Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	Multimedia
Core Processor	TM3260
Program Memory Type	-
Controller Series	Nexperia
RAM Size	-
Interface	I²C, 2-Wire Serial
Number of I/O	61
Voltage - Supply	1.14V ~ 1.26V
Operating Temperature	0°C ~ 85°C
Mounting Type	Surface Mount
Package / Case	456-BGA
Supplier Device Package	456-BGA (27x27)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/pnx1501e-557

Chapter 6: Boot Module

Chapter 7: PCI-XIO Module

Chapter 8: General Purpose Input Output Pins

Chapter 21: MPEG-1 and MPEG-2 Variable Length Decoder

Chapter 22: Digital Video Disc Descrambler

Chapter 23: LAN100 — Ethernet Media Access Controller

6. DC/AC I/O Characteristics

The characteristics listed in the following tables apply to the worst case operating condition defined in Section 5. on page 1-24. All voltages are referenced to VSS (0 V digital ground). The following I/O characteristics includes the effect of process variation.

6.3 BPX2T14MCP Type I/O Circuit

Table 23: BPX2T14MCP Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{OH}	Output High Voltage		$0.9V_{CCP}$			V
V_{OL}	Output Low Voltage				$0.1V_{CCP}$	V
V_{IHT}	DC Input High Voltage	Logic Threshold	2.0			V
V_{ILT}	DC Input Low Voltage	Logic Threshold			0.8	V
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			V_{CCP} + 0.3	V
V_{IL}	DC Input Low Voltage		-0.3			V
Z_O	Output AC Impedance	High/Low level output state		22		Ω
Pull	Pull-up/down Resistor	If applicable	38	66	165	KΩ
C_{IN}	Input pin capacitance				6	pF

BPX2T14MCP I/Os require a board level 27-33 Ω series resistor to reduce ringing.

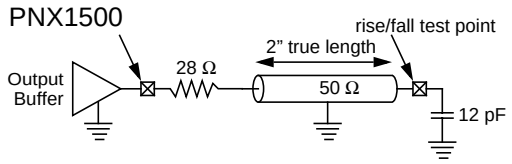

Figure 4: BPX2T14MCP Test Load Condition

5. References

[1] “PNX1300 Series Media Processor”, Feb. 15th 2002, Philips Semiconductors, Inc.

[2] “PCI Local Bus Specification, Rev 2.2”, Dec. 18th, 1998, PCI Special Interest Group.

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 Document title variable. 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.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
Offset 0x04 0820	***GPXIO_address***			
31:0	gpxio_addr	R/W	0	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.
Offset 0x04 0824	***GPXIO_write_data***			
31:0	gpxio_wdata	R/W	0	General Purpose XIO cycle data. This register is programmed with data for a write cycle.
Offset 0x04 0828	***GPXIO_read_data***			
31:0	gpxio_rdata	R	0	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.				
31:10	Reserved	R	0	
9	gpxio_cyc_pending	R	0	1 = GPXIO transaction on XIO is pending. 0 = GPXIO has completed or not yet started.
8	gpxio_done	R	0	General Purpose XIO cycle complete. This bit is cleared by writing 1 to bit 6 or 7. It will also be cleared by writing to the GPXIO interrupt clear register.
7	clr_gpxio_done	W	0	1 = Clear “gpxio_done.”
6	gpxio_init	R/W	0	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	Reserved	R	0	
4	gpxio_rd	R/W	0	1 = Read command on XIO 0 = Write command on XIO
3:0	gpxio_ben	R/W	0	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***			
31:22	Reserved			
21:16	nand_ctrls	R/W	17	This field controls the type of NAND-Flash access cycle. The bits are defined as follows: [21]: 1= 64-MB device support; 0 = 32 MB and smaller device support [20]: 1 = Include data in access cycle; 0 access does not include data phase(s) [19:18] = No. of commands to be used in NAND-Flash access [17:16] = No. of address phases to be used in NAND-Flash access. For 64-MB devices, 11 provide four address phases and 10 provide three address phases.
15:8	command_b	R/W	0	This is the second command for NAND-Flash when two commands are required to complete a cycle.
7:0	command_a	R/W	0	This is the command type to be used with NAND-Flash cycles when one or more commands are required to complete a cycle.

valid data and by putting their base addresses in the two BASEx_PTR registers, their maximum size into the SIZE register and the number of valid words for DMA buffer 1 into the PG_BUF_CTRLx.BUF_LEN bits.

When the FIFO queue is programmed into Pattern Generation mode, i.e. FIFO_MODE[1]=1, 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 FIFO queue has been enabled the BUF1_RDY flag must be cleared by software so that the GPIO module can load the BASE1_PTR and the PG_BUF_CTRLx.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_PTR and BUF_LEN values for DMA buffer 2 are loaded by the GPIO modules.

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 and transmitting the samples/timestamps on the selected GPIO pins. Once DMA buffer 1 is empty, 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_PTRx, the number of valid words in the new buffer by setting PG_BUF_CTRLx.BUF_LEN and then clear BUF1_RDY (write a ‘1’ to BUF1_RDY_CLR) before DMA buffer 2 fills up to avoid an Underrun condition.Transmission continues from buffer 2, until it is empty. At that time, BUF2_RDY is asserted, and transmission continues from the new buffer 1, and so on. If an Underrun condition is reached the GPIO module stops the transmission, holds current values on the pins and does not warn the CPU that an underrun condition occurred.

Remark: The BASEx_PTRx and PG_BUF_CTRLx.BUF_LEN values for a DMA buffer are only loaded into the GPIO pattern generation logic when the relevant BUFx_RDY signal has been cleared. Since the PG_BUF_CTRLx.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_PTRx and BUF_LEN values should be stable before software clears BUFx_RDY.

Remark: The DMA buffer sizes must be a multiple of 64 bytes. 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 pin when the reference timestamp clock reaches this value.

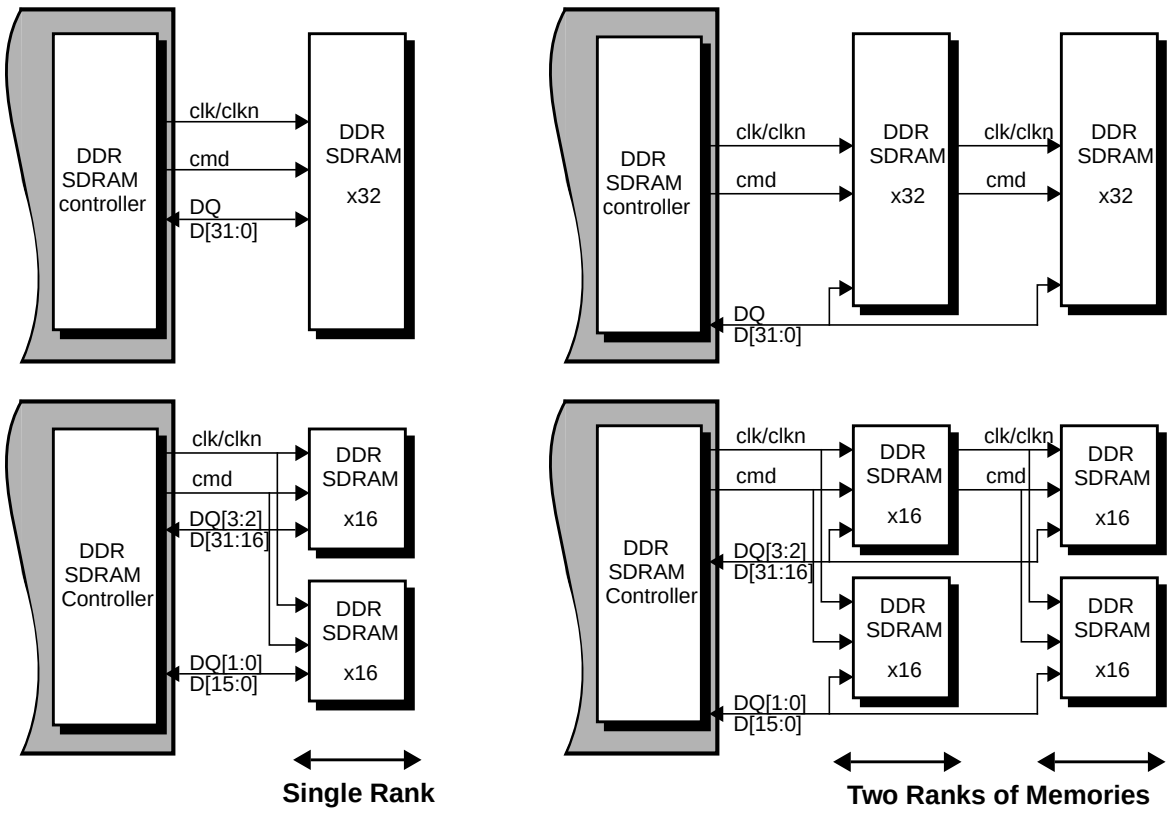

Figure 8: Examples of Supported Memory Configurations

3.2 Error Signaling

The MMIO port does not support error signaling. Reads from invalid addresses return the value “0”, writes to invalid addresses are ignored. The errors are not reported at system level.

Changing MMIO registers of an initiated DDR SDRAM Controller may cause incorrect behavior. Forcing the DDR controller into halt mode, programming MMIO registers while in halt mode, then unhalting the DDR controller when the MMIO registers have been programmed is the suggested series of actions to take.

3.3 Latency

The DDR SDRAM Controller uses two pipeline stages to calculate the command(s) that will be issued to the DDR memories after a MTL command is accepted by the DDR controller.

We will describe the latency of a MTL read command. Assume we have a MTL read command on one of the MTL ports in cycle 0 which is accepted by the DDR controller. In cycle 1, the DDR controller will determine the first DDR burst for the MTL read command. In cycle 2, the DDR SDRAM Controller will determine the DDR commands that need to be sent out on the DDR interface (we assume we do not have

The outputs of all mixers are connected to the back-end part of the QVCP --- the output formatter. The output formatter performs all necessary functions to adopt the final composited image to the display requirements. Among the functions performed in the output formatter are: gamma correction, chrominance downsampling, output formatting, and VBI insertion.

2.3 Layer Resources and Functions

This section focuses on the elements comprising a layer. Note that all of the described modules are present in each layer exactly once, the justification being that they (elements) are either always needed for the basic operation of a layer or they are so small (in design size) that assigning them to the resource pool would be inefficient due to the multiplexing and routing overhead associated with the pool elements.

2.3.1 Memory Access Control (DMA CTRL)

QVCP has 4 DMA agents, each of which connects to a 512-Byte buffer in the DMA adapter. DMA agents 1-2 are hard coded to layer1-2 respectively. DMA4 is used to fetch a VBI packet or a data packet for DMA-based control-register programming. DMA3 can be assigned to any of the two layers for supporting the semi-planar input format.

For video data fetches, the request block size is equal to the initial layer width (before horizontal scaling). If start_fetch is disabled (i.e., Enable bit 31 of register 0x10E2C8 is programmed to zero), the first DMA request starts right after the layer_enable is asserted and QVCP works as if prefetch is enabled. However, if start_fetch is enabled (i.e., Enable bit 31 of register 0x10E2C8 is programmed to one), then the DMA starts fetching only when QVCP s internal line counter reaches the 12-bit line threshold programmed in the Fetch Start bits [11:0] of register 0x10E2C8. Data fetched for the first field (interlaced) or frame (progressive) is not used and is flushed at the FCU (Flow Control Unit) FIFO. Thereafter, the pixels for the second field/frame start marching into the FCU FIFO, waiting for the correct layer position. The FCU FIFO releases pixels only if the x,y coordinates generated by the Screen Timing Generator (STG) match the layer position. In case of an interlaced output, the field ID is also checked. The DMA fetch request for the next active video line starts as soon as the last active pixel of the current line moves from the adapter FIFO into the processing pipeline and this request must be served in time to guarantee that the first active pixel of this new line is ready at the FCU FIFO before the STG signals the start of active video for the new line.

The DMA-based register-control programming only needs to be done once for a particular display scenario; thus, DMA4 is mainly used for VBI data fetch. QVCP is designed such that a VBI packet will only be inserted in the horizontal blanking interval and only one VBI packet is allowed in any one horizontal blanking interval. To insert this packet, there are two DMA requests. The first request has a block size of 1 since it is used to fetch only the header (which contains the size information). The second request is meant to fetch actual data of the required size and so, the maximum DMA request size (for the second request) is equal the length of horizontal blanking interval. The VBI data for the current horizontal blanking interval is always fetched in advance and stored in the DMA buffer (in the adapter). After sending out this prefetched data, the VBI DMA control unit (DMA4) requests a prefetch of the next packet (and correct operation requires that the sequence of the next two fetches must

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
8	PF_10B_MODE	R/W	0	10-bit Input format modes 0: 8-bit input format mode 1: 10-bit input format mode
7:0	PF_IPFMT	R/W	0	Input Formats 08 (hex) = YUV 4:2:2 semi-planar 24 (hex) = 1-bit indexed[Note1] 45 (hex) = 2-bit indexed[Note1] 66 (hex) = 4-bit indexed[Note1] 87 (hex) = 8-bit indexed[Note1] A0 (hex) = packed YUY2 4:2:2 A1 (hex) = packed UYVY 4:2:2 AC (hex) = 16 bits variable contents 4:4:4 CC (hex) = 24 bits variable contents 4:4:4 E8 (hex) = 32 bits variable contents 4:2:2 EC (hex) = 32 bits variable contents 4:4:4 **Note1:** For indexed modes Variable format register should be set 'E7E7E7E7' **Note 2**: Only YUV 4:2:2 semi-plana format (08) and packed formats (A0 & A1) are available when PF_10B_MODE is on
Offset 0x10 E2C0	***Layer Background Color***			
31	BG_enable	R/W	0	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	Upper	R/W	0	Upper channel of the background color (R/Y) (two's complement)
15:8	Middle	R/W	0	Middle channel of the background color (G/U) (two's complement)
7:0	Lower	R/W	0	Lower channel of the background color (B/V) (two's complement)
Offset 0x10 E2C4	***Variable Format register***			
31:29	PF_SIZE_A[2:0]	R/W	0	Size component for alpha Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
28:24	PF_OFFS_A[4:0]	R/W	0	Offset component for alpha Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.
23:21	PF_SIZE_L[2:0]	R/W	0	Size component for V or B Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
20:16	PF_OFFS_L[4:0]	R/W	0	Offset component for V or B Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.
15:13	PF_SIZE_M[2:0]	R/W	0	Size component for U or G Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
12:8	PF_OFFS_M[4:0]	R/W	0	Offset component for U or G Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.

Enabling the Dither Units

Immediately after the Dither unit is enabled or after a reset, the unit waits for the beginning of a newly-captured image. Only then the unit starts dithering.

Once the Dither unit is operational (enabled), it keeps track of the order in which the images arrive: we refer to the very first image at the unit dither as the *even* image, the second image as the *odd* image, and so on. A frame here is defined as two images: an even image followed by an odd image. This maintained state does not depend on the selected alternation options, it is maintained as long as the Dither unit is enabled. Any alternative activity corresponds to the internally-maintained state of a frame and field (even or odd) and has nothing to do if the signal is coming from the top or the bottom field.

Dithering operation also distinguishes between even and odd pixel-components of the same type (either Y, U or V) in a line.

The first occurrence of a Y or U or V component in the first line in the first received image is considered to be an **even** occurrence (or *set*).

2.5.5 Horizontal Video Filters (Sampling, Scaling, Color Space Conversion)

Interpolation Filter (Upsampling)

All horizontal video processing is based on equidistant sampled components. All 4:2:2 video streams, therefore, have to be upsampled before being scaled horizontally. The interpolation FIR filter used can interpolate interspersed or co-sited chroma samples. Mirroring of samples at the field boundaries compensate for run-in and run-out conditions of the filter.

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 (Downsampling)

After horizontal processing, the chrominance may be down-sampled to reduce memory bandwidth or allow a higher-quality vertical processing not available otherwise. Mirroring of samples at the field boundaries compensate for run in and run out conditions of the filter.

The following coefficients are used:

- co-sited: low pass A=(1,2,1)/4 or sub-sample A=(0,1,0)
- interspersed: B=(-3,19,19,-3)/32

Normal Polyphase Filter (Horizontal Scaling)

The normal polyphase filter can be used to zoom up (upscale) or downscale a video image. Depending on the number of components, the filter is used with 6 taps (three-component mode) or 3 taps (four-component mode).

Capture Enable Mode

Using the *cfen* bits, video capture can be limited to odd or even or both fields. If both fields are to be captured, the capture starts with the next odd field.

The status of the *osm* (one-shot) bit in the mode-control-register specifies the capture mode (one-shot or continuous):

- If *osm*=0, the corresponding incoming video stream is captured continuously. For example, in a video conference application the vanity image would be a continuous stream to the frame buffer.
- If *osm*=1, the corresponding incoming video stream is captured one field or frame at a time (depending on the *cfen* bits).

Programming hint: In a video conference application the captured image would be a one-shot stream to the host memory. If you write *osm*=1 and select field/frame in the register, it is captured on the next VSYNC and *cfen* bits are cleared to 0. To capture the next image, the *cfen and 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 every new line.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.

Double Buffer Mode

To avoid line tear caused by trying to display a frame at the same time that it is being updated, a double buffer mode is available. In this double buffer mode, a second set of DMA base addresses is available. After capturing and storing one complete frame in the location described by one set, the other set is used for the next frame. The idea is illustrated in Figure 11.

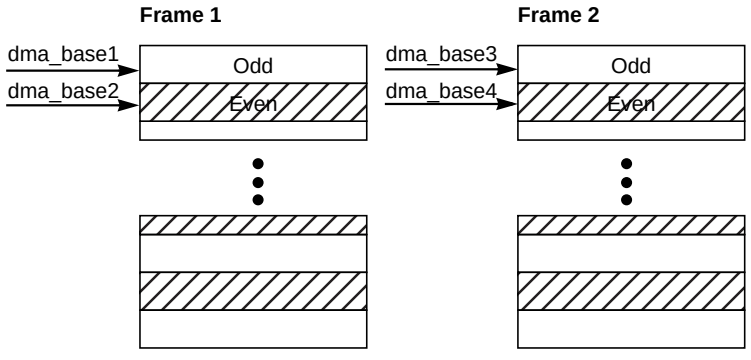

Figure 11: Double Buffer Mode

If the alpha value, after horizontal and vertical processing, is below a fixed threshold (0x80), the sample is replaced by the color key.

- **Fixed Alpha Insert**

  The alpha value defined in the Color Key register is inserted, as the fourth component after horizontal and vertical processing, in the PSU unit (if CKEY_K2A=1).

2.4.7 Alpha Processing

Alpha processing is only available when the horizontal and vertical filter blocks are either bypassed or operated in the four-component mode. If the horizontal filter is used for color space conversion, the alpha information is kept time-aligned with the other three components.

2.4.8 Video Data Output

After processing the video stream, the MBS can split the video data into one or more (up 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 then get buffered and transferred to the external memory in more effective clusters. A list of the supported output video formats is shown in Table 6. 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 Endian settings (according to the global Endian setting, the Endian bit in the output format register can invert the setting).

Table 6: 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 plane #2 plane #3		Y8 or R8 U8 or G8 V8 or B8
semi planar YUV (4:2:2 or 4:2:0)	plane #1 plane #2		Y8 or R8 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
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

Remark: Table 6 shows location of the first 'pixel unit' within a 64 bit word in the little endian mode. The selected endian mode will affect the position of the components within multi-byte pixel units!

Table 8: Memory Based Scaler (MBS) Registers *…Continued*

Bit	Symbol	Access	Value	Description
29:28	CKEY_A2K	R/W	0	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	CKEY_REPLACE	R/W	0	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	Reserved			
23: 16	CKEY_MASK1	R/W	0	Color key mask component 1 Defines bits of color component that are compared against color key value setting to key sample.
15: 8	CKEY_MASK2	R/W	0	Color key mask component 2 Defines bits of color component that are compared against color key value setting to key sample.
7: 0	CKEY_MASK3	R/W	0	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	CKEY_ALPHA	R/W	0	Alpha value Defines the alpha value to be used for keyed samples.
23: 16	CKEY_COMP1	R/W	0	Color key component 1 Defines value of color key for component 1 (red or Y).
15: 8	CKEY_COMP2	R/W	0	Color key component 2 Defines value of color key for component 2 (green or U).
7: 0	CKEY_COMP3	R/W	0	Color key component 3 Defines value of color key for component 3 (blue or V).
Video Output Format Control Registers				
Offset 0x10 C300	***Video Output Format***			
31:30	PSU_BAMODE	R/W	0	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	Reserved			
13	PSU_ENDIAN	R/W	0	Output format endian mode 0: same as system endian mode 1: opposite of system endian mode
12	Reserved			

Raster-ops are turned off if alpha blending is enabled.

2.3.2 Alpha Blending

The Drawing Engine supports alpha blending of source and destination data. The destination data can be either 32-bit αRGB:8888, 32-bit αYUV8888, 16-bit RGB:565, 16-bit aRGB 4444, or 16-bit RGBa 4534. Source data can come either from the memory or from the host data port. The alpha calculations are based on the source and "surface" alpha values.

Raster-ops are disabled if alpha blending is selected.

The following combinations of source and destination data are supported.

Table 1: Source and Destination Data

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
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

2.3.3 Source Data Location and Type

The data for the source operand can reside in either the frame buffer memory or be provided by the host processor via the MMIO bus. There are five source selection options currently defined:

1. Source data is held in frame buffer memory (always full color data)
2. Source data is provided by the processor via the MMIO bus and is full color
3. Source data is provided by the processor via the MMIO bus and is a monochrome bitmap

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 need 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 into all four byte lanes in 8-bpp mode. In 15 or 16-bpp mode, the lower word will be replicated into 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.

Table 28: MonoPatFColor

Bit	Symbol	Access	Value	Description
Offset 0x04 F5F8 MonoPatFColor				
31:0	MonoPatFColor[31:0]	R/W	NI	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 need 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 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-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 29: MonoPatBColor

Bit	Symbol	Access	Value	Description
Offset 0x04 F5FC MonoPatBColor				
31:0	MonoPatBColor[31:0]	R/W	NI	Holds the background color monochrome pattern expansion, lines, and solid fills.

5.17 Statistics Counters

In Ethernet applications generally, many counters that keep Ethernet traffic statistics must be maintained. There are a number of standards specifying such counters, such as IEEE Std 802.3 / Clause 30. Other standards are RFC 2665 and RFC 2233.

The approach taken here is as follows: by default, all counters are implemented in software. With the help of the StatusInfo field in the packet status word, many of the important statistics events listed in the standards can be counted by software. Currently, no counters are added in hardware.

5.18 Status Vectors

A transmit status vector and a receive status vector are available in registers TSV0, TSV1, and RSV. These registers can be polled with software. Normally, they are of limited use, because the communication between driver software and Ethernet module takes place primarily through the packet descriptors. Statistical events are counted by software in the device driver. However, these transmit and receive status vectors are made visible for debug purposes. The values in these registers are simple copies of the transmit and receive status vectors as produced by the MII Interface. They are valid as long as the status vectors are valid, and should typically only be read when the transmit and receive processes are halted.

The TSV0, TSV1 and RSV registers are defined in Section 3.2.

5.19 Reset

The LAN100 has a hard reset input and several soft resets which can be activated by setting the appropriate bits in its registers. All registers in the LAN100 have a reset value of 0 unless specified differently in Section 3.2.

5.19.1 Hard Reset

The LAN100 module experiences a hard reset when the PNX15xx Series is reset. After a hard reset, all register values in the software view will be set to their default value as specified in Section 3.2.

5.19.2 Soft Reset

Parts of the LAN100 can be reset by software by setting bits in the Command register and the MAC1 configuration register.

The MAC1 register has six different reset bits:

- SOFT_RESET: Setting this bit will put all components in the MII Interface in the reset state except for the MII Interface registers (in address locations 72 0000 to 72 00FC). The soft reset bit defaults to 1, and must be cleared after a hardware reset to enable the MII Interface.
- SIMULATION_RESET: Setting this bit resets the random number generator in the Transmit Function. The value after a hard reset is 0.

3.1 Register Tables

Table 2: IIC Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 5000	*I2C CONTROL*			
31:8	Unused		-	Ignore upon read. Write as zeroes.
7	AA	R/W	0	IIC acknowledge bit 0 = Acknowledge not returned during acknowledge clock pulse 1 = Acknowledge returned during acknowledge clock pulse
6	EN	R/W	0	IIC enable bit 0 = Disable IIC module 1 = Enable IIC module
5	STA	R/W	0	IIC start bit 0 = Slave mode, accept transactions 1 = Master mode, generate start condition if bus is free
4	STO	R	0	IIC stop bit 0 = Slave mode, accept transactions 1 = Generate stop condition on I2C bus when IIC module is master.
3	Unused		-	Ignore upon read. Write as zeroes.
2:0	CR	R/W	100	These three bits determine the serial clock frequency when IIC module is in master mode. This field shall be changed only when EN bit is 0. The IIC Clock is divided as follows to achieve the desired frequency. The table assumes the IIC module receives a 24 MHz clock from the Clock module. 0: 60 -> 400 KHz 1: 80 -> 300 KHz 2: 120 -> 200 KHz 3: 160 -> 150 KHz 4: 240 -> 100 KHz 5: 320 -> 75 KHz 6: 480 -> 50 KHz 7: 960 -> 25 KHz

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 IIC module is in the master receiver mode.
- A data byte has been received while IIC module is in the addressed slave receiver mode.

This powerdown state can be initiated by an external host processor by writing to bit TM32_CONTROL.TM_PWRDWN_REQ, see Chapter 3 System On Chip Resources. The TM3260 only exits this mode when this bit is de-asserted. At this point in time the TM3260 clock may be removed by the host.

The second method to shutdown the TM3260 clock as well as the MMIO clock is to follow the procedure defined in Chapter 5 The Clock Module Section 27 on page 27-1. This is solution for standalone systems where PNX15xx Series is the master of the system.

1.1.5 SDRAM Controller

Power consumption of the MMI is lowest when it is halted. There are two different ways to achieve halting the MMI:

- Writing the halt register field of a software programmable MMIO register.
- Programming the MMI to go into halt mode automatically after a certain period of inactivity.

Remark: Before halting the MMI, make sure that there are no pending memory transactions.

MMIO Directed Halt

The HALT bit of MMIO register IP_2031_CTL can be written with a ‘1’ to indicate a request for halting. Write a ‘0’ to this bit to indicate a request for taking the DDR controller out of halt mode.

Remark: It is recommended that putting the MMI in MMIO direct-halt mode (with MMIO registers) before reprogramming the configuration and timing registers in MMI so that the on-going transactions are not effected. When MMIO registers DDR_MR and DDR_EMR are reprogrammed, a start action has to be performed (after the MMI is unhalted), for the new DDR values to take effect.

Auto Halt

The MMI can be programmed such that it goes into halt mode when it has observed a certain period of inactivity. This is accomplished by programming the MMIO registers AUTO_HAL_LIMIT and IP_2031_CTL. The MMI will exit the halt mode automatically when a new MTL memory request is presented to one of its input ports. The MTL clock and DCS clock cannot be turned off to operate in this mode.

Remark: This modes introduces extra latency on memory transactions and it is not a recommended operating mode.