Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Active
Core Processor	ARM926EJ-S
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	454MHz
Co-Processors/DSP	Data; DCP
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	LCD, Touchscreen
Ethernet	-
SATA	-
USB	USB 2.0 + PHY (1)
Voltage - I/O	2.0V, 2.5V, 2.7V, 3.0V, 3.3V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	Cryptography, Hardware ID
Package / Case	169-LFBGA
Supplier Device Package	169-MAPBGA (11x11)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mcimx233cjm4c

Table 5-239. HW_ICOLL_INTERRUPT109 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD1**	RO	0x0	Always write zeroes to this bitfield.
4	**ENFIQ**	RW	0x0	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
3	**SOFTIRQ**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 turn off the software interrupt request. FORCE_INTERRUPT = 0x1 force a software interrupt
2	**ENABLE**	RW	0x0	Enable the interrupt bit through the collector. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY**	RW	0x0	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). LEVEL0 = 0x0 level 0, lowest or weakest priority LEVEL1 = 0x1 level 1 LEVEL2 = 0x2 level 2 LEVEL3 = 0x3 level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_INTERRUPT109_SET(0,0x00000001);
```

5.4.120 Interrupt Collector Interrupt Register 110 Description

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT110	0x800
HW_ICOLL_INTERRUPT110_SET	0x804
HW_ICOLL_INTERRUPT110_CLR	0x808
HW_ICOLL_INTERRUPT110_TOG	0x80C

Table 5-240. HW_ICOLL_INTERRUPT110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	

6.4.40 AHB Layer 1 Transfer Count Register Description

The AHB Layer 1 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 1.

HW_DIGCTL_L1_AHB_ACTIVE_CYCLES 0x370

Table 6-80. HW_DIGCTL_L1_AHB_ACTIVE_CYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-81. HW_DIGCTL_L1_AHB_ACTIVE_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 1.

DESCRIPTION:

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_AHB_STATS_SELECT_L1_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L1_AHB_ACTIVE_CYCLES_COUNT_RD();
```

6.4.41 AHB Layer 1 Performance Metric for Stalled Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

HW_DIGCTL_L1_AHB_DATA_STALLED 0x380

Table 6-82. HW_DIGCTL_L1_AHB_DATA_STALLED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 6-83. HW_DIGCTL_L1_AHB_DATA_STALLED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field counts the number of AHB cycles in which a master was stalled.

Then, the approximate write latency for a given word is:

*tHCLK * 32 * 32 * n*

In addition to this latency, software must allow for the 2-µs postamble (using HW_DIGCTL_MICROSECONDS), as described in Section 7.2.3, “Write Postamble,”

7.2.3 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 µs after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes. A recommended software sequence to meet the postamble requirements is as follows:

1. Issue the write and poll for BUSY (as per Section 7.2.2, “Software Write Sequence,”).
2. Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 µs.
3. Perform the next OTP operation.

7.2.4 Shadow Registers and Hardware Capability Bus

The on-chip customer hardware capability bus is generated using a direct connection to the HW_OCOTP_CUSTCAP shadow register. The bits are copied from the OTP on reset. They can be modified until HW_OCOTP_LOCK_CUSTCAP_SHADOW is set.

The user can force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device, which is useful for debugging code. To force a reload:

- Set HW_OCOTP_CTRL_RELOAD_SHADOWS.
- Wait for HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_RELOAD_SHADOWS to be cleared by the controller.
- Attempting to write to the shadow registers while the shadows are being reloaded will result in the setting of HW_OCOTP_CTRL_ERROR. In addition, the register will not take the attempted write (yielding to the reload instead).
- Attempting to write to a shadow register that is locked will result in the setting of HW_OCOTP_CTRL_ERROR.

HW_OCOTP_CTRL_RELOAD_SHADOWS can be set at any time. There is no need to wait for HW_OCOTP_CTRL_BUSY or HW_OCOTP_CTRL_ERROR to be clear.

- In the case of HW_OCOTP_CTRL_BUSY being set due to an active write, the controller will perform the bank opening and shadow reloading immediately after the completion of the write.
- In the case where HW_OCOTP_CTRL_RD_BANK_OPEN is set, the shadow reload will be performed immediately after the banks are closed by software (by clearing HW_OCOTP_CTRL_RD_BANK_OPEN). It should be noted that BUSY will take approximately 33 HCLK cycles to clear, so polling for HW_OCOTP_CTRL_BUSY immediately after clearing HW_OCOTP_CTRL_RD_BANK_OPEN is not recommended.

11.5.54 AHB to APBX DMA Channel 6 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

HW_APBX_CH6_DEBUG2 0x400

Table 11-110. HW_APBX_CH6_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 11-111. HW_APBX_CH6_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 6.

EXAMPLE:

```
Empty example.
```

11.5.55 APBX DMA Channel 7 Current Command Address Register Description

The APBX DMA Channel 7 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH7_CURCMDAR 0x410

Table 11-112. HW_APBX_CH7_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-113. HW_APBX_CH7_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 7.

EXAMPLE:

```
Empty example.
```

11.5.63 APBX DMA Channel 8 Next Command Address Register Description

The APBX DMA Channel 8 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH8_NXTCMDAR 0x490

Table 11-128. HW_APBX_CH8_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 11-129. HW_APBX_CH8_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	CMD_ADDR	RW	0x00000000	Pointer to next command structure for Channel 8.

DESCRIPTION:

APBX DMA Channel 8 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 8 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.64 APBX DMA Channel 8 Command Register Description

The APBX DMA Channel 8 command register specifies the cycle to perform for the current command chain item.

HW_APBX_CH8_CMD 0x4A0

Table 11-130. HW_APBX_CH8_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 12-26. HW_DRAM_CTL09 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:1	**RSVD1**	RO	0x0	Reserved.
0	**WRITEINTERP**	RW	0x0	Allow controller to interrupt a write bursts to the DRAMs with a read command. Defines whether the memory controller can interrupt a write burst with a read command. Some memory devices do not allow this functionality. 0 = The device does not support read commands interrupting write commands. 1 = The device does support read commands interrupting write commands.

DESCRIPTION:

DRAM Control registers. Individual fields control various aspects of the DRAM interface. See bit fields for descriptions

EXAMPLE:

```
Empty Example.
```

12.5.13 DRAM Control Register 10 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL10 0x028

Table 12-27. HW_DRAM_CTL10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4					AGE_COUNT			RSVD3					ADDR_PINS			RSVD2						TEMRS		RSVD1							Q_FULLNESS	

Table 12-28. HW_DRAM_CTL10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD4**	RO	0x0	Reserved.
26:24	**AGE_COUNT**	RW	0x0	Initial value of master aging-rate counter for command aging. Holds the initial value of the master aging-rate counter. When using the placement logic to fill the command queue, the command aging counters will be decremented one each time the master aging-rate counter counts down age_count cycles.
23:19	**RSVD3**	RO	0x0	Reserved.

engine with a transfer first, because each chain includes a wait4ready command structure. As a result, firmware should look at the HW_ECC8_STATUS0_COMPLETED_CE bit field to determine which block is being reported in the status register. There is also a 16 bit HANDLE field in the HW_GPMI_ECCCTRL register that is passed down the pipeline with each transaction. This handle field can be used to speed firmware's detection of which transaction is being reported.

These examples of reading and writing have focused on full page transfers of 4K page NAND devices. Set HW_GPMI_ECCCTRL_ECC_CMD to a value of HW_GPMI_ECCCTRL_ECC_CMD_ENCODE_4_BIT or to a value of HW_GPMI_ECCCTRL_ECC_CMD_DECODE_4_BIT to enable encode and decode of up to full page transfers of 2K page NAND devices.

To reiterate, you can select a single block to transfer within one transaction by setting only one bit in the HW_GPMI_ECCCTRL_BUFFER_MASK bit field. There is a 1:1 correspondence between a bit in this bit field and a 512-byte buffer address offset into the payload area pointed to by the HW_GPMI_PAYLOAD register. The ECC8 and GPMI blocks are designed to be very efficient at reading single 512-byte pages in one transaction. With no errors, the transaction takes less than 20 HCLKs longer than the time to read the raw data from the NAND.

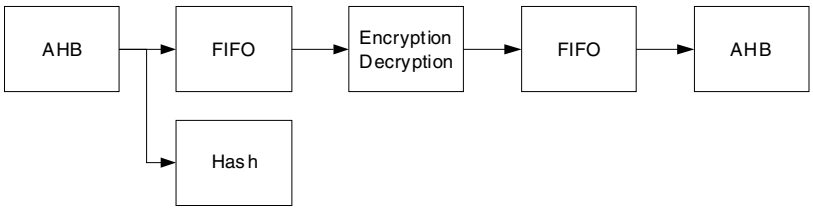
Additionally, you can select multiple contiguous blocks to transfer within one transaction by setting the respective bits in the HW_GPMI_BUFFER_MASK bit fields. The selected bits must represent a contiguous block of data in the NAND.

To summarize, the APBH DMA command chain for a Reed-Solomon decode operation is shown in Figure 14-11. Seven DMA command structures must be present for each NAND read transaction decoded by the ECC8. The seven DMA command structures for multiple NAND read transaction blocks can be chained together to make larger units of work for the ECC8, and each will produce an appropriate error report in the ECC8 PIO space. Multiple NAND devices can have such multiple chains scheduled. The results can come back out of order with respect to the multiple chains.

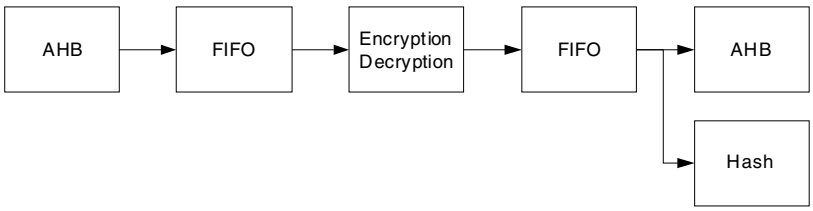
If uncorrectable errors occur, it is up to software to determine how to deal with the bad block. One strategy might be to reread the data from NAND flash in the hope that enough soft errors will have been removed to make correction possible on a second pass.

14.2.4 Interrupts

There are two interrupt sources used in processing ECC8 protected NAND read and write transfers. Since all ECC8 operations are initiated by GPMI DMA command structures, the DMA completion interrupt for the GPMI is an important ISR. Both of the flow charts of Figure 14-6 and Figure 14-9 show the GPMI DMA complete ISR skeleton. In both reads and writes, the GPMI DMA completion interrupt is used to schedule work ***INTO*** the error correction pipeline. As the front end processing completes, the DMA interrupt is generated and additional work, i.e. DMA chains, are passed to the GPMI DMA to keep it



- **Encryption and Output Hashing—**Data from source buffer is encrypted/decrypted into destination, and output data is hashed.



16.1.1 DCP Limitations for Software

While the DCP module has been designed to be as flexible as possible, there are a few limitations to which software must adhere:

- Buffer sizes for all operations MUST be aligned to the natural size of the transfer algorithm used. Memcopy operations can transfer any number of bytes (one-byte granularity) and AES operations must be multiples of 16 bytes (four-word granularity). For all operations, if the byte count is not a word granularity, the hardware rounds up to the next word. Hashing is supported at a byte granularity.
- The DCP module supports buffer operations to any byte alignment, but performance will be improved if buffers are aligned to a four-byte boundary, since fetch/store operations can be performed without having to do byte operations to accommodate the misaligned addresses.
- Hash operations are limited to a 512-Mbyte buffer size. The hardware only implements a 32-bit hash length counter instead of the 64-bit counter supported by the SHA-1 algorithm (counter counts bits, not bytes, therefore a total of 512 Mbytes).
- For chained hashing operations (operations involving multiple descriptors), every descriptor except the last must have a byte count that is a 16-word multiple (granularity of the hash algorithm).
- Key values cannot be written while the AES block is active. This limitation exists because the key RAM is in use while AES is operational. Any writes from the APB cannot be held in wait states; therefore, the RAM must be accessible during key writes.
- The byte-swap controls can only be used with modulo-4 length buffers. For non-modulo-4 lengths, the final partial word will contain incorrect data. Any address alignment can be used with byte swapping, however.

Table 16-32. HW_DCP_PACKET1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	**HASH_INIT**	RO	0x0	Reflects whether the current hashing block is the initial block in the hashing operation, so the hash registers should be initialized before the operation.
11	**PAYLOAD_KEY**	RO	0x0	When set, indicates the payload contains the key. This bit takes precedence over the OTP_KEY control
10	**OTP_KEY**	RO	0x0	Reflects whether a hardware-based key should be used. The KEY_SELECT field from the Control1 field is used to select from multiple hardware keys. The PAYLOAD_KEY bit takes precedence over the OTP_KEY bit.
9	**CIPHER_INIT**	RO	0x0	Reflects whether the cipher block should load the initialization vector from the payload for this operation.
8	**CIPHER_ENCRYPT**	RO	0x0	When the cipher block is enabled, this bit indicates whether the operation is encryption or decryption. ENCRYPT = 0x01 DECRYPT = 0x00
7	**ENABLE_BLIT**	RO	0x0	Reflects whether the DCP should perform a blit operation. Source data is always continuous and the destination buffer is written in run/stride format. When set, the BUFFER_SIZE field is treated as two 16-bit values for the X-Y extents of the blit operation.
6	**ENABLE_HASH**	RO	0x0	Reflects whether the selected hashing function should be enabled for this operation.
5	**ENABLE_CIPHER**	RO	0x0	Reflects whether the selected cipher function should be enabled for this operation.
4	**ENABLE_MEMCOPY**	RO	0x0	Reflects whether the selected hashing function should be enabled for this operation.
3	**CHAIN_CONTIGUOUS**	RO	0x000000	Reflects whether the next packet's address is located following this packet's payload.
2	**CHAIN**	RO	0x0	Reflects whether the next command pointer register should be loaded into the channel's current descriptor pointer.
1	**DECR_SEMAPHORE**	RO	0x0	Reflects whether the channel's semaphore should be decremented at the end of the current operation. When the semaphore reaches a value of zero, no more operations will be issued from the channel.
0	**INTERRUPT**	RO	0x0	Reflects whether the channel should issue an interrupt upon completion of the packet.

DESCRIPTION:

This register shows the contents of the Control0 register from the packet being processed.

EXAMPLE:

```
Empty Example.
```

16.3.11 DCP Work Packet 2 Status Register Description

This register displays the values for the current work packet offset 0x08 (Control1) field.

HW_DCP_PACKET2 0x0A0

Table 16-48. HW_DCP_CH0STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**HASH_MISMATCH**	RW	0x0	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0	**RSVD1**	RO	0x0	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

DESCRIPTION:

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

EXAMPLE:

```
Empty Example.
```

16.3.19 DCP Channel 0 Options Register Description

The DCP Channel 0 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH0OPTS	0x130
HW_DCP_CH0OPTS_SET	0x134
HW_DCP_CH0OPTS_CLR	0x138
HW_DCP_CH0OPTS_TOG	0x13C

Table 16-49. HW_DCP_CH0OPTS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																RECOVERY_TIMER															

Table 17-23. HW_PXP_S0UBUF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	Address pointer for the S0 (video) U/Cb Chroma input buffer. The address MUST be word-aligned for proper PXP operation.

DESCRIPTION:

This register contains the pointer to the Chroma U/Cb buffer when performing colorspace conversion. This register is unused when processing RGB data.

EXAMPLE:

```
HW_PXP_S0BUF_WR(image_y); // Y (luma) image data
HW_PXP_S0UBUF_WR(image_u); // U (Cb) image data
HW_PXP_S0VBUF_WR(image_v); // V (Cr) image data
```

17.4.8 Source 0 V/Cr Input Buffer Pointer Description

S0 Chroma (V/Cr) Input Buffer Pointer. This register points to the beginning of the Source 0 V/Cr input buffer.

HW_PXP_S0VBUF 0x070

Table 17-24. HW_PXP_S0VBUF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDR																															

Table 17-25. HW_PXP_S0VBUF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	Address pointer for the S0 (video) V/Cr Chroma input buffer. The address MUST be word-aligned for proper PXP operation.

DESCRIPTION:

This register contains the pointer to the Chroma V/Cr buffer when performing colorspace conversion. This register is unused when processing RGB data.

EXAMPLE:

```
HW_PXP_S0BUF_WR(image_y); // Y (luma) image data
HW_PXP_S0UBUF_WR(image_u); // U (Cb) image data
HW_PXP_S0VBUF_WR(image_v); // V (Cr) image data
```

17.4.9 PXP Source 0 (video) Buffer Parameters Description

This register contains buffer information for the S0 input RGB/YUV buffer.

HW_PXP_S0PARAM 0x080

DESCRIPTION:

The S1 Overlay 1 Parameter register provides additional controls for Overlay 1.

EXAMPLE:

```
u32 olparam;
olparam = BF_PXP_OLnPARAM_ENABLE (1);
olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
olparam |= BF_PXP_OLnPARAM_FORMAT (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
olparam |= BF_PXP_OLnPARAM_ROP (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(1,olparam); // enable overlay to perform XOR ROP using RGB8888 overlay
```

17.4.32 PXP Overlay 1 Parameters 2 Description

This register contains buffer parameters for the Overlay 1 input buffer.

HW_PXP_OL1PARAM2 0x270

Table 17-72. HW_PXP_OL1PARAM2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																															

Table 17-73. HW_PXP_OL1PARAM2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RSVD**	RO	0x00000000	Reserved, always set to zero.

DESCRIPTION:

The Overlay 1 Parameter 2 register is reserved for future use.

EXAMPLE:

```
Empty Example.
```

17.4.33 PXP Overlay 2 Buffer Pointer Description

Overlay 2 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 2 input buffer.

HW_PXP_OL2 0x280

Table 17-74. HW_PXP_OL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDR																															

Table 17-75. HW_PXP_OL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	Address pointer for the overlay 2 buffer. The address MUST be word-aligned for proper PXP operation.

DESCRIPTION:

The S1 Overlay 4 Parameter register provides additional controls for Overlay 4.

EXAMPLE:

```
u32 olparam;
olparam = BF_PXP_OLnPARAM_ENABLE (1);
olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
olparam |= BF_PXP_OLnPARAM_FORMAT (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
olparam |= BF_PXP_OLnPARAM_ROP (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(4,olparam); // enable overlay to perform XOR ROP using RGB8888 overlay
```

17.4.44 PXP Overlay 4 Parameters 2 Description

This register contains buffer parameters for the Overlay 4 input buffer.

HW_PXP_OL4PARAM2 0x330

Table 17-96. HW_PXP_OL4PARAM2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																															

Table 17-97. HW_PXP_OL4PARAM2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RSVD**	RO	0x00000000	Reserved, always set to zero.

DESCRIPTION:

The Overlay 4 Parameter 2 register is reserved for future use.

EXAMPLE:

```
Empty Example.
```

17.4.45 PXP Overlay 5 Buffer Pointer Description

Overlay 5 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 5 input buffer.

HW_PXP_OL5 0x340

Table 17-98. HW_PXP_OL5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDR																															

Table 17-99. HW_PXP_OL5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	Address pointer for the overlay 5 buffer. The address MUST be word-aligned for proper PXP operation.

DESCRIPTION:

The S1 Overlay 5 Parameter register provides additional controls for Overlay 5.

EXAMPLE:

```
u32 olparam;
olparam = BF_PXP_OLnPARAM_ENABLE (1);
olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
olparam |= BF_PXP_OLnPARAM_FORMAT (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
olparam |= BF_PXP_OLnPARAM_ROP (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(5,olparam); // enable overlay to perform XOR ROP using RGB8888 overlay
```

17.4.48 PXP Overlay 5 Parameters 2 Description

This register contains buffer parameters for the Overlay 5 input buffer.

HW_PXP_OL5PARAM2 0x370

Table 17-104. HW_PXP_OL5PARAM2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																															

Table 17-105. HW_PXP_OL5PARAM2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RSVD**	RO	0x00000000	Reserved, always set to zero.

DESCRIPTION:

The Overlay 5 Parameter 2 register is reserved for future use.

EXAMPLE:

```
Empty Example.
```

17.4.49 PXP Overlay 6 Buffer Pointer Description

Overlay 6 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 6 input buffer.

HW_PXP_OL6 0x380

Table 17-106. HW_PXP_OL6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDR																															

Table 17-107. HW_PXP_OL6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDR**	RW	0x0	Address pointer for the overlay 6 buffer. The address MUST be word-aligned for proper PXP operation.

19.4.3 TV Encoder Filter Control Register Description

This is filter control register of the TV Encoder Block

HW_TVENC_FILTCTRL	0x020
HW_TVENC_FILTCTRL_SET	0x024
HW_TVENC_FILTCTRL_CLR	0x028
HW_TVENC_FILTCTRL_TOG	0x02C

Table 19-5. HW_TVENC_FILTCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1												YSHARP_BW	YD_OFFSETSEL	SEL_YLPF	SEL_CLPF	SEL_YSHARP	YLPF_COEFSEL	COEFSEL_CLPF	YS_GAINSGN	YS_GAINSEL		RSRVD2	RSRVD3	RSRVD4							

Table 19-6. HW_TVENC_FILTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
19	**YSHARP_BW**	RW	0x0	Controls the luma sharpness filter bandwidth inside the luma filter sub-block.
18	**YD_OFFSETSEL**	RW	0x0	LU sub-block vector controls. Control the luma offset: 0 : do not substract 1 : subtract 16 from luma.
17	**SEL_YLPF**	RW	0x0	LU sub-block vector controls. Enables the luma low pass filter.
16	**SEL_CLPF**	RW	0x0	LU sub-block vector controls. Enables the chroma low pass filter.
15	**SEL_YSHARP**	RW	0x0	LU sub-block vector controls. Enables the luma sharpness filter.
14	**YLPF_COEFSEL**	RW	0x0	LU sub-block vector controls. Controls the luma low pass filter bandwidth: 0 : 5.5 MHz 1 : 4.2 MHz
13	**COEFSEL_CLPF**	RW	0x0	LU sub-block vector controls. Controls the chroma low pass filter bandwidth: 0 : 1.3 MHz 1 : 0.6 MHz
12	**YS_GAINSGN**	RW	0x0	LU sub-block vector controls. Controls the sign of the sharpness modification: 0 : positive 1 : negative
11:10	**YS_GAINSEL**	RW	0x0	LU sub-block vector controls. Controls the degree of luma sharpness enhancement by luma sharpness filter: 00 : 3dB 01 : 6dB 10 : 9dB 11 : 12dB

Chapter 21
Synchronous Serial Ports (SSP)

This chapter describes the two identical synchronous serial ports (SSP) included on the i.MX23. It includes sections on external pins, bit rate generation, frame formats, Winbond SPI mode, Motorola SPI mode, Texas Instruments Synchronous Serial Interface (SSI) mode, and SD/SDIO/MMC mode. Programmable registers are described in Section 21.10, “Programmable Registers.

21.1 Overview

The synchronous serial port is a flexible interface for inter-IC and removable media control and communication. The SSP supports master operation of SPI, Texas Instruments SSI and 1-bit, 4-bit, and 8-bit SD/SDIO/MMC. The SPI mode has enhancements to support 1-bit legacy MMC cards. SPI master dual (2-bit) and quad (4-bit) mode reads are also supported. The SSP also supports slave operation for the SPI and SSI modes. The SSP has a dedicated DMA channel in the bridge and can also be controlled directly by the CPU through PIO registers. Figure 21-1 illustrates one of the two SSP ports included on the i.MX23. The only interaction between SSP1 and SSP2 is that they share the same input, SSPCLK.

DESCRIPTION:

The IMSC register is the Interrupt Mask Set/Clear Register. On a read, this register gives the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

EXAMPLE:

```
No Example.
```

27.3.11 UART Raw Interrupt Status Register Description

The RIS register is the Raw Interrupt Status Register. It is a read-only register. On a read this register gives the current raw status value of the corresponding interrupt. A write has no effect. All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

HW_UARTDBGRIS 0x03C

Table 27-22. HW_UARTDBGRIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRMIS	DCDRMIS	CTSRMIS	RIRMIS

Table 27-23. HW_UARTDBGRIS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15:11	**RESERVED**	RO	0x0	Reserved, read as zero, do not modify.
10	**OERIS**	RO	0x0	Overrun Error Interrupt Status.
9	**BERIS**	RO	0x0	Break Error Interrupt Status.
8	**PERIS**	RO	0x0	Parity Error Interrupt Status.
7	**FERIS**	RO	0x0	Framing Error Interrupt Status.
6	**RTRIS**	RO	0x0	Receive Timeout Interrupt Status.
5	**TXRIS**	RO	0x0	Transmit Interrupt Status.
4	**RXRIS**	RO	0x0	Receive Interrupt Status.
3	**DSRRMIS**	RO	0x0	nUARTDSR Modem Interrupt Status.
2	**DCDRMIS**	RO	0x0	nUARTDCD Modem Interrupt Status.
1	**CTSRMIS**	RO	0x0	nUARTCTS Modem Interrupt Status.
0	**RIRMIS**	RO	0x0	nUARTRI Modem Interrupt Status.

DESCRIPTION:

The RIS register is the Raw Interrupt Status Register. It is a read-only register. On a read this register gives the current raw status value of the corresponding interrupt. A write has no effect. All the bits, except for

Table 37-8. HW_PINCTRL_MUXSEL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	**BANK0_PIN10**	RW	0x3	Pin 54, GPMI_D10 pin function selection: 00= gpmi_data10; 01= lcd_d20; 10= ssp1_d6; 11= GPIO.
19:18	**BANK0_PIN09**	RW	0x3	Pin 53, GPMI_D09 pin function selection: 00= gpmi_data09; 01= lcd_d19; 10= ssp1_d5; 11= GPIO.
17:16	**BANK0_PIN08**	RW	0x3	Pin 52, GPMI_D08 pin function selection: 00= gpmi_data08; 01= lcd_d18; 10= ssp1_d4; 11= GPIO.
15:14	**BANK0_PIN07**	RW	0x3	Pin 50, GPMI_D07 pin function selection: 00= gpmi_data07; 01= lcd_d15; 10= ssp2_d7; 11= GPIO.
13:12	**BANK0_PIN06**	RW	0x3	Pin 51, GPMI_D06 pin function selection: 00= gpmi_data06; 01= lcd_d14; 10= ssp2_d6; 11= GPIO.
11:10	**BANK0_PIN05**	RW	0x3	Pin 48, GPMI_D05 pin function selection: 00= gpmi_data05; 01= lcd_d13; 10= ssp2_d5; 11= GPIO.
9:8	**BANK0_PIN04**	RW	0x3	Pin 49, GPMI_D04 pin function selection: 00= gpmi_data04; 01= lcd_d12; 10= ssp2_d4; 11= GPIO.
7:6	**BANK0_PIN03**	RW	0x3	Pin 47, GPMI_D03 pin function selection: 00= gpmi_data03; 01= lcd_d11; 10= ssp2_d3; 11= GPIO.
5:4	**BANK0_PIN02**	RW	0x3	Pin 46, GPMI_D02 pin function selection: 00= gpmi_data02; 01= lcd_d10; 10= ssp2_d2; 11= GPIO.

Table 37-86. HW_PINCTRL_IRQEN0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**IRQEN**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 0. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT0, an interrupt will be propagated to the interrupt collector as interrupt GPIO0.

For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT0 (corresponding to pins GPIO0[2] and GPIO0[4]) will cause interrupts from bank 0.

EXAMPLE:

```
Empty Example.
```

37.4.42 PINCTRL Bank 1 Interrupt Mask Register Description

The PINCTRL Bank 1 Interrupt Mask Register contains interrupt enable masks for the pins in bank 1.

HW_PINCTRL_IRQEN1	0x910
HW_PINCTRL_IRQEN1_SET	0x914
HW_PINCTRL_IRQEN1_CLR	0x918
HW_PINCTRL_IRQEN1_TOG	0x91C

Table 37-87. HW_PINCTRL_IRQEN1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1	IRQEN																														

Table 37-88. HW_PINCTRL_IRQEN1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD1**	RO	0x0	Reserved - write 0 to this bit-field.
30:0	**IRQEN**	RW	0x00000000	Each bit in this register corresponds to one of the 31 pins in bank 1: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT1. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT1.

get fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (i.e., one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read/modify/write operations. When atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

39.2 Naming Convention

The generated include files and macros follow a consistent naming convention that matches the SOC documentation. This prevents name-space collisions and makes the macros easier to remember.

```
//
// The include file for a specific hardware module is named:
//
//     regs<module>.h
//
// Every register has an associated typedef that provides a C definition of
// the register. The definition is always a union of a 32-bit unsigned int
// (i.e., reg32_t), and an anonymous bit field structure.
//
//     hw_<module>_<regname>_t
//
// Macros and defines that relate to a register as a whole are named:
//
//     HW_<module>_<regname>_ADDR
//     HW_<module>_<regname>_<SET | CLR | TOG>_ADDR
//         - defines for the indicated register address
//
//     HW_<module>_<regname>
//         - a define for accessing the primary register using the typedef.
//           Should be used as an rvalue (i.e., for reading), but avoided as
//           an lvalue (i.e., for writing). Will usually generate RMW when
//           used as an lvalue.
//
//     HW_<module>_<regname>_RD()
//     HW_<module>_<regname>_WR()
//         - macros for reading/writing the primary register as a whole
//
//     HW_<module>_<regname>_<SET | CLR | TOG>()
//         - macros for writing the associated set | clear | toggle registers
//
```