength mod 4 is not zero, then zero-value bytes are added onto the end of the packet to guarantee a whole number of quadlets is sent.

Unformatted Transmit

The unformatted transmit format is shown in Table 17. The first quadlet contains packet control information. The remaining quadlets contain data that is transmitted without any formatting on the bus. No CRC is appended on the packet, nor is any data in the first quadlet sent. This is used to send PHY configuration and Link-on packets.

Remark: The bit-inverted check quadlet must be included in the FIFO since the AV Link core will not generate it.

Table 17: Unformatted Transmit Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n															spd										1110				0000			
n+1	destinationID																destinationOffsetHigh															
n+2	destinationOffsetLow																															
n+3	data length																extendedTcode															
...	Unformatted packet data																															
n+m																																

Asynchronous Stream Transmit

The asynchronous stream packet format is shown below. The first quadlet contains packet control information. The second quadlet contains data length, tag channel number, and synchronization code. The third quadlet contains the data length in quadlets. The data length can be zero for empty asynchronous stream packets.

Table 18: Asynchronous Stream Packet Transmit Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n															spd		tLabel						1		1110				priority			
n+1	data length																tag		channel						1010				sy			
...	block data																															
n+m												padding (if needed)																				

When a packet conforming to this format is written to either asynchronous transmit FIFO, an asynchronous stream packet (identical on the cable to an isochronous packet) will be transmitted during the asynchronous phase of a bus cycle.

Transaction Data Confirmation Formats

After a request, response, or asynchronous stream packet is transmitted, the asynchronous transmitter assembles a confirmation which is used to confirm the result of the transmission to the higher layers. Packets transmitted from the Transmit

Request FIFO are confirmed by a confirmation written into the Receive Request FIFO and packets transmitted from the Transmit Response FIFO are confirmed by a confirmation written into the Receive Response FIFO.

Table 19: Request and Response Confirmation Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																tLabel						00		1000				conf			

Table 20: Confirmation Codes

Table 21:

Code	Description
0	Non-broadcast packet transmitted; addressed node returned no acknowledge.
1	Broadcast packet transmitted or non-broadcast packet transmitted; addressed node returned an acknowledge complete.
2	Non-broadcast packet transmitted; addressed node returned an acknowledge pending.
4	Retry limit exceeded; destination node hasn't accepted the non-broadcast packet within the maximum number of retries.
D_{16}	Acknowledge data error received (transaction complete).
E_{16}	Acknowledge type error received (transaction complete).

[21-1] All other codes are reserved.

For every packet written in a transmitter queue by the CPU, there will be one confirmation written in the corresponding receiver queue by the AV layer logic.

14.2.9 Asynchronous Packet Receiver (ARX)

The asynchronous packet receiver module accepts incoming packets delivered by the link and delivers them depending on the packet type to the right memory queue. Response type packets are delivered to the response memory queue, all other packets (request, selfid and PHY packets) to the request memory queue. The receiver module also stores confirmations passed to it by the transmitter module in the right memory queue and detects incoming solicited responses. Solicited responses are detected by means of the pending request ID supplied to the receiver module (during the split transaction timeout interval) by the transmitter. During the split transaction timeout interval, the receiver can be programmed to discard all other (unsolicited) responses.

14.2.9.1 Asynchronous Receive Packet Formats

This section describes the asynchronous receive packet formats. Four basic asynchronous data packet formats and one confirmation format exist:

Table 22: Asynchronous Data Packet Formats

Format	Usage	Transaction Code
No-packet data	Quadlet read requests	4
	Quadlet/block write responses	2
Quadlet packet	Quadlet write requests	0
	Quadlet read responses	6
Block packet	Block read requests	5
-	Block write requests	1
-	Block read responses	7
-	Lock requests	9
-	Lock responses	B_{hex}
Self-ID / PHY packet	Concatenated self-ID / PHY packets	E_{hex}
Confirmation packet	Confirmation of packet transmission	8

Each packet format uses several fields. More information about most of these fields can be found in the IEEE 1394 Specification

.

Table 23: Asynchronous Receive Fields

Field Name	Description
destinationID	This field is the concatenation of bus numbers (or all ones for "local bus") and nodeNumbers (or all ones for broadcast) for this node.
tLabel	This field is the transaction label, which is used to pair up a response packet with its corresponding request packet.
rt	The retry code for this packet. 00=retry1, 01=retryX, 10=retryA, 11=retryB.
tCode	The transaction code for this packet.
priority	The priority level for this packet (0000 for cable environment).
sourceID	This is the node ID of the sender of this packet.
destinationOffsetHigh, destinationOffsetLow	The concatenation of these two field addresses a quadlet in this node's address space
rCode	Response code for response packets.
quadlet data	For quadlet write requests and quadlet read responses, this field holds the data received.
dataLength	The number of bytes of data to be received in a block packet.
extendedTcode	If the tCode indicates a lock transaction, this specifies the actual lock action to be performed with the data in this packet.
block data	The data received. If dataLength=0, no data will be written into the FIFO for this field. Regardless of the destination or source alignment of the data, the first byte of the block will appear in the high order byte of the first quadlet.

Table 23: Asynchronous Receive Fields

Field Name	Description
padding	If the dataLength mod 4 is not zero, then zero-value bytes are added onto the end of the packet to guarantee that a whole number of quadlets is sent.
u	Unsolicited response tag bit. This bit is set to one (1) if the received response was unsolicited.
ackSent	This field contains the acknowledge code that the link layer returned to the sender of the received packet. For packets that do not need to be acknowledged (such as broadcasts) the field contains the acknowledge value that would have been sent if an acknowledge had been required. The values for this field are listed in the following table.

Table 24: Acknowledge Codes

Code	Name	Description
0001	ack_complete	The node has successfully accepted the packet. If the packet was a request subaction, the destination node has successfully completed the transaction and no response subaction shall follow.
0010	ack_pending	The node has successfully accepted the packet. If the packet was a request subaction, a response subaction will follow at a later time. This code shall not be returned for a response subaction.
0100	ack_busy_X	The packet could not be accepted. The destination transaction layer may accept the packet on a retry of the subaction.
0101	ack_busy_A	The packet could not be accepted. The destination transaction layer will accept the packet when the node is not busy during the next occurrence of retry phase A.
0110	ack_busy_B	The packet could not be accepted. The destination transaction layer will accept the packet when the node is not busy during the next occurrence of retry phase B.
1101	ack_data_error	The node could not accept the block packet because the data field failed the CRC check, or because the length of the data block payload did not match the length contained in the dataLength field. This code shall not be returned for any packet that does not have a data block payload.
1110	ack_type_error	A field in the request packet header was set to an unsupported or incorrect value, or an invalid transaction was attempted (e.g., a write to a read-only address).
0000, 0011, 0111 - 1100, and 1111	reserved	This revision of the IEEE 1394_AVLINK_CORE will not generate other acknowledge codes, but may receive them from newer (1394A) links. In that case, these new values will show up here.

No-Data Receive

The no-data receive formats are shown below. The first quadlet contains the destination node ID and the rest of the packet header. The second and third quadlets contain16-bit source ID and either the 48-bit, quadlet-aligned destination offset (for requests) or the response code (for responses). The last quadlet contains packet reception status.

Table 25: Quadlet Read Request Receive Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
destinationID																tLabel						rt		tCode				priority			
sourceID																destinationOffsetHigh															
destinationOffsetLow																															
														spd														ackSent			

Table 26: Write Response Receive Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
destinationID																tLabel						rt		tCode				priority			
sourceID																rCode															
														spd													u	ackSent			

Quadlet Receive

The quadlet receive formats are shown below. The first quadlet contains the destination node ID and the rest of the packet header. The second and third quadlets contain16-bit source ID and either the 48-bit, quadlet-aligned destination offset (for requests) or the response code (for responses). The fourth quadlet is the quadlet data for read responses and write quadlet requests, and is the data length and reserved for block read requests. The last quadlet contains packet reception status.

Table 27: Quadlet Write Request Receive Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
destinationID																tLabel						rt		tCode				priority			
sourceID																destinationOffsetHigh															
destinationOffsetLow																															
quadlet data																															
														spd														ackSent			

Table 28: Quadlet Read Response Receive Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
destinationID																tLabel						rt		tCode				priority			
sourceID																rCode															
quadlet data																															
														spd													u	ackSent			

Table 29: Block Read Request Receive Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
destinationID																tLabel						rt		tCode				priority			
sourceID																destinationOffsetHigh															
destinationOffsetLow																															
data length																															
														spd														ackSent			

Block Receive

The block receive format is shown below. The first quadlet contains the destination node ID and the rest of the packet header. The second and third quadlets contain 16-bit source ID and either the 48-bit destination offset (for requests) or the response code and reserved data (for responses). The fourth quadlet contains the length of the data field and the extended transaction code (all zeros except for lock transactions). The block data, if any, follows the extended code. The last quadlet contains packet reception status.

Table 30: Block Write or Lock Request Receive Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n	destinationID																tLabel						rt		tCode				0000			
n+1	sourceID																destinationOffsetHigh															
n+2	destinationOffsetLow																															
n+3	data length																extendedTcode															
...	Block data																															
									padding (if needed)																							
n+m															spd														ackSent			

Table 31: Block Read or Lock Response Receive Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n	destinationID																tLabel						rt		tCode				0000			
n+1	sourceID																rCode															
n+2																																
n+3	data length																extendedTcode															
...	Block data																															
									padding (if needed)																							
n+m															spd													u	ackSent			

Self-ID and PHY Packets Receive

The self-ID and PHY packet receive formats are shown below. The first quadlet contains a synthesized packet header with a tCode of 0xE. For self-ID information, the remaining quadlets contain data that is received from the time a bus reset ends to the first subaction gap. This is the concatenation of all the self-ID packets received.

Remark: The bit-inverted check quadlet is included in the Read Request FIFO and the application must check it.

Table 32: Self-ID Receive Format

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
n																									1110				0000			
n+1	Self ID packet data																															
...																																
n+m															spd														ackSent			

The "ackSent" field will either be "ACK_DATA_ERROR" if a non-quadlet-aligned packet is received or there was a data overrun, or "ACK_COMPLETE" if the entire string of self-ID packets was received.

Table 33: PHY Packet Receive Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								1110				0000			
PHY packet first quadlet																															

For PHY packets, there is a single following quadlet which is the first quadlet of the PHY packet. The check quadlet has already been verified and is not included.

Transaction Data Confirmation Formats

After a packet from one of the queues has been transmitted, the asynchronous transmitter assembles a confirmation which is used to confirm the result of the transmission to the higher layers. Separate confirmations are assembled for request and response transmissions. Request confirmations are written into the request queue and response confirmations are written into the response queue.

Table 34: Request and Response Confirmation Format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">destinationID</td><td colspan="6">tLabel</td><td colspan="2">00</td><td colspan="4">1000</td><td colspan="4">conf</td></tr>
</table>

Table 35: Confirmation Codes

Code	Description
0	Non-broadcast packet transmitted; addressed node returned no acknowledge.
1	Broadcast packet transmitted or non-broadcast packet transmitted; addressed node returned an acknowledge complete.
2	Non-broadcast packet transmitted; addressed node returned an acknowledge pending.
4	Retry limit exceeded; destination node hasn't accepted the non-broadcast packet within the maximum number of retries.
D_{16}	Acknowledge data error received (transaction complete).
E_{16}	Acknowledge type error received (transaction complete).

[35-1] All other codes are reserved.

For every packet written in a transmitter queue by the CPU, there will be one confirmation written in the corresponding receiver queue by the AV layer logic.

14.2.10 Interrupts

The IEEE 1394_AVLINK_CORE provides a single interrupt.

To aid in determining the exact source of an interrupt, the area of the device generating an interrupt is indicated in a two-layer structure. (See the Interrupt Hierarchy diagram, Figure 9 on page 14-330, for more detail).

The first layer of the structure consists of bit [6] and bit [3:0] of the GLOBCSR. These contain interrupt status bits from general sections of the device; the link layer controller, the AV transmitter area, the AV receiver area, and the asynchronous transceiver area.

The second layer of the interrupt structure consists of several interrupt acknowledge registers (ITX0_INTACK, ITX1_INTACK, IRX_INTACK, ASYINTACK and LNKPHYINTACK), which contain interrupt bits for specific functions: AV transmitters, AV receivers, asynchronous transmitters and receivers and link layer controller.

Each interrupt bit in the interrupt acknowledge registers can be individually enabled to generate a chip-level interrupt.

14.2.10.1 Determining and Clearing Interrupts

When responding to an interrupt event generated by the IEEE 1394_AVLINK_ CORE, or when operating in polled mode, the first register to be examined is the GLOBCSR register. This will show which interrupt acknowledge register contains the bit(s) that is generating the interrupt.

To acknowledge and clear a standing interrupt, the bit in any of the interrupt acknowledge registers causing the interrupt has to be written to a logic `1'. Writing a value of `0' to the bit has no effect.

After all the interrupt bits are dealt with in the appropriate interrupt acknowledge register, the interrupt status indication will automatically clear in the GLOBCSR (ITX0INT, ITX1INT, IRXINT, ASYTX/RX and LNKPHYINTACK).

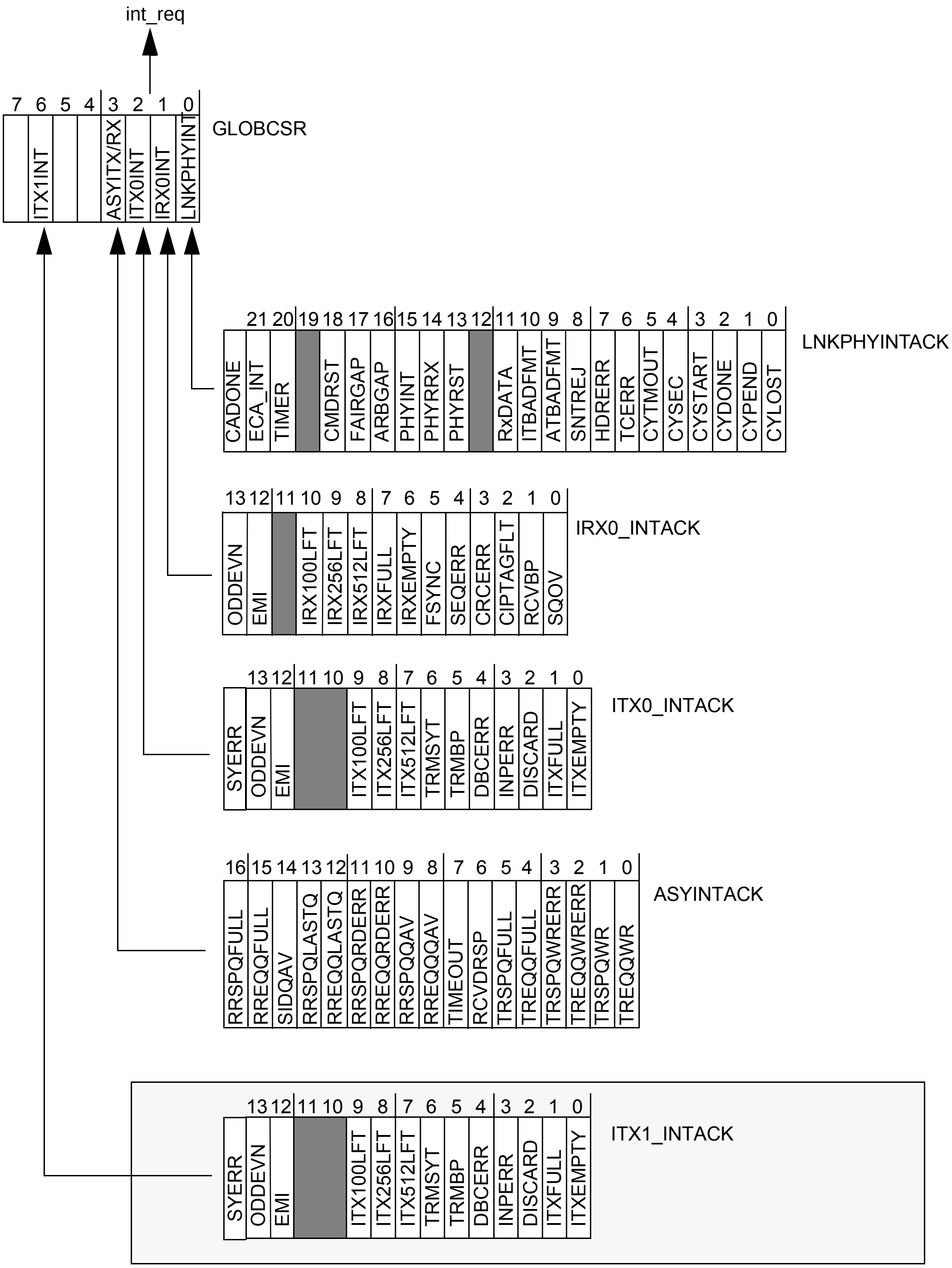


Figure 9: Interrupt Hierarchy Diagram of IEEE 1394_AVLINK_CORE

14.2.11 Reset

14.2.11.1 Reset from Host Bus Interface

During (asynchronous) power-on reset the (synchronous) reset bits for the Apple link core will be set. When reset_n is removed and a 24.576 MHz clock is present (generated at the top chip level from the incoming PHY clock) these synchronous reset bits will be automatically cleared after two cycles of the 24.576 MHz clock. This takes the Apple link core out of the reset state and enables processing of the incoming self-ID packets. If the reset control bits are set by the CPU then they will not clear themselves automatically; only the CPU can clear them or a new power-on reset.

PHY Register 0 Read

PHY register 0 contains important information for the link and node management software such as root status and phy_id. The link needs this information to function properly. Whenever the PHY sends register 0 to the link the relevant information will be extracted and placed in the corresponding link control and status registers.

Although the Apple code based PHY is supposed to send the contents of its register 0 to the link after power on and bus reset there are two reasons to let the link specifically read this register:

- Not all PHYs will be based on Apple code and thus not all PHYs are expected to send their register 0 contents automatically whenever it changes.
- The LNK may not be ready by the time the PHY sends its register 0. Therefore the AVLink contains a special circuit that will issue a PHY read command for register 0 directly after the first subaction gap reported by the PHY, and after a power-on reset or bus reset. If the CPU has already engaged the PHY/LNK interface in another read or write command, then the circuit will wait until the first possible opportunity to issue the read command.

14.2.11.2 Software Reset

The control registers for isochronous receivers and transmitters have a reset bit in the irx and itx.

The LNKCTL register contains the bit RSTTx. When set to one, this synchronously resets the transmitter within the link layer.

The LNKCTL register also contains the bit RSTRx. When set to one, this synchronously resets the receiver within the link layer.

Setting the isochronous transmitter reset bit (RST_Tx) to ‘1’ resets the transmitter. In order for synchronous reset of ITX to work properly, the application must supply an av<x>_aclk_tx to ensure that the reset bit is kept (programmed) HIGH for at least the duration of one av<x>_aclk_tx period. Failure to do so may cause the application interface of this module to be improperly reset (or not reset at all).

Setting the isochronous receiver reset bit (RST_Rx) to ‘1’ resets the receiver.
In order for synchronous reset of IRX to work properly, the application must supply an av<x>_aclk_rx to ensure that the reset bit is kept (programmed) HIGH for at least the duration of one av<x>_aclk_rx period. Failure to do so may cause the application interface of this module to be improperly reset (or not reset at all).

14.2.12 Power Management

Power Management of the PNX8526 is addressed in Chapter 5 Clock Reset and Power Management.

14.2.13 IEEE 1394 Physical Configuration Register

The IEEE 1394 link supports both IEEE 1394-1995, as well as IEEE 1394a-2000 compliant physical layer ICs. Selection of which type is defined in the C1394_PHY field of the Global 2 registers. This register is shown on page 14-357 of this chapter.

14.3 Register Descriptions

The base address for the PNX8526 IEEE 1394 module is 0x04 9000.

14.3.1 Register Address Map

Table 36: IEEE 1394 Register Summary

Offset	Name	Description
0x04 9000	IDREG	ID register
0x04 9004	LNKCTL	General Link Control
0x04 9008	LNKPHYINTACK	Link/PHY Interrupt Acknowledge
0x04 900C	LNKPHYINTE	Link/PHY Interrupt Enable
0x04 9010	CYCTM	Cycle Timer Register
0x04 9014	PHYACS	PHY Register Access
0x04 9018	GLOBCSR	Global Interrupt Status and TX Control
0x04 901C	TIMER	Timer
0x04 9020	ITXn_PKCTL	Isochronous Transmit Packing Control and Status
0x04 9024	ITXn_HQ1	Common Isochronous Transmit Packet Header Quadlet 1
0x04 9028	ITXn_HQ2	Common Isochronous Transmit Packet Header Quadlet 2
0x04 902C	ITXn_INTACK	Isochronous Transmitter Interrupt Acknowledge
0x04 9030	ITXn_INTE	Isochronous Transmitter Interrupt Enable
0x04 9034	ITXn_CTL	Isochronous Transmitter Control Register
0x04 9038	ITXn_MEM	Isochronous Transmitter Memory Status
0x04 9040	IRXn_PKCTL	Isochronous Receiver UnPacking Control and Status
0x04 9044	IRXn_HQ1	Common Isochronous Receiver Packet Header Quadlet 1
0x04 9048	IRXn_HQ2	Common Isochronous Receiver Packet Header Quadlet 2
0x04 904C	IRXn_INTACK	Isochronous Receiver Interrupt Acknowledge

Table 36: IEEE 1394 Register Summary *…Continued*

Offset	Name	Description
0x04 9050	IRXn_INTE	Isochronous Receiver Interrupt Enable
0x04 9054	IRXn_CTL	Isochronous Receiver Control Register
0x04 9058	IRXn_MEM	Isochronous Receiver Memory Status
0x04 9080	ASYCTL	Asynchronous RX/TX Control
0x04 9084	ASYMEM	Asynchronous RX/TX Memory Status
0x04 9088	TX_RQ_NEXT	Asynchronous Transmit Request Next
0x04 908C	TX_RQ_LAST	Asynchronous Transmit Request Last
0x04 9090	TX_RP_NEXT	Asynchronous Transmit Response Next
0x04 9094	TX_RP_LAST	Asynchronous Transmit Response Last
0x04 9098	RREQ	Asynchronous Receive Request
0x04 909C	RRSP	Asynchronous Receive Response
0x04 90A0	ASYINTACK	Asynchronous RX/TX Interrupt Acknowledge
0x04 90A4	ASYINTE	Asynchronous RX/TX Interrupt Enable
0x04 9100	RRSPSIZE	Asynchronous Receive Response Fifo Size
0x04 9104	RREQSIZE	Asynchronous Receive Request Fifo Size
0x04 9110	TRSPSIZE	Asynchronous Transmit Response Fifo Size
0x04 9114	TREQSIZE	Asynchronous Transmit Request Fifo Size
0x04 9120	IRX0_SIZE	Isochronous Receiver Fifo Size
0x04 9130	ITX0_SIZE	Isochronous Transmitter Fifo Size
0x04 9134	ITX1_SIZE	Isochronous Transmitter Fifo Size
0x04 9138—91FC	Reserved	
0x04 9200	ITX0_CNMKE1	Even “No more copies” encryption key 1
0x04 9204	ITX0_CNMKE2	Even “No more copies” encryption key 2
0x04 9208	ITX0_COGKE1	Even “Copy one generation” encryption key 1
0x04 920C	ITX0_COGKE2	Even “Copy one generation” encryption key 2
0x04 9210	ITX0_CNKE1	Even “Copy never” encryption key 1
0x04 9214	ITX0_CNKE2	Even “Copy never” encryption key 2
0x04 9218	ITX0_CNMKO1	Odd “No more copies” encryption key 1
0x04 921C	ITX0_CNMKO2	Odd “No more copies” encryption key 2
0x04 9220	ITX0_COGKO1	Odd “Copy one generation” encryption key 1
0x04 9224	ITX0_COGKO2	Odd “Copy one generation” encryption key 2
0x04 9228	ITX0_CNKO1	Odd “Copy never” encryption key 1
0x04 922C	ITX0_CNKO2	Odd “Copy never” encryption key 2
0x04 9240	ITX1_CNMKE1	Even “No more copies” encryption key 1
0x04 9244	ITX1_CNMKE2	Even “No more copies” encryption key 2
0x04 9248	ITX1_COGKE1	Even “Copy one generation” encryption key 1
0x04 924C	ITX1_COGKE2	Even “Copy one generation” encryption key 2
0x04 9250	ITX1_CNKE1	Even “Copy never” encryption key 1

Table 36: IEEE 1394 Register Summary *…Continued*

Offset	Name	Description
0x04 9254	ITX1_CNKE2	Even “Copy never” encryption key 2
0x04 9258	ITX1_CNMKO1	Odd “No more copies” encryption key 1
0x04 925C	ITX1_CNMKO2	Odd “No more copies” encryption key 2
0x04 9260	ITX1_COGKO1	Odd “Copy one generation” encryption key 1
0x04 9264	ITX1_COGKO2	Odd “Copy one generation” encryption key 2
0x04 9268	ITX1_CNKO1	Odd “Copy never” encryption key 1
0x04 926C	ITX1_CNKO2	Odd “Copy never” encryption key 2
0x04 9300	IRX0_CNMKE1	Even “No more copies” encryption key 1
0x04 9304	IRX0_CNMKE2	Even “No more copies” encryption key 2
0x04 9308	IRX0_COGKE1	Even “Copy one generation” encryption key 1
0x04 930C	IRX0_COGKE2	Even “Copy one generation” encryption key 2
0x04 9310	IRX0_CNKE1	Even “Copy never” encryption key 1
0x04 9314	IRX0_CNKE2	Even “Copy never” encryption key 2
0x04 9318	IRX0_CNMKO1	Odd “No more copies” encryption key 1
0x04 931C	IRX0_CNMKO2	Odd “No more copies” encryption key 2
0x04 9320	IRX0_COGKO1	Odd “Copy one generation” encryption key 1
0x04 9324	IRX0_COGKO2	Odd “Copy one generation” encryption key 2
0x04 9328	IRX0_CNKO1	Odd “Copy never” encryption key 1
0x04 932C	IRX0_CNKO2	Odd “Copy never” encryption key 2
0x04 9FFC	MODULE_ID	Module ID

Table 37: EEE 1394 Global2 Register Summary

Offset	Name	Description
0x04 D060	C1394_PHY	Global 2 register

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Global Control and Status Registers				
Offset 0x04 9000			*ID register (IDREG)*	
31:22	R/W	3FF	BUS_ID	The 10-bit bus number that is used with the Node ID in the source address for outgoing packets and used to accept or reject incoming packets.
21:16	R/W	3F	NODE_ID	Used in conjunction with BUS ID in the source address for outgoing packets. Used to accept or reject incoming packets. Automatically updated with the node ID assigned after the IEEE 1394 bus Tree-ID sequence.
15:8	R	03	PART_CODE	0x03 = VIPER

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7:0	R	01	VERSION_CODE	Shows the revision of the IEEE 1394_AVLINK_CORE.
Offset 0x04 9004			***General Link Control (LNKCTL)***	
31	R/W	0	IDVALID	When set to one, the IEEE 1394_AVLINK_CORE accepts the packets addressed to this node. Automatically set after self-ID is completed and node ID is updated.
30	R/W	1	RCVSELFID	When set to one, the self-ID packets generated by each PHY device on the bus during bus initialization are received and placed into the asynchronous request queue as a single packet. Also enables reception of PHY configuration packets in the asynchronous request queue.
29:27	R/W	0	BSYCTRL	These bits control what busy status the chip returns to incoming packets. 000 = Use protocol requested by received packet (either dual phase or single phase 001 = Send busyA when it is necessary to send a busy acknowledge (testing/diagnostics) 010 = Send busyB when it is necessary to send a busy acknowledge (testing/diagnostics) 011 = Use single phase retry protocol 100 = Use protocol requested in packet, always send a busy ack for all packets 101 = Busy A all incoming packets 110 = Busy B all incoming packets 111 = Use single phase retry protocol, always send a busy ack.
26	R/W	1	TxENABLE	Set to logic 1, the link layer transmitter will arbitrate and send bus packets.
25	R/W	1	RxENABLE	Set to logic 1, link layer transmitter will receive & respond to bus packets.
24:22		-	Unused	Bits read 0.
21	R/W	0	RSTTx	Reset the link layer transmitter.
20	R/W	0	RSTRx	Reset the link layer receiver.
19		-	Unused	Bits read 0.
18	R/W	0	RPL	Reset the PHY-LINK interface according to IEEE 1394.a requirements. This bit automatically resets to logic 0 after completion of the reset.
17:13		-	Unused	Bits read 0
12	R/W	0	STRICTISOCH	When set to logic 1, packets that are sent outside of specified isochronous cycles are rejected.
11	R/W	0	CYMASTER	When set and LNKCTL.ROOT = 1 and CYCTIM.CYCLENUMBER increments, the transmitter sends a cycle-start packet. The cycle master function will be disabled if a cycle timeout is detected (LNKPHYINTACK. CYTMOUT = 1). To restart the cycle master function in such a case, first reset CYMASTER, then set it again.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
10	R/W	0	CYSOURCE	1 = The CYCTM.CYCLENUMBER increments and CYCTM.CYCLE_ OFFSET resets for each positive transition of the cyclein port. 0 = The CYCTM.CYCLENUMBER increments when the CYCTM.CYCLE_ OFFSET rolls over.
9	R/W	0	CYTIMREN	Activates the CYCTM register when set.
8:7		-	Unused	
6	R/W	1	TxRDY	The transmitter is idle and ready.
5	R	0	ROOT	When set, this node is the root on the IEEE 1394 bus. This bit is automatically updated after the self-ID phase.
4	R	0	BUSYFLAG	0 = Busy A will send once acknowledge is required. 1 = Busy B will send once acknowledge is required.
3:0	R	0	ATACK	At Acknowledge received. The last acknowledge received by the transmitter in response to a packet sent from the transmit-FIFO interface while ATF is selected (diagnostic purposes).
Offset 0x04 9008			*Link/PHY Interrupt Acknowledge (LNKPHYINTACK)*	
31:22		-	Unused	
21	R/W	0	CADONE	Interrupt from external source (ECA in case of L41)
20	R/W	0	TIMER	Interrupt when timer has counted to 0.
19		-	Unused	
18	R/W	0	CMDRST	Command Reset: a write request to RESET-START has been received.
17	R/W	0	FAIRGAP	The serial bus has been idle for a fair gap (called subaction gap in the IEEE 1394 Specification, Ref [3].
16	R/W	0	ARBGAP	The serial bus has been idle for an arbitration gap.
15	R/W	0	PHYINT	The PHY chip has signaled an interrupt through the PHY interface after a bus reset of PHY reset. Activates for any of the following reasons: (1) PHY has detected a loop on the bus. (2) Cable power has fallen below the minimum voltage. (3) PHY arbitration state machine has timed out, indicating a bus loop. (4) Bus cable has been disconnected. Recognition and notification of any of the above events by the PHY requires between 166 and 500 ms; so this bit is not instantaneously set.
14	R/W	0	PHYRRX	PHY register information received; register data has been transferred by the PHY into the Link.
13	R/W	0	PHYRST	PHY reset started. A PHY-layer reconfiguration has started. This interrupt clears the ID valid bit. It is called a Bus Reset in IEEE 1394 Specification, Ref [3]. The asynchronous queues will be flushed during a bus reset.
12:11		-	Unused	

IEEE 1394 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
10	R/W	0	ITBADFMT	Isochronous Transmitter is stuck. The transmitter has detected invalid data at the transmit FIFO interface when the ITF is selected. Reset the isochronous transmitter to clear.
9	R/W	0	ATBADFMT	Asynchronous Transmitter is stuck. The transmitter expected the start of a new asynchronous packet in the FIFO, but found other data (out of sync with user). Reset the asynchronous transmitter to clear.
8	R/W	0	SNTREJ	Busy acknowledge sent by receiver. The receiver was forced to send a busy acknowledge to a packet in this node because the receiver response/request FIFO overflowed.
7	R/W	0	HDRERR	Header error. The receiver detected a header CRC error on an incoming packet that may have been addressed to this node.
6	R/W	0	TCERR	Transaction Code Error. The transmitter detected an invalid transaction code in the data at the transmit FIFO interface.
5	R/W	0	CYTMOUT	Cycle timed out. An isochronous cycle lasted more than 125 ms from cycle-start to fair gap. This disables the cycle master function.
4	R/W	0	CYSEC	Cycle second incremented. The CYCTM.CYCLE_SECONDS field has been incremented.
3	R/W	0	CYSTART	Cycle started. The transmitter has sent or the receiver has received a cycle start packet.
2	R/W	0	CYDONE	Cycle done. A fair gap has been detected on the bus after the transmission or reception of a cycle start packet. This indicates that the isochronous cycle is over.
1	R/W	0	CYPEND	Cycle pending is asserted when cycle timer offset is set to zero (rolled over or reset) and stays asserted until the isochronous cycle has ended.
0	R/W	0	CYLOST	Cycle lost. The cycle timer has rolled over twice without the reception of a cycle start packet. This only occurs when cycle master is not asserted.
Offset 0x04 900C			***Link/PHY Interrupt Enable (LNKPHYINTE)***	
Reference Offset 0x04 9008 Link/PHY Interrupt Acknowledge (LNKPHYINTACK) for more details.				
31:22		-	Unused	
21	R/W	0	EECA_INT	Enable interrupt from external source.
20	R/W	0	ETIMER	Enable interrupt when Timer has counted to 0.
19		-	Unused	
18	R/W	0	ECMDRST	Enable interrupt Command Reset Received.
17	R/W	0	EFAIRGAP	Enable interrupt to the serial bus that has been idle for a Fair Gap.
16	R/W	0	EARBGAP	Enable interrupt to the serial bus that has been idle for an Arbitration Gap.
15	R/W	0	EPHYINT	Enable interrupt to the PHY chip that has signaled an interrupt through the PHY interface after a bus reset of PHY reset.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
14	R/W	0	EPHYRRX	Enable interrupt to PHY Register information Received.
13	R/W	0	EPHYRST	Enable interrupt to PHY Reset that has started.
12:11		-	Unused	
10	R/W	0	EITBADFMT	Enable interrupt when the Isochronous Transmitter is stuck.
9	R/W	0	EATBADFMT	Enable interrupt when the Asynchronous Transmitter is stuck.
8	R/W	0	ESNTREJ	Enable interrupt to Busy Acknowledge sent by receiver.
7	R/W	0	EHDRERR	Enable interrupt to Header Error.
6	R/W	0	ETCERR	Enable interrupt to Transaction Code Error.
5	R/W	0	ECYTMOUT	Enable interrupt to Cycle Timed Out.
4	R/W	0	ECYSEC	Enable interrupt to Cycle Second incremented.
3	R/W	0	ECYSTART	Enable interrupt to Cycle Started.
2	R/W	0	ECYDONE	Enable interrupt to Cycle Done.
1	R/W	0	ECYPEND	Enable interrupt to Cycle Pending.
0	R/W	0	ECYLOST	Enable interrupt to Cycle Lost.
Offset 0x04 9010			***Cycle Timer Register (CYCTM)***	
31:25	R/W	0	CYCLE_SECONDS	Second counter
24:12	R/W	0	CYCLE_NUMBER	8 kHz bus cycle counter
11:0	R/W	0	CYCLE_OFFSET	24.576 MHz cycle counter
Offset 0x04 9014			***PHY Register Access (PHYACS)***	
31	R/W	0	RDPHY	When set, a read register request with accompanying address (RGPHYAD) is sent to the PHY. Bit is cleared after the request is sent.
30	R/W	0	WRPHY	When set, a write register request with accompanying address (RGPHYAD) and data (RGPHYDATA) is sent to the PHY. This bit is cleared after the request is sent.
29:28		-	Unused	
27:24	R/W	0	PHYRGAD	Address of the PHY register that is being accessed
23:16	R/W	0	PHYRGDATA	Data to be written to the PHY register PHYRGAD
15:12		-	Unused	
11:8	R	0	PHYRXAD	Address of register from which data (PHYRXDATA) was read
7:0	R	0	PHYRXDATA	Data as read from PHY register indicated in PHYRXAD
Offset 0x04 9018			***Global Interrupt Status and TX Control (GLOBCSR)***	
31:19		-	Unused	
20	R/W	0	SFTRST	1 = Generates a reset of the IEEE 1394 module, resetting all registers to their reset value. Automatically cleared after 2 cycles of CLK_1394. 0 = No effect
19		-	Unused	
18	R/W	0	ENOUTAV2	Controls the value of port enoutav2.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17	R/W	0	ENOUTAV1	Controls the value of port enoutav1.
16	R/W	1	DIRAV1	Controls the value of port dirav1.
15		-	Unused	
14	R/W	0	EITX1INT (see note opposite)	1 = Enables interrupt bit GLOBCSR.ITX1INT. 0 = Disables interrupt bit GLOBCSR.ITX1INT. Note: Bits EITX1INT and ITX1INT are only present when IEEE 1394_AVLINK _ CORE has been ordered with two isochronous transmitters.
13:12		-	Unused	
11	R/W	0	EASYTX/RX	1 = Enables interrupt bit GLOBCSR.ASYTX/RX. 0 = Disables interrupt bit GLOBCSR.ASYTX/RX.
10	R/W	0	EITX0INT	1 = Enables interrupt bit GLOBCSR.ITX0INT. 0 = Disables interrupt bit GLOBCSR.ITX0INT.
9	R/W	0	EIRX0INT	1 = Enables interrupt bit GLOBCSR.IRX0INT. 0 = Disables interrupt bit GLOBCSR.IRX0INT.
8	R/W	0	ELNKPHYINT	1 = Enables interrupt bit GLOBCSR.LNKPHYINT. 0 = Disables interrupt bit GLOBCSR.LNKPHYINT.
7		-	Unused	
6	R	0	ITX1INT (see Note)	1 = Signals that an interrupt is pending in the ITX1_INT register. 0 = No interrupt in ITX1_INT register is set.
5:4		-	Unused	
3	R	0	ASYTX/RX	1 = Signals that an interrupt is pending in the ASYINT register. 0 = No interrupt in ASYINT register is set.
2	R	0	ITX0INT	1 = Signals that an interrupt is pending in the ITX0_INT register. 0 = No interrupt in ITX0_INT register is set.
1	R	0	IRX0INT	1 = Signals that an interrupt is pending in the IRX0_INT register . 0 = No interrupt in IRX0_INT register is set.
0	R	0	LNKPHYINT	1 = Signals that an interrupt is pending in the LNKPHYINTACK register. 0 = No interrupt in LNKPHYINTACK register is set.
Offset 0x04 901C			*Timer (TIMER)*	
31	R/W	0	TMGOSTOP	1 = Start timer. This bit will be automatically reset to 0. 0 = No effect
30	R/W	0	TMCONT	1 = Continuously operate timer. 0 = Operate timer for one timing cycle, then stop.
29	R/W	0	TMBRE	Timer bus reset enable 1 = Start the timer at the beginning of a bus reset. 0 = Start the timer upon setting the TIMER.TMGOSTOP bit.
28:24		-	Unused	
23:0	R/W	0	PRELOAD	Timer preload value

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 9020			*Isochronous Transmit Packing Control and Status (ITXn_PKCTL)*	
31		-	Unused	
30	R/W	0	MLAN	mLAN mode bit 0 = Normal SYT timestamping operation is assumed. 1 = Sends SYT stamps with data payloads only. Additionally, the SYT value will automatically be incremented by two additional cycles.
29:28	R/W	0	TXAP_CLK	This field controls the frequency and mode of the isochronous transmitter application clock at the AV interface. 00 = An external application clock should be provided, itx_clkouten is de-asserted. 01 = Output frequency at irx_clkout = 24.576 MHz, itx_clkouten is asserted. 10 = Output frequency at irx_clkout = 12.288 MHz, itx_clkouten is asserted. 11 = Output frequency at irx_clkout = 6.144 MHz, itx_clkouten is asserted.
27:16	R/W	0	TRDEL	Transport delay. Value added to cycle timer to produce timestamps. Lower 4 bits add to upper 4 bits of cycle_offset, (Cycle Timer Register, CYCTM). Remainder adds to cycle_count field.
15:8	R/W	0	MAXBL	The (maximum) number of data blocks to be put in a payload.
7	R/W	0	ENXTMPSTP	Enable External Timestamp control. Allows an external timestamp (generated by the application) to be inserted in place of the link generated time-stamp. Defaults to link generated timestamp.
6:5	R/W	0	SYT_DELAY	Programmable delay of AV1FSYNC and AV2FSYNC. Each cycle is 1 bus cycle, 125 ms. Reset value is ‘00’, a 3 cycle delay. 01 = 2 cycles 00 = 3 cycles 10 = 4 cycles 11 = Reserved
4	R/W	0	EN_ITX	Enables receipt of new application packets and generation of isochronous bus packets in every cycle. Also enables the Link Layer to arbitrate for the transmitter in each subsequent bus cycle. When this bit is disabled (0), the current packet will be transmitted and then the transmitter will shut down.
3:2	R/W	0	PM	Packing Mode. 00 = Variable sized bus packets, most generic mode 01 = Fixed size bus packets 10 = MPEG-2 packing mode 11 = No data, just CIP headers are transmitted.
1	R/W	0	EN_FS	Enables generation/insertion of SYT stamps (Timestamps) in CIP header.

IEEE 1394 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
0	R/W	0	RST_ITX	Setting this bit (RST_ITX) to ‘1’ resets the transmitter. For synchronous reset of ITX to work properly, an itx_clkin must be present to ensure the reset bit is kept (programmed) HIGH for at least the duration of one itx_clkin period. Failure to do so may cause the application interface of this module to be improperly reset (or not reset at all). When reset is enabled, all bytes will be flushed from the FIFO and transmission will cease.
Offset 0x04 9024			***Common Isochronous Transmit Packet Header Quadlet 1(ITXn_HQ1)***	
31:24		-	Unused	
23:16	R/W	0	DBS	Size of data blocks in quadlets, from which AV payload is constructed (0 = 256 quadlets).
15:14	R/W	0	FN	Fraction Number. The encoding for the number of data blocks into which each source packet shall be divided. 00 = 1 datablock 01 = 2 datablocks 10 = 4 datablocks 11 = 8 datablocks
13:11	R/W	0	QPC	Number of zero filled dummy quadlets to append to each source packet before it is divided into data blocks of the specified sized. The value QPC must be less than DBS and less than 2^{FN}.
10	R/W	0	SPH	Indicates that a 25-bit CYCTM based timestamp has to be inserted before each application packet.
9:0		-	Unused	
Offset 0x04 9028			***Common Isochronous Transmit Packet Header Quadlet 2 (ITXn_HQ2)***	
31:30		-	Unused	Bits read 0.
29:24	R/W	0	FMT	Value to be inserted in the FMT field in the AV header.
23:0	R/W	0	FDF/SYT	Value to be inserted in the FDF field. When the EN_FS bit in the Isochronous Packing Transmit Control and Status register (ITX_n_CTL) is set (=1), the lower 16 bits of this register are replaced by an SYT stamp if a rising edge on av<x>_fsync_in has been detected or all `1s’ if no such edge was detected since the previous packet. The upper 8 bits are sent as they appear in the FDF register. When the EN_FS bit in the Transmit Control and Status register = 0, the full 24 bits can be set to any application specified value.
Offset 0x04 902C			***Isochronous Transmitter Interrupt Acknowledge (ITXn_INTACK)***	
31:15		-	Unused	Bits read 0.
14	R/W	0	SYERR	An overflow occurred at the buffer that stores events at emi, odd/ even or itx_sysync. As a result, the content of the SY field in the isochronous packet header can not be properly synchronized with the isochronous payload. Recommended action is to reset the isochronous transmitter. The internal event buffer can store up to 3 events for the current FIFO content. An event is a state change for one of the mentioned inputs.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13	R/W	0	ODDEVN	A packet with an updated odd/even key has just been transmitted.
12	R/W	0	EMI	A packet with updated emi bits has just been transmitted.
11:10		-	Unused	Bits read 0.
9	R/W	0	ITX100LFT	Set when the number of empty quadlets in the itx FIFO <= 100. This interrupt will be disabled when the FIFO size <= 100 Quadlets.
8	R/W	0	ITX256LFT	Set when the number of empty quadlets in the itx FIFO <= 256. This interrupt will be disabled when the FIFO size <= 256 Quadlets.
7	R/W	0	ITX512LFT	Set when number of empty quadlets in the itx FIFO <= 512. This interrupt will be disabled when the size of the FIFO size <= 512 Quadlets.
6	R/W	0	TRMSYT	Interrupt on transmission of a SYT in CIP header quadlet 2
5	R/W	0	TRMBP	Interrupt on payload transmission/discard complete
4	R/W	0	DBCERR	Acknowledge interrupt on Data Block Count (DBC) synchronization loss.
3	R/W	0	INPERR	Acknowledge interrupt on input error (input data discarded).
2	R/W	0	DISCARD	Interrupt on lost cycle (payload discarded).
1	R/W	0	ITXFULL	Set when the itx FIFO is full, the number of empty quadlets = 0. This is a fatal error. It is recommended to reset and re-initialize the transmitter.
0	R/W	0	ITXEMPTY	Set when the number of quadlets in the itx FIFO containing valid data = 0
Offset 0x04 9030			***Isochronous Transmitter Interrupt Enable (ITXn_INTE)***	
Reference Offset 0x04 9008 Link/PHY Interrupt Acknowledge ITXn_INTACK for more details.				
31:15		-	Unused	
14	R/W	0	ESYERR	Enable interrupt when an overflow occurred at the buffer that stores events at emi, oddeven or itx_sysync.
13	R/W	0	EODDEVN	Enable Interrupt when a packet with an updated odd/even key has just been transmitted.
12	R/W	0	EEMI	Enable Interrupt when a packet with updated emi bits has just been transmitted.
11:10		-	Unused	
9	R/W	0	EITX100LFT	Enables Interrupt when number of empty quadlets in the itx FIFO <= 100.
8	R/W	0	EITX256LFT	Enables Interrupt when number of empty quadlets in the itx FIFO <= 256.
7	R/W	0	EITX512LFT	Enables Interrupt when number of empty quadlets in the itx FIFO <= 512.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
6	R/W	0	ETRMSYT	Enables Interrupt on transmission of a SYT in CIP header quadlet 2.
5	R/W	0	ETRMBP	Enables Interrupt on payload transmission/discard complete.
4	R/W	0	EDBCERR	Enables interrupt on Data Block Count (DBC) synchronization loss.
3	R/W	0	EINPERR	Enables interrupt on input error (input data discarded).
2	R/W	0	EDISCARD	Enables Interrupt on lost cycle (payload discarded).
1	R/W	0	EITXFULL	Enables Interrupt when itx FIFO is full, the number of empty quadlets = 0.
0	R/W	0	EITXEMPTY	Enables Interrupt when the number of quadlets in the itx FIFO containing valid data = 0
Offset 0x04 9034			*Isochronous Transmitter Control Register (ITXn_CTL)*	
31:21		-	Unused	
20	R/W	0	CPHREN	Enable encryption cypher 0 = Application packets will not be encrypted. 1 = Application packets will be encrypted by the embedded stream cipher which operates according to the 5C content protection protocol. The choice of encryption key is determined by the status of the ITX_n_CTL.EMI and the ITX_n_CTL.ODDEVEN fields.
19	R/W	0	EMIPE	EMI Pin Enable 0 = Use EMI bits from this register ITX_n_CTL.EMI. 1 = Use EMI value from ports itx_emi.
18:16		-	Unused	
15:14	R/W	0	TAG	Tag code to be inserted into the isochronous bus packet header 01 = IEC 61883 international standard data
13:8	R/W	0	CHANNEL	Isochronous channel number
7		-	Unused	
6		0	Reserved	This bit is reserved for future extension of the SPD field according to IEEE 1394.a definition.
5:4	R/W	0	SPD	Cable transmission speed (S100, S200, S400) =00, 100MB/s =01, 200MB/s =10, 400MB/s =11, reserved.
3:2	R/W	0	EMI[1..0]	The EMI field specifies the encryption mode for the data stream. Functionality of this field depends on the status of the ITX_n_CTL. CPHREN bit and the ITX_n_CTL.EMIPE bit. If ITX_n_CTL.CPHREN is logic 0 then this field is R/W and its value will be transmitted into SY[3:2] of the Isochronous packet header. If ITX_n_CTL.CPHREN is logic 1 and ITX_n_CTL.EMIPE is logic 1 then this field is read only and reads the current status of port itx_emi. If ITX_n_CTL.CPHREN is logic 1 and ITX_n_CTL.EMIPE is logic 0 then this field is R/ W.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
1	R/W	0	ODDEVEN	Functionality of the ODDEVEN bit depends on the status of ITX_n_CTL.CPHREN. If ITX_n_CTL.CPHREN is logic 0 then the value of this bit will be transmitted into SY[1] of the Isochronous packet header. If ITX_n_CTL.CPHREN is logic 1 then toggling this bit will cause the cipher to swap its odd/even key. Upon reading, this bit will reflect the odd/even status of the cipher, not the value that was last written into it.
0	R	0	SYSYNC	The SYSYNC bit reads the value to be inserted in bit 0 of the SY field of the isochronous packet header.
Offset 0x04 9038			***Isochronous Transmitter Memory Status (ITXn_MEM)***	
31:7		-	Unused	
6	R	0	ITXM100LFT	Set when the number of empty quadlets in the itx FIFO<=100. Disabled when the FIFO size <= 128 Quadlets.
5	R	0	ITXM256LFT	Set when the number of empty quadlets in itx FIFO <= 256. Disabled when the FIFO size <= 256 Quadlets.
4	R	0	ITXM512LFT	Set when number of empty quadlets in itx FIFO <= 512. Disabled when the size of the FIFO size <= 512 Quadlets.
3	R	0	ITXMF	Set when the itx FIFO is full, the number of empty quadlets = 0.
2	R	0	ITXMAF	Set when the number of empty quadlets in the itx FIFO = 1.
1	R	1	ITXM5AV	Set when the number of empty quadlets in the itx FIFO >= 5.
0	R	1	ITXME	Set when the number of quadlets in the itx FIFO containing valid data = 0.
Offset 0x04 9040			***Isochronous Receiver UnPacking Control and Status (IRXn_PKCTL)***	
31:30		-	Unused	
29:28	R/W	0	RXAP_CLK	This field controls the frequency and mode of the isochronous receiver application clock at the AV interface. 00 = An external application clock should be provided, irx_clkouten is de-asserted. 01=Output frequency at irx_clkout = 24.576 MHz, irx_clkouten is asserted. 10=Output frequency at irx_clkout = 12.288 MHz, irx_clkouten is asserted. 11=Output frequency at irx_clkout = 6.144 MHz, irx_clkouten is asserted
27:9		-	Unused	
8	R/W	0	SNDIMM	Send application packet Immediately. 1 = A received isochronous application packet containing a CRC error will be output immediately, without regard for the timestamp value. 0 = A received isochronous application packet will be output with respect to the timestamp value.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7	R/W	0	DIS_TSC	Disable Timestamp Checking. 1 = The timestamp accompanying a packet is output before the isochronous application packet for use by the application. This adds an extra quadlet to the received data stream; the application must be capable of handling these 4 extra bytes. 0 = A received isochronous application packet will be output with respect to the timestamp value.
6	R/W	1	RMVUAP	Remove unreliable packets 1 = Received unreliable application packets (CRC error, data length error), which will be removed from the memory. 0 = All received application packets will be delivered.
5	R	0	SPAV	Source packet available for delivery in buffer memory.
4	R/W	0	EN_IRX	Enable receiver operation. Value is only checked whenever a new bus packet arrives, so enable/disable while running is ‘graceful’, meaning any transfers in process will be completed before this bit is asserted.
3:2	R/W	0	BPAD	Byte Padding. The value indicates the amount of byte to be removed from the last data quadlet of each source packet. This is in addition to quadlet padding.
1	R/W	0	EN_FS	Enable processing of SYT stamps (Timestamps) in CIP header.
0	R/W	1	RST_IRX	Setting this bit (RST_IRX) to ‘1’ resets the receiver. In order for synchronous reset of IRX to work properly, irx_clkin must be present to ensure the reset bit is kept (programmed) HIGH for at least the duration of one irx_clkin period. Failure to do so may cause the application interface of this module to be improperly reset (or not reset at all). When a receiver reset is required, first disable the receiver IRX_n_PKCTL.EN_IRX, wait until RX FIFO is emptied, then perform the reset. This allows previously received packets to go to the application instead of being lost.
Offset 0x04 9044			*Common Isochronous Receiver Packet Header Quadlet 1(IRXn_HQ1)*	
31	R	0	E0	End of Header. Always set to 0 for first CIP header quadlet.
30	R	0	F0	Format. Always set to 0 for first CIP header quadlet.
29:24	R	0	SID	Source ID. Contains the node address of the sender of the isochronous data.
23:16	R	0	DBS	Size of data blocks in quadlets from which AV payload is constructed (0 = 256 quadlets)
15:14	R	0	FN	Fraction Number. The encoding for the number of data blocks into which each source packet has been divided: 00 = 1 datablock 01 = 2 datablocks 10 = 4 datablocks 11 = 8 datablocks

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13:11	R	0	QPC	Number of zero filled dummy quadlets appended to each source packet before it was divided into data blocks of the specified sized.
10	R	0	SPH	1 = A 25-bit CYCTM based timestamp is inserted before each application packet. 0 = A 16-bit CYCTM based timestamp is inserted.
9:0		-	Unused	Bits read 0.
Offset 0x04 9048			*Common Isochronous Receiver Packet Header Quadlet 2 (IRXn_HQ2)*	
31	R	0	E1	End of Header. Always set to 1 for the second CIP header quadlet.
30	R	0	F1	Format. Always set to 0 for the second CIP header quadlet.
29:24	R	0	FMT	Value to be inserted in the FMT field in the CIP header.
23:0	R	FFFF	FDF/SYT	When the IRX_n_PKCTL.EN_FS is set (=1), the lower 16 bits of this register are interpreted as SYT.
Offset 0x04 904C			*Isochronous Receiver Interrupt Acknowledge (IRXn_INTACK)*	
31:14		-	Unused	
13	R/W	0	ODDEVN	An odd/even key change has appeared in a bus packet that was received.
12	R/W	0	EMI	Set when the received EMI bit values have changed in a bus packet that was received.
11		-	Unused	
10	R/W	0	IRX100LFT	Set when the number of empty quadlets in the irx FIFO <= 100. This interrupt will be disabled when the FIFO size <= 100 Quadlets.
9	R/W	0	IRX256LFT	Set when the number of empty quadlets in the irx FIFO <= 256. This interrupt will be disabled when the FIFO size <= 256 Quadlets.
8	R/W	0	IRX512LFT	Set when number of empty quadlets in the rtx FIFO <= 512. This interrupt will be disabled when the size of the FIFO size <= 512 Quadlets.
7	R/W	0	IRXFULL	Set when the irx FIFO is full and the number of empty quadlets = 0. This is a fatal error. Recommended action is to reset and re-initialize the receiver.
6	R/W	0	IRXEMPTY	Set when the number of quadlets in the irx FIFO containing valid data = 0.
5	R/W	0	FSYNC	A pulse was generated on the irx_fsync output (SYT expired).
4	R/W	0	SEQERR	A data block sequence error has occurred (DBC value incorrect).
3	R/W	0	CRCERR	An IEEE 1394 bus packet with a CRC error was received.
2	R/W	0	CIPTAGFLT	A Common Isochronous Packet with unknown header format was received (E,F bits).
1	R/W	0	RCVBP	A bus packet was received and processed correctly.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	R/W	0	SQOV	Status queue overflow. This is a fatal error. Recommended action is to reset and reinitialize the receiver.
Offset 0x04 9050			***Isochronous Receiver Interrupt Enable (IRXn_INTE)***	
Reference Offset 0x04 9008 Link/PHY Interrupt Acknowledge IRXn_INTACK for more details.				
31:14		-	Unused	
13	R/W	0	EODDEVN	Enable interrupt when an odd/even key change has appeared in a bus packet that was received.
12	R/W	0	EEMI	Enable interrupt when the received EMI bit values have changed in a bus packet that was received.
11		-	Unused	
10	R/W	0	EIRX100LFT	Enable interrupt when number of empty quadlets in the irx FIFO <= 100.
9	R/W	0	EIRX256LFT	Enable interrupt when number of empty quadlets in the irx FIFO <= 256.
8	R/W	0	EIRX512LFT	Enable interrupt when number of empty quadlets in the rtx FIFO <= 512.
7	R/W	0	EIRXFULL	Enable interrupt when irx FIFO is full, the number of empty quadlets = 0.
6	R/W	0	EIRXEMPTY	Enable interrupt when the number of quadlets in the irx FIFO containing valid data = 0.
5	R/W	0	EFSYNC	Enable interrupt when a pulse was generated on the irx_fsync output (SYT expired).
4	R/W	0	ESEQERR	Enable interrupt when a data block sequence error has occurred (DBC value incorrect).
3	R/W	0	ECRCERR	Enable interrupt when a IEEE 1394 bus packet with a CRC error was received.
2	R/W	0	ECIPTAGFLT	Enable interrupt when a Common Isochronous Packet with unknown header format was received (E,F bits).
1	R/W	0	ERCVBP	Enable interrupt when a bus packet was received and processed correctly.
0	R/W	0	ESQOV	Enable interrupt when status queue overflow occurs.
Offset 0x04 9054			***Isochronous Receiver Control Register (IRXn_CTL)***	
31:21		-	Unused	
20	R/W	0	DCPHREN	Enable encryption decyphering.
19:18		-	Unused	
17:16	R	0	SPD	Cable transmission speed (S100, S200, S400) 00 = 100MB/s 01 = 200MB/s 10 = 400MB/s 11 = reserved
15:14	R/W	0	TAG	Tag code as received in the isochronous bus packet header 01 = IEC 61883 international standard data.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13:8	R/W	0	CHANNEL	Isochronous channel number
7:4	R	0	ERR	Error code for the last receiver isochronous AV packet. 0001 = The node has successfully accepted the bus packet. 1101 = The node could not accept the bus packet because the payload failed the CRC check or because the length of the payload did not match the length that was specified in the isochronous packet header.
3:2	R	0	EMI[1..0]	EMI control bits
1	R	0	ODDEVEN	ODDEVEN control bit
0	R	0	SYSYNC	Bit 0 of the SY field of the isochronous packet header as received.
Offset 0x04 9058			*Isochronous Receiver Memory Status (IRXn_MEM)*	
31:7		-	Unused	
6	R	0	IRXM100LFT	Set when the number of empty quadlets in the itx FIFO <= 100. Disabled when the FIFO size <= 100 Quadlets.
5	R	0	IRXM256LFT	Set when the number of empty quadlets in itx FIFO <= 256. Disabled when the FIFO size <= 256 Quadlets.
4	R	0	IRXM512LFT	Set when number of empty quadlets in itx FIFO <= 512. Disabled when the size of the FIFO size <= 512 Quadlets.
3	R	0	IRXMF	Set when the itx FIFO is full, the number of empty quadlets = 0.
2	R	0	IRXMAF	Set when the number of empty quadlets in the itx FIFO = 1.
1	R	1	IRXM5AV	Set when the number of empty quadlets in the itx FIFO >= 5.
0	R	1	IRXME	Set when the number of quadlets in the itx FIFO containing valid data = 0.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Asynchronous Control and Status Interface Registers				
Offset 0x04 9080			***Asynchronous RX/TX Control (ASYCTL)***	
31	R/W	0	EN_LDTR	Controls how the transfer registers are being updated with new data from receive request queue and the receive response queue. In order to prevent speculative read actions by a CPU from destroying the FIFO contents, this bit should always be set to 1. 1 = The interrupt bit ASYINTACK.RRSPQQAV or ASYINTACK.RREQQQAV has to be cleared in order to load the latest data from the related queue into the transfer register. In this mode speculative read actions will not cause any harm to the contents of the FIFO. 0 = Reading a transfer register will cause the transfer register to immediately load the latest data from the related receive queue.
30:24		-	Unused	Bits read 0.
23	R/W	0	DIS_BCAST	Disable the reception of broadcast packets.
22	R/W	0	ARXRST	Asynchronous receiver reset. This bit will auto clear when the link layer state machine is idle.
21	R/W	1	ATXRST	Asynchronous transmitter reset. After every bus reset this bit is set to logic 1. This effectively disables the asynchronous transmitter. Re-enable the asynchronous transmitter by clearing this bit after each bus reset, especially if asynchronous transmission is used.
20	R/W	1	ARXALL	Receive and filter only RESPONSE packets. 1 = All responses are stored in the FIFO. 0 = Only solicited responses are stored in the FIFO.
19:16	R/W	0	MAXRC	Maximum number of asynchronous transmitter single phase retries.
15:13	R/W	0	TOS	Timeout Seconds, integer of 1 second
12:0	R/W	0320	TOF	Timeout Fractions, integer of 1/8000 second. Resets to 0x0320, which is 100 milliseconds.
Offset 0x04 9084			***Asynchronous RX/TX Memory Status (ASYMEM)***	
31:18		-	Unused	Bits read 0.
17	R	1	TRSPQIDLE	Set when the transfer register of the asynchronous transmitter response queue is empty.
16	R	1	TREQQIDLE	Set when the transfer register of the asynchronous transmitter request queue is empty.
15	R	0	RRSPQF	Set when the asynchronous receiver response FIFO is full, the number of empty quadlets = 0.
14	R	0	RRSPQAF	Set when the number of empty quadlets in the asynchronous receiver response FIFO=1.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
13	R	1	RRSPQ5AV	Set when the number of empty quadlets in the asynchronous receiver response FIFO >= 5.
12	R	1	RRSPQE	Set when the number of quadlets in the asynchronous receiver response FIFO containing valid data = 0.
11	R	0	RREQQF	Set when the asynchronous receiver request FIFO is full, the number of empty quadlets = 0.
10	R	0	RREQQAF	Set when the number of empty quadlets in the asynchronous receiver request FIFO = 1.
9	R	1	RREQQ5AV	Set when the number of empty quadlets in the asynchronous receiver request FIFO >= 5.
8	R	1	RREQQE	Set when the number of quadlets in the asynchronous receiver request FIFO containing valid data = 0.
7	R	0	TRSPQF	Set when the asynchronous transmitter response FIFO is full, the number of empty quadlets = 0.
6	R	0	TRSPQAF	Set when the number of empty quadlets in the asynchronous transmitter response FIFO= 1.
5	R	1	TRSPQ5AV	Set when the number of empty quadlets in the asynchronous transmitter response FIFO >= 5.
4	R	1	TRSPQE	Set when the number of quadlets in the asynchronous transmitter response FIFO containing valid data = 0.
3	R	0	TREQQF	Set when the asynchronous transmitter request FIFO is full, the number of empty quadlets = 0.
2	R	0	TREQQAF	Set when the number of empty quadlets in the asynchronous transmitter request FIFO = 1.
1	R	1	TREQQ5AV	Set when the number of empty quadlets in the asynchronous transmitter request FIFO >= 5.
0	R	1	TREQQE	Set when the number of quadlets in the asynchronous transmitter request FIFO containing valid data = 0.
Offset 0x04 9088			*Asynchronous Transmit Request Next (TX_RQ_NEXT)*	
31:0	W	0	TX_RQ_NEXT	First/Middle quadlet of packet for asynchronous transmitter request queue. Writing this register will clear the ASYINTACK.TRREQQWR flag until the quadlet has been written into its FIFO.
Offset 0x04 908C			*Asynchronous Transmit Request Last (TX_RQ_LAST)*	
31:0	W	0	TX_RQ_LAST	Last quadlet of packet for asynchronous transmitter request queue. Writing this register will clear the ASYINTACK.TRREQQWR flag until the quadlet has been written into its FIFO.
Offset 0x04 9090			*Asynchronous Transmit Response Next (TX_RP_NEXT)*	
31:0	W	0	TX_RP_NEXT	First/Middle quadlet of packet for asynchronous transmitter response queue. Writing this register will clear the ASYINTACK. TRRSPQWR flag until the quadlet has been written into its FIFO.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 9094			***Asynchronous Transmit Response Last (TX_RP_LAST)***	
31:0	W	0	TX_RP_LAST	Last quadlet of packet for asynchronous transmitter response queue. Writing this register will clear the ASYINTACK. TRRSPQWR flag until the quadlet has been written into its FIFO.
Offset 0x04 9098			***Asynchronous Receive Request (RREQ)***	
31:0	R	0	RREQ	Quadlet of packet from asynchronous receiver request queue which resides in the transfer register. Reading this register will clear the ASYINTACK.RREQQQAV flag until the next received quadlet is available for reading.
Offset 0x04 909C			***Asynchronous Receive Response (RRSP)***	
31:0	R	0	RREQ	Quadlet of packet from asynchronous receiver response queue which resides in the transfer register. Reading this register will clear the ASYINTACK.RRSPQQAV flag until the next received quadlet is available for reading.
Offset 0x04 90A0			***Asynchronous RX/TX Interrupt Acknowledge (ASYINTACK)***	
31:17		-	Unused	Bits read 0.
16	R/W	0	RRSPQFULL	Asynchronous receiver response queue has become full.
15	R/W	0	RREQQFULL	Asynchronous receiver request queue has become full.
14	R/W	0	SIDQAV	Current quadlet in the asynchronous receiver request queue is Self-ID data. This bit is set only after a bus reset, not after reception of PHY packets other than self-IDs. This interrupt automatically resets when the quadlet is read.
13	R/W	0	RRSPQLASTQ	Current quadlet in asynchronous receiver response queue is the last quadlet of a bus packet.
12	R/W	0	RREQQLASTQ	Current quadlet in asynchronous receiver request queue is the last quadlet of a bus packet.
11	R/W	1	RRSPQRDERR	Asynchronous receiver response queue read error (transfer error) or bus reset occurred. When set to logic 1, this queue is blocked for read access.
10	R/W	1	RREQQRDERR	Asynchronous receiver request queue read error (transfer error) or bus reset occurred. When set to logic 1, this queue is blocked for read access.
9	R/W	0	RRSPQQAV	Asynchronous receiver response queue quadlet is available.
8	R/W	0	RREQQQAV	Asynchronous receiver request queue quadlet is available.
7	R/W	0	TIMEOUT	Split transaction response timeout
6	R/W	0	RCVDRSP	Solicited response received (within timeout interval).
5	R/W	0	TRSPQFULL	Asynchronous transmitter response queue has become full.
4	R/W	0	TREQQFULL	Asynchronous transmitter request queue has become full.
3	R/W	0	TRSPQWRERR	Asynchronous transmitter response queue write error (transfer error)
2	R/W	0	TREQQWRERR	Asynchronous transmitter request queue write error (transfer error)

IEEE 1394 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
1	R/W	0	TRSPQWR	Asynchronous transmitter response queue written (transfer register emptied).
0	R/W	0	TREQQWR	Asynchronous transmitter request queue written (transfer register emptied).
Offset 0x04 90A4			***Asynchronous RX/TX Interrupt Enable (ASYINTE)***	
Reference Offset 0x04 9008 Link/PHY Interrupt Acknowledge ASYINTACK for more details.				
31:17		-	Unused	
16	R/W	0	ERRSPQFULL	Enable interrupt when the asynchronous receiver response queue has become full.
15	R/W	0	ERREQQFULL	Enable interrupt when the asynchronous receiver request queue has become full.
14	R/W	0	ESIDQAV	Enable interrupt when the current quadlet in the asynchronous receiver request queue is SelfID data.
13	R/W	0	ERRSPQLASTQ	Enable interrupt when the current quadlet in asynchronous receiver response queue is the last quadlet of a bus packet.
12	R/W	0	ERREQQLASTQ	Enable interrupt when the current quadlet in asynchronous receiver request queue is the last quadlet of a bus packet.
11	R/W	1	ERRSPQRDERR	Enable interrupt when the asynchronous receiver response queue read error (transfer error) or bus reset occurred.
10	R/W	1	ERREQQRDERR	Enable interrupt when the asynchronous receiver request queue read error (transfer error) or bus reset occurred.
9	R/W	0	ERRSPQQAV	Enable interrupt when the asynchronous receiver response queue quadlet is available.
8	R/W	0	ERREQQQAV	Enable interrupt when the asynchronous receiver request queue quadlet is available.
7	R/W	0	ETIMEOUT	Enable interrupt when split transaction response timeout.
6	R/W	0	ERCVDRSP	Enable interrupt when the solicited response is received (within timeout interval).
5	R/W	0	ETRSPQFULL	Enable interrupt when the asynchronous transmitter response queue has become full.
4	R/W	0	ETREQQFULL	Enable interrupt when the asynchronous transmitter request queue has become full.
3	R/W	0	ETRSPQWRERR	Enable interrupt when the asynchronous transmitter response queue write error (transfer error).
2	R/W	0	ETREQQWRERR	Asynchronous transmitter request queue write error (transfer error).
1	R/W	0	ETRSPQWR	Enable interrupt when the asynchronous transmitter response queue written (transfer register emptied).
0	R/W	0	ETREQQWR	Enable interrupt when the asynchronous transmitter request queue written (transfer register emptied).
Registers for FIFO Size Programming				
Offset 0x04 9100			***Asynchronous Receive Response Fifo Size (RRSPSIZE)***	

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:14		-	Unused	
13:8	R/W	0	base_fifo	Base address of the FIFO
7:6		-	Unused	
5:0	R/W	03	end_fifo	End address of the FIFO
Offset 0x04 9104			***Asynchronous Receive Request Fifo Size (RREQSIZE)***	
31:14		-	Unused	
13:8	R/W	04	base_fifo	Base address of the FIFO
7:6		-	Unused	
5:0	R/W	07	end_fifo	End address of the FIFO
Offset 0x04 9110			***Asynchronous Transmit Response Fifo Size (TRSPSIZE)***	
31:14		-	Unused	
13:8	R/W	08	base_fifo	Base address of the FIFO
7:6		-	Unused	
5:0	R/W	0B	end_fifo	End address of the FIFO
Offset 0x04 9114			***Asynchronous Transmit Request Fifo Size (TREQSIZE)***	
31:14		-	Unused	
13:8	R/W	0C	base_fifo	Base address of the FIFO
7:6		-	Unused	
5:0	R/W	0F	end_fifo	End address of the FIFO
Offset 0x04 9120			***Isochronous Receiver Fifo Size (IRX0_SIZE)***	
31:14		-	Unused	Bits read 0.
13:8	R/W	10	base_fifo	Base address of the FIFO
7:6		-	Unused	Bits read 0.
5:0	R/W	1F	end_fifo	End address of the FIFO
Offset 0x04 9130			***Isochronous Transmitter Fifo Size (ITX0_SIZE)***	
31:14		-	Unused	Bits read 0.
13:8	R/W	20	base_fifo	Base address of the FIFO
7:6		-	Unused	Bits read 0.
5:0	R/W	2F	end_fifo	End address of the FIFO
Offset 0x04 9134			***Isochronous Transmitter Fifo Size (ITX1_SIZE)***	
31:14		-	Unused	Bits read 0.
13:8	R/W	20	base_fifo	Base address of the FIFO
7:6		-	Unused	Bits read 0.
5:0	R/W	2F	end_fifo	End address of the FIFO
Offset 0x04 9138—91FC			***Reserved***	

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Copy No More Even/Odd Key Registers				
Offset 0x04 9200			*ITX0_CNMKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even No More Copies encryption key.
Offset 0x04 9204			*ITX0_CNMKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even No More Copies encryption key.
Offset 0x04 9208			*ITX0_COGKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even Copy One Generation encryption key.
Offset 0x04 920C			*ITX0_COGKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even Copy One Generation encryption key.
Offset 0x04 9210			*ITX0_CNKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even Copy Never encryption key.
Offset 0x04 9214			*ITX0_CNKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even Copy Never encryption key.
Offset 0x04 9218			*ITX0_CNMKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd No More Copies encryption key.
Offset 0x04 921C			*ITX0_CNMKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd No More Copies encryption key.
Offset 0x04 9220			*ITX0_COGKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd Copy One Generation encryption key.
Offset 0x04 9224			*ITX0_COGKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd Copy One Generation encryption key.
Offset 0x04 9228			*ITX0_CNKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd Copy Never encryption key.
Offset 0x04 922C			*ITX0_CNKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd Copy Never encryption key.
Offset 0x04 9240			*ITX1_CNMKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even No More Copies encryption key.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 9244			*ITX1_CNMKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even No More Copies encryption key.
Offset 0x04 9248			*ITX1_COGKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even Copy One Generation encryption key.
Offset 0x04 924C			*ITX1_COGKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even Copy One Generation encryption key.
Offset 0x04 9250			*ITX1_CNKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even Copy Never encryption key.
Offset 0x04 9254			*ITX1_CNKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even Copy Never encryption key.
Offset 0x04 9258			*ITX1_CNMKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	0	Key 1	Holds bits [56:32] of the odd No More Copies encryption key.
Offset 0x04 925C			*ITX1_CNMKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd No More Copies encryption key.
Offset 0x04 9260			*ITX1_COGKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd Copy One Generation encryption key.
Offset 0x04 9264			*ITX1_COGKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd Copy One Generation encryption key.
Offset 0x04 9268			*ITX1_CNKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd Copy Never encryption key.
Offset 0x04 926C			*ITX1_CNKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd Copy Never encryption key.
Offset 0x04 9300			*IRX0_CNMKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even No More Copies encryption key.
Offset 0x04 9304			*IRX0_CNMKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even No More Copies encryption key.
Offset 0x04 9308			*IRX0_COGKE1*	
31:24		-	Unused	Bits read 0.

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even Copy One Generation encryption key.
Offset 0x04 930C			*IRX0_COGKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even Copy One Generation encryption key.
Offset 0x04 9310			*IRX0_CNKE1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the even Copy Never encryption key.
Offset 0x04 9314			*IRX0_CNKE2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the even Copy Never encryption key.
Offset 0x04 9318			*IRX0_CNMKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd No More Copies encryption key.
Offset 0x04 931C			*IRX0_CNMKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd No More Copies encryption key.
Offset 0x04 9320			*IRX0_COGKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd Copy One Generation encryption key.
Offset 0x04 9324			*IRX0_COGKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd Copy One Generation encryption key.
Offset 0x04 9328			*IRX0_CNKO1*	
31:24		-	Unused	Bits read 0.
23:0	R/W	NI	Key 1	Holds bits [56:32] of the odd Copy Never encryption key.
Offset 0x04 932C			*IRX0_CNKO2*	
31:0	R/W	NI	Key 2	Holds bits [31:0] of the odd Copy Never encryption key.
Offset 0x04 9FF4			*POWER_DOWN*	
31	R/W	0	PowerDown	0 = Normal operation of the peripheral. 1 = Module is powered down and the module clock can be removed. Module responds to all reads with DEADABBA (except for read of POWER_DOWN) and all writes with ERR ack (except for writes to POWER_DOWN)
30:0		-	Unused	Bits read 0.
Module ID Register				
Offset 0x04 9FFC			*MODULE_ID*	
31:16	R	0x0101	Module ID	Module ID

IEEE 1394 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:12	R	1	MajRev	Major revision indicates any revisions that break software compatibility of the IEEE 1394 module.
11:8	R	0	MinRev	Minor revision indicates any revisions that maintain software compatibility of the IEEE 1394 module.
7:0	R	0	ApertureSize	Size = 4 kB*(ApertureSize + 1)

14.3.2 Global 2 Registers

GLOBAL 2 REGISTER				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
IEEE 1394 Physical Configuration Register				
Offset 0x04 D060			*C1394_PHY*	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x0	C1394_PHY	IEEE 1394 FireWire™ physical configuration 0 = 1394a PHY 1 = 1394-95 Annex J PHY

Chapter 15: I^{2}C Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

15.1 Introduction

The PNX8526 chip contains two I^{2}C modules that interface with a variety of peripherals: consumer electronic appliances, video image processing equipment, and miniature cards. It supports bi-directional data transfer between master and slave in slow and fast speeds. The I^{2}C bus supports multiple master and slave modes. Arbitration between simultaneously transmitting masters can be maintained without corruption of serial data on the bus. Serial clock synchronization allows devices with different bit rates to communicate via one serial bus, and it can be used as a handshake mechanism to suspend and resume serial transfer. The I^{2}C hardware architecture and software protocol is simple and versatile.

15.2 Functional Description

The on-chip I^{2}C module provides a serial interface that meets the I^{2}C bus specification and supports all transfer modes to and from the I^{2}C bus.
It supports the following functionality:

- Both normal mode and fast modes up to 400 kHz
- 32-bit word access from the CPU, and no buffering is supported.
- Generation of interrupts on I^{2}C state change.
- Four modes of operation: master transmitter and receiver, slave transmitter and receiver
- STO bit and the actual addresses of the Special Function Registers (SFRs).

Remark: The I^{2}C software interface is identical to that of 8XC558 from Philips Semiconductors.

The main features of the I^{2}C-bus are as follows:

- Bi-directional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization, which allows communication between devices with different bit rates

- Using serial clock synchronization as a handshake mechanism to suspend and resume serial transfer
- May be used for test and diagnostic purposes

The I^2C bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. Because more than one master could try to initiate a data transfer at the same time, a collision detection scheme is used to arbitrate between the multiple masters. If two or more masters attempt to transfer information onto the bus, the first to produce a ‘one’ (when the other produces a ‘zero’) will detect the collision and back off transferring information.

The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired-AND connection to the SCL line. Two wires, SDA (serial data) and SCL (serial clock), carry information between the devices
connected to the I^2C bus.

Each device can operate as either a transmitter or receiver, depending on the function of the device. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device addressed by a master is considered a slave.

Generation of clock signals on the I^2C bus is always the responsibility of the master device. Each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding down the clock line or by another master when arbitration occurs.

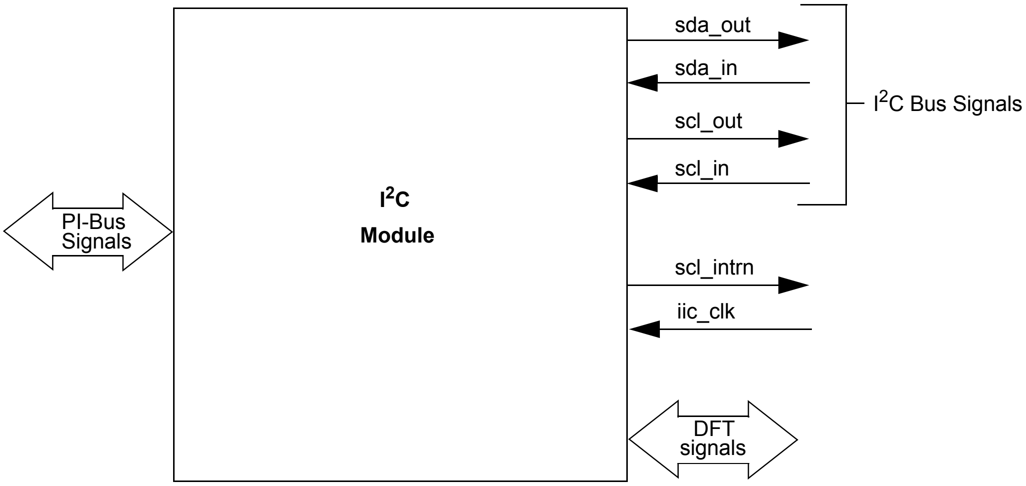


Figure 1: Top Level View of the I^2C Module

15.3 Operation

15.3.1 General Operation

The I^2C module supports a master/slave I^2C bus interface with an integrated shift register, shift timing generation and slave address recognition. It is compliant with the I2C Bus Specification. I^2C standard mode (100 kHz SCL) and fast mode (up to 400 kHz SCL) are supported.

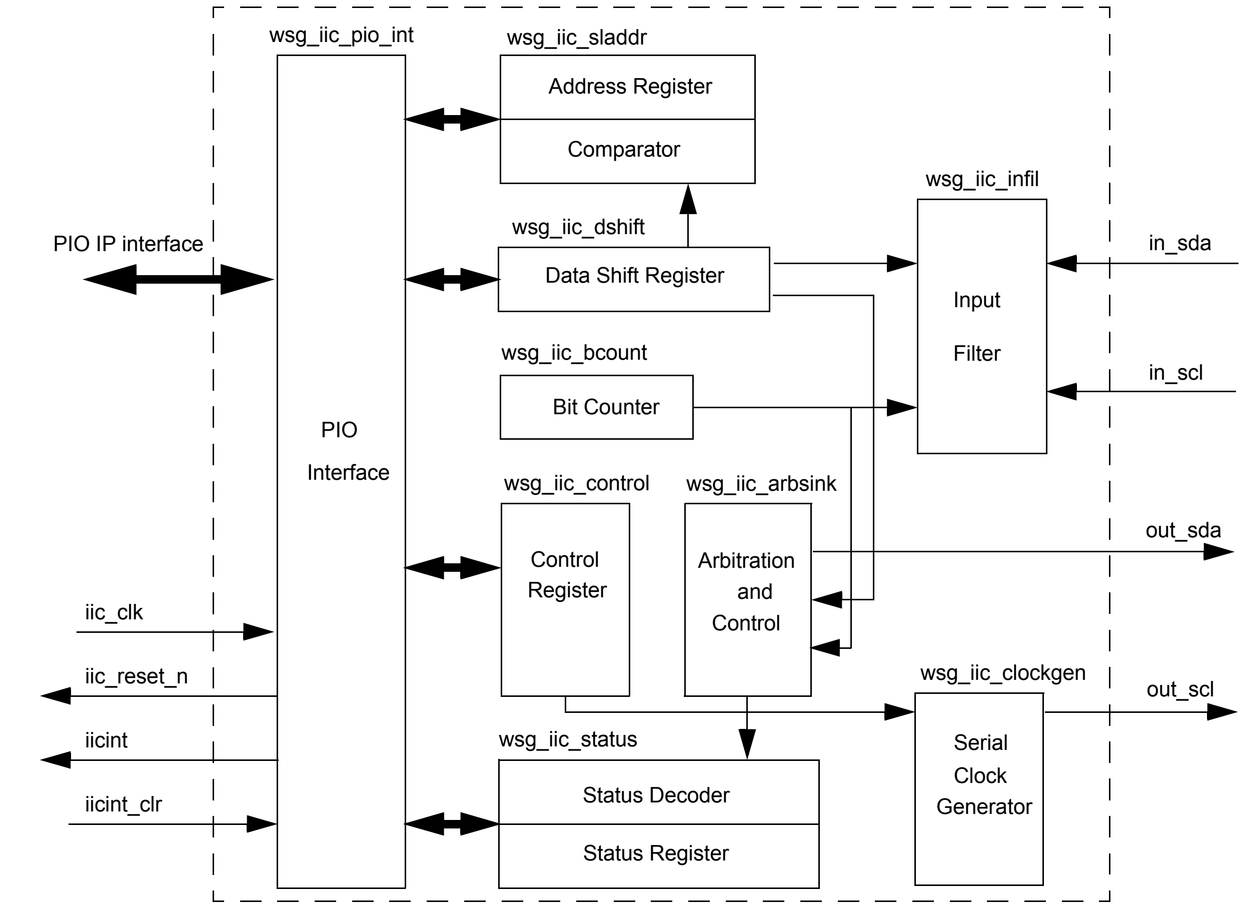


Figure 2: I^2C Core Submodules

15.3.1.1 I^2C Arbitration and Control Logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic ‘1’ actually appears as a logic 1 on the I^2C bus. If another device on the bus overrules a logic ‘1’ and pulls the SDA line low, arbitration is lost and the I^2C module immediately changes from master transmitter to slave receiver. The I^2C module will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete. Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^2C module is returning a “not acknowledge” (logic ‘1’) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, the I^2C module generates no further clock pulses.

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” (high level) duration is determined by the device that generates the shortest marks, and the “space” (low level) duration is determined by the device that generates the longest spaces.

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. The I^2C module will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. This block also controls all of the signals for serial byte handling. It provides the shift pulses for DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic and monitors the I^2C bus status.

15.3.1.2 Serial Clock Generator

This programmable clock pulse generator provides the SCL clock pulses when the I^2C module is in master transmitter or master receiver mode. It is switched off when the I^2C module is in a slave mode. The output frequency is dependent on the CR bits in the control register. The output clock pulses have a 50% duty cycle unless the clock generator is synchronized with other SCL clock sources, as described above.

15.3.1.3 Bit Counter

The bit counter tracks the number of bits that have been received during the byte transfer. The output from this counter is used to trigger events, such as address recognition and acknowledge generation, which occur at specific points during the byte transfer.

15.3.1.4 Control Register

This register is used by the microcontroller to control the generation of START and STOP conditions, enable the interface, control the generation of ACKs, and to select the clock frequency.

15.3.1.5 Status Decoder and Register

There are 26 possible bus states if all four modes of the I^2C module are used. The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I^2C bus status. The 5-bit code may be used for processing the various service routines. Each service routine processes a particular bus status. The 5-bit status code is stored in bits 7-3 of the status register. Bits 2-0 and 31-8 of the status register are always zero.

15.3.1.6 Input Filter

Input signals SDA and SCL from IO pad cells are synchronized with the internal clock. Spikes shorter than three clock periods are filtered out.

15.3.1.7 Address Register and Comparator

This SFR may be loaded with the 7-bit slave address to which I^{2}C module will respond when programmed as a slave. The least significant bit is used to enable the general call address recognition. The comparator compares the received 7-bit slave address with its own slave address. It also compares the first received 8-bit byte with the general call address. If an equality is found, the appropriate status bits are set and an interrupt is requested.

15.3.1.8 Data Shift Register

This register contains a byte of serial data to be transmitted or a byte which has just been received. Like all the registers in this module, only bits 7-0 are used. Data in DAT is always shifted from right to left; the first bit to be transmitted is the msb (bit 7) and, after a byte has been received, the first bit of received data is located at the msb (bit 7) of DAT. While data is being shifted out, data on the bus is simultaneously being shifted in. DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in DAT.

15.3.1.9 Related Interrupts

The serial interrupt signal (iic_intrn) issues an interrupt when any one of the 26 possible I^{2}C module states are entered. The only state that never causes an interrupt is state 0xF8, which indicates that no relevant state information is available.

15.3.2 Modes of Operation

The I^{2}C module hardware may operate in any of the following four modes:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

As a master, the I^{2}C module will generate all the serial clock pulses and the START and STOP conditions. A transfer ends with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I^{2}C bus will not be released.

Two types of data transfers are possible on the I^{2}C bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after each received byte except the last byte. At the end of the last received byte, a “not acknowledge” is returned.

In a given application, the I^{2}C module may operate as a master and as a slave. In the slave mode, the I^{2}C module hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^{2}C module switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

15.3.2.1 Master Transmitter Mode

Serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7-bit SLA) and the data direction bit as in Figure 3. In this case the data direction bit (R/W) will be a logic '0' (W). Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

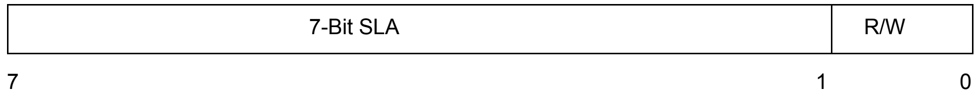


Figure 3: SDA First Transmitted Byte

In the master transmitter mode, a number of data bytes can be transmitted to the slave receiver. Before the master transmitter mode can be entered, the IIC_CONTROL register must be initialized with the EN bit set and the STA and STO bits reset. EN must be set to enable the I^{2}C module interface. If the AA bit is reset, the I^{2}C module will not acknowledge its own slave address or the general call address if they are present on the bus. This will prevent the I^{2}C module interface from entering a slave mode.

The master transmitter mode may now be entered by setting the STA bit. The I^{2}C module will then test the I^{2}C bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the status code in the status register (STA) will be 0x08. This status code must be used to vector to an interrupt service routine that loads IIC_DAT with the slave address and the data direction bit (SLA+W).

When the slave address and direction bit have been transmitted and an acknowledgment bit has been received, a number of status codes in STA are possible. The appropriate action to be taken for any of the status codes is detailed in Table on page 15-370. After a repeated start condition (state 0x10), the I2C module may switch to the master receiver mode by loading IIC_DAT with SLA+R.

15.3.2.2 Master Receiver Mode

The first byte transmitted contains the slave address of the transmitting device (7-bit SLA) and the data direction bit. In this case, the data direction bit (R/W) will be logic 1 (R). Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

In the master receiver mode, a number of data bytes are received from a slave transmitter. The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load IIC_DAT with the 7-bit slave address and the data direction bit (SLA+R).

When the slave address and data direction bits have been transmitted and an acknowledgment bit has been received, a number of status codes are possible in STA. The appropriate action to be taken for each of the status codes is detailed in Table . After a repeated start condition (state 0x10), the I2C module may switch to the master transmitter mode by loading IIC_DAT with SLA+W.

15.3.2.3 Slave Receiver Mode

Serial data and the serial clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

In the slave receiver mode, a number of data bytes are received from a master transmitter. To initiate the slave receiver mode, IIC_ADDRESS must be loaded with the 7-bit slave address to which the I^{2}C module will respond when addressed by a master. Also the least significant bit of IIC_ADDRESS should be set if the interface should respond to the general call address (0x00). The IIC_CONTROL register, should be initialized with EN and AA set and STA and STO reset in order to enter the slave receiver mode. Setting the AA bit will enable the logic to acknowledge its own slave address or the general call address and EN will enable the interface.

When IIC_ADDRESS and IIC_CONTROL have been initialized, the I^{2}C module waits until it is addressed by its own slave address, followed by the data direction bit which must be ‘0’ (W) for the I^{2}C module to operate in the slave receiver mode. After its own slave address and the W bit have been received, a valid status code can be read from IIC_DAT. This status code should be used to vector to an interrupt service routine and the appropriate action. Refer to the status codes in Table . The slave receiver mode may also be entered if arbitration is lost while the I^{2}C module is in the master mode.

If the AA bit is reset during a transfer, the I^{2}C module will return a not acknowledge (logic ‘1’) to SDA after the next received data byte. While AA is reset, the I^{2}C module does not respond to its own slave address or a general call address. However, the I^{2}C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to isolate the I^{2}C module from the I^{2}C bus temporarily.

15.3.2.4 Slave Transmitter Mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer.

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver. Data transfer is initialized as in the slave receiver mode. When IIC_ADDRESS and IIC_CONTROL have been initialized, the I^{2}C module waits until it is addressed by its own slave address followed by the data direction bit, which must be ‘1’ (R) for the I^2C module to operate in the slave transmitter mode. After its own slave address and the R bit have been received, a valid status code can be read from STA. This status code is used to vector to an interrupt service routine and the appropriate action Refer to the status codes in Table . The slave transmitter mode may also be entered if arbitration is lost while the I^2C module is in the master mode.

If the AA bit is reset during a transfer, the I^2C module will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^2C module is switched to the “not addressed” slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all ‘1s as serial data. While AA is reset, the I^2C module does not respond to its own slave address or a general call address. However, the I^2C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C module from the I^2C bus.

15.4 Register Descriptions

There are two I^2C modules in the PNX8526. I^2C 1 registers start at base address 0x04 5000 and I^2C 2 registers start at base address 0x04 6000. A 4 kB contiguous aperture space is reserved for these registers.

15.4.0.1 Register Address Map

Table 1: I^2C 1 Register Summary

Offset	Name	Description
0x04 5000	IIC_CONTROL	Controls the operation mode of the I^2C module
0x04 5004	IIC_DAT	Byte of data to be transmitted or received
0x04 5008	IIC_STATUS	Indicates the status of the I^2C module
0x04 500C	IIC_ADDRESS	Slave address of the I^2C module
0x04 5010	IIC_STOP	Set STO flag
0x04 5014	IIC_PD	Powerdown, reset I^2C sampling clock registers except for MMIO registers
0x04 5018	IIC_SET_PINS	Set I^2C bus SDA and/or SCL signals low
0x04 501C	IIC_OBS_PINS	Observe I^2C bus SDA and SCL signals
0x04 5020-5FDC	-	Reserved
0x04 5FE0	IIC_INT_STATUS	Interrupt Status register
0x04 5FE4	IIC_INT_EN	Interrupt Enable register
0x04 5FE8	IIC_INT_CLR	Interrupt Clear register
0x04 5FEC	IIC_INT_SET	Interrupt software set register
0x04 5FF4	IIC POWERDOWN	Powerdown mode, switch module clock off.
0x04 5FFC	Module ID	Module Identification and revision information

Table 2: I^{2}C 2 Register Summary

Offset	Name	Description
0x04 6000	IIC_CONTROL	Controls the operation mode of the I^{2}C module
0x04 6004	IIC_DAT	Byte of data to be transmitted or received
0x04 6008	IIC_STATUS	Indicates the status of the I^{2}C module
0x04 600C	IIC_ADDRESS	Slave address of the I^{2}C module
0x04 6010	IIC_STOP	Set STO flag
0x04 6014	IIC_PD	Powerdown, reset I^{2}C sampling clock registers except for mmio registers
0x04 6018	IIC_SET_PINS	Set I^{2}C bus SDA and/or SCL signals low
0x04 601C	IIC_OBS_PINS	Observe I^{2}C bus SDA and SCL signals
0x04 6020-6FDC	-	Reserved
0x04 6FE0	IIC_INT_STATUS	Interrupt Status register
0x04 6FE4	IIC_INT_EN	Interrupt Enable register
0x04 6FE8	IIC_INT_CLR	Interrupt Clear register
0x04 6FEC	IIC_INT_SET	Interrupt software set register
0x04 6FF4	IIC POWERDOWN	Powerdown mode, switch module clock off.
0x04 6FFC	Module ID	Module Identification and revision information

I^{2}C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 5000			*I2C CONTROL*	
31:8		-	Unused	Ignore upon read. Write as zeroes.
7	R/W	0	AA	I^{2}C acknowledge bit 0 = Acknowledge not returned during acknowledge clock pulse 1 = Acknowledge returned during acknowledge clock pulse
6	R/W	0	EN	I^{2}C enable bit 0 = Disable I^{2}C module 1 = Enable I^{2}C module
5	R/W	0	STA	I^{2}C start bit 0 = Slave mode, accept transactions 1 = Master mode, generate start condition if bus is free

I^2C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R	0	STO	I^2C stop bit 0 = Slave mode, accept transactions 1 = Generate stop condition on I^2C bus when I^2C module is master.
3		-	Unused	Ignore upon read. Write as zeroes.
2:0	R/W	000	CR2	These three bits determine the serial clock frequency. The 24 MHz I2C Clock is divided to achieve the desired frequency. [2:0]DividerFrequency (kHz) 00060400 00180300 010120200 011160150 100240100 10132075 11048050 11196025

Bit 7: AA Address Acknowledge

If the AA flag is set, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:

- The “own slave address” has been received.
- The general call address has been received while the general call bit (GC) in the ADR register is set.
- A data byte has been received while I^2C module is in the master receiver mode.
- A data byte has been received while I^2C module is in the addressed slave receiver mode.

If the AA flag is cleared and the I^2C-bus module is in the addressed slave transmitter mode, state 0xC8 will be entered after the last serial bit is transmitted. The I^2C module leaves state 0xC8, enters the “not addressed” slave receiver mode, and the SDA line remains at a high level. In state 0xC8, the AA flag can be set again for future address recognition.

If the AA flag is cleared and the I^2C module is in the “not addressed” slave mode, its own slave address and the general call address are ignored. Consequently, no acknowledge is returned, and a serial interrupt is not requested.

Thus, I^2C module can be temporarily released from the I^2C bus while the bus status is monitored. While I^2C module is released from the bus, START and STOP conditions are detected, and serial data is shifted in.

Address recognition can be resumed at any time by setting the AA flag. If the AA flag is set when the part's own slave address or the general call address has been partly received, the address will be recognized at the end of the byte transmission.

Bit 6: EN Enable

When EN is '0', the SDA and SCL outputs are constantly at 1 level leading to a high impedance state at the associated port lines SDA and SCL. The state of the SDA and SCL input lines SDA and SCL is ignored; I^2C module is in the "not addressed" slave state, and the STO bit in IIC_CONTROL is forced to '0'. No other bits are affected. SDA and SCL port lines may be used as open drain I/O ports.

When EN is '1', I^2C module is enabled. The port latches associated to SDA and SCL must be set to logic 1.

EN should not be used to temporarily release I^2C module from the I^2C bus because when EN is reset, the I^2C bus status is lost. The AA flag should be used to temporarily release the I^2C module.

Bit 5: STA Start bit

When the STA bit is set to enter a master mode, the I^2C module hardware checks the status of the I^2C bus and generates a START condition if the bus is free. If it is not free, the I^2C module waits for a STOP condition and generates a START condition after a delay of a low clock period in the internal serial clock generator.

If STA is set while I^2C module is already in master mode and one or more bytes are transmitted or received, the I^2C module transmits a repeated START condition. STA may be set at any time. It may also be set when the I^2C module is an addressed slave.

When the STA bit is reset, no START condition or repeated START condition will be generated.

Bit 4: STO Stop bit

A write to IIC_CONTROL Register will not affect this bit. It must be written via IIC_STOP Register. When the STO bit is set while I^2C module is in a master mode, a STOP condition is transmitted to the I^2C bus.

When the STOP condition is detected on the bus, the I^2C module hardware clears the STO flag. In a slave mode, the STO flag may be set to recover from an error condition. In this case, no STOP condition is transmitted to the I^2C bus. However, the I^2C module hardware behaves as if a STOP condition has been received and switches to the defined "not addressed" slave receiver mode. The STO flag is automatically cleared by the hardware.

If the STA and STO bits are both set, the STOP condition is transmitted to the I^2C bus if I^2C module is in master mode. In slave mode, the I^2C module generates an internal STOP condition, which is not transmitted. I^t then transmits a START condition.

When the STO bit is reset, no STOP condition will be generated.

There is also no interrupt generated for detection of a STOP condition which was created by the I^2C. (The I^2C needs to be in master mode to do this.) And there is no status code for this condition in the I^2C STATUS register.

Bits [2:0]: CR2

These three bits determine the serial clock frequency when I^2C module is in master mode. This register shall be changed only when EN bit is'0.'

CR	IIC CLK Divide by	I^2C Bit Frequency (kHz) Derived from a 24 MHz Clock
000	60	400
001	80	300
010	120	200
011	160	150
100	240	100
101	320	75
110	480	50
111	960	25

I^2C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 5004			*I2C DATA REGISTER*	
31:8		-	Unused	Ignore upon read. Write as zeroes.
7:0	R/W	0	DAT	Write byte data to be transmitted on the I2C bus. Read byte data has been received from the I2C bus.

Bit [7:0]: DAT

DAT contains a byte of serial data to be transmitted or a byte which has just been received. This special function register can be read from or written to while it is not in the process of shifting a byte. This occurs when I^2C module is in a defined state. Data in DAT is always shifted from right to left: the first bit to be transmitted is the msb (bit 7), and after a byte has been received, the first bit of received data is located at the msb of DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; DAT always contains the last data byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in DAT. Reset initializes DAT to 0x00.

A logic '1' in DAT corresponds to a high level on the I^2C bus, and a logic '0' corresponds to a low level on the bus. Serial data shifts through DAT from right to left.

DAT and the ACK flag form a 9-bit shift register which shifts in or shifts out 8 bits, followed by an acknowledge bit. The ACK flag is controlled by the I^2C module hardware and cannot be accessed by the CPU. Serial data is shifted through the ACK flag into DAT on the rising edges of clock pulses on the SCL line. When a byte has been shifted into DAT, the serial data is available in DAT, and the acknowledge bit is returned by the control logic during the 9th clock pulse. Serial data is shifted out from DAT via a buffer on the falling edges of clock pulses on the SCL line.

When the CPU writes to DAT, the buffer is loaded with the contents of DAT(7) which is the first bit to be transmitted to the SDA line. After nine serial clock pulses, the eight bits in DAT will have been transmitted to the SDA line, and the acknowledge bit will be present in ACK.

Remark: The eight transmitted bits are shifted back into DAT.

I^2C 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Offset 0x04 5008			*I2C STATUS REGISTER*	
31:8		-	Unused	Ignore upon read. Write as zeroes.
7:3	R	11111	STA	Indicates the status of the I2C module.
2:0		-	Unused	Ignore upon read. Write as zeroes.

Bit [7:3]: STA Status register

STA is a read-only special function register. Writing to this register has no affect. The three least significant bits are always zero. The bits 7-3 contain the status code. There are 26 possible status codes. When STA contains 0xF8, no relevant state information is available and no serial interrupt is requested. Reset initializes STA to 0xF8. All other STA values correspond to defined I^2C module states.

See the final row of the table for explanations of the terms used in Table .

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	**Bus Module Status**	**To/From DAT**	**STA**	**STO**	**SI**	**AA**	**Next Action Taken**
0x00	Bus Error	No DAT Action	0	1	0	X	HW will enter the “not addressed” slave mode.
0x08	A Start condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted, ACK bit will be received (I^2C bus module will be switched to MST/TRX mode).
		Load SLA+R	X	0	0	X	SLA+R will be transmitted, ACK bit will be received (I^2C bus module will be switched to MST/REC mode).

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	Bus Module Status	To/From DAT	STA	STO	SI	AA	Next Action Taken
0x10	A repeated Start condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted, ACK bit will be received (I^2C bus module will be switched to MST/TRX mode).
		Load SLA+R	X	0	0	X	SLA+R will be transmitted, ACK bit will be received (I^2C bus module will be switched to MST/REC mode).
0x18	SLA+W has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in DAT has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	Bus Module Status	To/From DAT	STA	STO	SI	AA	Next Action Taken
0x30	Data byte in DAT has been transmitted; NOT ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in the SLA+R/W or Data bytes	No DAT action	0	0	0	X	I^{2}C bus will be released; the “not addressed” slave mode will be entered.
		No DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received	No DAT action	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received	No DAT action	1	0	0	X	Repeated START condition will be transmitted.
		No DAT action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned	Read data byte	0	0	0	0	Data byte will be received; NOT ACK bit will also been returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	Bus Module Status	To/From DAT	STA	STO	SI	AA	Next Action Taken
0x58	Data byte has been received; NOT ACK has been returned	Read data byte	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO bit will be reset.
0x60	Own SLA+W has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General call address (00H) has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	Bus Module Status	To/From DAT	STA	STO	SI	AA	Next Action Taken
0x78	Arbitration lost in SLA+R/W as master; General call address has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLA; data byte has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; data byte has been received; NOT ACK has been returned	Read data byte	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		Read data byte	0	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		Read data byte	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when bus becomes free.
		Read data byte	1	0	0	1	Switched to no addressed SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’; A START condition will be transmitted when the bus becomes free.

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	Bus Module Status	To/From DAT	STA	STO	SI	AA	Next Action Taken
0x90	Previously addressed with general call; data byte has been received; ACK has been returned	Read data byte	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x98	Previously addressed with general call; data byte has been received; NOT ACK has been returned	Read data byte	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address
		Read data byte	0	0	0	1	Switched to “not addressed” SLV mode; Own SLA will be recognized; general call address will be recognized if ADR(0) =’1’
		Read data byte	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’; A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as SLV/(REC or TRX) (But for SLV/TRX a violation of I^{2}C bus format)	No STA action	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.

Table 3: I2C Module Status Decoded

		Application Software Response					
			To CON				
Status Code STA	Bus Module Status	To/From DAT	STA	STO	SI	AA	Next Action Taken
		No STA action	0	0	0	1	Switched to “not addressed” SLV mode; Own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		No STA action	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		No STA action	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’. A START condition will be transmitted when the bus becomes free.
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received; ACK has been received	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in DAT has been transmitted; ACK has been received	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.

Table 3: **I2C Module Status Decoded**

		Application Software Response					
			To CON				
Status Code STA	**Bus Module Status**	**To/From DAT**	**STA**	**STO**	**SI**	**AA**	**Next Action Taken**
0xC0	Data byte in DAT has been transmitted; NOT ACK has been received (AA = X)	No STA action	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		No STA action	0	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		No STA action	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		No STA action	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in DAT has been transmitted (AA = 0); ACK has been received	No STA action	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		No STA action	0	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		No STA action	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		No STA action	1	0	0	1	Switched to “not addressed” SLV mode; Own SLA will be recognized; general call address will be recognized if ADR(0) =’1’. A START condition will be transmitted when the bus becomes free.
0xF8	No information available	No DAT action	No IIC_CONTROL action				Wait or proceed with current transfer.

Key:

MST—Master	SLA—Slave address
SLV—Slave	W—Write bit
REC—Receiver	R—Read bit
TRX—Transmitter	X—Don’t care bit

I^{2}C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 500C			*I2C ADDRESS REGISTER*	
31:8		-	Unused	Ignore upon read. Write as zeroes.
7:1	R/W	0x00	SLAVE_ADDR	Slave address when in slave mode
0	R/W	0x0	GEN_CALL_ADDR	General call address 0 = Does not generate interrupt if general call address is detected on the I2C bus. 1= Generates interrupt if general call address is detected.

Bit [7:1]: SLAVE_ADDR Slave Address

SLAVE_ADDR is not affected by the I^{2}C module hardware. The contents of this register are irrelevant when I^{2}C module is in a master mode. In the slave mode, the bits 7:1 must be loaded with the microcontroller’s own slave address. These bits correspond to the 7-bit slave address which will be recognized on the incoming data stream from the I^{2}C bus. When the slave address is detected and the interface is enabled, a serial interrupt will be generated.

Bit 0: GEN_CALL_ADDR General Call Address

When this bit is set, the general call address (Slave Address[7:1] on I2C bus = 0x00, R/W bit on I2C bus = 0) is recognized. If not set, this bit is ignored.

I^2C 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Offset 0x04 5010			*I2C STOP REGISTER*	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0	STO	Set and read STO flag Writes: In master mode, set STO flag to indicate a STOP has been requested. Then generate a STOP condition on I2C bus. When the STOP condition is detected on the bus, the I2C module hardware clears the STO flag . In slave mode, set STO flag to indicate a STOP has been requested. No STOP condition is transmitted to the I2C bus. However, the I2C module hardware behaves as if a STOP condition has been received and switches to the defined “not addressed” slave receiver mode. The STO flag is immediately cleared by the hardware so that software can never see it set. Reads: View the STO flag to see if a STOP has been requested. STO flag can also be viewed by reading the STO bit of IIC_CONTROL. See STO bit of IIC_CONTROL register for more information

Bit 0: STO Stop

A write to this register regardless of what is on the data bus will set the STO flag. It will be reset by the hardware when a STOP condition is detected on the I^2C bus. The STO flag can be read from either IIC_STOP or IIC_CONTROL but writing to the STO bit of IIC_CONTROL will have no affect. Reset initializes STO to 0x0.

I^2C 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Offset 0x04 5014			*I2C PD REGISTER*	
31:3		-	Unused	Ignore upon read. Write as zeroes.
2	R/W	0	PD	This bit synchronously resets the I2C clock domain except for the MMIO registers. 0 = Don’t reset I2C clock domain. 1 = Reset I2C clock domain. Note: Do not reset the I2C clock domain until the I2C module is disabled using bit 6 of the I2C Control register.
1:0		-	Unused	Ignore upon read. Write as zeroes.
Offset 0x04 5018			*I2C BUS SET REGISTER*	
31:2		-	Unused	Ignore upon read. Write as zeroes.

I^{2}C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
1	W	0	SET_SCL_LOW	Pull I2C SCL bus signal to logic one or zero: 1 = Pulls SCL signal to logic zero. 0 = SCL signal is not pulled to logic zero.
0	W	0	SET_SDA_LOW	Pull I2C SDA bus signal to logic one or zero: 1 = Pulls SDA signal to logic zero. 0 = SDA signal is not pulled to logic zero.
Offset 0x04 501C			***I2C BUS OBSERVATION REGISTER***	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	R	0	OBSERVE_SCL	Observe I2C SCL bus signal: 1 = SCL signal is at logic one. 0 = SCL signal is at logic zero.
0	R	0	OBSERVE_SDA	Observe I2C SDA bus signal 1 = SDA signal is at logic one. 0 = SDA signal is at logic zero.
Offset 0x04 5020—5FDC			***Reserved***	
Offset 0x04 5FE0			***I2C INTERRUPT STATUS REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R	0	INT_STATUS	Interrupt status register. It reports any pending interrupts: 1 = Interrupt i pending. 0 = Interrupt i not pending.
Offset 0x04 5FE4			***I2C INTERRUPT ENABLE REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0	INT_ENABLE	Interrupt enable register 1 = Interrupt i is enabled. 0 = Interrupt is disabled.
Offset 0x04 5FE8			***I2C INTERRUPT CLEAR REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	W	0	INT_CLEAR	Interrupt clear register. 1 = Interrupt is cleared. 0 = Interrupt is not cleared.
Note: The I^{2}C module will look at a new transaction on the I^{2}C bus as soon as the previous interrupt has been cleared. Therefore, software must make sure that “interrupt clear” is the last transaction that is sent to the I^{2}C module before starting a new transaction.				
Offset 0x04 5FEC			***I2C INTERRUPT SET REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	W	0	INT_SET	Interrupt set register. Allows software to set interrupts. 1 = Interrupt is set 0 = Interrupt is not set.

I^2C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 5FF4			*I2C POWERDOWN REGISTER*	
31	R/W	0	POWER_DOWN	0 = Normal operation of peripheral. This is the reset value. 1 = Module is powerdown and module clock can be removed. Module returns DEADABBA on all reads except for reads of the powerdown bit. Module generates ERR ACK on all writes except for writes to the powerdown bit.
30:0		-	Unused	Ignore upon read. Write as zeroes.

I^2C 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 5FFC			*I2C MODULE ID REGISTER*	
31:16	R	0x0105	MODULE ID	Unique 16-bit code. Module ID 0 and -1 are reserved for future use.
15:12	R	0	MAJREV	Major Revision
11:8	R	1	MINREV	Minor Revision
7:0	R	0	MODULE APERTURE SIZE	Aperture size = 4 kB*(bit_value+1), so 0 means 4 kB (the default).

I^2C 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
I2C 2 registers begin at offset 0x04 6000. In all other respects, they are identical to the I2C 1 registers.				

Chapter 16: Smartcard UART

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

16.1 Introduction

This document describes the architecture and implementation of the PNX8526 Smartcard UART interface, which is compliant with the ISO 7816 specifications.

Smartcard UART is a serial interface designed to connect with a Smartcard “conditional access module” for program access verification. Use of the Smartcard allows cable TV companies to ensure that the customer has the correct (by subscription) access to programming.

The main features of the Smartcard UART interface are as follows:

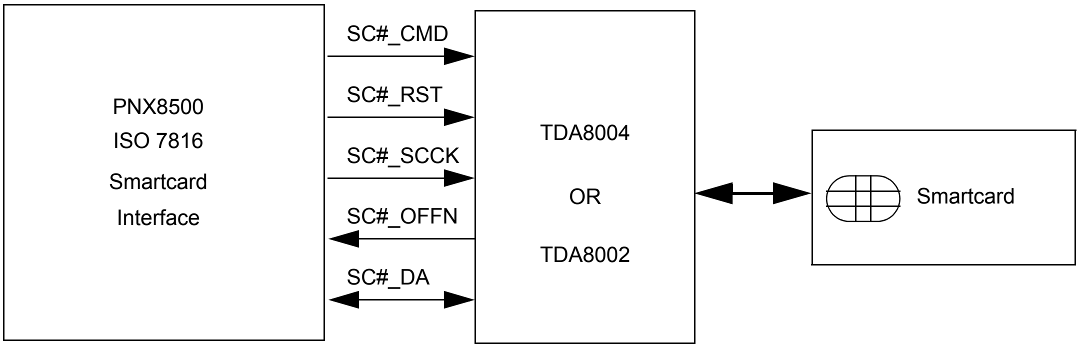
- Automatic convention processing
- Variable baud rates through frequency or division ratio programming
- Error management at character level for T=0
- An extra guard-time register
- Programmable 1 to 8-byte deep FIFO in reception mode
- Direct Memory Access (DMA) capabilities
- Parity error counter in reception mode
- Card clock STOP HIGH, clock STOP LOW
- Supports the asynchronous protocols T=0 and T=1 in accordance with ISO 7816 (T=0 is character transmission protocol and T=1 is block transmission protocol.)
- Versatile 24-bit timeout counter for ATR and waiting times processing
- 22-ETU counter for Block Guard Time
- Supports synchronous cards

16.1.1 Smartcard UART Interface

The PNX8526 Smartcard UART may be interfaced with a Philips TDA8004 (or any other Philips’ analog interface for asynchronous and synchronous smartcards, such as the TDA8002).

The TDA8002/TDA8004 perform all supply, protection and control functions for the card, and they are directly compatible with ISO 7816 specifications.

Additional information on these products can be found at www.semiconductors.com.



Note: # = 1 or 2

Figure 1: Connecting the Smartcard UART to the TDA8002/TDA8004

Table 1: Smartcard Signals

Pad Name	Description
SC1_RST SC2_RST	Card reset input from ISO UART (active low). This port can be directly managed by the software through the RSTIN bit in the UART2 Configuration Register (UCR2).
SC1_SCCK SC2_SCCK	Input for external clock. It is the clock signal to the card. Note: The TDA8004 /TDA8002 must be configured so the frequency on the TDA8004/ TDA8002 CLK pin is half the frequency of the XTAL1 pin. The frequency of the signal CLK_CARD is determined through the Clock Configuration Register (CCR) in the PNX8526 Smartcard UART.
SC1_CMD SC2_CMD	Smartcard activation input from the PNX8526 UART (active low). This port can be directly managed by the software through the START bit in UART2 Configuration Register (UCR2) in the PNX8526 UART.
SC1_OFFN SC2_OFFN	Interrupt to the PNX8526 UART. When a change occurs on this pad, the PNX8526 UART generates an internal interrupt. Then the software can read the state of this port on the $\overline{\text{BOF}}$ bit in the Mixed Status Register (MSR).
SC1_DA SC2_DA	Input/output serial line.

16.2 Functional Description

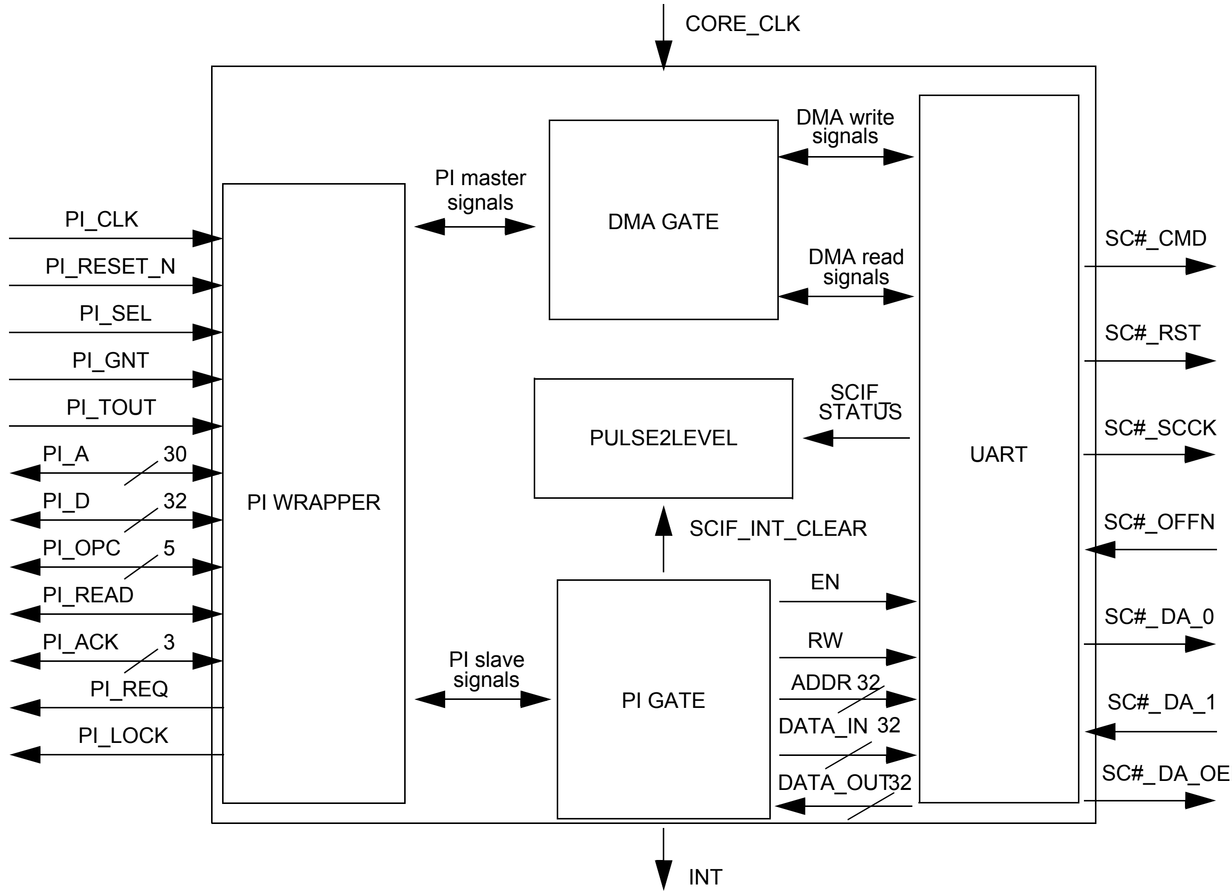


Figure 2: PNX8526 Interface Block Diagram

The PNX8526 Smartcard Interface can be divided into five blocks: UART, DMA GATE, PULSE2LEVEL, PI GATE and PI WRAPPER.

The **PI WRAPPER** block is the interface between the PI-Bus and the Smartcard Interface block.

The **DMA GATE** block allows the Smartcard Interface to store data, thanks to its internal 32-word FIFO (1 word = 4 bytes). This FIFO can be used both in transmission and in reception mode.

The **PI GATE** block is the interface between the PI WRAPPER module and the UART module.

The **UART** block is in charge of the transmission and reception of characters. It is the core of the PNX8526 Smartcard Interface. It contains an internal 8-bit FIFO which can be used *only* in reception mode.

16.3 Operation

The PNX8526 Smartcard Interface has two operation modes: the interrupt mode and the DMA mode.

- In interrupt mode, an interrupt will be generated each time a character has been sent or received.
- In DMA mode (DMAE bit set in UART Configuration Register 2 (UCR2)), an interrupt will be generated only at the end of the DMA transfer or if a problem happened during the DMA transfer (parity error, card removed from the reader, etc.).

When the general parameters of the card have been programmed, the UART may be used with the following registers: UART Receive and UART Transmit Registers (URR and UTR), UART Status Register (USR) and Mixed Status Register (MSR). In reception mode, a FIFO of 1 to 8 bytes may be used. This is configured with the FIFO Control Register (FCR).

The timeout counter helps the controller process different real-time tasks (ATR, WWT, BWT, mute card, etc.). Please refer to the ISO/IEC 7816 Specification for descriptions of these wait times.

The timeout counter is configured with the Timeout Configuration (TOC) register. The number of Elementary Time Units (ETUs) to count is programmed in registers TOR1, TOR2 and TOR3. It may be used as 24 bits, as “16 bits + 8 bits” or as “8 bits + 8 bits + 8 bits.” Each counter may be set to start counting once the mode is written in the TOC, on detection of a start bit on I/O, or as auto-reload.

At the end of the communication with the Smartcard, the controller may reset the START bit in UCR2 register. Then the TDA8004/TDA8002 performs a deactivation sequence. (Refer to the Philips TDA8004/8002 IC Card Interface Data Sheet for details on “Deactivation Sequence”).

16.3.1 Clock Circuitry

The clock to the card (clk_card) which is output on pin SC[2:1]_SCCK, the clock to the UART core (clk_uart) and the clock giving the Elementary Time Unit (clk_etu) are derived from the main internal clock of f_{SC} = 54 MHz.

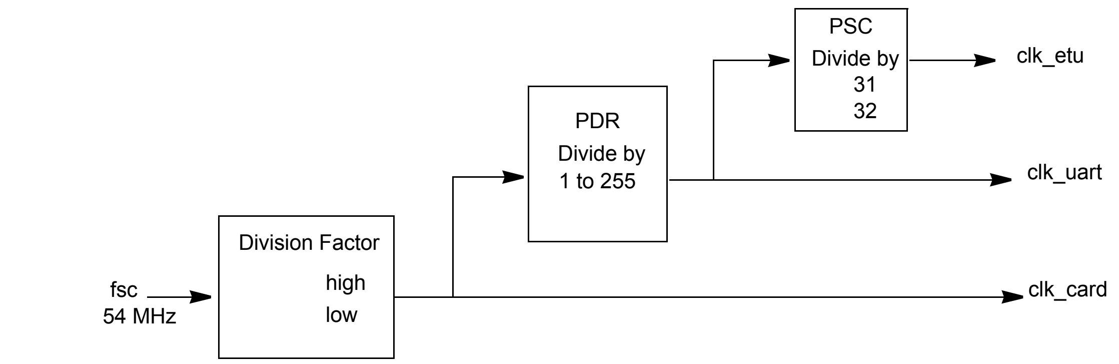


Figure 3: ISO UART Clock Diagram

clk_card

The clk_card frequency is derived from the main clock by programming the division factor field (AC) of the Clock Configuration register to get the desired frequency or to stop the clock High or Low. All transitions are synchronous, ensuring correct pulse length during start or change in accordance with ISO 7816. After power on, clk_card is set at Stop Low.

Table 2: Supported clk_card Frequencies

clk_card Frequency	Division Factor
27 MHz	2
13.5 MHz	4
9 MHz	6
6.75 MHz	8
4.5 MHz	12
3 MHz	18
2.25 MHz	24
1.6875 MHz	32

clk_uart

The clk_uart frequency is derived from clk_card using the division factor loaded in the Programmable Divider Register (PDR). This division factor depends on the F and D factors received during the Answer To Reset (ATR) session of the card. This allows the interface to achieve different baud rates. See Section 16.3.3 for details.

clk_etu

clk_etu gives the baud rate. It is derived from clk_uart using a Programmable Pre-scaler (PSC) of either 31 or 32 (field PSC of UART Configuration Register 2).

16.3.1.1 ISO UART

The ISO UART handles all the specific requirements defined in ISO T = 0 and T = 1 protocol types. It is clocked by clk_uart, which gives the sampling rate for start bit detection (the start bit is detected at the first low level on I/O) and the f_{clk_card} /372 frequency for ETU timing (in the reception mode, the bit is sampled at the middle of each ETU period). Changing the card clock frequency interferes with the baud rate.

16.3.2 Timeout Counters

The ISO UART has three timeout counters which are configurable with the Timeout Configuration register (with the field 8/16, 16/24) in the following ways:

- Three independent 8-bit counters
- One 8-bit counter and one 16-bit counter
- One 24-bit counter.

This timeout counter is useful for processing the clk_card counting during Answer To Reset (ATR) sequence, the Work Waiting Time (WWT), or the Block Waiting Time (BWT) defined in T = 1 protocol. Please refer to the ISO/IEC 7816 Specification for descriptions of these wait times.

One 24-bit Counter

The three counters are linked together to form a 24-bit counter. TOR1 is linked to TOR2 which is linked to TOR3. TOR1 is the lsb part of the 24-bit counter.

An interrupt is generated when the terminal count is reached and only the TO3 status bit is set in the USR/SCIF_INT_STATUS register.

A 24-bit timeout counter may be started for giving an interrupt after a number of ETUs programmed in registers TOR1, TOR2 and TOR3. It will help the controller process different real-time tasks (ATR, WWT, BWT etc.) if the controllers and card clocks are asynchronous.

An 8-bit and 16-bit Counter

Counter 1 is an independent 8-bit counter. Counters 2 and 3 are linked together to form a 16-bit counter (TOR2 is the lsb part).

One interrupt can be generated by each of these two counters when the terminal count is reached and the corresponding event bits are set in the USR/ SCIF_INT_STATUS register.

Three 8-bit Counters

The three counters are independent and controlled by their associated mode field in the Timeout Configuration register. They can be individually set in auto-reload mode or software trigger mode or start counting on start bit detection.

Each counter can generate an interrupt when the terminal count occurs and the corresponding event bit is set in the SCIF_INT_STATUS (TO1, TO2, TO3). All the interrupts are ORed and only one physical interrupt line is connected to the PIC.

16.3.2.1 Counter Operation

Counters TOR1, TOR2 and TOR3 have four operation modes, defined in the Timeout Configuration (TOC) register: Stop, Autoreload, Software-Triggered and “Triggered on start bit on I/O.”

For starting the action defined in a particular mode, the software must write the specific bits for all counters in the TOC register. The count will start upon writing to the TOC register or on the start bit detected on I/O.

- When operating the three as one 24-bit counter, set the mode for Counter3 (MODE31, MODE30) in the TOC register. The TO3 bit in the UART Status register reports when the counter has terminated.
- When operating as one 8-bit and one 16-bit counter, set the mode for Counter1 and Counter3 in the TOC register. The TO1 and TO3 bits in the UART Status register report when the 8-bit and 16-bit counters have terminated.
- When operating as three 8-bit counters, set the mode for each counter in the TOC register. The TO[3:1] bits in the UART Status register report when each counter terminates.

16.3.2.2 Programming Rules

Timers are a powerful control/check mechanism however, some rules must be followed to avoid basic mistakes:

- When the software decides to change the configuration of a timer before the end of the count (AUTORELOAD, SOFTWARE TRIGGERED, TRIGGERED ON START BIT ON I/O), first it must stop the timer (reset the appropriate bits in the TOC register) and second it can set the new configuration (set the appropriate bits in theTOC register). In TRIGGERED ON START BIT ON I/O mode, the timer is automatically reset each time there is a start bit and the related TORx register is loaded.
- The value stored in TORx register can be changed when the TIMERx has started its count. In SOFTWARE TRIGGERED mode, the corresponding TORx register can be changed just after the mode has been programmed in the TOC register. In TRIGGERED ON START BIT ON I/O mode and AUTORELOAD mode, the corresponding TORx register can be modified after the first start bit on I/O line.

16.3.3 Baud Rates

The baud rate is the frequency of clk_etu, which is determined by the settings of the Programmable Divider Register (PDR) and the PSC field of the UART 2 Configuration register. The baud rate should be set to “match the card” according to the F and D factors received during the Answer To Reset (ATR) session of the card.

Use Table 3 to determine how to set up these registers.

Table 3: PSC and PDR Settings According to F and D Factors

F D	0	1	2	3	4	5	6	9	10	11	12	13
1	31:12	31:12	31:18	31:24	31:36	31:48	31:60	32:16	32:24	32:32	32:48	32:64
2	31:6	31:6	31:9	31:12	31:18	31:24	31:30	32:8	32:12	32:16	32:24	32:32
3	31:3	31:3		31:6	31:9	31:12	31:15	32:4	32:6	32:8	32:12	32:16
4				31:3		31:6		32:2	32:3	32:4	32:6	32:8
5						31:3		32:1		32:2	32:3	32:4
6										32:1		32:2
8	31:1	31:1		31:2	31:3	31:4	31:5		32:2		32:4	
9							31:3					

The table cells contain the PSC:PDR settings for the related F and D factors.

16.3.4 Transmission Mode

The following occurs in transmission mode:

- Transmission according to the convention detected during ATR, consequently the CPU only has to write characters in direct convention into the UART Transmit register. Transmission of the next character may start at 12 ETUs + the Extra Guard time value in case of no error or 13 ETUs in case of error.
- Parity calculation and detection of repetition request from the card in case of error
- The Last Character to Transmit (LCT) bit in the UART Configuration Register 1 allows fast reconfiguration for receiving the answer 12 ETUs after the start bit of the last transmitted character.

The data transmitted to the card are stored in direct convention in the UART Transmit register. The ISO UART can automatically recognize the convention during ATR, then process the data according to the convention.

Between the transmission of two characters to the card, the ISO UART automatically inserts a number of guard ETUs equal to the value stored in the Guard Time register. GT[7:0] is in the range of 0 to 255. When GT[7:0] = 255, it indicates that the minimum delay between the start leading edge of two consecutive characters is 12 ETUs for T = 0 protocol, 11 ETUs for T = 1 protocol.

Guard Time corresponds to modulo -255 binary value stored in the Guard Time register with an exception for protocol T = 1 as shown in Table 4 below.

Table 4: Guard Time Selection

GT[7:0] (in decimal value)	T = 0	T = 1
0	0	0
1	1	1

Table 4: Guard Time Selection *…Continued*

GT[7:0] (in decimal value)	T = 0	T = 1
...	...	...
254	254	254
255	0	-1

When the CPU needs to transmit data to the card, it first sets the T/R bit in UART Configuration register 1, which sets the UART in transmission mode. As soon as a character has been written in the transmit register, the UART makes the conversion, calculates the parity and starts the transmission. When the character has been transmitted, it surveys the I/O line at 11 ETUs in order to know if an error has been detected by the card.

If no error has occurred, the UART sets the Transmission Buffer Empty (TBE) bit in the UART Status and Mixed Status registers and sends a request to get the next character to be sent. That request is either an interrupt fed to the interrupt controller if the DMA is disabled (interrupt mode) or a DMA request if the DMA is enabled. When a new character is loaded in the transmit register, TBE is cleared.

If an error has occurred, it sets the bits Buffer Empty and PE (parity error during transmission), disables the DMA if it was enabled (DMAE is automatically cleared) and by means of an interrupt, warns the CPU to rewrite the previous character in the transmit register. If the character has been rewritten in Transmit before 13 ETUs, the transmission will start at 13 ETUs; otherwise, it will start after Transmit has been loaded with the character to transmit. If the DMA channel was enabled for transferring the data, the software has to read DMA Read Address and DMA Length to know on which address and byte number the transfer was failing. Then it must re-initialize the DMA channel to restart the transfer—either from the beginning of the block or at the point of the failed character.

To finish a transmission in interrupt mode, the Last Character to Transmit (LCT) bit must be set so that the UART will force reception mode into “ready” to get the reply from the card at 11 ETUs. This bit is automatically cleared when the Smartcard controller switches to reception mode.

When the session is completed, the CPU re-initializes UART with the Reset register.

16.3.5 Reception Mode

The following occurs in reception mode:

- Detection of the inverse or direct convention at the beginning of ATR
- Automatic convention setting, so the CPU only receives characters in direct convention
- Parity checking and automatic request for character repetition in case of error (reception is possible at 12 ETUs)

The data received from the card are stored in direct convention in the UART Receive register. The ISO UART automatically recognizes the convention during the ATR, then processes the data according to the convention.

The ISO UART is configured in the reception mode. When the CPU wants to start a session, it sets the Start Session (SS) bit in UART Configuration 1. The UART controller will sample the I/O line (sampling rate fclk_card) to detect the first start bit.

The convention is recognized on the first character of the ATR and the UART configures itself in order to exchange direct data without parity processing with the CPU whatever the convention of the card is. The SS bit must be reset by software when the first character is received. At the end of every character, the UART tests the parity and resets what is necessary for receiving another character.

If no parity error is detected upon reception of a character, the UART sets the Reception Buffer Full (RBF) bit in the UART Status and Mixed Status registers, and sends a request to get the character read before the reception of the next one. That request is either an interrupt fed to the interrupt controller if the DMA is disabled (interrupt mode), or a DMA request if the DMA is enabled. The RBF bit is cleared in these registers by setting the RUR bit in the Command register.

If a parity error has been detected, the UART pulls down the I/O line low between 10.5 and 11.5 ETUs. It also sets the RBF and PE (parity error during reception) bits in protocol T=1 and PE bit only in protocol T=0. Then it sends an interrupt to the interrupt controller, which warns the CPU that an error has occurred. If the DMA was enabled, the transfer is aborted and the DMA is stopped. The retry procedure should be handled by software.

At the end of a successful DMA transfer in reception mode, the UART flushes the internal DMA gate.

16.3.6 Activation and Deactivation Sequence

After reset, the SC#_CMD line is set high to disable the voltage on the card, and the reset to the card is active (SC#_RST= ‘0’). The clock to the card (SC#SCCK) is disabled and stopped low. The I/O line is in high impedance.

16.3.6.1 Activation Sequence

To start an activation sequence with the card, the software driver should first wait for the interrupt indicating the card presence (SC#_OFFN = ‘1’) or poll at least once the $\overline{\text{BOF}}$ bit of the Mixed Status register. It should then initialize the clock parameters if the default value is not correct and set the START bit of the UART Configuration 2 register. The activation sequence will start and the signal SC#_CMD will be asserted. All the timings of the activation sequence are under software control and software might use the timer/counter for this purpose.

The recommended activation sequence is depicted in Figure 4 and consists of the following phases:

- START bit is set by software and then SC#_CMD is set to ‘0’ by hardware
- The I/O line SC_DA is set in high impedance
- clk_card is enabled and output on the pin SC#_SCCK. Software must ensure that clk_card is initialized at the right speed.

- Reset (SC#_RST) is kept low for about 45000 clk_card periods (t_{rst}), then it is set high, except if the ATR is received when RSTIN is at low level. In that case, the low level on SC#_RST is maintained.
- The ISO 7816 controller is now expecting the ATR sequence. The software has to take care that the card is answering in a certain delay by means of the timeout counter.

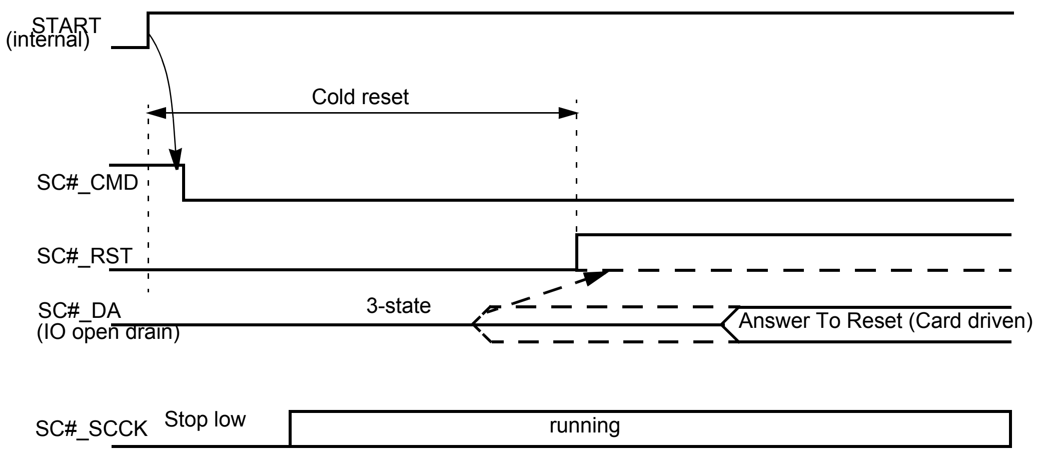


Figure 4: Activation Sequence

16.3.6.2 Warm Reset

To activate a warm reset sequence (SC#_CMD is low for awhile), the user has to set the RSTIN bit of the UART Configuration 2 register. This will assert the signal SC#_RST and the pin will stay asserted as long as the control bit is set. After the SC#_RST line is set to ‘1,’ the sequencer waits for the ATR session. In case of reception of the ATR sequence during SC#_RST at low level, SC#_RST is kept low for about 45000 clk_card periods (trst), then it is set high, except if the ATR is received when RSTIN is at low level. In that case, the low level on SC#_RST is maintained.

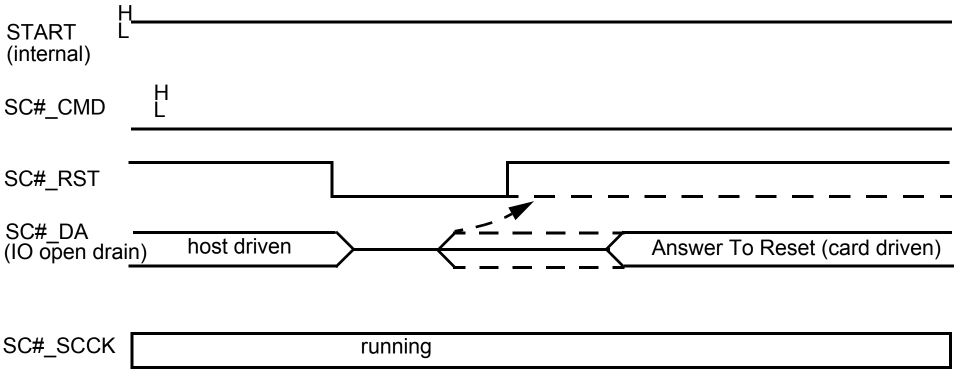


Figure 5: Warm Reset Sequence

16.3.6.3 Emergency Sequence

When a fault is detected by the external Smartcard circuitry, the SC#_OFFN line has to be externally pulled low so the Smartcard controller will detect that something is wrong and generate an interrupt with the OCH bit set in the Mixed Status register. Then the software reacts immediately by starting an emergency sequence which has the exact same timing as the deactivation sequence.

In a standard application the SC#_OFFN line may be pulled low when one of the following conditions occurs:

- Removing the card during a session
- Short circuit or high current on VCC
- Overheating
- VDD spikes

16.3.6.4 Deactivation Sequence

To stop a session with the card, software has to clear the START bit of the UART Configuration 2 register so the controller will de-assert SC#_CMD. Then the software starts the deactivation sequence in the following order:

- SC#_RST is set low
- clk_card is disabled and kept low
- I/O line is set in high impedance
- SC#_CMD is set high

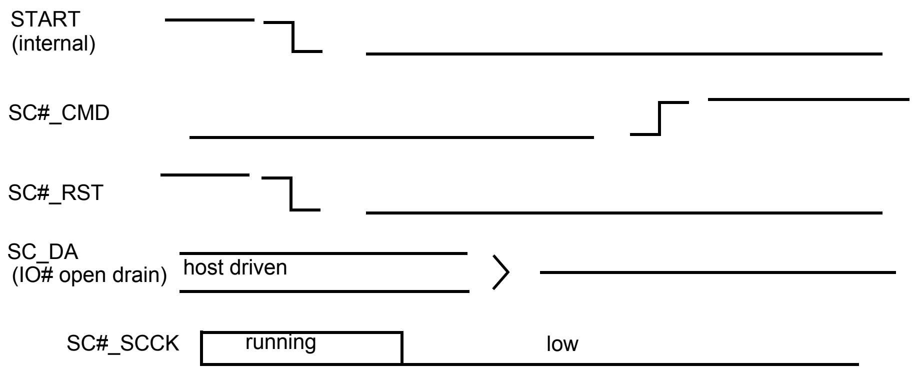


Figure 6: Deactivation/Emergency Sequence

16.3.7 ISO UART Interrupts

Each ISO UART can generate an interrupt fed to the interrupt controller on two separated lines. It is assumed that the interrupt controller is correctly initialized to handle the ISO UART interrupts. Two cases should be considered: the first is when the ISO UART is configured in interrupt mode, the second is when the ISO UART is configured in DMA mode.

16.3.7.1 Interrupt Mode

In interrupt mode (DMAE = 0), an interrupt is generated each time an event occurs. Either a data byte is received/transmitted or a status bit raises. During the ATR session, the character transfer should be done in interrupt mode (DMA disabled).

An interrupt is generated in the following cases:

- Each time a data byte is either transmitted or received.
- When a parity error is detected at reception of a data byte (protocol T = 0 or 1).
- When an early answer is detected.
- When the OFFN line is toggling (card insertion or power supply problem).
- When an overrun is detected.

16.3.7.2 DMA Mode

In DMA mode (DMAE = 1), the data bytes are transferred through the DMA channel and an interrupt is generated only when the complete block is transferred (DONE = 1), except if an anomaly is detected during the DMA transfer. In that case, an interrupt is immediately generated to signal the software driver about the error or event. When the interrupt is generated (DONE = 1 or anomalies during DMA transfer), the DMA is disabled by hardware (the DONE bit is cleared), to avoid further spurious DMA transfers polluting the memory buffer.

An interrupt is generated in the following cases:

- When a DMA block transfer is successfully finished (DONE = 1), (although the interrupt mode is still possible). After the ATR session is successfully performed, character transfer should always be done using the DMA channel to avoid throughput error and data losses.
- When either a parity error (protocol T = 0 or 1) or an early answer is detected during a DMA block transfer.
- When the OFFN line is toggling (card insertion or power supply problem).

16.3.7.3 Interrupt Mechanism

The block diagram of the Smartcard interrupt is shown in Figure 7.

When an event occurs, at least one of the status bits located in either the UART Status or Mixed Status registers is set to flag the event. The status bits are gathered into the SCIF_INT_STATUS register to flag the interrupt cause. Each of these interrupt flags can be individually enabled and then “ORed” to generate one interrupt line which is fed to the Priority Interrupt Controller (PIC). When the Smartcard interrupt is acknowledged by the CPU, the interrupt service routine should do the following tasks in the right order:

- Read SCIF_INT_STATUS register to identify the cause (more than one bit can be set).

- Read the UART Status and Mixed Status registers to know which event has occurred.
- Clear the status bit in the UART Status and Mixed Status registers.
- Reset the bit in SCIF_INT_STATUS by writing a ‘1’ in the SCIF_INT_CLEAR register in the bit position to be cleared. The status bit of UART Status and Mixed Status registers must be cleared before clearing the interrupt bit, otherwise an interrupt will raise again.
- Take the necessary actions to continue the running process.

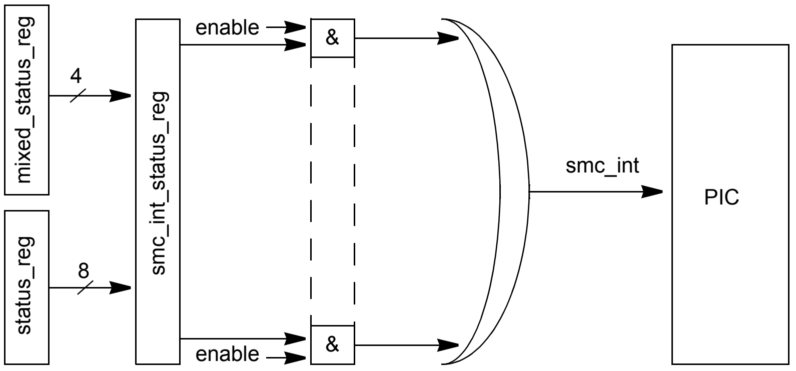


Figure 7: Interrupt Mechanism

16.4 Register Descriptions

The PNX8526 has two Smartcard UART interfaces. The Smartcard 1 registers begin at
0x04 3000. The Smartcard 2 registers begin at 0x04 4000.

16.4.1 Register Address Map

Table 5: Smartcard UART Register Summary

Offset	Name	Description
Smartcard 1		
0x04 3000	Reset Register (RER)	Reset UART Status Registers (USR) and some bits in Mixed Status Register (MSR)
0x04 3004	Clock Configuration Register (CCR)	Determines the Smartcard clock frequency.
0x04 3008	Programmable Divider Register (PDR)	This register is used for counting the cards clock cycles forming the ETUs.
0x04 300C	UART Configuration Register 2 (UCR2)	Used to configure the Smartcard Interface.
0x04 3010	DMA Read Address Register (DRA)	It points at the memory address at which the transfer is supposed to start when the UART is in transmission mode.
0x04 3014	Guard Time Register (GTR)	This register is used to store the number of guard ETUs given by the card during ATR.

Table 5: Smartcard UART Register Summary *…Continued*

Offset	Name	Description
0x04 3018	UART Configuration Register 1 (UCR1)	Used to configure the Smartcard Interface.
0x04 301C	DMA Length Register (DLR)	Number of bytes to send to the Smartcard or to receive from the Smartcard when in DMA mode.
0x04 3020	Timeout Configuration Register (TOC)	Manages timeout counters (TOR1, TOR2 and TOR3).
0x04 3024	Timeout Register 1 (TOR1)	Number of ETUs that timer1 will count before generating an interrupt.
0x04 3028	Timeout Register 2 (TOR2)	Number of ETUs that timer2 will count before generating an interrupt.
0x04 302C	Timeout Register 3 (TOR3)	Number of ETUs that timer3 will count before generating an interrupt.
0x04 3030	Mixed Status Register (MSR) (Read)	A status register
0x04 3030	FIFO Control Register (FCR) (Write)	Controls the depth of the FIFO and the parity error counter.
0x04 3034	UART Transmit Register (UTR) (Write)	This register contains the next character to transmit to the Smartcard.
0x04 3034	UART Receive Register (URR) (Read)	Characters received from the Smartcard are stored in this register.
0x04 3038	UART Status Register (USR)	A status register
0x04 303C	DMA Write Address Register (DWA)	It points at the memory address at which the transfer is supposed to start when the UART is in reception mode.
0x04 3040	Command Register (CRE)	Flushes the DMA gate and manages FIFO read operations.
0x04 3FE0	SCIF_INT_STATUS	All bits of UART which can trigger an interrupt (all bits of UART Status register plus 4 bits in Mixed Status register)
0x04 3FE4	SCIF_INT_ENABLE	Enable interrupts
0x04 3FE8	SCIF_INT_CLEAR	Clear interrupts
0x04 3FEC	SCIF_INT_SET	Set interrupts
0x04 3FF4	SCIF_POWERDOWN	Locks access to registers except for the module ID and powerdown registers.
0x04 3FFC	SCIF_MODULE_ID	Module ID register (0x01060000)
Smartcard 2		
0x04 4000	Reset Register (RER)	The Smartcard 2 registers, starting at 0x04 4000, are identical to the Smartcard 1 registers, described above.
0x04 4004	Clock Configuration Register (CCR)	
0x04 4008	Programmable Divider Register (PDR)	
...	...	
0x04 4FFC	SCIF_MODULE_ID	

SMARTCARD UART 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Note: bit [7], RIU bar, of register UCR1 must be set to 1 before any action on the Smartcard Uart registers.				
Offset 0x04 3000			*Reset Register (RER)*	
31:16	W	0	Unused	
15	W	0	RS_DONE	Reset the bit DONE in Mixed Status Register. RS_DONE is automatically reset by hardware.
14	W	0	RS_BGT	Reset the bit BGT in Mixed Status Register. RS_BGT is automatically reset by hardware.
13	W	0	RS_OCH	Reset the bit OCH in Mixed Status Register. RS_OCH is automatically reset by hardware.
12	W	0	RS_DMAERR	Reset the bit DMA_ERR in Mixed Status Register. RS_DMAERR is automatically reset by hardware.
11	W	0	RS_DMAABORT	Reset the bit DMA_ABORT in Mixed Status Register. RS_DMAABORT is automatically reset by hardware.
10	W	0	RS_TO3	Reset the bit TO3 in UART Status Register. RS_TO3 is automatically reset by hardware.
9	W	0	RS_TO2	Reset the bit TO2 in UART Status Register. RS_TO2 is automatically reset by hardware.
8	W	0	RS_TO1	Reset the bit TO1 in UART Status Register. RS_TO1 is automatically reset by hardware.
7	W	0	RS_EA	Reset the bit EA in UART Status Register. RS_EA is automatically reset by hardware.
6	W	0	RS_PE	Reset the bit PE in UART Status Register. RS_PR is automatically reset by hardware.
5	W	0	RS_OVR	Reset the bit OVR in UART Status Register. RS_OVR is automatically reset by hardware.
4	W	0	RS_FER	Reset the bit FE in UART Status Register. RS_FER is automatically reset by hardware.
3	W	0	RS_TBERBF	Reset the bit TBE/RBF in UART Status Register. RS_TBERBF is automatically reset by hardware.
2	W	0	RFI	Reset FIFO pointers. This bit is automatically reset by hardware.
1	W	0	RMS	Reset Mixed Status (RMS) register. When set, this bit clears bits OCH, DONE, DMA_ABORT, DMA_ERR and BGT in the MSR register. It is automatically reset by hardware. Note that bit BGT in MSR does not generate interrupts.
0	W	0	RUS	Reset UART Status (RUS) register. When set, this bit clears all bits in the USR register. This bit is automatically reset by hardware.
Offset 0x04 3004			*Clock Configuration Register (CCR)*	
7	R/W	0	SFD	When set to ‘1’ , 1 ETU = 624 clock card cycles

SMARTCARD UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
6	R/W	0	RPB	Remove Parity Bit. When this bit is set to ‘1’: • In transmission mode : the parity bit is removed (allowing a 10 ETU data frame). • In reception mode : there is no parity error check.
5	R/W	0	SHL	When the bit CST (see below) is set to 1, then clock card signal (output CLOCK_CARD) is stopped low if SHL=0, and high if SHL=1.
4	R/W	0	CST	In case of an asynchronous card, the bit CST (Clock Stop) defines whether the clock card (output pin CLK_CARD) is stopped or not.
3	R/W	0	SC	In case of a synchronous card (bit SAN in UCR2 set to 1), the output CLK_CARD is a copy of the bit SC.
2:0	R/W	0	AC2:AC0	If CST is reset (=0), then the clock card frequency (output CLK_CARD) is determined by those bits according to the following table: AC2 :AC1: AC0 = 000 : CLK_CARD = CORE_CLK/2 AC2 :AC1: AC0 = 001 : CLK_CARD = CORE_CLK/4 AC2 :AC1: AC0 = 010 : CLK_CARD = CORE_CLK/6 AC2 :AC1: AC0 = 011 : CLK_CARD = CORE_CLK/8 AC2 :AC1: AC0 =1 00 : CLK_CARD = CORE_CLK/12 AC2 :AC1: AC0 =101 : CLK_CARD = CORE_CLK/16 AC2 :AC1: AC0 =110 : CLK_CARD = CORE_CLK/24 AC2 :AC1: AC0 =111 : CLK_CARD = CORE_CLK/32 All frequency changes are synchronous, ensuring that no spike or unwanted pulse width occurs during changes.
Offset 0x04 3008			***Programmable Divider Register (PDR)***	
7:0	R/W	00000001	PD7:PD0	This register is used for counting the card clock cycles forming the ETUs. The value 00000000 should never be written in this register.
Offset 0x04 300C			***UART Configuration Register 2 (UCR2)***	
7	R/W	0	RSTIN	Card reset signal from the controller. The output pin RSTIN is a copy of this bit.
6	R/W	0	DISTBE/RBF	If bit DISTBE/RBF (disable TBE/RBF interrupt) is set, then reception or transmission of a character will not generate an interrupt.
5	R/W	0	START	If the controller sets START to 1, the card is activated. If the controller resets START to 0, the card is deactivated. The output CMDVCC = NOT (START). (For more details, see the Philips TDA8004 IC Card Interface Data Sheet.)

SMARTCARD UART 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
4	R/W	0	DMAE	If bit DMAE (DMA enable) is set (=1), DMA mode is enable; when reset, interrupt mode is active.’ The bit DMAE is automatically reset (= 0) by hardware in the following conditions: • DMA transfer successfully finished (bit DONE set in MSR) • A parity error occured during the DMA mode in T= 0 protocole (bit PE set in USR) • A change occured on input pad OFF (bit OCH set in MSR) • An error occured on the PI-bus (bit DMA_ERR set in MSR).
3	R/W	0	SAN	Synchronous/Asynchronous is set by software if a synchronous card is expected. The UART is bypassed and only bits 0 in URR and in UTR are connected to I/O. In this case, CLK_CARD output is controlled by bit SC in the CCR register.
2	R/W	0	AUTOCONV	If bit $\overline{\text{AUTOCONV}}$ is set, the convention is set by software with bit CONV in the UCR1 register. If it is reset, the convention is automatically detected on the first received character while bit SS (Start Session) is set.
1	R/W	0	CKU	When bit CKU is set, one ETU will last to half of the formula below.
0	R/W	0	PSC	If PSC is set high, the prescaler value is 32. If PSC is set low, the prescaler value is 31. One ETU will last the number of card clock cycles equal to PRESCALER x PDR.
Offset 0x04 3010			***DMA Read Address Register (DRA)***	
31:0	R/W	0	DRA[31:0]	It points at the memory at which the transfer is supposed to start when the UART is in transmission mode. This register must be programmed before setting the DMAE bit in UCR2.
Offset 0x04 3014			***Guard Time Register (GTR)***	
31:8		-	Unused	
7:0	R/W	0	GT[7:0]	This register is used to store the number of guard ETUs given by the card during ATR. In transmission mode, the UART will wait this number of ETUs before transmitting the character stored in UTR. In T=1 protocol, GTR=FF means operation at 11 ETUs. In protocol T=0, GTR=FF means operation at 12 ETUs. See Section 16.3.4 on page 16-389 for more details.
Offset 0x04 3018			***UART Configuration Register 1 (UCR1)***	
31:8		-	Unused	
7	R/W	0	RIU	Reset ISO UART ($\overline{\text{RIU}}$). This must be set to 1 before any action on the UART. When set to 0, this bit resets all UART registers to their initial value. UART is inactive if $\overline{\text{RIU}}$ = 0.
6	R/W	0	FIP	Force Inverse Parity 1 = The UART will NACK a correct received character and will transmit characters with wrong parity bits.
5	R/W	0	FC	Flow Control is set if the flow control is used.

SMARTCARD UART 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
4	R/W	0	PROT	Determines the protocol type when in asynchronous transmission. 0 = Protocol is T = 0 1 = Protocol is T = 1 In protocol T=1, if there is a parity error in reception mode the character is loaded and an interruption is generated but the DMA transfer is not aborted. In protocol T=0, when the parity error counter reaches its terminal count, the DMA transfer is aborted and the character is not loaded.
3	R/W	0	T/R	Transmit/Receive is set by software for transmission mode. A change from 0 to 1 will set the TBE bit in the USR. T/R is automatically reset by hardware if LCT has been used after transmitting the last character.
2	R/W	0	LCT	Last Character to Transmit is set by software after writing in the UART Transmit register (UTR) the last character to transmit. It allows automatic change to reception mode. Reset by hardware at the end of a successful transmission.
1	R/W	0	SS	Start Session is set before ATR for automatic convention detection and early answer detection (must be reset by software after reception of a correct initial character).
0	R/W	0	CONV	Convention is set HIGH if the convention is direct, LOW in case of inverse convention. CONV is either automatically written by hardware according to the convention detected during ATR, or by software if the AUTOCONV bit is set.
Offset 0x04 301C			***DMA Length Register (DLR)***	
31:16	R/W	0	DLW[15:0]	Determines the number of characters (bytes) to be transferred in memory (65535 bytes max) when in reception mode. This register must be programmed before setting the DMAE bit in UCR2.
15:12		-	Unused	Read and write as zeroes.
11:0	R/W	0	DLR[11:0]	Determines the number of characters (bytes) to send (4095 bytes max) to the Smartcard when in transmission mode. This register must be programmed before setting the DMAE bit in UCR2.
Offset 0x04 3020			***Time Out Configuration Register (TOC)***	
31:8		-	Unused	

SMARTCARD UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7:6	R/W	0	MODE31, MODE30	Counter3 mode control 00 = STOP: the counter stops counting. 01 = AUTORELOAD: the counter starts counting the value stored in the associated register on the first start bit after this mode has been programmed, and automatically restarts counting this value when it has reached the terminal count. An interrupt is generated at every terminal count. 10 = SOFTWARE TRIGGERED: the counter starts counting the value stored in the associated registers. When the counter reaches its terminal count, it generates an interrupt and stops. The mode bits are reset to (0,0) by hardware. 11 = TRIGGERED ON START BIT ON I/O: In this mode, the counter will automatically start counting the value stored in the associated registers when a start bit occurs on I/0 (reception or transmission). An interrupt is only generated if the counter reaches the terminal count. Then the counter is stopped. The mode bits are reset to (0,0) by hardware.
5:4	R/W	0	MODE21, MODE20	Counter2 mode control Same as the Counter3 mode control [7:6] described above.
3:2	R/W	0	MODE11, MODE10	Counter1 mode control Same as the Counter3 mode control [7:6] described above.
1:0	R/W	0	8/16, 16/24	00 = TOR3, TOR2 and TOR1 are linked as a 24-bit ETU counter 01 = TOR3 and TOR2 are linked as a 16-bit ETU counter and TOR1 is an independent 8-bit ETU counter. 1X = TOR3, TOR2 and TOR1 are 3 independent 8-bit ETU counters
Offset 0x04 3024			***Time Out Register 1 (TOR1)***	
31:8		-	Unused	
7:0	W	0	TOL[7:0]	8-bit ETU counter. Can be linked to TOR2 and TOR3 registers to form a 24-bit ETU counter (see the TOC register description for more information)
Offset 0x04 3028			***Time Out Register 2 (TOR2)***	
31:8		-	Unused	
7:0	W	0	TOL[15:8]	8-bit ETU counter. Can be linked: • to TOR1 and TOR3 registers to form a 24-bit ETU counter • to TOR3 only to form a 16-bit ETU counter (see the TOC register description for more information)
Offset 0x04 302C			***Time Out Register 3 (TOR3)***	
31:8		-	Unused	
7:0	W	0	TOL[23:16]	8-bit ETU counter. Can be linked: • to TOR1 and TOR2 registers to form a 24-bit ETU counter • to TOR2 only to form a 16-bit ETU counter (see the TOC register description for more information.)

SMARTCARD UART 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Offset 0x04 3030			*Mixed Status Register (MSR)*	
31:8		-	Unused	
7	R	0	DONE	1 = DMA transfer ended successfully. When this bit is set, INT goes high. The bit is reset when the RMS bit is set in Reset register.
6	R	0	FE	FIFO Empty 1 = The reception FIFO is empty. The bit is reset when at least one character has been loaded in the FIFO.
5	R	0	BGT	The BGT bit is linked with a 22-ETU counter, which is started at every start bit on I/O. If the count is finished before the next start bit, the BGT bit is set. This helps check that the card has not answered before 22 ETUs after the last transmitted character, or transmitted a character before 22 ETUs after the last received character. This bit is reset by setting the RMS bit in the Reset register.
4	R	0	BOF	The $\overline{\text{BOF}}$ bit is a copy of the pin $\overline{\text{OFF}}$.
3	R	0	OCH	1 = A change occurred on pin $\overline{\text{OFF}}$. When this bit is set, INT goes high. The bit is reset when the RMS bit is set in the Reset register.
2	R	0	DMA_ERR	1 = A problem occurred on the PI bus during a transaction (timeout, wrong memory address...). In that case the DMA transfer is aborted and the DMA_ABORT status bit is also set.
1	R	0	DMA_ABORT	1 = An anomaly occurred during a DMA transfer (Parity Error counter reaches its terminal count, OCH and DMA_ERROR). When DMA_ABORT is set, then the DMAE bit in UCR2 is reset (DMA transfer stopped).
0	R	0	TBE/RBF	Transmit buffer empty/receive buffer full. 1 = One of the following occurred: • Change from reception mode to transmission mode • A character has been transmitted by the UART • The reception FIFO is full The TBE/RBF bit is reset when the $\overline{\text{RIU}}$ bit is reset, or when a character has been written in UTR, or when the RUR bit in the Command Register (CRE) is set HIGH, or when changing from transmission mode to reception mode.
Offset 0x04 3030			*FIFO Control Register (FCR)*	
31:7		-	Unused	
6:4	W	0	PEC[2:0]	PEC2, PEC1 and PEC0 determine the number of accepted parity errors before setting the PE (Parity Error) bit within the UART Status Register (USR) and setting INT high in protocol T=0 000 = Only 1 parity error will be accepted 111 = 8 parity errors will be accepted
3		-	Unused	

SMARTCARD UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:0	W	0	FL[2:0]	FL[2:0] determines the depth of the FIFO 000 = Length 1 111 = Length 8 When DMA is enabled FL[2:0] must be 000.
In protocol T=0, if a correct character is received before the programmed error number is reached, then the error counter is reset. If the programmed number of allowed parity errors is reached, then the PE bit within the UART Status Register (USR) is set as long as the RUS bit in the Reset register has not been set. In protocol T=1, the error counter has no action. (PE is set at the first wrong received character). When DMA is enabled (DMAE bit set in UCR2), bits FL[2:0] must be reset.				
Offset 0x04 3034			***UART Transmit Register (UTR)***	
31:8		-	Unused	
7:0	W	0	UT[7:0]	To transmit a character to the card, the controller writes the data to this register in direct convention . The transmission: • starts at the end of this writing if the previous character has been transmitted and the extra guard time has expired • starts at the end of the extra guard time (if it has not expired) • does not start if the transmission of the previous character is not completed In synchronous mode, the UT0 data is written on the I/O line of the card when the UTR has been written and remains unchanged.
Offset 0x04 3034			***UART Receive Register (URR)***	
31:8		-	Unused	
7:0	R	0	UR[7:0]	To read data from the card, the controller reads it from this register in direct convention. When needed, this register may be tied to the FIFO whose length n is programmable between 1 and 8 (see bit FL[2:0] in FIFO Control register). Reading this register does not affect the UART, which means the TBE/RBF bits present in the USR & MSR registers are not affected and FIFO pointers are not updated. To update the UART, the controller must set the RUR bit in the Command register. In synchro- nous mode, the UR0 bit is directly linked to the I/O line of the card.
Offset 0x04 3038			***UART Status Register (USR)***	
This register is used by the controller to monitor the activity of the ISO UART and timeout counters.				
31:8		-	Unused	
7:5	R	0	TO[3:1]	Bit TO1 is set when counter1 has reached its terminal count. Bit TO2 is set when counter2 has reached its terminal count. Bit TO3 is set when counter3 has reached its terminal count.
4	R	0	EA	Early Answer is high if the first start bit on I/O DURING ATR has been detected between 200 and 384 CLK card pulses (all activities on I/O line during the first 200 CLK card pulses with RST low or high are not taken into account).
3	R	0	PE	Parity Error (see bits PEC2:PEC0 in FIFO Control register).

SMARTCARD UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2	R	0	OVR	Overrun is high if the UART has received a new character while the URR was full. In this case, at least one character has been lost.
1	R	0	FER	Framing Error is high when I/O was not in high impedance state at 10.25 ETUs after a start bit.
0	R	0	TBE/RBF	Transmission Buffer Empty is 1 when the UART is in transmission mode and when the controller may write the next character to transmit in UTR. It is reset when the controller has written data in the transmit register or when the T/R bit within UCR1 has been reset either automatically or by software. Reception Buffer Full is high when the FIFO is full. The controller may set the RUR bit in the Command register which clears the RBF bit after it has read the URR register. TBE and RBF share the same bit within the USR. (When in transmission mode, TBE is relevant; when in reception mode, RBF is relevant.)

In T=0 protocol, PE is high if the UART has detected a number of received characters with parity error equal to the number written in PEC2, PEC1 and PEC0, or if a transmitted character has been NACKed by the card.

In T=0 protocol, a character received with a parity error is not stored in the FIFO. In T=1 protocol, a character with parity error is stored in the FIFO and the parity error counter is not operating.

If any of the status bits FER, OVR, PE, EA, TO1, TO2 and TO3 is set, then INT is high. The bit having caused the interrupt is reset when the RUS bit is set in the Reset register. If TBE/RBF is set, and if the mask bit DISTBE/RBF within the UCR2 is not set, then INT is also high. TBE/RBF is reset when data has been written in the UTR, or when the RUR bit is set in the Command register, or when changing from transmission mode to reception mode.

Offset 0x04 303C			***DMA Write Address Register (DWA)***	
31:0	R/W	0	DWA[31:0]	Points to the memory address at which the transfer is supposed to start (32-bit register) when the UART is in reception mode. This register must be programmed before setting the DMAE bit in UCR2. DMA buffers have to start on a word-aligned boundary.
Offset 0x04 3040			***Command Register (CRE)***	
31:4		-	Unused	
3	W	0	RAF	Reset the DMA gate autoflush mode.
2	W	0	SDA	When this bit is set to 1, DMA gate autoflush mode is enable. (DMA gate autoflush mode : When in reception mode, the DMA gate is automatically flushed if no character have been received 100 us after the last received character).
1	W	0	FDG	Flushes the UART DMA Gate. It is automatically reset by hardware.
0	W	0	RUR	After reading the UART Receive register, the controller may set this bit in order to update: • the FIFO pointers • TBE/RBF bits in USR and MSR registers (in reception mode)
Offset 0x04 3FE0			***SCIF_INT_STATUS***	
31:12		-	Unused	
11	R	0	DONE	See description in Mixed Status Register (MSR).

SMARTCARD UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
10	R	0	OCH	See description in Mixed Status Register (MSR).
9	R	0	DMA_ERR	See description in Mixed Status Register (MSR).
8	R	0	DMA_ABORT	See description in Mixed Status Register (MSR).
7	R	0	TO3	See description in UART Status Register (USR).
6	R	0	TO2	See description in UART Status Register (USR).
5	R	0	TO1	See description in UART Status Register (USR).
4	R	0	EA	See description in UART Status Register (USR).
3	R	0	PE	See description in UART Status Register (USR).
2	R	0	OVR	See description in UART Status Register (USR).
1	R	0	FER	See description in UART Status Register (USR).
0	R	0	TBE/RBF	See description in UART Status Register (USR).
Offset 0x04 3FE4			*SCIF_INT_ENABLE*	
31:12		-	Unused	
11	R/W	0	ENA_DONE	Enable DONE interrupt.
10	R/W	0	ENA_OCH	Enable OCH interrupt.
9	R/W	0	ENA_DMA_ERR	Enable DMA_ERR interrupt.
8	R/W	0	ENA_DMA_ABORT	Enable DMA_ABORT interrupt.
7	R/W	0	ENA_TO3	Enable TO3 interrupt.
6	R/W	0	ENA_TO2	Enable TO2 interrupt.
5	R/W	0	ENA_TO1	Enable TO1 interrupt.
4	R/w	0	ENA_EA	Enable EA interrupt.
3	R/W	0	ENA_PE	Enable PE interrupt.
2	R/W	0	ENA_OVR	Enable OVR interrupt.
1	R/W	0	ENA_FER	Enable FER interrupt.
0	R/W	0	ENA_TBE/RBF	Enable TBE/RBF interrupt.
Offset 0x04 3FE8			*SCIF_INT_CLEAR*	
31:12		-	Unused	
11	W	0	CLR_DONE	Write 1 to clear DONE interrupt.
10	W	0	CLR_OCH	Write 1 to clear OCH interrupt.
9	W	0	CLR_DMA_ERR	Write 1 to clear DMA_ERR interrupt.
8	W	0	CLR_DMA_ABORT	Write 1 to clear DMA_ABORT interrupt.
7	W	0	CLR_TO3	Write 1 to clear TO3 interrupt.
6	W	0	CLR_TO2	Write 1 to clear TO2 interrupt.
5	W	0	CLR_TO1	Write 1 to clear TO1 interrupt.
4	W	0	CLR_EA	Write 1 to clear EA interrupt.
3	W	0	CLR_PE	Write 1 to clear PE interrupt.

SMARTCARD UART 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2	W	0	CLR_OVR	Write 1 to clear OVR interrupt.
1	W	0	CLR_FER	Write 1 to clear FER interrupt.
0	W	0	CLR_TBE/RBF	Write 1 to clear TBE/RBF interrupt.
Offset 0x04 3FEC			***SCIF_INT_SET***	
31:12		-	Unused	
11	W	0	SET_DONE	Set DONE interrupt.
10	W	0	SET_OCH	Set OCH interrupt.
9	W	0	SET_DMA_ERR	Set DMA_ERR interrupt.
8	W	0	SET_DMA_ABORT	Set DMA_ABORT interrupt.
7	W	0	SET_TO3	Set TO3 interrupt.
6	W	0	SET_TO2	Set TO2 interrupt.
5	W	0	SET_TO1	Set TO1 interrupt.
4	W	0	SET_EA	Set EA interrupt.
3	W	0	SET_PE	Set PE interrupt.
2	W	0	SET_OVR	Set OVR interrupt.
1	W	0	SET_FER	Set FER interrupt.
0	W	0	SET_TBE/RBF	Set TBE/RBF interrupt.
Offset 0x04 3FF4			***SCIF_POWERDOWN***	
31	R/W	0	PWDN	When set, access to Smartcard Interface registers is forbidden, except for SCIF_MODULE_ID and SCIF_POWERDOWN.
30:0		-	Unused	
Offset 0x04 3FFC			***SCIF_MODULE_ID***	
31:0	R	0x0106	MODULE_ID	Smartcard Interface ID code
15:12	R	1	REV_MAJOR	Major revision
11:8	R	0	REV_MINOR	Minor revision
7:0	R	0	APP_SIZE	Apperture size is 0 = 4 kB.

SMARTCARD UART 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
The Smartcard 2 registers, starting at 0x04 4000, are identical to the Smartcard 1 registers, described above.				

Chapter 17: DV Input Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

17.1 Introduction

The PNX8526 has three DV input interfaces (DV1, DV2 and DV3).

- The DV1 port can go to either Video Input Processor (VIP1 or VIP2). The DV1 input pins can also be used for GPIO. (Refer to Chapter 10 GPIO/IR for programming details.)
- The DV2 and DV3 ports can go to VIP1 or VIP2 or to the Transport Stream Network (TSN) for processing.

Remark: DV pins can carry only one function at a time.

DV input port routing is controlled in the I/O Mux register. (Refer to the I/O Mux register in Chapter 2 Multi-Function Pins of the PNX8526 Data Sheet for information on routing DV input.)

17.2 DV Input to VIP

Any of the three DV input ports can route digital video to VIP1 or VIP2.

The data stream may come from a device such as the SAA 7111A, which can digitize analog video from any source. The DV stream may be either 8-bit or 10-bit. Refer to Chapter 26 Video Input Processor (VIP) for details.

17.3 TS Mode of DV2 and DV3

There are two input ports available for MPEG2 or DIRECTV transport streams with identical functionality. The transport stream interface feeds the signals from those pins to the Transport Stream Network (TSN).

Each transport stream input supports 8-bit parallel, single serial or dual serial mode for data rates up to 108 Mbits/sec (108 MHz clock in serial mode). In dual serial mode, up to four input streams are supported.

Serial transport streams are converted to parallel streams in the transport stream input TSIN interface. See block diagram in Chapter 1 Functional Specification for illustration.

17.3.1 Input Signal Descriptions

The input pins and their function of the TSIN interfaces are described in the following tables for the parallel and serial modes.

The mode settings for the interface are described in Chapter 30 Transport Stream Network.

17.3.1.1 Parallel Mode

Table 1: Parallel Mode Signals

Pins		Function
DV2/ITU656	**TSIN1**	
DV2_DATA[7:0]	DATA[7:0]	These are data lines from the FEC, which `are read on the rising edge of the clock signal.
DV2_ERR	error	This line indicates an error (error = high) for a packet. It has to be stable for entire packet duration. (The usage of this signal is optional)
DV2_SOP	sync	This line indicates the start of a new MPEG2 packet. This signal is active high on the first byte of a packet.
DV2_VALID	valid	This line indicates valid data on the data input lines. A valid = high indicates valid data for the clock cycle. If valid is not supplied to the input pins, a logical '1' has to be connected.
DV2_CLOCK	clock	This clock is used to sample the data lines on the rising edge.

17.3.1.2 Serial Mode

Table 2: Serial Mode Signals

Pins		Function
DV2ITU656	**TSIN 1S1**	
DV2_DATA[0]	Data1	This line is the data line from the FEC. The data line is read on the rising edge of the clock signal.
DV2_ERR	error1	This line indicates an error (error = high) for a packet. It has to be stable for entire packet duration. (The usage of this signal is optional)
DV2_SOP	sync1	This line indicates the start of a new MPEG2 packet. This signal is active high on the first byte (bit) of a packet.
DV2_VALID	valid1	This line indicates valid data on the data input lines. A valid = high indicates valid data for the clock cycle. If valid is not supplied to the input pins, a logical '1' has to be connected.
DV2_CLOCK	clock1	This clock is used to sample the data line on the rising edge.
DV2ITU656	**TSIN 1S2**	
DV2_DATA[7]	Data2	AS DATA1
DV2_DATA[6]	error2	AS ERROR1
DV2_DATA[5]	sync2	AS SYNC1
DV2_DATA[4]	valid2	AS VALID1
DV2_DATA[3]	clock2	AS CLOCK1

The TSIN2 is connected to the DV3 pins in exactly the same way as TSIN1 is connected to DV2
(Refer to the Pin List in Chapter 1 Signal Descriptions of the PNX8526 Data Sheet).

17.3.2 Digital Video/TS Interface Timing Diagrams

For information on the functions of the IO pins, refer to the timing diagrams in AC/DC Characteristics, Chapter 3 Electrical and Mechanical Data.

Chapter 18: Transport Stream Output Port

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

18.1 Introduction

The transport stream output interface supports the common Digital Video Bus (DVB) transport stream protocols for serial and parallel data transmission. All other signals are relative to this clock.

Remark: There is no error signal output at this interface.

18.1.1 Output Signal Descriptions

Table 1: Output Signals

TSOUT	Function
TS_DATA[7:0]	These are data lines, which are valid on the rising edge of the clock signal. In case of serial mode, DATA[0] holds the serial data.
TS_SYNC	This line indicates the start of a new MPEG2 (DIRECTV) packet. This signal is active high on the first byte of a packet.
TS_CLOCK	All data and signals are valid relative to the riding edge of this clock.
TS_VALID	This line indicates valid data on the data output lines. A valid = high indicates valid data for the clock cycle.

18.1.2 Transport Stream Interface Timing Diagrams

For information on the functions of the IO Pins, refer to the following timing diagrams.

The mode settings for the interface are described in Chapter 30 Transport Stream Network. Timing is the same for TSIN1, TSIN2 and TSOUT.

For Setup and Hold Timing, refer to Chapter 3 Electrical and Mechanical Data of the PNX8526 Data Sheet.

Chapter 19: DV Output Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — October 13 2003

19.1 Introduction

The PNX8526 has two Advanced Image Composition Processors (AICPs). The video stream output of each AICP is sent to a set of pins, the DV Output Port, where it is routed to external devices. Typically, the digital video stream goes to the Analog Companion Chip, the PNX8510-11, which handles the digital-to-analog conversion and related tasks leading to the actual display.

Each DV Output Port is comprised of 10 data pins and one clock pin. (Refer to Chapter 1 Signal Descriptions and Chapter 2 Multi-Function Pins of the PNX8526 Data Sheet, for more information.) The Analog Companion Chip (PNX8510-11) has matching pins for its DV input ports. (Refer to the PNX8510-11 analog Companion Chip Data Sheet for details.)

Both the host and companion devices are capable of a variety of video formats. Which format to use will depend on the particular application and what display devices are connected. For example, use a high definition format only when an HD TV is available.

In any case, it is essential that both the host and the companion are set to the same video format.

- The PNX8526 digital video output format is controlled in the AICP module. There are two AICP modules, corresponding to DV Output Ports 1 & 2. Those ports can be used either as dedicated ports for AICP1/2 or they can be used for high data rate output modes by the AICP1 only. (Refer to Chapter 28 Advanced Image Composition Processor (AICP) for details and register descriptions.)
- The PNX8510-11 video input format is controlled in the Video In Control register. (Refer to the PNX8510-11 analog Companion Chip Data Sheet for details.)

19.2 Clocking the DV Output Stream

This section provides details on synchronizing the host output with the companion input.

The PNX8526 has three general video output modes:

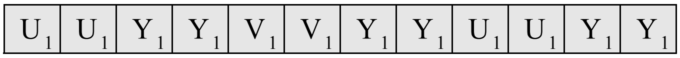
- 444 RGB
- 444YUV
- 422 YUV

All 444 formats do not go through the CVBS data path of the PNX8510-11 Analog Companion Chip. The 422 Standard Definition formats always go through the Digital Video Encoder (DENC) of the PNX8510-11 companion chip. The results of the Standard Definition clock modes for the PNX8526 are shown in Table 1.

Table 1: Standard Definition Interface Clocks

Mode #	Interface Modes	PNX8526 Processing Speed	Interface Speed
1	422 YUV input to the DENC	13.5 MHz	13.5Y, 6.75U/V = 27 MHz/54 MHz
2	444 RGB/YUV (DENC provides CVBS Sync output)	13.5 MHz	theoretical: 40.5 MHz practical = 54 MHz 3x13.5 RGB/YUV

Mode #1 describes the “normal” DENC 27 MHz output mode. A clock of 27 MHz for the 422 YUV output to the DENC is sufficient in terms of quality. However, a 54 MHz DAC clock may be desirable to achieve 4x oversampling of the video data. The PNX8510-11 27 MHz clock can be easily derived from the 54 MHz interface clock with “divide by 2.” The output data stream would look as shown in Figure 1.

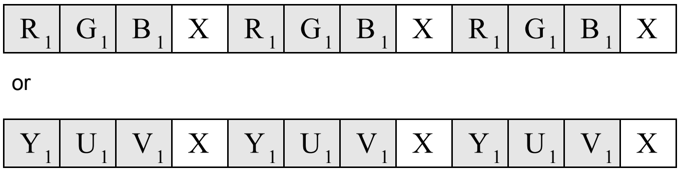


U_1	U_1	Y_1	Y_1	V_1	V_1	Y_1	Y_1	U_1	U_1	Y_1	Y_1

Figure 1: Single D1 4:2:2 Stream 54 MHz

All components are doubled, which does not introduce more complexity to the interface.

In 444 RGB/YUV (mode #2) the “natural” clock frequency would be 40.5 MHz. By slightly modifying the data stream, the output clock could be set to 54 MHz. That will result in an easier clock divide (divide by 2 for 27 MHz processing or 4 for 13.5) and would open the opportunity to do 4x oversampling for those modes as well. The data stream is shown in Figure 2



R_1	G_1	B_1	X	R_1	G_1	B_1	X	R_1	G_1	B_1	X

or

Y_1	U_1	V_1	X	Y_1	U_1	V_1	X	Y_1	U_1	V_1	X

Figure 2: Serial RGB/YUV 4:4:4 Stream 54 MHz

The “X” represents a “don’t care” data slot. Because current video encoders don’t support 444 RGB/YUV input streams, there are no legacy issues. The contents of the pins at this time is most likely the previous value. However to have the possibility to avoid high interface speeds, a divide by 3 might be needed for a 40.5 MHz input clock. For that mode, the output stream would look like Figure 3.

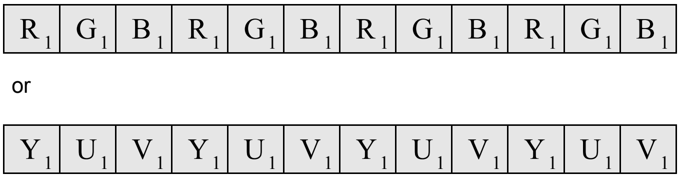

R_1 G_1 B_1 R_1 G_1 B_1 R_1 G_1 B_1 R_1 G_1 B_1
or
Y_1 U_1 V_1 Y_1 U_1 V_1 Y_1 U_1 V_1 Y_1 U_1 V_1

Figure 3: Serial RGB/YUV Stream 40.5 MHz

Aside from the standard definition modes, the PNX8526 provides support for High Definition (HD) modes. The supported modes are described in Table 2.

Table 2: HD Interface Clocks

Mode#	Interface Modes	PNX8526 Processing Speed	Interface Speed
3	444YUV/RGB 480P	27 MHz	3x27 MHz serial = 81 MHz
4	422 YUV, PNX8510-11 internal 422 to 444 conversion, parallel output, usage of two interfaces	XMHz	XMHz max 81 MHz (see examples below)
5	RGB 24-bit mode, parallel output, usage of two interfaces	XMHz	XMHz max 81 MHz (see examples below)

Serial pixel rates up to 81 MHz are supported, which covers 480 progressive (mode #1).

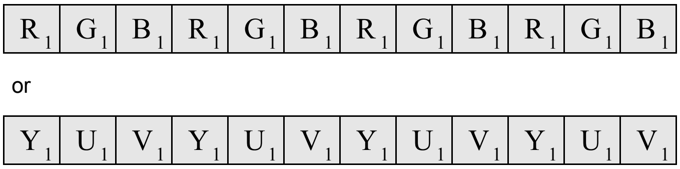

R_1 G_1 B_1 R_1 G_1 B_1 R_1 G_1 B_1 R_1 G_1 B_1
or
Y_1 U_1 V_1 Y_1 U_1 V_1 Y_1 U_1 V_1 Y_1 U_1 V_1

Figure 4: 480P Serial 4:4:4 Data Stream

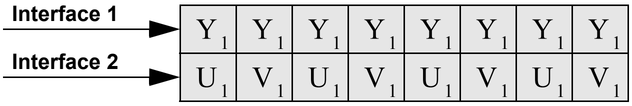
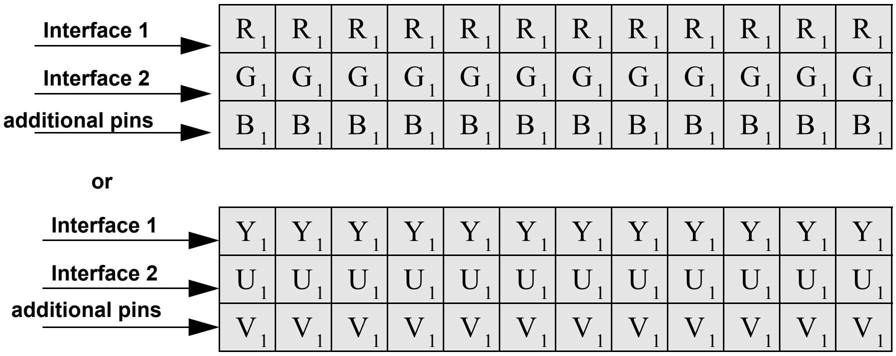

Interface 1
Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1
Interface 2
U_1 V_1 U_1 V_1 U_1 V_1 U_1 V_1

Figure 5: Parallel 4:2:2 HD Data Stream (uses two interfaces)

All pixel rates that result in an interface frequency beyond 81 MHz (if transferred in serialized form) will use two physical interfaces. YUV 422 is preferred to utilize the 10-bit data path capabilities.


Interface 1
Interface 2
additional pins
R_1 R_1 R_1 R_1 R_1 R_1 R_1 R_1 R_1 R_1 R_1 R_1
G_1 G_1 G_1 G_1 G_1 G_1 G_1 G_1 G_1 G_1 G_1 G_1
B_1 B_1 B_1 B_1 B_1 B_1 B_1 B_1 B_1 B_1 B_1 B_1
or
Interface 1
Interface 2
additional pins
Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1 Y_1
U_1 U_1 U_1 U_1 U_1 U_1 U_1 U_1 U_1 U_1 U_1 U_1
V_1 V_1 V_1 V_1 V_1 V_1 V_1 V_1 V_1 V_1 V_1 V_1

Note: Uses two interfaces +, components 8 instead of 10-bit

Figure 6: Parallel 4:4:4 HD Data Stream

Example:

1080i: 75 MHz Y
75 MHz chroma
->
75 MHz interface speed
pixel clock at DAC's 75 MHz, no oversampling

1024x768 24 bit RGB mode, 75Hz refresh:
78MHz R, G, B -> 8bit each component
->
interface speed 78 MHz
pixel clock at DAC’s 78 MHz, no oversampling

Aside from those single stream modes, the interleaved modes shown in Table 3 are supported.

Table 3: Interleaved Modes, Internal/Interface Clocks

Mode #	Interface Modes	PNX8526 Processing Speed	Interface Speed
6	2x27MHz D1 (422)	13.5 MHz	2x(13.5Y, 6.75U/V) = 54 MHz (natural clock see to #1
7	444RGB+422YUV (422 is transferred as 444)	13.5 MHz	2x40.5 MHz (3x13.5) = 81 MHz
8	2x444RGB requires two PNX8510-11!	13.5 MHz	2x40.5 MHz (3x13.5) = 81 MHz

The interleaved modes described above require certain data stream manipulations in the PNX8510-11 input interface. Mode #6 is straightforward—it's like #1 with a 54 MHz interface speed. The data stream is shown in Figure 7.

U_1	U_2	Y_1	Y_2	V_1	V_2	Y_1	Y_2	U_1	U_2	Y_1	Y_2	V_1	V_2

Figure 7: Two Interleaved D1 YUV 4:2:2 Streams (typical = 54 MHz)

That means slice one is always stream 1, slice two is always stream 2. Mode #7 requires dummy data in the 422 stream to keep the two streams balanced. It will look like the following:

R_1	Y_2	G_1	U_2	B_1	V_2	R_1	Y_2	G_1	U_2	B_1	V_2

or

Y_1	Y_2	U_1	U_2	V_1	V_2	Y_1	Y_2	U_1	U_2	V_1	V_2

Figure 8: Two Interleaved RGB/YUV 4:4:4 Streams (typical = 81 MHz)

Stream #2 will have doubled chrominance data, which have to be subsampled in the PNX8510-11 to provide a 422 "uYvYuYvY... " stream to the DENC. A 444 data stream, in either RGB or YUV from stream 1 will bypass the PNX8510-11 DENC and go out directly to the DACs. (This is the concurrent 444 RGB + 422 YUV same contents, single 10-bit interface mode, with SCART and CVBS at the same time.)

Mode #8, a 2x444 RGB high quality mode will require two PNX8510-11 chips, because the PNX8510-11 has only one RGB data path.

19.2.1 PNX8526 Interface Clock Speeds

The PNX8526 covers four interface clock speeds:

- 27 MHz single D1
- 54 MHz for 2xD1, accommodates also 40.5 for 444 3x13.5 with RGBx or YUVx as well as 2x D1 interleaved
- 81 MHz for interleaved 444 formats

19.3 DV Out Pin Descriptions

The following table lists all DV Out signals and various modes of operation. The DV Out pins can be configured for two independent DV Out streams or operate together as a single 24-bit RGB (with additional bits multiplexed on Audio Out2). See Chapter 2 Multi-Function Pins of the PNX8526 Data Sheet.

Table 4: DV Output Signals

Internal AICP Signal	Chip Port	Type	Mode Description
	AICP 1		
clk_icp_mux1	DV_CLK1	O	AICP interface clock 1
clk_icp_mux2	DV_CLK2	O	AICP interface clock 2
icp_d1_out1	DV_OUT1 [9:0]	O	Single D1 output mode for primary AICP output
icp_d1_out2	DV_OUT2 [9:0]	O	Single D1 output mode for secondary AICP output
icp_24bit_out[9:0]	DV_OUT1 [9:0]	O	24-bit parallel RGB/YUV mode, bit [9:0]
icp_24bit_out[19:10]	DV_OUT2 [19:10]	O	24-bit parallel RGB/YUV mode, bit [19:10]
icp_24bit_out[23]	AO2_SDOUT	O	24-bit parallel RGB/YUV mode, bit 23
icp_24bit_out[22]	AO2_WSOUT	O	24-bit parallel RGB/YUV mode, bit 22
icp_24bit_out[21]	AO2_SCKOUT	O	24-bit parallel RGB/YUV mode, bit 21
icp_24bit_out[20]	AO2_OSCLK	O	24-bit parallel RGB/YUV mode, bit 20
icp_hd_y_out	DV_OUT2 [9:0]	O	Double D1 output mode for HD 4:2:2 YUV video
icp_hd_uv_out	DV_OUT1 [9:0]	O	Double D1 output mode for HD 4:2:2 YUV video
icp_hsync	HSYNC	O	Horizontal sync signal
icp_vsync	VSYNC	O	Vertical sync signal
icp_cblank	BLANK	O	Composite blank signal
icp_slice_qualifier	HSYNC	O	The slice qualifier to distinguish between 2 D1 slices
trigger	VSYNC	I	External field/frame based trigger (odd/even signal is assumed)

Chapter 20: Audio Input Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

20.1 Introduction

The PNX8526 has three Audio Input ports, identified as Audio Input 1, 2 and 3. Each audio in port supports single or dual-channel sources. All three Audio Input ports are driven by identical Audio In modules.

20.1.1 Audio In Overview

The Audio In module provides a DMA-driven serial interface to an off-chip stereo A/D converter, I^2S subsystem or other serial data source. Audio In provides all signals needed to connect to high quality, low cost oversampling A/D converters. It includes a programmable clock generator for a precise oversampling A/D system clock.

The Audio In module and external A/D converter (or I^2S subsystem) together provide the following capabilities:

- One or two channels of audio input per port
- 8 or 16-bit samples per channel
- Programmable 1 Hz to 100 kHz sampling rate

Remark: This is a practical range. The actual sample rate is application dependent.

- Internal or external sampling clock source
- Audio In autonomously writes sampled audio data to memory using double buffering (DMA)
- 8-bit and 16-bit mono and stereo PC standard memory data formats
- Little or big-endian memory formats set by Global Registers.

The Audio In module can be used with many serial A/D converter devices, including the Philips SAA7366 (stereo A/D), UDA1360, Crystal Semiconductor CS5331, or CS5336 (stereo A/Ds).

20.2 Functional Description

20.2.1 Overview

The Audio In module has three major subsystems: a programmable sample clock generator, a serial-to-parallel converter, and a DMA engine.

The sampling clock can be used as either master or slave to the external A/D device. The sampling clock synchronizes the serial-to-parallel converter with the source data stream. The samples enter the serial-to-parallel converter, which reformats the data for the DMA engine. The DMA engine uses double-buffering to write mono/stereo 8 to 16-bit samples to memory. The Audio In is compatible with the TM3100 and software can be easily ported.

20.2.2 External Interface

The Audio In has four pins. The OSCLK is a precise, programmable clock output intended to serve as the master system clock for the external A/D subsystem. Three other pins (SCK, WS and SD) constitute a flexible serial input interface. Using the Audio In MMIO registers, these pins can be configured to operate in a variety of serial interface framing modes, including but not limited to the following:

- Standard stereo I^{2}S (msb first, 1-bit delay from WS, left and right data in a frame). (For further details on I^{2}S, refer to the “I^{2}S Bus Specification” dated June 5 1996, in the *Multimedia ICs Data Handbook IC22* by Philips Semiconductors, 1998.)
- lsb first with 1–16 bit data per channel
- Complex serial frames of up to 512 bits/frame with “valid sample” qualifier bit.

Table 1: Audio-In Unit External Signals

Signal	Type	Description
I^{2}S_IN1_OSCLK I^{2}S_IN2_OSCLK I^{2}S_IO_OSCLK	OUT	Oversampling Clock. This output can be programmed to emit any frequency up to 40 MHz with a resolution of better than 0.3 Hz. It is intended for use as the 256 f_s or 384 f_s oversampling clock by external A/D subsystem.
I^{2}S_IN1_SCK I^{2}S_IN2_SCK I^{2}S_IO_SCK	I/O	When Audio In is programmed as the serial-interface timing slave (power-up default), SCK is an input. SCK receives the serial bit clock from the external A/D subsystem. This clock is treated as fully asynchronous to the main chip level clock. When Audio In is programmed as the serial-interface timing master, SCK is an output. SCK drives the serial clock for the external A/D subsystem. The frequency is a programmable integral divide of the OSCLK frequency. SCK is limited to 30 MHz. The sample rate of valid samples embedded within the serial stream is limited to 100 kHz.
I^{2}S_IN1_SD I^{2}S_IN2_SD I^{2}S_IO_SD[0]	IN	Serial Data from external A/D subsystem. Data on this pin is sampled on positive or negative edges of SCK as determined by the CLOCK_EDGE bit in the AI_SERIAL register.
I^{2}S_IN1_WS I^{2}S_IN2_WS I^{2}S_IO_WS	I/O	When Audio In is programmed as the serial-interface timing slave (power-up default), WS acts as an input. WS is sampled on the same edge as selected for SD. When Audio In is programmed as the serial-interface timing master, WS acts as an output. It is asserted on the opposite edge of the SD sampling edge. WS is the word-select or frame-synchronization signal from/to the external A/D subsystem.
Note: The pins for Audio Input port 3 (I^{2}S_IO_) have multiple functions. (Refer to Chapter 10 GPIO/IR for details on selecting the function.) This table assumes the pins are configured for Audio In.		

20.2.3 Clock System

Figure 1 illustrates the clocking capabilities of the Audio In unit. Driving the system is a square wave Direct Digital Synthesizer (DDS). The DDS can be programmed to emit frequencies from approximately 1 Hz to 40 MHz with a resolution of better than 0.3 Hz.

The DDS and its control registers reside in the Clocks module. Refer to the clocks section in Chapter 5 Clock Reset and Power Management for details.

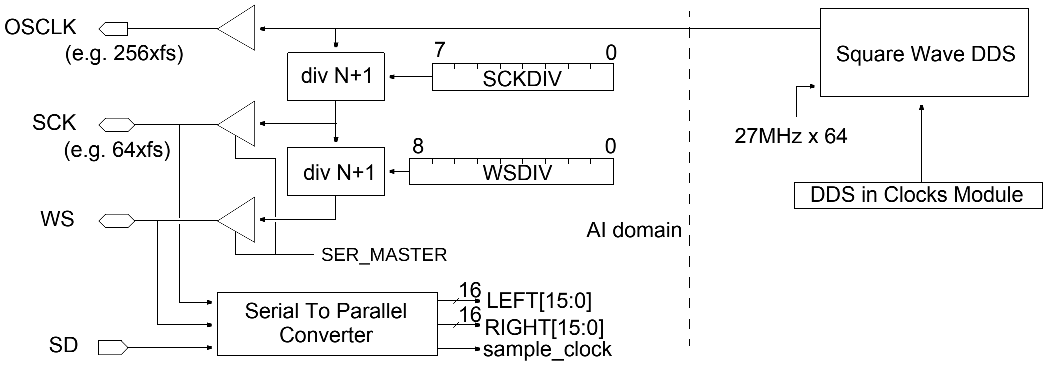


Figure 1: Audio In Clock System and I/O Interface

The output of the DDS is always sent on the OSCLK output pin. This output is intended to be used as the 256 f_s or 384 f_s system clock source for oversampling A/D converters.

Software may change the DDS frequency setting dynamically, so as to adjust the input sampling rate to track an application dependent master reference. Using the DDS function, a high quality, low-jitter OSCLK is generated.

20.2.3.1 Clock System Operation

SCK and WS can be configured as input or output, as determined by the SER_MASTER control field. As an output, SCK is a divided form of the OSCLK output frequency. The SCKDIV register value is used to divide down the OSCLK frequency. See Section 20.4 on page 20-425 for more details. Whether input or output, the SCK pin signal is used as the bit clock for serial-parallel conversion. The value of SCKDIV is determined by the equation:

$$f_{AISCK} = \frac{f_{AIOSCLK}}{SCKDIV + 1}$$

Remark: SCKDIV is in the range 0-255.

If set as output, WS can similarly be programmed using WSDIV to control the serial frame length from 1 to 512 bits. The number of bits per frame is equal to WSDIV + 1. Table 2 presents several sample rates with the appropriate SCKDIV necessary to achieve a bit clock of 64 f_s.

Table 2: Sample Rate Settings

f_s	OSCLK	SCKDIV	SCK
44.1 kHz	256 f_s	3	64 f_s

Table 2: Sample Rate Settings *…Continued*

f_s	OSCLK	SCKDIV	SCK
48.0 kHz	256 f_s	3	64 f_s
44.1 kHz	384 f_s	5	64 f_s
48.0 kHz	384 f_s	5	64 f_s

The preferred application of the clock system options is to use OSCLK as A/D master clock, and let the A/D converter be timing master over the serial interface (SER_MASTER = 0).

In case of an external codec for common Audio In and Audio Out use, it may not be possible to independently control the A/D and D/A system clocks. It is recommended that the Audio Out clock system DDS be used to provide a single master A/D and D/A clock. Audio Out or the D/A converter can be used as serial interface timing master, and Audio In can be set as slave to the serial frame determined by Audio Out (Audio In SER_MASTER = 0, SCK and WS externally wired to the corresponding Audio Out pins). In such systems, independent software control over A/D and D/A sampling rate is not possible, but component count is minimized.

20.3 Operation

20.3.1 Overview

The Audio In unit is reset by a chip level hardware reset or by writing logic ‘1’ to the I2S_IN_CTL.RESET register bit. Upon RESET, capture is disabled (CAP_ENABLE = 0), and buffer1 is the active buffer (BUF1_ACTIVE = 1). A minimum of five valid SCK clock cycles is required to allow internal Audio In circuitry to stabilize before enabling capture. This can be accomplished by programming AI DDS control register(s) and AI_SERIAL, then delaying for the appropriate time interval.

Software initiates capture by providing two equal size empty buffers and putting their base address and size in the BASE1, BASE2 and SIZE registers. Once two valid (local memory) buffers are assigned, capture can be enabled by writing a ‘1’ to CAP_ENABLE. The Audio In unit hardware will proceed to fill buffer 1 with input samples. Once buffer 1 fills up, BUF1_FULL is asserted, and capture continues without interruption in buffer 2. If BUF1_INTEN is enabled, a level triggered interrupt request is generated to the chip level interrupt controller.

Remark: The buffers must be 64-byte aligned and be a multiple of 64 samples in size (the six lsbits of AI_BASE1, AI_BASE2 and AI_SIZE are always zero).

Software is required to assign a new, empty buffer to BASE1 and perform an ACK1, before buffer 2 fills up. Capture continues in buffer 2, until it fills up. At that time, BUF2_FULL is asserted and capture continues in the new buffer 1, etc.

Upon receipt of an ACK, the Audio In hardware removes the related interrupt request line assertion at the next main clock edge.

In normal operation, the chip level system controller and Audio In hardware continuously exchange buffers without ever losing a sample. If the system controller fails to provide a new buffer in time, the OVERRUN error flag is raised. This flag is *not affected* by ACK1 or ACK2; it can only be cleared by an explicit write of logic ‘1’ to ACK_OVR.

Remark: Reserved bits in MMIO registers should be ignored when read and written as zeroes. See Section 20.4.

20.3.2 Serial Data Framing

The Audio In unit can accept data in a wide variety of serial data framing conventions. Figure 2 illustrates the notion of a serial frame. If POLARITY = 1 and CLOCK_EDGE = 0, a frame is defined with respect to the positive transition of the WS signal as observed by a positive clock transition on SCK. (See Section 20.4.) Each data bit sampled on positive SCK transitions has a specific bit position—i.e., once the clock edge detects the WS transition, the next sample will be data bit position 0.

Each subsequent clock edge defines a new bit position. Other combinations of POLARITY and CLOCK_EDGE can be used to define a variety of serial frame bit position definitions. (See Section 20.4 on page 20-425.)

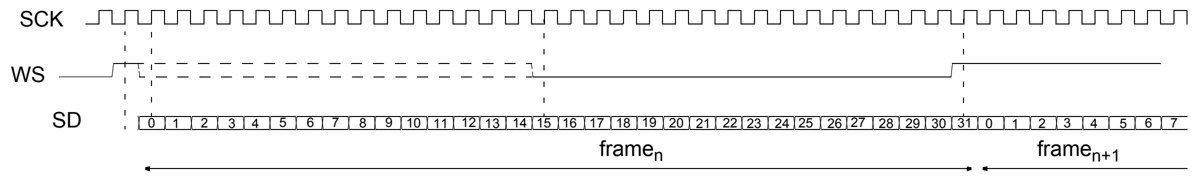


Figure 2: Audio In Serial Frame and Bit Position Definition (POLARITY = 1, CLOCK_EDGE = 0)

The capturing of samples is governed by FRAMEMODE. If FRAMEMODE = 00, every serial frame results in one sample from the serial-parallel converter. A sample is defined as a left/right pair in stereo modes or a single left channel value in mono modes. If FRAMEMODE = 1y, the serial frame data bit in bit position VALIDPOS is examined. If it has value ‘y’, a sample is taken from the data stream (the valid bit is allowed to precede or follow the left or right channel data provided it is in the same serial frame as the data).

The left and right sample data can be in a lsb-first or msb-first form at an arbitrary bit position and with an arbitrary length. (See Section 20.4.) In msb-first mode, the serial-to-parallel converter assigns the value of the bit at LEFTPOS to LEFT[15]. Subsequent bits are assigned, in order, to decreasing bit positions in the LEFT data word, up to and including LEFT[SSPOS]. Bits LEFT[SSPOS–1:0] are cleared. Hence, in msb-first mode, an arbitrary number of bits are captured. They are left-adjusted in the 16-bit parallel output of the converter.

In lsb-first mode, the serial to parallel converter assigns the value of the bit at LEFTPOS to LEFT[SSPOS]. Subsequent bits are assigned, in order, to increasing bit positions in the LEFT data word, up to and including LEFT[15]. Bits LEFT[SSPOS–1:0] are cleared. Hence, in lsb-first mode, an arbitrary number of bits are captured. They are returned left-adjusted in the 16-bit parallel output of the converter.

See Figure 3 and Table 3 for an example of how the Audio In module registers are set to collect 16-bit samples using the Philips SAA7366 I^2S 18-bit A/D converter. (See Section 20.4.) The setup assumes the SAA7366 acts as the serial master.

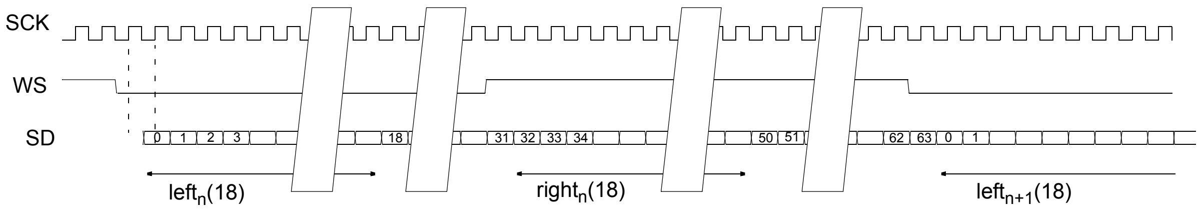


Figure 3: Serial Frame of the SAA7366 18-Bit I^2S A/D Converter (Format 2 SWS)

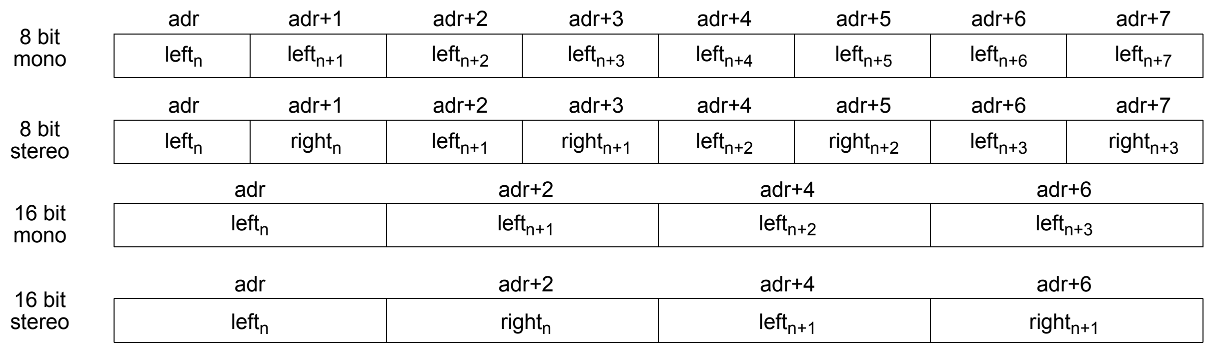
For example, if it were desired to use only the 12 msbits of the A/D converter in Figure 3, use the settings of Table 3 with SSPOS set to four. This results in LEFT[15:4] being set with data bits 0..11 and LEFT[3:0] being set equal to zero. RIGHT[15:4] is set with data bits 32..43 and RIGHT[3:0] is set to zero.

Table 3: Example Setup For SAA7366

Field	Value	Explanation
SER_MASTER	0	SAA7366 is serial master
SCKDIV	3	SCK set to OSCLK/4 (not needed since SER_MASTER = 0)
WSDIV	63	Serial frame length of 64 bits (not needed since SER_MASTER = 0)
POLARITY	0	Frame starts with negative WS
FRAMEMODE	00	Take a sample each serial frame
VALIDPOS	n/a	Don’t care
LEFTPOS	0	Bit position 0 is msb of left channel and will go to LEFT[15]
RIGHTPOS	32	Bit position 32 is msb of right channel and will go to RIGHT[15]
DATAMODE	0	msb first
SSPOS	0	Stop with LEFT/RIGHT[0]
CLOCK_EDGE	0	Sample WS and SD on positive SCK edges for I^2S

20.3.3 Memory Data Formats

The Audio In unit autonomously writes samples to memory in mono and stereo 8 and 16-bit per sample formats, as shown in Figure 4. Successive samples are always stored at increasing memory address locations.



8 bit mono

adr	adr+1	adr+2	adr+3	adr+4	adr+5	adr+6	adr+7
left$_n$	left$_{n+1}$	left$_{n+2}$	left$_{n+3}$	left$_{n+4}$	left$_{n+5}$	left$_{n+6}$	left$_{n+7}$

8 bit stereo

adr	adr+1	adr+2	adr+3	adr+4	adr+5	adr+6	adr+7
left$_n$	right$_n$	left$_{n+1}$	right$_{n+1}$	left$_{n+2}$	right$_{n+2}$	left$_{n+3}$	right$_{n+3}$

16 bit mono

adr	adr+2	adr+4	adr+6
left$_n$	left$_{n+1}$	left$_{n+2}$	left$_{n+3}$

16 bit stereo

adr	adr+2	adr+4	adr+6
left$_n$	right$_n$	left$_{n+1}$	right$_{n+1}$

Figure 4: Audio In Memory DMA Formats

20.3.3.1 Endian Control

The Audio In block imports a chip level endianness control signal and will slave to this signal for normal capture operation. This global “big_endian” control bit resides in the RST_CTL register (refer to PNX8526 Register Summary) and determines how increasing memory addresses map to byte positions within audio words. The Audio In unit power-on default state uses this “global” system endian control input.

20.3.4 Memory Buffers and Capture

The Audio In unit hardware implements a double buffering scheme to ensure that no samples are lost, even if the chip level controller is highly loaded and slow to respond to interrupts. The software assigns buffers by writing a base address and size to the MMIO control fields (see Section 20.4 on page 20-425). Refer to Section 20.3 for details on hardware/software synchronization.

In 8-bit capture modes, the eight msbits of the serial-to-parallel converter output data are written to memory. In 16-bit capture modes, all bits of the parallel data are written to memory. If SIGN_CONVERT is set to one, the msb of the data is inverted, which is equivalent to translating from two’s complement to offset binary representation. This allows the use of an external two’s complement 16-bit A/D converter to generate eight-bit unsigned samples, which is often used in PC audio.

Remark: The Audio In hardware does *not* generate A-law or µ-law 8-bit data formats. If such formats are desired, additional processing is necessary, via software, to convert from 16-bit linear data to A-law or µ-law data.

20.3.5 Data Bus Latency and HBE

Audio Out uses two 64-byte internal buffers to capture the audio input samples. When one 64-byte buffer is full, the next incoming samples are stored in the second 64-byte buffer. The DMA unit starts the storing process to main memory as soon as one 64-byte buffer is full. One 64-byte buffer holds 16 “two by 16-bit” stereo samples. Under normal operation, the 64-byte buffer gets written to memory while the second 64-byte buffer is capable of receiving sixteen more samples. This normal operation will be maintained as long as the data bus arbiter is set to guarantee a latency for Audio In that matches the incoming audio samples rate.

Given a sample rate f_s, and an associated sample interval T (in nSec), the bus latency should be at most 16*T nSec. If this max latency is not satisfied, the HBE (Bandwidth Error) condition will result. This error flag gets set when one of the two internal 64-byte buffers is not free and a new sample arrives for that buffer. Table 4 shows the required data bus arbitration latency requirements for a number of common operating modes. The right column in Table 4 shows the nature of the resulting 64-byte burst data bus requests.

Table 4: Audio In Data Bus Arbiter Latency Requirement Examples

CapMode	f_s (kHz)	T (nS)	Max Arbiter Latency (16*T) (uSec)	Access Pattern
Stereo 2x16-bit/sample	44.1	22,676	362.816	1 64-byte request minimally every 362.816 uS
Stereo 2x16-bit/sample	48.0	20,833	333.328	1 64-byte request minimally every 333.328 uS
Stereo 2x16-bit/sample	96.0	10,417	166.672	1 64-byte request minimally every 166.672 uS

20.3.6 Error Behavior

If either an OVERRUN or HBE error occurs, input sampling is temporarily halted and incoming samples will be lost. In the case of OVERRUN, sampling resumes as soon as the control software makes one or more new buffers available through an ACK1 or ACK2 operation. In the case of HBE, sampling will resume as soon as the internal buffer can be written to memory. HBE and OVERRUN are ‘sticky’ error flags meaning they will remain set until an explicit software write of logic ‘1’ to ACK_HBE or ACK_OVR is performed. See Section 20.4.

20.3.7 Interrupts

The AI_STATUS register provides all sources of Audio In generated interrupt: BUF1_FULL, BUF2_FULL, HBE and OVERRUN. All interrupts sourced by Audio In to the chip level interrupt controller are level triggered. An interrupt will be generated from Audio In only if the corresponding interrupt enable bit is set in the AI_CTL register. For example, to assert an interrupt to the system upon the occurrence of a bandwidth error (HBE asserted), set the HBE_INTEN bit to logic ‘1’. See Section 20.4.

Interrupt status bits within AI_STATUS are persistent, meaning that once the interrupt is triggered, it will remain asserted until cleared by setting the corresponding ACK bit in AI_CTL. Using the above example, assuming the HBE interrupt is asserted, write a logic ‘1’ to ACK_HBE to clear the AI_STATUS.HBE bit and deactivate the interrupt to the system.

Remark: The Audio In module is based upon a legacy function and, as such, is not strictly compliant with DVP module interrupt conventions.

20.3.8 Timestamp Events

Audio In exports event signals associated with audio capture to the central timestamp/timer function on-chip. The central timestamp/timer function can be used to count the number of occurrences of each event or timestamp the occurrence of the event or both. The event will be a positive edge pulse with the duration of the event to be greater than or equal to 200 ns. The specific event exported is:

- The TSTAMP event will occur when the last sample in the internal Audio In buffer is written. One precise event will occur for each of the two DMA buffers.
- Audio In 3 only: The WS event indicates the arrival of a particular audio frame on the I^2S interface. Timer Mux Control in the Global 1 Registers (See Global Regsisters in Chapter 25 Internal TM32 CPU Core Processor) can select AIO3's WS for timestamping.

The occurrence of this event represents a precise, periodic time interval which can be used by system software for audio/video synchronization.

20.3.9 Diagnostic Mode

This mode can be used during the diagnostic phase of system testing to verify the correct operation of both Audio In and Audio Out units. This loopback mode internally feeds the I2S Audio Out to the I2S Audio In. This test can be used to check the data flow from the Audio Out buffer to the Audio In buffer.

Special care must be taken to enable diagnostic mode. The recommended way of entering diagnostic mode is as follows:

1. Configure Audio Out to generate AO_SCK.
2. Set the AI_CTL.DIAGMODE bit followed by a five (SCK) cycle delay.
3. Perform a software reset of Audio In and immediately set back the DIAGMODE bit to logic ‘1’. Then enable Audio In capture.

Audio In Operation

The Audio (I2S) Input Ports Registers (Section 20.4) describe the function of the control and status fields of the AI unit. To ensure compatibility with future devices, undefined bits in MMIO registers should be ignored when read and written as 0s.

The AI unit is reset by a PNX8526 hardware reset, or by writing 0x80000000 to the AI_CTL register. Upon RESET, capture is disabled (CAP_ENABLE = 0), and buffer1 is the active buffer (BUF1_ACTIVE = 1). A minimum of 5 valid clock cycles is required to allow internal AI circuitry to stabilize before enabling capture. This can be accomplished by programming AI_FREQ and AI_SERIAL and then delaying for the appropriate time interval.

Programming the AI_SERIAL MMIO register must follow this sequence:

- Set AI_FREQ to ensure that a valid clock is generated (only when AI is the master of the audio clock system).
- MMIO(AI_CTL) = 1<<31; /* Software Reset */

- MMIO(AI_SERIAL) = 1<<31; /* sets serial-master mode, starts AI_SCK */
- MMIO(AI_SERIAL) = (1<<31) | (SCKDIV value); /* then set DIVIDER values */

The TM32 CPU core initiates capture by providing two equal size empty buffers and putting their base address and size in the BASEn and SIZE registers. Once two valid (local memory) buffers are assigned, capture can be enabled by writing a '1' to CAP_ENABLE. The AI unit now proceeds to fill buffer 1 with input samples. Once buffer 1 fills up, BUF1_FULL is asserted, and capture continues without interruption in buffer 2. If BUF1_INTEN is enabled, a SOURCE 11 interrupt request is generated.

Remark: The buffers must be 64-byte aligned and multiple of 64 samples in size (the six lsbits of AI_BASE1, AI_BASE2 and AI_SIZE are always '0').

The TM32 CPU is required to assign a new, empty buffer to BASE1 and perform a ACK1 before buffer 2 fills up. Capture continues in buffer 2 until it fills up. At that time, BUF2_FULL is asserted and capture continues in the new buffer 1.

Upon receipt of an ACK, the AI hardware removes the related interrupt request line assertion at the next TM32 CPU clock edge. The AI interrupt should always be operated in level-sensitive mode since AI can signal multiple conditions that each need independent ACKs over the single internal SOURCE 11 request line.

In normal operation, the TM32 CPU and AI hardware continuously exchange buffers without ever losing a sample. If the TM32 CPU fails to provide a new buffer in time, the OVERRUN error flag is raised. The flag is not affected by ACK1 or ACK2; it can only be cleared by an explicit ACK-OVR.

20.4 Register Descriptions

The PNX8526 has three Audio Input modules. The registers for Audio In 1 begin at 0x11 1000. The registers for Audio In 2 begin at 0x11 3000. The registers for Audio In 3 begin at 0x11 5000. The register sets for the three modules are identical, so only the Audio In 1 registers are detailed in the following tables.

20.4.1 Register Address Map

Table 5: Audio Input Ports 1-3, Register Summary

Offset	Name	Description
Audio Input 1 (I2S_IN1)		
0x11 1000	AI_STATUS	Provides status of Audio In components/situations.
0x11 1004	AI_CTL	Control register to configure Audio In options.
0x11 1008	AI_SERIAL	Control register to configure Audio In serial timing and data options.
0x11 100C	AI_FRAMING	Control register to configure data framing format.
0x11 1010	Reserved	
0x11 1014	AI_BASE1	Base address of buffer 1.
0x11 1018	AI_BASE2	Base address of buffer 2.
0x11 101C	AI_SIZE	The DMA Buffer size in samples.
0x11 1020—1FF0	Reserved	
0x11 1FF4	AI_PWR_DWN	Powerdown function. Not implemented.
0x11 1FFC	AI_MODULE_ID	Module ID number, including major and minor revision levels.
Audio Input 2 (I2S_IN2)		
0x11 3000	AI_STATUS	Provides status of Audio In components/situations.
0x11 3004	AI_CTL	Control register to configure Audio In options.
0x11 3008	AI_SERIAL	Control register to configure Audio In serial timing and data options.
0x11 300C	AI_FRAMING	Control register to configure data framing format.
0x11 3010	Unused	
0x11 3014	AI_BASE1	Base address of buffer 1.
0x11 3018	AI_BASE2	Base address of buffer 2.
0x11 301C	AI_SIZE	The DMA Buffer size in samples.
0x11 3020-3FF0	Reserved	
0x11 3FF4	AI_PWR_DWN	Powerdown function. Not implemented.
0x11 3FFC	AI_MODULE_ID	Module ID number, including major and minor revision levels.
Audio Input 3 (I2S_IO)		
0x11 5000	AI_STATUS	Provides status of Audio In components/situations.
0x11 5004	AI_CTL	Control register to configure Audio In options.
0x11 5008	AI_SERIAL	Control register to configure Audio In serial timing and data options.
0x11 500C	AI_FRAMING	Control register to configure data framing format.
0x11 5010	Unused	
0x11 5014	AI_BASE1	Base address of buffer 1.
0x11 5018	AI_BASE2	Base address of buffer 2.
0x11 501C	AI_SIZE	The DMA Buffer size in samples.
0x11 5020-5FF0	Reserved	
0x11 5FF4	AI_PWR_DWN	Powerdown function. Not implemented.
0x11 5FFC	AI_MODULE_ID	Module ID number, including major and minor revision levels.

AUDIO (I2S) INPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Audio In 1 Registers (Offset 0x11 1000), Audio In 2 Registers (Offset 0x11 3000) and Audio In 3 Registers (Offset 0x11 5000) The Audio In 1, 2 and 3 registers are identical except for their offsets. A triplicate set has been created to facilitate programming. The tables for Audio In 2 (0x11 3000) and 3 (0x11 5000) registers follow this table.				
Note: The clock frequency emitted by the AI_OSCLK output is set in Chapter 5 Clock Reset and Power Management at Offset 0x04 7310 AI1_OSCLK_CTL. See PNX8526 Register Summary List for more details.				
Offset 0x11 1000			*AI_STATUS*	
31:5		-	Unused	
4	R	1	BUF1_ACTIVE	1 = Buffer will be used for the next incoming sample. 0 = Buffer 2 will receive the next sample.
3	R	0	OVERRUN	An OVERRUN error has occurred i.e., software failed to provide an empty buffer in time and 1 or more samples have been lost.
2	R	0	HBE	Bandwidth Error
1	R	0	BUF2_FULL	1 = Buffer 2 is full. If BUF2_INTEN is also 1, an interrupt request is pending.
0	R	0	BUF1_FULL	1 = Buffer 1 is full. If BUF1_INTEN is also 1, an interrupt request is pending.
Offset 0x11 1004			*AI_CTL*	
31	R/W	0	RESET	The Audio In logic is reset by writing a 0x80000000 to AI_CTL. This bit always reads as a ‘0’.
30	R/W	0	CAP_ENABLE	Capture Enable flag: 0 = Audio In is inactive. 1 = Audio In captures samples and acts as DMA master to write samples to local memory.
29:28	R/W	00	CAP_MODE	00 = Mono (left ADC only), 8 bits/sample 01 = Stereo, 2 times 8 bits/sample 10 = Mono (left ADC only), 16 bits/sample 11 = Stereo, 2 times 16 bits/sample
27	R/W	0	SIGN_CONVERT	0 = Leave msb unchanged. 1 = Invert msb.
26		-	Unused	
25	R/W	0	DIAGMODE	0 = Normal operation 1 = Diagnostic mode
24:8		-	Unused	
7	R/W	0	OVR_INTEN	Overrun Interrupt Enable: 0 = No interrupt 1 = Interrupt if an overrun error occurs.
6	R/W	0	HBE_INTEN	HBE Interrupt Enable: 0 = No interrupt 1 = Interrupt if a bandwidth error occurs.
5	R/W	0	BUF2_INTEN	Buffer 2 full interrupt Enable: 0 = No interrupt 1 = Interrupt if buffer 2 full.

AUDIO (I2S) INPUT PORTS 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
4	R/W	0	BUF1_INTEN	Buffer 1 full Interrupt Enable: 0 = No interrupt 1 = Interrupt if buffer 1 full.
3	R/W	0	ACK_OVR	Write a 1 to clear the OVERRUN flag and remove any pending OVERRUN interrupt request. This bit always reads as 0.
2	R/W	0	ACK_HBE	Write a 1 to clear the HBE flag and remove any pending HBE interrupt request. This bit always reads as 0.
1	R/W	0	ACK2	Write a 1 to clear the BUF2_FULL flag and remove any pending BUF2_FULL interrupt request. This bit always reads as 0.
0	R/W	0	ACK1	Write a 1 to clear the BUF1_FULL flag and remove any pending BUF1_FULL interrupt request. This bit always reads as 0.
Offset 0x11 1008			*AI_SERIAL*	
31	R/W	0	SER_MASTER	Sets clock ratios and internal/external clock generation. 0 = The A/D converter is the timing master over the serial interface. AI_SCK and AI_WS pins are set to be input. 1 = Audio In serial interface is the timing master over the external A/D. The AI_SCK and AI_WS pins are set to be outputs.
30	R/W	0	DATAMODE	0 = msb first 1 = lsb first
29:28	R/W	00	FRAMEMODE	This mode governs capturing of samples. 00 = Accept a sample every serial frame. 01 = Unused, reserved 10 = Accept sample if valid bit = 0. 11 = Accept sample if valid bit = 1.
27	R/W	0	CLOCK_EDGE	0 = The SD and WS pins are sampled on positive edges of the SCK pin. If SER_MASTER = 1, WS is asserted on SCK negative edge. 1 = SD and WS are sampled on negative edges of SCK. As output, WS is asserted on SCK positive edge.
26:17		-	Unused	
16:8	R/W	0	WSDIV	Sets the divider used to derive AI_WS from AI_SCK. Set to 0..511 for a serial frame length of 1..512.
7:0	R/W	0	SCKDIV	Sets the divider used to derive AI_SCK from AI_OSCLK. Set to 0..255, for division by 1..256.
Offset 0x11 100C			*AI_FRAMING*	
31	R/W	0	POLARITY	Sets format of serial data stream. 0 = Serial frame starts on WS negative edge. 1 = Serial frame starts on WS positive edge.
29:22	R/W	0	VALIDPOS	Defines bit position within a serial frame where the valid bit is found.
21:13	R/W	0	LEFTPOS	Defines bit position within a serial frame where the first data bit of the left channel is found.

AUDIO (I2S) INPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
12:4	R/W	0	RIGHTPOS	Defines bit position within a serial frame where the first data bit of the right channel is found.
3:0	R/W	0	SSPOS	Start/Stop bit position. If DATAMODE = msb first, SSPOS determines the bit index (0..15) in the parallel word of the *last* data bit. Bits 15 (msb) up to and including SSPOS are taken in order from the serial frame data. All other bits are set to zero. If DATAMODE = lsb first, SSPOS determines the bit index (0..15) in the parallel word of the first data bit. Bits SSPOS up to and including 15 are taken in order from the serial frame data. All other bits are set to zero.
Offset 0x11 1010			***Reserved***	
Offset 0x11 1014			***AI_BASE1***	
31:6	R/W	0	BASE1	Base Address of buffer 1 must be a 64-byte aligned address in local memory.
5:0			Reserved	
Offset 0x11 1018			***AI_BASE2***	
31:6	R/W	0	BASE2	Base Address of buffer 1 must be a 64-byte aligned address in local memory.
5:0			Reserved	
Offset 0x11 101C			***AI_SIZE***	
31:6	R/W	0	SIZE	Sets number of samples in buffers before switching to other buffers. In stereo modes, a pair of 8-bit or 16-bit data counts as 1 sample. In mono modes, a single value counts as a sample.
5:0			Reserved	
Offset 0x11 1020—1FF0			***Reserved***	

AUDIO (I2S) INPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 1FF4			***AI_PWR_DWN***	
31:1		-	Unused	
0	R/W	0	PWR_DWN	The bit is used to provide power control status for system software block power management. Not implemented.
Offset 0x11 1FFC			***AI_MODULE_ID***	
31:16	R	0x010D	ID	Module ID. This field identifies the block as type Audio In.

AUDIO (I2S) INPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:12	R	0	MAJ_REV	Major Revision ID. This field is incremented by 1 when changes introduced in the block result in software incompatibility with the previous version of the block. First version default = 0.
11:8	R	0	MIN_REV	Minor Revision ID. This field is incremented by 1 when changes introduced in the block result in software compatibility with the previous version of the block. First version default = 0.
7:0	R	0	APERTURE	Aperture size. Identifies the MMIO aperture size in units of 4 kB for the AI block. AI has an MMIO aperture size of 4 kB. Aperture = 0: 4 kB.

AUDIO (I2S) INPUT PORTS 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Audio In 2 registers begin at 0x11 3000. In all other respects, they are identical to the Audio In 1 registers.				

AUDIO (I2S) INPUT PORTS 3 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Audio In 3 registers begin at 0x11 5000. In all other respects, they are identical to the Audio In 1 registers.				

Chapter 21: Audio Output Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

21.1 Introduction

The PNX8526 has three digital audio output ports; Audio Out 1, 2 and 3. Audio Out 1 and 2 each support two channels (stereo). Audio Out 3 supports up to eight channels of audio.

All audio out ports are driven by identical audio out modules. Only the number of external data lines differ. Audio Out ports 1 and 2 have one data pin; port 3 has four.

21.1.1 Audio Out Module

The audio out module provides a DMA-driven serial interface designed to support stereo audio D/A converters. The Audio Out module can support up to eight PCM audio channels by driving up to four external stereo D/As. The audio out unit provides a glueless interface to high quality, low cost oversampling D/A converters. This includes a precisely programmable oversampling clock.

The Audio Out unit and external D/As together provide the following capabilities:

- Up to 8 channels of audio output
- 16-bit or 32-bit samples per channel
- Programmable 1 Hz to 100 kHz sampling rate

Remark: This is a practical range. The actual sample rate is application dependent.

- Internal or external bit clock source
- Autonomously reads processed audio data from memory using double buffering (DMA)
- 16-bit mono and stereo PC standard memory data formats
- Little or big-endian memory formats, set by Global Registers
- Control capability for highly integrated PC codecs.

Remark: AC-97 codecs are not supported.

21.2 Functional Description

21.2.1 Overview

The Audio Out module has three major subsystems: a DMA engine, a programmable sample clock generator, and a parallel-to-serial converter. The Audio Out pins provide the digital audio stream, clock and control signals to external D/A converters.

The DMA engine reads 16 or 32-bit samples from memory using a double buffered DMA approach. Software initially assigns two full sample buffers containing an integral number of samples for all active channels. The DMA engine retrieves samples from the first buffer until exhausted and continues from the second buffer, while requesting a new first sample buffer from the system controller, etc. The Audio Out is compatible with the TM3100 and software can be ported easily.

The samples are given to the data serializer (parallel-to-serial converter), which sends them out in a msb-first or lsb-first serial frame format that can also contain one or two codec control words of up to 16 bits. The output frame structure is programmable.

21.2.2 External Interface

The Audio Out module has seven signals: OSCLK, SCK, WS and SD[3:0].

Audio Out port 3 (I2s_IO) has four data pins, but these additional data ports can be used to support up to eight audio channels (four stereo channels). Any unused data channels can be programmed for alternate functions. (See Chapter 10 GPIO/IR for details.)

Table 1: Audio Out Unit External Signals

Signal	Type	Description
I^2S_OUT1_OSCLK I^2S_OUT2_OSCLK I^2S_IO_OSCLK	OUT	Oversampling Clock. This output can be programmed to emit any frequency up to 40 MHz. It is intended for use as the 256 f_s or 384 f_s oversampling clock by the external D/A conversion subsystem.
I^2S_OUT1_SCK I^2S_OUT2_SCK I^2S_IO_SCK	IO	When Audio Out is programmed to act as the serial interface timing slave (RESET default), SCK acts as input. It receives the Serial Clock from the external audio D/A subsystem. The clock is treated as fully asynchronous to the chip main clock. When Audio Out is programmed to act as serial interface timing master, SCK acts as output. It drives the Serial Clock for the external audio D/A subsystem. The clock frequency is a programmable integral divide of the OSCLK frequency. SCK is limited to the frequency of the OSCLK or lower.
I^2S_OUT1_WS I^2S_OUT2_WS I^2S_IO_WS	IO	When Audio Out is programmed as the serial-interface timing slave (RESET default), WS acts as an input. WS is sampled on the opposite SCK edge at which SD is asserted. When Audio Out is programmed as serial-interface timing master, WS acts as an output. WS is asserted on the same SCK edge as SD. WS is the word-select or frame-synchronization signal from/to the external D/A subsystem. Each audio channel receives one sample for every WS period. WS can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.
I^2S_OUT1_SD[0] I^2S_OUT2_SD[0] I^2S_IO_SD[0]	OUT	Serial Data to stereo external audio D/A subsystem. SD[0] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.

Table 1: Audio Out Unit External Signals

Signal	Type	Description
I^2S_IO_SD[1]	OUT	Serial Data to stereo external audio D/A subsystem. SD[1] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit. Available only on I^2S_IO ports.
I^2S_IO_SD[2]	OUT	Serial Data to stereo external audio D/A subsystem. SD[2] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit. Available only on I^2S_IO ports.
I^2S_IO_SD[3]	OUT	Serial Data to stereo external audio D/A subsystem. SD[3] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit. Available only on I^2S_IO ports.

The OSCLK output is an accurately programmable clock output intended to be used as the master system clock for the external D/A subsystem. The other pins (SCK, WS and SD[3:0]) constitute a flexible serial output interface.

Using the Audio Out MMIO registers, these pins can be configured to operate in a variety of serial interface framing modes, including but not limited to:

- Standard stereo I^2S (msb first, one-bit delay from WS, left and right data in a frame).
(For further details on I^2S, refer to the I2S Bus Specification).
- lsb first with 1–16 bit data per channel.
- Complex serial frames of up to 512 bits/frame.

21.2.3 Sample Clock Generator

The clock generator is programmable to support various sample frequencies. Figure 1 illustrates the different clock capabilities of the Audio Out unit. A square wave Direct Digital Synthesizer (DDS) drives the clock system. This DDS is part of the Clocks module. Refer to the clocks section in Chapter 5 Clock Reset and Power Management for more information.

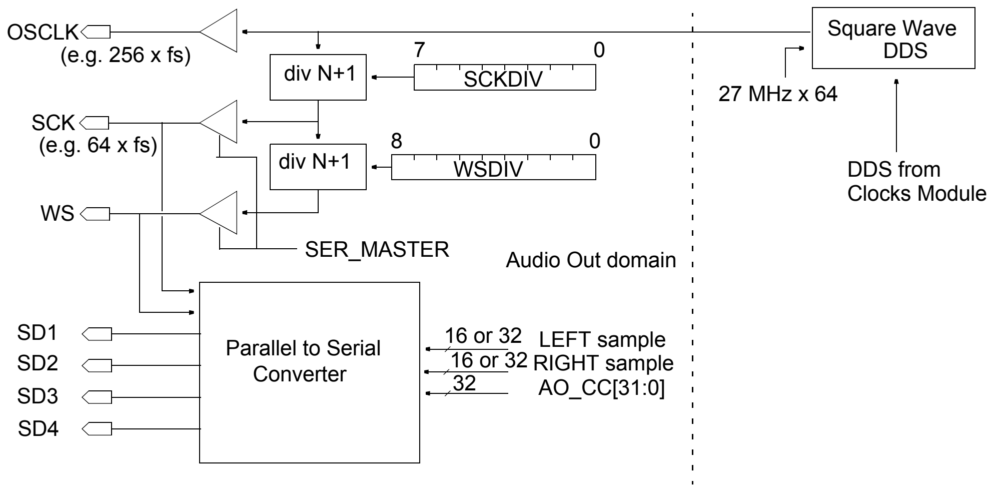


Figure 1: Audio Out Clock System and I/O Interface

Using the DDS as a clock source allows software to control the coarse and fine clock rate so that complex forms of synchronization can be implemented without external hardware. Examples include locking the audio to a broadcast clock, or locking it to an SPDIF input without changing the system's hardware.

The output of the DDS is always sent to the OSCLK output pin. (See Clock registers: DDS Audio Out—Offsets 0x04 710C, 7114, 7118 in Chapter 5 Clock Reset and Power Management for more details.) This output is intended to be used as the 256 fs or 384 fs system clock source for oversampling D/A converters.

The software may change the oversampling clock frequency dynamically (via the DDS register) to adjust the outgoing audio sample rate. In ATSC transport stream decoding, this is the method used by which the system software locks the audio output sample rate to the original program provider sample rate.

Table 2 presents several sample rates with the SCKDIV setting necessary to achieve a bit clock of 64 Fs.

Table 2: Clock System Setting

f_s	OSCLK	SCKDIV	SCK
44.1 kHz	256 fs	3	64 fs
48.0 kHz	256 fs	3	64 fs
44.1 kHz	384 fs	5	64 fs
48.0 kHz	384 fs	5	64 fs

The values of SCKDIV given assume the oversampling clock supplied to Audio Out is either 256 Fs or 384 Fs. The value of SCKDIV is determined by the following equation.

$$f_{SCK} = \frac{f_{OSCLK}}{SCKDIV + 1}$$

Remark: SCKDIV is in the range 0-255.

21.2.3.1 Clock System Operation

WS and SCK are sent to each external D/A converter in the master mode. WS, the word strobe, determines the sample rate: each active channel receives one sample for each WS period. SCK is the data bit clock. The number of SCK clocks in a WS period is the number of data bits in a serial frame required by the attached D/A converter.

WS is a divided form of the SCK bit clock, set to use WSDIV to control the serial frame length. The number of bits per frame is equal to WSDIV+1. There are some minimum length requirements for a serial frame. Refer to Section 21.3.4.1 on page 21-439 for details.

SCK and WS can be configured as input or output as determined by the SER_MASTER control field. If set as output, SCK can be set to a divider of the OSCLK output frequency. (See Section 21.4 on page 21-443 for more details.)

Whether set as input or output, the SCK pin signal is always used as the bit clock for parallel-to-serial conversion. The WS pin always acts as the trigger to start the generation of a serial frame. WS can similarly be programmed using WSDIV to control the serial frame length. The number of bits per frame is equal to WSDIV+1.

The preferred use of the clock system options is to use OSCLK as D/A master clock and let the D/A converter be timing slave of the serial interface (SER_MASTER = 1).

Some D/A converters however, provide somewhat better SNR properties if they are configured as serial masters instead, with Audio Out as slave (SER_MASTER = 0). As illustrated by Figure 1, the internal parallel to serial converter that constructs the serial frame is oblivious to who is the serial master.

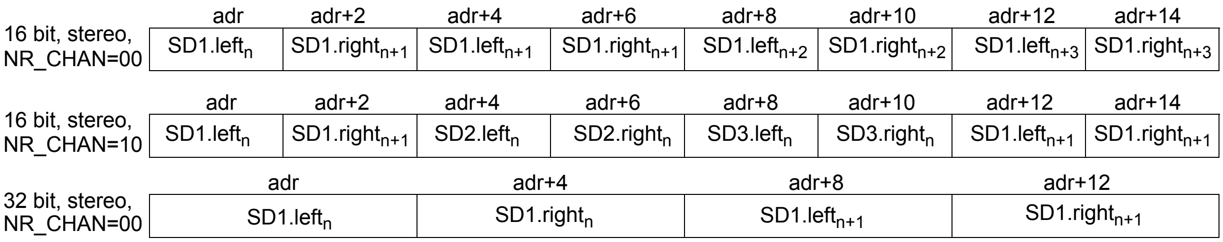
21.3 Operation

This section describes Audio Out operation and the configuration features to achieve the available modes. This section also covers interrupts and error conditions.

21.3.1 Memory Data Formats

The Audio Out unit autonomously reads samples from memory in 16 or 32-bit per sample memory formats, as shown in Figure 2. Successive samples are always read from increasing memory address locations.

The Audio Out unit hardware implements a double buffering scheme to ensure that there are always samples available to transmit, even though the system controller is highly loaded and slow to respond to interrupts. The software assigns two equal size buffers by writing a base address value and size value to the MMIO control fields described in Section 21.4. Refer to Section 21.3.2 for details on hardware/software synchronization.



	adr	adr+2	adr+4	adr+6	adr+8	adr+10	adr+12	adr+14
16 bit, stereo, NR_CHAN=00	SD1.left$_n$	SD1.right$_{n+1}$	SD1.left$_{n+1}$	SD1.right$_{n+1}$	SD1.left$_{n+2}$	SD1.right$_{n+2}$	SD1.left$_{n+3}$	SD1.right$_{n+3}$

	adr	adr+2	adr+4	adr+6	adr+8	adr+10	adr+12	adr+14
16 bit, stereo, NR_CHAN=10	SD1.left$_n$	SD1.right$_{n+1}$	SD2.left$_n$	SD2.right$_n$	SD3.left$_n$	SD3.right$_n$	SD1.left$_{n+1}$	SD1.right$_{n+1}$

	adr	adr+4	adr+8	adr+12
32 bit, stereo, NR_CHAN=00	SD1.left$_n$	SD1.right$_n$	SD1.left$_{n+1}$	SD1.right$_{n+1}$

Figure 2: Audio Out Memory DMA Formats

Table 3: Operating Modes and Memory Formats

NR_CHAN	MODE	Destination of Successive Samples
00	mono	SD1.left
00	stereo	SD1.left, SD1.right
01	mono	SD1.left, SD2.left
01	stereo	SD1.left, SD1.right, SD2.left, SD2.right
10	mono	SD1.left, SD2.left, SD3.left
10	stereo	SD1.left, SD1.right, SD2.left, SD2.right, SD3.left, SD3.right
11	mono	SD1.left, SD2.left, SD3.left, SD4.left
11	stereo	SD1.left, SD1.right, SD2.left, SD2.right, SD3.left, SD3.right, SD4.left, SD4.right.

Prior to output transmission, if SIGN_CONVERT = 1, the msb of the memory data is inverted. This allows the use of external two's complement 16-bit D/A converters to generate audio from 16-bit unsigned samples. This msb inversion also applies to the '0' values transmitted to non-active output channels.

Remark: The Audio Out hardware does *not* support A-law or m-law data formats. If such formats are required, the system DSP processor should be used to convert from A-law or m-law data to 16-bit linear data.

21.3.1.1 Endian Control

Audio Out supports both little-endian and big-endian byte ordering. The module reads the chip-level endianness control signal and will slave to this signal for normal capture operation. This global "sys_big_end" control bit resides in the reset control register RST_CTL (see the PNX8526 Register Summary List) determines how increasing memory addresses map to byte positions within audio words. The Audio Out unit power-on default state uses this "global" system endian control input.

21.3.2 Audio Out MMIO Description

Upon reset, transmission is disabled (TRANS_ENABLE = 0), and buffer1 is the active buffer (BUF1_ACTIVE = 1). The system software initiates transmission by providing two full equal size buffers (with valid audio data) and putting their base address and size in the two AO_BASEx registers and the AO_SIZE register. Once two valid buffers are assigned, transmission can be enabled by writing a '1' to TRANS_ENABLE. The Audio Out unit hardware now proceeds to empty buffer 1 by transmission of output samples. Once buffer 1 empties, BUF1_EMPTY is asserted, and transmission continues without interruption from buffer 2. If BUF1INTEN is enabled, a level triggered system level interrupt request is generated.

Remark: Buffers must be 64-byte aligned (the six lsbits of AO_BASE1, AO_BASE2 are zero). Buffer sizes must be a multiple of 64 samples (the six lsbits of AO_SIZE are zero).

The system software is required to assign a new, full buffer to AO_BASE1 and perform an ACK1 before buffer 2 empties. Transmission continues from buffer 2 until it is empty. At that time, BUF2_EMPTY is asserted, and transmission continues from the new buffer 1, etc. An ACK performs two functions:

- It notifies Audio Out that the corresponding AO_BASEx register now points to a buffer filled with samples.
- It clears BUF_EMPTY. Upon receipt of an ACK, the Audio Out hardware removes the BUF_EMPTY related interrupt request line assertion at the next system controller clock edge.

21.3.3 Interrupts

Audio Out has a private, level triggered interrupt request line to the system interrupt controller. An interrupt is asserted as long as one or more of the UNDERRUN, HBE, BUF1_EMPTY or BUF2_EMPTY condition flags and the corresponding INTEN bit are asserted. Interrupts are sticky—i.e., an interrupt remains asserted until the software explicitly clears the condition flag by an ACK action.

Remark: For legacy reasons, the MMIO interrupt mechanism for Audio Out has not been changed. As such, the Audio Out module is not strictly compliant with DVP block interrupt architecture recommendations.

21.3.3.1 Interrupt Latency

During normal error free transmission, the source of an Audio Out interrupt will be either BUF1_EMPTY, BUF2_EMPTY or both. The DMA buffer sizes configured in the AO_SIZE register will directly affect the frequency of these 'empty' interrupts. Software should set up the AO_SIZE register with a large enough value so that interrupts occur no more frequently than 1 ms in order to meet system interrupt latency requirements.

21.3.4 Serial Data Framing

The Audio Out unit can generate data in a wide variety of serial data framing conventions. Figure 3 illustrates the notion of a serial frame. If POLARITY = 1, a frame starts with a positive edge of the WS signal. If POLARITY = 0, a serial frame starts with a negative edge on WS. (See Section 21.4 on page 21-443.) If CLOCK_EDGE = 0, the parallel-to-serial converter samples WS on a positive clock edge transition and outputs the first bit (bit 0) of a serial frame on the next falling edge of SCK.

If CLOCK_EDGE = 1, the parallel-to-serial converter samples WS on the negative edge of SCK, while audio data is output on the positive edge. That is, the SCK polarity would be reversed with respect to Figure 3.

Every serial frame transmits a single left and right channel sample and optional codec control data to each D/A converter. The sample data can be in an lsb-first or msb-first form at an arbitrary serial frame bit position and with an arbitrary length. (See Section 21.4.)

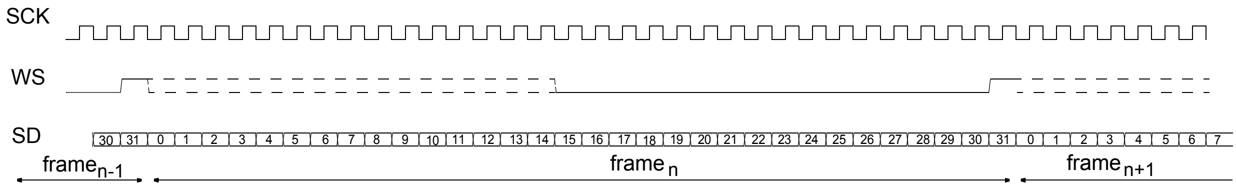


Figure 3: Definition of Serial Frame Bit Positions (POLARITY = 1, CLOCK_EDGE = 0)

In msb-first mode (DATAMODE = 0), the parallel-to-serial converter sends the value of LEFT[MSB] in bit position LEFTPOS in the serial frame. Subsequently, bits from decreasing bit positions in the LEFT data word, up to and including LEFT[SSPOS], are transmitted in order.

In lsb-first mode (DATAMODE = 1), the parallel-to-serial converter sends the value of LEFT[SSPOS] in bit position LEFTPOS in the serial frame. Subsequent bits from the LEFT data word, up to and including LEFT[MSB], are transmitted in order. The exact bits transmitted for a data item ‘S’ are shown in Table 4.

Table 4: Bits Transmitted for Each Memory Data Item

Operating Mode	First Bit	Last Bit	Valid SSPOS Values
16 bit/sample, msb-first	S[15]	S[SSPOS]	0..15
16 bit/sample, lsb-first	S[SSPOS]	S[15]	0..15
32 bit/sample, msb-first	S[31]	S[SSPOS]	0..31
32 bit/sample, lsb-first	S[SSPOS]	S[31]	0..31

Frame bits that do not belong to either LEFT[MSB:SSPOS] or RIGHT[MSB:SSPOS] or a codec control field (Section 21.3.5) are shifted out as zero. This zero extension ensures that Audio Out can be used in combination with D/A converters which expect more bits than the actual number of transmitted bits in the current operating mode e.g., 18-bit D/As operating with 16-bit memory data.

21.3.4.1 Serial Frame Limitations

Due to the implementation, there is a minimum serial frame length requirement that is operating-mode dependent according to Table 5.

Table 5: Minimum Serial Frame Length in Bits

Operating Mode	Minimum Serial Frame Length
16 bit/sample, mono	13 bits
32 bit/sample, mono	13 bits
16 bit/sample, stereo	13 bits
32 bit/sample, stereo	36 bits

21.3.4.2 I^{2}S Serial Framing Example

Figure 4 and Table 6 show how the Audio Out unit MMIO registers should be set to transmit 16 or 32 bits of stereo data via an I^{2}S serial standard to an 18-bit D/A converter with a 64-bit serial frame.

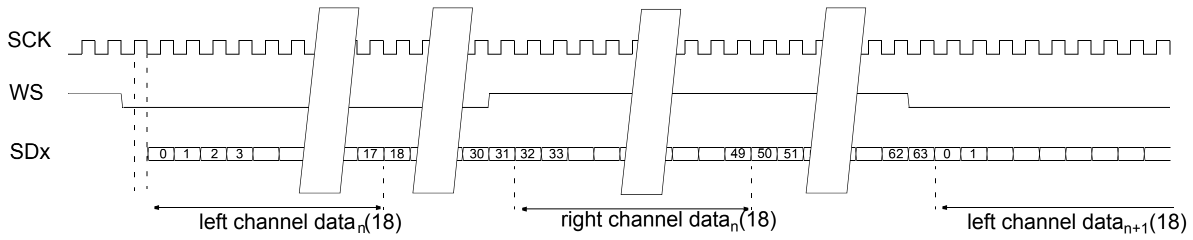

SCK
WS
SDx
0
1
2
3
17
18
30
31
32
33
49
50
51
62
63
0
1
left channel data$_n$(18)
right channel data$_n$(18)
left channel data$_{n+1}$(18)

Figure 4: Serial Frame (64 Bits) of a 18-Bit Precision I^{2}S D/A Converter

Table 6: Example Setup For 64-Bit I^{2}S Framing

Field	Value	Explanation
POLARITY	0	Frame starts with negative edge Audio Out WS.
LEFTPOS	0	LEFT[MSB] will go to serial frame position 0.
RIGHTPOS	32	RIGHT[MSB] will go to serial frame position 32.
DATAMODE	0	msb first.
SSPOS	0	Stop with LEFT/RIGHT[0], send 0s after. (For 32 bit/sample mode, this field could be set to 14 to ensure zeroes in all unused bit positions.)
CLOCK_EDGE	0	Audio Out SD change on negative edge Audio Out SCK
WSDIV	63	Serial frame length = 64.
WS_PULSE	0	Emit 50% duty cycle Audio Out WS.

Remark: The transfer of data from SDRAM into the Audio Out module's buffers is initiated by the transition of WS, not by transmit enable. As a consequence, there is a delay between the receipt of the first WS pulse and the transmission of the first data. The length of this delay is dependent on system load and is not easily predictable.

21.3.5 Codec Control

In addition to the left and right data fields that are generated based on autonomous DMA action, a serial frame generated by Audio Out can be set to contain one or two control fields up to 16 bits in length. Each control field can be independently enabled or disabled by the CC1_EN, CC2_EN bits in AO_CTL.

The content shifted into the frame is taken from the CC1 and CC2 field in the AO_CC register. The CC1_POS and CC2_POS fields in the AO_CFC register determine the first bit position in the frame where the control field is emitted. The field is emitted observing the setting of DATAMODE i.e., lsb or msb first.

The CC_BUSY bit in AO_STATUS indicates if the Audio Out unit is ready to receive another CC1, CC2 value pair. Writing a new value pair to AO_CC writes the value into a buffer register and raises the CC_BUSY status. (See Section 21.4 on page 21-443.) As soon as both CC1 and CC2 values have been copied to a shadow register in preparation for transmission, CC_BUSY is negated, indicating that the Audio Out logic is ready to accept a new codec control pair. The old CC1/CC2 data keeps being transmitted—i.e., software is not required to provide new CC1 and CC2 data.

Software must ensure that the CC_BUSY status is negated before writing a new CC1, CC2 pair. The user, by the process of waiting on CC_BUSY, can reliably emit a sequence of individual audio frames with distinct control field values. This can, for example, be used during codec initialization.

Remark: No provision is made for interrupt-driven operation of such a sequence of control values. It is assumed, after initialization, that the value of control fields determines slowly changing, asynchronous parameters such as output volume.

It is legal to program the control field positions within the frame such that CC1 and CC2 overlap each other and/or left and right data fields. If two fields are defined to start at the same bit position, the priority is left (highest), right, CC1 then CC2. The field with the highest priority will be emitted starting at the conflicting bit position. If a field *f2* is defined to start at a bit position *i* that falls within a field *f1* starting at a lower bit position, *f2* will be emitted starting from *i* and the rest of *f1* will be lost. Any bit positions not belonging to a data or control field will be emitted as zero.

Figure 5 shows a 64-bit frame suitable for use with the CS4218 codec. It is obtained by setting POLARITY=1, LEFTPOS=0, RIGHTPOS=32, DATAMODE=0, SSPOS=0, CLOCK_EDGE=1, WS_PULSE=1, CC1_POS= 16, CC1_EN=1, CC2_POS=48 and CC2_EN=1.

Remark: Frames are generated (externally or internally) even when TRANS_ENABLE is de-asserted. Writes to CC1 and CC2 should only be done after TRANS_ENABLE is asserted. The 'first' CC values will then go out on the next frame.

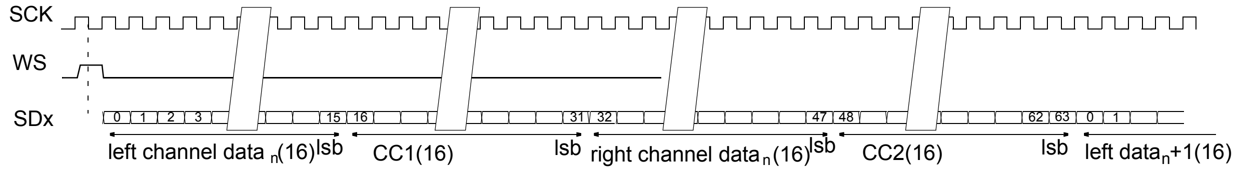


Figure 5: Example Codec Frame Layout for a Crystal Semiconductor CS4218

21.3.6 Timestamp Events

Audio Out exports event signals associated with audio transmission to the central timestamp/timer function on-chip. The central timestamp/timer function can be used to count the number of occurrences of each event or timestamp, the occurrence of the event, or both. The event will be a positive edge pulse with the duration of the event to be greater than or equal to 200 ns. The specific event exported is as follows:

- BUF_DONE — The occurrence of this TSTAMP event indicates that one of the internal 64-byte hardware buffers corresponding to the *last* 64 bytes of the current external DMA memory buffer. This event is not dependent upon PI-Bus latency.

- *Audio Out 3 only:* The WS event indicates the arrival of a particular audio frame on the I^2S interface. Timer Mux Control in the Global 1 Registers (See Global Registers in Chapter 25 Internal TM32 CPU Core Processor) can select AIO3's WS for timestamping.

This event represents a precise, periodic time interval which can be used by system software for audio/video synchronization.

21.3.7 Reset

Audio Out is reset by the external chip reset pin or by writing AO_CTLRESET = 1. Either reset method sets all MMIO fields to their default values as indicated in the tables. Audio Out is not affected by any other reset signal or mechanism. In addition, any ongoing data bus request, DMA and interrupt request activity is deactivated.

After an Audio Out reset, five Audio Out SCK clock cycles are required to stabilize the internal circuitry prior to enabling Audio Out. This can be accomplished by first programming the AO_SERIAL register to start Audio Out SCK generation, and then waiting for the appropriate five Audio Out SCK cycle interval.

21.3.8 Powerdown Behavior

Audio Out has no internal powerdown functionality, however the chip power management software can remove the main block level clock to Audio Out. If the Audio Out module enters a powerdown state, SCK, WS and SDx hold their value stable but OSCLK continues to provide a D/A converter oversampling clock.

Once the system wakes up, the signals resume their original transitions at the point where they were halted. The external D/A converter subsystem is most likely confused by this behavior, so it is recommended that Audio Out transmission be stopped (by deactivating TRANS_ENABLE) prior to software enable of Audio Out powerdown.

21.3.9 Data Bus Latency and HBE

The Audio Out unit uses two internal hardware 64-byte buffers as well as an output holding register that holds a single mono sample or single stereo sample pair. For Audio Out there are four separate stereo output channels and each output channel has one output holding register. The holding register width is 64 bits.

Under normal operation, the internal hardware buffers get refreshed from memory fast enough to avoid any missing samples. Meanwhile, data is being emitted from one 64-byte hardware buffer and holding register. If the data bus arbiter is set up with an insufficient latency guarantee, the situation can arise that one of the 64-byte hardware buffers is not refilled in time and the buffer and holding register are exhausted by the time a new output sample is due. In that case the HBE flag is raised. The last sample for each channel will be repeated until the buffer is refreshed. The HBE condition is sticky and can only be cleared by an explicit ACK_HBE. This condition indicates an incorrect setting of the data bus bandwidth arbiter.

Table 7 shows the maximum tolerable latency for a number of common operating modes. The right-most column in the table indicates the maximum tolerable latency for Audio Out under normal operating condition. To sustain error free audio playback, one

64-byte DMA transfer must be completed within the maximum latency period indicated for each operating mode.

Remark: For high sample rates with more than four channels (96 kHz, 32-bit, >4ch), Audio Out cannot guarantee error-free operation due to data bus latency restrictions.

Table 7: Audio Out Latency Tolerance Examples

Transfer Mode	fs (kHz)	T (nSec)	* Max Latency (uSec) (with T=1/fs)
2-ch stereo 16 bit/sample	48.0	20833	354.17
4-ch stereo 16 bit/sample	48.0	20833	187.50
6-ch stereo 16 bit/sample	48.0	20833	104.17
8-ch stereo 16 bit/sample	48.0	20833	104.17
2-ch stereo 32 bit/sample	48.0	20833	187.50
4-ch stereo 32 bit/sample	48.0	20833	104.17
6-ch stereo 32 bit/sample	48.0	20833	62.50

Table 7: Audio Out Latency Tolerance Examples *…Continued*

Transfer Mode	fs (kHz)	T (nSec)	* Max Latency (uSec) (with T=1/fs)
8-ch stereo 32 bit/sample	48.0	20833	62.50
2-ch stereo 16 bit/sample	96.0	10417	177.08
4-ch stereo 16 bit/sample	96.0	10417	93.75
6-ch stereo 16 bit/sample	96.0	10417	52.08
8-ch stereo 16 bit/sample	96.0	10417	52.08
2-ch stereo 32 bit/sample	96.0	10417	93.75
4-ch stereo 32 bit/sample	96.0	10417	52.08

* Max Latency (uSec) (with T=1/fs)

2ch 16it stereo: (17 * T)
4ch 16bit stereo: (9 * T)
6ch-16bit stereo: (5 * T)
8ch 16bit stereo: (5 * T)
--
2ch 32bit stereo: (9 * T)
4ch 32bit stereo: (5 * T)
6ch 32bit stereo: (3 * T)
8ch 32bit stereo: (3 * T)

21.3.10 Error Behavior

In normal operation, the system controller and Audio Out hardware continuously exchange buffers without ever failing to transmit a sample. If the system controller fails to provide a new buffer in time, the UNDERRUN error flag is raised and the last valid sample or sample pair is repeated until a new buffer of data is assigned by an ACK1 or ACK2. The UNDERRUN flag is *not affected* by ACK1 or ACK2; it can only be cleared by an explicit ACK_UDR.

If an HBE error occurs, the last valid sample or sample pair is repeated until the Audio Out hardware retrieves a new sample buffer across the data bus.

21.4 Register Descriptions

The PNX8526 has three digital audio output ports. The registers for Audio Out 1 begin at 0x11 0000. The registers for Audio Out 2 begin at 0x11 2000. The registers for Audio Out 3 begin at 0x11 4000. The register sets for the three modules are identical, so only the Audio Out 1 registers are detailed in the following tables.

Remark: Audio Out ports 1 and 2 each support two channels (stereo). Audio Out 3 supports up to eight channels of audio.

21.4.1 Register Address Map

Table 8: Audio Output Ports 1-3, Register Summary

Offset	Name	Description
Audio Out 1 (I2S_OUT1)		
0x11 0000	AO_STATUS	Provides status of buffers and other Audio Out components/situations.
0x11 0004	AO_CTL	Control register to configure Audio Out options.
0x11 0008	AO_SERIAL	Control register to configure Audio Out serial timing and data options.
0x11 000C	AO_FRAMING	Control register to configure data framing.
0x11 0010	Reserved	
0x11 0014	AO_BASE1	Base address of buffer 1.
0x11 0018	AO_BASE2	Base address of buffer 2.
0x11 001C	AO_SIZE	The DMA Buffer size in samples.
0x11 0020	AO_CC	Codec Control data content.
0x11 0024	AO_CFC	Codec data position.
0x11 0028—0FF0	Reserved	
0x11 0FF4	AO_PWR_DWN	Powerdown function.
0x11 0FFC	AO_MODULE_ID	Provides module ID number, including major and minor revision levels.
Audio Out 2 (I2S_OUT2)		
0x11 2000	AO_STATUS	Provides status of buffers and other Audio Out components/situations.
0x11 2004	AO_CTL	Control register to configure Audio Out options.
0x11 2008	AO_SERIAL	Control register to configure Audio Out serial timing and data options.
0x11 200C	AO_FRAMING	Control register to configure data framing.
0x11 2010	Reserved	
0x11 2014	AO_BASE1	Base address of buffer 1.
0x11 2018	AO_BASE2	Base address of buffer 2.
0x11 201C	AO_SIZE	The DMA Buffer size in samples.
0x11 2020	AO_CC	Codec Control data content.
0x11 2024	AO_CFC	Codec data position.
0x11 2028—2FF0	Reserved	
0x11 2FF4	AO_PWR_DWN	Powerdown function.
0x11 2FFC	AO_MODULE_ID	Provides module ID number, including major and minor revision levels.
Audio Out 3 (I2S_IO)		
0x11 4000	AO_STATUS	Provides status of buffers and other Audio Out components/situations.
0x11 4004	AO_CTL	Control register to configure Audio Out options.
0x11 4008	AO_SERIAL	Control register to configure Audio Out serial timing and data options.
0x11 400C	AO_FRAMING	Control register to configure data framing.

Table 8: Audio Output Ports 1-3, Register Summary *…Continued*

Offset	Name	Description
0x11 4010	Reserved	
0x11 4014	AO_BASE1	Base address of buffer 1.
0x11 4018	AO_BASE2	Base address of buffer 2.
0x11 401C	AO_SIZE	The DMA Buffer size in samples.
0x11 4020	AO_CC	Codec Control data content.
0x11 4024	AO_CFC	Codec data position.
0x11 4028—4FF0	Reserved	
0x11 4FF4	AO_PWR_DWN	Powerdown function.
0x11 4FFC	AO_MODULE_ID	Provides module ID number, including major and minor revision levels.

AUDIO (I2S) OUTPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Audio Out 1 Registers(Offset 0x11 0000), Audio Out 2 Registers (Offset 0x11 2000) and Audio Out 3 Registers (Offset 0x11 4000) The Audio Out 1, 2 and 3 registers are identical except for their offsets. A triplicate set has been created to facilitate programming. The tables for Audio Out 2 (0x11 2000) and 3 (0x11 4000) registers follow this table.				
Note: The clock frequency emitted by the OSCLK output can be found in Chapter 5 Clock Reset and Power Management at Offset 0x04 7314 AO1_OSCLK_CTL. See the PNX8526 Register Summary List for more details.				
Offset 0x11 0000			***AO_STATUS***	
31:6		-	Unused	
5	R	0	CC_BUSY	0 = Audio Out is ready to receive a CC1, CC2 pair. 1 = Audio Out is not ready to receive a CC1, CC2 pair. Try again in a few SCK clock intervals.
4	R	1	BUF1_ACTIVE	1 = buffer 1 will be used for the next sample to be transmitted. 0 = buffer 2 will contain the next sample.
3	R	0	UNDERRUN	An UNDERRUN error has occurred i.e., the system controller/ software failed to provide a full buffer in time and no samples were transmitted, although requested by the D/A converter. If UDR_ INTEN is also 1, an interrupt request is pending. The UNDERRUN flag can ONLY be cleared by writing a ‘1’ to ACK_UDR.
2	R	0	HBE	Bandwidth Error indicates that no data was transmitted due to an inability to read the local AO buffer from memory in time.
1	R	0	BUF2_EMPTY	If 1, buffer 2 is empty. If BUF2_INTEN is also 1, an interrupt request is asserted. BUF2_EMPTY is cleared by writing a ‘1’ to ACK2, at which point the Audio Out hardware will assume that AO_BASE2 and AO_SIZE describe a new full buffer.
0	R	0	BUF1_EMPTY	If 1, buffer 1 is empty. If BUF1_INTEN is also 1, an interrupt request is asserted. BUF1_EMPTY is cleared by writing a ‘1’ to ACK1, at which point the Audio Out hardware will assume that AO_BASE1 and AO_SIZE describe a new full buffer.

AUDIO (I2S) OUTPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 0004			*AO_CTL*	
31	R/W	0	RESET	Resets the Audio Out logic. See Section 21.3.7 on page 21-441 for a description of the recommended procedure.
30	R/W	0	TRANS_ENABLE	Transmission Enable flag 0 = Audio Out is inactive. 1 = Audio Out transmits samples and acts as DMA master to read samples from local memory. Do not change the POLARITY bit while transmission is enabled.
29:28	R/W	00	TRANS_MODE	00 = Mono, 32 bits/sample. Left and right data sent to each active output are the same. 01 = Stereo, 32 bits/sample 10 = Mono, 16 bits/sample. Left and right data are the same. 11 = Stereo, 16 bits/sample
27	R/W	0	SIGN_CONVERT	0 =Leave msb unchanged. 1 = Invert msb (not applied to codec control fields).
26:25		-	Unused	
24	R/W	0	CC1_EN	0 = CC1 emission disabled. 1 = CC1 emission enabled.
23	R/W	0	CC2_EN	0 = CC2 emission disabled. 1 = CC2 emission enabled.
22	R/W	0	WS_PULSE	0 = Emit 50% AO_WS. 1 = Emit single AO_SCK cycle AO_WS.
21:8		-	Unused	
7	R/W	0	UDR_INTEN	UNDERRUN Interrupt Enable. 0 = No interrupt 1 = Interrupt if an UNDERRUN error occurs.
6	R/W	0	HBE_INTEN	HBE Interrupt Enable: 0 = No interrupt 1 = Interrupt if a data bus bandwidth error occurs.
5	R/W	0	BUF2_INTEN	Buffer 2 Empty Interrupt Enable: 0 = No interrupt 1 = Interrupt if buffer 2 empty
4	R/W	0	BUF1_INTEN	Buffer 1 Empty Interrupt Enable: 0 = No interrupt 1 = Interrupt if buffer 1 empty.
3	R/W	0	ACK_UDR	Write a 1 to clear the UNDERRUN flag and remove any pending UNDERRUN interrupt request. ACK_UDR always reads 0.
2	R/W	0	ACK_HBE	Write a 1 to clear the HBE flag and remove any pending HBE interrupt request. ACK_HBE always reads as 0.
1	R/W	0	ACK2	Write a 1 to clear the BUF2_EMPTYflag and remove any pending BUF2_EMPTY interrupt request. ACK2 always reads 0.
0	R/W	0	ACK1	Write a 1 to clear the BUF1_EMPTY flag and remove any pending BUF1_EMPTY interrupt request. ACK1 always reads 0.

AUDIO (I2S) OUTPUT PORTS 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Offset 0x11 0008			***AO_SERIAL***	
31	R/W	0	SER_MASTER	0 = The D/A subsystem is the timing master over the Audio Out serial interface. SCK and WS act as inputs. 1 = AO is the timing master over serial interface. SCK and WS act as outputs. This mode is required for 4, 6 or 8 channel operation. The SER_MASTER bit should only be changed while Audio Out is disabled i.e. ,TRANS_ENABLE = 0.
30	R/W	0	DATAMODE	0 = msb first 1 = lsb first
29	R/W	0	CLOCK_EDGE	0 = The parallel-to-serial converter samples WS on positive edges of SCK and outputs data on the negative edge of SCK. 1 = The parallel-to-serial converter samples WS on negative edges of SCK and outputs data on positive edges of SCK.
28:19		-	Unused	
18:17	R/W	00	NR_CHAN	00 = Only SD[0] is active. 01 = SD[0] and [1] are active. 10 = SD[0], [1], and [2] are active. 11 = SD[0]..SD[3] are active. Each SD output receives either 1 or 2 channels depending on TRANS_MODE. Non-active channels receive 0 value samples. In mono modes, each channel of a SD output receives identical left and right samples. Audio Out 1 and 2 have one data line which must be set to 00.
16:8	R/W	0x0	WSDIV	Sets the divider used to derive WS from SCK. Set to 0..511 for a serial frame length of 1..512.
7:0	R/W	0x0	SCKDIV	Sets the divider used to derive SCK from OSCLK. Set to 0..255 for division by 1..256.
Offset 0x11 000C			***AO_FRAMING***	
31	R/W	0	POLARITY	0 = Serial frame starts with a WS negedge. 1 = Serial frame starts with a WS posedge. This bit should not be changed during operation of Audio Out i.e., only update this bit when TRANS_ENABLE = 0.
30	R/W	0	SSPOS4	Start/Stop bit position msb. Note that SSPOS is a 5-bit field, while bit SSPOS4 is non-adjacent for backwards compatibility in 16-bit/sample modes. Program this field along with AO_FRAMING[3:0].
29:22		-	Unused	
21:13	R/W	0x0	LEFTPOS	Defines the bit position within a serial frame where the first data bit of the left channel is placed.
12:4	R/W	0x0	RIGHTPOS	Defines the bit position within a serial frame where the first data bit of the right channel is placed.

AUDIO (I2S) OUTPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
3:0	R/W	0x0	SSPOS	Start/Stop bit position. Note that SSPOS is a 5-bit field, while bit SSPOS4 is non-adjacent for backwards compatibility in 16-bit/ sample modes. Program this field along with AO_FRAMING[3:0]. If DATAMODE = msb first, transmission starts with the msb of the sample i.e., bit 15 for 16-bit/sample modes or bit 31 for 32-bit/ sample modes. SSPOS determines the bit index (0..31) in the parallel input word of the last transmitted data bit. If DATAMODE = lsb first, SSPOS determines the bit index (0..31) in the parallel word of the first transmitted data bit. Bits SSPOS up to and including the msb are transmitted i.e., up to bit 15 in 16-bit/sample mode and bit 31 in 32-bit/sample mode.
Offset 0x11 0010			***Reserved***	
Offset 0x11 0014			***AO_BASE1***	
31:6	R/W	0x0	BASE1	Base Address of buffer1 must be a 64-byte aligned address in local memory.
5:0		-	Unused	
Offset 0x11 0018			***AO_BASE2***	
31:6	R/W	0x0	BASE2	Base Address of buffer2 must be a 64-byte aligned address in local memory.
5:0		-	Unused	
Offset 0x11 001C			***AO_SIZE***	
31:6	R/W	0	SIZE	DMA buffer size in samples. The number of mono samples or stereo sample pairs is read from a DMA buffer before switching to the other buffer. Buffer size in bytes is as follows: 16 bps, mono: 2 * SIZE 32 bps, mono: 4 * SIZE 16 bps, stereo: 4 * SIZE 32 bps, stereo: 8 * SIZE
5:0		-	Unused	
Offset 0x11 0020			***AO_CC***	
31:16	R/W	0x0	CC1	The 16-bit value of CC1 is shifted into each emitted serial frame starting at bit position CC1_POS, as long as CC1_EN is asserted.
15:0	R/W	0x0	CC2	The 16-bit value of CC2 is shifted into each emitted serial frame starting at bit position CC2_POS, as long as CC2_EN is asserted.
Offset 0x11 0024			***AO_CFC***	
31:18		-	Unused	
17:10	R/W	0x0	CC1_POS	Defines the bit position within a serial frame where the first data bit of CC1 is placed.
9:0	R/W	0x0	CC2_POS	Defines the bit position within a serial frame where the first data bit of CC2 is placed.
Offset 0x11 0028—0FF0			***Reserved***	

AUDIO (I2S) OUTPUT PORTS 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 0FF4			*AO_PWR_DWN*	
31:1		-	Unused	
0	R/W	0	PWR_DWN	The bit is used to provide power control status for system software block power management.
Offset 0x11 0FFC			*AO_MODULE_ID*	
31:16	R	0x0120	ID	Module ID. This field identifies the block as type Audio Out.
15:12	R	0	MAJ_REV	Major Revision ID. This field is incremented by 1 when changes introduced in the block result in software incompatibility with the previous version of the block. First version default = 0.
11:8	R	0	MIN_REV	Minor Revision ID. This field is incremented by 1 when changes introduced in the block result in software *compatibility* with the previous version of the block. First version default = 0.
7:0	R	0	APERTURE	Aperture size. Identifies the MMIO aperture size in units of 4 kB for the AO block. AO has an MMIO aperture size of 4 kB. Aperture = 0: 4 kB.

AUDIO OUT 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Audio Out 2 registers begin at 0x11 2000. In all other respects, they are identical to the Audio Out 1 registers.				

AUDIO OUT 3 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Audio Out 3 registers begin at 0x11 4000. In all other respects, they are identical to the Audio Out 1 registers.				

Chapter 22: SPDIF Input Ports

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

22.1 Introduction

The SPDIF Input port accepts serial digital input that complies with the IEC60958 format specification for audio bitstreams. The interface locks onto and decodes the incoming “bi-phase mark” encoded signal and recognizes all preambles associated with the IEC60958 audio format.

The SPDIF Input is intended to accommodate digital audio generated by peripheral devices, such as CD and DVD players.

Key features are as follows:

- Clock extraction and decoding of incoming “bi-phase-mark” encoded serial bitstream
- Recognition of all preamble types: B, M, W
- Support for 16 to 24-bit PCM coded or non-PCM coded data types
- DMA of incoming audio samples into memory
- Raw mode 32-bit capture of incoming subframes
- Interrupt on parity or validity error as well as others
- Memory data formats compatible with PNX2700 Audio Out function
- Internal loopback with SPDIF out (diagnostic mode)
- Support for IEC61937 non-PCM bitstream format
- Capture of channel status and user information to MMIO registers

22.2 Functional Description

The SPDIF block diagram is shown in Figure 1. The SPDIF receiver resamples the incoming bitstream at a much higher clock rate than the source bit rate. During this process, the SPDIF bit clock and data are recovered. This resampled “synchronous” audio stream is then passed to the SPDIF decoder.

Internally, the SPDIF decoder produces a decoded binary representation of the ‘bi-phase’ data stream. At its output, the decoder produces a framed audio data format with separate framing, data and clock signals. This framed audio data is fed to the DMA unit for storage in main memory. In addition, during the decode phase, the input stream is processed to extract parity, validity and selected channel status information for each IEC60958 block.

The two hardware DMA buffers are 64 bytes each. During SPDIF capture, the hardware alternately fills one buffer while unloading the other into main memory.

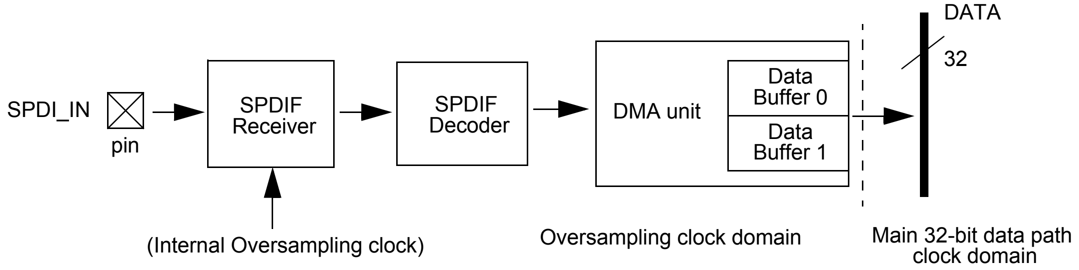

SPDI_IN
pin
SPDIF Receiver
SPDIF Decoder
DMA unit
Data Buffer 0
Data Buffer 1
DATA
32
(Internal Oversampling clock)
Oversampling clock domain
Main 32-bit data path clock domain

Figure 1: . SPDIF Input Block Diagram

The SPDIF In module operates using two clock domains. From Figure 1, the SPDIF receiver, decoder and DMA unit use an oversampling clock. The outbound side of the DMA unit uses the main PI data bus clock for all processing. Due to the need to use different clocks for the SPDIF In block, clock synchronization is done as necessary between the outbound and inbound sides of the DMA unit.

22.2.1 External Interface

The SPDIF input has a single input pin. The signal applied to this pin must have TTL level voltage swing.

Table 1: SPDIF Pin Summary

Signal	Type	Description
SPDIF_IN	IN	Single ended SPDIF input pin. Input sample rate can be 32KHz, 44.1KHz, 48KHz or 96KHz. Input signal must be TTL compatible.

For the commonly found 0.5 Vpp SPDIF signal (IEC60958 consumer mode), the user must externally restore the signal to TTL voltage levels. In all cases, external isolation of the input signal is recommended.

IEC60958 specifies that consumer systems have a 0.5 Vpp signal driven from the transmitter into an unbalanced cable with a 75 ohm nominal impedance. The load side must present a 75 ohm resistive impedance over the frequency band of 0.1 to 6 Mhz. Figure 2 presents an input circuit that satisfies the load requirements. The circuit presents a simple RS422 differential receiver. The chosen receiver should have good input hysteresis. Also, the signal applied to the SPDIF input pin should ideally have a 50% duty cycle. It is recommended that the system designer add an isolation transformer to the input circuit. Other consumer input circuits may be possible.

Remark: In Figure 2 the actual system design may vary.

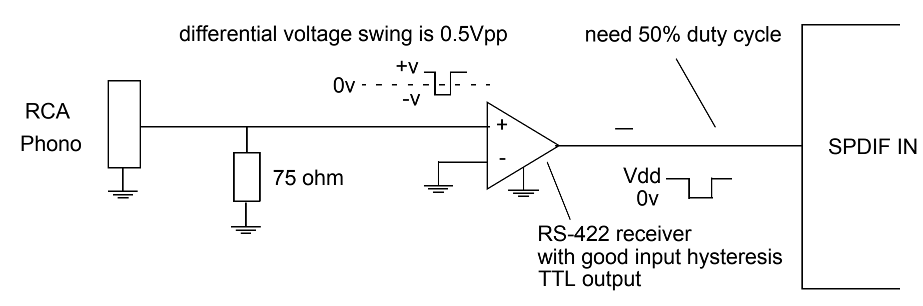


Figure 2: Sample SPDIF Consumer Interface

22.3 Operation

22.3.1 The SPDIF Bitstream

The SPDIF bitstream is composed of a single signal that is organized into a block structure of 192 frames. The signal contains both data and an embedded clock. Each frame is composed of two 32-bit subframes. The stream is encoded with a line code called “bi-phase mark” encoding. Figure 3 shows the organization of the IEC60958 SPDIF stream format.

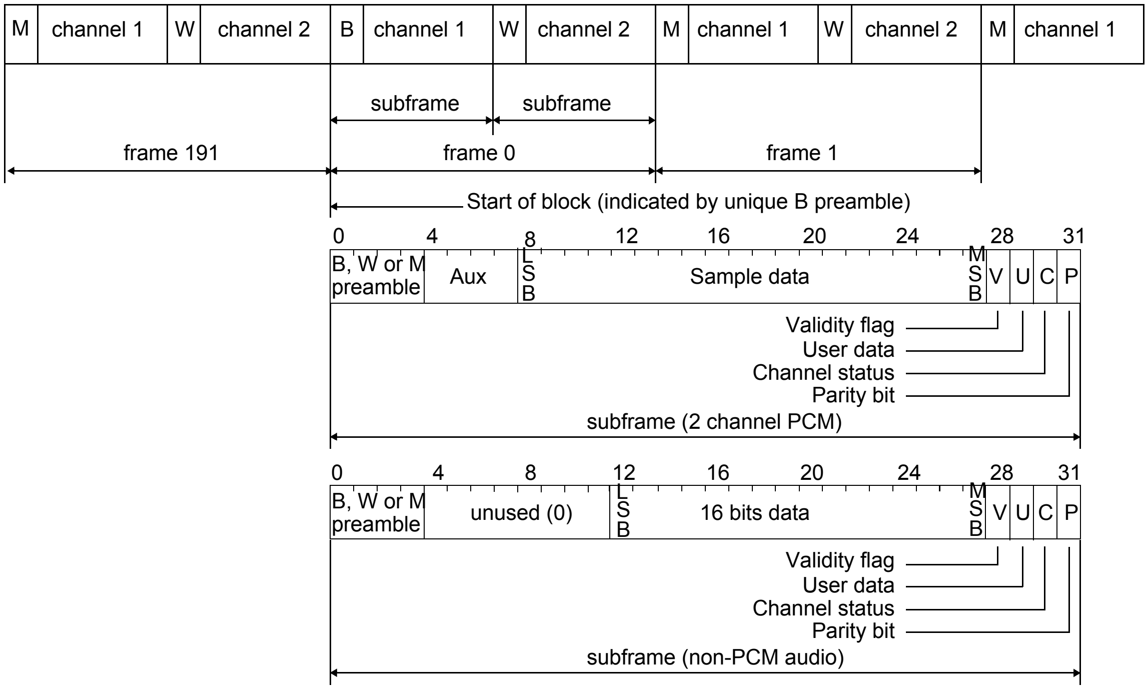


Figure 3: Serial Format of an IEC60958 Block

The input stream is parsed by the hardware using an extracted bit clock that is synchronous to the oversampling clock. The audio data, validity flag, channel status and parity bits are extracted and the SPDI_STATUS and SPDI_CBITS registers are updated. The audio portion of each subframe can hold samples that are up to 24 bits in length.

22.3.2 SPDIF Receiver Oversampling Clock

The source input stream is oversampled by the SPDIF receiver and a representation of the bi-phase data bitstream is produced along with a separate internal 64 *Fs* bit clock. The oversampling clock used to resample the input stream is a low jitter, divided form of the system PLL clock. The frequency of the oversampling clock must be within a certain frequency range described by the following equation:

$$1220\ \mathrm{Fs} \le \mathrm{Fosclk} \le 2400\ \mathrm{Fs}$$

where *Fs* is the incoming sample rate.

To guarantee error-free capture for all sample rates, the oversampling frequency *Fosclk* must be set to a nominal value. The SPDIF receiver's internal oversampling clock frequency can be programmed by selecting a clock divider setting in the central clock functional block. (For oversampling/clock programming details, see register CLK_SPDI_CTL (offset 0x04 7258) in Chapter 5 Clock Reset and Power Management.) Table 2 shows the divider selections and clocks that are produced.

Table 2: SPDIF In Oversampling Clock Value Settings

Input Audio Sample Rate: fs (KHz)	Central PLL Base Frequency (64x27MHz)/4	Central Clock Divider n	Fosclk: Oversampling Clock Freq (MHz)
96	432 Mhz	3	144.00
32.0, 44.1 and 48.0	432 Mhz	6	72.0

22.3.2.1 SPDIF Receiver Sample Rate Tolerance and IEC60958

Three levels of sampling frequency accuracy are specified in the IEC60958 document. The SPDIF receiver will achieve lock onto a level III signal (*variable pitch shift of +/- 12.5% of Fs*) with respect to all of the standard sampling frequencies; 32 KHz, 44.1 KHz and 48 KHz as well as the higher 96 KHz. For this design, the SPDIF receiver is classified as a level III compliant receiver. (Reference *Digital SPDIF Input for MADRE2* by H. Veerkamp, A. Mol, R. Takken, Nijmegen, February 1999.)

22.3.2.2 SPDIF Receiver Jitter Tolerance

The maximum tolerable input jitter of the SPDIF receiver is described by the equation below, or 0.26 UI pk-pk (1 UI = 1/128*Fs*).

$$\mathrm{Tmax(jitter)} = \pm 0.13 \times \frac{1}{128\mathrm{Fs}}$$

For a particular *Fs*, the max jitter is shown in Table 3.

Table 3: Input Jitter for Different Sample Rates

Fs (kHz)	1 UI = 1/(128Fs) (nsec)	Max Jitter = 0.26UI pk-pk (nsec)
32	244.1	31.7
44.1	177.2	23.0
48	162.8	21.2
96	81.4	10.6

The SPDIF receiver will reproduce the input data and clock without error if the maximum input jitter remains within the specified max jitter tolerance above. The receiver design meets and exceeds the IEC60958-3 consumer jitter requirements specification (i.e. 0.25 UI pk-pk between 200Hz and 400KHz jitter frequency).

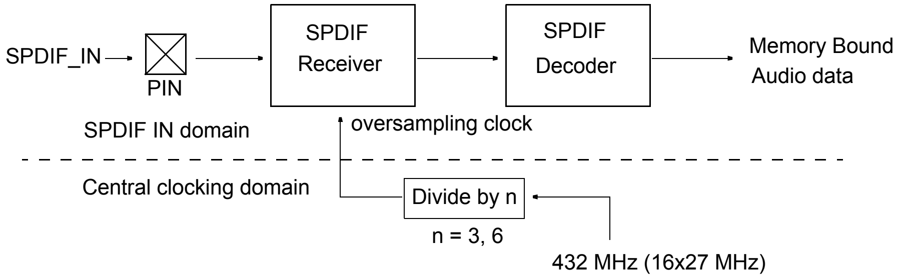


Figure 4: Oversampling Clock Generation

22.3.3 Memory Formats

The input bitstream is parsed and audio samples are captured and packed as 32-bit words prior to being written to memory. The SPDIF decoder always decodes the bi-phase encoded samples into binary prior to transfer to memory. See Figure 6 for the memory formatting for each of the sample sizes supported by the SPDIF IN block.

- **16-bit mode**: The input samples are packed into 32-bit words consisting of two 16-bit samples per 32-bit word.
- **32-bit mode**: For 17 through 24-bit audio, the samples are formatted into 32-bit words and placed in memory at consecutive 32-bit addresses. For these sample sizes, the sample is first combined with a programmable bitmask SPDI_SMPMASK.SMASK. The result of the mask operation is zero extended, at the least significant end, to the full 32-bits before being placed in memory. The resultant 32-bit words are of the form: 0xnnnnmm00 where the “n’s” are the 16 msbits of the sample, the “m”s are the masked eight lsbits subject to SMASK (see Section 22.3.6).
- **Raw capture mode**: Input subframes are captured and all ‘bi-phase’ encoding (bits [31:4]) is replaced with a binary representation. The preamble portion (bits [3:0]) of the subframe is replaced with a code indicating what preamble was present (see Figure 5). All parts of the subframe are assembled in order and this

32-bit word is then placed in memory. This mode can be used for "pass-through" of audio to an external SPDIF OUT block. The external SPDIF OUT block must be configured appropriately. (Reference Chapter 23 SPDIF Output Port.)

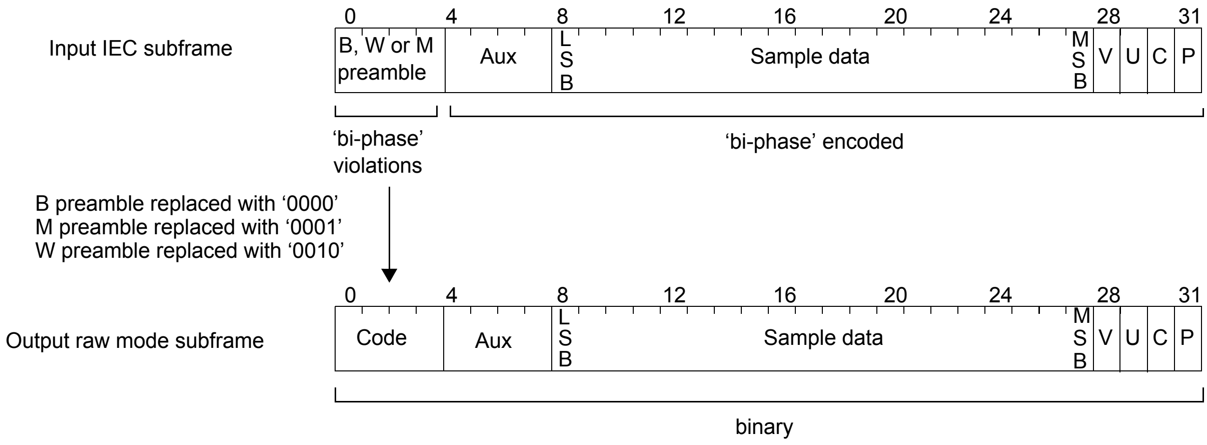
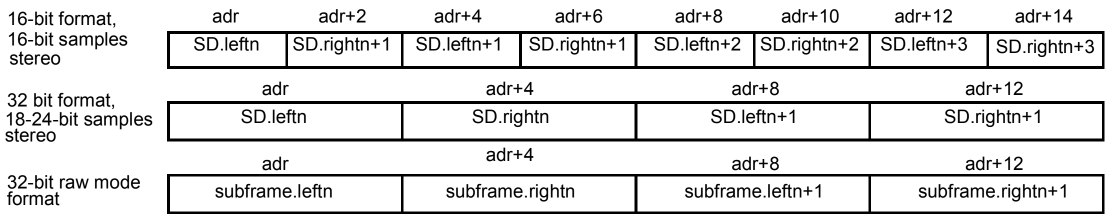


Figure 5: Raw Mode Format



16-bit format, 16-bit samples stereo	adr	adr+2	adr+4	adr+6	adr+8	adr+10	adr+12	adr+14
	SD.leftn	SD.rightn+1	SD.leftn+1	SD.rightn+1	SD.leftn+2	SD.rightn+2	SD.leftn+3	SD.rightn+3

32 bit format, 18-24-bit samples stereo	adr	adr+4	adr+8	adr+12
	SD.leftn	SD.rightn	SD.leftn+1	SD.rightn+1

32-bit raw mode format	adr	adr+4	adr+8	adr+12
	subframe.leftn	subframe.rightn	subframe.leftn+1	subframe.rightn+1

Figure 6: Sample Order View of Memory

22.3.3.1 SPDIF In Endianness

The SPDIF In block can store data in memory using either big-endian or little-endian formatting. The SPDIF In module reads a chip level endianness control signal and will slave to this signal for normal capture operation. This global "sys_big_end" control bit determines how increasing memory addresses map to byte positions within 16-bit or 32-bit words. The SPDIF In default state uses this "global" system endianness control input. The endian control can be changed for SPDI without resetting the module. However, it is necessary to disable capture prior to changing the system endian control for SPDI.

The format in memory for both little and big-endian byte ordering is shown in Figure 7.

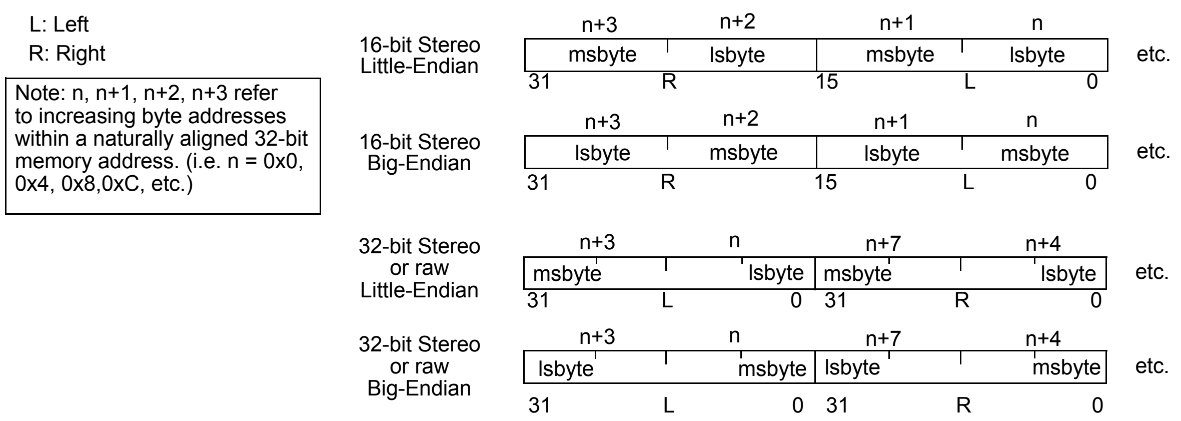

L: Left
R: Right
Note: n, n+1, n+2, n+3 refer to increasing byte addresses within a naturally aligned 32-bit memory address. (i.e. n = 0x0, 0x4, 0x8,0xC, etc.)
16-bit Stereo Little-Endian
n+3
n+2
n+1
n
msbyte
lsbyte
msbyte
lsbyte
31
R
15
L
0
etc.
16-bit Stereo Big-Endian
n+3
n+2
n+1
n
lsbyte
msbyte
lsbyte
msbyte
31
R
15
L
0
etc.
32-bit Stereo or raw Little-Endian
n+3
n
n+7
n+4
msbyte
lsbyte
msbyte
lsbyte
31
L
0
31
R
0
etc.
32-bit Stereo or raw Big-Endian
n+3
n
n+7
n+4
lsbyte
msbyte
lsbyte
msbyte
31
L
0
31
R
0
etc.

Figure 7: Endianness Byte Address Memory Format

22.3.4 SPDI_CBITSx and Channel Status Bits

The channel status block indicates the status of the currently received audio stream. The channel status block structure is different for each of the consumer or professional IEC60958 formats. Within each 32-bit subframe is a channel status bit at location bit[30] in the 32-bit word. The two ‘C’ bits, one for each of the subframes within a frame, can be different. Registers SPDI_CBITS1...SPDI_CBITS6 hold channel status bits embedded in the source SPDIF stream.

Register SPDI_CTL.UCBITS_SEL determines which set (subframe 1 or 2) of 192-channel status bits will be captured.

The SPDI_CBITSx registers are updated on a block basis. Upon the occurrence of each new B preamble in the source stream, these registers are updated with selected channel status information from the previous block. Programmers can use the SPDI_CBITSx registers to determine the state of the SPDIF source material. Information can be determined on whether the source stream type is consumer or professional, as well as the stream’s sample rate and size.

A selected set of the channel status bits captured by the SPDI_CBITS registers are shown in Table 4 and Table 5. (Reference *IEC60958-1 ed 2, Digital Audio Interface*, “Part 1: General; Part 2: Professional Applications; Part 3: Consumer Applications” for complete information regarding all consumer and professional channel status bit meanings.)

Table 4: SPDI_CBITS1 Channel Status Meaning

SPDI_CBITS 1[n]	IEC Consumer Channel Status Bit No.	IEC Consumer Meaning	AES/EBU Professional Channel Status Bit No.	AES/EBU Professional Meaning
0	0	Consumer mode	0	Professional mode
1	1	PCM/non-PCM data	1	PCM/non-PCM data
2	2	Copyright	2	Emphasis
3	3	Format	3	Emphasis

Table 4: SPDI_CBITS1 Channel Status Meaning *…Continued*

SPDI_CBITS 1[n]	IEC Consumer Channel Status Bit No.	IEC Consumer Meaning	AES/EBU Professional Channel Status Bit No.	AES/EBU Professional Meaning
4	4	Format	4	Emphasis
5	5	Format	5	Locked
6	6	Mode	6	Sample rate
7	7	Mode	7	Sample rate
8	8	Category code	8	Channel mode
9	9	Category code	9	Channel mode
10	10	Category code	10	Channel mode
11	11	Category code	11	Channel mode
12	12	Category code	12	User bit mgnmt
13	13	Category code	13	User bit mgnmt
14	14	Category code	14	User bit mgnmt
15	15	Category code	15	User bit mgnmt
16	16	Source Number	16	Use of aux bits
17	17	Source Number	17	Use of aux bits
18	18	Source Number	18	Use of aux bits
19	19	Source Number	19	Source word length - Source encode history
20	20	Channel number	20	Source word length - Source encode history
21	21	Channel number	21	Source word length - Source encode history
22	22	Channel number	22	Source word length - Source encode history
23	23	Channel number	23	Source word length - Source encode history
24	24	Sample rate	24	Multi-channel function
25	25	Sample rate	25	Multi-channel function
26	26	Sample rate	26	Multi-channel function
27	27	Sample rate	27	Multi-channel function
28	28	Clock accuracy	28	Multi-channel function
29	29	Clock accuracy	29	Multi-channel function
30	30	Reserved	30	Multi-channel function
31	31	Reserved	31	Multi-channel function

Table 5: SPDI_CBITS2 Channel Status Meaning

SPDI_CBITS 2[n]	IEC Consumer Channel Status Bit Number	IEC Consumer Meaning	AES/EBU Professional Channel Status Bit Number	AES/EBU Professional Meaning
0	32	Word length	32	Digital audio reference signal
1	33	Word length	33	Digital audio reference signal
2	34	Word length	34	Reserved
3	35	Word length	35	Reserved
4:31	36:63	*	36:63	*

* Reference *IEC60958-1 ed 2, Digital Audio Interface*, “Part 1: General; Part 2: Professional Applications; Part 3: Consumer Applications”

22.3.5 SPDI_UBITSx and User Bits

The complete set of user data channel bits are available in the SPDI_UBITSx registers. The SPDI_UBITSx registers are updated on a block basis. Upon the occurrence of each new B preamble in the source stream, the SPDI_UBITSx registers are updated with user data bit information from the previous block. The meaning of the user data is application dependent. The SPDI_CTL.UCBITS_SEL register determines which set (subframe 1 or 2) of 192 user data bits will be captured. (Refer to IEC60958-1 ed2, Digital Audio Interface, Part 1: General, Part 2: Professional Applications and Part 3: Consumer Applications for complete information regarding all consumer and professional user data channel bit meanings.)

22.3.6 SPDI_SMPMASK and Sample Size Masking

The SPDI_SMPMASK register allows per bitmasking on the least significant eight bits of the incoming samples (corresponding to subframe bits [11:4]). The SMASK setting only applies to 32-bit capture mode (i.e., SAMP_MODE = 01). The eight bits of SMASK will determine which subframe bits [11:4] will be captured and stored in memory. To reject a particular bit (within subframe bits [11:4]) in an audio sample, set the corresponding SMASK[7:0] bit to logic ‘1’. The default value of SMASK is 0x00.

Note the sense of the mask operation. Setting SMASK[7:0] bits to logic ‘1’ will zero the corresponding subframe bit [11:4]. Others will pass unchanged. The masking operation applies to all memory bound samples.

22.3.7 SPDI_BPTR and the Start of an IEC60958 Block

During SPDIF In capture, memory buffers are continuously filled with input data. As input blocks are filling the memory buffers, the address of the first instance of a frame 0 in a particular memory buffer changes continuously. To aid software with the task of finding the start of a block in memory, the SPDI_BPTR contains the address of the first occurrence of a frame 0 (indicating the starting boundary of a complete 192 frame block) within the currently filling memory buffer—either BUF1 or BUF2. This function is useful during capture of non-PCM coded data as found in IEC61937 data streams. The software driver can use SPDI_BPTR to find the beginning of an IEC60958 block and then quickly determine the location of any sync condition thereafter embedded in the non-PCM data structure.

22.3.8 Signal Lock and Errors

Figure 8 is a start-up software process flow for capture of an SPDIF In stream.

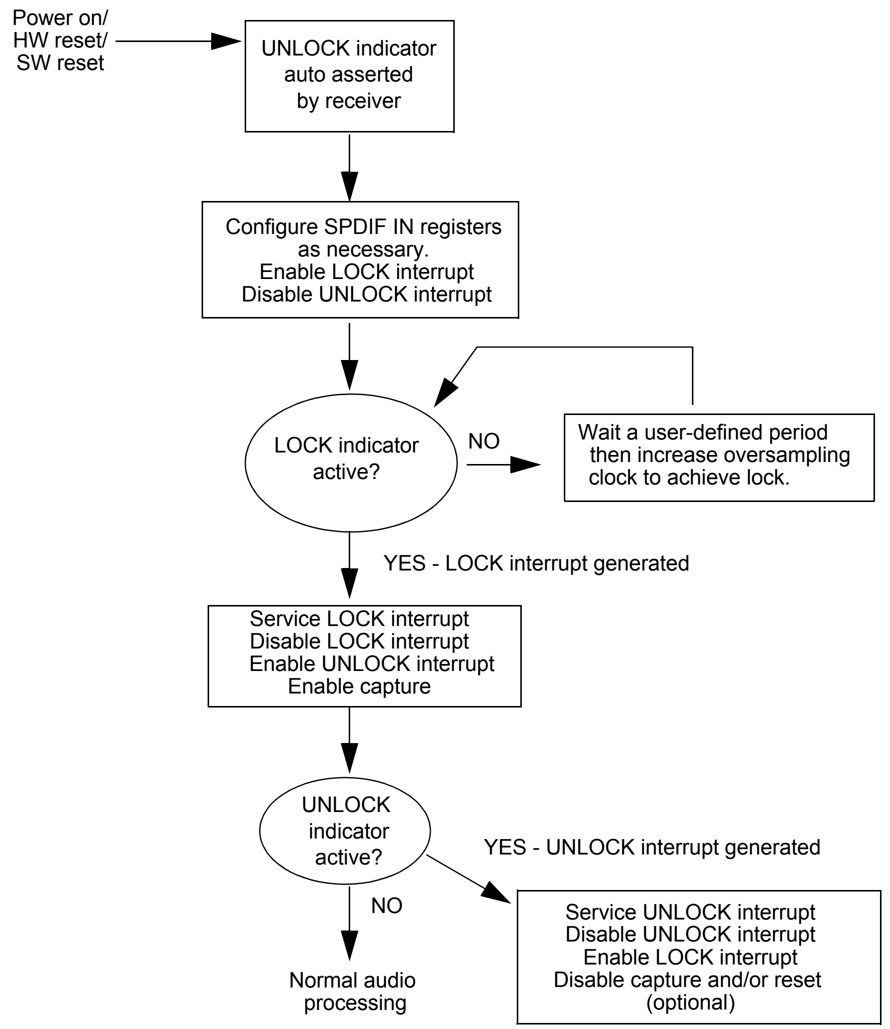


Figure 8: Lock/Unlock Processing for SPDIF In

22.3.8.1 LOCK and UNLOCK State Behavior

Shortly after power on (or any reset), the receiver will indicate that it is not locked to an input stream by asserting the UNLOCK bit in SPDI_STATUS. Later, once a valid SPDIF stream is applied to the interface, the receiver will assert the LOCK status bit.

LOCK means that the internal PLL is locked. A valid sequence of preambles is not required for LOCK. At this point, the receiver has determined that the input stream is valid and that DMA capture can be started by the user.

Remark: The status bits, LOCK and UNLOCK are sticky and may become activated regardless of the state of the SPDIF capture enable bit. Software must explicitly clear the LOCK and UNLOCK status bits using the appropriate SPDI_INTCLR register bits.

Also, the LOCK and UNLOCK behavior applies to all capture modes. For raw mode processing, parity and validity bits are ignored. PERR or VERR status bits are not updated.

22.3.8.2 UNLOCK Error Behavior and DMA

If the receiver should encounter an error condition in the stream, it will indicate this by asserting the UNLOCK bit. During run time the conditions that can cause the UNLOCK state are:

- an unexpected bi-phase error is encountered
- the input stream is not present, or is suddenly removed
- the input signal contains too much jitter.

If the UNLOCK_ENBL bit is set, the SPDIF IN will generate an interrupt. The parity and validity errors (PERR and VERR) do not cause out-of-lock conditions.

From the point where the error condition occurred, the contents of the currently filling internal 64-byte buffers are muted (zeroed). The external memory buffers will receive muted data from this point forward. If the receiver does not relock before the current external memory buffer is filled to completion with muted data, DMA will halt. DMA is halted in this way so that bus resources are not further utilized. Otherwise, DMA continues with valid data soon after lock is reacquired.

From the error point onward, the last stable capture sample rate will be maintained by the hardware automatically during this error condition processing.

22.3.9 Interrupts

The SPDI_STATUS register contains status flags that indicate certain conditions that may need the attention of the chip level controller. Each of these conditions can be used as interrupt sources.

The SPDI_INTEN register is used to provide the capability to enable any of the interrupt source bits in the SPDI_STATUS register. To enable one of the interrupts shown in the SPDI_STATUS register, the programmer must set the corresponding SPDI_INTEN bits for that interrupt source. For example, to allow an interrupt to be passed to the chip level interrupt controller upon the occurrence of a parity error in the incoming stream during capture, the software must write a logic ‘1’ to the PERR_ENBL bit in SPDI_INTEN. To disable an interrupt source, write a logic ‘0’ to the appropriate bit in SPDI_INTEN. The effect of writing a logic ‘0’ to an enable bit while the particular interrupt is active is that the interrupt is unconditionally de-asserted and disabled.

The status conditions in the SPDI_STATUS register will be “sticky,” meaning the SPDI_STATUS bit will remain active until explicitly cleared. Using the same example above, if the parity error bit PERR is enabled (SPDI_INTEN.PERR_ENBL = 1) and the PERR interrupt is active currently, to clear the PERR bit and deactivate the interrupt the user must write a logic ‘1’ to the PERR_CLR bit in the SPDI_INTCLR register.

The SPDI_INTSET register is useful for software diagnostic generation of interrupts. Setting any of these bits to logic ‘1’ will generate an interrupt to the chip level interrupt controller. To use the SPDI_INTSET register to generate interrupts, the same enable rule applies as outlined above.

For SPDIF In, the hardware interrupt signal is “level triggered.” This means the interrupt signal passed to the chip level interrupt controller will be logic ‘1’ when active and will remain so until cleared explicitly by the system interrupt handler software.

22.3.10 Event Timestamping

SPDIF In exports several event notification signals to the central DVP timestamping function. The central timestamp function includes timers and timestamp registers to provide event counts and event triggered “snapshot” clock values. The event signal is a positive edge going pulse with positive level duration greater than or equal to 160 ns.

The specific events that are exported to the central timestamp function are as follows:

- WS (Word strobe)—This event signals the arrival of a sample pair on the SPDIF IN interface. The rising edge of the signal indicates the beginning of either the B or M subframe. The logic “high” duration is as stated above.
- SWS (Last subframe)—This event signal indicates that a single 32-bit subframe *corresponding to the last sample in the currently filling memory buffer* has been received at the input to SPDIF In. The event is NOT qualified with a particular block boundary. This represents a precise, periodic event for use by system software to achieve audio/video synchronization.
- All SPDI_STATUS register bits (except LOCK)—These events can be used by software to either count or timestamp any interrupt generated by SPDIF IN. Refer to Figure 8 for details regarding SPDIF In interrupt sources.

22.3.11 Bandwidth and Latency Requirements

Normally, the rate of transmission of frames corresponds exactly to the source sampling frequency. The maximum data bus latency requirement will be for 96KHz streams (i.e. frame rate = 96KHz) with the SPDIF input set up for any of the 32-bit capture modes:

(96K frames/sec) x (8bytes/ frame) = 0.768Mbytes/sec

The maximum data bus latency allowed in order to sustain this transfer rate is (assuming data transfers are 64 bytes each):

64 bytes/N sec = 0.768 Mbytes/sec

Solving for N and providing a relation:

$N \leq 83.33$ uSec

For error-free operation during sustained DMA, there needs to be one 64-byte DMA write transfer completed to memory every 83 uSecs (or better) after a 64-byte hardware buffer has been filled. This will guarantee the latency requirement for the worst case input sample rate. If the latency requirement is not met, the hardware will set the HBE bit in the SPDI_STATUS register to logic ‘1’ indicating a data bus bandwidth error. For this condition, one or more audio samples will have been lost and will not be recoverable. The data bus arbitration for the SPDIF input block should be adjusted to satisfy this latency requirement.

22.3.12 Timestamping Control Register

Timestamping is available for a number of SPDIF input events. This control is done via the Global 1 register SPDI_MUX_SEL. This register is shown on page 22-471 of this chapter.

22.4 Register Descriptions

The register set is composed of status and control functions necessary to configure SPDIF In data capture and DMA of audio data to main memory.

The base address for the PNX8500 SPDIF Input Port module is 0x10 A000.

22.4.1 Register Address Map

Table 6: SPDIF Input Module Register Summary

Offset	Name	Description
0x10 A000	SPDI_CTL	SPDIF In general control register.
0x10 A004	SPDI_BASE1	Base address of buffer 1.
0x10 A008	SPDI_BASE2	Base address of buffer 2.
0x10 A00C	SPDI_SIZE	Size of the buffers.
0x10 A010	SPDI_BPTR	Contains the address of the first occurrence of a frame 0.
0x10 A014	SPDI_SMPMASK	Masking for the least significant 8 bits of the incoming sample.
0x10 A018	SPDI_CBITS1	Channel Status register 1.
0x10 A01C	SPDI_CBITS2	Channel Status register 2.
0x10 A020	SPDI_CBITS3	Channel Status register 3.
0x10 A024	SPDI_CBITS4	Channel Status register 4.
0x10 A028	SPDI_CBITS5	Channel Status register 5.
0x10 A02C	SPDI_CBITS6	Channel Status register 6.
0x10 A030	SPDI_UBITS1	User bits register 1.
0x10 A034	SPDI_UBITS2	User bits register 2.
0x10 A038	SPDI_UBITS3	User bits register 3.
0x10 A03C	SPDI_UBITS4	User bits register 4.
0x10 A040	SPDI_UBITS5	User bits register 5.
0x10 A044	SPDI_UBITS6	User bits register 6.
0x10 A048—AFDC	Reserved	
0x10 AFE0	SPDI_STATUS	SPDIF In status register.
0x10 AFE4	SPDI_INTEN	SPDIF In interrupt enable register.
0x10 AFE8	SPDI_INTCLR	SPDIF In interrupt clear register.

Table 6: SPDIF Input Module Register Summary …Continued

Offset	Name	Description
0x10 AFEC	SPDI_INTSET	SPDIF In interrupt select register.
0x10 AFF4	SPDI_PWR_DWN	Powerdown register
0x10 AFFC	SPDI_MODULE_ID	Module ID.

Table 7: SPDIF Input Global 1 Register Summary

Address	Name	Description
0x06 3800	SPDI_MUX_SEL	Global 1 register

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 A000			*SPDI_CTL*	
31:9		-	Unused	
11:8	R/W	0	GL_FILTER	Input glitch filter control. These bits control the rejection of a glitch on the SPDI interface. 0000 = The Glitch Rejection Filter is disabled. 0001 .. 1111 = An incoming signal transition must remain stable for (programmed value + 1) rising edges of OSC_CLK, otherwise it is rejected as a glitch.
7	R/W	0	UCBITS_SEL	User/Channel status bits select. Selects the set of subframes from which the user and channel status bits are extracted and written to the SPDI_UBITSx and SPDI_CBITSx registers. This bit is activated only on a block boundary, meaning that the bit can be changed at any time via software, but the update of the SPDI_UBITSx and SPDI_CBITSx registers with the new information will wait until a complete block has been received at the SPDIF input. 0 = subframe 1 is selected, user and channel status bits are extracted/written to UBITSx and CBITSx registers. 1 = subframe 2 is selected. User and channel status bits are extracted/written to UBITSx and CBITSx registers.
6:5	R/W	0	CHAN_MODE[1:0]	00 = Capture stereo left/right sample pairs (Default). 01 = Capture mono primary (subframe 1) channel. 10 = Capture mono secondary (subframe 2) channel. 11 = Reserved Note: The channel mode should only be changed while capture is disabled (i.e. CAP_ENABLE = 0).

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4:3	R/W	0	SAMP_MODE[1:0]	**00 = 16-bit mode**. Subframe bits [27:12] inclusive are selected and stored. Hardware stores a single 16-bit word per subframe. If audio samples are actually larger than 16 bits, the most significant 16 bits of the audio sample will be selected and stored. **01= 32-bit mode**. Subframe bits [27:4] inclusive are selected and stored subject to SMASK. A 32-bit word is formed by bitwise masking the sample (subject to the value of SPDI_SMPMASK.SMASK) and padding ‘0’ bits to the least significant end of the 24 bits. The resultant 32-bit words are of the form: 0xnnnnmm00 where the *n*s are the 16 subframe bits [27:12] and the *m*s are the eight masked subframe bits [11:4]. This provides for any audio sample size from 17 to 24 bits. (See the SPDI_SMPMASK register description for operation of the SMASK feature). SMASK only applies for this sample mode. **10 = Raw capture mode**. The entire subframe is captured and stored. The bi-phase portion of the subframe (i.e., bits [31:4]) are decoded into binary. Bits [3:0] are replaced with a code. The entire 32 bits are then stored as one unit. 11 = Reserved Note: The sample mode should only be changed while capture is disabled (i.e., CAP_ENABLE = ‘0’).
2	R/W	0	DIAG_MODE	Diagnostic loopback mode. Used to diagnose the SPDIF IN block. 0 = The SPDIF IN input source is set to the SPDIF input pin (default). 1 = The SPDIF IN input source is set to the SPDIF OUT pin.
1	R/W	0	CAP_ENABLE	Writing a ‘1’ to this bit enables capture per the selected mode. Writing a ’0’ here stops any ongoing capture after completing any actions related to the current audio sample.
0	R/W	0	RESET	Writing a ‘1’ to this bit resets the SPDI block. The registers of the SPDI will all be reset to ‘0s’. This should be used with caution. Any ongoing capture will be interrupted.
Offset 0x10 A004			*SPDI_BASE1*	
31:6	R/W	0	BASE1	Selects the main memory buffer starting addresses used for DMA of audio data samples. Note: Any change to the SPDI_BASE1 register should only be done while a memory buffer is not being used by the hardware DMA.
5:0	R	0	Reserved	

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 A008			*SPDI_BASE2*	
31:6	R/W	0	BASE2	Selects the main memory buffer starting addresses used for DMA of audio data samples. Note: Any change to the SPDI_BASE2 register should only be done while a memory buffer is not being used by the hardware DMA.
5:0	R	0	Reserved	Hardwired to logic ‘0’
Offset 0x10 A00C			*SPDI_SIZE*	
31:6	R/W	0	SIZE (in bytes)	The size of the DMA buffers is specified in the SPDI_SIZE register. Note hardware limits the buffer size and starting address to be aligned to 64-byte addresses. Assignment to SPDI_BASE1, SPDI_BASE2 and SPDI_SIZE have no effect on the state of the SPDI_STATUS flags.
5:0	R	0	Reserved	Hardwired to logic ‘0’
Offset 0x10 A010			*SPDI_BPTR*	
31:0	R	0	ADDRESS	To aid software with finding the start of a block in memory, the SPDI_BPTR contains the address of the first occurrence of a frame 0 (indicating the starting boundary of a complete 192-frame block) within the currently filling memory buffer: BUF1 or BUF2. This is useful during capture of non-PCM coded data found in IEC61937 data streams.
Offset 0x10 A014			*SPDI_SMPMASK*	
31:8		-	Unused	
7:0	R/W	0x00	SMASK	Allows per bitmasking the least significant 8 bits of the incoming samples (corresponding to subframe bits [11:4]). The SMASK setting only applies to 32-bit capture mode (i.e., SAMP_MODE = 01). The 8 bits of SMASK will determine which subframe bits [11:4] will be captured and stored in memory. Note: Setting SMASK[7:0] bits to logic ‘1’ will zero the corresponding subframe bit [11:4]. Others will pass unchanged.
Offset 0x10 A018			*SPDI_CBITS1*	
31:0	R	0	CBITS [31:0]	Channel Status register 1 contains bytes 0, 1, 2 and 3 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL. It will always reflect the condition of the current decoded block of 192 frames and will always start at the block boundary. Register bit meaning will depend upon the source transmission (i.e., consumer vs. professional).
Offset 0x10 A01C			*SPDI_CBITS2*	
31:0	R	0	CBITS [31:0]	Channel Status register 2 contains bytes 4, 5, 6 and 7 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A020			*SPDI_CBITS3*	
31:0	R	0	CBITS [31:0]	Channel Status register 3 contains bytes 8, 9,10 and 11 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 A024			*SPDI_CBITS4*	
31:0	R	0	CBITS [31:0]	Channel Status register 4 contains bytes 12, 13, 14 and 15 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A028			*SPDI_CBITS5*	
31:0	R	0	CBITS [31:0]	Channel Status register 5 contains bytes 16,17,18 and 19 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A02C			*SPDI_CBITS6*	
31:0	R	0	CBITS [191:159]	Channel Status register 6 contains bytes 20, 21, 22 and 23 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A030			*SPDI_UBITS1*	
31:0	R	0	UBITS [31:0]	User bit 1 contains the state of user bytes 0,1, 2 and 3 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A034			*SPDI_UBITS2*	
31:0	R	0	UBITS [31:0]	User bit 2 contains the state of user bytes 4, 5, 6 and 7 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A038			*SPDI_UBITS3*	
31:0	R	0	UBITS [31:0]	User bit 3 contains the state of user bytes 8, 9, 10 and 11 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A03C			*SPDI_UBITS4*	
31:0	R	0	UBITS [31:0]	User bit 4 contains the state of user bytes 12, 13, 14 and 15 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A040			*SPDI_UBITS5*	
31:0	R	0	UBITS [31:0]	User bit 5 contains the state of user bytes 16, 17, 18 and 19 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 A044			*SPDI_UBITS6*	
31:0	R	0	UBITS [191:159]	User bit 6 contains the state of user bytes 20, 21, 22 and 23 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A048—AFDC Reserved				

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 AFE0			*SPDI_STATUS*	
31:10		-	Unused	
9	R	0	UNLOCK	UNLOCK active. This flag gets set to logic ‘1’ if the SPDIF receiver is NOT locked onto an incoming stream. Programmers can use this UNLOCK indication, in conjunction with the LOCK bit, to determine the state of the receiver or to make a decision to adjust the oversampling frequency. See the definition of the LOCK bit. Possible causes of an out-of-lock state are: i) The oversampling frequency is too high or too low with respect to the applied input SPDIF sample rate. ii) Too much jitter in SPDIF input stream. iii) Absent, invalid or corrupted SPDIF stream applied to the interface/receiver. The flag can be cleared by a software write to UNLOCK_CLR.
8	R	0	UCBITS	User/Channel bits available. This flag gets set if a new set of user data bits and channel status bits have been written to the SPDI_UBITSx and SPDI_CBITSx registers. Updated on a block basis.
7	R	0	LOCK	LOCK active. 1 = The SPDIF receiver achieved lock onto the incoming stream. Use this LOCK flag, in conjunction with the UNLOCK flag, to determine the state of the receiver or to make a decision to adjust the oversampling frequency. The flag can be cleared by a software write to LOCK_CLR. LOCK means that the internal PLL is locked. A valid sequence of preambles is not required for LOCK.
6	R	0	VERR	Validity Error. 1 = The hardware encounters a subframe that has the validity flag set to 1, indicating that the payload portion of the subframe is not reliable. The flag can be cleared by a software write to VERR_CLR.

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
5	R	0	PERR	Parity Error. 1 = The hardware encounters a subframe that has a parity error. Parity is even for the subframe and applies to subframe bits [31:4] inclusive. Normally, the external SPDIF transmitter will set the subframe P bit to logic ‘1’ or logic ‘0’ so that bits [31:4] have an even number of logic “1s” and “0s”. The flag can be cleared by a software write to PERR_CLR.
4	R	0	OVERRUN	1 = Both external main memory DMA buffers are filled before a new empty buffer is assigned by the system control CPU. Hardware has performed a normal buffer switch over and is overwriting fresh, unconsumed data. This flag can be cleared by software write to OVR_CLR.
3	R	0	HBE (Bandwidth error)	Bandwidth Error. 1 = The internal hardware DMA buffers in SPDI are full and at least one of them was not emptied before new input data arrived on the SPDI interface, indicating that DMA service latency is too long. This flag can be cleared by a software write to HBE_CLR.
2	R	0	BUF1_ACTIVE	This flag gets set to logic ‘1’ if the hardware is currently filling memory DMA buffer 1. Otherwise, it is reset to logic ‘0’. This flag can be cleared by a software write to BUF1_ACTIVE_CLR.
1	R	0	BUF2_FULL	This flag gets set to logic ‘1’ if memory DMA buffer 2 has been filled by the SPDI hardware. It can be cleared by a software write to BUF2_FULL_CLR.
0	R	0	BUF1_FULL	This flag gets set to logic ‘1’ if memory DMA buffer 1 has been filled by the SPDI hardware. It can be cleared by a software write to BUF1_FULL_CLR.
Offset 0x10 AFE4			*SPDI_INTEN*	
31:10		-	Unused	
9	R/W	0	UNLOCK_ENBL	1 = UNLOCK bit in SPDI_STATUS is enabled for interrupts. 0 = UNLOCK bit in SPDI_STATUS is disabled for interrupts.
8	R/W	0	UCBITS_ENBL	1 = UCBITS bit in SPDI_STATUS is enabled for interrupts. 0 = UCBITS bit in SPDI_STATUS is disabled for interrupts.
7	R/W	0	LOCK_ENBL	1 = LOCK bit in SPDI_STATUS is enabled for interrupts. 0 = LOCK bit in SPDI_STATUS is disabled for interrupts.
6	R/W	0	VERR_ENBL	1 = VERR bit in SPDI_STATUS is enabled for interrupts. 0 = VERR bit in SPDI_STATUS is disabled for interrupts.
5	R/W	0	PERR_ENBL	1 = PERR bit in SPDI_STATUS is enabled for interrupts. 0 = PERR bit in SPDI_STATUS is disabled for interrupts.
4	R/W	0	OVR_ENBL	1 = OVERRUN bit in SPDI_STATUS is enabled for interrupts. 0 = OVERRUN bit in SPDI_STATUS is disabled for interrupts.
3	R/W	0	HBE_ENBL	1 = HBE bit in SPDI_STATUS is enabled for interrupts. 0 = HBE bit in SPDI_STATUS is disabled for interrupts.

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2	R/W	0	BUF1_ACTIVE_ENBL	1 = BUF1_ACTIVE bit in SPDI_STATUS is enabled for interrupts. 0 = BUF1_ACTIVE bit in SPDI_STATUS is disabled for interrupts.
1	R/W	0	BUF2_FULL_ENBL	1 = BUF2_FULL bit in SPDI_STATUS is enabled for interrupts. 0 = BUF2_FULL bit in SPDI_STATUS is disabled for interrupts.
0	R/W	0	BUF1_FULL_ENBL	1 = BUF1_FULL bit in SPDI_STATUS is enabled for interrupts. 0 = BUF1_FULL bit in SPDI_STATUS is disabled for interrupts.
Offset 0x10 AFE8			*SPDI_INTCLR*	
31:10		-	Unused	
9	R/W	0	UNLOCK_CLR	1 = Clear UNLOCK bit in SPDI_STATUS. 0 = No effect
8	W	0	UCBITS_CLR	1 = Clear UCBITS in SPDI_STATUS. 0 = No effect.
7	W	0	LOCK_CLR	1 = Clears LOCK in SPDI_STATUS. 0 = No effect.
6	W	0	VERR_CLR	1 = Clear VERR in SPDI_STATUS. 0 = No effect
5	W	0	PERR_CLR	1 = Clear PERR in SPDI_STATUS. 0 = No effect.
4	W	0	OVR_CLR	1 = Clear OVERRUN in SPDI_STATUS. 0 = No effect.
3	W	0	HBE_CLR	1 = Clear HBE in SPDI_STATUS. 0 = No effect.
2	W	0	BUF1_ACTIVE_CLR	1 = Clear BUF1_ACTIVE in SPDI_STATUS. 0 = No effect.
1	W	0	BUF2_FULL_CLR	1 = Clear BUF2_FULL in SPDI_STATUS. 0 = No effect.
0	W	0	BUF1_FULL_CLR	1 = Clear BUF1_FULL in SPDI_STATUS. 0 = No effect.
Offset 0x10 AFEC			*SPDI_INTSET*	
31:10		-	Unused	
9	W	0	UNLOCK_SET	1 = UNLOCK bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
8	W	0	UCBITS_SET	1 = UCBITS bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
7	W	0	LOCK_SET	1 = LOCK bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
6	W	0	VERR_SET	1 = VERR bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
5	W	0	PERR_SET	1 = PERR bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
4	W	0	OVR_SET	1 = OVERRUN bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
3	W	0	HBE_SET	1 = HBE bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
2	W	0	BUF1_ACTIVE_SET	1 = BUF1_ACTIVE bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
1	W	0	BUF2_FULL_SET	1 = BUF2_FULL bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect
0	W	0	BUF1_FULL_SET	1 = BUF1_FULL bit in SPDI_STATUS is to be set to logic ‘1’. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic ‘1’. 0 = No effect

SPDIF INPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 AFF4			*SPDI_PWR_DWN*	
31:1		-	Unused	
0	R/W	0	PWR_DWN	The bit is used to provide power control status for system software block power management.
Offset 0x10 AFFC			*SPDI_MODULE_ID*	
31:16	R	0x0110	Module ID	This field identifies the block as type SPDIF IN. SPDIF IN ID = 0x0110
15:12	R	0	MAJ_REV	Major Revision ID
11:8	R	0	MIN_REV	Minor Revision ID
7:0	R	0	APERTURE	Aperture size

22.4.2 Global 1 Registers

GLOBAL 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
SPDIF IN Timestamping Register				
Offset 0x06 3800			*SPDI_MUX_SEL*	
31:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/ W	0x0	SPDI_MUX_SEL	SPDIF IN timestamping. The specific events that may be timestamped are: 0000 = WS - Word strobe 0001 = SWS - Last subframe 0010 = SPDI_STATUS[0] - Buffer 1 full 0011 = SPDI_STATUS[1] - Buffer 2 full 0100 = SPDI_STATUS[2] - Buffer 1 active 0101 = SPDI_STATUS[3] - Bandwidth Error 0110 = SPDI_STATUS[4] - Parity Error 0111 = SPDI_STATUS[5] - Validity Error 1000 = SPDI_STATUS[6] - User/Channel bits available 1001 = SPDI_STATUS[7] - Unlock active

Chapter 23: SPDIF Output Port

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

23.1 Introduction

The PNX8526 has one SPDIF output port driven by the SPDIF Output module. The SPDIF Output module (SPDO) generates a 1-bit high speed serial data stream. The primary application is to provide SPDIF (Sony/Philips Digital Interface) data for use by external audio equipment. The SPDIF output has the following features:

- Fully compliant with IEC-60958 for both consumer and professional applications
- Supports two channel linear PCM audio with 16-24 bit/sample
- Supports one or more Dolby Digital AC-3 6-channel data streams embedded as IEC-61937
- Supports one or more MPEG-1 or MPEG-2 audio streams embedded per IEC-61937
- Allows arbitrary programmable sample rates from 1 Hz to 300 kHz
- SPDO can carry data with a sample rate independent of and asynchronous to the Audio Out sample rate
- SPDO hardware performs autonomous DMA of memory-resident IEC-60958 subframes assembled by software
- SPDO hardware performs parity generation and bi-phase mark encoding
- Software has full control over all data content, including User and Channel data

Remark: IEC-61937 is a generic specification for transmitting non PCM data with an IEC-60958 transport. It supports AC3, PTS, MPEG1 Layer 2, MPEG2 Layer 3 (MP3), MPEG 2 BC, MPEG 2 AAC and others.

It is also possible to use the SPDO signal as a general purpose high-speed data stream. Potential applications include use as a high-speed UART or high-speed serial data channel. In this case features are as follows:

- Up to 40 Mbits/s data rate
- Full software control over each bit cell transmitted
- lsb first or msb first data format

23.2 Functional Description

The SPDO module has two basic components: a DMA engine and an emitter. The emitter is clocked from the DDS (in the Clocks module) and can be programmed to the desired sample rate. The emitter delivers the data stream to the SPDIF Output pin.

Software initially gives SPDO two memory data buffers, then enables the SPDO block. As soon as the first memory buffer is drained, SPDO requests a new buffer from software while switching over to use the other memory buffer, and so on.

With exception of the DDS operation, the SPDO block is generally software compatible with that of the TM1300.

23.2.1 External Interface

Table 1: SPDIF Out External Signals

Signal	Type	Description
SPDIF_OUT	O	SPDIF output. Self-clocking interface carrying either two-channel PCM data with samples up to 24 bits, or encoded Dolby Digital (AC-3) or MPEG audio data for decoding by an external AC-3 or MPEG capable audio amplifier.

An external circuit as shown in Figure 1 is required to provide an electrically isolated output and convert the 3.3 V output pin to a drive level of 0.5 V peak-to-peak into a 75 Ohm load, as required for consumer applications of IEC-60958.

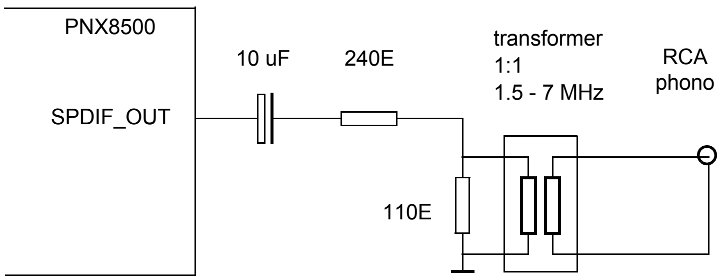


Figure 1: Suggested External SPDIF Out Interface Circuitry

23.3 Operation

Before enabling the SPDO block, software must assign two buffers with data to SPDO_BASE1, SPDO_BASE2 and a buffer size value (in bytes) to SPDO_SIZE. Each memory buffer size must be a multiple of 64 bytes, regardless of the operating mode.

The SPDO block is enabled by writing a ‘1’ to SPDO_CTL.TRANS_ENABLE. Once enabled, the first DMA buffer is sent out at the programmed sample rate. Once the first buffer is empty, BUF1_ACTIVE is negated and the BUF1_EMPTY flag in SPDO_STATUS is asserted. If BUF1_INTEN in SPDO_CTL is also asserted, an interrupt to the chip level interrupt controller is generated.

The SPDO block continues by emitting the data in DMA buffer 2. In normal operation, software assigns a new buffer 1 full of data to SPDO and signals this by writing a ‘1’ to ACK_BUF1. The SPDO block immediately negates the BUF1_EMPTY condition and the related interrupt request. Once buffer 2 is empty, similar signalling occurs and the hardware switches back to using buffer 1. Transmission continues uninterrupted until the unit is disabled.

The SPDO module has two operating modes: SPDIF and transparent DMA mode.

23.3.1 SPDIF Mode

SPDIF driver software assembles SPDIF data in each memory data buffer. Each memory data buffer consists of groups of 32-bit words. Each word describes the data to be transmitted for a single IEC-60958 subframe, including what type of preamble to include. Each subframe is transmitted in 64 clock intervals of the SPDO clock, a programmable clock generated by the SPDO Direct Digital Synthesizer (DDS). Note that the DDS resides in the central Clocks module.

The SPDIF mode may also be used as a source input stream for the SPDIF In block.

23.3.1.1 Sample Rate Programming

In SPDIF, the frame rate always equals f_s, the sample rate of embedded audio. This relation holds for PCM as well as for AC-3 and MPEG audio. Each frame consists of 128 Unit Intervals (UIs). The length of a UI is determined by the frequency setting of the SPDO Direct Digital Synthesizer (DDS) in the central Clocks module.

$$f_s = \frac{(f_{DDS})}{128}$$

The DDS can be programmed to emit on-chip frequencies from approximately 1 Hz to 80 MHz with a maximum jitter of less than 0.579 ns. Refer to Table 11 in Chapter 5 Clock Reset and Power Management for details.

Table 2 shows settings for common sample rate and main clock values.

Table 2: SPDIF Out Sample Rate Setting

f_s (kHz)	UI (nSec)	Jitter (nSec)
32.000	244.14	0.579
44.100	177.15	0.579
48.000	162.76	0.579
96.000	81.38	0.579

23.3.2 Transparent Mode

In transparent mode, software prepares each data bit exactly as it is to be transmitted, in a series of 32-bit words in each memory data buffer. (The 32-bit word is constructed according to the same rules for byte ordering as described in Section 23.4.2) Each 32-bit word is transmitted (LSB or MSB first) in 32 clock intervals of the DDS. The data is shifted out “as is,” without bi-phase mark encoding, parity generation or preamble insertion.

23.4 Data Formatting

23.4.1 IEC-60958 Serial Format

Figure 2 shows the serial format layout of an IEC-60958 block. A block starts with a special “B” preamble and consists of 192 frames. The sample rate of all embedded audio data is equal to the frame rate. Each frame consists of two subframes. Subframe 1 always starts with an “M” preamble, except for subframe 1 in frame 0, which starts with a “B.” Subframe 2 always starts with a “W” preamble.

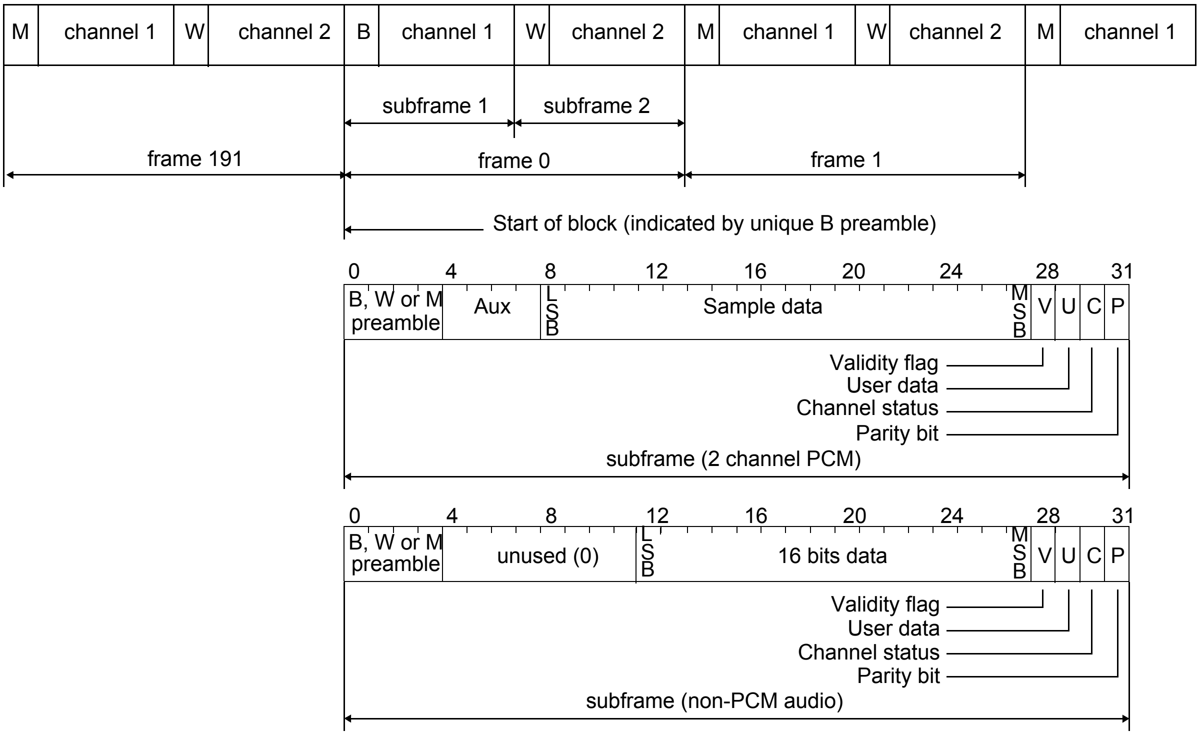


Figure 2: Serial Format of a IEC-60958 Module

When IEC-60958 data carries two-channel PCM data, one audio sample is transmitted in each subframe, “left” in subframe 1 and “right” in subframe 2. Each sample can be 16-24 bits, where the msb is always aligned with bit slot 28 of the subframe. In case of more than 20 bits per sample, the aux field is used for the four lsbits.

When IEC-60958 data carries non-PCM audio, such as one or more streams of encoded AC-3 data and/or MPEG audio, each subframe carries 16 bits of data. The data of successive frames adds up to a payload data stream which carries its own burst data.

This is described in the “Interface for non-PCM Encoded Audio Bitstreams Applying IEC958,” *Philips Consumer Electronics*, June 6 1997, IEC 100c/WG11 (prelim IEC-61937). Programmers should also refer to the I*EC-60958 Digital Audio Interface*, "Part 1: General; Part 2: Professional Applications; Part 3: Consumer Applications" for a precise description of the required values in each field for different types of consumer equipment.

The SPDO block hardware only generates B, W and M preambles as well as the P (Parity) bit. All other bits in the subframe are determined by software and copied "as is" from memory to output, subject only to bit-cell coding.

Software must construct valid IEC-60958 blocks by using the right sequence of 32-bit words as described in Section 23.4.2

23.4.1.1 IEC-60958 Bit Cell and Preamble

Each data bit in IEC-60958 is transmitted using bi-phase mark encoding. In bi-phase mark encoding, each data bit is transmitted as a cell consisting of two consecutive binary states. The first state of a cell is always inverted from the second state of the previous cell. The second state of a cell is identical to the first state if the data bit value is a "0," and inverted if the data bit value is a "1."

Preambles are coded as bi-phase mark violations, where the first state of a cell is not the inverse of the last state of the previous cell.

The duration of each state in a cell is called a Unit Interval (UI), so that each cell is two UIs long. In SPDO, the length of a UI is one SPDO clock cycle, as determined by the settings of the DDS (see Section 23.3.1.1).

Figure 3 illustrates the transmission format of an 8-bit data value "10011000", as well as the transmission format of the three preambles. Note that each preamble always starts with a rising edge. This is made possible by the presence of the parity bit, which always guarantees an even number of '1' bits in each subframe.

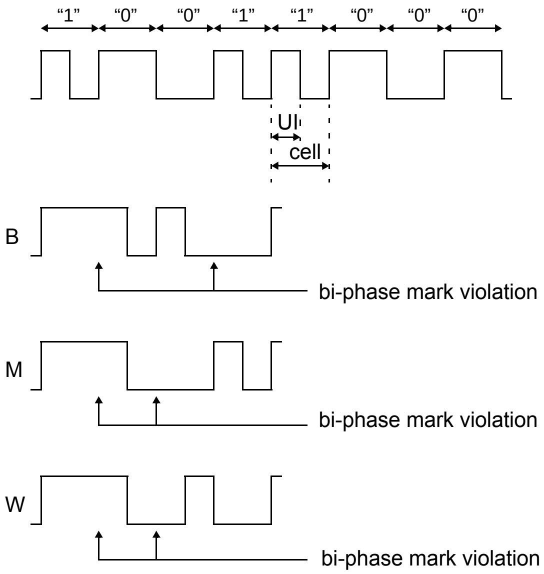


Figure 3: Bi-Phase Mark Data Transmission

23.4.1.2 IEC-60958 Parity

The parity bit, or P bit in Figure 2, is computed by the SPDIF Out hardware. The P bit value should be set such that bit cells 4 to 31 inclusive contain an even number of ones (and hence even number of zeroes). The P bit is bi-phase mark encoded using the same method as for all other bits.

23.4.2 IEC-60958 Memory Data Format

The system software must prepare a memory data structure that instructs the SPDIF block hardware to generate correct IEC-60958 blocks. This data structure consists of 32-bit words with the content described in Table 3.

Table 3: SPDIF Subframe Descriptor Word

Bits	Definition
31 (msb)	This bit must be a ‘0’ for future compatibility
30..4	Data value for bits 4..30 of the subframe, exactly as they are to be transmitted. Hardware will perform the bi-phase mark encoding and Parity generation.
3..0 (lsb)	0000 - generate a B preamble 0001 - generate an M preamble 0010 - generate a W preamble 0011 .. 1111 reserved for future

The data structure for a block consists of 384 of these 32-bit descriptor words, one for each subframe of the block, with the correct B, M, W values. All data content, including the U, C and V flag are fully under control of the software that builds each block.

A DMA buffer handed to the hardware is required to be a multiple of 64 bytes in length. It can contain one or more complete blocks, or a block may straddle DMA buffer boundaries. The 64-byte length will result in DMA buffers that contain a multiple of 16 subframes.

23.4.2.1 Endianness

The SPDIF descriptor is a 32-bit word memory data structure, and therefore is subject to endianness considerations. The SPDO block imports a chip level endianness control signal and will slave to this signal for normal transmit operation. This global “sys_big_end” control bit determines how increasing memory addresses map to byte positions within 32-bit descriptor words. The SPDO unit power-on default state uses this “global” system endianness control input.

23.5 Errors and Interrupts

23.5.1 DMA Error Conditions

Two types of errors can occur during DMA operation.

If the software fails to provide a new buffer of data in time, and both DMA buffers empty out, the SPDO hardware raises the UNDERRUN flag in SPDO_STATUS. Transmission switches over to the use of the next buffer, but the data transmitted is from the previously transmitted buffer. IF UDR_INTEN is asserted, an interrupt will be generated. The UNDERRUN flag is sticky—it will remain asserted until the software clears it by writing a ‘1’ to ACK_UDR.

A lower level error can also occur when the limited size internal buffer empties out before it can be refilled across the data bus. This situation can arise only if insufficient bandwidth is allocated to SPDO from the bus arbiter. In this case, the Highway Bandwidth Error (HBE) error flag is raised.

23.5.2 HBE and Data Bus Latency

The SPDO unit uses two internal 64-byte buffers and two 32-bit holding registers. Under normal operation, one internal buffer gets refreshed from SDRAM fast enough to avoid any missing samples, while data is meanwhile being emitted from the other 64-byte buffer and its holding register.

If the data bus arbiter is set up with an insufficient latency guarantee, a situation can arise where one 64-byte buffer is not refilled in time, so that the other buffer/holding register are exhausted by the time a new output sample is due. In that case the HBE error is raised, and the last sample for each channel will be repeated until the new buffer is refreshed. The HBE condition is sticky and can only be cleared by an explicit ACK_HBE, which indicates an incorrect setting of the data bus bandwidth arbiter.

The data bus arbiter needs to guarantee that the maximum latency required by the SPDO block can always be met.

Given an output data rate f_s in samples/sec, 2 x 32 bits are required for each sample interval. The arbiter should be set to have a latency so that the buffer is refilled before a sample interval expires. Refer to Table 4 for example latency requirements.

Table 4: SPDO Block Data Bus Latency Requirements

f_s (kHz)	1/f_s (nSec)	Max. latency (9/f_s) (uSec)
32.000	31250	281
44.100	22675	204
48.000	20833	187
96.000	10416	94

23.5.3 Interrupts

The SPDO block generates an interrupt if one of the following status bit flags, and its corresponding INTEN flag are set: BUF1_EMPTY, BUF2_EMPTY, HBE, UNDERRUN. See Section 23.3 for details.

All these status flags are ‘sticky.’ They are asserted by hardware when a certain condition occurs, and remain set until the interrupt handler explicitly clears them by writing a ‘1’ to the corresponding ACK bit in SPDO_CTL. The SPDO hardware takes the flag away in the clock cycle after the ACK is received. This allows immediate return from interrupt after the ACK.

The SPDIF Out module is a legacy function and is not strictly compliant with DVP module level interrupt hardware guidelines.

23.5.4 Timestamp Events

SPDO exports event signals associated with audio transmission to the central timestamp/timer function on-chip. The central timestamp/timer function can be used to count the number of occurrences of each event or timestamp the occurrence of the event or both. The event will be a positive edge pulse with the duration of the event to be greater than or equal to 200 ns. The specific event exported is as follows:

- BUF_DONE—signals a request by one of the internal 64-byte hardware buffers corresponding to the last 64 bytes of the current external DMA memory buffer. Note that this event is not dependent upon PI-Bus latency.

The occurrence of this event represents a precise, periodic time interval which can be used by system software for audio/video synchronization.

23.6 Register Summary and Descriptions

The base address for the PNX8526 SPDIF Output Port module is 0x10 9000.

23.6.1 Register Address Map

Table 5: SPDIF Output Module Register Summary

Offset	Name	Description
0x10 9000	SPDO_STATUS	SPDIF Out Status
0x10 9004	SPDO_CTL	SPDIF Out general control register
0x10 9008	Reserved	
0x10 900C	SPDO_BASE1	Base address of buffer 1
0x10 9010	SPDO_BASE2	Base address of buffer 2
0x10 9014	SPDO_SIZE	Size of the buffers
0x10 9018—9FF0	Reserved	
0x10 9FF4	SPDO_PWR_DWN	Powerdown
0x10 9FFC	SPDO_MODULE_ID	Module ID

SPDIF OUTPUT PORT REGISTERS

Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 9000			*SPDO_STATUS*	
Note: The clock frequency emitted by the DDS output can be found in Chapter 5 Clock Reset and Power Management registers at *Offset 0x04 711C DDS7_SPDO_CTL*				
31:5		-	Unused	
4	R	1	BUF1_ACTIVE	This flag gets set if the hardware is currently emitting DMA buffer 1 data and is negated when emitting DMA buffer 2 data.
3	R	0	UNDERRUN	This flag gets set if both DMA buffers were emptied before a new full buffer was assigned by software. The hardware has performed a normal buffer switch over and is emitting old data. IT can only be cleared by software write to ACK_UDR.

SPDIF OUTPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2	R	0	HBE (Bandwidth error)	Bandwidth Error. This flag gets set if the internal buffers in SPDO were emptied before new memory data was brought in. This flag can be cleared only by a software write to ACK_HBE.
1	R	0	BUF2_EMPTY	This flag gets set if DMA buffer 2 has been emptied by the SPDO hardware. The flag can be cleared only by a software write to ACK_BUF2.
0	R	0	BUF1_EMPTY	This flag gets set if DMA buffer 1 has been emptied by the SPDO hardware. The flag can be cleared only by a software write to ACK_BUF1.
Offset 0x10 9004			*SPDO_CTL*	
31	R/W	0	RESET	1 = Software Reset. Immediately resets the SPDO block. This should be used with extreme caution. Any ongoing transmission will be interrupted and receivers may be left in a strange state.
30	R/W	0	TRANS_ENABLE	1 = Enables transmission per the selected mode. 0 = Transmission Disabled. Stops any ongoing transmission after completing any actions related to the current data descriptor word.
29:27	R/W	000	TRANS_MODE	Transmission mode. 000 = IEC-60958 mode. Hardware performs bi-phase mark encoding, preamble and parity generation, and transmits one IEC-60958 subframe for each data descriptor word. 010 = Transparent mode, lsb first. The 32-bit data descriptor words are transmitted as is, lsb first. 011 = Transparent mode, msb first. The 32-bit data descriptor words are transmitted as is, msb first. Other codes are reserved for future extensions. Note: The transmission mode should only be changed while transmission is disabled.
26:8		-	Unused	
7	R/W	0	UDR_INTEN	If UDR_INTEN = 1 and UNDERRUN = 1, an interrupt is asserted to the chip level interrupt controller.
6	R/W	0	HBE_INTEN	If HBE_INTEN = 1 and HBE = 1, an interrupt is asserted to the chip level interrupt controller.
5	R/W	0	BUF2_INTEN	If BUF2_INTEN = 1 and BUF2_EMPTY = 1, an interrupt is asserted to the chip level interrupt controller.
4	R/W	0	BUF1_INTEN	If BUF1_INTEN = 1 and BUF1_EMPTY = 1, an interrupt is asserted to the chip level interrupt controller.
3	R/W	0	ACK_UDR	1 = Clear UNDERRUN. 0 = No effect. Always reads as 0.
2	R/W	0	ACK_HBE	1 = Clear HBE. 0 = No effect. Always reads as 0.
1	R/W	0	ACK_BUF2	1 = Clear BUF2_EMPTY. Informs SPDO that DMA buffer 2 is full. 0 = No effect. Always reads as 0.

SPDIF OUTPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	R/W	0	ACK_BUF1	1 = Clear BUF1_EMPTY. Informs SPDO that DMA buffer 1 is full. 0 = No effect. Always reads as 0.
Offset 0x10 9008			*Reserved*	
31:0	R	0x0	Reserved	Legacy register. Always reads as ‘0’.
Offset 0x10 900C			*SPDO_BASE1*	
31:6	R/W	0x0	BASE1	SPDO_BASE1 contains the memory address of DMA buffer 1.
5:0	R	0x0	Reserved	Always reads as ‘0’.
Offset 0x10 90010			*SPDO_BASE2*	
31:6	R/W	0x0	BASE2	SPDO_BASE2 contains the memory address of DMA buffer 2.
5:0	R	0x0	Reserved	Always reads as ‘0’.
Offset 0x10 9014			*SPDO_SIZE*	
31:6	R/W	0x0	SIZE	SPDO_SIZE determines the size, in bytes, of both DMA buffers.
5:0	R	0x0	Reserved	Always reads as ‘0’.
Offset 0x10 9018—9FF0			*Reserved*	

SPDIF OUTPUT PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 9FF4			*SPDO_PWR_DWN*	
31:1		-	Unused	
0	R/W	0	PWR_DWN	Used to provide power control status for system software block power management.
Offset 0x10 9FFC			*SPDO_MODULE_ID*	
31:16	R	0x0121	ID	Module ID. This field identifies the block as type SPDO.
15:12	R	0	MAJ_REV	Major Revision ID
11:8	R	0	MIN_REV	Minor Revision ID
7:0	R	0	APERTURE	Aperture size. Identifies the MMIO aperture size in units of 4 kB for the SPDO block. SPDO has an MMIO aperture size of 4 kB. Aperture = 0:4 kB.

Chapter 24: MIPS RISC Processor Core

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

24.1 Introduction

The PR3940 MIPS RISC Processor Core is a high-performance 32-bit processor core. The Reduced Instruction Set Computer (RISC) core is based on the MIPS-II and MIPS-16 Instruction Set Architectures (ISAs), developed by MIPS Technologies, Inc. This chapter provides an overview of the PR3940 MIPS RISC Processor Core. For complete information, refer to the PR3940 MIPS RISC Core Data Book.

24.2 Functional Description

Figure 1 shows a block diagram of the PR3940, which has the following components:

- Instruction Execution Unit (IU)
- Coprocessor 0 (CP0)
- Multiply/Accumulate/Divide Unit (MAD)
- Memory Management Unit (MMU)
- PI-Bus Interface Unit (PI-BIU)
- Debug Support Unit (DSU)
- Instruction and Data Caches

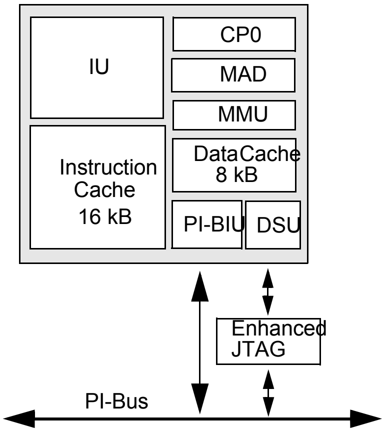


Figure 1: Block Diagram of the PR3940 Processor Core

The PR3940 can achieve an execution rate of one instruction per cycle through a six-stage pipeline. These stages, in the order of execution, are as follows:

- Instruction fetch (I-stage)
- Instruction pre-decode (P-stage)
- Instruction decode and register assignment (R-stage)
- Arithmetic/logic execution (A-stage)
- Memory access (M-stage)
- Writeback to registers (W-stage).

24.2.1 Instruction Execution Unit

The instruction execution unit contains the following blocks:

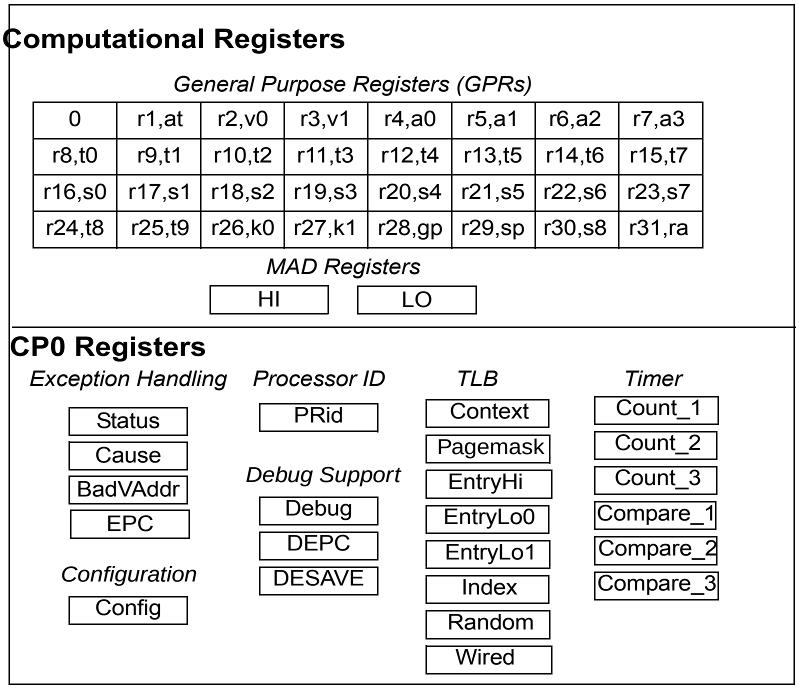
- Computational registers - 32 General Purpose Registers (GPRs) and HI/LO registers. All are 32 bits wide.
- ALU/Shifter - Computational unit for most of MIPS-II instructions

The instruction execution unit has the following 32-bit registers:

- 32 General Purpose Registers (GPRs) r0 to r31. These GPRs are treated symmetrically, with two exceptions:
  - Register r0 always contains the value 0. It can be a target register of an instruction whose result is not needed, or it can be a source register of an instruction of value 0.
  - Register r31 is the link return address register for the Jump And Link instruction.
- A pair of registers, labeled as HI and LO, for storing the results of MAD instructions

The registers HI and LO store the double-word (64-bit) operands and results of integer multiply and divide instructions. In the case of a multiply instruction, HI and LO store the high word and low word of the register, respectively. In the case of a divide instruction, the HI and LO are used to store the remainder and the quotient, respectively

.



Computational Registers

General Purpose Registers (GPRs)

0	r1,at	r2,v0	r3,v1	r4,a0	r5,a1	r6,a2	r7,a3
r8,t0	r9,t1	r10,t2	r11,t3	r12,t4	r13,t5	r14,t6	r15,t7
r16,s0	r17,s1	r18,s2	r19,s3	r20,s4	r21,s5	r22,s6	r23,s7
r24,t8	r25,t9	r26,k0	r27,k1	r28,gp	r29,sp	r30,s8	r31,ra

MAD Registers

HI LO

CP0 Registers

Figure 2: PR3940 Processor Registers

24.2.2 System Control Coprocessor (CP0)

The System Control Coprocessor (CP0) has a number of special purpose registers that are used during exception handling and to configure certain processor options. Refer to the PR3940 MIPS RISC Core Data Book for more information.

24.2.3 MAD Unit

The PR3940 incorporates a Multiply/Accumulate/Divide (MAD) unit with a 64-bit accumulator for fast multiply and multiply/accumulate computation. It can continuously execute one 16-bit signed or unsigned multiply/multiply-add instruction in one cycle, and 32-bit multiply/multiply-add instruction in four cycles. The MAD has two special registers, HI and LO, that store the results of the computation. Special instructions allow contents to be moved between the HI/LO registers and GPRs. MAD instruction also allows writeback capability into a GPR.

24.2.4 Memory Architecture

A full 32-bit address space is available in the PR3940 core. This gives access to a maximum of 4 GB of virtual memory space.

The PR3940 MMU logically expands the physical address space of the CPU by translating addresses composed in a large virtual address space into the physical address space. The memory address space is divided into four memory segments in which the virtual addresses are translated into physical addresses.

Static Translation

The translation of all the segments is guided by a pre-defined table built in the PR3940. This translation is enabled by setting the Translation Look-aside Buffer (TLB) bit in the CP0 Configuration register to 0. Furthermore, an application can choose one of the predefined translation tables: scheme 1 and scheme 2, which are selected according to the MAP bit in the configuration register.

Dynamic Translation

When the TLB bit in the Configuration register is enabled, the translation of some kernel segments is still based on a predefined table, but the rest use a TLB for page-based translation. This provides support for most high-end operating systems.

24.2.4.1 Memory Protection

Two operating modes are supported.

- Kernel mode provides the highest privilege. When in this mode, all registers, all memory space and the complete instruction set are accessible.
- User mode can access all computational registers, but has limited access to CP0 registers, memory, and instructions. User mode allows access to the lower 2 GB of memory.
- In addition, the TLB allows memory protection on a page-by-page basis.

24.2.4.2 Endianness

The PR3940 allows byte ordering of operands in either big or little-endian configuration. Software can specify the order of bytes within a word via the Configuration register.

24.2.5 Instruction and Data Caches

This section describes the cache and buffers along the memory path from the PR3940 through the PI-Bus to the memory. A cache is a high-speed memory store which contains instructions and operands in anticipation of their need by the processor. By retrieving data from the cache, longer duration of memory accesses are avoided and therefore the processor improves overall performance. In the PR3940, instructions and operands are stored separately in the instruction cache and data cache respectively.

The PR3940 cache organization is essentially software-transparent to maintain the binary compatibility with other standard MIPS II processors. There are a few operating modes to provide flexibility for various applications. For example, memory areas can be defined as cacheable or non-cacheable within the memory architecture and mapping scheme.

The PR3940 core has a separate set of associative instruction and data caches.

- Instruction Cache: The instruction cache is 16 kB, 2-way set associative, with 32 bytes per line. The refill size is fixed at one cache line.

- Data Cache: The data cache is 8 kB, 4-way set associative, with 16 bytes per line. The refill size is fixed at one line.

The data cache uses a write policy of writeback with allocate on write misses. That means on a store hit, data cache is only updated with the new operand value, but on a store miss, the missing line is fetched from the memory, merged with the new operand value and loaded into the cache. The data cache can be read and written internally and can be snooped by external devices.

24.2.5.1 Snooping

All memory accesses to the main memory from a processor or an I/O device are monitored by the bus controller. By supporting data snooping on the data cache, the PR3940 guarantees cache coherency.

24.2.6 Debug Support

The PR3940 core includes a Debug Support Unit (DSU) for instruction and data level break points, single stepping and real-time trace. The DSU provides access to the internal state of the PR3940 through dedicated pins. The DSU is in compliance with the MIPS Standard Enhanced JTAG interface.

For additional information, refer to Chapter 38 MIPS EJTAG Software Debug Port and to the PR3940 MIPS RISC Core Data Book.

The DSU provides the following features:

- Two additional instructions: Software Debug Breakpoints (SDBBP) and Debug Exception Return (DERET).
- Single step execution
- Hardware breakpoints can be set at:
  - Virtual instruction address (with address bitmasking)
  - Virtual data address (with address bitmasking) and data value (with byte lane masking)
  - Physical address (with lower address bitmasking) and physical data (with data bitmasking)
- Trace Trigger points can be specified instead of hardware breakpoints
- Debug breaks can be initiated by the Processor Probe via a JTAG pin
- PC Trace information is provided by additional status pins and the processor clock.

24.2.7 Instruction Set Overview

The PR3940 core implements a superset of the MIPS-II ISA that includes several extended computational, jump, system coprocessor instructions and branch-likely instructions.

Chapter 25: Internal TM32 CPU Core Processor

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

25.1 Introduction

The TriMedia32 CPU Core Processor (TM32 CPU core) is a media processor for a wide range of high-performance multimedia applications that deal with high quality video and audio.

25.2 Functional Description

The TM32 CPU core on the PNX8526 includes the following features:

- Full instruction set of the Philips TM1300 Media Processor
- Five-issue VLIW with 200 MHz instruction issue rate
- 32-bit integer and 32-bit floating point arithmetic
- Extensive set of SIMD 8 and 16-bit multimedia operations
- Integrated 16-kB, 8-way set-associative dual ported data cache
- integrated 32-kB, 8-way set-associative instruction cache
- Built-in clock generator to run at 1.0, 1.25, 1.333, 1.50, 1.666, 1.75 or 2.0x SDRAM clock
- Built-in multi-processor semaphore device
- Built-in 32 input vectored interrupt controller with programmable vectors
- Four built-in timers/counters
- Instruction and data breakpoint logic
- Cache performance analysis logic

For full documentation on all aspects of the TM32 CPU core, refer to the *TriMedia32 Architecture* Data Book, published by TriMedia Technologies, Inc.

25.3 Memory Map

The memory map seen by the TM32 CPU core contains three apertures as shown in Figure 1

Accesses outside the three apertures return a “0” on loads, and have no effect for stores. Each aperture is independently positioned. Apertures should not be set to overlap or extend across the 0xFFFF FFFF limit of 32-bit addressing conventions.

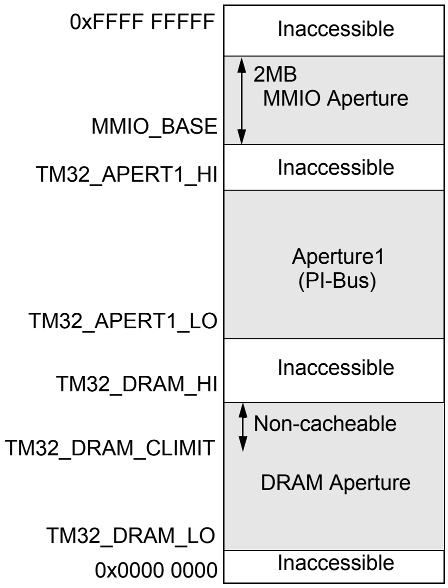


Figure 1: TM32 CPU Core Memory Map

25.3.1 DRAM Aperture

The range of the DRAM aperture varies. It must be a multiple of 64 kB, and can only be positioned on a 64-kB boundary. Its value is determined by the content of the TM32_DRAM_LO MMIO register inside the TM32 CPU core module. The last byte of the DRAM aperture is at address TM32_DRAM_HI-1.

The DRAM aperture is divided into two parts:

- Cacheable area from TM32_DRAM_LO to TM32_DRAM_CLIMIT-1
- Non-cacheable area from TM32_DRAM_CLIMIT to TM32_DRAM_HI-1

The TM32 CPU core can access the entire DRAM aperture data with all load and store instructions. Loads and stores in the cacheable area use the data cache. Loads and stores in the non-cacheable area bypass the data cache and go directly to memory across the memory interface. Execution is supported from the entire DRAM aperture, which always uses the instruction cache.

25.3.2 MMIO Aperture

The MMIO aperture is 2 MB in size. Its origin is determined by the value on the tm32_pci_mmio_base core input pins. The origin must be a multiple of 2 MB. Its value is only allowed to be changed when the TM32 CPU core is stopped.

In the PNX8526, the MMIO_BASE is provided by the PCI block. The MMIO_BASE is the same as the PCI BASE_14 value i.e., the origin of the second aperture in the PCI space.

The TM32 CPU core can only perform 32-bit aligned loads and stores to the MMIO aperture. Execution from this aperture, or access of sizes other than 32 bits, is not supported.

Figure 2 summarizes the layout of the MMIO aperture. Access to a 32 bit location in the MMIO aperture will access the Icache tags/(LRU) bits, a TM32 internal MMIO register, or a PI-Bus system device register.

MMIO locations with offset 0x00 FFFF constitute a 64-kB read-only Icache tag or Least Recently Used (LRU) area, which is only visible to the TM32 CPU core and not accessible by other PI-Bus masters.

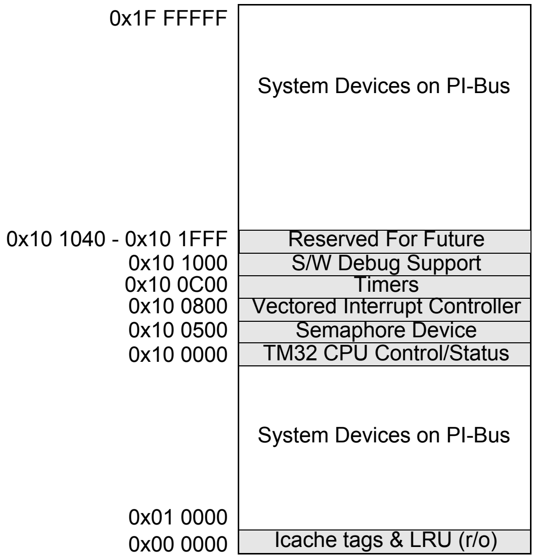


Figure 2: MMIO Locations Inside TM32 CPU Core

Accesses to MMIO locations with offsets 0x10 0000 - 0x10 1FFF resolve to core internal MMIO registers. The full set of TM32 internal registers in this range are defined in Section 25.11. A write to a non-existent register address has no effect. A read from a non-existent register returns zeroes and has no side effects.

Any 32-bit access to an MMIO location outside the Icache tags or core internal register area causes a PI-Bus access.

Remark: For security reasons, some of the TM32 internal MMIO registers are read-only and can only be written by PI masters (e.g., internal MIPS CPU, external PCI host, and Boot block). Writing to such registers from the TM32 CPU core has no effect. No exception is raised i.e., the value written is discarded.

25.3.3 Aperture1 (PI-Bus Aperture)

The range of Aperture1 varies and is intended to provide access to PI-Bus system resources outside the MMIO aperture. It must be a multiple of 64 kB, and can only be positioned on a 64-kB boundary. Its value is determined by the content of the TM32_APERT1_LO MMIO register inside the TM32 CPU core module. The last byte of the Aperture1 is at address TM32_APERT1_HI-1. If TM32_APERT1_HI is less than or equal to TM32_APERT1_LO, Aperture1 is disabled.

The TM32 CPU core can access Aperture1 data with all load and store instructions. Such loads and stores always bypass the data cache and go directly to the PI-Bus.

Remark: The TM32 CPU core always reads a full 32-bit word across the PI-Bus on 8 and 16-bit load operations, and internally selects the appropriate byte(s) based on address and endian mode.

TM32_APERT1_LOW and TM32_APERT1_HI are both 64-kB aligned. They are read-only for the TM32 and read/write for external PI masters. On reset, TM32_APERT1_LOW is set to 0x1C00 0000 and TM32_APERT1_HI is set to 0x2000 0000.

In the PNX8526, Aperture1 is not used in systems that obey the standard system memory map. It can be used in special configurations to map a part of the PCI or XIO peripheral address space into the TM32 CPU memory map.

25.4 Special Event Handling

The TM32 CPU core responds to the special events listed in Table 1 in priority order.

With the exception of RESET, which is enabled at all times, the TM32 architecture allows special event handling to begin only during an interruptible jump operation (ijmpt, ijmpf or ijmpi) that succeeds. EXC, NMI and INT handling can be initiated during handling of an EXC or an INT, but only during successful interruptible jumps.

Table 1: Special Events and Event Vectors

Event	Vector
RESET	(Highest priority) vector to TM32_START_ADR (taken after receiving start command).
EXC	(All exceptions) vector to EXCVEC (programmable)
NMI, INT	(Non-maskable interrupt, maskable interrupt) use the programmed vector (one of 32 vectors depending on the interrupt source).

The instruction scheduler uses interruptible jumps exclusively for interdecision tree jumps. Hence, within a decision tree, no special-event processing can be initiated. If a tree-to-tree jump is taken, special-event processing is allowed. Since the only registers live at this point (i.e., that contain useful data) are the global registers allocated by the ANSI C compiler, only a subset of the registers needs to be preserved by the event handlers. (Refer to the TriMedia SDE Reference Manual for register details.) The TM32 register state can be described by the contents of this subset of general purpose registers and the contents of the PCSW and the DPC value (the target of the inter-tree jump).

The priority resolution mechanism built into the TM32 hardware dispatches the highest priority, non-masked special-event request at the time of a successful interruptible jump operation. In view of the simple, real time oriented nature of the mechanisms provided, only limited nesting of events should be allowed.

25.4.1 Reset and Start

RESET is the highest priority special event. There are two ways to reset the TM32 CPU core. Both have identical effects.

- Hardware reset by asserting and then negating the tm32_reset_n input to the core. In the PNX8526, this is accomplished by “peri_rst_n” output from the reset module, as documented in Chapter 5 Clock Reset and Power Management.
- MMIO reset by writing the code for “stop and reset” to the TM32_CTL MMIO register

Upon either reset, the TM32 CPU puts all registers and cache control in its defined initial state. It does not start program execution.

After reset, program execution must be started as follows by either the PNX8526 boot block or a host processor:

- Write the desired frequency ratio and code for “enable pll” and “stop and reset” to the TM32_CTL register.
- Delay for approximately 10 μs to allow the internal CPU PLL to lock and stabilize.
- Configure DRAM and Aperture1 apertures.
- Write the desired TM32 CPU core start address to the TM32_START_ADDR MMIO register.
- Write the code for “start” and the *same* frequency ratio and “enable pll” to the TM32_CTL MMIO register.

Execution starts at the address contained in the TM32_START_ADDR MMIO register. TM32_START_ADDR is 64-byte aligned and is read-only for the TM32, and read/write for external PI masters.

25.4.2 EXC (Exceptions)

The TM32 CPU core enters EXC special-event processing under the following conditions:

- RESET is de-asserted.
- The intersection PCSW[15,6:0] and PCSW[31,22:16] is non-empty or PCSW.TFE is set.

- A successful interruptible jump is in the final jump execution stage.

The TM32 hardware takes the following actions on the initiation of EXC processing:

- DPC is assigned the intended destination address of the successful jump.
- Instruction processing starts at EXCVEC.

All other actions are the responsibility of the EXC handler software.

Remark: No other special event processing will take place until the handler decides to execute an interruptible jump that succeeds.

25.4.3 INT and NMI (Maskable and Non-Maskable Interrupts)

The on-chip Vectored Interrupt Controller (VIC) provides 32 INT request input hardware lines. The interrupt controller prioritizes and maps attention requests from several different peripherals onto successive INT requests to the TM32.

INT special event processing will occur under the following conditions:

- RESET is de-asserted.
- The intersection PCSW[15,6:0] and PCSW[31,22:16] is empty and PCSW.TFE is not set.
- The intersection of IPENDING and IMASK is non-empty.
- The interrupt is at level NMI or PCSW.IEN = 1.
- A successful interruptible jump is in the final jump execution stage.

The TM32 hardware takes the following actions on initiation of NMI or INT processing:

- DPC gets assigned the intended destination address of the successful jump.
- Instruction processing starts at the appropriate interrupt vector.

All other actions are the responsibility of the INT handler software.

Remark: No other special event processing will take place until the handler decides to execute an interruptible jump that succeeds.

25.5 Timers

The TM32 contains four programmable timers/counters, all with the same function. The first three (TIMER1, TIMER2, TIMER3) are intended for general use. The fourth timer/counter (SYSTIMER) is reserved for system software and should not be used by applications. Each timer has three registers as shown in Table . The MMIO register addresses shown are offset addresses with respect to the timer's base address.

Table 2: MMIO addresses

Timer Base	MMIO Address
TIMER1	MMIO_BASE+0x10 0C00
TIMER2	MMIO_BASE+0x10 0C20
TIMER3	MMIO_BASE+0x10 0C40
SYSTIMER	MMIO_BASE+0x10 0C60

Each timer/counter can be set to count one of the event types specified in Table 3.

Table 3: PNX8526 Timer Source Selections

Source Name	Source Bits Value	Source Description
CLOCK	0	TM32 CPU clock
PRESCALE	1	Prescaled TM32 CPU clock
TIMER1MUX	2	External expansion mux 1
DATABREAK	3	Data breakpoints
INSTBREAK	4	Instruction breakpoints
CACHE1	5	Cache event 1
CACHE2	6	Cache event 2
EXT0	7	DV1_CLK (DV1 input clock)
EXT1	8	DV_CLK1 (AICP1 output clock)
EXT2	9	Audio In 1 Word Strobe
EXT3	10	Audio Out 1 Word Strobe
TIMER2MUX	11	External expansion mux 2
TIMER3MUX	12	External expansion mux 3
TIMER4MUX	13	External expansion mux 4
EXT7	14	Audio In 2 Word Strobe
EXT8	15	Audio Out 2 Word Strobe

The DATABREAK event is special, in that the timer/counter may increment by zero, one or two in each clock cycle. For all other event types, increments are by zero or one.

Remark: The CACHE1 and CACHE2 events serve as cache performance monitoring support. The actual event selected for CACHE1 and CACHE2 is determined by the MEM_EVENTS MMIO register. If a TM32 pin signal is selected as an event, each rising edge of the signal is counted.

Each PNX8526 external expansion multiplexer allows for 16 additional timer/counter source selections. TIMER1MUX is only used for TIMER1, etc. To select a secondary source, set TIMER1 primary source to TIMER1MUX and set the controls for the expansion mux indicated in Table 4 to the value in Table 5.

Table 4: PNX8526 Expansion Mux Control MMIO Registers

Name	MMIO Offset
timer1mux_ctl	0x4D704
timer2mux_ctl	0x4D708
timer3mux_ctl	0x4D70c
timer4mux_ctl	0x4D710

Table 5: PNX8526 Timer Expansion Mux Selection

mux_ctl value	Source Description
0	DV2 input clock
1	DV3 input clock
2	AICP2 output clock
3	SPDIF In Word Strobe
4	Audio I/O 3 Word Strobe
5	TS_CLK, Transport Stream Out clock
6	1394 Master Clock
7	SSI input clock
8	GPIO1 output (selectable in GPIO)
9	GPIO2 output (selectable in GPIO)
10..15	RESERVED for future

25.6 Debug Support

This section describes the special debug support offered by the TM32 CPU core. Instruction and data breakpoints can be defined through a set of registers in the MMIO register space. When a breakpoint is matched, an event is generated that can be used as a timer source (see Section 25.5). The timer TMODULUS has to be set to generate a TM32 interrupt after the desired number of breakpoint matches.

For more information, see Chapter 39 TM32 JTAG Software Debug Port.

25.7 System Provisions

25.7.1 TM32_CTL

The TM32_CTL register (offset 0x10 0030) allows the boot block or a host processor to set the TM32 operating frequency and to start and (software) reset TM32 (see Section 25.4.1).

TM32_CTL is read-only for the TM32 CPU core, and read/write for external PI masters. On reset, TM32_CTL.PLL_EN is set to 0, while TM32_CTL.RATIO is set to 0x09 (1.0x). All other bits are undefined after reset. See Section 25.11 for details.

25.7.2 TM32_MODID

The read only TM32_MODID register (offset 0x10 0FFC) provides a unique ID to allow software to identify the TM32 CPU core and version number. See Section 25.11 for details.

25.7.3 Endian Mode

The TM32 CPU core supports both little-endian and big-endian modes of operation. Endian mode is solely determined by the PCSW BSX bit. Programs initiating switching of endianness (by writing a new PCSW value) are supported. Endianness only affects how data is loaded and stored. The TM32 CPU core instructions regarding endianness are coded in an invariant manner.

The 64-bit DVP memory bus uses address invariant rules: the 8 lsbits of the bus are associated with byte address “A” (A is divisable by 8). The 8 msbits of the bus are associated with address “A+7.” Byte enables are used to implement 8, 16 or 32-bit data transactions, but bytes travel over the lane associated with their memory byte address. This convention holds irrespective of system endianness.

The PI-Bus has a big-endian signal, that indicates to all attached blocks which endianness to use. The transfers on the bus always use right hand side (lsb) alignment (a 32-bit load/store value travels over all 32 wires). A 16-bit value travels over the 16 lsb wires (an 8-bit value travels over the 8 lsb wires). For each transfer size, the value travels with the msb on the left and the lsb on the right.

The TM32 CPU core has been designed to support dynamic endian-mode switching. However in normal use, software immediately sets it to operate in the same endian mode as the PI-Bus and does not change it thereafter.

25.8 Powerdown

25.8.1 Partial Powerdown

The TM32 CPU core can voluntarily powerdown into a low dissipation state. This is accomplished by any store to the POWER_DOWN MMIO register at offset 0x10 0108.

During partial powerdown, the CCCOUNT register and the TM32 CPU core timers (TIMER1, 2, 3 and SYSTIMER) continue operation.

Any enabled interrupt request, or any incoming PI-Bus access will wake up the TM32 CPU core and execution continues where it left off. The PI-Bus transaction value/ effect will not be affected by powerdown.

The partial powerdown mechanism is intended to be used by the idle loop or idle task of the real-time kernel. The TM32 CPU core should be powered down whenever possible to reduce heat dissipation to a minimum.

Table 6: Timing of Partial Powerdown

Event	Typical Time (TM32 CPU Clocks)
Powerdown store	6 clocks to halt activity
Interrupt assertion	9 clocks to wakeup
PI-Bus read or write	10 clocks to wakeup

25.8.2 Full Powerdown

Asserting the external input signal “tm32_pwrdwn_req” causes the TM32 CPU core to finish any outstanding bus transactions. It then shuts down all internal clocks and the internal PLL and asserts “tm32_pwrdwn_ack.”

In the PNX8526, powerdown can be initiated respectively through the TM32_PWRDWN_REQ and TM32_PWRDWN_ACK MMIO registers.

During full powerdown, the CCOUNT register and TM32 CPU core timers are not counting. In addition, the core will not grant access to its registers for incoming PI transactions. Instead, it returns a bogus value on read and acknowledges but ignores PI-Bus writes. PI-Bus transactions do not cause wakeup.

Wakeup is accomplished by de-asserting “tm32_pwrdwn_req,” which causes the TM32 CPU core to re-start its internal PLL, then re-start execution where it left off. Once started, the core de-asserts “tm32_pwrdwn_ack.” At that point, the core will again handle incoming PI-Bus transactions.

The time required from “powerdown request” to “powerdown complete” depends on the status of ongoing transactions at the time of the request. If no transactions are ongoing, powerdown is acknowledged in a few TM32 clocks. However, in the case of a complex transaction that needs to timeout, extensive simulation has shown that “powerdown acknowledge” occasionally takes more than 512 TM32 clock cycles, but never more than 1024 clocks.

Wakeup, from “de-assert request” to “de-assert acknowledge,” takes several hundred TM32 clocks, due to the need to re-start and stabilize the CPU PLL.

25.9 Interface Descriptions

The TM32 CPU core has two main interfaces: a 32-bit PI-Bus and a 64-bit DVP Memory bus.

25.9.1 PI-Bus Master

As bus master, the TM32 CPU core can initiate the following PI-Bus transactions:

- WDU read
- WDU write

- HW0, HW1 write
- BY0, BY1, BY2, BY3 write

The relationship between CPU load operations in the PI aperture and bus transactions is specified in Table 7. The WD16 read transactions are generated only by Icache misses due to TM32 CPU core PI aperture execution. For stores, refer to Table 8.

Table 7: PI Aperture Operation (CPU Loads)

TM32 CPU Operation	Endian Mode	Address	PI-Bus Opcode	PI Data	CPU Register
32 bit load	little	0x1000	WDU	01020304	01020304
16 bit load	little	0x1000	WDU	01020304	00000304
16 bit load	little	0x1002	WDU	01020304	00000102
8 bit load	little	0x1000	WDU	01020304	00000004
8 bit load	little	0x1001	WDU	01020304	00000003
8 bit load	little	0x1002	WDU	01020304	00000002
8 bit load	little	0x1003	WDU	01020304	00000001
32 bit load	big	0x1000	WDU	01020304	01020304
16 bit load	big	0x1000	WDU	01020304	00000102
16 bit load	big	0x1002	WDU	01020304	00000304
8 bit load	big	0x1000	WDU	01020304	00000001
8 bit load	big	0x1001	WDU	01020304	00000002
8 bit load	big	0x1002	WDU	01020304	00000003
8 bit load	big	0x1003	WDU	01020304	00000004

Table 8: PI Aperture Operation (CPU Stores)

TM32 CPU Operation	Endian Mode	Address	PI-Bus Opcode	CPU Register	PI Data
32 bit store	either	0x1000	WDU	01020304	01020304
16 bit store	either	0x1000	HW0	01020304	xxxx0304
16 bit store	either	0x1002	HW1	01020304	xxxx0102
8 bit store	either	0x1000	BY0	01020304	xxxxxx04
8 bit store	either	0x1001	BY1	01020304	xxxxxx03
8 bit store	either	0x1002	BY2	01020304	xxxxxx02
8 bit store	either	0x1003	BY3	01020304	xxxxxx01

Remark: The TM32 CPU core always retrieves a full 32-bit word across the PI-Bus on 8 and 16-bit load operations, and selects the appropriate byte(s) based on address and endian mode.

The core handles slave-initiated retract by re-issuing the request forever.

25.9.2 PI-Bus Slave

As a slave target, the TM32 CPU core supports only 32-bit WDU read and write PI-Bus transactions. Any other transactions lead to an error ACK.

The TM32 CPU core will frequently issue a slave retract on both WDU reads and writes. This is done to avoid locking down internal CPU buses for the relatively long duration of PI transactions.

25.9.3 PI-Bus Deadlock Issues

The TM32 CPU core as master will retry a slave retracted transaction forever.

For a TM32 PI-Bus aperture load/store, internal resources are not held during such a retry period, and the TM32 CPU core will allow incoming PI-Bus MMIO reads/writes while the retract/retry is going on.

For an MMIO aperture load/store, the TM32 CPU core does not allow incoming PI-Bus transactions during a retract/retry sequence. This limitation can in certain systems increase the risk of PI-Bus deadlock.

25.9.4 DVP Memory Bus

DVP transactions generated by the TM32 CPU core are:

- 64-byte read/write; 64-byte aligned, sequential order
- 64-byte read; 64-byte aligned, critical word first, XOR style

25.10 Global 1 and 2 Registers

The TM32 CPU core Timer (1-4) Input Select registers are used to select various signals that can be used as timer inputs. Global 1 registers select the signals via TIMER1MUX_CNTL to TIMER4MUX_CNTL. These registers are shown on page 25-513 of this chapter.

The TM32 CPU core Powerdown Request and Powerdown Acknowledge registers are used to set and check the powerdown condition of the TM32 CPU core. This control is done via the Global 2 registers TM_PWRDWN_REQ and TM_PWRDWN_ACK. These registers are shown on page 25-514 of this chapter.

25.11 Register Descriptions

The base address of the TM32 CPU Core module is 0x10 0000 for system registers.

25.11.1 TM32 CPU Core Address Map

The following table lists all the MMIO registers implemented inside the TM32 CPU core. For compatibility with future devices, any undefined MMIO bits should be ignored when read and written as zeroes.

Table 9: TM32 CPU Core Register Summary

Offset	Name	TM32 CPU	Other PI Masters	Description
Cache And Memory System				
0x10 000C	MEM_EVENTS	R/W	R/W	Selects two cache-related events for counting.
0x10 0010	DC_LOCK_CTL	R/W	R/W	Enable bit for data-cache locking, also PCI hole disable
0x10 0014	DC_LOCK_ADDR	R/W	R/W	Start of address range that will be locked into the data cache
0x10 0018	DC_LOCK_SIZE	R/W	R/W	Size of address range that will be locked into the data cache
0x10 001C	DC_PARAMS	R	R	Data-cache geometry (block size, association, # of sets)
0x10 0020	IC_PARAMS	R	R	Instruction-cache geometry (block size, association, # of sets)
TM32 CPU Core Registers				
0x10 0030	TM32_CTL	R	R/W	Controls (start, stop, frequency setting) the TM32 CPU core.
0x10 0034	TM32_DRAM_LO	R	R/W	Lowest DRAM address the TM32 CPU core can access
0x10 0038	TM32_DRAM_HI	R	R/W	Highest DRAM address the TM32 CPU core can access + 1
0x10 003C	TM32_DRAM_CLIMIT	R	R/W	Start of non-cacheable region in DRAM (very similar to old DRAM_CACHEABLE_LIMIT @10 0008)
0x10 0040	TM32_APERT1_LO	R	R/W	Lowest TM32 CPU core address that resolves to a PI-Bus access
0x10 0044	TM32_APERT1_HI	R	R/W	Highest TM32 CPU core address that resolves to a PI-Bus access + 1
0x10 0048	TM32_START_ADDR	R	R/W	Address that TM32 CPU core execution starts from upon START
0x10 004C	TM32_PC	R	R	TM32 CPU core Program Counter
Cache And Memory System				
0x10 0108	POWER_DOWN	R/W	W	Write to this register initiates the TM32 CPU core powerdown.
0x10 0210	IC_LOCK_CTL	R/W	R/W	Enable bit for instruction-cache locking
0x10 0214	IC_LOCK_ADDR	R/W	R/W	Start of address range that will be locked into instruction cache
0x10 0218	IC_LOCK_SIZE	R/W	R/W	Size of address range that will be locked into instruction cache
Miscellaneous				
0x10 0500	SEM	R/W	R/W	Semaphore assist device
TM32 CPU Core Registers				

Table 9: TM32 CPU Core Register Summary *…Continued*

Offset	Name	TM32 CPU	Other PI Masters	Description
0x10 0800	EXCVEC	R/W	R/W	Interrupt vector (handler start address) for exceptions
0x10 0810	ISETTING0	R/W	R/W	Interrupt mode and priority settings for sources 0-7
0x10 0814	ISETTING1	R/W	R/W	Interrupt mode and priority settings for sources 8-15
0x10 0818	ISETTING2	R/W	R/W	Interrupt mode and priority settings for sources 16-23
0x10 081C	ISETTING3	R/W	R/W	Interrupt mode and priority settings for sources 24-31
0x10 0820	IPENDING	R/W	R/W	Interrupt-pending status bit for all 32 sources
0x10 0824	ICLEAR	R/W	R/W	Interrupt-clear bit for all 32 sources
0x10 0828	IMASK	R/W	R/W	Interrupt-mask bit for all 32 sources
0x10 0880	INTVEC0	R/W	R/W	Interrupt vector (handler start address) for source 0
0x10 0884	INTVEC1	R/W	R/W	Interrupt vector (handler start address) for source 1
0x10 0888	INTVEC2	R/W	R/W	Interrupt vector (handler start address) for source 2
0x10 088C	INTVEC3	R/W	R/W	Interrupt vector (handler start address) for source 3
0x10 0890	INTVEC4	R/W	R/W	Interrupt vector (handler start address) for source 4
0x10 0894	INTVEC5	R/W	R/W	Interrupt vector (handler start address) for source 5
0x10 0898	INTVEC6	R/W	R/W	Interrupt vector (handler start address) for source 6
0x10 089C	INTVEC7	R/W	R/W	Interrupt vector (handler start address) for source 7
0x10 08A0	INTVEC8	R/W	R/W	Interrupt vector (handler start address) for source 8
0x10 08A4	INTVEC9	R/W	R/W	Interrupt vector (handler start address) for source 9
0x10 08A8	INTVEC10	R/W	R/W	Interrupt vector (handler start address) for source 10
0x10 08AC	INTVEC11	R/W	R/W	Interrupt vector (handler start address) for source 11
0x10 08B0	INTVEC12	R/W	R/W	Interrupt vector (handler start address) for source 12
0x10 08B4	INTVEC13	R/W	R/W	Interrupt vector (handler start address) for source 13
0x10 08B8	INTVEC14	R/W	R/W	Interrupt vector (handler start address) for source 14
0x10 08BC	INTVEC15	R/W	R/W	Interrupt vector (handler start address) for source 15
0x10 08C0	INTVEC16	R/W	R/W	Interrupt vector (handler start address) for source 16
0x10 08C4	INTVEC17	R/W	R/W	Interrupt vector (handler start address) for source 17
0x10 08C8	INTVEC18	R/W	R/W	Interrupt vector (handler start address) for source 18
0x10 08CC	INTVEC19	R/W	R/W	Interrupt vector (handler start address) for source 19
0x10 08D0	INTVEC20	R/W	R/W	Interrupt vector (handler start address) for source 20
0x10 08D4	INTVEC21	R/W	R/W	Interrupt vector (handler start address) for source 21
0x10 08D8	INTVEC22	R/W	R/W	Interrupt vector (handler start address) for source 22
0x10 08DC	INTVEC23	R/W	R/W	Interrupt vector (handler start address) for source 23
0x10 08E0	INTVEC24	R/W	R/W	Interrupt vector (handler start address) for source 24
0x10 08E4	INTVEC25	R/W	R/W	Interrupt vector (handler start address) for source 25
0x10 08E8	INTVEC26	R/W	R/W	Interrupt vector (handler start address) for source 26
0x10 08EC	INTVEC27	R/W	R/W	Interrupt vector (handler start address) for source 27
0x10 08F0	INTVEC28	R/W	R/W	Interrupt vector (handler start address) for source 28

Table 9: TM32 CPU Core Register Summary *…Continued*

Offset	Name	TM32 CPU	Other PI Masters	Description
0x10 08F4	INTVEC29	R/W	R/W	Interrupt vector (handler start address) for source 29
0x10 08F8	INTVEC30	R/W	R/W	Interrupt vector (handler start address) for source 30
0x10 08FC	INTVEC31	R/W	R/W	Interrupt vector (handler start address) for source 31
0x10 0C00	TIMER1_TMODULUS	R/W	R/W	Contains the maximum count value for timer 1 + 1.
0x10 0C04	TIMER1_TVALUE	R/W	R/W	Current value of timer 1 counter
0x10 0C08	TIMER1_TCTL	R/W	R/W	Timer 1 control (prescale value, source select, run bit)
0x10 0C20	TIMER2_TMODULUS	R/W	R/W	Contains the maximum count value for timer 2 + 1.
0x10 0C24	TIMER2_TVALUE	R/W	R/W	Current value of timer 2 counter
0x10 0C28	TIMER2_TCTL	R/W	R/W	Timer 2 control (prescale value, source select, run bit)
0x10 0C40	TIMER3_TMODULUS	R/W	R/W	Contains the maximum count value for timer 3 + 1.
0x10 0C44	TIMER3_TVALUE	R/W	R/W	Current value of timer 3 counter
0x10 0C48	TIMER3_TCTL	R/W	R/W	Timer 3 control (prescale value, source select, run bit)
0x10 0C60	SYSTIMER_TMODULUS	R/W	R/W	Contains the maximum count value for system timer + 1.
0x10 0C64	SYSTIMER_TVALUE	R/W	R/W	Current value of system timer/counter
0x10 0C68	SYSTIMER_TCTL	R/W	R/W	System timer control (prescale value, source select, run bit)
0x10 0FFC	TM32_MODID	R	R	Module ID register for software PI-Bus device enumeration
0x10 1000	BICTL	R/W	R/W	Instruction breakpoint control
0x10 1004	BINSTLOW	R/W	R/W	Start of address range that causes instruction breakpoints
0x10 1008	BINSTHIGH	R/W	R/W	End of address range that causes instruction breakpoints
0x10 1020	BDCTL	R/W	R/W	Data breakpoint control
0x10 1030	BDATAALOW	R/W	R/W	Start of address range that causes data breakpoints
0x10 1034	BDATAAHIGH	R/W	R/W	End of address range that causes data breakpoints
0x10 1038	BDATAVAL	R/W	R/W	Compare value for data breakpoints
0x10 103C	BDATAMASK	R/W	R/W	Compare mask for compare value for data breakpoints

Table 10: M32 CPU Core Global 1 and 2 Register Summary

Offset	Name	Description
Global 1 Registers		
0x06 3704	TIMER1MUX_CNTL	Timer 1 input select register
0x06 3708	TIMER2MUX_CNTL	Timer 2 input select register
0x06 370C	TIMER3MUX_CNTL	Timer 3 input select register
0x06 3710	TIMER4MUX_CNTL	Timer 4 input select register
Global 2 Registers		
0x04 D700	TM_PWRDWN_REQ	TM32 CPU core powerdown request
0x04 D704	TM_PWRDWN_ACK	TM32 CPU core powerdown acknowledge

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Cache and Memory System				
Offset 0x10 000C			***MEM_EVENTS***	
31:8		-	Unused	
7:4	R/W	0	Event2	Selects the source for TM32 CPU TIMER CACHE2 event: 0 = No event 1 = Instruction Cache Misses 2 = Instruction Cache stall cycles (includes cycles where I and Dcache both stall) 3 = Data Cache Bank Conflicts 4 = Data Cache Read Misses 5 = Data Cache Write Misses 6 = Data Cache stall cycles (excludes cycles where I and D cache both stall) 7 = Data Cache copy backs to SDRAM 8 = Copyback Buffer Full 9 = Data cache write miss with all fetch units occupied 10 = Data Cache stream miss 11 = Prefetch Operation Started and not discarded 12 = Prefetch Operation Discarded (already in Dcache, or no fetch unit) 13 = Prefetch Operation Discarded (already in Dcache) 14, 15 = RESERVED
3:0	R/W	0	Event1	Selects the source for TM32 CPU TIMER CACHE1 event.
Offset 0x10 0010			***DC_LOCK_CTL***	
31:1		-	Unused	
0	R/W	0	DC_LOCK_ENABLE	0 = Disable Data Cache Locking 1 = Enable locking for all bytes in the range DC_LOCK_ADDRESS <= a <= DC_LOCK_ADDRESS + DC_LOCK_SIZE -1, and bring this data in from main memory.
Offset 0x10 0014			***DC_LOCK_ADDR***	
31:14	R/W	0	DC_LOCK_ADDRESS[31:14]	Lower bound of memory byte address range that becomes data cache locked when DC_LOCK_ENABLE is asserted. The full range is DC_LOCK_ADDRESS <= a <= DC_LOCK_ADDRESS + DC_LOCK_SIZE -1.
13:0	R	0	DC_LOCK_ADDRESS[13:0]	These bits of DC_LOCK_ADDRESS must be zero due to alignment restrictions for the locked range.
Offset 0x10 0018			***DC-LOCK_SIZE***	
31:14		0	Unused	
13:6	R/W	0	DC_LOCK_SIZE[13:6]	Size of memory byte address range that becomes data cache locked when DC_LOCK_ENABLE is asserted. The full range is DC_LOCK_ADDRESS <= a <= DC_LOCK_ADDRESS + DC_LOCK_SIZE -1.
5:0	R	0	DC_LOCK_SIZE[5:0]	These bits must be 0 due to locked address range alignment restrictions
Offset 0x10 001C			***DC_PARAMS***	

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:26			Reserved	
25:16	R	64	BLOCKSIZE	Block Size of Data Cache, in bytes
15:11	R	8	ASSOCIATIVITY	Associativity of Data Cache
10:0	R	32	NUMBER_OF_SETS	Number of sets in Data Cache
Offset 0x10 0020			***IC_PARAMS***	
31:26			Reserved	
25:16	R	64	BLOCKSIZE	Block Size of Instruction Cache, in bytes
15:11	R	8	ASSOCIATIVITY	Associativity of Instruction Cache
10:0	R	32	NUMBER_OF_SETS	Number of sets in Instruction Cache

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
TM32 Control and Status Registers				
Offset 0x10 0030			***TM32_CTL***	
The TM32_CTL register is not writable by the TM32 CPU core.				
31:30	W	NI	ACTION	00 = No action 01 = Stop and reset TM32 CPU core immediately. 10 = Start TM32 CPU core (execution starts at TM32_START_ADR). 11 = Reserved
29:7		-	Unused	
6	R/W	0x1	PLL_EN	1 = PLL is generating the TM32 CPU core internal clock. 0 = PLL is disabled. Take TM32 CPU core internal clock directly from clk_tm32 core input, i.e. SDRAM_CLK in PNX8526.
5:0	R/W	0x9	RATIO	TM32 CPU core speed to clk_tm32 core input clock ratio 001001 = 1.0 (Note: In 1.0 RATIO, PLL_EN should be set to 0, i.e., the internal PLL should not be enabled.) 101100 = 1.25 100011 = 1.3333333 111101 = 1.4 011010 = 1.50 101011 = 1.6666666 111100 = 1.75 010001 = 2.0 Other = Reserved
This register allows the on-chip MIPS CPU, an external PCI host CPU, or any other PI-Bus master to STOP or START the TM32 CPU core and to set its operating frequency. Upon START, the TM32 CPU core starts execution from the address in the TM32_START_ADR MMIO register. Note that RATIO should only be changed while the TM32 CPU is stopped. After changing RATIO, a period of 1 µs delay should be taken before performing a TM32 CPU core start.				
Offset 0x10 0034			***TM32_DRAM_LO***	

TM32 CPU CORE REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
The TM32_DRAM_LO register is not writable by the TM32 CPU core.				
31:16	R/W	0x0000	TM32_DRAM_LO[31:16]	The TM32 CPU core goes out across the DVP memory bus for all loads/stores with TM32_DRAM_LO <= a < TM32_DRAM_HI
15:0	R	0x0000	TM32_DRAM_LO[15:0]	Must be zeroes due to 64-kB address alignment requirement
The TM32 CPU core is restricted to DRAM accesses from the 32-bit address TM32_DRAM_LO...TM32_DRAM_HI-1. During system boot and configuration, the values of TM32_DRAM_LO and TM32_DRAM_HI are set by the host CPU or the Boot block. The TM32 CPU core can read the values, but not change them. This prevents accidental or intentional overwriting of critical system data in DRAM, such as MIPS kernel data structures.				
Offset 0x10 0038			***TM32_DRAM_HI***	
The TM32_DRAM_HI register is not writable by the TM32 CPU core.				
31:16	R/W	0x0100	TM32_DRAM_HI[31:16]	This register is the highest DRAM address that TM32 CPU core can access. See register TM32_DRAM_LO for more details.
15:0	R	0x0000	TM32_DRAM_HI[15:0]	Must be zeroes due to 64-kB address alignment requirement
Offset 0x10 003C			***TM32_DRAM_CLIMIT***	
The TM32_DRAM_CLIMIT register is not writable by the TM32 CPU core. NOTE: this restriction may be removed in future implementations of the TM32 CPU core.				
31:16	R/W	0x00ff	TM32_DRAM_CLIMIT[31:16]	TM32 CPU.Dcache is allowed to cache DRAM accesses from TM32_DRAM_LO...TM32_DRAM_CLIMIT-1. Addresses from TM32_DRAM_CLIMIT...TM32_DRAM_HI-1 are non-cached.
15:0	R	0x0000	TM32_DRAM_CLIMIT[15:0]	Must be zeroes due to 64-kB address alignment requirement
Offset 0x10 0040			***TM32_APERT1_LO***	
The TM32_APERT1_LO register is not writable by the TM32 CPU core.				
31:16	R/W	0x1c00	TM32_APERT1_LO[31:16]	Lowest TM32 CPU core address that resolves to a PI access.
15:0	R	0x0000	TM32_APERT1_LO[15:0]	Must be zeroes due to 64-kB address alignment requirement
Any TM32 CPU core data access that falls between TM32_APERT1_LO...TM32_APERT1_HI-1 is resolved to a PI-Bus access. This aperture is intended to be mapped on the XIO bus to allow TriMedia to execute from Flash and access selected XIO peripherals. But it can be used for other purposes. There is no built-in interpretation of the aperture structure, size or position.				
Offset 0x10 0044			***TM32_APERT1_HI***	
The TM32_APERT1_HI register is not writable by the TM32 CPU core.				
31:16	R/W	0x2000	TM32_APERT1_HI[31:16]	TM32_APERT1_HI-1 is the highest TM32 CPU core address that resolves to a PI access. See register TM32_APERT1_LO for more details.
15:0	R	0x0000	TM32_APERT1_HI[15:0]	Must be zeroes due to 64-kB address alignment requirement
Offset 0x10 0048			***TM32_START_ADR***	
The TM32_START_ADR register is not writable by the TM32 CPU core.				
31:6	R/W	0x0	TM32_START_ADR[31:6]	PC start value for TM32 CPU core upon start of execution. Must be 64 byte aligned.
5:0	R	0x0	TM32_START_ADR[5:0]	Must be zeroes due to 64-byte alignment restriction

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
This MMIO register should be given a defined value before performing a TM32 CPU core ‘start’ by writing to TM32_CTL.ACTION. The value in TM32_START_ADR should lie inside TM32_DRAM_LO...TM32_DRAM_HI-1. This register should only be written to while the TM32 CPU core is in the stopped state.				
Offset 0x10 004C			*TM32_PC*	
31:0	R	NI	n/a	PC value of TM32 CPU core
This read-only register reflects the instantaneous value of the TM32 CPU core program counter. It can be read by any PI-Bus master, as well as by the TM32 CPU core itself. When read by the TM32 CPU core using 32-bit load operations, the value returned is precisely defined as the address of the VLIW instruction containing the load operation.				

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Partial Powerdown				
Offset 0x10 0108			*POWER_DOWN*	
This register should only be written by the TM32 CPU core, to perform a voluntary powerdown. The effect of externally initiated writes to this register is undefined!				
31:0	W			A TM32 CPU write to this register initiates a voluntary powerdown of the TM32 CPU. Execution continues where it left off upon any non-masked interrupt to the TM32 CPU core, or upon any incoming PI-Bus MMIO write to the TM32 CPU.
Instruction Cache Locking				
Offset 0x10 0210			*IC_LOCK_CTL*	
31:1		-	Unused	
0	R/W	0	IC_LOCK_ENABLE	0 = Instruction Cache Locking is disabled 1 = Enable locking for addresses in the range IC_LOCK_ADDR <= a <= IC_LOCK_ADDR + IC_LOCK_SIZE -1, and bring all instructions in this range into the Icache.
Offset 0x10 0214			*IC_LOCK_ADDR*	
31:14	R/W	0	IC_LOCK_ADDRESS[31:14]	Lower bound of memory byte address range that becomes instruction cache locked when IC_LOCK_ENABLE is asserted. The full range is IC_LOCK_ADDRESS <= a <= IC_LOCK_ADDRESS + IC_LOCK_SIZE -1.
13:0	R	0	IC_LOCK_ADDRESS[13:0]	Must be all zeroes due to alignment restrictions
Offset 0x10 0218			*IC_LOCK_SIZE*	
31:14		-	Unused	
13:6	R/W	0	IC_LOCK_SIZE[13:6]	Size of memory byte address range that becomes instruction cache locked when IC_LOCK_ENABLE is asserted. The full range is IC_LOCK_ADDRESS <= a <= IC_LOCK_ADDRESS + IC_LOCK_SIZE -1.
5:0	R	0	IC_LOCK_SIZE[5:0]	Must be all zeroes due to alignment restrictions

TM32 CPU CORE REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Misc. Registers				
Offset 0x10 0500			***SEM***	
31:12	R	0	Unused	
11:0	R/W	0	SEM	Multi-Processor Semaphore assist device. Writing a zero to this field makes the field zero. Writing a non-zero to this field succeeds if and only if the field was zero.

TM32 CPU CORE REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Exception and Interrupt Control and Vectors				
Offset 0x10 0800			***EXCVEC***	
31:0	R/W	0	Exception Vector	Address of exception handler routine
Offset 0x10 0810			***ISETTING0***	
Interrupt Mode and Priority for sources 0..7				
31:28	R/W	0	MP7	Each MP Field [xxxx]: 0xxx: source operates in edge-triggered mode. 1xxx: source operates in level-sensitive mode. Each MP Field [xxxx]: x111 NMI (highest) priority x110 maskable level 6 ... x000 maskable level 0
27:24	R/W	0	MP6	
23:20	R/W	0	MP5	
19:16	R/W	0	MP4	
15:12	R/W	0	MP3	
11:8	R/W	0	MP2	
7:4	R/W	0	MP1	
3:0	R/W	0	MP0	
Offset 0x10 0814			***ISETTING1***	
Interrupt Mode and Priority for sources 8..15				
31:28	R/W	0	MP15	Each MP Field [xxxx]: 0xxx: source operates in edge-triggered mode. 1xxx: source operates in level-sensitive mode. Each MP Field [xxxx]: x111 NMI (highest) priority x110 maskable level 6 ... x000 maskable level 0
27:24	R/W	0	MP14	
23:20	R/W	0	MP13	
19:16	R/W	0	MP12	
15:12	R/W	0	MP11	
11:8	R/W	0	MP10	
7:4	R/W	0	MP9	
3:0	R/W	0	MP8	

			TM32 CPU CORE REGISTERS	
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 0818			***ISETTING2***	
Interrupt Mode and Priority for sources 16..23				
31:28	R/W	0	MP23	Each MP Field [xxxx]: 0xxx: source operates in edge-triggered mode. 1xxx: source operates in level-sensitive mode. Each MP Field [xxxx]: x111 NMI (highest) priority x110 maskable level 6 ... x000 maskable level 0
27:24	R/W	0	MP22	
23:20	R/W	0	MP21	
19:16	R/W	0	MP20	
15:12	R/W	0	MP19	
11:8	R/W	0	MP18	
7:4	R/W	0	MP17	
3:0	R/W	0	MP16	
Offset 0x10 081C			***ISETTING3***	
Interrupt Mode and Priority for sources 24..31				
31:28	R/W	0	MP31	Each MP Field [xxxx]: 0xxx: source operates in edge-triggered mode. 1xxx: source operates in level-sensitive mode. Each MP Field [xxxx]: x111 NMI (highest) priority x110 maskable level 6 ... x000 maskable level 0
27:24	R/W	0	MP30	
23:20	R/W	0	MP29	
19:16	R/W	0	MP28	
15:12	R/W	0	MP27	
11:8	R/W	0	MP26	
7:4	R/W	0	MP25	
3:0	R/W	0	MP24	
Interrupt Controller Request, Clear and Mask MMIO				
Offset 0x10 0820			***IPENDING***	
31:0	R/W	0	IPENDING	Each IPENDING(i) bit: On read, 1=source i interrupt request is currently pending. On write, 1=assert source i interrupt request. Note: Functionality of this register may be affected by other TM32 registers.
Offset 0x10 0824			***ICLEAR***	
31:0	R/W	0	ICLEAR	Each ICLEAR(i) bit: On read, same as IPENDING(i) On write, 1=clear source i interrupt request Note: Functionality of this register may be affected by other TM32 registers.
Offset 0x10 0828			***IMASK***	
31:0	R/W	0	IMASK	Each IMASK(*i*) bit: On read or write, 0=disallow source i interrupt request. On read or write, 1=allow source i interrupt request.
Offset 0x10 0880			***INTVEC0***	
31:0	R/W	NI	Source 0 vector	Address of interrupt handler routine for source 0
Offset 0x10 0884			***INTVEC1***	

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:0	R/W	NI	Source 1 vector	Address of interrupt handler routine for source 1
Offset 0x10 0888			*INTVEC2*	
31:0	R/W	NI	Source 2 vector	Address of interrupt handler routine for source 2
⇓	⇓	⇓	⇓	⇓
Offset 0x10 08F8			*INTVEC30*	
31:0	R/W	NI	Source 30 vector	Address of interrupt handler routine for source 30
Offset 0x10 08FC			*INTVEC31*	
31:0	R/W	NI	Source 31 vector	Address of interrupt handler routine for source 31

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
CPU Timers/Counters				
Offset 0x10 0C00			*TIMER1_TMODULUS*	
31:0	R/W	NI	TMODULUS	Timer Modulus value. When the Timer reaches this value it wraps around to 0 (or to 1 in the case of increment by 2).
Offset 0x10 0C04			*TIMER1_TVALUE*	
31:0	R/W	NI	TVALUE	Timer Value
Offset 0x10 0C08			*TIMER1_TCTL*	
31:20		-	Unused	
19:16	R/W	0	PRESCALE	Prescale value is 2^PRESCALE, i.e., in the range [1..32768]
15:12		-	Unused	
11:8	R/W	0	SOURCE	Timer Source select: 0 = TM32 CPU clock 1 = pre-scaled CPU clock 2 = TIMERMUX1 external expansion mux 3 = data breakpoints 4 = instruction breakpoints 5 = cache event 1 6 = cache event 2 7 = DV1_CLK (DV1 input clock) 8 = DV_CLK1 (AICP1 output clock) 9 = Audio In 1 word strobe 10 = Audio Out 1 word strobe 11 = do not use 12 = do not use 13 = do not use 14 = Audio In 2 word strobe 15 = Audio Out 2 word strobe
7:1		-	Unused	

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	R/W	0	RUN	“RUN” bit: 0 = Timer1 stopped 1 = Timer1 running
Offset 0x10 0C20			***TIMER2_TMODULUS***	
31:0	R/W	NI	TMODULUS	Timer Modulus value. When the Timer reaches this value it wraps around to 0 (or to 1 in the case of increment by 2).
Offset 0x10 0C24			***TIMER2_TVALUE***	
31:0	R/W	NI	TVALUE	Timer Value
Offset 0x10 0C28			***TIMER2_TCTL***	
31:20		-	Unused	
19:16	R/W	0	PRESCALE	Prescale value is 2^PRESCALE, i.e., in the range [1..32768]
15:12		-	Unused	
11:8	R/W	0	SOURCE	Timer Source select: 0 = TM32 CPU clock 1 = pre-scaled CPU clock 2 = do not use 3 = data breakpoints 4 = instruction breakpoints 5 = cache event 1 6 = cache event 2 7 = DV1_CLK (DV1 input clock) 8 = DV_CLK1 (AICP1 output clock) 9 = Audio In 1 word strobe 10 = Audio Out 1 word strobe 11 = TIMER2MUX external expansion mux 12 = do not use 13 = do not use 14 = Audio In 2 word strobe 15 = Audio Out 2 word strobe
7:1		-	Unused	
0	R/W	0	RUN	“RUN” bit: 0 = Timer2 stopped 1 = Timer2 running
Offset 0x10 0C40			***TIMER3_TMODULUS***	
31:0	R/W	NI	TMODULUS	Timer Modulus value. When the Timer reaches this value it wraps around to 0 (or to 1 in the case of increment by 2).
Offset 0x10 0C44			***TIMER3_TVALUE***	
31:0	R/W	NI	TVALUE	Timer Value
Offset 0x10 0C48			***TIMER3_TCTL***	
31:20		-	Unused	
19:16	R/W	0	PRESCALE	Prescale value is 2^PRESCALE, i.e., in the range [1..32768]
15:12		-	Unused	

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
11:8	R/W	0	SOURCE	Timer Source select: 0 = TM32 CPU clock 1 = pre-scaled CPU clock 2 = do not use 3 = data breakpoints 4 = instruction breakpoints 5 = cache event 1 6 = cache event 2 7 = DV1_CLK (DV1 input clock) 8 = DV_CLK1 (AICP1 output clock) 9 = Audio In 1 word strobe 10 = Audio Out 1 word strobe 11 = do not use 12 = TIMER3MUX external expansion mux 13 = do not use 14 = Audio In 2 word strobe 15 = Audio Out 2 word strobe
7:1		-	Unused	
0	R/W	0	RUN	“RUN” bit: 0=Timer3 stopped 1=Timer3 running
Offset 0x10 0C60			*SYSTIMER_TMODULUS*	
31:0	R/W	NI	TMODULUS	Timer Modulus value. When the Timer reaches this value it wraps around to 0 (or to 1 in the case of increment by 2).
Offset 0x10 0C64			*SYSTIMER_TVALUE*	
31:0	R/W	NI	TVALUE	Timer Value
Offset 0x10 0C68			*SYSTIMER_TCTL*	
31:20		-	Unused	
19:16	R/W	0	PRESCALE	Prescale value is 2^PRESCALE, i.e., in the range [1..32768]
15:12		-	Unused	

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
11:8	R/W	0	SOURCE	Timer Source select: 0 = TM32 CPU clock 1 = pre-scaled CPU clock 2 = do not use 3 = data breakpoints 4 = instruction breakpoints 5 = cache event 1 6 = cache event 2 7 = DV1_CLK (DV1 input clock) 8 = DV_CLK1 (AICP1 output clock) 9 = Audio In 1 word strobe 10 = Audio Out 1 word strobe 11 = do not use 12 = do not use 13 = TIMER4MUX external expansion mux 14 = Audio In 2 word strobe 15 = Audio Out 2 word strobe
7:1		-	Unused	
0	R/W	0	RUN	“RUN” bit: 0=SysTimer stopped 1=SysTimer running

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Module ID				
Offset 0x10 0FFC			*TM32_MODID*	
31:16	R	0x2B80	Module	Module Code 0x2b80 designates TM32 CPU core
15:12	R	0	Major rev	Major design revision number of core
11:8	R	0	Minor rev	Minor design rev number (metal update)
7:0	R	0x01	Aperture size	MMIO Aperture size code = 8 kB

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Instruction Breakpoint Control and Address Range Registers				
Offset 0x10 1000			*BICTL*	
31:9		-	Unused	

TM32 CPU CORE REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
8	R/W	0	IAC	Instruction Address breakpoint Control 0 = Breakpoint if BINSTLOW <= addr <= BINSTHIGH 1 = Breakpoint if addr < BINSTLOW or addr > BINSTHIGH
7:1		-	Unused	
0	R/W	0	IC	“IC” Instruction breakpoint Control bit: 0 = Disable instruction breakpoints. 1 = Enable instruction breakpoints.
Offset 0x10 1004			***BINSTLOW***	
31:0	R/W	0	BINSTLOW	Instruction Breakpoint Range Low Address
Offset 0x10 1008			***BINSTHIGH***	
31:0	R/W	0	BINSTHIGH	Instruction Breakpoint Range High Address

TM32 CPU CORE REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Data Breakpoint Control Registers				
Offset 0x10 1020			***BDCTL***	
31:17		-	Unused	
16	R/W	0	DVC	Control field for data breakpoints: 0 = Break if data (under BDATAMASK) equal BDATAVAL 1 = Break if data (under BDATAMASK) not equal BDATAVAL
15:9		-	Unused	
8	R/W	0	DAC	“DAC” Data Address Control: 0 = Break if BDATALOW <= addr <= BDATAHIGH 1 = Break if addr < BDATALOW or addr >= BDATAHIGH
7:4		-	Unused	
3	R/W	0	BS	“BS” Break on Store: 0 = Don’t check data stores. 1 = Do check data stores.
2	R/W	0	BL	“BL” Break on Load: 0 = Don’t check data loads. 1 = Do check data loads.
1:0	R/W	0	DC	“DC” Data Control: 0 = Disable Data Breakpoints 1 = Enable break on data addresses only 2 = Enable break on data values only 3 = Enable break only when both address and value meet conditions
Data Breakpoint Address Range and Value Compare Registers				
Offset 0x10 1030			***BDATAALOW***	

TM32 CPU CORE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:0	R/W	0	BDATAALOW	Address Range Start
Offset 0x10 1034			*BDATAAHIGH*	
31:0	R/W	0	BDATAAHIGH	Address Range End
Offset 0x10 1038			*BDATAVAL*	
31:0	R/W	0	BDATAVAL	Data Breakpoint Value
Offset 0x10 103C			*BDATAMASK*	
31:0	R/W	0	BDATAMASK	Data Breakpoint Value Mask - only bits that are ‘1’ are taken into account when doing the data breakpoint value compare (for equal or non-equal). Use a nonzero value in this field when enabling data value breakpoints.

25.11.2 Global 1 Registers

GLOBAL 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Timer Multiplexer Control Registers				
Offset 0x06 3704			*TIMER1MUX_CNTL*	
31:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	0x0	TIMER1MUX_CNTL[3:0]	Timer1 input select. 0000 = DV2_CLK (DV2 input clock) 0001 = DV3_CLK (DV3 input clock) 0010 = DV_CLK2 (ICP2 pixel out clock) 0011 = SPDI (from SPDIF In pre-mux "spdi_tstamp") 0100 = I2S_IO_WS (from AIO3 Audio block) 0101 = TS_CLK (Transport Stream Out clock) 0110 = CLK_1394 (1394 master clock) 0111 = SSI_SCLK (external SSI input clock) 1000 = GPIO TIMER1 (source selectable in GPIO block) 1001 = GPIO TIMER2 (source selectable in GPIO block)
Offset 0x06 3708			*TIMER2MUX_CNTL*	
31:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	0x0	TIMER2MUX_CNTL[3:0]	Timer2 input select. 0000 = DV2_CLK (DV2 input clock) 0001 = DV3_CLK (DV3 input clock) 0010 = DV_CLK2 (ICP2 pixel out clock) 0011 = SPDI (from SPDIF In pre-mux "spdi_tstamp") 0100 = I2S_IO_WS (from AIO3 Audio block) 0101 = TS_CLK (Transport Stream Out clock) 0110 = CLK_1394 (1394 master clock) 0111 = SSI_SCLK (external SSI input clock) 1000 = GPIO TIMER1 (source selectable in GPIO block) 1001 = GPIO TIMER2 (source selectable in GPIO block)

GLOBAL 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x06 370C			*TIMER3MUX_CNTL*	
31:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	0x0	TIMER3MUX_CNTL[3:0]	Timer3 input select. 0000 = DV2_CLK (DV2 input clock) 0001 = DV3_CLK (DV3 input clock) 0010 = DV_CLK2 (ICP2 pixel out clock) 0011 = SPDI (from SPDIF In pre-mux "spdi_tstamp") 0100 = I2S_IO_WS (from AIO3 Audio block) 0101 = TS_CLK (Transport Stream Out clock) 0110 = CLK_1394 (1394 master clock) 0111 = SSI_SCLK (external SSI input clock) 1000 = GPIO TIMER1 (source selectable in GPIO block) 1001 = GPIO TIMER2 (source selectable in GPIO block)
Offset 0x06 3710			*TIMER4MUX_CNTL*	
31:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	0x0	TIMER4MUX_CNTL[3:0]	Timer4 input select. 0000 = DV2_CLK (DV2 input clock) 0001 = DV3_CLK (DV3 input clock) 0010 = DV_CLK2 (ICP2 pixel out clock) 0011 = SPDI (from SPDIF In pre-mux "spdi_tstamp") 0100 = I2S_IO_WS (from AIO3 Audio block) 0101 = TS_CLK (Transport Stream Out clock) 0110 = CLK_1394 (1394 master clock) 0111 = SSI_SCLK (external SSI input clock) 1000 = GPIO TIMER1 (source selectable in GPIO block) 1001 = GPIO TIMER2 (source selectable in GPIO block)

25.11.3 Global 2 Registers

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
TM32 Powerdown Registers				
Offset 0x04 D700			*TM32_PWRDWN_REQ*	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R/W	0x0	TM32_PWRDWN_REQ	TriMedia powerdown request 0 = Do not request a TriMedia powerdown 1 = Request a TriMedia powerdown
Offset 0x04 D704			*TM32_PWRDWN_ACK*	
31:1		-	Unused	Ignore during writes and read as zeroes.
0	R	0x0	TM32_PWRDWN_ACK	0 = Trimedia does not acknowledge powerdown request. 1 = Trimedia acknowledges powerdown request.

Chapter 26: Video Input Processor (VIP)

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

26.1 Functional Description

The PNX8526 has two Video Input Processors. The Video Input Processor (VIP) handles incoming digital video and processes it for use by other components of the PNX8526. This enables applications such as picture-in-picture and video teleconferencing on the TV screen.

The VIP provides the following functions:

- Receives ITU-R-656 digital video data from the video port. The data stream may come from a device such as the SAA 7111A, which can digitize analog video from any source.
- Provides an internal Test Pattern Generator with NTSC, PAL, and variable format support.
- Separate acquisition windows for video and VBI data
- Performs horizontal scaling, cropping and pixel packing on video data from a continuous video data stream or a single field or frame.
- Raw data capture in either 8 or 10-bit packed mode with double buffering
- ANC header decoding or window mode for VBI data extraction
- Routes the video stream and VBI data to the frame buffer in a variety of formats.
- Horizontal down scaling or zoom up to 2x, 8-bit input only
- Linear horizontal aspect ratio conversion using normal or transposed 6-tap polyphase filter
- Non-linear horizontal aspect ratio conversion using normal 6-tap polyphase filter
- 4:2:2 to 4:4:4 conversions
- Optional linear phase interpolation / non-linear phase interpolation (as in MBS)
- Provides last pixel in signals for VBI and video to GPIO block for timestamping.
- Interrupt generation for VBI or video written to memory
- Pixel frequency up to 40 MHz; 80 MHz input clock
- Color space conversion (mutually exclusive with scaling)

The functional blocks of the VIP and the main data paths are shown in Figure 1.

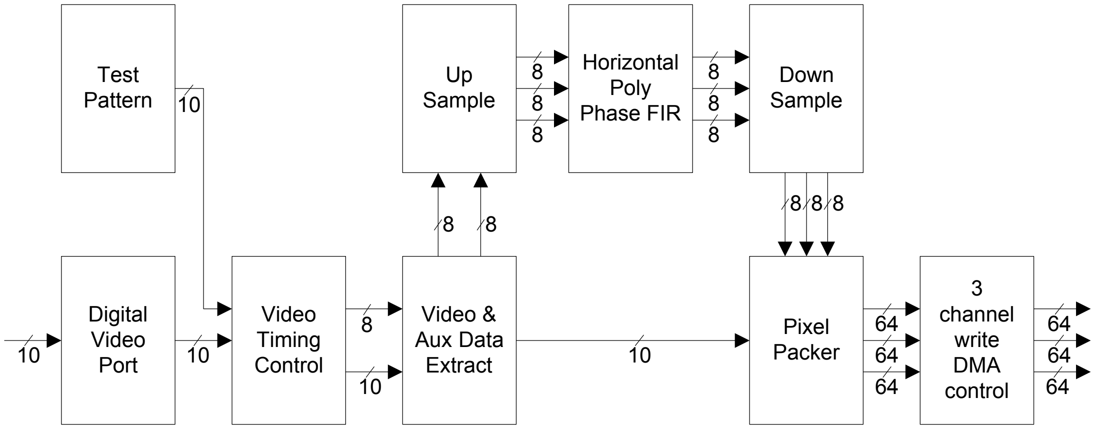


Figure 1: VIP Block Diagram

26.1.1 Operation

26.1.1.1 Video Input Source

The VIP accepts data from two sources:

- External Video Input Port
- Internal Test Pattern Generator

The input signal is selected by the Video Timing Control block.

The Video Mode Control block recovers sync information from the external sync signals or can decode sync signals that are encoded into the video data stream. The incoming encoded sync signal format is based upon D1 video timing reference codes, which also include protection bits (see ITU-R-656).

External Digital Video Input Port

The use of the video data port bits are shown in Table 1. In all modes, the msb is always placed on pin VDI 9.

Table 1: Digital Video Bus Signals

Bus Signal	8-bit	10-bit
VDI 9	D07	D09
VDI 8	D06	D08
VDI 7	D05	D07
VDI 6	D04	D06
VDI 5	D03	D05
VDI 4	D02	D04
VDI 3	D01	D03
VDI 2	D00	D02

Table 1: Digital Video Bus Signals *...Continued*

Bus Signal	8-bit	10-bit
VDI 1	VREF*	D01
VDI 0	HREF*	D00
* = Optional signal		

The external sync signals VREF and HREF are only available in 8-bit mode. A separate signal (DV_VALID) is provided to validate all incoming data.

Test Pattern Generator

The Test Pattern Generator produces a video stream with a pixel frequency of half the VIP input clock (e.g. the 27 MHz encoder clock from the clocks module). The sync generation is NTSC-like, with 525 lines per frame and 858 pixels per line. The active video range is 720x462 bordered by a white frame.

The test pattern is shown in Figure 2. It contains the following elements:

- A white 2-pixel wide frame—size 720x462
- A color bar—white 100%, yellow 75%, cyan 75%, green 75%, magenta 75%, red 75%, blue 75%, and black 0%
- A gray ramp—full value range 0–255
- A vertical multi-burst
- A horizontal multi-burst—first rectangle solid in odd, second solid in even field
- Vertical lines
- A moving cursor
- Test Pattern

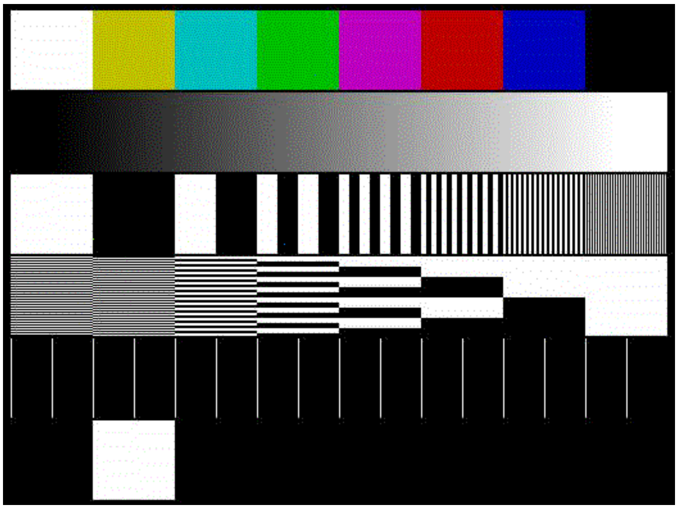


Figure 2: Test Pattern

To capture a picture using the built-in test pattern generator (odd and even field), set up the registers as shown in Table 2 to start capturing at the upper left corner of the white frame.

Table 2: Test Pattern Generator Setup

Mode	Reference	Window Start	Window End
		(x,y)	(x,y)
NTSC	HREF- / VREF+	8A,0 (138,0)	359,F1 (857,241)
PAL	HREF- / VREF+	90,0 (144,0)	35f,11F (863,287)

26.1.1.2 Video Input Formats

The VIP accepts the following external input streams via its Digital Video Port:

- 10-bit with encoded syncs YUV 4:2:2 (D1 mode)
- 8-bit with external syncs YUV 4:2:2 (VMI mode)
- 10-bit raw data samples (Raw mode)

The YUV 4:2:2 sampling scheme assumed by all modes is defined by CCIR 601.

D1 Mode

D1 Mode expects a 10-bit 4:2:2 video data stream (defined by CCIR 656) with syncs encoded in the video data stream. (For compatibility with 8-bit D1 interfaces, the two LSBs are not used for timing reference extraction, as defined in CCIR 656-2.) Timing reference codes recognized are 80h, 9Dh, ABh, B6h, C7h, DAh, ECh and F1h. Single bit errors in the reference codes are corrected, double bit errors are rejected.

VMI Mode

VMI Mode is 8-bit YUV (4:2:2) with external horizontal or vertical syncs. Chrominance and luminance samples are multiplexed into a single 8-bit data stream. The Field Identifier (FID) can be derived from the horizontal and vertical sync timing relation.

Raw Mode

In Raw Mode 8-bit or 10-bit data are continuously captured and written into system memory. In 10-bit raw mode each sample is packed into a 16-bit unit with either setting the MSBs to zero or by doing 16-bit sign extension. Raw Mode is only available in the auxiliary capture path of the VIP. It can be enabled independent of D1 or VMI mode.

26.1.1.3 Video Data Acquisition

The Video and Auxiliary Data Extract block receives a continuous pixel stream from the Video Timing Control block and outputs active window data and synchronization signals. Bit fields in the windowing registers specify the start and end of the source windows relative to the reference edges of H and V syncs and size of the target windows.

Internal Timing

Window start is defined relative to either the rising or falling edges of the VREF and HREF inputs (or similar D1 events). See Figure 3.

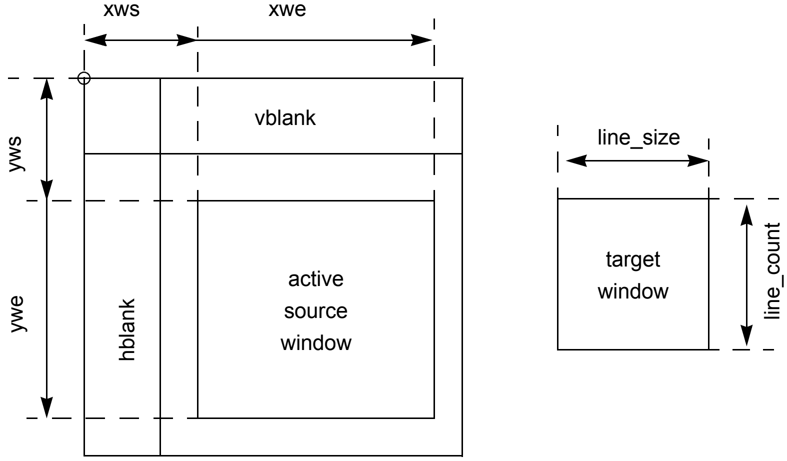


Figure 3: Source and Target Window Parameters

The first qualified data aligned with the REHS reference edge is interpreted as a U-sample. If the UYVY data stream is out of sync, it can be realigned with the *vsra* bits in the Video Input Format register.

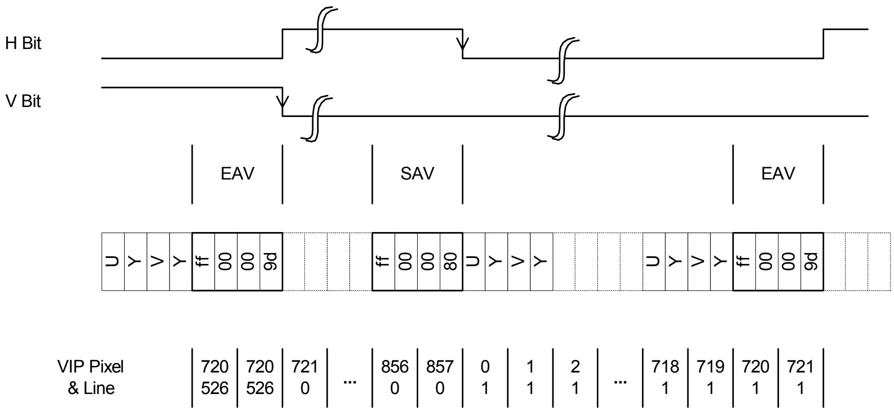


Figure 4: Acquisition Window Counter Reference

Refer to Table 2 and Table 3 for how to setup the windower and scaler e.g., to capture the entire test pattern.

Field Identifier Generation

The Field Identifier in D1 mode is extracted from the F bit in every valid video header. In VMI mode the Field Identifier is derived from the value of the HREF signal during the negative edge of the VREF signal. Instead of using the Field Identifier derived from the video stream the Field Identifier can also be forced to zero or forced to toggle after each new incoming field. The forced value takes effect after the selected vertical reference edge occurs at the input. The *SF* bit controls how the Field Identifier value is interpreted. A change of the Field Identifier interpretation takes effect immediately.

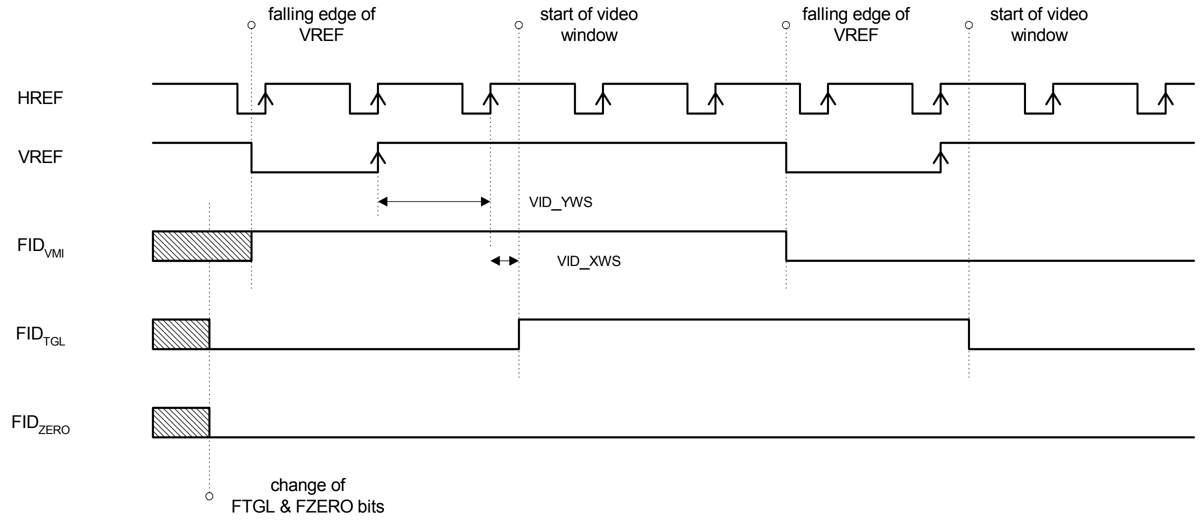


Figure 5: Field Identifier Timing

Table 3: Field Identifier Generation Modes

VDI8	HREF	VSEL	FZERO	FTGL	Change at	FID	FID	FSWP	Meaning
x	f	VMI	0	0	negedge VREF	!f	0	0	Odd
f	x	D1	0	0	valid D1 Header	f	1	0	Even
x	x	x	1	0	immediately	0	0	1	Even
x	x	x	x	1	immediately*	0,1,0,..	1	1	Odd

[3-1] *= FID toggles after detection of video window start

Video Acquisition Window

The start location of the window to be captured, relative to the input stream, is specified in the Video Acquisition Window Start registers (*VID_XWS, VID_YWS*).

The stop location of the window to be captured, relative to the input stream, is specified in the Video Acquisition Window End register (*VID_XWE, VID_YWE*).

Additional Target window cropping, which might be necessary after scaling, can be done with the PSU_*LSIZE* and PSU_*LCOUNT* values in the Target Window Size register.

26.1.1.4 Auxiliary Data Acquisition

Capturing auxiliary data utilizes the same DMA engine used for the third video plane. Capture of overlapping Video and Auxiliary regions is therefore only possible when semi-planar or packed formats are being used. Data can be captured in either 8 or 10 bits. In 10-bit mode data is extended to 16 bits by either adding leading zeroes or by sign-extension.

Three different types of auxiliary data capture are defined:

- Ancillary Data Capture (ANC)
- Auxiliary data acquisition window (AUX)
- Raw data capture (RAW)

A buffer size register can be used to limit data acquisition by size (one shot mode) or define a ring-buffer length.

Even though ANC and AUX capture can be enabled separately, simultaneous capture of ANC and AUX is not advisable. The timing and sequence of ANC and AUX data are not necessarily related and therefore, simultaneous capture may lead to unpredictable results.

Ancillary Data Capture

Ancillary data embedded in the stream and marked by special ITU-R-656 header codes can be decoded and extracted for software processing. ANC identifier codes (DID) recognized by the VIP are odd and even field VBI ANC codes (91 and 51 hex). The packed length is taken form the lower 6 bits of the NN byte (see Figure 6)

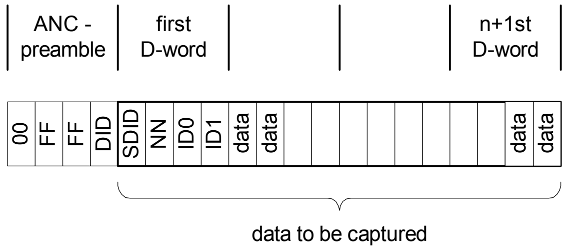


Figure 6: ANC Data Structure

A value of NN=0 will capture exactly one DWORD consisting of SDID, NN, ID0 and ID1 bytes. Parity bits for SDID and NN bytes located in bits 6 and 7 are not checked.

Auxiliary Data Acquisition Window

The auxiliary data acquisition window can be used to capture VBI data or an additional region of video data. The field identifier is compared against *AUX_CFEN* in the VIP Mode Control register bits at the start of the programmed window to control whether or not a field is captured. The start and end points of the auxiliary window are defined by the Auxiliary Acquisition Window Start and End registers at offsets 180 and 184 (*AUX_XWS, AUX_YWS, AUX_XWE, AUX_YWE*).

The *AUX_XWS* parameter specifies the number of the first pixel to be captured after the HREF reference edge.The *AUX_YWS* parameter specifies the number of the first line to be captured after the AUX reference edge.

The *AUX_XWE* parameter specifies the number of the last pixel to be captured. The *AUX_YWE* parameter specifies the number of the last line to be captured.

Pixel and lines start counting at 0.

Raw Data Capture

Raw data capture overrides ANC or AUX data capture modes when enabled. In raw capture mode any validated data at the video port is captured regardless of external or embedded synchronization signals.

26.1.1.5 Horizontal Video Filters

The horizontal video processing features in the VIP are the same as those in the MBS, with one exception: there is no fourth component data path in the VIP. However, the four component mode setting is needed to enable linear phase interpolation mode.

For more information, refer to Section 27.1.2.3 in Chapter 27 Memory Based Scaler (MBS).

26.1.1.6 Video Data Capture

The VIP can produce a variety of output formats. Video formats range from single component up to three component formats like a 4:4:4 YUV. Up to three write planes can be defined. On input, the video format is restricted to YUV 4:2:2 as defined in ITU-R-656 or 8/10 raw data. On output, true color and compressed formats are supported. For a complete list of supported video formats refer to Section 26.2 on page 26-525.

The Pixel Packing Unit takes care of quantization and packing of the color components into 64-bit units. A list of the most common video formats supported is shown in Table 4. Packing of a pixel into 64-bit units is always done from right to left, while bytes within one pixel unit are ordered according to the endianness settings (the global endianness setting and the endianness bit in the output format register).

Table 4: Output Pixel Formats

<table>
<tr><th>Format</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="3">planar YUV or RGB (4:4:4, 4:2:2 or 4:2:0)</td><td colspan="24" rowspan="3">plane #1
plane #2
plane #3</td><td colspan="8">Y8 or R8</td></tr>
<tr><td colspan="8">U8 or G8</td></tr>
<tr><td colspan="8">V8 or B8</td></tr>
<tr><td rowspan="2">semi planar YUV (4:2:2 or 4:2:0)</td><td colspan="16" rowspan="2">plane #1
plane #2</td><td colspan="8"></td><td colspan="8">Y8 or R8</td></tr>
<tr><td colspan="8">V8</td><td colspan="8">U8</td></tr>
<tr><td>packed 4/4/4 RGBa</td><td colspan="16"></td><td colspan="4">alpha</td><td colspan="4">R4</td><td colspan="4">G4</td><td colspan="4">B4</td></tr>
<tr><td>packed 4/5/3 RGBa</td><td colspan="16"></td><td colspan="4">alpha</td><td colspan="4">R4</td><td colspan="5">G5</td><td colspan="3">B3</td></tr>
<tr><td>packed 5/6/5 RGB</td><td colspan="16"></td><td colspan="5">R5</td><td colspan="6">G6</td><td colspan="5">B5</td></tr>
<tr><td>packed YUY2 4:2:2</td><td colspan="16"></td><td colspan="8">U8 or V8</td><td colspan="8">Y8</td></tr>
<tr><td>packed UYVY 4:2:2</td><td colspan="16"></td><td colspan="8">Y8</td><td colspan="8">U8 or V8</td></tr>
<tr><td>packed 888 RGB(a)</td><td colspan="8">(alpha)</td><td colspan="8">R8 or Y8</td><td colspan="8">G8 or U8</td><td colspan="8">B8 or V8</td></tr>
<tr><td>packed 4:4:4 VYU(a)</td><td colspan="8">(alpha)</td><td colspan="8">V8</td><td colspan="8">Y8</td><td colspan="8">U8</td></tr>
</table>

Table 4 shows the location of the first “pixel unit” within a 64-bit word in little-endian mode. The selected endianness will affect the position of components within multi-byte pixel units. Refer to Chapter 7 Pixel Formats for more information.

Capture Enable and Mode

Video capture can be limited to odd or even fields or both fields using the VID_CFEN bits in the VIP Mode Control register. If both fields are captured, capturing starts with the next odd field.

The status of the VID_OSM (one-shot) bit in the VIP Mode Control register specifies the capture mode (one-shot or continuous):

- If VID_OSM = 0, the corresponding incoming video stream is being captured continuously. For example, in a video conference application the vanity image would be a continuous stream to the frame buffer.
- If VID_OSM = 1, the corresponding incoming video stream is being captured one field or frame at a time (depending on the VID_CFEN bits).
  For example, in a video conference application the captured image would be a one-shot stream to the host memory. If you write VID_OSM = 1 and select field/frame in the register, it is captured on the next VSYNC and VID_CFEN bits are cleared to 0. To capture the next image, the VID_CFEN and VID_OSM bits must be reprogrammed.

Address Generation

The line address is generated by loading the base address from the corresponding register set at the beginning of each field and adding the line pitch to it at the beginning of each new line. The lower three bits of the first three base address registers are used as a byte offset for the first pixel components of each line. The offset has to be a multiple of the number of bytes per component.

Double Buffer Mode

To avoid line tear caused by displaying a frame being updated at the same time, a double buffer mode is available. In double buffer mode, a second set of DMA base addresses is available. After capturing one complete frame into the location described by the one set, the other set is used for the next frame.

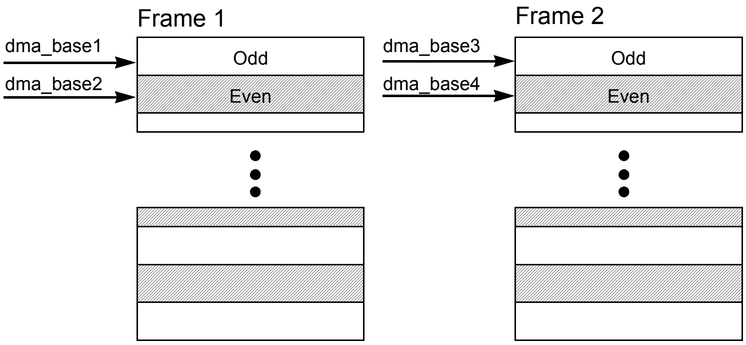


Figure 7: Double Buffer Mode

26.1.1.7 AUX Data Capture

Auxiliary data capture formats for writing into the frame buffer are limited to raw luma and chroma samples in 8-bit or 10-bit format (extended to 16-bit). Optionally, writing of chroma samples can be omitted.

Capture Enable and Mode

Bits in the VIP Mode Control register perform the following functions as described below:

- The AUX_CFEN bits specify the fields from which the device is to capture AUX data. If AUX_CFEN=0, no auxiliary data is captured. Once capture of an auxiliary window has started, resetting these bits has no effect until the end of the video window.
- The AUX_ANC bits specify the type of ancillary data blocks to be fetched. Once capture of an ANC block has started, resetting these bits has no effect until the end of the data block.
- The AUX_RAW bit enables continuous capturing of raw samples regardless of external or internal syncs. If raw capture is enabled AUX_CFEN and AUX_ANC bit settings are ignored. Changes to this bit take effect immediately.
- The AUX_BSIZE value specifies the buffer size available for AUX data in number of bytes.
- The AUX_PITCH bits specify the AUX line pitch i.e., the difference in the address from a pixel on a line to the same pixel on the next line when pitch mode is enabled. Pitch mode is also defined for ANC data capture where each packet is treated as a new video line.

- The AUX_OSM bit can be used to automatically limit capture by stopping after any wrap condition is reached. End of AUX window wrap condition also applies to ANC capture, even if AUX capture is disabled.

Data buffered in local FIFOs is flushed when a wrap condition is reached or in pitch mode at the end of each video line or ANC packet. For raw mode, data is also flushed when disabling raw capture.

Address Generation

The Auxiliary Capture Base Address register provides 26 bits to specify a destination address for storing the auxiliary data in the frame buffer. Address generation is similar to video capture address generation except for the missing double buffer and field modes.

26.1.1.8 Interrupt Generation

The VIP contains a DVP-compliant interrupt generation mechanism. Interrupts can be generated for the following events:

- Start of video
- End of video (written to memory)
- Start of AUX In
- End of AUX In (written to memory)
- Line threshold reached
- Pipeline error. The error may be due to an illegal scaling ratio e.g., >2x scaling or memory bus bandwidth error (FIFO overflow).

In addition to these interrupts the VIP provides “last pixel in” signals from VBI and video to the GPIO block for timestamping.

26.2 Register Descriptions

The PNX8526 has two Video Input Processors, VIP 1 and VIP 2. The registers for VIP 1 begin at 0x10 6000. The VIP 2 registers begin at 0x10 7000.

26.2.1 Register Address Map

…Continued

Table 5: VIP Module Register Summary

Offset	Name	Description
Video Input Processor (VIP) 1		
0x10 6000	VIP_MODE	VIP operation mode
0x10 6040	VIP_LINETHR	Video line count threshold
0x10 6100	VIN_FORMAT	Video input format and mode
0x10 6104	VIN_TESTPGEN	Video test pattern generator
0x10 6140	WIN_XYSTART	Video horizontal and vertical acquisition window start

Table 5: VIP Module Register Summary

Offset	Name	Description
0x10 6144	WIN_XYEND	Video horizontal and vertical acquisition window end
0x10 6180	AUX_XYSTART	Auxiliary horizontal and vertical acquisition window start
0x10 6184	AUX_XYEND	Auxiliary horizontal and vertical acquisition window stop
0x10 6200	HSP_ZOOM_0	Initial zoom for 1st pixel in line (unsigned)
0x10 6204	HSP_PHASE	Horizontal phase control
0x10 6208	HSP_DZOOM_0	Initial zoom delta for 1 pixel in line (signed)
0x10 620C	HSP_DDZOOM	Zoom delta change (signed)
0x10 6220	CSM_COEFF0	Color space matrix coefficients C_{00} - C_{02}
0x10 6224	CSM_COEFF1	Color space matrix coefficients C_{10} - C_{12}
0x10 6228	CSM_COEFF2	Color space matrix coefficients C_{20} - C_{22}
0x10 622C	CSM_OFFS1	Color space matrix offset coefficients D_0-D_2
0x10 6230	CSM_OFFS2	Color space matrix rounding coefficients E_0-E_2
0x10 6284	CSM_CKEY	Color key components
0x10 6300	PSU_FORMAT	Output format and mode
0x10 6304	PSU_WINDOW	Target window size
0x10 6340	PSU_BASE1	Target base address DMA #1
0x10 6344	PSU_PITCH1	Target line pitch component 1
0x10 6348	PSU_BASE2	Target base address DMA #2
0x10 634C	PSU_PITCH2	Target line pitch component 2 and 3
0x10 6350	PSU_BASE3	Target base address DMA #3
0x10 6354	PSU_BASE4	Target base address DMA #4
0x10 6358	PSU_BASE5	Target base address DMA #5
0x10 635C	PSU_BASE6	Target base address DMA #6
0x10 6380	AUX_FORMAT	Auxiliary capture output format and mode
0x10 6390	AUX_BASE	Auxiliary capture base address
0x10 6394	AUX_PITCH	Auxiliary capture line pitch
0x10 6800—69FC	COEFF_TABLE	Coefficient table for horizontal filter
0x10 6FE0	INT_STATUS	Interrupt status register
0x10 6FE4	INT_ENABLE	Interrupt enable register
0x10 6FE8	INT_CLEAR	Interrupt clear register
0x10 6FEC	INT_SET	Interrupt set register
0x10 6FF4	POWERDOWN	Powerdown mode
0x10 6FFC	MODULE_ID	Module Identification and revision information
Video Input Processor (VIP) 2		
0x10 7000	VIP_MODE	VIP operation mode
0x10 7040	VIP_LINETHR	Video line count threshold
...	...	Refer to VIP 1 Registers above.
0x10 7FFC	MODULE_ID	Module Identification and revision information

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
VIP 1 Registers (Offset 0x10 6000) and VIP 2 Registers (Offset 0x10 7000) The VIP registers 1 and 2 are identical except for their offsets.				
Operating Mode Control Registers				
Offset Offset 0x10 6000			*VIP Mode Control*	
31:30	R/W	0	VID_CFEN[1:0]	Video window capture field enable 00 = Capture disabled. 01 = Capture odd only. 10 = Capture even only. 11 = Capture both.
29	R/W	0	VID_OSM	Video capture one shot mode 0 = Continuously capture fields selected by CFEN. 1 = Capture fields selected by CFEN only once.
28	R/W	0	VID_FSEQ	Video capture field sequence 0 = Capture fields starting with any field. 1 = Capture fields starting with odd field. Setting has no effect unless VID_CFEN is set to capture both.
27:26	R/W	0	AUX_CFEN[1:0]	Auxiliary window capture enable 00 = Capture disabled. 01 = Capture odd only 10 = Capture even only 11 = Capture both
25	R/W	0	AUX_OSM	Auxiliary capture one shot mode 0 = When auxiliary wrap event is reached, buffer wraps around. 1 = When auxiliary wrap event is reached, capturing stops.
24	R/W	0	AUX_FSEQ	Auxiliary capture field sequence 0 = Capture fields starting with any field. 1 = Capture fields starting with odd field. Setting has no effect unless AUX_CFEN is set to capture both.
23:22	R/W	0	AUX_ANC[1:0]	ANC data capture enable 00 = No ANC data captured 01 = Capture ANC blocks with DID=55 (odd field VBI data) 10 = Capture ANC blocks with DID=91 (even field VBI data) 11 = Capture ANC blocks with DID=91 and 55
21	R/W	0	AUX_RAW	Auxiliary raw capture enable 0 = Raw capture disabled. 1 = Raw capture enabled, all samples will be captured. When enabled, AUX_ANC and AUX_CFEN settings are ignored.
20:18		-	Unused	

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS

Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17	R/W	0	RST_ON_ERR	Reset on error. Writing a one into this bit will automatically reset the block in case of a pipeline error (e.g. illegal scaling ratio / FIFO overflow).
16	W	0	SOFT_RESET	Soft reset. Writing a one into this bit will reset the block.
15		-	Unused	
14	R/W	0	IFF_CLAMP	Clamp mode for IFF (affects U/V only) 0 = Clamp to 0-255. 1 = Clamp to 16 - 240 (CCIR range).
13:12	R/W	0	IFF_MODE	Interpolation mode 00 = Bypass 01 = Reserved 10 = Co-sited 11 = Interspersed
11		-	Unused	
10	R/W	0	DFF_CLAMP	Clamp mode for DFF (affects U/V only) 0 = Clamp to 0-255. 1 = Clamp to 16 - 240 (CCIR range).
9:8	R/W	0	DFF_MODE	Decimation mode 00 = Bypass 01 = Co-sited (subsample) 10 = Co-sited (lowpass) 11 = Interspersed
7:4		-	Unused	
3	R/W	0	HSP_CLAMP	Clamp mode for HSP 0 = Clamp to 0-255. 1 = Clamp to CCIR range defined by bit 2.
2	R/W	0	HSP_RGB	Color space mode, defines CCIR clamping range for HSP 0 = Processing in YUV color space 1 = Processing in RGB color space
1:0	R/W	0	HSP_MODE	Horizontal processing mode 00 = Bypass mode 01 = Color space matrix mode 10 = Normal polyphase mode 11 = Transposed polyphase mode
Video Information Registers				
Offset 0x10 6040			***VIP Line Threshold***	
31:11		-	Unused	
10:0	R/W	0	LCTHR[10:0]	Video line count threshold Line threshold status bit is set if video line count (SVLC) reaches this value.

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Input Format Control Registers				
Offset 0x10 6100			***Video Input Format***	
31:30	R/W	0	VSRA[1:0]	Video stream realignment 00 = Normal 01 = Ignore 1st sample after HREF. 1x = Reserved
29:21		-	Unused	
20	R/W	0	NHDAUX	Header detect during AUX window 0 = D1 header detection enabled inside AUX window. 1 = D1 header detection disabled inside AUX window.
19	R/W	0	NPAR	Parity check disable 0 = Parity check enabled for D1 header detection. 1 = Parity check disabled for D1 header detection.
18:16		-	Unused	
15:14	R/W	0	VSEL[1:0]	Video source select 00 = Reserved 01 = Video port, encoded sync (D1-Mode) 10 = Video port, external sync (VMI-Mode) 11 = Reserved
13	R/W	0	TWOS	UV data type 0 = Offset binary 1 = Two’s complement
12	R/W	0	TPG	Test pattern generator 0 = Video stream selected by VSEL. 1 = Internal test pattern generator
11:10		-	Unused	
10	R/W	0	FREF	Field toggle reference mode 0 = Normal, use VREF. 1 = Toggling Field bit is used as vertical reference.
9	R/W	0	FTGL	Field toggle mode 0 = Normal 1 = Free toggle (sequence starts with FID = 0).
8:4		-	Unused	
3	R/W	0	SF	Swap field interpretation 0 = Odd (first) field = 0, even (second) field = 1 1 = Odd (first) field = 1, even (second) field = 0
2	R/W	0	FZERO	Force FID value to zero 0 = Field identifier derived from input stream. 1 = Force field identifier value to 0

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
1	R/W	0	REVS	Vertical sync reference edge 0 = Falling edge / start of active video 1 = Rising edge / end of active video
0	R/W	0	REHS	Horizontal sync reference edge 0 = Falling edge / SAV 1 = Rising edge / EAV
Offset 0x10 6104			***Video Test Pattern Generator Control***	
31	R/W	0	PAL	Field generation mode 0 = NTSC timing 1 = PAL timing
30		-	Unused	
29	R/W	0	VSEL	Vertical timing signal select 0 = Generate VREF. 1 = Generate VS.
28	R/W	0	HSEL	Horizontal timing signal select 0 = Generate HREF. 1 = Generate HS.
27	R/W	0	SWAP	Alternative test pattern 0 = Normal test pattern 1 = Test pattern with diagonal patterns, etc.
26	R/W	0	MOVE	Scrolling enable for alternative test pattern 0 = No scrolling 1 = Scrolling enabled.
25:0		-	Unused	
Video Acquisition Window Control Registers				
Offset 0x10 6140			***Video Acquisition Window Start***	
31:27		-	Unused	
26:16	R/W	0	VID_XWS[10:0]	Horizontal video window start The pixel co-sited with the reference edge REHS is numbered 0.
15:11		-	Unused	
10:0	R/W	0	VID_YWS[10:0]	Vertical video window start The first line indicated by the reference edge REVS is numbered 0.
Offset 0x10 6144			***Video Acquisition Window End***	
31:27		-	Unused	
26:16	R/W	0	VID_XWE[10:0]	Horizontal video window end pixels from XWS up to and including XWE are processed.
15:11		-	Unused	
10:0	R/W	0	VID_YWE[10:0]	Vertical video window end lines from YWS up to and including YWE are processed.

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
VBI Acquisition Window Control Registers				
Offset 0x10 6180			*Auxiliary Acquisition Window Start*	
31:27		-	Unused	
26:16	R/W	0	AUX_XWS[10:0]	Horizontal auxiliary window start . The pixel co-sited with the reference edge REHS is numbered 0.
15:11		-	Unused	
10:0	R/W	0	AUX_YWS[10:0]	Vertical auxiliary window start. The line co-sited with the reference edge REVS is numbered 0.
Offset 0x10 6184			*Auxiliary Acquisition Window End*	
31:27		-	Unused	
26:16	R/W	0	AUX_XWE[10:0]	Horizontal auxiliary window end pixels from XWS up to and including XWE are processed.
15:11		-	Unused	
10:0	R/W	0	AUX_YWE[10:0]	Vertical auxiliary window end lines from YWS up to and including YWE are processed.

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Horizontal Video Processing Control Registers				
Offset 0x10 6200			*Initial Zoom*	
31:29	R/W	0	HSP_PHASE_MODE[2:0]	Phase mode 0 = 64 phases 1 = 32 phases 2 = 16 phases 3 = 8 phases 4 = 4 phases 5 = 2 phases 6 = Fixed phase 7 = Linear phase interpolation (only valid for 4 component mode)
28:27		-	Unused	
26	R/W		HSP_FIR_COMP[1:0]	Horizontal filter components 0 = three components, 6-tap FIR each 1 = four components, 3-tap FIR each (4th component unused). In color space matrix mode this value has to remain zero.
25:20		-	Unused	

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19:0	R/W	0	HSP_ZOOM_0[19:0]	Initial zoom for 1st pixel in line (unsigned, LSB = 2^{-16}) 2 0000 (hex) = downscale 50% 1 0000 (hex) = no scaling = 2^0 0 8000 (hex) = zoom 2 x (transposed: downscale 50%). Values above 10000 (hex) are not valid in transposed mode.
Offset 0x10 6204			*Phase Control*	
31		-	Unused	
30:28	R/W	0	HSP_QSHIFT[2:0]	Quantization shift control used to change quantization before being multiplied with HSP_MULTIPLY. 100 (bin) = divide by 16. 101 (bin) = divide by 8. 110 (bin) = divide by 4. 111 (bin) = divide by 2. 000 (bin) = multiply by 1. 001 (bin) = multiply by 2. 010 (bin) = multiply by 4. 011 (bin) = multiply by 8. Warning: A value range overflow caused by an improper quantization shift can not be compensated for later by multiplying a HSP_MULTIPLY value below 0.5.
27:26		-	Unused	
25	R/W	0	HSP_QSIGN	Quantization sign bit
24:16	R/W	0	HSP_QMULTIPLY[8:0]	Quantization multiply control used to compensate for different weight sums in transposed polyphase or color space matrix mode, remaining bits are fraction (largest number is 511/512). Value range: $0 \leq m < 1.0$. Instead of using values in the range of $m < 0.5$ the quantization shift HSP_QSHIFT should be modified to gain more precision in the truncated result.
15:13		-	Unused	
12:0	R/W	0	HSP_OFFSET_0	Initial start offset for DTO
Offset 0x10 6208			*Initial Zoom delta*	
31:26		-	Unused	
25:0	R/W	0	HSP_DZOOM_0[25:0]	Initial zoom delta for 1 pixel in line (signed, LSB = 2^{-27}) used for non-constant scaling ratios.
Offset 0x10 620C			*Zoom delta change*	
31:29		-	Unused	
28:0	R/W	0	HSP_DDZOOM[28:0]	Zoom delta change (signed, LSB = 2^{-40}) used for non-constant scaling ratios.

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Color Space Matrix Registers				
Offset 0x10 6220			*Color space matrix coefficients C_{00} - C_{02}*	
31:30		-	Unused	
29:20	R/W	0	CSM_C02[9:0]	Coefficient C02 (signed, LSB = 2^{-9})
19:10	R/W	0	CSM_C01[9:0]	Coefficient C01 (signed, LSB = 2^{-9})
9:0	R/W	0	CSM_C00[9:0]	Coefficient C00 (signed, LSB = 2^{-9})
Note: Signed values are represented as two's complement.				
Offset 0x10 6224			*Color space matrix coefficients C_{10} - C_{12}*	
31:30		-	Unused	
29:20	R/W	0	CSM_C12[9:0]	Coefficient C12 (signed, LSB = 2^{-9})
19:10	R/W	0	CSM_C11[9:0]	Coefficient C11 (signed, LSB = 2^{-9})
9:0	R/W	0	CSM_C10[9:0]	Coefficient C10 (signed, LSB = 2^{-9})
Offset 0x10 6228			*Color space matrix coefficients C_{20} - C_{22}*	
31:30		-	Unused	
29:20	R/W	0	CSM_C22[9:0]	Coefficient C22 (signed, LSB = 2^{-9})
19:10	R/W	0	CSM_C21[9:0]	Coefficient C21 (signed, LSB = 2^{-9})
9:0	R/W	0	CSM_C20[9:0]	Coefficient C20 (signed, LSB = 2^{-9})
Offset 0x10 622C			*Color space matrix offset coefficients D_0 - D_2*	
31:29		-	Unused	
28	R/W	0	CSM_D2_TWOS	Offset coefficient D_2 type 0 = Unsigned 1 = Signed
27:20	R/W	0	CSM_D2[7:0]	Offset coefficient D_2 (LSB = 2^0)
19		-	Unused	
18	R/W	0	CSM_D1_TWOS	Offset coefficient D_1 type 0 = Unsigned 1 = Signed
17:10	R/W	0	CSM_D1[7:0]	Offset coefficient D_1 (LSB = 2^0)
9		-	Unused	
8	R/W	0	CSM_D0_TWOS	Offset coefficient D_0 type 0 = Unsigned 1 = Signed
7:0	R/W	0	CSM_D0[7:0]	Offset coefficient D_0 (LSB = 2^0)
Offset 0x10 6230			*Color space matrix offset coefficients E_0 - E_2*	
31:30		-	Unused	
29:20	R/W	0	CSM_E2[9:0]	Offset coefficient E2 (signed, LSB = 2^{-2})
19:10	R/W	0	CSM_E1[9:0]	Offset coefficient E1 (signed, LSB = 2^{-2})

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
9:0	R/W	0	CSM_E0[9:0]	Offset coefficient E0 (signed, LSB = 2^{-2})
Color Keying Control Registers				
Offset 0x10 6284			*Color Key Components*	
31:24	R/W	0	CKEY_ALPHA	Alpha value. Defines the alpha value to be used for keyed samples.
23:0		-	Unused	

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Video Output Format Control Registers				
Offset 0x10 6300			*Video Output Format*	
31:30	R/W	0	PSU_BAMODE	Base address mode 00 = Single set (e.g., progressive video source) base 1-3 according to number of planes (plane 1-3) 01 = Reserved 10 = Alternate sets each field (interlaced video source). base 1-3, odd field (plane 1-3) base 4-6, even field (plane 1-3) 11 = Alternate sets each field and frame (double buffer mode). packed modes only, frame index is set to 1 if cfen=0, frame index is incremented after capturing even field before capturing odd, base address byte offset is defined in PSU_OFFSET1 base 1, odd field 1st frame (plane 1 only) base 2, even field 1st frame (plane 1 only) base 3, odd field 2nd frame (plane 1 only) base 4, even field 2nd frame (plane 1 only)
29:14		-	Unused	
13	R/W	0	PSU_ENDIAN	Output format endianness 0 = Same as system endianness 1 = Opposite of system endianness
12		-	Unused	
11:10	R/W	0	PSU_DITHER	Output format dither mode 00 = No dithering 01 = Error dispersion (never reset pattern). 10 = Error dispersion (reset pattern at first capture enable). 11 = Error dispersion (reset pattern every field).

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
9:8	R/W	0	PSU_ALPHA	Output format alpha mode 00 = No alpha (alpha byte not written). 01 = Alpha byte written, value from CKEY_ALPHA (offset 284). 10 = Reserved 11 = Reserved Setting 00 is ignored if size of alpha component is less than 8 bits.
7:0	R/W	0	PSU_OPFMT	Output formats 08 (hex) = YUV 4:2:2, semi-planar 0B (hex) = YUV 4:2:2, planar 0F (hex) = RGB or YUV 4:4:4, planar A9 (hex) = Compressed 4/4/4 + (4 bit alpha) AA (hex) = Compressed 4/5/3 + (4 bit alpha) AD (hex) = Compressed 5/6/5 A0 (hex) = Packed YUY2 4:2:2 A1 (hex) = Packed UYVY 4:2:2 E2 (hex) = YUV or RGB 4:4:4 + (8 bit alpha) E3 (hex) = VYU 4:4:4 + (8 bit alpha)
Offset 0x10 6304			***Target Window Size***	
31:27		-	Unused	
26:16	R/W	0	PSU_LSIZE	Line size. Used for horizontal cropping after scaling. 0 = Cropping disabled. 1 = One pixel
15:11		-	Unused	
10:0	R/W	0	PSU_LCOUNT	Line count. Used for vertical cropping after scaling. 0 = Cropping disabled. 1 = One line

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Video Output Address Generation Control Registers				
Offset 0x10 6340			***Target Base Address #1***	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE1	Base address DMA #1 used depending on PSU_BAMODE setting.
2:0	R/W	0	PSU_OFFSET1	Base address byte offset plane 1 bits define pixel offset within multi pixel 64 bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 6344			***Target Line Pitch #1***	
31:15		-	Unused	

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
14:3	R/W	0	PSU_PITCH1	Line pitch DMA #1 . Signed value (two's complement) used for all packed formats and for plane 1.
2:0		-	Unused	
Offset 0x10 6348			*Target Base Address #2*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE2	Base address DMA #2 used depending on PSU_BAMODE setting.
2:0	R/W	0	PSU_OFFSET2	Base address byte offset plane 2 bits define pixel offset within multi pixel 64 bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 634C			*Target Line Pitch #2*	
31:15		-	Unused	
14:3	R/W	0	PSU_PITCH2	Line pitch DMA #2, signed value (two's complement) used for planes 2 and 3.
2:0		-	Unused	
Offset 0x10 6350			*Target Base Address #3*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE3	Base address DMA #3 used depending on PSU_BAMODE setting.
2:0	R/W	0	PSU_OFFSET3	Base address byte offset plane 3 bits define pixel offset within multi pixel 64 bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 6354			*Target Base Address #4*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE4	Base address DMA #4 used depending on PSU_BAMODE setting.
2:0		-	Unused	
Offset 0x10 6358			*Target Base Address #5*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE5	Base address DMA #5 used depending on PSU_BAMODE setting.
2:0		-	Unused	
Offset 0x10 635C			*Target Base Address #6*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE6	Base address DMA #6 used depending on PSU_BAMODE setting.
2:0		-	Unused	

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Auxiliary Data Output Format Control Registers				
Offset 0x10 6380			*Auxiliary Capture Output Format*	
31:30		0	AUX_BAMODE	Base address mode 00 = Pitch mode, wrap at end of buffer or window. 01 = Pitch mode, wrap at end of buffer. 10 = Append mode, wrap at end of buffer or window. 11 = Reserved
29:27		-	Unused	
26	R/W	0	AUX_SGNEX	Auxiliary capture sign extension 0 = No sign extension 1 = Sign extension enabled for 10-bit samples.
25	R/W	0	AUX_BPS	Auxiliary capture bits per sample 0 = 8-bit samples 1 = 10-bit samples
24	R/W	0	AUX_SUBSAMPLE	Auxiliary capture subsample 0 = All samples 1 = Luma (even) samples only
23:22		-	Unused	
21:0	R/W	0	AUX_BZSIZE[21:0]	Auxiliary capture ring buffer size Size of ring buffer in bytes; 0 = unlimited buffer size.
Auxiliary Data Output Address Generation Control Registers				
Offset 0x10 6390			*Auxiliary Capture Base Address*	
31:26		-	Unused	
25:0	R/W	0	AUX_BASE	Auxiliary capture base address. Lower 3 bits define byte offset within 64-bit words. Offset has to be a multiple of the byte per unit size (e.g., a 16-bit unit can be placed on any 16-bit boundary)
Offset 0x10 6394			*Auxiliary Capture Line Pitch*	
31:15		-	Unused	
14:3	R/W	0	AUX_PITCH	Auxiliary capture line pitch. Signed value
2:0		-	Unused	

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Miscellaneous Registers				
Offset 0x10 6800 - 69FC			*Coefficient Table Taps 0-5 (Horizontal) (64 entries x 64 bits)*	
63:62		-	Unused	
61:52	W	NI	TAP_5[X][9:0]	Inverted coefficient, tap #5, two's complement

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
51:42	W	NI	TAP_4[X][9:0]	Inverted coefficient, tap #4, two's complement
41:32	W	NI	TAP_3[X][9:0]	Inverted coefficient, tap #3, two's complement
31:30		-	Unused	
29:20	W	NI	TAP_2[X][9:0]	Inverted coefficient, tap #2, two's complement
19:10	W	NI	TAP_1[X][9:0]	Inverted coefficient, tap #1, two's complement
9:0	W	NI	TAP_0[X][9:0]	Inverted coefficient, tap #0, two's complement

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Interrupt and Status Control Registers				
Offset 0x10 6FE0			***Interrupt Status***	
31	R	0	STAT_FID_AUX	Field identifier at start of auxiliary window
30	R	0	STAT_FID_VID	Field identifier at start of video window
29	R	0	STAT_FID_VPI	Field identifier at video input port
28		-	Unused	
27:16	R	0	STAT_LINE_COUNT[11:0]	Source video line count. Refer to Section 26.1.1.3 on page 26-518 for information on how lines are counted.
15:10		-	Unused	
9	R	0	STAT_AUX_OVRFLW	Auxiliary buffer overflow event
8	R	0	STAT_VID_OVRFLW	Video buffer overflow event
7	R	0	STAT_WIN_SEQBRK	Windower sequence break event
6	R	0	STAT_FID_SEQBRK	Field identifier sequence break event
5	R	0	STAT_LINE_THRESH	Line counter threshold reached event
4	R	0	STAT_AUX_WRAP	Auxiliary capture write pointer wrap around event
3	R	0	STAT_AUX_START_IN	Start of auxiliary data acquisition event
2	R	0	STAT_AUX_END_OUT	End of auxiliary data write to memory event
1	R	0	STAT_VID_START_IN	Start of video data acquisition event
0	R	0	STAT_VID_END_OUT	End of video data write to memory event
Offset 0x10 6FE4			***Interrupt Enable***	
31:10		-	Unused	
9	R/W	0	IEN_AUX_OVRFLW	Auxiliary buffer overflow event
8	R/W	0	IEN_VID_OVRFLW	Video buffer overflow event
7	R/W	0	IEN_WIN_SEQBRK	Windower sequence break event
6	R/W	0	IEN_FID_SEQBRK	Field identifier sequence break event
5	R/W	0	IEN_LINE_THRESH	Line counter threshold reached event

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R/W	0	IEN_AUX_WRAP	Auxiliary capture write pointer wrap around event
3	R/W	0	IEN_AUX_START_IN	Start of auxiliary data acquisition event
2	R/W	0	IEN_AUX_END_OUT	End of auxiliary data write to memory event
1	R/W	0	IEN_VID_START_IN	Start of video data acquisition event
0	R/W	0	IEN_VID_END_OUT	End of video data write to memory event
Offset 0x10 6FE8			*Interrupt Clear*	
31:10		-	Unused	
9	W	0	CLR_AUX_OVRFLW	Auxiliary buffer overflow event
8	W	0	CLR_VID_OVRFLW	Video buffer overflow event
7	W	0	CLR_WIN_SEQBRK	Windower sequence break event
6	W	0	CLR_FID_SEQBRK	Field identifier sequence break event
5	W	0	CLR_LINE_THRESH	Line counter threshold reached event
4	W	0	CLR_AUX_WRAP	Auxiliary capture write pointer wrap around event
3	W	0	CLR_AUX_START_IN	Start of auxiliary data acquisition event
2	W	0	CLR_AUX_END_OUT	End of auxiliary data write to memory event
1	W	0	CLR_VID_START_IN	Start of video data acquisition event
0	W	0	CLR_VID_END_OUT	End of video data write to memory event
Offset 0x10 6FEC			*Interrupt Set*	
31:10		-	Unused	
9	W	0	SET_AUX_OVRFLW	Auxiliary buffer overflow event
8	W	0	SET_VID_OVRFLW	Video buffer overflow event
7	W	0	SET_WIN_SEQBRK	Windower sequence break event
6	W	0	SET_FID_SEQBRK	Field Identifier sequence break event
5	W	0	SET_LINE_THRESH	Line counter threshold reached event
4	W	0	SET_AUX_WRAP	Auxiliary capture write pointer wrap around event
3	W	0	SET_AUX_START_IN	Start of auxiliary data acquisition event
2	W	0	SET_AUX_END_OUT	End of auxiliary data write to memory event
1	W	0	SET_VID_START_IN	Start of video data acquisition event
0	W	0	SET_VID_END_OUT	End of video data write to memory event
Offset 0x10 6FF4			*POWERDOWN*	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.

VIDEO INPUT PROCESSOR (VIP) 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 6FFC			*Module ID*	
31:16	R	0x011A	MOD_ID	Module ID (unique 16-bit code)
15:12	R	0	REV_MAJOR	Major revision counter
11:8	R	1	REV_MINOR	Minor revision counter
7:0	R	0	APP_SIZE	Aperture Size is 0 = 4 kB.

Video Input Processor (VIP) 2 Registers				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
VIP 2 registers begin at offset 0x10 7000. In all other respects, they are identical to the VIP 1 registers.				

Chapter 27: Memory Based Scaler (MBS)

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

27.1 Functional Description

This document describes the Memory Based Scaler (MBS), an advanced video scaling module.

27.1.1 Overview

Memory based scaling is done independently of any video clocks by reading the video data from memory and writing the processed pictures back to memory. An advantage of memory based scaling is that a single scaler can be used to scale more than one video stream.

The memory based scaler (MBS) in the PNX8526 performs the following operations:

- Scaling
- De-interlacing
- Linear and non-linear aspect ratio conversion
- Anti-flicker filtering
- Conversions between 4:2:0, 4:2:2 and 4:4:4
- Indexed color to true color conversion
- Color expansion / compression
- De-planarization / planarization
- Variable color space conversion

Most of the above functions can be performed during a single pass, though the filter quality (length) may vary depending on the operations performed. Table 1 shows some typical combinations with resulting filter lengths

.

Table 1: Pipeline Processing (Horizontal First Mode)

Input	IFF[a]	Horizontal	DFF[b]	Vertical	Output
- 4:2:0 input (semi) planar - 4:2:2 input	up-sample to 4:4:4	- color space conversion (full matrix plus offset) or - scaling only (6-tap direct or transposed polyphase)	down-sample to 4:2:x	- scaling only (6-tap direct polyphase) - de-interlacing [c] (see Table 3)	- 4:2:0 output (semi) planar - 4:2:2 output (1-3 planes)
- 4:4:4 input (8-32 BPP) - LUT mode (8/4/2/1 BPP)[d]	bypass		bypass	- scaling only (4-tap direct polyphase) - Anti Flicker (static 4-tap filter)	- 4:4:4 output (16-32 BPP) - 4:4:4:4 output (16-32 BPP)
- 4:4:4:4 input (16-32 BPP) - LUT mode (8/4/2/1 BPP)[e]	bypass	- scaling only (3-tap direct polyphase)	bypass	- scaling only (3-tap direct polyphase) - Anti Flicker (static 3-tap filter)	

[a] Interpolation FIR Filter

[b] Decimation FIR Filter

[c] Vertical Temporal Median (VTM) or two field Majority Select Algorithms (MSA) only

[d] alpha component from LUT is not used

[e] alpha component from LUT is used

Table 2: Pipeline Processing (Vertical First Mode)

Input	Vertical	IFF	Horizontal	DFF	Output
- 4:2:0 input - 4:2:2 input	- scaling only (6-tap direct polyphase) - de-interlacing (see Table 3)	up-sample to 4:4:4	- color space conversion (full matrix plus offset) or - scaling only (6-tap direct or transposed polyphase)	down-sample to 4:2:x	- 4:2:0 output (2-3 planes) - 4:2:2 output (1-3 planes)
- 4:4:4 input (8-32 BPP) - LUT mode (8/4/2/1 BPP)	- scaling only (4-tap direct polyphase) - Anti Flicker (static 4-tap filter)	bypass		bypass	- 4:4:4 output (16-32 BPP) - 4:4:4:4 output (16-32 BPP)
- 4:4:4:4 input (16-32 BPP) - LUT mode (8/4/2/1 BPP)	- scaling only (3-tap direct polyphase) - Anti Flicker (static 3-tap filter)	bypass	- scaling only (3-tap direct polyphase)	bypass	

Table 3: De-Interlacing Mode Filter Lengths

Input Format	Vertical Scaling	Majority Select (Y:UV taps)	Median(Y:UV taps)
4:2:0 or 4:2:2 (semi) planar	downscale	6:3 (5:4)*	6:5
	zoom	5:3 (4:4)*	5:5 (6:4)*
4:2:2 single plane	downscale	not supported	6:6
	zoom	not supported	5:5

* numbers in parenthesis show alternate tap mode

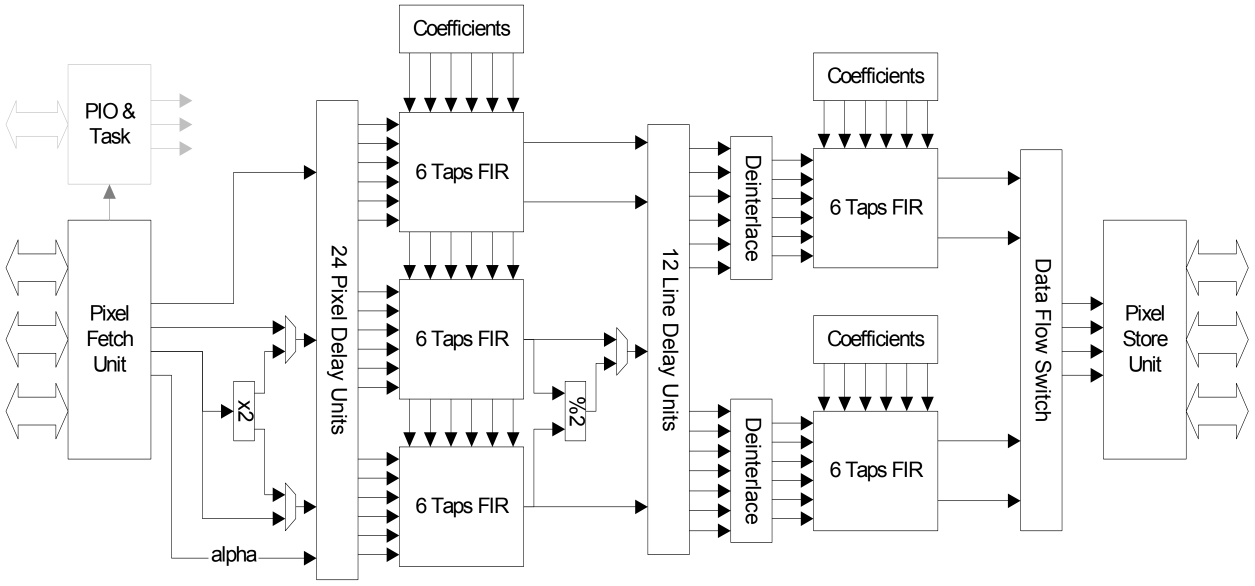


Figure 1: Simplified MBS Block Diagram

27.1.2 Operation

This section provides the details about how the Memory Based Scaler functions. A description of each functional group is provided.

27.1.2.1 Task Control

Because the MBS is capable of processing several video streams in sequence, a pipeline mechanism is implemented for scaling a sequence of tasks. A task is described by a data structure stored in memory. Writing the base address of a task into the Task FIFO schedules it to be executed after previously scheduled tasks have been processed.

In addition to the task list in the FIFO, each task structure in memory consists of a linked list of subtasks that will be executed in sequence (e.g., High Definition (HD) scaling task via partitioning). The software scheduling algorithm is responsible for keeping the task FIFO from overflowing. An interrupt can be generated once the last task in the FIFO gets executed to request new tasks from the scheduler. Other interrupt events exist to aid in keeping the task FIFO filled and to avoid overflowing the four available FIFO slots.

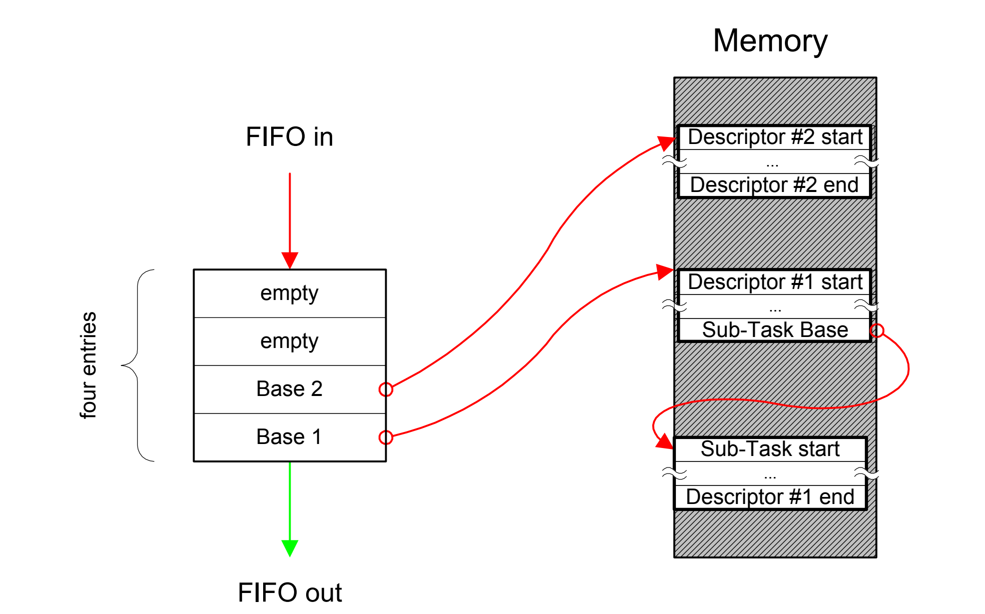


Figure 2: Four-Slot Task FIFO and Linked List

The table below shows the opcodes allowed in a descriptor list. The data structure consists of 32-bit words, therefore endianness rules apply. The command type is defined in the lower two bits of the 32-bit word. The remaining bits are decoded depending on the command type.

Table 4: Task Descriptor Opcode Table

Command	Bits	Function	Description
Jump	Link to address		
	25:3	address	Defines location where processing of commands continues
	1:0	opcode	= 00 (binary)
Exec / Stop	End of task list		
	2	stop flag	If this task is set, processing of the MBS task is stopped.
	1:0	opcode	= 01 (binary)
Queue	Start processing and queue next task		
	25:3	address	Task at this location is started after processing of current task finished
	1:0	opcode	= 10 (binary)
Load	Load register		
	31:24	count	Define number of registers to be loaded minus one
	19:16	mask	Write mask, if bit is zero according byte is written, if bit is set byte is not written
	11:2	Index	Load registers starting at offset
	1:0	opcode	= 11 (binary)

The following example shows how to setup a scaling task using the Color Space Matrix setup values shown in Table 13.

Table 5: Task Programming Example

Value	Meaning
0x0000_0207	Load one DWord @ offset 204
0x3100_0000	Value to be written to offset 204
0x0300_0223	Load four DWords @ offset 220
0x0CC0_0095	Value to be written to offset 220
0x398F_3895	Value to be written to offset 224
0x0004_0895	Value to be written to offset 228
0x1806_01f0	Value to be written to offset 22C
0x0000_0005	End of task / stop processing

27.1.2.2 Video Source Controls

Source video data for the MBS can be fetched using three dedicated DMA engines. It can be fetched from memory in several packed or planar formats. The source window is defined by one set of width and height values which is automatically translated into the corresponding number of 64-bit words to be fetched per line for each plane. Video lines can be fetched in reverse to allow horizontal flipping of images for applications like video conferencing.

To allow fetching of interlaced video lines from different locations in memory, each plane can be assigned a second set of base address registers. Pitches are defined separately only for luma and chroma planes. The lower three bits of the first three base address registers are used as an “intra long word offset” for the left-most pixel components of each line. The offset has to be a multiple of the number of bytes per component.

Fixed Input Formats

The fixed input formats available consist of indexed, packed and planar modes.

Table 6: Input Pixel Formats

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1-bit indexed																									I							
2-bit indexed																									2							
4-bit indexed																									4-bit							
8-bit indexed																									8-bit Index							
YUV 4:4:4, planar YUV 4:2:x, planar RGB 4:4:4, planar	plane #1																								Y8 or R8							
	plane #2																								U8 or G8							
	plane #3																								V8 or B8							

Table 6: Input Pixel Formats *…Continued*

Format	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
YUV 4:2:x, semi planar	plane #1		Y8
	plane #2	V8	U8
packed YUY2 4:2:2		U8 or V8	Y8
packed UYVY 4:2:2		Y8	U8 or V8
packed variable 16-bit		variable YUV or RGB 4:4:4	
packed variable 32-bit	variable YUV 4:4:4, 4:2:2 or RGB 4:4:4		

For indexed formats, sizes of 1, 2, 4 or 8 bits per pixel can be selected. A 32-bit color look-up table is used to expand indexed modes into true color including alpha values.

Predefined packed modes are available for YUY2 and UYVY formats. Other packed modes can be selected by using one of the three variable input formats described below.

Planar formats can be used to fetch pixel components from different locations in memory. In semi-planar modes a video field is defined by one plane for Y samples and one shared plane for U/V. In full planar formats, each component is fetched from a separate plane.

When processing 4:2:2 or 4:2:0 input stream the first sample from memory has to be a Y/U pair.

Variable Input Formats

In addition to the pre-defined fixed input pixel formats above, the MBS also includes three variable input format modes for packed 4:4:4 and 4:2:2 pixel extraction. In these modes the pixel components can be extracted from programmable locations within 32 or 16-bit units.

Table 7 shows an example for a 4:4:4, 32-bit pixel format. Table 8 shows an example for a 4:2:2 format. Since 4:2:2 pixel formats are based on meta-pixels (double pixels), programming of component location spans a 32-bit range.

Table 7: Variable Input Format Programming Example (32-Bit Pixels, aRGB)

Field	Value	Meaning
PFU_SIZE_C4 [2:0]	7	Size component #4 (alpha or V) Number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C4 [4:0]	24	Offset component #4 (alpha or V) Index of MSB position within 32-bit word (0-31)
PFU_SIZE_C3 [2:0]	7	Size component #3 (V or B or Y2) number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C3 [4:0]	0	Offset component #3 (V or B or Y2) index of MSB position within 32-bit word (0-31)

Table 7: Variable Input Format Programming Example (32-Bit Pixels, aRGB) *…Continued*

Field	Value	Meaning
PFU_SIZE_C2[2:0]	7	Size component #2 (U or G) number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C2[4:0]	8	Offset component #2 (U or G) index of MSB position within 32-bit word (0-31)
PFU_SIZE_C1[2:0]	7	Size component #1 (Y or R) number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C1[4:0]	16	Offset component #1 (Y or R) index of MSB position within 32-bit word (0-31)

Table 8: Variable Input Format Programming Example (16-Bit Pixels, YUY2)

Field	Value	Meaning
PFU_SIZE_C4 [2:0]	7	Size component #4 (V) Number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C4 [4:0]	24	Offset component #4 (V) Index of MSB position within 32-bit word (0-31)
PFU_SIZE_C3 [2:0]	7	Size component #3 (Y2) number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C3 [4:0]	16	Offset component #3 (Y2) index of MSB position within 32-bit word (0-31)
PFU_SIZE_C2[2:0]	7	Size component #2 (U) number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C2[4:0]	8	Offset component #2 (U) index of MSB position within 32-bit word (0-31)
PFU_SIZE_C1[2:0]	7	Size component #1 (Y) number of bits minus 1 (7 = 8 bits per component)
PFU_OFFS_C1[4:0]	0	Offset component #1 (Y) index of MSB position within 32-bit word (0-31)

27.1.2.3 Horizontal Video Filters

Interpolation Filter

All horizontal video processing is based on equidistant sampled components. All 4:2:2 video streams must therefore be up-sampled before being scaled horizontally. The interpolation FIR filter can interpolate interspersed or co-sited chroma samples. Mirroring of samples at the field boundaries compensates for run-in and run-out conditions of the filter.

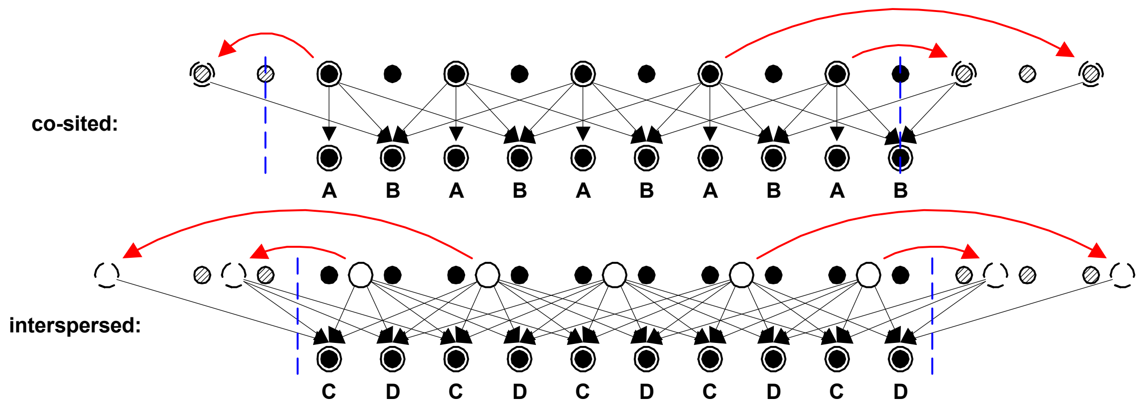


Figure 3: Interpolation FIR Filter Mirror Modes

The following coefficients are used:

- Co-sited: A=(1) and B=(-3,19,19,-3)/32
- Interspersed: C=(-1,5,13,-1)/16 and D=(-1,13,5,-1)/16

Decimation Filter

After horizontal processing, the chrominance may be down-sampled to allow higher quality vertical processing not otherwise available or to reduce memory bandwidth. Figure 4 shows the decimation FIR filter and the coefficients used. Mirroring of samples at the field boundaries compensates for run-in and run-out conditions of the filter.

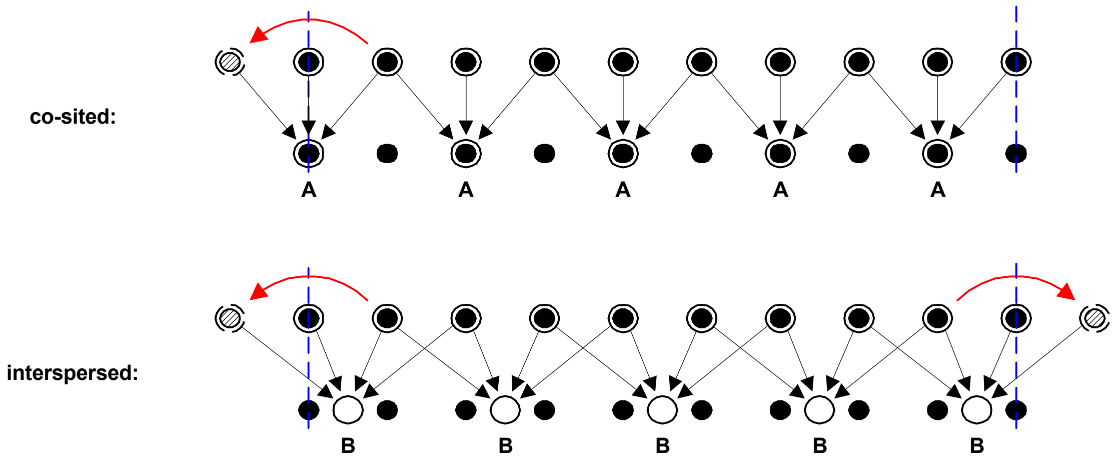


Figure 4: Decimation FIR Filter Mirror Modes

The following coefficients are used:

- Co-sited: lowpass A=(1,2,1)/4 or subsample A=(0,1,0)
- Interspersed: B=(-3,19,19,-3)/32

Normal Polyphase Filter Mode

The normal polyphase filter can be used to zoom or downscale a video image. Depending on the number of components, the filter has 6 taps (three component mode) or 3 taps (four component mode). Figure 5 shows the principle of a 6-tap FIR filter. Figure 5 shows how pixels between the six reference pixels can be interpolated using a lowpass filter. Figure 6 shows the interpolated pixel as the reference location which can be used to visualize the polyphase filter principle.

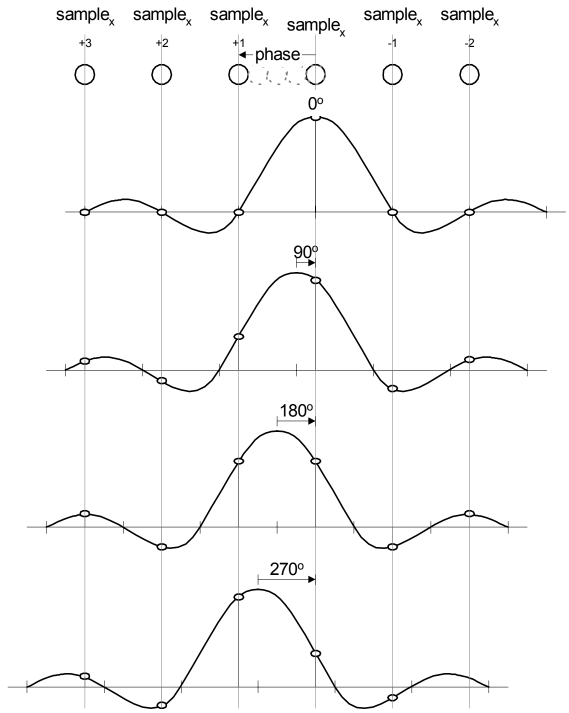


Figure 5: Normal Polyphase Filter (Zoom 400%)

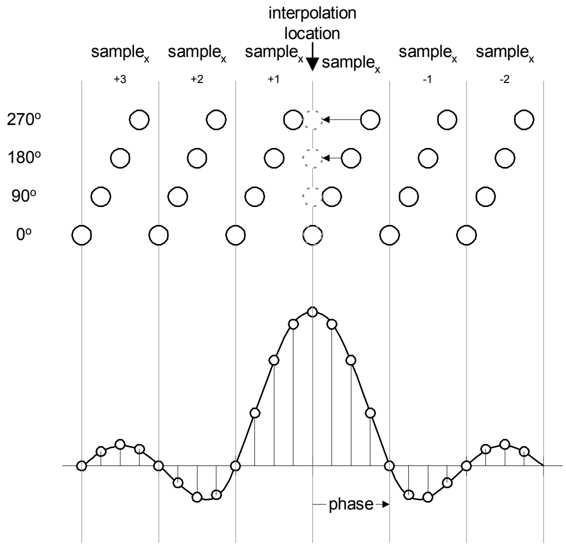


Figure 6: Interpolated Pixel Referenced to Sample

For the normal polyphase filter, run-in and run-out behavior are compensated by appropriate mirroring of samples at the field boundaries. In order to align the first output pixel with the geometrical location of the first input pixel, the start phase has to be chosen as:

$$\texttt{dto\_offset} = \texttt{0x200} \cdot \texttt{filter\_length}$$

For a 6-tap filter the default start phase would be 0x0C00, a 2-tap filter (linear phase interpolation) would lead to a start phase of 0x0400.

If other values are used (e.g., to skip mirrored field boundary samples) the effective input window width is reduced by one pixel for each additional 0x0400 added to the start phase value.

The phase is proportional to the zoom ratio 360^{o} = 0x1 0000h. This can also be expressed as *HSP_ZOOM_0* = 0x1 0000h / *zoom_x*, with *zoom_x* = *pixel_out* / *pixel_in*. The variable *pixel_in* does not include pixels needed during run-in or run-out phase of the filter. The amount of these pixels depends on the initial phase offset of the filter, as described above.

Three Component Mode

In three component mode, each component is assigned one of the three 6-tap FIR units. Coefficients for components 1, 2 and 3 are shared and taken from taps 0-5 in the horizontal coefficient table. Alpha processing in three component mode is not possible.

Four Component Mode

In four component mode, the three 6-tap FIR units are split into four 3-tap FIR units:

- Coefficients for component 1 (R or Y) are taken from tap 0-2 in the coefficient table.
- Coefficients for component 2 (G or U) are taken from tap 0-2 in the coefficient table.
- Coefficients for component 3 (B or V) are taken from tap 0-2 in the coefficient table.
- Coefficients for component 4 (alpha) are taken from tap 3-5 in the coefficient table.

Table 9: Normal Polyphase Programming Example Scaling 200%

Field	Value	Meaning
HSP_MODE	0x2	Normal polyphase filter mode
HSP_QSIGN	1	Compensates for negative coefficients
HSP_QMULTIPLY	0x100	m = 0.5, s = 0, to compensate for sum of coefficient being 2
HSP_QSHIFT	0x0	
HSP_PHASE_MODE	0x0	64 phases, optionally for scaling 200% the 2 phase setting could be used as well (affects coefficient position in table)
HSP_ZOOM_0	0x8000	Scaling ratio 200%

Table 9: Normal Polyphase Programming Example Scaling 200% *…Continued*

Field	Value	Meaning
HSP_OFFSET_0	0x0c00	Filter pre-loads 3 samples and then starts at phase 0 (the 1st input sample will be in the center of the 6-tap filter leading to 1st output sample being same as 1st input sample)
HSP_DZOOM_0	0x00000	Fixed scaling ratio
HSP_DDZOOM	0x0000	

Non-Linear Scaling

Panoramic scaling can be achieved by using the non-linear scaling mechanism. Non-linear scaling is based on two cascaded integrators which allow changing the scaling ratio in the shape of a parabola. The shape of the parabola is controlled by three coefficients. Based on these coefficients the zoom ratio at a specific output pixel calculates to:

$$z(x) = z_0 + x \cdot dz_0 + \frac{x \cdot (x \angle 1)}{2} \cdot ddz_0$$

with

$$z_0 = ZOOM\_0,\ \ dz_0 = DZOOM\_0,\ \ ddz_0 = DDZOOM$$

Figure 7 shows a diagram of the two cascaded integrators and the type of output signal after each integrating stage.

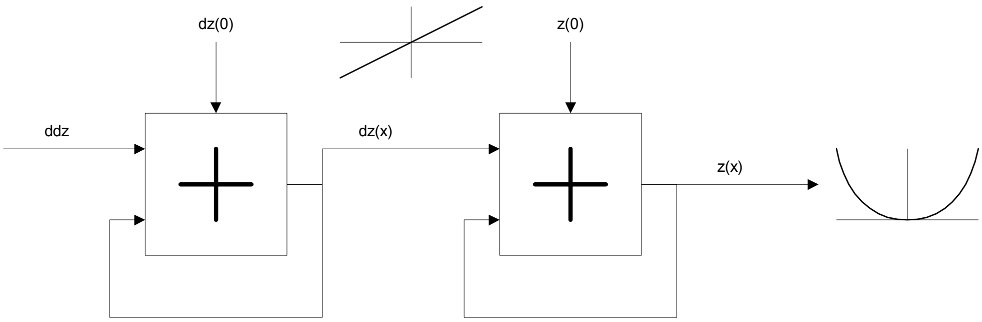


Figure 7: Cascaded Integrators

Based on the relation between zoom ratio at the corner and the middle of a line

$$\gamma = \frac{z_0}{z_{mid}}$$

and the effective zoom ratio

$$z_{eff} = \frac{pixel\_out}{pixel\_in}$$

the coefficients calculate to:

$$z_0 = z_{eff} \cdot \frac{3 \cdot n_{last} \cdot \gamma}{(\gamma + 2) \cdot n_{last} + 2 \cdot (\gamma \angle 1)}$$

$$dz_0 = \frac{4 \cdot z_0 \cdot (1 \angle \gamma) \cdot (n_{last} \angle 1)}{\gamma \cdot n_{last}^2}$$

$$ddz_0 = \frac{8 \cdot z_0 \cdot (\gamma \angle 1)}{\gamma \cdot n_{last}^2}$$

Table 10: Non-Linear Scaling Example

Field	Value	Meaning
HSP_MODE	0x2	Normal polyphase filter mode
HSP_QSIGN	1	Compensates for negative coefficients
HSP_QMULTIPLY	0x100	m = 0.5, s = 0, to compensate for sum of coefficient being 2
HSP_QSHIFT	0x0	
HSP_PHASE_MODE	0x0	64 phases
PFU_LSIZE	0x0c8	Input line length 200 pixels
PSU_LSIZE	0x122	Output line length 290 pixels (cropping)
HSP_OFFSET_0	0x0c00	Filter pre-loads 3 samples and then starts at phase 0 (the 1st input sample will be in the center of the 6-tap filter leading to 1st output sample being same as 1st input sample)
HSP_ZOOM_0	0x78a3	Panoramic scaling ratio, initial scaling ratio 212%, scaling ratio in center (145 output pixels) 120%, effective scaling ratio (290 output pixels) 145%
HSP_DZOOM_0	0x94969	
HSP_DDZOOM	0x1defaf8f	

Transposed Polyphase Filter Mode

The transposed polyphase filter can only be used to downscale. By basically reversing the flow through the normal polyphase filter, the transposed filter takes the output pixels as reference and accumulates input pixels. The advantage of the transposed polyphase filter is the longer effective filter length, which allows for filtering out high frequencies that interfere with downscaling. A drawback of this approach is that the filter coefficients must be optimized for the scaling ratio to avoid visible DC ripple effects. Compensation for run-in and run-out behavior of the filter is not available. Use of the transposed filter is limited to three components.

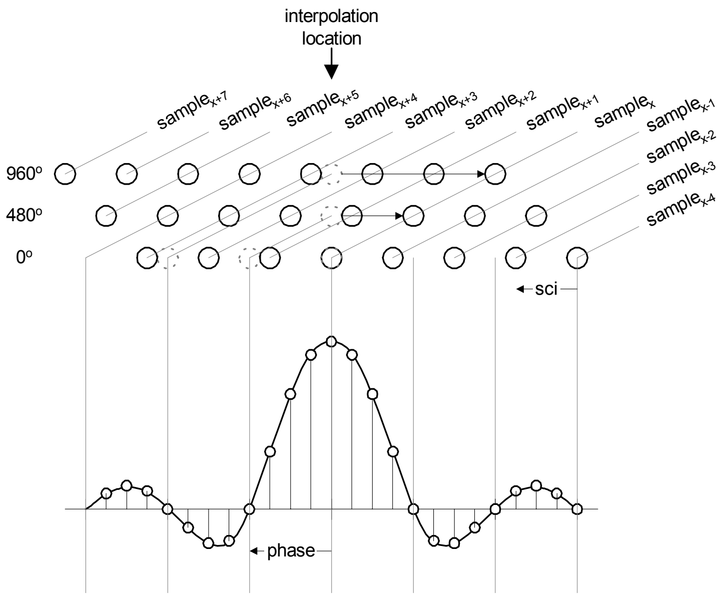


Figure 8: Transposed Polyphase Filter (Downscale 75%)

The scaling ratio for the transposed polyphase filter calculates to *HSP_ZOOM_0* = 0x1 0000h * *zoom_x*, with *zoom_x* = *pixel_out* / *pixel_in,* (not taking into account run-in and run-out losses of the filter).

Table 11: Transposed Polyphase Programming Example Scaling 50%

Field	Value	Meaning
HSP_MODE	0X3	Transposed polyphase filter mode
HSP_QSIGN	1	Compensates for negative coefficients
HSP_QMULTIPLY	0x100	m = 0.5, s = 0, to compensate for sum of coefficient being 2
HSP_QSHIFT	0x0	
HSP_PHASE_MODE	0x0	64 phases, optionally for scaling 50% the 2 phase setting could be used as well (affects coefficient position in table)
HSP_ZOOM_0	0x08000	Scaling ratio 50%
HSP_OFFSET_0	0x0000	Filter starts at phase 0
HSP_DZOOM_0	0x00000	No panorama mode available in transposed mode!
HSP_DDZOOM	0x0000	

Linear Phase Interpolation Mode

In linear phase interpolation, mode samples between two pixels are interpolated by using the linear equation px = (1-phase) * x_{left} + phase * x_{right} where * is phase E[0,1] or 0 $\leq$ phase <1. Linear phase interpolation creates adequate results especially when used with computer generated graphics which, unlike motion video, are normally not bandwidth limited.

LPI mode is enabled by setting HSP_PHASE_MODE = 7. Horizontal processing has to be set to normal polyphase (HSP_MODE = 2) and four component mode has to be enabled (HSP_FIR_COMP = 1).

The coefficients for tap 0-2 of all four components are then derived directly from the phase:

tap_0 = -phase;

tap_1 = -1 + phase;

tap_2 = 0;

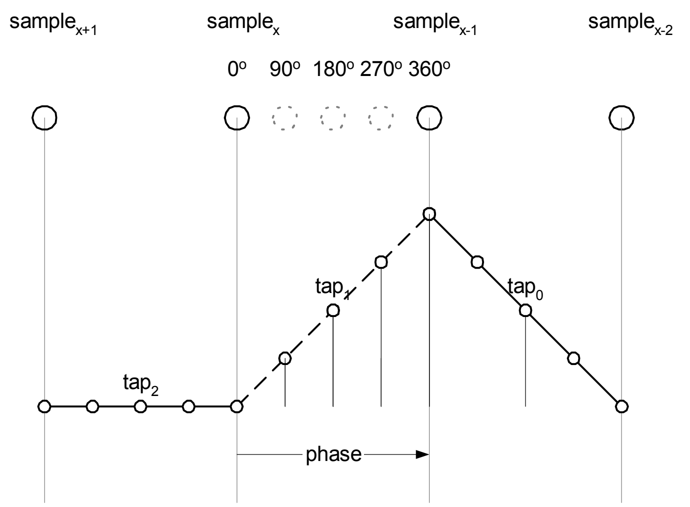


Figure 9: Linear Phase Interpolation (Zoom 200%)

The phase is proportional to the zoom ratio, 360^{o} = 0x1_0000h. This leads to *HSP_ZOOM_0* = 0x1 0000h / *zoom_x*, with *zoom_x* = *pixel_out* / *pixel_in*.

Table 12: Linear Phase Interpolation Programming Example Scaling 50%

Field	Value	Meaning
HSP_MODE	0X2	Normal polyphase filter mode
HSP_FIR_COMP	0X1	Four component mode
HSP_QSIGN	1	Negative coefficients
HSP_QMULTIPLY	0x100	m = 0.5, s = 1, no compensation required
HSP_QSHIFT	0x1	
HSP_PHASE_MODE	7	Linear interpolation
HSP_ZOOM_0	0x8000	Scaling ratio 50%
HSP_OFFSET_0	0x0400	Filter starts at phase 0 after prefetching one pixel
HSP_DZOOM_0	0x00000	Fixed scaling ratio
HSP_DDZOOM	0x0000	

Color Space Matrix Mode

In addition to normal and transposed polyphase filtering (scaling) the FIR filter structure can also be programmed to perform color space conversion functions. A dedicated set of registers holds the coefficients for the color space matrix (see Figure 10 for matrix variables and ranges).

Horizontal scaling and color space conversion are mutually exclusive.

$$\begin{bmatrix} a'_0 \\ a'_1 \\ a'_2 \end{bmatrix} = \left\{ \begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} \times \begin{bmatrix} a_0 + d_0 \\ a_1 + d_1 \\ a_2 + d_2 \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \end{bmatrix} \right\} \cdot 2^s \cdot m + \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

$$\angle 128 \le e_x \le 127.75 \quad \text{(in increments of 0.25)}$$
$$\angle 1.0 < m < 1.0$$
$$\angle 3 \le s \le 2$$
$$\angle 1.0 \le c_x < 1.0$$
$$\angle 128 \le d_x \le 255$$

Figure 10: Color Space Matrix Equation

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = 4 \cdot \begin{bmatrix} \frac{1.164}{4} & 0 & \frac{1.596}{4} \\ \frac{1.164}{4} & \angle\frac{0.392}{4} & \angle\frac{0.813}{4} \\ \frac{1.164}{4} & \frac{2.017}{4} & 0 \end{bmatrix} \times \begin{bmatrix} Y \angle 16 \\ U \angle 128 \\ V \angle 128 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

Figure 11: Example Color Space Matrix Equation

Figure 11 shows an example for YUV (CCIR) to RGB (full range) conversion. The multiply and quantization shift setting of 2^3 * 0.5 = 4 (s=3, m=0.5) are chosen based on the highest coefficient value of 2.017. Rounding is done to symmetrically distribute the error introduced by truncating of lower bits.

Table 13: Color Space Matrix Programming Example

Field	Value	Meaning
HSP_QSIGN	0	Positive
HSP_QMULTIPLY	0x100	m = 0.5
HSP_QSHIFT	3	s = 3

Table 13: Color Space Matrix Programming Example *…Continued*

Field	Value	Meaning
CSM_C_{00}	0x095	c_{00} = 149/512 = 1.164 / 4
CSM_C_{01}	0x000	c_{01} = 0
CSM_C_{02}	0x0CC	c_{02} = 204/512 = 1.596 / 4
CSM_C_{10}	0x095	c_{10} = 149/512 = 1.164 / 4
CSM_C_{11}	0x3CE	c_{11} = - 50/512 = - 0.392 / 4
CSM_C_{12}	0x398	c_{12} = - 104/512 = - 0.813 / 4
CSM_C_{20}	0x095	c_{20} = 149/512 = 1.164 / 4
CSM_C_{21}	0x102	c_{21} = 258/512 = 2.017 / 4
CSM_C_{22}	0x000	c_{22} = 0
CSM_D0_TWOS	1	Two’s complement
CSM_D1_TWOS	1	Two’s complement
CSM_D2_TWOS	1	Two’s complement
CSM_D_0	0xF0	d_0 = -16
CSM_D_1	0x80	d_1 = -128
CSM_D_2	0x80	d_2 = -128
CSM_E_0	0x00	e_0 = 0
CSM_E_1	0x00	e_1 = 0
CSM_E_2	0x00	e_2 = 0
SM = 0x3100_0000	Shift and signed magnitude	
C0 = 0x0CC0_0095	Matrix coefficients	
C1 = 0x398F_3895	Matrix coefficients	
C2 = 0x0004_0895	Matrix coefficients	
DE = 0x1806_01f0	Offsets	

27.1.2.4 Vertical Video Filters

The polyphase filter used for vertical processing can only be operated in normal polyphase mode. The filter can be used as a two component 6-tap filter, three component 4-tap filter or four component 3-tap filter (optionally linear phase interpolation). In two component mode, each filter unit can be run independently to allow 4:2:0 to 4:2:2 conversion or vice versa. See page 27-549 for details on normal polyphase filters.

Three Component Mode

In three component mode, the two 6-tap FIR units are split up into three 4-tap FIR units:

- Coefficients for component 1 (R or Y) are taken from tap 0-3 in the luma table.
- Coefficients for component 2 (G or U) are taken from tap 0-3 in the chroma table.
- Coefficients for component 3 (B or V) are taken from tap 4,5 in the luma and chroma table (order L4,L5,C4,C5).

Four Component Mode

In four component mode, the two 6-tap FIR units are split up into four 3-tap FIR units:

- Coefficients for component 1 (R or Y) are taken from tap 0-2 in the luma table.
- Coefficients for component 2 (G or U) are taken from tap 0-2 in the chroma table.
- Coefficients for component 3 (B or V) are taken from tap 3-5 in the luma table.
- Coefficients for component 4 (alpha) are taken from tap 3-5 in the chroma table.

.

Table 14: Vertical Scaling Four Component Processing

Field	Value	Meaning
VSP_MODE	0x2	Normal polyphase filter mode
VSP_FIR_COMP	0x1	Four component mode
VSP_QSIGN	1	Compensates for negative coefficients
VSP_QMULTIPLY	0x200	m = 0.5, s = 0, to compensate for sum of coefficient being 2
VSP_QSHIFT	0x0	
VSP_PHASE_MODE	0x0	64 phases
VSP_ZOOM_0	0x10000	Scaling ratio 100% (e.g. for anti-flicker filter)
VSP_OFFSET_0	0x0600	Filter starts at phase 32 after pre-fetching two lines
VSP_DZOOM_0	0x00000	Fixed scaling ratio
VSP_DDZOOM	0x0000	

Non-Linear Scaling

Same as horizontal scaling (see Section Non-Linear Scaling on page 27-552).

De-Interlacing and 4:2:0 Video Format Processing

The different de-interlacing modes available are shown in Table 3. Depending on the filter sizes, the vertical scaler start phase and line length difference must be programmed according to Table 15 below.

Table 15: De-Interlacing and 4:2:0 Mode Vertical Scaler Settings

(Y:UV taps)	Phase Luma*	Phase Chroma* (4:2:0)	Phase Chroma* (4:2:2)	Line Length Difference (4:2:0)	Line Length Difference (4:2:2)
6:6	0x0C00	0x0B	0x0C	-3	0
6:5	0x0C00	0x09	0x0A	-2	0
6:4	0x0C00	0x07	0x08	-1	0
6:3	0x0C00	0x05	0x06	+2	+1
5:5	0x0A00	0x08	0x08	-4	0
5:4	0x0A00	0x06	0x06	-3	0
5:3	0x0A00	0x04	0x04	0	+1
4:4	0x0800	0x07	0x08	0	0

*Add 400 when in Median mode, add 800 when using Majority Select Algorithm

If de-interlacing is enabled, the previous and current field have to be fed into the scaler as one interleaved frame. For Majority Select Algorithm, the first line has to originate in the previous field; while in Median mode, the first line has to be fetched from the current field. Base Address Mode = 1 can be selected to assist in weaving the two fields together into one frame.

Table 16: Vertical Scaling 200% 4:2:0 Processing

Field	Value	Meaning
VSP_MODE	0x2	Normal polyphase filter mode
VSP_QSIGN	1	Compensates for negative coefficients
VSP_QMULTIPLY	0x200	m = 0.5, s = 0, to compensate for sum of coefficient being 2
VSP_QSHIFT	0x0	
VSP_PHASE_MODE	0x0	64 phases, optionally for scaling 200% the 2 phase setting could be used as well (affects coefficient position in table)
VSP_ZOOM_0	0x8000	Scaling ratio 200%
VSP_OFFSET_0	0x0C00	Filter starts at phase 0 after pre-fetching three lines
VSP_OFFSET_C	0x0B	Filter pre-loads 2 lines and then starts at phase 48 (this setting reflects the geometrical alignment between luma and chroma lines in 4:2:0 format when using 6-tap filters each, see Table 15)
VSP_LDIFF_C	0x9	Line difference between luma and chroma at end of field set to a value of -3 (meaning fetching of chroma lines will stop three lines before last luma line is fetched, see Table 15)
VSP_DZOOM_0	0x00000	Fixed scaling ratio
VSP_DDZOOM	0x0000	

27.1.2.5 Color Key Processing

Color key processing (to convert alpha into color key values and/or vice versa) is only available on 4:4:4 video formats.

Color Key to Alpha Convert

To avoid color key values getting altered during video processing, color key matching samples can be tagged as transparent by generating an alpha value of zero at input of the MBS. Non-keyed samples get assigned a programmable alpha value. After video processing the alpha value can later be stored in memory together with the other components or can be used to re-key the keying color back into the video stream. (See “Alpha to Color Key Convert,” below.)

Color Key Replace

Scaling video streams that contain color keys can lead to artifacts at the corners between keyed and non-keyed samples. This is due to the “smear” effect of the lowpass filter and the highly saturated colors normally used for keying. To avoid these artifacts color keyed samples are replaced with either gray, black or the previous sample (gray if at the start of a line).

Alpha to Color Key Convert

If alpha after horizontal and vertical processing is below fixed threshold (0x80) sample is replaced by color key.

Fixed Alpha Insert

The alpha value defined in the Color Key register is inserted as a fourth component after horizontal and vertical processing.

27.1.2.6 Alpha Processing

Alpha processing is only available when the horizontal and vertical filter blocks are either bypassed or operated in four component mode. If the horizontal filter is operating in color space matrix mode alpha information is kept time aligned with the other three components.

27.1.2.7 Video Data Output

After processing the video stream the MBS can split the video data into one to three different streams. For each stream there is a memory base address. There are two line pitch registers further defining the DMA streams. The number of streams (planes) is defined by the output format register. Depending on its value the video components get packed into 64-bit words. These words will then get buffered and transferred to external memory in more effective clusters. A list of the supported output video formats is shown in Table 17. Packing of a pixel into 64-bit units is always done from right to left while bytes within one pixel unit are ordered according to endianness settings (according to the global endianness setting, the endianness bit in the output format register can invert that setting).

Table 17: Output Pixel Formats

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
planar YUV or RGB (4:4:4, 4:2:2 or 4:2:0)	plane #1																								Y8 or R8							
	plane #2																								U8 or G8							
	plane #3																								V8 or B8							
semi planar YUV (4:2:2 or 4:2:0)	plane #1																								Y8							
	plane #2																V8								U8							
packed 4/4/4 RGBa																	alpha				R4				G4				B4			
packed 4/5/3 RGBa																	alpha				R4				G5					B3		
packed 5/6/5 RGB																	R5					G6						B5				
packed YUY2 4:2:2																	U8 or V8								Y8							

Table 17: Output Pixel Formats *…Continued*

Format	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
packed UYVY 4:2:2			Y8	U8 or V8
packed 888 RGB(a)	(alpha)	R8 or Y8	G8 or U8	B8 or V8
packed 4:4:4 VYU(a)	(alpha)	V8	Y8	U8

Table 17 shows the location of the first “pixel unit” within a 64-bit word in little-endian mode. The selected endianness affects the position of components within multi-byte pixel units. Refer to Chapter 7 Pixel Formats, for more information.

Address Generation

Each of the three video planes is assigned a set of two base address registers and two line address pointers. Depending on the base address mode, video data of each plane is written using one pointer or using both pointers alternating each line. At the end of the line the pitch value is added to the active line address pointer. The pitch defines the difference in address between two vertically adjacent pixels. Pitch values are defined separately for chroma and luma components only. The lower three bits of the first three base address registers are used as an intra long word offset for the left-most pixel components of each line. The offset has to be a multiple of the number of bytes per component.

27.1.2.8 Interrupt Generation

The MBS contains a DVP-compliant interrupt generation mechanism. The following interrupt events are defined:

- **TASK_ERROR**: Current processing task leads to pipeline error. MBS has to be reset to resume new scaling task.
- **TASK_END**:Current task processing done, but some data might still remain in the output FIFO. This Interrupt denotes the point when the MBS is ready to start processing of a new task.
- **TASK_OVERFLOW**: Task FIFO overflow. Task request written into Task FIFO register got ignored.
- **TASK_IDLE**: Task finished (TASK_IDLE) and the task FIFO is empty.
- **TASK_EMPTY**: Task FIFO runs empty. Generation of this interrupt requires that there was a pending task in the task FIFO. This interrupt will not be generated if tasks are only submitted at MBS idle state.
- **TASK_DONE**: Current task finished and all writes complete. On reception of this interrupt all data will be accessible in main memory.

27.2 Register Descriptions

27.2.1 Register Map Address

The base address for the PNX8526 MBS registers is 0x10 C000.

Table 18: MBS Module Register Summary

Offset	Name	Description
0x10 C000	MBS_MODE	MBS operation mode
0x10 C040	TASK_QUEUE	Task queue FIFO
0x10 C044	TASK_STATUS	Task queue status
0x10 C100	PFU_FORMAT	Input format and mode
0x10 C104	PFU_WINDOW	Source window size
0x10 C108	PFU_VARFORM	Variable source format
0x10 C140	PFU_BASE1	Source base address DMA #1
0x10 C144	PFU_PITCH1	Source line pitch component 1
0x10 C148	PFU_BASE2	Source base address DMA #2
0x10 C14C	PFU_PITCH2	Source line pitch component 2 and 3
0x10 C150	PFU_BASE3	Source base address DMA #3
0x10 C154	PFU_BASE4	Source base address DMA #4
0x10 C158	PFU_BASE5	Source base address DMA #5
0x10 C15C	PFU_BASE6	Source base address DMA #6
0x10 C200	HSP_ZOOM_0	Initial zoom for 1st pixel in line (unsigned)
0x10 C204	HSP_PHASE	Horizontal phase control
0x10 C208	HSP_DZOOM_0	Initial zoom delta for 1 pixel in line (signed)
0x10 C20C	HSP_DDZOOM	Zoom delta change (signed)
0x10 C220	CSM_COEFF0	Color space matrix coefficients C_{00} - C_{02}
0x10 C224	CSM_COEFF1	Color space matrix coefficients C_{10} - C_{12}
0x10 C228	CSM_COEFF2	Color space matrix coefficients C_{20} - C_{22}
0x10 C22C	CSM_OFFS1	Color space matrix offset coefficients D_0-D_2
0x10 C230	CSM_OFFS2	Color space matrix rounding coefficients E_0-E_2
0x10 C240	VSP_ZOOM_0	Initial zoom for 1st pixel in line (unsigned)
0x10 C244	VSP_PHASE	Vertical phase control
0x10 C248	VSP_DZOOM_0	Initial zoom delta for 1 pixel in line (signed)
0x10 C24C	VSP_DDZOOM	Zoom delta change (signed)
0x10 C280	CKEY_MASK	Color key and alpha control
0x10 C284	CKEY_VALUE	Color key and alpha components
0x10 C300	PSU_FORMAT	Output format and mode
0x10 C304	PSU_WINDOW	Target window size
0x10 C340	PSU_BASE1	Target base address DMA #1
0x10 C344	PSU_PITCH1	Target line pitch component 1

Table 18: MBS Module Register Summary *…Continued*

Offset	Name	Description
0x10 C348	PSU_BASE2	Target base address DMA #2
0x10 C34C	PSU_PITCH23	Target line pitch component 2 and 3
0x10 C350	PSU_BASE3	Target base address DMA #3
0x10 C354	PSU_BASE4	Target base address DMA #4
0x10 C358	PSU_BASE5	Target base address DMA #5
0x10 C35C	PSU_BASE6	Target base address DMA #6
0x10 C400—C7FC	COLOR_TABLE	Color look-up table
0x10 C800—C9FC	COEFF_TABLE1	Coefficient table for horizontal filter (0-5)
0x10 CA00—CDFC	COEFF_TABLE2	Coefficient table for vertical luma filter (0-5)
0x10 CC00—CDFC	COEFF_TABLE3	Coefficient table for vertical chroma filter (0-5)
0x10 CFE0	INT_STATUS	Interrupt status register
0x10 CFE4	INT_ENABLE	Interrupt enable register
0x10 CFE8	INT_CLEAR	Interrupt clear register
0x10 CFEC	INT_SET	Interrupt set register
0x10 CFF4	POWERDOWN	Powerdown mode
0x10 CFFC	MODULE_ID	Module Identification and revision information

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Operating Mode Control Registers				
Offset 0x10 C000			*MBS Mode Control*	
31:30	R/W	0	DPM_SEQ	Data path sequence 0x = Bypass all filter stages (format conversion only). 10 = HSP -> VSP 11 = VSP -> HSP
29:26		-	Unused	
25:24	R/W	0	VSP_DEINTERLACE	De-interlacing mode 00 = No de-interlacing 01 = Vertical temporal median 10 = Majority selection algorithm (2 field) 11 = Majority selection algorithm (3 field) (Note: 3-field MS function is not implemented.)
23:22		-	Unused	
21:20	R/W	0	COEFF_LUT_MODE	Coefficient look-up table access mode 00 = Each table gets written separately. 01 = Writes to coefficient table 2 are copied into table 3. 10 = Writes to coefficient table 1 are copied into table 2. 11 = Writes to coefficient table 1 are copied into tables 2 and 3.
19		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
18	R/W	0	NO_AUTO_SKIP	Disable Auto Skip When set to zero (default) the MBS will automatically skip all remaining operations within a task once the TASK_DONE interrupt was generated. If enable the MBS will continue untill all pipeline stages are idle.
17	W	0	SKIP_TASK	Skip current task. Writing a one into this bit will reset the current task and continue with previously scheduled tasks.
16	W	0	SOFT_RESET	Soft reset Writing a one into this bit will reset the block and all outstanding scaling tasks.
15		-	Unused	
14	R/W	0	IFF_CLAMP	Clamp mode for IFF (affects U/V only). 0 = Clamp from 0-255. 1 = Clamp from 16 - 240 (CCIR range.
13:12	R/W	0	IFF_MODE	Interpolation mode 00 = Bypass 01 = Reserved 10 = Co-sited 11 = Interspersed
11		-	Unused	
10	R/W	0	DFF_CLAMP	Clamp mode for DFF (affects U/V only) 0 = Clamp from 0-255. 1 = Clamp from 16 - 240 (CCIR range).
9:8	R/W	0	DFF_MODE	Decimation mode 00 = Bypass 01 = Co-sited (subsample) 10 = Co-sited (lowpass) 11 = Interspersed
7	R/W	0	VSP_CLAMP	Clamp mode for VSP 0 = Clamp from 0-255. 1 = Clamp to CCIR range defined by VSP_RGB (bit 6.)
6	R/W	0	VSP_RGB	Color space mode, defines CCIR clamping range for VSP 0 = Processing in YUV color space (CCIR range: 16 - 235(Y), 16 - 240(U/V)) 1 = Processing in RGB color space (CCIR range: 16 - 235)
5:4	R/W	0	VSP_MODE	Vertical processing mode 00 = Bypass mode 01 = Reserved 10 = Normal polyphase mode 11 = Reserved

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
3	R/W	0	HSP_CLAMP	Clamp mode for HSP 0 = Clamp from 0-255. 1 = Clamp to CCIR range defined by HSP_RGB (bit 2).
2	R/W	0	HSP_RGB	Color space mode, defines CCIR clamping range for HSP 0 = Processing in YUV color space (CCIR range: 16 - 235(Y), 16 - 240(U/V)) 1 = Processing in RGB color space (CCIR range: 16 - 235)
1:0		0	HSP_MODE	Horizontal processing mode 00 = Bypass mode 01 = Color space matrix mode 10 = Normal polyphase mode 11 = Transposed polyphase mode

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Video Information Registers				
Offset 0x10 C040			*Task FIFO*	
31:3	W	0	TASK_BASE	Scale task FIFO Must be aligned to 8-byte boundary.
2		-	Unused	
1:0	W	0	TASK_CMD	Command mode 00 = Process task descriptor at given base. 01 = Start scaling with current register settings. 10 = Start and queue next. 11 = Reserved
Offset 0x10 C044			*Task Status*	
31:4		-	Unused	
3:0	R	0	TASK_PENDING	Number of pending scaling tasks (including current)

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Input Format Control Registers				
Offset 0x10 C100			*Input Format*	
31:30	R/W	0	PFU_BAMODE	Base address mode 00 = Single set (e.g., progressive video source) base 1-3 according to number of planes (plane 1-3) 01 = Alternate sets each line (e.g., interlaced video source) base 1-3, first line in, etc. (plane 1-3) base 4-6, second line in, etc. (plane 1-3) 1x = Reserved
29:16		-	Unused	
15	R/W	0	PFU_HFLIP	Mirror input mode 0 = Normal 1 = Mirrored input
14	R/W	0	PFU_AMI	Alternate memory interleaving format 0 = Default memory interleaving 1 = Memory interleaving used by MPEG block
13	R/W	0	PFU_ENDIAN	Input format Endianness 0 = Same as system Endianness 1 = Opposite of system Endianness
12:8		-	Unused	
7:0	R/W	0	PFU_IPFMT	Input formats 00 (hex) = YUV 4:2:0, semi-planar 03 (hex) = YUV 4:2:0, planar 08 (hex) = YUV 4:2:2, semi-planar 0B (hex) = YUV 4:2:2, planar 0F (hex) = RGB or YUV 4:4:4, planar 24 (hex) = 1-bit indexed 45 (hex) = 2-bit indexed 66 (hex) = 4-bit indexed 87 (hex) = 8-bit indexed A0 (hex) = Packed YUY2 4:2:2 A1 (hex) = Packed UYVY 4:2:2 AC (hex) = 16-bit variable contents 4:4:4 E8 (hex) = 32-bit variable contents 4:2:2 EC (hex) = 32-bit variable contents 4:4:4
Offset 0x10 C104			*Source Window Size*	
31:27		-	Unused	
26:16	R/W	0	PFU_LSIZE	Line size Defines size of input window. 1 = One pixel
15:11		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
10:0	R/W	0	PFU_LCOUNT	Line count Defines size of input window. 1 = One line
Offset 0x10 C108			***Variable Format Register***	
31:29	R/W	0	PFU_SIZE_C4 [2:0]	Size component #4 (alpha or V) Number of bits minus 1 (e.g., 7 = 8 bits per component)
28:24	R/W	0	PFU_OFFS_C4 [4:0]	Offset component #4 (alpha or V) Index of MSB position within 32-bit word (0-31)
23:21	R/W	0	PFU_SIZE_C3 [2:0]	Size component #3 (V or B or Y2) number of bits minus 1 (e.g., 7 = 8 bits per component)
20:16	R/W	0	PFU_OFFS_C3 [4:0]	Offset component #3 (V or B or Y2) index of MSB position within 32-bit word (0-31)
15:13	R/W	0	PFU_SIZE_C2[2:0]	Size component #2 (U or G) number of bits minus 1 (e.g., 7 = 8 bits per component)
12:8	R/W	0	PFU_OFFS_C2[4:0]	Offset component #2 (U or G) index of MSB position within 32-bit word (0-31)
7:5	R/W	0	PFU_SIZE_C1[2:0]	Size component #1 (Y or R) number of bits minus 1 (e.g., 7 = 8 bits per component)
4:0	R/W	0	PFU_OFFS_C1[4:0]	Offset component #1 (Y or R) index of MSB position within 32-bit word (0-31)

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Video Input Address Generation Control Registers				
Offset 0x10 C140			***Source Base Address #1***	
31:26		-	Unused	
25:3	R/W	0	PFU_BASE1	Base address DMA #1, used depending on PFU_BAMODE setting.
2:0	R/W	0	PFU_OFFSET1	Base address byte offset DMA #1. Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C144			***Source Line Pitch #1***	
31:15		-	Unused	
14:3	R/W	0	PFU_PITCH1	Line pitch DMA #1, signed value (two's complement) is used for all packed formats and for plane 1.
2:0		-	Unused	
Offset 0x10 C148			***Source Base Address #2***	
31:26		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
25:3	R/W	0	PFU_BASE2	Base address DMA #2, used depending on PFU_BAMODE setting.
2:0	R/W	0	PFU_OFFSET2	Base address byte offset DMA #2. Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C14C			***Source Line Pitch #2***	
31:15		-	Unused	
14:3	R/W	0	PFU_PITCH2	Line pitch DMA #2, signed value (two’s complement) is used for planes 2 and 3.
2:0		-	Unused	
Offset 0x10 C150			***Source Base Address #3***	
31:26		-	Unused	
25:3	R/W	0	PFU_BASE3	Base address DMA #3, used depending on PFU_BAMODE setting.
2:0	R/W	0	PFU_OFFSET3	Base address byte offset DMA #3. Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C154			***Source Base Address #4***	
31:26		-	Unused	
25:3	R/W	0	PFU_BASE4	Base address DMA #4, used depending on PFU_BAMODE setting.
2:0		-	Unused	
Offset 0x10 C158			***Source Base Address #5***	
31:26		-	Unused	
25:3	R/W	0	PFU_BASE5	Base address DMA #5, used depending on PFU_BAMODE setting.
2:0		-	Unused	
Offset 0x10 C15C			***Source Base Address #6***	
31:26		-	Unused	
25:3	R/W	0	PFU_BASE6	Base address DMA #6, used depending on PFU_BAMODE setting.
2:0		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Horizontal Video Processing Control Registers				
Offset 0x10 C200			*Initial Zoom*	
31:29	R/W	0	HSP_PHASE_MODE[2:0]	Phase mode 0 = 64 phases 1 = 32 phases 2 = 16 phases 3 = 8 phases 4 = 4 phases 5 = 2 phases 6 = Fixed phase 7 = Linear phase interpolation (only valid for 4 component mode)
28	R/W	0	HSP_NO_CROP	Disable line length cropping. 0 = Cropping enabled (default). 1 = Cropping disabled, used for striped scaling tasks.
27		-	Unused	
26	R/W	0	HSP_FIR_COMP[1:0]	Horizontal filter components 0 = Three components, 6-tap FIR each 1 = Four components, 3-tap FIR each In color space matrix mode this value has to remain zero.
25:20		-	Unused	
19:0	R/W	0	HSP_ZOOM_0[19:0]	Initial zoom for 1st pixel in line (unsigned, lsb = 2^{-16}) 2 0000 (hex) = Downscale 50% 1 0000 (hex) = No scaling = 100% 0 8000 (hex) = Zoom 2 x (transposed: downscale 50%) Values above 10000 (hex) are not valid in transposed mode.
Offset 0x10 C204			*Phase Control*	
31		-	Unused	
30:28	R/W	0	HSP_QSHIFT[2:0]	Quantization shift control is used to change quantization before being multiplied with HSP_MULTIPLY. 100 (bin) = Divide by 16. 101 (bin) = Divide by 8. 110 (bin) = Divide by 4. 111 (bin) = Divide by 2. 000 (bin) = Multiply by 1. 001 (bin) = Multiply by 2. 010 (bin) = Multiply by 4. 011 (bin) = Multiply by 8. Warning: A value range overflow caused by an improper quantization shift can not be compensated later by multiplying with a HSP_MULTIPLY value below 0.5.
27:26		-	Unused	
25	R/W	0	HSP_QSIGN	Quantization sign bit

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
24:16	R/W	0	HSP_QMULTIPLY[8:0]	Quantization multiply control is used to compensate for a different weight sum in transposed polyphase or color space matrix mode, remaining bits are fraction (largest number is 511/512). Value range: $0 \le m < 1.0$. Instead of using values in the range of $m < 0.5$ the quantization shift HSP_QSHIFT should be modified to gain more precision in the truncated result.
15:13		-	Unused	
12:0	R/W	0	HSP_OFFSET_0	Initial start offset for DTO
Offset 0x10 C208			***Initial Zoom delta***	
31:26		-	Unused	
25:0	R/W	0	HSP_DZOOM_0[25:0]	Initial zoom delta for 1 pixel in line (signed, lsb = 2^{-27}) is used for non constant scaling ratios.
Offset 0x10 C20C			***Zoom delta change***	
31:29		-	Unused	
28:0	R/W	0	HSP_DDZOOM[28:0]	Zoom delta change (signed, lsb = 2^{-40}) is used for non-constant scaling ratios.

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Color Space Matrix Registers				
Offset 0x10 C220			***Color space matrix coefficients C_{00} - C_{02}***	
31:30		-	Unused	
29:20	R/W	0	CSM_C02[9:0]	Coefficient C02 (signed, LSB = 2^{-9})
19:10	R/W	0	CSM_C01[9:0]	Coefficient C01 (signed, LSB = 2^{-9})
9:0	R/W	0	CSM_C00[9:0]	Coefficient C00 (signed, LSB = 2^{-9})
Note: Signed values are represented as two’s complement.				
Offset 0x10 C224			***Color space matrix coefficients C_{10} - C_{12}***	
31:30		-	Unused	
29:20	R/W	0	CSM_C12[9:0]	Coefficient C12 (signed, LSB = 2^{-9})
19:10	R/W	0	CSM_C11[9:0]	Coefficient C11 (signed, LSB = 2^{-9})
9:0	R/W	0	CSM_C10[9:0]	Coefficient C10 (signed, LSB = 2^{-9})
Offset 0x10 C228			***Color space matrix coefficients C_{20} - C_{22}***	
31:30		-	Unused	
29:20	R/W	0	CSM_C22[9:0]	Coefficient C22 (signed, LSB = 2^{-9})
19:10	R/W	0	CSM_C21[9:0]	Coefficient C21 (signed, LSB = 2^{-9})
9:0	R/W	0	CSM_C20[9:0]	Coefficient C20 (signed, LSB = 2^{-9})

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 C22C			*Color space matrix offset coefficients D_0 - D_2*	
31:29		-	Unused	
28	R/W	0	CSM_D2_TWOS	Offset coefficient D_2 type 0 = Unsigned 1 = Signed
27:20	R/W	0	CSM_D2[7:0]	Offset coefficient D_2 (LSB = 2^0)
19		-	Unused	
18	R/W	0	CSM_D1_TWOS	Offset coefficient D_1 type 0 = Unsigned 1 = Signed
17:10	R/W	0	CSM_D1[7:0]	Offset coefficient D_1 (LSB = 2^0)
9		-	Unused	
8	R/W	0	CSM_D0_TWOS	Offset coefficient D_0 type 0 = Unsigned 1 = Signed
7:0	R/W	0	CSM_D0[7:0]	Offset coefficient D_0 (LSB = 2^0)
Offset 0x10 C230			*Color space matrix offset coefficients E_0 - E_2*	
31:30		-	Unused	
29:20	R/W	0	CSM_E2[9:0]	Offset coefficient E2, two's complement (signed, LSB = 2^{-2})
19:10	R/W	0	CSM_E1[9:0]	Offset coefficient E1, two's complement (signed, LSB = 2^{-2})
9:0	R/W	0	CSM_E0[9:0]	Offset coefficient E0, two's complement (signed, LSB = 2^{-2})

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Vertical Video Processing Control Registers				
Offset 0x10 C240			*Initial Zoom*	
31:29	R/W	0	VSP_PHASE_MODE[2:0]	Phase mode 0 = 64 phases 1 = 32 phases 2 = 16 phases 3 = 8 phases 4 = 4 phases 5 = 2 phases 6 = Fixed phase 7 = Linear phase interpolation (valid only for 4 component modes).
28		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
27:26	R/W	0	VSP_FIR_COMP[1:0]	Vertical filter components 00 = Two components, 6-tap FIR each* 01 = Two components (alternative tap ratio) 10 = Three components, 4-tap FIR each 11 = Four components, 3-tap FIR each *Filter lengths differ when in de-interlacing mode.
25:20		-	Unused	
19:0	R/W	0	VSP_ZOOM_0[19:0]	Initial zoom for 1st pixel in line (unsigned, LSB = 2^{-16}) 2 0000 (hex) = Downscale 50% 1 0000 (hex) = No scaling = 2^{0} 0 8000 (hex) = Zoom 2 x
Offset 0x10 C244			***Phase Control***	
31		-	Unused	
30:28	R/W	0	VSP_QSHIFT[2:0]	Quantization shift control used to change quantization before being multiplied with VSP_MULTIPLY. 100 (bin) = Divide by 16. 101 (bin) = Divide by 8. 110 (bin) = Divide by 4. 111 (bin) = Divide by 2. 000 (bin) = Multiply by 1. 001 (bin) = Multiply by 2. 010 (bin) = Multiply by 4. 011 (bin) = Multiply by 8.
27:26		-	Unused	
25	R/W	0	VSP_QSIGN	Quantization sign bit
24		-	Unused	
23:21		-	VSP_LDIFF_C[2:0]	Line offset for chroma line count (signed) This setting can be used to manipulate the automatically calculated chroma line count. Usually this setting can be left at zero. (used for 4:2:0 scaling and deinterlacing only)
20:16	R/W	0	VSP_OFFSET_C[12:8]	Initial start offset for chroma DTO (is used for 4:2:0 scaling and de-interlacing only).
15:13		-	Unused	
12:0	R/W	0	VSP_OFFSET_0[12:0]	Initial start offset for DTO
Offset 0x10 C248			***Initial Zoom delta***	
31:26		-	Unused	
25:0	R/W	0	VSP_DZOOM_0[25:0]	Initial zoom delta for 1 pixel in line (signed, LSB = 2^{-27}) is used for non-constant scaling ratios
Offset 0x10 C24C			***Zoom delta change***	
31:29		-	Unused	
28:0	R/W	0	VSP_DDZOOM[28:0]	Zoom delta change (signed, LSB = 2^{-40}) is used for non-constant scaling ratios.

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Color Keying Control Registers				
Offset 0x10 C280			*Color Key Control*	
31:30	R/W	0	CKEY_K2A	Color key to alpha convert 00 = No alpha manipulation 01 = Fixed alpha at output 10 = Reserved 11 = Color key to alpha convert Alpha value for non-key sample is taken from CKEY_ALPHA register, alpha value for key sample is set to zero.
29:28	R/W	0	CKEY_A2K	Alpha mode to color key convert 00 = No alpha manipulation 01 = Reserved 10 = Reserved 11 = Alpha to color key convert Samples with alpha component below 80 (hex) are replaced with values defined in CKEY_COMP 1-3.
27:26	R/W	0	CKEY_REPLACE	Color keying replace mode 00 = No color component manipulation 01 = Replace keyed color components with black (10, 10, 10.) 10 = Replace keyed color components with gray (80, 80, 80). 11 = Replace keyed color components with last non-key value.
25:24		-	Unused	
23:16	R/W	0	CKEY_MASK1	Color key mask component 1defines bits of color component that are compared against color key value setting to key sample.
15:8	R/W	0	CKEY_MASK2	Color key mask component 2 defines bits of color component that are compared against color key value setting to key sample.
7:0	R/W	0	CKEY_MASK3	Color key mask component 3 defines bits of color component that are compared against color key value setting to key sample.
Offset 0x10 C284			*Color Key Components*	
31:24	R/W	0	CKEY_ALPHA	Alpha value defines the alpha value to be used for keyed samples.
23:16	R/W	0	CKEY_COMP1	Color key component 1 defines value of color key for component 1 (red or Y).
15:8	R/W	0	CKEY_COMP2	Color key component 2 defines value of color key for component 2 (green or U).
7:0	R/W	0	CKEY_COMP3	Color key component 3 defines value of color key for component 3 (blue or V).

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Video Output Format Control Registers				
Offset 0x10 C300			*Video Output Format*	
31:30	R/W	0	PSU_BAMODE	Base address mode 00 = Single set (e.g, progressive video source) base 1-3 according to number of planes (plane 1-3) 01 = Alternate sets each line (e.g. anti-flicker mode) base 1-3, first line out, etc. (plane 1-3) base 4-6, second line out, etc. (plane 1-3) 1x = Reserved
29:14		-	Unused	
13	R/W	0	PSU_ENDIAN	Output format Endianness 0 = Same as system Endianness 1 = Opposite of system Endianness
12		-	Unused	
11:10	R/W	0	PSU_DITHER	Output format dither mode 00 = No dithering 01 = Error dispersion (never reset pattern). 10 = Error dispersion (reset pattern at startup). 11 = Error dispersion (reset pattern every field.)
9:8	R/W	0	PSU_ALPHA	Output format alpha mode 00 = No alpha (alpha byte is not written to memory). 01 = Alpha byte written (see Color Keying Control). 10 = Reserved 11 = Reserved
7:0	R/W	0	PSU_OPFMT	Output formats 00 (hex) = YUV 4:2:0, semi-planar 03 (hex) = YUV 4:2:0, planar 08 (hex) = YUV 4:2:2, semi-planar 0B (hex) = YUV 4:2:2, planar 0F (hex) = RGB or YUV 4:4:4, planar A9 (hex) = Compressed 4/4/4 + (4-bit alpha) AA (hex) = Compressed 4/5/3 + (4-bit alpha) AD (hex) = Compressed 5/6/5 A0 (hex) = Packed YUY2 4:2:2 A1 (hex) = Packed UYVY 4:2:2 E2 (hex) = YUV or RGB 4:4:4 + (8-bit alpha) E3 (hex) = VYU 4:4:4 + (8-bit alpha)
Offset 0x10 C304			*Target Window Size*	
31:27		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
26:16	R/W	0	PSU_LSIZE	Line size is used for horizontal cropping after scaling: 0 = Cropping disabled. 1 = One pixel
15:11		-	Unused	
10:0	R/W	0	PSU_LCOUNT	Line count is used for vertical cropping after scaling: 0 = Cropping disabled. 1 = One line

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Video Output Address Generation Control Registers				
Offset 0x10 C340			***Target Base Address #1***	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE1	Base address DMA #1, used depending on PSU_BAMODE setting.
2:0	R/W	0	PSU_OFFSET1	Base address byte offset DMA #1 bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C344			***Target Line Pitch #1***	
31:15		-	Unused	
14:3	R/W	0	PSU_PITCH1	Line pitch DMA #1, signed value (two’s complement) is used for all packed formats and for plane 1.
2:0		-	Unused	
Offset 0x10 C348			***Target Base Address #2***	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE2	Base address DMA #2, used depending on PSU_BAMODE setting.
2:0	R/W	0	PSU_OFFSET2	Base address byte offset DMA #2 bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C34C			***Target Line Pitch #2***	
31:15		-	Unused	
14:3	R/W	0	PSU_PITCH2	Line pitch DMA #2, signed value (two’s complement) is used for planes 2 and 3.
2:0		-	Unused	
Offset 0x10 C350			***Target Base Address #3***	
31:26		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
25:3	R/W	0	PSU_BASE3	Base address DMA #3, used depending on PSU_BAMODE setting.
2:0	R/W	0	PSU_OFFSET3	Base address byte offset DMA #3 bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C354			*Target Base Address #4*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE4	Base address DMA #4, used depending on PSU_BAMODE setting.
2:0		-	Unused	
Offset 0x10 C358			*Target Base Address #5*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE5	Base address DMA #5, used depending on PSU_BAMODE setting.
2:0		-	Unused	
Offset 0x10 C35C			*Target Base Address #6*	
31:26		-	Unused	
25:3	R/W	0	PSU_BASE6	Base address DMA #6, used depending on PSU_BAMODE setting.
2:0		-	Unused	

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Miscellaneous Registers				
Offset 0x10 C400 - C7FC			*Color Look-Up Table*	
31:24	W	NI	LUT_ALPHA[x]	Alpha
23:16	W	NI	LUT_RED[X][7:0]	Red or Y
15:8	W	NI	LUT_GREEN[X][7:0]	Green or U
7:0	W	NI	LUT_BLUE[X][7:0]	Blue or V
Offset 0x10 C800 - C9FC			*Coefficient Table #1 Taps 0-5 (Horizontal) (64 entries x 64 bits)*	
63:62		-	Unused	
61:52	W	NI	TAP_5[X][9:0]	Inverted coefficient tap #5, two’s complement
51:42	W	NI	TAP_4[X][9:0]	Inverted coefficient tap #4, two’s complement
41:32	W	NI	TAP_3[X][9:0]	Inverted coefficient tap #3, two’s complement
31:30		-	Unused	
29:20	W	NI	TAP_2[X][9:0]	Inverted coefficient tap #2, two’s complement

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19:10	W	NI	TAP_1[X][9:0]	Inverted coefficient tap #1, two's complement
9:0	W	NI	TAP_0[X][9:0]	Inverted coefficient tap #0, two's complement
Offset 0x10 CA00 - CBFC			*Coefficient Table #2 Taps 0-5 (Vertical - Luma) (64 entries x 64 bits)*	
63:62		-	Unused	
61:52	W	NI	TAP_5[X][9:0]	Inverted coefficient tap #5, two's complement
51:42	W	NI	TAP_4[X][9:0]	Inverted coefficient tap #4, two's complement
41:32	W	NI	TAP_3[X][9:0]	Inverted coefficient tap #3, two's complement
31:30		-	Unused	
29:20	W	NI	TAP_2[X][9:0]	Inverted coefficient tap #2, two's complement
19:10	W	NI	TAP_1[X][9:0]	Inverted coefficient tap #1, two's complement
9:0	W	NI	TAP_0[X][9:0]	Inverted coefficient tap #0, two's complement
Offset 0x10 CC00 - CDFC			*Coefficient Table #3 Taps 0-5 (Vertical - Chroma) (64 entries x 64 bits)*	
63:62		-	Unused	
61:52	W	NI	TAP_5[X][9:0]	Inverted coefficient tap #5, two's complement
51:42	W	NI	TAP_4[X][9:0]	Inverted coefficient tap #4, two's complement
41:32	W	NI	TAP_3[X][9:0]	Inverted coefficient tap #3, two's complement
31:30		-	Unused	
29:20	W	NI	TAP_2[X][9:0]	Inverted coefficient tap #2, two's complement
19:10	W	NI	TAP_1[X][9:0]	Inverted coefficient tap #1, two's complement
9:0	W	NI	TAP_0[X][9:0]	Inverted coefficient tap #0, two's complement

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Interrupt and Status Control Registers				
Offset 0x10 CFE0			*Interrupt Status*	
31:30	R	0	STAT_PSU[1:0]	Status of pixel store unit (test signal)
29:27		-	Unused	
26:16	R	0	STAT_PSU_LINE	Status of pixel store unit, line currently written
15:14	R	0	STAT_VSP[1:0]	Status of vertical scale pipe (test signal)
13:12	R	0	STAT_DFF[1:0]	Status of decimation FIR filter (test signal)
11:10	R	0	STAT_HSP[1:0]	Status of vertical scale pipe (test signal)
9:8	R	0	STAT_IFF[1:0]	Status of interpolation FIR filter (test signal)
7:6	R	0	STAT_PFU[1:0]	Status of pixel fetch unit (test signal)
5	R	0	STAT_TASK_ERROR	Processing error

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R	0	STAT_TASK_END	Current task processing done
3	R	0	STAT_TASK_OVERFLOW	Task FIFO overflow
2	R	0	STAT_TASK_IDLE	Task finished and task FIFO is empty.
1	R	0	STAT_TASK_EMPTY	Task FIFO runs empty .
0	R	0	STAT_TASK_DONE	Current task finished and all writes are complete.
Offset 0x10 CFE4			***Interrupt Enable***	
31:6		-	Unused	
5	R/w	0	IEN_TASK_ERROR	Processing error
4	R/w	0	IEN_TASK_END	Current task processing done
3	R/W	0	IEN_TASK_OVERFLOW	Task FIFO overflow
2	R/W	0	IEN_TASK_IDLE	Task finished and task FIFO is empty.
1	R/W	0	IEN_TASK_EMPTY	Task FIFO runs empty .
0	R/W	0	IEN_TASK_DONE	Current task finished and all writes are complete.
Offset 0x10 CFE8			***Interrupt Clear***	
31:6		-	Unused	
5	W	0	CLR_TASK_ERROR	Processing error
4	W	0	CLR_TASK_END	Current task processing done
3	W	0	CLR_TASK_OVERFLOW	Task FIFO overflow
2	W	0	CLR_TASK_IDLE	Task finished and task FIFO is empty.
1	W	0	CLR_TASK_EMPTY	Task FIFO runs empty .
0	W	0	CLR_TASK_DONE	Current task finished and all writes are complete.
Offset 0x10 CFEC			***Interrupt Set***	
31:6		-	Unused	
5	W	0	SET_TASK_ERROR	Processing error
4	W	0	SET_TASK_END	Current task processing done
3	W	0	SET_TASK_OVERFLOW	Task FIFO overflow
2	W	0	SET_TASK_IDLE	Task finished and task FIFO is empty.
1	W	0	SET_TASK_EMPTY	Task FIFO runs empty .
0	W	0	SET_TASK_DONE	Current task finished and all writes are complete.
Offset 0x10 CFF4			***POWERDOWN***	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.

MEMORY BASED SCALER (MBS) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 CFFC			*Module ID*	
31:16	R	0x0119	MOD_ID	Module ID
15:12	R	0	REV_MAJOR	Major revision counter
11:8	R	1	REV_MINOR	Minor revision counter
7:0	R	0	APP_SIZE	Aperture Size 0 = 4kB

Chapter 28: Advanced Image Composition Processor (AICP)

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

28.1 Overview

This chapter describes the Advanced Image Composition Processor (AICP). The AICP provides a high resolution graphics controller with graphics and video processing. The AICP, in combination with other modules such as the 2D Drawing Engine, provide class-leading graphics and video capabilities.

Operations performed by the AICP include:

- Display image layer control, color/chroma keying, alpha-blending and processing for graphics and video such as 4:4:4 to 4:2:2 and 4:2:2 to 4:4:4 conversion and color space conversion.
- Motion Video processing (Motion video is the term used for moving images like motion pictures on an otherwise static graphic display.)
- Graphics modes beyond standard VGA
- Hardware Cursor management

The AICP provides advanced functionality with a series of layers and mixers. The AICP creates a series of display data layers (pixel streams) and mixes them logically in sequence to create the composited output picture.

The PNX8526 has two independent AICP modules. Due to the required display modes, AICP 1 contains a total of four layers and is intended to be connected to a TV or monitor. The AICP 2 contains two layers and is intended to connect to a VCR.

28.2 About Layers

A layer is essentially a display surface that provides a pixel stream to the image processor. The final display output is the intelligent composition of significant pixels from the several layers. Pixels from higher layers may replace or blend with those of lower layers. These operations are handled on a per-pixel basis in the layer mixer.

A layer obtains data from a particular data source. The data may provide a desktop image, a motion video image, or a cursor or sprite image. Registers in the layer select the data source and set display parameters such as size for the layer. In addition, functionality in a layer may include image filtering, gamma correction and color space conversion.

Input to a layer comes from the frame buffer or from other available data sources. Output from a layer goes to the accompanying mixer.

28.3 About Mixers

A Mixer is a functional block that selects between and manipulates data streams from two sources: the pixel stream from its companion layer and the pixel stream output of the previous mixer. The functions include pixel inversion, pixel selection (between sources), and alpha-blending. The mixer operates on a per-pixel basis using programmable logical raster operations to determine which functions to apply. The keys used in the raster operations include chroma keying (color keying on a color range) and color keying.

The output of a mixer is a continuous stream of pixels at the video clock rate going to the display device.

28.4 Video Overlay

The layering system is efficient for overlaying motion video on a underlying layer. A video stream may be “played” in off-screen memory and delivered to a certain position of the display. This allows the use of different color depths for each layer. The layering offers other advantages, including the ability to use color keying, chroma keying, or alpha-blending to define a non-rectangular (and/or semi-transparent) video or graphic overlay.

28.5 Supported Data Types

The AICP consists of four (for the primary display path) or two (for the secondary display path) generic layers and a background layer (24-bit registers). The background layer is represented by a 24-bit YUV/RGB color register value which is used if no other layer is active.

The other four layers have a generic structure which allows them to operate on the data types listed in the table below. The table shows the basic assignment of a certain portion of the data stream to a specific color component. The table does not show unpacked modes.

Every non-planar data format can have 4 or 8-bit alpha values associated with it. The alpha portion is usually located at the upper bit positions of the data chunk belonging to a pixel. However with the help of the channel-swap-register it is possible to move the position of the alpha portion of a pixel to another position. The actual number of bits belonging to a particular pixel can be set to any value within 32-bits to achieve arbitrary unpacked pixel formats. A more detailed description about data formats and how to program the AICP can be found in the Pixel Formatter section of this document.

Table 1: Data Formats

1 bpp RGB/YUV Indexed																															
31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P24						P31		P16							P23	P8							P15	P0						P7	
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
P56						P63		P48							P55	P40						P47		P32						P39	

2 bpp RGB/YUV Indexed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P12		P13		P14		P15		P8		P9		P10		P11		P4		P5		P6		P7		P0		P1		P2		P3	
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
P28		P29		P30		P31		P24		P25		P26		P27		P20		P21		P22		P23		P16		P17		P18		P19	

4 bpp RGB/YUV Indexed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P6				P7				P4				P5				P2				P3				P0				P1			
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
P14				P15				P12				P13				P10				P11				P8				P9			

8 bpp RGB/YUV Indexed / 8 bpp Gray RGB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P3								P2								P1								P0							
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
P7								P6								P5								P4							

3:3:2 RGB/YUV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C3_3			C2_3			C1_3		C3_2			C2_2			C1_2		C3_1			C2_1			C1_1		C3_0			C2_0			C1_0	
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
C3_7			C2_7			C1_7		C3_6			C2_6			C1_6		C3_5			C2_5			C1_5		C3_4			C2_4			C1_4	

4:4:4 RGB/YUV Packed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2_2				C1_2				C3_1				C2_1				C1_1				C3_0				C2_0				C1_0			
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
C1_5				C3_4				C2_4				C1_4				C3_3				C2_3				C1_3				C3_2			

5:3:4 RGB/YUV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	C2_2			C1_2				C3_1					C2_1			C1_1				C3_0					C2_0			C1_0			
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
C1_5				C3_4					C2_4			C1_4				C3_3					C2_3			C1_3				C3_2			

5:5:5 RGB/YUV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	C3_1					C2_1					C1_1						C3_0					C2_0					C1_0				
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
	C3_3					C2_3					C1_3						C3_2					C2_2					C1_2				

5:6:5 RGB/YUV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C3_1					C2_1						C1_1					C3_0					C2_0						C1_0				
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
C3_3					C2_3						C1_3					C3_2					C2_2						C1_2				

6:3:3 RGB/YUV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				C3_1						C2_1			C1_1							C3_0						C2_0			C1_0		
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
				C3_3						C2_3			C1_3							C3_2						C2_2			C1_2		

8:8:8 RGB/YUV

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1_1								C3_0								C2_0								C1_0							
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
C2_2								C1_2								C3_1								C2_1							

There is no support for YUV 4:2:2 indexed data formats and no support of any 4:2:0 formats. The color expansion for non 24-bit RGB colors can be done in two selectable ways:

1. Use source data left-aligned and fill lsbits with zeros.
2. Use source data left-aligned and replicate higher data bits into remaining lower bits.

Indexed data types always use the built-in 3x8x256 color look-up-table. The range depends on the width of the index used.

28.6 Operation

Figure 1 shows a high level block diagram of the AICP.

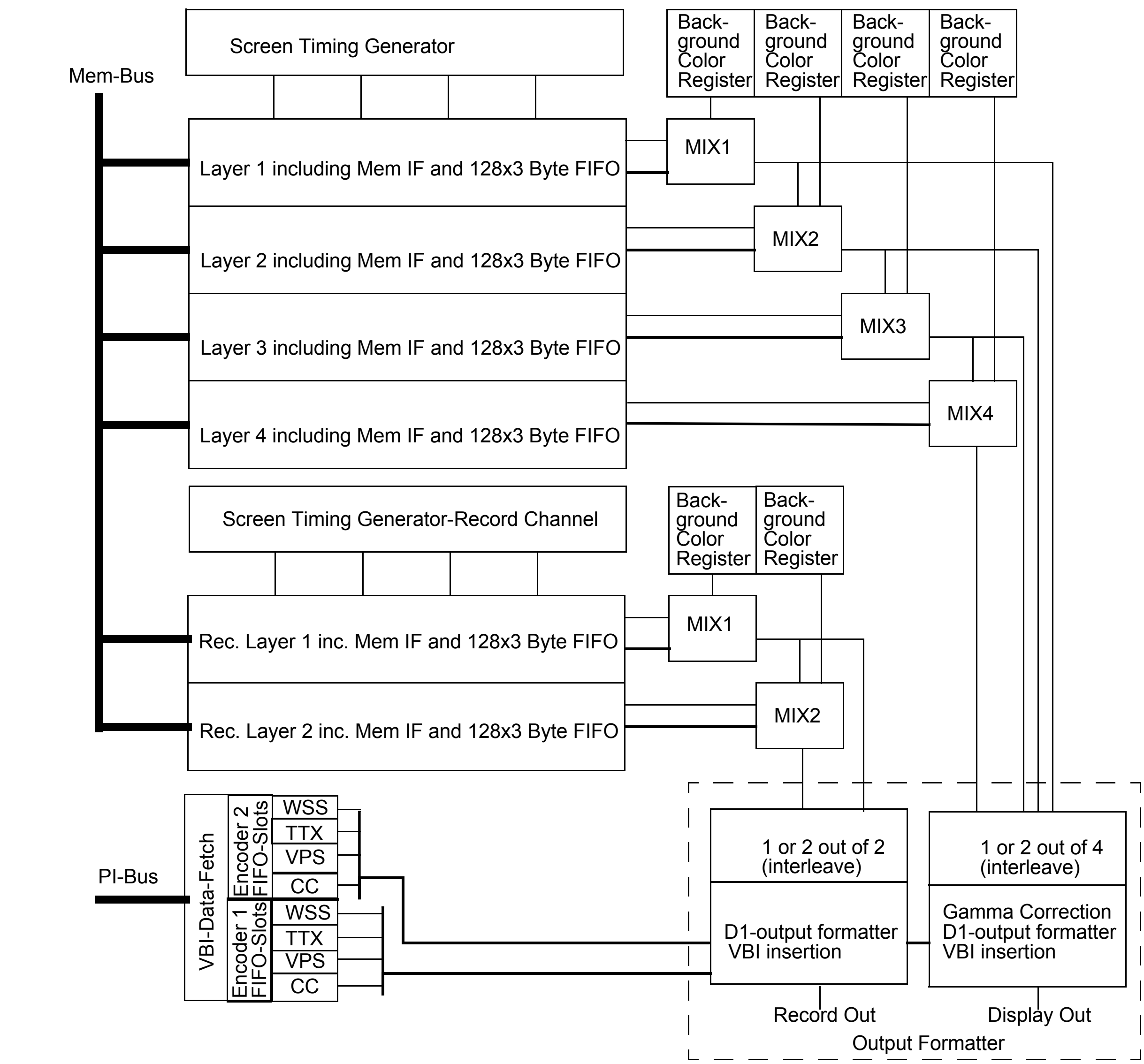

Mem-Bus
Screen Timing Generator
Back-ground Color Register
Back-ground Color Register
Back-ground Color Register
Back-ground Color Register
Layer 1 including Mem IF and 128x3 Byte FIFO
MIX1
Layer 2 including Mem IF and 128x3 Byte FIFO
MIX2
Layer 3 including Mem IF and 128x3 Byte FIFO
MIX3
Layer 4 including Mem IF and 128x3 Byte FIFO
MIX4
Screen Timing Generator-Record Channel
Back-ground Color Register
Back-ground Color Register
Rec. Layer 1 inc. Mem IF and 128x3 Byte FIFO
MIX1
Rec. Layer 2 inc. Mem IF and 128x3 Byte FIFO
MIX2
PI-Bus
VBI-Data-Fetch
Encoder 2 FIFO-Slots
Encoder 1 FIFO-Slots
WSS
TTX
VPS
CC
WSS
TTX
VPS
CC
1 or 2 out of 2 (interleave)
D1-output formatter VBI insertion
1 or 2 out of 4 (interleave)
Gamma Correction D1-output formatter VBI insertion
Record Out
Display Out
Output Formatter

Figure 1: AICP Block Diagram

The function of each of the components in the AICP is explained in more detail in the following sections.

28.6.1 Layer Structure

The AICP mainly consists of a multiple instantiation of two main components—layers and mixers. An AICP layer contains all the necessary functions to process and handle a single image surface. It is the layer's responsibility to produce a constant pixel stream for an image stored in the frame buffer based on the programmed pixel format, the layer position within the overall screen and the required pixel manipulation operations. The layer has a generic, highly programmable structure that allows it to process a wide variety of different images, such as live video and computer generated graphics.

The operation mode of a layer depends on the incoming pixel data format and the selected output mode of the AICP. To preserve image quality, it is best to perform as few conversions as possible on a specific image.

The layers are constructed in a way that allows the whole AICP pipeline to operate in either YUV or RGB mode. With this approach, the full RGB graphics bandwidth can be utilized for graphics-intensive applications.

Remark: The bandwidth can only be used by TVs capable of displaying RGB or YCRCB content, since the CVBS output will be bandwidth-limited in the video encoder. The RGB and encoder DACs cannot be used simultaneously (i.e., if the primary display should appear on a standard TV with a CVBS or S-Video input, the AICP pipe has to operate in YUV mode and RGB graphics content has to be converted to YUV 4:2:2).

If the display is supposed to appear at an RGB monitor type device or at a TV using the SCART input, the AICP will operate in RGB mode and YUV 4:2:2 material will be up-sampled and converted to RGB 8:8:8.

Standard video data coming from either a D1 input or from the MPEG decoder will be available in the frame buffer as a 4:2:2 YUV data stream either directly written by the relevant agent or converted from 4:2:0 to 4:2:2 by the memory based scaler (MBS). The AICP will bring this format to YUV 4:4:4 either by duplicating the color information to both neighbor Y samples or by shifting the existing and interpolating the missing chroma samples. The mode of operation depends on the display media. If it is a standard TV with either composite or S-Video input it is preferable to keep the video format in the 4:2:2 color resolution and just expand it by sample duplication. As the final video data stream is composed and converted to a CCIR-656 data stream going into a digital encoder, the duplicated samples can be discarded again.

Figure 2 shows the function of one AICP layer.

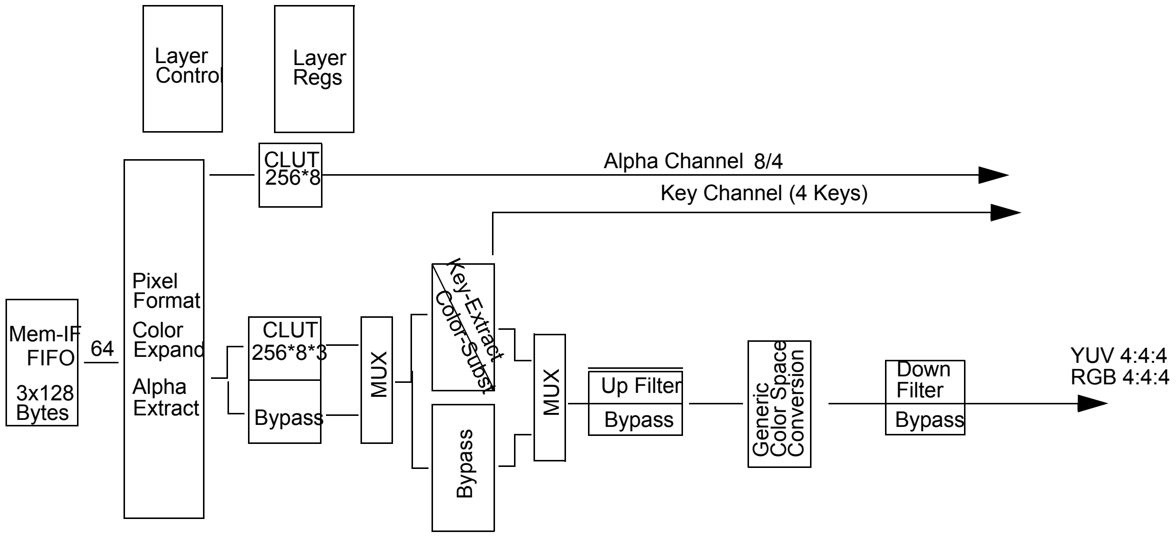


Figure 2: AICP Layer Structure

28.6.2 Pixel Formatter

The way an image or video data is stored in the frame buffer depends on the operating system that generates the graphics data and the format of the incoming video data stream. Different RGB and YUV data formats are defined by various operating systems. The Pixel Formatter follows a rather generic approach to deal with this wide variation in formats. Two register settings are important in order to program the correct pixel format.

1. The total number of bits per pixel must be defined. This number has to account for all bits including unused bits (for unpacked formats) and potential alpha values.
2. The multiplex order must be specified. This register controls the way the incoming data stream is de-multiplexed—i.e., how many bits of the incoming data word each of the three internal channels (RGB/YUV) gets assigned.

Each of the formats can also contain a per pixel 4 or 8-bit alpha value or a global surface alpha value programmed into a register. A layer’s alpha behavior is specified by the alpha parameter.

28.6.3 Color Look-Up Table

Each of the layers contains a look-up table for each color component and for the potential alpha value of a pixel. The look-up tables have a depth of 256 8-bit words each. The LUTs can be loaded and read out independently.

The basic function of a LUT is to expand indexed color formats. However, since the addresses of the LUTs are not linked together, gamma correction on a component basis is also possible.

28.6.4 Chroma/Color Keying

Chroma keying allows overlaying video dependent on whether the pixel lies within a range of color values. This feature allows video transparency or “green screening”, which is a technique for compositing foreground imagery with a background. Chroma key matching is performed on a range of values rather than on a single value, as in color keying.

A Key Mask is provided to allow chroma keying on specific bits within the data field.

Color keying is considered to be a subset of chroma keying and can be achieved by setting the color range accordingly. The chroma/color key result is used for mixing functions downstream. For example, color keying can be used to determine which pixels should contain motion video. The color key function will compare the pixels of a layer with the color key register and place motion video from another layer in those pixels that match.

The Color Key is generated in each layer (before color space conversion). Each byte lane of the expanded RGB 8:8:8 or YUV 4:4:4 new pixel is passed through an 8-bit Color Key AND mask. The result in each channel is compared to the register values for Color Key Lower Range and Color Key Upper Range. Every layer can use up to four color keys.

When the pixel component is equal to a range register value or is within the range, the result for that byte lane is a true (1). If it’s outside the range, the result is 0. The results from the three-byte lanes are used as keys in the 8-bit Color Key Combining ROP. This ROP is located in the layer and should not to be confused with the ROP in the mixer that follows each layer.

- **CnKey0_Layer** is determined by checking if the B(V) color component is within the chroma key range.
- **CnKey1_Layer** is determined by checking if the G(U) color component is within the chroma key range.
- **CnKey2_Layer** is determined by checking if the R(Y) color component is within the chroma key range.

“Cn” represents one of the four possible key colors. Since four colors are possible per layer, four different ROPs exist for determining which component of which key color should lead to a color key hit.

If the color component is within the range, the key is set to 1. If it’s not within the range, the key is set to 0.

Figure 3 illustrates the function of one color key/color exchange operation. Each AICP layer color key unit contains four color key blocks. Each color key block operates on a separate register set.

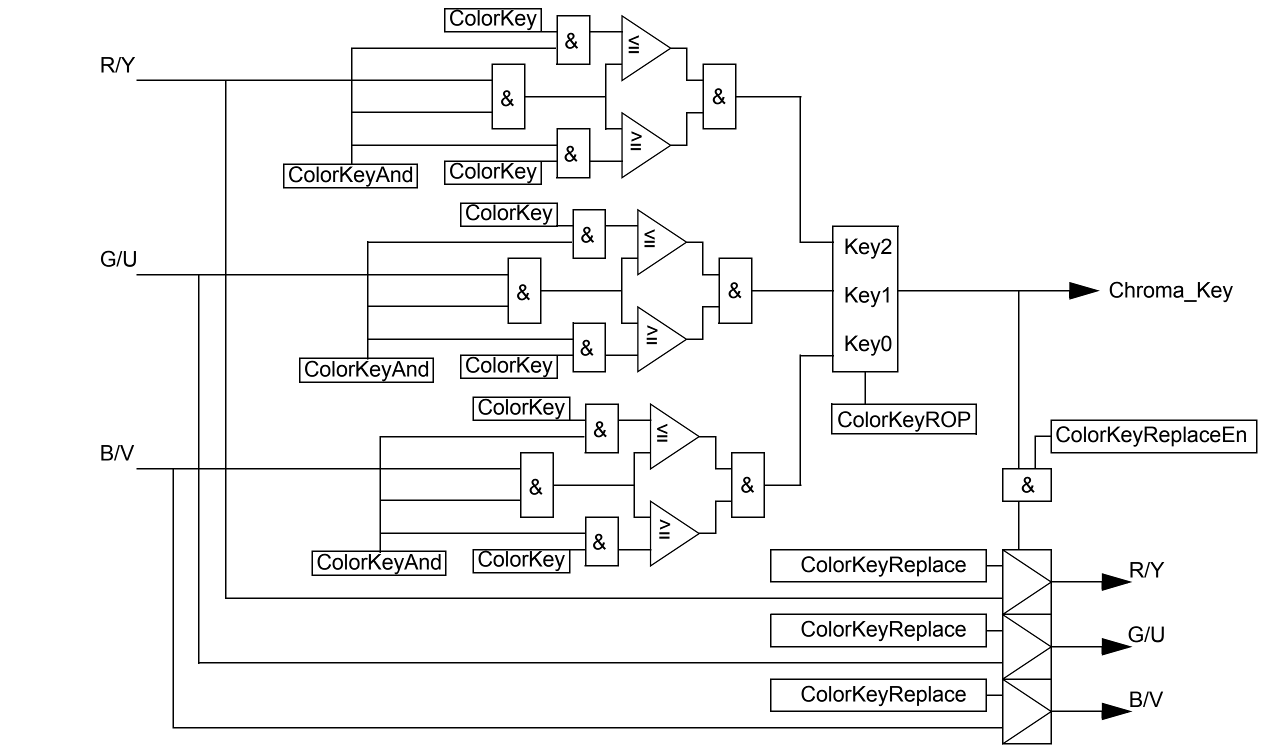


Figure 3: Color Key Unit

28.6.5 Up Sampling Filter/Shift Filter

The up-sampling/shift filters take over the task of converting 4:2:2 based material to a 4:4:4 chrominance resolution. This is necessary if an AICP layer is supposed to output RGB or YUV data in full color resolution to be displayed either on a TV with SCART input, an HDTV, or other CRT-like devices. Two flavors of 4:2:2 formats are known: co-sited and interspersed.

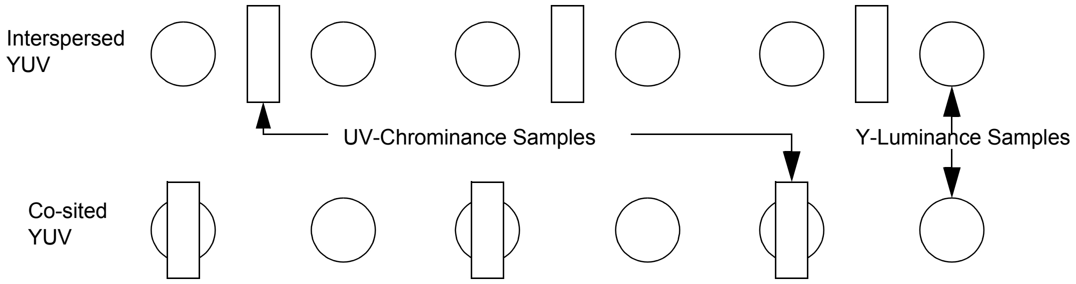


Figure 4: Interspersed and Co-Sited Formats

The up-sampling from 4:2:2 YUV data to 4:4:4 can be done in two separate filters, depending on the input format of the YUV data stream. In the case of interspersed chrominance pixels, a conversion to a co-sited and/or 4:4:4 format is done by applying a phase shift filter with the following transfer function:

1. $-1+5z^{-1}+13z^{-2}-1z^{-3}$ (for phase shifting only)
2. $-1+13z^{-1}+5z^{-2}-1z^{-3}$ (additional output for 4:2:2 to 4:4:4 conversion)

For phase shifting (4:2:2 interspersed to 4:2:2 co-sited) only filter 1 above is applied. For transfer of 4:2:2 interspersed to 4:4:4 filters 1 and 2 are used

For transformation of a 4:2:2 co-sited to a 4:4:4 format a different filter is used. This filter has the following transfer function:

3. $-3+19z^{-1}+19z^{-2}-3z^{-3}$ (up-sample filter)

Different filter settings can achieve similar results. However, for interspersed to 4:4:4 or to 4:2:2 co-sited conversion, only the shift-filter should be used. For 4:2:2 co-sited to 4:4:4 conversion, only the up-sample filter should be used. If the shift filter is used together with the up-sample filter, additional distortions due to rounding effects will result. (A bad example would be: shift interspersed to co-sited and use up-sample filter for final 4:2:2 to 4:4:4 conversion.)

28.6.6 Color Space Matrix

The color space matrix can be used to transform color components from one color space into another. Each layer contains a universal color space matrix with a set of programmable coefficients to assure that all possible YUV and RGB data format conversions can be accomplished. With this approach any flavor of RGB to YUV, YUV to RGB and YUV to Y’U’V’ conversion is covered.

The color space conversion results in a data path expansion to 9 bits. Due to the generic nature of the color space matrix it is also possible to change the contrast brightness and saturation settings for YUV data streams by changing the coefficients accordingly.

The matrix works mathematically as follows:

$$\begin{bmatrix} \mathrm{POut1} \\ \mathrm{POut2} \\ \mathrm{POut3} \end{bmatrix} = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{In1} \\ \mathrm{In2} \\ \mathrm{In3} \end{bmatrix}$$

$$\mathrm{Out1} = \mathrm{POut1} + \mathrm{Offset1}$$

$$\mathrm{Out2} = \mathrm{POut2} + \mathrm{Offset2}$$

$$\mathrm{Out3} = \mathrm{POut3} + \mathrm{Offset3}$$

The matrix coefficients a11 to a33 are representing 10-bit wide two’s complement values. See Table 2.

Remark: It is important to program the two's complement bits correctly since they assure correct rounding, clipping and sign extension.

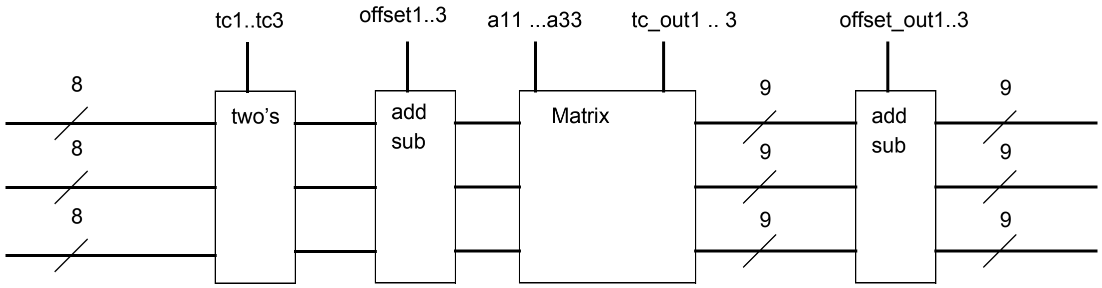


Figure 5: Matrix Block Diagram

Table 2: Coefficients for Color Space Conversion—RGB to YUV

a11=256*0.859*0.299	a12=256*0.859*0.587	a13=256*0.859*0.114
a21=-256*0.564*0.299	a22=-256*0.564*0.587	a23=256*0.564*0.886
a31=256*0.713*0.701	a32=-256*0.713*0.587	a33=-256*0.713*0.114

The inverse of the coefficient matrix shown above, with the assumption that the YUV data entering the matrix is limited in range as mentioned above, results in the coefficients shown in Table 3:

Table 3: Coefficients for Color Space Conversion—YUV to RGB

a11=256*1	a12=256*0	a13=256*1.371
a21=256*1	a22=-256*0.336	a13=-256*0.698
a31=256*1	a32=256*1.732	a33=256*0

28.6.7 Down Filter

The function of the down filter is to limit the chroma frequency components to meet the Nyquist criteria before down sampling the chrominance data stream from 4:4:4 to 4:2:2. This operation will only be carried out if the layer output is intended to be converted to a CCIR-656 D1 data stream, which is taken by a digital video encoder to generate S-Video or CVBS output formats. The filter is a combination of several subfilters operating according to the following transfer function:

$$(-0.5 + 4*z^{2} - 0.5*z^{-4}) * (1 + z^{-1} + z^{-2}) - z^{-3}) * (0.5 + z^{-1} + 0.5*z^{-2})$$

28.6.8 Screen Timing Generator

The Screen Timing Generator (STG) creates the required synchronization signals for the monitor or other display devices. The screen timing generator usually operates as the timing master in the system. However, it is also possible to synchronize the operation of the screen timing generator to external events i.e., a vertical synchronization signal. The screen timing generator also defines the active display region. The coordinate system for the STG is (x, y), with (0, 0) referring to the top left of the screen. The coordinate (Horizontal Total, Vertical Total) defines the bottom right of the screen. Horizontal and vertical blanking intervals, synchronization signals, and the visible display are within these boundaries.

The following rules apply to the register settings specifying the screen timing:

- Total number of pixel per line: HTOTAL + 1
- Total number of lines per field: VTOTAL + 1
- 0,1 < horizontal blanking start <=HTOTAL
- 0,1 < horizontal sync start <= HTOTAL
- 0 < vertical blanking start <= VTOTAL
- 0 < vertical sync start <= VTOTAL
- HSYNC - must be asserted or negated for at least one clock
- VSYNC - must be asserted or negated for at least one scanline
- HBLNK - must be asserted or negated for at least two clocks
- VBLNK - must be asserted or negated for at least two scanlines

The state change of the odd_even signal is always tied to the rising edge of the vsync signal.

Figure 6 identifies screen display parameters controlled by fields in the STG registers.

,

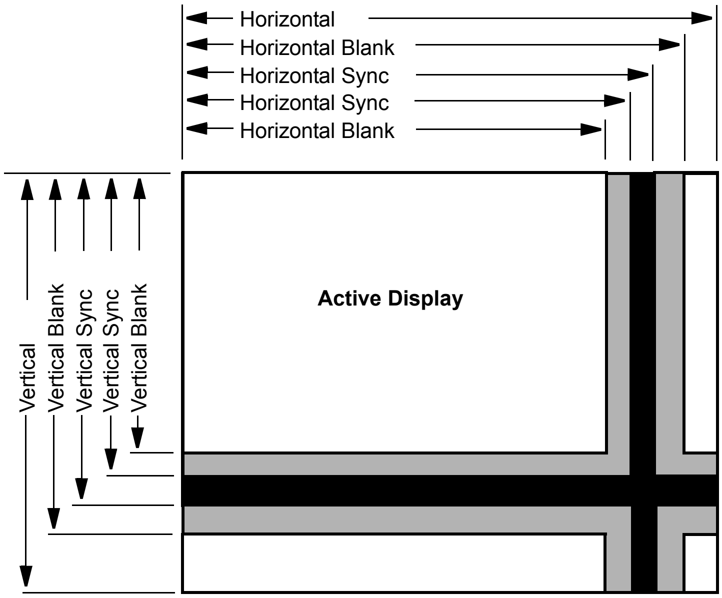


Figure 6: Video Frame Screen Timing

28.6.9 Programming the STG

Because the STG coordinate system begins at (0,0), it is necessary to program certain registers to be one less than the desired value. For example, a scanline has 800 pixels total. The horizontal total should be set to 799 because 0—799 is a total of 800. The same applies to programming the vertical total.

In the vertical domain, there are three main timing intervals to set: vertical active time, vertical blank time, and vertical sync time. The position of the vertical sync defines the vertical front and back porches.

Remark: The vertical sync interval (and therefore vertical blanking) must be a minimum of one line in duration.

The STG has no specific requirement for horizontal blank and sync. The location, duration and even existence of horizontal blank and sync times are entirely dependent on the display surface. If the display surface does not require horizontal blanking, it’s not necessary to program it into the STG.

Non-blanked area occurs when the currently active line is not within the vertical blanking interval or in the column of the horizontal blanking interval. Display layers can be programmed to reside on any portion of the screen. Any non-blanked screen position that does not have an active display layer pixel assigned to it will result in the background color or the previous layer pixel being displayed.

The AICP provides clipping support at the edge of the defined H and V Total. If a layer is positioned so that some part of it would exceed the overall screen dimensions, no wrapping occurs. Instead, the pixel layering in this area is marked as invalid and is not displayed.

The AICP also supports negative screen positions i.e., top and left side clipping of layers. For negative x and y layer start positions, the following equations must be used:

```
if StartX < 0 then
StartX = Xtotal + 1 - ABS(StartX)
set StartX sign bit
StartY = Ydisplay - 1
else
StartX = StartX
StartY = StartY

if StartY < 0 then
StartY = Ytotal + 1 - ABS(StartY)
set StartY sign bit
else
StartY = StartY
```

In addition to the standard progressive AICP display mode, another mode called “interlaced” can be switched on by setting the Interlaced control bit. In this mode the V Total register no longer specifies the height of a frame but the height of a field. The field height alternates by one line depending on whether an odd or even field needs to be processed by the AICP. Four registers are provided for this mode to specify the actual location of the vsync signal within a line in odd and even fields.

28.6.10 Changing Timing

All register settings to the timing generator take effect immediately and are not clock resynchronized. (The start/stop bit is an exception. It takes effect immediately and is clock re-synchronized.) The only safe way to change screen timing is as follows:

1. Turn off the timing generator.
2. Program all registers needed in the new display mode.
3. Turn the timing generator back on. In the process, the entire display pipeline is reset. All display layers are reset, and the screen timing starts at the vertical total, which guarantees a complete vertical blank period and vertical sync signal at the start of any mode change.

28.6.11 Mixer

The main functionality of the mixer stages is to compose the outgoing pixel streams from each layer to the final display image. The mixer data path operates at 9 bits. This includes clipping, alpha-blending and inverting colors. Which of those functions is applied, and how, is defined in a set of raster operations (ROPs). A raster operation is always a logical combination of several input keys and a specific ROP register which enables one or more of the different key combinations.

Each mixer knows four different keys (Key0, Key1, Key2, Key3) as illustrated in the mixer block diagram.

- -Key0 (output key from previous layer, output of KeyPass ROP)
- -Key1 (current color/chroma key pixel key) or (new pixel key 1)
- -Key2 (New Pixel Key 2)
- -Key3 (color key of the previous pixel)

Since each layer can have four color keys, the output is a 4-bit vector specifying which keys match the pixel. This 4-bit vector is masked by the ColorKeyMASK. The result is Key2. For Key3 the procedure is similar, with the only difference being the color key vector is passed from the previous mixer. The result of masking the color key vector with the ColorKeyMask register is Key3. However, one can do selective color keying for the current pixel. A ColorKeyROP specifies whether color keying is performed on the current layer pixel or not. Inputs for this ROP are Key0,1,2, 3.

The ROP block decides if a certain operation is done on the pixel or not.

The outputs of the Select, Alpha, Invert, Key_Pass and Alpha_pass ROPs are based on Table 4.

Table 4: ROP Table for Invert/Select/Alpha/KeyPass/AlphaPass ROP’s

ROP BIT	Key3	Key2	Key1	Key0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1

Table 4: ROP Table for Invert/Select/Alpha/KeyPass/AlphaPass ROP's *…Continued*

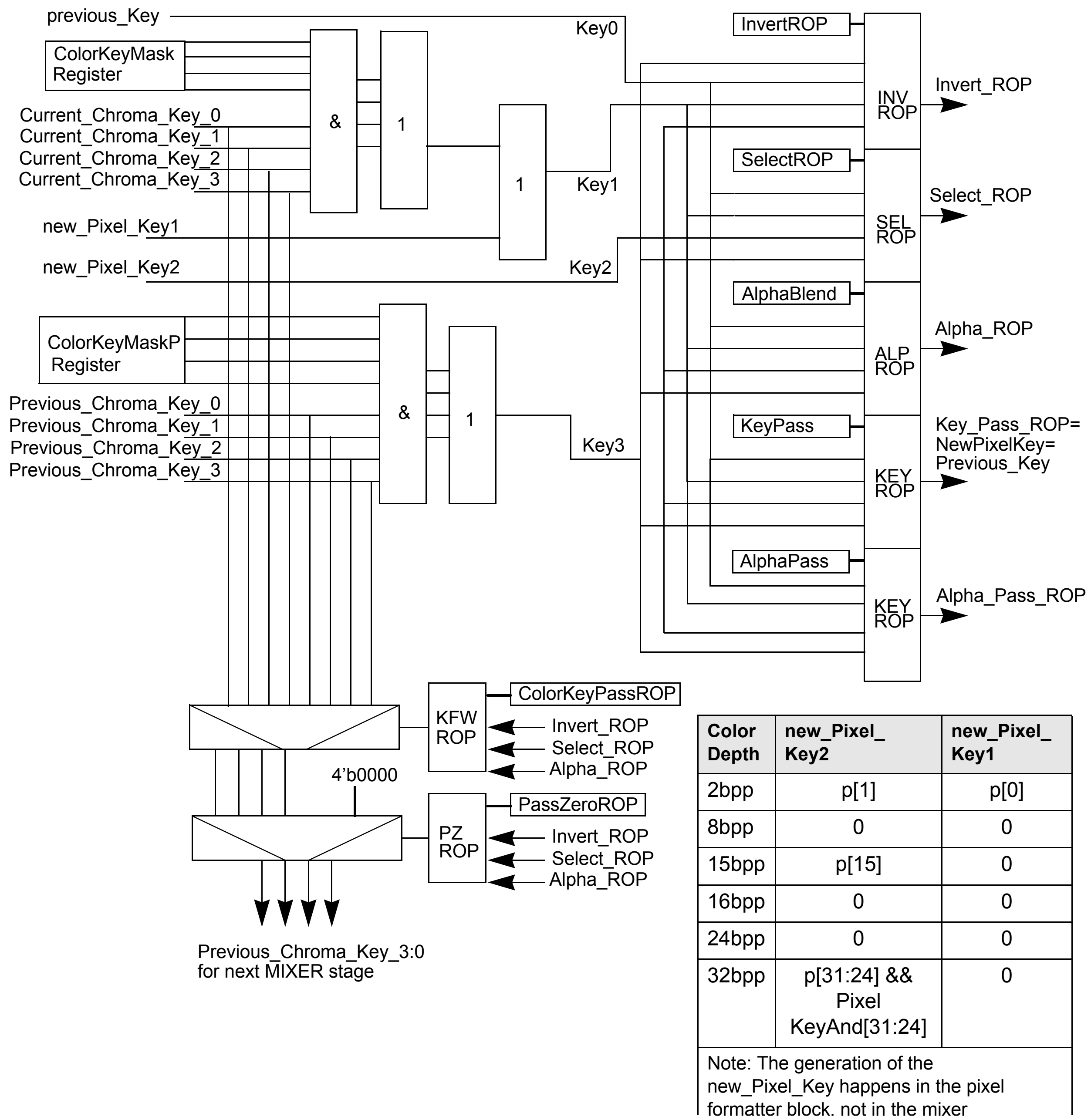
ROP BIT	Key3	Key2	Key1	Key0
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

The output of the ColorKeyPass and PassZero ROP is shown in Table 5.

Table 5: Color Key ROP Table

ROP Bit	Invert_ROP	Select_ROP	Alpha_ROP
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

The following drawing illustrates the Mixer function. The upper part shows the generation of the signals, which are used to control the actual pixel manipulation functions, shown in the lower part of the block diagram.



Color Depth	**new_Pixel_ Key2**	**new_Pixel_ Key1**
2bpp	p[1]	p[0]
8bpp	0	0
15bpp	p[15]	0
16bpp	0	0
24bpp	0	0
32bpp	p[31:24] && Pixel KeyAnd[31:24]	0

Note: The generation of the new_Pixel_Key happens in the pixel formatter block. not in the mixer

Figure 7: Mixer Block Diagram—Pixel Selection

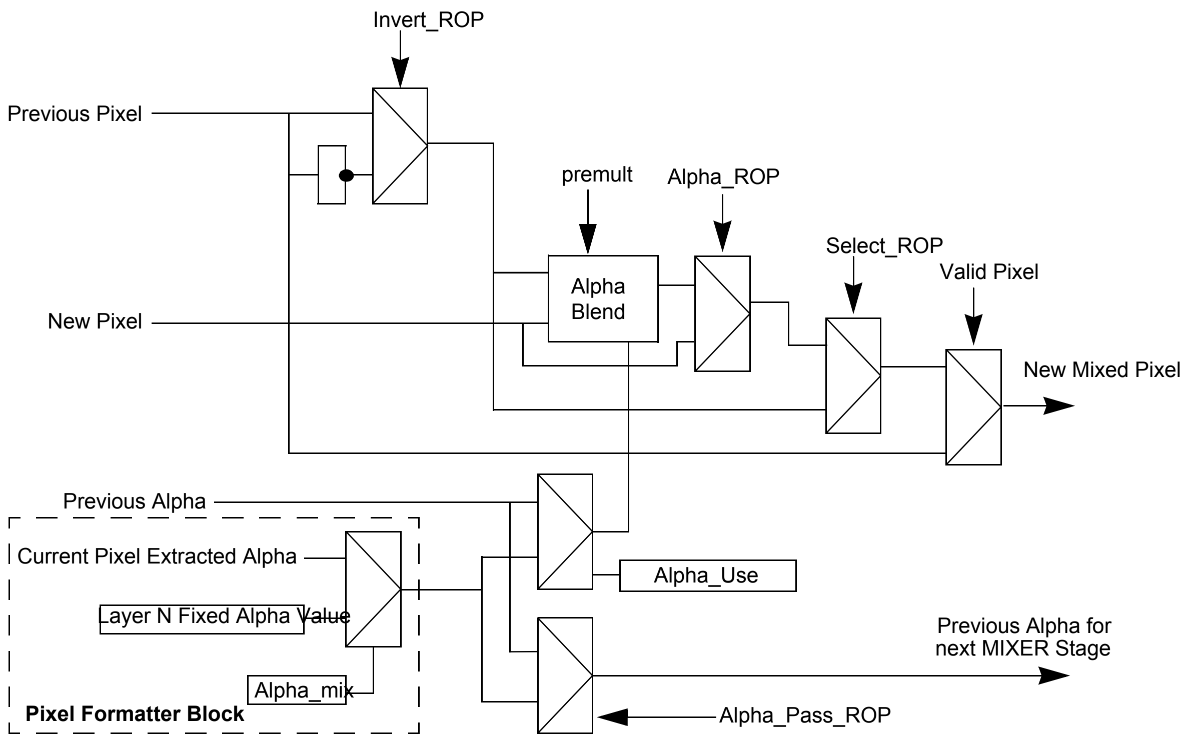


Figure 8: Mixer Block Diagram—Pixel Processing

28.6.12 Alpha-Blending

Blending allows video and graphics to be combined with either 16 or 256 levels of transparency. Blending is possible only when both current and previous layer pixels are valid. The blend value may come from a layer's alpha register or from the upper 4 or 8 bits of an incoming pixel or as a multiplication of both.

The blending is done according to the following equation:

Pixel_result = alpha x Pixel_current + (1-alpha) x Pixel_previous

Pre-multiplied pixel formats are supported. The Premult bit is set, which means the incoming pixel stream is already premultiplied with the per-pixel alpha value. The resulting alpha blend equation is as follows:

Pixel_result = Pixel_current + (1-alpha) x Pixel_previous

An additional per-component pre-multiply with a constant can be achieved by proper programming of the color space matrix. Fading of alpha values is controlled by the alpha_mix bit. If it is set, the pixel alpha gets multiplied by the fixed alpha value/256.

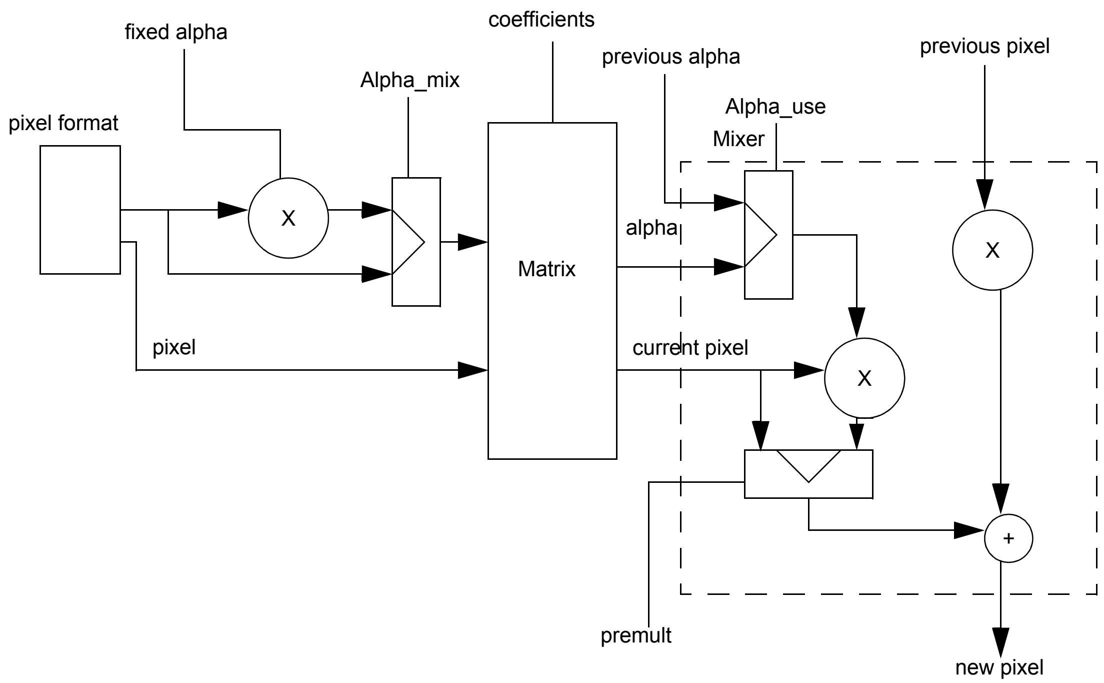


Figure 9: Alpha-Blending Block Diagram

28.6.13 Output Formatter

The output formatter arranges the pixel streams coming from the various mixers according to the selected output format. Because there is a special companion chip (PNX8510-11) to accommodate the analog back-end functions, the output interface functionality is tightly linked to the interface capabilities of the PNX8510-11. However it is still possible to connect standalone digital video encoders (SAA7128...) to the PNX8526.

Two AICPs exist in the PNX8526. Each AICP supports one physical output interface. One physical interface is primarily intended to serve as the display interface connected to a TV or monitor and is associated with the four layer AICP. The other one is used to connect to a recording device, like a VCR, and is bound to the two-layer AICP 2.

Each of the two physical AICP output interfaces can carry up to (operation mode dependent) two video streams in an interleaved manner. With this feature up to four display or record devices can be supported. For these special modes, two PNX8510-11 companion chips must be in the system. (One PNX8510-11 supports only two simultaneous video channels.)

Remark: Because each of the two physical output interfaces is driven by one and only one AICP, two general modes of operation are possible for each AICP. However in special cases where a high interface speed would be required, both interfaces can be combined and connected to the primary four-layer AICP to provide high data rates such as 1080i modes. Note that this implies the loss of the secondary AICP for record purposes.

28.6.13.1 Single Stream Mode

In single stream mode each AICP provides one video stream.

Table 6: Single Stream Mode Interface Formats

Display Modes	Mode	Interface Speed
4:4:4 RGB or YUV or YCrCb PAL/NTSC/ SECAM resolution 4:4:4 i.e. PAL: 864 pixel/line x 312.5 lines/field x 50 Hz = 13.5 MHz/component	4:4:4 Muxed Components	40.5 MHz
4:2:2 CVBS or Y/C PAL/NTSC/SECAM resolution 4:2:2 i.e. PAL: 864 pixel/line x 312.5 lines/field x 50Hz = 13.5 MHz/Y samples 7.5MHz/U samples 7.5MHz/V samples	4:2:2 Muxed components	27 MHz
4:4:4 RGB or YUV or YCrCb 2FH 864 pixel/line x 312.5 lines/field x 50 Hz x2 = 27 MHz/component	4:4:4 Muxed Components	81 MHz
4:4:4 RGB or YUV or YCrCb 480P 864 pixel/line x 625 lines/field x 50Hz = 27 MHz/component	4:4:4 Muxed Components	81 MHz
generic D1 mode; the interface clock can run up to 100 MHz, the components can have either 4:2:2 or 4:4:4 color resolution, but must be in the correct color space.	4:4:4 Muxed components/ 4:2:2 Muxed components	up to 100 MHz

28.6.13.2 Interleaving Mode

In interleaved mode, each AICP module can output two interleaved video streams over one physical interface.

Table 7: Interleaved Stream Mode Interface Formats

Display Modes	Mode	Interface Speed
Interleaved mode 1 accommodates 2 synchronous multiplexed D1 streams	2x muxed 4:2:2 single D1	54 MHz
Interleaved mode 2 accommodates 2 synchronous multiplexed video streams each running at 40.5 MHz	2x muxed 4:4:4 RGB/YUV	81 MHz

28.6.13.3 Combined Mode

This mode offers a combination of the two physical interfaces to serve high data rate needs. In this mode both interfaces are combined and connected to the primary four-layer AICP. No secondary output is available in this mode.

Table 8: Combined Mode Interface Formats

Display Modes	Mode	Interface Speed
24-bit RGB/YUV both D1 interfaces and additional pins are combined to provide high-speed direct access to video DACs	24-bit direct RGB/ YUV	up to 100 MHz
Combined, double D1 mode; the two D1 interfaces are combined to carry a single HDTV stream in 4:2:2 YUV or 4:2:2 YPrPb format primary D1: Y channel secondary D1: muxed UV or PrPb channel i.e.: 1920x1080 50 Hz interlaced 1125 pixel/line x 562.5 lines/filed x 50 Hz = 74.25 MHz/ Y samples 37.125 MHz/Cr/Pr samples 37.125 MHz/Cb/Pb samples	2 combined D1	up to 100 MHz per D1

28.6.14 VBI Insertion

VBI data preparation is a software-intensive task. The following software preparation must be done prior to its transfer over the D1 interface.

- Packaging of VBI data so that they fit into the h blanking interval (this may differ with different h blank programming values)
- Generation of the appropriate ancillary data header for D1 transfer
- Generation of a linked list in memory for the PI master

As mentioned, the communication between the D1 interface and software is handled via a linked list DMA approach. This list data structure looks as follows:

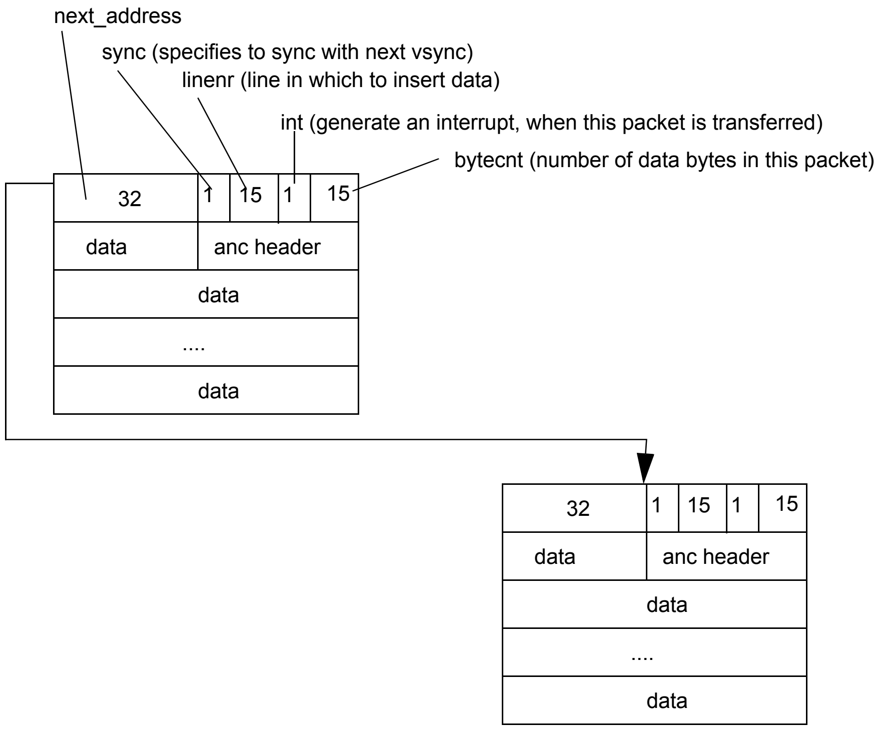

next_address
sync (specifies to sync with next vsync)
linenr (line in which to insert data)
int (generate an interrupt, when this packet is transferred)
bytecnt (number of data bytes in this packet)
32
1
15
1
15
data
anc header
data
....
data
32
1
15
1
15
data
anc header
data
....
data

Figure 10: VBI Data Structure

The ancillary data header matches the scheme used in the PNX8510-11 analog companion chip.

- Next address specifies the next physical memory address, where the next data packet is located.
- Sync with next vsync specifies whether this data packet has to be transferred immediately or after the next vsync.
- Linenr specifies the line number in which this data packet has to be inserted.
- Bytecnt specifies the number of data bytes including the ANC header associated with this data packet. This value is used to request the appropriate number of bytes from memory.

After the linked list is set up in memory, the internal PI master is initiated by setting the first start address for the linked list, and a write to the appropriate enable bit. After that, it will fetch the first data packet header to determine the byte count that it has to fetch, plus the address for the next packet in the linked list. The additional information about synchronizing and linenr are stored in the appropriate internal registers.

After that, it will request the number of bytes needed from the pi interface, plus the header for the next packet in the linked list. After the data have arrived, it will wait until the appropriate line number occurs and the output formatter signals that it can receive VBI data. If so, the data get transferred to the output formatter and inserted into the D1 data stream. This is handled via a simple request acknowledge handshake. In parallel, whenever there is enough space in the VBI FIFO, the next data packet will be requested until the VBI engine reaches a packet which either has a next_address equal to 32'h00000000 or the sync bit in the packet header tells it to wait until the next vsync comes in.

By specifying the vsync bit or simply a linenr, which the current field has already passed, the insertion of the data packets is synchronized with the fields syncs. Quasi empty fields can be achieved by specifying empty packets (bytecnt = 0) and the setting of the vsync bit in the data header.

In addition to normal VBI data, the PNX8510-11 provides a mechanism to reprogram its video encoders via special ANC packets. With this feature it should be possible to change the encoder programming on a deterministic field-by-field basis. The reprogramming is needed for teletext data transfer implementations. One does a setup of the encoder to transfer teletext data in lines n through m. The reprogramming ANC data packets have to be sent at least in line n-2 to allow the reprogramming to become active. The first teletext data packet has to arrive at line n-1 to give the whole VBI engine inside the PNX8510-11 at least one line to decode the information and to put it into the appropriate places (mostly FIFOs and registers).

At line n the encoder will request teletext data, which then gets transferred to its interface out of a PNX8510-11 internal teletext buffer. In the meantime, the VBI engine inside the AICP will keep sending teletext data to the PNX8510-11, which continuously fills and drains data of the teletext buffer.

28.6.15 Hardware Software Interface

28.6.15.1 What the Software Needs

To implement a dynamic and flexible hardware software interface, the software needs to know some status information, provided either by readable programming registers or via interrupts. The required information is as follows:

- Layer done interrupt (plus indicating used buffer A or B)
- FIFO underflow detection and interrupt (bandwidth error)
- Interrupt, if data are fetched twice or more from the same specified buffer location (buffer reprogramming occurred too late -> underrun condition)
- Timestamps for each outgoing video field/frame

The "data fetched twice from the same location" interrupt is needed by the software to identify when a reprogramming of a buffer occurred too late for a layer. This, plus the timestamping information, can be used to adjust the scheduling software accordingly so that a buffer reprogramming doesn't occur too late.

Software is able to reprogram some registers dynamically without caring about the current status/position of a layer. These registers are as follows:

- source address a
- source address b
- byte count a

- byte count b
- line pitch a
- line pitch b
- layer width/height
- layer startx/y
- layer start fetch
- layer pixel format
- pixel processing

For a flexible VBI insertion, a VBI done interrupt is needed. This indicates when the VBI DMA hardware reaches the end of the provided linked list in memory.

28.6.15.2 Hardware Implementation

The dynamically changeable layer registers can be reprogrammed at all times in the normal memory mapped IO space. A layer enable/re-enable (writing a 1 to the layer enable register) will upload those registers into a shadow area. After that, reprogramming is permitted again. The software shall wait until the upload bit is reset (layer status/control register) before the next upload is initiated.

The involved internal AICP blocks then take the programming registers from the shadow area into their work area. Once all uploaded registers are transferred into the work register area, the upload bit is reset and the next upload procedure can begin. If any other layer register of the AICP is reprogrammed, a layer reset should be initiated. If registers from the global register space are altered, a global AICP reset should be done.

Some other probably useful information/interrupts are provided:

- Two programmable line interrupts
- Instant readable x/y position
- A per-layer end of layer interrupt (same as end of buffer interrupt)

Additionally, software needs the following:

- FIFO underflow interrupt
- Interrupt, if data is fetched twice from a buffer by the same layer
- Timestamping in the GPIO block (vsync is routed to this block as timestamp signal)
- End of VBI list interrupt

A programming value to specify the line to start fetching is needed to provide the capability to reprogram the buffer-specific information for a layer as late as possible. The layer starts as a default at screen timing generator position (0,0) fetching. This programming value moves the fetching point to a specific line.

28.7 Dynamic Reprogramming

After reprogramming the AICP, a global reset of the whole module should be done. The sequence should be as follows:

- Reset screen timing generator enable
- Switch screen timing generator back on
- Enable one layer after another

Register shadowing for some registers allows a dynamic reprogramming without reset. These registers are as follows:

- layer_s_adra
- layer_s_adrb
- layer_byte_count_a
- layer_byte_count_b
- layer_pitch_a
- layer_pitch_b
- width
- height
- startx
- starty

A re-enable of the layer (writing a 1 to layer_enable) will move those registers into an upload area. The programming values are then taken from this area whenever it is safe for the targeted blocks. For DMA reprogramming, this is at the end of fetching data for the current field/frame. For the layer control (startx, starty, width, height), this is at the end of the active layer area. That implies that each layer needs some time between two active areas (fields/frames).

28.7.1 Programming Restrictions

When programming the AICP, the following programming restrictions should be observed:

- layer width <= htotal
- layer height <= vtotal
- only 8-bit alpha allowed for format (32 bpp, YUV, 888)
- only 4-bit alpha allowed for format (16 bpp, YUV/RGB 534)
- hsyncs/e != 0, 1 < htotal
- hblnks/e != 0, 1 < htotal

- vsyncs/e != 0 < vtotal
- vblnks/e != 0 < vtotal

28.8 Register Summary and Descriptions

The PNX8526 has two AICP modules. The registers for AICP 1 begin at 0x10 E000. The registers for AICP 2 begin at 0x10 F000.

AICP 1 supports four layers, while AICP 2 supports only two. Refer to the following tables for the locations of these registers.

All registers are assumed to be reset by the system-wide reset signal to the value 0. Readback of unused registers is 0.

Table 9: AICP Module Register Summary

Offset	Name	Description
AICP 1 Registers		
0x10 E000	TOTAL	Total number of horizontal and vertical pixels for the display.
0x10 E004	HBLANK	Location of horizontal blanking start and end.
0x10 E008	VBLANK	Location of vertical blanking start and end.
0x10 E00C	HSYNC	Location of horizontal sync start and end.
0x10 E010	VSYNC	Location of vertical sync start and end.
0x10 E014	VINTERRUPT	Vertical line interrupt setting.
0x10 E018	FEATURES	Feature control for Gamma LUTs and number of layers.
0x10 E01C	STATUS	Status of current field, vertical sync and blanking, and XY position.
0x10 E020	CONTROL	Main display controls.
0x10 E028	INTLCTRL1	Control horizontal offset for VSYNC start.
0x10 E02C	INTLCTRL2	Control horizontal offset for VSYNC end.
0x10 E030	VBI SRC Address	Vertical Blanking Interval data source address
0x10 E034	VBI_CTRL	Control VBI data fetch engine.
0x10 E038	VBI_SENT_OFFSET	Add to the linecnt value in the packet identifier.
0x10 E03C	OUT_CTRL	Output control.
0x10 E040	OUTPUT GAMMA LUT CTRL	Output gamma LUT. (AICP1 ONLY)
0x10 E044	GAMMA LUT CTRL	Output gamma LUT control. (AICP1 ONLY)
0x10 E048—E04C	Reserved	
0x10 E050	Signature1	Signature
0x10 E054	Signature2	
0x10 E058	Signature3	
0x10 E05C—E0FC	Reserved	
AICP 1: Layer 1		
0x10 E100	Layer Source Address A	Layer Source Address A

Table 9: AICP Module Register Summary *…Continued*

Offset	Name	Description
0x10 E104	Layer Pitch A	Layer Pitch A
0x10 E108	Layer Source Width A	Layer Source Width A
0x10 E10C	Layer Source Address B	Layer Source Address B
0x10 E110	Layer Pitch B	Layer Pitch B
0x10 E114	Layer Source Width B	Layer Source Width B
0x10 E118—E12C	Reserved	
0x10 E130	Layer Start	Layer Start
0x10 E134	Layer Size	Layer Size
0x10 E138	Layer Pixel Format	Layer Pixel Format
0x10 E13C	Layer Pixel Processing	Layer Pixel Processing
0x10 E140	Layer Status/Control	Layer Status/Control
0x10 E144	LUT Programming	LUT Programming
0x10 E148	LUT Addressing	LUT Addressing
0x10 E14C	Mixer Pixel Key AND Register	Mixer Pixel Key AND Register
0x10 E150	Color Key1 AND Mask	Color Key1 AND Mask
0x10 E154	Color Key Up1	Color Key Up1
0x10 E158	Color Key Low1	Color Key Low1
0x10 E15C	Color Key Replace1	Color Key Replace1
0x10 E160	Color Key2 AND Mask	Color Key2 AND Mask
0x10 E164	Color Key Up2	Color Key Up2
0x10 E168	Color Key Low2	Color Key Low2
0x10 E16C	Color Key Replace2	Color Key Replace2
0x10 E170	Color Key3 AND Mask	Color Key3 AND Mask
0x10 E174	Color Key Up3	Color Key Up3
0x10 E178	Color Key Low3	Color Key Low3
0x10 E17C	Color Key Replace3	Color Key Replace3
0x10 E180	Color Key4 AND Mask	Color Key4 AND Mask
0x10 E184	Color Key Up4	Color Key Up4
0x10 E188	Color Key Low4	Color Key Low4
0x10 E18C	Color Key Replace4	Color Key Replace4
0x10 E190	Color Key Mask/ROP	Color Key Mask/ROP
0x10 E194	Pixel Invert/Select ROP	Pixel Invert/Select ROP
0x10 E198	Alpha Blend/Key Pass	Alpha Blend/Key Pass
0x10 E19C	Alpha Pass	Alpha Pass
0x10 E1A0	Color Key ROPs 1/2	Color Key ROPs 1/2
0x10 E1A4	Color Key ROPs	Color Key ROPs
0x10 E1A8	Matrix Coefficients1	Matrix Coefficients1
0x10 E1AC	Matrix Coefficients2	Matrix Coefficients2

Table 9: AICP Module Register Summary *…Continued*

Offset	Name	Description
0x10 E1B0	Matrix Coefficients3	Matrix Coefficients3
0x10 E1B4	Matrix Coefficients4	Matrix Coefficients4
0x10 E1B8	Matrix Coefficients5	Matrix Coefficients5
0x10 E1BC	DC-Offsets CSM Output	DC-Offsets CSM Output
0x10 E1C0	Layer Background Color	Layer Background Color
0x10 E1C4	DC-Offsets CSM Input	DC-Offsets CSM Input
0x10 E1C8	Start Fetch	Start Fetch
0x10 E1CC—E1F8	Reserved	
AICP 1: Layer 2		
0x10 E200—E2F8	...	The Layer 2 registers begin at 0x10 E200. They are identical to the Layer 1 registers, described above.
AICP 1: Layer 3		
0x10 E300—E3F8	...	The Layer 3 registers begin at 0x10 E300. They are identical to the Layer 1 registers, described above.
AICP 1: Layer 4		
0x10 E400-E4F8	...	The Layer 4 registers begin at 0x10 E400. They are identical to the Layer 1 registers, described above.
0x10 E4FC-EFDC	Reserved	
0x10 EFE0	Interrupt Status AICP1	Interrupt Status for AICP 1
0x10 EFE4	Interrupt Enable AICP1	Interrupt Enable for AICP 1
0x10 EFE8	Interrupt Clear AICP1	Interrupt Clear for AICP 1
0x10 EFEC	Interrupt Set AICP1	Interrupt Set for AICP 1
0x10 EFD0- EFF0	Reserved	
0x10 EFF4	Powerdown	Powerdown
0x10 EFFC	Module ID	AICP 1 Module ID is 0x0111 0000
AICP 2 Registers		
0x10 F000-F03C	...	These AICP 2 registers are identical to the AICP 1 registers.
0x10 F040-F04C	Reserved	Note: Output gamma LUT control is not available in AICP 2.
0x10 F050-F0FC	...	These AICP 2 registers are identical to the AICP 1 registers.
0x10 F100-F1F8	...	The Layer 1 registers begin at 0x10 F100. They are identical to the AICP 1: Layer 1 registers, described above.
0x10 F200-F2F8	...	The Layer 2 registers begin at 0x10 F200. They are identical to the AICP 1: Layer 1 registers, described above.
0x10 F300-FFD8	Reserved	Note that AICP 2 supports only layers 1 and 2.
0x10 FFE0-FFFC	...	These AICP 2 registers are identical to the AICP 1 registers. AICP 2 Module ID is 0x0118 0000

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Screen Timing Generator Registers				
Offset 0x10 E000			*TOTAL*	
31:28		-	Unused	
27:16	R/W	0	HTOTAL	Horizontal Total sets the number of horizontal pixels for the display. Total # of pixels per line = HTOTAL+1.
15:12		-	Unused	
11:0	R/W	0	VTOTAL	Vertical Total sets the number of vertical pixels for the display. Total # of lines per field = VTOTAL +1.
Offset 0x10 E004			*HBLANK*	
31:28		-	Unused	
27:16	R/W	0	HBLANKS	Horizontal Blank Start sets the pixel location where horizontal blanking starts.
15:12		-	Unused	
11:0	R/W	0	HBLNKE	Horizontal Blank End sets the pixel location where horizontal blanking ends.
Offset 0x10 E008			*VBLANK*	
31:28		-	Unused	
27:16	R/W	0	VBLANKS	Vertical Blank Start sets the pixel location where vertical blanking starts.
15:12		-	Unused	
11:0	R/W	0	VBLANKE	Vertical Blank End sets the pixel location where vertical blanking ends.
Offset 0x10 E00C			*HSYNC*	
31:28		-	Unused	
27:16	R/W	0	HSYNCS	Horizontal Sync Start sets the pixel location where horizontal sync starts.
15:12		-	Unused	
11:0	R/W	0	HSYNCE	Horizontal Sync End sets the pixel location where horizontal sync ends.
Offset 0x10 E010			*VSYNC*	
31:28		-	Unused	
27:16	R/W	0	VSYNCS	Vertical Sync Start sets pixel location where Vertical sync starts.
15:12		-	Unused	
11:0	R/W	0	VSYNCE	Vertical Sync End sets pixel location where Vertical sync ends.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Control and Interrupt Registers				
Offset 0x10 E014			*VINTERRUPT*	
31:28		-	Unused	
27:16	R/W	0	VLINTA	Vertical Line Interrupt A sets a vertical line number where an interrupt will be generated when the scanline matches this value. The interrupt is monitored by the Event Monitor (EVM).
15:12		-	Unused	
11:0	R/W	0	VLINTB	Vertical Line Interrupt B sets a vertical line number where an interrupt will be generated when the scanline matches this value. The interrupt is monitored by the Event Monitor (EVM).
Offset 0x10 E018			*FEATURES*	
31:5		-	Unused	
4:3	R	0x1	NOGAMMALUTS	Number of gamma LUTS binary coded for the output channels (The first slice of the AICP out goes through the gamma LUT.): Value for AICP1 is 0x1.
2:0	R	0x4	NOLAYERS	Number of layers in this AICP: Value for AICP1 is 0x4.
Note: For the interrupt to occur, it must be enabled in the STG control register.				
Offset 0x10 E01C			*STATUS*	
31		-	Unused	
30	R	0	O_E_STAT	Odd/Even flag status (interlaced mode) 0 = First field (odd) 1 = Second field (even)
29	R	0	VSYNCSTAT	Read back current status of VSYNC 0 = Not in Vertical Sync area 1 = Sweep is in Vertical Sync.
28	R	0	VBLNKSTAT	Read back current status of VBLANK 0 = Not in Vertical Blanking area 1 = Sweep is in Vertical Blanking.
27:16	R	0	YCNT	Read back line current count
15:12		-	Unused	
11:0	R	0	XCNT	Read back position in current line
Offset 0x10 E020			*CONTROL*	
31:30		-	Unused	
29	R/W	0	Interlaced	Interlaced mode bit 0 = Non-interlaced mode; VTotal=frame height. 1 = Interlaced mode Field height = VTotal+1 for odd fields. Field height = VTotal for even fields.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
28	R/W	0	BlankPol	BLANK Polarity 0 = Positive blank 1 = Negative blank
27		-	Unused	
26	R/W	0	HSYNCPol	HSYNC Polarity 0 = Positive going 1 = Negative going
25		-	Unused	
24	R/W	0	VSYNCPol	VSYNC Polarity 0 = Positive going 1 = Negative going
23:21		-	Unused	
20	R/W	0	BlankCtl	Blank Control allows either normal blanking or forces blanking to occur immediately. 0 = Force Blank 1 = Normal Blank
19		-	Unused	
18	R/W	0	HSYNCCtl	HSYNC Control enables or disables the horizontal sync output of the chip. 0 = Disable 1 = Enable
17		-	Unused	
16	NI	0	VSYNCCtl	VSYNC Control enables or disables vertical sync output of the chip. 0 = Disable 1 = Enable
15:3		-	Unused	
2	R/W	0x1	TRIGGER_POL	External trigger polarity 1 = Positive edge (default) 0 = Negative edge
1	R/W	0	MASTER	STG master/slave/operation 0 = Master mode 1 = Slave mode
0	R/W	0	TGRST	Timing generator reset 0 = Disable 1 = Enable Disable will reset all layer_enable bits (global AICP reset).
Offset 0x10 E028			*INTLCTRL1*	
31:28		-	Unused	
27:16	R/W	0	INT_START_E	Horizontal offset for VSYNC start even field (interlaced mode only) Vsync appears at INT_START_E + 1.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:12		-	Unused	
11:0	R/W	0	INT_START_O	Horizontal offset for VSYNC start odd field (interlaced mode only) Vsync appears at INT_START_O + 1.
Offset 0x10 E02C			*INTLCTRL2*	
31:28		-	Unused	
27:16	R/W		INT_END_E	Horizontal offset for VSYNC end even field (interlaced mode only)
15:12		-	Unused	
11:0	R/W	0	INT_END_O	Horizontal offset for VSYNC end odd field (interlaced mode only)
Offset 0x10 E030			*VBI SRC Address*	
31:26		-	Unused	
25:0	R/W	0	VBI_SRC_ADDR	VBI data source address
Offset 0x10 E034			*VBI_CTRL*	
31:1		-	Unused	
0	R/W	0	VBI_EN	Enable VBI data fetch engine.
Offset 0x10 E038			*VBI_SENT_OFFSET*	
31:12		-	Unused	
11:0	R/W	0	VBI_SENT_OFFSET	This programming value specifies the number of lines to add to the linecnt value in the packet identifier.
Offset 0x10 E03C			*OUT_CTRL*	
31:20		-	Unused	
19	R/W	1	10Bit	1 = ouput formatter works in 10 bit per color component mode 0 = ouput formatter works in 8 bit per color component mode
18	R/W	0	qualifier	1 = slice qualifier is put out 0 = hsync is put out
17:16	R/W	0	outmode	00 = output interface runs is two-d1 mode 01 = output interface runs in double-d1 mode 10 = output interface operates in 24bit parallel mode 11 = unused
15	R/W	0	parallel_mode	This bit controls the sync delay compensation. 1 = syncs are delayed (needed for 24bit parallel output mode) 0 = no additional sync delay
14	R/W	1	TC_outS2	1 = invert the MSB of the second D1 slice 0 = leave second D1 slice untouched
13	R/W	1	TC_outS1	1 = invert the MSB of the first D1 slice 0 = leave first D1 slice untouched

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
12	R/W	0	oversample	This bit enables the output statemachine for oversampling. This bit should be 0 for interleaved output modes. If one (only supported for a single d1 stream, either 444 or 422 or 444x) the output clock should be 2x the streaming clock i.e. 422 SD mode: streaming clock 27Mhz, output clock 54Mhz results in a 2x oversampling of the datastream. 1 = oversampling enabled 0 = no oversampling
11	R/W	0	RGBX	1 = RGBX/YUVX mode 0 = RGB/YUV mode
10	R/W	1	D1_MODE	1 = 4:2:2 D1 mode 0 = 4:4:4 D1 mode
9	R/W	0	INTERLEAVE	1 = Interleaved mode (2 D1 streams are interleaved clk_icp_interl must be set to 2x clk_icp_mux) 0 = non interleaved mode
8	R/W	0	SLICE_SEL	Interleaved mode: determines the position of the two slices in the stream. Non-interleaved mode: determines which of the selected slices goes in the stream. 0 = Mux_sel_1 1 = Mux_sel_2
7:6			Unused	
5:4	R/W	0	MUX_SEL_2	Tap off selection for second slice 0 = tap off after first mixing stage 1 = tap off after second mixing stage 2 = tap off after third mixing stage(AICP1 ONLY) 3 = tap off after fourth mixing stage (AICP1 ONLY)
3:2			Unused	
1:0	R/W	0	MUX_SEL_1	Tap off selection for first slice 0 = tap off after first mixing stage 1 = tap off after second mixing stage 2 = tap off after third mixing stage(AICP1 ONLY) 3 = tap off after fourth mixing stage (AICP1 ONLY)
Offset 0x10 E040			***OUTPUT GAMMA LUT CTRL (AICP1 ONLY)***	
31:30			Unused	
29:20	R/W	0	LUT_RED	Gamma LUT data port red This register reads alyways 0 if the “Host_Enable” bit is 0
19:10	R/W	0	LUT_GREEN	Gamma LUT data port green This register reads alyways 0 if the “Host_Enable” bit is 0
9:0	R/W	0	LUT_BLUE	Gamma LUT data port blue This register reads alyways 0 if the “Host_Enable” bit is 0
Offset 0x10 E044			***GAMMA LUT CTRL (AICP1 ONLY)***	

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31	R/W	0	LUT_ENABLE	Gamma LUT enable bit 1 = Active 0 = Bypass
30	R/W	0	HOST_ENABLE	This bit enables Host read/write access to the LUT: 1 = Host access enabled. 0 = Host access disabled, no access possible.
29:25		-	Unused	
24:16	R/W	0	LUT_ADDR	Gamma LUT address, no auto increment supported.
15:1		-	Unused	
0	R/W	0	LUT_RW	Gamma LUT read/write 1 = Write access to gamma LUT 0 = Read access to gamma LUT
Offset 0x10 E050			*Signature1*	
31:16	R	0	middle signature	Middle path signature
15:0	R	0	lower signature	Lower path signature
Offset 0x10 E054			*Signature2*	
31:16	R	0	alpha signature	Alpha path signature
15:0	R	0	upper signature	Upper path signature
Offset 0x10 E058			*Signature3*	
31:16	R	0	misc signature	Other signature
15:9		-	Unused	
8	R	0	sig_done	Signature done
7:6		-	Unused	
5:4	R/W	0	sig_select	Signature select
3:1		-	Unused	
0	R/W	0	sig_enable	Signature enable

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Layer & Mixer Registers				
The structure of each layer function block is identical. The register for a function such as Source Address in Layer 1, has the same structure as the corresponding register in Layer 2 to Layer 4. Layer one starts at offset 0x100 from the AICP base address Layer two starts at offset 0x200 from the AICP base address Layer three (AICP1 only) starts at offset 0x300 from the AICP base address Layer four (AICP1 only) starts at offset 0x400 from the AICP base address				
Offset 0x10 E100			***Layer Source Address A***	
31:26		-	Unused	
25:0	R/W	0	Layer N Source Address A	Layer N Source Data Start Address A in bytes. This sets starting address A for data transfers from the linear Frame Buffer memory to Layer N. **Note:** It should be aligned on a 128-byte boundary for memory performance reasons. It has to be 8-byte aligned.
Offset 0x10 E104			***Layer Pitch A***	
31:23		-	Unused	
22:0	R/W	0	Layer N Pitch A	Layer N Source Data Pitch B in bytes. This sets pitch A for data transfers from the linear Frame Buffer memory to Layer N. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E108			***Layer Source Width A***	
31:23		-	Unused	
22:0	R/W	0	Layer N Source Width A	Layer N source width in bytes A. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E10C			***Layer Source Address B***	
31	W	0	Source address upload	Writing a 1 to this bit will initiate an address upload of layer source address a and b into the register shadow area.
30:26		-	Unused	
25:0	R/W	0	Layer N Source Address B	Layer N Source Data Start Address B in bytes. This sets starting address B for data transfers from the linear Frame Buffer memory to Layer N. **Note:** It should be aligned on a 128-byte boundary. It has to be 8-byte aligned.
Offset 0x10 E110			***Layer Pitch B***	
31:23		-	Unused	
22:0	R/W	0	Layer N Pitch B	Layer N Source Data Pitch B in bytes sets pitch B for data transfers from the linear Frame Buffer memory to Layer N. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E114			***Layer Source Width B***	
31:23		-	Unused	
22:0	R/W	0	Layer Source Width B	Layer N source width in bytes B. The value has to be rounded up to the next 64-bit word.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 E130			*Layer Start*	
31	R/W	0	Fine	Fine positioning enable for interlaced modes (layer size needs to be set to odd + even number of lines). This feature should only be used for graphic contents as it is changing the temporal relationship between subsequent fields. If fine positioning is enabled, the entire layer must be completely positioned within the visible screen area. Cropping has to be performed by setting the layer dimensions and position to the appropriate values.
30:29		-	Unused	
28:16	R/W	0	LayerNStartX	Layer N Start x position (from zero at left edge) in pixels.
15:13		-	Unused	
12:0	R/W	0	LayerNStartY	Layer N Start y position (from zero at top) in lines.
Offset 0x10 E134			*Layer Size*	
31:28		-	Unused	
27:16	R/W	0	LayerNHeight	Layer N height in lines.
15:12		-	Unused	
11:0	R/W	0	LayerNWidth	Layer N width, in pixels.
Offset 0x10 E138			*Layer Pixel Format*	
31	R/W	0x1	LayerExpansionParameter	Controls how data that is not 24-bit data is converted to 24-bit data when necessary. 0 =Use source data left-aligned into msbits and fill lsbits with 0s. 1 = Use source data left-aligned into msbits and replicate highest msb data into remaining lower order bits.
30	R/W	0	Indexed	0 = No indexed colors, CLUT is shut off. 1 = Indexed colors
29:24	R/W	0	Offset	6-bit offset value specifies how many bits to offset in the first-fetched Long Word at the beginning of a line.
23:16	R/W	0	LayerNFixedAlphaValue	Alpha blend value to be applied in Mixer. This 8-bit field provides 256 levels of fixed alpha-blending. The AlphaSelect ROP must be set appropriately to use this feature.
15:14	R/W	0	Alpha	00 = No Alpha information in pixel data 01 = 4-bit Alpha information in pixel data 10 = 8-bit Alpha information in pixel data 11 = Reserved
13:8	R/W	0	Bits Per Pixel	The total number of bits per pixel needs to be specified here. This number has to include potential alpha bits or unused bits for unpacked formats.
7	R/W	0	Premult	If this bit is set, the incoming pixels are premultiplied with alpha. That disables the new x alpha multiplication in the mixer stage if alpha-blending is enabled.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
6:4	R/W	0	Channel Swap/UV_Swap	Ext/Int channel relationship (assignment of A/Y/U/V or A/R/G/B to alpha/upper/middle/lower AICP layer processing channel). 000 = (A)UML - yvyu/vyuy 001 = (A)ULM - yuyv/uyvy 010 = (A)MUL 011 = (A)MLU 100 = (A)LUM 101 = (A)LMU 110 = M(A)UL 111 = L(A)UM Swaps the UV pixel information in the pixel formatter for 4:2:2 formats to provide YVYU and YUYV formats.
3:0	R/W	0	Mux Order	Non-Indexed Mode determines how the bits within a pixel are assigned to the rgb/yuv channels. 0000 = mode_332 0001 = mode_444 0010 = mode_534 0011 = mode_555 0100 = mode_565 0101 = mode_633 0110 = mode_888 0111 = mode_vyuy 1110 = mode_yvyu Indexed Mode specifies the actual index size without alpha i.e., 2bpp indexed = 4’b0010 Note: In indexed mode the following formula should be satisfied: BPP >= Mux_Order + Alpha x 4.
Offset 0x10 E13C			***Layer Pixel Processing***	
31	R/W	0	pu2c	Controls the upper previous channel format input to the mixer. 0 = Data left untouched. 1 = Data conversion two’s complement <-> binary offset
30	R/W	0	pm2c	Controls the middle previous channel format input to the mixer. 0 = Data left untouched. 1 = Data conversion two’s complement <-> binary offset
29	R/W	0	pl2c	Controls the lower previous channel format input to the mixer. 0 = Data left untouched. 1 = Data conversion two’s complement <-> binary offset
28	R/W	0	cu2c	Controls the upper current channel format input to the mixer. 0 = Data left untouched. 1 = Data conversion two’s complement <-> binary offset
27	R/W	0	cm2c	Controls the middle current channel format input to the mixer. 0 = Data left untouched. 1 = Data conversion two’s complement <-> binary offset
26	R/W	0	cl2c	Controls the lower current channel format input to the mixer. 0 = Data left untouched. 1 = Data conversion two’s complement <-> binary offset

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
25	R/W	0	ou2c	Controls the upper channel format output of the mixer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset
24	R/W	0	om2c	Controls the middle channel format output of the mixer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset
23	R/W	0	ol2c	Controls the lower channel format output of the mixer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset
22:20	R/W	0	endianness swap	This specifies the used endianness swapping: 000 = swapping is controlled by bits per pixel and global endianness signal. The incoming 32 bits are interpreted as byte order "4321". Swapping will look like: 100 = No swapping 101 = 3412 110 = 2143 111 = 1234
19	R/W	0	Alpha_index	1 = Take incoming pixel value and put it into the alpha channel. 0 = No alpha indexing, take a separate value according to pixel format specification.
18	R/W	0	Alpha_mix	Enables alpha scaling. 0 = Switched off 1 = Per pixel alpha is multiplied with (fixed_alpha)/256.
17		-	Unused	
16	R/W	0	Alpha_use	Controls which alpha value is used for blending in the layer mixer stage 1 = Use previous alpha 0 = Use alpha of current layer
15:13		-	Unused	
12	R/W	0	A2C	Controls the alpha channel format within a layer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset
11	R/W	0	U2C	Controls the upper data channel format within a layer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset
10	R/W	0	M2C	Controls the middle data channel format within a layer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset
9	R/W	0	L2C	Controls the lower data channel format within a layer. 0 = Data left untouched. 1 = Data conversion two's complement <-> binary offset

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
8	R/W	0	LUT_Enable	This bit enables the LUT. 0 = LUT is bypassing pixel. 1 = Use output of the LUT.
7:6		-	Unused	
5	R/W	0	Upsample	Up-sample filter enable 0 = Bypass 1 = Up-sample filter is enabled. Works on two's complement 422 co-sited data.
4	R/W	0	422_444_en	This bit works in conjunction with the shift_en bit. If it is enabled, it converts 422 interspersed data into 444 data. Shift_en has to be switched on. 0 = Bypass 1 = Enable
3	R/W	0	Shift_en	This bit enables a filter to convert 422 interspersed data into 422 co-sited data. 0 = Bypass 1 = Enable
2:1		-	Unused	
0	R/W	0	Bandlimit	This bit specifies if the band limiting filter and subsampler for 444 to 422 conversion is switched on. 0 = No band limiting/subsampling applied in the data path. 1 = Apply band limiting/subsampling.
Offset 0x10 E140			*Layer Status/Control*	
31:27		-	Unused	
26	R	0	Layer_FUSF_Status	Layer FIFO Underflow Status. Read 0 = No Underflow Read 1 = Underflow took place Write 1 to clear Underflow Status.
25:12		-	Unused	
11	R	NI	Address upload	This bit indicates if a fetch start address upload is currently in progress: 1 = Upload is in progress, do not reprogram source addr a or b. 0 = Upload is done, reprogramming is safe.
10	R/W	0	Buffer toggle	This bit controls the dma buffer mode: 1 = Always toggle between buffer A and B (A=odd field, B=even field). 0 = No buffer toggle, always fetch from buffer spec A.
9	R	NI	Layer upload	This bit indicates if the register upload into the shadow area is still in progress. 1 = Upload is in progress. 0 = Upload done.
8	R	0	Layer_recover	0 = Layer doesn't recover after detecting a FIFO underflow. 1 = Layer recovers after FIFO underflow.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7:6		-	Unused	
5	R	1	DownFilter	0 = Layer does not contain an 4:4:4 to 4:2:2 down filter. 1 = Layer contains an 4:4:4 to 4:2:2 down filter.
4	R	1	UpsampleFilter	0 = Layer does not contain an 4:2:2 to 4:4:4 up-sampling filter. 1 = Layer contains an 4:2:2 to 4:4:4 up-sampling filter.
3	R	1	CSC	0 = Layer does not contain a color space matrix. 1 = Layer contains a color space matrix.
2	R	1	AlphaLUT	0 = Layer does not contain the Alpha LUT. 1 = Layer contains the Alpha LUT.
1	R	1	ComponentLUT	0 = Layer does not contain the RGB/YUV component LUTs. 1 = Layer contains the RGB/YUV component LUTs.
0	R/W	0	LayerN_Enable	0 = Disable layer N 1 = Enable layer N This register reads always 0 if the screen timing generator is not enabled.
Offset 0x10 E144			*LUT Programming*	
31:24	R/W	0	Alpha	Alpha value for LUT programming This register reads alyways 0 if the “Host_Enable” bit is 0
23:16	R/W	0	Red	Red value for LUT programming This register reads alyways 0 if the “Host_Enable” bit is 0
15:8	R/W	0	Green	Green value for LUT programming This register reads alyways 0 if the “Host_Enable” bit is 0
7:0	R/W	0	Blue	Blue value for LUT programming This register reads alyways 0 if the “Host_Enable” bit is 0
Offset 0x10 E148			*LUT Addressing*	
31:24	R/W	0	LUTAddress	Address register for LUT programming, no auto-increment is supported.
23:9		-	Unused	
8	R/W	0	Host_Enable	This enables read/write access by the host: 1 = Host access enabled. 0 = Host access disabled.
7:1		-	Unused	
0	R/W	0	Read/Write	0 = Read 1 = Write
Offset 0x10 E14C			*Mixer Pixel Key AND Register*	
31:24	R/W	0xFF	PixelKeyAND	The bits 31:24 in 32 bpp mode are ANDed with this mask (input for KEY2).
23:0		-	Unused	
Offset 0x10 E150			*Color Key1 AND Mask*	
31:24		-	Unused	

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
23:0	R/W	0xFFFFFF	ColorKeyAND1	Defines a 24-bit mask where the pixel is ANDed before it's keyed with the COLORKEY value.
Offset 0x10 E154			***Color Key Up1***	
31:24		-	Unused	
23:0	R/W	0	ColorKeyup1	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E158			***Color Key Low1***	
31:24		-	Unused	
23:0	R/W	0	ColorKeylow1	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E15C			***Color Key Replace1***	
31	R/W	0	Colorkeyreplaceen	Enables color replacement.
30:24		-	Unused	
23:0	R/W	0	ColorKeyreplace1	Defines a 24-bit color to be put into the data path if the color key matches.
Offset 0x10 E160			***Color Key2 AND Mask***	
31:24		-	Unused	
23:0	R/W	0xFFFFFF	ColorKeyAND2	Defines a 24-bit mask where the pixel is ANDed before it's keyed with the COLORKEY value.
Offset 0x10 E164			***Color Key Up2***	
31:24		-	Unused	
23:0	R/W	0	ColorKeyup2	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E168			***Color Key Low2***	
31:24		-	Unused	
23:0	R/W	0	ColorKeylow2	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E16C			***Color Key Replace2***	
31	R/W	0	Colorkeyreplaceen	Enables color replacement.
30:24		-	Unused	
23:0	R/W	0	ColorKeyreplace2	Defines a 24-bit color to be put into the data path if the color key matches.
Offset 0x10 E170			***Color Key3 AND Mask***	
31:24		-	Unused	
23:0	R/W	0xFFFFFF	ColorKeyAND3	Defines a 24-bit mask where the pixel is ANDed before it's keyed with the COLORKEY value.
Offset 0x10 E174			***Color Key Up3***	
31:24		-	Unused	
23:0	R/W	0	ColorKeyup3	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E178			***Color Key Low3***	
31:24		-	Unused	
23:0	R/W	0	ColorKeylow3	Defines a 24-bit color key used for color keying inside the layer.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 E17C			***Color Key Replace3***	
31	R/W	0	Colorkeyreplaceen	Enables color replacement.
30:24		-	Unused	
23:0	R/W	0	ColorKeyreplace3	Defines a 24-bit color to be put into the data path if the color key matches.
Offset 0x10 E180			***Color Key4 AND Mask***	
31:24		-	Unused	
23:0	R/W	0xFFFFFF	ColorKeyAND4	Defines a 24-bit mask where the pixel is ANDed before it’s keyed with the COLORKEY value.
Offset 0x10 E184			***Color Key Up4***	
31:24		-	Unused	
23:0	R/W	0	ColorKeyup4	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E188			***Color Key Low4***	
31:24		-	Unused	
23:0	R/W	0	ColorKeylow4	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E18C			***Color Key Replace4***	
31	R/W	0	Colorkeyreplaceen	Enables color replacement.
30:24		-	Unused	
23:0	R/W	0	ColorKeyreplace4	Defines a 24-bit color to be put into the data path if the color key matches.
Offset 0x10 E190			***Color Key Mask/ROP***	
31:28		-	Unused	
27:24	R/W	0x0	KeyFwd Note: This function is not implemented in the PNX8526.	These bits can revert the results of the ColorKeyPassROP for the individual 4 color keys of a layer. The results are 4 individual decision signals: KeyFwd[0] XOR ColorKeyPassROP KeyFwd[1] XOR ColorKeyPassROP KeyFwd[2] XOR ColorKeyPassROP KeyFwd[3] XOR ColorKeyPassROP 0= Select previous pixel color key. 1= Select new pixel color key n.
23:20	R/W	0	ColorKeyMask	This mask specifies which color to key in for the current pixel coming out of the layer.
19:16	R/W	0	ColorKeyMaskP	This color Mask is used to decide which color key to use for the incoming previous pixel.
15:8	R/W	0	PassZeroROP	This ROP decides if the output key value is 0 or is the one selected by ColorKeyPassROP. 1 = No color key (4-bit vector is 0). 0 = Color key vector selected by ColorKeyPassROP.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7:0	R/W	0	ColorKeyPassROP	This ROP determines which color key to select for the current mixer output. ROP output: 1= Select previous pixel color key. 0= Select new pixel color key.
Offset 0x10 E194			*Pixel Invert/Select ROP*	
31:16	R/W	0	InvertROP	This ROP decides if the previous pixel is inverted or not. ROP output: 1 = Invert previous pixel. 0 = Do not invert previous pixel.
15:0	R/W	0	SelectROP	This ROP determines which pixel to select for the current mixer output. ROP output: 1 = Select previous pixel. 0 = Select new pixel.
Offset 0x10 E198			*Alpha Blend/Key Pass*	
31:16	R/W	0	AlphaBlend	This ROP value determines whether or not to do an alpha blend. ROP output: 1 = Do alpha-blending. 0 = No alpha-blending
15:0	R/W	0	KeyPass	This ROP generates the key which is passed to the next layer mixer and is used as KEY0 in those ROPs.
Offset 0x10 E19C			*Alpha Pass*	
31:16	R/W	0	AlphaPass	This ROP value determines which alpha is passed to the next mixer stage. ROP output: 1 = Alpha of previous pixel 0 = Alpha of current pixel
15:0		-	Unused	
Offset 0x10 E1A0			*Color Key ROPs 1/2*	
31:16		-	Unused	
15:8	R/W	0	ColorKeyROP1	This ROP determines if results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn’t match. 1 = Color key matched.
7:0	R/W	0	ColorKeyROP2	This ROP determines if results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn’t match. 1 = Color key matched.
Offset 0x10 E1A4			*Color Key ROPs 3/4*	
31:16		-	Unused	

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:8	R/W	0	ColorKeyROP3	This ROP determines if results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn't match. 1 = Color key matched.
7:0	R/W	0	ColorKeyROP4	This ROP determines if the results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn't match. 1 = Color key matched.
Offset 0x10 E1A8			***Matrix Coefficients1***	
31:26		-	Unused	
25:16	R/W	0	CSCa12	2. cs matrix coefficient
15:10		-	Unused	
9:0	R/W	0x100	CSCa11	1. cs matrix coefficient
Offset 0x10 E1AC			***Matrix Coefficients2***	
31:26		-	Unused	
25:16	R/W	0	CSCa21	4. cs matrix coefficient
15:10		-	Unused	
9:0	R/W	0	CSCa13	3. cs matrix coefficient
Offset 0x10 E1B0			***Matrix Coefficients3***	
31:26		-	Unused	
25:16	R/W	0	CSCa23	6. cs matrix coefficient
15:10		-	Unused	
9:0	R/W	0x100	CSCa22	5. cs matrix coefficient
Offset 0x10 E1B4			***Matrix Coefficients4***	
31:26		-	Unused	
25:16	R/W	0	CSCa32	8. cs matrix coefficient
15:10		-	Unused	
9:0	R/W	0	CSCa31	7. cs matrix coefficient
Offset 0x10 E1B8			***Matrix Coefficients5***	
31:22	R/W	-	Unused	
21	R/W	0	TC_out3	Lower output two's complement switch 1 = Two's complement data 0 = Binary data
20	R/W	0	TC_out2	Middle output two's complement switch 1 = Two's complement data 0 = Binary data

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19	R/W	0	TC_out1	Upper output two's complement switch 1 = Two's complement data 0 = Binary data
18	R/W	0	TC3	Lower input two's complement switch 1 = Two's complement data 0 = Binary data
17	R/W	0	TC2	Middle input two's complement switch 1 = Two's complement data 0 = Binary data
16	R/W	0	TC1	Upper input two's complement switch 1 = Two's complement data 0 = Binary data
15:10		-	Unused	
9:0	R/W	0x100	CSCa33	9. cs matrix coefficient
Offset 0x10 E1BC			***DC-Offsets CSM Output***	
31:24		-	Unused	
23:16	R/W	0x0	Offset3	DC-offset lower output channel 8-bit two's complement
15:8	R/W	0x0	Offset2	DC-offset middle output channel 8-bit two's complement
7:0	R/W	0x0	Offset1	DC-offset upper output channel 8-bit two's complement
Offset 0x10 E1C0			***Layer Background Color***	
31	R/W	0	BG_enable	This bit enables the replacement of the previous input by the specified background color. 1 = Replace 0 = Use previous mixer output.
30:24		-	Unused	
23:16	R/W	0	Upper	Upper channel of the background color (R/Y)
15:8	R/W	0	Middle	Middle channel of the background color (G/U)
7:0	R/W	0	Lower	Lower channel of the background color (B/V)
Offset 0x10 E1C4			***DC-Offsets CSM Input***	
31		-	Unused	
23:16	R/W	0	Offset_in3	DC-offset lower input channel 8-bit two's complement
15:8	R/W	0	Offset_in2	DC-offset middle input channel 8-bit two's complement
7:0	R/W	0	Offset_in1	DC-offset upper input channel 8-bit two's complement
Offset 0x10 E1C8			***Start Fetch***	
31	R/W	0	Enable	This bit enables the start fetch functionality.
30:12	R/W	-	Unused	
11:0	R/W	0	Fetch Start	The current layer starts fetching data whenever the internal line counter reaches this point.

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 EFE0			*Interrupt Status AICP1*	
31:20	R	NI	y_positon	Current line position of the STG timing generator
19:16	R	NI	buffer status flags	Bit 16 corresponds to the current buffer status of layer4. Bit 17 corresponds to the current buffer status of layer3. Bit 18 corresponds to the current buffer status of layer2. Bit 19 corresponds to the current buffer status of layer1. 0 = buffer A 1 = buffer B
15	R	0	end_layer1_int	This bit is set when an end of layer 1 interrupt occurs.
14	R	0	end_layer2_int	This bit is set when an end of layer 2 interrupt occurs.
13	R	0	end_layer3_int	This bit is set when an end of layer 3 interrupt occurs. Unused in AICP2.
12	R	0	end_layer4_int	This bit is set when an end of layer 4 interrupt occurs. Unused in AICP2.
11	R	0	fifo_underflow1_int	This bit is set when a FIFO underflow in layer 1 occurs.
10	R	0	fifo_underflow2_int	This bit is set when a FIFO underflow in layer 2 occurs.
9	R	0	fifo_underflow3_int	Bit is set when a FIFO underflow in layer 3 occurs. Unused in AICP2.
8	R	0	fifo_underflow4_int	Bit is set when a FIFO underflow in layer 4 occurs. Unused in AICP2.
7	R	0	vlinta	This bit is set when the VLINTA specified in vinterrupt register occurs.
6	R	0	vlintb	This bit is set when the VLINTB specified in vinterrupt register occurs.
5	R	0	buffer_reuse_layer1_int	This bit is set when a buffer reuse interrupt for layer 1 occurs.
4	R	0	buffer_reuse_layer2_int	This bit is set when a buffer reuse interrupt for layer 2 occurs.
3	R	0	buffer_reuse_layer3_int	This bit is set, when a buffer reuse interrupt for layer 3 occurs. Unused in AICP2.
2	R	0	buffer_reuse_layer4_int	This bit is set when a buffer reuse interrupt for layer 4 occurs. Unused in AICP2.
1	R	0	vbi_done_int	This bit is set when a VBI done interrupt occurs. All VBI packets are transferred.
0	R	0	vbi_packet_int	This bit is set when a VBI packet is specified to issue an interrupt.
Offset 0x10 EFE4			*Interrupt Enable AICP1*	
31:16		-	Unused	
15:0	R/W	0	Interrupt Enables	A ‘1’ in the appropriate bit will enable the interrupt according to the specification in register 0xFE0.
Offset 0x10 EFE8			*Interrupt Clear AICP1*	
31:16		-	Unused	

AICP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:0	W	0	Interrupt Clears	A ‘1’ in the appropriate bit will clear the interrupt according to the specification in register 0xFE0.
Offset 0x10 EFEC			*Interrupt Set AICP1*	
31:16		-	Unused	
15:0	W	0	Interrupt Sets	A ‘1’ in the appropriate bit will set the interrupt according to the specification in register 0xFE0.
Offset 0x10 EFF4			*Powerdown*	
31	R/W	0	Powerdown	This bit has no effect i.e., there is no powerdown implemented for this module.
30:0		-	Unused	
Offset 0x10 EFFC			*Module ID*	
31:16	R	0x0111 AICP1 0x0118 AICP2	Module ID	Unique revision number
15:12	R	1	REV_MAJOR	Major revision counter
11:8	R	0	REV_MINOR	Minor revision counter
7:0	R	0	APP_SIZE	Aperture Size 0 = 4 kB

AICP 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
The AICP 2 registers begin at offset 0x10 F000 and correspond to the AICP 1 registers, with the following exceptions: - AICP 2 does not support Output gamma LUT control. (Registers 0x10 F040-F04C are reserved.) - AICP 2 supports only two layers. (Registers 0x10 F300-FFD8 are reserved.) - Because of these differences, note the settings in the Features register.				
Offset 0x10 F018			*FEATURES*	
31:5		-	Unused	
4:3	R	0x0	NOGAMMALUTS	Number of gamma LUTS binary coded for the output channels (The first slice of the AICP out goes through the gamma LUT.): Value for AICP2 is 0x0.
2:0	R	0x2	NOLAYERS	Number of layers in this AICP: Value for AICP2 is 0x2.
Note: For the interrupt to occur, it must be enabled in the STG control register.				
Offset 0x10 FFE0			*Interrupt Status AICP2*	
31:20	R		y_position	Current line position of the STG timing generator
19:18		-	Unused	

AICP 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17:16	R		buffer status flags	Bit 17 corresponds to the current buffer status of layer1. Bit 16 corresponds to the current buffer status of layer2.
15	R	0	end_layer1_int	This bit is set when an end of layer 1 interrupt occurs.
14	R	0	end_layer2_int	This bit is set when an end of layer 2interrupt occurs.
13		-	Unused	
12		-	Unused	
11	R	0	fifo_underflow1_int	This bit is set, when a fifo underflow in layer 1 occurred
10	R	0	fifo_underflow2_int	This bit is set, when a fifo underflow in layer 2 occurred
9		-	Unused	
8		-	Unused	
7	R	0	vlinta	This bit is set when the vertical line interrupt a specified in vinterrupt register occurs.
6	R	0	vlintb	This bit is set when the vertical line interrupt b specified in vinterrupt register occurs.
5	R	0	buffer_reuse_layer1_int	This bit is set when a buffer reuse interrupt for layer 1 occurs.
4	R	0	buffer_reuse_layer2_int	This bit is set when a buffer reuse interrupt for layer 2 occurs.
3		-	Unused	
2		-	Unused	
1	R	0	vbi_done_int	This bit is set when a VBI done interrupt occurs. All VBI packets are transferred.
0	R	0	vbi_packet_int	This bit is set when a VBI packet is specified to issue an interrupt

Chapter 29: 2D Drawing Engine

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

29.1 Overview

The 2D Drawing Engine (DE) module accelerates the 2D drawing operations that are most heavily used in a graphics environment. The 2D Drawing Engine interfaces to both the internal PI-Bus and the internal memory bus. DE operation may be synchronous to the memory interface, with a small portion of the module operating at the host clock frequency.

The major features of the 2D Drawing Engine are listed below:

- 3 operand bit BLT (256 Raster operations)
- Alpha-blending
- Transparent bit BLT
- Fast host monochrome to full color expansion for text, mono bitmaps, and patterns
- Lines

29.2 Operation

The Drawing Engine supports two types of operations: lines and bit BLTs. Line capability is limited to solid lines, with the software responsible for the calculation of the initial error terms of the Bresenham algorithm. Lines are provided as a compatibility check mark and are not highly optimized.

Bit BLTs are the primary function of the Drawing Engine and are highly optimized for performance. Bit BLT is the generic term used to indicate the transfer and processing of a block of visual data from one location to another. To specify the type of bit BLT, the following information is required:

- Raster Operation (ROP) of 256 possible ROPs
- Alpha Blend Mode: Source or Surface
- Source data location and type: system memory or CPU, mono, color, or alpha
- Transparency: On Source, On Destination, or none
- Pattern: 8 X 8 mono, 8 X 8 color, or solid

29.2.1 Raster Operations

The Drawing Engine supports a three operand bit BLT. The three operands are source, destination, and pattern. There are eight logical operations that can be performed with any combination of the three operands: one, zero, and, or, nand, nor, xor, nxor. This yields a total of 256 combinations of ROPs that can be performed. Although all 256 ROPs are possible, only a handful are typically used.

Examples of a single operand BLT are a pattern fill or a basic screen-to-screen copy where the source overwrites the destination. An example of a two operand BLT in the source is “xor’d” with the destination. A three operand BLT could “and” the source, pattern, and destination. The raster operation is defined by an 8-bit field specifying one of the possible 256 operations.

Raster ops are turned off if alpha-blending is enabled.

29.2.2 Alpha-Blending

The Drawing Engine supports alpha-blending of source and a global SurfaceAlpha. The alpha calculations are based on the source and “surface” alpha values.

Destination pixels always reside in memory. Source pixels may either come from memory (if they are in the same color format as the destination) or from the host data port. In addition to accepting pixels in the destination format, the host data port also accepts 4 or 8-bit alpha values. In this case, the color values for each source pixel come from the MonoHostBColor register (address 1424h).

The source pixels may contain either ‘normal’ or ‘pre-multiplied’ color values. If the source pixel is pre-multiplied, the alpha value in the source pixel is not applied to the source pixel. ‘Normal’ color values will have the source alpha applied.

In addition to blending source and destination data using the source’s alpha, the Drawing Engine also has a global “surface alpha.” The surface alpha is applied to all pixels in an alpha BLT. Surface alpha is stored in the MonoHostFColor register (address 1420h).

Alpha values in 32-bit pixels are always in the upper byte of the DWORD. If the destination color format is RGBα:4534 or αRGB:4444, the blended results can be dithered to present the appearance of higher color resoltion and to reduce banding artifacts.

Raster ops are disabled if alpha-blending is selected.

The following combinations of source and destination data are supported:

Table 1: Source and Destination Pixel Formats

Source	Source Format	Destination Format	Notes
Memory	αRGB:8888	αRGB:8888	Std alpha calculations
Host	αRGB:8888	αRGB:8888	Std alpha calculations
Host	α:4	αRGB:8888	MonoHostBColor reg provides src color

Table 1: Source and Destination Pixel Formats

Source	Source Format	Destination Format	Notes
Host	α:8	αRGB:8888	MonoHostBColor reg provides src color
Memory	RGB:565	RGB:565	Surface reg specifies alpha
Host	RGB:565	RGB:565	Surface reg specifies alpha
Host	α:4	RGB:565	MonoHostBColor reg provides src color
Host	α:8	RGB:565	MonoHostBColor reg provides src color
Memory	RGBα:4534	RGBα:4534	Std alpha calculations
Host	RGBα:4534	RGBα:4534	Std alpha calculations
Host	α:4	RGBα:4534	MonoHostBColor reg provides src color
Host	α:8	RGBα:4534	MonoHostBColor reg provides src color
Memory	αRGB:4444	αRGB:4444	Std alpha calculations
Host	αRGB:4444	αRGB:4444	Std alpha calculations
Host	α:4	αRGB:4444	MonoHostBColor reg provides src color
Host	α:8	αRGB:4444	MonoHostBColor reg provides src color

29.2.3 Source Data Location and Type

The data for the source operand can reside in either frame buffer memory or system memory. There are four source selection options currently defined:

1. Source data is held in frame buffer memory (always full color data)
2. Source data is held in system memory and is full color
3. Source data is held in system memory and is a monochrome bitmap
4. Source data is held in system memory and is a monochrome text font

Option 1 is used for normal screen-to-screen operations such as a source-to-destination copy. Option 2 is used for a system memory-to-screen copy when the source data is a full color bitmap. Option 3 is used for a system memory-to-screen copy when the source data is a monochrome bitmap. The monochrome bitmap will be expanded to full color using the foreground and background color registers within the Drawing Engine. Option 4 is similar to option 3 except that it is designed to handle tightly packed fonts that are used to render text. A 2-bit field is used to select the appropriate source data option.

29.2.4 Patterns

The Drawing Engine provides an 8-by-8 pattern or “brush.” The upper left corner of the pattern is mapped to the origin of the destination surface when the DstXY register is loaded. This pattern can be solid, monochrome, or full color. If the pattern is solid, the color value will be taken from the foreground color register. A monochrome pattern is stored in system memory as two DWORDs (64 bits) of monochrome data.

This monochrome data is written to the Drawing Engine where it is color expanded to the appropriate color depth and stored in the pattern RAM. A full color pattern will be directly transferred from system memory into the pattern RAM. Only one pattern may be stored in the pattern RAM at a given time.

29.2.5 Transparency

The Drawing Engine supports transparency on either source or destination data. Transparent patterns are not supported. Transparency works by comparing either source or destination to a value stored in a color compare register and then allowing (or disallowing) a write to occur based on the result of the compare. Transparency is implemented on a per-pixel basis: at 8 bpp one byte is used for the compare value. At 16 bpp one word is used for the compare, and at 32 bpp the entire color compare register is used for the compare. A transparency mask is provided in order to exclude certain bits within a pixel from being used in the color compare.

29.3 Programming Information

29.3.1 Mono Expand

The Drawing Engine needs to deal with two types of monochrome data from the host: fonts and bitmaps. The primary difference between fonts and bitmaps is how data are padded to byte boundaries.

First, it is worthwhile to note the bit ordering of monochrome data in a DWORD. Bit 7 is the left-most pixel, bit 24 is the right-most pixel. It is unfortunate having this mixture of data formats, since pixels in a byte are big-endian, but bytes are little-endian ordered. Therefore, in a DWORD bits are processed in the following order:

bit7, bit6, bit5,... bit0, bit15, bit8, bit 23... bit16, bit31... bit24

Fonts can be transferred to the DE in a highly packed format with no pad data between bits on adjacent scanlines. Pad is added after the last bits in the last data byte. Since the font data in system memory always begins on a byte boundary, the host processor can easily arrange to deliver a series of 32-bit aligned DWORDs to the DE. This font format has been widely used by Microsoft since Windows 95.

In the following example, the data stream for a 5*6 font requires 1 DWORD to be sent to the Engine.

```
            31                                    0
Data Stream:00110001001001111010000110000110


Rendered Font:  10000
                11010
                10000
                10010
                01110
                01100
                            Note the 2 pad bits (25 & 24) in the last byte.
```

Since font data starts on a byte boundary, a source shift parameter is not required. The Width field of the BltSize register determines how many host bits will be processed before advancing to the next scanline. There are no pad bits between data bits of adjacent scanlines—the data are highly packed. Excess bits in the very last DWORD of host data will be discarded. The Engine will require

(BltSize.Width*BltSize.Height +31)/32

DWORD for each glyph drawn.

Mono bitmaps sent to the Engine from Windows are not necessarily byte aligned on the left or right edges. The starting and ending bits of each scanline may be in the middle of a byte.

For mono bitmaps, the five lsbits of the SrcLinear register determine which bit in the first DWORD has the first valid data bit. The BltSize.Width register determines how many bits will be expanded/drawn for each scanline. The host will send

(BltSize.Width + (SrcLinear&31) + 31)/32

DWORDs for each scanline. The Engine will discard unused bits in the last DWORD of each scanline as pad bits. The driver must always send the correct number of DWORDs for each scanline in the bitmap.

29.3.2 Mono BLT Register Setup

To deal with both host-to-screen mono bitmap and text data in a general manner, the Engine uses up to eight parameters as shown in Table 2.

Table 2: Mono BLT Parameters

Parameter	Description
DstXY or DstLinear	Specifies the drawing destination on the screen
SrcLinear	The three lsbits specify the leading pad bits in the first byte of data on each scanline.
DstStride	Specifies the number of pixels between scanlines in the destination
MonoHostFColor	Foreground color
MonoHostBColor	Background color
CCColor	Color compare color for transparency
BltCtl	Drawing function
BltSize	Destination width and height

When the BltCtl.SRC register is set for font rendering, the Engine automatically accommodates the predefined padding and alignment requirements for fonts. This means that the host data will be tightly packed with no padding between scanlines or before the very first pixel. After the first font glyph has been sent from the host, only two registers will need to be re-loaded to initiate the next text BLT: DstXY/DstLinear and BltSize.

29.3.3 Solid Fill Setup

Solid fill is a common graphics operation. To achieve low programming overhead, the Drawing Engine only uses five parameters to set up a solid color fill.

Table 3: Solid Fill Parameters

Parameter	Description
DstXY or DstLinear	Specifies the drawing destination on the screen
DstStride	Specifies the number of pixels between scanlines in the destination
MonoPatFColor	Specifies the drawing color.
BltCtl	This register is set for a solid fill operation.
BltSize	Specifies destination width and height, initiates drawing function.

29.3.4 Color BLT Setup

The Engine uses up to 10 parameters to set up a color BLT.

Table 4: Color BLT Parameters

Parameters	Description
DstXY or DstLinear	Specifies the drawing destination on the screen.
SrcXY or SrcLinear	Specifies the location of source data for screen-to-screen BLTs. Specifies start of line alignment for host-to-screen operations.
SrcStride	Specifies the number of pixels between scanlines in the source for screen-to-screen BLTs. Unused for host-to-screen BLTs.
DstStride	Specifies the number of pixels between scanlines in the destination.
MonoPatFColor	Specifies the drawing color for solid fill BLTs using block write. Also may be used to load the PatRam quickly.
MonoPatBColor	The background color register may be used to help load the PatRam quickly.
CCColor	The Color Compare Color may be loaded if transparency is enabled in the BltCtl register.
TransMask	The transparency mask is used to mask out bits for color compare operations.
BltCtl	drawing function: ROP type, color compare enables, source data path
BltSize	Specifies destination width and height, initiates drawing function.

Simple Screen-to-Screen BLTs use four registers to initiate the operation: SrcXY/SrcLinear, DstXY/DstLinear, BltCtl, and BltSize. This assumes that the stride registers have already been initialized with the current screen pitch.

Simple Color Host-to-Screen BLTs use four registers to initiate the operation: DstXY/DstLinear, SrcXY/SrcLinear, BltCtl, and BltSize. This assumes that the DstStride register has already been initialized with the current screen pitch.

The SrcXY/SrcLinear register specifies the data alignment at the start of each scanline. The first DWORD of data will have (SrcLinear & 3) pixels of pad data in the least significant bytes. Each scanline of host data is padded to end on a DWORD boundary. The Engine will draw BltSize.Width pixels for each scanline of the BLT. The number of DWORDs of host data for each scanline is

((BltSize.Width + (SrcLinear & 3))*(PSize/8) + 3)/4

Advanced BLTs will initialize a small group of additional registers.

If transparency is desired, the Color Compare Color and Transparency Mask registers are loaded.

If patterns are used, the DstXY register and the PatRam (see below) must be loaded. The pattern is usually tiled to the screen assuming the upper left corner of the pattern is anchored to the upper left corner of the screen. To easily do this, the three low bits of the X and Y fields of the DstXY register are used to derive the initial alignment of the pattern data. The pattern register can be “un-anchored” by first writing to the DstXY register to specify the pattern alignment, then writing the DstLinear with the actual destination screen address. The last write to the DstXY register will set the pattern alignment. This means that you must always load the DstXY register before starting a BLT that uses the pattern.

29.3.5 PatRam

The PatRam is an 8*8 pixel pattern cache that provides pattern data for the ROP ALU. The PatRam operates at either 8, 16, or 32 bits per pixel as specified in the PSize register:

In 8 bit-per pixel mode, only the first 64 bytes of the PatRam are used. The host must initialize bytes 0 to 63 prior to initiating a BLT that uses a pattern. In this mode, byte 0 of the PatRam is the upper left-most pixel in the ram.

In 16-bit per pixel mode, only the first 128 bytes of the PatRam are used. The host must initialize bytes 0 to 127 prior to initiating a BLT that uses a pattern. In this mode, bytes 0 and 1 of the PatRam are the upper left-most pixel in the ram. Byte 0 is the LSB of the pixel and contains five bits of the blue component and three bits of the green component. Byte 1 contains the remaining bits of the green component and five bits of the red component.

In 32-bit per pixel mode, all 256 bytes of each scanline of the PatRam are used. The host must load the entire PatRam prior to initiating a BLT that uses a pattern. In this mode, bytes 0..3 of the PatRam are the upper left-most pixel in the ram. Byte 0 is the LSB of the pixel and contains the blue component, byte 1 contains the green component, byte two contains the red component, and byte three is the alpha component (usually unused).

The PatRam can be loaded in two different ways to load either color data or monochrome data.

The 256-byte PatRamColor section of the register space allows direct byte/word/DWORD access to the PatRam. This allows arbitrary pattern data to be loaded. 64 pixels of data must be loaded into the PatRam prior to use. This ranges from 64 to 256 bytes of data depending on color depth. Byte 0 of this space is the first byte of PatRam scanline 0.

To assist in loading mono patterns, the 8-byte PatRamMono section of the register space provides automatic color expansion of mono data while loading the PatRam. Thus, the host only sends 16 bytes (four DWORDS) of data to initialize the entire pattern regardless of color depth: MonPatFColor, MonoPatBColor, and 8 bytes of mono data (64 bits). Each bit in the mono data stream loads the appropriate byte(s) of the PatRam with either the foreground color if the bit is a 1, or else the background color if the bit is a 0. Byte0/Bit7 loads the left-most pixel of scanline 0, byte1/Bit7 loads the left-most pixel of scanline 1, etc.

29.4 Register Descriptions

The Drawing Engine uses five areas of memory space:

- The first memory space area is for the DE command registers. These registers are used to setup and execute DE commands.
- The next area decoded is for monochrome pattern data. Although only 8 bytes of monochrome pattern data are required, a 256-byte decode is implemented to allow color expansion to occur to different areas of the pattern RAM.
- The third decoded area is for full color pattern data: 64, 128, or 256 bytes of data may be written to the pattern RAM.
- The fourth area decoded is for “real time” drawing registers. Unlike command registers which are pipelined, real time registers can be accessed immediately.
- The last memory area decoded is a 64-kbyte area used to transfer host data to the DE.

The addresses in the following table refer to memory space or indexed I/O space, with the exception of the host data space that has separate memory and I/O addresses.

The base address for the PNX8526 2D Drawing Engine is 0x04 F000.

29.4.1 Register Address Maps

The following tables provide a map of the 2D Drawing Engine registers.

Table 5: Memory Space Addresses

Address Range	Decoded For
0x04 F400—F47F 0x04 F5F8—F5FF	Drawing Engine Command Registers
0x04 F480—F5F7	Reserved for future use
0x04 F600—F6FF	Monochrome Pattern Data
0x04 F700—F7FF	Color Pattern Data
0x04 F800—F80B	Real Time Drawing Engine Registers
0x05 0000—FFFF	Drawing Engine Host Data

Table 6: Command Register Map

Register Name	Address	Function
SrcAddrBase	0x04 F400	SRC Base address for XY->linear
DstAddrBase	0x04 F404	DST Base address for XY->linear
Psize	0x04 F408	Color depth for drawing operations
SrcLinear	0x04 F40C	BLT src address (linear)
DstLinear	0x04 F410	Vector/BLT dst address (linear)
SrcStride	0x04 F414	Scanline src pitch for BLTs
DstStride	0x04 F418	Scanline dst pitch for BLTs/vectors
CCColor	0x04 F41C	Color compare target
MonoHostFColor and SurfAlpha	0x04 F420	Foreground color for mono host expansion, SurfAlpha register for alpha-blending
MonoHostBColor and HAlphaColor	0x04 F424	Background color for mono host expansion, color value for host alpha data for alpha-blending
BltCtl	0x04 F428	Raster Op, etc. selection
SrcXY	0x04 F42C	BLT src address (XY)
DstXY	0x04 F430	Vector/BLT dst address (XY)
BltSize	0x04 F434	BLT Size (width and height) *Initiates BLT Drawing*
DstXY2	0x04 F438	Vector/BLT dst address (XY)
VecConst	0x04 F43C	Vector Bresenham constants
VecCount	0x04 F440	Vector length, octant, error term. *Initiates Line Drawing*
TransMask	0x04 F444	Transparency Mask
reserved	0x04 F448—F5F4	Reserved for future use
MonoPatFColor	0x04 F5F8	Foreground color for mono pattern expansion, lines, and solid fills
MonoPatBColor	0x04 F5FC	Background color for mono pattern expansion, lines, and solid fills

Table 7: Real Time Drawing Register Map

Register Name	Address	Function
EngineStatus	0x04 F800	Engine Status
PanicControl	0x04 F804	Engine Abort
EngineConfig	0x04 F808	Engine Configuration
HostFIFOStatus	0x04 F80C	Host FIFO entries available
DeviceID	0x04 FFFC	Indicates the device ID
PatRamMono	0x04 F600—F6FF	Pattern RAM Mono
PatRamColor	0x04 F700—F7FF	Pattern RAM Color
Host Data	0x05 0000—5FFFF	Host Data

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Drawing Engine Command Registers				
Offset 0x04 F400			*Source Address Base*	
31:29	R/W	0	Swap[2:0]	Specifies endian-swapping on reads from memory. When Swap[2] is 0, swapping is determined by the global endian setting, possibly modified by BSI[0] in EngineConfig. When Swap[2] is 1, swapping is determined by Swap[1:0] as shown: 00=No swapping. Memory is little-endian. 01=Bytes are swapped within each 16-bit word. 10=Words are swapped within each 32-bit double word. 11=Bytes are swapped within each 32-bit double word.
28:24			Reserved	
23:16	R/W	0	Off[22:16]	Specifies the offset for pixel (0,0) of a source bitmap for XY to Linear conversion of addresses. Bits 2:0 must be set to 0.
15:8	R/W	0	Off[15:8]	
7:0	R/W	0	Off[7:0]	

This register specifies the offset for pixel (0,0) of a source bitmap for XY to Linear conversion of addresses.

It is interpreted as a byte address. The lower three bits of this register always read 0, regardless of the value written. This implies that the source base address must be aligned to a 64-bit boundary. Under many circumstances, this register will be initialized to the proper offset and then changed only for special effect.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F404			*Destination Address Base*	
31:29	R/W	0	Swap[2:0]	Specifies endian-swapping on reads from memory. When Swap[2] is 0, swapping is determined by the global endian setting, possibly modified by BSI[0] in EngineConfig. When Swap[2] is 1, swapping is determined by Swap[1:0] as shown: 00=No swapping. Memory is little-endian. 01=Bytes are swapped within each 16-bit word. 10=Words are swapped within each 32-bit double word. 11=Bytes are swapped within each 32-bit double word.
28:24			Reserved	
23:16	R/W	0	Off[22:16]	Specify the offset for pixel (0,0) of a destination bitmap for XY to Linear conversion of addresses. Bits 2:0 must be set to 0.
15:8	R/W	0	Off[15:8]	
7:0	R/W	0	Off[7:0]	

This register specifies the offset for pixel (0,0) of a destination bitmap for XY to Linear conversion of addresses.

It is interpreted as a byte address. The lower three bits of this register are always interpreted as 0, regardless of the value written. When reading the contents of this register, the lower three bits will be read back as 0. This implies that the source base address must be aligned to a 64-bit boundary. Under many circumstances, this register will be initialized to the proper offset and then changed only for special effects.

2D DRAWING ENGINE REGISTERS

Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F408			*Pixel Size*	
31:16			Reserved	
15:8	R/W	0	PFormat[7:0]	Currently used only during alpha-blending operations. The format is dependent on the color depth. Refer to Table on page 29-638 for bit assignments. 16 bpp: 0000_0000RGB 565 0000_0001αRGB 4444 0000_0010RGBα 4534 0000_1000RGB 565 dithered 0000_1001αRGB 4444 dithered 0000_1010RGBα 4534 dithered 32 bpp: 0000_0000αRGB 8888 0000_0001αVYU 8888 0000_0010αYUV 8888 Note: Specifying any other combination of bits will result in an invalid command being executed.
7:0	R/W	0	Depth[5:0]	Specifies the number of bits per pixel. 00100000b32 bpp 00010000b16 bpp 00001000b8 bpp All other values are reserved.

This register specifies the pixel size and format for the Drawing Engine for BLTs and vectors. It is unchanged by any drawing operations.

The following table shows the pixel bit assignments for each of the six color formats above.

Remark: In the 4534 case, the format name is not indicative of the actual order of the bits:

Table 8: Pixel Format Bit Assignments

Format	31 24	23 16	15 8	7 0
RGB 565			RRRRRGGG	GGGBBBBB
αRGB 4444			aaaaRRRR	GGGGBBBB
RGBα 4534			aaaaRRRR	GGGGGBBB
αRGB 8888	aaaaaaaa	RRRRRRRR	GGGGGGGG	BBBBBBBB
αVYU 8888	aaaaaaaa	VVVVVVVV	YYYYYYYY	UUUUUUUU
αYUV 8888	aaaaaaaa	YYYYYYYY	UUUUUUUU	VVVVVVVV

The YUV values have ranges which match the D1 format in the CCIR-656. Y is an unsigned value ranging from 16 to 235. U and V are signed offset values ranging from 16 to 240, where 128 represents the zero point.

When Depth is 16, PFormat[3] enables dithering the results of an alpha blend operation. All alpha blend operations are done with 8 bits of precision for each component. The components of the source pixels are expanded to 8 bits by replicating the high-order bits into the low-order bits. The results of the computations can be thought of as being in fixed point with 3, 4, 5 or 6 bits of precision to the left of the decimal point, depending on the component and color format. When dithering is enabled, a 4x4 dither is applied by adding a constant fraction to each component and truncating the results to fit in the resulting pixel. The constant fraction is taken from the following table, based on the X and Y coordinates of the destination pixel:

Table 9: Dithering Constant Fractions

X mod 4 = 0	X mod 4 = 1	X mod 4 = 2	X mod 4 = 3	Y mod 4 = 0
7/8	3/8	1/8	5/8	Y mod 4 = 1
1/8	5/8	7/8	3/8	Y mod 4 = 2
5/8	7/8	3/8	1/8	Y mod 4 = 3
3/8	1/8	5/8	7/8	

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F40C			*Source Linear*	
31:24			Reserved	
23:16	R/W	0	Adr[22:16]	Used to load the linear source address for a BLT operation.
15:8	R/W	0	Adr[15:8]	
7:0	R/W	0	Adr[7:0]	

This register is used to load the linear source address for a BLT operation. It must be loaded with a pixel aligned address.

Remark: Loading the SrcXY register actually causes the register to be loaded with the proper linear pixel address.

This register is interpreted as a byte address, except during a monochrome host to screen bit BLT or a 4-bit or 8-bit expand alpha BLT. In the monochrome BLT case, Adr[4:0] specifies the first valid bit within the first DWORD transferred by the host. Adr[4:3] specifies the correct byte and Adr[2:0] specify the correct bit. In the 4-bit expand case, Adr[3:0] specifies the first valid nibble within the alpha data transferred by the host. In the 8-bit expand case, Adr[2:0] specifies the first valid byte within the alpha data transferred by the host.

This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F410			*Destination Linear*	
31:24			Reserved	
23:16	R/W	0	Adr[22:16]	Used to load the starting linear pixel address for a vector or the destination linear address for a BLT operation.
15:8	R/W	0	Adr[15:8]	
7:0	R/W	0	Adr[7:0]	

This register is used to load the starting linear pixel address for a vector or the destination linear address for a BLT operation.

It is interpreted as a byte address and must be loaded with a pixel aligned address.

Remark: Loading the DstXY register actually causes this register to be loaded with the proper linear pixel address. This register may not be used to specify the destination address for a command that utilizes patterns.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F414			*Source Stride*	
31:14			Reserved	
13:0	R/W	0	SrcStr	Used to load the starting linear pixel address for a vector or the destination linear address for a BLT operation. Bits 2:0 must be set to 0.

This register holds the unsigned offset between adjacent source scanlines for screen-to-screen BLTs. Under many circumstances, this register will be initialized to the screen pitch and then changed only for special effects.

This 14-bit register is interpreted as an unsigned byte value. It is used during a BLT to step from scanline to scanline. It is also used to convert a SrcXY address to a SrcLinear address according to the following formula:

$$\mathrm{SrcLinear} = \mathrm{SrcXY.Y} * \mathrm{SrcStride} + \mathrm{SrcXY.X}^{(1)} + \mathrm{SrcAddrBase}$$

(1) This value is adjusted for pixel color depth.

There are no restrictions on this register except that the lower three bits are always interpreted as 0, regardless of the value written. When reading the contents of this register, the lower three bits will be read back as 0. This implies the source stride must be a multiple of 8 bytes.

As this is an unsigned register, it is always interpreted as a positive value. The direction in which a source is traversed is controlled by the BLT direction field in the BltCtl register. This register may be useful for BLTing bitmaps stored in offscreen memory in a 1D format to the screen. It is unchanged by any drawing operation.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F418			***Destination Stride***	
31:14			Reserved	
13:8	R/W	0	DstStr[13:8]	Hold the offset between adjacent scanlines for BLTs and vectors. Bits 2:0 must be set to 0.
7:0	R/W	0	DstStr[7:0]	

This register holds the offset between adjacent scanlines for BLTs and vectors. Under many circumstances, this register will be initialized to the screen pitch and then changed only for special effects. It is interpreted as an unsigned byte value.

This 14-bit signed register is used during BLTs and vectors to step from scanline to scanline. It is also used to convert a DstXY address to a DstLinear address according to the following formula:

$$\mathrm{DstLinear} = \mathrm{DstXY.Y} * \mathrm{DstStride} + \mathrm{DstXY.X}^{(1)} + \mathrm{DstAddrBase}$$

(1) This value is adjusted for pixel color depth

There are no restrictions on this register except that the lower three bits are always interpreted as 0, regardless of the value written. When reading the contents of this register, the lower three bits will be read back as 0. This implies that the destination stride must be a multiple of 8 bytes.

As an unsigned register, it is always interpreted as a positive value. The direction in which the destination is traversed is controlled by the BLT direction field in the BltCtl register.

This register may be useful for BLTing bitmaps stored in offscreen memory in a 1D format to the screen. It is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F41C			*Color Compare*	
31:24	R/W	0	CCCol[31:24]	Hold the color compare target color.
23:16	R/W	0	CCCol[23:16]	
15:8	R/W	0	CCCol[15:8]	
7:0	R/W	0	CCCol[7:0]	

This register holds the color compare target color. It should be initialized prior to any BLT that enables color compare. The appropriate number of bytes needs to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8 bpp mode. In 16 bpp mode, the lower word will be replicated into the upper word. In 32 bpp mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F420			*Mono Host F Color or SurfAlpha*	
31:24	R/W	0	FCol[31:24] alpha[7:0]	Specify the foreground color for host bitmap monochrome expansion. For alpha-blending, this register specifies the global surface alpha values.
23:16	R/W	0	FCol[23:16] R/V/Y[7:0]	
15:8	R/W	0	FCol[15:8] G/Y/U[7:0]	
7:0	R/W	0	FCol[7:0] B/U/V[7:0]	

Specifies the foreground color for host bitmap monochrome expansion. For alpha-blending, this register specifies the global surface alpha values.

It holds the foreground color for host bitmap monochrome expansion. The appropriate number of bytes needs to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When used as SurfAlpha, this register always contains four unsigned 8-bit alpha values, regardless of color depth.

When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8 bpp mode. In 16 bpp mode, the lower word will be replicated in the upper word. In 32 bpp mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F424			***Mono Host B Color or HAlpha Color***	
31:24	R/W	0	BCol[31:24]	Hold the background color for host bitmap monochrome expansion. For alpha-blending this register may provide color data.
23:16	R/W	0	BCol[23:16] R/V/Y[7:0]	
15:8	R/W	0	BCol[15:8] G/Y/U[7:0]	
7:0	R/W	0	BCol[7:0] B/U/V[7:0]	

This register holds the background color for host bitmap monochrome expansion. The appropriate number of bytes need be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When used as HostAlphaColor, the least significant three bytes of this register always contain 8-bit color values, regardless of the current depth. The most significant byte is supplied by the expanded host data during a 4-bit or 8-bit alpha expand BLT. In the YUV formats, the order of the components in the least significant three bytes matches that of the current PFormat.

When reading the value of this register, the lower byte will be replicated into all four byte lanes in 8 bpp mode. In 16 bpp mode, the lower word will be replicated into the upper word. In 32-bpp mode, or when A[3:0] in BltCtl are non-zero, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F428			*Blt Control*	
31:27			Reserved	
26:24	R/W	0	BD[2:0]	The BD[2:0] field specifies the bitblt direction. BD[0] specifies the horizontal BLT direction. It is bit 24 of this register: 0 = Blt direction is left to right. 1 = Blt direction is right to left. BD[1] specifies the vertical BLT direction. It is bit 25 of this register: 0 = Blt direction is top to bottom. 1 = Blt direction is bottom to top. BD[2] enables vertical flipping during the BLT by allowing the source data to be read in the opposite vertical direction from the destination data. Bd[2] is bit 26 of this register: 0 = Src is read as specified in BD[1.] 1 = Src is read in reverse of BD[1].
23:20	R/W	0	A[3:0]	The A[3:0] field controls alpha-blending operations and is in bits 23:20 of this register. A[1:0] controls enabling of alpha-blending and the format of the source data. The valid values are: 0 = Alpha-blending is disabled. 1 = Src data contains normal alpha data. 2 = Src data contains pre-multiplied alpha data. 3 = Src data does not have alpha data, surface alpha is the only alpha value. Bit 2 of this field controls the updating of the alpha field in the destination: 0 = Preserve destination alpha field. 1 = Update destination alpha field. Bit 3 of this field controls whether destination data participates in the alpha blend operation. It is used to implement “unary” blends: 0 = Blend source with destination data 1 = Blend source with black IN RGB mode, “black” means RGB = (0, 0, 0). In YUV mode, “black” means YUV = (16, 128, 128).

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19			Reserved	
18:16	R/W	0	CC[2:0]	The CC[2:0] specifies the operation of the color compare hardware. CC[2:0] are bits 19:16 of this register. Legal values are: 0 = Color compare disabled. 1 = SRC data is used for color compare, perform write operation if colors match. 2 = DST data is used for color compare, perform write operation if colors match. 3 = Reserved 4 = Reserved 5 = SRC data is used for color compare, perform write operation if colors don’t match. 6 = DST data is used for color compare, perform write operation if colors don’t match. 7 = Reserved
15:13			Reserved	
12	R/W	0	SP	The SP field indicates if the pattern is a solid color. SP is bit 12 of this register. Legal values are: 0 = Patterns are handled normally. 1 = The value held in the MonoPatFColor foreground color register will be used as pattern data.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
11			Reserved	
10:8	R/W	0	SRC[2:0]	The SRC[2:0] field is a 3-bit parameter specifying how the source data are created. SRC[2:0] are in bits 11:8 of this register: 0 = SRC data is color data from SGRAM. This is used for screen-to-screen BLTs with either ROPs or alpha blends. 1 = SRC data is color bitmap data from the host processor. This is used for host-to-screen BLTs with either ROPs or alpha blends. 2 = SRC data is PC mono bitmap data from the host processor. This is used for color expanding host-to-screen BLTs. This encoding implies that host data is padded to a DWORD boundary at the end of scanlines. 3 = SRC data is PC mono font data from the host processor. This is used for text rendering. This encoding implies highly packed host data and forces the Drawing Engine to use SrcStride=BltSize. Width and SrcLinear=0. 4 = SRC data is 4-bit alpha values from the host. This option is used with alpha-blending. 5 = SRC data is 8-bit alpha values from the host. This option is used with alpha-blending. 6 = Use only surface values for alpha-blending, no SRC data present. 7 = Reserved
7:0	R/W	0	ROP[7:0]	The ROP[7:0] field is an 8-bit parameter that specifies the raster op. It is the same format used by GDI. ROP[7:0] are in bits 7:0 of this register.

This register specifies the current raster op, alpha, and other parameters for the Drawing Engine data path. This register must be properly initialized for BLTs, Alpha Blends, and vectors.

This register is unchanged by any drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F42C			*Source Address, XY Coordinates*	
31:27			Reserved	
26:24	R/W	0	Y[10:8]	Unsigned 11-bit Y source address
23:16	R/W	0	Y[7:0]	
15:11			Reserved	
10:8	R/W	0	X[10:8]	Unsigned 11-bit X source address
7:0	R/W	0	X[7:0]	

This register is used to load the source XY address for a BLT operation.

The X and Y fields are unsigned 11-bit numbers allowing a 2K by 2K address space. Since the Drawing Engine uses linear addresses internally, the X and Y coordinates in this register will be converted to a linear address. It is byte accessible; a write to the high byte of this register begins the conversion process from XY to linear. It is not used for vectors.

Remark: SrcLinear should be utilized when using monochrome bitmaps or text. The lower six bits of SrcLinear specify the starting bit position within a DWORD. Also loads the SrcLinear register with the converted XY address.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F430			*Destination Address, XY Coordinates*	
31:27			Reserved	
26:24	R/W	0	Y[10:8]	Unsigned 11-bit Y destination address
23:16	R/W	0	Y[7:0]	
15:11			Reserved	
10:8	R/W	0	X[10:8]	Unsigned 11-bit X destination address
7:0	R/W	0	X[7:0]	

This register is used to load the starting XY pixel destination coordinate for a drawing operation.

The X and Y fields are unsigned 11-bit numbers allowing a 2K by 2K address space. Since the Drawing Engine uses linear addresses internally, the X and Y coordinates in this register will be converted to a linear address. It is byte accessible; a write to the high byte of this register begins the conversion process from XY to linear.

This register causes the same behavior as writing to DstXY2, which is provided to allow for command register bursting during vector commands.

Drawing commands that require patterns must use this register to specify the destination coordinate. Using the DstLinear register to specify destination during a pattern command will cause errant results. Also loads the DstLinear register with the converted XY address.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F434			*BLT Size*	
31:28			Reserved	
27:16	R/W	0	H	Height of the BLT in scanlines
15:12			Reserved	
11:0	R/W	0	W	Width of the BLT in pixels

This register specifies the height and width of a BLT operation in pixels/scanlines.

W[11:0] specifies the width of a BLT in pixels. The minimum allowed value is 1 and the maximum value is 4k-1. Zero is not a valid value for this field. W[11:0] corresponds to bits 11:0 of this register. H[11:0] specifies the height of a BLT in lines. The minimum allowed value is 1 and the maximum value is 4k-1. Zero is not a valid value for this field. H[11:0] corresponds to bits 27:16 of this register.

Remark: Loading the high byte of this register starts a BLT or Alpha-Blending operation. This register is unchanged by any drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F438			*Destination Address, XY2 Coordinates*	
31:27			Reserved	
26:24	R/W	0	Y[10:8]	Unsigned 11-bit Y destination address
23:16	R/W	0	Y[7:0]	
15:11			Reserved	
10:8	R/W	0	X[10:8]	Unsigned 11-bit X destination address
7:0	R/W	0	X[7:0]	

This register is used to load the starting XY pixel destination coordinate for a drawing operation.

The X and Y fields are unsigned 11-bit numbers allowing a 2K by 2K address space. Since the Drawing Engine uses linear addresses internally, the X and Y coordinates in the register will be converted to a linear address. It is byte accessible; a write to the high byte of this register begins the conversion process from XY to linear.

This register causes the same behavior as writing to DstXY, which is provided to allow for command register bursting during bitblt commands.

Drawing commands that require the use of patterns must use this register to specify the destination coordinate. Using the DstLinear register to specify the destination during a pattern command will cause errant results. Loading this register also loads the DstLinear register with the converted XY address.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F43C			*Vector Constant*	
31:24	R/W	0	Const1 [15:8]	Const1 = 2 x dmin
23:16	R/W	0	Const1 [7:0]	
15:8	R/W	0	Const0 [15:8]	Const0 = 2 x dmin - 2 x dmax
7:0	R/W	0	Const0 [7:0]	

This register holds the two error term values for the Bresenham line algorithm. These values are computed by the host processor as follows:

Const0 = 2 x dmin - 2 x dmax
Const1 = 2 x dmin

This register is unchanged after the vector is completed.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F440			*Vector Count Control*	
31:24	R/W	0	InitErr[15:8]	Initial error value for the Bresenham line algorithm
23:16	R/W	0	InitErr[7:0]	
15:8	R/W	0	Oct, Len[11:8]	Bits 15:13 specify the drawing octant as follows: Bit 15 1Y is major axis 0 X is major axis Bit 14 1 negative X step 0 positive X step Bit 13 1 negative Y step 0 positive Y step
7:0	R/W	0	Len[7:0]	Length of the major axis in pixels

This register holds the vector length, drawing octant, and the initial error value for the Bresenham line algorithm. The initial value of the error term is held in bits 31:16 and is computed by the host processor:

InitErr = 2 x dmin - dmax
Len = dmax + 1

where dmin = min (dx,dy), dmax = max(dx,dy)

The last pixel can be left undrawn by simply decrementing the Len parameter prior to loading.

Remark: Loading the high byte of this register starts the vector Drawing Engine. A Len value of 0 is illegal and should be avoided.

This register is unchanged after the vector is completed, though it must be reloaded to start the next vector.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F444			*TransMask*	

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:24	R/W	0	TMask[31:24]	Transparency mask used in the color compare
23:16	R/W	0	TMask[23:16]	
15:8	R/W	0	TMask[15:8]	
7:0	R/W	0	TMask[7:0]	

This register holds the transparency mask used in the color compare. It should be initialized prior to any BLT that enables color compare. The appropriate number of bytes needs to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 15 or 16 bits, the lowest two bytes need to be written. In 32-bit mode, all bytes should be written.

Setting a bit to 1 in this register enables the corresponding bit within a pixel to be used in the color compare. Clearing a bit to 0 in this register excludes the corresponding bit within a pixel from being used in the color compare. This effectively causes that bit(s) to be considered a match in the color compare. Correct programming values are shown below:

```
08bpp: 0x000000FF     // All 8 bits included in Color Compare
15bpp: 0x00007FFF     // 15 lower bits included in Color Compare
16bpp: 0x0000FFFF     // All 16 bits included in Color Compare
32bpp: 0x00FFFFFF     // 24 lower bits included in Color Compare
```

When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8-bpp mode. In 15 or 16-bpp mode, the lower word will be replicated in the upper word. In 32-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F5F8			*MonoPatFColor*	
31:0	R/W	0	MonoPatFColor[31:0]	Specifies the foreground color for monochrome pattern expansion, lines, and solid fills.

This register specifies the foreground color for monochrome pattern expansion, lines, and solid fills. The appropriate number of bytes needs to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8-bpp mode. In 16-bpp mode, the lower word will be replicated in the upper word. In 32-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F5FC			*MonoPatBColor*	
31:0	R/W	0	MonoPatBColor[31:0]	Holds the background color monochrome pattern expansion, lines, and solid fills.

This register holds the background color for monochrome pattern expansion, lines, and solid fills. The appropriate number of bytes needs to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8-bpp mode. In 16-bpp mode, the lower word will be replicated in the upper word. In 32-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Drawing Engine Real Time Registers				
Offset 0x04 F800			*EngineStatus*	
31:11			Reserved	
10	R	0	IRQ	Draw Engine interrupt request status 1 = A 2D interrupt is being requested. This reflects the actual state of the IRQ signal leaving the Drawing Engine.
9	R	0	DEDone	DeDone and DEBusy are the primary Drawing Engine activity indicators.They are the complement of each other. When DEBusy is logic 1 (DEDone a 0), the Drawing Engine is active. If EngineConfig bit 9 is a 1, "active" is defined as: processing a register access emptying commands or data from the Host FIFO performing an operation such as a bitblt or line waiting for a memory transaction to complete If EngineConfig bit 9 is a 0, the engine is active when a blt/vector starts and becomes inactive when the blt/vector finishes. All memory writes are complete, AND the host FIFO is empty. DEBusy is read as logic 0, the Drawing Engine is guaranteed to be idle.
8	R	0	DEBusy	
7:4			Reserved	
3	R	0	HFIFO_not empty	Host FIFO is not empty.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2	R	0	HFIFO_full	Host FIFO is full. BLT is busy; another BLT is pending in shadow registers.
1	R	0	Vector active	Vector is in process.
0	R	0	BLT active	BLT is in process.

This read-only register returns the Drawing Engine status. Writes to this register have no effect and do not hang the bus.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F804			*PanicControl*	
31:8			Reserved	
7:0	W	0	RST	Used to reset the state machines in the Drawing Engine. Writing 0x0000_0001 to this register will halt the Drawing Engine and place it in an idle state. Writing 0x0000_0000 will allow the DE to be reprogrammed and resume normal operation. It may be necessary for software to implement a delay between setting the RST signal to 1 and resetting it to 0.

This register is currently used to reset the state machines in the Drawing Engine. It does not reset any other registers in the Engine.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F808			*EngineConfig*	
31:11			Reserved	
10	R/W	0	IRQ_CLR	IRQ_CLR (bit 10) is a self-clearing bit used to reset Drawing Engine IRQ flip-flop. The IRQ flip-flop is set when the DEBusy bit in the EngineStatus register transitions from 1 to 0. It is cleared under software control by setting IRQ_CLR to 1. See BMODE for a description of DEBusy.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
9	R/W	0	BMODE	BMODE selects between two slightly different behaviors of DEBusy and DEDone (EngineStatus register). 0=the DEBusy bit is set to 1 when the BLT/vector state machine becomes active i.e., a drawing operation is starting. The bit is cleared only when the state machine is idle, all memory writes are completed, and the command FIFO is empty. 1=the DEBusy bit is set whenever the BLT/vector state machine is active OR the command FIFO is not empty. The DEBusy bit will be cleared when the engine is idle AND the command FIFO is empty. Note that in this mode, the Draw Engine will go busy whenever a register is loaded. Using interrupts can be tricky in this mode.
8	R/W	-	IRQ_EN	IRQ_EN (bit 8) is used to enable the interrupt signal leaving the Drawing Engine module. When IRQ_EN is set to a 1, the interrupt signal is enabled. When set to 0, the interrupt signal leaving the Drawing Engine is masked. The IRQ_EN does not affect the actual IRQ flip-flop. It merely masks the IRQ bit leaving the module.
7:3			Reserved	
2	R/W	0	BSM	BSI (bit 1) and BSM (bit 2) toggle byte and word swapping. Setting BSI to 1 reverses the sense of the global endian bit for data read from the host data input port, so that data are swapped when written when they normally would not be, and vice versa. This bit affects host BLT data, pattern loading. It is intended for debugging and should be set to 0 for normal operation.
1	R/W	0	BSI	BSI (bit 1) and BSM (bit 2) toggle byte and word swapping. Setting BSM to 1 reverses the sense of the global endian bit for data being read and written to the memory port. It is intended for debugging and should be set to 0 for normal operation.
0			Reserved	

This register configures specific Drawing Engine features and enables the interrupt request signal.

BSI (bit 1) and BSM (bit 2) toggle byte and word swapping. Certain registers that contain byte or word-oriented data, but must be manipulated as DWORDS, need to be byte or word-swapped when written on big-endian architectures. These include HostData, PatRamMono and PatRamColor. The Drawing Engine automatically determines whether swapping is necessary based on the global “endian” flag.

The bottom byte of this register should not be altered while drawing commands are in progress.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F80C			*HostFIFOStatus*	
31:6			Reserved	
5:0	R	0	Level[5:0]	Used to provide software with a method of determining the number of available entries in the Host FIFO.

This register provides software with a method of determining the number of available entries in the Host FIFO.

There are 32 FIFO locations in the Host FIFO. If read as 0, it indicates there are no available FIFO locations available for writing. A write to the Host FIFO while the FIFO is full (or almost full) will result in wait states on the PI-Bus.

If read as 0x20, it indicates that all 32 entries are available for writing.

A software driver can read this register and subsequently write the corresponding number of DWORDs of data to the Host FIFO. This will prevent the PNX8526 from generating PCI retries since a write will not occur when the Host FIFO is full.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 FFF4			*POWERDOWN*	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x04 FFFC			*DeviceID*	
31:16	R	0x0117	ModuleID	ModuleID is 0117h
15:12	R	0	MajRev	The initial Major revision field is 0h.
11:8	R	0	MinRev	The initial Minor revision field is 0h.
7:0	R	0x10	Size	The size is 10h (68 kB).

This register provides software with the module type, revision, and aperture size.

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Drawing Engine Data Registers				
Offset 0x04 F600—F6FF			*PatRamMono*	
This address range allows write-only access to the pattern RAM. It is used to load monochrome patterns into the pattern cache with automatic color expansion. Each bit in this address range represents one pixel in the pattern cache. A ‘1’ written here is converted to the foreground color and written to the Pattern RAM. A ‘0’ written here is converted to the background color and written to the Pattern RAM. A monochrome pattern always consists of 64 bits of data regardless of the pixel color depth. The amount of color expanded data, however, is dependent on color depth. The monochrome data should be written to the address range 0x1600 - 0x1607. Note that patterns must be loaded sequentially because the lower order address bits are ignored as a pattern is loaded.				

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 F700—F7FF			*PatRamColor (256 bytes)*	
This address range allows write access to the pattern cache ram. It is only for full color patterns. A full color pattern consists of 64 bytes of data at 8 bpp, 128 bytes at 16 bpp, and 256 bytes of data at 32 bpp. Full color patterns should be loaded starting at address 0x1700. Patterns must be loaded sequentially because the lower order address bits are ignored as a pattern is loaded.				

2D DRAWING ENGINE REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x05 0000—FFFF			*Host Data (64 kB - Memory Space)*	
This address space receives host data for BLTs that require host data. All addresses in this range map to the host write buffer. It allows the host to use REP MOVSD instructions to efficiently copy data from system memory to the BLT engine. When doing a BLT that requires host data (host to screen), it is necessary to program SRC[1:0] in the BltCtl register to select host data. This register is byte accessible, but... Host-to-Screen BLTs always require an integer number of DWORDs to be transferred from the host. Although the HostData port will accept byte writes to load individual bytes of data, writing the high byte actually transfers the host data into the Engine. Reads from this register return unknown data but do not hang the bus. Writing excess data to this register space is not recommended, but will not cause the bus to hang.				

Chapter 30: Transport Stream Network

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

30.1 Introduction

The Transport Stream IO (TSIO) router in the PNX8526 is responsible for interfacing to external transport stream source chips and connecting external and internal transport stream sources, like the MPEG System Processors (MSP) and the IEEE 1394 link interfaces, with external and internal transport stream destinations.

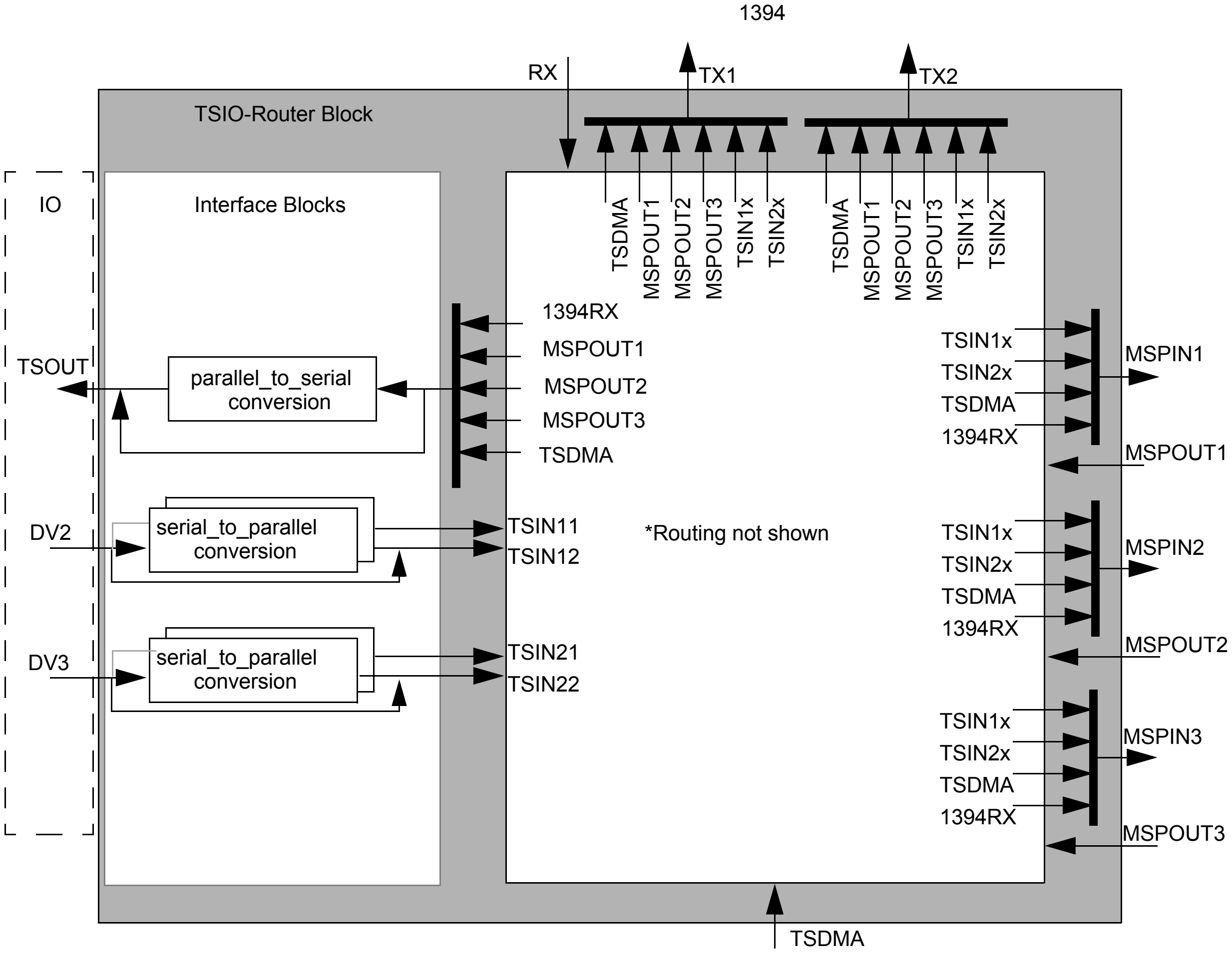


Figure 1: Transport Stream Network Block Diagram

The TSIO control register and other multiplexing information are in the Global 2 register space (see the PNX8526 Register Summary List, Global 2 Registers for additional information at offset 0x04 D600: IO_MUX_CTRL).

30.2 TS Input Interface

The TS Input Interface supports both parallel and serial interface modes. In serial mode, the interface converts a serial into a parallel data stream. In dual serial mode, each data pin carries two transport streams. The interface DV2 has two serial interfaces each of which is mapped to the internal parallel streams TSIN11 and TSIN12, and the interface DV3 has two serial interfaces each of which is mapped to the internal parallel streams TSIN21 and TSIN22.

The following figure describes the two different modes of the DV2 interface.

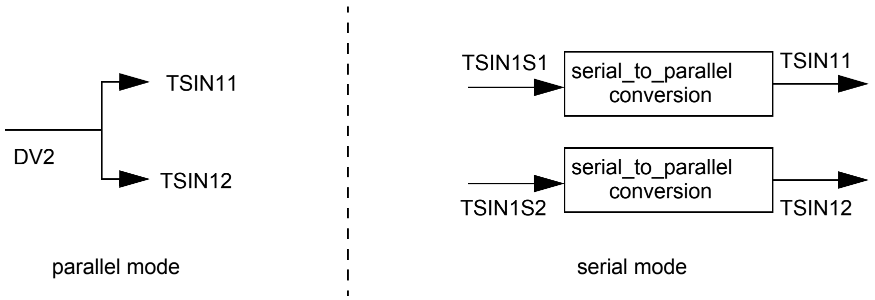


Figure 2: DV Interface Modes

The same configuration is possible for the DV3 interface.

Remark: Same data is going to both TSIN11/12 in parallel mode.

For information on parallel mode, refer to Table 17.3.1.1in Chapter 17 DV Input Ports. For information on serial mode, refer to Table 17.3.1.2 in the same chapter.

30.3 TS Input Router

The TS Input router works on 8-bit data and three side signals: a valid, an error and the sync signal.

Possible sources of TS streams are as follows:

- Two (four) TS input interfaces (TSIN 1/2) (each TSIN can work as a parallel interface,
  a single serial, or a dual serial interface)
- One 1394 receiver channel (1394 RX)
- One internal DMA agent dedicated to software generated transport streams (TSDMA)
- Three MSP feedback channels for PID filtered, optionally de-scrambled data (MSPOUT 1/3)

Possible destinations for the TS data are:

- Three internal de-scrambler/decrypt section filter data paths (MSP 1/3)
- Two 1394 transmitter channels (1394 TX 1/2)
- One TS output (TSOUT)

The router internally works on parallel data streams.

Input data in serial mode have to be converted into parallel mode in the TSIN block before they get routed. Inputs from the TSIN2 pins, for example, are finally routed to two parallel inputs of the router called TSIN21 and TSIN22. They are connected according to the register settings in Section 30.4.

Output data for serial mode have to be converted into serial mode by the TSOUT interface.

There are “serial to parallel” converters in each TS/Input paths, and there is one “parallel to serial” converter in the TSOUT path.

The clock inputs of the receiving units must be turned off while switching from one input source to another. This avoids “unpredictable behavior” caused by changing input signal clocks.

Remark: In serial mode an external sync signal is required.

30.3.1 Transport Stream Interface Timing Diagrams

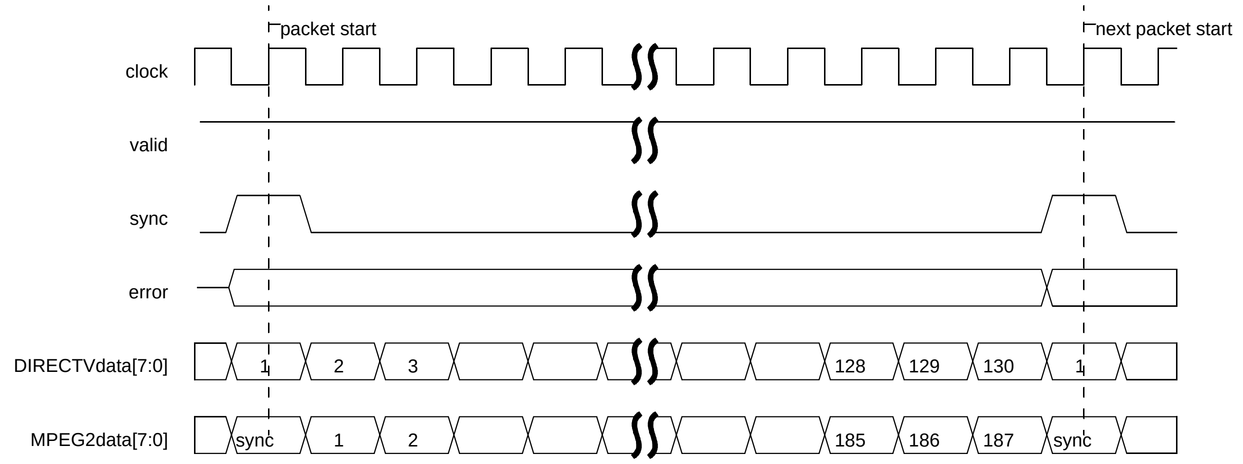


Figure 3: Input Interface Timing Diagram for Parallel Mode

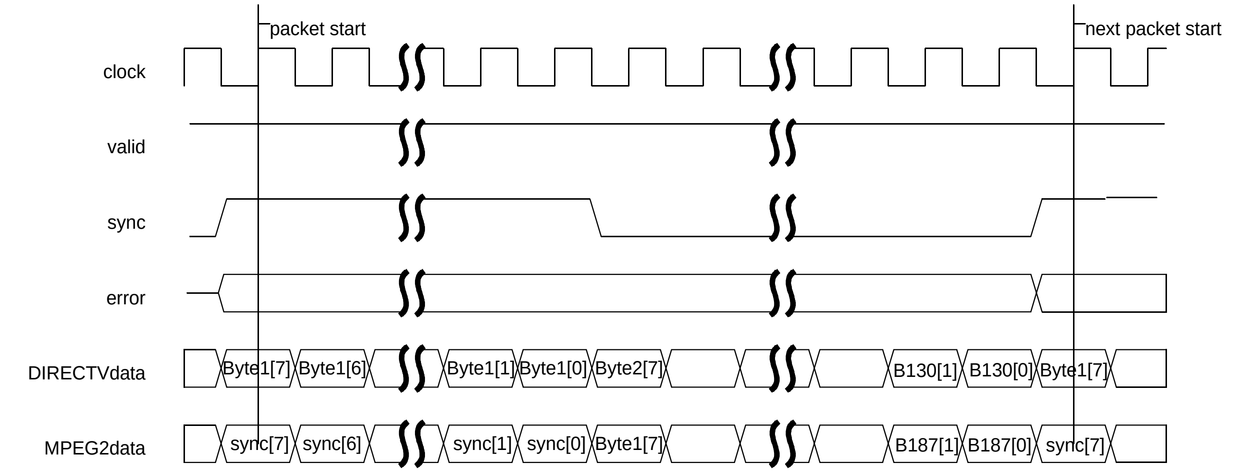


Figure 4: Input Interface Timing Diagram for Serial Mode with a Byte Style Sync

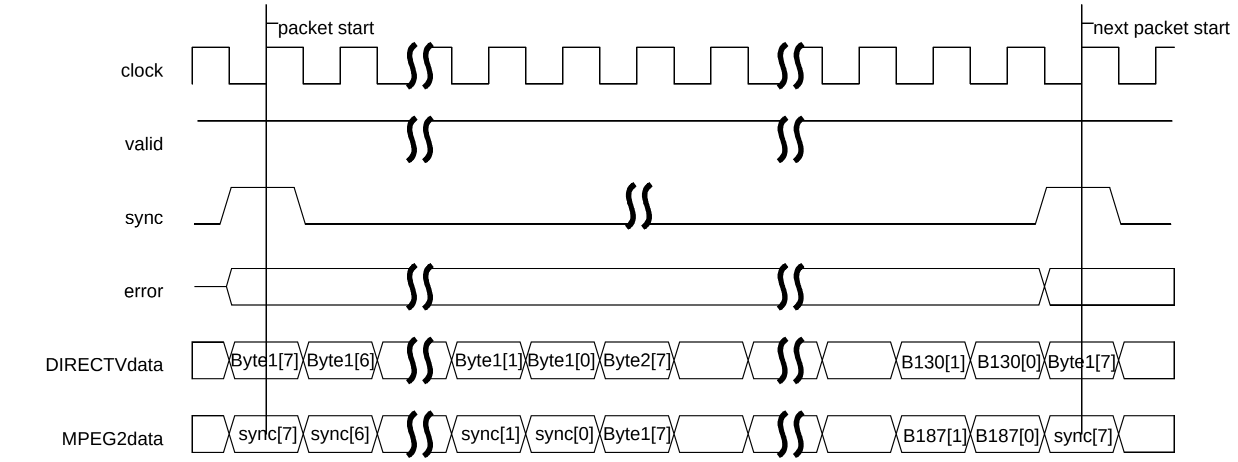


Figure 5: Input Interface Timing Diagram for Serial Mode with a Bit Style Sync

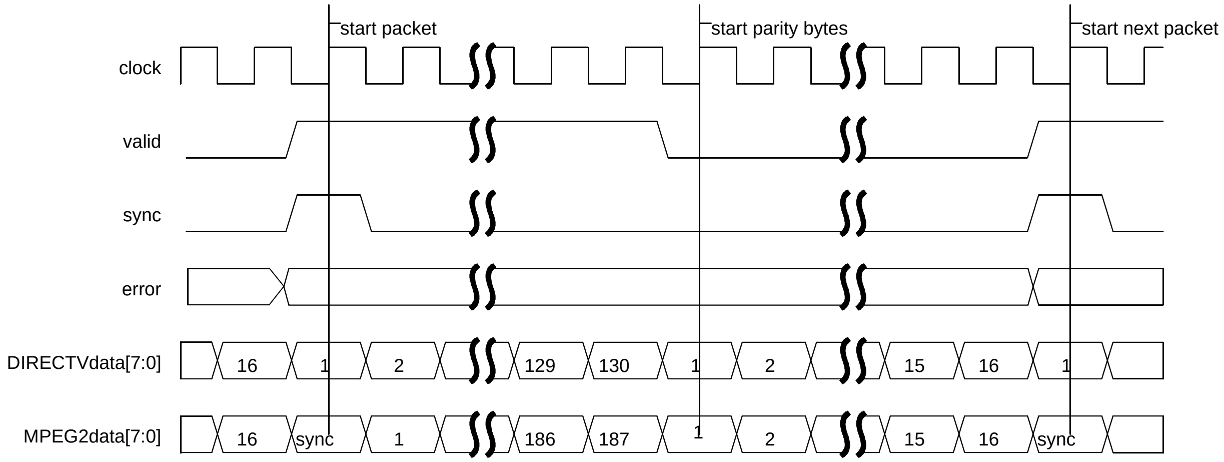


Figure 6: Input Interface Timing Diagram for Parallel Mode Including Parity Bytes of the FEC

30.4 Register Descriptions

The base address for the Transport Stream Network module is offset 0x04 D300. The TSIO control register and other multiplexing information are in the Global 2 register space (see the PNX8526 Register Summary List, Global 2 Registers for additional information at offset 0x04 D600: IO_MUX_CTRL).

TSIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 D300			*TSIO_REG*	
31:30	R	00	Reserved	
29	R/W	0	TSIN2 mode	Specifies the mode of the interface: 0 = Parallel mode TSIN2 on TSIN21 and TSIN22 1 = Serial mode, TSIN21 and TSIN22 Default = parallel mode
28	R/W	0	TSIN1 mode	Specifies the mode of the interface: 0 = Parallel mode TSIN1 on TSIN11 and TSIN12 1 = Serial mode, TSIN11 and TSIN12 Default = parallel mode
27:26	R	00	Reserved	
25	R/W	0	TSOUT_mode1	TSOUT_mode1: sync style for serial mode 0 = Byte sync 1 = Bit sync Default = byte sync
24	R/W	0	TSOUT_mode0	Specifies the mode of the interface: TSOUT_mode0: 0 = Parallel mode 1 = Serial mode Selection of serial mode enables the parallel to serial conversion. Default = parallel mode
23:20	R/W	0001	MSP3IN_route[3:0]	Specifies the input source for MSP3IN: 0000 = TSIN11 0001 = TSIN12 0010 = TSIN21 0011 = TSIN22 0100 = 1394 RX 0101 = TSDMA others = reserved Default = TSIN12
19:16	R/W	0100	TSOUT_route[3:0]	Specifies the input source for TSOUT: 0100 = 1394 RX 0101 = TSDMA 0110 = MSPOUT1 0111 = MSPOUT2 1000 = MSPOUT3 others = reserved Default = TSIN12

TSIO REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:12	R/W	0010	TX2_route[3:0]	Specifies the input source for TX2: 0000 = TSIN11 0001 = TSIN12 0010 = TSIN21 0011 = TSIN22 0101 = TSDMA 0110 = MSPOUT1 0111 = MSPOUT2 1000 = MSPOUT3 others = reserved Default = TSIN12
11:8	R/W	0000	TX1_route[3:0]	Specifies the input source for TX1: 0000 = TSIN11 0001 = TSIN12 0010 = TSIN21 0011 = TSIN22 0101 = TSDMA 0110 = MSPOUT1 0111 = MSPOUT2 1000 = MSPOUT3 others = reserved Default = TSIN12
7:4	R/W	0011	MSP2IN_route[3:0]	Specifies the input source for MSP2IN: 0000 = TSIN11 0001 = TSIN12 0010 = TSIN21 0011 = TSIN22 0100 = 1394 RX 0101 = TSDMA others = reserved Default = TSIN12
3:0	R/W	0001	MSP1IN_route[3:0]	Specifies the input source for MSP1IN: 0000 = TSIN11 0001 = TSIN12 0010 = TSIN21 0011 = TSIN22 0100 = 1394 RX 0101 = TSDMA others = reserved Default = TSIN12

Chapter 31: Transport Stream Direct Memory Access

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

31.1 Introduction

The Transport Stream Direct Memory Access (TSDMA) module reads packets from memory and generates an 8-bit parallel bitstream.

The TSDMA module is connected to the PI-Bus interface and is a read master on this bus.

The 8-bit parallel data stream of the TSDMA module is fed into the internal TS Network and used as an input to any destination of the TS Network, like the 1394 TX AV channels or the TSOUT interface. (Refer to Chapter 1 Functional Specification.)

The TSDMA can handle packet sizes up to 2047 bytes. It either transfers each packet immediately after fetching it from memory or uses an optional packet header containing a 31-bit timestamp to schedule the delivery of each packet.

The overall bandwidth for the data transmission cannot exceed 3 MB/s (24 Mbits/s).

The frequency of the transport stream interface is selected in the clocks module. The optional clock sources for this clock are either a fully programmable synthesizer, a subdivided clock from a PLL or one of the input clocks of the transport stream input interfaces.

31.2 Functional Description

The TSDMA is programmed by the CPU to read packet data from one or two DMA buffers.

Each DMA buffer has its own set of control registers for DMA_PACKET_LENGTH and DMA_SIZE.

If the TSDMA runs in header mode, each packet must start with a 32-bit HEADER. This HEADER contains a 31-bit timestamp and an error bit and is not sent or stored. It is only used to control the delivery of the packet. The timestamp in the HEADER is compared with an internal counter counting on 13.5 MHz. When the counter reaches the value specified in the timestamp, the DMA buffer starts transferring the packet data.

This time counter is a copy of the master time counter in the GPIO module. Its value can be monitored in the GPIO.

In no-header mode, the wait time packets match the delivery offset value (in tsdma clock cycles).

The TSDMA generates a valid bitstream out of the packet data using an external applied clock as the interface clock.

If both DMA channels are enabled and the DMA_LOOP option is set, the DMA module automatically switches to the other DMA buffers when finishing the first one. The status of each DMA channel is flagged in the TSDMA_IR_STATUS register:

- DMA Active Status indicates a channel reading a buffer.
- DMA Threshold Status is an interrupt that is set when a certain number of packets remains in the DMA buffer
- DMA End Status is an interrupt that is set when a buffer transfer is complete.

31.2.1 Output Signal Descriptions

The output of the TSDMA module is an input to the TS Network module.

The output format is the common byte parallel stream format including a valid, an error and a sync signal. These signals are active high and relative to the rising edge of the interface clock, which is selected in the clocks module.

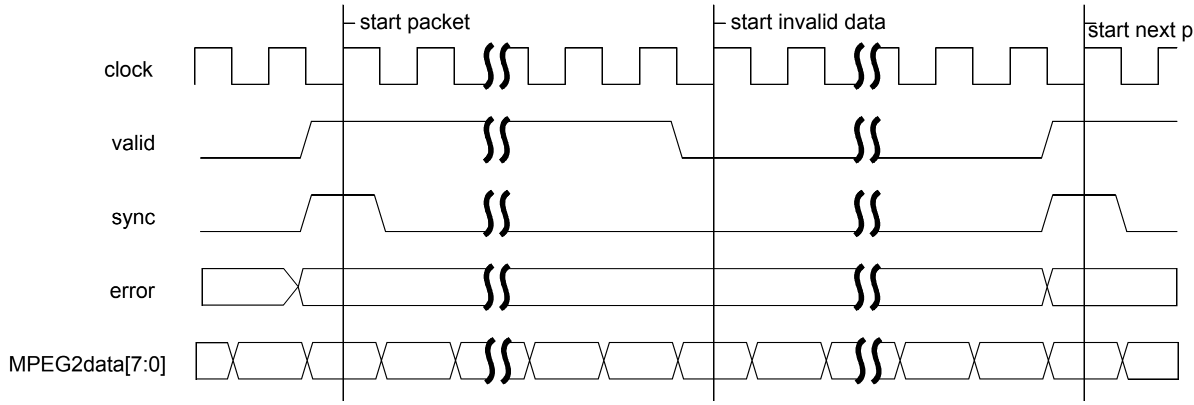


Figure 1: Signal Timing

There is an additional end of packet signal (eop not shown in the figure), indicating the last byte of a packet if the packet length is not a multiple of four. This signal is only connected to the IEEE 1394 link core.

31.3 Operation

31.3.1 DMA Operation

Software can program one or two DMA channels in the TSDMA by setting their values for DMA_START_ADDRESS, DMA_THRESHOLD, DMA_PACKET_LENGTH, DMA_SIZE DMA_DELIVERY_OFFSET.

The DMA_PACKET_LENGTH is the length of a packet in bytes (not including the HEADER in header mode or the stuffing bytes).

The DMA_SIZE of a buffer specifies the amount of packet data in 32-bit words which is DMA_PACKET_LENGTH (plus stuffing bytes) divided by four and multiplied by the number of packets.

DMA_START_ADDRESS must be 32-byte aligned (the lower two address bits are ignored).

31.3.1.1 Memory Data Formats

There are two different modes of operation which require two different memory formats: The header mode requires a HEADER at the beginning of each packet. The non-header mode must not contain a HEADER.

Table 1: Memory Data Format for TSDMA Data

Non-Header Mode Byte Address	Header Mode Byte Address	Data	
	A - A+3	Optional HEADER: ERROR_FLAG, timestamp[30:0]	
A - A+3	A+4	First byte of “A” packet (for this byte the sync signal will be created)	
A+4	A+5	Second byte of “A” packet	
	...		
A+packet_length-1	A+packet_length-1	Last byte of “A” packet	
A+packet_length	A+packet_length	Optional stuffing byte 0	Note: If the packet length is not a multiple of four, the packet has to be padded so the following packet starts at a 32-bit boundary.
A+packet_length +1	A+packet_length +1	Optional stuffing byte 1	
A+packet_length + 2	A+packet_length + 2	Optional stuffing byte 2	
	B - B+3	Optional HEADER: ERROR_PIN_FLAG, timestamp[30:0]	
B - B+3	B+4	First byte of “B” packet (for this byte the sync signal will be created)	

31.3.1.2 DMA Channel Control

When a DMA channel is enabled (DMA_ENABLE = 1), the TSDMA module reads packets from memory until it has completely read the external DMA buffer or it gets disabled during transfer. The DMA_END flag is asserted when the DMA_SIZE from the active DMA channel is reached and when there are no unread data left in the DMA buffer.

After being idle and if both channels are enabled at the same time, DMA1 will be used first.

While streaming a packet out, the TSDMA module sets the DMA_ACTIVE flags in TSDMA_IR_STATUS. The DMA_ACTIVE flag stays active until the last byte of a packet is transferred.

When a DMA channel has reached the DMA_THRESHOLD, the DMA_THRES flag in TSDMA_STATUS is asserted. The DMA_THRESHOLD specifies the number of 32-bit data remaining in the buffer before the DMA_THRES flag is set.

If both DMA1 and DMA2 get enabled simultaneously and DMA_LOOP is not enabled, TSDMA starts with DMA1 and continues to read from DMA2 after DMA1 is done. After DMA2 has finished TSDMA will stop if the DMA1_END flag is not cleared in TSDMA_IR_CLR. If DMA1_END is cleared TSDMA will start over with DMA1 again. The same for the reverse situations: the DMA2_END bit has to be cleared before it starts again after DMA1 has finished.

If both DMA channels are enabled and DMA_LOOP is set, the TSDMA block switches from one DMA to the other DMA and continues reading the new DMA buffer. When DMA2 is finished, the TSDMA will return to DMA1 and automatically reset its end flag.

If only DMA1 or DMA2 is enabled and DMA_LOOP is set, TSDMA starts with the enabled DMA and jumps back to the beginning after it has finished. DMA_LOOP needs to be cleared to stop the TSDMA.

Disabling a channel stops all current transfers and sets the DMA channel to idle.

31.3.1.3 Packet Delivery Schedule

In header mode, the timestamp (bits [30:0] of the HEADER), is fed to the transport stream control unit to schedule the packet’s delivery. The timestamp of the HEADER is a 31-bit value which is compared with the local copy of the master time counter (See Chapter 10 GPIO/IR for more information). If the value of the local time counter reached the “HEADER value plus programmable DELIVERY_OFFSET of the active channel,” the TSDMA module starts sending the packet.

From the time when the timestamp is evaluated, a timeout counter starts counting. The value of this timeout counter is compared with the programmable timeout value (TSDMA_TIMEOUT_VALUE). If the timeout value is reached and the transport stream control unit has not started transferring the packet, the DMA_timeout flag is set. This flag is used to prevent stalling of the bitstream transfer due to improper timestamp generation. The duration of the timeout is defined by the following equation:

timeout_time = TSDMA_TIMEOUT_VALUE * 1/TSTAMP_clk

The TSTAMP_clk is the clock frequency set for all global timestamping units in the clocks module. (See Chapter 5 Clock Reset and Power Management.)

In non-header mode, the delivery schedule is done with the DELIVERY_OFFSET value of the channel. In this mode, the offset defines the number of clock cycles before the next packet is sent. An offset value of zero corresponds to a delay of one cycle (TSDMA clock cycle) between the last byte of a packet and the first byte of the next packet.

31.3.2 Stream Control

In header-mode, the evaluation of the ERROR_FLAG (bit 31 of the HEADER) selects the error bit for the following packet data. The beginning of a packet is always the first data of a module. This is used to generate the side signals of the transport stream packet protocol including error, sync and valid.

The ERROR_FLAG is evaluated for each packet in the DMA buffer. If the error_signal_enable is set, the packet gets transferred and the error signal for this packet is set for the entire duration of a packet on the data bus. Otherwise the packet associated to the ERROR_FLAG is discarded.

Remark: Normally there should be no unwanted erroneous packets in the memory since it is already parsed for this flag in the MSP. The ERROR_FLAG makes sense only in debug mode for the MSP—the other destinations connected to the TS Network do not support error signals.

The sync signal is active high for the first byte of a packet on the bus.

The valid signal validates data on the bus (active high).

There is an additional end of packet signal (eop not shown in the figure), indicating the last byte of a packet if the packet length is not a multiple of four. This signal is only connected to the IEEE 1394 link core.

31.4 Register Summary and Descriptions

The base address of the TSDMA module in the PNX8526 begins at offset 0x11 6000.

Table 2: TSDMA Module Register Summary

Offset	Name	Description
0x11 6000	TSDMA_CTRL	Control bits for TSDMA
0x11 6004	TSDMA1_START_ADDRESS	Start address of DMA1
0x11 6008	TSDMA1_PACKET_LENGTH	Packet length of DMA1
0x11 600C	TSDMA1_SIZE	DMA size of DMA1
0x11 6010	TSDMA1_THRESHOLD	Threshold of DMA1
0x11 6014	TSDMA1_DELIVERY_OFFSET	Delivery offset of DMA1
0x11 6018	TSDMA2_START_ADDRESS	Start address of DMA2
0x11 601C	TSDMA2_PACKET_LENGTH	Packet length of DMA2
0x11 6020	TSDMA2_SIZE	DMA size of DMA2
0x11 6024	TSDMA2_THRESHOLD	Threshold of DMA2
0x11 6028	TSDMA2_DELIVERY_OFFSET	Delivery offset of DMA2
0x11 602C	TSDMA_TIMEOUT_VALUE	Timeout value
0x11 6030	TSDMA_TSTAMP_SEL	Timestamp select
0x11 6034	TSDMA_DMA_COUNT	Current DMA count
0x11 6038—0x11 6FDC	Reserved	-
0x11 6FE0	TSDMA_INT_STATUS	Interrupt Status
0x11 6FE4	TSDMA_INT_EN	Interrupt enable
0x11 6FE8	TSDMA_INT_CLR	Interrupt clear
0x11 6FEC	TSDMA_INT_SET	Interrupt set
0x11 6FF0	Reserved	-

Table 2: TSDMA Module Register Summary *…Continued*

Offset	Name	Description
0x11 6FF4	Powerdown	Powerdown signal
0x11 6FF8	Reserved	-
0x11 6FFC	TSDMA_MOD_ID	Module ID

TSDMA REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 6000			*TSDMA_CTRL*	
31:18		-	Unused	Read returns 00.
17	R/W	0	tsdma_eop2_en	Enable to route the eop signal to the tx2 interface (of the TS Network).
16	R/W	0	tsdma_eop1_en	Enable to route the eop signal to the tx1 interface (of the TS Network).
15:14		-	Unused	Read returns 00.
13	R/W	0	tsdma_header2_dis	Disables use of the header structure for DMA 2.
12	R/W	0	tsdma_header1_dis	Disables use of the header structure for DMA 1.
11:10		-	Unused	Read returns 00.
9	R/W	0	TSDMA2_SEND_ERROR_PACKETS	0 = Erroneous packets are not sent. 1 = Erroneous packets are sent. Using the error bit (bit 31 in the HEADER), the packet is evaluated if it is valid or not. If this bit is ‘1’ the erroneous packets are sent like other packets setting the error signal. If this bit is ‘0’ erroneous packets are discarded. Default = Erroneous packets are not sent. This bit has no meaning in non-header mode.
8	R/W	0	TSDMA1_SEND_ERROR_PACKETS	0 = Erroneous packets are not sent. 1 = Erroneous packets are sent. Using the error bit (bit 31 in the HEADER) the packet is evaluated if it is valid or not. If this bit is ‘1’ the erroneous packets are sent like other packets setting the error signal. If this bit is ‘0’ erroneous packets are discarded. Default = Erroneous packets are not sent. This bit has no meaning in non-header mode.
7:5		-	Unused	Read returns 00.

TSDMA REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4	R/W	0	DMA loop	0 = No looping 1 = Looping Setting this bit will force the TSDMA to automatically restart a DMA channel. If DMA1 and DMA2 are enabled, looping will jump from DMA1 to DMA2, then back to DMA1 and so forth. If only DMA1 or DMA2 is enabled, looping will start with the enabled DMA again after it is done. If looping is disabled and DMA1 and DMA2 are enabled, first DMA1 will be finished, then DMA2, and then the TSDMA stops (unless DMA_END event was cleared by software). If looping is disabled and either DMA1 or DMA2 is enabled, the enabled DMA channel will be finished and then TSDMA stops (unless DMA_END event was cleared by software). Default = No looping.
3:2		-	Unused	Read returns 00s
1	R/W	0	TSDMA2_ENABLE	0 = Disable 1 = Enable This bit enables the DMA channel 2. The DMA controller should be enabled after all other settings are done for this channel. Default = Disable.
0	R/W	0	TSDMA1_ENABLE	0 = Disable 1 = Enable This bit enables the DMA channel 1. The DMA controller should be enabled after all other settings are done for this channel. Default = Disable.
TSDMA Read Channel 1 Setup				
Offset 0x11 6004			***TSDMA1_START_ADDRESS***	
31:2		0	TSDMA1_START_ ADDRESS	Start address for the DMA read buffer for DMA channel 1.
1:0		-	Unused	Read returns 00.
Offset 0x11 6008			***TSDMA1_PACKET_LENGTH***	
31:11		-	Unused	Read returns 00.
10:0	R/W	0	TSDMA1_PACKET_ LENGTH	This register defines the length of one packet in bytes. The specified length does not include the HEADER or padding bytes. The maximum packet length is 2047 bytes = 0x1FF.
Offset 0x11 600C			***TSDMA1_SIZE***	
31:18		-	Unused	Read returns 00.
17:0	R/W	0	TSDMA1_SIZE	TSDMA1_SIZE register defines size of the DMA buffer in 32-bit words. DMA_SIZE =[(PACKET_LENGTH + HEADER + PADDING BYTES) * Number of Packets]/4 with HEADER being 4 in header mode and 0 in non-header mode
Offset 0x11 6010			***TSDMA1_THRESHOLD***	
31:18		-	Unused	Read returns 00.

TSDMA REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17:0	R/W	0	TSDMA1_THRESHOLD	The threshold is the number of 32-bit words which must remain unsent when the threshold flag is raised.
Offset 0x11 6014			***TSDMA1_DELIVERY OFFSET***	
31:0	R/W	0	TSDMA1_DELIVERY_ OFFSET	In header mode: delivery offset is added to the timestamp of the packet structure for delivery schedule. In non-header mode: delivery offset is used as a delay for the delivery of the next packet.
TSDMA Read Channel 2 Setup				
Offset 0x11 6018			***TSDMA2_START_ADDRESS***	
31:2	R/W	0	TSDMA2_START_ADDRES S	Same as for TSDMA1
1:0		-	Unused	Read returns 00.
Offset 0x11 601C			***TSDMA2_PACKET_LENGTH***	
31:11		-	Unused	Read returns 00.
10:0	R/W	0	TSDMA2_PACKET_ LENGTH	Same as for TSDMA1
Offset 0x11 6020			***TSDMA2_SIZE***	
31:18		-	Unused	Read returns 00.
17:0	R/W	0	TSDMA2_SIZE	Same as for TSDMA1
Offset 0x11 6024			***TSDMA2_THRESHOLD***	
31:18		-	Unused	Read returns 00.
17:0	R/W	0	TSDMA2_THRESHOLD	Same as for TSDMA1
Offset 0x11 6028			***TSDMA2_DELIVERY_OFFSET***	
31:0	R/W	0	TSDMA2_DELIVERY_ OFFSET	Same as for TSDMA1
Offset 0x11 602C			***TSDMA_TIMEOUT_VALUE***	
31:16		-	Unused	Read returns 00.
15:0	R/W	0	TSDMA_TIMEOUT_ VALUE	Relevant only in header mode. The TSDMA_TIMEOUT_VALUE is used to check if a packet is sent within a certain time after evaluating its timestamp. While the TSDMA module evaluates the timestamp to schedule the delivery of the packet, a timeout counter is running. The value of this counter is compared against this TSDMA_TIMEOUT_VALUE. If a packet is not sent after the counter reaches TSDMA_TIMEOUT_VALUE, a timeout flag is raised. The duration of the timeout is defined by the following equation: timeout-time =TSDMA_TIMEOUT_VALUE * 1/TSTAMP_clk. The TSTAMP_clk is the clock frequency set in the clocks module. Refer to Section 31.3.1.3 on page 31-665.
Offset 0x11 6030			***TSDMA_TSTAMP_SEL***	
31:19		-	Unused	Read returns 00.

TSDMA REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
18:16	R/W	0	TSDMA_TSTAMP_SEL2	Selects the source of the signal to be timestamped in the GPIO module with TSDMA2: 0 = irx_fsync 1 = irx_sysync 2 = irx_seq_err 3 = irx_emi [0] 4 = irx_emi [1] 5 = tsdma_sync 6 = tsdma_last_byte 7 = wait Notes: All irx signals are outputs of the IEEE 1394 link core. The tsdma_last_byte signals the last byte of a packet being transferred. Wait is set when TSDMA is idling because it gets no data from the bus.
15:3		-	Unused	Read returns 00.
2:0	R/W	0	TSDMA_TSTAMP_SEL1	Selects the source of the signal to be timestamped in the GPIO module with TSDMA1: 0 = irx_fsync 1 = irx_sysync 2 = irx_seq_err 3 = irx_emi [0] 4 = irx_emi [1] 5 = tsdma_sync 6 = tsdma_last_byte 7 = wait Notes: All irx signals are outputs of the IEEE 1394 link core. The tsdma_last_byte signals the last byte of a packet being transferred. Wait is set when TSDMA is idling because it gets no data from the bus.
Offset 0x11 6034			***TSDMA_DMA_COUNT***	
31:18		-	Unused	
17:0	R	0	TSDMA_DMA_COUNT	This register gives the current value of the DMA COUNT. It only gives a stable value if the DMA is stopped.
Offset 0x11 6038—6FDC			***Reserved***	

TSDMA REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 6FE0			*TSDMA_INT_STATUS*	
31:9		-	Unused	Returns 0.
8	R	0	TSDMA_ACTIVE	This flag indicates an active TSDMA.
7	R	0	TSDMA2_ACTIVE	This flag indicates an active channel DMA 2.
6	R	0	TSDMA1_ACTIVE	This flag indicates an active channel DMA 1.
5	R	0	TSDMA_STALL	Indicates the TSDMA is stalling because the system cannot deliver data (DMA read underflow).
4	R	0	TSDMA_TIMEOUT	This interrupt indicates the timeout condition as specified for the TSDMA_TIMEOUT_VALUE.
3	R	0	TSDMA2_THRES	This interrupt indicates that the DMA threshold is reached for channel2.
2	R	0	TSDMA1_THRES	This interrupt indicates that the DMA threshold is reached for channel1.
1	R	0	TSDMA2_END	This interrupt indicates the end of DMA channel 2.
0	R	0	TSDMA1_END	This interrupt indicates the end of DMA channel 1.
Offset 0x11 6FE4			*TSDMA_INT_EN*	
31:6		-	Unused	
5	R/W	0	TSDMA_STALL	Each of these bits enables generation of an interrupt by the appropriate bit in the TSDMA_INT_STATUS register.
4	R/W	0	TSDMA_TIMEOUT	
3	R/W	0	TSDMA2_THRES	
2	R/W	0	TSDMA1_THRES	
1	R/W	0	TSDMA2_END	
0	R/W	0	TSDMA1_END	
Offset 0x11 6FE8			*TSDMA_INT_CLR*	
31:6		-	Unused	
5	W	0	TSDMA_STALL	Each of these bits clears the interrupt bit in the status register TSDMA_INT_STATUS.
4	W	0	TSDMA_TIMEOUT	
3	W	0	TSDMA2_THRES	
2	W	0	TSDMA1_THRES	
1	W	0	TSDMA2_END	
0	W	0	TSDMA1_END	
Offset 0x11 6FEC			*TSDMA_INT_SET*	
31:6		-	Unused	

TSDMA REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
5	W	0	TSDMA_STALL	Each of these bits sets the interrupt bit in the status register TSDMA_INT_STATUS.
4	W	0	TSDMA_TIMEOUT	
3	W	0	TSDMA2_THRES	
2	W	0	TSDMA1_THRES	
1	W	0	TSDMA2_END	
0	W	0	TSDMA1_END	
Offset 0x11 6FF0			*Reserved*	
Offset 0x11 6FF4			*Power Down*	
31	R/W	0	Powerdown	TSDMA Powerdown indicator. 1 = Powerdown 0 = Power up When this bit equals 1, no other registers are accessible.
30:0		-	Unused	
Offset 0x11 6FF8			*Reserved*	
Offset 0x11 6FFC			*TSDMA_MOD_ID*	
31:16	R	0x0125	TSDMA_MOD_ID	This is the module ID for the TSDMA module.
15:12	R	1	MAJREV	Major Revision
11:8	R	0	MINREV	Minor Revision
7:0	R	0	MODULE APERTURE SIZE	Encoded as: Aperture size = 4 k*(bit_value+1), so 0 means 4 kB (the default).

Chapter 32: DMA Controller and CRC Checker

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

32.1 Introduction

The STB_DMA Controller is a DMA Controller and Cyclic Redundancy Check (CRC) coprocessor that is connected to the PNX8526 internal PI-Bus. The DMA Controller supports four independent DMA channels.

The DMA Controller has four main components, as shown in Figure 1:

- DMA and CRC Data Path: The DMA Controller has a 16-byte buffer to do a buffered mode DMA. During every DMA transaction, the DMA Controller reads the DMA data, realigns it to the destination address and then writes it back. The same DMA data path is used to compute CRC.
- DMA Engine Core: The DMA Engine Core performs a single DMA transfer until the count is done or a throttle is reached. It manages the data path and the CRC operation. It reports the status of the DMA transfer back to the DMA Control Engine.
- DMA Scatter-Gather and Control Engine: This portion of the DMA Controller arbitrates among the four channels and manages all scatter-gather operations. It then issues the selected DMA transfer to the DMA Engine Core for execution. This engine is controlled via the DMA Controller Configuration Registers.
- Configuration Registers: The operation of the DMA Controller is determined by the DMA Controller Configuration Registers. The operation of each channel can be controlled separately.

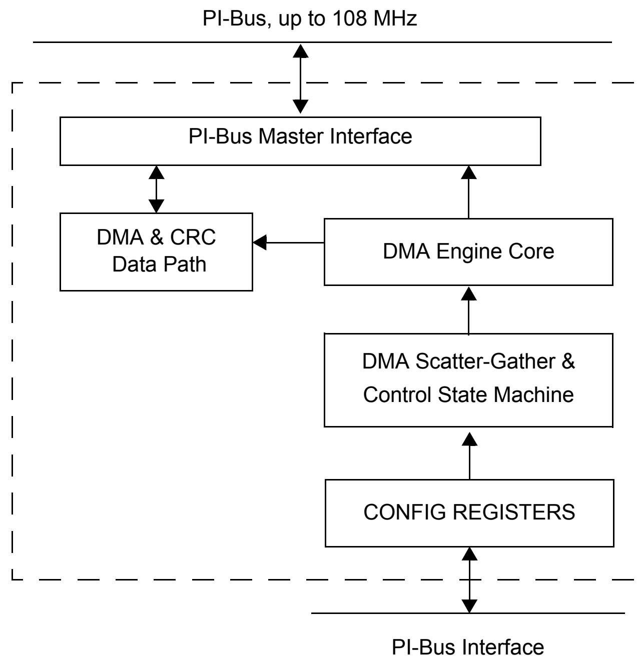


Figure 1: STB_DMA Controller Block Diagram

32.1.1 Features

The STB_DMA Controller has the following features:

- PI System Bus DMA Controller
- Support for four independent DMA channels
- Buffered Mode DMA with a 16-byte data buffer
- Support for both Demand Mode and Scatter-Gather Mode DMA
- Configurable throttle support with throttle sizes ranging from 16 bytes to 112 bytes
- CRC support (DMA Controller can also be used to perform CRC)

32.2 Functional Description

32.2.1 DMA Engine Core

The DMA Engine core transfers data from a source region to a destination region. These two regions can reside anywhere within the PNX8526.

Transfer Size

The DMA Engine Core can perform a single transfer of 1 to 65 535 (64 kB-1) bytes in size. Longer transfers will have to be split into separate DMA transactions. The transfer must not cross a 32-MB boundary. If a DMA transfer needs to be longer than 64 kB or needs to cross a 32-MB address boundary, the software will have to break the transfer into two or more commands to the DMA Controller, each following the above rules.

Source and Destination Location

The source and destination must lie within the PNX8526 address map. The internal PI address decoder performs the address decoding and routes the cycle to the corresponding device.

Throttle

The DMA Engine Core will transfer data in approximately 16-byte chunks. At every 16-byte boundary, the DMA Engine checks to see if a throttle limit was reached. If it was reached, the DMA Engine Core stops the current transfer and reports the status back to the main DMA Control Engine. If it wasn’t reached, the DMA Engine Core will continue to transfer another 16 bytes. The throttle limit can be set in multiples of 16 bytes from 16 bytes to 128 bytes. If the throttle is not set, the transfer will go on until all the bytes have been transferred.

Transfers on PI-Bus

The STB_DMA will try to align all read and write requests to perform aligned 64-byte requests whenever possible. Transfers that are not 64 bytes are only done for alignment to reach a 64-byte boundary.

Remark: The DMA Controller can manage address alignment between the source and destination. However, for optimal DMA performance both the source and destination must be aligned.

32.2.1.1 The DMA Engine Core Operation

The DMA Scatter-Gather and Control Engine arbitrates among the four channels and issues the selected transfer to the DMA Engine Core. The DMA Engine Core picks up the source and destination address, the count of the bytes to be transferred, and the transfer attributes. The DMA Engine core then transfers 16 bytes or up to a 16-byte boundary. It first performs the read, collects the data in the buffer, and then performs the write. When this mini-transfer is done, the core updates the current source, destination address and the current count in the configuration registers.

The DMA Engine Core then checks if the throttle is reached. If it has been reached, the transfer is terminated and control is returned to the DMA Control Engine. If the throttle hasn’t been reached, the transfer continues until all the requested bytes have been transferred or the throttle limit is reached. When the count reaches 0 or the throttle has been reached, the transfer is stopped and all the status information is relayed back to the configuration registers.

32.2.2 DMA and CRC Data Path

The STB_DMA has a 16-byte data buffer that is used to buffer data. (This DMA Controller is a buffered mode engine).

During DMA transfers, the DMA Controller reads and writes data in 16-byte chunks or less. This 16-byte data is first read in and stored in the data buffer, aligned to the destination address. During the write operation, this aligned data is written out. Software alignment is not necessary; it is managed by the data buffer. The data buffer supports both big and little-endian modes.

The DMA Controller has a byte-wide CRC Engine which operates on the write byte data during the write operation. The CRC vector can be initialized by the Host CPU via a configuration register. The DMA Data Path keeps computing the CRC during each byte write (when enabled), and the final result can be read back from the same configuration register.

CRC cannot be performed while DMA data is being moved within memory. The data to be CRC’d must be collected in memory and the CRC operation fired off. During CRC operations, the DMA Controller goes through the motions of performing a read and write (like a regular DMA), but it does not actually write the data. It uses this data for CRC only. The CRC engine is only effective during byte-write operations to the destination address, so the destination must be programmed as an 8-bit wide I/O device.

32.2.3 DMA Scatter-Gather and Control Engine

The DMA Scatter-Gather and Control Engine is the back-end “brain” of the STB_DMA Engine Controller. The DMA controller register file generates a request after collection from either the processor enabling a DMA operation or a descriptor being loaded from memory.

The DMA Control Engine arbitrates between these requests and issues a single transaction to the DMA Engine Core. It manages Scatter-Gather DMA by reading preset descriptors from memory and executing them until it encounters a STOP command.

32.3 DMA Transfer Operation

There are two modes of initiating a DMA transfer: the Demand Transfer mode and the Scatter-Gather mode.

- The Demand Transfer Mode (DM) is the basic method of performing a DMA. In the Demand Transfer mode, all source and destination attributes are programmed in registers within the DMA Controller and a DMA transfer is initiated.
- The Scatter-Gather Mode (SGM) is a more complex, descriptor-based method of performing a DMA, wherein the descriptors are prepared in advance and stored in memory. The DMA Controller fetches the descriptors from memory and executes them. This mode is used to initialize and perform multiple DMAs together.

32.3.1 Demand Mode DMA

Demand Mode transfers are initiated by the host by programming the entire DMA Descriptor in the corresponding channel DMA registers. When all information has been programmed, the DMA is enabled by writing a ‘1’ to the CHEN bit. The proper sequence for programming a DMA is to write the registers in the following order: SRC_ADDR, DST_ADDR and finally the CTL_CNT. The CHEN bit in the CTL_CNT must be set to a “1” and the LLEN bit must be set to a “0.”

Remark: All commands are listed in the DMACMD bit field description of the CTL_CNT0 register.

The DMA Controller then triggers the demand mode transfer. When the transfer is complete, an indication is returned in two ways:

- The corresponding DMACHBUSY bit in the DMA_STATUS register will return to “0” and an interrupt will be raised (if enabled).
- If the CPU intends to perform the DMA in the polled mode, then it must poll either the corresponding DMACHBUSY bit in the DMA_STATUS register or the STB_DMAINTST bit in the Interrupt Status register.

When the DMA is running, the CPU may want to stop the DMA transfer either temporarily or permanently. The DMA can be stopped by writing a “0” to the corresponding DMACHEN bit in the DMA_CONTROL register. DMA can later be restarted by turning this same bit on.

Remark: The host processor must not change either the CHEN or the LLEN bits while the DMA is running. Any necessary control must be done with the DMACHEN bit.

32.3.2 Scatter-Gather DMA

For performing a scatter-gather DMA, the processor must prepare a complete list of descriptors in memory, each descriptor being identical in configuration to the descriptor registers. Each descriptor must be placed at a 16-byte aligned address and descriptors in memory must be in consecutive addresses. All descriptors in memory, however, must have both the CHEN and LLEN bits set to a “1.” Each descriptor contains a command, usually a combination of data MOVE, CRC, JUMP and STOP. (All commands are listed in the DMACMD bit field description in the CTL_CNT0 register.) The DMA Controller will process descriptors until it encounters a descriptor with a STOP command.

Once the descriptor table in memory is initialized, the host processor proceeds to write the address of the first descriptor entry into the LL_PTR register. It will then write a “1” to the corresponding channel descriptor register bit LLEN (none of the other values in the current channel descriptor registers need to be valid). The LLEN bit being a “1” will automatically trigger the Scatter-Gather DMA operation, provided the corresponding DMA Channel is enabled in the DMA_CONTROL register. When a “1” is being written into the LLEN bit, none of the other bits in that register is overwritten.

The DMA Controller will first “fetch” the DMA descriptor in memory pointed to by the Link List register. This descriptor is loaded in the corresponding channel descriptor registers (and can be read for monitoring by the processor). Upon loading the descriptor, it is “executed.” The execution is similar in operation to the demand mode

DMA. When this execution is complete, the DMA Controller updates the LL_PTR register. For a regular command, the LL_PTR register is automatically incremented by 16. For a JUMP command, the LL_PTR register is loaded with the DST_ADDR field in the descriptor. For a STOP command, the LL_PTR register is not changed. (All commands are listed in the DMACMD bit field description in the CTL_CNT0 register.)

The DMA then fetches the next command descriptor from memory (using the updated Link List Pointer) and executes it. The DMA Controller will continue until it encounters a STOP command. On encountering a STOP, it will not increment the Link List Pointer and continue pointing at the current descriptor location.

When the DMA is running, the CPU may want to stop the DMA transfer either temporarily or permanently. The processor can be stopped by writing a “0” to the corresponding DMACHEN bit in the DMA_CONTROL register. DMA execution can later be restarted by turning this same bit on.

Remark: The host processor must not change either the CHEN or the LLEN bits while the DMA is running. Any desired control must be done with the DMACHEN bit.

32.3.2.1 Scatter-Gather Mode Retriggering

When a DMA channel is operating in scatter-gather mode, the application might need to add more commands to the scatter-gather linked-list chain on the same channel. Normally, software will have to poll the status of the DMA channel and wait for it to finish before triggering the second DMA. The retriggering mechanism allows the application to avoid polling and waiting for the current transfer to finish.

To do this, the software application must follow the procedure listed below:

- Every set of descriptors that are stored in memory must have the last descriptor be a STOP. Do not combine a data move, jump or CRC operation in the last descriptor. Trigger the first scatter-gather DMA by writing a “1” to the LLEN bit of the corresponding channel register.
- While this DMA is going on, the application needs to append more descriptors to the same DMA channel. Create and append the new descriptors just after the last STOP command in memory. End the new descriptor set with another STOP command.
- After the new set of descriptors have been written into memory, the application must modify the last descriptor of the currently executing set from a STOP to a NOP.
- After this conversion, the application must then retrigger the DMA Controller channel by writing a “1” to the LLEN bit. If the DMA Controller is already executing the current transfer, it will save the retrigger information to be used and retrigger the DMA after it is done. If the DMA Controller was already done with execution of the old block, it will be retriggered as though it was a new scatter-gather request.

32.3.3 CRC Computation

CRC computation in the STB_DMA is done by the DMA Controller, and it must be done as a post-process. The data to be CRC'd must be stored in memory before the CRC operation can begin. Once the data is ready, the CRC operation is initiated as a DMA. The source address and byte-count must be programmed into the channel registers (or in memory as a part of a descriptor). The destination address is a dummy address location and need not be programmed. The destination must be programmed as an I/O device with a size of 8 bits. The CRC register must be initialized with an initial value.

32.4 Register Descriptions

The base address for the DMA Controller and CRC Checker registers is 0x06 2000.

32.4.1 Configuration Registers

The configuration registers take up 4 kB of PI address space. All registers are 32 bits wide and must be accessed as an atomic 32-bit entity. The read value for all reserved locations and "unused" bits must be ignored, and they must be written to zero.

The DMA Controller supports four channels. Each channel has its descriptor and LL_PTR registers. An interrupt is provided for each channel. There is also an additional global DMA control and status register to control and monitor the overall operation.

All four channels have their own channel descriptor registers and each descriptor is made up of four 4-byte registers, one of which is reserved. The descriptor has a fixed format, and this same format is also used to prepare descriptor tables in memory. Descriptors are loaded from memory into the channel descriptor registers in the Scatter-Gather mode. In the Demand mode, the descriptor registers can be loaded directly and prepared by the host processor.

The two most important bits in the Channel Control register (CHAN_CTL) are the CHEN and the LLEN bits. When either of these two bits is set, that particular channel will attempt to trigger its DMA. Before CHEN is set, the whole channel descriptor must be valid. (The LL_PTR need not be set to a valid value.) Before LLEN is set, the LL_PTR must be valid (the channel descriptor may not be set to valid values).

Software must never program both CHEN and LLEN to be a "1" at the same time. It is either CHEN or LLEN. Setting CHEN to a "1" via a host write triggers the Demand mode transfer. Setting LLEN to a "1" via a host write triggers the Scatter-Gather mode transfer.

32.4.2 Channel Link List Pointer Register

The Link List Pointer (LL_PTR) register is 4 bytes long. It contains the memory address of the next command descriptor that needs to be loaded into the channel descriptor registers in Scatter-Gather mode. This register is not used in Demand mode.

32.4.3 CRC Register

This register is the CRC accumulator. Because the CRC is a single shared register, only one of the four channels can be computing a CRC at a given time. The CRC process begins when the processor writes to this register and initializes the CRC value. CRC can be done for an entire block of data either completely or partially. (If a CRC is done partially, then the processor must store intermediate CRC accumulator results in a temporary space to be restored later.)

32.4.3.1 DMA Status Register

This is a read-only register denoting the status of the DMA Controller. Some of the bits in the status register will automatically clear on a read. Software has to copy the status and further process it to prevent loss of information.

32.4.4 Register Address Map

Table 1: DMA Controller and CRC Checker Register Summary

Offset	Name	Description
0x06 2000	CTL_CNT0	DMA Control for channel 0
0x06 2004	SRC_ADDR0	Source Address for channel 0
0x06 2008	DST_ADDR0	Destination Address for channel 0
0x06 200C	Reserved	
0x06 2010	Channel Link List Pointer0	Link List Pointer for channel 0
0x06 2014—201C	Reserved	
0x06 2020	CTL_CNT1	DMA Control for channel 1
0x06 2024	SRC_ADDR1	Source Address for channel 1
0x06 2028	DST_ADDR1	Destination Address for channel 1
0x06 202C	Reserved	
0x06 2030	Channel Link List Pointer1	Link List Pointer for channel 1
0x06 2034—203C	Reserved	
0x06 2040	CTL_CNT2	DMA Control for channel 2
0x06 2044	SRC_ADDR2	Source Address for channel 2
0x06 2048	DST_ADDR2	Destination Address for channel 2
0x06 204C	Reserved	
0x06 2050	Channel Link List Pointer2	Link List Pointer for channel 2
0x06 2054—205C	Reserved	
0x06 2060	CTL_CNT3	DMA Control for channel 3
0x06 2064	SRC_ADDR3	Source Address for channel 3
0x06 2068	DST_ADDR3	Destination Address for channel 3
0x06 206C	Reserved	
0x06 2070	Channel Link List Pointer3	Link List Pointer for channel 3
0x06 2074—207C	Reserved	
0x06 2080—20FC	Reserved	

Table 1: DMA Controller and CRC Checker Register Summary *...Continued*

Offset	Name	Description
0x06 2100	CRC	CRC Accumulator register
0x06 2104	DMA_Control	DMA Control register
0x06 2108—210C	Reserved	
0x06 2110	DMA Status Register	DMA Status Register
0x06 2114	DMA Soft Reset Register	DMA Software reset register
0x06 2118—23FC	Reserved	
0x06 2FE0	Interrupt Status Register	Interrupt Status Register
0x06 2FE4	Interrupt Enable Register	Interrupt Enable Register
0x06 2FE8	Interrupt Clear Register	Interrupt Clear Register
0x06 2FEC	Interrupt Set Register	Interrupt Set Register
0x06 2FF0	Reserved	
0x06 2FF4	Powerdown	Powerdown mode
0x06 2FF8	Reserved	
0x06 2FFC	Module ID	Module ID

DMA CONTROLLER REGISTERS				
Bits	Read/ Write	Reset Value	Name	Description
Offset 0x06 2000			*CTL_CNT0*	
31:28	R/W	0	DMACMD	DMA Command Demand Mode: 0100 = MOVE - Move Data 0101 = CRC - Compute CRC Scatter-Gather Mode: 000x = NOP - No data movement. LL_PTR = LL_PTR + 16 x01x = STOP-No data movement. LL_PTR is unchanged. STOP after the current descriptor. 0100 = MOVE - Move Data. LL_PTR = LLPTR + 16 0101 = CRC - Compute CRC. LL_PTR = LL_PTR + 16 x110 = MOVE & STOP - Move Data. LL_PTR is unchanged. STOP after the current descriptor. x111 = CRC & STOP - Compute CRC. LL_PTR is unchanged. STOP after the current descriptor. 100x = JUMP - No data movement. LLPTR = DST_ADDR All other values reserved. (x = “Don’t care.”)
27	R/W	0	LLEN	Link List Enable. (Scatter-Gather Mode (SGM) enable) When this bit is enabled, none of the other bits in this register is affected or changed. Used to support the scatter-gather retrigger mechanism. 0 = SGM transfer off 1 = Start an SGM transfer (reads “SGM transfer in progress”). When this bit is ‘1’, the LL_PTR field must be valid.

DMA CONTROLLER REGISTERS				
Bits	Read/ Write	Reset Value	Name	Description
26	R/W	0	CHEN	Channel Enable. (Demand Mode (DM) enable) 0 = DM transfer off 1 = Start a DM transfer (reads “DM transfer in progress”). When this bit is ‘1’, the whole channel descriptor must be valid. Note: This bit can be set by the MIPS CPU to start a DMA transfer by loading a descriptor from memory.
25	R/W	0	EOTINTEN	End of Transfer Interrupt Enable. Enables interrupt generation at the end of the current demand-mode transfer. 0 = Disable 1 = Enable
24:23	R/W	0	SRCTYPE	Source Type. Defines the type of the source device: 00 = Memory Device 01 = Reserved 10 = Reserved 11 = Reserved
22	R/W	0	SRCINC	Source Increment. Defines source incrementability: 0 = The Source Address cannot be incremented 1 = The Source Address can be incremented
21:20	R/W	0	DSTTYPE	Destination Type. Defines the type of the destination device: Memory or I/O device width. 00 = Memory Device 01 = 8-bit wide I/O Device (for CRC only) 10 = Reserved 11 = Reserved
19	R/W	0	DSTINC	Destination Increment. Defines destination incrementability: 0 = The Destination Address cannot be incremented. 1 = The Destination Address can be incremented.
18:16	R/W	0	THRCNT	Throttle Count. Sets the threshold limit to the DMA Engine core. 000 = Throttle Disable 001 = 16 bytes 010 = 32 bytes 011 = 48 bytes ... 111 = 112 bytes
15:0	R/W	0	CHAN_CNT	Channel Data Count field is the 16-bit DMA Transfer count. 0 = No transfer 1 to 0xFFFF = Number of bytes of data that will be transferred
Offset 0x06 2004			*SRC_ADDR0*	
31:0	R/W	0	SRC_ADDR	Physical Source Address of the DMA transaction
Offset 0x06 2008			*DST_ADDR0*	
31:0	R/W	0	DST_ADDR	Physical Destination Address of the DMA transaction. For scatter-gather DMA, a new “JUMP” address is stored in this field to reprogram the LL_PTR register.
Offset 0x06 200C			*Reserved*	

DMA CONTROLLER REGISTERS				
Bits	Read/ Write	Reset Value	Name	Description
Offset 0x06 2010			***Link List Pointer0***	
31:0	R/W	0	LL_PTR	This is the address in memory of the next command descriptor to be fetched and executed (valid only in Scatter-Gather mode).
Offset 0x06 2014 - 201C			***Reserved***	
Offset 0x06 2020			***CTL_CNT1***	
31:16	R/W	0	CHAN_CTL	Channel Control field that controls the DMA transfer attributes. Refer to the bit description of CTL_CNT0.
15:0	R/W	0	CHAN_CNT	Channel Data Count field is the 16-bit DMA transfer count.
Offset 0x06 2024			***SRC_ADDR1***	
31:0	R/W	0	SRC_ADDR	Physical Source Address of the DMA transaction
Offset 0x06 2028			***DST_ADDR1***	
31:0	R/W	0	DST_ADDR	Physical Destination Address of the DMA transaction. For scatter-gather DMA, a new “JUMP” address is stored in this field to reprogram the LL_PTR register.
Offset 0x06 202C			***Reserved***	
Offset 0x06 2030			***Channel Link List Pointer1***	
31:0	R/W	0	LL_PTR	This is the address in memory of the next descriptor to be fetched and executed (valid only in Scatter-Gather mode).
Offset 0x06 2034 - 203C			***Reserved***	
Offset 0x06 2040			***CTL_CNT2***	
31:16	R/W	0	CHAN_CTL	Channel Control field that controls the DMA transfer attributes. Refer to the bit description of CTL_CNT0.
15:0	R/W	0	CHAN_CNT	Channel Data Count field is the 16-bit DMA transfer count.
Offset 0x06 2044			***SRC_ADDR2***	
31:0	R/W	0	SRC_ADDR	Physical Source Address of the DMA transaction
Offset 0x06 2048			***DST_ADDR2***	
31:0	R/W	0	DST_ADDR	Physical Destination Address of the DMA transaction. For scatter-gather DMA, a new “JUMP” address is stored in this field to reprogram the LL_PTR register.
Offset 0x06 204C			***Reserved***	
Offset 0x06 2050			***Channel Link List Pointer2***	
31:0	R/W	0	LL_PTR	This is the address in memory of the next descriptor to be fetched and executed (valid only in the Scatter-Gather mode).
Offset 0x06 2054 - 205C			***Reserved***	
Offset 0x06 2060			***Control and Count Register3 CTL_CNT3***	
31:16	R/W	0	CHAN_CTL	Channel Control field that controls the DMA transfer attributes. Refer to the bit description of CTL_CNT0.
15:0	R/W	0	CHAN_CNT	Channel Data Count field is the 16-bit DMA transfer count.

DMA CONTROLLER REGISTERS				
Bits	Read/ Write	Reset Value	Name	Description
Offset 0x06 2064			*SRC_ADDR3*	
31:0	R/W	0	SRC_ADDR	Physical Source Address of the DMA transaction
Offset 0x06 2068			*DST_ADDR3*	
31:0	R/W	0	DST_ADDR	Physical Destination Address of the DMA transaction. For scatter-gather DMA, a new “JUMP” address is stored in this field to reprogram the LL_PTR register.
Offset 0x06 206C			*Reserved*	
Offset 0x06 2070			*Channel Link List Pointer3*	
31:0	R/W	0	LL_PTR	This is the address in memory of the next descriptor to be fetched and executed (valid only in the Scatter-Gather mode).
Offset 0x06 2074 - 207C			*Reserved - For future use (additional channels)*	
Offset 0x06 2080 - 20FC			*Reserved*	
Offset 0x06 2100			*CRC*	
31:0	R/W	0xFFFF _FFFF	CRC Registers	This register is the CRC accumulator register.
Offset 0x06 2104			*DMA_Control*	
31:9		-	Unused	
8	R/W	0	DMAGEN	DMA Controller Global Enable. Enables/Disables DMA Controller 0 = Disable the DMA Controller centrally, regardless of what other configuration bits are set. 1 = Enables the DMA Controller.
7:4		-	Unused	
3:0	R/W	0	DMACHEN [3:0]	DMA Channel Enable. Selectively enables or disables a particular DMA Channel. Bits 3:0 correspond to DMA channels 3:0. 0 = Disable channel. 1 = Enabled channel.
Offset 0x06 2108 - 210C			*Reserved*	
Offset 0x06 2110			*DMA Status Register*	
31:4		-	Unused	
3:0	R	0	DMACHBUSY	DMA Channel Busy. Denotes the busy status of the channels. Bits 3:0 correspond to DMA channels 3:0. 0 = Channel is not running. 1 = Channel is busy.
Offset 0x06 2114			*DMA Soft Reset Register*	
31:1		-	Unused	
0	W	0	STBDMA_RST	STB_DMA oft Reset Enable. This register allows software a software reset of the DMA Controller engine. 0 = No Effect. 1 = The DMA Controller is reset by software, except for all the registers and the PI-Bus Interface.

DMA CONTROLLER REGISTERS				
Bits	Read/ Write	Reset Value	Name	Description
Offset 0x06 2118 - 23FC			***Reserved***	
Offset 0x06 2FE0			***Interrupt Status Register***	
31:4		-	Unused	
3:0	R	0	STB_DMAINTST	STB_DMA Interrupt Status. This register allows software to check an interrupt status for pending interrupts. Each bit 3:0 reports the status of the corresponding channel interrupt source 3:0. 0 = Interrupt is not pending. 1 = Interrupt is pending. Writing to these bits has no effect.
Offset 0x06 2FE4			***Interrupt Enable Register***	
31:4		-	Unused	
3:0	R/W	0	STB_DMAINTENA	STB_DMA Interrupt Enable. This register allows software to selectively enable an interrupt. Each bit 3:0 controls the corresponding internal interrupt source 3:0. 0 = Interrupt is disabled. 1 = Interrupt is enabled.
Offset 0x06 2FE8			***Interrupt Clear Register***	
31:4		-	Unused	
3:0	W	0	STB_DMAINTCLR	STB_DMA Interrupt Clear. This register allows software to reset an interrupt. Each bit 3:0 controls the corresponding internal interrupt source 3:0. Ignore read data 0 = Interrupt is not cleared. 1 = Interrupt is cleared.
Offset 0x06 2FEC			***Interrupt Set Register***	
31:4		-	Unused	
3:0	W	0	STB_DMAINTSET	STB_DMA Interrupt Set. This register allows software to set an interrupt. Each bit controls each internal interrupt source. Ignore read data 0 = Interrupt is not set. 1 = Interrupt is set.
Offset 0x06 2FF0			***Reserved***	
Offset 0x06 2FF4			***POWERDOWN***	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x06 2FF8			***Reserved***	

DMA CONTROLLER REGISTERS				
Bits	Read/ Write	Reset Value	Name	Description
Offset 0x06 2FFC			*Module ID*	
31:16	R	0x0130	MODULEID	STB_DMA Module ID Returns "0x0130” Writing has no effect.
15:12	R	0	REV_MAJOR	Major revision counter
11:8	R	1	REV_MINOR	Minor revision counter
7:0	R	0	APP_SIZE	Aperture Size 0 = 4kB

Chapter 33: MPEG System Processor (MSP)

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

33.1 Introduction

The PNX8526 has three MPEG System Processors. The MPEG System Processor (MSP) is in charge of parsing an MPEG2 or DIRECTV transport stream, including de-scrambling, de-multiplexing and routing the data to memory. The MSP is compliant with DVB, DES, ICAM and MULTI2 de-scrambler standards.

The MSP receives transport streams through a versatile stream input interface, capable of handling byte-parallel streams in various formats at rates up to 81 megabits per second (Mbits/s). The stream is de-multiplexed and de-scrambling is applied to separate streams. Up to 48 individual data streams are routed to different queues in SDRAM memory for further dispatch to either the MPEG2 Video Decoder, the Audio Decoder or the MIPS CPU. The MSP provides a high level of integration of all the system level functions required in a typical set top box.

The MSP features a fully customized Transport RISC microprocessor dedicated to the transport/de-multiplexing function. The Transport RISC microprocessor has dedicated on-chip RAM for instructions and data. This RISC microprocessor performs the primary transport de-multiplexer functions (such as stream parsing, redirection, filtering, de-scrambling, etc.) and thereby keeps the PR3940 MIPS CPU in the system free to perform more complex tasks, such as handling the real time OS, providing on-screen display, and controlling the I/O processing and web browser functions of the system. This architecture is optimized so that the Transport RISC and PR3940 MIPS CPU function in parallel without contending for the same resources.

Program Specific Information (PSI), Service Information (SI), Conditional Access (CA) messages, and private data are selected and stored in external memory for subsequent off-line processing. De-scrambled stream content can be filtered with a high degree of flexibility and routed to the relevant location by the Transport RISC microprocessor (e.g., audio decoder, video decoder, several private data buffer pools for application-specific processing).

The MSP supports the following external interfaces:

- Transport stream input @ 81 Mbits/s in the parallel mode
- Transport stream output @ 81 Mbits/s, which interfaces to the internal transport stream network. It is then routed to either the transmitter channels (TX) of the internal 1394 link core or output to the external output pins (TSOUT).

33.1.1 MSP Features

MSP features include the following items:

- Flexible Packet framer
  - Frames DVB/DIRECTV packets.
  - Performs sync lock/drop hysteresis.
  - Contains integrated 64 PID filters. All PID filters can be dedicated to filter packets containing sections, transport packet headers, or to parse audio, video, teletext, subtitle and other PES data.
- GP/HS interface, which can serve as an alternate input from IEEE1394 devices, for example. The IEEE1394 GP/HS mode supports packet insertion and has an internal SRAM for storing two packets. It can also output either scrambled or de-scrambled transport streams to IEEE1394 devices.
- Dedicated 108 MHz Transport RISC processor, which is flexible enough to provide a de-multiplexing scheme that is fully compliant with CANAL+ Media Web Box, Kirch DBox-2 and BSkyB specifications.
  - 108 MHz RISC engine dedicated to transport stream parsing
  - Downloadable microcode for greater system flexibility
  - Performs software parsing of Transport Stream (TS), Program Stream (PS) and proprietary (SW) data streams.
  - Filters section data by interfacing to the dedicated hardware filter.
  - Controls de-scrambling by interfacing with the integrated de-scramblers.
- Integrated hardware filter
  - 32-section filters to retrieve PSI, SI, private data, EPG, etc.
  - Can be used in the Section Filter mode (to support Canal+) or the Section Range Filter mode (to support Kirch).
  - Section filters can be configured to either 8 or 12 bytes wide. (Section range filters are only 12 bytes wide.)
  - PTS/DTS range filtering
- Queue Manager for dispatching data to memory
  - Flexible 48-channel DMA-based storage of section substreams, TS/PES data substreams and TS/PES packet headers in external memory.
  - Ability to capture data and, if the queue fills, “protect” the data that is already in memory and discontinue further dispatch of data to that memory queue.
- Real time de-scrambler, supporting different de-scrambler algorithms and consisting of four modules:
  - Control word bank, containing 16 pairs (odd, even) of control words
  - DVB de-scrambler core, implementing the stream decipher and block decipher algorithms
  - MULTI2 de-scrambler algorithm

- Integrated Conditional Access Module (ICAM), including an ISO7816-UART to interface with the conditional access Smartcard
- DIRECTV (Single and triple-DES) de-scrambler algorithm
- Copy Protection (single DES(ECB))

33.2 Functional Description

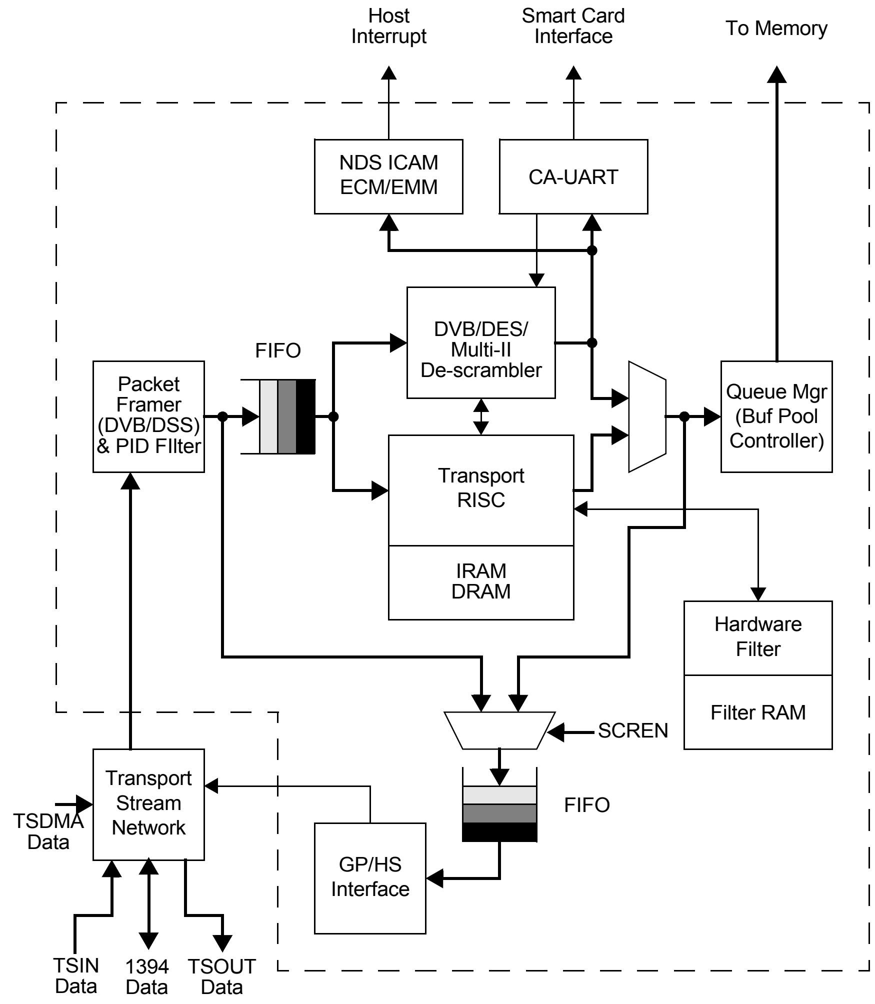


Figure 1: MSP Block Diagram

33.2.1 GP/HS Data Interface

The General Purpose and High Speed (GP/HS) interface can be used to filter specific parts from the transport stream at a specific level (transport, PES, section). It can also be used to bypass the transport stream and output parts to the internal Transport Stream Network (TSN) for routing to the internal IEEE1394 link core or to the TSOUT interface.

The GP/HS filter has two basic modes: a GP mode, which outputs data at the frequency of the transport stream rate, and an HS mode, which outputs data at a 13.5 MHz clock rate. Filtering in GP mode is not possible, so the GP/HS filter outputs all packets selected by the PID filter and the Transport RISC Processor. In the HS mode, there are four filter modes available.

IEEE 1394 Mode

- Intended to interface with a link level IEEE 1394 Device (like the PDI1394L11).
- This is a Packet Mode. Entire Transport Packets of fixed size are output via the GP/HS Interface. The packet size is programmable.
- Output packets can be scrambled or de-scrambled.
- The GP/HS Strobe signal is always an output from the device.
- Packet insertion is supported.

General Purpose (GP) Mode

- Provides a generic Packet Interface to any external device. The interface is similar to the IEEE 1394 Mode except that the GP/HS Strobe signal is an input in the Input Mode.
- Entire Transport Packets of fixed size are output via the GP/HS Interface. The packet size is programmable.
- Output packets can be scrambled or de-scrambled.
- Packet insertion of two different packets is supported.

High Speed Data (HS) Mode

- Outputs a filtered Transport Stream through the GP/HS port.
- The filtering is done in software by the Transport RISC Engine.
- Only de-scrambled data can be filtered and output.
- This is not a packet mode, only portions (or the whole) of a transport packet are output.
- Packet Insertion is not supported.

33.2.2 Packet Framer

The packet framer accepts an MPEG2 or DIRECTV transport stream at its channel interface. It receives this data stream from the Input/Output router, clocked in at the channel clock rate provided by the corresponding source clock. The packet framer then performs synchronization detection on data arriving in serial or parallel formats. The packet framer also provides synchronization lock, drop hysteresis, and packet alignment. During synchronization lock, the packet framer continues to output framed packets to the framer FIFO. During synchronization drop, the packet framer does not output framed packets to the input FIFO.

The packet framer also contains a 64-entry programmable PID filter. Each entry can be defined to be valid, tagged to extract a PCR, and routed to the GPHS output interface.

Features and functions of the packet framer are:

- Only parallel format input data are supported.
- Input data can be clocked in with its own clock source, the packet framer will perform the corresponding frequency conversion and rate buffering.
- Packet framing and synchronization, used to detect start and end of packets.
- DVB/DIRECTV support
- Synchronization Lock/Drop Hysteresis support
- Integrated PID filter for 64 PIDs
- Integrated timestamp insertion unit

33.2.2.1 Framer FIFO

A 256x8 FIFO between the packet framer and the RISC Engine buffers the continuous stream of channel data. This FIFO also accommodates the different processing requirements of the various data in the transport packet. The FIFO allows for asynchronous clocks between the channel interface and packet synchronizer/ framer logic and remaining logic in the MSP Core.

The packet framer input runs on the channel (or other source) clock, while the RISC engine and remaining logic run off an internal 108 MHz (SYSCLK) clock.

33.2.2.2 PID Filter (TS Parsing/PS Parsing)

The PID filter is part of the packet framer and supports up to 64 PIDs for DVB/ DIRECTV respectively. Each PID can be defined to work in the positive or negative mode. The negative mode is done with the help of the RISC.

The PID filter performs two types of filtering:

- All packets that match a PID in the PID table will be sent downstream for further processing or discarded.
- All packets that do not match any entry in the PID table (such packets are called NOMATCH packets) can be programmed to either be dropped from the stream or sent into a default queue in memory.

Once the PID filtering is done, selected packets are sent downstream to the Transport/De-Multiplexer RISC Engine for further processing. The RISC does further parsing of data in the packets and other kinds of filtering.

33.2.2.3 Local Header Unit and Timestamp Insertion Support

The packet framer has a local header unit that uses the PID attributes from the PID unit to build a local header. This local header is used by the RISC Engine to further process the packet. The local header provides the following:

- Generates timestamp information. A 31-bit timestamp is inserted for every packet based on a counter running at a system-defined timestamp clock (programmable option).
- Generates matched PID location information.
- Adds a 10-byte IEEE 1394 header for 130-byte input DIRECTV packets for recording to an IEEE 1394 device.
- Generates host interrupts on the arrival of a PCR packet or any packet (programmable interrupt generation).

33.2.2.4 Transport/De-Multiplexer RISC Engine

The custom RISC engine has been designed to provide a compact, high-performance configurable engine to handle the processing of the incoming MPEG2 transport packets. Using customizable microcode, the RISC engine supports a number of baseline functions. It can be modified further to support additional application-specific requirements.

The RISC engine has its own dedicated Instruction SRAM and Data SRAM. It operates at a clock speed of 108 MHz and executes microcode that resides in the on-chip SRAMs. The microcode is downloaded from external memory during initialization by the host processor.

Features supported by the RISC engine include the following:

- MPEG2 transport parsing
  - Output of entire transport packet and payload
  - Output of PES packet and PES payload
  - Output of section data
  - Output of adaptation field private data
  - Output to any of the MQM queues (See Section 33.2.5 on page 33-695 for more details.)
- MPEG2 stream de-multiplexing
  - Video
  - Audio
  - General Purpose
  - Application messages
  - Adaptation fields containing PCR
- De-scrambling and conditional access support
  - Transport layer de-scrambling support
  - PES layer de-scrambling support
  - 16 de-scrambler odd/even Control Word (CW) pairs
- Section Filtering support for PSI tables
  - PSI, EMM, ECM, or user data
  - Up to 32-section table entries

  - Support for section payload straddle over more than a single transport packet
  - Section Filtering interface with a dedicated hardware section filter
- Error management
  - Error reporting on TEI bit set on TP Header
- Event generation
  - Interrupt events can be generated by the RISC under any parsing error or event conditions with total software control.

For more information see the PNX8526 MSP RISC Engine Core Instruction Set document.

33.2.2.5 Transport RISC Engine Memories

The Transport/De-Multiplexer RISC engine and the MIPS microprocessor share two memories, which are used for Transport/De-Multiplexer RISC code storage and data tables. The RISC Instruction RAM (I-RAM) contains the Transport RISC microcode and can be initialized by the CPU at any time when the RISC is stopped. The Transport RISC Data RAM (D-RAM) is dual-ported and can be written and read by both the RISC engine and the MIPS CPU. The D-RAM is used for inter-processor communication.

The Transport RISC is a byte-processing engine that is controlled by microcode downloaded by the MIPS CPU. A variety of microcode can be downloaded, depending on the application. This feature is what makes the Transport RISC engine flexible and versatile.

33.2.2.6 Transport Parsing and De-Multiplexing

The Transport/De-Multiplexer RISC engine parses transport data from the FIFO and de-multiplexes the data for delivery to a number of destinations in the system. The RISC engine writes the data into a FIFO. There, the data waits for the Memory Queue Manager which then places it into an appropriate memory queue. The parsed data can be routed to 48 separate data queues in D-RAM. The queues are not directly accessible by the RISC engine, but are accessed by selecting a queue number.

33.2.3 Hardware Section Filtering

Section filtering support is provided by a dedicated Hardware Section Filter (HSF) in the Transport/De-Multiplexer. This block performs two of the major filtering functions: filtering of the section header and filtering of the Program Timestamp. Performing these two functions in hardware allows the RISC engine more bandwidth to perform other functions.

The section header filtering has two modes: section filtering and section range filtering. Section filtering allows comparison of a 96/64-bit filtering programmable section header with a section filter. A section filter consists of a 96/64-bit mask vector, a 96/64-bit mode vector, and a 96/64-bit compare value. This mode is primarily intended to support Canal+ Media Web Box 3.0 type filtering applications. Section range filtering allows comparison of a 64-bit section header with a minimum/ maximum value that includes an “inner range” or “outer range” setting per byte. This mode is primarily intended to support Kirch DBox2 style filtering.

Table 1: 8-Byte Section Filter Organization in the SFRAM

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x00	X	X	X	LNK	M3	M2	M1	M0	V7	V6	V5	V4	V3	V2	V1	V0
0x10	X	X	X	X	N7	N6	N5	N4	N3	N2	N1	N0	M7	M6	M5	M4

Table 2: 12-Byte Section Filter Organization in the SFRAM

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x00	X	X	X	LNK	V11	V10	V9	V8	V7	V6	V5	V4	V3	V2	V1	V0
0x10	X	X	X	X	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
0x20	X	X	X	X	N11	N10	N9	N8	N7	N6	N5	N4	N3	N2	N1	N0

Table 3: Section Range Filter Organization in the SFRAM

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0x00	X	X	X	LNK	Mi3	Mi2	Mi1	Mi0	Ma7	Ma6	Ma5	Ma4	Ma3	Ma2	Ma1	Ma0
0x10	X	X	X	MaM 7	MaM 6	MaM 5	MaM 4	MaM 3	MaM 2	MaM 1	MaM 0	Mi7	Mi6	Mi5	Mi4	FUN C
0x20	X	X	X	X	X	X	X	MiM7	MiM6	MiM5	MiM4	MiM3	MiM2	MiM1	MiM0	X

The RISC engine interfaces with the HSF through a control register in its address space. The HSF includes Section Filtering RAM (SFRAM), which is used to store the 32 section filter values. Each filter occupies two or three entries in the SFRAM.

The host loads the section filters into the SFRAM when new filters are available. It can also read from this RAM for diagnostic purposes. The filters are stored in the SFRAM as a linked list. The RISC microprocessor has no direct access to this RAM.

The RISC loads the section header to be filtered into the HSF and initiates section filtering. It then polls the HSF registers to find out when filtering has been completed. When done, the RISC engine will read back the compare vector from the HSF.

33.2.4 De-Scrambler Core

The De-Scrambler core is intended for de-scrambling MPEG-encoded information packets that are received by the Set Top Box receiver, with optional Copy Protection (DES) that is applied for local use. The de-scrambler can support up to four different de-scrambling standards and can handle up to sixteen control word pairs. It is made up of five blocks. The De-Scrambler core has the following features:

- Supports de-scrambling on data rates up to 81 Mbits/s.
- Control word (CW) bank containing 16 pairs (odd, even) of control words
- DVB de-scrambler core implementing the stream and block decipher algorithms (see Common Scrambling Specifications, Rev 1 (E), European project for Digital Video Broadcasting.(64-bit key)
- MULTI2 de-scrambler algorithm (64-bit key)

- ICAM de-scrambling control unit is included with MSP1 and MSP2 only.
- DIRECTV (DES) de-scrambling control unit with support for ECB and CBC modes. (Single DES with 56-bit key and a Triple DES with 168-bit key).
- Copy Protection (Single DES with 56-bit key) applies to packets that were scrambled in the original MPEG stream. Copy protection is applied to protect the stream content as it flows from the MSP to other devices in the Set Top Box system.

In transport level de-scrambling mode, de-scrambling is performed on the payload of a transport packet only. The transport header and the (optional) adaptation field are expected to be unscrambled. In PES-level de-scrambling mode, PES payloads are de-scrambled, and transport packet headers, adaptation fields, and PES headers are expected to be unscrambled.

The de-scrambler is capable of supporting PES-level de-scrambling in accordance with the recommendations of the DVB standard. De-scrambler core restrictions are actually less rigid than recommendations made in the DVB standard.

33.2.5 Memory Queue Manager Controller

Sections, TS/PES data, ECMs, EMMs, etc. that pass filtering are stored via the Memory Queue manager into a buffer pool in external memory.

- Interfaces with the RISC Engine and collects output data, queue and control information.
- "Unrolls" virtual queue number information into physical addresses for dispatch to memory.
- Maintains physical addresses for all the queues.
- Interfaces to system memory via 1 to 3 DVP-L2 Interfaces.
- Supports 48 queues.
- Each queue has a separate base address and a separate size.
  - Buffer size is programmable to either 64, 128, 256, 512 Bytes, 1 kB, 2 kB, 4 kB, 8 kB, 16 kB, 32 kB, 64 kB, 128 kB, 256 kB, 320K, 512 kB or 1MB.
  - Data for any queue can be routed to memory via any one of the DVP-L2 Interfaces.
  - An interrupt for any queue can be routed to MQM_INT[0] or MQM-INT[1].
- Tagging and Flushing mechanisms supported for queue data under control of the RISC Microprocessor

Several data sections (e.g., TS/PES packets, SI-sections or proprietary data) may be stored in one buffer. The Memory Queue manager can handle up to 48 data streams that are de-multiplexed from the TS stream. These data streams are filtered by the MSP. Data in these queues can start or stop at any byte boundary. Queues must be aligned to multiples of their sizes. Queues are circular in nature and will wrap data around their boundaries.

33.2.6 MPEG System Processor Interrupt Handler

The MSP interrupt handler combines the interrupts sources [9:0] and [21:12] into two interrupt lines. Interrupts are stored in an interrupt pending register, which is ANDed with the interrupt mask register of a specific interrupt line before forming the vectored output interrupt lines that are connected to the interrupt handler.

33.2.7 MSP Interrupt Sources and Control

The MSP interrupt scheme is shown in Figure 2 below. There are ten internal sources of interrupts within the MSP, and two output sources. Each one of the ten internal sources is routed to both outputs, and software can selectively enable/disable them to specifically route them to the desired output. The ten internal interrupt sources make up bits [9:0] and bits [21:12] in all the interrupt registers. Software is expected to mask each internal interrupt source in at least one of the two interrupt outputs

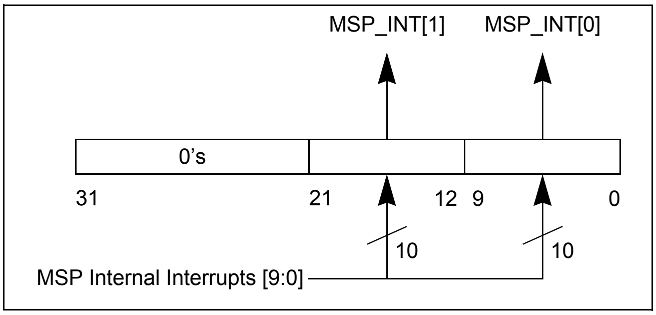


Figure 2: MSP Interrupt Scheme

Table 4: MSP Internal Interrupt Sources

Bit Number	Interrupt Source
0	MQM_INT[0] - Memory Queue Manager Interrupt 0. Interrupts when any of the queues generates an interrupt and this interrupt is enabled and directed to MQM_INT[0].
1	MQM_INT[1] - Memory Queue Manager Interrupt 1. Interrupts when any of the queues generates an interrupt and this interrupt is enabled and directed to MQM_INT[1].
2	SYNCLOCK - The Packet Framer generates this interrupt when it enters a Synchronization Lock.
3	SYNCDROP - The Packet Framer generates this interrupt when it loses Packet Synchronization.
4	PCRINT - The Packet Framer generates this interrupt every time a new PCR is captured from the stream.
5	RISCINT - This interrupt is generated when the Transport RISC Engine executes the INT instruction.
6	PKTAINT - This interrupt is generated when the GP/HS Module has finished inserting Packet A into the GP/HS output stream.

Table 4: MSP Internal Interrupt Sources *…Continued*

Bit Number	Interrupt Source
7	PKTBINT - This interrupt is generated when the GP/HS Module has finished inserting Packet B into the GP/HS output stream.
8	ICAMECM - This interrupt is generated by the ICAM ECM/EMM processor.
9	ICAM_CAUART - This interrupt is generated by the ICAM CA/UART module.
10:11	Reserved

33.3 Register Summary and Descriptions

33.3.1 Clarifications

33.3.1.1 Memory Queue Manager Registers

The Memory Queue Manager (MQM) supports 48 queues in hardware and can generate interrupts for each of them (based on the TAGINT command). Each queue has its own base address and size, which are accessible in a table. All of the 48 interrupts are funneled into two interrupts as an input to the interrupt configuration block. Configuration registers for the MQM are shown in the following register tables.

33.3.1.2 Data Queue Head Pointer Register

The MQM Data Queue Head Pointer register table implements a 26-bit address and a 6-bit attribute field for each of the 48 queues. This table does not power up with valid values after reset. The power up software must write valid values into these registers before starting any data flow in the MSP.

33.3.1.3 MSP Base Address

The PNX8526 has three MSP modules. The register base address for MSP 1 is 0x11 8000. The base address for the MSP 2 registers is 0x12 0000 and the base address for the MSP 3 is 0x12 8000. MSP 3 does not include the ICAM function, otherwise the three MSP modules are identical.

Table 5: MSP Module Register Summary

Offset	Name	Description
MSP 1		
0x11 8000—80BC	DQHDPTR	Data Queue N (N = 0 to 47) Head Pointer Registers
0x11 80D0	DQHDPTRBASE	Data Queue Head Pointer Base Register
0x11 80D4	DQI0MSKHI	Data Queue Interrupt 0 Mask Register High
0x11 80D8	DQI0MSKLO	Data Queue Interrupt 0 Mask Register Low
0x11 80DC	DQI1MSKHI	Data Queue Interrupt 1 Mask Register High
0x11 80E0	DQI1MSKLO	Data Queue Interrupt 1 Mask Register Low
0x11 80E4	DQISTATHI	Data Queue Interrupt Status Register High
0x11 80E8	DQISTATLO	Data Queue Interrupt Status Register Low
0x11 80F0	DQTHRESH0	Data Queue Interrupt Threshold Register 0

Table 5: MSP Module Register Summary *…Continued*

Offset	Name	Description
0x11 80F4	DQTHRESH1	Data Queue Interrupt Threshold Register 1
0x11 80F8	DQTHRESH2	Data Queue Interrupt Threshold Register 2
0x11 80FC	DQCTL	Data Queue Control Register
0x11 8100	GPHSCTL	GP/HS Interface Control Register
0x11 8104	PKTSIZE	GP/HS Packet Size Register
0x11 8108	INSTHRESH	Packet Insertion Threshold Register
0x11 8110	INSSTAT	Packet Insertion Status Register
0x11 8114—81FC	Reserved	
0x11 8200	FRMRCTL	Packet Framer Format Control
0x11 8204	FRMRSYNC	Packet Framer Sync Control
0x11 8208	PIDMASK	Packet Framer PID Mask
0x11 820C	PIDCTL	Packet Framer PID Match Control
0x11 8210	PEAKVAL	Packet Framer FIFO Peak Value
0x11 8214—8258	Reserved	
0x11 825C	DSS_BITRATE	DSS BitRate Register
0x11 8300—83FC	PIDTABLE	PID Table N (N = 0, to 63) Register
0x11 8400	RISCDBG0	RISC Debug Register 0
0x11 8404	RISCDBG1	RISC Debug Register 1
0x11 8408	RISCDBG2	RISC Debug Register 2
0x11 840C	RISCDBG3	RISC Debug Register 3
0x11 8410	RISCDBG4	RISC Debug Register 4
0x11 8414	RISCDBG5	RISC Debug Register 5
0x11 8418	RISCDBG6	RISC Debug Register 6
0x11 841C	RISCDBG7	RISC Debug Register 7
0x11 8420	RISCDBG8	RISC Debug Register 8
0x11 8424	RISCDBG9	RISC Debug Register 9
0x11 8428	RISCDBG10	RISC Debug Register 10
0x11 842C	RISCDBG11	RISC Debug Register 11
0x11 8430	RISCCTL	RISC Engine Control Register
0x11 8434	HSFCTL	HSF Control Register
0x11 8600 + n*8	KEYnEVN0 (n = 0 to F)	De-scrambler Even Key Register 0
0x11 8604 + n*8	KEYnEVN1 (n = 0 to F)	De-scrambler Even Key Register 1
0x11 8680 + n*8	KEYnODD0 (n = 0 to F)	De-scrambler Odd Key Register 0
0x11 8684 + n*8	KEYnODD1 (n = 0 to F)	De-scrambler Odd Key Register 1
0x11 8700	M2SYSKEY0	Multi-2 System Key Register 0

Table 5: MSP Module Register Summary *…Continued*

Offset	Name	Description
0x11 8704	M2SYSKEY1	Multi-2 System Key Register 1
0x11 8708	M2SYSKEY2	Multi-2 System Key Register 2
0x11 870C	M2SYSKEY3	Multi-2 System Key Register 3
0x11 8710	M2SYSKEY4	Multi-2 System Key Register 4
0x11 8714	M2SYSKEY5	Multi-2 System Key Register 5
0x11 8718	M2SYSKEY6	Multi-2 System Key Register 6
0x11 871C	M2SYSKEY7	Multi-2 System Key Register 7
0x11 8720	CPKEYEVEN0	Copy Protect Even Key Register 0
0x11 8724	CPKEYEVEN1	Copy Protect Even Key Register 1
0x11 8728	CPKEYODD0	Copy Protect Odd Key Register 0
0x11 872C	CPKEYODD1	Copy Protect Odd Key Register 1
0x11 8730	DSCIVCBC0	De-scrambler IV CBC Register 0
0x11 8734	DSCIVCBC1	De-scrambler IV CBC Register 1
0x11 8738	DSCIVOFB0	De-scrambler IV OFB Register 0
0x11 873C	DSCIVOFB1	De-scrambler IV OFB Register 1
0x11 8740	CPIVCBC0	Copy Protect IV CBC Register 0
0x11 8744	CPIVCBC1	Copy Protect IV CBC Register 1
0x11 8748	CPIVOFB0	Copy Protect IV OFC Register 0
0x11 874C	CPIVOFB1	Copy Protect IV OFB Register 1
0x11 8750	MODEPAD	De-scrambler Mode/Pad Register
0x11 8754—8758	Reserved	
0x11 8C00	MSPRST	MSP Module Reset Register
0x11 8FE0	MSPINTST	MSP Interrupt Status Register
0x11 8FE4	MSPINTENA	MSP Interrupt Enable Register
0x11 8FE8	MSPINTCLR	MSP Interrupt Clear Register
0x11 8FEC	MSPINTSET	MSP Interrupt Set Register
0x11 8FF4	POWERDOWN	Powerdown mode
0x11 8FFC	MODULEID	MSP Module ID Register
MSP 2		
MSP 2 Registers begin at 0x12 0000. They are identical to the MSP 1 registers, described above.		
MSP 3		
MSP 3 Registers begin at 0x12 8000. They are identical to the MSP 1 registers, described above. Note that MSP 3 does NOT implement the ICAM.		

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Data Queue *N* (*N* = 0 to 47) Head Pointer Registers				
Offset 0x11 8000—80BC			*DQHDPTR*	
31:30	R/W	NI	DQL2NUM [1:0]	Defines which DVP-L2 Interface the data for this queue must be routed though. 0x0 = Route data to DVP-L2 IF #0. 0x1 = Route data to DVP-L2 IF #1. 0x2 = Route data to DVP-L2 IF #2. 0x3 = Reserved
29:26	R/W	NI	DQSIZE [3:0]	Defines the Data Queue size: 0x0 = 320k bytes 0x1 = 64 bytes 0x2 = 128 bytes 0x3 = 256 bytes 0x4 = 512 bytes 0x5 = 1K bytes 0x6 = 2K bytes 0x7 = 4K bytes 0x8 = 8K bytes 0x9 = 16K bytes 0xA = 32K bytes 0xB = 64K bytes 0xC = 128K bytes 0xD = 256K bytes 0xE = 512K bytes 0xF = 1M bytes
25:0	R/W	NI	DQHP [25:0]	This field defines the lower 26 bits of the 32-bit data queue head pointer. It is updated only when a NEWQ command is received.
Offset 0x11 80C0			*DQLIMIT0*	
31:24	R	0	Reserved	
23:21	R/W	0	DQLIMIT[143:141]	Controls the data queue limit interrupt for queue #47. Once the limit is reached, the interrupt will occur. If DQENLOCK of Data Queue Control Register is enabled, the remaining data for the queue will be discarded until the limit is changed (or disabled). Data queue limit interrupt is disabled. Data queue limit interrupt is at 1/4 queue. Data queue limit interrupt is at 2/4 queue. Data queue limit interrupt is at 3/4 queue. Data queue limit interrupt is at 4/4 queue. Reserved. Reserved. Writing this value has no effect on this field, and this field will remain unchanged.
20:18	R/W	0	DQLIMIT[140:138]	Same as above for data queue #46

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17:15	R/W	0	DQLIMIT[137:135]	Same as above for data queue #45
14:12	R/W	0	DQLIMIT[134:132]	Same as above for data queue #44
11:9	R/W	0	DQLIMIT[131:129]	Same as above for data queue #43
8:6	R/W	0	DQLIMIT[128:126]	Same as above for data queue #42
5:3	R/W	0	DQLIMIT[125:123]	Same as above for data queue #41
2:0	R/W	0	DQLIMIT[122:120]	Same as above for data queue #40
Offset 0x11 80C4			***DQLIMIT1***	
31:30	R	0	Reserved	
29:27	R/W	0	DQLIMIT[119:117]	Controls the data queue limit interrupt for queue #39. Once the limit is reached, the interrupt will occur. If DQENLOCK of Data Queue Control Register is enabled, the remaining data for the queue will be discarded until the limit is changed (or disabled). Data queue limit interrupt is disabled. Data queue limit interrupt is at 1/4 queue. Data queue limit interrupt is at 2/4 queue. Data queue limit interrupt is at 3/4 queue. Data queue limit interrupt is at 4/4 queue. Reserved. Reserved. Writing this value has no effect on this field, and this field will remain unchanged.
26:24	R/W	0	DQLIMIT[116:114]	Same as above for data queue #38
23:21	R/W	0	DQLIMIT[113:111]	Same as above for data queue #37
20:18	R/W	0	DQLIMIT[110:108]	Same as above for data queue #36
17:15	R/W	0	DQLIMIT[107:105]	Same as above for data queue #35
14:12	R/W	0	DQLIMIT[104:102]	Same as above for data queue #34
11:9	R/W	0	DQLIMIT[101:99]	Same as above for data queue #33
8:6	R/W	0	DQLIMIT[98:96]	Same as above for data queue #32
5:3	R/W	0	DQLIMIT[95:93]	Same as above for data queue #31
2:0	R/W	0	DQLIMIT[92:90]	Same as above for data queue #30
Offset 0x11 80C8			***DQLIMIT2***	
31:30	R	0	Reserved	

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
29:27	R/W	0	DQLIMIT[89:87]	Controls the data queue limit interrupt for queue #29. Once the limit is reached, the interrupt will occur. If DQENLOCK of Data Queue Control Register is enabled, the remaining data for the queue will be discarded until the limit is changed (or disabled). Data queue limit interrupt is disabled. Data queue limit interrupt is at 1/4 queue. Data queue limit interrupt is at 2/4 queue. Data queue limit interrupt is at 3/4 queue. Data queue limit interrupt is at 4/4 queue. Reserved. Reserved. Writing this value has no effect on this field, and this field will remain unchanged.
26:24	R/W	0	DQLIMIT[86:84]	Same as above for data queue #28
23:21	R/W	0	DQLIMIT[83:81]	Same as above for data queue #27
20:18	R/W	0	DQLIMIT[80:78]	Same as above for data queue #26
17:15	R/W	0	DQLIMIT[77:75]	Same as above for data queue #25
14:12	R/W	0	DQLIMIT[74:72]	Same as above for data queue #24
11:9	R/W	0	DQLIMIT[71:69]	Same as above for data queue #23
8:6	R/W	0	DQLIMIT[68:66]	Same as above for data queue #22
5:3	R/W	0	DQLIMIT[65:63]	Same as above for data queue #21
2:0	R/W	0	DQLIMIT[62:60]	Same as above for data queue #20
Offset 0x11 80CC			***DQLIMIT3***	
31:30	R	0	Reserved	
29:27	R/W	0	DQLIMIT[59:57]	Controls the data queue limit interrupt for queue #19. Once the limit is reached, the interrupt will occur. If DQENLOCK of Data Queue Control Register is enabled, the remaining data for the queue will be discarded until the limit is changed (or disabled). Data queue limit interrupt is disabled. Data queue limit interrupt is at 1/4 queue. Data queue limit interrupt is at 2/4 queue. Data queue limit interrupt is at 3/4 queue. Data queue limit interrupt is at 4/4 queue. Reserved. Reserved. Writing this value has no effect on this field, and this field will remain unchanged.
26:24	R/W	0	DQLIMIT[56:54]	Same as above for data queue #18
23:21	R/W	0	DQLIMIT[53:51]	Same as above for data queue #17
20:18	R/W	0	DQLIMIT[50:48]	Same as above for data queue #16
17:15	R/W	0	DQLIMIT[47:45]	Same as above for data queue #15

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
14:12	R/W	0	DQLIMIT[44:42]	Same as above for data queue #14
11:9	R/W	0	DQLIMIT[41:39]	Same as above for data queue #13
8:6	R/W	0	DQLIMIT[38:36]	Same as above for data queue #12
5:3	R/W	0	DQLIMIT[35:33]	Same as above for data queue #11
2:0	R/W	0	DQLIMIT[32:30]	Same as above for data queue #10
Offset 0x11 80EC			*DQLIMIT4*	
31:30	R	0	Reserved	
29:27	R/W	0	DQLIMIT[29:27]	Controls the data queue limit interrupt for queue #9. Once the limit is reached, the interrupt will occur. If DQENLOCK of Data Queue Control Register is enabled, the remaining data for the queue will be discarded until the limit is changed (or disabled). Data queue limit interrupt is disabled. Data queue limit interrupt is at 1/4 queue. Data queue limit interrupt is at 2/4 queue. Data queue limit interrupt is at 3/4 queue. Data queue limit interrupt is at 4/4 queue. Reserved. Reserved. Writing this value has no effect on this field, and this field will remain unchanged.
26:24	R/W	0	DQLIMIT[26:24]	Same as above for data queue #8
23:21	R/W	0	DQLIMIT[23:21]	Same as above for data queue #7
20:18	R/W	0	DQLIMIT[20:18]	Same as above for data queue #6
17:15	R/W	0	DQLIMIT[17:15]	Same as above for data queue #5
14:12	R/W	0	DQLIMIT[14:12]	Same as above for data queue #4
11:9	R/W	0	DQLIMIT[11:9]	Same as above for data queue #3
8:6	R/W	0	DQLIMIT[8:6]	Same as above for data queue #2
5:3	R/W	0	DQLIMIT[5:3]	Same as above for data queue #1
2:0	R/W	0	DQLIMIT[2:9]	Same as above for data queue #0
Offset 0x11 80D0			*DQHDPTRBASE*	
31:26	R/W	0	DQHP	This is the upper 6 bits of the 32-bit data queue head pointer, and is the same for all 64 queues.
25:0			Reserved	
Data Queue Interrupt Mask Registers				
Offset 0x11 80D4			*DQI0MSKHI*	
31:16			Reserved	
15:0	R/W	0	DQI0MSK [47:32]	0 = Corresponding data queue interrupt is masked. 1 = Corresponding data queue interrupt is enabled and routed to MQMINT[0].

MSP 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
Offset 0x11 80D8			***DQI0MSKLO***	
31:0	R/W	0	DQI0MSK [31:0]	0 = Corresponding data queue interrupt is disabled. 1 = Corresponding data queue interrupt is enabled and routed to MQMINT[0].
Offset 0x11 80DC			***DQI1MSKHI***	
31:16			Reserved	
15:0	R/W	0	DQI1MSK [47:32]	0 = Corresponding data queue interrupt is masked. 1 = Corresponding data queue interrupt is enabled and routed to MQMINT[1].
Offset 0x11 80E0			***DQI1MSKLO***	
31:0	R/W	0	DQI1MSK [31:0]	0 = Corresponding data queue interrupt is disabled. 1 = Corresponding data queue interrupt is enabled and routed to MQMINT[1].
Data Queue Interrupt Status Registers				
Offset 0x11 80E4			***DQISTATHI***	
31:16		-	Unused	
15:0	R W	0	DQISTAT [47:32]	These bits reflect the interrupt pending status of the queues. The MQM will set the corresponding bit in this register regardless of whether the corresponding bit in the DQIMSK registers was masked or not. 0 = Corresponding data queue interrupt has not occurred. 1 = Corresponding data queue interrupt is pending. 0 = No effect 1 = Corresponding data queue interrupt is reset by the host.
Offset 0x11 80E8			***DQISTATLO***	
31:0	R W	0	DQISTAT [31:0]	These bits reflect the interrupt pending status of the queues. The MQM will set the corresponding bit in this register regardless of whether the corresponding bit in the DAIMSK register was masked or not. 0 = Corresponding data queue interrupt has not occurred. 1 = Corresponding data queue interrupt is pending. 0 = No effect 1 = Corresponding data queue interrupt is reset by the host.
Data Queue Interrupt Threshold Registers				
Offset 0x11 80F0			***DQTHRESH0***	
31:30	R/W	0	DQITHRESH [95:94]	Controls the Data Queue Threshold Interrupt for Queue #47. 0x0 = Data Queue Threshold Interrupt is disabled. 0x1 = DQ Threshold Interrupt is at 1/4 queue. 0x2 = DQ Threshold Interrupt is at 1/2 queue. 0x3 = Writing this value has no effect on this field, and this field will remain unchanged.
29:28	R/W	0	DQITHRESH[93:92]	Same as above for Queue #46.
27:26	R/W	0	DQITHRESH[91:90]	Same as above for Queue #45.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
25:24	R/W	0	DQITHRESH[89:88]	Same as above for Queue #44.
23:22	R/W	0	DQITHRESH[87:86]	Same as above for Queue #43.
21:20	R/W	0	DQITHRESH[85:84]	Same as above for Queue #42.
19:18	R/W	0	DQITHRESH[83:82]	Same as above for Queue #41.
17:16	R/W	0	DQITHRESH[81:80]	Same as above for Queue #40.
15:14	R/W	0	DQITHRESH[79:78]	Same as above for Queue #39.
13:12	R/W	0	DQITHRESH[77:76]	Same as above for Queue #38.
11:10	R/W	0	DQITHRESH[75:74]	Same as above for Queue #37.
9:8	R/W	0	DQITHRESH[73:72]	Same as above for Queue #36.
7:6	R/W	0	DQITHRESH[71:70]	Same as above for Queue #35.
5:4	R/W	0	DQITHRESH[69:68]	Same as above for Queue #34.
3:2	R/W	0	DQITHRESH[67:66]	Same as above for Queue #33.
1:0	R/W	0	DQITHRESH[65:64]	Same as above for Queue #32.
Offset 0x11 80F4			*DQTHRESH1*	
31:30	R/W	0	DQITHRESH [63:62]	Controls the Data Queue Threshold Interrupt for Queue #31. 0x0 = Data Queue Threshold Interrupt is disabled. 0x1 = DQ Threshold Interrupt is at 1/4 queue. 0x2 = DQ Threshold Interrupt is at 1/2 queue. 0x3 = Writing this value has no effect on this field, and this field will remain unchanged.
29:28	R/W	0	DQITHRESH [61:60]	Same as above for Queue #30.
27:26	R/W	0	DQITHRESH [59:58]	Same as above for Queue #29.
25:24	R/W	0	DQITHRESH [57:56]	Same as above for Queue #28.
23:22	R/W	0	DQITHRESH [55:54]	Same as above for Queue #27.
21:20	R/W	0	DQITHRESH [53:52]	Same as above for Queue #26.
19:18	R/W	0	DQITHRESH [51:50]	Same as above for Queue #25.
17:16	R/W	0	DQITHRESH [49:48]	Same as above for Queue #24.
15:14	R/W	0	DQITHRESH [47:46]	Same as above for Queue #23.
13:12	R/W	0	DQITHRESH [45:44]	Same as above for Queue #22.
11:10	R/W	0	DQITHRESH [43:42]	Same as above for Queue #21.
9:8	R/W	0	DQITHRESH [41:40]	Same as above for Queue #20.
7:6	R/W	0	DQITHRESH [39:38]	Same as above for Queue #19.
5:4	R/W	0	DQITHRESH [37:36]	Same as above for Queue #18.
3:2	R/W	0	DQITHRESH [35:34]	Same as above for Queue #17.
1:0	R/W	0	DQITHRESH [33:32]	Same as above for Queue #16.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 80F8			*DQTHRESH2*	
31:30	R/W	0	DQITHRESH [31:0]	Controls the Data Queue Threshold Interrupt for Queue #15. 0x0 = Data Queue Threshold Interrupt is disabled. 0x1 = DQ Threshold Interrupt is at 1/4 queue. 0x2 = DQ Threshold Interrupt is at 1/2 queue. 0x3 = Writing this value has no effect on this field, and this field will remain unchanged.
29:28	R/W	0	DQITHRESH [29:28]	Same as above for Queue #14.
27:26	R/W	0	DQITHRESH [27:26]	Same as above for Queue #13.
25:24	R/W	0	DQITHRESH [25:24]	Same as above for Queue #12.
23:22	R/W	0	DQITHRESH [23:22]	Same as above for Queue #11.
21:20	R/W	0	DQITHRESH [21:20]	Same as above for Queue #10.
19:18	R/W	0	DQITHRESH [19:18]	Same as above for Queue #9.
17:16	R/W	0	DQITHRESH [17:16]	Same as above for Queue #8.
15:14	R/W	0	DQITHRESH [15:14]	Same as above for Queue #7.
13:12	R/W	0	DQITHRESH [13:12]	Same as above for Queue #6.
11:10	R/W	0	DQITHRESH [11:10]	Same as above for Queue #5.
9:8	R/W	0	DQITHRESH [9:8]	Same as above for Queue #4.
7:6	R/W	0	DQITHRESH [7:6]	Same as above for Queue #3.
5:4	R/W	0	DQITHRESH [5:4]	Same as above for Queue #2.
3:2	R/W	0	DQITHRESH [3:2]	Same as above for Queue #1.
1:0	R/W	0	DQITHRESH [1:0]	Same as above for Queue #0.
Data Queue Control Registers				
Offset 0x11 80FC			*DQCTL*	
31:4		-	Reserved	
3	R/W	0	DQENLOCK	0 = Disable queue lock mechanism for Limit Interrupt. 1 = Enable the queue lock mechanism for limit interrupt to prevent data queues from overrunning. If the data queue reaches the limit, the following data for the corresponding queue will be discarded until the limit value is changed or the limit interrupt is disabled.
2:1	R/W	0	DQRUN_STOP	11 = Reserved 10 = RUN - Enable MQM to process data. 01 = STOP - The MQM will stop at the next NEWQ command. 00 = NO CHANGE - No change from the last status (If the MQM was running, it will continue to run. If the MQM was stopped, it will not start).
0	R/W	0	DQRST	0 = No effect 1 = Software reset. Resets the whole MQM and the MQM FIFO but does not reset the MQM register values. This bit will auto-clear after the MQM reset procedure is complete.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
GP/HS Registers				
Offset 0x11 8100			*GPHSCTL*	
31:16		-	Unused	
15	R/W	0	AVSYNC_FN	GP/HS Interface mode decides the function of AVSYNC output. 0 = The AVSYNC output creates a pulse on the first valid byte of the packet. This is intended to interface with Philips Link Devices. 1 = The AVSYNC output creates a "blank" or a PACKETGAP when no packets are output.
14	R/W	0	CC_LOCK_AB	Packet Insertion Continuity Count Control 0 = Packets A and B have their own continuity counters. 1 = Packets A and B both increment their continuity counters on any packet insertion.
13	R/W	0	CC_UPD_EN	Packet Insertion Continuity Counter Enable 0 = Continuity counter insertion disabled. 1 = Continuity counter insertion enabled. When this field is written with a ’1’, the CC_INIT_VAL register field will also be updated with the host write data.
12	R/W	0	INS_MODE	Packet insertion mode 0 = Automatic packet insertion mode 1 = Manual packet insertion mode
11	R/W	0	INS_PKTB	0 = Packet B insert disabled 1 = Packet B insert enabled
10	R/W	0	INS_PKTA	0 = Packet A insert disabled 1 = Packet A insert enabled
9	R/W	0	INS_EN	0 = Packet insertion disabled 1 = Packet insertion enabled
8	R/W	0	INPUTSEL	0 = GP/HS uses de-scrambled input. 1 = GP/HS uses scrambled Input.
7:6	R/W	0x3	GPIF_MODE [1:0]	Defines the GP/HS interface mode 00 = GP interface mode 01 = HS Data interface mode 10 = IEEE 1394 interface mode 11 = Disabled
5:2	R/W	0	CC_INIT_VAL [3:0]	Defines the Initial value of the packet insertion Continuity counters. This value will only be loaded into the registers when the CC_UPD_EN bit is written with a ’1’. If the CC_UPD_EN bit is written with a ’0’, this field will not change. This is to prevent accidental change of this field.
1			Reserved	
0	R/W	1	GP_DIR	0 = GP Interface is in the output mode. 1 = GP Interface is in the input mode.
Offset 0x11 8104			*PKTSIZE*	
31:16		-	Unused	

MSP 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
15:10			Reserved	
9	R/W	0	DVB_DSS	This field is only used by the packet insertion logic to insert the Continuity count into inserted packets at the correct place. 0 = GP/HS Output interface is in DSS mode. 1 = GP/HS Output interface is in DVB mode.
8	R/W	0	DSS_PKTSIZE	This field is only used by the packet insertion logic to insert the Continuity count into the inserted packets at the correct place. 0 = DSS output packet is 130 bytes. 1 = DSS output packet is 140 bytes, of which the first 10 bytes comprise an additional header.
7:0	R/W	0xBC	PKTSIZE	Defines the packet size. Use 188 for DVB, 130 for DSS, and 140 for DSS with IEEE1394 padding.
Offset 0x11 8108			***INSTHRESH***	
31:16		-	Unused	
15:0	R/W	0	INS_THRESH	Defines the packet insertion threshold. If enabled, the packet insertion logic will insert packets into the GP/HS output interface after the number of packets specified in this field.
Offset 0x11 8110			***INSSTAT***	
31:2		-	Unused	
1	R	0	INSB_STATUS	Reports the insertion status for PKT-B. 0 = Packet B not being inserted. 1 = Packet B will be inserted next.
0	R	0	INSA_STATUS	Reports the insertion status for PKT-A. 0 = Packet A not being inserted. 1 = Packet A will be inserted next.
Offset 0x11 8114			***INSCOUNT***	
31:16		-	Unused	
15:0	R/W	0	INS_COUNT	Returns the current value of the packet insertion counter.
Offset 0x11 8118			***PKTADDR***	
31:9		-	Unused	
8:0	W	0	PKTADDR	Returns 0. PacketRAM address where data has to be written. Two bytes must be written into the PacketRAM at the same time. Addresses for packet A - 188 to 281 Addresses for packet B - 282 to 375
Offset 0x11 811C			***PKTDATA***	
31:16		-	Unused	
15:0	W	0	PKTDATA	Returns 0. Data to be written into the PacketRAM. Writing to this register also writes data into the RAM. The PKTADDR register must be written to first with the PacketRAM address before data is written in this register. PKTDATA[7:0] = lower addressed byte. PKTDATA[15:8] = higher addressed byte.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Packet Framer Registers				
Offset 0x11 8200			*FRMRCTL*	
31:16		-	Unused	
15	R/W	1	DVB_DSS	0 = The Framer is in DSS mode. 1 = The Framer is in DVB mode.
14	R/W	0	SYNDET	0 = Normal Sync detection 1 = Forces the Packet Framer into sync detection at the end of the current transport packet, even if the framer is in sync. The Packet Framer will clear this bit automatically on sync detection.
13	R/W	0	TEI_EN	Defines the behavior of the Packet Framer on Transport Error Indicator (TEI) detection for DVB packets only. No effect for DSS packets. 0 = If the TEI bit is '1', the entire transport packet is discarded and is not passed on to the RISC or used for PCR recovery 1 = Ignores the TEI bit in the TP header.
12	R/W	0	DSS140_IN	Defines the input DSS packet size. 0 = Input DSS packet size is 130 bytes. 1 = Input DSS packet size is 140 bytes. This is a playback from the IEEE 1394 interface. The first 10 bytes is the IEEE 1394 header followed by a 130-byte DSS packet.
11	R/W	0	DSS140_OUT	Defines the output DSS packet size. This bit is only effective when the DSS140_IN bit is a '1'. 0 = DSS output packets are 130 bytes. 1 = DSS output packets are 140 bytes. The first 10 bytes are a IEEE 1394 header followed by a 130-byte DSS Packet.
10	R/W	0	CHSYNC_DLY	Delays the CHSYNC input signal for 140-byte input DSS Packets by 10 clocks. This is only effective when the DSS140_IN bit is a '1'. 0 = No delay for SYNC. No PID matching is possible in the Packet Framer. 1 = SYNC is delayed by 10 clocks to line up with the first valid 130-byte DSS packet so that the Framer can do a PID Matching on a 140-byte input DSS Packet.
9	R/W	0	CHORD	Defines the bit order in the CHDATA[7:0] input. 0 = Sets bit 7 to be msb. 1 = Sets bit 0 to be msb.
8	R/W	0	CHERR_EN	0 = Ignores CHERR input signal. 1 = Reserved
7:0	R/W	47	SYNCBP	Defines the Sync byte pattern for DVB. The default is 47h. This field is not used for DSS.
Offset 0x11 8204			*FRMRSYNC*	
31:16		-	Unused	
15	R/W	0	FRAMEEN	0 = Disables Packet Framer. No data is passed to the RISC Engine. 1 = Enables Packet Framer. Data is passed to the RISC Engine.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
14	R/W	0	CHCLK_SYNC	0 = The CHCLK input signal is asynchronous to MSP Clock. The max frequency of the CHCLK signal cannot be more than 27 MHz. 1 = CHCLK is same frequency and is synchronous to MSP Clock.
13	R/W	0	SYNCSEL	DSS Synchronization Select 0 = Synchronizes with the Sync bit in the packet. 1 = Synchronizes with the CHSYNC input signal.
12	R/W		BYPASSEN	0 = Normal packet framing. All input data received in the channel interface is passed on to the RISC Engine after framing and synchronizing. 1 = Bypass. All input data is passed on to the RISC Engine without any framing or synchronizing.
11	R/W	-	BYTE_ALIGN	Force byte alignment - used for testing purposes. 0 = Normal operation - allows the Packet Framer to sync on any bit alignment. 1 = Byte alignment - forces the Packet Framer to align to the bytes that are presented on the parallel data input and search for the sync byte.
10		0	Reserved	
9:5	R/W	1	SYNCDRP	SYNC DROP - The number of consecutive Sync bytes (DVB) or bits (DSS) that must be lost to constitute a sync drop. (Sync bytes/bits are spaced one TP apart). A value of "0" is invalid.
4:0	R/W	1	SYNCLCK	SYNC LOCK - This field contains the number of consecutive sync bytes that must be detected before sync is locked. (Sync bytes/bits are spaced one TP apart). This value must be between 1 and 31. A value of ’0’ disables Sync Locking.
Offset 0x11 8208			*PIDMASK*	
31:16		-	Unused	
15:0	R/W	1FFF	PIDMASK	This field defines the 16-bit PID Masking pattern used by the PID Filter. A ’1’ in a bit field allows that bit to be compared, and a ’0’ masks that bit out. For DVB, use a value of 0x1FFF. For DSS, use a value of 0x0FFF. All other values must only be used for diagnostic purposes.
Offset 0x11 820C			*PIDCTL*	
31:13		-	Unused	
12:5	R/W	0	Reserved	
4	R/W	0	TSEN	0 = Disables Timestamp insertion. 1 = Enables Timestamp Insertion. If enabled, 4 extra bytes (32-bit timestamp) are inserted into the stream.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
3	R/W	0	PIDEN	0 = Disables PID filtering. If PID filtering is disabled, the entire packet is sent to the RISC Engine. 1 = Enables PID filtering. If enabled, two extra bytes denoting the PID match number are sent to the RISC engine with the entire packet.
2	R/W	0	NMTCHEN	0 = Drop packets are not matched by the PID filter. 1 = Send packets are not matched by the PID filter downstream for processing.
1	R/W	0	NUMPIDS	0 = Number of PIDs = 40 1 = Number of PIDs = 64
0	R/W	0	RSTPEAK	Ignore read value. 0 = No effect 1 = Resets FIFO Peak Detector in the PEAKVAL register.
Offset 0x11 8210			***PEAKVAL***	
31:8		-	Unused	
7:0	R	0	PEAKVAL	This field contains the maximum value of data in the FIFO. It is only intended to be used for debugging.
Offset 0x11 8214—8258			***Reserved***	
DSS Bit Rate Registers				
Offset 0x11 825C			***DSS_BITRATE***	
31:16		-	Unused	
15:0	R/W	0	DSSBITRATE	DSS Bit Rate [15:0] The value set by the host in this register is used by the Packet Framer as part of the 10-byte DSS packet header. Specifically, DSSBITRATE[15:8] is byte 4 and DSSBITRATE[7:0] is byte 5.

33.3.2 PID Filter Registers

The PID Filter contains 64 registers for the 64 PIDs. Each PID can be enabled or disabled. If a PID is disabled (PIDVALID = 0), the rest of the bits in that register are ignored. If a PID is enabled (PIDVALID = 1), any packet matching this PID will be sent to the RISC Engine. Additionally, if the 1394EN bit is set, this packet is sent to the GP/HS interface. If the PCRPKT bit is set, a PCR will be extracted from a packet with this PID (if a PCR is present in that packet). Every enabled PID in this table is unique. If two enabled PIDs have the same value, then the PID with the higher offset is ignored.

At power up, the entire PID table contents is treated as invalid and each of the PID registers is written with a ’0’ to disable all PIDs. Software can then load valid PIDs into these registers.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 8300—83FC			*PIDTABLE*	
31:16		-	Unused	
15	R/W	NI	PIDVALID	0 = PID entry is Invalid. 1 = PID entry is Valid.
14	R/W	NI	PCRPKT	0 = PID does not contain PCR. 1 = PID contains PCR. A PCR will be extracted from a packet with this PID (if a PCR is present in it).
13	R/W	NI	1394EN	0 = Do not route this packet to the GP/HS interface. 1 = This packet can be sent to the GP/HS interface.
12:0	R/W	NI	PID	13-bit PID value

33.3.3 RISC Engine Registers

The Risc Engine has various internal registers that can be read by the host CPU for debugging purposes. These registers are read-only and reflect the internal general purpose and special purpose register values and internal flags.

In addition, the RISC engine also has a control register to start/stop or to single step.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
RISC Engine Registers				
Offset 0x11 8400			*RISCDBG0*	
31:16		-	Unused	
15:8	R		RISC_R0	Returns the value of register r0.
7:0	R		RISC_R1	Returns the value of register r1.
Offset 0x11 8404			*RISCDBG1*	
31:16		-	Unused	
15:8	R		RISC_R2	Returns the value of register r2.
7:0	R		RISC_R3	Returns the value of register r3.
Offset 0x11 8408			*RISCDBG2*	
31:16		-	Unused	
15:8	R		RISC_R4	Returns the value of register r4.
7:0	R		RISC_R5	Returns the value of register r5.
Offset 0x11 840C			*RISCDBG3*	
31:16		-	Unused	
15:8	R	NI	RISC_R6	Returns the value of register r6.
7:0	R	NI	RISC_R7	Returns the value of register r7.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x11 8410			*RISCDBG4*	
31:16		-	Unused	
15:8	R	NI	RISC_RC1	Returns the value of register rc1.
7:0	R	NI	RISC_RC2	Returns the value of register rc2.
Offset 0x11 8414			*RISCDBG5*	
31:16		-	Unused	
15:8	R	NI	RISC_LOOPC	Returns the value of register loopc.
7:0	R	NI	RISC_TCCNT	Returns the value of register tccnt.
Offset 0x11 8418			*RISCDBG6*	
31:16		-	Unused	
15:8	R	NI	RISC_DSCRM	Returns the value of register de-scram.
7:0	R	NI	RISC_RP	Returns the value of register rp.
Offset 0x11 841C			*RISCDBG7*	
31:16		-	Unused	
15:0	R	NI	RISCCPUCTL	Returns the value of register cpuctl.
Offset 0x11 8420			*RISCDBG8*	
31:16		-	Unused	
9:0	R	NI	RISC_PC	Returns the value of register pc.
Offset 0x11 8424			*RISCDBG9*	
31:10		-	Unused	
9:0	R	NI	RISC_BTA	Returns the value of register bta.
Offset 0x11 8428			*RISCDBG10*	
31:10		-	Unused	
9:0	R	NI	RISC_BBTA	Returns the value of register bbta.
Offset 0x11 842C			*RISCDBG11*	
31:8		-	Unused	
7:0	R	NI	RISC_FLAGS	Returns the value of risc flags. Bit 7 = ZERO Bit 6 = ZERO Bit 5 = GT Bit 4 = GT Bit 3 = LT Bit 2 = LT Bit 1 = OVR Bit 0 = LOOPC_ZERO
Offset 0x11 8430			*RISCCTL*	
31:2		-	Unused	
1	R/W	1	RISC_STOP	0 = RISC is running. 1 = RISC is stopped.

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
0	W	0	RISC_SSTEP	Always returns 0. 0 = No Effect 1 = Forces RISC Engine to execute one instruction.
Offset 0x11 8434			*HSFCTL*	
31:2		-	Unused	
1:0	R/W	0	SEC_MODE	Defines the Section Filtering Type. 00 = 8-byte Section Filter Type 01 = 12-byte Section Filter Type 10 = 8-byte Section Range Filter Type 11 = Reserved

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
De-Scrambler Keys				
*Offset 0x11 8600 + n*8*			*KEYnEVN0 (n = 0 to F)*	
31:0	R/W	NI	KEY_EVN [31:0]	De-Scrambler Even Key. At power up, the de-scrambler key table is invalid and must be initialized by software.
*Offset 0x11 8604 + n*8*			*KEYnEVN1 (n = 0 to F)*	
31:0	R/W	NI	KEY_EVN [63:32]	De-Scrambler Even Key. At power up, the de-scrambler key table is invalid and must be initialized by software.
*Offset 0x11 8680 + n*8*			*KEYnODD0 (n = 0 to F)*	
31:0	R/W	NI	KEY_ODD [31:0]	De-Scrambler Odd Key. At power up, the de-scrambler key table is invalid and must be initialized by software.
*Offset 0x11 8684 + n*8*			*KEYnODD1 (n = 0 to F)*	
31:0	R/W	NI	KEY_ODD [63:32]	De-Scrambler Odd Key. At power up, the de-scrambler key table is invalid and must be initialized by software.
Multi-2 and Copy Protect Keys				
Offset 0x11 8700			*M2SYSKEY0*	
31:0	R/W	0	SYS_KEY [31:0]	Multi-2 System Key
Offset 0x11 8704			*M2SYSKEY1*	
31:0	R/W	0	SYS_KEY [63:32]	Multi-2 System Key
Offset 0x11 8708			*M2SYSKEY2*	
31:0	R/W	0	SYS_KEY [95:64]	Multi-2 System Key
Offset 0x11 870C			*M2SYSKEY3*	
31:0	R/W	0	SYS_KEY [127:96]	Multi-2 System Key
Offset 0x11 8710			*M2SYSKEY4*	
31:0	R/W	0	SYS_KEY [159:128]	Multi-2 System Key
Offset 0x11 8714			*M2SYSKEY5*	

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:0	R/W	0	SYS_KEY [191:160]	Multi-2 System Key
Offset 0x11 8718			*M2SYSKEY6*	
31:0	R/W	0	SYS_KEY [223:192]	Multi-2 System Key
Offset 0x11 871C			*M2SYSKEY7*	
31:0	R/W	0	SYS_KEY [255:224]	Multi-2 System Key
Offset 0x11 8720			*CPKEYEVEN0*	
31:0	R/W	0	CP_KEY_EVN [31:0]	Copy Protection Even Key
Offset 0x11 8724			*CPKEYEVEN1*	
31:0	R/W	0	CP_KEY_EVN [63:32]	Copy Protection Even Key
Offset 0x11 8728			*CPKEYODD0*	
31:0	R/W	0	CP_KEY_ODD [31:0]	Copy Protection Odd Key
Offset 0x11 872C			*CPKEYODD1*	
31:0	R/W	0	CP_KEY_ODD [63:32]	Copy Protection Odd Key
De-Scrambler/Copy Protect Initialization Vectors				
Offset 0x11 8730			*DSCIVCBC0*	
31:0	R/W	0	DSC_IV_CBC [31:0]	De-scrambler IV CBC
Offset 0x11 8734			*DSCIVCBC1*	
31:0	R/W	0	DSC_IV_CBC [63:32]	De-scrambler IV CBC
Offset 0x11 8738			*DSCIVOFB0*	
31:0	R/W	0	DSC_IV_OFB [31:0]	De-scrambler IV OFB
Offset 0x11 873C			*DSCIVOFB1*	
31:0	R/W	0	DSC_IV_OFB [63:32]	De-scrambler IV OFB
Offset 0x11 8740			*CPIVCBC0*	
31:0	R/W	0	CP_IV_CBC [31:0]	Copy Protect IV CBC
Offset 0x11 8744			*CPIVCBC1*	
31:0	R/W	0	CP_IV_CBC [63:32]	Copy Protect IV CBC
Offset 0x11 8748			*CPIVOFB0*	
31:0	R/W	0	CP_IV_OFB [31:0]	Copy Protect IV OFB
Offset 0x11 874C			*CPIVOFB1*	
31:0	R/W	0	CPIV_OFB [63:32]	Copy Protect IV OFB
De-Scrambler Modes				
Offset 0x11 8750			*MODEPAD*	
31:21		-	Unused	

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
20:16	R/W	0	CP_MODE [4:0]	Copy Protect Mode Full BlockResidual BlockSolitary Block 0x0 = NoneNoneNone 0x1 = DES-ECBNoneNone 0x2 = DES-ECBDES-OFB(PB)None 0x3 = DES-ECBDES-OFB(PB)DES-OFB(IV) 0x4 = DES-ECBDES-OFB(IV)None 0x5 = DES-ECBDES-OFB(IV)DES-OFB(IV) 0x6 = ReservedReservedReserved 0x7 = DES-CBCNoneNone 0x8 = DES-CBCDES-OFB(PBNone 0x9 = DES-CBCDES-OFB(PB)DES-OFB(IV) 0xa = DES-CBCDES-OFB(IV)None 0xb = DES-CBCDES-OFB(IV)DES-OFB(IV) 0xc—0xf =ReservedReservedReserved
15:13		-	Unused	
12:8	R/W	0	DSC_MODE [4:0]	De-Scrambler Mode Full BlockResidual BlockSolitary Block 0x00 = NoneNoneNone 0x01 = DES-ECBNoneNone 0x02 = DES-ECBDES-OFB(PB)None 0x03 = DES-ECBDES-OFB(PB)DES-OFB(IV) 0x04 = DES-ECBDES-OFB(IV)None 0x05 = DES-ECBDES-OFB(IV)DES-OFB(IV) 0x06 = ReservedReservedReserved 0x07 = DES-CBCNoneNone 0x08 = DES-CBCDES-OFB(PB)None 0x09 = DES-CBCDES-OFB(PB)DES-OFB(IV) 0x0a = DES-CBCDES-OFB(IV)None 0x0b = DES-CBCDES-OFB(IV)DES-OFB(IV) 0x0c = ReservedReservedReserved 0x0d = DVBDVBDVB 0x0e = MULTI2 MULTI2MULTI2 0x0f—0x10 = ReservedReservedReserved 0x11 = 3DES-ECBNoneNone 0x12 = 3DES-ECB3DES-OFB(PB)None 0x13 = 3DES-ECB3DES-OFB(PB)3DES-OFB(IV) 0x14 = 3DES-ECB3DES-OFB(IV)None 0x15 = 3DES-ECB3DES-OFB(IV)3DES-OFB(IV) 0x16 = ReservedReservedReserved 0x17 = 3DES-CBCNoneNone 0x18 = 3DES-CBC3DES-OFB(PB)None 0x19 = 3DES-CBC3DES-OFB(PB)3DES-OFB(IV) 0x1a = 3DES-CBC3DES-OFB(IV)None 0x1b = 3DES-CBC3DES-OFB(IV)3DES-OFB(IV) 0x1c—0x1f = ReservedReservedReserved
7:0	R/W	0	PAD_CHAR	Pad Character
Offset 0x11 8754			***Reserved***	
Offset 0x11 8758			***Reserved***	

33.3.4 PI-Bus Configuration Registers

The PI-Bus Configuration registers are a standard set of registers in a device's configuration address space for the purpose of providing automatic configuration by software. These registers reside just below the first 4 kB region, which includes information on the module size, the module ID, and the module interrupt registers

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
PI-Bus Configuration Registers				
Offset 0x11 8C00			*MSPRST*	
31:5		-	Unused	
4	W	NI	RST_GPHS	Ignore read value. 0 = No Effect 1 = Writing a '1' will automatically reset the GP/HS. This bit will auto-clear.
3	W	NI	RST_MQM	Ignore read value 0 = No Effect 1 = Writing a '1' will automatically reset the MQM. This bit will auto-clear.
2	W	NI	RST_DESC	Ignore read value 0 = No Effect 1 = Writing a '1' will automatically reset the de-scrambler. This bit will auto-clear.
1	W	NI	RST_RISC	Ignore read value 0 = No Effect 1 = Writing a '1' will automatically reset the RISC Engine. This bit will auto-clear.
0	W	NI	RST_FRMR	Ignore read value 0 = No Effect 1 = Writing a '1' will automatically reset the Packet Framer and the PCR logic. This bit will auto-clear.

33.3.5 MSP Interrupt Registers

All of the interrupt registers have 20 bits, two groups of 10. The ten bits as shown in Table 4 on page 33-696 are the ten internal interrupt sources. In all the registers, bits 0 and 10, 1 and 11, 2 and 12, 3 and 13, etc. are the same internal sources. Software must be careful to mask/ignore one of each pair.

MSP 1 REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
MSP Interrupt Registers				
Offset 0x11 8FE0			***STB_MSPINTST***	
31:20		-	Unused	
19:0	R	0	STB_MSPINTST	The MSP Interrupt Status This register allows software to check an interrupt status for pending interrupts. Each bit controls each internal interrupt source. 0 = Corresponding interrupt is not pending. 1 = Corresponding interrupt is pending.
Offset 0x11 8FE4			***STB_MSPINTENA***	
31:20		-	Unused	
19:0	R/W	0	STB_MSPINTENA	The MSP Interrupt Enable This register allows software to selectively enable an interrupt. Each bit controls each internal interrupt source. 0 = Corresponding interrupt is disabled. 1 = Corresponding interrupt is enabled.
Offset 0x11 8FE8			***STB_MSPINTCLR***	
31:20		-	Unused	
19:0	W	NI	STB_MSPINTCLR	The MSP Interrupt Clear This register allows software to reset an interrupt. Each bit controls each internal interrupt source. Ignore read data. 0 = Corresponding interrupt is not cleared. 1 = Corresponding interrupt is cleared.
Offset 0x11 8FEC			***STB_MSPINTSET***	
31:20		-	Unused	
19:0	W	NI	STB_MSPINTSET	The MSP Interrupt Set This register allows software to set an interrupt. Each bit controls each internal interrupt source. Ignore read data. 0 = Corresponding interrupt is not set. 1 = Corresponding interrupt is set.
Offset 0x11 8FF4			***POWERDOWN***	
31	R/W	0	POWER_DOWN	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0		-	Unused	Ignore during writes and read as zeroes.
Offset 0x11 8FFC			***MODULEID***	
31:16	R	0x010E	MODULEID [15:0]	Returns "0x010E"

MSP 1 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:12	R	0x2	MAJREV [3:0]	Major Revision of the MSP. 0 = PNX8526 rev A 2 = PNX8525 rev B with ICAM 4 = PNX8525 rev B without ICAM
11:8	R	0	MINREV [3:0]	Minor Revision of the MSP
7:0	R	0x07	APERSIZE [7:0]	0x07 = Returns ”0x07.” The MSP needs an aperture size of 32 kB.

MSP 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
MSP 2 Registers begin at 0x12 0000. They are identical to the MSP 1 registers, described above.				

MSP 3 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
MSP 3 Registers begin at 0x12 8000. They are identical to the MSP 1 registers, described above. Note that MSP 3 does NOT implement the ICAM.				

Chapter 34: MPEG Video Decoder

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

34.1 Introduction

The MPEG unit, also known as the MPEG-PIPE, is a slice-level video decoder comprised of a pipeline of functional blocks. It supports MPEG2 video decoding at a maximum resolution of main profile at high-level (MP@HL) and also MPEG1 video decoding. It is capable of decoding all eighteen video formats of the ATSC digital television standard. The MPEG-PIPE does not support the D picture-type in MPEG1; however, the decoding of a D picture can be handled by the TM32 CPU core.

In order to save main memory usage, the MPEG-PIPE can optionally decode the video bitstream in half-horizontal resolution mode. The amount of main memory storage required in half-resolution mode is half of what is required in full-resolution (normal) mode.

The video lines of each field (top or bottom) of all picture types (I, P and B) are normally stored in one contiguous main memory block. To optimize main memory usage in decoding MPEG2 video however, each field of a type B picture can be optionally divided into two, four or eight chunks of video lines, where each chunk can be stored in a location in main memory that is non-contiguous with other chunks. Chunking of type B picture fields saves main memory usage. The buffer space of a chunk is reused immediately after it has been displayed.

Another important feature of the MPEG-PIPE is its ability to perform error concealment. For terrestrial ATSC broadcast, errors in transmission can be quite frequent, especially in areas having poor reception. Error concealment is performed by patching the missing pixels (or errors) with corresponding pixels from the stored backward or forward reference frames.

The MPEG-PIPE is a slice-level decoder. It decodes all the macroblocks in a given slice, reconstructs the pixels of each macroblock and stores the pixels in designated locations in main memory. All other aspects of MPEG decoding functions are carried out by the TM32 CPU core (with proper programming). This includes parsing of the sequence header down to the slice header of the MPEG stream. The relevant information decoded by software, such as picture type, picture structure, quantization matrix, etc. must be programmed into corresponding MMIO registers to direct the MPEG-PIPE to decode the macroblocks within each slice. Some other functions performed by the TM32 CPU core relating to MPEG decoding are as follows:

- Frame buffer management
- Trick modes, such as frame/field freeze implementation
- Decoding of I frames only for mosaic implementation
- Slow motion and fast forward or fast reverse implementation

34.2 MPEG-PIPE Overview

The MPEG-PIPE consists of a Variable Length Decoder (VLD) block, a Run-Level Decoder and Inverse Scan (RL/IS) block, an Inverse Quantization (IQ) block, an Inverse Discrete Cosine Transform (IDCT) block and a Motion Compensation (MC) block.

The VLD decodes the Huffman-encoded MPEG1 and MPEG2 video elementary bitstreams. The VLD unit, enabled by the TM32 CPU core, operates independently during slice-level decoding. The remaining decoding of the bitstream is carried out by the TM32 CPU core, and the VLD assists the TM32 CPU core in slave mode. The VLD outputs a stream of macroblock headers and a stream of run-level pairs. The VLD sends the macroblock header stream to the MC block and the run-level stream to the RL/IS block for further decoding.

Expansion of the run-level pairs produced by the VLD into actual quantized DCT values and rearrangement of these values in natural order are carried out by the RL/IS block.

The IQ block is responsible for restoring or “dequantizing” the quantized DCT values by multiplying them by the corresponding values in the 8x8 DCT quantization matrix. The IQ block also optionally performs the frequency domain filtering in association with the half-resolution mode.

The output of the IQ block is passed to the IDCT block, which performs the inverse discrete cosine transformation on each of the 8x8 dequantized blocks. The output data from the IDCT process is either the video samples or the differential values to be used by the MC block to reconstruct the final video samples.

The MC reconstructs the final video samples from the macroblock header data decoded by the VLD, the reference frame data stored in main memory and the differential data from the IDCT. Half-resolution compression and error concealment are carried out mainly in the MC block.

34.3 VLD Operation

After initialization, the TM32 CPU core controls the VLD through the VLD command register. The normal mode of operation is for the TM32 CPU core to request the VLD to parse some number of macroblocks.

Once the VLD has begun parsing macroblocks, it may stop for one of the following reasons:

- The command was completed with no exceptions
- A start code was detected
- An error was encountered in the bitstream
- The input DMA was completed and the VLD is stalled waiting for more data

Under normal circumstances, the TM32 CPU core can be interrupted whenever the VLD halts.

Consider the case in which the VLD has encountered a start code. At this point, the VLD will halt and set the status flag which indicates that a start code has been detected. This flag will generate an interrupt to the TM32 CPU core. Upon entering the interrupt service routine, the TM32 CPU core will read the VLD status register to determine the source of the interrupt. Once it has been determined that a start code has been encountered, the CPU will read 8 bits from the VLD shift register to determine the type of start code that has been encountered. If a slice start code has been encountered, the TM32 CPU core will read from the shift register the slice quantization scale code and any extra slice information (from the slice header). The slice quantization scale code will then be written back to the VLD_QS register. Before exiting the interrupt service routine, the TM32 CPU core will clear the start code detected status bit in the status register and issue a new command to process the remaining macroblocks.

34.3.1 VLD Input

The VLD reads the video bitstream from main memory and performs the variable length decoding process. The TM32 CPU core writes the main memory buffer address from which the bitstream will be read by the VLD in the VLD_INP_ADR register. The number of bytes to be read by the VLD is updated by the TM32 CPU core in the VLD_INP_CNT register.

The VLD unit uses two 64-byte buffers to store the input bitstream. The VLD reads the bitstream data from the main memory and updates the VLD_INP_ADR and the VLD_INP_CNT register. The content of the VLD_INP_ADR register reflects the next fetch address of the bitstream data. The content of the VLD_INP_CNT register reflects the number of bytes to be read from the main memory. When the number of bytes to be read from the main memory transitions from non-zero to zero, the DMA_INPUT_DONE flag in the VLD_MC_STATUS is set. An interrupt will be sent to the TM32 CPU core also if the corresponding interrupt enable bit in the VLD_IE register is set. The TM32 CPU core should then provide the new bitstream buffer address and the number of bytes in the bitstream buffer to the VLD.

34.3.2 VLD Output

The output of the VLD is sent to the downstream pipeline blocks. Macroblock headers are passed to the MC block and the run-level encoded DCT coefficients to the RL/IS block for further decoding of the macroblocks.

34.3.3 VLD Registers

34.3.3.1 VLD Status (VLD_MC_STATUS)

The VLD_MC_STATUS register contains current status information which is pertinent to the normal operation of an MPEG video decoder application. Writing a logic ‘1’ to any of the status bits (other than bit-0) clears the corresponding bit. Writing a logic ‘0’ has no effect. *Exception: Bit 0 (Command Done) is cleared only by issuing a new command. Writing a logic ‘1’ to bit zero of the status register will result in undefined behavior of the VLD*.

Remark: Several status bits may be asserted simultaneously.

When an error occurs in the VLD or RL/IS, the corresponding error flag (Bitstream error or RL overflow error) is set and an interrupt is generated (if the corresponding bit in the VLD_IE register is set). See Section 34.3.5 on page 34-725 for details on the error handling mechanism.

34.3.3.2 VLD Interrupt Enable (VLD_IE)

This VLD_IE read/write register allows the TM32 CPU core to control the initiation of the interrupt for the corresponding bits in the VLD_MC_STATUS register. Writing a one to any of these bits in the VLD_IE register enables the interrupt for the corresponding bit in the status register.

34.3.3.3 VLD Control (VLD_CTL)

When VLD detects a new slice start code in the bitstream, it writes the lower 8 bits of the start code into the slice_start_code field of the VLD_CTL register before interrupting the CPU. When restarted, VLD reads the slice_start_code from the VLD_CTL register and writes them into bits 16-23 of the last word in the first mb_header and sets the mb_first bit to 1. To allow the CPU to switch bitstreams at the slice level, CPU can write the desired slice start code and slice_start_code_strobe in the VLD control register. The value of ‘1’ in the slice_start_code_strobe will cause the VLD to update the slice_start_code field with the given slice_start_code value. The other fields in the VLD_CTL register are not updated when the input data contains a value of ‘1’ in the slice_start_code_strobe field. The slice_start_code_strobe bit is always read as 0. The CPU will write the slice_start_code only when the VLD is not active.

In order to update the DMA_INPUT_DONE_MODE fields, the slice_start_code_strobe bit value must be set to ‘0’.

34.3.3.4 VLD DMA Current Read Address (VLD_INP_ADR) and VLD DMA Current Read Count (VLD_INP_CNT)

The TM32 CPU core writes the main memory buffer address where the bitstream will be read in the VLD_INP_ADR register. The number of bytes to be read by the VLD is updated by the TM32 CPU core in the VLD_INP_CNT register.

The VLD unit uses two 64-byte buffers to store the input bitstream. The VLD reads the bitstream data from the main memory and updates the VLD_INP_ADR and the VLD_INP_CNT register. The content of the VLD_INP_ADR register reflects the current or the next fetch address of the bitstream data.

VLD interrupts the TM32 CPU core when it has consumed all the given bitstream data in the main memory (the DMA_INPUT_DONE condition). The value of the DMA_INPUT_DONE_MODE bit in the VLD_CTL register is used to select the condition for raising the DMA_INPUT_DONE flag.

The VLD input address is word-aligned and the count value in number of bytes is also word aligned.

34.3.3.5 VLD Command (VLD_COMMAND)

This read/write register indicates the next action to be taken by the VLD. A command is sent to the VLD by writing the corresponding 4-bit command code in the COMMAND field of the VLD_COMMAND register. Some commands require an associated count value which resides in the least significant 8 bits of this register. The following VLD commands are available:

Table 1: VLD Commands

Code	Command	Flags Set after Completion of the Command	Description
0x1	Shift the bitstream by “‘count” bits	Command Done	VLD shifts the number of bits in its internal shift register. The shift register value is available in the VLD_SR register. The flag is reset by issuing the new command.
0x2	Parse for a given number of macroblocks	Command Done and/or Start Code Detected	VLD parses for a given number of macroblocks,but if VLD encounters a start code, the parsing action will be terminated and VLD will set only the start code detected flag. If VLD parses the given number of macroblocks without encountering a start code, VLD will set the command done flag. The start code detected flag is reset by writing a ‘1’ value to the flag. The command done flag is reset by issuing the new command
0x3	Search for the next start code	Start Code Detected and Command Done	VLD search for a start code. The search code has a 0x000001 prefix and an additional 8-bit value. The start code detected flag is reset by writing a ‘1’ value to the flag. The command done flag is reset by issuing the new command
0x4	Reset MPEG-PIPE	Command Done	See Section 34.3.5 on page 34-725.
0x5	Initialize VLD	None	The bit count register is initialized to zero. The initialization action is immediate without any delay.
0x6	Search for the given start code	Start Code Detected and Command Done	VLD search for a start code with a given 8-bit lsb of the 32-bit start code. The search code has 0x000001 prefix and an additional 8-bit value is given in the ‘count’ field of the VLD_COMMAND register. The start code detected flag is reset by writing a ‘1’ value to the flag. The command done flag is reset by issuing the new command
0x7	Parse one row of macroblocks	Start Code Detected	This command instructs the VLD to parse one complete row of macroblocks. If the row contains more than one slice, VLD parses the intermediate slice headers without CPU intervention, provided these slice headers have a 0 bit after the 5-bit quantizer_scale_code. If the VLD encounters a start code different from the start code of the current slice, or if the slice header has a 1 bit after the quantizer_scale_code, it sets the Start-Code-Detected flag and ends the operation. *WARNING:* ‘Count’ field of the VLD_COMMAND register is still in effect as in the ‘Parse A Number Of Macroblocks’ Command. VLD stops and sets the Command-Done flag after ‘Count’ macroblock headers are parsed. ‘Count’ must be set to at least mb_width (number of macroblocks per row in the picture) to guarantee entire row is parsed before the VLD stops.

The TM32 CPU core must wait for the VLD to halt before the next command can be issued. Note that there are several ways in which a command may be completed. Only a successful completion is indicated by the Command Done bit in the status register. A command may complete unsuccessfully if a start code or an error is encountered before the requested number of items has been processed. Also, expiration of a DMA count does not constitute completion of a command. When a DMA count expires, the VLD is stalled as it waits for a new DMA to be initiated. It is not halted.

The ‘Initialize VLD’ command initializes VLD_BIT_CNT register (bit counter) to zero.

The ‘search for the given start code’ command searches for the start code pattern given in the COUNT field (in the least significant 8 bits) of the VLD_COMMAND register. A valid MPEG start code is a 32-bit pattern with the upper 24 bits equal to 0x000001. For each start code encountered in the bitstream, the 8 bits following the 0x000001 pattern are compared against the given 8-bit start code pattern until a perfect match is obtained. Once the given start-code is found, the VLD sets the COMMAND_DONE bit in the VLD_MC_STATUS register. At that point, the VLD will also interrupt the TM32 CPU core if the corresponding bit in the VLD_IE register is set.

34.3.3.6 VLD Shift Register (VLD_SR)

This read-only register is a shadow of the VLD’s operational shift register. It allows the TM32 CPU core to access the bitstream through the VLD. Bits 0 through 15 are currently in use in the VLD shift register. Bits 16 to 31 are reserved.

34.3.3.7 VLD Quantizer Scale (VLD_QS)

This 5-bit register read/write register contains the quantization scale code to be output by the VLD until it is overridden by a macroblock quantizer scale code. The quantizer scale code is part of the macroblock header output.

34.3.3.8 VLD Picture Info (VLD_PI)

This 32-bit read/write register contains the picture layer information necessary for the VLD (and MC) to parse the macroblocks within that picture. Again, the value of each of these fields is determined by the appropriate standard (MPEG1 or MPEG2)

34.3.3.9 VLD Bit Count (VLD_BIT_CNT)

The number of bits consumed by the VLD is updated in the VLD_BIT_CNT register. It counts upward when bits are shifted out and consumed by the VLD. This counter wraps around after reaching the maximum value. VLD_BIT_CNT can be initialized to zero by issuing the ‘Initialize VLD’ command.

34.3.4 Interrupt

The interrupt source number for the MPEG-PIPE on the TM32 Vectored Interrupt Controller (VIC) is 14.

34.3.5 Error Handling

The VLD can generate two types of errors and the MC can generate six types of errors. The VLD_MC_STATUS register has two VLD error flags and six motion compensation error flags.The VLD errors are 1) bitstream parsing error and 2) run-level overflow error. See Table 2 for details on the VLD error handling procedure when full MPEG-PIPE is in operation i.e., when the write-to-memory bit is not set in the VLD_CTL register.

34.3.5.1 Unexpected Start Code

When the VLD encounters an unexpected start code, the VLD sets the ‘Start Code Detected’ and ‘Bitstream Error’ flags (in the VLD_MC_STATUS register). The start code value is left in the shift register (VLD_SR). The VLD interrupts the TM32 CPU core if one of the corresponding interrupt bits in the VLD_IE register is enabled.

Table 2: VLD Error Handling

Cycle No.	Action	Remarks
i	VLD sets appropriate error bit in the VLD_MC_STATUS register	The *vld_mc_error* signal is formed by ORing together all the error bits in the VLD_MC_STATUS register. Hence any MC error also drives the vld_mc_error signal high and the following error handling steps still apply.
i to j	When the *vld_mc_error* signal is high, VLD will complete any pending MMIO or memory highway transaction. The valid data in the highway output buffers, in ‘write-to-memory’ mode, will be flushed to main memory. Then VLD will assert the *vld_ready_to_reset* signal and wait for the TM32 CPU core to reset the MPEG-PIPE.	Any highway transaction, once started, will not be aborted.
i to k	After receiving the *vld_mc_error* signal, MC will complete any pending MMIO or memory highway transaction. Then MC will assert the *mc_ready_to_reset* signal and wait for the TM32 CPU core to reset the MPEG-PIPE.	Any highway transaction, once started, will not be aborted.
k	If (vld_ready_to_reset AND mc_ready_to_reset) occur, then VLD interrupts the TM32 CPU core.	Assumes k > j; otherwise it is cycle j. The corresponding IE bit in VLD_IE register must be enabled.
l	TM32 CPU core will perform the software reset.	See Table 3 on page 34-728 for the SW reset procedure.

The error handling in the VLD will be synchronized with the error handling in the MC block as well. Synchronization of error handling in the VLD and MC is achieved through the use of three interface signals between them: the *vld_mc_error*, the *vld_ready_to_reset*, and the *mc_ready_to_reset* signal. These internal signals are active high and low after hardware or software reset.

As soon as it encounters an error, VLD sets the appropriate error flag in the VLD_MC_STATUS register. The vld_mc_error signal is obtained by ORing together all the MC and VLD error flags. Hence any MC error will also drive the vld_mc_error signal high and the subsequent VLD error handling steps apply as well. As soon as the vld_mc_error signal is high, VLD completes any outstanding highway (MMIO or main memory) transaction and then asserts the vld_ready_to_reset signal to the MC. Meanwhile MC also performs similar error handling steps when the vld_mc_error_signal goes high and eventually asserts the mc_ready_to_reset signal to the VLD (see Section 34.5 on page 34-733 for more details). When the vld_ready_to_reset and mc_ready_to_reset signals are both high, VLD interrupts the TM32 CPU core if any of the error interrupt enable bits (in the VLD_IE register) is set. Then the TM32 CPU core can issue a ‘Flush the VLD output buffers’ command to optionally empty the VLD output buffer before sending a software reset command to the VLD.

34.3.5.2 MC Flush

The MPEG-PIPE stores the intermediate data at various stages within the MPEG-PIPE. In order to terminate the decoding process at a particular point of the bitstream, the TM32 CPU core will issue the *flush_cmd* by setting the *flush_cmd* bit in the MC_COMMAND register. The *flush_cmd* is issued in two cases:

- When the bitstream contains error bits for a known amount of bytes and would like to terminate the decoding at a particular byte-offset of the bitstream buffer. The TM32 CPU core will set the DMA_input_done_mode bit to ‘1’ in the VLD_CTL register.
- When the TM32 CPU core decides to switch the bitstream at the start of a new slice-start-code in a new row.

The flush_cmd will be issued when the VLD stops after interrupting the TM32 CPU core for the dma_input_done reason, in the first case, and when the VLD stops after interrupting the TM32 CPU core for the start_code_detected reason, in the second case. When VLD detects the assertion of the flush_cmd bit in the MC_COMMAND register, the RL/IS, IQ, and IDCT blocks continue their operations until they have exhausted their input data buffers and then go into their respective idle states. Then the VLD unit raises the *vld_mc_done_flush* signal to the MC. The MC block processes the flush command and sets the DONE_FLUSH bit in the VLD_MC_STATUS register when the flush operation is completed. The VLD unit de-asserts the *vld_mc_done_flush* signal only after the TM32 CPU core clears the *DONE_FLUSH* bit in the VLD_MC_STATUS register. Any status bit can be cleared by writing a ‘1’ into the status bit.

There may be some partial data in the RL/IS-IQ-IDCT chain and in the macroblock header buffers (in the first case) and there will not be any partial data in the RL/IS-IQ-IDCT chain or in the macroblock header buffers (in the second case). The TM32 CPU core should issue a ‘Reset MPEG-PIPE’ command when flush is completed in the first case. Software reset is optional in the second case.

34.3.5.3 Timeout

The timeout mechanism is used by the MPEG-PIPE in order to interrupt the TM32 CPU core when the MPEG-PIPE does not make the expected progress in decoding the given bitstream. The expected MPEG-PIPE performance is given in the MC_PICINFO0 register. See Section 34.5.16 on page 34-741 for more details.

The current implementation of this timeout mechanism relies on two interface signals, *mc_timeout* and *reset_mc_timeout* between the VLD and the MC, plus a timeout down-counter in the MC block. The reset_mc_timeout signal is obtained by ORing together all the bits in the VLD_MC_STATUS register. As long as reset_mc_timeout is low, the timeout counter continues to decrement once per cycle until it reaches 0. The timeout counter stops if its current value is 0 or when reset_mc_timeout is high. At the falling edge of reset_mc_timeout, the timeout counter is reset to a number N, according to the value specified in the 4-bit *mc_time_period* field of the MC_PICINFO0 register, where N = 2048*mc_timeout_period. *mc_timeout_period* should be set to zero in the ‘Write-to-memory’ mode to effectively disable the timeout mechanism.

The MC asserts the mc_timeout signal to the VLD for one cycle when no progress is detected in the MC. See Section 34.5.16 for details regarding how progress is detected in the MC block. When the mc_timeout signal is asserted, VLD sets the TIMEOUT bit in the VLD_STATUS register only if the VLD is also in a parsing mode (that is, when the VLD is in the middle of executing a parse command). When the TIMEOUT bit is set, the VLD also raises an interrupt to the TM32 CPU core, if the corresponding interrupt enable bit is set in the VLD_IE register.

Remark: Timeout will not occur when the MC is carrying out an ERRCON2 command (type 2 error concealment). The VLD is idle in this case.

34.3.5.4 Reset

The MPEG-PIPE can be reset by a hardware or software reset. The hardware reset signal is generated from the global reset pin. The software reset is initiated by writing a 'Reset MPEG_PIPE' command in the VLD_COMMAND register. See Table 3 for details.

The COMMAND_DONE bit in the VLD_MC_STATUS register is set upon completion of a software reset, which will lead to an interrupt to the TM32 CPU core (if the corresponding interrupt is enabled).

Remark: The VLD_INP_CNT is cleared after reset. However, this is not treated as a DMA_INPUT_DONE condition in the VLD, and the TM32 CPU core will not be interrupted by the VLD after reset with a DMA_INPUT_DONE condition. The MPEG video bitstream buffer should be refilled as needed and the VLD_INP_CNT rewritten with a proper value before issuing new commands to the VLD after reset.

Table 3: Software Reset Procedure

Cycle No.	Action	Remarks
i	TM32 CPU core issues 'Reset the MPEG-PIPE' command by writing the corresponding command code into the VLD_COMMAND register.	

Table 3: Software Reset Procedure *…Continued*

Cycle No.	Action	Remarks
i to j	1) VLD will complete any MMIO or memory highway transaction in progress. 2) Any new highway transaction is aborted. 3) VLD raises the *vld_ready_to_reset* signal.	Any highway transaction, once started, will not be aborted midway.
i to k	1) MC will complete any MMIO or memory highway transaction in progress. 2) Any new highway transaction is aborted. 3) MC raises the *mc_ready_to_reset* signal.	Any highway transaction, once started, will not be aborted midway.
k	If (vld_ready_fo_reset AND mc_ready_to_reset) occurs, the VLD and MC will perform the full reset.	Assumes k > j; otherwise it is cycle j. The full reset resets all internal buffers, state machines. After the reset, the VLD_MC_STATUS register will have 0x1 value and the following MMIO registers will have 0x0 value: VLD_COMMAND, VLD_CTL, VLD_BIT_CNT, MC_STATUS, MC_PFCOUNT, MC_COMMAND, VLD_INP_CNT, VLD_MBH_CNT, VLD_RL_CNT and VLD_STATUS have 0x1 value

34.4 Run-Length Decoder/Inverse Scan Overview

Run-Length Decoder/Inverse Scanning (RL/IS) and Inverse Quantization (IQ) follow the VLD. It takes the VLD run-level output and generates scanned 8x8 DCT coefficient blocks, puts them back in natural order, inverse-quantizes them, then passes the reconstructed 8x8 two-dimensional matrices to the IDCT block.

The RL Decoder finishes a block when one of the following is encountered:

- A run value of EOB (end of block) means successful decode of a block.
- The total accumulated run is equal to or greater than 64. In this case, the RL Decoder has encountered an out-of-bounds error, and the error flag in the status register is set.

After the RL Decoder step, inverse scan is performed on the RL Decoder output. There are two types of scans: Zig-Zag scan and Alternate scan. The type of inverse scan used is controlled by the ALTERNATE_SCAN bit in the control register.

Inverse Quantization is the step that dequantizes the DCT coefficients based on the quantization matrix, the quantizer_scale, sign of the quantized DCT coefficient, and MPEG coding type (MPEG1/MPEG2). After the arithmetic process, saturation and mismatch control are performed on the results to generate the final IQ output and pass the results to the IDCT block.

34.4.1 Run-Length Decoder

When an error is encountered, such as the DCT coefficient index is out of bounds, the Run-Length Decoder sets the error flag in the status register and generate an interrupt to inform the TM32 CPU core that the corresponding interrupt enable bit is set in the control register.

34.4.2 Inverse Scan

The inverse scan step is processed in the same pipeline stage with the Run-Length Decoder. The ALTERNATE_SCAN bit in the control register determines whether zig-zag or alternate scan is applied to the current picture. The zig-zag scan table is as follows:

Table 4: Zig-Zag Scan Table

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

The alternate scan is as follows:

Table 5: Alternate Scan Table

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

34.4.3 Inverse Quantization

Inverse Quantization provides the following:

- Two quantization matrices are mapped into MMIO space, initialized with default matrices but can be overwritten by downloadable matrices at picture level and sequence level.
- Global quantizer scale can be updated at slice and MB levels.
- Supports both MPEG1 and MPEG2 mode Inverse Quantization.

- Inverse quantization for DCT coefficients other than Intra-DC coefficients

Inverse quantization is a process that takes natural ordered, quantized DCT coefficients in 8x8 matrix format and produces a reconstructed 8x8 DCT coefficient matrix. There are three steps involved in this process. The first step is to select the dequantization matrix and perform the arithmetic required by the video encoding algorithm (MPEG1 or MPEG2) and macroblock type. The result then goes through a saturation process to limit the reconstructed DCT coefficients to 12-bit signed values. Finally, a mismatch control step is performed in an effort to alleviate picture quality degradation due to different IDCT implementation in the encoder and decoder.

For all other DCT coefficients, including DC coefficients for non-intra blocks, the reconstructed DCT coefficients are recovered by the arithmetic involving the current *quantizer_scale_code* and the quantization matrix (weighting matrix). Default quantization matrix for intra blocks is shown in Table and default quantization matrix for non-intra blocks is shown in Table .

Table 6: Default Quantization Matrix for Intra Blocks

8	**16**	**19**	**22**	**26**	**27**	**29**	**34**
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Table 7: Default Quantization Matrix for Non-Intra Blocks

16	**16**	**16**	**16**	**16**	**16**	**16**	**16**
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

The *quantizer_scale_code* can be updated on the slice and macroblock levels. It is updated whenever a new *quantizer_scale_code* is decoded by the VLD. A new *quantization matrix* can only be loaded on the sequence or picture levels. The currently active quantization matrices are loaded into MMIO registers so the Inverse Quantization block can access them.

When a loadable quantizer matrix is downloaded, it has to go through an inverse zig-zag scan process.

Remark: The alternate scan does not apply to downloadable quantizer matrices.

The flags in bits 2 and 3 of Extra_Picture_Info indicate if the default or downloaded quantizer matrices are currently being used by the Inverse Quantization block.

When a downloadable quantization matrix is loaded into MMIO, the corresponding Default_Quantizer_Matrix flag in Extra_Picture_Info is also cleared. At the beginning of a picture, the two Default_Quantizer_Matrix bits are set after the default matrices are loaded.

34.4.4 Quantization Coefficients for Half-Resolution Scheme

The MPEG2 pipe supports the two-to-one compression scheme to reduce the main memory requirement for decoding by 50%. The half-resolution scheme is enabled by asserting the half-resolution mode bit in the MC_PICINFO1 register. If half-resolution mode is not enabled, the coefficients from the VLD should always be selected.

The half-resolution mode uses frequency domain filtering in order to improve the quality of the video image. The filtering scheme either selects a coefficient value generated from the VLD or selects a zero coefficient value. The coefficient selection scheme is programmable and operates at an 8-by-8 block level. The TM32 CPU core writes 64-bit values into the IQ_SEL_0 and IQ_SEL_1 MMIO registers. Each bit in the MMIO register is used to select either a coefficient value from VLD or a zero coefficient value within an 8-by-8 block. The bit value of ‘0’ selects the coefficient from the VLD and the bit value of ‘1’ selects the zero value. Refer to the IQ_SEL register description for more information.

34.4.5 RL/IQ Registers

Communications between the TM32 CPU core and the MPEG2 pipeline blocks are carried out through MMIO transactions.

Any undefined MMIO bit should be ignored when read and written as zero.

34.4.5.1 Run-Length Decoder Statistics Registers

The Run-Length Decoder has a status register in the MMIO space called RL_STATS. This is a read/writable register. It has two fields used only for performance analysis.

Total Symbol Count

This counts the number of symbols (non-zero DCT coefficients) in multiples of 1024 in the decoded bitstream. The TM32 CPU core is responsible for initializing, reading, and resetting this value.

Total_Coded_Block_Count

This counts the number of coded blocks in multiples of 1024 in the current bitstream decoder process. The TM32 CPU core is responsible for initializing, reading and resetting this value.

34.4.5.2 Inverse Quantization Control Register (IQ_CONTROL)

There are a number of parameters that are used by the Run-Length Decoder, Inverse Scan, Inverse Quantization, and Inverse DCT. Bits 2 and 3 of the IQ_CONTROL register indicate if the default or loaded quantizer matrices are being used by the Inverse Quantization block. The TM32 CPU core will set the appropriate value for the Default_intra_quant_matrix and the Default_non_intra_quant_matrix fields. At the beginning of a picture, the TM32 CPU core will set the value of '1' for the Default_intra_quant_matrix and the Default_non_intra_quant_matrix fields.

34.5 Motion Compensation

34.5.1 Introduction

The Motion Compensation (MC) unit is compatible with MPEG1 and MPEG2 MP@ML and MP@HL (ATSC compliant) video standards.

The MC block supports a 2:1 compression mode (also referred to as the half-resolution or half-res mode) in order to reduce main memory usage. The reconstructed frames are compressed by 50% before storage into main memory.

The MC block requires macroblock headers from the VLD block, YUV data from the IDCT block, and reference YUV data from main memory (SDRAM), in order to perform motion compensation. The output of motion compensation is stored directly into main memory. In half-resolution mode each 8x8 YUV IDCT block is compressed to 4x8 before motion compensation and storage into main memory. Picture and sequence level parameters required for motion compensation and SDRAM base addresses are programmed in the registers.

Reference and decoding pictures are stored on a per field basis. That is, top-field and bottom-field of each picture are stored in a separate buffer. The Y-component of each field is stored in a separate buffer, while the U- and V- components of each field are interleaved in one buffer in the order of UVUVUV.... Therefore, up to 12 field buffers are required to decode one motion-compensated picture. The width of all field buffers in number of bytes must be the same and is specified in the LINE_SIZE register. The LINE_SIZE can be larger than the actual picture width, if necessary.

Features

- Supports motion compensation operations in MPEG1 and MPEG2 MP@ML and MP@HL
- Supports full-resolution (normal) and half-resolution (2:1 compression) modes
- Error concealment under CPU control
- Error detection
- Optimal 'B' frame buffer switching at macroblock row level
- Allows pipeline flush under CPU control
- Optional short-cuts available to reduce memory bandwidth usage

34.5.2 MC Control Registers

The MC_PICINFO0 and MC_PICINFO1 registers contain parameters that are extracted or derived from the decoding MPEG video bitstream at the picture layer and above (such as mb_width and picture_type). (Note that MC_PICINFO1 is the same as the VLD_PI register in the VLD block.) MC_PICINFO2 register contains parameters for error concealment operation and picture_stride, which can be different from the coded picture width if necessary. The rest of the control registers store the SDRAM base addresses of the forward, backward, and destination YUV field buffers.

Remark: All SDRAM base addresses must be at least 16-byte aligned.

The allocation for all MPEG picture data (together) must be in chunks of 4 kB and with 4 kB aligned addresses.

34.5.3 MC Command Register

The MC Command register contains a 3-bit command field through which the CPU instructs the MC to perform certain operations. The definition of the command code and the corresponding operation is given in Table 8. The command field will be automatically cleared by the MC once the last command is completed.

Table 8: MC Command Codes

Name	Code	Description
cmd_done	000	If a command is issued previously, this indicates completion of the last requested operation.
flush_cmd	001	When the VLD is stopped (due to a start code for instance), the CPU may issue this command to flush the MPEG pipe. The MC continues to read and process mb_header from the mb_header_queue and will initiate an internal flush sequence when the queue becomes empty. MC will be stalled and the DONE_FLUSH bit (in the VLD_MC_STATUS register) set after the last macroblock in the pipe is stored into SDRAM. CPU will be interrupted also if the DONE_FLUSH_IE bit (in VLD_IE register) is 1. CPU must acknowledge the DONE_FLUSH condition by writing 1 into the DONE_FLUSH status bit to resume MC normal operation. MC will clear the MC_COMMAND register once the flush operation is completed.
errcon1_cmd	010	When the VLD is stopped due to a new slice start code, the CPU issues this command to perform error concealment. Before restarting the VLD to parse the new slice, the CPU sets this command code and writes the desired macroblock position (row and column) to start error concealment into err_mb_row and err_mb_col registers. Once the VLD restarts parsing (from the beginning of the new slice), the MC first determines the starting macroblock position of the new slice based on the first macroblock header of the slice parsed by the VLD and performs error concealment from (err_mb_row, err_mb_col) to just before the starting position of the new slice. At that point, the MC clears the MC_COMMAND register, sets the DONE_ERRCON bit (in the VLD_MC_STATUS register), and waits in idle for CPU acknowledgement of the condition. If DONE_ERRCON_IE (in VLD_IE register) is 1, the CPU will also be interrupted by the VLD when the DONE_ERRCON bit is set. The CPU acknowledges the DONE_ERRCON condition by writing to the DONE_ERRCON status bit. The MC then resumes normal operation and continues processing of the new slice. Note that error concealment is allowed to span multiple macroblock rows.
errcon2_cmd	100	This command is similar to errcon1_cmd, except that the CPU needs to specify both the starting macroblock position (in err_mb_row and err_mb_col) and the ending macroblock position (in err_end_mb_row and err_end_mb_col) for the desired error concealment operation. In this case, the MC performs error concealment from the given starting position to the macroblock just before the given ending macroblock position. If error concealment is up to and includes the last macroblock of the current picture, software should specify err_end_mb_col as 0 and err_end_mb_row as mb_height for frame pictures or as mb_height>>1 for field pictures.

34.5.4 MC Status Register

The MC Status register is a read-only register that is updated continuously while the MC is in operation. It is normally read by the CPU when the block is completely idle and the pipe is empty. The (state_mb_col, state_mb_row) status indicates the macroblock position of the last macroblock successfully reconstructed and stored into SDRAM. start_mb_col is the macroblock column position of the first macroblock of the last slice (this information allows the CPU to detect, for example, if there are any “holes” in the decoding slice).

34.5.5 MC_PFCOUNT

The MC_PFCOUNT MMIO register contains two counters: count_24x4 and count_24x5, which record the total number of 24x4 and 24x5 (16x4 and 16x5 in half-res mode) 2-D memory fetches generated by the PFU, respectively. The CPU can reset these counters by writing 0 into the register. These numbers reflect the memory bandwidth utilization for motion compensation and would allow the CPU to derive a contingency strategy to assure graceful degradation in picture quality when memory bandwidth requirement is at the limit.

34.5.6 LINE_SIZE

The LINE_SIZE register contains the size in bytes of the Y component for one video line stored in memory. For example, if the system is receiving and decoding a 1920 by 1080 pixel MPEG2 bitstream in full resolution mode, LINE_SIZE should contain 1920.

Certain limitations apply to the value that LINE_SIZE can contain:

LINE_SIZE cannot be 512, 680, 1024, 1360, 1368, 1536, 2040

Recommended values are 320, 352, 368, 640, 704, 720, 960, 1280, 1440, 1920.

If the desired LINE_SIZE value is not on this recommended list, use padding (and lose a bit of memory space) to achieve one of the recommended values listed above.

This register must be set before the motion compensation unit is enabled. The stored value has to be 8 bytes aligned (i.e., the three least significant bits are forced to 0s).

34.5.7 VLD_MC_STATUS

The MC updates the fields in the VLD_MC_STATUS register (For a complete description of this status register, refer to Section 34.3 on page 34-721).

When the MC detects a bitstream error (see Section 34.5.9), it writes the corresponding error code into the mc_error_flags field, which causes an interrupt to the CPU (if the MC_ERROR interrupt is enabled). The DONE_FLUSH bit is set when the MC completes a flush command. If the DONE_FLUSH_IE bit (in VLD_IE register) is set, a VLD interrupt is sent to CPU as well. The DONE_ERRCON bit is set when the MC completes an error concealment operation. CPU also receives a VLD interrupt if the DONE_ERRCON_IE bit (in VLD_IE register) is set.

34.5.8 Error Concealment Operation

The MC saves the starting macroblock column position in the start_mb_col field of the MP_MC_STATUS register to allow the CPU to perform further error checking.

The MC maintains an internal database of concealment motion vectors during normal operation. This is in preparation for error concealment.

In error concealment mode, the MC must reconstruct a number of macroblocks where the motion parameters to reconstruct a macroblock are derived from the internal concealment motion vector database.

Two error concealment commands are available:

- **errcon1_cmd** is used when error concealment is applied from a user-programmed starting macroblock position to the beginning of the next slice being parsed by VLD (where the starting column position of the new slice is unknown to the user at the moment)
- **errcon2_cmd** is used when error concealment is applied from a user-programmed starting macroblock position to a user-programmed ending macroblock position. As described below, both commands allow error concealment to span over multiple macroblock rows, and the MC_COMMAND register is cleared automatically when either command is done.

The errcon1_cmd bit is checked when the MC fetches a mb_header which is the beginning of a new slice. If errcon1_cmd is 1, the MC performs error concealment before processing the current mb_header. The error concealment starts at (err_mb_col, err_mb_row), as programmed by the CPU, until the macroblock just before the given new slice. On the other hand, the errcon2_cmd bit is checked by the MC while it is waiting for a new macroblock header from the VLD. If this bit is set, the MC performs error concealment from the programmed starting macroblock position to the ending macroblock position.

Upon completion of either command, the MC clears the MC_COMMAND register, sets the DONE_ERRCON bit (in VLD_MC_STATUS register), and waits in an idle loop for CPU acknowledgement of the condition by writing a 1 into the DONE_ERRCON bit. If the user sets the DONE_ERRCON_IE bit in the VLD_IE register, the VLD also interrupts the CPU when the DONE_ERRCON bit is set. The MC then resumes normal operation. Notice that no macroblocks are repaired if (err_mb_col + err_mb_row) >= (start_mb_col + slice_start_code – 1) for errcon1_cmd, or if (err_mb_col + err_mb_row) >= (err_end_mb_col + err_end_mb_row) for errcon2_cmd. For these cases, the MC declares completion of error concealment immediately.

Notice that the DONE_ERRCON bit is set only when all the error-concealed macroblocks are stored into main memory. That is, the MPEG pipe is flushed and idle upon completion of an error concealment operation.

When a bitstream error is detected, the user should send a command to clean out the MPEG pipe before attempting error concealment of the missing macroblocks. The command can be an MPEG-RESET or an MC FLUSH command, depending on the error situation at hand. After the MPEG pipe is flushed or reset, the user commands the VLD to scan the bitstream for the next undamaged slice start code. At that point, CPU can decide whether to perform an error concealment operation to “repair” the damaged macroblocks from the point where error was first detected up to the beginning of the new slice.

Two cases are possible:

1. The error macroblocks to be repaired and the new slice are all located in the current destination buffer.

   The following steps will repair the missing macroblocks: i) write the desired starting position of error concealment into err_mb_row and err_mb_col fields of the MC_PICINFO2 register, ii) set the errcon1_cmd bit in MC_COMMAND register, iii) send a command to the VLD to start parsing the new slice, iv) wait for the DONE_ERRCON flag in the VLD_MC_STATUS register to be set by the MC, and v) acknowledge DONE_ERRCON by writing 1 to this bit field.
2. Some of the error macroblocks and/or the new slice are not located in the current destination buffer.

   The following steps will conceal the missing macroblocks: i) write the desired starting
   position of error concealment into err_mb_row and err_mb_col fields of the MC_PICINFO2 register, and the beginning position of the macroblock row right below the last row of the current destination buffer into the err_end_mb_row and err_end_mb_col fields of the MC_PICINFO2 register, ii) set the errcon2_cmd bit, iii) wait for DONE_ERRCON flag to be set by the MC, iv) acknowledge DONE_ERRCON, v) program the new buffer address (and any new picture-information if the buffer corresponds to a new picture) into corresponding MMIO registers, vi) if there are more macroblocks to be repaired in the new buffer, write the position of the first macroblock of the new buffer into the err_mb_row and err_mb_col, and repeat steps (ii) through (v) in Case 1. Otherwise, command the VLD to start parsing the new slice.

34.5.9 Bitstream Error Detection

Before processing an mb_header from the VLD, the MC checks the contents of the header for certain parameter errors. Table 9 lists the codes of the error types that are checked.

Table 9: MC Error Types

Error Code	Type	Description
000000	No error	
000001	Invalid Motion Type	This error is detected when dual prime motion is specified for a B picture.
000010	Invalid Macroblock Address Increment	This error is detected when the given macroblock address increment in an mb_header is too large, such that it goes beyond the end of the current row in MPEG2, or beyond the end of the current picture in MPEG1
000100	Macroblock Overflow	This error is raised when a slice continues while the last coded macroblock is already at the end of the current row for MPEG2, or at the end of the current picture for MPEG1

Table 9: MC Error Types *…Continued*

Error Code	Type	Description
001000	Reserved	
010000	Pre-Fetch Coordinates Out-of-Bounds	This error is detected when the computed pre-fetch request parameters refer to a reference macroblock which goes beyond the boundary of the reference picture
100000	No Prediction for B Skipped Macroblocks	This error occurs if the last coded values of mv_forward and mv_backward are both 0 when skipped macroblocks are encountered in a B picture. That is, skipped macroblocks are immediately preceded by an intra macroblock in a B picture, which is not allowed

When one of these bitstream errors is detected, the MC writes the corresponding error code into the mc_error_flags field of the VLD_MC_STATUS register (refer to Section 34.3 on page 34-721). This eventually causes the MPEG pipe to go into an idle state and await reset.

34.5.10 Options for Reducing Memory Bandwidth

The following options are available to reduce memory bandwidth:

- Drop the vertical half-pel flags of Y reference in all non-intra macroblocks in a B picture, to reduce the number of 24x5 (16x5 in half-res mode) MMI requests. This is controlled by setting the no_y_yhalf bit in the VLD_PI register.
- Drop the vertical half-pel flags of UV reference in all non-intra macroblocks in a B picture, to reduce the number of 24x5 (16x5 in half-res mode) MMI requests. This is controlled by setting the no_uv_yhalf bit in the VLD_PI register.
- Drop backward prediction for all non-intra macroblocks requiring both forward and backward predictions. This is controlled by setting the no_backward bit in the VLD_PI register.

34.5.11 Fetching Macroblocks

For each macroblock the MC fetches 2-D reference data blocks from one of the eight reference field buffers (Y or UV, top-field or bottom-field, forward or backward).

For bandwidth efficiency in memory access, the MC makes special 2-D memory fetch requests to the MMI to read 24x4 or 24x5 (16x4 or 16x5 in half-res mode) bytes per request. It then selects the desired reference data from the fetched data and stores them into internal reference macroblock buffers. The extra data that were fetched are discarded. A list of all possible pre-fetch sizes is shown in Table 10.

Table 10: Possible Pre-Fetch Sizes

No. of Pre-Fetch Lines per Request	No. of MMI Requests	Fetch Sizes
Note: w = 24 in full-res mode, w = 16 in half-res mode		
4	1	wx4
5	1	wx5

Table 10: Possible Pre-Fetch Sizes *…Continued*

No. of Pre-Fetch Lines per Request	No. of MMI Requests	Fetch Sizes
8	2	wx4, wx4
9	2	wx4, wx5
16	4	wx4, wx4, wx4, wx4
17	4	wx4, wx4, wx4, wx5

To assist the CPU in determining memory bandwidth utilization due to the fetching of reference data, MC increments the count_24x4 and count_24x5 MMIO registers for every wx4 and wx5 MMI request respectively (where w = 16 or 24). These two (16-bit unsigned) counters can be reset by the CPU by writing 0 directly into the MC_PFCOUNT register.

34.5.12 Storing Macroblocks

The MC stores the reconstructed macroblocks into main memory. The MC computes the physical main memory addresses to store the macroblocks based on the base addresses of the destination buffer and LINE_SIZE parameters programmed into the MMIO registers.

After the macroblocks are successfully stored into SDRAM, MC writes the macroblock position of the last macroblock stored into the state_mb_col and state_mb_row fields of the MC Status register.

For bandwidth efficiency in SDRAM access, the MC makes special 2-D storage requests to the MMI to write 32x4 bytes into main memory per request. To cover the cases in which the output buffer is not completely filled, however, special byte-enable masks are devised to write only the first 8x4, 16x4, or 24x4 bytes of the 32x4 bytes sent to main memory.

Since every MPEG2 picture slice starts and ends in the same row and each field is decoded sequentially starting from the first row, it is possible to divide the entire top or bottom field into multiple destination buffers. For instance, a complete picture field may be divided into four uniform quarters, such that each quarter is stored in a different SDRAM buffer. Software can take advantage of this multi-buffer approach to reduce the total memory buffer size to hold the decoding picture. Hence for each new destination buffer, the CPU programs the starting macroblock row to which the first row of the destination buffer corresponds into the mb_row_offset register. The MC computes the main memory address of the macroblocks in the output buffer taking into account also the mb_offset parameter.

It is clear that mb_row must not be less than mb_row_offset. The CPU should check the validity of mb_row (through slice_start_code) for every new slice to detect any potential error in the decoding bitstream.

Splitting a destination picture field into multiple buffers should be done only when decoding a B picture. Because I and P pictures are used as reference, it is mandatory that the entire I or P picture field be stored in a single contiguous memory buffer and mb_row_offset must be 0 in these cases. Splitting a destination picture field into multiple buffers should never be done for any picture type when decoding an MPEG1 bitstream.

34.5.13 MC/VLD Error Recovery

Both the VLD and MC may detect certain errors in the decoding bitstream as they operate. If the MC detects a bitstream error, it writes the corresponding error code into the mc_error_flags field of the VLD_MC_STATUS register. When either block detects an error, all the MPEG units will idle synchronously. When all the MPEG units become idle, the VLD will interrupt the CPU (if the corresponding interrupt is enabled). The CPU acknowledges the error interrupt by writing 1 into any of the error bit fields, which clears all the error bits. To complete the error recovery process, the CPU must now send a reset command to the MPEG pipe to resume normal operation.

34.5.14 MPEG Software Reset

During a software reset of the MPEG pipe, all the MPEG units will first enter an idle state as quickly as possible. As soon as all the MPEG units become idle, all of their internal states are reset synchronously. For the MC block, the MC_STATUS, MC_COMMAND, and MC_PFCOUNT MMIO registers are also cleared after reset (while other MC MMIO registers are unchanged). Then all the MPEG units resume normal operation after software removes the RESET command from the VLD_COMMAND register.

Remark: The first mb_header sent to the MC after a software reset must be the first of a slice. The MC will not check for this condition. It is the responsibility of the software to make sure that this is the case when restarting the MPEG unit after a soft reset.

34.5.15 MC Flush

There are two situations under which the CPU will need to make sure the MPEG pipe is empty with all pending macroblocks reconstructed and saved into main memory:

- When the VLD detects the start code of a new slice which starts in a new destination frame buffer.
- When the VLD is stalled right before the start of an error packet, as controlled by the CPU. Under either situation, the CPU issues a flush command to the VLD/MC by setting the flush_cmd bit in the MC Command register.

Upon completion of a flush command, MC clears the MC_COMMAND register and sets the DONE_FLUSH bit in the VLD_MC_STATUS register. VLD will also interrupt the CPU if the DONE_FLUSH_IE bit (in VLD_IE register) is 1. From then on, the entire MPEG pipe is empty and idle. The CPU determines the completion of the flush operation by reading the DONE_FLUSH status bit. The CPU acknowledges the flush-completion condition by writing 1 into the DONE_FLUSH status bit, which clears the bit and resumes normal MC operation. The CPU may also send a software reset command to reset the MPEG pipe. Software reset is necessary to ensure proper MPEG operation after flushing if the flush command is sent to handle a packet error.

34.5.16 MC Timeout

Whenever the state_mb_col or state_mb_row fields of the MC_STATUS register are updated by the MC, or when the reset_mc_timeout signal transitions from high to low, the internal 15-bit mc_timeout_counter in the MC block is reset to N = mc_timeout_period*2048, where mc_timeout_period is a 4-bit value between 0 and 15, programmed into the MC_PICINFO0 register by the CPU.

If the reset_mc_timeout signal is low, the mc_timeout_counter is decremented for every CPU clock cycle; otherwise, countdown is stalled. When the counter reaches 0, the internal mc_timeout one-shot signal between the MC and VLD is triggered to indicate to the VLD that the MC has not been storing any output macroblocks for the last N clocks. The mc_timeout_counter stays at 0 until state_mb_col or state_mb_row is updated, or when the reset_mc_timeout signal transitions from high to low, or after software reset. The timeout mechanism is disabled if mc_timeout_period equals 0. The maximum timeout period is 30720 cycles.

If enabled by the CPU, the VLD will interrupt the CPU in response to the mc_timeout signal if it is in the parsing mode.

Remark: Timeout will not occur if the MC is carrying out an ERRCON2 command (type 2 error
concealment).

34.6 Register Summary and Descriptions

The base address for the PNX8526 MPEG Video Decoder module is 0x10 5000.

Table 11: MPEG Video Decoder Module Register Summary

Offset	Name	Description
0x10 5000	VLD_COMMAND	VLD Command
0x10 5004	VLD_SR	Shift Register (shadow)
0x10 5008	VLD_QS	Quantization Scale code to be output by the VLD
0x10 500C	VLD_PI	VLD Picture Information
0x10 5010	VLD_MC_STATUS	VLD MC Status register
0x10 5014	VLD_IE	VLD Interrupt Enable
0x10 5018	VLD_CTL	VLD Control
0x10 501C	VLD_INP_ADR	VLD Input Memory Address
0x10 5020	VLD_INP_CNT	VLD Input Count shows the number of bytes read from main memory
0x10 5024—5030	Unused	
0x10 5034	VLD_BIT_CNT	BLD Bit Count shows the number of bits consumed by the VLD.
0x10 5038	LINE_SIZE	Contains the size in bytes of the Y component for one video line stored in memory

Table 11: MPEG Video Decoder Module Register Summary *…Continued*

Offset	Name	Description
0x10 5040	W_TBL0_W0	Quantization matrix table 0
0x10 5044	W_TBL0_W1	
0x10 5048	W_TBL0_W2	
0x10 504C	W_TBL0_W3	
0x10 5050	W_TBL0_W4	
0x10 5054	W_TBL0_W5	
0x10 5058	W_TBL0_W6	
0x10 505C	W_TBL0_W7	
0x10 5060	W_TBL0_W8	
0x10 5064	W_TBL0_W9	
0x10 5068	W_TBL0_W10	
0x10 506C	W_TBL0_W11	
0x10 5070	W_TBL0_W12	
0x10 5074	W_TBL0_W13	
0x10 5078	W_TBL0_W14	
0x10 507C	W_TBL0_W15	
0x10 5080	W_TBL1_W0	Quantization matrix table 1
0x10 5084	W_TBL1_W1	
0x10 5088	W_TBL1_W2	
0x10 508C	W_TBL1_W3	
0x10 5090	W_TBL1_W4	
0x10 5094	W_TBL1_W5	
0x10 5098	W_TBL1_W6	
0x10 509C	W_TBL1_W7	
0x10 50A0	W_TBL1_W8	
0x10 50A4	W_TBL1_W9	
0x10 50A8	W_TBL1_W10	
0x10 50AC	W_TBL1_W11	
0x10 50B0	W_TBL1_W12	
0x10 50B4	W_TBL1_W13	
0x10 50B8	W_TBL1_W14	
0x10 50BC	W_TBL1_W15	
0x10 50C0	IQ_CONTROL	Inverse Quantization Control
0x10 50C4	RL_STATS	Run Length Statistics
0x10 50C8	MP_IQ_SEL_0	Inverse Quantization Select
0x10 50CC	MP_IQ_SEL_1	Inverse Quantization Select
0x10 5200	MC_PICINFO0	MC Picture Information 0
0x10 5208	MC_PICINFO2	MC Picture Information 1

Table 11: MPEG Video Decoder Module Register Summary *…Continued*

Offset	Name	Description
0x10 520C	MC_FREFY0	MC Forward Reference base address for the Y component of top field
0x10 5210	MC_FREFY1	MC Forward Reference base address for the Y component of bottom field
0x10 5214	MC_FREFUV0	MC Forward Reference base address for the UV component of top field
0x10 5218	MC_FREFUV1	MC Forward Reference base address for the UV component of bottom field
0x10 521C	MC_BREFY0	MC Backward Reference base address for the Y component of top field
0x10 5220	MC_BREFY1	MC Backward Reference base address for the Y component of bottom field
0x10 5224	MC_BREFUV0	MC Backward Reference base address for the UV component of top field
0x10 5228	MC_BREFUV1	MC Backward Reference base address for the UV component of bottom field
0x10 522C	MC_DESTY0	MC Destination base address for the Y component of top field
0x10 5230	MC_DESTY1	MC Destination base address for the Y component of bottom field
0x10 5234	MC_DESTUV0	MC Destination base address for the UV component of top field
0x10 5238	MC_DESTUV1	MC Destination base address for the UV component of bottom field
0x10 523C	MC_COMMAND	MC Command
0x10 5240	MC_PFCOUNT	Records the total number of 2-D memory fetches generated
0x10 5244	MC_STATUS	MC Status
0x10 5FFA	PD	Powerdown Register
0x10 5FFC	MODULE ID	Module Identification Register

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 5000			***VLD_COMMAND***	
31:12		-	Unused	
11:8	R/W	0	Command	Command code of the VLD command to be executed. Refer to Section 1 on page 34-724 for more information. 0x1 = Shift the bitstream by ‘”count’” bits. 0x2 = Parse for a given number of macroblocks. 0x3 = Search for the next start code. 0x4 = Reset the MPEG-PIPE. 0x5 = Initialize the VLD. 0x6 = Search for the given start code. 0x7 = Parse one row of macroblocks.
7:0	R/W	0	Mblock/Shift Count or Start Code	For the ‘Shift Bitstream’ command, only the lower 4 bits are used; the upper 4 bits should be set to 0. All 8 bits are used for the ‘Parse macroblocks’ and ‘Search for given start code’ commands.
Offset 0x10 5004			***VLD_SR***	
31:15			Reserved	
15:0	R	NI	Shift Register	This read-only register is a shadow of the VLD’s operational shift register. It allows the TM32 CPU core to access the bitstream through the VLD. Bits 0 through 15 are the current bits of the VLD shift register.
Offset 0x10 5008			***VLD_QS***	
31:5		-	Unused	
4:0	R/W	NI	Quant scale	This 5-bit read/write register contains the quantization scale code to be output by the VLD until it is overridden by a macroblock quantizer scale code. The quantizer scale code is part of the macroblock header output.
Offset 0x10 500C			***VLD_PI***	
31:28	R/W	NI	Vertical back.rsize	No. of bits per backward vertical motion vector residual in the picture
27:24	R/W	NI	Horizontal back.rsize	No. of bits per backward horizontal motion vector residual in No. picture
23:20	R/W	NI	Vertical for.rsize	No. of bits per forward vertical motion vector residual in picture
19:16	R/W	NI	Horizontal for.rsize	No. of bits per forward horizontal motion vector residual in picture
15		-	Unused	
14	R/W	NI	no_backward	Controls use of backward prediction in macroblocks with bi-directional motion prediction. 0 = Keep backward prediction. 1 = Drop backward prediction.
13	R/W	NI	mpeg2mode	0 = The current sequence is MPEG1. 1 = The current sequence is MPEG2. Note: For error checking only

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
12	R/W	NI	no_uv_yhalf	Controls use of vertical half-pel flag for UV component in all macroblocks of a B picture. 0 = Keep UV vertical half-pel flag. 1 = Drop UV vertical half-pel flag.
11	R/W	NI	no_y_yhalf	Controls use of vertical half-pel flag for Y component in all macroblocks of a B picture. 0 = Keep Y vertical half-pel flag 1 = Drop Y vertical half-pel flag
10	R/W	NI	top_field_first	0 = Capture bottom field first. 1= Capture top field first.
9	R/W	NI	full_pel_backward	0 = Backward motion vectors in a picture have half-pel units. 1 = Backward motion vectors in a picture have full-pel units.
8	R/W	NI	full_pel_forward	0 = Forward motion vectors in a picture have half-pel units. 1 = Forward motion vectors in a picture have full-pel units.
7	R/W	NI	half_res_mode	0 = full_res_mode 1 = half_res_mode
6	R/W	NI	mv_concealment	Indicates whether motion vectors are coded in all intra macroblock headers of a picture. 0 = Forward motion vectors are not coded. 1 = Forward motion vectors are coded.
5	R/W	NI	intra_vlc	0 = Use DCT table zero. 1 = Use DCT table one.
4	R/W	NI	frame_prediction_frame_dct	0 = motion_type and dct_type follow the decoded values in the mb_header from the VLD. 1 = motion_type = FRAME, and dct_type = 0. CPU should set it to 0 for Field Pictures and 1 for MPEG1.
3:2	R/W	NI	picture_structure	0x1 = Top-field 0x2 = Bottom-field 0x3 = Frame picture 0x0 = Reserved
1:0	R/W	NI	picture_type	1=I 2=P 3=B 0=D (MPEG1 only)
Offset 0x10 5010			***VLD_MC_STATUS***	
31:24		-	Unused	
23	R	0	DONE_FLUSH	1 signals the completion of the last flush command. CPU writes 1 to this field to acknowledge condition and clear this flag.
22	R/W	0	EOR_ERRCON	1 signals the completion of the last error concealment command. CPU writes 1 to this field to acknowledge condition and clear the flag

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
21:16	R	0	mc_error_flags	MC error code. 000000 = No error 000001 = Invalid motion type 000010 = Invalid macroblock address increment (>1) 000100 = Macroblock overflow 001000 = Reserved 010000 = Pre-fetch coordinates out of bound 100000 = No prediction for B skipped macroblocks
15:8		-	Unused	
7	RW	0	Time-out	1 = MPEG-PIPE timed out. Refer to Section 34.3.5.3 for details on the timeout mechanism. Bit is cleared by writing a logic ‘1.’
6	RW	0	RL overflow	1 = Overflow of run/level values within a block. Refer to Section 34.3.5 for details on the error handling procedure. This bit is cleared by writing a logic ‘1’ to it.
5:4		-	Unused	
3	RW	0	DMA input done	Conditions for setting this bit depend on the value of the DMA_Input_Done field in the VLD_CTL register. Refer to Section 34.3 for details. This bit is cleared by writing a logic ‘1’ to it.
2	RW	0	Bitstream error	1 = VLD encountered an illegal Huffman code or an unexpected start code. Refer to Section 34.3.5 for details on the error handling procedure. This bit is cleared by writing a logic ‘1’ to it.
1	RW	0	Start code detected	1 = VLD encountered 0x000001 while executing current command. This bit is cleared by writing a logic ‘1’ to it.
0	R		VLD Command done	1 = Successful completion of current command. This bit is cleared by issuing a new command.
Offset 0x10 5014			***VLD_IE***	
31:24		-	Unused	
23	R/W	0	DONE_FLUSH_IE	This bit enables the matching bit from the status register 0x010 to issue an IR to the CPU.
22	R/W	0	EOR_ERRCON_IE	This bit enables the matching bit from the status register 0x010 to issue an IR to the CPU.
21:16	R/W	0	MC Int. Enables	Each bit enables the matching bit from the status register 0x010 to issue an IR to the CPU.
15:8		-	Unused	
7:0	R/W	0	VLD Int. Enables	Each bit enables the matching bit from the status register 0x010 to issue an IR to the CPU.
Offset 0x10 5018			***VLD_CTL***	
31:17		-	Unused	
16	W	0	slice_start_code_strobe	When CPU writes 1 to this field, the VLD copies the value of slice_start_code to its internal register. CPU should do this only when the VLD is stopped. This bit is always read as 0.

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15:8	R/W	0	slice start code	Slice start code when the VLD is restarted; the slice_start_code_ strobe bit field must be set to ‘1’ in order to update this field.
7:3		-	Unused	
2	R/W	0	DMA-input-done-mode	0 = VLD sets the DMA_INPUT_DONE flag (in VLD_MC_STATUS register) when the DMA_INP_CNT transitions from non-zero to zero. 1 = The same flag is set only with the additional condition that both highway input buffers are empty. The slice_start_code_strobe bit field must be set to ‘0’ in order to update this field.
1:0		-	Unused	
Offset 0x10 501C			***VLD_INP_ADR***	
31:0	R/W	NI	VLD Input Memory Address	Memory address from which the VLD is reading (updated when DMA read transfer is completed.)
Offset 0x10 5020			***VLD_INP_CNT***	
31:15		-	Unused	
14:0	R/W	0	VLD Input Count	Number of bytes to be read from main memory
Offset 0x10 5024—5030			***UNUSED***	
Offset 0x10 5034			***VLD_BIT_CNT***	
31:18		-	Unused	
17:0	R	0	Actual Bit Count	Number of bits consumed by the VLD. Counts upward when bits are shifted out and consumed by the VLD. This counter wraps around after reaching the maximum value. Value can be initialized to zero by issuing the ‘Initialize VLD’ command.
Offset 0x10 5038			***LINE_SIZE***	
31:13		-	Unused	
12:3	R/W	0	Line size	The line size must be a multiple of 8 bytes. Recommended values are: 320, 352, 368, 640, 704, 720, 960, 1280, 1440, 1920. LINE_SIZE cannot be 512, 680, 1024, 1360, 1368, 1536, 2040
2:0	R/W	0	Line size	Always 0.
Offset 0x10 5040			***W_TBL0_W0***	
31:24	R/W	NI	w[0][0][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][0][2]	
15:8	R/W	NI	w[0][0][1]	
7:0	R/W	NI	w[0][0][0]	
Offset 0x10 5044			***W_TBL0_W1***	
31:24	R/W	NI	w[0][0][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][0][6]	
15:8	R/W	NI	w[0][0][5]	
7:0	R/W	NI	w[0][0][4]	

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 5048			***W_TBL0_W2***	
31:24	R/W	NI	w[0][1][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][1][2]	
15:8	R/W	NI	w[0][1][1]	
7:0	R/W	NI	w[0][1][0]	
Offset 0x10 504C			***W_TBL0_W3***	
31:24	R/W	NI	w[0][1][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][1][6]	
15:8	R/W	NI	w[0][1][5]	
7:0	R/W	NI	w[0][1][4]	
Offset 0x10 5050			***W_TBL0_W4***	
31:24	R/W	NI	w[0][2][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][2][2]	
15:8	R/W	NI	w[0][2][1]	
7:0	R/W	NI	w[0][2][0]	
Offset 0x10 5054			***W_TBL0_W5***	
31:24	R/W	NI	w[0][2][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][2][6]	
15:8	R/W	NI	w[0][2][5]	
7:0	R/W	NI	w[0][2][4]	
Offset 0x10 5058			***W_TBL0_W6***	
31:24	R/W	NI	w[0][3][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][3][2]	
15:8	R/W	NI	w[0][3][1]	
7:0	R/W	NI	w[0][3][0]	
Offset 0x10 505C			***W_TBL0_W7***	
31:24	R/W	NI	w[0][3][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][3][6]	
15:8	R/W	NI	w[0][3][5]	
7:0	R/W	NI	w[0][3][4]	
Offset 0x10 5060			***W_TBL0_W8***	
31:24	R/W	NI	w[0][4][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][4][2]	
15:8	R/W	NI	w[0][4][1]	
7:0	R/W	NI	w[0][4][0]	

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 5064			***W_TBL0_W9***	
31:24	R/W	NI	w[0][4][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][4][6]	
15:8	R/W	NI	w[0][4][5]	
7:0	R/W	NI	w[0][4][4]	
Offset 0x10 5068			***W_TBL0_W10***	
31:24	R/W	NI	w[0][5][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][5][2]	
15:8	R/W	NI	w[0][5][1]	
7:0	R/W	NI	w[0][5][0]	
Offset 0x10 506C			***W_TBL0_W11***	
31:24	R/W	NI	w[0][5][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][5][6]	
15:8	R/W	NI	w[0][5][5]	
7:0	R/W	NI	w[0][5][4]	
Offset 0x10 5070			***W_TBL0_W12***	
31:24	R/W	NI	w[0][6][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][6][2]	
15:8	R/W	NI	w[0][6][1]	
7:0	R/W	NI	w[0][6][0]	
Offset 0x10 5074			***W_TBL0_W13***	
31:24	R/W	NI	w[0][6][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][6][6]	
15:8	R/W	NI	w[0][6][5]	
7:0	R/W	NI	w[0][6][4]	
Offset 0x10 5078			***W_TBL0_W14***	
31:24	R/W	NI	w[0][7][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][7][2]	
15:8	R/W	NI	w[0][7][1]	
7:0	R/W	NI	w[0][7][0]	
Offset 0x10 507C			***W_TBL0_W15***	
31:24	R/W	NI	w[0][7][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 040 to 07C show either the default_intra_matrix or the downloaded_intra_matrix.
23:16	R/W	NI	w[0][7][6]	
15:8	R/W	NI	w[0][7][5]	
7:0	R/W	NI	w[0][7][4]	

MPEG VIDEO DECODER REGISTERS

Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 5080			*W_TBL1_W0*	
31:24	R/W	NI	w[1][0][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][0][2]	
15:8	R/W	NI	w[1][0][1]	
7:0	R/W	NI	w[1][0][0]	
Offset 0x10 5084			*W_TBL1_W1*	
31:24	R/W	NI	w[1][0][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][0][6]	
15:8	R/W	NI	w[1][0][5]	
7:0	R/W	NI	w[1][0][4]	
Offset 0x10 5088			*W_TBL1_W2*	
31:24	R/W	NI	w[1][1][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][1][2]	
15:8	R/W	NI	w[1][1][1]	
7:0	R/W	NI	w[1][1][0]	
Offset 0x10 508C			*W_TBL1_W3*	
31:24	R/W	NI	w[1][1][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][1][6]	
15:8	R/W	NI	w[1][1][5]	
7:0	R/W	NI	w[1][1][4]	
Offset 0x10 5090			*W_TBL1_W4*	
31:24	R/W	NI	w[1][2][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][2][2]	
15:8	R/W	NI	w[1][2][1]	
7:0	R/W	NI	w[1][2][0]	
Offset 0x10 5094			*W_TBL1_W5*	
31:24	R/W	NI	w[1][2][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][2][6]	
15:8	R/W	NI	w[1][2][5]	
7:0	R/W	NI	w[1][2][4]	
Offset 0x10 5098			*W_TBL1_W6*	
31:24	R/W	NI	w[1][3][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][3][2]	
15:8	R/W	NI	w[1][3][1]	
7:0	R/W	NI	w[1][3][0]	
Offset 0x10 509C			*W_TBL1_W7*	

MPEG VIDEO DECODER REGISTERS				
Bits	**Read/ Write**	**Reset Value**	**Name (Field or Function)**	**Description**
31:24	R/W	NI	w[1][3][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][3][6]	
15:8	R/W	NI	w[1][3][5]	
7:0	R/W	NI	w[1][3][4]	
Offset 0x10 50A0			*W_TBL1_W8*	
31:24	R/W	NI	w[1][4][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][4][2]	
15:8	R/W	NI	w[1][4][1]	
7:0	R/W	NI	w[1][4][0]	
Offset 0x10 50A4			*W_TBL1_W9*	
31:24	R/W	NI	w[1][4][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][4][6]	
15:8	R/W	NI	w[1][4][5]	
7:0	R/W	NI	w[1][4][4]	
Offset 0x10 50A8			*W_TBL1_W10*	
31:24	R/W	NI	w[1][5][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][5][2]	
15:8	R/W	NI	w[1][5][1]	
7:0	R/W	NI	w[1][5][0]	
Offset 0x10 50AC			*W_TBL1_W11*	
31:24	R/W	NI	w[1][5][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][5][6]	
15:8	R/W	NI	w[1][5][5]	
7:0	R/W	NI	w[1][5][4]	
Offset 0x10 50B0			*W_TBL1_W12*	
31:24	R/W	NI	w[1][6][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][6][2]	
15:8	R/W	NI	w[1][6][1]	
7:0	R/W	NI	w[1][6][0]	
Offset 0x10 50B4			*W_TBL1_W13*	
31:24	R/W	NI	w[1][6][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][6][6]	
15:8	R/W	NI	w[1][6][5]	
7:0	R/W	NI	w[1][6][4]	
Offset 0x10 50B8			*W_TBL1_W14*	

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:24	R/W	NI	w[1][7][3]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][7][2]	
15:8	R/W	NI	w[1][7][1]	
7:0	R/W	NI	w[1][7][0]	
Offset 0x10 50BC			***W_TBL1_W15***	
31:24	R/W	NI	w[1][7][7]	The two quantizer matrices that are being used are mapped into MMIO space. Each value in the 8x8 quantizer matrix occupies eight bits. These two matrices take 128 bytes total. Offsets 080 to 0BC show either the default_non_intra_matrix or downloaded_non_intra_matrix.
23:16	R/W	NI	w[1][7][6]	
15:8	R/W	NI	w[1][7][5]	
7:0	R/W	NI	w[1][7][4]	
Offset 0x10 50C0			***IQ_CONTROL***	
31:6		-	Unused	
5:4	R/W	00	Intra_DC_Precision	Used to compute intra_dc_mult and reset values of intra_dc_pred.
3	R/W	0	Default_Non_Intra_Q_Matrix	If set, indicates the non-intra default quantizer matrix is being used.
2	R/W	1	Default_Intra_Q_Matrix	If set, indicates the default intra quantizer matrix is being used
1	R/W	0	Alt_Scan	0 = The Zig-Zag scan table is used for the current block. 1 = The alternate scan table is used.
0	R/W	0	Quant_Scale_Type	This field is used to select the quantizer scale table.
Offset 0x10 50C4			***RL_STATS***	
31:20	R/W	0	Total_Coded_Block_Ct	Total number of coded blocks in multiples of 1024. Can be initialized by writing the required values through MMIO write transaction.
19:0	R/W	0	Total_Symbol_Cnt	Total number of non-zero DCT coefficients in multiples of 1024. This field can be initialized by writing the required values through MMIO write transaction.
Offset 0x10 50C8			***MP_IQ_SEL_0***	
31:0	R/W	0	MP_IQ_SEL_0	The MP_IQ_SEL_0 and MP_IQ_SEL_1 is initialized to zero after hardware or software reset. The coefficient selection scheme is programmable and operates at an 8-by-8 block level. The TM32 CPU core will write 64-bit values into the MP_IQ_SEL_0 and MP_IQ_SEL_1 MMIO registers. Each bit in the MMIO register is used to select either a coefficient value from VLD or a zero coefficient value within an 8-by-8 block. The bit value of ‘0’ selects the coefficient from the VLD and the bit value of ‘1’ selects the zero value.
Offset 0x10 50CC			***MP_IQ_SEL_1***	

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
31:0	R/W	0	MP_IQ_SEL_1	The MP_IQ_SEL_0 and MP_IQ_SEL_1 is initialized to zero after hardware or software reset. The coefficient selection scheme is programmable and operates at an 8-by-8 block level. The TM32 CPU core will write 64-bit values into the MP_IQ_SEL_0 and MP_IQ_SEL_1 MMIO registers. Each bit in the MMIO register is used to select either a coefficient value from VLD or a zero coefficient value within an 8-by-8 block. The bit value of ‘0’ selects the coefficient from the VLD and the bit value of ‘1’ selects the zero value.
Offset 0x10 5200			*MC_PICINFO0*	
31		-	Unused	
30:24	R/W	NI	mb_row_offset	Starting macroblock row in the decoding picture which is the first macroblock row the current destination field buffers correspond to. For I and P pictures, this number should always be 0 because reference pictures are not allowed to span multiple non-continuous SDRAM buffers. For B pictures, the decoding picture is allowed to span multiple non-continuous SDRAM buffers. This offset allows the Storage Unit to compute the correct storage coordinates in the current destination buffers.
23:20		-	Unused	
19:16	R/W	NI	mc_timeout_period	If non-zero VLD interrupts the CPU if no macroblocks are stored in SDRAM by the MC within mc_timeout_period*2048 CPU cycles while the VLD is in parsing mode. 0 has no effect.
15:8	R/W	NI	mb_height	Number of luminance (Y) macroblocks per column in a frame picture, which is a fixed number for a given video sequence. The MC relies on this number to detect out-of-bounds reference errors.
7:0	R/W	NI	mb_width	Number of macroblocks per row in a picture in the current sequence
Offset 0x10 5208			*MC_PICINFO2*	
31:24	R/W	NI	err_end_mb_row	Ending macroblock row in error concealment (for each errcon2_cmd)
23:16	R/W	NI	err_end_mb_col	Ending macroblock column in error concealment (for each errcon2_cmd)
15:8	R/W	NI	err_mb_row	Macroblock row-to-start error concealment (for each errcon1_cmd or errcon2_cmd)
7:0	R/W	NI	err_mb_col	Macroblock column-to-start error concealment (for each errcon1_cmd or errcon2_cmd)
Offset 0x10 520C			*MC_FREFY0*	
31:0	R/W	NI	fb_fref_Y_f0	SDRAM base address of forward reference top-field buffer for the Y component
Offset 0x10 5210			*MC_FREFY1*	
31:0	R/W	NI	fb_fref_Y_f1	SDRAM base address of forward reference bottom-field buffer for the Y component
Offset 0x10 5214			*MC_FREFUV0*	
31:0	R/W	NI	fb_fref_UV_f0	SDRAM base address of forward reference top-field buffer for the UV component
Offset 0x10 5218			*MC_FREFUV1*	
31:0	R/W	NI	fb_fref_UV_f1	SDRAM base address of forward reference bottom-field buffer for the UV component

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 521C			*MC_BREFY0*	
31:0	R/W	NI	fb_bref_Y_f0	SDRAM base address of backward reference top-field buffer for the Y component
Offset 0x10 5220			*MC_BREFY1*	
31:0	R/W	NI	fb_bref_Y_f1	SDRAM base address of backward reference bottom-field buffer for the Y component
Offset 0x10 5224			*MC_BREFUV0*	
31:0	R/W	NI	fb_bref_UV_f0	SDRAM base address of backward reference top-field buffer for the UV component
Offset 0x10 5228			*MC_BREFUV1*	
31:0	R/W	NI	fb_bref_UV_f1	SDRAM base address of backward reference bottom-field buffer for the UV component
Offset 0x10 522C			*MC_DESTY0*	
31:0	R/W	NI	fb_dest_Y_f0	SDRAM base address of destination top-field buffer for the Y component
Offset 0x10 5230			*MC_DESTY1*	
31:0	R/W	NI	fb_dest_Y_f1	SDRAM base address of destination bottom-field buffer for the Y component
Offset 0x10 5234			*MC_DESTUV0*	
31:0	R/W	NI	fb_dest_UV_f0	SDRAM base address of destination top-field buffer for the UV component
Offset 0x10 5238			*MC_DESTUV1*	
31:0	R/W	NI	fb_dest_UV_f1	SDRAM base address of destination bottom-field buffer for the UV component
Offset 0x10 523C			*MC_COMMAND*	
31:3		-	Unused	
2:0	R/W	0	mc_cmd	The CPU instructs the MC to perform certain special operations with this command. Refer to Section 8 on page 34-734 for a description. 000 = cmd_done 001 = flush_cmd 010 = errcon1_cmd 100 = errcon2_cmd
Offset 0x10 5240			*MC_PFCOUNT*	
31:16	R/W	0000	count_24x5	Records the total number of 24x5 2-D memory fetches generated.
15:0	R/W	0000	count_24x4	Records the total number of 24x4 2-D memory fetches generated.
Offset 0x10 5244			*MC_STATUS*	
31:24		-	Unused	
23:16	R	NI	start_mb_col	Macroblock column position in the decoding picture where the current slice starts
15:8	R	0	state_mb_row	Macroblock row position in the decoding picture where the last reconstructed macroblock is stored in SDRAM

MPEG VIDEO DECODER REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7:0	R	0	state_mb_col	Macroblock column position in the decoding picture where the last reconstructed macroblock is stored in SDRAM
Offset 0x10 5FF4			*POWERDOWN*	
31:0	R/W	0	PD	MPEG Powerdown indicator 1 = Powerdown 0 = Power up When this bit equals 1, no other registers are accessible.
Offset 0x10 5FFC			*MODULE_ID*	
31:16	R	0x0100	Module ID	MPEG Video Decoder Module ID register
15:12	R	0	MajRev	Major Revision
11:8	R	0	MinRev	Minor Revision
7:0		0	Aperture Size [7:0]	Returns 00 = MPEG Video Aperture = 4kB.

Chapter 35: PI-Bus Architecture

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

35.1 Introduction

The PNX8526 PI-Bus architecture is designed to support the following:

- MIPS Icache, Dcache traffic is separated from DMA devices and TriMedia Programmable IO (PIO) (i.e., non-DMA traffic). The MIPS core has a high performance path, low latency path to memory.
- TriMedia Icache, Dcache traffic is separated from DMA devices and the MIPS PIO. The TriMedia (TM) core has a high performance, low latency path to memory.
- The MIPS core has a low latency access to peripherals that typically will be accessed by MIPS.
- The TM core has a low latency access to peripherals that typically will be accessed by TriMedia.
- All peripheral registers are accessible from MIPS, TriMedia, PCI, BOOT, and EJTAG.
- The MIPS PIO is separated from the TriMedia PIO to provide low latency interrupt handling.

35.2 Functional Description

The PI-Bus architecture block diagram is shown in Figure 1.

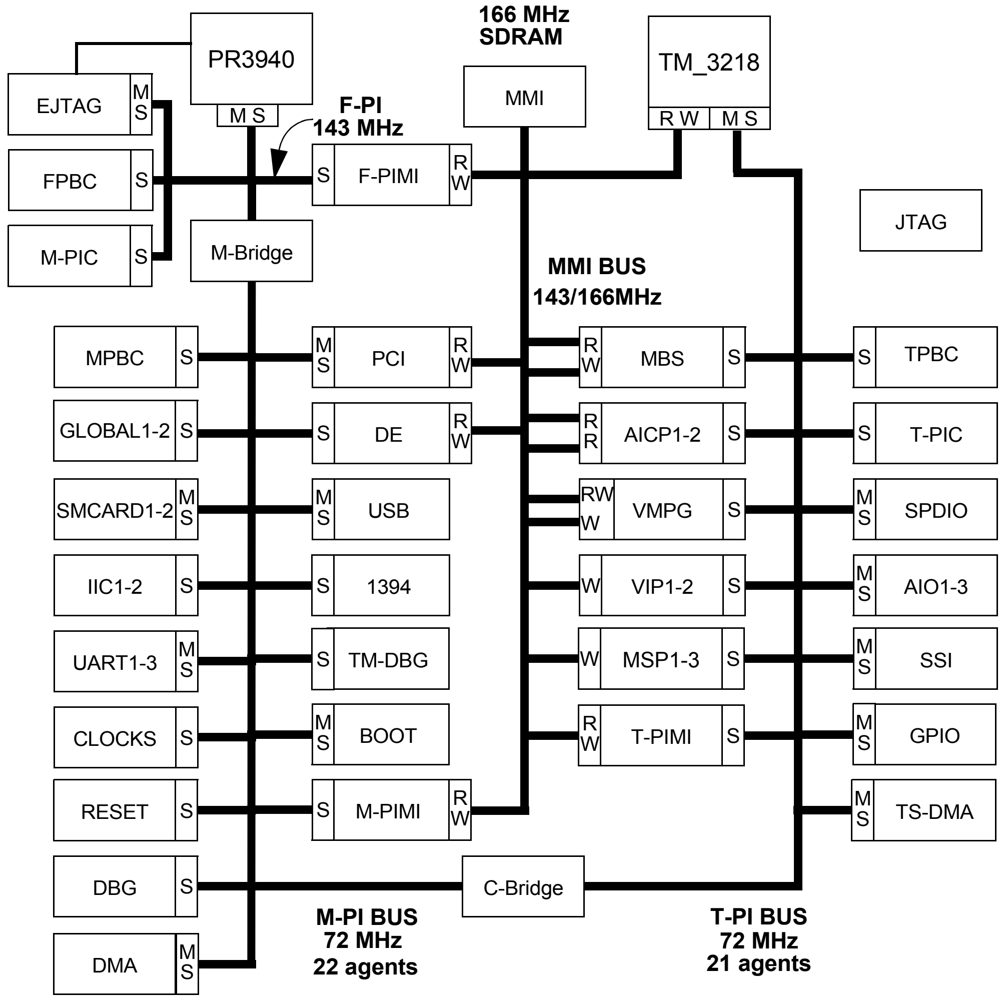


Figure 1: PNX8526 PI-Bus Architecture

There are three PI-Buses in the PNX8526: the Fast PI-Bus (F-PI), the MIPS Peripheral PI-Bus (M-PI), and the TriMedia PI-Bus (T-PI). Each of these buses has its own bus controller (FPBC, MPBC, and TPBC respectively) as shown in the figure above.

Each peripheral on the PI-Bus is located as close as possible to the CPU that is expected to typically control that peripheral.

The GPIO interrupts that may require fast service will be serviced by the TM32. Therefore the GPIO module is located on the T-PI-Bus.

Global (1 and 2) register peripherals contain configuration registers for the chip. Most of the system optimization features are programmable via the Global 2 register.

All peripherals can be accessed from the PR3940, PCI interface, Boot module, EJTAG and DMA controller. Due to system security issues, TM32 access to peripherals can be blocked (see Chapter 2 System Memory Map and Protection). Access between the buses is provided by the M-Bridge and the C-Bridge. All other modules in the PNX8526 are not allowed to access peripherals.

35.3 Operation

35.3.1 Tuning System Performance

Following boot with the internal boot scripts, the system is not performing optimally because the boot scripts are designed to provide a safe system boot that can be used generically for different customers e.g., different memory bus speeds, etc.

There are many ways to optimize system performance. It is essential that all the clocks be running at the desired speed. See the clock module specification for details on how to change the clock frequencies of the system.

The following is recommended for tuning system performance.

- Clock frequency of the MMI bus must be set to the speed of the SDRAMs. This provides maximum bandwidth for the system and also minimizes the latency for CPU accesses.
- The refresh rate of the MMI is programmable in the Global 2 register. This should be changed after boot to allow optimum system performance. Note that the refresh rate is highly temperature-dependent and the number of refreshes can be minimized if a low temperature of the external SDRAM can be guaranteed.
- The F-PIMI has two operating modes: synchronous and asynchronous. When using the synchronous operating mode the F-PI clock speed must be the same as the MMI clock speed. This synchronous mode allows lower latency memory access for the MIPS. When using a 143 MHz memory system it is recommended not to run the MIPS at the maximum speed (150 MHz) but rather, execute at 143 MHz and ensure the F-PIMI is operating in synchronous mode. The performance gain is application dependent. However, unless the cache miss rate is extremely low, using synchronous mode is likely to give the best performance.
  The F-PIMI also has an option for posted writes, which can be enabled from the global register file. If posting is enabled (always recommended) the PR3940 will receive acknowledgement that a write is completed as soon as the write data is transferred to the buffer inside the F-PIMI (i.e., before data actually reaches the SDRAM). If the MIPS CPU wants to ensure that the write buffer in the F-PIMI is empty, it can either execute a SYNC or a load instruction from SDRAM. Both will guarantee the write buffer is flushed.
- The M-Bridge can be set to operate in synchronous and asynchronous mode. The synchronous mode will provide faster access across the bridge and thus minimize the latency for the MIPS to access the peripherals.
  Another feature of the M-Bridge is optional blocking of transfers of error acknowledges (ERR ACK) across the bridge. This is desirable to avoid the MIPS CPU taking an exception after seeing the ERR ACK on the F-PI-Bus. For WIN CE, the exception handler for the ERR ACK is controlled by Microsoft, and

therefore, the action taken when seeing the ERR ACK can not be PNX8526-specific. There are other programmable options for the M-Bridge, which are for testing only and should not be modified.
If there is a collision of accesses from both sides of the bridge, the access from the M-PI-Bus will have priority. Assuming the internal MIPS processor is in use, there should be no need for any accesses from the M-PI-Bus to the F-PI-Bus. The only accesses that could happen would be speculative loads from the TM32, which would limit system performance. There is a feature in the M-PI-Bus controller to block accesses from the M-PI-Bus to the F-PI-Bus.

- The C-Bridge should always operate in asynchronous mode. The latency across the bridge is only a few clock cycles. If there is a collision of accesses from both sides of the bridge, the access from the T-PI-Bus will have priority. When doing accesses across the C-Bridge, several of the PI-buses may be occupied. If so, this will increase the access time for other bus peripherals, and a large number of accesses across the C-Bridge will limit system performance.
The C-Bridge has a feature that can optionally retract a transaction in case there is a pending access across the C-Bridge. This can be done after a programmable interval in the global register (roughly 0.3μs, 0.6μs, 1μs or none). The retract feature will retract the bus transaction all the way back to the originator, and will re-issue the transaction after a minimum 1 clock cycle. This allows the buses to be freed up for other peripherals to do DMA, etc.

- It is possible to program the M-PIMI and the T-PIMI to synchronous mode and to allow write posting. The synchronous mode is not supported and is included for testing only. The write posting is likely to create data coherency issues and should not be enabled.

- The MMI arbiter will be programmed to allocate bandwidth to the peripherals on the MMI bus. To support all PI peripherals being active at the same time and still have low utilization of the peripheral PI-Buses, it is recommended that the maximum waiting time be 2.5 μs for the M-PIMI and T-PIMI to get to the MMI bus. Considering a 143 MHz system, this can be achieved by allocating 1/18 slots to each PIMI. This corresponds to approximately 50 MB/s bandwidth to each PIMI. In general this bandwidth allocation can be minimized if some of the peripherals are not active.

35.3.2 Bus Performance

Assuming the system optimization features described above have been configured, the bus loading of the peripheral PI-buses are expected to be very low. The T-PI Bus is likely to have the highest loading which under a worst case scenario can be as much as 15%. Most systems are expected to have T-PI Bus loading of less than 5% and M-PI Bus loading of less than 2%. This is very low utilization and will guarantee the average access time to MMIO registers, for example, to be very low. On average, the CPU will be waiting less than 1 bus clock cycle from requesting the bus until bus ownership is granted.

35.3.2.1 Round-Robin Arbitration Scheme (Chosen for the PI-Buses)

Round-robin arbitration is designed to avoid deadlock and starvation of any module. It also ensures that peripherals can get sufficient bandwidth and the CPU will get low latency access to the bus. Assuming the system is configured as described above, the maximum waiting time for a peripheral to get access to the bus is 40μs. All peripherals are designed to support this latency. The maximum waiting time for the CPU to get the bus is 3μs. Such long waiting time is extremely unlikely and as described above, the average waiting time for the CPU to get the bus is less than 1 clock cycle. The arbitration scheme and the allocation of slots in the arbiter is described in detail in the following section.

35.4 PI-Bus Controllers

This section describes the PI-Bus Controller Units (PBC) used in the PNX8526. There are three PI-Buses in the PNX8526. The fast PI-Bus (F-PI), the MIPS peripheral PI-Bus (M-PI) and the TriMedia PI-Bus (T-PI). Each of these buses has its own bus controller (FPBC, MPBC and TPBC respectively) as shown in Figure 1.

The PI-Bus controllers provide the following functions:

- Bus arbitration
- Slave selection
- PI slave
- Error generation and status
- Interrupt generation
- Null module
- Generate MIPS external write buffer empty signal in F-PI Bus controller
- MIPS remapping registers in F-PI Bus controller
- Bus keeper
- Bus state machine
- Timeout counter

35.4.1 Bus Arbitration

The arbiter receives request signals (pi_req) from bus masters and assigns bus ownership to one of the masters via its output grant lines (pi_gnt). It operates under control of the bus state machine that enables bus arbitration only in certain cycles. Bus arbitration is done asynchronously within a single bus cycle.

- All the PI-Bus controllers uses a 1 or 2 level round-robin arbitration scheme.
- In the PI-Bus controllers, round-robin means that the priority to own the PI-Bus is determined by the order of the masters in the arbitration scheme following the last owner of the bus.
- If only one master requests the PI-Bus it will get the bus.
- If multiple masters have their PI request lines asserted, the bus will be given to the master with the highest priority.

- Arbitration is done so that no master can starve any other master from accessing the PI-Bus.

35.4.2 Slave Selector

The PI-Bus controllers generate selects for all the PI slave modules according to the address on the PI-Bus. See Table 1 in the PNX8526 Register Summary List for addresses of PI slave apertures.

35.4.3 PI Slave

The PI slave function is activated when a bus operation accesses an address that is mapped to the bus controller’s MMIO aperture or an address that is not mapped to any other bus slave. Therefore the PI slave contains the MMIO registers for the PI-Bus controller and handles PI addresses not looked after by anyone else. See Section 35.4.3.1 below to see what is done for accesses to undefined PI spaces or undefined MMIO apertures.

35.4.3.1 Error Generation

The PI-Bus controller generates error acknowledges (ERR) to support software debug. Below are the rules for error acknowledge generation:

- Due to TriMedia’s speculative reads, PI-Bus error acknowledges are not generated on WDU read accesses to the PNX8526’s MMIO space. Instead, a ready acknowledge and the pi_d = 0xdeadabba is returned (pi_data = 0xdeadabba is used to hint during debug that this is not a true PI read).
- Generate a PI-Bus error acknowledge for WDU read accesses to undefined PI address space and also ignore the read when en_err_u_pi_space bit is enabled. When bit is disabled generate a ready acknowledge and pi_d = 0xdeadabba is returned. The reasoning behind this enable bit is that an external CPU that does speculative reads to undefined PI space may be used and if an error is returned for these reads it will hang the system. If no such CPU is used then you want the errors generated to let software know these reads are incorrect.
- Generate a PI-Bus error acknowledge for non-WDU read accesses to all undefined areas and also ignore the read.
- Generate a PI-Bus error acknowledge for write accesses to undefined areas and also ignore the write.
- Generate a PI-Bus error acknowledge for accesses by tm_owned peripherals to addresses outside the range tm_region_hi to tm_region_lo.
- Generate a PI-Bus error acknowledge for write accesses by TriMedia to non tm_owned peripherals and ignore the write.
- Due to TriMedia’s speculative reads, PI-Bus error acknowledges are not generated on read accesses by TriMedia to non TM-owned peripherals. Instead, a ready acknowledge and the pi_d = 0xdeadabba is returned
- Generate a PI-Bus error if a peripheral DMAs to any peripheral’s MMIO space.

- Generate a PI-Bus error if a T-PI peripheral DMAs to PCI or XIO space.

35.4.4 Interrupt Generation

The bus controllers generate an interrupt for:

- Error acknowledge detected on the PI-Bus
- PI-Bus timeout
- Performance Monitor Overflow (not implemented)

These interrupts can be enabled, cleared, software set and status seen by accessing registers on top of the PI-Bus controllers MMIO space.

35.4.5 Null Module

The PI-Bus controllers will have a 'Null' module which is used for patching all holes in the MMIO space. The Null module will have module ID (0x0124). When the software tries to read the module ID from a 'hole' in the MMIO space, the bus controller will respond with the Null module ID and the size of the hole.

For example, if there is a 4-kB hole, the bus controller will respond with 0x0124.0000 where the last two bytes identify the 4 kB aperture size. Software must recognize the Null ID and realize this is not a real peripheral, but is simply referring to a hole in the MMIO space.

35.4.6 MIPS External Write Buffer Empty

The F-PI Bus controller indicates to the MIPS when its external buffer is empty using the mips_ext_wbempty signal. This feature is enabled using the en_mips_ext_wbempty register. When it is not enabled, mips_ext_wbempty is kept high. This feature should only be enabled when the MIPS external write buffer i.e., the Fast PIMI is configured to do posted writes. See Table 1 below for the MIPS External Write Buffer Empty Truth Table and the F-PI Bus controller register description for more information.

Table 1: MIPS External Write Buffer Empty Truth Table

en_mips_ext_wbempty	fpimi_busy	mips_ext_wbempty
0	0	1
0	1	1
1	0	1
1	1	0

35.4.7 MIPS Remapping Registers

When the MIPS reset is released, execution starts at physical address 1FC0_0000. Depending on the value of the Boot Exception Vector (BEV) bit inside the MIPS, exception vectors can be mapped to addesses 1FC0_0100 or 1FC0_0180. In the PNX8526, SDRAM memory starts at PI address 0000_0000 (due to MIPS) and can be 64 MB at most (highest PI address 3FFFFFC). The execution of the reset vector and boot exception vectors must be mapped to SDRAM or ROM. Therefore the F-PI

Bus controller has registers that respond to these 1FC0_0xxx fetches from MIPS and give back a load address and jump instruction, followed by a few “No Operations” (NOPs), to force the MIPS to jump to an SDRAM or ROM address where the boot software has been located.

Figure 2 shows the addresses these registers are mapped to.

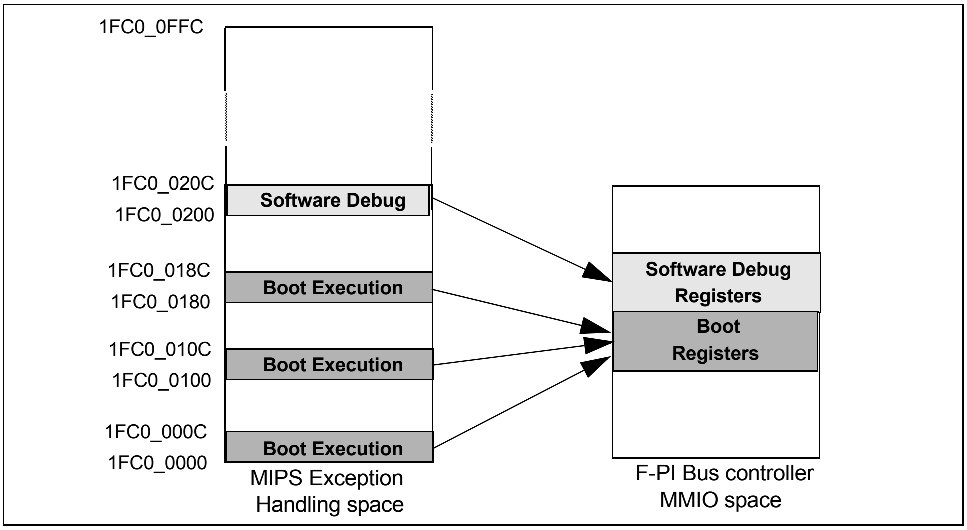


Figure 2: Remapping of MIPS Exception Addresses

PI addresses in the range 1FC0.0000-1FC0.000C, 1FC0.0100-1FC0.010c and 1FC0.0180-1FC0.018C are mapped to the same four exception handling registers for boot in the F-PI Bus controller.

PI addresses in the range 1fc0.0200-1fc0.020c are mapped to four exception handling registers for software debug in the F-PI Bus controller.

Accesses to addresses outside the shaded regions shown above will return NOPs. This is to ensure that if the MIPS fetches more than the four exception handling addresses (1FC0_0X00 to 1FC0_0X0C), it will only fetch instructions that do nothing, (NOPs).

Access to these registers is enabled by register bit en_mips_remap_fpbc. Address remapping is enabled after reset.

The following is a description of both sets of registers:

Exception Handling Registers for Exceptions During Boot

MIPS_BOOT_ADR0 - Programmable register reset to 0x3c08a010 lui t0,0xA010

MIPS_BOOT_ADR4 - Programmable register reset to 0x01000008 jr t0

MIPS_BOOT_ADR8 - Programmable register reset to 0x00000000 nop

MIPS_BOOT_ADRC - Programmable register reset to 0x00000000 nop

The registers will be R/W and typically controlled by the Boot module. The reset values allow boot of the MIPS CPU from 0xA0100000, which is 1 MB from address 0x0 uncached kernel mode.

Due to the internal pipelining of the MIPS CPU, several instructions will be fetched after the jump instruction. However, only the first instruction (NOP) after the jump will be executed.

Exception Handling Registers for Software Debug

SW_EXC_ADR0 - Programmable register reset to 0x3c08a020   lui   t0,0xA020

SW_EXC_ADR4 - Programmable register reset to 0x01000008   jr   t0

SW_EXC_ADR8 - Programmable register reset to 0x00000000   nop

SW_EXC_ADRC - Programmable register reset to 0x00000000   nop

The registers will be R/W and typically controlled by the MIPS CPU. The reset values allow exception handling from 0xA0200000, which is 2 MB from address 0x0 uncached kernel mode.

Remark: The hexidecimal data values shown in this Section 35.4.7 on page 35-762 are correct for litle-endian mode.

35.4.7.1 Typical Programming

The BOOT block programs MIPS_BOOT_ADR0-MIPS_BOOT_ADRC registers with the desired boot location for the MIPS or it will rely on the reset value. More than likely, only MIPS_BOOT_ADR0 register needs to be modified.

When the BOOT block releases the MIPS reset, the MIPS will start fetching code from address 1fc0.0000. The MIPS will execute a 'load register' followed by a 'jump register' and a NOP instruction. Following execution of 'jump register,' the execution will be transferred to the SDRAM.

Later in the MIPS boot code, en_MIPS_remap_fpbc register should be set to 0. This will enable 1fc0.0000-1fc0.1000 to be used as ordinary PI addresses.

During software debug en_MIPS_remap_fpbc must be set to 1. This enables support for the debug exception. The MIPS should program SW_EXC_ADR0-SW_EXC_ADRC registers with the desired location for the exception handler. It is likely that only the SW_EXC_ADR0 register needs to be modified.

Any address in this 4 kB aperture that is not supported will respond with zeros (NOP instructions) on read accesses.

35.4.8 Timeout

The timeout signal can be asserted by the bus controller in two ways:

1. The timeout counter counts the number of wait acknowledges during the data cycles in a bus operation. When the programmed limit is exceeded, the TOUT signal is asserted for one bus cycle. The timeout counter can be totally switched off.

2. TOUT is also asserted if the bus controller receives request timeout from one of the PI-to-PI bridges on its PI-Bus.

Remark: The bus controllers have no way of flagging a slave that is generating infinite retracts.

35.5 Register Descriptions

35.5.1 Register Address Maps

The following tables summarize the PNX8526 PI-Bus registers. PI Bridge and PIMI registers begin at 0x04 D100. They are Global 2 registers.

F-PI Bus Controller registers begin at 0x03 F000. M-PI Bus Controller registers begin at 0x04 E000. T-PI Bus Controller registers begin at 0x10 3000.

Table 2: PI Bridge and PIMI Register Summary (Global 2 Registers)

Offset	Name	Description
0x04_D100	MBRIDGE_CTL	MIPS PI-PI bridge control register
0x04_D104	FPIMI_CTL	F-PI to MI gateway control register
0x04_D108	MPIMI_CTL	M-PI to MI gateway control register
0x04_D10C	TPIMI_CTL	T-PI to MI gateway control register
0x04_D110	CBRIDGE_CTL	PI-PI cross over bridge control register

Table 3: F-PI Bus Controller Register Summary

Offset	Name	Description
0x03_F000	FPBC_CTRL	F-PI Bus timeout control register
0x03_F00C	FPBC_ADDR	F-PI Bus error and timeout address register
0x03_F010	FPBC_STAT	F-PI Bus error and timeout status register
0x03_F014	FPBC_MON	F-PI Bus monitoring register
0x03_F018	EN_EJT_DSU_APERTURE	Enable/Disable EJTAG and DSU
0x03_F01C	EN_MIPS_EXT_WBEMPTY	Enable MIPS external write buffer empty signal
0x03_F020	MIPS_BOOT_ADR0	Exception handling register for Boot
0x03_F024	MIPS_BOOT_ADR4	Exception handling register for Boot
0x03_F028	MIPS_BOOT_ADR8	Exception handling register for Boot
0x03_F02C	MIPS_BOOT_ADRC	Exception handling register for Boot
0x03_F034	EN_MIPS_REMAP_FPBC	Enable mapping of 1fc0.0000-1fc0.1000 to F-PI-bus controller registers
0x03_F040	SW_EXC_ADR0	Exception handling register for software debug
0x03_F044	SW_EXC_ADR4	Exception handling register for software debug
0x03_F048	SW_EXC_ADR8	Exception handling register for software debug
0x03_F04c	SW_EXC_ADRC	Exception handling register for software debug
0x03_FFE0	FPBC_INT_STATUS	Interrupt status register
0x03_FFE4	FPBC_INT_EN	Interrupt enable register

Table 3: F-PI Bus Controller Register Summary *…Continued*

Offset	Name	Description
0x03_FFE8	FPBC_INT_CLR	Interrupt clear register
0x03_FFEC	FPBC_INT_SET	Interrupt software set register
0x03_FFFC	FPBC_MODULE_ID	F-PI Bus controller module identification and revision information

35.5.1.1 PI Bridge and PIMI Registers

Table 4: M-PI Bus Controller Register Summary

Offset	Name	Description
0x04_E000	MPBC_CTRL	M-PI Bus timeout control register
0x04_E00C	MPBC_ADDR	M-PI Bus error and timeout address register
0x04_E010	MPBC_STAT	M-PI Bus error and timeout status register
0x04_E014	MPBC_MON	M-PI Bus monitoring register
0x04_E01C	EN_F_PI_APERTURES	Enable F-PI MMIO space to T-PI master register
0x04_EFE0	MPBC_INT_STATUS	Interrupt status register
0x04_EFE4	MPBC_INT_EN	Interrupt enable register
0x04_EFE8	MPBC_INT_CLR	Interrupt clear register
0x04_EFEC	MPBC_INT_SET	Interrupt software set register
0x04_EFFC	MPBC_MOD_ID	M-PI Bus controller module identification and revision information

Table 5: T-PI Bus Controller Register Summary

Offset	Name	Description
0x10_3000	TPBC_CTRL	T-PI Bus timeout control register
0x10_300C	TPBC_ADDR	T-PI Bus error and timeout address register
0x10_3010	TPBC_STAT	T-PI Bus error and timeout status register
0x10_3014	TPBC_MON	T-PI bus monitoring register
0x10_3FE0	TPBC_INT_STATUS	Interrupt Status register
0x10_3FE4	TPBC_INT_EN	Interrupt Enable register
0x10_3FE8	TPBC_INT_CLR	Interrupt Clear register
0x10_3FEC	TPBC_INT_SET	Interrupt software set register
0x10_3FFC	TPBC_MOD_ID	T-PI Bus controller module identification and revision information

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 D100			*MBRIDGE_CTL*	
31:9		-	Unused	Ignore during writes and read as zeroes.
8	R/W	0x0	MPIB_CLKS_SYNC_CF	Clock relationship across the bridge 0 = Clocks are asynchronous. 1 = F-PI to M-PI Bus clock frequency relationship is 2:1.
7	R/W	0x1	MPIB_SEC_PRIORITY_CF	Priority in case of requests from both sides at the same time 0 = F-PI Bus side has priority. 1 = M-PI Bus side has priority. Note: This bit needs to be set to logic 1 in the PNX8526 so that TriMedia can not accept a transaction to its slave MMIO registers while its master is being retracted. To avoid this, the Trimedia side needs priority, which is the M-PI side of the bridge in this case.
6	R/W	0x0	MPIB_BLOCK_ERR_CF	Blocking error acknowledge across the MIPS PI-PI bridge 0 = Do not block error acknowledge. 1 = Block error acknowledge.
5	R/W	0x0	MPIB_POSTED_WRITE_CF	Posted writes 0 = No posting 1 = All writes across bridges are posted writes. (Must be logic 0 for the PNX8526 as posted writes are not supported in the system.)
4:3	R/W	0x3	MPIB_P_RETRACT_CF[1:0]	The bridge can abort an F-PI to M-PI transaction by sending back a retract to the F-PI Bus if it has been waiting too long to get onto the M-PI Bus. The wait time is defined below. 00 = Waits 22 F-PI clock cycles before it retracts. 01 = Waits 43 F-PI clock cycles before it retracts. 10 = Waits 72 F-PI clock cycles before it retracts. 11 = Never retracts.
2:1	R/W	0x3	MPIB_S_RETRACT_CF[1:0]	The bridge can abort an M-PI to F-PI transaction by sending back a retract to the M-PI Bus if it has been waiting too long to get onto the F-PI Bus. The wait time is defined below. 00 = Waits 22 M-PI clock cycles before it retracts. 01 = Waits 43 M-PI clock cycles before it retracts. 10 = Waits 72 M-PI clock cycles before it retracts. 11 = Never retracts.
0	R/W	0x0	MPIB_WINDEBUG_CF	0 = All read types (halfword, tri-byte and byte) are not changed by the bridge. 1 = Halfword, tri-byte and byte read transactions on the source side of the bridge are translated into word reads on the target side (for debug only).
Offset 0x04 D104			*FPIMI_CTL*	
31:2		-	Unused	Ignore during writes and read as zeroes.

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
1	R/W	0x0	FPIMI_POSTED_WRITE	Posted writes 0 = No posting 1 = All writes across the PI to MI gateway are posted writes.
0	R/W	0x0	FPIMI_CLOCKS_ARE_SYNC	Clock relationship across the PI-to-MI gateway 0 = Clocks are asynchronous. 1 = Clocks are synchronous.
Offset 0x04 D108			*MPIMI_CTL*	
31:2		-	Unused	Ignore during writes and read as zeroes.
1	R/W	0x0	MPIMI_POSTED_WRITE	Posted writes 0 = No posting 1 = All writes across the PI-to-MI gateway are posted writes (must be logic zero for the PNX8526).
0	R/W	0x0	MPIMI_CLOCKS_ARE_SYNC	Clock relationship across the PI-to-MI gateway 0 = Clocks are asynchronous. 1 = Clocks are synchronous. (Must be logic 0 for the PNX8526.)
Offset 0x04 D10C			*TPIMI_CTL*	
31:2		-	Unused	Ignore during writes and read as zeroes.
1	R/W	0x0	TPIMI_POSTED_WRITE	Posted writes 0 = No posting 1 = All writes across the PI-to-MI gateway are posted writes. (Must be logic zero for the PNX8526.)
0	R/W	0x0	TPIMI_CLOCKS_ARE_SYNC	Clock relationship across the PI-to-MI gateway 0 = Clocks are asynchronous. 1 = Clocks are synchronous. (Must be logic zero for the PNX8526.)
Offset 0x04 D110			*CBRIDGE_CTL*	
31:9		-	Unused	Ignore during writes and read as zeroes.
8	R/W	0x0	CPIB_CLKS_SYNC_CF	Clock relationship across the bridge 0 = Clocks are asynchronous. 1 = M-PI to T-PI Bus clock frequency relationship is 2:1.
7	R/W	0x1	CPIB_SEC_PRIORITY_CF	Priority in case of requests from both sides at the same time 0 = M-PI Bus side has priority. 1 = T-PI Bus side has priority. Note: This bit needs to be set to logic 1 in the PNX8526 so that TriMedia can not accept a transaction to its slave MMIO registers while its master is being retracted. To avoid this, the Trimedia side needs priority, which is the T-PI side of the bridge in this case.
6	R/W	0x0	CPIB_BLOCK_ERR_CF	Blocking error acknowledge across the cross over PI-PI bridge 0 = Do not block error acknowledge. 1 = Block error acknowledge.

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
5	R/W	0x0	CPIB_POSTED_WRITE_CF	Posted writes 0 = No posting 1 = All writes across bridges are posted writes. (Must be logic zero for the PNX8526 as posted writes are not supported in the system.)
4:3	R/W	0x0	CPIB_P_RETRACT_CF[1:0]	The bridge can abort an M-PI to T-PI transaction by sending back a retract to the M-PI Bus if it has been waiting too long to get onto the T-PI Bus. The wait time is defined below. 00 = Waits 22 M-PI clock cycles before it retracts. 01 = Waits 43 M-PI clock cycles before it retracts. 10 = Waits 72 M-PI clock cycles before it retracts. 11 = Never retracts.
2:1	R/W	0x0	CPIB_S_RETRACT_CF[1:0]	The bridge can abort a T-PI to M-PI transaction by sending back a retract to the T-PI Bus if it has been waiting too long to get onto the M-PI Bus. The wait time is defined below. 00 = Waits 22 T-PI clock cycles before it retracts. 01 = Waits 43 T-PI clock cycles before it retracts. 10 = Waits 72 T-PI clock cycles before it retracts. 11 = Never retracts.
0	R/W	0x0	CPIB_WINDEBUG_CF	0 = All read types (Halfword, tri-byte and byte) are not changed by the bridge. 1 = Halfword, tri-byte and byte read transactions on the source side of the bridge are translated into word reads on the target side (for debug only).

35.5.1.2 F-PI Bus Controller Registers

F-PI BUS CONTROLLER (FPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 F000			*FPBC_CTRL*	
31:5		-	Unused	Ignore upon read. Write as zeroes.
4:1	R/W	0x0	TOUT_SEL[3:0]	Timeout select 0x0 = Timeout generated after 1 wait cycle. 0x1 = Timeout generated after 3 consecutive wait cycles. 0x2 = Timeout generated after 7 consecutive wait cycles. 0x3 = Timeout generated after 15 consecutive wait cycles. 0x4 = Timeout generated after 31 consecutive wait cycles. 0x5 = Timeout generated after 63 consecutive wait cycles. 0x6 = Timeout generated after 127 consecutive wait cycles. 0x7 = Timeout generated after 255 consecutive wait cycles. 0x8 = Timeout generated after 511 consecutive wait cycles. 0x9 = Timeout generated after 1,023 consecutive wait cycles. 0xa = Timeout generated after 2,047 consecutive wait cycles. 0xb = Timeout generated after 4,095 consecutive wait cycles. 0xc = Timeout generated after 8,191 consecutive wait cycles. 0xd = Timeout generated after 16,383 consecutive wait cycles. 0xe = Timeout generated after 32,767 consecutive wait cycles. 0xf = Timeout generated after 65,535 consecutive wait cycles.
0	R/W	0x1	TOUT_OFF	Timeout disable 0x0 = Timeout enabled. 0x1 = Timeout disabled.
Offset 0x03 F00C			*FPBC_ADDR*	
31:2	R	0x0000 0000-	ERR_TOUT_ADDR[31:2]	Full 30 bits of the F-PI address bus which causes a PI error or timeout.
1:0		-	Unused	Ignore upon read. Write as zeroes.
Offset 0x03 F010			*FPBC_STAT*	
31:27		-	Unused	Ignore upon read. Write as zeroes.
26:24	R	0x0	ERR_TOUT_GNT[2:0]	Active master causing error or timeout 0x1 = EJTAG 0x2 = MIPS side of M-bridge 0x4 = MIPS processor.
23:20		-	Unused	Ignore upon read. Write as zeroes.

F-PI BUS CONTROLLER (FPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
19:16	R	0x0	ERR_TOUT_SEL[3:0]	Selected agent during error or timeout 00 = MIPS PIC or None 01 = Fast PI to Memory highway bus Interface (F-PIMI) 02 = MIPS side of M-bridge 03 = F-PI Bus controller 04 = MIPS processor debug (DSU) 05 = EJTAG 1MB probe memory 06 = EJTAG data registers 07 = MIPS exception handling space 08 = Null 09 = Default slave. **Note:** MIPS PIC is not specifically identified.
15:13		-	Unused	Ignore upon read. Write as zeroes.
12:8	R	0x00	ERR_TOUT_OPC[4:0]	Opcode causing an error or timeout 0x00 = No-operation (no communication with bus slave agent) 0x01 = Block of 32 words, regular address space 0x02 = 1 word, regular address space 0x03 = 1 word, control address space 0x04 = Block of 2 words, regular address space 0x05 = Block of 4 words, regular address space 0x06 = Block of 8 words, regular address space 0x07 = Block of 16 words, regular address space 0x08 = 1 halfword address [1:0] = 00, regular address space 0x09 = 1 triple-byte address [1:0] = 00, regular address space 0x0a = 1 halfword address [1:0] = 10, regular address space 0x0b = 1 triple-byte address [1:0] = 01, regular address space 0x0c = 1 byte address [1:0] = 00, regular address space 0x0d = 1 byte address [1:0] = 01, regular address space 0x0e = 1 byte address [1:0] = 10, regular address space 0x0f = 1 byte address [1:0] = 11, regular address space 0x10 = Reserved 0x11 = Reserved 0x12 = 1 double word, regular address space 0x13 = Reserved 0x14 = block of 2 double words, regular address space 0x15 = block of 4 double words, regular address space 0x16 = block of 8 double words, regular address space 0x17 = block of 16 double words, regular address space The rest are reserved.
7:3		-	Unused	Ignore upon read. Write as zeroes.

F-PI BUS CONTROLLER (FPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
2:0	R	0x0	ERR_TOUT_ACK[2:0]	Acknowledge during error or timeout 0x0 = Wait: bus operation not completed in current bus cycle. 0x1 = Ready-More: bus operation successfully completed in current bus cycle and slave is immediately able to process a further bus operation of the same data direction. 0x2 = Error: bus operation has lead to error condition 0x3 = Ready: bus operation, successfully completed in current bus cycle. 0x4 = Retract: refuse to complete bus operation. The rest are reserved.
Offset 0x03 F014			*FPBC_MON*	
31:19		-	Unused	Ignore upon read. Write as zeroes.
18:16	R	0x0	GNT_LAST[2:0]	Last active master 0x1 = EJTAG 0x2 = MIPS side of M-bridge 0x4 = MIPS processor.
15:3		-	Unused	Ignore upon read. Write as zeroes.
2:0	R	0x0	REQ_CURR[2:0]	Current requests 0x1 = EJTAG 0x2 = MIPS side of M-bridge 0x4 = MIPS processor.
Offset 0x03 F018			*EN_EJT_DSU_APERTURE*	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0x1	EN_EJT_DSU_APERTURE	Enable EJTAG probe memory and MIPS DSU spaces 0 = PI addresses from FF20_0000 to FF2F_FFFFdo not generate an EJTAG probe memory. PI select and PI addresses from FF30_0000 to FF3F_FFFF don’t generate a MIPS DSU PI select. 1 = PI addresses from FF20_0000 to FF2F_FFFFgenerate an EJTAG probe memory. PI select and PI addresses from FF30_0000 to FF3F_FFFF generate a MIPS DSU PI select.
Offset 0x03 F01C			*EN_MIPS_EXT_WBEMPTY*	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0x1	EN_MIPS_EXT_WBEMPTY	Enable signal to tell MIPS that its external buffer is full 0 = MIPS external write buffer empty signal is kept always high. Used when the MIPS external write buffer (F-PIMI) is programmed to non-posted writes. 1 = MIPS external write buffer empty signal equals the inverted version of F-PIMI busy. Used when the MIPS external write buffer F-PIMI is programmed to posted writes.

F-PI BUS CONTROLLER (FPBC) REGISTERS				
	Read/	Reset	Name	
Bits	Write	Value	(Field or Function)	Description
Offset 0x03 F020			***MIPS_BOOT_ADR0***	
31:0	R/W	0x3c08 a010	MIPS_BOOT_ADR0[31:0]	Exception handling register for Boot. This register usually contains a load register command. It is used to get the MIPS to load the jump address into a MIPS register. The reset value of this register = lui t0,0xA0F0 which means load the jump address A0F0_0000 into MIPS register t0. A010_0000 is 1Mbyte above zero uncached kernel mode.
Offset 0x03 F024			***MIPS_BOOT_ADR4***	
31:0	R/W	0x0100 0008	MIPS_BOOT_ADR4[31:0]	Exception handling register for Boot. This register usually contains a jump register command. It is used to get the MIPS to jump to the address that is stored in a MIPS register. The reset value of this register = jr t0 which means jump to the address stored in MIPS register t0.
Offset 0x03 F028			***MIPS_BOOT_ADR8***	
31:0	R/W	0x0000 0000	MIPS_BOOT_ADR8[31:0]	Exception handling register for Boot. This register usually contains a NOP command. It is used to get the MIPS to do No Operation. Reset value = NOP.
Offset 0x03 F02C			***MIPS_BOOT_ADRC***	
31:0	R/W	0x0000 0000	MIPS_BOOT_ADRC[31:0]	Exception handling register for Boot. This register usually contains a NOP command. It is used to get the MIPS to do No Operation. Reset value = NOP.
Offset 0x03 F034			***EN_MIPS_REMAP_FPBC***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0x1	EN_MIPS_REMAP_FPBC	Enables remapping of MIPS exception handling addresses to F-PI Bus controller registers. 0 = Do not remap MIPS exception handling addresses. 1 = Remap MIPS exception handling addresses to F-PI Bus controller registers.
Offset 0x03 F040			***SW_EXC_ADR0***	
31:0	R/W	0x3c08 a020	SW_EXC_ADR0[31:0]	Exception handling register for software debug. This register usually contains a load register command. It is used to get the MIPS to load the jump address into a MIPS register. The reset value of this register = lui t0,0xA020 which means load the jump address A020_0000 into MIPS register t0. A020_0000 is 2 Mbytes above zero uncached kernel mode.
Offset 0x03 F044			***SW_EXC_ADR4***	
31:0	R/W	0x0100 0008	SW_EXC_ADR4[31:0]	Exception handling register for Boot. This register usually contains a jump register command. It is used to get the MIPS to jump to the address that is stored in a MIPS register. The reset value of this register = jr t0 which means jump to the address stored in MIPS register t0.

F-PI BUS CONTROLLER (FPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 F048			*SW_EXC_ADR8*	
31:0	R/W	0x0000 0000	SW_EXC_ADR8[31:0]	Exception handling register for Boot. This register usually contains a NOP command. It is used to get the MIPS to do No Operation. Reset value = NOP.
Offset 0x03 F04C			*SW_EXC_ADRC*	
31:0	R/W	0x0000 0000	SW_EXC_ADRC[31:0]	Exception handling register for Boot. This register usually contains a NOP command. It is used to get the MIPS to do No Operation. Reset value = NOP.
Offset 0x03 FFE0			*FPBC_INT_STATUS*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	R	0	INT_STATUS_TOUT	Timeout interrupt status register. It reports any pending timeout interrupts: 1 = Timeout interrupt pending, meaning F-PI Bus controller has generated a timeout because some F-PI device has violated the programmable limit for timeout (see FPBC_TOUT) or M-bridge generated a timeout request (see PI-PI bridge). 0 = Timeout interrupt not pending.
0	R	0	INT_STATUS_ERROR	Error interrupt status register. It reports any pending error interrupts 1 = Error interrupt pending. F-PI Bus controller has detected an error acknowledge on the F-PI Bus. 0 = Error interrupt not pending.
Offset 0x03 FFE4			*FPBC_INT_EN*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	R/W	0	INT_ENABLE_TOUT	Timeout interrupt enable register 1 = Timeout interrupt is enabled. 0 = Timeout interrupt is disabled.
0	R/W	0	INT_ENABLE_ERROR	Timeout interrupt enable register 1 = Error interrupt is enabled 0 = Error interrupt is disabled.
Offset 0x03 FFE8			*FPBC_INT_CLR*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	INT_CLEAR_TOUT	Timeout interrupt clear register. This is written by software to clear the interrupt. 1 = Timeout interrupt is cleared. 0 = Timeout interrupt is not cleared.
0	W	0	INT_CLEAR_ERROR	Error interrupt clear register. This is written by software to clear the interrupt. 1 = Error interrupt is cleared. 0 = Error interrupt is not cleared.

F-PI BUS CONTROLLER (FPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 FFEC			***FPBC_INT_SET***	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	INT_SET_TOUT	Timeout interrupt set register. Allows software to set interrupts. 1 = Timeout interrupt is set. 0 = Timeout interrupt is not set.
0	W	0	INT_SET_ERROR	Error interrupt set register. Allows software to set interrupts. 1 = Error interrupt is set. 0 = Error interrupt is not set.
Offset 0x03 FFF0—FFF8			***Reserved***	
Offset 0x03 FFFC			***FPBC_MODULE_ID***	
31:16	R	0x0102	MODULE_ID[15:0]	Unique 16-bit code. Module ID 0 and 1 are reserved for future use. Value 0x0102 is for the PNX8526 F-PI Bus controller module.
15:12	R	0	MAJREV[3:0]	Major Revision: any revision that might break software compatibility.
11:8	R	1	MINREV[3:0]	Minor Revision: any revision that keeps software compatible.
7:0	R	0	MODULE_APERTURE_SIZE	Aperture size = 4 kB*(bit_value+1), so 0 means 4 kB (the default).

35.5.1.3 M-PI Bus Controller Registers

M-PI BUS CONTROLLER (MPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 E000			*MPBC_CTRL*	
31:5		-	Unused	Ignore upon read. Write as zeroes.
4:1	R/W	0x0	TOUT_SEL[3:0]	Timeout select 0x0 = Timeout generated after 1 wait cycle. 0x1 = Timeout generated after 3 consecutive wait cycles. 0x2 = Timeout generated after 7 consecutive wait cycles. 0x3 = Timeout generated after 15 consecutive wait cycles. 0x4 = Timeout generated after 31 consecutive wait cycles. 0x5 = Timeout generated after 63 consecutive wait cycles. 0x6 = Timeout generated after 127 consecutive wait cycles. 0x7 = Timeout generated after 255 consecutive wait cycles. 0x8 = Timeout generated after 511 consecutive wait cycles. 0x9 = Timeout generated after 1,023 consecutive wait cycles. 0xa = Timeout generated after 2,047 consecutive wait cycles. 0xb = Timeout generated after 4,095 consecutive wait cycles. 0xc = Timeout generated after 8,191 consecutive wait cycles. 0xd = Timeout generated after 16,383 consecutive wait cycles. 0xe = Timeout generated after 32,767 consecutive wait cycles. 0xf = Timeout generated after 65,535 consecutive wait cycles.
0	R/W	0x1	TOUT_OFF	Timeout disable 0x0 = Timeout enabled. 0x1 = Timeout disabled.
Offset 0x04 E00C			*MPBC_ADDR*	
31:2	R	0x0000 0000-	ERR_TOUT_ADDR[31:2]	Full 30 bits of the M-PI address bus which causes a PI error or timeout.
1:0		-	Unused	Ignore upon read. Write as zeroes.
Offset 0x04 E010			*MPBC_STAT*	
31:20	R	0x000	ERR_TOUT_GNT[11:0]	Active master causing error or timeout 0x001 = M-PI Noise generator 0x002 = DMA Engine 0x004 = PCI 0x008 = USB 0x010 = UART 3 0x020 = UART 2 0x040 = UART 1 0x080 = Smartcard 2 0x100 = Smartcard 1 0x200 = Boot 0x400 = MIPS side of C-bridge 0x800 = Peripheral side of M-bridge.
19:18		-	Unused	Ignore upon read. Write as zeroes.

M-PI BUS CONTROLLER (MPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
17:12	R	0x00	ERR_TOUT_SEL[5:0]	Selected agent during error or timeout 0x00 = None 0x01 = PCI 0x02 = XIO 0x03 = M-PI to Memory highway bus Interface (M-PIMI) 0x04 = Peripheral side of M-bridge 0x05 = MMIO registers in PCI block 0x06 = Debug 0x07 = Boot 0x08 = Smartcard 1 0x09 = Smartcard 2 0x0a = I2C 1 0x0b = I2C 2 0x0c = Reset 0x0d = USB 0x0e = Link 1394 0x0f = UART 1 0x10 = UART 2 0x11 = UART 3
17:12 cont'd.	R	0x00	ERR_TOUT_SEL[5:0]	Selected agent during error or timeout (cont'd.) 0x12 = Global register 1 0x13 = M-PI Bus controller 0x14 = 2D Drawing Engine 0x15 = Clock 0x16 = TrIMedia debug 0x17 = DMA 0x18 = Global Register 2 0x19 = MIPS side of C-bridge 0x20 = Null 0x21 = Default slave
11:9		-	Unused	Ignore upon read. Write as zeroes.
8:4	R	0x00	ERR_TOUT_OPC[4:0]	Opcode causing error or timeout. See bits 12:8 of the FPBC_STAT register (offset 0x03_F010) for the Opcode table.
3		-	Unused	Ignore upon read. Write as zeroes.
2:0	R	0x0	ERR_TOUT_ACK[2:0]	Acknowledge during error or timeout. See bits 2:0 of the FPBC_STAT register (offset 0x03_F010) for the ACKs table.
Offset 0x04 E014			*MPBC_MON*	
31:28		-	Unused	Ignore upon read. Write as zeroes.

M-PI BUS CONTROLLER (MPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
27:16	R	0x000	GNT_LAST[11:0]	Last active master 0x001 = M-PI Noise generator 0x002 = DMA Engine 0x004 = PCI 0x008 = USB 0x010 = UART 3 0x020 = UART 2 0x040 = UART 1 0x080 = Smartcard 2 0x100 = Smartcard 1 0x200 = Boot 0x400 = MIPS side of C-bridge 0x800 = Peripheral side of M-bridge
15:12		-	Unused	Ignore upon read. Write as zeroes.
11:0	R	0x000	REQ_CURR[11:0]	Current requests 0x001 = M-PI Noise generator 0x002 = DMA Engine 0x004 = PCI 0x008 = USB 0x010 = UART 3 0x020 = UART 2 0x040 = UART 1 0x080 = Smartcard 2 0x100 = Smartcard 1 0x200 = Boot 0x400 = MIPS side of C-bridge 0x800 = Peripheral side of M-bridge
Offset 0x04 E01C			*EN_F_PI_APERTURES*	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R	0x1	EN_F_PI_APERTURES	Enable F-PI MMIO space to T-PI master registers. 0x0 = Disable F-PI MMIO space to T-PI masters 0x1 = Enable F-PI MMIO space to T-PI masters.
Offset 0x04 EFE0			*MPBC_INT_STATUS*	
31:2			Unused	Ignore upon read. Write as zeroes.
1	R	0	INT_STATUS_TOUT	Timeout interrupt status register. It reports any pending timeout interrupts: 0x1 = Timeout interrupt pending. M-PI Bus controller has generated a timeout because some M-PI device has violated the programmable limit for timeout (see MPBC_TOUT). Or M-bridge or C-bridge generated a timeout request (see PI-PI bridge). 0x0 = Timeout interrupt not pending.
0	R	0	INT_STATUS_ERROR	Error interrupt status register. It reports any pending error interrupts. 0x1 = Error interrupt pending: F-PI Bus controller has detected an error acknowledge on the M-PI Bus. 0x0 = Error interrupt not pending.

M-PI BUS CONTROLLER (MPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 EFE4			*MPBC_INT_EN*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	R/W	0	INT_ENABLE_TOUT	Timeout interrupt enable register 0x1 = Timeout interrupt is enabled. 0x0 = Timeout interrupt is disabled.
0	R/W	0	INT_ENABLE_ERROR	Timeout interrupt enable register 0x1 = Error interrupt is enabled. 0x0 = Error interrupt is disabled.
Offset 0x04 EFE8			*MPBC_INT_CLR*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	INT_CLEAR_TOUT	Timeout interrupt clear register. This is written by software to clear the interrupt. 0x1 = Timeout interrupt is cleared. 0x0 = Timeout interrupt is not cleared.
0	W	0	INT_CLEAR_ERROR	Error interrupt clear register. This is written by software to clear the interrupt. 0x1 = Error interrupt is cleared. 0x0 = Error interrupt is not cleared.
Offset 0x04 EFEC			*MPBC_INT_SET*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	INT_SET_TOUT	Timeout interrupt set register. Allows software to set interrupts. 1 = Timeout interrupt is set. 0 = Timeout interrupt is not set.
0	W	0	INT_SET_ERROR	Error interrupt set register. Allows software to set interrupts. 0x1 = Error interrupt is set. 0x0 = Error interrupt is not set.
Offset 0x04 EFFC			*MPBC_MODULE_ID*	
31:16	R	0x010B	MODULE_ID[15:0]	Unique 16-bit code. Module ID 0 and 1 are reserved for future use. Value 0x010B is for the PNX8526 M-PI Bus controller module.
15:12	R	0	MAJREV[3:0]	Major Revision: any revision that might break software compatibility.
11:8	R	1	MINREV[3:0]	Minor Revision: any revision that keeps software compatible.
7:0	R	0	MODULE_APERTURE_SIZE[7:0]	Aperture size = 4kB*(bit_value+1), so 0 means 4kB (the default).

35.5.1.4 T-PI Bus Controller Registers

T-PI BUS CONTROLLER (TPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x10 3000			*TPBC_CTRL*	
31:5		-	Unused	Ignore upon read. Write as zeroes.
4:1	R/W	0x0	TOUT_SEL[3:0]	Timeout select 0x0 = Timeout generated after 1 wait cycle. 0x1 = Timeout generated after 3 consecutive wait cycles. 0x2 = Timeout generated after 7 consecutive wait cycles. 0x3 = Timeout generated after 15 consecutive wait cycles. 0x4 = Timeout generated after 31 consecutive wait cycles. 0x5 = Timeout generated after 63 consecutive wait cycles. 0x6 = Timeout generated after 127 consecutive wait cycles. 0x7 = Timeout generated after 255 consecutive wait cycles. 0x8 = Timeout generated after 511 consecutive wait cycles. 0x9 = Timeout generated after 1,023 consecutive wait cycles. 0xa = Timeout generated after 2,047 consecutive wait cycles. 0xb = Timeout generated after 4,095 consecutive wait cycles. 0xc = Timeout generated after 8,191 consecutive wait cycles. 0xd = Timeout generated after 16,383 consecutive wait cycles. 0xe = Timeout generated after 32,767 consecutive wait cycles. 0xf = Timeout generated after 65,535 consecutive wait cycles.
0	R/W	0x1	TOUT_OFF	Timeout disable 0x0 = Timeout enabled. 0x1 = Timeout disabled.
Offset 0x10 300C			*TPBC_ADDR*	
31:2	R	0x0000 0000-	ERR_TOUT_ADDR[31:2]	Full 30 bits of the T-PI address bus which causes a PI error or timeout.
1:0		-	Unused	Ignore upon read. Write as zeroes.
Offset 0x10 3010			*TPBC_STAT*	
31:30		-	Unused	Ignore upon read. Write as zeroes.
29:16	R	0x00	ERR_TOUT_GNT[13:0]	Active master causing error or timeout 0x0001 = T-PI noise generator 0x0002 = GPIO 0x0004 = TS DMA 0x0008 = Audio Input 3 0x0010 = Audio Output 3 0x0020 = Audio Input 2 0x0040 = Audio Output 2 0x0080 = Audio Input 1 0x0100 = Audio Output 1 0x0200 = Sony Philips Digital Input 0x0400 = Sony Philips Digital Output 0x0800 = Synchronous Serial Interface 0x1000 = PI-PI cross over bridge 0x2000 = Trimedia
15		-	Unused	Ignore upon read. Write as zeroes.

T-PI BUS CONTROLLER (TPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
14:10	R	0x00	ERR_TOUT_SEL[4:0]	Selected agent during error or timeout 0x00 = None 0x01 = PI-PI cross over bridge 0x02 = T-PI to Memory highway bus Interface (T-PIMI) 0x03 = Trimedia 0x04 = T-PI interrupt controller 0x05 = T-PI Bus controller 0x06 = GPIO 0x07 = Video MPEG2 0x08 = Video Input Processor 1 0x09 = Video Input Processor 2 0x0a = Synchronous Serial Interface 0x0b = Sony Philips Digital Output 0x0c = Sony Philips Digital Input 0x0d = Memory Based Scala 0x0e = Image Composition Processor 1 0x0f = Image Composition Processor 2
14:10 cont'd.	R	0x00	ERR_TOUT_SEL[4:0]	Selected agent during error or timeout (cont'd.) 0x10 = Audio Output 1 0x11 = Audio Input 1 0x12 = Audio Output 2 0x13 = Audio Input 2 0x14 = Audio Output 3 0x15 = Audio Input 3 0x16 = MPEG System Processor 1 0x17 = MPEG System Processor 2 0x18 = MPEG System Processor 3 0x19 = TSDMA 0x1a = Null 1 0x1b = Null 2 0x1c = Null 3 0x1d = Null 4 0x1e = Default slave
9		-	Unused	Ignore upon read. Write as zeroes.
8:4	R	0x00	ERR_OPC[4:0]	Opcode causing error or timeout. See bits 12:8 of the FPBC_STAT register (offset 0x03_F010) for the Opcode table.
3		-	Unused	Ignore upon read. Write as zeroes.
2:0	R	0x00	ERR_ACK[2:0]	Acknowledge during error or timeout. See bits 2:0 of the FPBC_STAT register (offset 0x03_F010) for the ACKs table.
Offset 0x10 3014			*TPBC_MON*	
31:30		-	Unused	Ignore upon read. Write as zeroes.

T-PI BUS CONTROLLER (TPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
29:16	R	0x000	GNT_LAST[13:0]	Last active master 0x0001 = T-PI noise generator 0x0002 = GPIO 0x0004 = TS DMA 0x0008 = Audio Input 3 0x0010 = Audio Output 3 0x0020 = Audio Input 2 0x0040 = Audio Output 2 0x0080 = Audio Input 1 0x0100 = Audio Output 1 0x0200 = Sony Philips Digital Input 0x0400 = Sony Philips Digital Output 0x0800 = Synchronous Serial Interface 0x1000 = PI-PI cross over bridge 0x2000 = TriMedia
15:14		-	Unused	Ignore upon read. Write as zeroes.
13:0	R	0x000	REQ_CURR[13:0]	Current requests 0x0001 = T-PI noise generator 0x0002 = GPIO 0x0004 = TS DMA 0x0008 = Audio Input 3 0x0010 = Audio Output 3 0x0020 = Audio Input 2 0x0040 = Audio Output 2 0x0080 = Audio Input 1 0x0100 = Audio Output 1 0x0200 = Sony Philips Digital Input 0x0400 = Sony Philips Digital Output 0x0800 = Synchronous Serial Interface 0x1000 = PI-PI cross over bridge 0x2000 = TriMedia
Offset 0x10 3FE0			*TPBC_INT_STATUS*	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	R	0	INT_STATUS_TOUT	Timeout interrupt status. Reports any pending timeout interrupts: 0x1 = Timeout interrupt pending: T-PI Bus controller has generated a timeout because a T-PI device has violated the programmable limit for timeout (see TPBC_TOUT). Or M-bridge or C-bridge generated a timeout request (see PI-PI bridge). 0x0 = Timeout interrupt is not pending.
0	R	0	INT_STATUS_ERROR	Error interrupt status. Reports any pending error interrupts: 0x1 = Error interrupt pending, meaning F-PI Bus controller has detected an error acknowledge on the M-PI Bus. 0x0 = Error interrupt is not pending.
Offset 0x10 3FE4			*TPBC_INT_EN*	
31:2		-	Unused	Ignore upon read. Write as zeroes.

T-PI BUS CONTROLLER (TPBC) REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
1	R/W	0	INT_ENABLE_TOUT	Timeout interrupt enable register 0x1 = Timeout interrupt is enabled 0x0 = Timeout interrupt is disabled.
0	R/W	0	INT_ENABLE_ERROR	Timeout interrupt enable register 0x1 = Error interrupt is enabled 0x0 = Error interrupt is disabled.
Offset 0x10 3FE8			***TPBC_INT_CLR***	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	INT_CLEAR_TOUT	Timeout interrupt clear register. This is written by software to clear the interrupt. 0x1 = Timeout interrupt is cleared 0x0 = Timeout interrupt is not cleared.
0	W	0	INT_CLEAR_ERROR	Error interrupt clear register. This is written by software to clear the interrupt. 0x1 = Error interrupt is cleared 0x0 = Error interrupt is not cleared.
Offset 0x10 3FEC			***TPBC_INT_SET***	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	INT_SET_TOUT	Timeout interrupt set register. Allows software to set interrupts. 0x1 = Timeout interrupt is set 0x0 = Timeout interrupt is not set.
0	W	0	INT_SET_ERROR	Error interrupt set register. Allows software to set interrupts. 0x1 = Error interrupt is set 0x0 = Error interrupt is not set.
Offset 0x10 3FFC			***TPBC_MODULE_ID***	
31:16	R	0x0112	MODULE_ID[15:0]	Unique 16-bit code. Module ID 0 and 1 are reserved for future use. Value 0x0112 is for the PNX8526 T-PI Bus controller module.
15:12	R	1	MAJREV[3:0]	Major Revision: any revision that might break software compatibility.
11:8	R	0	MINREV[3:0]	Minor Revision: any revision that maintains software compatibility.
7:0	R	0	MODULE_APERTURE_SIZE[7:0]	Aperture size = 4 kB*(bit_value+1), so 0 means 4 kB (the default).

Chapter 36: MMI Arbiter

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

36.1 Functional Description

36.1.1 Overview

The MMI Arbiter performs memory arbitration for the PNX8526. The arbiter receives 16 request signals from the internal modules (referred to as agents) desiring to access memory. Using an internal list, the arbiter selects which agent will be granted the bus and forwards the selected agent information to the memory controller. The list specifies the order and frequency that agents are allowed to request access to the memory subsystem.

The arbiter operates in one of three modes: Boot, Linear and “Buddy”.

36.1.2 Boot Mode

At power up, the arbiter is placed in boot mode. In this mode, the arbiter sequentially grants each agent access to the memory if the agent has asserted its request. This mode is not intended to intelligently allocate memory bandwidth—its goal is to make sure that the chip can boot up. The host processor is expected to initialize the list data to allow intelligent memory arbitration and set the arbiter to Linear or “Buddy” mode for better performance.

36.1.3 Linear Mode

After the internal list has been initialized, the arbiter can be placed in linear mode. In this mode, the arbiter sequentially examines each entry in the list and gives the agent specified in the list the opportunity to use the memory subsystem. If the agent does not require a memory access (its request is inactive), the agent is skipped and the next agent in the list is given a chance to access memory (See Section 36.1.3.1 on the next page for a slight exception to this rule.) When the end of the list is reached, the search starts again at the beginning of the list. The list contains up to 512 entries. The list must contain an even number of agents. The MaxAddr register specifies the last list address i.e., one-half the number of entries in the list -1. Thus, if the list has eight entries, the MaxAddr register would be programmed to 3.

When loading the list, each 32-bit write by the host processor loads two list entries. To allow future expansion, the data for the two list entries are in bits 3:0 and 11:8 as shown in Table 1.

Table 1: List Data, Linear Mode

List Address	List Data 11:8	List Data 3:0
0	Entry 1	Entry 0
1	Entry 3	Entry 2
2	Entry 5	Entry 4
3	Entry 7	Entry 6
…	…	…
255	Entry 511	Entry 510

36.1.3.1 Linear Mode Default Agents

In Linear Mode, up to two default agents can be enabled that will be granted access to memory if the current agent is not requesting access. If neither the current nor default agents needs to access memory, the next agent in the list is checked. If both default agents are enabled, the priority for the arbitration is:

List agent1st (highest) priority

DefaultAgent02nd priority

DefaultAgent13rd (lowest) priority

Thus, DefaultAgent1 will only be granted access if the list agent is inactive and DefaultAgent0 is either inactive or not enabled.

36.1.4 Buddy Mode

After the internal list has been initialized, the arbiter can be placed in buddy mode. In this mode, each entry in the list is used to specify two agents that are “buddies.” When an entry is read from the list, bits[3:0] specify the primary agent (most likely real time agent) that is given a chance to access memory. If the primary agent uses memory, the arbiter moves to the next entry in the list and the buddy agent does not get access to memory at this time. If the primary agent did not need the bus, the buddy agent specified in bits[11:8] of the entry is given a chance to access memory. If neither the primary or buddy agents require a memory access, the entry in the list is skipped and the arbiter moves on to the next entry. (See Section 36.1.4.1 for an exception to this rule.) The list can contain up to 256 entries. The MaxAddr register specifies the number of entries in the list.

Table 2: List Data, Buddy Mode

List Address	List Data 11:8	List Data 3:0
0	Buddy 0	Primary 0
1	Buddy 1	Primary 1
2	Buddy 2	Primary 2
3	Buddy 3	Primary 3
…	…	…
255	Buddy 255	Primary 255

36.1.4.1 Buddy Mode Default Agents

In Buddy Mode, up to two default agents can be enabled that will be granted access to memory if neither the current or buddy agents are requesting access. If neither the primary, buddy, or default agents needs to access memory, the next primary agent in the list is checked. If both default agents are enabled, the priority for the arbitration is as follows:

Primary agent1st (highest) priority

Buddy agent2nd priority

DefaultAgent03rd priority

DefaultAgent14th (lowest) priority

36.1.5 Hunting

In all operating modes, the arbiter sequentially examines agents based on the memory clock. In Linear mode, two entries of the list are examined on each clock cycle while in Buddy mode both the primary and buddy agent are examined in a single clock. Since arbitration is performed concurrently with memory operation, there is considerable overlap between servicing the previous agent by the memory controller and the selection of the next agent to be granted access to the memory bus. It is expected that very little time will be lost due to this overlap.

The arbiter is in hunt mode while it is sequentially processing the list to find the next active agent. The arbiter enters hunt mode almost immediately after granting an agent access to the memory. Because of the internal memory bus protocol, some care needs to be taken to avoid reissuing the bus to the current owner before the owner has a chance to remove its request signal. The arbiter can operate in one of two modes to avoid this situation:

- If DelayHunt is set (mode register, bit 5), the arbiter delays entering hunt mode for two clock cycles to allow the newly granted owner time to remove his request signal. This slows down arbitration slightly, but it allows the list to safely contain “back-to-back” entries for the same agent.
- If DelayHunt is 0, the arbiter masks the newly granted agent’s request signal for two clock cycles. This ensures that the agent has time to remove its request signal while allowing the arbiter to immediately begin hunting for the next active agent. The drawback to this mode is that “back-to-back” list entries for the same agent will not provide 2 contiguous chances to access memory. The “back-to-back” entries will only provide more chances for the agent to access memory.

In general, if the same agent is entered in the list in a “near back-to-back” fashion, then DelayHunt mode should probably be set. “Near back-to-back” is defined as within four entries of each other for Linear mode and two entries in Buddy mode.

Remark: Using the DelayHunt mode will enable back-to-back grants to a default agent if desired.

The list only indirectly controls the allocation of memory bandwidth. Since each entry in the arbitration list simply allows an agent access to the memory for one transaction, agents that consistently use 128-byte transactions will use more bandwidth than an agent that only uses an 8-byte transaction.

36.1.6 Agent IDs

The mapping of the PNX8526 memory agents to MMI Arbiter IDs is shown in Table 3.

Table 3: Agent IDs

Agent	ID	Agent	ID
ICP2	0	VMPGW – write	8
ICP1	1	FPIMI	9
MBS – read	2	TM32	A
MBS – write	3	MPIMI	B
MSP	4	TPIMI	C
VIP	5	D2D	D
VMPGL – read	6	MSP3	E
VMPGR – read	7	PCI	F

36.1.7 Dual Lists

To support real-time reprogramming of the arbitration table, the arbiter actually has two banks of list RAM. In normal operation, one bank is “active”—the list data is being read from this bank for allocating memory accesses. The “inactive” bank is not being used by the arbiter and may be accessed by the host processor without disturbing the arbitration process. When the “inactive” bank has been properly programmed, the software sets the “BankSwitch” bit in the Mode register to initiate a graceful transition between banks. The BankSwitch bit takes effect when the arbiter reaches the end of the current list. Once the hardware has changed banks, the BankSwitch bit is automatically cleared. Each bank has matching MaxAddr and DefaultAgent registers that must be programmed to the proper value before switching banks.

Regardless of the current mode (boot, linear, or buddy), changes to the list RAM, MaxAddr, and DefaultAgent registers by the host processor will not take effect until a bank switch has happened.

The Mode register is common to (i.e., shared with) both banks.

36.2 Register Summary and Descriptions

36.2.1 Clarifications

- Four 32-bit registers and a 256-location RAM control operation of the arbiter.
- The Mode register controls the basic operation. It is used to select operating mode (boot, linear, or buddy) as well as to report minor status conditions.
- The MaxAddr register specifies the value of the last valid entry in the list. This register is typically programmed to the last RAM address programmed divided by 4 (of course, truncated to 8 bits). For example:

```
// init the list ram
poke (0x0004D800, 00000102)
poke (0x0004D804, 00000304)
poke (0x0004D808, 00000506)
poke (0x0004D80C, 00000702)
poke (0x0004D810, 00000804)
poke (0x0004D814, 00000906)
poke (0x0004DC04, 00000005)
```

Remark: There are actually two MaxAddr registers that share a common programming interface—the “active” and “inactive” registers as described in Section 36.1.7.

- The DefaultAgent registers specify up to two agents that will be given a chance to use the bus if the current list entries do not require access. These registers also enable the respective default agent.
- There are two sets of DefaultAgent registers that share a common programming interface—the “active” and “inactive” registers as described in Section 36.1.7.
- The List is a 256 location RAM that contains arbitration entries. Since the physical RAM is 8 bits wide, in linear mode, each byte in the RAM contains two arbitration entries. In buddy mode, each byte of RAM specifies the primary and buddy agents. There are two sets of list RAMs that share a common programming interface—the “active” and “inactive” registers as described in Section 36.1.7.

The base address for the PNX8526 MMI Arbiter is 0x04 D800. These registers are part of the Global 2 Registers. For more information see the Register Summary List for PNX8526

Table 4: MMI Arbiter Register Summary (Global 2 Registers)

Address	Name	Descriptions
0x04 D800	RAM0	The lowest RAM
0x04 D804	RAM1	...
0x04 D808	RAM2	...
0x04 D80C	RAM3	...
0x04 D810	RAM4	...
...	...	...
0x04 DBFC	RAMFF	The highest RAM
0x04 DC00	Mode	This register controls the operating mode of the MMI Arbiter.
0x04 DC04	MaxAddr	This register specifies the last used RAM entry.
0x04 DC08	DefaultAgent0	This register specifies and enables the first (higher priority) default grant agent. This agent has higher priority than DefaultAgent1.
0x04 DC0C	DefaultAgent1	This register specifies and enables the second (lower priority) default grant agent. This agent has lower priority than DefaultAgent0.

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
MMI Arbiter Control Registers For more details on the RAM0—RAM255 registers see Section 36.1.2 on page 36-784.				
Offset 0x04 D800			*RAM0*	
31:12		-	Unused	Ignore during writes and read as zeroes.
11:8	R/W	NI	Entry1 or Buddy0	In Linear mode, this field = Entry 1 In Buddy mode, this field = Buddy 0
7:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	NI	Entry0 or Primary 0	In Linear mode, this field = Entry 0 In Buddy mode, this field = Primary 0
Offset 0x04 D804			*RAM1*	
31:12		-	Unused	Ignore during writes and read as zeroes.
11:8	R/W	NI	Entry3 or Buddy1	In Linear mode, this field = Entry 3 In Buddy mode, this field = Buddy 1
7:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	NI	Entry2 or Primary1	In Linear mode, this field = Entry 2 In Buddy mode, this field = Primary 1
Offset 0x04 D808			*RAM2*	
31:12		-	Unused	Ignore during writes and read as zeroes.
11:8	R/W	NI	Entry5 or Buddy2	In Linear mode, this field = Entry 5 In Buddy mode, this field = Buddy 2
7:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	NI	Entry4 or Primary2	In Linear mode, this field = Entry 4 In Buddy mode, this field = Primary 2
Offset 0x04 D80C			*RAM3*	
31:12		-	Unused	Ignore during writes and read as zeroes.
11:8	R/W	NI	Entry7 or Buddy3	In Linear mode, this field = Entry 7 In Buddy mode, this field = Buddy 3
7:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	NI	Entry6 or Primary3	In Linear mode, this field = Entry 6 In Buddy mode, this field = Primary 3
Offset 0x04 D810			*RAM4*	
⇓	⇓	⇓	⇓	⇓
Offset 0x04 DBFC			*RAM255*	
31:12		-	Unused	Ignore during writes and read as zeroes.
11:8	R/W	NI	Entry511 or Buddy 255	In Linear mode, this field = Entry 511 In Buddy mode, this field = Buddy 255
7:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	NI	Entry510 or Primary 255	In Linear mode, this field = Entry 510 In Buddy mode, this field = Primary 255

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x04 DC00			*MODE*	
31:10		-	Unused	Ignore during writes and read as zeroes.
9	R	0x0	ActiveBank	Indicates which RAM bank is active. When it is 1, bank A is active. Primarily for diagnostics.
8	R/W	0x0	BankSwitch	Writing a 1 to this register requests a BankSwitch. The hardware automatically clears this register when the BankSwitch is completed.
7		-	Unused	
6	R/W	0x0	delayhunt	When 1, hunting is delayed 2 clocks after the next agent is granted. When 0, the new owner's request signal is masked for 2 clocks after it is granted the bus.
5	R	0x0	dagent1_enabled	Indicates that default_agent1 is enabled in the active bank. Primarily for diagnostics.
4	R	0x0	dagent0_enabled	Indicates that default_agent0 is enabled in the active bank. Primarily for diagnostics.
3	R/W	0x0	buddy	When 1, buddy mode is enabled. When 0, linear mode is enabled. This bit does not become active until the next BankSwitch command is completed.
2	R	-	hunting	A 1 indicates the MMI Arbiter is hunting for the next requesting agent. Primarily for diagnostics.
1	R	0x0	boot_change	A 1 indicates that the Arbiter is changing to/from boot mode. Primarily for diagnostics.
0	R/W	1	boot	Writing a 0 to this bit turns off boot mode. When 1, the arbiter is in boot mode. When 0, the arbiter is in list mode.
Offset 0x04 DC04			*MAXADDR*	
31:8		-	Unused	Ignore during writes and read as zeroes.
7:0	R/W	0x00	MaxAddr	This register specifies the last valid RAM address that should be used by the arbiter.
Offset 0x04 DC08			*DEFAULTAGENT0*	
31:8		-	Unused	Ignore during writes and read as zeroes.
7	R/W	0x0	da0_enable	This bit will enable default agent 0 when the bank is made active.
6:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	0x0	dagent0	These bits specify the higher priority default agent.
Offset 0x04 DC0C			*DEFAULTAGENT1*	
31:8		-	Unused	Ignore during writes and read as zeroes.

GLOBAL 2 REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
7	R/W	0x0	da1_enable	This bit will enable default agent 1 when the bank is made active.
6:4		-	Unused	Ignore during writes and read as zeroes.
3:0	R/W	0x0	dagent1	These bits specify the lower priority default agent.

Chapter 37: Boundary-Scan Hardware

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

37.1 Introduction

The PNX8526 is equipped with logic supporting the Boundary-Scan Architecture standard, specified in the IEEE 1149.1 standard. This chapter gives a brief introduction to the architecture and application of this standard.

The IEEE 1149.1 (JTAG) standard can be used for various goals, including testing connections between integrated circuits, testing the internal structures of integrated circuits, and monitoring and communicating with a running system.

Boundary-Scan hardware is comprised of on-chip test logic, five dedicated pins, called the Test Access Port (TAP), and a TAP controller.

The JTAG standard defines some mandatory instructions for a TAP controller and allows for user-defined and private instructions. The PNX8526 provides user-defined and private instructions for hardware and software debug and for production testing.

For software debug, there is communication between a debug monitor running on the TriMedia CPU and a debugger front-end, running on a host computer. Refer to Chapter 39 TM32 JTAG Software Debug Port, for more information.

37.2 Overview

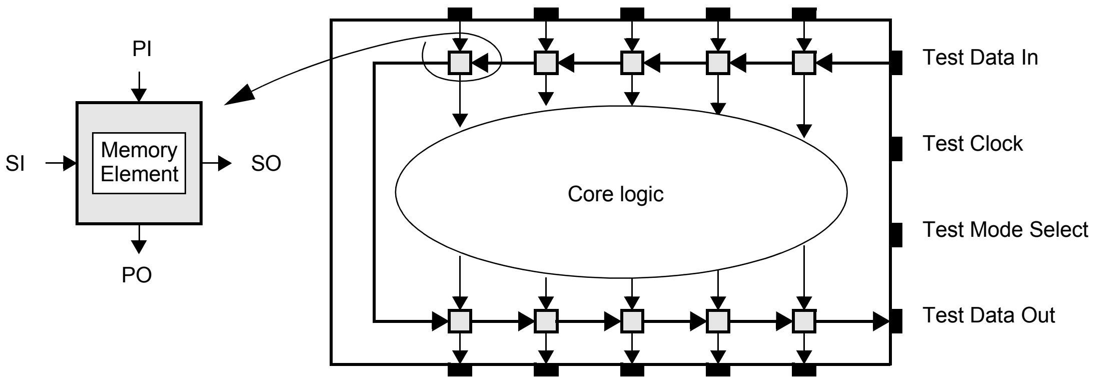


Figure 1: Boundary-Scan Architecture

All primary input and output signals have an associated boundary-scan cell. The boundary-scan cell is a memory element that has several functions:

- Captures data on its parallel input (PI)
- Updates data on its parallel output (PO)

- Serially scans data to its neighbor (SI to SO).

Functionally, the boundary-scan cells behave transparently (PI to PO).

The set of boundary-scan cells are configured into a parallel I/O shift register. A load operation samples the signal value on all pins (capture state). The unload operation updates the boundary-scan cell values to pass to the core logic (update state).

The Test Access Port is comprised of five pins. The Test Data In (TDI) pin accepts test data, which can be shifted serially around the register until reaching the Test Data Out (TDO) pin. The Test Mode Select (TMS) signal controls the mode of operation. And the Test Clock (TCK) provides the clock. The Test Reset (TRST) pin provides an asynchronous reset of the controller.

37.2.1 Testing

Boundary-Scan can be used to test an individual device and the connections between devices:

- Testing the functionality of a single device is called “internal test,” or Intest. This allows the operator to check if a device actually exists and identify gross defects.
- Testing the interconnect structure between two devices is called “external test,” or Extest. Using Extest to search for shorts and opens on a printed circuit board (PCB) is the major application of Boundary-Scan.

Figure 2 shows a sample PCB containing four boundary-scan devices. Notice that there is a connector input called TDI, connected to the TDI of the first device. TDO of the first device is then connected to the TDI of the second device and so on, creating a global scan path terminating at the connector output called TDO. TCK is connected to each device TCK input, TMS works similarly.

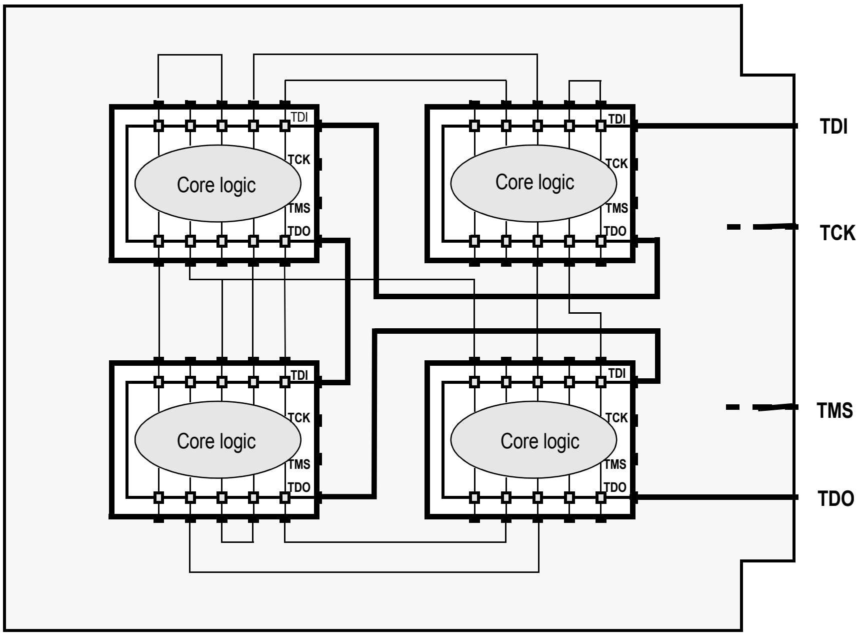


Figure 2: Using the Boundary-Scan Path on a PCB

37.2.2 IEEE 1149.1 Boundary-Scan Standard

The standardized Boundary-Scan device architecture, as described in IEEE standard 1149.1, is shown in Figure 3. At any time, only one register can be connected from TDI to TDO (e.g., IR, Bypass, Boundary-Scan, Ident or even internal device registers). The selected register is identified by the decoded output of the IR. Certain instructions are mandatory, such as Extest, whereas others are optional, such as the Idcode instruction.

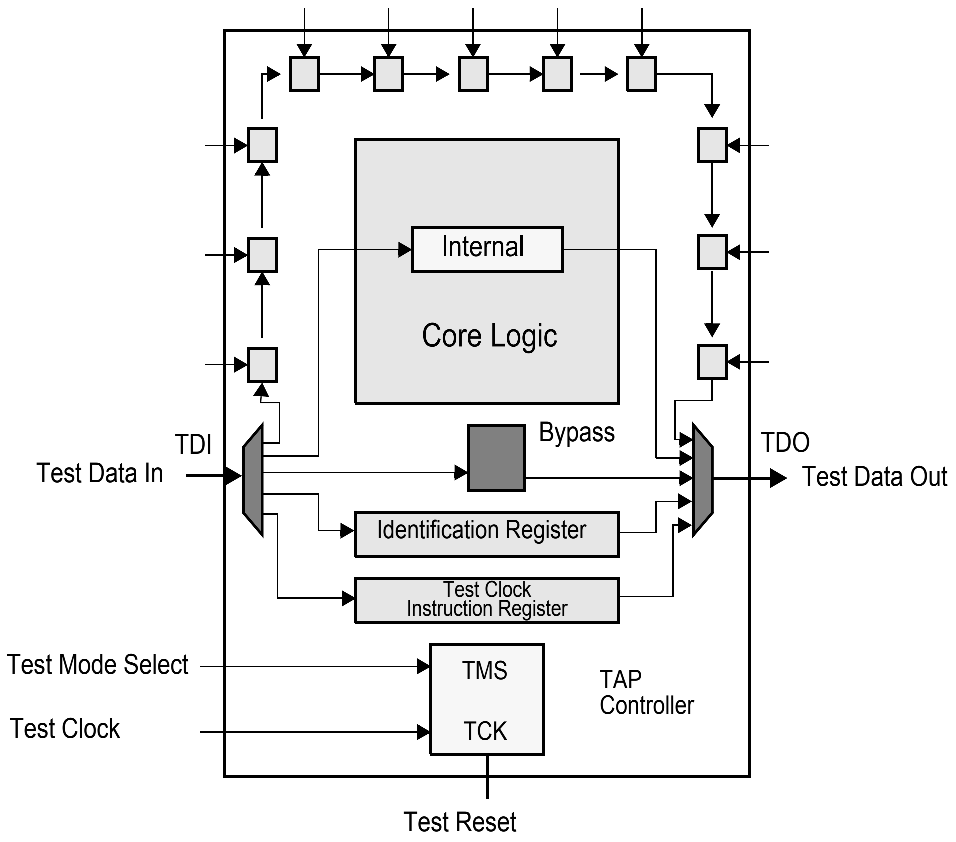


Figure 3: IEEE 1149.1 Chip Architecture

37.2.2.1 Test Access Port (TAP)

The Test Access port has four dedicated input pins and one output pin:

- TCK (Test Clock)
- TMS (Test Mode Select)
- TDI (Test Data In)
- TRST (Test Reset)
- TDO (Test Data Out)

TCK provides the clock for the test logic. TCK is asynchronous to any system clock. Stored state devices in the JTAG controller retain their state indefinitely when TCK is stopped at 0 or 1.

The signal received at TMS is decoded by the TAP controller to control test functions. The test logic is required to sample TMS at the rising edge of TCK.

Serial test instructions and test data are received at TDI. The signal is sampled at the rising edge of TCK. When test data is shifted from TDI to TDO, the data must appear without inversion at TDO after a set number of rising and falling edges of TCK. The number is determined by the length of the instruction or test data register selected.

TDO is the serial output for test instructions and data from the TAP controller. Changes in the state of TDO must occur at the falling edge of TCK. This is because devices connected to TDO must sample TDO at the rising edge of TCK. The TDO driver must be in an inactive state (i.e., TDO line HIghZ) except when scanning data.

37.2.2.2 TAP Controller

The TAP controller is a finite state machine. It synchronously responds to changes in TCK and TMS signals. The TAP instructions and data are serially scanned into the TAP controller's instruction and data registers via the common input line TDI. The TMS signal tells the TAP controller to select either the TAP instruction register or a TAP data register as the destination for serial input from the common line TDI. An instruction scanned into the instruction register selects a data register to be connected between TDI and TDO and hence to be the destination for serial data input.

The TAP controller's state changes are determined by the TMS signal. The states are used for scanning in/out TAP instruction and data, updating instruction and data registers, and for executing instructions.

The controller's state diagram (Figure 4) shows separate states for "Capture," "Shift" and "Update" of data and instructions. This leaves the contents of a data or instruction register undisturbed until serial "scan in" is finished and the data is ready for Update. By separating Shift and Update states, a register's contents (the parallel stage) are not affected during scan in/out.

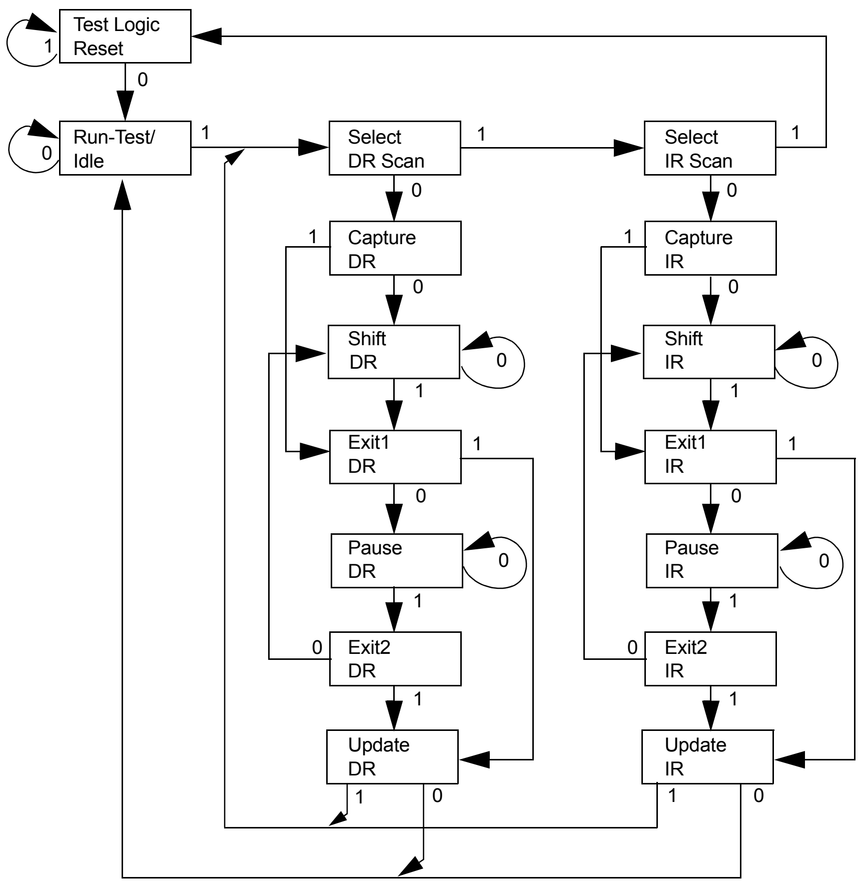


Figure 4: State Diagram of TAP Controller

The TAP controller must be in Test Logic Reset state after power up. It remains in that state as long as TMS is held at 1. The controller transitions to Run-Test/Idle state from Test Logic Reset state when TMS = 0. The Run-Test/Idle state is an idle state between scanning an instruction/data register in and out. The “Run-Test” part of the name refers to the start of built-in tests. The “Idle” part of the name refers to all other cases.

Remark: There are two similar substructures in the state diagram, one for scanning in an instruction and another for scanning in data. To scan in/out a data register, it is necessary to scan in an instruction first.

An instruction or data register must have at least two stages, the shift register stage and the parallel input/output stage. When an n-bit data register is to be read, the process is as follows:

1. The register is selected by an instruction.
2. The register’s contents are “captured” (loaded in parallel into shift register stage).
3. n bits are shifted in and at the same time n bits are shifted out.
4. The register is “updated” with the new n bits shifted in.

Remark: When a register is scanned, its old value is shifted out of TDO and the new value, shifted in via TDI, is written to the register at the Update state. Hence, scan in/out involves the same steps. This also means that reading a register via JTAG destroys its contents (unless otherwise stated).

Some registers are specified as read-only via JTAG so that when the controller transitions to the Update state for the read-only register, the update has no effect. Some times, it is necessary to have read/write registers (for example, control registers used for handshake) that must be read non-destructively. In these cases, the value shifted in determines whether the old value is "remembered."

37.2.2.3 Boundary-Scan Register

Each logic input and output signal pin on the device is supplemented with a boundary-scan cell. The total set of boundary-scan cells are configured as a parallel-in, parallel-out shift register.

37.2.2.4 JTAG Public Instruction Set

The PNX8526 uses a 5-bit JTAG instruction register. Some of these instructions are listed in Table 1. The unspecified opcodes are private, and their effects are undefined. The JTAG instructions EXTEST, SAMPLE/PRELOAD, BYPASS and IDCODE are standard instructions. The other instructions are discussed in Chapter 39 TM32 JTAG Software Debug Port

.

Table 1: JTAG Instruction Encoding

Encoding	Instruction Name	Action
00000	EXTEST	Select boundary-scan register
00001	SAMPLE/PRELOAD	Select boundary-scan register
11111	BYPASS	Select bypass register
00010	IDCODE	Select ID code register
TriMedia Debug		
10000	TM_DBG_RESET	Reset TriMedia to power on state
10001	TM_DBG_SEL_DATA_IN	Select DATA_IN register
10010	TM_DBG_SEL_DATA_OUT	Select DATA_OUT register
10011	TM_DBG_SEL_IFULL_IN	Select IFULL_IN register
10100	TM_DBG_SEL_OFULL_OUT	Select OFULL_OUT register
10101	TM_DBG_SEL_JTAG_CTL1	Select JTAG_CTRL register 1
10110	TM_DBG_SEL_JTAG_CTL2	Select JTAG_CTRL register 2
Other	Reserved	

37.2.3 Boundary-Scan Description Language

The Boundary-Scan Description Language (BSDL) provides a standard machine and human-readable data format for describing how IEEE 1149.1 is implemented in a device. The BSDL description for the PNX8526 is provided in a data file.

Many IEEE 1149.1 tools on the market support BSDL as a data input format. These tools offer different capabilities to engineers implementing IEEE 1149.1 in their designs, including board interconnect Automatic Test Pattern Generation (ATPG) and Automatic Test Equipment (ATE).

37.2.4 More Information

The following documents provide more information on boundary-scan:

IEEE Standard 1149.1-1990, “Test Access Port and Boundary-Scan Architecture,” IEEE, 1990/1993/1995.

B. Bennetts, “Boundary-Scan Tutorial,” ASSET Intertech Inc.

H. Bleeker et.al., “Boundary-Scan Test: A Practical Approach,” Kluwer Academic Publishers, 1993.

C. Maunder, R. Tullos, “Test Access Port and Boundary-Scan Architecture,” IEEE Computer Society Press, 1992.

K. Parker, “The Boundary-Scan Handbook,” Kluwer Academic Publishers, 1992.

Chapter 38: MIPS EJTAG Software Debug Port

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

38.1 Introduction

The MIPS Enhanced JTAG (EJTAG) module incorporates DMA support from the JTAG pins to the PI-bus and support for probe memory access by software running on the processor via the JTAG pins. EJTAG is an industry standard. It gives the host computer total control over the target system IC (e.g., to perform reset or debug break, to observe the status of clocks in the system, etc.)

Through the EJTAG PI-bus master functionality, the host computer is able to perform read/write access to all PI-bus slave agents (e.g., for observation of status and control of the DSU or for software download to memory).

Through the EJTAG PI-bus slave functionality a processor on the PI-bus can access program or data memory within the host computer.

In a special operation mode, the JTAG TDO line outputs serial real-time trace information reflecting the instruction flow of the CPU. That information is further extended by 3-bit state information supplied directly by the DSU.

The Enhanced JTAG (EJTAG) module has the following features:

- Compliant with IEEE 1149.1, EJTAG 2.0.0, and PI-Bus V3.3 standards
- PI-Bus master/slave functionality
- Acts as companion to the Debug Support Unit in the PR3940 MIPS Processor

38.1.1 Using EJTAG

Instructions for using EJTAG are available in the MIPS EJTAG Debug Solution document.

38.2 Functional Description

The EJTAG consists of two separate modules. One is the EJTAG module, and the other is the Debug Support Unit (DSU). The EJTAG module contains the JTAG data registers, probe memory access logic and DMA logic. The DSU contains the breakpoint registers, comparators and tracing logic. The DSU is located inside the PR3940 MIPS processor.

The maximum EJTAG clock (TCK) is 40 MHz.

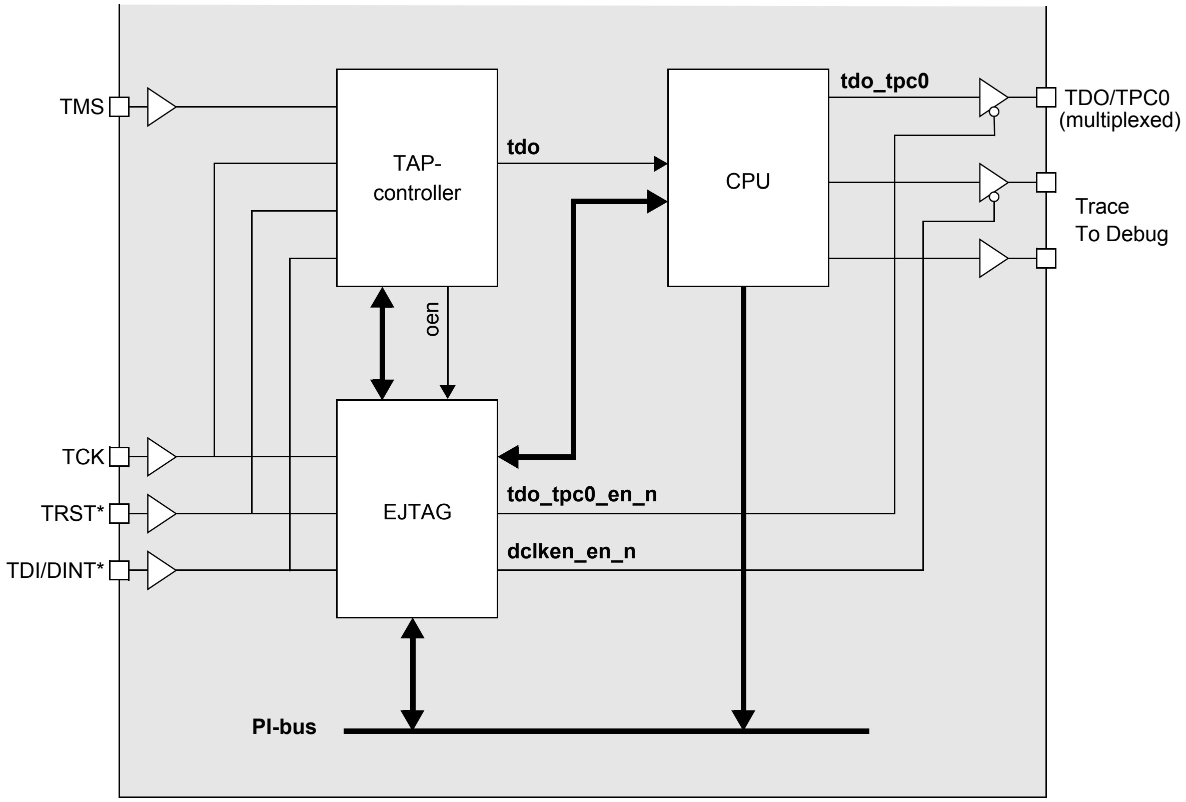


Figure 1: Placement of the EJTAG Module

In addition to the standard IEEE 1149.1 JTAG TAP controller functionality, the EJTAG Debug Interface provides a means of communication between a development system host computer and the target system IC utilizing the 4-5 wire JTAG bus.

A block diagram of the EJTAG core is depicted in Figure 2. The core is divided into several functional blocks, which communicate by means of point-to-point channels.

The core operates on two functional clocks: the fast PI-bus clock pi_clk and the JTAG clock tck. In Figure 2 the two clock domains are separated by a dotted line. The functional blocks that lie in the area above this dotted line are clocked with the JTAG clock tck. The blocks that lie in the area beneath the dotted line are synchronous with the PI-bus clock pi_clk. The blocks that lie on the dotted line synchronize the communication between the two clock domains and operate on both clocks.

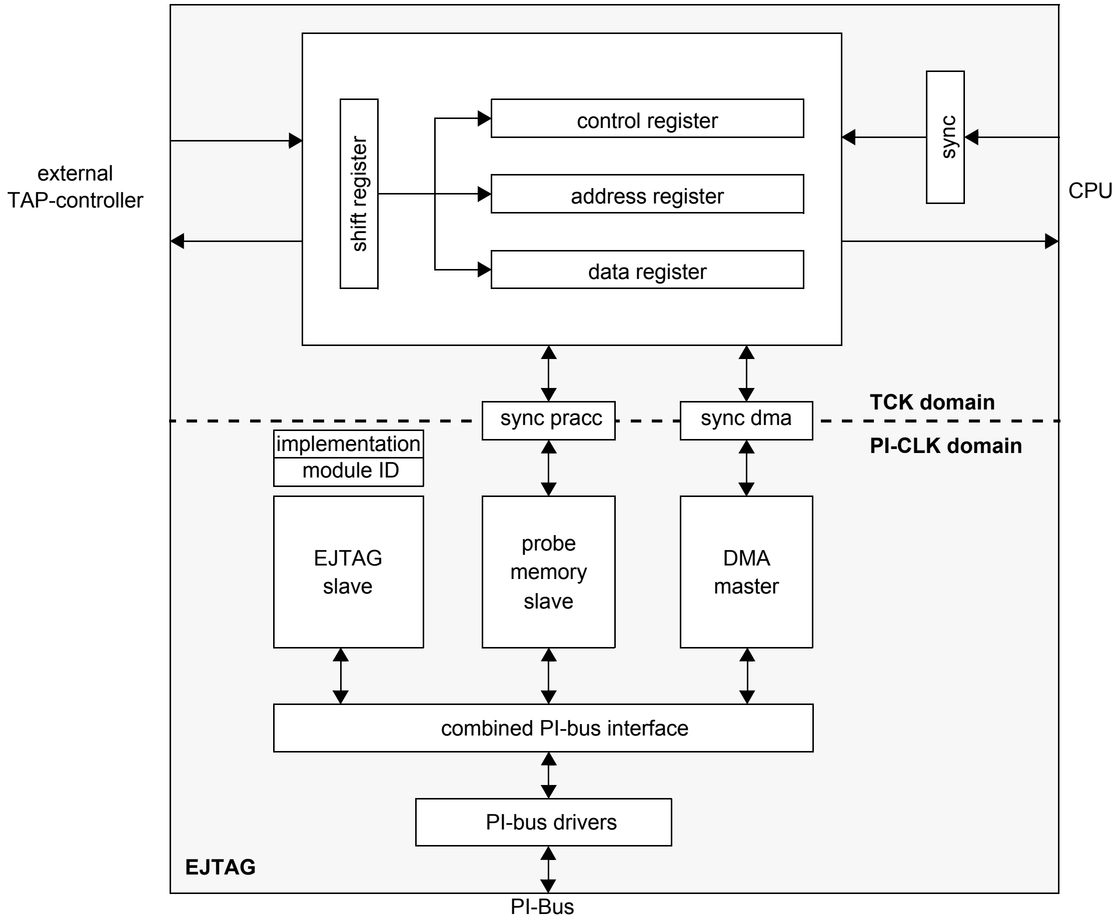


Figure 2: Simplified Block Diagram of the EJTAG Module

The functional blocks of the EJTAG are as follows:

- DMA master—handles a DMA. DMAs are initiated by the probe.
- Probe memory slave—handles a probe memory access. Probe memory accesses are initiated by masters on the PI-bus.
- EJTAG slave—allows reading out of the implementation and module identification register via the PI-bus.
- Combined PI-bus interface—merge the PI-bus control of the DMA master, probe memory slave and EJTAG slave.
- PI-bus drivers—the PI-bus tri-state drivers.
- Shift register—96-bit wide register in which selected register is captured and shifted in/out.
- Control register—the EJTAG control register.

- Address register—the address register that holds the address that is shifted in through tdi. This register supplies the address to the DMA master. The address that is shifted out is supplied either by the address register in the DMA master or by the address register in the probe memory slave. One of the two sources is selected by the DmaAcc bit in the control register.
- Data register—the data register that holds the data that is shifted in through tdi. This register supplies the write data to the DMA master and the read data to the probe memory slave. The data that is shifted out through tdo is supplied either by the data register in the DMA master or by the data register in the probe memory slave. One of two sources is selected by the DmaAcc bit in the control register.

38.3 External Pin Descriptions

The following table lists the EJTAG pins. Refer to Chapter 1 Signal Descriptions and Chapter 2 Multi-Function Pins of the PNX8526 Data Sheet for more information.

Table 1: EJTAG Contacts

EJTAG v2.5 Name	PNX8525 Name	Type (I/O)	Description
TDI	DBG_TDI	I	PR3940 Debug Port Data In
TDO/TPC	DBG_TDO	O	PR3940 Debug Port Data Out
TCK	DBG_CLK	I	PR3940 Debug Port Clock
TMS	DBG_TMS	I	PR3940 Debug Port Mode Select
TRST	JTAG_TRST	I	JTAG Port Reset (shared)

Table 2: Mapping of MIPS EJTAG Signals onto Chip Output Pins

EJTAG v2.5 Name	PNX8525 Signal	Output Pin**	Function
TCP[2]	DSU_TPC1	DV_OUT2[8]	Debug Support Unit 1, TPC1
Not Used	DSU_TPC0	DV_OUT2[7]	Debug Support Unit 0, TPC0
PCST2[2]	DSU_PCST1[2]	DV_OUT2[6]	Program Counter Status1, Bit 2
PCST2[1]	DSU_PCST1[1]	DV_OUT2[5]	Program Counter Status1, Bit 1
PCST2[0]	DSU_PCST1[0]	DV_OUT2[4]	Program Counter Status1, Bit 0
PCST1[2]	DSU_PCST0[2]	DV_OUT2[3]	Program Counter Status0, Bit 2
PCST1[1]	DSU_PCST0[1]	DV_OUT2[2]	Program Counter Status0, Bit 1
PCST1[0]	DSU_PCST0[0]	DV_OUT2[1]	Program Counter Status0, Bit 0
DCLK	DSU_CLK	DV_OUT2[0]	Debug Support Unit Clock
** These Output Pins can also be used as SPY outputs.			

38.4 Register Descriptions

Table 3: MIPS E-JTAG Register Summary

Address	Name	Description
0x03 D000	Implementation Register	
0x03 DFFC	Module ID	

MIPS E-JTAG SOFTWARE DEBUG PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x03 D000			*Implementation Register*	
31:26	R		reserved	
25:24	R		ProfSup	PROFILING SUPPORT: Defines if profiling is supported. 00 No profiling support. 01 Simple DMA profiling supported. 10,11 Reserved.
23	R		SDBBPCode	SPECIAL SDBBP CODE 0: SDBBP is encoded according to 1.3 specification 1: SDBBP is encoded using a Special2 Opcode
22:21	R		ASIDsize	SIZE OF ASID IN IMPLEMENTATION: Defines the size of the ASID in the implementation: 00 No ASID in implementation. 01 6 bit ASID. 10 8 bit ASID. 11 Reserved.
20	R		CplxBrk	COMPLEX BREAK SUPPORT_ This bit is set to ‘1’ when Complex Break is supported, and otherwise set to ‘0’.
19	R		PhysAW	PHYSICAL ADDRESS WIDTH Informs the size of EJTAG_Address_register 0: Physical addresses are 32-bit in length 1: Physical addresses is from 33 to 64-bits in length The exact length of can be determined by shifting a pattern through the EJTAG Address Register.
18	R		DCacheC	DATA CACHE COHERENCY 0: Data Cache does not keep coherency with DMA 1: Data Cache keeps coherency with DMA
17	R		ICacheC	INSTRUCTION CACHE COHERENCY 0: Instruction Cache does not keep DMA coherency 1: Instruction Cache keeps coherency with DMA
16	R		MIPS16	MIPS 16 SUPPORT 0: MIPS CPU does not support MIPS16 1: MIPS CPU supports MIPS16

MIPS E-JTAG SOFTWARE DEBUG PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
15	R		NoPCTrace	PC TRACE SUPPORT 0: PC Trace is supported by implementation 1: PC Trace is not supported by implementation
14	R		NoDMA	DMA SUPPORT 0: EJTAG DMA is supported by implementation 1: EJTAG DMA is not supported by implementation
13:11	R		TPCW	TPC WIDTH 000,111 1 bit 000 is Standard EJTAG 001 2 bits Extended EJTAG 010 4 bits Extended EJTAG 011 8 bits Extended EJTAG others reserved Extended EJTAG
10:8	R		PCSTW	PCST WIDTH AND TELEVISION FACTOR 000,111 3 bits (DCLK is 1/1 of MIPS CPU CLK) 001 6 bits (DCLK is 1/2 of MIPS CPU CLK) 010 9 bits (DCLK is 1/3 of MIPS CPU CLK) 011 12 bits (DCLK is 1/4 of MIPS CPU CLK) others reserved Note: 000 is for standard EJTAG, the other values for the Extended EJTAG.
7	R		NoProcBrk	PROCESSOR BUS BREAK: this bit indicates if the Processor Bus Break function is implemented in the DSU. 0: Processor Bus Break is implemented 1: Processor Bus Break is not implemented
6	R		NoDataBrk	DATA ADDRESS BREAK: this bit indicates if the Data Address Break function is implemented in the DSU. 0: Data Address Break is implemented 1: Data Address Break is not implemented
5	R		NoInstBrk	INSTRUCTION ADDRESS BREAK: this bit indicates if the Instruction Address Break function is implemented in the DSU. 0: Instruction Address Break is implemented 1: Instruction Address Break is not implemented

MIPS E-JTAG SOFTWARE DEBUG PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
4:1	R		Ch[3:0]	OBSOLETE FIELD: debug SW should check InstrBrk, DataBrk and ProcBrk to know what break types are implemented. NUMBER OF BREAK CHANNELS: these 4 bits used to in-dicate the number of break channels implemented in the DSU. 0000 = no break channels 0001 = 1 break channel ... 1111 = 15 break channels
0	R		MIPS32/64	INHIBITOR__ indicates the type of MIPS CPU, informing the width of the MIPS CPU datapath, the de-bug registers implemented in the DSU, and the EJTAG_Data_register. 0: 32-bit wide data registers 1: 64-bit wide data registers
Offset 0x03 DFFC			*Module ID*	
31:16	R	0x0104	MODULE_ID	Module ID Number
15:12	R	0	REV_MAJOR	Major Revision
11:8	R	0	REV_MINOR	Minor Revision
7:0	R	0	APP_SIZE	Apperture Size is 0 = 4 kB.

Chapter 39: TM32 JTAG Software Debug Port

Programmable Source Decoder with Integrated Peripherals

Rev. 01 — 8 October 2003

39.1 Functional Description

The IEEE 1149.1 (JTAG) Standard can be used for various goals including testing connections between integrated circuits on the board level, control over testing the internal structures of the integrated circuits, and monitoring and communicating with a running system.

The JTAG standard defines on-chip test logic, four or five dedicated pins, collectively called the Test Access Port (TAP), and a TAP controller.

In order to guarantee correct behavior on the board level, the JTAG standard defines instructions that must always be implemented by a TAP controller. Apart from mandatory instructions, the standard also allows user defined and private instructions. In the PNX8526, user defined and private instructions exist for hardware and software debug purposes and for production testing. For software debug, there is communication between a debug monitor running on the TriMedia CPU and a debugger front-end running on a host computer. This is explained in Section 39.1.2.

39.1.1 Overview

The TriMedia debug interface consists of the TAP, TAP Controller, JTAG Instruction register and the internal debug registers.

39.1.1.1 Test Access Port (TAP)

The Test Access Port includes four dedicated input pins and one output pin:

- TCK (Test Clock)
- TMS (Test Mode Select)
- TDI (Test Data In)
- TRST (Test Reset)
- TDO (Test Data Out)

TCK provides the clock for test logic required by the JTAG standard. TCK is asynchronous to any system clock. Stored state devices in the JTAG controller will retain their state indefinitely when TCK is stopped at 0 or 1.

The signal received at TMS is decoded by the TAP controller to control test functions. The test logic is required to sample TMS at the rising edge of TCK.

Serial test instructions and test data are received at TDI. The TDI signal is required to be sampled at the rising edge of TCK. When test data is shifted from TDI to TDO, the data must appear without inversion at TDO after a number of rising and falling edges of TCK determined by the length of the instruction or test data register selected.

TDO is the serial output for test instructions and data from the TAP controller. Changes in the state of TDO must occur at the falling edge of TCK. This is because devices connected to TDO are required to sample TDO at the rising edge of TCK. The TDO driver must be in an inactive state (i.e., TDO line High Z) except when the scanning of data is in progress.

39.1.1.2 TAP Controller

The TAP controller is a finite state machine that synchronously responds to changes in TCK and TMS signals. The TAP instructions and data are serially scanned into the TAP controller's instruction and data registers via the common input line TDI. The TMS signal tells the TAP controller to select either the TAP instruction register or a TAP data registers as the destination for serial input from the common line TDI. An instruction scanned into the instruction register selects a data register to be connected between TDI and TDO to become the destination for serial data input.

The TAP controller's state changes are determined by the TMS signal. The states are used for scanning in/out TAP instruction and data, updating instruction, and data registers, and for executing instructions.

The controller's state diagram (Figure 1) shows separate states for "Capture," "Shift" and "Update" of data and instructions in order to leave the contents of a data register or an instruction register undisturbed until serial scan in is finished and the Update state is entered. By separating the Shift and Update states, the contents of a register (the parallel stage) is not affected during scan in/out.

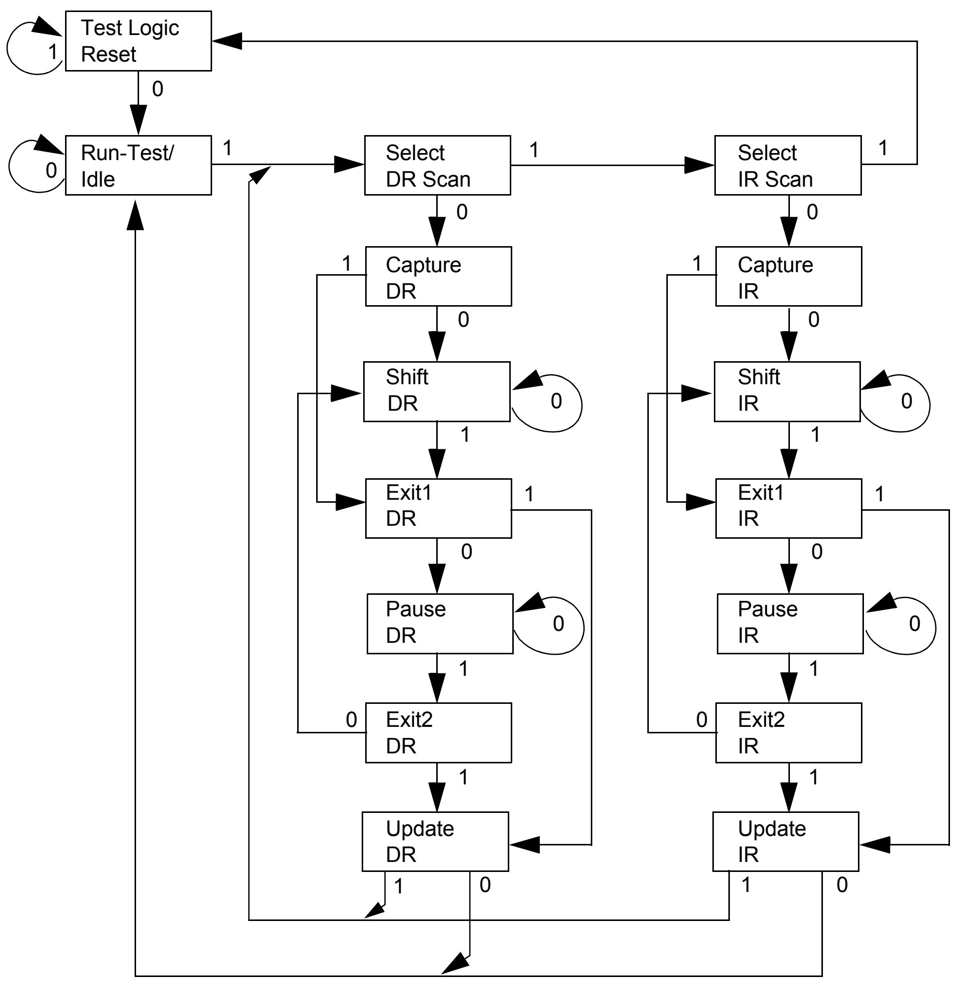


Figure 1: State Diagram of TAP Controller

The TAP controller must be in Test Logic Reset state after power up. It remains in that state as long as TMS is held at 1. The controller transitions to Run-Test/Idle state from Test Logic Reset state when TMS = 0. The Run-Test/Idle state is an idle state of the controller in between scanning in/out an instruction/data register. The “Run-Test” part of the name refers to start of built-in tests. The “Idle” part of the name refers to all other cases.

Remark: There are two similar substructures in the state diagram, one for scanning in an instruction and another for scanning in data. To scan in/out a data register, one has to scan in an instruction first.

An instruction or data register must have at least two stages, the shift register stage and the parallel input/output stage. When an n-bit data register is to be “read,” the register is selected by an instruction. The register’s contents are “captured” first (loaded in parallel into the shift register stage), n bits are shifted in and simultaneously, n bits are shifted out. Finally the register is “updated” with the new n bits shifted in.

Remark: When a register is scanned, its old value is shifted out of TDO and the new value shifted in via TDI is written to the register at the Update state. Hence, scan in/out involve the same steps. This also means that reading a register via JTAG destroys its contents unless otherwise stated.

We can specify some registers as read-only via JTAG so that when the controller transitions to the Update state for the read-only register, the update has no effect. Some times, we need read/write registers (for example, control registers used for handshake) which must be read non-destructively. In such cases, the value shifted in determines whether the old value is “remembered” or something else happens.

39.1.1.3 PNX8526 JTAG Instruction Set

PNX8526 uses a 5-bit JTAG instruction register. The unspecified opcodes are private and their effects are undefined. Table 1 lists the public JTAG instructions.

Table 1: JTAG Instruction Encoding

Encoding	Instruction name	Action
00000	EXTEST	Select boundary-scan register
00001	SAMPLE/PRELOAD	Select boundary-scan register
11111	BYPASS	Select bypass register
00010	IDCODE	Select ID code register
10000	TM_DBG_RESET	Reset TriMedia to power on state
10001	TM_DBG_SEL_DATA_IN	Select DATA_IN register
10010	TM_DBG_SEL_DATA_OUT	Select DATA_OUT register
10011	TM_DBG_SEL_IFULL_IN	Select IFULL_IN register
10100	TM_DBG_SEL_OFULL_OUT	Select OFULL_OUT register
10101	TM_DBG_SEL_JTAG_CTL1	Select JTAG_CTRL register 1
10110	TM_DBG_SEL_JTAG_CTL2	Select JTAG_CTRL register 2

The JTAG instructions EXTEST, SAMPLE/PRELOAD, BYPASS and IDCODE are standard instructions and are not discussed here. All other instructions are discussed in Section 39.2.

39.1.2 Operation

Figure 2 shows an overview of the JTAG access path from a host machine to a target PNX8526 system and a simplified block diagram of the PNX8526 processor. The JTAG Interface Module, shown separately in the diagram, may be a PC add-on card or a similar module connected to a PC serial or parallel port. The JTAG interface module is necessary for PNX8526 systems that are not plugged into a PC. For PC-hosted PNX8526 systems, the host based TriMedia debugger front-end can communicate with the target resident debug monitor via the PCI bus.

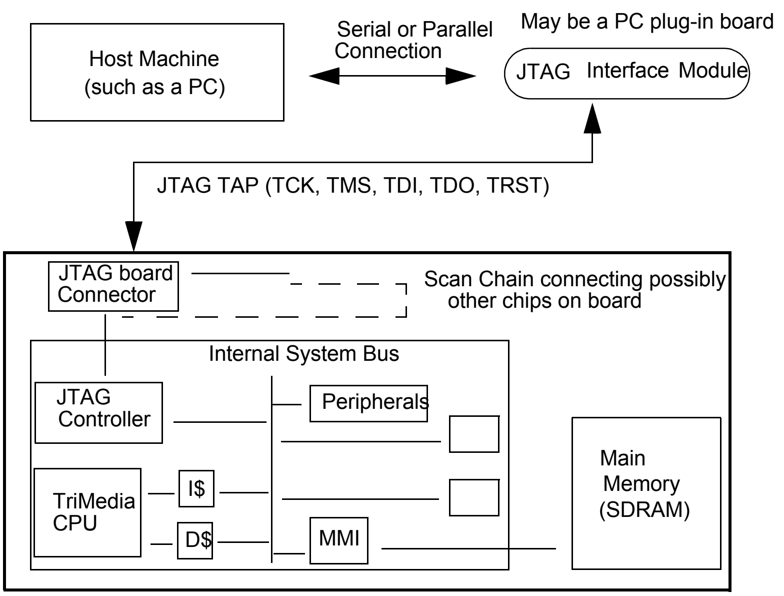


Figure 2: System with JTAG Access

Enhancements to the standard JTAG functionality include a handshake mechanism for transferring data to and from a PNX8526 processor's MMIO registers, support for posting an interrupt, and resetting processor state.

The actual interpretation of the contents of the MMIO registers is determined by a software protocol used by the debug monitor running on the internal TriMedia CPU and the debug front-end running on a host machine.

The communication between a host computer and a target system via JTAG requires the following major components:

1. A Host computer with a serial or parallel interface

   The host computer transfers data to and from the JTAG interface module, preferably in word-parallel fashion. Also needed is JTAG interface device driver software to access and modify the registers of the JTAG interface module.

2. A JTAG interface module (hardware) that asynchronously transfers data to and from the host computer

   The interface module synchronously transfers data to and from the JTAG TAP on a PNX8526 processor, supplies the test clock TCK and other signals to the JTAG controller on TriMedia. The interface module may be a PC plug-in board.

   This module may transfer data from and to the host computer in bit-serial or word-parallel fashion. It transfers data from and to the JTAG registers on the PNX8526 processor in bit-serial fashion in accordance with the IEEE 1149.1 Standard. The JTAG interface module connects to a 4 or 5-pin JTAG connector on the PNX8526 board which provides a path to the JTAG pins on the PNX8526 processor. It is the responsibility of the interface module to scan data in and out of the PNX8526 processor into its internal buffers and make them available to the host computer.

3. A JTAG controller on the PNX8526 processor which provides a bridge between the external JTAG TAP and the internal system.

   The controller transfers data from/to the TAP to/from its scannable registers asynchronous to the internal system clock. A monitor running on the internal TriMedia CPU and the debugger front-end running on a host computer exchange data via JTAG by reading/writing the MMIO registers reserved for this purpose, including two control registers used for handshaking.

39.2 Programming Information

Because the JTAG data registers live in MMIO space and are accessible by both the TriMedia CPU and the JTAG controller at the same time, race conditions must not exist either in hardware or in software. The communication protocol uses a handshake mechanism to avoid software race conditions.

39.2.1 Handshaking and Communication Protocol

The following describes the mechanism for transferring data via JTAG.

39.2.1.1 Transfer from Debug Front-End to Debug Monitor

The debugger front-end running on a host transfers data to a debug monitor via the JTAG_DATA_IN register. It must poll the JTAG_CTRL2.ifull bit to check if the JTAG_DATA_IN register can be written to. If the JTAG_CTRL2.ifull bit is clear, the front-end may scan data into the JTAG_DATA_IFULL_IN register.

Remark: Data and control bits may be shifted in with SEL_IFULL_IN instruction and the bit shifted into JTAG_CTRL.ifull register must be 1. This action triggers an interrupt. The debug monitor must copy the data from JTAG_DATA_IN register into its private area when servicing the interrupt and then clear the JTAG_CTRL2.ifull bit. This allows the JTAG interface module to write the next piece of data to the JTAG_DATA_IN register.

39.2.1.2 Transfer from Monitor to Front-End

The monitor running on TriMedia must check if JTAG_CTRL1.ofull is clear and if so, it can write data to JTAG_DATA_OUT. After that, the monitor must set the JTAG_CTRL1.ofull bit. The debugger front-end polls the JTAG_CTRL1.ofull bit. When set, it can scan out the JTAG_DATA_OUT register and clear the JTAG_CTRL1.ofull bit. Since JTAG_DATA_OUT is read-only via JTAG, the update action at the end of scan out has no effect on JTAG_DATA_ OUT. The JTAG_CTRL1.ofull bit however, must be cleared by shifting in the value 1.

39.2.1.3 Controller States

In the power on reset state, JTAG_CTRL2.ifull and JTAG_CTRL1.ofull must be cleared by the JTAG controller.

39.2.1.4 Example Data Transfer Via JTAG

Scanning in a 5-bit instruction will take 12 TCK cycles from the Run-Test/Idle state: 4 cycles to reach Shift-IR state, 5 cycles for actual shifting in, 1 cycle to exit1-IR state, 1 cycle to update-IR state, and 1 cycle back to Run-Test/Idle state. Likewise, scanning in a 32-bit data register will take 38 TCK cycles, and transferring an 8-bit JTAG_CTRL data register will take 14 TCK cycles from Idle state. However, if a data transfer follows instruction transfer, then the transition to DR scan stage can be done without going through Idle state, thereby saving 1 cycle.

39.2.1.5 Transfer of Data to TriMedia Via JTAG

Poll control register to check if input buffer is empty or not and scan in data when it is empty and set the ifull control bit to 1 triggering an interrupt.

Remark: Scanning in any instruction automatically scans out the 3 least significant bits (including the ifull or ofull bit) of the selected JTAG_CTRL register.

Table 2: Transfer of Data In via JTAG

Action	Number of TCK Cycles
IR shift in SEL_IFULL_IN instruction	12
While JTAG_CTRL2.ifull = 1, scan in SEL_IFULL_IN instruction	11+
DR scan 33 bits of register JTAG_IFULL_IN	38
TOTAL	61+ cycles

39.3 Register Descriptions

The PNX8526 has two JTAG data registers and two JTAG control registers (see Figure 3) in MMIO space and a number of JTAG instructions to manipulate those registers. Table 3 lists the MMIO addresses of the JTAG data and control registers. The addresses are offsets from the MMIO_BASE. All references to instruction and data registers below refer to JTAG instructions and data registers only (not TriMedia instruction or data registers)

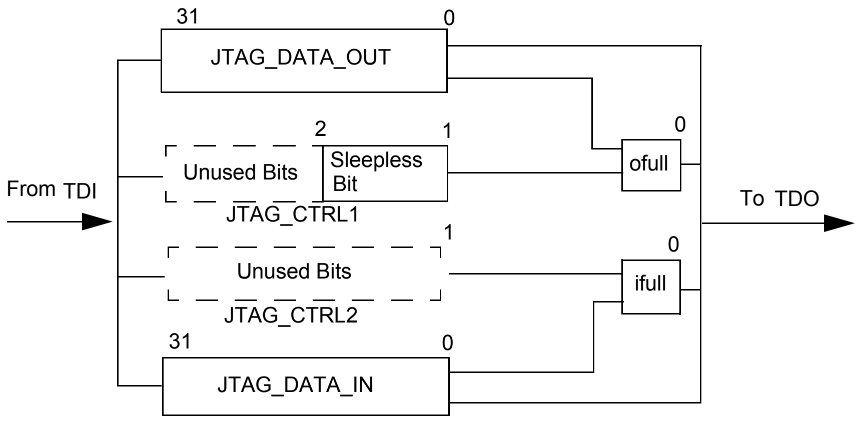


Figure 3: Additional JTAG Data and Control Registers

Data Registers

There are two 32-bit data registers, JTAG_DATA_IN and JTAG_DATA_OUT in the MMIO space. Both can be connected in between TDI and TDO like the standard Bypass and Boundary-Scan registers of JTAG (not shown in Figure 3).

- The JTAG_DATA_IN register can be read or written to via the JTAG port. S
- The JTAG_DATA_OUT register is read-only via the JTAG port, so that scanning out JTAG_DATA_OUT is non-destructive.
- The JTAG_DATA_IN and JTAG_DATA_OUT are readable/writable from the TriMedia processor via the usual load/store operations.

Control Registers

There are two control registers, JTAG_CTRL1 and JTAG_CTRL2, in MMIO space.

- The JTAG_CTRL registers are used for handshake between a debug monitor running on a TriMedia CPU and a debugger front-end running on a host.
- JTAG_CTRL1.ofull = 1 means that JTAG_DATA_OUT has valid data to be scanned out. On the power-on reset of the PNX8526, JTAG_CTRL1.ofull = 0. JTAG_CTRL1.ofull is both readable and writable via the JTAG tap. Writing 0 to JTAG_CTRL1.ofull via JTAG is a “remember” operation i.e., JTAG_CTRL1.ofull retains its previous state. Writing 1 to JTAG_CTRL1.ofull via JTAG is a ‘clear’ operation i.e., JTAG_CTRL1.ofull becomes 0.
- JTAG_CTRL2.ifull = 0 means that the JTAG_DATA_IN register is empty. JTAG_CTRL2.ifull = 1 means that JTAG_DATA_IN has valid data and the debug monitor has not yet copied it to its private area. Upon power on reset of the TriMedia processor, JTAG_CTRL2.ifull = 0. JTAG_CTRL2.ifull is readable and writable via JTAG. Writing 0 to JTAG_CTRL2.ifull via JTAG is a ‘remember’ operation i.e., JTAG_CTRL2.ifull retains it previous state. Writing 1 to JTAG_CTRL2.ifull posts a TriMedia interrupt on hardware line 49.
- The peripheral blocks on a PNX8526 processor may enter a “powerdown” state to reduce power consumption. The JTAG_CTRL1.sleepless bit determines if the JTAG block participates in a powerdown state. In the power on RESET state, JTAG_CTRL1.sleepless bit is 1 meaning the JTAG block does not powerdown. It can be read and written to by the PNX8526 processors via load/store operations and by the debugger front-end running on a host by scan in/out.

JTAG Virtual Registers

There are two virtual registers, JTAG_IFULL_IN and JTAG_OFULL_OUT:

- The first virtual register JTAG_IFULL_IN connects the registers JTAG_CTRL2.ifull and JTAG_DATA_IN in series. Likewise, the virtual register JTAG_OFULL_OUT connects JTAG_CTRL1.ofull and JTAG_DATA_OUT in series.
- The reason for the virtual registers is to shorten the time for scanning the JTAG_DATA_IN and JTAG_DATA_OUT registers. Without virtual registers, one must scan in an instruction to select JTAG_DATA_IN, scan in data, scan an instruction to select JTAG_CTRL register and finally scan in the control register. With virtual register, one can scan in an instruction to select JTAG_IFULL_IN and then scan in both control and data bits. Similar savings can be achieved for scan out using virtual registers.

JTAG Instructions

There are six JTAG Instructions:

- Six instructions SEL_DATA_IN, SEL_DATA_OUT, SEL_IFULL_IN, SEL_OFULL_OUT, SEL_JTAG_CTRL1 and SEL_JTAG_CTRL2 for selecting the registers to be connected between TDI and TDO for serial input/output.
- An instruction RESET for resetting the TriMedia CPU to power on state.
- In the capture-IR state of the TAP controller, the 2 least significant bits (bits 0 and 1) of the shift register stage must be loaded with the ‘01’ as required in the standard. The standard allows the remaining bits of the IR shift stage to be loaded with design specific data. The bits 2, 3 and 4 of the IR shift stage are loaded with bits 0, 1 and 2 of the JTAG_CTRL register. This means that shifting in any instruction allows the 3 least significant bits of the JTAG_CTRL register to be inspected. This reduces the polling overhead for data transfer.

The base address of the PNX8526 TM32 JTAG SW Debug interface is 0x06 1000.

Table 3: TM32 JTAG SW Debug Register Summary

Offset	Name	Description
0x06 1000	TM_DBG_DATA_IN	Input register for data coming from the JTAG
0x06 1004	TM_DBG_DATA_OUT	Output register for data going to the JTAG
0x06 1008	TM_DBG_CTRL1	Control register 1 for output data
0x06 100C	TM_DBG_CTRL2	Control register 2 for input data
0x06 1FE0	INT_ST	Interrupt Status Register
0x06 1FE4	INT_EN	Interrupt Enable Register
0x06 1FE8	INT_CLR	Interrupt Clear Register
0x06 1FEC	INT_SET	Interrupt Set Register
0x06 1FF0	Reserved	
0x06 1FF4	POWER_DOWN	Powerdown Register
0x06 1FF8	Reserved	
0x06 1FFC	Module ID	Module Identification Register

TM32 JTAG SOFTWARE DEBUG PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
TriMedia JTAG Debug Registers				
Offset 0x06 1000			*JTAG_DATA_IN*	
31:0	R/W	0	JTAG_DATA_IN[31:0]	JTAG debugger input data
Offset 0x06 1004			*JTAG_DATA_OUT*	
31:0	R/W	0	JTAG_DATA_OUT[31:0]	JTAG debugger output data

TM32 JTAG SOFTWARE DEBUG PORT REGISTERS				
Bits	Read/ Write	Reset Value	Name (Field or Function)	Description
Offset 0x06 1008			*JTAG_CTRL1*	
31:2		-	Unused	
1	R/W	0	sleepless	Set bit to prevent JTAG debug module from going into powerdown.
0	R/W	0	ofull	JTAG output data available handshake bit
Offset 0x06 100C			*JTAG_CTRL2*	
31:1		-	Unused	
0	R/W	0	ifull	JTAG input data available handshake bit
Offset 0x06 1FE0			*Interrupt Status Register*	
31:1		-	Unused	
0	R	0	INTR_STATUS	A logic “1” indicates JTAG interrupt detected.
Offset 0x06 1FE4			*Interrupt Enable Register*	
31:1		-	Unused	
0	R/W	0	INTR_EN	A logic “1” written to a specific bit location will enable the corresponding interrupt in the Interrupt Status register.
Offset 0x06 1FE8			*Interrupt Clear Register*	
31:1		-	Unused	
0	W	0	INTR_CLR	A logic “1” written to a specific bit location will clear the corresponding interrupt in the Interrupt Status register.This bit is self-clearing.
Offset 0x06 1FEC			*Interrupt Set Register*	
31:1		-	Unused	
0	W	0	INTR_SET	A logic “1” written to a specific bit location will set the corresponding interrupt in the Interrupt Status register.
Offset 0x06 1FF4			*Powerdown Register*	
31	R/W	0	POWER_DOWN	JTAG Powerdown indicator 1 = Powerdown 0 = Power up When this bit equals 1, no other registers are accessible.
30:0		-	Unused	
Offset 0x06 1FFC			*Module ID Register*	
31:16	R	0x0127	MOD_ID	JTAG Module ID Number
15:12	R	0	REV_MAJOR	Major revision
11:8	R	0	REV_MINOR	Minor revision
7:0	R	0	APP_SIZE	Aperture size is 0 = 4 kB.

Index

40. Disclaimers

Life support – These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes – Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status ‘Production’), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

41. Licenses

Purchase of Philips I^{2}C components

Purchase of Philips I^{2}C components conveys a license under the Philips’ I^{2}C patent to use the components in the I^{2}C system provided the system conforms to the I^{2}C specification defined by Philips. This specification can be ordered using the code 9398 393 40011.

42. Trademarks

Nexperia – is a trademark of Koninklijke Philips Electronics N.V.

43. Contact information

For additional information, please visit **http://www.semiconductors.philips.com.**

For sales office addresses, send an email to: **sales.addresses@www.semiconductors.philips.com.**

Date of release: 8 October 2003
Document order number: UM10104_1