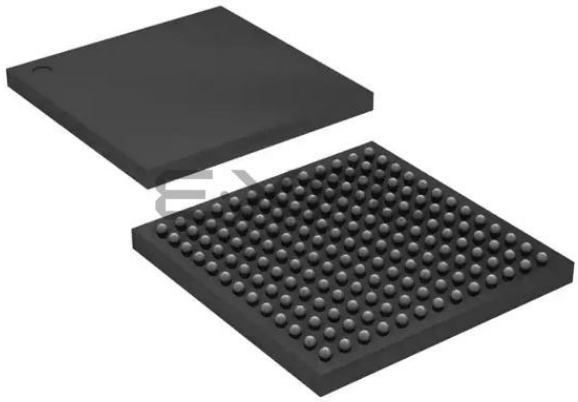

Welcome to E-XFL.COM

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	Obsolete
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/product-detail/nxp-semiconductors/mcimx233cjm4cr2

Tables

DESCRIPTION:

This register indicates the CPU Frequency limit.

EXAMPLE:

```
HW_CLKCTRL_STATUS_RD(BF_CLKCTRL_STATUS_CPU_LIMIT());
```

4.8.20 ClkCtrl Version Description

The VERSION control register is a read only status of the clkctrl block version.

HW_CLKCTRL_VERSION 0x140

Table 4-40. HW_CLKCTRL_VERSION

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
MAJOR								MINOR								STEP															

Table 4-41. HW_CLKCTRL_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**MAJOR**	RO	0x4	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	**MINOR**	RO	0x0	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	**STEP**	RO	0x0	Fixed read-only value reflecting the stepping of the RTL version.

DESCRIPTION:

This register indicates the RTL version in use.

EXAMPLE:

```
HW_CLKCTRL_VERSION_RD(BF_CLKCTRL_VERSION_MAJOR());
```

CLKCTRL Block v4.0, Revision 1.48

Table 5-255. HW_ICOLL_INTERRUPT117 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_INTERRUPT117_SET(0,0x00000001);
```

5.4.128 Interrupt Collector Interrupt Register 118 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_INTERRUPT118	0x880
HW_ICOLL_INTERRUPT118_SET	0x884
HW_ICOLL_INTERRUPT118_CLR	0x888
HW_ICOLL_INTERRUPT118_TOG	0x88C

Table 5-256. HW_ICOLL_INTERRUPT118

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	

Table 5-278. HW_ICOLL_DBGREAD0

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
VALUE																															

Table 5-279. HW_ICOLL_DBGREAD0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	VALUE	RO	0xECA94567	Fixed read-only value.

DESCRIPTION:

This register is used to test the read mux paths on the APBH.

EXAMPLE:

```
if (HW_ICOLL_DBGREAD0_RD != 0xECA94567)
Error();
```

5.4.140 Interrupt Collector Debug Read Register 1 Description

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD1	0x1140
HW_ICOLL_DBGREAD1_SET	0x1144
HW_ICOLL_DBGREAD1_CLR	0x1148
HW_ICOLL_DBGREAD1_TOG	0x114C

Table 5-280. HW_ICOLL_DBGREAD1

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
VALUE																															

Table 5-281. HW_ICOLL_DBGREAD1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	VALUE	RO	0x1356DA98	Fixed read-only value.

DESCRIPTION:

This register is used to test the read mux paths on the APBH.

EXAMPLE:

```
if (HW_ICOLL_DBGREAD1_RD != 0x1356DA98)
Error();
```

5.4.141 Interrupt Collector Debug Flag Register Description

The Interrupt Collector debug flag register is used to post diagnostic state into simulation.

HW_ICOLL_DBGFLAG	0x1150
HW_ICOLL_DBGFLAG_SET	0x1154

- In all cases, the controller will clear HW_OCOTP_CTRL_RELOAD_SHADOWS after the successful completion of the operation.

7.3 Behavior During Reset

The OCOTP is always active. The shadow registers described in Section 7.4, “Programmable Registers,” automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is approximately 32 HCLK cycles after the deassertion of reset. These shadow registers can be reloaded as described in Section 7.2.4, “Shadow Registers and Hardware Capability Bus.”

7.4 Programmable Registers

The following registers are available for programmer access and control of the OCOTP.

7.4.1 OTP Controller Control Register Description

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

HW_OCOTP_CTRL	0x000
HW_OCOTP_CTRL_SET	0x004
HW_OCOTP_CTRL_CLR	0x008
HW_OCOTP_CTRL_TOG	0x00C

Table 7-1. HW_OCOTP_CTRL

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
WR_UNLOCK																RSRVD2		RELOAD_SHADOWS	RD_BANK_OPEN	RSRVD1		ERROR	BUSY	RSRVD0				ADDR			

Table 8-48. HW_USBCTRL_ENDPOINTLISTADDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**EPBASE**	RW	0x0	Endpoint List Pointer (Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). (i.e., one queue head per endpoint and direction.)
10:0	**RSVD**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

DESCRIPTION:

EndPoint List Address

EXAMPLE:

```
Empty Example.
```

8.6.25 Embedded TT Asynchronous Buffer Status and Control Register (Host Controller mode) Description

This register contains parameters needed for internal TT operations. This register is not used in the device controller operation. This is a read/write register. Writes must be DWORD writes.

HW_USBCTRL_TTCTRL 0x15c

Table 8-49. HW_USBCTRL_TTCTRL

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
RSVD1	TTHA							RSVD2																							

Table 8-50. HW_USBCTRL_TTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD1**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
30:24	**TTHA**	RW	0x0	Internal TT Hub Address Representation. Default is 0 (Read/Write). This field is used to match against the Hub Address field in QH and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in the QH/siTD.
23:0	**RSVD2**	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

Chapter 9
Integrated USB 2.0 PHY

This chapter describes the integrated USB 2.0 full-speed and high-speed PHY available on the i.MX23. Programmable registers are described in Section 9.4, “Programmable Registers.”

9.1 Overview

The i.MX23 contains an integrated USB 2.0 PHY macrocell capable of connecting to PC host systems at the USB full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s. See Figure 9-1 for a block diagram of the PHY. The integrated PHY provides a standard UTM interface. The USB_DP and USB_DN pins connect directly to a USB device connector.

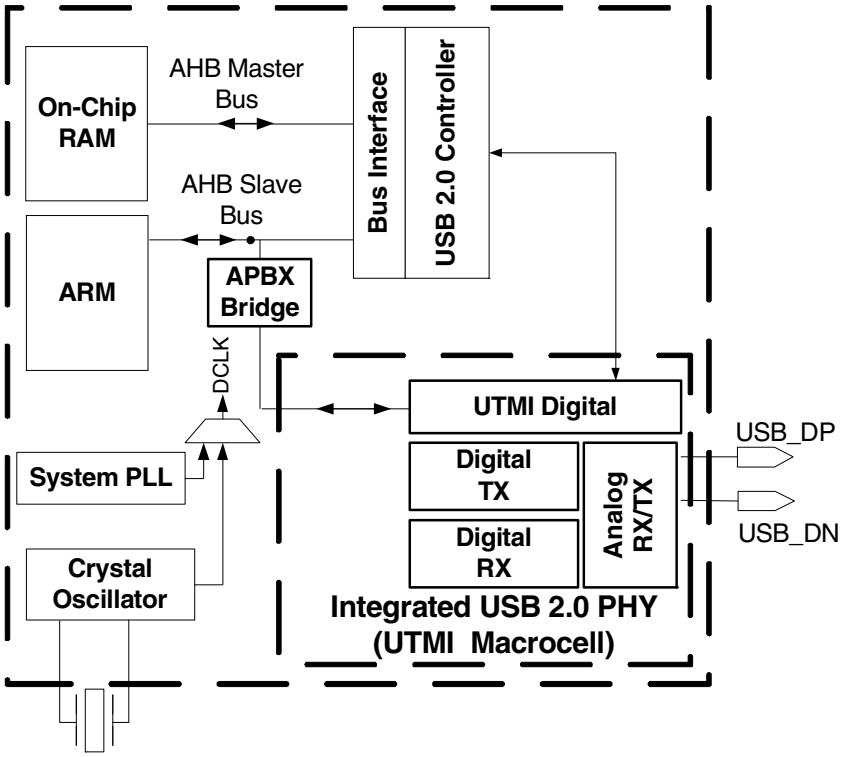

Figure 9-1. USB 2.0 PHY Block Diagram

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

Table 10-24. HW_APBH_CH0_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 10-25. HW_APBH_CH0_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 APBH DMA Channel 0.

EXAMPLE:

```
Empty example.
```

10.5.12 APBH DMA Channel 1 Current Command Address Register Description

The APBH DMA Channel 1 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH1_CURCMDAR 0x0B0

Table 10-26. HW_APBH_CH1_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 10-27. HW_APBH_CH1_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 1.

DESCRIPTION:

APBH DMA Channel 1 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Table 10-93. HW_APBH_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 5 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. TERMINATE = 0x14 When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. HALT_AFTER_TERM = 0x1D If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts. WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 5.

EXAMPLE:

```
Empty example.
```

10.5.46 AHB to APBH DMA Channel 5 Debug Information Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

HW_APBH_CH5_DEBUG2 0x2D0

Descriptor Legend

NEXT CMD ADDR										
CMD	<=	xfer_count	cmdwords	wait4endcmd	semaphore	nandwait4ready	nandlock	irqoncmplt	chain	command
BUFFER ADDR										
HW_GPMI_CTRL0	<=	command_mode	word_length	lock_cs	CS	address	address_increment	xfer_count		
HW_GPMI_COMPARE	<=	mask				reference				
HW_GPMI_ECCCTRL	<=	ecc_cmd			enable_ecc			buffer_mask		
HW_GPMI_ECCCOUNT										
HW_GPMI_PAYLOAD										
HW_GPMI_AUXILIARY										

Note: Refer to this legend when examining Figure 14-8 and Figure 14-11.

Figure 14-7. ECC8 DMA Descriptor Legend

14.4.1 Hardware ECC Accelerator Control Register Description

The Hardware ECC Accelerator Control Register provides overall control of the hardware ECC accelerator.

HW_ECC8_CTRL	0x000
HW_ECC8_CTRL_SET	0x004
HW_ECC8_CTRL_CLR	0x008
HW_ECC8_CTRL_TOG	0x00C

Table 14-1. HW_ECC8_CTRL

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
SFTRST	CLKGATE	AHBM_SFTRST	RSRVD2	THROTTLE				RSRVD1													DEBUG_STALL_IRQ_EN	DEBUG_WRITE_IRQ_EN	COMPLETE_IRQ_EN	RSRVD0				BM_ERROR_IRQ	DEBUG_STALL_IRQ	DEBUG_WRITE_IRQ	COMPLETE_IRQ

Table 14-2. HW_ECC8_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	0 = Normal ECC8 operation. 1 = Disable clocking with the ECC8 and hold it in its reset (lowest power) state (default). This bit can be turned on and then off to reset the ECC8 block to its default state. This bit resets all state machines except for the AHB master state machine. RUN = 0x0 Allow ECC8 to operate normally. RESET = 0x1 Hold ECC8 in reset.
30	**CLKGATE**	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block. RUN = 0x0 Allow ECC8 to operate normally. NO_CLKS = 0x1 Do not clock ECC8 gates in order to minimize power consumption.
29	**AHBM_SFTRST**	RW	0x1	Resets the AHB state machine. 0 = Normal ECC8 operation. 1 = Disable clocking with the ECC8 and hold it in its reset (lowest power) state (default). This bit can be turned on and then off to reset the ECC8 block to its default state. Do not use this bit for normal device soft-resets unless instructed to do so by Freescale. RUN = 0x0 Allow ECC8 to operate normally. RESET = 0x1 Hold ECC8 in reset.
28	**RSRVD2**	RO	0x0	Reserved.
27:24	**THROTTLE**	RW	0x0	Non-zero values will hold off that number of HCLKs between success burst requests on the AHB.
23:11	**RSRVD1**	RO	0x0	Reserved.
10	**DEBUG_STALL_IRQ_EN**	RW	0x0	1 = Interrupt on debug stall mode is enabled. The IRQ is raised on every block

```
write[7].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        | // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check

write[7].dma_bar = dma_error_handler;                   // if sense check fails, branch to error handler

//-----------------------------------------------------------------------------
// Descriptor 9: emit GPMI interrupt
//-----------------------------------------------------------------------------
write[8].dma_nxtcmdar = NULL;                                 // not used since this is last descriptor

write[8].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)          | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)          | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)          | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)          |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)          |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)          |
                   BF_APBH_CHn_CMD_IRQONCMPLT    (1)          | // emit GPMI interrupt
                   BF_APBH_CHn_CMD_CHAIN         (0)          | // terminate DMA chain processing
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
```

15.4.1.2 Using the BCH Encoder

To use the BCH encoder, first turn off the module-wide soft reset bit in both the GPMI and BCH blocks before starting any DMA activity. Note that turning off the soft reset must take place by itself, prior to programming the rest of the control registers. Turn off the BCH bus master soft reset bit (bit 29). Turn off the clock gate bits.

Program the remainder of the GPMI, BCH and APBH DMA as follows:

```
// bring APBH out of reset
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_SFRST);
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_CLKGATE);

// bring BCH out of reset
HW_BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
HW_BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);

// bring gpmi out of reset
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
HW_GPMI_CTRL1_SET(BM_GPMI_CTRL1_DEV_RESET | // deassert reset
                  BM_GPMI_CTRL1_BCH_MODE  );// enable BCH mode

// enable pinctrl
HW_PINCTRL_CTRL_WR(0x00000000);

// enable GPMI through alt pin wiring
HW_PINCTRL_MUXSEL0_CLR(0xff000000);
HW_PINCTRL_MUXSEL0_SET(0xaa000000);

// to use the primary pins do the following
//    HW_PINCTRL_MUXSEL4_CLR(0xff000000);
//    HW_PINCTRL_MUXSEL4_SET(0x55000000);

// enable gpmi pins
HW_PINCTRL_MUXSEL0_CLR(0x0000ffff);// data bits
HW_PINCTRL_MUXSEL1_CLR(0x000fffff);// control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The BCH engine is used for writing to the NAND but may optionally produce an interrupt. From the sample code in Section 15.4.1.1, “DMA Structure Code Example”:

Table 15-27. HW_BCH_FLASH2LAYOUT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**NBLOCKS**	RW	0x07	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23:16	**META_SIZE**	RW	0x0A	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15:12	**ECC0**	RW	0x8	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. NONE = 0x0 No ECC to be performed ECC2 = 0x1 ECC 2 to be performed ECC4 = 0x2 ECC 4 to be performed ECC6 = 0x3 ECC 6 to be performed ECC8 = 0x4 ECC 8 to be performed ECC10 = 0x5 ECC 10 to be performed ECC12 = 0x6 ECC 12 to be performed ECC14 = 0x7 ECC 14 to be performed ECC16 = 0x8 ECC 16 to be performed ECC18 = 0x9 ECC 18 to be performed ECC20 = 0xA ECC 20 to be performed
11:0	**DATA0_SIZE**	RW	0x200	Indicates the size of the data 0 block (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. If set to zero, the first block will only contain metadata.

DESCRIPTION:

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE:

```
HW_BCH_FLASH2LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH2LAYOUT1_WR(0x04408200);
```

HW_TIMROT_TIMCTRL2_CLR	0x068
HW_TIMROT_TIMCTRL2_TOG	0x06C

Table 22-15. HW_TIMROT_TIMCTRL2

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
RSRVD2																IRQ	IRQ_EN	RSRVD1					POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			

Table 22-16. HW_TIMROT_TIMCTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
15	**IRQ**	RW	0x0	This bit is set to one when Timer 2 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14	**IRQ_EN**	RW	0x0	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13:9	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
8	**POLARITY**	RW	0x0	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7	**UPDATE**	RW	0x0	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6	**RELOAD**	RW	0x0	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.

If I^{2}C interrupts are enabled, a level-sensitive interrupt will be signaled to the processor upon one of the events listed in Table 25-1.

Table 25-1. I^{2}C Interrupt Condition in HW_I2C_CTRL1

SOURCE	BIT NAME	DESCRIPTION
Slave Address	SLAVE_IRQ	This interrupt is generated when an address match occurs. It indicates that the CPU should read the captured RW bit from the I^{2}C address byte to determine the type of DMA to use for the data transfer phase.
Slave Stop	SLAVE_STOP_IRQ	This interrupt is generated when a stop condition is detected after a slave address has been matched.
Oversize Xfer	OVERSIZE_XFER_TERM_IRQ	The DMA and I^{2}C controller are initialized for an expected transfer size. If the data phase is not terminated within this transfer size then oversize transfer processing goes into effect and the CPU is alerted via this interrupt.
Early Termination	EARLY_TERM_IRQ	The DMA and I^{2}C controller are initialized for an expected transfer size. If the data phase is terminated before this transfer size then early termination processing goes into effect and the CPU is alerted via this interrupt.
Master Loss	MASTER_LOSS_IRQ	A master begins transmission on an idle I^{2}C bus and monitors the data line. If it ever attempts to send a one on the line and notes that a zero has been sent instead, then it notes that it has lost mastership of the I^{2}C bus. It terminates its transfer and reports the condition to the CPU via this interrupt. This detection only happens on master transmit operations.
No Slave Ack	NO_SLAVE_ACK_IRQ	When a start condition is transmitted in master mode, the next byte contains an address for a targeted slave. If the targeted slave does not acknowledge the address byte, then this interrupt is set, no further I^{2}C protocol is processed, and the I^{2}C bus returns to the idle state.
Data Engine Complete	DATA_ENGINE_CMPLT_IRQ	This bit is set whenever the DMA interface state machine completes a transaction and resets its run bit. This is useful for PIO mode transmit transactions that are not mediated by the DMA and therefore cannot use the DMA command completion interrupt. This bit is still set for master completions when the DMA is used, but can be ignored in that case.
Bus Free	BUS_FREE_IRQ	When bus mastership is lost during the I^{2}C arbitration phase, the bus becomes busy running services for another master. This interrupt is set whenever a stop command is detected so the master transaction can attempt a retry.

The interrupt lines are tied directly to the bits of Control Register 1. Clearing these bits through software removes the interrupt request.

25.2.2 I^{2}C Bus Protocol

The I^{2}C interface operates as shown in Figure 25-2 and Figure 25-3.

- A START condition is defined as a high-to-low transition on the data line while the I2C_SCL line is held high.
- After this has been transmitted by the master, the bus is considered busy.
- The next byte of data transmitted after the start condition contains the address of the slave in the first seven bits, and the eighth bit tells whether the Master is receiving data from the slave or transmitting data to the slave.

To receive one data byte from a slave device such as an FM tuner, the following bus transaction takes place.

Table 25-4. I²C Transfer "FM Tuner" Read of One Byte

ST	SAD+R	SAK	DATA	NMAK	SP

In this transaction:

- The master first generates a start condition, ST.
- It then sends the seven-bit slave address for the FM tuner plus a read bit (SAD+R).
- The slave in the FM tuner responds with a slave acknowledge bit (SAK).
- The master then generates I²C clocks for a data byte to be transferred (DATA).
- The slave provides data to the I²C data bus during the DATA byte transfer.
- Next, the master generates a master non-acknowledge to the slave (NMAK), indicating the end of the data transfer to the slave. The slave will then release the data line.
- Finally, the master generates a stop condition (SP), terminating the transaction and freeing the I²C bus for other masters to use.

The following example shows a multiple byte read from an FM tuner or other slave device:

Table 25-5. I²C Transfer "FM Tuner" Read of Three Bytes

ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

25.2.2.2 Typical EEPROM Transactions

I²C EEPROMs typically have a specific transaction sequence for reading and writing data bytes to and from the EEPROM array. Table 25-6 through Table 25-9 show the first two bytes of data as a sub-address for purposes of illustration. The sub-address is used to address the memory space inside the device. Table 25-3 defines each element of the transactions shown. When writing a single byte of data to the EEPROM, one must first transfer two bytes of sub-address as follows:

Table 25-6. I²C Transfer When Master is Writing One Byte of Data to a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	SP

The sub-address only needs to be specified once for a multibyte transfer, as shown here. Note that the sub-address must be sent for each start condition that initiates a transaction.

Table 25-7. I²C Transfer When Master is Writing Multiple Bytes to a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	DATA	SAK	SP

One must also provide the sub-address before reading bytes from the EEPROM. The sub-address is transmitted from the master to the slave before it can receive data bytes. The two transfers are joined into a single bus transaction though the use of a repeated start condition (SR). Normally, a stop condition precedes a start condition. However, when a start condition is preceded by another start condition, it is

Table 33-33. HW_LRADC_DELAY3

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
TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_DELAYS				LOOP_COUNT					DELAY										

Table 33-34. HW_LRADC_DELAY3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TRIGGER_LRADCS**	RW	0x00	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23:21	**RSRVD2**	RO	0x0	Reserved
20	**KICK**	RW	0x0	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19:16	**TRIGGER_DELAYS**	RW	0x0	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15:11	**LOOP_COUNT**	RW	0x00	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels. ERRATA: TA1 and TA2 silicon revisions do not correctly support the LOOP_COUNT field, do not use.
10:0	**DELAY**	RW	0x000	This 11-bit field counts down to zero. At zero it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single even. This counter operates on a 2KHz clock derived from crystal clock.

If a gate GPIO is used, then the driver will use the SD_POWER_UP_DELAY eFuse to determine the amount of time, in 10-ms increments, to wait until starting the 1-ms initialization sequence. This eFuse field is 6-bits wide, providing from 10–600 ms of delay. If the field is 000000b, then the delay is a default 20 ms. If no gate GPIO is specified in SD_POWER_GATE_GPIO, then the delay is skipped.

The SSP ports on the i.MX23 top out at 50 MHz with 20–40 pF loading. By default, the serial clock is set to 12 MHz. If the SD_SPEED_ENABLE persistent bit is set, then the driver will use a maximum speed based on the results of device identification and limited by choices available in the SSP clock index.

This mode supports the 1-bit, 4-bit, and 8-bit data MMC/SD buses. The SD_BUS_WIDTH efuse bits selects how many bus pins are physically available for the SSP port. Bus width will be limited based on these bits, as well as the bus width capabilities indicated by the connected device.

Table 35-10. Bus Pin Selection

SD_BUS_WIDTH	Width
00b	4-bit
01b	1-bit
10b	8-bit
11b	Reserved

The SD/MMC boot mode requires either a Boot Control Block (BCB) or Master Boot Record (MBR) on the device. The boot loader will first search for a MBR. If found, it will use the MBR data to find the boot image. If the MBR is not present, the boot loader will search for a BCB. If found, the BCB will provide the boot image located and size to the boot loader. If neither structure is fount, the boot loader will return an error to the ROM.

35.7.1 Boot Control Block (BCB)

The last physical sector of the device contains a media configuration block. This block contains the sector address of the boot image. The config block has the following format:

```
typedef struct {
    drive_type_t    eDriveType;
    uint32_t        Tag;
    uint32_t        Reserved[5];
} media_regions_t;

typedef struct {
    uint32_t        Signature;
    uint32_t        Version
    uint32_t        Reserved[1];
    uint32_t        NumRegions;
    media_regions_t Regions[];
} media_config_block_t;
```

The driver will first verify the Signature and Version, then search all NumRegions for the appropriate Tag. Table 35-11 shows the expected values for these parameters.

Table 37-16. HW_PINCTRL_MUXSEL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:14	**BANK2_PIN07**	RW	0x3	Pin 37, ROTARYA pin function selection: 00= timrot1; 01= auart2_rts; 10= spdif; 11= GPIO.
13:12	**BANK2_PIN06**	RW	0x3	Pin 127, SSP1_SCK pin function selection: 00= ssp1_sck; 01= reserved; 10= alt_jtag_trst_n; 11= GPIO.
11:10	**BANK2_PIN05**	RW	0x3	Pin 125, SSP1_DATA3 pin function selection: 00= ssp1_d3; 01= reserved; 10= alt_jtag_tms; 11= GPIO.
9:8	**BANK2_PIN04**	RW	0x3	Pin 124, SSP1_DATA2 pin function selection: 00= ssp1_d2; 01= i2c_sd; 10= alt_jtag_rtck; 11= GPIO.
7:6	**BANK2_PIN03**	RW	0x3	Pin 123, SSP1_DATA1 pin function selection: 00= ssp1_d1; 01= i2c_clk; 10= alt_jtag_tck; 11= GPIO.
5:4	**BANK2_PIN02**	RW	0x3	Pin 122, SSP1_DATA0 pin function selection: 00= ssp1_d0; 01= reserved; 10= alt_jtag_tdi; 11= GPIO.
3:2	**BANK2_PIN01**	RW	0x3	Pin 126, SSP1_DETECT pin function selection: 00= ssp1_detect; 01= gpmi_ce3n; 10= usb_id; 11= GPIO.
1:0	**BANK2_PIN00**	RW	0x3	Pin 121, SSP1_CMD pin function selection: 00= ssp1_cmd; 01= reserved; 10= alt_jtag_tdo; 11= GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

EXAMPLE:

```
Empty Example.
```

37.4.7 PINCTRL Pin Mux Select Register 5 Description

The PINCTRL Pin Mux Select Register provides pin function selection for 16 pins in bank2.

Table B-1. Register Names and Addresses (continued)

Register Name	Address
HW_ICOLL_INTERRUPT79_CLR	0x80000618
HW_ICOLL_INTERRUPT79_SET	0x80000614
HW_ICOLL_INTERRUPT79_TOG	0x8000061C
HW_ICOLL_INTERRUPT8	0x800001A0
HW_ICOLL_INTERRUPT8_CLR	0x800001A8
HW_ICOLL_INTERRUPT8_SET	0x800001A4
HW_ICOLL_INTERRUPT8_TOG	0x800001AC
HW_ICOLL_INTERRUPT80	0x80000620
HW_ICOLL_INTERRUPT80_CLR	0x80000628
HW_ICOLL_INTERRUPT80_SET	0x80000624
HW_ICOLL_INTERRUPT80_TOG	0x8000062C
HW_ICOLL_INTERRUPT81	0x80000630
HW_ICOLL_INTERRUPT81_CLR	0x80000638
HW_ICOLL_INTERRUPT81_SET	0x80000634
HW_ICOLL_INTERRUPT81_TOG	0x8000063C
HW_ICOLL_INTERRUPT82	0x80000640
HW_ICOLL_INTERRUPT82_CLR	0x80000648
HW_ICOLL_INTERRUPT82_SET	0x80000644
HW_ICOLL_INTERRUPT82_TOG	0x8000064C
HW_ICOLL_INTERRUPT83	0x80000650
HW_ICOLL_INTERRUPT83_CLR	0x80000658
HW_ICOLL_INTERRUPT83_SET	0x80000654
HW_ICOLL_INTERRUPT83_TOG	0x8000065C
HW_ICOLL_INTERRUPT84	0x80000660
HW_ICOLL_INTERRUPT84_CLR	0x80000668
HW_ICOLL_INTERRUPT84_SET	0x80000664
HW_ICOLL_INTERRUPT84_TOG	0x8000066C
HW_ICOLL_INTERRUPT85	0x80000670
HW_ICOLL_INTERRUPT85_CLR	0x80000678
HW_ICOLL_INTERRUPT85_SET	0x80000674
HW_ICOLL_INTERRUPT85_TOG	0x8000067C
HW_ICOLL_INTERRUPT86	0x80000680
HW_ICOLL_INTERRUPT86_CLR	0x80000688