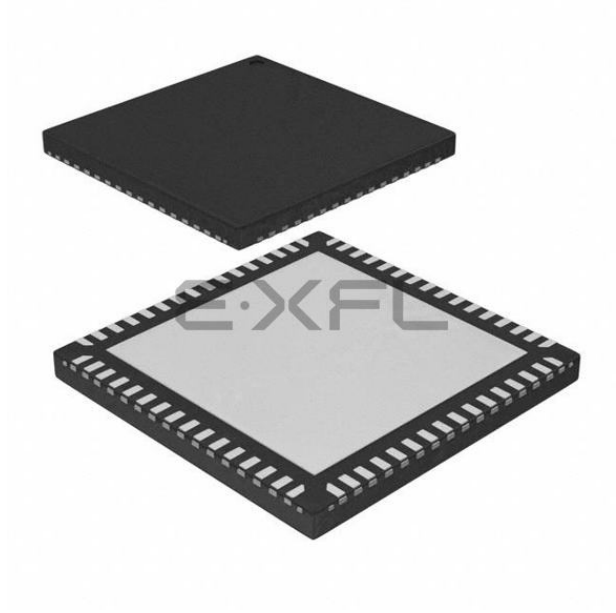

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8/16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	50
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.6V ~ 3.6V
Data Converters	A/D 16x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atxmega64d3-mh

Chapter 1
Overview

1.1 Introduction

The PXD10 family represents a new generation of 32-bit microcontrollers based on the Power Architecture®. These devices provide a cost-effective, single chip display solution for the industrial market. An integrated TFT driver with digital video input ability from an external video source, significant on-chip memory, and low power design methodologies provide flexibility and reliability in meeting display demands in rugged environments. The advanced processor core offers high performance processing optimized for low power consumption, operating at speeds as high as 64 MHz. The family itself is fully scalable from 512 KB to 1 MB internal flash memory. The memory capacity can be further expanded via the on-chip QuadSPI serial flash controller module.

The PXD10 platform has a single level of memory hierarchy and supports a wide range of on-chip SRAM and internal flash memories. The 1 MB flash memory version (PXD1010) outlined in detail within this document features 160 KB of on-chip graphics SRAM to buffer cost-effective color TFT displays driven via the on-chip Display Control Unit (DCU). Refer to Table 1-1, Table 1-2, and Table 1-3 for specific memory and feature sets of the product family members.

The PXD10 family benefits from the extensive development infrastructure for Power Architecture devices, which is already well established. This includes full support from available software drivers, operating systems, and configuration code to assist with users' implementations. See Section 1.6, Developer environment, for more information.

1.2 PXD10 family comparison

Table 1-1 and Table 1-2 report the memory scaling of code flash memory and RAM.

Table 1-1. Code flash memory scaling

Memory size	Start address	End address
512 KB (PXD1005)	0x0000_0000	0x0007_FFFF
1 MB (PXD1010)	0x0000_0000	0x000F_FFFF

Table 1-2. RAM memory scaling

Memory size	Start address	End address
48 KB (PXD1005 and PXD1010)	0x4000_0000	0x4000_BFFF

Table 2-1. PXD10 system memory map (continued)

AP		Size [KB]	PXD1005	PXD1010	Region[1]
Start Address	End Address				
0xFFF38000	0xFFF3BFFF	16	Yes	Yes	Software Watchdog (SWT0)
0xFFF3C000	0xFFF3FFFF	16	Yes	Yes	System Timer Module (STM0)
0xFFF40000	0xFFF43FFF	16	Yes	Yes	Error Correction Status Module
0xFFF44000	0xFFF47FFF	16	Yes	Yes	Direct Memory Access Controller 2 (DMA2x)
0xFFF48000	0xFFF4BFFF	16	Yes	Yes	Interrupt Controller (INTC)
0xFFF4C000	0xFFF8FFFF	272	—	—	Reserved
0xFFF90000	0xFFF93FFF	16	Yes	Yes	DSPI 0
0xFFF94000	0xFFF97FFF	16	Yes	Yes	DSPI 1
0xFFF98000	0xFFFA7FFF	64	—	—	Reserved
0xFFFA8000	0xFFFABFFF	16	No	Yes	QuadSPI 0
0xFFF9C000	0xFFFBFFFF	144	—	—	Reserved
0xFFFC0000	0xFFFC3FFF	16	Yes	Yes	FlexCan 0 (CAN0)
0xFFFC4000	0xFFFC7FFF	16	Yes	Yes	FlexCan 1 (CAN1)
0xFFFC8000	0xFFFDBFFF	80	—	—	Reserved
0xFFFDC000	0xFFFDFFFF	16	Yes	Yes	DMA Channel Multiplexer (DMA_MUX)
0xFFFE0000	0xFFFFBFFF	112	—	—	Reserved
0xFFFFC000	0xFFFFFFFF	16	Yes	Yes	Boot Assist Module (BAM)

1 The contents of memory addresses marked as reserved and individual bits within a memory address that are marked as reserved may return any value when read unless otherwise indicated.

2 Flash sector can be accessed via both slave ports. Arbitration logic required in case two different masters do access the same address at the same time.

Table 5-7. Main Configuration Register (MCR) field descriptions (continued)

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>5</td><td>Start trigger edge/ level detection. The following table shows the interaction between the EDGE bit and the TRGEN and EDGLEV bits.
<table>
<tr><th>TRGEN</th><th>EDGLEV</th><th>EDGE</th><th>Trigger Detection</th></tr>
<tr><td>0</td><td>n</td><td>n</td><td>External triggering disabled</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>External trigger on falling edge of trigger</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>External trigger on rising edge of trigger</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>External trigger on low edge of trigger</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External trigger on high edge of trigger</td></tr>
</table></td></tr>
<tr><td>6</td><td>Reserved
Must be kept at 0.</td></tr>
<tr><td>7</td><td>NSTART: Normal Start conversion
Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation.
This bit stays high while the conversion is ongoing (or pending during injection mode).
0 Causes the current chain conversion to finish and stops the operation
1 Starts the chain or scan conversion</td></tr>
<tr><td>8</td><td>Reserved
Write of any value has no effect, read value is always 0.</td></tr>
<tr><td>9</td><td>JTRGEN: Injection external trigger enable
0 External trigger disabled for channel injection (injected conversion cannot be started using an external signal)
1 External trigger enabled for channel injection</td></tr>
<tr><td>10</td><td>JEDGE: Injection trigger edge selection
Edge selection for external trigger, if JTRGEN = 1.
0 Selects falling edge for the external trigger
1 Selects rising edge for the external trigger</td></tr>
<tr><td>11</td><td>JSTART: Injection start
Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted.</td></tr>
<tr><td>12:13</td><td>Reserved
Write of any value has no effect, read value is always 0.</td></tr>
<tr><td>14</td><td>Reserved
Must be kept at 0.</td></tr>
<tr><td>15:22</td><td>Reserved
Write of any value has no effect, read value is always 0.</td></tr>
<tr><td>23</td><td>ADCLKSEL: Analog clock frequency selector
If this bit is set the AD_clk frequency is equal to ipg_clk frequency. Otherwise, it is half of ipg_clk frequency. This bit can be written in power-down only.</td></tr>
</table>

- Contains a set of registers to control peripheral clock selection
- Supports multiple clock sources and maps their address spaces to its memory map
- Generates an output clock
- Guarantees glitch-less clock transitions when changing the system clock selection
- Supports 8-, 16- and 32-bit wide read/write accesses

8.4.1.3 Modes of Operation

This section describes the basic functional modes of the MC_CGM.

8.4.1.3.1 Normal and Reset Modes of Operation

During normal and reset modes of operation, the clock selection for the system clock is controlled by the MC_ME.

8.4.2 External Signal Description

The MC_CGM delivers an output clock to the PH[4] pin for off-chip use and/or observation.

8.4.3 Memory Map and Register Definition

Table 8-2. MC_CGM Register Description

Address	Name	Description	Size	Access
0xC3FE_0370	CGM_OC_EN	Output Clock Enable	word	read/write
0xC3FE_0374	CGM_OCDS_SC	Output Clock Division Select	byte	read/write
0xC3FE_0378	CGM_SC_SS	System Clock Select Status	byte	read
0xC3FE_037C	CGM_SC_DC0	System Clock Divider Configuration 0	byte	read/write
0xC3FE_037D	CGM_SC_DC1	System Clock Divider Configuration 1	byte	read/write
0xC3FE_037E	CGM_SC_DC2	System Clock Divider Configuration 2	byte	read/write
0xC3FE_0380	CGM_AC0_SC	Aux Clock 0 Select Control	word	read/write
0xC3FE_0388	CGM_AC1_SC	Aux Clock 1 Select Control	word	read/write
0xC3FE_038C	CGM_AC1_DC0	Aux Clock 1 Divider Configuration 0	byte	read/write
0xC3FE_0398	CGM_AC2_SC	Aux Clock 2 Select Control	word	read/write
0xC3FE_0394	CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	byte	read/write
0xC3FE_0398	CGM_AC3_SC	Aux Clock 3 Select Control	word	read/write

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- Not change register content

8.5.1 Main features

- External crystal oscillator (FXOSC) digital interface
- Oscillator clock available interrupt
- Oscillator bypass mode
- Output clock division factors ranging from 1,2,3....32

8.5.2 Functional Description

The crystal oscillator circuit includes an internal oscillator driver and an external crystal circuitry. It provides an output clock that can be provided to PLL or used as a reference clock to specific modules depending on system needs.

The crystal oscillator is controlled by the MC_ME module. The OSCON bit of ME_XXX_MCR registers controls the powerdown of oscillator based on the current device mode while S_OSC of ME_GS register provides the oscillator clock available status.

After system reset, the oscillator is put to power down state and software has to switch on when required. Whenever the crystal oscillator is switched on from off state, OSCCNT counter starts and when it reaches the value EOCV[7:0]*512, oscillator clock is made available to the system. Also an interrupt pending bit I_OSC of OSC_CTL register is set. An interrupt will be generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by setting OSC_CTL[OSCBYP]. This bit can only be set by the software. System reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as external clock applied on EXTAL pin and the oscillator status is forced to ‘1’. The bypass configuration is independent of the powerdown mode of the oscillator.

Table 8-14 shows the truth table of different configurations of oscillator.

Table 8-14. Truth table of crystal oscillator

ENABLE	BYP	XTAL	EXTAL	CK_OSCM	OSC MODE
0	0	No crystal, Hiz	No crystal, Hiz	0	Power down, IDDQ
x	1	x	Ext clock	EXTAL	Bypass, OSC Disabled
1	0	Crystal	Crystal	EXTAL	Normal, OSC Enabled
		Gnd	Ext clock	EXTAL	Normal, OSC Enabled

The crystal oscillator clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the OSCDIV[4:0] bits of OSC_CTL register.

Table 12-5. Register descriptions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
THRESHOLD 0x1E8	R	0	0	0	0	0	0	LS_BF_VS									
	W																
	R	OUT_BUF_HIGH								OUT_BUF_LOW							
	W																
INT_STATUS 0x1EC	R	0	0	0	0	0	0	0	0	0	0	0	0	P4_FIFO_HI_FLAG	P4_FIFO_LO_FLAG	P3_FIFO_HI_FLAG	P3_FIFO_LO_FLAG
	W													w1c	w1c	w1c	w1c
	R	0	DMA_TRANS_FINISH	0	0	IPM_ERROR	PROG_END	P2_FIFO_HI_FLAG	P2_FIFO_LO_FLAG	P1_FIFO_HI_FLAG	P1_FIFO_LO_FLAG	CRC_OVERFLOW	CRC_READY	VS_BLANK	LS_BF_VS	UNDRUN	VSYNC
	W		w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
INT_MASK 0x1F0	R	0	0	0	0	0	0	0	0	0	0	0	0	M_P4_FIFO_HI_FLAG	M_P4_FIFO_LO_FLAG	M_P3_FIFO_HI_FLAG	M_P3_FIFO_LO_FLAG
	W																
	R	0	M_DMA_TRANS_FINISH	0	0	M_IPM_ERROR	M_PROG_END	M_P2_FIFO_HI_FLAG	M_P2_FIFO_LO_FLAG	M_P1_FIFO_HI_FLAG	M_P1_FIFO_LO_FLAG	M_CRC_OVERFLOW	M_CRC_READY	M_VS_BLANK	M_LS_BF_VS	M_UNDRUN	M_VSYNC
	W																
COLBAR_1 0x1F4	R	1	1	1	1	1	1	1	1	COLBAR_1_R							
	W																
	R	COLBAR_1_G								COLBAR_1_B							
	W																

Offset: 0x234 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP_BUF_p2_hi								INP_BUF_p2_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP_BUF_p1_hi								INP_BUF_p1_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 37.

Figure 12-40. Threshold input buffer 1 Register (THRESHOLD_INP_BUF_1)

Table 12-35. THRESHOLD_INP_BUF_1 Field Descriptions

Field	Description
0–7 INP_BUF_p2_hi	High Threshold for input buffer for blend stage 2.
8–15 INP_BUF_p2_lo	Low Threshold for input buffer for blend stage 2.
16–23 INP_BUF_p1_hi	High Threshold for input buffer for blend stage 1 (background).
24–31 INP_BUF_p1_lo	Low Threshold for input buffer for blend stage 1 (background plane).

12.3.4.31 THRESHOLD_INP_BUF_2 Register

Figure 12-41 represents the threshold register for input buffer for plane 3 and plane 4.

Offset: 0x238 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP_BUF_p4_hi								INP_BUF_p4_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP_BUF_p3_hi								INP_BUF_p3_lo							
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 38.

Figure 12-41. THRESHOLD_INP_BUF_2 Register

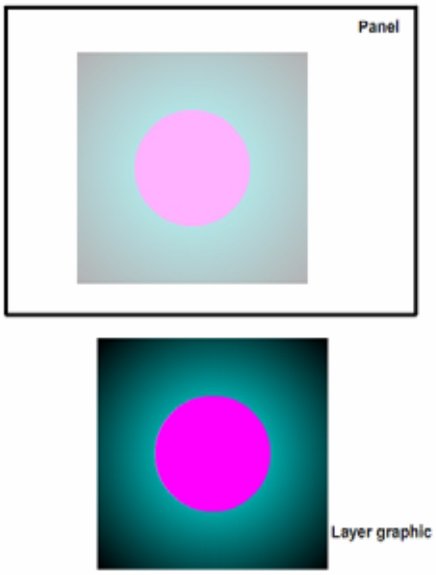

Figure 12-63. Case 3 example (all pixels transparent)

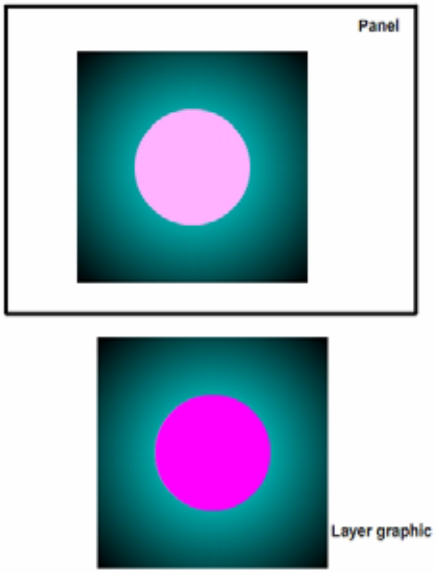

Figure 12-64. Case 4 example (selected pixels transparent)

15.1.1 Overview

The DMA is a highly-programmable data transfer engine, which has been optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known, and is *not* defined within the data packet itself. The DMA hardware supports:

- Single design supporting 16-, 32- and 64-channel implementations, dependent on size of the TCD memory and design parameters
- Connections to the AMBA-AHB crossbar switch for bus mastering the data movement, slave bus for programming the module
  - Parameterized support for 32- and 64-bit AMBA-AHB datapath widths
- 32-byte transfer control descriptor per channel stored in local memory
- 32 bytes of data registers, used as temporary storage to support burst transfers

Throughout this document, *n* is used to reference the channel number. Additionally, data sizes are defined as byte (8-bit), halfword (16-bit), word (32-bit) and doubleword (64-bit).

15.1.2 Features

The DMA module supports the following features:

- 16 programmable channels
- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes
- Transfer control descriptor organized to support two-deep, nested transfer operations
  - An *inner* data transfer loop defined by a “minor” byte transfer count
  - An *outer* data transfer loop defined by a “major” iteration count
- Channel service request via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
    - Independent channel linking at end of minor loop and/or major loop
  - Peripheral-paced hardware requests (one per channel)
  - For all three methods, *one service request per execution of the minor loop is required*
- Support for fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Error terminations are optionally enabled per channel, and logically summed together to form a small number of error interrupt outputs
- Support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software or hardware

Table 15-11. DMA Clear Error (DMACERR) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 6-0
CERR[0:6]	Clear Error Indicator	0-63 Clear corresponding bit in DMAERR{H,L} 64-127 Clear all bits in DMAERR{H,L}

15.2.1.11 DMA Set START Bit (DMASSRT)

The DMASSRT register provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding Transfer Control Descriptor to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing all START bits to be set. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Table 15-28 for the TCD START bit definition.

Register address: DMA_Offset + 0x001e

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	SSRT[0:6]						
RESET:		0	0	0	0	0	0	0

= Unimplemented

Figure 15-12. DMA Set START Bit (DMASSRT) Register

Table 15-12. DMA Set START Bit (DMASSRT) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 6-0
SSRT[0:6]	Set START Bit (Channel Service Request)	0-63 Set the corresponding channel's TCD.start 64-127 Set all TCD.start bits

15.2.1.12 DMA Clear DONE Status (DMACDNE)

The DMACDNE register provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding Transfer Control Descriptor to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing all DONE bits to be cleared. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Table 15-28 for the TCD DONE bit definition.

Register address: DMA_Offset + 0x001f

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W	NOP	CDNE[0:6]						
RESET:		0	0	0	0	0	0	0

= Unimplemented

Figure 15-13. DMA Clear DONE Status (DMACDNE) Register

Chapter 22
LCD Driver (LCD64F6B)

22.1 Information Specific to This Device

This section presents device-specific parameterization and customization information not specifically referenced in the remainder of this chapter.

22.1.1 Number of Front and Back Planes

Table 22-1. Number of Front and Back Planes

Parameter	Value
Number of front planes	38 or 40[1]
Number of back planes	4 or 6[1]

1 Software-configurable

22.1.2 LCD Clock Selection

Table 22-2 shows the clocks selected by the LCDCR[LCDOCS] bit.

Table 22-2. LCD Clock Selection Based on LCDCR[LCDOCS]

Value of LCDCR[LCDOCS]	Clock selected
1	32 kHz OSC
0	128 kHz OSC

22.1.3 Settings during STANDBY mode

To keep the LCD driver shut down in STANDBY mode, the following settings are needed:

- LCDCR[LCDRST] = 0
- LCDCR[LCDRCS] = 0

To keep the LCD driver on (functioning) in STANDBY mode, the following settings are needed:

- LCDCR[LCDRST] = 1
- LCDCR[LCDRCS] = 1
- LCDCR[LCDOCS] = 0 or 1
  - — If this field is 0, the LCD driver will operate from SIRC.
  - — If this field is 1, the LCD driver will operate from SXOSC.

Table 23-25. IFMI field descriptions

Field	Description
IFMI[0:4] 27:31	Filter match index This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Section 23.8.2.2, Slave mode for more details). When no filter matches, IFMI[0:4] = 0. When Filter *n* is matching, IFMI[0:4] = *n* + 1.

23.7.2.19 Identifier filter mode register (IFMR)

This register is not implemented on LINFlex_1.

Address: Base + 0x0048 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	IFM			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-25. Identifier filter mode register (IFMR)

Table 23-26. IFMR field descriptions

Field	Description
IFM	Filter mode 0 Filters 2*n* and 2*n* + 1 are in identifier list mode. 1 Filters 2*n* and 2*n* + 1 are in mask mode (filter 2*n* + 1 is the mask for the filter 2*n*). (Refer to Table 23-27.)

Table 23-27. IFMR[IFM] configuration

Bit	Value	Result
IFM[0]	0	Filters 0 and 1 are in identifier list mode.
	1	Filters 0 and 1 are in mask mode (filter 1 is the mask for the filter 0).
IFM[1]	0	Filters 2 and 3 are in identifier list mode.
	1	Filters 2 and 3 are in mask mode (filter 3 is the mask for the filter 2).
IFM[2]	0	Filters 4 and 5 are in identifier list mode.
	1	Filters 4 and 5 are in mask mode (filter 5 is the mask for the filter 4).
IFM[3]	0	Filters 6 and 7 are in identifier list mode.
	1	Filters 6 and 7 are in mask mode (filter 7 is the mask for the filter 6).

30.5.3.2 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check that the TX Buffer is empty. If the QSPI_SFMSR[TXNE] bit is set the TX Buffer must be cleared by writing 1 into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register. Optionally one can clear the QSPI_TBSR[TRCTR] field by writing 1 into QSPI_MCR[CLR_TXF].
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR). At least one word of data must be written into the TX Buffer.
4. Program the required instruction code options (i.e. size of data) into the QSPI_ICR[ICO] register.
5. Trigger the IP Command to program the serial flash device by writing the instruction code into the QSPI_ICR[IC] register.
6. Depending from the amount of data required step 3 must be repeated until all the required data have been written into the QSPI_TBDR register. At any time the QSPI_TBDR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Steps 4 and 5 may be executed together.

Upon writing the QSPI_ICR[IC] field (refer to step 5) the QuadSPI module will start to execute the command by transferring instruction code, address and then data to the external device. The data are fetched from the TX Buffer. It consists of 15 entries with 32-bit and is organized as a circular FIFO, whose read pointer is incremented after each fetch. When all data are transmitted, the QuadSPI module will return from ‘busy’ to ‘idle’. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

30.5.3.3 Flash Read

Host access to the data stored in the external serial flash device is done in two steps: First the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

30.5.3.3.1 Reading Serial Flash Data into the QuadSPI Module

Read access to the external serial flash device can be triggered in two different ways:

- **IP Command Read**: For **flash read via the register interface** the user must provide the required components of a SFM command to the QSPI_SFAR and the QSPI_ICR registers. All available read commands supported by the external serial flash are possible.

  Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QSPI_MCR[CLR_RXF] bit.

  From these inputs the complete transaction is built when the QSPI_ICR[IC] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash device into the RX Buffer. Since the read access is triggered via the register interface the IP_ACC status bit is set driving in turn the BUSY bit (both are located in the QSPI_SFMSR register).

Chapter 33
Sound Generation Logic (SGL)

33.1 Introduction

This document describes the Sound Generation Logic (SGL) module.

Refer to Figure 33-1 for the detailed block diagram of the SGL.

33.1.1 Overview

The SGL provides a single output to the speaker or buzzer interface.

It selects outputs from the PWM channels being used for sound generation. In case of monotonic sound, two PWM channels are required while, in the case of polyphonic sound, only one PWM channel is required. Monotonic / polyphonic sound can be selected through the signal mono/poly_b.

A 32-bit counter value determines the duration for which sound will be played.

NOTE

In this document, the term “polyphonic” refers to PCM-based sound generation as described in Section 33.4, Functional description.

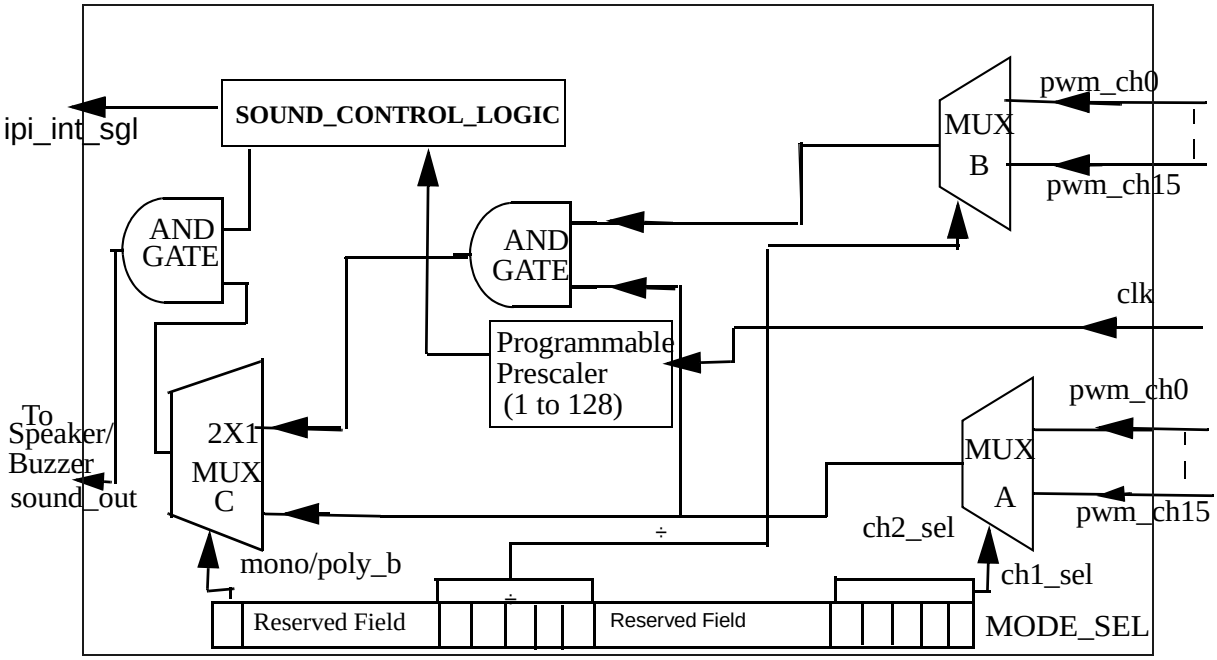

Figure 33-1. Block diagram

33.3 Memory map and register definition

33.3.1 Memory map

Table 33-2. SGL Memory Map

Address Offset	Register	Access	Reset Value
0x00	MODE_SEL register	R/W[1]	0x0000_0000
0X04	SOUND_DURATION register	R/W	0x0000_0000
0X08	HIGH_PERIOD register	R/W	0x0000_0000
0X0C	LOW_PERIOD register	R/W	0x0000_0000
0X10	SGL_STATUS register	R	0x00

[1] Note that R/W registers may contain some read-only or write-only bits.

33.3.2 Register summary

The conventions in Figure 33-3 serve as a key for the register summary and individual register diagrams.

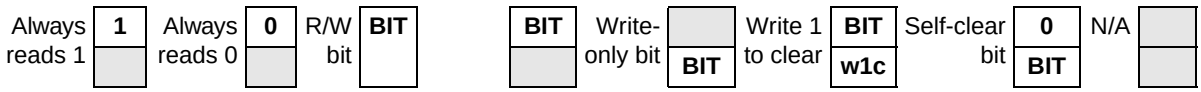

Figure 33-3. Key to register fields

33.3.3 Register descriptions

33.3.3.1 MODE_SEL register

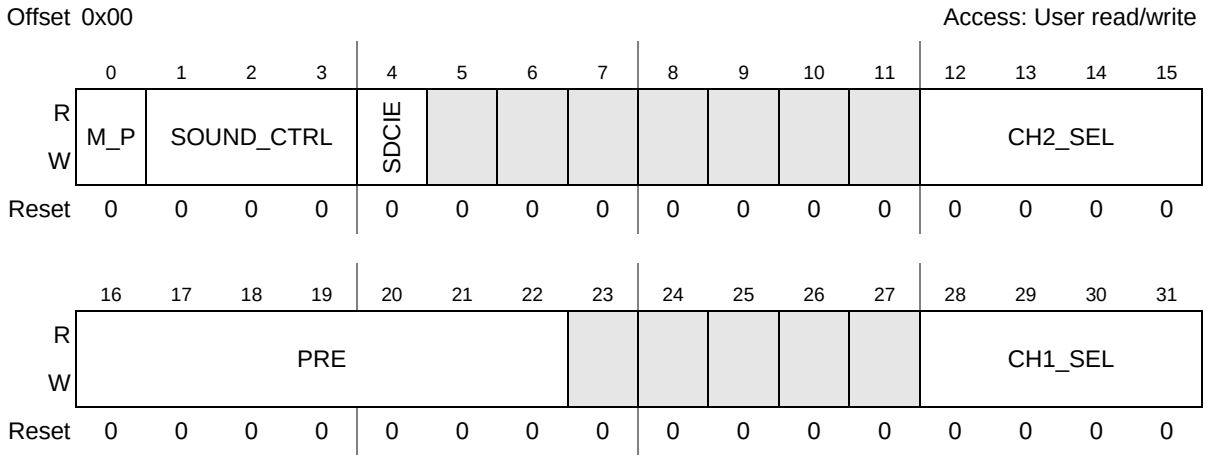

Figure 33-4. MODE_SEL REGISTER

33.3.3.5 SGL_STATUS register

Offset 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																SDCIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-8. SGL_STATUS register

Table 33-8. SGL_STATUS field descriptions

Field	Description
SDCIF	Sound Duration Complete Interrrupt Flag bit. Reflects the status of the interrupt. 1 SDCIF set event has occurred due to generation of interrupt 0 SDCIF cleared

33.4 Functional description

The SGL can be used to produce two types of sounds as described below:

- Monotonic sound
  The tone amplitude modulation is based on two different mixed signals.The first of these two signals has a variable frequency and fixed duty cycle (signal1 in Figure 33-9). The second signal has a fixed frequency and a variable pulse width for generation of the amplitude (signal2 in Figure 33-9). The duty cycle of this signal represents the amplitude of the generated sound. The sound generator is generally a PWM that generates 2 different signals and a mixer to generate a tone at the output as shown in signal 3 of Figure 33-9. It is passed through a first-order low-pass filter to produce signal4 in Figure 33-9. This signal can be fed to the speaker interface.

Table 34-2. Low power configuration

Mode	Configuration
Run, Test, Safe and Stop	All general purpose and graphics SRAM is powered and operational.
Standby (1)	Only 8 KB of the SRAM remains powered. Upper RAM is disabled by the MC_PCU.
Standby (2)	All system SRAM remains powered. Upper RAM is enabled by the MC_PCU.

34.5 Register memory map

The internal SRAM has no registers. Registers for the SRAM ECC are located in the ECSM.

34.6 SRAM ECC mechanism

The SRAM ECC detects the following conditions and produces the following results:

- Detects and corrects all 1-bit errors
- Detects and flags all 2-bit errors as non-correctable errors
- Detects 39-bit reads (32-bit data bus plus the 7-bit ECC) that return all zeros or all ones, asserts an error indicator on the bus cycle, and sets the error flag

SRAM does not detect all errors greater than 2 bits.

Internal SRAM write operations are performed on the following byte boundaries:

- 1 byte (0:7 bits)
- 2 bytes (0:15 bits)
- 4 bytes or 1 word (0:31 bits)

If the entire 32 data bits are written to SRAM, no read operation is performed and the ECC is calculated across the 32-bit data bus. The 8-bit ECC is appended to the data segment and written to SRAM.

If the write operation is less than the entire 32-bit data width (1-, or 2-byte segment), the following occurs:

1. The ECC mechanism checks the entire 32-bit data bus for errors, detecting and either correcting or flagging errors.
2. The write data bytes (1-, or 2-byte segment) are merged with the corrected 32 bits on the data bus.
3. The ECC is then calculated on the resulting 32 bits formed in the previous step.
4. The 7-bit ECC result is appended to the 32 bits from the data bus, and the 39-bit value is then written to SRAM.

34.6.1 Access timing

The system bus is a two-stage pipelined bus, which makes the timing of any access dependent on the access during the previous clock. Table 34-3 lists the various combinations of read and write operations to SRAM and the number of wait states used for the each operation. The table columns contain the following information:

- Current operation: Lists the type of SRAM operation executing currently

Chapter 35
Stepper Motor Controller (SMC)

35.1 Introduction

The SMC block is a PWM motor controller suitable for driving small stepper and air core motors used in instrumentation applications. The module can also be used for other motor control or PWM applications that match the frequency, resolution and output drive capabilities of the module. The SMC has 12 PWM channels associated with two pins each (24 pins in total).

35.1.1 Features

The SMC includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- Left, right, or center aligned PWM
- Short-circuit detection in each PWM channel with programmable time-out

35.1.2 Modes of operation

35.1.2.1 Functional modes

35.1.2.1.1 Dither Function

Dither function can be selected or deselected by setting or clearing the MCCTL0[DITH] bit. This bit influences all PWM channels. For details, please refer to Section 35.4.1.3.5, Dither Bit (MCCTL0[DITH]).

35.1.2.2 PWM Channel Configuration Modes

The 12 PWM channels can operate in three functional modes. Those modes are, with some restrictions, selectable for each channel independently.

35.1.2.2.1 Dual Full H-Bridge Mode

This mode is suitable to drive a stepper motor or a 360^{o} air gauge instrument. For details, please refer to Section 35.4.1.1.1, Dual Full H-Bridge Mode”. In this mode two adjacent PWM channels are combined, and two PWM channels drive four pins.

35.1.2.2.2 Full H-Bridge Mode

This mode is suitable to drive any load requiring a PWM signal in a H-bridge configuration using two pins. For details please refer to Section 35.4.1.1.2, Full H-Bridge Mode”.

Table 35-7. MCCCx Field Descriptions

Field	Description
MCOM	**Output Mode** — MCOM controls the PWM channel's output mode. 00 Half H-bridge mode, PWM on pin MnCxM, pin MnCxP is released 01 Half H-bridge mode, PWM on pin MnCxP, pin MnCxM is released 10 Full H-bridge mode 11 Dual full H-bridge mode
MCAM	**PWM Channel Alignment Mode** — MCAM controls the PWM channel's PWM alignment mode and operation. MCAM and MCOM are double buffered. The values used for the generation of the output waveform will be copied to the working registers either at once (if all PWM channels are disabled or MCPER[PER] is set to 0) or if a timer counter overflow occurs. Reads of the register return the most recent written value, which are not necessarily the currently active values. 00 Channel disabled 01 Left aligned 10 Right aligned 11 Center aligned
CD	**PWM Channel Delay** — Each PWM channel can be individually delayed by a programmable number of PWM timer counter clocks. The delay will be n/f_{TC}. 00 Zero PWM clocks channel delay 01 One PWM clock channel delay 10 Two PWM clocks channel delay 11 Three PWM clocks channel delay

NOTE

The SMC will release the pins after the next PWM timer counter overflow without accommodating any channel delay if a single channel has been disabled or if the period register has been cleared or all channels have been disabled. Program one or more inactive PWM frames (duty cycle = 0) before writing a configuration that disables a single channel or the entire SMC.

35.3.2.5 Motor Controller Duty Cycle Register (MCDC0..11)

Each duty cycle register sets the sign and duty functionality for the respective PWM channel. The number of each register refires directly the PWM channel it controls. The relation between channels, pin names and register names is shown in Table 35-19.

Offset Module Base + 0x0020 . . . 0x0037

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIGN [4]	SIGN[3:0]				DUTY										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-6. Motor Controller Duty Cycle Register (MCDC0..11)

Table B-2. Detailed register map (continued)

Register Description	Register Name	Used Size	Address
Reserved	-	-	Base + (0x000C - 0x000E)
Miscellaneous Reset Status Register	ECSM_MRSR	8-bit	Base + 0x000F
Reserved	-	-	Base + (0x00010 - 0x0012)
Miscellaneous Wakeup Control Register	ECSM_MWCR	8-bit	Base + 0x0013
Reserved	-	-	Base + (0x0014 - 0x001E)
Miscellaneous Interrupt Register	ECSM_MIR	8-bit	Base + 0x001F
Reserved	-	-	Base + (0x0020 - 0x0023)
Miscellaneous User Defined Control Register	ECSM_MUDCR	32-bit	Base + 0x0024
Reserved	-	-	Base + (0x0028 - 0x0042)
ECC Configuration Register	ECSM_ECR	8bit	Base + 0x0043
Reserved	-	-	Base + (0x0044 - 0x0046)
ECC Status Register	ECSM_ESR	8bit	Base + 0x0047
Reserved	-	-	Base + (0x0048 - 0x0049)
ECC Error Generation Register	ECSM_EEGR	16-bit	Base + 0x004A
Reserved	-	-	Base + (0x04C - 0x004F)
Platform Flash ECC Error Address Register	ECSM_PFEAR	32-bit	Base + 0x0050
Reserved	-	-	Base + (0x054 - 0x0055)
Platform Flash ECC Master Number Register	ECSM_PFEMR	8-bit	Base + 0x0056
Platform Flash ECC Attributes Register	ECSM_PFEAT	8-bit	Base + 0x0057
Reserved	-	-	Base + (0x058 - 0x005B)
Platform Flash ECC Data Register	ECSM_PFEDR	32-bit	Base + 0x005C
Platform RAM ECC Address Register	ECSM_PREAR	32-bit	Base + 0x0060
Reserved	-	-	Base + 0x064
Platform RAM ECC Syndrome Register	ECSM_PRESR	8-bit	Base + 0x0065
Platform RAM ECC Master Number Register	ECSM_PREMR	8-bit	Base + 0x0066
Platform RAM ECC Attributes Register	ECSM_PREAT	8-bit	Base + 0x0067