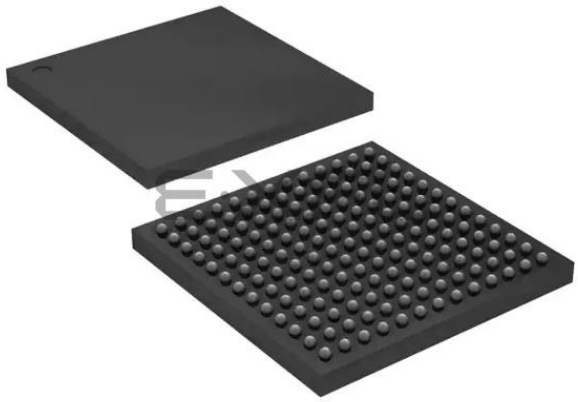


**Welcome to E-XFL.COM**

## 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/pro/item?MUrl=&PartUrl=mcimx233djm4b |

# Contents

- ITU-R BT.656 compliant D1 digital video output mode with on-the-fly RGB to YCbCr color-space-conversion. This output also feeds the integrated TV-Encoder
- Supports wide variety of input and output formats allowing for conversion between input and output (e.g., RGB565 input to RGB888 output). Also supports packed pixel formats.

See Chapter 18, "LCD Interface (LCDIF)," for more information.

### 1.2.19.2 Pixel Processing Pipeline (PXP)

The PXP performs all necessary post display frame pre-processing in hardware with minimal memory overhead. In a video-centric system such as the i.MX23, this allows for the CPU to have maximum processing bandwidth for video-decode operation. The PXP operation and features can be described as follows:

- The *background* image (e.g. decoded video frames) is read from external memory into separate Y/U/V buffers as 8x8 pixel macroblocks. These buffers are then fed into a color-space converter (e.g. YUV to RGB) followed by the scaling engine which utilizes an advanced bi-linear weighted scaling algorithm. The scaling operation is defined in terms of the output image (via programmable offsets and cropping registers). The output of the scaler is fed into yet another internal buffer called S0. If the background image is already in the RGB color-space it is assumed to be scaled appropriately for the required output format and can thus be read directly into the internal S0 buffer. In order to maintain efficient use of external memory, only the relevant (visible) portion of the background image is fetched.
- The scaled RGB image (in the internal S0 buffer) can be blended with up to eight programmable *overlays*. The co-ordinates of the overlays can once again be described in terms of the resultant output image. Each overlay can have a either a global programmable opacity or a per-pixel resolution if constructed with ARGB color-space. In addition to this, each overlay can have a relative priority level such that when constructing the output image, the PXP only fetches the visible overlay in the current 8x8 macroblock. The overlays are fetched into the internal S1 buffer. Alpha blending is performed on the S0 and S1 buffers to generate the blended output into the internal S3 buffer. Other operations such as BITBLT and color-keying can also be performed at this stage.
- The final stage of the PXP operation is the rotator which can perform flips and 90, 180 and 270 rotations. The rotator operates on the 8x8 pixel macroblocks in the S3 buffer to maximize external memory fetch efficiency. It writes 8x8 macroblocks to external memory in this final stage.

It should be noted that the PXP supersedes all pixel operations of the DCP.

See Chapter 17, "Pixel Pipeline (PXP)," for more information on the PXP.

### 1.2.19.3 PAL/NTSC TV-Encoder

The PAL/NTSC TV-Encoder is part of the integrated TV-Out functionality of i.MX23. The encoder takes input directly from the LCDIF without intermediate memory access. In order to utilize the TV-Out path, the LCDIF must be configured to output the ITU-R BT.656/BT.601 D1 digital video stream mode. This stream is synchronized to the internal 108MHz clock of the TV-Encoder. After this point, the block

**Table 4-31. HW_CLKCTRL_FRAC Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 6 | **CPU_STABLE** | RO | 0x0 | This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divide should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state. |
| 5:0 | **CPUFRAC** | RW | 0x12 | This field controls the CPU clock fractional divider. The resulting frequency shall be 480 * (18/CPUFRAC) where CPUFRAC = 1-35. |

**DESCRIPTION:**

This register controls the 9-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

**EXAMPLE:**

```
HW_CLKCTRL_FRAC_WR(BF_CLKCTRL_FRAC_CPUFRAC(4));
```

## 4.8.16 Fractional Clock Control Register 1 Description

The FRAC1 control register provides control for PFD clock generation.

| | |
|---|---|
| HW_CLKCTRL_FRAC1 | 0x100 |
| HW_CLKCTRL_FRAC1_SET | 0x104 |
| HW_CLKCTRL_FRAC1_CLR | 0x108 |
| HW_CLKCTRL_FRAC1_TOG | 0x10C |

**Table 4-32. HW_CLKCTRL_FRAC1**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLKGATEVID | VID_STABLE | RSRVD1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

## 7.4.10 Value of OTP Bank0 Word7 (Crypto Key) Description

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

HW_OCOTP_CRYPTO3 0x090

**Table 7-19. HW_OCOTP_CRYPTO3**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BITS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 7-20. HW_OCOTP_CRYPTO3 Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 31:0 | **BITS** | RO | 0x0 | Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). If LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR] |

### DESCRIPTION:

Non-shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

### EXAMPLE:

```
Empty Example.
```

## 7.4.11 HW Capability Shadow Register 0 Description

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

HW_OCOTP_HWCAP0 0x0A0

**Table 7-21. HW_OCOTP_HWCAP0**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BITS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 7-22. HW_OCOTP_HWCAP0 Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 31:0 | **BITS** | RW | 0x0 | Shadow register for HW capability bits 31:0 (copy of OTP bank 1, word 0 (ADDR = 0x08)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set. |

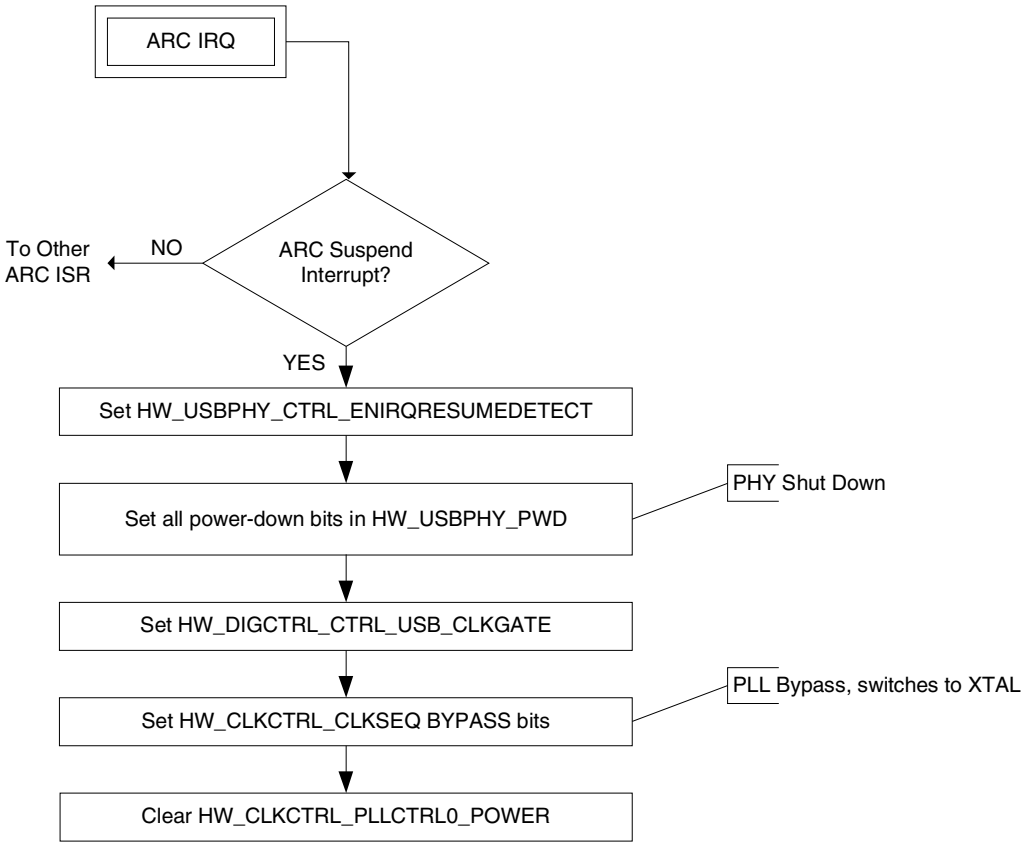



**Figure 8-4. USB 2.0 PHY PLL Suspend Flowchart**

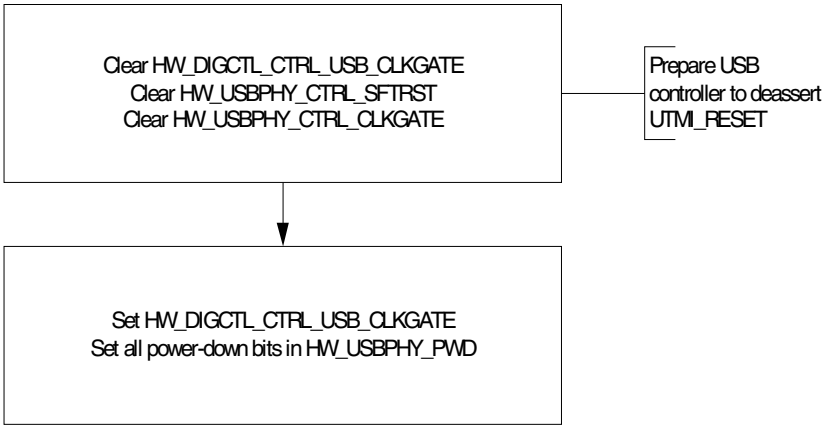



**Figure 8-5. UTMI Powerdown**

**Table 11-194. HW_APBX_CH12_DEBUG2**

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

**Table 11-195. HW_APBX_CH12_DEBUG2 Bit Field Descriptions**

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

### DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 12.

### EXAMPLE:

```
Empty example.
```

## 11.5.97 APBX DMA Channel 13 Current Command Address Register Description

The APBX DMA Channel 13 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH13_CURCMDAR 0x6B0

**Table 11-196. HW_APBX_CH13_CURCMDAR**

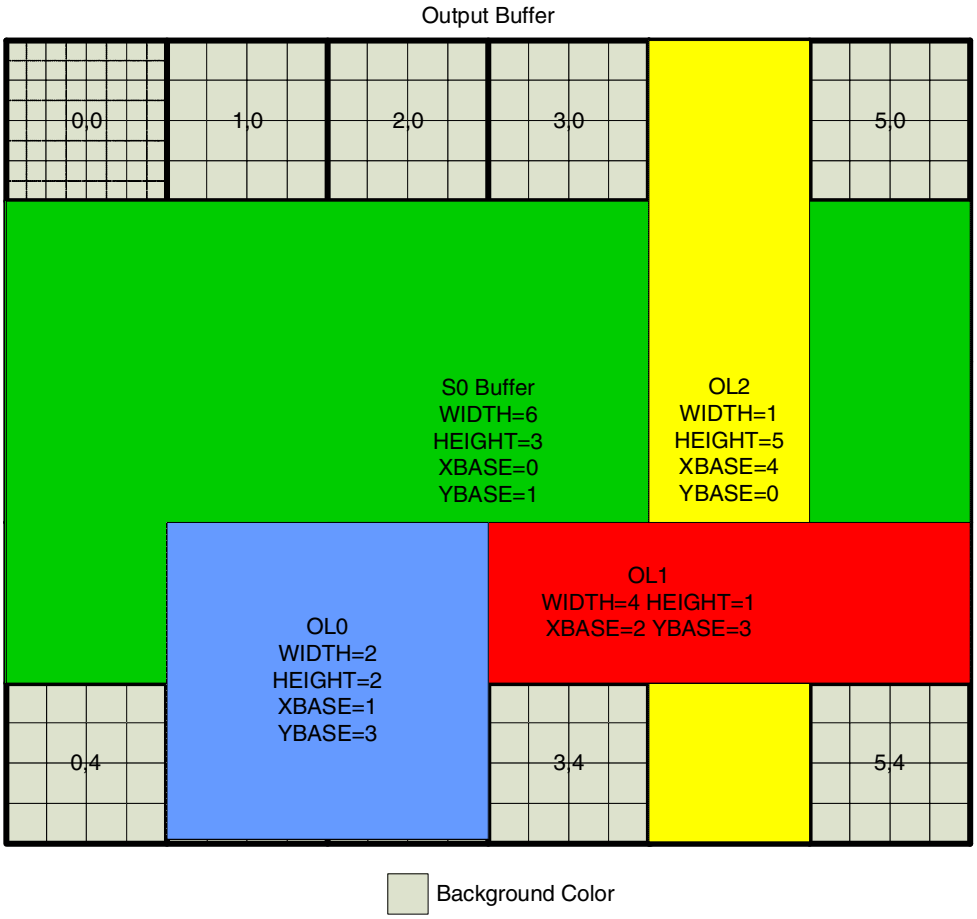
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMD_ADDR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

**Table 11-197. HW_APBX_CH13_CURCMDAR Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 31:0 | **CMD_ADDR** | RO | 0x00000000 | Pointer to command structure currently being processed for Channel 13. |

### DESCRIPTION:

APBX DMA Channel 13 is controlled by a variable sized command structure. This register points to the command structure currently being executed.




The S0 buffer and each overlay can be placed within the output buffer using their XBASE and YBASE registers and the dimensions of each region are set using their WIDTH and HEIGHT parameters. Overlay 0 has the highest priority (effectively it is the highest in the stacking order) and the S0 buffer and background color have the lowest priority.

Overlays can be blended with the background or S0 planes, but not with each other. Effectively only a single overlay is active for each 8x8 pixel block.

**Figure 17-12. Pixel Pipeline Overlay Support**

Each overlay can perform one of three classes of operations between the overlay and the underlying background (S0) image: alpha blending, color keying, or raster operations.

An overlay can be enabled by writing the address of the overlay image to the OLn register, the overlay's size and location information into the OLnSIZE register, and then setting the OLnPARAM_ENABLE bit. The OLnPARAM registers also contain further controls to select the modes of operation (below).

**Table 17-8. Register Use for Conversion (continued)**

| Register | Value | Description |
|---|---|---|
| OL2PARAM | 0x00000000 | Overlay 2 disabled |
| OL3PARAM | 0x00000000 | Overlay 3 disabled |
| OL4PARAM | 0x00000000 | Overlay 4 disabled |
| OL5PARAM | 0x00000000 | Overlay 5 disabled |
| OL6PARAM | 0x00000000 | Overlay 6 disabled |
| OL7PARAM | 0x00000000 | Overlay 7 disabled |
| CTRL | 0x000C9003 | SCALE=1<br>CROP=1<br>S0_FORMAT=9<br>(YUV420) IRQ_ENABLE=1<br>ENABLE=1 |

Note that the scaling factors are computed as (source/dest)*4096, thus in the horizontal direction 640/480*4096=5461=0x1555. In the vertical direction, the scaling factor is computed as 480/272*4096=7228=0x1C3C. The original source image and resulting scaled images are shown below:

**Table 22-8. HW_TIMROT_TIMCTRL0 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 0 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.<br>0: Positive edge detection.<br>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. |
| 5:4 | **PRESCALE** | RW | 0x0 | Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency.<br>DIV_BY_1 = 0x0 PreScale: Divide the APBX clock by 1.<br>DIV_BY_2 = 0x1 PreScale: Divide the APBX clock by 2.<br>DIV_BY_4 = 0x2 PreScale: Divide the APBX clock by 4.<br>DIV_BY_8 = 0x3 PreScale: Divide the APBX clock by 8. |
| 3:0 | **SELECT** | RW | 0x0 | Selects the source for the timer "tick" that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior.<br>NEVER_TICK = 0x0 Never tick.<br>PWM0 = 0x1 Input from PWM0.<br>PWM1 = 0x2 Input from PWM1.<br>PWM2 = 0x3 Input from PWM2.<br>PWM3 = 0x4 Input from PWM3.<br>PWM4 = 0x5 Input from PWM4.<br>ROTARYA = 0x6 Input from Rotary A.<br>ROTARYB = 0x7 Input from Rotary B.<br>32KHZ_XTAL = 0x8 Input from 32-kHz crystal.<br>8KHZ_XTAL = 0x9 Input from 8-kHz (divided from 32-kHz crystal).<br>4KHZ_XTAL = 0xA Input from 4-kHz (divided from 32-kHz crystal).<br>1KHZ_XTAL = 0xB Input from 1-kHz (divided from 32-kHz crystal).<br>TICK_ALWAYS = 0xC Always tick. |

### DESCRIPTION:

This control register specifies control parameters, as well as interrupt status and the enable for Timer 0.

### EXAMPLE:

```
HW_TIMROT_TIMCTRLn_WR(0, 0x00000008); // Set up control fields for timer0
```

## 22.4.4 Timer 0 Count Register Description

The Timer 0 Count Register contains the timer counter values for Timer 0.

HW_TIMROT_TIMCOUNT0 0x030

```
// Poll for reset to clear the channel.
for (retries = 0; retries < RESET_TIMEOUT; retries++)
     if ((reset_mask & HW_APBX_CTRL0_RD()) == 0)
          break;
if( retries == RESET_TIMEOUT) exit(1);

// Setup dma channel configuration.
BF_WRn(APBX_CHn_NXTCMDAR,I2C_CHANNEL_NUM,
       CMD_ADDR,(reg32_t) I2C_DMA_CMD);
BF_WR(APBX_CTRL1, CH3_CMDCMPLT_IRQ, 0); // clear interrupt

// Start the dma channel by incrementing semaphore.
BF_WRn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, INCREMENT_SEMA, 1);

// Poll for the semaphore to decrement to zero on the DMA channel.
for (retries = 0; retries < SEMAPHORE_TIMEOUT; retries++)
     if (0 == BF_RDn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, PHORE))
          break;

// a frame with one byte of address and five bytes of data was just sent
}
```

### 25.2.3.2 Reading 256 Bytes from an EEPROM

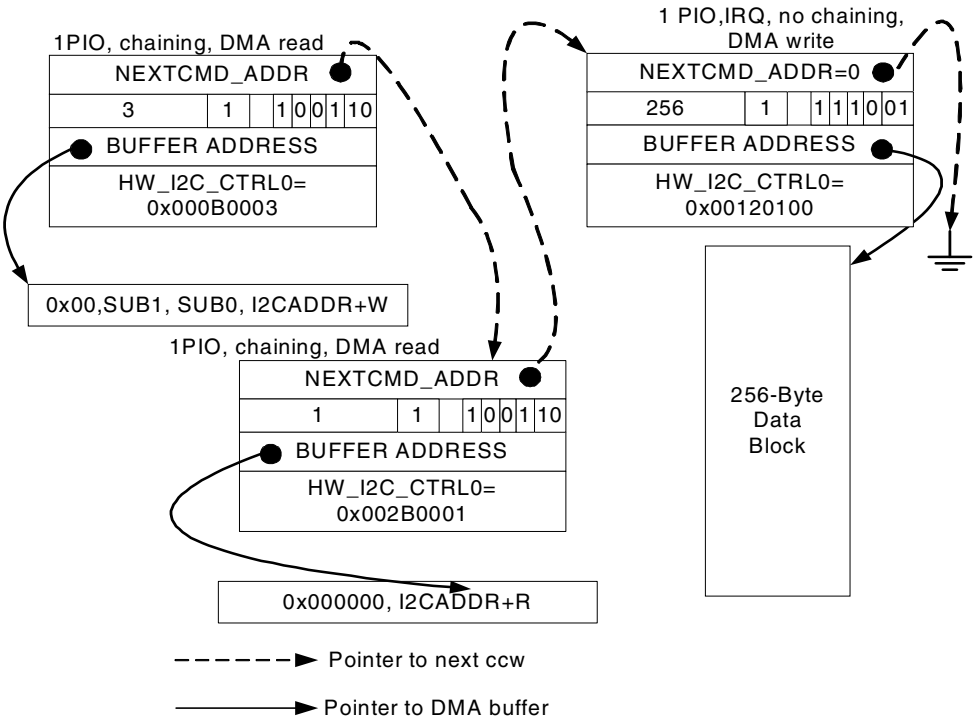



**Figure 25-9. I²C Reading 256 Bytes from an EEPROM**

```
//-----------------------------------------------------------------------
// dma buffers to hold i2c command string for slave addres+W plus sub0,
// sub 1 and the second command, a slave address+R
// eePROM write address == 0xA0,  read address == 0xA1
//-----------------------------------------------------------------------
unsigned char eeprom_command_buffer[4] = {0xA0,0x34,0x12,0xA1};

//-----------------------------------------------------------------------
// I2C DMA chain
//-----------------------------------------------------------------------
const static reg32_t  I2C_DMA_CMD3[4] =
{
    0x0,
    (BF_APBX_CHn_CMD_XFER_COUNT(256) |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)  |
```

the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

**EXAMPLE:**

```
No Example.
```

## 27.3.12 UART Masked Interrupt Status Register Description

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

HW_UARTDBGMIS 0x040

**Table 27-24. HW_UARTDBGMIS**

| 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 | | | | | OEMIS | BEMIS | PEMIS | FEMIS | RTMIS | TXMIS | RXMIS | DSRMMIS | DCDMMIS | CTSMMIS | RIMMIS |

**Table 27-25. HW_UARTDBGMIS 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 | **OEMIS** | RO | 0x0 | Overrun Error Masked Interrupt Status. |
| 9 | **BEMIS** | RO | 0x0 | Break Error Masked Interrupt Status. |
| 8 | **PEMIS** | RO | 0x0 | Parity Error Masked Interrupt Status. |
| 7 | **FEMIS** | RO | 0x0 | Framing Error Masked Interrupt Status. |
| 6 | **RTMIS** | RO | 0x0 | Receive Timeout Masked Interrupt Status. |
| 5 | **TXMIS** | RO | 0x0 | Transmit Masked Interrupt Status. |
| 4 | **RXMIS** | RO | 0x0 | Receive Masked Interrupt Status. |
| 3 | **DSRMMIS** | RO | 0x0 | nUARTDSR Modem Masked Interrupt Status. |
| 2 | **DCDMMIS** | RO | 0x0 | nUARTDCD Modem Masked Interrupt Status. |
| 1 | **CTSMMIS** | RO | 0x0 | nUARTCTS Modem Masked Interrupt Status. |
| 0 | **RIMMIS** | RO | 0x0 | nUARTRI Modem Masked Interrupt Status. |

**DESCRIPTION:**

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

Figure 29-1 shows a high-level diagram of the AUDIOOUT module. See Figure 1-4 for a diagram of the audio path and control options.

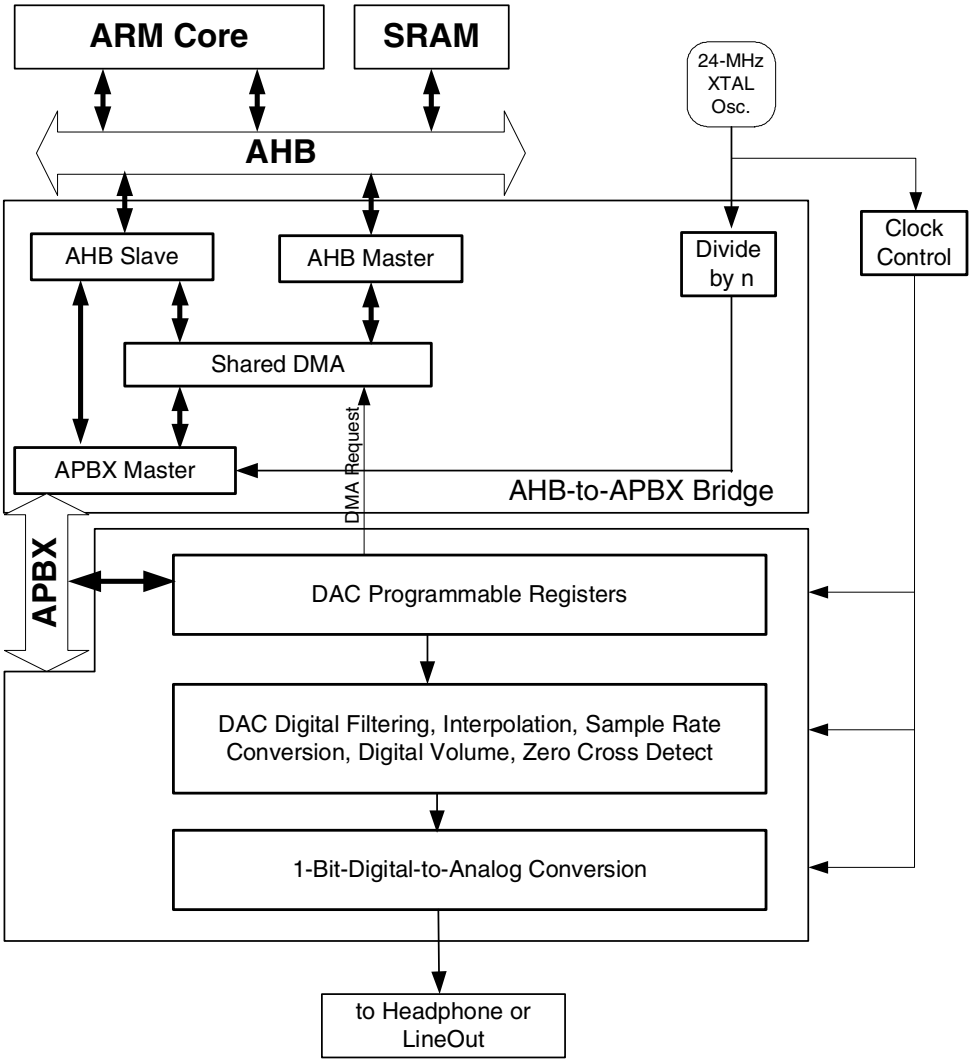



**Figure 29-1. Functional AUDIOOUT/DAC Block Diagram**

## 29.2 Operation

Audio data conversion begins by either using the i.MX23's AHB-APBX bridge DMA engine to write two's complement PCM data to the AUDIOOUT's input FIFO, or by writing the data directly to the AUDIOOUT's data register via the host CPU. The data is then normalized to 24-bit samples and then filtered using a 3-stage FIR filter, consisting of 33, 11, and 7 taps, respectively. The AUDIOOUT contains a sequencer, multiply-accumulate hardware, and a set of filter coefficients that performs successive iterations on the data stored in RAM. Intermediate data calculated along the taps/stages of the FIR are also stored in the AUDIOOUT's RAM. The AUDIOOUT module includes hardware for interpolation, sample

**Table 30-1. HW_SPDIF_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 | RSRVD1 | | | | | | | | | DMAWAIT_COUNT | | | | | RSRVD0 | | | | | | | | | | WAIT_END_XFER | WORD_LENGTH | FIFO_UNDERFLOW_IRQ | FIFO_OVERFLOW_IRQ | FIFO_ERROR_IRQ_EN | RUN |

**Table 30-2. HW_SPDIF_CTRL Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 31 | **SFTRST** | RW | 0x1 | Setting this bit to one forces a reset to the entire block and then gates the clocks off. This bit must be set to zero for normal operation. |
| 30 | **CLKGATE** | RW | 0x1 | This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. WARNING: First set the CLKGATE bit in the HW_CLKCTRL_SPDIF register to 1. Only then, set this bit to 1 to prevent any extra samples from being transmitted. When removing clock gating, follow the reverse order: First reset this CLKGATE bit to 0, and then reset the CLKGATE bit in the HW_CLKCTRL_SPDIF register to 0 . |
| 29:21 | **RSRVD1** | RO | 0x00 | Reserved |
| 20:16 | **DMAWAIT_COUNT** | RW | 0x00 | DMA Request Delay Count. This bit field specifies the number of APBX clock cycles (0 to 31) to delay before each DMA request. This field acts as a throttle on the bandwidth consumed by the SPDIF block. This field can be loaded by the DMA. |
| 15:6 | **RSRVD0** | RO | 0x000 | Reserved |
| 5 | **WAIT_END_XFER** | RW | 0x1 | Set this bit to a one if the SPDIF Transmitter should wait until the internal FIFO is empty before halting transmission based on deassertion of RUN. Use in conjuntion with HW_SPDIF_STAT_END_XFER to determine transfer completion |
| 4 | **WORD_LENGTH** | RW | 0x0 | Set this bit to one to enable 16-bit mode. Set this bit to zero for 32-bit mode. In either case, the SPDIF frame allows transmission of only 24 bits. In 16-bit mode, eight zeros will be appended to the LSB's of the input sample; in 32-bit mode, the 24 MSB's of HW_SPDIF_DATA will be transmitted. |
| 3 | **FIFO_UNDERFLOW_IRQ** | RW | 0x0 | This bit is set by hardware if the FIFO underflows during SPDIF transmission. Reset this bit by writing a one to the SCT clear address space and not by a general write. |

**Table 32-3. HW_POWER_CTRL Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 10 | **ENIRQ_VDDIO_BO** | RW | 0x0 | Enable interrupt for VDDIO brownout. |
| 9 | **VDDA_BO_IRQ** | RW | 0x0 | Interrupt Status for VDDA_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write. |
| 8 | **ENIRQ_VDDA_BO** | RW | 0x0 | Enable interrupt for VDDA brownout. |
| 7 | **VDDD_BO_IRQ** | RW | 0x0 | Interrupt Status for VDDD_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write. |
| 6 | **ENIRQ_VDDD_BO** | RW | 0x0 | Enable interrupt for VDDD brownout. |
| 5 | **POLARITY_VBUSVALID** | RW | 0x1 | Set to 1 to check for 5V connected using VBUSVALID status bit. Set to 0 to check for 5V disconnected. |
| 4 | **VBUSVALID_IRQ** | RW | 0x0 | Interrupt status for VBUSVALID signal. Interrupt polarity is set using POLARITY_VBUSVALID. Reset this bit by writing a one to the SCT clear address space and not by a general write. |
| 3 | **ENIRQ_VBUS_VALID** | RW | 0x0 | Enable interrupt for 5V detect using VBUSVALID. |
| 2 | **POLARITY_VDD5V_GT_VDDIO** | RW | 0x1 | Set to 1 to check for 5V connected. Set to 0 to check for 5V disconnected. |
| 1 | **VDD5V_GT_VDDIO_IRQ** | RW | 0x0 | Interrupt status for VDD5V_GT_VDDIO signal. Interrupt polarity is set using POLARITY_VDD5V_GT_VDDIO. Reset this bit by writing a one to the SCT clear address space and not by a general write. |
| 0 | **ENIRQ_VDD5V_GT_VDDIO** | RW | 0x0 | Enable interrupt for 5V detect. |

### DESCRIPTION:

Empty Description.

### EXAMPLE:

```
Empty Example.
```

## 32.11.2 DC-DC 5V Control Register Description

This register contains the configuration options of the power management subsystem that are available when external 5V is applied.

| | |
|---|---|
| HW_POWER_5VCTRL | 0x010 |
| HW_POWER_5VCTRL_SET | 0x014 |
| HW_POWER_5VCTRL_CLR | 0x018 |
| HW_POWER_5VCTRL_TOG | 0x01C |

**DESCRIPTION:**

Empty Description.

**EXAMPLE:**

```
Empty Example.
```

## 32.11.4 Battery Charge Control Register Description

This register cotrols the battery charge features for both NiMH slow charge and Li-Ion charge.

| | |
|---|---|
| HW_POWER_CHARGE | 0x030 |
| HW_POWER_CHARGE_SET | 0x034 |
| HW_POWER_CHARGE_CLR | 0x038 |
| HW_POWER_CHARGE_TOG | 0x03C |

**Table 32-8. HW_POWER_CHARGE**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSVD5 | | | | ADJ_VOLT | | | | RSRVD3 | ENABLE_LOAD | ENABLE_CHARGER_RESISTORS | ENABLE_FAULT_DETECT | CHRG_STS_OFF | RSVD4 | RSVD3 | PWD_BATTCHRG | RSVD2 | | | | STOP_ILIMIT | | | | RSVD1 | | | | BATTCHRG_I | | | |

**Table 32-9. HW_POWER_CHARGE Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 31:27 | **RSVD5** | RO | 0x0 | Empty Description. |
| 26:24 | **ADJ_VOLT** | RW | 0x0 | Adjustments to the final LiIon final voltage. These bits shouldn't be set unless recommended by Freescale.<br>0b000: no change<br>0b001: -0.25%<br>0b010: +0.50%<br>0b011: -0.75%<br>0b100: +0.25%<br>0b101: -0.50%<br>0b110: +0.75%<br>0b111: -1.00% |
| 23 | **RSRVD3** | RW | 0x0 | Empty Description. |
| 22 | **ENABLE_LOAD** | RW | 0x0 | Enable 100ohm load on the regulated 4.2V output. This bit shouldn't be set unless recommended by Freescale. |

**Table 37-36. HW_PINCTRL_DRIVE6 Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 13:12 | **BANK1_PIN19_MA** | RW | 0x0 | Pin 33, LCD_RS pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= reserved. |
| 11:10 | **RSRVD2** | RO | 0x0 | Always write zeroes to this field. |
| 9:8 | **BANK1_PIN18_MA** | RW | 0x0 | Pin 31, LCD_RESET pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= reserved. |
| 7:6 | **RSRVD1** | RO | 0x0 | Always write zeroes to this field. |
| 5:4 | **BANK1_PIN17_MA** | RW | 0x0 | Pin 11, LCD_D17 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= reserved. |
| 3:2 | **RSRVD0** | RO | 0x0 | Always write zeroes to this field. |
| 1:0 | **BANK1_PIN16_MA** | RW | 0x0 | Pin 13, LCD_D16 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= reserved. |

### DESCRIPTION:

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

### EXAMPLE:

```
Empty Example.
```

## 37.4.17 PINCTRL Drive Strength and Voltage Register 7 Description

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 7 pins of bank 1.

| | |
|---|---|
| HW_PINCTRL_DRIVE7 | 0x270 |
| HW_PINCTRL_DRIVE7_SET | 0x274 |
| HW_PINCTRL_DRIVE7_CLR | 0x278 |
| HW_PINCTRL_DRIVE7_TOG | 0x27C |

Table 37-44. HW_PINCTRL_DRIVE10 Bit Field Descriptions

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 17:16 | **BANK2_PIN20_MA** | RW | 0x0 | Pin 102, EMI_A11 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= 16 mA. |
| 15 | **RSRVD3** | RO | 0x0 | Always write zeroes to this field. |
| 14 | **BANK2_PIN19_V** | RW | 0x1 | Pin 104, EMI_A10 pin voltage selection:<br>0= 1.8V (mDDR) or 2.5V (DDR1);<br>1= reserved. |
| 13:12 | **BANK2_PIN19_MA** | RW | 0x0 | Pin 104, EMI_A10 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= 16 mA. |
| 11 | **RSRVD2** | RO | 0x0 | Always write zeroes to this field. |
| 10 | **BANK2_PIN18_V** | RW | 0x1 | Pin 103, EMI_A09 pin voltage selection:<br>0= 1.8V (mDDR) or 2.5V (DDR1);<br>1= reserved. |
| 9:8 | **BANK2_PIN18_MA** | RW | 0x0 | Pin 103, EMI_A09 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= 16 mA. |
| 7 | **RSRVD1** | RO | 0x0 | Always write zeroes to this field. |
| 6 | **BANK2_PIN17_V** | RW | 0x1 | Pin 106, EMI_A08 pin voltage selection:<br>0= 1.8V (mDDR) or 2.5V (DDR1);<br>1= reserved. |
| 5:4 | **BANK2_PIN17_MA** | RW | 0x0 | Pin 106, EMI_A08 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= 16 mA. |
| 3 | **RSRVD0** | RO | 0x0 | Always write zeroes to this field. |
| 2 | **BANK2_PIN16_V** | RW | 0x1 | Pin 105, EMI_A07 pin voltage selection:<br>0= 1.8V (mDDR) or 2.5V (DDR1);<br>1= reserved. |
| 1:0 | **BANK2_PIN16_MA** | RW | 0x0 | Pin 105, EMI_A07 pin output drive strength selection:<br>00= 4 mA;<br>01= 8 mA;<br>10= 12 mA;<br>11= 16 mA. |

## DESCRIPTION:

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

## EXAMPLE:

```
Empty Example.
```

| | |
|---|---|
| HW_PINCTRL_PULL1_TOG | 0x41C |

**Table 37-55. HW_PINCTRL_PULL1**

| 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 | | | BANK1_PIN28 | RSRVD2 | | | | | BANK1_PIN22 | RSRVD1 | | | BANK1_PIN18 | RSRVD0 | | | | | | | | | | | | | | | | | |

**Table 37-56. HW_PINCTRL_PULL1 Bit Field Descriptions**

| BITS | LABEL | RW | RESET | DEFINITION |
|---|---|---|---|---|
| 31:29 | **RSRVD3** | RO | 0x0 | Always write zeroes to this field. |
| 28 | **BANK1_PIN28** | RW | 0x0 | Set this bit to one to disable the internal pad keeper and enable the internal pull up resistor on pin 129, PWM2. |
| 27:23 | **RSRVD2** | RO | 0x0 | Always write zeroes to this field. |
| 22 | **BANK1_PIN22** | RW | 0x0 | Set this bit to one to disable the internal pad keeper and enable the internal pull up resistor on pin 36, LCD_DOTCK. |
| 21:19 | **RSRVD1** | RO | 0x0 | Always write zeroes to this field. |
| 18 | **BANK1_PIN18** | RW | 0x0 | Set this bit to one to disable the internal pad keeper and enable the internal pull up resistor on pin 31, LCD_RESET. |
| 17:0 | **RSRVD0** | RO | 0x0 | Always write zeroes to this field. |

### DESCRIPTION:

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

### EXAMPLE:

```
Empty Example.
```

## 37.4.27 PINCTRL Bank 2 Pull Up Resistor Enable Register Description

The PINCTRL Bank 2 PULL Register enables/disables the internal pull up resistors for those pins in bank 2 which support this operation.

| | |
|---|---|
| HW_PINCTRL_PULL2 | 0x420 |
| HW_PINCTRL_PULL2_SET | 0x424 |
| HW_PINCTRL_PULL2_CLR | 0x428 |
| HW_PINCTRL_PULL2_TOG | 0x42C |

01: for NTSC

10: for PAL-M, PAL-N and Pal-60

11: other PAL cases

H_pal_fsc_phase_alt goes to SG to enable PAL-manner of phase alternation by field of color subcarrier

H_ntsc_ln_cnt goes to SG to align even-odd field identification with internal field count; expected value is "1" for ntsc and "0" for PAL-B

H_lpf_rst_off[8:0] goes to ES to program the time to generate a pulse so as to preload a pipeline in the y_delay module of the L_unit. The value is found by simulation to be 272 for NTSC in D1 mode, 284 for PAL in D1 mode, and 136 for progressive in external sync mode.

If "default_picform" in register 0 is set to1, all the parameters programmable through the register-10 group are instead assigned hardwired values determined by T_ENCD_MODE. However, if even one of these parameters needs to be given a different value, the entire register-10 group must be written and default_picform set to 0. To make this process easier, the hardwired values for all registers and modes are given in Table 38-2.

**Table 38-2. Hardwired Registers Values**

| mode | 10.0 | 10.1 | 10.2 | 10.3 | 10.4 |
|---|---|---|---|---|---|
| 000 NTSC | 17b1340e | 3a42242d | 07b14459 | 8320ca51 | 884a92c5 |
| 001 PAL-B | 17c1340e | 3a72642d | 077140 58 | 9c270c4e | 0471f244 |
| 010 PAL-M | 17b1340e | 3a72642d | 07f1505c | 8320ca51 | 0443a2c5 |
| 011 PAL-N | 17b1340e | 3a725c2d | 07e1505c | 9c270c51 | 0471b2c5 |
| 100 PAL-CN | 17c1340e | 3a72642d | 07a140 58 | 9c270c4e | 0471f244 |
| 101 mixed NTSC | 17b1340e | 3a42242d | 07b14459 | 8320ca51 | 044252c5 |
| 110 PAL-60 | 17b1340e | 3a72642d | 07f1505c | 8320ca51 | 0443a2c5 |
| 111 prog NTSC | 32a1380f | 359224xx (*) | xxxxxxxx (*) | 8320bacx (*) | 022202c5 |

(*) The parameters sync_eqend, wbrst_strt, nbrst_strt, brst_end, vstrt_subph are not used for progressive. The corresponding fields are "don't care."



This page ıration.

# Appendix C
# Acronyms and Abbreviations

This appendix includes definitions for many of the acronyms and abbreviations found in this product data sheet.

**AAC**: Advanced Audio Coding
**AC**: Audio Coding
**ADC**: Analog-to-Digital Converter
**AES**: Advanced Encryption Standard
**AHB**: Advanced High-performance Bus
**AIO**: Analog Input/Output
**AMBA**: Advanced Microcontroller Bus Architecture
**APB**: Advanced Peripheral Bus
**APBH**: Advanced Peripheral Bus—HCLK Domain
**APBX**: Advanced Peripheral Bus—XCLK Domain
**ARC**: ARC International (corporate name)
**ARM**: Advanced RISC Machine (formerly Acorn RISC Machine)
**AVC**: Adaptive Voltage Control
**BATT**: Battery
**BCB**: Boot Control Block
**BGA**: Ball Grid Array
**BIST**: Built-In Self-Test
**BKPT**: Breakpoint
**BLTC**: Boot Loader Transaction Controller
**CBC** Cipher Block Chaining
**CCS**: Command Completion Signaling
**CE**: Consumer Electronics
**CLKCTRL**: Clock Control
**CP**: Charge Pump
**CPUCLK**: Processor (ARM CPU) Clock
**CRC**: Cyclic Redundancy Check
**CSC**: Color-Space Conversion
**CTS**: Clear To Send
**DABT**: Data Abort
**DAC**: Digital-to-Analog Converter
**dB**: Decibel