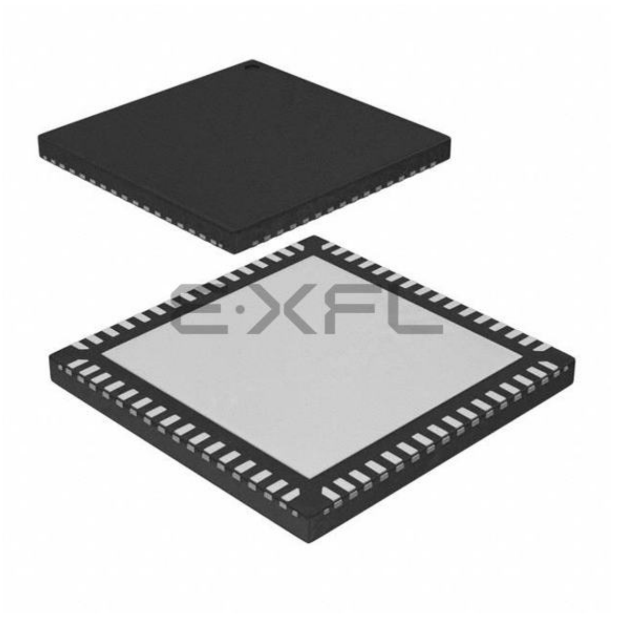


Welcome to E-XFL.COM

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	ARM® Cortex®-M4
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	I²C, SPI, UART/USART, USB
Peripherals	DMA, I²S, POR, PWM, WDT
Number of I/O	48
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	176K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 8x12b
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 (7.5x7.5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsamg55j19a-mut

5.3 Voltage Regulator

The SAM G55 embeds a core voltage regulator that is managed by the Supply Controller and that supplies the Cortex-M4 core, internal memories (SRAM, ROM and Flash logic) and the peripherals. An internal adaptive biasing adjusts the regulator quiescent current depending on the required load current.

For adequate input and output power supply decoupling/bypassing, refer to Table 39-4 “VDDCORE Voltage Regulator Characteristics” in section “Electrical Characteristics”.

In case of dual supply, the voltage regulator must be enabled and VDDOUT must be used as input control of the external DC/DC. This will allow a correct slope at first startup and for low power mode.

5.4 Typical Powering Schematics

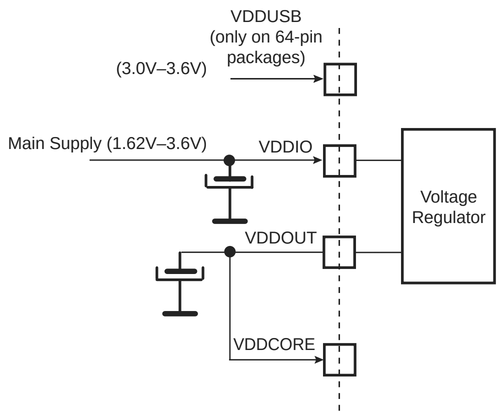
The SAM G55 supports single and dual voltage supply, with VDDIO from 1.62V to 3.6V and VDDCORE from external DC/DC controlled by the internal regulator. Figure 5-2 and Figure 5-3 illustrate the power schematics.

To achieve system performances, the internal voltage regulator must be used.

5.4.1 Single Supply

The SAM G55 supports a 1.62V to 3.6V single supply mode. The internal voltage regulator input is connected to the source and its output feeds VDDCORE. Figure 5-2 illustrates the power schematics.

Figure 5-2. Single Supply



11.6 Functional Description

11.6.1 Test Pin

One dedicated pin, TST, is used to define the device operating mode. When this pin is at low level during powerup, the device is in normal operating mode. When at high level, the device is in test mode. The TST pin integrates a permanent pull-down resistor of about 15 kΩ, so that it can be left unconnected for normal operation. Note that when setting the TST pin to low or high level at power up, it must remain in the same state during the duration of the whole operation.

11.6.2 NRST Pin

The NRST pin is bidirectional. It is handled by the on-chip reset controller and can be driven low to provide a reset signal to the external components or asserted low externally to reset the microcontroller. It will reset the Core and the peripherals except the Backup region (RTC, RTT and Supply Controller). There is no constraint on the length of the reset pulse and the reset controller can guarantee a minimum pulse length. The NRST pin integrates a permanent pull-up resistor to VDDIO of about 100 kΩ. By default, the NRST pin is configured as an input.

11.6.3 ERASE Pin

The ERASE pin is used to reinitialize the Flash content (and some of its NVM bits) to an erased state (all bits read as logic level 1). The ERASE pin and the ROM code ensure an in-situ reprogrammability of the Flash content without the use of a debug tool. When the security bit is activated, the ERASE pin provides the capability to reprogram the Flash content. It integrates a pull-down resistor of about 100 kΩ to GND, so that it can be left unconnected for normal operations.

This pin is debounced by SCLK to improve the glitch tolerance. To avoid unexpected erase at powerup, a minimum ERASE pin assertion time is required. This time is defined in Table 39-50 “AC Flash Characteristics”.

The ERASE pin is a system I/O pin and can be used as a standard I/O. At startup, the ERASE pin is not configured as a PIO pin. If the ERASE pin is used as a standard I/O, startup level of this pin must be low to prevent unwanted erasing. Also, if the ERASE pin is used as a standard I/O output, asserting the pin to low does not erase the Flash. For details, please refer to Section 9.2 “Peripheral Signal Multiplexing on I/O Lines”.

11.6.4 Debug Architecture

Figure 11-4 shows the Debug Architecture used in the SAM G55. The Cortex-M4 embeds five functional units for debug:

- SWJ-DP (Serial Wire/JTAG Debug Port)
- FPB (Flash Patch Breakpoint
- DWT (Data Watchpoint and Trace)
- ITM (Instrumentation Trace Macrocell)
- TPIU (Trace Port Interface Unit)

The debug architecture information that follows is mainly dedicated to developers of SWJ-DP emulators/probes and debugging tool vendors for Cortex M4-based microcontrollers. For further details on SWJ-DP see the Cortex M4 technical reference manual.

13.5.2.2 Wakeup from WFE

The processor wakes up if:

- It detects an exception with sufficient priority to cause an exception entry
- It detects an external event signal. See “External Event Input”
- In a multiprocessor system, another processor in the system executes an SEV instruction.

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause an exception entry. For more information about the SCR, see “System Control Register”.

13.5.2.3 External Event Input

The processor provides an external event input signal. Peripherals can drive this signal, either to wake the processor from WFE, or to set the internal WFE event register to 1 to indicate that the processor must not enter sleep mode on a later WFE instruction. See “Wait for Event” for more information.

13.5.3 Power Management Programming Hints

ISO/IEC C cannot directly generate the WFI and WFE instructions. The CMSIS provides the following functions for these instructions:

```
void __WFE(void) // Wait for Event
void __WFI(void) // Wait for Interrupt
```

13.6.5.24 USUB16 and USUB8

Unsigned Subtract 16 and Unsigned Subtract 8

Syntax

```
        op{cond}{Rd,} Rn, Rm
```

where

op	is any of:
	USUB16 Unsigned Subtract 16.
	USUB8 Unsigned Subtract 8.
cond	is an optional condition code, see "Conditional Execution".
Rd	is the destination register.
Rn	is the first operand register.
Rm	is the second operand register.

Operation

Use these instructions to subtract 16-bit and 8-bit data before writing the result to the destination register:

The USUB16 instruction:

1. Subtracts each halfword from the second operand register from the corresponding halfword of the first operand register.
2. Writes the unsigned result in the corresponding halfwords of the destination register.

The USUB8 instruction:

1. Subtracts each byte of the second operand register from the corresponding byte of the first operand register.
2. Writes the unsigned byte result in the corresponding byte of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
USUB16 R1, R0  ; Subtracts halfwords in R0 from corresponding halfword of R1
               ; and writes to corresponding halfword in R1USUB8  R4, R0, R5
               ; Subtracts bytes of R5 from corresponding byte in R0 and
               ; writes to the corresponding byte in R4.
```

13.6.6.5 SMLAL and SMLALD

Signed Multiply Accumulate Long, Signed Multiply Accumulate Long (halfwords) and Signed Multiply Accumulate Long Dual.

Syntax

```
op{cond} RdLo, RdHi, Rn, Rm
op{XY}{cond} RdLo, RdHi, Rn, Rm
op{X}{cond} RdLo, RdHi, Rn, Rm
```

where:

op	is one of: MLAL Signed Multiply Accumulate Long. SMLAL Signed Multiply Accumulate Long (halfwords, X and Y). X and Y specify which halfword of the source registers *Rn* and *Rm* are used as the first and second multiply operand: If *X* is B, then the bottom halfword, bits [15:0], of *Rn* is used. If *X* is T, then the top halfword, bits [31:16], of *Rn* is used. If *Y* is B, then the bottom halfword, bits [15:0], of *Rm* is used. If *Y* is T, then the top halfword, bits [31:16], of *Rm* is used. SMLALD Signed Multiply Accumulate Long Dual. SMLALDX Signed Multiply Accumulate Long Dual Reversed. If the *X* is omitted, the multiplications are bottom × bottom and top × top. If *X* is present, the multiplications are bottom × top and top × bottom.
cond	is an optional condition code, see “Conditional Execution”.
RdHi, RdLo	are the destination registers. *RdLo* is the lower 32 bits and *RdHi* is the upper 32 bits of the 64-bit integer. For SMLAL, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD and SMLA LDX, they also hold the accumulating value.
Rn, Rm	are registers holding the first and second operands.

Operation

The SMLAL instruction:

- Multiplies the two’s complement signed word values from *Rn* and *Rm*.
- Adds the 64-bit value in *RdLo* and *RdHi* to the resulting 64-bit product.
- Writes the 64-bit result of the multiplication and addition in *RdLo* and *RdHi*.

The SMLALBB, SMLALBT, SMLALTB and SMLALTT instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from *Rn* and *Rm*.
- Adds the resulting sign-extended 32-bit product to the 64-bit value in *RdLo* and *RdHi*.
- Writes the 64-bit result of the multiplication and addition in *RdLo* and *RdHi*.

The non-specified halfwords of the source registers are ignored.

The SMLALD and SMLALDX instructions interpret the values from *Rn* and *Rm* as four halfword two’s complement signed 16-bit integers. These instructions:

- If *X* is not present, multiply the top signed halfword value of *Rn* with the top signed halfword of *Rm* and the bottom signed halfword values of *Rn* with the bottom signed halfword of *Rm*.
- Or if *X* is present, multiply the top signed halfword value of *Rn* with the bottom signed halfword of *Rm* and the bottom signed halfword values of *Rn* with the top signed halfword of *Rm*.

13.6.11.20 VMSR

Move to floating-point System Register from ARM Core register.

Syntax

```
VMSR{cond} FPSCR, Rt
```

where:

cond	is an optional condition code, see “Conditional Execution”.
Rt	is the general-purpose register to be transferred to the FPSCR.

Operation

This instruction moves the value of a general-purpose register to the FPSCR. See “Floating-point Status Control Register” for more information.

Restrictions

The restrictions are:

- *Rt* cannot be PC or SP.

Condition Flags

This instruction updates the FPSCR.

13.6.12.1 BKPT

Breakpoint.

Syntax

```
BKPT #imm
```

where:

imm	is an expression evaluating to an integer in the range 0–255 (8-bit value).

Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

imm is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The BKPT instruction can be placed inside an IT block, but it executes unconditionally, unaffected by the condition specified by the IT instruction.

Condition Flags

This instruction does not change the flags.

Examples

```
 BKPT 0xAB   ; Breakpoint with immediate value set to 0xAB (debugger can
             ; extract the immediate value by locating it using the PC)
```

Note: ARM does not recommend the use of the BKPT instruction with an immediate value set to 0xAB for any purpose other than Semi-hosting.

13.8.3.6 Interrupt Priority Registers

Name: NVIC_IPRx [x=0..12]

Access: Read/Write

Reset: 0x000000000

31	30	29	28	27	26	25	24
PRI3							

23	22	21	20	19	18	17	16
PRI2							

15	14	13	12	11	10	9	8
PRI1							

7	6	5	4	3	2	1	0
PRI0							

The NVIC_IPR0–NVIC_IPR12 registers provide a 8-bit priority field for each interrupt. These registers are byte-accessible. Each register holds four priority fields that map up to four elements in the CMSIS interrupt priority array IP[0] to IP[49].

- **PRI3: Priority (4m+3)**

Priority, Byte Offset 3, refers to register bits [31:24].

- **PRI2: Priority (4m+2)**

Priority, Byte Offset 2, refers to register bits [23:16].

- **PRI1: Priority (4m+1)**

Priority, Byte Offset 1, refers to register bits [15:8].

- **PRI0: Priority (4m)**

Priority, Byte Offset 0, refers to register bits [7:0].

Notes:
1. Each priority field holds a priority value, 0–15. The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:4] of each field; bits[3:0] read as zero and ignore writes.
2. For more information about the IP[0] to IP[49] interrupt priority array, that provides the software view of the interrupt priorities, see Table 13-30 “CMSIS Functions for NVIC Control” .
3. The corresponding IPR number *n* is given by *n* = *m* DIV 4.
4. The byte offset of the required Priority field in this register is *m* MOD 4.

13.11.2.4 MPU Region Base Address Register

Name: MPU_RBAR

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR			VALID	REGION			

The MPU_RBAR defines the base address of the MPU region selected by the MPU_RNR, and can update the value of the MPU_RNR.

Write MPU_RBAR with the VALID bit set to 1 to change the current region number and update the MPU_RNR.

- **ADDR: Region Base Address**

Software must ensure that the value written to the ADDR field aligns with the size of the selected region (SIZE field in the MPU_RASR).

If the region size is configured to 4 GB, in the MPU_RASR, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x00000000.

The base address is aligned to the size of the region. For example, a 64 KB region must be aligned on a multiple of 64 KB, for example, at 0x00010000 or 0x00020000.

- **VALID: MPU Region Number Valid**

Write:

0: MPU_RNR not changed, and the processor updates the base address for the region specified in the MPU_RNR, and ignores the value of the REGION field.

1: The processor updates the value of the MPU_RNR to the value of the REGION field, and updates the base address for the region specified in the REGION field.

Always reads as zero.

- **REGION: MPU Region**

For the behavior on writes, see the description of the VALID field.

On reads, returns the current region number, as specified by the MPU_RNR.

- **SIZE: Size of the MPU Protection Region**

The minimum permitted value is 3 (b00010).

The SIZE field defines the size of the MPU memory region specified by the MPU_RNR. as follows:

(Region size in bytes) = $2^{(SIZE+1)}$

The smallest permitted region size is 32B, corresponding to a SIZE value of 4. The table below gives an example of SIZE values, with the corresponding region size and value of N in the MPU_RBAR.

SIZE Value	Region Size	Value of N[1]	Note
b00100 (4)	32 B	5	Minimum permitted size
b01001 (9)	1 KB	10	–
b10011 (19)	1 MB	20	–
b11101 (29)	1 GB	30	–
b11111 (31)	4 GB	b01100	Maximum possible size

Note: 1. In the MPU_RBAR; see “MPU Region Base Address Register”

- **ENABLE: Region Enable**

Note: For information about access permission, see “MPU Access Permission Attributes”.

16.6.28 PIO Input Filter Slow Clock Status Register

Name: PIO_IFSCSR

Address: 0x400E0E88 (PIOA), 0x400E1088 (PIOB)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Glitch or Debouncing Filter Selection Status**

0: The glitch filter is able to filter glitches with a duration < $t_{peripheral\ clock}/2$.

1: The debouncing filter is able to filter pulses with a duration < $t_{div\_slck}/2$.

Table 18-4. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0104	Peripheral Clock Disable Register 1	PMC_PCDR1	Write-only	–
0x0108	Peripheral Clock Status Register 1	PMC_PCSR1	Read-only	0x0000_0000
0x010C	Peripheral Control Register	PMC_PCR	Read/Write	0x0000_0000
0x0110	Oscillator Calibration Register	PMC_OCR	Read/Write	0x0040_4040
0x0114	SleepWalking Enable Register 0	PMC_SLPWK_ER0	Write-only	–
0x0118	SleepWalking Disable Register 0	PMC_SLPWK_DR0	Write-only	–
0x011C	SleepWalking Status Register 0	PMC_SLPWK_SR0	Read-only	0x0000_0000
0x0120	SleepWalking Activity Status Register 0	PMC_SLPWK_ASR0	Read-Only	–
0x0130	PLL Maximum Multiplier Value Register	PMC_PMMR	Read/Write	0x07FF_07FF

Note: If an offset is not listed in the table it must be considered as “reserved”.

23.5.4 EEFC Flash Result Register

Name: EEFC_FRR

Address: 0x400E0A0C

Access: Read-only

31	30	29	28	27	26	25	24
FVALUE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FVALUE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FVALUE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FVALUE							

- **FVALUE: Flash Result Value**

The result of a Flash command is returned in this register. If the size of the result is greater than 32 bits, the next resulting value is accessible at the next register read.

30.6.8.1 Modes of Operation

The USART can operate in SPI Master mode or in SPI Slave mode.

Operation in SPI Master mode is programmed by writing 0xE to the USART_MODE field in US_MR. In this case, the SPI lines must be connected as described below:

- The MOSI line is driven by the output pin TXD.
- The MISO line drives the input pin RXD.
- The SCK line is driven by the output pin SCK.
- The NSS line is driven by the output pin RTS.

Operation in SPI Slave mode is programmed by writing 0xF to the USART_MODE field in US_MR. In this case, the SPI lines must be connected as described below:

- The MOSI line drives the input pin RXD.
- The MISO line is driven by the output pin TXD.
- The SCK line drives the input pin SCK.
- The NSS line drives the input pin CTS.

In order to avoid unpredicted behavior, any change of the SPI mode must be followed by a software reset of the transmitter and of the receiver (except the initial configuration after a hardware reset). See Section 30.6.8.4 ”Receiver and Transmitter Control”.

30.6.8.2 Bit Rate

In SPI mode, the bit rate generator operates in the same way as in USART Synchronous mode (see Section 30.6.1.3 ”Baud Rate in Synchronous Mode or SPI Mode”). However, some restrictions apply:

In SPI Master mode:

- The external clock SCK must not be selected (USCLKS ≠ 0x3), and bit CLKO must be set to ‘1’ in the US_MR, in order to generate correctly the serial clock on the SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the value programmed in CD must be greater than or equal to 6.
- If the divided peripheral clock is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the SCK pin; this value can be odd if the peripheral clock is selected.

In SPI Slave mode:

- The external clock (SCK) selection is forced regardless of the value of the USCLKS field in the US_MR. Likewise, the value written in US_BRGR has no effect, because the clock is provided directly by the signal on the USART SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the external clock (SCK) frequency must be at least 6 times lower than the system clock.

30.6.8.3 Data Transfer

Up to nine data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the US_MR. The nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI mode (Master or Slave).

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the US_MR. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

- **FCS: Force SPI Chip Select**

Applicable if USART operates in SPI Master mode (USART_MODE = 0xE):

0: No effect.

1: Forces the Slave Select Line NSS (RTS pin) to 0, even if USART is not transmitting, in order to address SPI slave devices supporting the CSAAT mode (Chip Select Active After Transfer).

- **RCS: Release SPI Chip Select**

Applicable if USART operates in SPI Master mode (USART_MODE = 0xE):

0: No effect.

1: Releases the Slave Select Line NSS (RTS pin).

32.6 TWI Functional Description

32.6.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 32-3).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 32-2).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

Figure 32-2. START and STOP Conditions

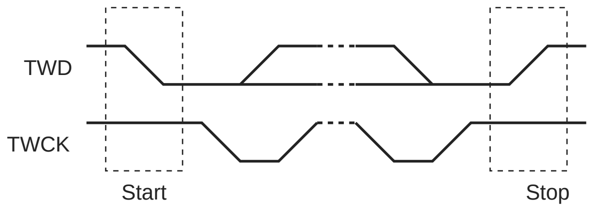
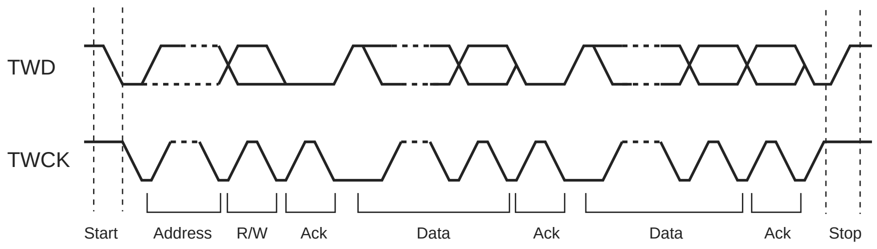


Figure 32-3. Transfer Format



32.6.2 Modes of Operation

The TWI has different modes of operation:

- Master Transmitter mode (Standard and Fast modes only)
- Master Receiver mode (Standard and Fast modes only)
- Multi-master Transmitter mode (Standard and Fast modes only)
- Multi-master Receiver mode (Standard and Fast modes only)
- Slave Transmitter mode (Standard, Fast and High-speed mode)
- Slave Receiver mode (Standard, Fast and High-speed modes)

These modes are described in the following sections.

Figure 32-4. Master Write with One Data Byte

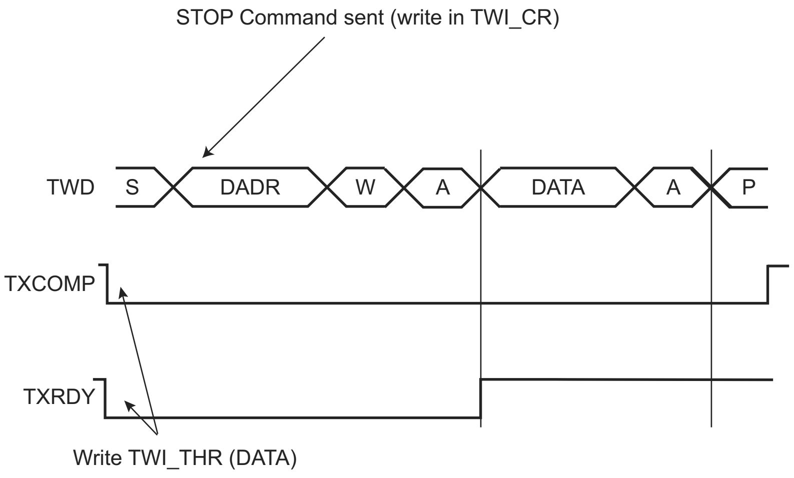


Figure 32-5. Master Write with Multiple Data Bytes

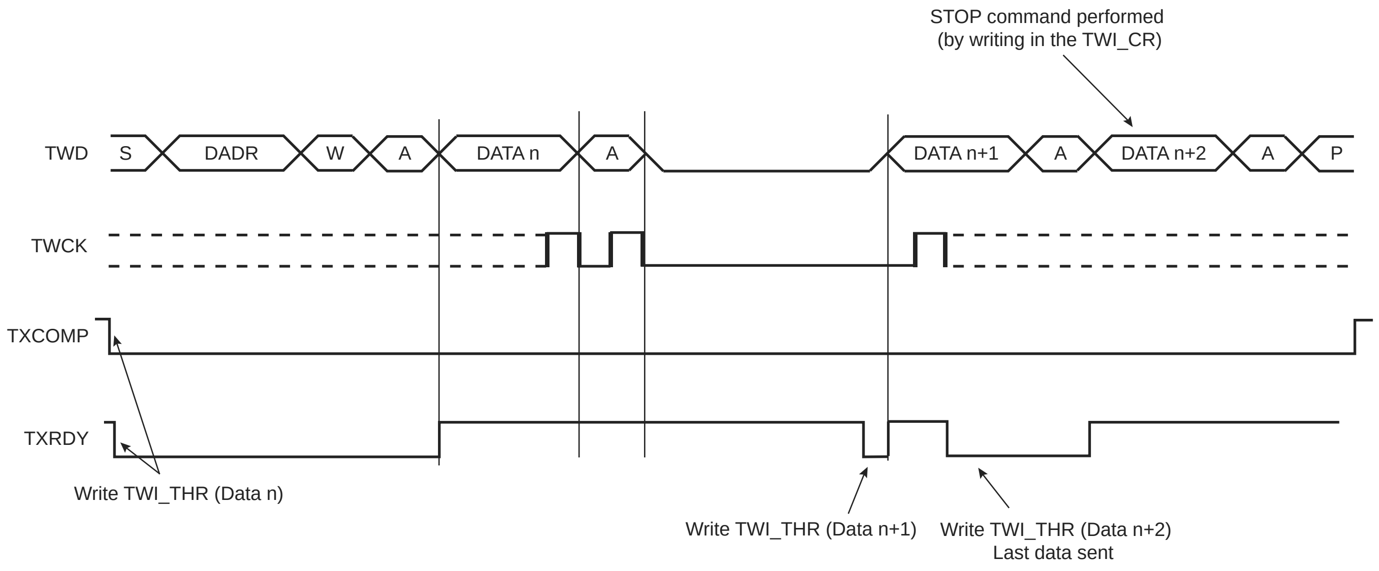
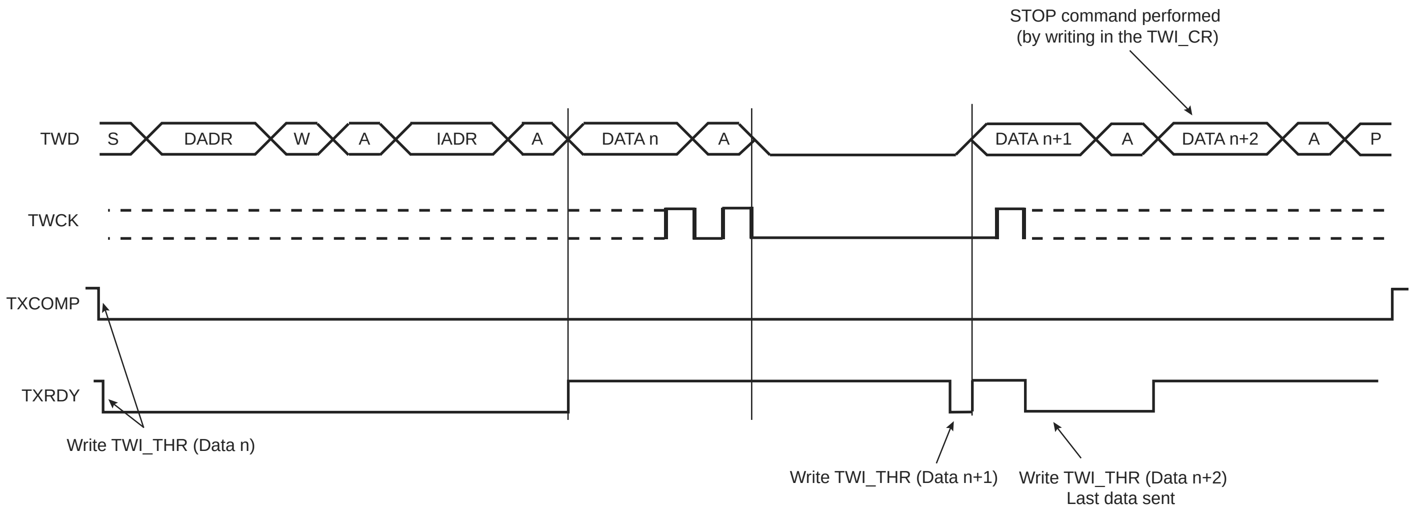


Figure 32-6. Master Write with One Byte Internal Address and Multiple Data Bytes



34.7 Pulse Density Modulation Interface Controller (PDMIC) User Interface

Table 34-4. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00	Control Register	PDMIC_CR	Read/Write	0x00000000
0x04	Mode Register	PDMIC_MR	Read/Write	0x00F00000
0x08–0x10	Reserved	–	–	–
0x14	Converted Data Register	PDMIC_CDR	Read-only	0x00000000
0x18	Interrupt Enable Register	PDMIC_IER	Write-only	–
0x1C	Interrupt Disable Register	PDMIC_IDR	Write-only	–
0x20	Interrupt Mask Register	PDMIC_IMR	Read-only	0x00000000
0x24	Interrupt Status Register	PDMIC_ISR	Read-only	0x00000000
0x28–0x54	Reserved	–	–	–
0x58	DSP Configuration Register 0	PDMIC_DSPR0	Read/Write	0x00000000
0x5C	DSP Configuration Register 1	PDMIC_DSPR1	Read/Write	0x00000001
0x60–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	PDMIC_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	PDMIC_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–
0x100–0x124	Reserved for PDC Registers	–	–	–

Notes: 1. If an offset is not listed in the table, it must be considered as “reserved”.

Access to the USB host operational registers is achieved through the AHB bus slave interface. The OpenHCI host controller initializes master DMA transfers through the ASB bus master interface as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory
- Access to the HC communication area
- Write status and retire transfer Descriptor

Memory access errors (abort, misalignment) lead to an “UnrecoverableError” indicated by the corresponding flag in the host controller operational registers.

The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub’s operational registers. Device connection is automatically detected by the USB host port logic.

USB physical transceivers are integrated in the product and driven by the root hub’s ports.

36.4 Product Dependencies

36.4.1 I/O Lines

DPs and DMs are not controlled by any PIO controllers. The embedded USB physical transceivers are controlled by the USB host controller.

36.4.2 Power Management

The USB host controller requires a 48 MHz clock. This clock must be generated by a PLL with a correct accuracy of ± 0.25%.

Thus the USB device peripheral receives two clocks from the Power Management Controller (PMC): the Master clock (MCK) used to drive the peripheral user interface (MCK domain) and the UHPCLK 48 MHz clock used to interface with the bus USB signals (Recovered 12 MHz domain).

36.4.3 Interrupt Sources

The USB host interface has an interrupt line connected to the interrupt controller.

Handling USB host interrupts requires programming the interrupt controller before configuring the UHP.

Table 36-1. Peripheral IDs

Instance	ID
UHP	47

- **PSSC: Port 2 suspend status changed (read/write, write ‘1’ to clear)**

A write of 1 clears this bit; a write of 0 has no effect.

0: Port 2 port suspend status has not changed.

1: Port 2 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed.

- **OCIC: Port 2 Overcurrent Indicator Change (read/write)**

0: Because the device does not provide inputs for signaling external overcurrent indication to the USB1.1 host controller, this bit is always 0. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS.

- **PRSC: Port 2 Reset Status Change (read/write, write ‘1’ to clear)**

A write of 1 clears this bit; a write of 0 has no effect.

0: Port 2 port reset status bit has not changed.

1: Port 2 port reset status bit has changed.