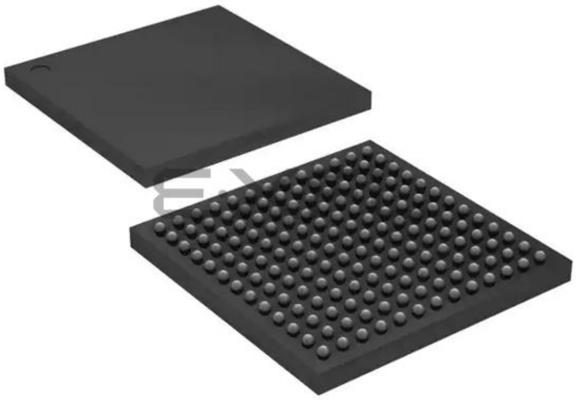
NXP USA Inc. - MCIMX233DJM4CR2 Datasheet



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	-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/mcimx233djm4cr2

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

Chapter 2
Characteristics and Specifications

This chapter describes the characteristics and specifications of the i.MX23 and includes sections on absolute maximum ratings, recommended operating conditions, and DC characteristics.

2.1 Absolute Maximum Ratings

Table 2-1. Absolute Maximum Ratings

Parameter	Min	Max	Units
Storage Temperature	–40	125	°C
Battery Pin - BATT, VDD4P2V	–0.3	4.242	V
5-Volt Source Pin - VDD5V (transient, t < 30ms, duty cycle < 0.05%)	-0.3	7.00	V
5-Volt Source Pin - VDD5V (static)	–0.3	6.00	V
PSWITCH (Note 1)	–0.3	VDDXTAL + 1.575	V
Analog Supply Voltage—VDDA	–0.3	2.10	V
Speaker Amplifier Supply Voltage—VDDS	–0.3	4.242	V
Digital Core Supply Voltage —VDDD	–0.3	1.575	V
Non-EMI Digital I/O Supply—VDDIO	–0.3	3.63	V
EMI Digital I/O Supply—VDDIO.EMI	–0.3	3.63	V
DC-DC Converter—DCDC_BATT (Note 2)	–0.3	BATT	V
Input Voltage on Any Digital I/O Pin Relative to Ground	–0.3	VDDIO+0.3	V
Input Voltage on USB_DP and USB_DN Pins Relative to Ground (Note 3)	–0.3	3.63	V
Input Voltage on Any Analog I/O Pin Relative to Ground	–0.3	VDDA+0.3	V

[1] VDDIO can be applied to PSWITCH through a 10 kΩ resistor. This is necessary in order to enter the chip's firmware recovery mode. (The on-chip circuitry prevents the actual voltage on the pin from exceeding acceptable levels.)

[2] Application should include a Schottky diode between BATT and VDD4P2.

[3] USB_DN and USB_DP can tolerate 5V for up to 24 hours. Note that while 5V is applied to USB_DN or USB_DP, LRADC readings can be corrupted.

Table 2-2. Electro-Static Discharge Immunity

169-Pin BGA & 128-Pin LQFP Packages	Tested Level
Human Body Model (HBM)	2 kV
Charge Device Model (CDM)	500 V

[4] Max load includes capacitive load due to PCB traces, pad capacitance and driver self-loading.

[5] Setting is for worst case. Freescale's EMI interface uses less powerful drivers than those typically used in mDDR devices. A possible transmission-line effect on the PC board must be suppressed by minimizing the trace length combined with Freescale's slower edge-rate drivers. The i.MX23 provides up to 16 mA programmable drive strength. However, the 16-mA mode is an experimental mode. With the 16-mA mode, the EMI function may be impaired by simultaneous switching output (SSO) noise. In general, the stronger the driver mode, the noisier the on-chip power supply. Freescale recommends not using a stronger driver mode than is required. Because on-chip power and ground noise is proportional to the inductance of its return path, users should make their best effort to reduce inductance between the EMI power and ground balls and the PC board power and ground planes.

[6] IOH is the maximum output current at which the VOH specification is met. IOL is the maximum input current at which the VOL specification is met.

2.3.1 Recommended Operating Conditions for Specific Clock Targets

NOTE

At this time, all data is preliminary and subject to change without notice.

Table 2-11. System Clocks

Name	Min. Freq. (MHz)	Max. Freq. (MHz)	Description
clk_gpmi	-	102.858	General Purpose memory interface clock domain
clk_ssp	-	102.858	Internal SSP Interface clock.
External SSP Clock	-	51.429	External SSP clock.

Table 2-12. Recommended Operating States - 169BGA Package

VDDD (V)	VDDD Brown-out (V)	HW_ DIGCTRL ARMCACHE (note 1)	CPUCLK / clk_p Frequency (MHz)	HW_ CLKCTRL CPU_DIV_CPU	HW_ CLKCTRL FRAC_ CPUFRC / PFD	AHBCLK / clk_h Frequency (MHz)	HW_ CLKCTRL HBUS_DIV	EMICLK / clk_emi Frequency (MHz)	HW_ CLKCTRL EMI_ DIV_EMI	HW_ CLKCTRL FRAC_ EMIFRAC	SUPPORTED DRAM
1.050	0.975		24.00			24.00	1	24.00			DDR, mDDR
1.050	0.975	11	64.00	5	27	64.00	1	64.00	5	27	DDR, mDDR
1.275	1.175	00	261.82	1	33	130.91	2	130.91	2	33	DDR, mDDR
1.375	1.275	00	360.00	1	24	120.00	3	120.00	3	24	DDR, mDDR
1.475	1.375	00	392.73	1	22	196.36	2	130.91	2	33	DDR, mDDR
1.550	1.450	00	454.74	1	19	151.58	3	151.58	3	19	mDDR
1.550	1.450	00	454.74	1	19	151.58	3	130.91	2	33	DDR

Table 5-49. HW_ICOLL_INTERRUPT14 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_INTERRUPT14_SET(0,0x00000001);
```

5.4.25 Interrupt Collector Interrupt Register 15 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_INTERRUPT15	0x210
HW_ICOLL_INTERRUPT15_SET	0x214
HW_ICOLL_INTERRUPT15_CLR	0x218
HW_ICOLL_INTERRUPT15_TOG	0x21C

Table 5-50. HW_ICOLL_INTERRUPT15

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 8-72. HW_USBCTRL_ENDPTPRIME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSVD1**	RO	0x0	Reserved.

20:16	**PETB**	RW	0x0	Prime Endpoint Transmit Buffer.
				For each endpoint, a corresponding bit is used to request that a buffer be prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a 1 to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.
				Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
				PETB[4] = Endpoint 4.
				PETB[3] = Endpoint 3.
				PETB[2] = Endpoint 2.
				PETB[1] = Endpoint 1.
				PETB[0] = Endpoint 0.

15:5	**RSVD0**	RO	0x0	Reserved.

4:0	**PERB**	RW	0x0	Prime Endpoint Receive Buffer.
				For each endpoint, a corresponding bit is used to request a buffer be prepared for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a 1 to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.
				Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
				PERB[4] = Endpoint 4.
				PERB[3] = Endpoint 3.
				PERB[2] = Endpoint 2.
				PERB[1] = Endpoint 1.
				PERB[0] = Endpoint 0.

DESCRIPTION:

endpoint prime request

EXAMPLE:

```
Empty Example.
```

8.6.37 Endpoint Flush Register Description

This register is used in device-mode only.

HW_USBCTRL_ENDPTFLUSH 0x1b4

Table 11-88. HW_APBX_CH5_CMD

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="16">XFER_COUNT</td><td colspan="4">CMDWORDS</td><td colspan="4">RSVD1</td><td>WAIT4ENDCMD</td><td>SEMAPHORE</td><td colspan="2">RSVD0</td><td>IRQONCMPLT</td><td>CHAIN</td><td colspan="2">COMMAND</td></tr>
</table>

Table 11-89. HW_APBX_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DRI device HW_DRI_DATA register. A value of 0 indicates a 64 KBytes transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DRI, starting with the base PIO address of the DRI (HW_DRI_CTRL) and increment from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH5_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

12.2.2.1 DDR Address Mapping Options

The address structure of DDR devices consists of these fields:

- Datapath
- Column
- Row
- Chip Select
- Bank

The DRAM controller extracts these fields from the lower 30 bits of the system address. The exact bit positions for each field are defined in the programmable registers of the controller. The order of extraction, however, is always fixed as:

Bank-Chip Select-Row-Column-Datapath

The maximum widths of each of these fields are fixed at:

- Bank = 2 bits
- Chip Select = 2 bits
- Row = 13 bits
- Column = 12 bits
- Datapath = 1 bit

The actual width of the column and row fields are programmable using the device address width bit fields (ADDR_PINS and COLUMN_SIZE) in the memory controller.

These maximum values, when combined, define the maximum 512-Mbyte addressable DRAM memory space. Figure 12-4 shows the positioning of the fields within the system address:

Note that practically, the maximum addressable external memory is limited to 128MB for the 169BGA and 64MB for the 128QFP packages. This is because most larger DRAMs require 14 row bits and the EMI controller supports a maximum of 13 row bits.

29 28	27 26	25 13	12 1	0
Bank	Chip Select	Row	Column	Datapath

Figure 12-4. Memory Controller Memory Map: Maximum

The ADDR_PINS and COLUMN_SIZE bit fields can each range from their maximum values down to a minimum value defined only by the size of the attached device. This allows the memory controller to function with a wide variety of memory device sizes. The settings for the ADDR_PINS and COLUMN_SIZE bit fields control how the address map is used to decode the user address to the DRAM chip selects and row and column addresses. It is assumed that the values in these bit fields never exceed the maximum values of 13 rows and 12 columns. Using the example shown in Figure 12-4, if the memory

12.5.7 DRAM Control Register 04 Description

DRAM control register. See bit fields for detailed descriptions.

HW_DRAM_CTL04 0x010

Table 12-15. HW_DRAM_CTL04

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							DLL_BYPASS_MODE	RSVD3							DLLLOCKREG	RSVD2							CONCURRENTAP	RSVD1							BANK_SPLIT_EN

Table 12-16. HW_DRAM_CTL04 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD4**	RO	0x0	Reserved.
24	**DLL_BYPASS_MODE**	RW	0x0	Enable the DLL bypass feature of the controller. Defines the behavior of the DLL bypass logic and establishes which set of delay parameters will be used. 0 = The values programmed in the DLL_DQS_DELAY_X, DQS_OUT_SHIFT, and WR_DQS_SHIFT are used. These parameters add fractional increments of the clock to the specified lines. 1 = The values programmed into the DLL_DQS_DELAY_BYPASS_X, DQS_OUT_SHIFT_BYPASS, and WR_DQS_SHIFT_BYPASS are used. These parameters specify the actual number of delay elements added to each of the lines. If the total delay time programmed into the delay parameters exceeds the number of delay elements in the delay chain, then the delay will be set to the maximum number of delay elements in the delay chain. 0 = Normal operational atuo-sync mode. 1 = Bypass the auto-sync DLL master delay line.
23:17	**RSVD3**	RO	0x0	Reserved.
16	**DLLLOCKREG**	RO	0x0	Status of DLL lock coming out of master delay. DLL lock/unlock.
15:9	**RSVD2**	RO	0x0	Reserved.
8	**CONCURRENTAP**	RW	0x0	Allow controller to issue commands to other banks while a bank is in auto pre-charge. Enables concurrent auto pre-charge. Some DRAM devices do not allow one bank to be auto pre-charged while another bank is reading or writing. The JEDEC standard allows concurrent auto pre-charge. Set this parameter for the DRAM device being used. 0 = Concurrent auto pre-charge disabled. 1 = Concurrent auto pre-charge enabled.

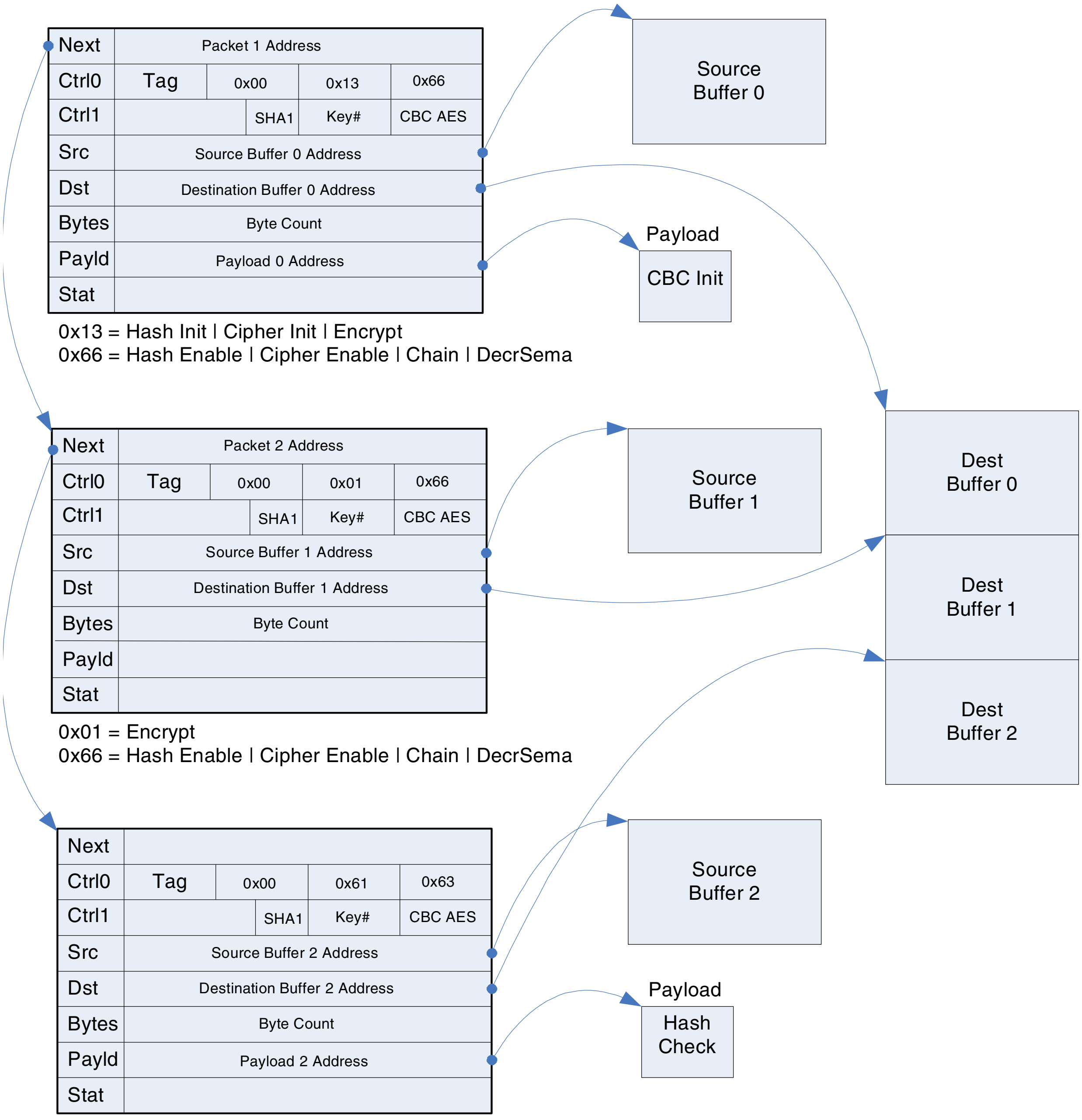
```
HW_PINCTRL_MUXSEL0_CLR(0x0000ffff);// data bits
HW_PINCTRL_MUXSEL1_CLR(0x000fffff);// control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The ECC8 engine is used for writing to the NAND but never produces an interrupt. From the sample code in Section 14.2.2.1, “DMA Structure Code Example:”

- DMA descriptor 1 prepares the NAND for data write by using the GPMI to issue a write setup command byte under “CLE”, then sends a 5-byte address under “ALE”. The ECC8 engine is disabled and not used for these commands.
- DMA descriptor 2 enables the ECC8 engine for t=8 encoding to begin the initial writing of the NAND data by specifying where the data payload is coming from in system memory.
- DMA descriptor 3 continues the writing of NAND data by specifying where the auxiliary data is coming from in system memory.
- DMA descriptor 4 issues the write commit command byte under “CLE” to the NAND.
- DMA descriptor 5 waits for the NAND to complete the write commit/transfer by watching the NAND’s ready line status. This descriptor relinquishes the NANDLOCK on the GPMI to enable the other DMA channels to initiate NAND transactions on different NAND CS lines.
- DMA descriptor 7 issues a NAND status command byte under “CLE” to check the status of the NAND device following the page write.
- DMA descriptor 8 reads back the NAND status and compares the status with an expected value. If there are differences, then the DMA processing engine follows an error-handling DMA descriptor path.
- DMA descriptor 9 disables the ECC8 engine and emits a GPMI interrupt to indicate that the NAND write has been completed.

14.2.3 Reed-Solomon ECC Decoding for NAND Reads

When a page is read from NAND flash, RS syndromes will be computed and, if correctable errors are found, they will be corrected on a per block basis within the NAND page. This decoding process is fully overlapped with other NAND data reads and with CPU execution. The RS decoder flowchart in Figure 14-9 shows the steps involved in programming the ECC8 Reed-Solomon decoder. The hardware flow of reading and decoding a 512-byte page encoded for t=8 error correction (18 parity bytes) is shown in Figure 14-10.



0x13 = Hash Init | Cipher Init | Encrypt
0x66 = Hash Enable | Cipher Enable | Chain | DecrSema

0x01 = Encrypt
0x66 = Hash Enable | Cipher Enable | Chain | DecrSema

0x61 = Hash Check | Hash Term | Encrypt
0x63 = Hash Enable | Cipher Enable | DecrSema | Interrupt

Figure 16-9. Multi-Buffer Scatter/Gather Cipher and Hash Operation

```
<code>

typedef struct _dcp_descriptor
{
    u32                *next;
    hw_dcp_packet1_t   ctrl0;
    hw_dcp_packet2_t   ctrl1;
    u32                *src,
                       *dst,
                        buf_size,
                       *payload,
                        stat;
}  DCP_DESCRIPTOR;
```

Table 19-32. HW_TVENC_CLOSEDCAPTION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
19:18	**CC_ENBL**	RW	0x0	Closed Caption Enable. Set mutually exclusive????? bit 1: enable insertion in even (bottom) field bit 0: enable insertion in odd (top) field }
17:16	**CC_FILL**	RW	0x0	Holds bits used to determine whether the two bytes in CC_DATA are to be inserted in the odd (upper) field or the even (lower) field.
15:0	**CC_DATA**	RW	0x0	Data to be inserted.

DESCRIPTION:

Closed Caption register of TV Encoder

EXAMPLE:

```
Empty example.
```

19.4.17 TV Encoder Color Burst Register Description

This is Color Burst Register of the TV Encoder Block

HW_TVENC_COLORBURST	0x140
HW_TVENC_COLORBURST_SET	0x144
HW_TVENC_COLORBURST_CLR	0x148
HW_TVENC_COLORBURST_TOG	0x14C

Table 19-33. HW_TVENC_COLORBURST

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
NBA								PBA								RSRVD1				RSRVD2											

Table 19-34. HW_TVENC_COLORBURST Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**NBA**	RW	0xc8	Parameter to define color burst. Should be set to 0xc8 for NTSC and 0xd6 for PAL-B
23:16	**PBA**	RW	0x0	Parameter to define color burst. Should be set to 0x00 for NTSC and 0x2a for PAL-B
15:12	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
11:0	**RSRVD2**	RO	0x0	Always write zeros to this bit field.

DESCRIPTION:

Color Burst Register of TV Encoder

EXAMPLE:

```
Empty example.
```

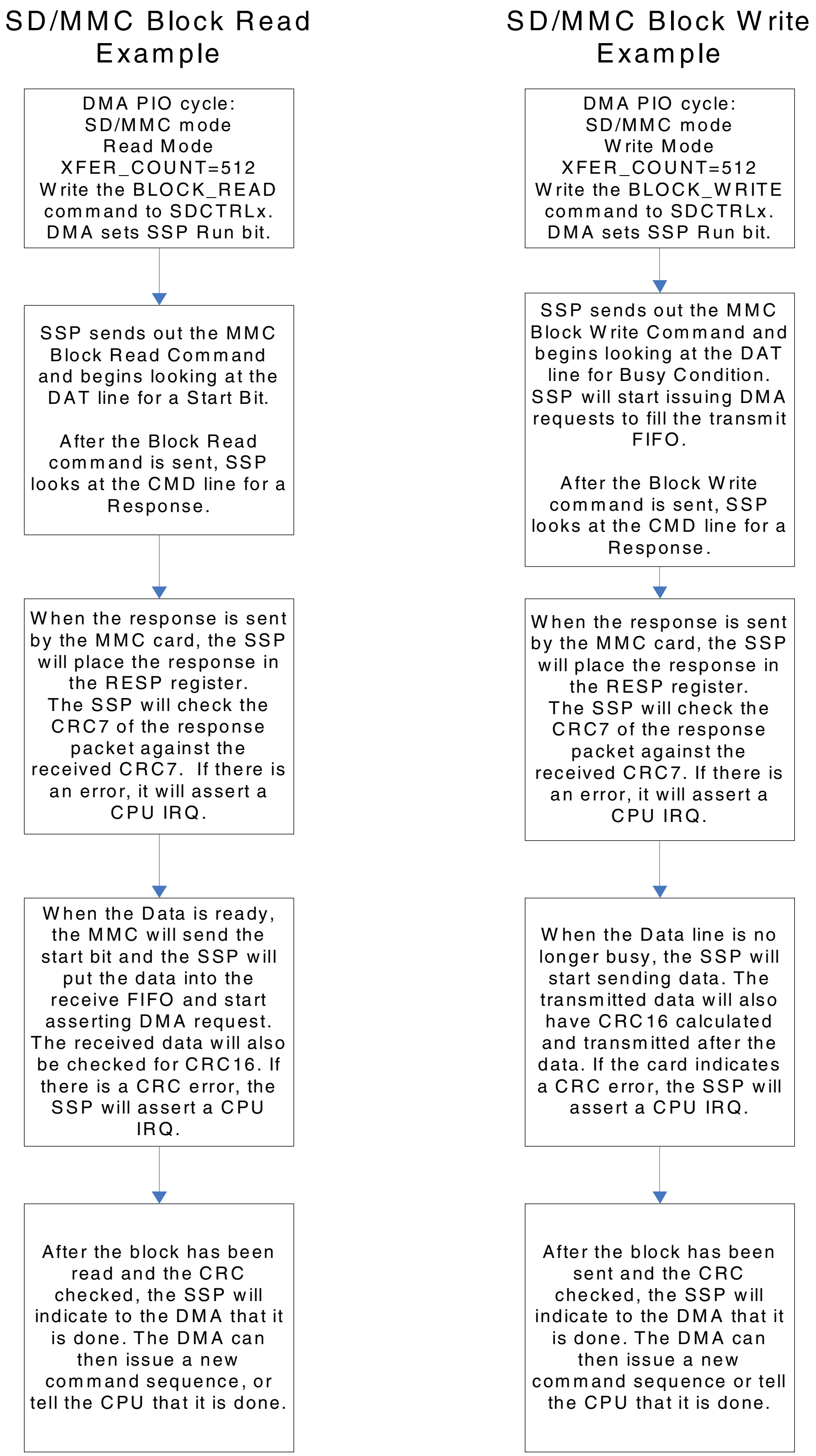

SD/MMC Block Read Example
DMA PIO cycle: SD/MMC mode Read Mode XFER_COUNT=512 Write the BLOCK_READ command to SDCTRLx. DMA sets SSP Run bit.
SSP sends out the MMC Block Read Command and begins looking at the DAT line for a Start Bit.
After the Block Read command is sent, SSP looks at the CMD line for a Response.
When the response is sent by the MMC card, the SSP will place the response in the RESP register. The SSP will check the CRC7 of the response packet against the received CRC7. If there is an error, it will assert a CPU IRQ.
When the Data is ready, the MMC will send the start bit and the SSP will put the data into the receive FIFO and start asserting DMA request. The received data will also be checked for CRC16. If there is a CRC error, the SSP will assert a CPU IRQ.
After the block has been read and the CRC checked, the SSP will indicate to the DMA that it is done. The DMA can then issue a new command sequence, or tell the CPU that it is done.
SD/MMC Block Write Example
DMA PIO cycle: SD/MMC mode Write Mode XFER_COUNT=512 Write the BLOCK_WRITE command to SDCTRLx. DMA sets SSP Run bit.
SSP sends out the MMC Block Write Command and begins looking at the DAT line for Busy Condition. SSP will start issuing DMA requests to fill the transmit FIFO.
After the Block Write command is sent, SSP looks at the CMD line for a Response.
When the response is sent by the MMC card, the SSP will place the response in the RESP register. The SSP will check the CRC7 of the response packet against the received CRC7. If there is an error, it will assert a CPU IRQ.
When the Data line is no longer busy, the SSP will start sending data. The transmitted data will also have CRC16 calculated and transmitted after the data. If the card indicates a CRC error, the SSP will assert a CPU IRQ.
After the block has been sent and the CRC checked, the SSP will indicate to the DMA that it is done. The DMA can then issue a new command sequence or tell the CPU that it is done.

Figure 21-11. SD/MMC Block Transfer Flowchart

DESCRIPTION:

SD/MMC status can be compared with a reference value.

EXAMPLE:

```
No Example.
```

21.10.5 SD/MMC compare mask Description

MMC/SD Status response mask .

HW_SSP_COMPMASK 0x040

Table 21-16. HW_SSP_COMPMASK

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
MASK																															

Table 21-17. HW_SSP_COMPMASK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**MASK**	RW	0x0	SD/MMC Compare mode Mask. If CHECK_RESP is set, the response is compared to REFERENCE, and the results are masked by this bitfield.

DESCRIPTION:

A mask allows the comparison of one or more bit fields in the reference value.

EXAMPLE:

```
No Example.
```

21.10.6 SSP Timing Register Description

SSP Timing Config Register

HW_SSP_TIMING 0x050

Table 21-18. HW_SSP_TIMING

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
TIMEOUT																CLOCK_DIVIDE								CLOCK_RATE							

Table 25-30. HW_I2C_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
28	**DMATERMINATE**	RO	0x0	Read-only view of the toggle state of the DMA Terminate signal.
27:26	**TBD**	RW	0x0	Reserved
25:16	**DMA_STATE**	RO	0x010	Current state of the DMA state machine.
15	**START_TOGGLE**	RO	0x0	Read-only view of the start detector. Toggles once for each detected start condition.
14	**STOP_TOGGLE**	RO	0x0	Read-only view of the stop detector. Toggles once for each detected stop condition.
13	**GRAB_TOGGLE**	RO	0x0	Read-only view of the grab receive data timing point. Toggles once for each read timing point, as delayed from rising clock.
12	**CHANGE_TOGGLE**	RO	0x0	Read-only view of the change xmit data timing point. Toggles once for each change xmit data timing point, as delayed from falling clock.
11	**TESTMODE**	RW	0x0	To be completed by designer.
10	**SLAVE_HOLD_CLK**	RO	0x0	Current State of the Slave Address Search FSM clock hold register.
9:0	**SLAVE_STATE**	RO	0x0000	Current State of the Slave Address Search FSM.

DESCRIPTION:

This register provides access to various internal states and controls that are used in diagnostic modes of operation.

EXAMPLE:

```
while(HW_I2C_DEBUG0.DMAREQ == old_dma_req_value); // wait for next dma request toggle
old_dma_req_value = HW_I2C_DEBUG0.DMAREQ; // remember the new state of the dma request toggle
```

25.4.9 I2C Device Debug Register 1 Description

The I2C Device Debug Register 1 provides a diagnostic view of the external bus and provides OE control for the clock and data.

HW_I2C_DEBUG1	0x080
HW_I2C_DEBUG1_SET	0x084
HW_I2C_DEBUG1_CLR	0x088
HW_I2C_DEBUG1_TOG	0x08C

Table 25-31. HW_I2C_DEBUG1

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
I2C_CLK_IN	I2C_DATA_IN	RSVD4		DMA_BYTE_ENABLES				CLK_GEN_STATE								RSVD2					LST_MODE		LOCAL_SLAVE_TEST	RSVD1			FORCE_CLK_ON	FORCE_ARB_LOSS	FORCE_RCV_ACK	FORCE_I2C_DATA_OE	FORCE_I2C_CLK_OE

Table 28-9. HW_AUDIOIN_ADCVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11:8	**RSRVD1**	RO	0x0	Reserved
7:0	**VOLUME_RIGHT**	RW	0xfe	Right Channel Volume Setting. This bit field is used to establish the incoming audio signal strength during record. Volume ranges from -0.5 dB (0xFF) 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.

DESCRIPTION:

The AUDIOIN Volume Register allows independent volume control of the left and right channels. Input audio can be attenuated in 0.5-dB steps, from full scale down to a minimum of -100 dB. This register is also used to enable/control volume updates such that they are only applied when PCM values cross zero to prevent unwanted audio artifacts.

EXAMPLE:

```
HW_AUDIOIN_ADCVOLUME.U = 0x00ff00ff; maximum volume for left and right channels.
```

28.4.5 AUDIOIN Debug Register Description

The AUDIOIN Debug Register is used for testing and debugging the AUDIOIN block.

HW_AUDIOIN_ADCDEBUG	0x040
HW_AUDIOIN_ADCDEBUG_SET	0x044
HW_AUDIOIN_ADCDEBUG_CLR	0x048
HW_AUDIOIN_ADCDEBUG_TOG	0x04C

Table 28-10. HW_AUDIOIN_ADCDEBUG

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
ENABLE_ADCDMA	RSRVD1																											ADC_DMA_REQ_HAND_SHAKE_CLK_CROSS	SET_INTERRUPT3_HAND_SHAKE	DMA_PREQ	FIFO_STATUS

Table 29-3. HW_AUDIOOUT_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
5	**DAC_ZERO_ENABLE**	RW	0x0	Never set DAC_ZERO_ENABLE! Make sure it is always cleared. (Since this bit is clear on reset, you'll be fine if you never touch it at all.) If you want to silence the audio output, but not go into a lower power mode, set HW_AUDIOOUT_DACVOLUME.MUTE_LEFT and HW_AUDIOOUT_DACVOLUME.MUTE_RIGHT simultaneously. This will immediately mute the digital audio signal sent to the DAC. There are two important things to note: 1. These bits ignore zero-crossing detection. They immediately mute the digital audio signal, so this can cause a "pop." The "pop" can be avoided by ramping the volume. 2. These bits cannot be toggled when clock gated. If you want to minimize power consumption, mute the amplifiers first, and then power down the DAC. Reverse these operations to come back out of the low power state. The amplifiers will hold the output signal to VAG so everything will be quiet, and the DAC can power back up very quickly when you have to start playing sound again. To enter the low-power state: 1. Set HW_AUDIOOUT_HPVOL.MUTE and HW_AUDIOOUT_LINEOUTCTRL.MUTE to mute the Headphone and Line Out amplifiers, respectively. 2. Set HW_AUDIOOUT_PWRDN.DAC to power down the DAC. To exit the low-power state: 1. Clear HW_AUDIOOUT_PWRDN.DAC to power up the DAC. 2. Clear HW_AUDIOOUT_HPVOL.MUTE and HW_AUDIOOUT_LINEOUTCTRL.MUTE to un-mute the Headphone and Line Out amplifiers, respectively.
4	**LOOPBACK**	RW	0x0	AUDIOIN-to-AUDIOOUT Loopback Enable. Setting this bit to one routes the audio data received by the AUDIOIN's FIFO to the AUDIOOUT's FIFO. This test mode provides a loopback that does not use the DIGFILT's DMA memory inteface. This bit should be cleared to zero for normal operation.
3	**FIFO_UNDERFLOW_IRQ**	RW	0x0	FIFO Underflow Interrupt Status Bit. This bit is set by hardware if the AUDIOOUT's FIFO underflows any time during operation due to a DMA request that is not serviced in time. It is reset by software by writing a one to the corresponding bit in the HW_AUDIOOUT_CTRL_CLR register. An interrupt is issued to the host processor if this bit is set and FIFO_ERROR_IRQ_EN=1.

EXAMPLE:

```
BW_LRADC_CTRL3_HIGH_TIME(BV_LRADC_CTRL3_HIGH_TIME__83NS);
BW_LRADC_CTRL3_INVERT_CLOCK(BV_LRADC_CTRL3_INVERT_CLOCK__NORMAL);
```

33.4.5 LRADC Status Register Description

The LRADC status register returns various read-only status bit field values.

HW_LRADC_STATUS	0x040
HW_LRADC_STATUS_SET	0x044
HW_LRADC_STATUS_CLR	0x048
HW_LRADC_STATUS_TOG	0x04C

Table 33-9. HW_LRADC_STATUS

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
RSRVD3					TEMP1_PRESENT	TEMP0_PRESENT	TOUCH_PANEL_PRESENT	CHANNEL7_PRESENT	CHANNEL6_PRESENT	CHANNEL5_PRESENT	CHANNEL4_PRESENT	CHANNEL3_PRESENT	CHANNEL2_PRESENT	CHANNEL1_PRESENT	CHANNEL0_PRESENT	RSRVD2															TOUCH_DETECT_RAW

Table 33-10. HW_LRADC_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD3**	RO	0x0	Reserved
26	**TEMP1_PRESENT**	RO	0x1	This read-only bit returns a one when the temperature sensor 1 current source is present on the chip.
25	**TEMP0_PRESENT**	RO	0x1	This read-only bit returns a one when the temperature sensor 0 current source is present on the chip.
24	**TOUCH_PANEL_PRESENT**	RO	0x1	This read-only bit returns a one when the touch panel controller function is present on the chip.
23	**CHANNEL7_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 7 converter function is present on the chip.
22	**CHANNEL6_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 6 converter function is present on the chip.
21	**CHANNEL5_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 5 converter function is present on the chip.
20	**CHANNEL4_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 4 converter function is present on the chip.
19	**CHANNEL3_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 3 converter function is present on the chip.
18	**CHANNEL2_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 2 converter function is present on the chip.
17	**CHANNEL1_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 1 converter function is present on the chip.
16	**CHANNEL0_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 0 converter function is present on the chip.

33.4.8 LRADC 2 Result Register Description

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel two.

HW_LRADC_CH2	0x070
HW_LRADC_CH2_SET	0x074
HW_LRADC_CH2_CLR	0x078
HW_LRADC_CH2_TOG	0x07C

Table 33-15. HW_LRADC_CH2

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
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 33-16. HW_LRADC_CH2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18 bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

Table 37-2. Pin Multiplexing for 169-Pin BGA Package

Bank 2	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
Mux Reg 4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00	emi_a06		emi_a05		emi_a04		emi_a03		emi_a02		emi_a01		emi_a00		timrot2		timrot1		ssp1_sck		ssp1_d3		ssp1_d2		ssp1_d1		ssp1_d0		ssp1_det		ssp1_cmd	
select = 01															auart2_cts		auart2_rts						i2c_sd		i2c_clk				gpmi_ce3n			
select = 10															gpmi_ce3n		spdif		jtag_trst_n		jtag_tms		jtag_rtck		jtag_tck		jtag_tdi		usb_id		jtag_tdo	
select = 11	GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO	

Bank 2	31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
Mux Reg 5	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00	emi_wen		emi_rasn		emi_cke		gpmi_ce0n		gpmi_ce1n		emi_ce1n		emi_ce0n		emi_casn		emi_ba1		emi_ba0		emi_a12		emi_a11		emi_a10		emi_a09		emi_a08		emi_a07	
select = 01																																
select = 10																																
select = 11	GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO	

Bank 3	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
Mux Reg 6	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00	emi_d15		emi_d14		emi_d13		emi_d12		emi_d11		emi_d10		emi_d9		emi_d8		emi_d7		emi_d6		emi_d5		emi_d4		emi_d3		emi_d2		emi_d1		emi_d0	
select = 01																																
select = 10																																
select = 11	disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled	

Bank 3	31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
Mux Reg 7	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00																					emi_clkn		emi_clk		emi_dqs1		emi_dqs0		emi_dqm1		emi_dqm0	
select = 01																																
select = 10																																
select = 11																					disabled		disabled		disabled		disabled		disabled		disabled	

Table 37-15. HW_PINCTRL_MUXSEL4

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
BANK2_PIN15		BANK2_PIN14		BANK2_PIN13		BANK2_PIN12		BANK2_PIN11		BANK2_PIN10		BANK2_PIN09		BANK2_PIN08		BANK2_PIN07		BANK2_PIN06		BANK2_PIN05		BANK2_PIN04		BANK2_PIN03		BANK2_PIN02		BANK2_PIN01		BANK2_PIN00	

Table 37-16. HW_PINCTRL_MUXSEL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION

31:30	**BANK2_PIN15**	RW	0x3	Pin 108, EMI_A06 pin function selection:
				00= emi_addr06;
				01= reserved;
				10= reserved;
				11= GPIO.

29:28	**BANK2_PIN14**	RW	0x3	Pin 107, EMI_A05 pin function selection:
				00= emi_addr05;
				01= reserved;
				10= reserved;
				11= GPIO.

27:26	**BANK2_PIN13**	RW	0x3	Pin 109, EMI_A04 pin function selection:
				00= emi_addr04;
				01= reserved;
				10= reserved;
				11= GPIO.

25:24	**BANK2_PIN12**	RW	0x3	Pin 110, EMI_A03 pin function selection:
				00= emi_addr03;
				01= reserved;
				10= reserved;
				11= GPIO.

23:22	**BANK2_PIN11**	RW	0x3	Pin 111, EMI_A02 pin function selection:
				00= emi_addr02;
				01= reserved;
				10= reserved;
				11= GPIO.

21:20	**BANK2_PIN10**	RW	0x3	Pin 112, EMI_A01 pin function selection:
				00= emi_addr01;
				01= reserved;
				10= reserved;
				11= GPIO.

19:18	**BANK2_PIN09**	RW	0x3	Pin 113, EMI_A00 pin function selection:
				00= emi_addr00;
				01= reserved;
				10= reserved;
				11= GPIO.

17:16	**BANK2_PIN08**	RW	0x3	Pin 38, ROTARYB pin function selection:
				00= timrot2;
				01= auart2_cts;
				10= gpmi_ce3n;
				11= GPIO.

37.4.21 PINCTRL Drive Strength and Voltage Register 11 Description

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 2.

HW_PINCTRL_DRIVE11	0x2b0
HW_PINCTRL_DRIVE11_SET	0x2b4
HW_PINCTRL_DRIVE11_CLR	0x2b8
HW_PINCTRL_DRIVE11_TOG	0x2bC

Table 37-45. HW_PINCTRL_DRIVE11

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
RSRVD7	BANK2_PIN31_V	BANK2_PIN31_MA		RSRVD6	BANK2_PIN30_V	BANK2_PIN30_MA		RSRVD5	BANK2_PIN29_V	BANK2_PIN29_MA		RSRVD4		BANK2_PIN28_MA		RSRVD3		BANK2_PIN27_MA		RSRVD2	BANK2_PIN26_V	BANK2_PIN26_MA		RSRVD1	BANK2_PIN25_V	BANK2_PIN25_MA		RSRVD0	BANK2_PIN24_V	BANK2_PIN24_MA	

Table 37-46. HW_PINCTRL_DRIVE11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD7**	RO	0x0	Always write zeroes to this field.
30	**BANK2_PIN31_V**	RW	0x1	Pin 114, EMI_WEN pin voltage selection: 0= 1.8V (mDDR) or 2.5V (DDR1); 1= reserved.

29:28	**BANK2_PIN31_MA**	RW	0x0	Pin 114, EMI_WEN pin output drive strength selection:
				00= 4 mA;
				01= 8 mA;
				10= 12 mA;
				11= 16 mA.

27	**RSRVD6**	RO	0x0	Always write zeroes to this field.
26	**BANK2_PIN30_V**	RW	0x1	Pin 98, EMI_RASN pin voltage selection: 0= 1.8V (mDDR) or 2.5V (DDR1); 1= reserved.

25:24	**BANK2_PIN30_MA**	RW	0x0	Pin 98, EMI_RASN pin output drive strength selection:
				00= 4 mA;
				01= 8 mA;
				10= 12 mA;
				11= 16 mA.

23	**RSRVD5**	RO	0x0	Always write zeroes to this field.
22	**BANK2_PIN29_V**	RW	0x1	Pin 115, EMI_CKE pin voltage selection: 0= 1.8V (mDDR) or 2.5V (DDR1); 1= reserved.

21:20	**BANK2_PIN29_MA**	RW	0x0	Pin 115, EMI_CKE pin output drive strength selection:
				00= 4 mA;
				01= 8 mA;
				10= 12 mA;
				11= 16 mA.

19:18	**RSRVD4**	RO	0x0	Always write zeroes to this field.