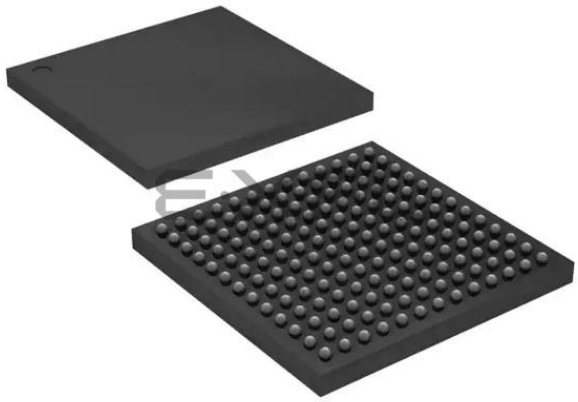

Welcome to E-XFL.COM

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	-10°C ~ 70°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/mcimx233djm4br2

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

http://www.freescale.com/support

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: IMX23RM
Rev. 1, 11/2009

Contents

Contents

Chapter 33
Low-Resolution ADC and Touch-Screen Interface

Figures

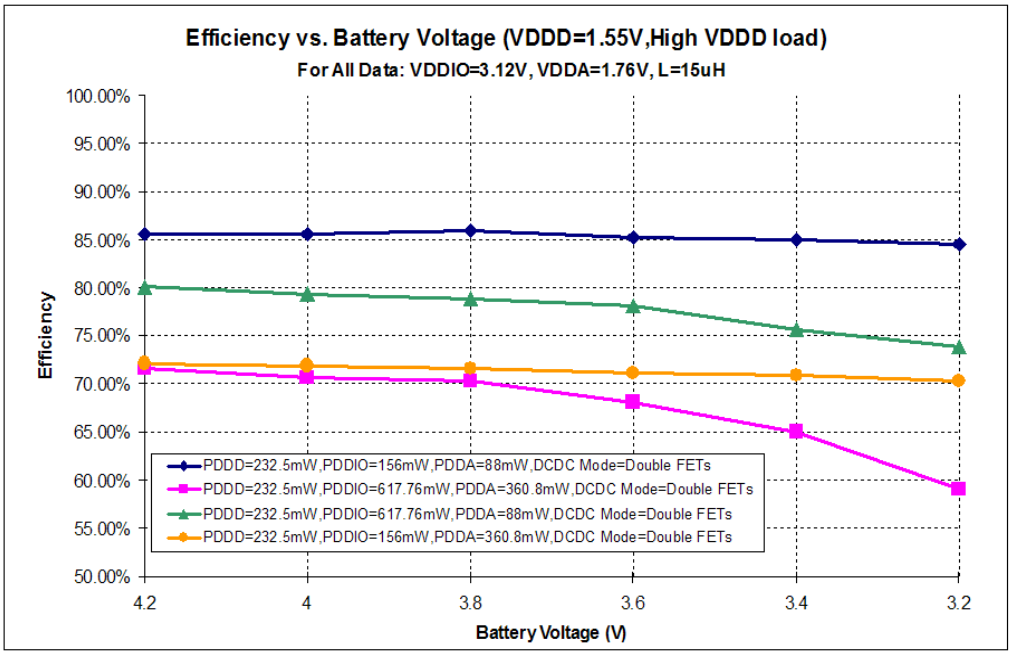

Figure 2-3. DCDC Efficiency vs. Battery Voltage (VDDD=1.55V, High VDDD Load)

Table 2-8. Non-EMI Digital Pin DC Characteristics

		VDDIO = 3.3 V		
Parameter	**Name**	**Min**	**Max**	**Units**
Non-EMI Regular & High Drive I/O Input Voltage	VIH	2.00	VDDIO	V
	VIL	-	0.80	V
Non-EMI Regular & High Drive I/O Output Voltage	VOH	0.8 * VDDIO	-	V
	VOL	-	0.40	V
Non-EMI Regular I/O Output Current (see notes 1 and 6 of Table 2-10)	IOH - 4mA	3.60	-	mA
	IOH - 8mA	7.20	-	mA
	IOH - 12mA	10.80	-	mA
	IOL - 4mA	-3.60	-	mA
	IOL - 8mA	-7.20	-	mA
	IOL - 12mA	-10.80	-	mA
Non-EMI High Drive I/O (PWM4) Output Current (see notes 2 and 6 of Table 2-10)	IOH - 8mA	-6.50	-	mA
	IOH - 16mA	-11.00	-	mA
	IOH - 24mA	-16.80	-	mA
	IOL - 8mA	-8.00	-	mA
	IOL - 16mA	-14.50	-	mA
	IOL - 24mA	-19.00	-	mA
External Pull-Up / Pull-Down Resistor Value Required to Overdrive Internal Gate Keeper		-	50	kΩ
Internal Pull-Up Resistor Accuracy	-20	-	+20	%

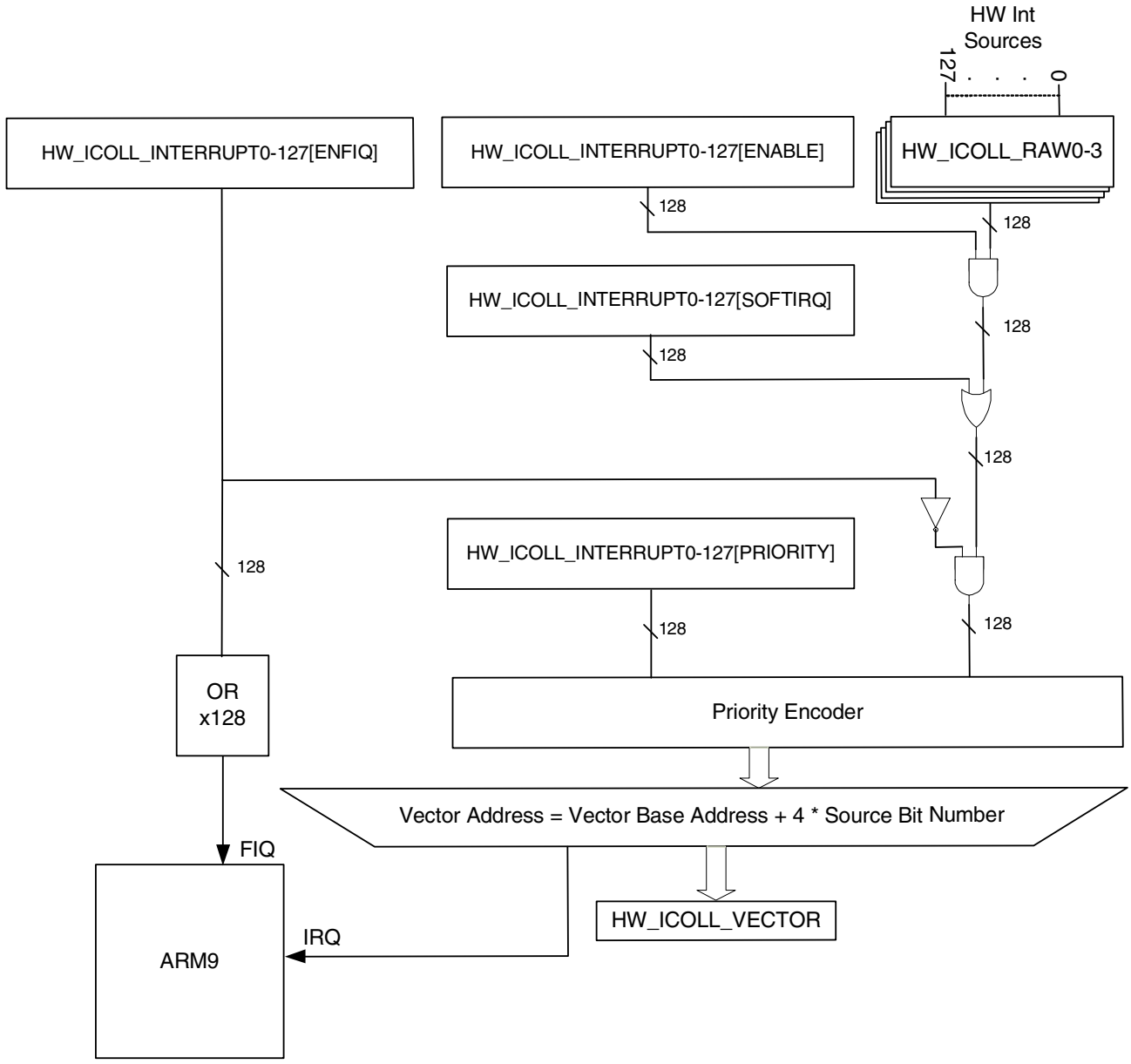

Figure 5-1. Interrupt Collector System Diagram

5.2 Operation

Within an individual interrupt request line (IRQ only), the ICOLL offers four-level priority (above base level) for each of its interrupt sources. Preemption of a lower priority interrupt by a higher priority is supported (interrupt nesting). Interrupts assigned to the same level are serviced in a strict linear priority order within level from lowest to highest interrupt source bit number. FIQ interrupts are not prioritized, nor are they vectorized. All interrupt lines can be configured as a FIQ. If more than one is routed to the FIQ, then they must be discriminated by software. It is highly recommended to reserve FIQ assignment to time critical events such as voltage brownouts or timers.

Table 5-155. HW_ICOLL_INTERRUPT67 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_INTERRUPT67_SET(0,0x00000001);
```

5.4.78 Interrupt Collector Interrupt Register 68 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_INTERRUPT68	0x560
HW_ICOLL_INTERRUPT68_SET	0x564
HW_ICOLL_INTERRUPT68_CLR	0x568
HW_ICOLL_INTERRUPT68_TOG	0x56C

Table 5-156. HW_ICOLL_INTERRUPT68

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 6-21. HW_DIGCTL_SJTAGDBG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD2**	RO	0x0	Reserved.
26:16	**SJTAG_STATE**	RO	0x2	This bitfield shows the state of the sjtag_state flip-flops inside the SJTAG controller. These bits implement the second state machine in the SJTAG block.
15:11	**RSVD1**	RO	0x0	Reserved.
10	**SJTAG_TDO**	RO	0x0	This bit shows the state of the ARM JTAG TDO signal as seen inside the SJTAG controller.
9	**SJTAG_TDI**	RO	0x0	This bit shows the state of the JTAG TDI capture FF inside the SJTAG controller.
8	**SJTAG_MODE**	RO	0x0	This bit shows the state of the JTAG mode capture FF inside the SJTAG controller.
7:4	**DELAYED_ACTIVE**	RO	0x0	This bitfield shows the state of the delay_onewire_active_reg FF inside the SJTAG controller. These bits implement the first state machine in the SJTAG block.
3	**ACTIVE**	RO	0x0	This bit shows the state of the onewire_active_reg FF inside the SJTAG controller.
2	**SJTAG_PIN_STATE**	RO	0x0	This bit reflects the state of the input driver sampling the SJTAG pin. When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, external source can pull the SJTAG pin high without starting the SJTAG state machines. In this mode, the SJTAG_PIN_STATE bit is used to confirm continuity from the pad to the SJTAG block.
1	**SJTAG_DEBUG_DATA**	RW	0x0	When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, then the SJTAG pin is placed in a diagnostic mode. In that case, this bit controls the input to the pad data drive signal. If HW_DIGCTL_CTRL_SJTAG_DEBUG_OE is set to 1, then this bit also controls the state of the SJTAG pin itself.
0	**SJTAG_DEBUG_OE**	RW	0x0	When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, then the SJTAG pin is placed in a diagnostic mode. In that case, this bit controls the input to the pad data output enable signal. Setting this bit to 1 turns on the SJTAG pad and drives it to the state indicated by HW_DIGCTL_CTRL_SJTAG_DEBUG_DATA.

6.4.11 Digital Control Microseconds Counter Register Description

The Digital Control Microseconds Counter Register is a read-only test value register.

HW_DIGCTL_MICROSECONDS	0x0C0
HW_DIGCTL_MICROSECONDS_SET	0x0C4
HW_DIGCTL_MICROSECONDS_CLR	0x0C8
HW_DIGCTL_MICROSECONDS_TOG	0x0CC

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_L3_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L3_AHB_ACTIVE_CYCLES_COUNT_RD();
```

6.4.47 AHB Layer 3 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_L3_AHB_DATA_STALLED 0x3E0

Table 6-94. HW_DIGCTL_L3_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 6-95. HW_DIGCTL_L3_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.

DESCRIPTION:

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_L3_AHB_DATA_STALLED_COUNT_RD();
```

6.4.48 AHB Layer 3 Performance Metric for Valid Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

HW_DIGCTL_L3_AHB_DATA_CYCLES 0x3F0

EXAMPLE:

```
Empty example.
```

10.5.27 APBH DMA Channel 3 Next Command Address Register Description

The APBH DMA Channel 3 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_APBH_CH3_NXTCMDAR 0x1A0

Table 10-56. HW_APBH_CH3_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 10-57. HW_APBH_CH3_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 3.

DESCRIPTION:

APBH DMA Channel 3 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 3 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.28 APBH DMA Channel 3 Command Register Description

The APBH DMA Channel 3 command register specifies the cycle to perform for the current command chain item.

HW_APBH_CH3_CMD 0x1B0

Table 10-58. HW_APBH_CH3_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	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 10-107. HW_APBH_CH6_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 6 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 6.

EXAMPLE:

```
Empty example.
```

10.5.53 AHB to APBH DMA Channel 6 Debug Information Description

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

HW_APBH_CH6_DEBUG2 0x340

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel will wait for the device to signal completion of a command by toggling the apx_endcmcd signal before proceeding to load and execute the next command structure. The semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in Table 11-3, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer count mechanism is duplicated in the associated peripheral, either as an implied or specified count in the peripheral.

Table 11-3. DMA Channel Command Word in System Memory

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
NEXT_COMMAND_ADDRESS																															
Number DMA Bytes to Transfer																Number PIO Words to Write				Reserved				WAIT4ENDCMD	DECREMENT SEMAPHORE	Reserved		IRQ_COMPLETE	CHAIN	COMMAND			
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBX Bus																															

Figure 11-3 shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1 if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it will not be detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by software. It can be used to interrupt the CPU.

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands and DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by software. When the semaphore goes to

There are four counters in all to cover the five low-power modes. There are separate counters for each of the three memory self-refresh low-power modes (Modes 3, 4 and 5). Memory Power-Down mode (Mode 1) and Memory Power-Down with Memory Clock Gating mode (Mode 2) share the same counter.

The counters determine the number of idle cycles before entry into the associated low-power mode. All of these counters are re-initialized each time there is a new read or write transaction entering or executing in the memory controller. This ensures that the memory controller does not enter any of the low-power modes when active.

All five low-power modes can be entered through automatic entry and are exited automatically when any of the following conditions occur:

- A new read or write transaction appears at the memory controller interface.
- The memory controller must refresh the memory when in either of the power-down modes (Modes 1 or 2). After completing the memory refresh, the memory controller re-enters power-down.
- The counter for a deeper low-power mode expires. The memory controller must exit the current low-power mode in order to enter the deeper low-power mode. A minimum of 15 cycles occur between exit from one low-power mode before entering into the next low-power mode, even if the counters expire within 15 cycles of each other. Note that the memory controller does not enter a less deep low-power mode, regardless of which counters expire.

12.2.6.4 Manual “On-Demand” Entry

Manual entry occurs if all of the following conditions are true:

- The hardware entry interface is not active or transitioning.
- The mode is programmed for manual entry by clearing the relevant bit in the LOWPOWER_AUTO_ENABLE bit field to 0.
- The particular mode is set to 1 in the LOWPOWER_CONTROL bit field.

For manual entry, the LOWPOWER_CONTROL bit field triggers entry into the low-power modes. The memory controller does not need to be idle when a low-power mode bit is enabled. When a particular mode that is programmed for manual entry is enabled, the memory controller completes the current memory burst access, and then, regardless of the activity inside the memory controller or at the memory interface, it enters the selected low-power mode.

If new transactions enter the memory controller while it is in one of the low-power modes, they accumulate inside the memory controller’s command queue until the queue is full. Exit from a manually-entered low-power mode is also manual. Clearing the LOWPOWER_CONTROL bit field bits to 0 disables the low-power mode of the memory controller, and command processing resumes. In the deepest low-power mode (Mode 5), the clock to the programming registers module is gated off. However, manual low-power mode exit requires the user to clear the LOWPOWER_CONTROL bit field to 0, which is not possible when the clock is off. As a result, the user should not manually activate the deepest low-power mode. If Memory Self-Refresh with Memory and Controller Clock Gating mode (Mode 5) is entered manually, the device cannot be brought out of low-power mode again!

desired decoded data addresses. To initiate a decode, software must set the M2M_ENCODE bit to 0 while writing the M2M_ENABLE bit.

15.4 Programming the BCH/GPMI Interfaces

Programming the BCH for NAND operations consists largely of disabling the soft reset and clock bits (SFTRST and CLKGATE) from the HW_BCH_CTRL register and then programming the flash layout registers to correspond to the format of the attached NAND device(s). The HW_BCH_LAYOUTSELECT register should also be programmed to map the chip select of each attached device into one of the four layout registers.

The bulk of the programming is actually applied to the GPMI via PIO operations embedded in DMA command structures. The DMA will perform all the requisite handshaking with the GPMI interface to negotiate the address portion of the transfer, then the BCH will handle all the movement of data from memory to the GPMI (writes) or the GPMI to memory (reads). The BCH will direct all data blocks to the buffer pointed to by the PAYLOAD_BUFFER and the metadata will be written to the AUXILIARY_BUFFER. Both of these registers are located in the GPMI PIO data space and are communicated to the BCH hardware at the beginning of the transfer. Thus, the normal multi-NAND DMA based device interleaving is preserved, i.e., four NANDs on four separate chip selects can be scheduled for read or write operations using the BCH. Whichever channel finishes its ready wait first and enters the DMA arbiter with its lock bit set will “own” the GPMI command interface and through it will own the BCH resources for the duration of its processing.

15.4.1 BCH Encoding for NAND Writes

The BCH encoder flowchart in Figure 15-6 shows the detailed steps involved in programming and using the BCH encoder. This flowchart shows how to use the BCH block with the GPMI.

To use the BCH encoder with the GPMI’s DMA, create a DMA command chain containing nine descriptor structures, as shown in Figure 15-8 and detailed in the DMA structure code example that follows it in Section 15.4.1.1, “DMA Structure Code Example.” The nine descriptors perform the following tasks:

1. Disable the BCH block (in case it was enabled) and issue NAND write setup command byte (under “CLE”) and address bytes (under “ALE”).
2. Configure and enable the BCH and GPMI blocks to perform the NAND write.
3. Disable the BCH block and issue NAND write execute command byte (under “CLE”).
4. Wait for the NAND device to finish writing the data by watching the ready signal.
5. Check for NAND timeout via “PSENSE”.
6. Issue NAND status command byte (under “CLE”).
7. Read the status and compare against expected.
8. If status is incorrect/incomplete, branch to error handling descriptor chain.
9. Otherwise, write is complete and emit GPMI interrupt.

15.6.9 Hardware BCH ECC Flash 0 Layout 1 Register Description

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH0LAYOUT0 register to control the format for the device on chip select 0.

HW_BCH_FLASH0LAYOUT1 0x090

Table 15-20. HW_BCH_FLASH0LAYOUT1

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
PAGE_SIZE																ECCN				DATAN_SIZE											

Table 15-21. HW_BCH_FLASH0LAYOUT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**PAGE_SIZE**	RW	0x10DA	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15:12	**ECCN**	RW	0x8	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 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	**DATAN_SIZE**	RW	0x200	Indicates the size of the subsequent data blocks (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

EXAMPLE:

```
HW_BCH_FLASH0LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH0LAYOUT1_WR(0x04408200);
```

15.6.10 Hardware BCH ECC Flash 1 Layout 0 Register Description

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH0LAYOUT1 register to control the format for the devices selecting layout 0 in the LAYOUTSELECT register.

Table 17-9. Register Use for Conversion (continued)

Register	Value	Description
S0PARAM	0x0000503C	WIDTH=0x50=80=640 pixels HEIGHT=0x3C=60=480 pixels
S0BACKGROUND	0x00000040	Dark Blue background region
S0CROP	0x0000281E	WIDTH=0x28=320 pixels HEIGHT=0x1E=240 pixels
S0SCALE	0x20002000	YSCALE=0x2000=1/2x XSCALE=0x1555=1/2x
S0OFFSET	0x08000800	XOFFSET=0x0800 (1/2 pixel) YOFFSET=0x0800 (1/2 pixel)
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*prev_rgb	Pointer to “previous” graphic
OL0SIZE	0x0B1B0402	XBASE=0x0B=88pixels YBASE=0x1B=216pixels WIDTH=0x04=32 pixels HEIGHT=0x02=16pixels
OL0PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL1	*next_rgb	Pointer to “next” graphic
OL1SIZE	0x2D1B0402	XBASE=0x2D=360pixels YBASE=0x1B=216pixels WIDTH=0x04=32 pixels HEIGHT=0x02=16pixels
OL1PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL2	*text_overlay	Pointer to text graphic
OL2SIZE	0x00000A1E	XBASE=0x00=0pixels YBASE=0x00=0pixels WIDTH=0x0A=80pixels HEIGHT=0x1E=240pixels
OL2PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL3	*border_rgb	Pointer to rectangular border graphic
OL3SIZE	0x0A00281E	XBASE=0x0A=80pixels YBASE=0x00=0pixels WIDTH=0x28=320pixels HEIGHT=0x1E=240pixels
OL3PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL4PARAM	0x00000000	Overlay 4 disabled

EXAMPLE:

```
No Example.
```

27.3.13 UART Interrupt Clear Register Description

The ICR register is the Interrupt Clear Register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

HW_UARTDBGICR 0x044

Table 27-26. HW_UARTDBGICR

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					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC

Table 27-27. HW_UARTDBGICR 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	**OEIC**	WO	0x0	Overrun Error Interrupt Clear.
9	**BEIC**	WO	0x0	Break Error Interrupt Clear.
8	**PEIC**	WO	0x0	Parity Error Interrupt Clear.
7	**FEIC**	WO	0x0	Framing Error Interrupt Clear.
6	**RTIC**	WO	0x0	Receive Timeout Interrupt Clear.
5	**TXIC**	WO	0x0	Transmit Interrupt Clear.
4	**RXIC**	WO	0x0	Receive Interrupt Clear.
3	**DSRMIC**	WO	0x0	nUARTDSR Modem Interrupt Clear.
2	**DCDMIC**	WO	0x0	nUARTDCD Modem Interrupt Clear.
1	**CTSMIC**	WO	0x0	nUARTCTS Modem Interrupt Clear.
0	**RIMIC**	WO	0x0	nUARTRI Modem Interrupt Clear.

Table 28-8. HW_AUDIOIN_ADCVOLUME

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
RSRVD5			VOLUME_UPDATE_LEFT	RSRVD4		EN_ZCD	RSRVD3	VOLUME_LEFT								RSRVD2			VOLUME_UPDATE_RIGHT	RSRVD1				VOLUME_RIGHT							

Table 28-9. HW_AUDIOIN_ADCVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSRVD5**	RO	0x00	Reserved
28	**VOLUME_UPDATE_LEFT**	RO	0x0	Left Channel Volume Update Pending. This bit is set to one by the hardware when an AUDIOIN volume update is pending, i.e., waiting on a zero crossing on the left channel. The bit is set following a write to the VOLUME_LEFT bit field and is cleared when the attenuation value is applied to the PCM input stream (at a zero-crossing). This status bit is not used when EN_ZCD=0.
27:26	**RSRVD4**	RO	0x00	Reserved
25	**EN_ZCD**	RW	0x0	Enable Zero Cross Detect. This bit enables/disables use of the zero cross detect circuit in the ADC (rather than enabling the circuit itself). When enabled, changes to the volume bit fields are not applied until it is detected that the input signal's sign bit toggles (crosses zero amplitude). When disabled, changes to the volume bit fields take effect immediately when written.
24	**RSRVD3**	RO	0x0	Reserved
23:16	**VOLUME_LEFT**	RW	0xfe	Left Channel Volume Setting. This bit field is used to establish the incoming audio signal strength during record. Volume ranges from -0.5 dB (0xFE) to -100 dB (0x37). Each increment of this bit field causes a half-dB increase in volume. Note that values 0x00-0x37 all produce the same attenuation level of -100 dB. Also note that a setting of 0xFF is reserved.
15:13	**RSRVD2**	RO	0x00	Reserved
12	**VOLUME_UPDATE_RIGHT**	RO	0x0	Right Channel Volume Update Pending. This bit is set to one by the hardware when an AUDIOIN volume update is pending, i.e., waiting on a zero crossing on the right channel. The bit is set following a write to the VOLUME_RIGHT bit field and is cleared when the attenuation value is applied to the PCM input stream (at a zero-crossing). This status bit is not used when EN_ZCD=0.

Table 32-5. HW_POWER_5VCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
10:8	**VBUSVALID_TRSH**	RW	0x0	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000: 2.9V 001: 4.0V 010: 4.1V 011: 4.2V 100= 4.3V 101: 4.4V 110: 4.5V 111: 4.6V
7	**PWDN_5VBRNOUT**	RW	0x1	The purpose of this bit is to power down the device if 5V is removed before the system is completely initialized. Clear this bit to disable automatic hardware powerdown AFTER the system is configured for 5v removal. This bit should not be set if DCDC_XFER is set.
6	**ENABLE_LINREG_ILIMIT**	RW	0x0	Enable the current limit in the linear regulators. The current limit is active during powerup from 5v and automatically disables before the ROM executes. This limit is needed to meet the USB in rush current specification of 100mA + 50uC. Note this linear regulator current limit is not related to the CHARGE_4P2_ILIMIT.
5	**DCDC_XFER**	RW	0x0	Enable automatic transition to switching DC-DC converter when VDD5V is removed. The LRADC must be operational and the BATT_VAL field must be written with the battery voltage using 8-mV step-size. It is also important to set the EN_BATADJ field.
4	**VBUSVALID_5VDETECT**	RW	0x0	Power up and use VBUSVALID comparator as detection circuit for 5V in the switching converter. Default is for the switching converter to use the VDD5V_GT_VDDIO status bit to determine the presence of 5V in the system. The VBUSVALID comparator provides a more accurate and adjustable threshold to determine the presence of 5V in the system, and is the recommended method of detecting 5V.
3	**VBUSVALID_TO_B**	RW	0x0	This bit muxes the Bvalid comparator to the VBUSVALID comparator and is used for test purposes only.
2	**ILIMIT_EQ_ZERO**	RW	0x0	The amount of current the device will consume from the 5V rail is minimized. The VDDIO linear regulator current limit is set to zero mA. Also, the source of current for the crystal oscillator and RTC is switched to the battery. Note that this functionality does not affect battery charge.

Table 33-6. HW_LRADC_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13	**EXT_EN1**	RW	0x0	These bits are not supported. DISABLE = 0x0 . ENABLE = 0x1 .
12	**EXT_EN0**	RW	0x0	When set to zero(default) the mux amp is bypassed when the LRADC input channel is not using the divide-by-two. When set to one the mux amp is never bypassed (old behavior).
11:10	**RSRVD2**	RO	0x00	Reserved
9	**TEMP_SENSOR_IENABLE1**	RW	0x0	Set this bit to one to enable the current source onto LRADC1. DISABLE = 0x0 Disable Temperature Sensor Current Source. ENABLE = 0x1 Enable Temperature Sensor Current Source.
8	**TEMP_SENSOR_IENABLE0**	RW	0x0	Set this bit to one to enable the current source onto LRADC0. DISABLE = 0x0 Disable Temperature Sensor Current Source. ENABLE = 0x1 Enable Temperature Sensor Current Source.
7:4	**TEMP_ISRC1**	RW	0x0	When the output voltage is lower than 1V the output current is 1uA higher than the decode shown above. This extra current drops to zero as the output voltage raises above 1.5V. This four bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC1. 300 = 0xF 300uA. 280 = 0xE 280uA. 260 = 0xD 260uA. 240 = 0xC 240uA. 220 = 0xB 220uA. 200 = 0xA 200uA. 180 = 0x9 180uA. 160 = 0x8 160uA. 140 = 0x7 140uA. 120 = 0x6 120uA. 100 = 0x5 100uA. 80 = 0x4 80uA. 60 = 0x3 60uA. 40 = 0x2 40uA. 20 = 0x1 20uA. ZERO = 0x0 0uA.
3:0	**TEMP_ISRC0**	RW	0x0	When the output voltage is lower than 1V the output current is 1uA higher than the decode shown above. This extra current drops to zero as the output voltage raises above 1.5V. This four bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC0. 300 = 0xF 300uA. 280 = 0xE 280uA. 260 = 0xD 260uA. 240 = 0xC 240uA. 220 = 0xB 220uA. 200 = 0xA 200uA. 180 = 0x9 180uA. 160 = 0x8 160uA. 140 = 0x7 140uA. 120 = 0x6 120uA. 100 = 0x5 100uA. 80 = 0x4 80uA. 60 = 0x3 60uA. 40 = 0x2 40uA. 20 = 0x1 20uA. ZERO = 0x0 0uA.