AXM0F343 MCU Programming Manual

UM70012/D

ON Semiconductor®

www.onsemi.com

USER MANUAL

Device Overview

The AXM0F343 System−on−Chip (SoC) family is available with AXM0F343−64 or AXM0F343−256 ultra−low power Micro Controller Units optimized for short range radio application. AXM0F343−64 MCU and AXM0F343−256 MCU integrate the powerful and energy efficient Arm® Cortex®−M0+ microprocessor, Program Flash memory, Embedded RAM, a DMA controller, and 19 GPIO. Peripherals include: USART, SPI, I^2C, timers, Capture/PWM, 12−bit 1M sample/s ADC, and analog comparators. Security features include AES and CRC acceleration engines and a true random number generator. AXM0F343 implements advanced low−power modes for ultra−low power consumption.

Features

Arm Cortex−M0+

- ARMv6−M Architecture
- Thumb®/Thumb−2 subset instruction set
- 2 stage pipeline
- Nested Vectored Interrupt Controller (NVIC) includes 15 built−in Arm core exceptions and is configured with an additional 24 interrupts
- Non−Maskable Interrupt (NMI)
- Sleep support
- Wake−up Interrupt Controller (WIC)
- SysTick timer for scheduler
- Low latency General Purpose Input/Output (GPIO) on the single cycle Input/Output port (IOP) Bus
- Single cycle 32x32 multiply

Debugger

- Serial−Wire Debug Access Port (SW−DAP)
- Breakpoint and single stepping support
- Micro Trace Buffer (MTB)
- Debug port lockout

Memory

- AXM0F343−64 MCU:
  - Contains 64 kB of FLASH
  - 8 kB of RAM: 2 kB and 6 kB banks with separate power and retention control
- AXM0F343−256 MCU:
  - 256 kB of FLASH
  - 32 kB of RAM: 8 kB and 24 kB banks with separate power and retention control

Flexible Clocking

- On−chip high speed (32/40 MHz) RC oscillator
- On−chip low power (10 kHz/640 Hz) RC oscillator
- High speed crystal oscillator driver
- Low power 32.768 kHz crystal oscillator driver
- External clock through GPIO
- Fully automatic calibration of on−chip RC oscillators to a reference clock
- Clock monitor can detect failures of the system clock and switch to another clock

Timers

- 32−bit wakeup timer
- 32−bit TICK timer
- General Purpose 16−bit timers (3X)
  - Up count, down count (saw tooth) and up/down count (triangle) modes
  - Sigma−Delta DAC modulator mode
  - Flexible clocking and pre−scale options
- Watchdog timer

16−bit Input Capture/Output Compare/PWM Units (4X)

- Paired with any 16−bit general purpose timer
  - Timer value capture at event trigger
  - Compare flag at timer match
- Differential PWM with programmable dead time
- AXM0F343−64 MCU TIMER/PWM
  - Shutdown Operation
  - Comparator DAC for Reset
  - Asynchronous Reset
  - Variable Frequency Operation

19 GPIO

- 6 GPIO (PA0−PA5), 8 GPIO (PB0−PB7), and 5 GPIO (PC0−PC4) can be used for both digital and analog functions
- Programmable pull−up/pull−down
- Open drain capable
- Programmable interrupts (edge/level, polarity)

Features (continued)

- Flexible allocation of GPIO pins to peripherals through programmable crossbar

USART (2x)

- 5−9 bit word length, 1−2 stop bits
- Uses any of the 16−bit general purpose timers as baud rate generator

Master/Slave SPI

- Programmable data width and direction
- Programmable phase and polarity
- Support 3−wire or 4−wire modes

12−bit ADC

- 12−bit Max 1M samples/second
- Up to 6 external sample channels (3 differential)
- Single ended or differential sampling
- 1/4x, 1x, and 10x gain amplifier
- External or internal (1 V) reference
- Programmable conversion schedule
- Can sample internal supply and reference signals including the built−in temperature sensor

Analog Comparator (2x)

- Internal or external reference
- Output signal may be routed to GPIO, read by software, or used as input capture trigger

DMA Controller

- Arm PL230 μDMA
- 3 channels with programmable source and priority
- Supports memory−to−memory, memory−to−peripheral, and peripheral−to−memory
- Supports ping−pong, memory scatter−gather, and peripheral scatter−gather DMA cycle types
- Supports multiple transfer data widths
- 1−1024 transfers in a DMA cycle

Security Features

- Hardware AES Acceleration
  - Significant reduction in encryption/decryption time compared to software only solution
  - Supports 128−bit, 192−bit, and 256−bit encryption/decryption
- CRC Engine
  - Standard Ethernet 32−bit polynomial
  - Used to verify integrity of the flash memory
- True random number generator
- Debug port lock

Table of Contents

BLOCK DIAGRAM

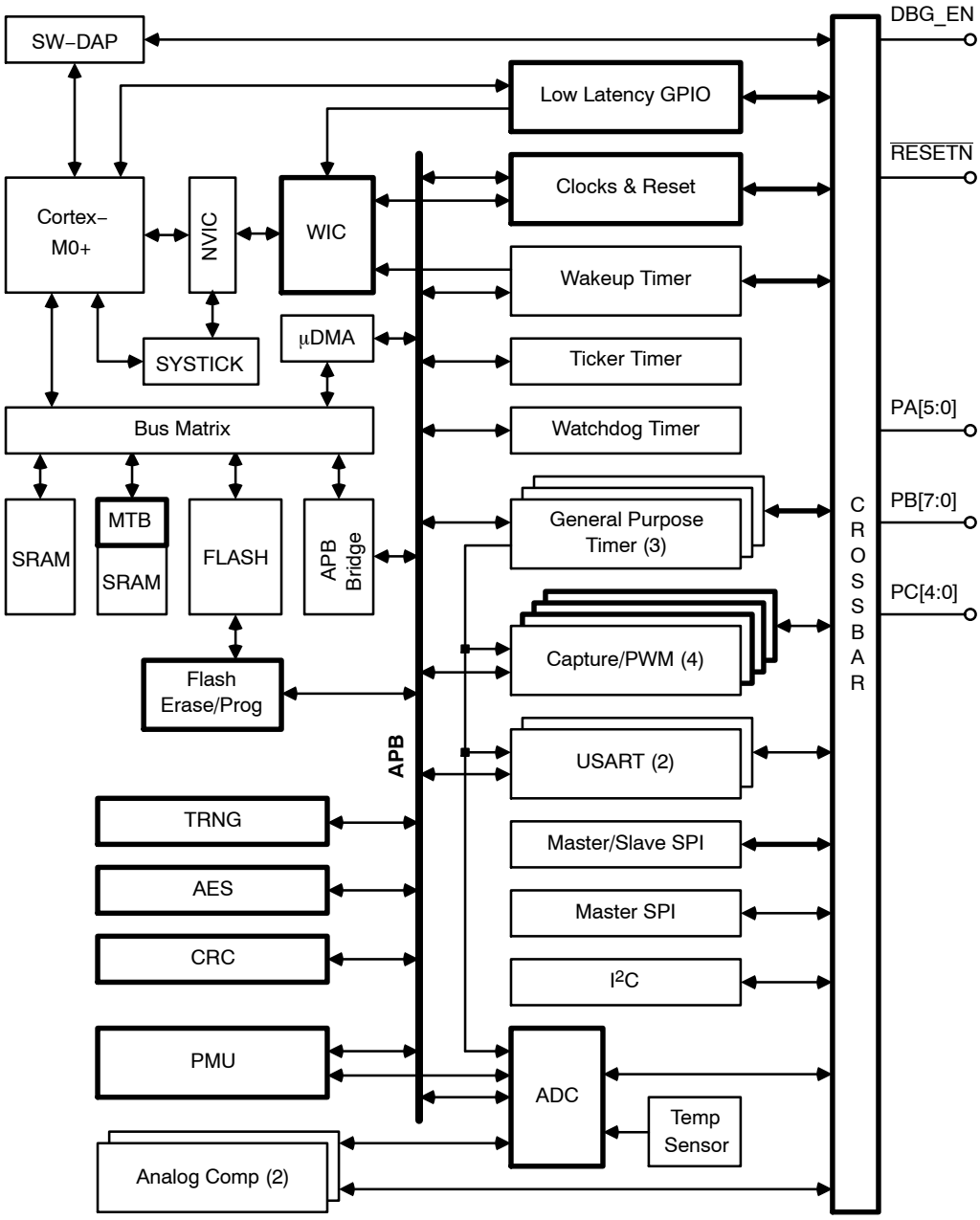


Figure 1. Block Diagram

ACRONYMS

Table 1. ACRONYMS

Acronym	Description
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AHB	Advanced High Performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
CCPWM	Compare/Capture/Pulse Width Modulation
CMP	Comparator
CMU	Clock Monitor Unit
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DAP	Debug Access Port
DMA	Direct Memory Access Controller
GPIO	General Purpose Input Output
I2C	Inter−Integrated Circuit Controller
MCU	Micro Controller Unit
MTB	Micro Trace Buffer
NMI	Non−Maskable Interrupt
NVIC	Nested Vectored Interrupt Controller
NVM	Non Volatile Memory
PMU	Power Management Unit
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
SW−DAP	Serial−Wire Debug Access Port
TRNG	True Random Number Generator
USART	Universal Synchronous Asynchronous Receiver Transmitter
WDOG	Watchdog timer
WIC	Wake−up Interrupt Controller
WUT	Wake Up Timer
XBAR	Crossbar

Cortex−M0+ MICROCONTROLLER AND BUS FABRIC

AXM0F343 MCU integrates the powerful and energy efficient Arm Cortex−M0+ processor that includes the integrated Nested Vectored Interrupt Controller (NVIC), Wake−up Interrupt Controller(WIC), and Debug Access Port (DAP). The processor uses the Thumb instruction set and is optimized for high performance with reduced code size and low power operation. The Arm Cortex−M0+ efficiently handles multiple parallel peripherals and has integrated sleep modes. With industry standard tool chain and support, developing applications on the AXM0F343 platform reduces time to market. Test and debug capability is enhanced with the Arm Serial Wire Debug Port and Micro Trace Buffer. The microprocessor uses little−endian formatting.

The microprocessor, debug port, and memories are interconnected using the Advanced Microcontroller Bus Architecture (AMBA bus) AHB−Lite system interface bus. An AHB to APB Bridge is included to connect the peripherals.

Next to the regular Arm Cortex−M0+ processor interrupts, the AXM0F343 MCU implements multiple external source interrupts for peripheral devices. A powerful nested, pre−emptive and priority based interrupt handling assure timely and flexible response to external events.

Low power features on AXM0F343 MCU include the WIC, adjustable clock rates, and different software controlled power modes to maximize opportunities to save power in final application.

Serial Wire Debug Access Port (SW−DAP)

The Debug Access Port is included in the Arm Cortex−M0+ implementation. The basic debug functionality includes processor halts, single−step, processor core register access, Reset and HardFault Vector Catch, unlimited software breakpoints, and full system memory access. The debug mode implementation also includes 4 hardware breakpoints and 2 hardware watch points. The Debug Port is disabled at power−up if the part is locked, and may be enabled by firmware using the Lock Override Register. Driving the debug enable pin high will prevent the part from entering a low power mode (see the PMU description).

The Debug Access Port interface implementation is the Arm Serial Wire Debug Port (SW−DP) connected to pins SWCLK and SWDIO. SWO is not implemented. The Serial Wire Debug Port Interface uses a single bi−directional data connection. Each operation consists of three phases: Packet request, Acknowledge response, and Data transfer phase. Use any Serial Wire Debug (SWD) compliant hardware debugger interface to interact with the internals of the AXM0F343 MCU.

Idle and reset: Between transfers, the host must either drive the line LOW to the IDLE state, or continue immediately with the start bit of a new transfer. The host is also free to leave the line HIGH, either driven or tri−stated, after a packet. This reduces the static current drain, but if this approach is used with a free running clock, a minimum of 50 clock cycles must be used, followed by a READ−ID as a new re−connection sequence. There is no explicit reset signal for the protocol. A reset is detected by either host or target when the expected protocol is not observed. It is important that both ends of the link become reset before the protocol can be restarted with a reconnection sequence. Re−synchronization following the detection of protocol errors or after reset is achieved by providing 50 clock cycles with the line HIGH, or tristate, followed by a read ID request. If the SW−DP detects that it has lost synchronization, for example if no stop bit is seen when expected, it leaves the line un−driven and waits for the host to either re−try with a new header after a minimum of one cycle with the line LOW, or signals a reset by not driving the line itself. If the SW−DP detects two bad data sequences in a row, it locks out until a reset sequence of 50 clock cycles with DBGDI HIGH is seen. If the host does not see an expected response from SW−DP, it must allow time for SW−DP to return a data payload. The host can then retry with a read to the SW−DP ID code register. If this is unsuccessful, the host must attempt a reset.

Nested Vectored Interrupt Controller (NVIC)

The Cortex−M0+ Nested Vectored Interrupt Controller (NVIC) supports priority based nested vectored interrupts. It includes 15 built−in or reserved exceptions and is configured with an additional 22 interrupts. Most interrupts have programmable priority. Priority levels available in the NVIC are 0, 64, 128, and 192. Lower numbers are higher priority. The priority of each group can be set separately by the firmware. While an interrupt is being serviced, only interrupts from a higher priority group will be serviced. If two interrupts of the same priority arrive at the same time, the earlier one (according to polling order) will be serviced first. The optional Wake−up Interrupt Controller (WIC) is included for low power mode support. Only a subset of the interrupts are included in the wake−up controller.

Exceptions and Interrupts Table

Table 2. EXCEPTIONS AND INTERRUPTS TABLE

Interrupt	Exception Number	Priority	Available in Sleep	Available in Hibernate	Available in Shutdown
Cortex M0+ – Reset	1	–3 (Highest)	Yes	No	No
Cortex M0+ – NMI (external pin, clock loss, or brownout)	2	–2	Yes	Yes	No
Cortex M0+ – Hard Fault	3	–1	No	No	No
Cortex M0+ Reserved	4–10	N/A	N/A	N/A	N/A
Cortex M0+ SVC	11	Programmable	No	No	No
Cortex M0+ Reserved	12–13	N/A	N/A	N/A	N/A
Cortex M0+ PendSV	14	Programmable	No	No	No
Cortex M0+ SysTick	15	Programmable	No	No	No
General Purpose I/O (GPIO)	16	Programmable	Yes	Yes	PB3
Wakeup Timer (WUT)	17	Programmable	Yes	Yes	No
Tick Timer (TICK)	18	Programmable	Yes	No	No
External Pin	19	Programmable	Yes	Yes	No
Analog Comparator (CMP)	20	Programmable	Yes	No	No
Analog to Digital Convertor (ADC)	21	Programmable	Yes	No	No
True Random Number Generator (TRNG)	22	Programmable	Yes	No	No
FLASH	23	Programmable	No	No	No
Watchdog (WDOG)	24	Programmable	No	No	No
Clock/System Config (CMU)	25	Programmable	Yes	No	No
Timer0 (TIM0)	26	Programmable	Yes	No	No
Timer1 (TIM1)	27	Programmable	Yes	No	No
Timer2 (TIM2)	28	Programmable	Yes	No	No
Capture/PWM 0 (CPMW0)	29	Programmable	Yes	No	No
Capture/PWM 1 (CPMW1)	30	Programmable	Yes	No	No
Capture/PWM 2 (CPMW2)	31	Programmable	Yes	No	No
Capture/PWM 3 (CPMW3)	32	Programmable	Yes	No	No
––Reserved––	33	N/A	N/A	N/A	N/A
Master/Slave SPI (SPI)	34	Programmable	Yes	No	No
Universal Synchronous/Asynchronous Receiver/Transmitter 0 (USART0)	35	Programmable	Yes	No	No
Universal Synchronous/Asynchronous Receiver/Transmitter 1 (USART1)	36	Programmable	Yes	No	No
I^2C (I2C)	37	Programmable	Yes	No	No
DMA_ERROR	38	Programmable	Yes	No	No
DMA_DONE	39	Programmable	Yes	No	No

AHB−APB Bridge with Atomic RMW

The APB peripheral bus is connected to the AHB bus using a bridge that includes support for Atomic Read−Modify−Write capability. APB peripheral reads ignore the Atomic RMW bits, reading the target register contents normally for all settings.

Address bits [27:26] are used to configure an APB write as one of four modes:

Table 3.

Address [27:26]	Write Operation
00	Direct Write − a normal write operation which transfers the write data directly into the targeted peripheral register overwriting the previous contents.
01	Clear or AND'ed Write − used for bit clearing as it first AND's the contents of the target register with the write data and places the result in the target register.
10	Set or OR'ed Write − used for bit setting as it first OR's the contents of the target register with the write data and places the result in the target register.
11	Inverted or XOR'ed Write − used for bit toggling as it first XOR's the contents of the target register with the write data and places the result in the target register.

Usage of the atomic read−modify−write capability can facilitated through macros:

```
#define ATOMIC_CLEAR(x)    *((volatile uint32_t *) ((uint32_t) (&(x)) + (1<<26))
#define ATOMIC_SET(x)      *((volatile uint32_t *) ((uint32_t) (&(x)) + (2<<26))
#define ATOMIC_TOGGLE(x)   *((volatile uint32_t *) ((uint32_t) (&(x)) + (3<<26))
```

Or alternatively using #define aliases for each peripheral:

```
#define AND_OFFSET         0x04000000UL
#define OR_OFFSET          0x08000000UL
#define XOR_OFFSET         0x0C00000UL
#define GPIO_AND           ((GPIO_Type*) (GPIO_BASE + AND_OFFSET))
#define GPIO_OR            ((GPIO_Type*) (GPIO_BASE + OR_OFFSET))
#define GPIO_XOR           ((GPIO_Type*) (GPIO_BASE + XOR_OFFSET))
```

MEMORY

The 32−bit memory address space is broken up into regions for code, data, and multi−use. Memory elements consist of registers, SRAM, and NVM (Flash). A memory region is dedicated to the IOP and APB peripheral access. There are also built in regions for built in Arm registers and peripherals. There are many unused portions of the memory space. Attempted access to these unused regions will result in a memory fault. Individual memory regions and elements are described in the next few sections.

The AXM0F343 MCU is available in two different memory configurations:

- AXM0F343−64 which contains 64 kB of FLASH and 8 kB of RAM
- AXM0F343−256 which contains 256 kB of FLASH and 32 kB of RAM

AXM0F343−64 MCU Memory Map

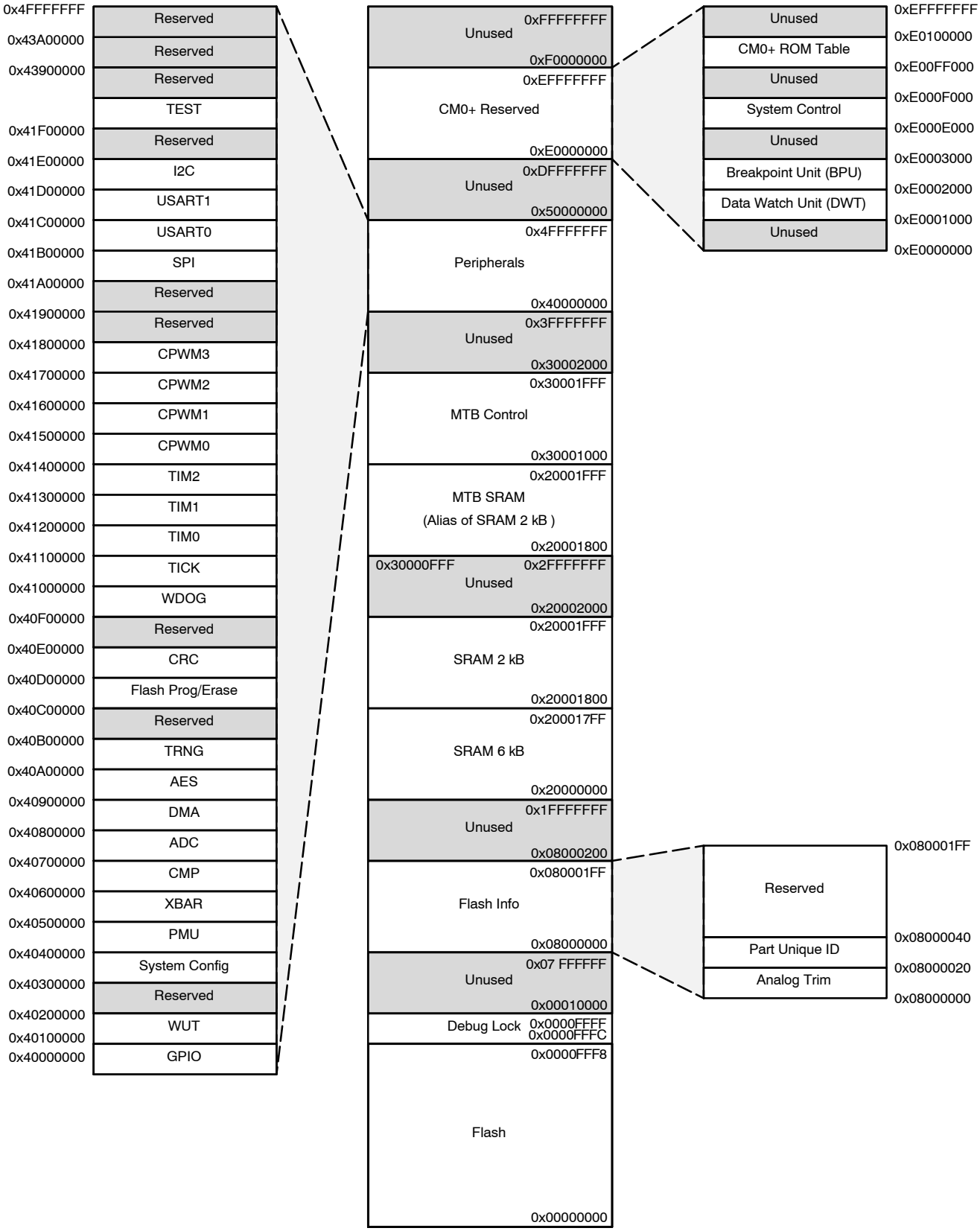


Figure 2. AXM0F343−64 MCU Memory Map

AXM0F343 MCU Memory Map

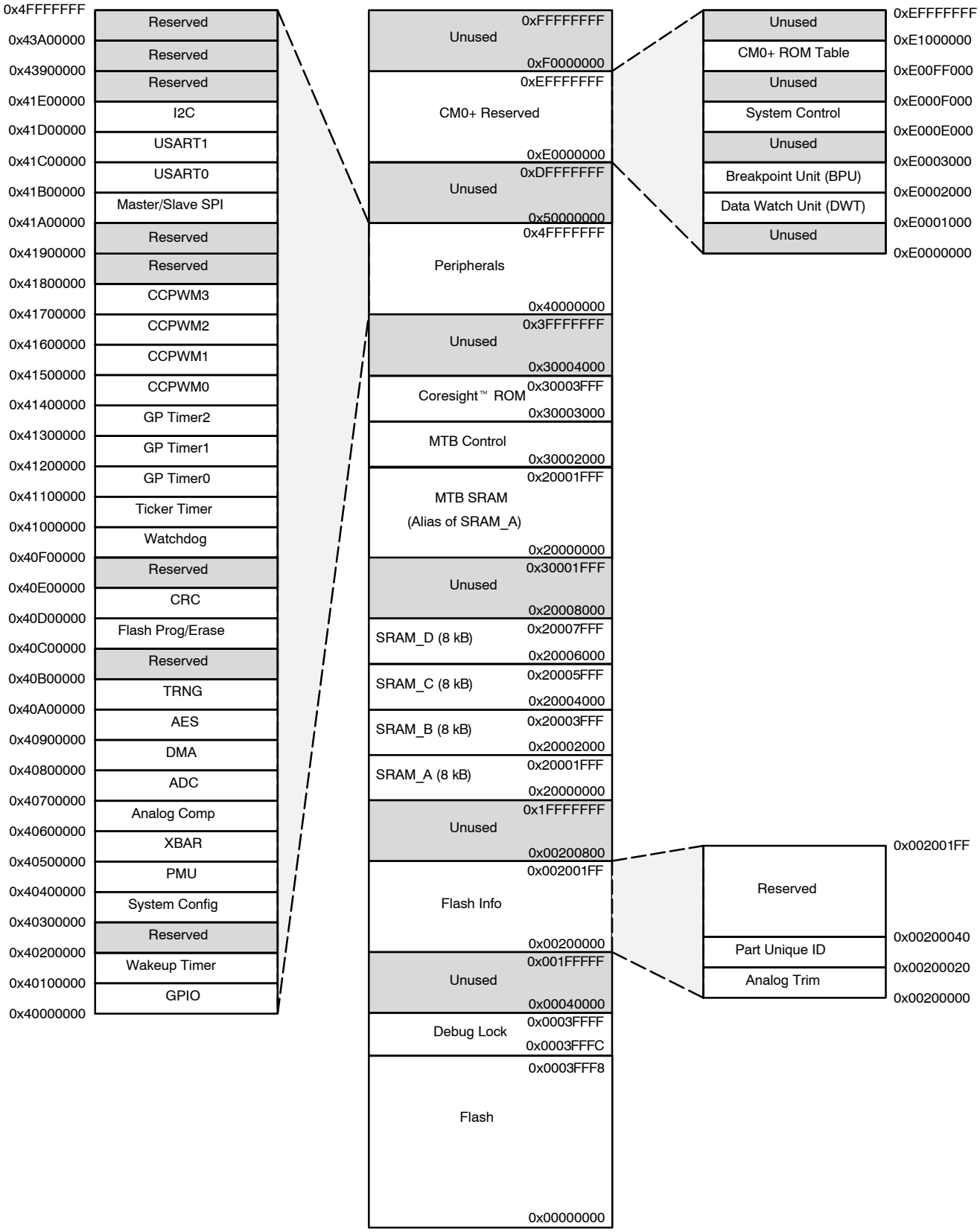


Figure 3. AXM0F343−256 MCU Memory Map

Peripheral Memory Address Decode

The APB and IOP peripherals are accessed in the address range of 0x40000000 to 0x4FFF_FFFF. Within this range peripheral addresses are decoded as such.

Table 4.

Bit Range	Purpose
[31:28]	Must be 0x4 to access peripheral space.
[27:26]	Atomic read−modify−write operation. Ignored for reads. 00 − Direct write, 01 − Clear write, 10 − Set write, 11 − Toggle write
[25]	Generally this bit is ignored and will alias if used.
[24:20]	Peripheral select decode.
[19:12]	Generally these bits are ignored and will alias if used.
[11:2]	Available for register offset within peripheral.
[1:0]	Byte select within peripheral register. Most, but not all peripherals support byte and half−word accesses.

Peripheral Register Table Summary

Table 5. PERIPHERAL REGISTER TABLE SUMMARY

Peripheral Register	Address	Access	Description
GPIO			
DATA	0x40000000	RO	Current GPIO value for read
DATAOUT	0x40000004	R/W	Value to apply to GPIO outputs
OUT_EN	0x40000008	R/W	Output enable for GPIO
INT_EN	0x4000000C	R/W	GPIO interrupt enable
INT_POL	0x40000010	R/W	GPIO interrupt polarity
INT_TYP	0x40000014	R/W	GPIO interrupt type
DMA_TRIG_EN	0x40000018	R/W	GPIO DMA enable
INT_STS	0x4000001C	R/W	GPIO interrupt status
WAKEUP TIMER (WUT)			
CFG	0x40100000	R/W	Wakeup timer configuration including clock source and pre−scale
STAT	0x40100004	RO	Wakeup timer status register
CNT	0x40100008	RO	Wakeup timer value
EVT	0x4010000C	R/W	Wakeup timer event matching value
CLOCK AND SYSTEM CONFIGURATION (CMU)			
CFG	0x40300000	R/W	System clock configuration
STS	0x40300004	RO	Clock status
ADCCLK_CFG	0x40300008	R/W	ADC clock source and pre−scale
EXTCLK_CFG	0x4030000C	R/W	External clock source select and clock divide
PCLK_EN	0x40300010	R/W	Enables the individual peripheral clocks
LPOSC_CFG	0x40300014	R/W	Low power RC oscillator configuration
LPOSC_FILT	0x40300018	R/W	Low power RC oscillator calibration filter
LPOSC_REF_DIV	0x4030001C	R/W	Low power RC oscillator calibration reference divide
LPOSC_FREQ_TUNE	0x40300020	R/W	Low power RC oscillator calibration frequency tune value
LPOSC_PER	0x40300024	RO	Low power RC oscillator calibration measured value
HS_OSC_CFG	0x40300028	R/W	High speed RC oscillator configuration

Table 5. PERIPHERAL REGISTER TABLE SUMMARY (continued)

Peripheral Register	Address	Access	Description
CLOCK AND SYSTEM CONFIGURATION (CMU)			
HSOSC_FILT	0x4030002C	R/W	High speed RC oscillator calibration filter
HSOSC_REF_DIV	0x40300030	R/W	High speed RC oscillator calibration reference divide
HSOSC_40M_FREQ_TUNE	0x40300034	R/W	High speed RC oscillator 40MHz calibration frequency tune value
HSOSC_32M_FREQ_TUNE	0x40300038	R/W	High speed RC oscillator 32MHz calibration frequency tune value
HSOSC_PER	0x4030003C	RO	High speed RC oscillator calibration measured value
RESERVED	0x40300040	–	
OSC_READY_STS	0x40300044	RO	Indicates if clock sources are enabled and stable
INT_STS	0x40300048	R/W	Clock interrupt enables
LPXOSC_GM_CFG	0x4030004C	R/W	Low power crystal oscillator transconductance configuration
RESERVED	0x40300050	–	
HS_XOSC_GM_CFG	0x40300054	R/W	High speed crystal oscillator transconductance configuration
HSXOSC_AMPL_CFG	0x40300058	R/W	Controls the crystal transconductance servo loop
HSXOSC_READY	0x4030005C	R/W	Crystal oscillator ready mode and status
AO_SCRATCH	0x40300060	R/W	Always on scratch register for retention in ultra−low power modes
IRQLATENCY	0x40300064	R/W	Controls the latency controls for interrupts
POWER MANAGEMENT UNIT (PMU)			
MOD	0x40400000	R/W	Controls power mode options
PD	0x40400004	R/W	Powers down the SRAM banks
STS	0x40400008	RO	Power mode status
CROSSBAR (XBAR)			
ANA_EN	0X40500000	R/W	PA0−PA5 analog function enable
OUT_CFG	0X40500004	R/W	Pin drive type configuration
PULL_UP_CFG	0X40500008	R/W	Pin pullup enables
PULL_DOWN_CFG	0X4050000C	R/W	Pin pulldown enables
PA_CFG	0X40500010	R/W	Bank A output pin function configuration
PB_CFG	0X40500014	R/W	Bank B output pin function configuration
PC_CFG	0X40500018	R/W	Bank C output pin function configuration
RESERVED	0X4050001C	–	
IN_USART0_CFG	0X40500020	R/W	USART0 input pin configuration
IN_USART1_CFG	0X40500024	R/W	USART1 input pin configuration
IN_SPI_CFG	0X40500028	R/W	Master/Slave SPI input pin configuration
RESERVED	0X4050002C	–	
IN_DMA_CFG	0X40500030	R/W	DMA trigger source configuration
IN_ADC_CFG	0X40500034	R/W	ADC trigger source configuration
IN_TIM0_CFG	0X40500038	R/W	Timer0 source clock configuration
IN_TIM1_CFG	0X4050003C	R/W	Timer1 source clock configuration
IN_TIM2_CFG	0X40500040	R/W	Timer2 source clock configuration
IN_CAPT0_CFG	0X40500044	R/W	Capture0 source trigger configuration
IN_CAPT1_CFG	0X40500048	R/W	Capture1 source trigger configuration
IN_CAPT2_CFG	0X4050004C	R/W	Capture2 source trigger configuration
IN_CAPT3_CFG	0X40500050	R/W	Capture3 source trigger configuration

Table 5. PERIPHERAL REGISTER TABLE SUMMARY (continued)

Peripheral Register	Address	Access	Description
CROSSBAR (XBAR)			
IN_EXTCLK_CFG	0X40500054	R/W	External source clock configuration
<Test Reserved>	0X40500058	R/W	Don't use
IN_MTB_CFG	0X4050005C	R/W	MTB (TSTART/TSTOP) source configuration
EXT_INT_SEL	0X40500060	R/W	Connects external interrupt source and set polarity
EXT_INT_EN	0X40500064	R/W	Enables interrupt generation from the selected external interrupt pin
EXT_INT_STS	0X40500068	R/W	Shows the interrupt status selected in EXT_IN_SEL
NMI_CFG	0X4050006C	R/W	Configure External Non−Maskable Interrupt
BRO_STS	0X40500070	R/W	Shows the Brownout status
PWM_EXT_IN	0X40500074	R/W	AXM0F343−64 MCU Timer or PWM Shutdown reset external pin
ANALOG COMPARATOR (CMP)			
OUT	0x40600000	RO	Read the output of the comparator values
CFG	0x40600004	R/W	Analog comparator configuration
INT_EN	0x40600008	R/W	Analog comparator interrupt enables
INT_STS	0x4060000C	R/W	Analog comparator interrupt status
ANALOG DIGITAL CONVERTER (ADC)			
DATA	0x40700000	RO	ADC result
CFG	0x40700004	R/W	ADC configuration
CTL	0x40700008	R/W	ADC trigger and mode controls
CAL	0x4070000C	R/W	ADC calibration trigger
INT_EN	0x40700010	R/W	ADC interrupt enables
INT_STS	0x40700014	R/W	ADC interrupt status
RESERVED	0x40700018	R/W	Don't use
CH_STS	0x40700018	RO	ADC channel from last completed measurement
STS	0x4070001C	R/W	ADC status
DIRECT MEMORY ACCESS CONTROLLER (DMA)			
STS	0x40800000	RO	DMA status
CFG	0x40800004	WO	DMA configuration
CTL_BASE_PTR	0x40800008	R/W	Pointer to the address of the primary data structure
ALT_CTL_BASE_PTR	0x4080000C	RO	Pointer to the address of the alternative data structure
WAIT_ON_REQ_STS	0x40800010	RO	DMA wait on request status
SW_REQ	0x40800014	WO	SW initiated DMA request trigger
USE_BURST_SET	0x40800018	R/W	Sets the option to use bursts
USE_BURST_CLR	0x4080001C	WO	Clears the option to use bursts
REQ_MASK_SET	0x40800020	R/W	Sets the DMA request mask
REQ_MAST_CLR	0x40800024	WO	Clears the DMA request mask
EN_SET	0x40800028	R/W	Enables the DMA channel
EN_CLR	0x4080002C	WO	Disables the DMA channel
PRI_ALT_SET	0x40800030	R/W	Sets the DMA primary/alternate select
PRI_ALT_CLR	0x40800034	WO	Clears the DMA primary/alternate select
PRI_SET	0x40800038	R/W	Sets the DMA channel priority register
PRI_CLR	0x4080003C	WO	Clears the DMA channel priority register

Table 5. PERIPHERAL REGISTER TABLE SUMMARY (continued)

Peripheral Register	Address	Access	Description
DIRECT MEMORY ACCESS CONTROLLER (DMA)			
RESERVED	0x40800040 – 0x40800048	–	
ERR_CLR	0x4080004C	R/W	DMA error status and clear
INT_EN	0x40800050	R/W	DMA interrupt enables
INT_STS	0x40800054	R/W	DMA interrupt status
ADVANCED ENCRYPTION STANDARD (AES)			
CTL	0x40900000	R/W	AES operation to perform
INPUT	0x40900004	WO	AES data to operate on
OUTPUT	0x40900008	RO	AES result
TRUE RANDOM NUMBER GENERATOR (TRNG)			
CFG	0x40A00000	R/W	Random number configuration and status
DATA	0x40A00000	RO	Random number result
FLASH CONTROL (FLASH)			
CTL	0x40C00000	R/W	Flash program and erase control
ADDR	0x40C00004	R/W	Address to be programmed or erased
DATA	0x40C00008	R/W	Data to program into the flash
INT_EN	0x40C0000C	R/W	Flash interrupt enable
RESERVED	0x40C00010	R/W	Don't use
INT_STS	0x40C00014	R/W	Flash interrupt status
PROT0	0x40C00018	R/W	Flash page 0−31 protection
PROT1	0x40C0001C	R/W	Flash page 32−63 protection
PROT2	0x40C00020	R/W	Flash page 64−95 protection
PROT3	0x40C00024	R/W	Flash page 96−127 protection
DMA_EN	0x40C00028	R/W	Enable DMA trigger generation on erase/program complete
CYCLIC REDUNDANCY CHECK (CRC)			
RESULT	0x40D00000	R/W	Current CRC value
INPUT	0x40D00004	WO	Data to add to the CRC
WATCHDOG (WDOG)			
LOAD	0x40F00000	R/W	Watchdog load value
CNT	0x40F00004	RO	Current watchdog counter value
CFG	0x40F00008	R/W	Watchdog configuration
RESERVED	0x40F0000C	–	Don't use
INT_STS_RAW	0x40F00010	RO	Watchdog raw interrupt status
INT_STS	0x40F00014	R/W	Watchdog interrupt status
RESERVED	0x40F00018 – 0x40F00BFC	–	Don't use
LOCK	0x40F00C00	R/W	Watchdog unlock
RESERVED	0x40F00C04 – 0x40F00EFC	–	Don't use
RESERVED	0x40F00F00 – x40F00F04	–	Don't use

Table 5. PERIPHERAL REGISTER TABLE SUMMARY (continued)

Peripheral Register	Address	Access	Description
TICK TIMER (TICK)			
CFG	0x41000000	R/W	TICK timer configuration
STS	0x41000004	RO	TICK timer status
CNT	0x41000008	RO	TICK timer counter value
EVT	0x4100000C	R/W	TICK timer event match value
16−bit GENERAL PURPOSE TIMER 0 (TIM0)			
CNT	0x41100000	R/W	Timer 0 count value
PER	0x41100004	R/W	Timer 0 rollover value, step value, or ΣΔ DAC code
CFG	0x41100008	R/W	Timer 0 configuration
INT_STS	0x4110000C	R/W	Timer 0 status
16−bit GENERAL PURPOSE TIMER 1 (TIM1)			
CNT	0x41200000	R/W	Timer 1 count value
PER	0x41200004	R/W	Timer 1 rollover value, step value, or ΣΔ DAC code
CFG	0x41200008	R/W	Timer 1 configuration
STS	0x4120000C	R/W	Timer 1 status
16−bit GENERAL PURPOSE TIMER 2 (TIM2)			
CNT	0x41300000	R/W	Timer 2 count value
PER	0x41300004	R/W	Timer 2 rollover value, step value, or ΣΔ DAC code
CFG	0x41300008	R/W	Timer 2 configuration
STS	0x4130000C	R/W	Timer 2 status
COMPARE/CAPTURE/PWM UNIT 0 (CPWM0)			
CFG	0x41400000	R/W	CPWM0 configuration
STS	0x41400004	R/W	CPWM0 status
DATA	0x41400008	R/W	CPWM0 capture data and PWM threshold
Shutdown	0x4140000C	R/W	CPWM0 Shutdown control
COMPARE/CAPTURE/PWM UNIT 1 (CPWM1)			
CFG	0x41500000	R/W	CPWM1 configuration
STS	0x41500004	R/W	CPWM1 status
DATA	0x41500008	R/W	CPWM1 capture data and PWM threshold
Shutdown	0x4150000C	R/W	CPWM1 Shutdown control
COMPARE/CAPTURE/PWM UNIT 2 (CPWM2)			
CFG	0x41600000	R/W	CPWM2 configuration
STS	0x41600004	R/W	CPWM2 status
DATA	0x41600008	R/W	CPWM2 capture data and PWM threshold
Shutdown	0x4160000C	R/W	CPWM2 Shutdown control
COMPARE/CAPTURE/PWM UNIT 3 (CPWM3)			
CFG	0x41700000	R/W	CPWM3 configuration
STS	0x41700004	R/W	CPWM3 status
DATA	0x41700008	R/W	CPWM3 capture data and PWM threshold
Shutdown	0x4170000C	R/W	CPWM3 Shutdown control

Table 5. PERIPHERAL REGISTER TABLE SUMMARY (continued)

Peripheral Register	Address	Access	Description
MASTER/SLAVE SPI (SPI)			
DATA	0x41A00000	R/W	Transmit data (write) and receive data (read)
CFG	0x41A00004	R/W	Master/Slave SPI configuration
STS	0x41A00008	R/W	Master/Slave SPI status
UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER/TRANSMITTER 0 (USART0)			
DATA	0x41B00000	R/W	USART0 transmit data (write) and receive data (read)
CFG	0x41B00004	R/W	USART0 configuration
STS	0x41B00008	R/W	USART0 status
UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER/TRANSMITTER 1 (USART1)			
DATA	0x41C00000	R/W	USART1 transmit data (write) and receive data (read)
CFG	0x41C00004	R/W	USART1 configuration
STS	0x41C00008	R/W	USART1 status
I^2C			
CFG	0x41D00000	R/W	I^2C configuration
CTL	0x41D00004	WO	I^2C status and control
DATA	0x41D00008	R/W	I^2C transmit data (write) and receive data (read)
DATA_M	0x41D0000C	R/W	I^2C mirrored transmit/receive data
ADDR_START	0x41D00010	R/W	I^2C address to use for the transaction
STS	0x41D00014	RO	I^2C status register
INT_STS	0x41D00018	R/W	I^2C interrupt status register
MISCELLANEOUS			
REV_ID	0x41F00050	RO	Revision ID
LOCK	0x41F00060	R/W	Debug port lock

FLASH

Flash memory is directly addressable and may be used for system software or data storage.

The AXM0F343−64 MCU final word (address 0x0000_FFFC) of the 64 kB of directly addressable memory is used as a LOCK word. See the Lock Control description for more detail on its use.

The AXM0F343−256 MCU final word (address 0x0003_FFFC) of the 256 kB of directly addressable memory is used as a LOCK word. See the Lock Control description for more detail on its use.

Under normal operation, the flash block operates as if it were a ROM, providing single cycle read access via the AHB. Under software control, however, pages of the flash may be erased or programmed. These operations make read access to the flash memory unavailable until completed. This necessitates that the processor execute code in RAM during the program or erase time or the processor will simply stall waiting for the AHB Ready signal from the Flash to return high.

The registers used for programming and erasing the flash are mapped onto the APB.

AXM0F343−64 MCUflash is divided into 128 pages of 512 bytes each.

AXM0F343−256 MCU flash is divided into 256 pages of 1 kbytes each.

Each page may be independently erased, to allow for new content to be programmed. An erased address contains all 1s. Programming a word (or half word) can alter 1s to 0s, but cannot convert 0s to 1s. Only an erase can convert 0s to 1s.

To avoid inadvertent alteration of the flash content by errant code, each program/erase operation requires writing the keyword 0xA45B to the upper 16 bits of the FLASH CTL register. Reading the FLSHCNTL register will always return all 0s for the upper 16 bits.

To program data into the flash, the starting address is written to the FLASH ADDR register, and a word, or half−word, of data is written to the FLASH DATA register. If half word programming is selected bits 15−0 of the FLASH DATA register will be programmed into the selected half word of the address. The PGM bit is then written to a 1 in the FLASH CTL register along with the half word selection, and the keyword 0xA45B in bits 31:16. The PGM bit will clear itself. When programming is complete, the PGERDONE bit in the FLASH CTL register will be high. If a full word, or the upper half of a word is programmed, the FLASH ADDR register will also automatically increment by 4 to allow for programming the next address in the flash. If the program interrupt is enabled, an interrupt will be generated when the PGERDONE bit is set. Programming a word/half−word in the flash requires 20 microseconds. If the page protection bit for that page is set, the programming will fail. If the protection interrupt is enabled, an interrupt will be generated immediately.

To erase a page of data, any address within the target page may be written to the FLASH ADDR register. The PGERASE bit is then written to a 1 in the FLASH CTL register, along with the keyword 0xA45B in bits 31:16 of the same register. The PGERASE bit will clear itself. When erase is complete, the PGERDONE bit in the FLASH CTL register will be high. If the erase interrupt is enabled, an interrupt will be generated when the PGERDONE bit is set. Erasing a page of data requires 10 milliseconds. If the page protection bit for that page is set, the erase will fail. If the protection interrupt is enabled, an interrupt will be generated immediately.

It is also possible to erase the entire user memory in the flash, returning the part to factory condition. This is done by writing a 1 to the MASSERASE bit in the FLASH CTL register, along with the keyword 0xA45B in bits 31:16 of the FLASH CTL register. A mass erase requires 10 milliseconds.

It is possible for application code executing from FLASH to perform a page erase, word, and half−word write. The FLASH_EN bit in the PCLK_EN register must be set prior to writing the PGERASE or PGM bit. Also it is best to bad the FLASH → CTL write with NOP to make certain there are no FLASH controller/micro contentions upon exiting the FLASH ERASE/WRITE.

Flash Access Times:

Table 6.

Flash Access Type	Time
Flash Read	40 ns
Flash Write	20 μs
Flash Page Erase	10 ms
Flash Full Macro Erase	10 ms

AXM0F343 MCU Flash Comparison:

Table 7.

	AXM0F343−64 MCU	AXM0F343−256 MCU
FLASH Size	64 kB	256 kB
FLASH Size	128 x 512 B Pages	256 x 1 kB Pages
Lock Word Address	0x0000_FFFC	0x0003_FFFC

Flash Endurance:

Table 8.

Flash Endurance Type	Minimum (−40°C to +85°C)
Data Retention	10 years
Program Erase	100,000 cycles @ 25°C 10,000 cycles @ 85°C

Flash Register Summary (Base Address:0x40C0_0000):

Table 9.

Name	Offset	Access	Description
CTL	0x00	R/W	Flash program and erase control bits
ADDR	0x04	R/W	Address to be programmed or erased
DATA	0x08	R/W	Data to program into the flash
INT_EN	0x0C	R/W	Interrupt Enable Register
Reserved	0x10		
INT_STS	0x14	R/W	Interrupt Status Register
PROT0	0x18	R/W	Flash Page Protect Register 0 (Pages 0−31)
PROT1	0x1C	R/W	Flash Page Protect Register 1 (Pages 32−63)
PROT2	0x20	R/W	Flash Page Protect Register 2 (Pages 64−95)
PROT3	0x24	R/W	Flash Page Protect Register 3 (Pages 96−127)
DMA_EN	0x28	R/W	Flash DMA Signal Enable Register

In the AMBA system, three bits of PADDR, bits[4:2] are used to select a register. Bits [1:0] are not used because all registers are 32−bit word aligned. Typically in an AMBA system, PADDR[31:28] are used to decode AHB slaves (HSEL) and PADDR[27:24] are used to decode APB slaves (PSEL). In this example, since bits PADDR[23:7] are unused, this register area is aliased many times throughout the address range selected with PSEL. In other words, the address for the first register is:

32”bHHHH_PPPP_XXXX_XXXX_XXXX_XXXX_XXX0_00XX.

Flash Control Register Definition (Offset 0x00):

Table 10.

Function	Bits	Default	Type	Symbol	Description
CTL	[31:16]	0x0000	WO	KEYWD	Keyword to prevent inadvertent changes to the flash Must be written to 0xA45B to enable setting any of the bits 0−2. Also only one of the bits 0−2 may be a 1, or none of the bits will be written.
	[15:4]	–	–	–	
	[5]	0x1	RO	PGERDONE	Program/Erase done 0 = Program/Erase in process 1 = Program/Erase done/inactive
	[4]	0x0	R/W	UPHALF	Upper Half Word Select 0 = Program bits [15:0] 1 = Program bits [31:16]
	[3]	0x0	R/W	HALFWD	Half Word Programming 0 = Program 32 bit word 1 = Program 16 bit word
	[2]	0x0	R/W	MASSERASE	Set to a 1 to start a mass erase
	[1]	0x0	R/W	PGERASE	Set to a 1 to start a page erase
	[0]	0x0	R/W	PGM	Set to a 1 to start programming operation

Flash Address Register Definition (Offset 0x04):

Table 11.

Function	Bits	Default	Type	Symbol	Description
ADDR	[31:0]	0x0000_0000	R/W	ADDR	Address to be programmed/erased

Flash Data Register Definition (Offset 0x08):

Table 12.

Function	Bits	Default	Type	Symbol	Description
DATA	[31:0]	0x0000_0000	R/W	DATA	Data to be programmed (only bits [15:0] for half word programming)

Flash INT Enable Register Definition (Offset 0x0C):

Table 13.

Function	Bits	Default	Type	Symbol	Description
INT_EN	[31:3]	–	–	–	Reserved
	[2]	0x0	R/W	PROT	Enable interrupt for protected segment violation. 1 = enabled, 0 = disabled
	[1]	0x0	R/W	ERASE	Enable erase interrupt. 1 = enabled, 0 = disabled
	[0]	0x0	R/W	PGM	Enable programming interrupt. 1 = enabled, 0 = disabled

Flash INT Status Register Definition (Offset 0x14):

Table 14.

Function	Bits	Default	Type	Symbol	Description
INT_STS	[31:2]	–	–	–	Reserved
	[2]	0x0	R/W	PROT	Protection interrupt pending if set. Write a 1 to clear
	[1]	0x0	R/W	ERASE	Erase interrupt pending if set. Write a 1 to clear
	[0]	0x0	R/W	PGM	Program interrupt pending if set. Write a 1 to clear

Flash Page Protect 0/1/2/3 Register Definition (Offset 0x18, 0x1C, 0x20, 0x24):

The Flash Page Protect 0 Register contains page protect bits for Flash pages 0−31. The Flash Page Protect 1 Register contains page protect bits for Flash pages 32−63. The Flash Page Protect 2 Registers contains page protect bits for Flash pages 64−95. The Flash Page Protect 3 Registers contains page protect bits for Flash pages 96−127. When the Page Protect bit is set for a page, the page becomes Read Only and cannot be written or erased. For AXM0F343−256 MCU only pages 0−127 are protected, pages 128−255 do not have protection capability.

Table 15.

Function	Bits	Default	Type	Symbol	Description
PROT0	[31:0]	0x00000000	R/W	PROT0	PAGE_PROT0[31:0] = Protect page [31:0]
PROT1	[31:0]	0x00000000	R/W	PROT1	PAGE_PROT1[31:0] = Protect page [63:32]
PROT2	[31:0]	0x00000000	R/W	PROT2	PAGE_PROT2[31:0] = Protect page [95:64]
PROT3	[31:0]	0x00000000	R/W	PROT3	PAGE_PROT3[31:0] = Protect page [127:96]

Flash DMA Signal Enable Register Definition (Offset 0x28):

Table 16.

Function	Bits	Default	Type	Symbol	Description
DMA_EN	[31:1]	–	–	–	Reserved
	[0]	0x0	R/W	PGM_SIGNAL	Enable DMA signal on programming complete. 1= enabled, 0= disabled

POWER MANAGEMENT

Power Modes

There are 4 operational modes with varying power consumption. These modes can be used by application software to optimize consumption of both dynamic and static power. Reduced functionality and retention occurs with each consecutive level of lower power mode.

The part will default to the highest power mode (run mode) on startup. The lower power modes are all entered through software (wfi/wfe) commands. The software needs to configure which mode will be entered upon these commands. See Power Management Unit.

The power modes are for functional operation, thus the part will not go into the low−power modes when DBG_EN is enabled.

Run Mode

In Run mode all digital systems are powered and running including external and/or internal oscillators. The processor is executing code. Various options exist to reduce power within this mode. Individual peripheral clocks can be gated based on configuration. You can also reduce the clock frequency of the system clock for power reductions. Additionally the software can power down one of the SRAM blocks if not needed by the application.

Sleep Mode

Sleep mode is the same as Run mode except the Arm Cortex−M0+ processor is in Sleep Mode with the processor clock gated. Since the processor clock is gated, the processor is not executing code. The flash is also powered down in this state. Some peripherals will maintain their clock in order to generate interrupts based on configuration. When an interrupt is detected, the processor wakes−up and enters run mode. Code execution starts from the last known location.

The HSOSC can be enabled or disabled during this mode. The HSOSC Draws significant current, thus for use cases where the 10.24 kHz internal LPOSC is sufficient for wake−up timing it is recommended to disable the HSOSC and enable it upon exit from Sleep mode.

Hibernate Mode

Hibernate mode powers off the CPU, flash, and most peripherals. Each SRAM block may configured to either power−off or retain its contents. The wake−up timer, GPIO, and external interrupt detection remain operational in Hibernate mode with a 1 V regulated supply to provide the wake−up condition. The Low−Power RC Oscillator is the only internal clock source available, if enabled. Waking up from hibernate mode re−initializes the processor and code execution starts from the reset vector, and switches to operating on the High Speed RC oscillator in 32 MHz mode, to be similar to power on. The PMU Status register indicates if the reset was due to a POR (cold start), hibernate, or shutdown. Wake−up from Hibernate requires approximately 320 μs.

Shutdown Mode

In Shutdown mode all circuits are powered−off except for the always on scratch register and minimal VDDIO domain logic used to detect a toggle on the **PB3** pin for wake−up. Waking up from Shutdown mode re−initializes the processor and code execution starts from the reset vector. The scratch register can be used to differentiate from a POR reset (cold start) and a shutdown mode wakeup reset (warm start).

Power Mode Table

Table 17. POWER MODE TABLE

Power Mode	Clocks	Powered Domains	Mode Exit/ Wakeup	Enabled Peripherals	Retention	Description
Run	HSOSC/HSXOSC LPOSC/LPXOSC FCLK/HCLK/PCLK DCLK (Note 1)	3.3 V VDDIO 3.3 V Wakeup 1.8 V DVDD 1.8 V AVDD RVDD2/6	N/A	All enabled	All registers and SRAM	Core and enabled clocks/peripherals active
Sleep	HSOSC/HSXOSC LPOSC/LPXOSC FCLK PCLK Limited	3.3 V VDDIO 3.3 V Wakeup 1.8 V DVDD 1.8 V AVDD RVDD2/6	Any enabled interrupt	All sleep enabled	All registers and SRAM	Core not executing. Most peripheral can be enabled. Switch to slow clock for lower power.

Table 17. POWER MODE TABLE (continued)

Power Mode	Clocks	Powered Domains	Mode Exit/ Wakeup	Enabled Peripherals	Retention	Description
Hibernate	LPOSC/LPXOSC PCLK – Wakeup Timer Only	3.3 V VDDIO 3.3 V Wakeup 1 V DVDD RVDD2/6 (Note 2)	Wakeup timer Any GPIO pin DGB_EN Ext. Reset	Wakeup timer GPIO DBG_EN	AO Scratch GPIO state SRAM (Note 3)	Core powered down. Wakeup and GPIO_WAKE powered. Low voltage SRAM retention. Resets core upon mode exit.
Shutdown	None	3 V VDDIO 3 V Wakeup	PB3 DBG_EN Ext. Reset	PB3 DBG_EN	AO Scratch GPIO state	Lowest power. GPIO wakeup only. Resets core upon mode exit.

1. If DBG_EN = 1
2. If ADC is enabled
3. Optional independent retention of SRAM banks

Power Management Unit (PMU)

The power management unit (PMU) with four operating modes: run, sleep, hibernate, and shutdown. Use of a debugger prevents the part from entering any low power mode. Driving the DBG_EN pin will cause the part to wake from hibernate or shutdown mode. All low power modes are entered with the software wfi/wfe command.

Table 18.

Function	Bits	Default	Type	Symbol	Description
PMU POWER MODE REGISTER: 0x40400000 The PMU Power Mode Register is used to configure low−power modes. Entry into these modes may be a combination of setting the register configuration along with specific processor code.					
POWER_MODE	[2]	0x0	R/W	PAD_LATCH	Latch I/O state. This bit will automatically be set high upon exit from hibernate or shutdown mode. 1: I/O state is latched and will ignore GPIO/XBAR settings 0: I/Os obey GPIO/XBAR settings
	[1:0]	0x0	R/W	POWER_MODE	Power Mode Configuration: 1x: Enter Shut−Down Mode 01: Enter Hibernate Mode 00: Run Mode Configuration (Sleep entered and exited by processor)
PMU POWER−DOWN REGISTER: 0x40400004 The PMU Power−down Register is used to power down individual SRAM blocks.					
POWER_DOWN	[1]	0x0	R/W	PD_SRAM1	Power Down SRAM1 AXM0F343−64 MCU = 6K SRAM bank AXM0F343−256 MCU = 24K SRAM bank
	[0]	0x0	R/W	PD_SRAM0	Power Down AXM0F343−64 MCU = 2K SRAM bank AXM0F343−256 MCU = 8K SRAM bank
PMU STATUS REGISTER: 0x40400008 The PMU Status Register contains reset information.					
STS	[3]	0x0	RO	DBG_STAT	State of DBG_EN pin (available as a convenience)
	[2]	0x0	RO	SHUTWU	Wakeup Reset; Reading 1 indicates that the last reset was caused by a wakeup from shutdown.
	[1]	0x0	RO	HIBWU	Wakeup Reset; Reading 1 indicates that the last reset was caused by a wakeup from hibernate.
	[0]	0x0	RO	WDOG_RES	Watchdog Reset; Reading 1 indicates that the last reset was caused by a watchdog reset

RESET AND BROWNOUT

Reset Sources

The chip has various sources of reset including:

- *Internal Power−On Reset (POR)* – The POR reset asserts when the supply is below threshold levels for proper operation. It also asserts during shutdown mode. By writing a non−default value to the always on scratch register software can differentiate between these events. The POR resets the entire chip including core, debug port, peripherals, wakeup timer, and watchdog.
- *External Pin Reset* – The external reset is under user control with the external RESETn pin. External pin reset resets the entire chip including core, debug port, peripherals, wakeup timer, and watchdog.
- *Software Issued Reset* – The software reset can be called by writing to a given register in the Cortex address space. It is typically called on exit from a processor exception. Software reset resets the entire chip including core, debug port, peripherals, wakeup timer, and watchdog.
- *Watchdog Timer Reset* – The watchdog timer reset is caused by the watchdog timeout and is used to prevent errant software from locking up the device. The watchdog reset resets the entire chip including core, debug port, peripherals, wakeup timer, and watchdog. The watchdog timer is disabled upon power up and must be enabled by software. The watchdog is paused when the debugger halts the processor.
- *Sleep Exit* – A partial reset of the flash logic is issued on sleep exit to make sure it is a proper state to start reading the flash.
- *Hibernate Mode Exit* – Hibernate mode exit generates a reset for the core and most peripherals. It does not reset the power management logic (PMU, wakeup timer, etc.). This allows software to differentiate a “cold start” reset from a “warm start” reset on hibernate exit.

Upon reset de−assertion, an internal state machine holds the core in reset until the flash has been checked for trim programming and the trim values have been loaded. Once this has occurred, the processor can start to read from the flash.

Brownout

The internal supply voltage is monitored and if it gets too low, the brownout indicator is asserted (before POR) to warn that power is likely going away. The brownout signal is connected to the processor non−maskable interrupt (NMI). The brownout indicator can be used to record or modify state before losing power. Recovery from a brownout event may be possible if a POR is not issued but it is safest to issue a software reset as part of the brownout service routine.

CLOCKS (CMU)

Clock Sources

There are 5 main clock sources available depending on power, accuracy, and application requirements. Internal clock sources include a high speed oscillator with 32 MHz and 40 MHz modes and a low power oscillator with 10.24 kHz and 640 Hz modes. External clock sources include a high speed (20 MHz) crystal oscillator and a low power (32.768 kHz) crystal oscillator. External input clock from GPIO through the XBAR is available as RSYSCLK in the diagram below, with a max speed of 20 MHz.

Clock Diagram

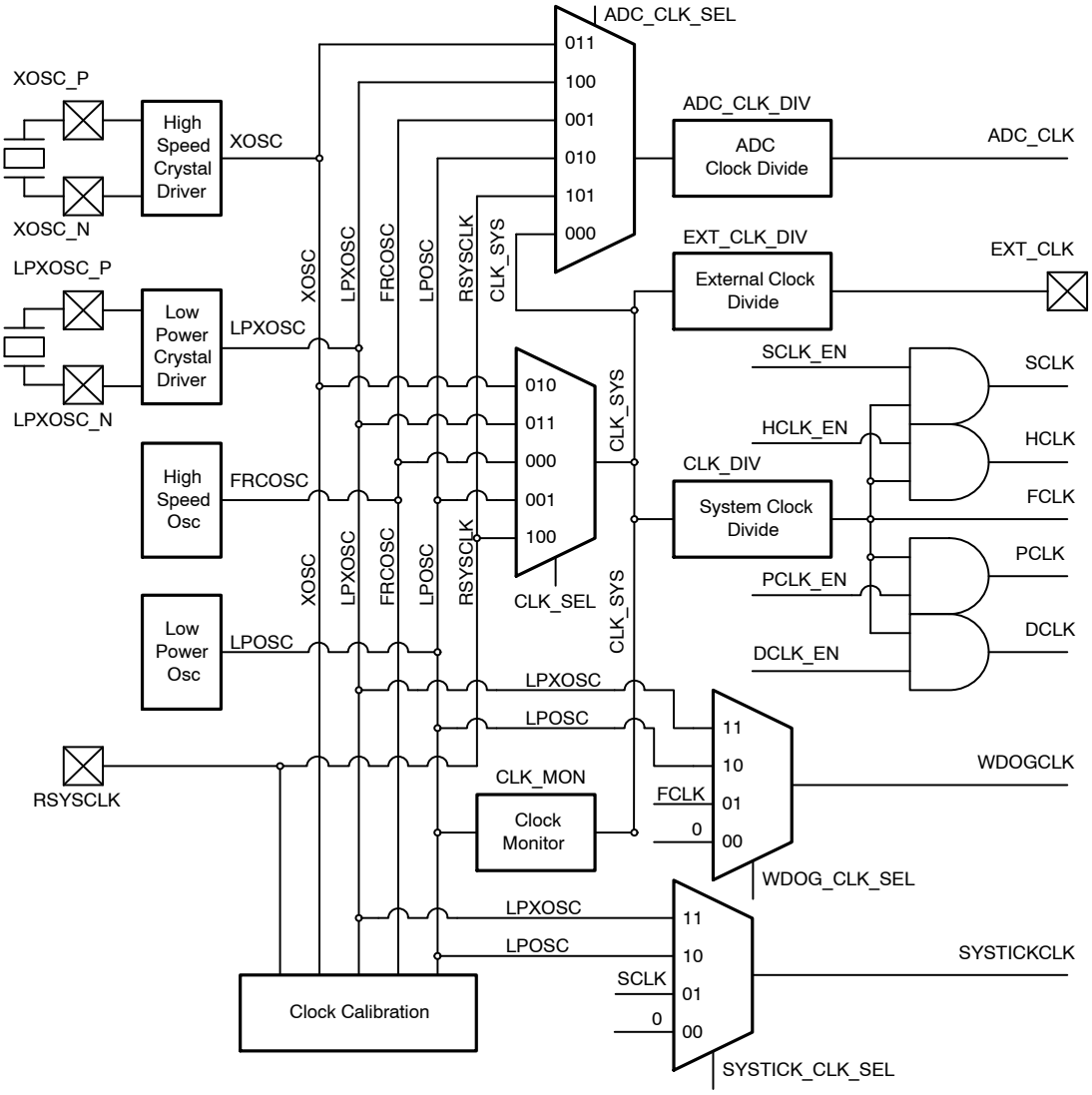


Figure 4. Clock Diagram

Clock Operation and Calibration

Each individual clock source can be powered down independently based on application and power requirements. Care should be taken to ensure that not all clock sources are powered down at once as this can result in a non−operational part. The clock monitor can be enabled to detect the loss of system clock but this is not default behavior.

The internal clocks are calibrated during manufacturing this calibration trim is written into the flash. These values are automatically loaded into calibration shadow registers on startup. The internal clocks can be optionally re−calibrated by the user if necessary.

Clock Configuration and Status Register Table

Table 19. CLOCK CONFIGURATION AND STATUS REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
CLOCK CONFIGURATION REGISTER: 0x40300000 The Clock Configuration Register is used to select the system clock frequency.					
CFG	[22]	0x0	R/W	CONS_READ_OV	Consecutive read override, in case of external RC clock, if the user wants to override consecutive read delay 0 = no override 1 = override consecutive read latency
	[21]	0x1	R/W	FLASH_WS_OV	Flash wait state override 0 = no override 1 = override flash wait state when using 20 MHz or lower external crystal clock
	[20]	0x1	R/W	CLK_LOSS_INT_EN	Clock Loss Interrupt Enable
	[19]	0x0	R/W	LPOSC_CAL_INT_EN	Low Power Oscillator Calibration Interrupt Enable
	[18]	0x0	R/W	HSOSC_CAL_INT_EN	High Speed RC Oscillator Calibration Interrupt Enable
	[17:16]	0x0	R/W	SYSTICK_CLK_SEL	SYSTICK Clock Select: 0x0: SYSTICKCLK clock disabled 0x1 = SYSTICKCLK = SYSCLK 0x2 = SYSTICKCLK = Low Power RC Oscillator 0x3: SYSTICKCLK = Low Power Crystal Oscillator
	[15:14]	0x0	R/W	WDOG_CLK_SEL	Watchdog Clock Select: 0x0: WDOGCLK disabled 0x1 = WDOGCLK = SYSCLK 0x2 = WDOGCLK = Low Power RC Oscillator 0x3: WDOGCLK = Low Power Crystal Oscillator
	[13]	0x1	R/W	LPXOSC_PD	Low Power Crystal Oscillator Power−down: 0x0 = Low Power Crystal Oscillator Enabled 0x1 = Low Power Crystal Oscillator Powered−down
	[12]	0x1	R/W	HSXOSC_PD	High Speed Crystal Oscillator Power−down: 0x0 = High Speed Crystal Oscillator Enabled 0x1 = High Speed Crystal Oscillator Powered−down
	[11]	0x1	R/W	LPOSC_PD	Low Power RC Oscillator Power−down: 0x0 = Low Power RC Oscillator Enabled 0x1 = Low Power RC Oscillator Powered−down
	[10]	0x0	R/W	HSOSC_PD	High Speed RC Oscillator Power−down: 0x0 = High Speed RC Oscillator Enabled 0x1 = High Speed RC Oscillator Powered−down
	[9:8]	0x0	R/W	RESERVED	RESERVED
	[7:6]	0x0	R/W	SYSCLK_MON_PER	Set Clock Monitor Period: 0x0 = Off 0x1 = 4 LPOSC Periods 0x2 = 16 LPOSC Periods 0x3 = 64 LPOSC Periods
	[5:3]	0x0	R/W	SYSCLK_DIV	Selects the System Clock Divide. 0x0 = HCLK = SYSCLK 0x1 = HCLK = SYSCLK ÷2 0x2 = HCLK = SYSCLK ÷4 0x3 = HCLK = SYSCLK ÷8 0x4 = HCLK = SYSCLK ÷16 0x5 = HCLK = SYSCLK ÷32 0x6 = HCLK = SYSCLK ÷64 0x7 = HCLK = SYSCLK ÷128
	[2:0]	0x0	R/W	SYSCLK_SEL	Requests the System Clock (SYSCLK) Source: 0x0 = High−Speed RC Oscillator Clock (HSOSC) 0x1 = Low Power 10kHz RC Oscillator (LPOSC) 0x2 = External High−Speed Crystal Oscillator (HSXOSC) 0x3 = External Low Power Crystal Oscillator (LPXOSC) 0x4 = External Clock on CLK_IN (EXTCLK) 0x5 − 0x7: RESERVED

Table 19. CLOCK CONFIGURATION AND STATUS REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
CLOCK STATUS REGISTER: 0x40300004 The Clock Status Register is a read−only register used to indicate clock status.					
STS	[3]	0x0	RO	CLK_LOSS	Clock Loss Detected and Switched Back to LPOSC
	[2:0]	0x0	RO	CLK_SEL	Active System Clock Source: 0x0 = High−Speed RC Oscillator Clock (HSOSC) 0x1 = Low Power 10 kHz RC Oscillator (LPOSC) 0x2 = External High−Speed Crystal Oscillator (HSXOSC) 0x3 = External Low Power Crystal Oscillator (LPXOSC) 0x4 = External Clock on CLK_IN (EXTCLK) 0x5 − 0x7: RESERVED **Note:** This register reflects the currently active clock settings. Since clock switching takes time, however, it will not immediately take on new values.
ADC CLOCK CONFIGURATION REGISTER: 0x40300008 The ADC Clock Configuration Register is used to select the ADC Clock source and clock frequency divide down value.					
ADCCLK_CFG	[6:4]	0x7	R/W	SEL	Requests the ADC Clock Source: 0x0: HSOSC 0x1: LPOSC 0x2: HSXOSC 0x3: LPXOSC 0x4: EXTCLK 0x5: PRESYSCLK 0x6 RESERVED 0x7: ADC clock disabled
	[3]	0x0	RO	Reserved	
	[2:0]	0x0	R/W	DIV	ADC Clock Divide Register. 0x0: ADCCLK = ÷1 0x1: ADCCLK = ÷2 0x2: ADCCLK = ÷4 0x3: ADCCLK = ÷8 0x4: ADCCLK = ÷16 0x5: ADCCLK = ÷32 0x6: ADCCLK = ÷64 0x7: ADCCLK = ÷128
EXTERNAL CLOCK CONFIGURATION REGISTER: 0x4030000C The External Clock Configuration Register is used to select the clock source and frequency divide for the external clock output on EXTCLK_OUT.					
EXTCLK_CFG	[10:8]	0x7	R/W	SEL	External clock source select: 0x0: HSOSC 0x1: LPOSC 0x2: HSXOSC 0x3: LPXOSC 0x4: EXTCLK_IN 0x5: PRESYSCLK 0x6: RESERVED 0x7: External clock disabled
	[7:0]	0x0	R/W	DIV	External Clock Divide Register. The external clock frequency will be the source clock divided by DIV+1

Table 19. CLOCK CONFIGURATION AND STATUS REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
PCLK ENABLE FORCE REGISTER: 0x40300010 The PCLK Enable Force Register is used to force individual PCLK sources to an enabled or ungated state. Normally these clocks are automatically enabled by the configuration or function of the peripheral. This register should not be used in normal conditions. Any write to the DMA will automatically set the PCLK_DMA_EN register. To save power after the DMA is disabled this bit should be cleared.					
PCLK_EN	[31]	0x0	R/W	TEST_EN	Set associated bit to 1 to force on PCLK to each peripheral.
	[30]	0x0	R/W	Reserved	
	[29]	0x0	R/W	I2C_EN	
	[28]	0x0	R/W	USART1_EN	
	[27]	0x0	R/W	USART0_EN	
	[26]	0x0	R/W	SPI_EN	
	[25]	0x0	R/W	Reserved	
	[24]	0x0	R/W	Reserved	
	[23]	0x0	R/W	CPWM3_EN	
	[22]	0x0	R/W	CPWM2_EN	
	[21]	0x0	R/W	CPWM1_EN	
	[20]	0x0	R/W	CPWM0_EN	
	[19]	0x0	R/W	TIM2_EN	
	[18]	0x0	R/W	TIM1_EN	
	[17]	0x0	R/W	TIM0_EN	
	[16]	0x0	R/W	TICK_EN	
	[15]	0x0	R/W	WDOG_EN	
	[14]	0x0	R/W	Reserved	
	[13]	0x0	R/W	CRC_EN	
	[12]	0x0	R/W	FLASH_EN	
	[11]	0x0	R/W	Reserved	
	[10]	0x0	R/W	TRNG_EN	
	[9]	0x0	R/W	AES_EN	
	[8]	0x0	R/W	DMA_EN	
	[7]	0x0	R/W	ADC_EN	
	[6]	0x0	R/W	CMP_EN	
	[5]	0x0	R/W	XBAR_EN	
	[4]	0x0	R/W	PMU_EN	
	[3]	0x0	R/W	SYSCLK_EN	
	[2]	0x0	R/W	Reserved	
	[1]	0x0	R/W	WUT_EN	
	[0]	0x0	R/W	GPIO_EN	
OSCILLATOR READY REGISTER: 0x40300044 The Oscillator Ready Register indicates which oscillator is turned on and is operating. The crystal oscillators take some time between switching on and achieving stable oscillation.					
OSC_READY_STS	[3]	0x0	RO	LPXOSC	1 indicates the Low Power Crystal oscillator is running and stable
	[2]	0x0	RO	HSXOSC	1 indicates the Crystal oscillator is running and stable
	[1]	0x0	RO	LPOSC	1 indicates the Low Power oscillator is running and stable
	[0]	0x0	RO	HSOSC	1 indicates the Fast RC oscillator is running and stable

Table 19. CLOCK CONFIGURATION AND STATUS REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
OSCILLATOR INTERRUPT REGISTER: 0x40300048 The Oscillator Interrupt Register reports the events that generate a clock management interrupt. Clear by writing a 1 to the corresponding bit(s).					
OSC_INT_STS	[4]	0x0	R/W	LPOSC_CAL	Low Power Oscillator Calibration Updated Interrupt
	[3]	0x0	R/W	HSOSC_CAL	Fast RC Oscillator Calibration Updated Interrupt
	[2]	0x0	R/W	LPOSC_CAL_MISSED	Low Power Oscillator Calibration Missed
	[1]	0x0	R/W	HSOSC_CAL_MISSED	Fast RC Oscillator Calibration Missed
	[0]	0x0	R/W	CLK_LOSS	Clock Loss Status (same Clock Status Register)

Low Power Oscillator Configuration and Calibration Register Table

Table 20. LOW POWER OSCILLATOR CONFIGURATION AND CALIBRATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
LOW POWER OSCILLATOR CONFIGURATION REGISTER: 0x40300014 The Low Power Oscillator Configuration Register is used to setup the configuration and calibration of the Low Power Oscillator. Note that the selected reference oscillator is not automatically turned on. The purpose of this feature is to allow “opportunistic” calibration, i.e. the calibration logic is always turned on, but is inactive until the reference oscillator is needed for another purpose. The reference oscillator can, however, be forced to run using the OSCFORCERUN register. Set CAL_SEL = 0x1 after calibration is complete to stop the calibration engine and lock in the Cal frequency trim value into the Tune Register.					
LPOSC_CFG	[6]	0x0	R/W	FREQ_SEL	Select the Frequency of the Low Power Oscillator. 0 = 640 Hz, 1 = 10.24 kHz
	[5:3]	0x0	R/W	CAL_DIV	Low Power Oscillator pre−scaler: 000: Reserved 001: Reserved 010: ÷2 011: ÷4 100: ÷8 101: ÷16 110: ÷32 111: ÷64
	[2:0]	0x0	R/W	CAL_SEL	Low Power Oscillator Calibration Source 000: HSOSC 001: Off (no Calibration) 010: HSXOSC 011: invalid 100: EXTCLK 101−111: invalid

LOW POWER OSCILLATOR CALIBRATION FILTER REGISTER: 0x40300018

The Low Power Oscillator Calibration Filter Register is used to setup the calibration filter for the Low Power Oscillator. The maximum value of k_{FILT}, that results in quickest calibration (single cycle), but no jitter suppression, is:

$$k_{FILT} = \left\lceil \frac{PRESCALER \cdot 2^{20}}{k_{LPOSC} \cdot \tau_{base} \cdot f_{REF}} \right\rceil = \left\lceil \frac{512\ kHz \cdot PRESCALER \cdot 2^{20}}{f_{REF}} \right\rceil \quad \text{(eq. 1)}$$

Example: FREF = 20 MHz, CAL_DIV = b010

$$k_{FILT} = \left\lceil \frac{512\ kHz \cdot \left(\frac{1}{2}\right) \cdot 2^{20}}{20\ MHz} \right\rceil = 0x346E$$

For reference when CAL_DIV = 010 ($f_{REF} \div 2$) then PRESCALER = (1/2)

Smaller values of k_{FILT} result in longer calibration, but increased jitter suppression. If *kFILT* is greater than 12 bits use 0xFFF for the LPOSC_FILT value. Since the calibration is only required to run once at boot, it is recommended a smaller *kFILT* be used to improve the calibration.

LPOSC_FILT	[15:0]	0x20C4	R/W	LPOSC_FILT	k_{FILT} (Low Power Oscillator Calibration Filter Constant)

Table 20. LOW POWER OSCILLATOR CONFIGURATION AND CALIBRATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
LOW POWER OSCILLATOR CALIBRATION REFERENCE FREQUENCY DIVIDER REGISTER: 0x4030001C The Low Power Oscillator Calibration Reference Frequency Divider Register is used to select the reference frequency divide.					
LPOSC_REF_DIV	[15:0]	0x61A8	R/W	LPOSC_REF_DIV	LP Oscillator Reference Frequency Divider; set to: $\frac{f_{REF} \cdot PRESCALER}{10\ kHz \mid 640\ Hz}$ $\frac{20\ MHz \cdot \left(\frac{1}{2}\right)}{10.24\ kHz} = 0x3D1$ (eq. 2)
LOW POWER OSCILLATOR CALIBRATION FREQUENCY TUNE REGISTER 0x40300020 The Low Power Oscillator Calibration Frequency Tune Register is used to select the frequency tune value.					
LPOSC_TRIM	[11:0]	0x000	R/W	LPOSC_TRIM	LP Oscillator Frequency Tune Value; in 1/32%.
LOW POWER OSCILLATOR CALIBRATION MEASURED PERIOD REGISTER: 0x4030003C The Low Power Oscillator Calibration Measured Period Register is used to read the last measured Lower Power Oscillator period. This is the number of reference clock cycles in a single LPOSC oscillator period.					
LPOSC_PER	[15:0]	0x0000	RO	LPOSC_PER	Last measured Low Power Oscillator Period

High Speed Oscillator Configuration and Calibration Register Table

Table 21. HIGH SPEED OSCILLATOR CONFIGURATION AND CALIBRATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
FAST RC OSCILLATOR CALIBRATION MEASURED FREQUENCY REGISTER: 0X40300024 The Fast RC Oscillator Calibration Measured Frequency Register is used to read the last measured HSOSC oscillator period. This is the number of oscillator clocks in one period of the pre−scaled reference clock.					
HSOSC_PER	[15:0]	0x0000	RO	HSOSC_PER	Last measured High Speed Oscillator Period

Table 21. HIGH SPEED OSCILLATOR CONFIGURATION AND CALIBRATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
HIGH SPEED OSCILLATOR CONFIGURATION REGISTER: 0x40300028 The Fast RC Oscillator Calibration Configuration Register is used to setup the configuration and calibration of the Fast RC Oscillator. Choose a reference clock source, and choose the reference clock pre−scaler such that the resulting reference frequency is above 500 Hz. For best frequency measurement precision, the resulting reference frequency should be between 500 Hz and 1 kHz, but faster reference frequencies are permissible for faster acquisition. Note that the selected reference oscillator is not automatically turned on. The purpose of this feature is to allow “opportunistic” calibration, i.e. the calibration logic is always turned on, but is inactive until the reference oscillator is needed for another purpose. The reference oscillator can, however, be forced to run using the OSCFORCERUN register.					
HSOSC_CFG	[8]	0x0	R/W	PER_GAIN	Period Gain; if 1, the measured period is multiplied by 16
	[7]	0x0	R/W	FREQ_SEL	Select the Frequency of the Fast Oscillator. 1 = MAX (40 MHz), 0 = MIN (32 MHz) Default is 32 MHz.
	[6:3]	0x4	R/W	DIV	Fast RC Oscillator pre−scaler: 0000: Reserved 0001: Reserved 0010: ÷2 0011: ÷4 0100: ÷8 0101: ÷16 0110: ÷32 0111: ÷64 1000: ÷128 1001: ÷256 1010: ÷512 1011: ÷1024 1100: ÷2048 1101: ÷4096 1110: ÷8192 1111: ÷16384
	[2:0]		R/W	CAL_SEL	Fast RC Oscillator Calibration Source 000: Off (no Calibration) 001: LPOSC 010: XOSC 011: LPXOSC 100: EXTCLK 101: invalid 110: invalid 111: invalid

FAST RC OSCILLATOR CALIBRATION FILTER REGISTER: 0x4030002C

The Fast RC Oscillator Calibration Filter Register is used to setup the calibration filter for the Fast RC Oscillator. The maximum value of k_{FILT}, that results in quickest calibration (single cycle), but no jitter suppression, is:

$$\text{For PERGAIN} = 0,\ \text{it is}: \ k_{FILT} = \left\lceil \frac{2^{20}}{f_{base} \cdot \tau_{REF} \cdot k_{HSOSC}} \right\rceil = \left\lceil \frac{f_{REF} \cdot 2^{20}}{15\ \text{kHZ} \cdot \text{PRESCALER}} \right\rceil \quad \text{(eq. 3)}$$

$$\text{For PERGAIN} = 1,\ \text{it is}: \ k_{FILT} = \left\lceil \frac{2^{16}}{f_{base} \cdot \tau_{REF} \cdot k_{HSOSC}} \right\rceil = \left\lceil \frac{f_{REF} \cdot 2^{16}}{15\ \text{kHZ} \cdot \text{PRESCALER}} \right\rceil \quad \text{(eq. 4)}$$

Smaller values of k_{FILT} result in longer calibration, but increased jitter suppression. If *kFILT* is greater than 12 bits use 0xFFF for the HSOSC_FILT value. Since the calibration is only required to run once at boot, it is recommended a smaller *kFILT* be used to improve the calibration. Recommendation is to use 51.2 kHz instead of 15 kHz in the calculation above.

HSOSC_FILT	[15:0]	0x20C4	R/W	HSOSC_FILT	k_{FILT} (Fast RC Oscillator Calibration Filter Constant)

FAST RC OSCILLATOR CALIBRATION REFERENCE FREQUENCY DIVIDER REGISTER: 0x40300030

The Fast RC Oscillator Calibration Reference Frequency Divider Register is used to select the calibration reference frequency divide.

HSOSC_REF_DIV	[15:0]	0x61A8	R/W	HSOSC_REF_DIV	High Speed RC Oscillator Reference Frequency Divider; set to $\frac{32\ \text{MHz} \mid 40\ \text{MHz} \cdot \text{PRESCALER}}{f_{REF}}$ (eq. 5)

Table 21. HIGH SPEED OSCILLATOR CONFIGURATION AND CALIBRATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
FAST RC OSCILLATOR 40 MHZ CALIBRATION FREQUENCY TUNE REGISTER: 0x40300034 The Fast RC Oscillator 40 MHz Calibration Frequency Tune Register is used to select the frequency trim value for 40 MHz operation.					
HSOSC_MAX_TRIM	[11:0]	0x000	R/W	HSOSC_MAX_TRIM	Fast RC Oscillator 40 MHz Frequency Tune Value; in 1/32%.
FAST RC OSCILLATOR 32 MHZ CALIBRATION FREQUENCY TUNE REGISTER: 0x40300038 The Fast RC Oscillator 32 MHz Calibration Frequency Tune Register is used to select the frequency trim value for 32 MHz operation.					
HSOSC_MIN_TRIM	[11:0]	0x000	R/W	HSOSC_MIN_TRIM	Fast RC Oscillator 32 MHz Frequency Tune Value; in 1/32%.

High Speed and Low Power Crystal Configuration Register Table

Table 22. HIGH SPEED AND LOW POWER CRYSTAL CONFIGURATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
LOW POWER CRYSTAL TRANSCONDUCTANCE CONFIGURATION REGISTER: 0x4030004C This register configures the low power crystal oscillator.					
LPXOSC_GM_CFG	[4:0]	0x1	R/W	LPXOSC_GM_CFG	Oscillator Transconductance Configuration : 00110 = 3.5 μS : 01000 = 4.6 μS : 01100 = 6.9 μS : 10000 = 9.1 μS
HIGH SPEED CRYSTAL TRANSCONDUCTANCE CONFIGURATION REGISTER: 0x40300054 This register allows the transconductance of the crystal oscillator to be configured. Normally, setting this register is not necessary, as a servo loop controls the transconductance to achieve low amplitude oscillation. This ensures the minimum possible current consumption.					
HSXOSC_GM_CFG	[3:0]	0x4	R/W	HSXOSC_GM_CFG	Gm (Gain) of the Crystal Oscillator 0000 = 0 μS 0001 = 25 μS 0010 = 50 μS 0011 = 75 μS : 1110 = 350 μS 1111 = 1000 μS
HIGH SPEED CRYSTAL AMPLITUDE CONFIGURATION REGISTER: 0x40300058 This register controls the transconductance servo loop. This register should be left at the default value.					
HSXOSC_AMPL_CFG	[6]	0x0	R/W	REG_EN	Crystal Oscillator Amplitude Regulator Enable
	[5]	0x0	R/W	AMPL_DET_EN	Amplitude detector enable
	[4]	0x0	R/W	READY_DET_EN	Ready detector enable
	[2:0]	0x4	R/W	AMPL	Crystal Oscillator Amplitude 000 : 180 mV 001 : 195 mV 010 : 230 mV : 111 : 460 mV

Table 22. HIGH SPEED AND LOW POWER CRYSTAL CONFIGURATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
HIGH SPEED CRYSTAL READY REGISTER: 0x4030005C This register controls the generation of the crystal oscillator ready signal. It should not normally be changed from the default.					
HSXOSC_READY	[7]	0x0	RO	SIG_DET_STS	Crystal Oscillator SIG DET signal
	[6]	0x0	RO	READY_STS	Crystal Oscillator READY signal
	[3]	0x0	R/W	OUT_AMP_DIS	If set, the crystal oscillator output amplifier is disabled
	[2:0]	0x1	R/W	MOD	Crystal Oscillator Ready Mode 000: Always Ready 001: Ready when HSXOSC_READY = 1 010: Ready when HSXOSC_SIG_DET = 1 011: Ready when either HSXOSC_READY = 1 or HSXOSC_SIG_DET = 1 (or both) 100: Never Ready 101: Invalid (never ready) 110: Invalid (never ready) 111: Ready when both HSXOSC_READY = 1 and HSXOSC_SIG_DET = 1

SYSTEM CONFIGURATION

The system configuration logic shares address space with the clock configuration. It includes a scratch register that is always−on during all power modes. It also includes controls for the programmable IRQ latency.

System Configuration Register Table

Table 23. SYSTEM CONFIGURATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
ALWAYS−ON SCRATCH REGISTER: 0x40300060 The main purpose of the scratch registers is to provide 32bit of state storage during low power modes. This is helpful in implementing a software flag to differentiate between startup modes (cold start/warm start).					
AO_SCRATCH	[31:0]	0x00000000	R/W	AO_SCRATCH	Always−On Scratch register. Retains value in all power−modes.
INT LATENCY REGISTER: 0x40300064 The Cortex−M0+ processor supports zero jitter interrupt latency for zero wait−state memory. The IRQ latency register value specifies the minimum number of cycles between an interrupt that becomes pended in the NVIC, and the vector fetch for that interrupt being issued on the AHB−Lite interface. If this bus is set to 0, interrupts are taken as quickly as possible. For non−zero values, the processor ensures that a minimum of INT_LAT+1 SCLK cycles exist between an interrupt becoming pended in the NVIC and the vector fetch for the interrupt being performed. For zero jitter predictable interrupts set this bus to at least a decimal value of 9. This causes an interrupt latency of 15 cycles from the assertion of the NVIC IRQ pin to execution of the exception handler.					
INT_LAT	[7:0]	0x00	R/W	INT_LAT	Interrupt Latency input to the M0+ core

GENERAL PURPOSE INPUT/OUTPUT (GPIO)

The AXM0F343 SoC owns a General Purpose Input/Output (GPIO) block that provides a 19−bit I/O interface, following are the properties:

- Bi−directional capability.
- Push pull or open drain configuration. (Configured in XBAR)
- Programmable pullup, pulldown or neither. (Configurable In XBAR)
- Individually configurable interrupt lines.
- Individually configurable DMA trigger lines.
- Rising or falling edge interrupt.
- High or low level interrupt.
- Thread safe atomic single−cycle Read−Modify−Write accesses.
- Inputs are sampled using a double flip−flop to avoid meta−stability issues.
- GPIO default state: Input Mode with Pull−down enabled.

GPIO Register Table

All GPIO registers are organized with 1−bit per GPIO. The 4 I/O banks are configured in the GPIO registers as follows:

Table 24.

Bits [20:16]	Bits [15:8]	Bits [5:0]
PC4−PC0	PB7−PB0	PA5−PA0

Table 25. GPIO REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
GPIO DATA REGISTER DEFINITION: 0x40000000 The GPIO Data register is a Read−Only register that returns the synchronized value currently on the pin.					
DATA	[20:8] [5:0]*	0xXXXXXXXX	RO	DATA	Data value: Read Sampled at pin. Read back value goes through double flip−flop synchronization logic with a delay of two cycles.
GPIO DATA OUT REGISTER DEFINITION: 0x40000004 The GPIO Data Out Register is a Read/Write register that contains the value driven out on the GPIO pin if the associated GPIO Output Enable is set to 1.					
DATA_OUT	[20:8] [5:0]*	0x00000000	R/W	DATAOUT	Data output register value: **Read** Current value of data output register. **Write** To data output register.
GPIO OUTPUT ENABLE REGISTER DEFINITION: 0x40000008 The GPIO Output Enable Register is a Read/Write register that contains the current direction configuration of the GPIO.					
OUT_EN	[20:8] [5:0]*	0x00000000	R/W	OUT_EN	Output enable: **Read** Current value of output enables **Write** to set the output enables 0 = Input 1 = Output
GPIO INTERRUPT ENABLE REGISTER DEFINITION: 0x4000000C The GPIO Interrupt Enable Register is a Read/Write register that contains the Interrupt Enable for each GPIO.					
INT_EN	[20:8] [5:0]*	0x00000000	R/W	INT_EN	Interrupt enable: **Read** Current value of interrupt enables **Write** to set the interrupt enables 0 = Disabled 1 = Enabled
GPIO INTERRUPT POLARITY REGISTER DEFINITION: 0x40000010 The Interrupt Polarity Register is a Read/Write register used to configure the Polarity of each GPIO's interrupt.					
INT_POL	[20:8] [5:0]*	0x00000000	R/W	INT_POL	Interrupt Polarity: **Read** Current value of Interrupt Polarity settings **Write** to set the Interrupt Polarity 0 = Low or Falling Edge 1 = High or Rising Edge

Table 25. GPIO REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
GPIO INTERRUPT TYPE REGISTER DEFINITION: 0x40000014 The Interrupt Type Register is a Read/Write register used to configure the type of each GPIO's interrupt.					
INT_TYP	[20:8] [5:0]*	0x00000000	R/W	INT_TYP	Interrupt Type: **Read** Current value of Interrupt Type settings **Write** to set the Interrupt Types 0 = Level Triggered 1 = Edge Triggered
GPIO DMA ENABLE REGISTER DEFINITION: 0x40000018 The GPIO DMA Enable Register is a Read/Write register that is used to enable the DMA trigger for each GPIO. There are four total DMA triggers from the GPIO block to the XBAR – one for each GPIO bank. The DMA trigger for each bank of 8 GPIO's is the OR of each of the 8 GPIO's DMA trigger status.					
DMA_TRIG_EN	[20:8] [5:0]*	0x00000000	R/W	DMA_TRIG_EN	DMA enable: **Read** Current value of DMA enables **Write** to set the DMA enables 0 = Disabled 1 = Enabled **Note**: The DMA Polarity and Type configuration are the same as the Interrupt Polarity and Type.
GPIO INTERRUPT STATUS REGISTER DEFINITION: 0x4000001C The GPIO Interrupt Status Register can be used to detect and clear active interrupts. A read of this register will report status of all interrupts. A read will not clear the interrupt as it does not allow individual interrupt clearing. A write will clear all interrupts with corresponding ones in the corresponding write data bits. Writing zeros to a particular bit will not clear the associated interrupt. If the source of the interrupt has not been removed before clearing the interrupt, the interrupt status will not be cleared. Because of the unique nature of this register, atomic read/modify/write should not be used.					
INT_STS	[20:8] [5:0]*	0x00000000	R/W	INT_STS	Interrupt status: **Read** Current value of interrupt status **Write** ones to all bits that require an interrupt clear

*Bits [31:21] are reserved
Bits [7:6] are reserved

CROSSBAR (XBAR)

The pins are shared between the serial wire debug interface and the internal peripheral functions. The application needs to configure the cross bar to wire the desired peripheral device to the desired I/O pins, given the system requirements. Typically multiple pin options exist for each function to avoid conflicts with other functions. The tables below indicate what peripheral device can be wired to what pin. Each pin may only be connected to one output source, however a pin may be connected to multiple input destinations and it may function as an output of one function and an input to another. This can be useful for operations such as using the timer outputs to be the synchronous clock for the USART.

The crossbar is also used to select sources for DMA trigger, ADC trigger, timer event counting source, and timer capture trigger functions. It is also used to select a pin for external clocking and for debug (trace) and test functions.

Crossbar IO Pin Usage Table

Table 26. CROSSBAR IO PIN USAGE TABLE

PA BANK

OUT CFG	**PA0**	**PA1** (Note 4)	**PA2** (Note 4)	**PA3** (Note 4)	**PA4**	**PA5** (Note 4)
0	GPIO0	GPIO1	GPIO2	GPIO3	GPIO4	GPIO5
1	TIM0OUT	TIM2OUT	PWM0H	TIM1OUT	EXTCLK_OUT	PWM3H
2	PWM2H	PWM1H	SDA	SCL	ACMPO0	USART1_TX
3	SPI_SEL1	PWM0L	SPI_SCK	SPI_SEL0	SPI_DOUT	SPI_SEL2
4	–	–	–	–	–	PWM1L
5	–	–	–	–	–	–
6	–	TIM1OUT	–	–	–	–
7	–	TIM0OUT	–	–	–	–
IN	CAPT1	TIM0CLK USART1_CLK TSTART	USART1_RX SPI_SEL_IN TSTOP CAPT2 SDA	EXTCLK_IN SPI_SCK_IN SCL	TIM1CLK EXT_INT	CAPT0
ANA	ANA_CH0 HSXOSC_P CMP0/1_MI	ANA_CH1 HSXOSC_N CMP0/1_PL	ANA_CH2 CMP0/1_MI	ANA_CH3 LPXOSC_P CMP0/1_PL	ANA_CH4 LPXOSC_N CMP0/1_MI	ANA_CH5 CMP0/1_PL

PB BANK

OUT CFG	**PB0** (Note 4)	**PB1** (Note 4)	**PB2**	**PB3** (Note 4)	**PB4** (Note 4)	**PB5** (Note 4)	**PB6** (Note 4, 5)	**PB7** (Note 4, 5)
0	GPIO8	GPIO9	GPIO10	GPIO11	GPIO12	GPIO13	GPIO14	GPIO15
1	USART1_TX	PWM2H	PWM3H	PWM0H	USART0_TX	TIM1OUT	PWM1H	PWM1L
2	ACMPO0	PWM1H	TIM2OUT	TIM1OUT	–	SPI_SCK	SPI_SEL0	SPI_DOUT
3	EXTCLK_OUT	SDA	SCL	–	SDA	SCL	–	–
4	PWM0L	SPI_DOUT	SPI_SCK	SPI_SEL0	PWM2H	PWM3H	SPI_SEL1	SPI_SEL2
5	–	–	–	–	PC4	–	–	TIM2OUT
6	–	–	TIM1OUT	–	–	–	–	TIM1OUT
7	–	–	TIM0OUT	–	–	–	–	TIM0OUT
IN	CAPT1 ADCTRIG SPI_DIN	USART0_CLK USART1_RX TIM0CLK SDA	CAPT0 SPI_SEL_IN SCL	TIM2CLK WAKEUP (Note 6) SPI_SCK_IN	TIM1CLK SPI_DIN TSTART SDA	USART0_RX SPI_SEL_IN TSTOP SCL	SPI_SCK_IN	USART1_CLK EXT_INT USART0_CLK

PC BANK

OUT CFG	**PC0** (Note 4)	**PC1** (Note 4)	**PC2** (Note 4)	**PC3** (Note 4)	**PC4**
0	GPIO16	GPIO17	GPIO18	GPIO19	GPIO20
1	SPI_SEL0	SPI_SCK	SPI_DOUT	–	ACMPO1
2	TIM0OUT	ACMPO1	USART0_TX	ACMPO0	PWM2H
3	EXTCLK_OUT	TIM2OUT	SPI_SEL1	SPI_SEL2	TIM2OUT
4	PWM3L	PWM0L	PWM1L	PWM2L	–
5	–	–	–	–	PB4
6	–	–	–	–	TIM1OUT
7	–	–	–	–	TIM0OUT
IN	ADCTRIG SPI_SCK_IN CAPT3 TSTART TSTOP	TIM0CLK SPI_SEL_IN EXTCLK_IN	CAPT2 EXT_INT TIM2CLK	USART0_RX SPI_DIN	ADCTRIG TIM1CLK USART1_CLK
AXM0F343−64 MCU: IN		–	–	–	–

4. Required in package. Must be bonded out in package.
5. When DBG_EN = 1 the PB6 = SWDIO and PB7 = SWCLK overriding any other configurations
6. Only valid in shutdown low power mode

Legend:

GPIO# = General purpose I/O #
T#OUT = Timer # roll−over output
T#CLK = Timer # external clock input
IC# = Timer capture # input
PWM#H/PWM#L = High side/low side PWM# output
ACMPO# = Analog comparator outputs
WAKEUP = Shutdown mode wake−up pin (PB3)
SDA/SCL = I^2C data I/O and clock
SPI_[SEL#|SCK] = Master/slave SPI select # out and clock out (master mode)
SPI_[SEL_IN|SCK_IN] = Master/slave SPI select in and clock in (slave mode)
SPI_[DOUT|DIN] = Master/slave SPI data output and data input
USART#_[RX|TX|CLK] = USART # receive data, transmit data, and clock
HSXOSC_P & HSXOSC_N = High Speed Crystal Pins
LPXOSC_P & LPXOSC_N = Low power Crystal Pins
EXT_INT = External interrupt
EXTCLK_[IN|OUT] = External clock in and external clock out
TSTART/TSTOP = Trace start and stop controls
ADCTRIG = ADC trigger

Crossbar Pin Configuration Register Table

Table 27. CROSSBAR PIN CONFIGURATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
XBAR ANALOG ENABLE REGISTER: 0x40500000 The XBAR Analog Enable Register Disconnects the digital inputs from the enabled pads, and turns off digital drive to the pin so that it can be used for analog function.					
ANA_EN	[5:0]	0x0	R/W	ANA_EN	1 = Analog function 0 = Digital function Register bit assignments: [5:0] = PA5−PA0 configuration
XBAR DRIVE TYPE REGISTER: 0x40500004 The XBAR Drive type sets the pin to be either push/pull or open drain. The pin must have the output enabled in the GPIO block or the I/O will not be driven regardless of the drive type setting. Configuring the pin as an output to a peripheral overrides this setting.					
DRIVE_TYPE	[31:0]	0x0	R/W	DRIVE_TYPE	1 = Open Drain 0 = Push/Pull Register bit assignments: [5:0] = PA5−PA0 configuration [15:8] = PB7−PB0 configuration [20:16] = PC4−PC0 configuration [31:24] = Reserved
XBAR PULL−UP REGISTER: 0x40500008 The XBAR Pull−up register allows a weak pull−up to be attached to the pin internally. The weak pull−up is often useful for open drain functions. If both the pull−up and pull−down bits are set for a particular I/O, the I/O bus keeper function is enabled.					
PULL_UP	[31:0]	0x0	R/W	PULL_UP	1 = Pull−up attached 0 = No internal pull−up Register bit assignments: [5:0] = PA5−PA0 configuration [15:8] = PB7−PB0 configuration [20:16] = PC4−PC0 configuration [31:24] = Reserved
XBAR PULL−DOWN REGISTER: 0x4050000C The XBAR Pull−down register allows a weak pull−down to be attached to the pin internally. If both the pull−up and pull−down bits are set for a particular I/O, the I/O bus keeper function is enabled.					
PULL_DOWN	[31:0]	0xFFFFFFFF	R/W	PULL_DOWN	1 = Pull−down attached 0 = No internal pull−down Register bit assignments: [5:0] = PA5−PA0 configuration [15:8] = PB7−PB0 configuration [20:16] = PC4−PC0 configuration [31:24] = Reserved

Crossbar Output Configuration Register Table

Table 28. CROSSBAR OUTPUT CONFIGURATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
XBAR I/O BANK OUTPUT CONFIGURATION REGISTERS: **PA Bank: 0x40500010** **PB Bank: 0x40500014** **PC Bank: 0x40500018** The XBAR PIO Output Configuration Registers are used to configure the output function for the I/O banks.					
PA_CFG PB_CFG PC_CFG	[31:0]	0x0	R/W	PA_CFG PB_CFG PC_CFG	Each IO in each bank has 3−bits to select its configuration according to the XBAR Pin usage configuration table above. The unused bits always read back as 0s: b000 = Config0/GPIO b001 = Config1 b010 = Config2 b011 = Config3 b100 = Config4 b101 = Config5 b110 = Config6 b111 = Config7 Register Bit Assignments: [2:0] = P[A|B|C]0 configuration [6:4] = P[A|B|C]1 configuration [10:8] = P[A|B|C]2 configuration [14:12] = P[A|B|C]3 configuration [18:16] = P[A|B|C]4 configuration [22:20] = P[A|B|C]5 configuration [26:24] = P[A|B|C]6 configuration [30:28] = P[A|B|C]7 configuration

Crossbar Input Function Configuration Register Table

Table 29. CROSSBAR INPUT FUNCTION CONFIGURATION REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
XBAR USART0 INPUT CONFIGURATION REGISTER: 0x40500020 The USART0 Input Configuration Register connects the USART 0 inputs to selected input pins. These pin should normally be un−driven by the GPIO and peripherals.					
IN_USART0_CFG	[4:2]	0x0	R/W	CLK	b000 − CLK = 0 b001 − CLK = PB1 b010 = Reserved b011 − CLK = PB7 b100 = Reserved b101 − b111 = Reserved
	[1:0]	0x0	R/W	RX	b00 − RX = 0 b01 − RX = PB5 b10 − RX = PC3 b11 = Reserved
XBAR USART1 INPUT CONFIGURATION REGISTER: 0x40500024 The USART1 Input Configuration Register connects the USART 1 inputs to selected input pins. These pin should normally be un−driven by the GPIO and peripherals.					
IN_USART1_CFG	[4:2]	0x0	R/W	CLK	b000 − CLK = 0 b001 − CLK = PA1 b010 = Reserved b011 − CLK = PB7 b100 − CLK = PC4 b101 − b111 = Reserved
	[1:0]	0x0	R/W	RX	b00 − RX = 0 b01 − RX = PA2 b10 − RX = PB1 b11 = Not Available

Table 29. CROSSBAR INPUT FUNCTION CONFIGURATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
XBAR SPI INPUT CONFIGURATION REGISTER: 0x40500028 The SPI Input Configuration Register connects the MS SPI inputs to selected input pins. These pins should normally be un−driven by the GPIO and peripherals.					
IN_SPI_CFG	[2:0]	0x0	R/W	IN	b000 − SELECT = 1, SCK = 0, DATAI = 0 b001 = Not Available b010 − SELECT = PB2, SCK = PB3 (slave mode) DATAI = PB0 (master & slave mode) b011 − SELECT = PB5, SCK = PB6 (slave mode), DATAI = PB4 (master & slave mode) b100 − SELECT = PC1, SCK = PC0 (slave mode), DATAI = PC3 (master & slave mode) **Note:** DATAI = MISO in Master Mode, MOSI in Slave Mode
XBAR DMA TRIGGER SELECT REGISTER: 0x40500030 The XBAR DMA Trigger Select Register is used to select the trigger for the DMA Channels. For Each DMA Trigger Select: 0x00 = No connection; 0x01 = ADC Sample; 0x02 = XXX; 0x03 = USART0_RX; 0x04 = USART0_TX; 0x05 = USART1_RX; 0x06 = USART1_TX; 0x07 = SPI_RX; 0x08 = SPI_TX; 0x09 = I2C_TX; 0x0A = I2C_TX; 0x0B = TIM0; 0x0C = TIM1; 0x0D = TIM2; 0x0E = CAPT0; 0x0F = PWM0; 0x10 = CAPT1; 0x11 = PWM1; 0x12 = CAPT2; 0x13 = PWM2; 0x14 = CAPT3; 0x15 = PWM3; 0x16 = TRNG; 0x17 = FLASH; 0x18 = GPIO BANK A; 0x19 = GPIO BANK B; 0x1A = GPIO BANK C; 0x1B = GPIO BANK R; 0x1C−1F = Reserved					
IN_DMA_CFG	[21]	0x0	R/W	CH2_TRIG_TYP	DMA channel 2 trigger type − 0 = SREQ, 1 = REQ
	[20:16]	0x00	R/W	CH2_TRIG_SEL	DMA channel 2 trigger select
	[15:14]	NA	NA	Reserved	
	[13]	0x0	R/W	CH1_TRIG_TYP	DMA channel 1 trigger type − 0 = SREQ, 1 = REQ
	[12:8]	0x00	R/W	CH1_TRIG_SEL	DMA channel 1 trigger select
	[7:6]	NA	NA	Reserved	
	[5]	0x0	R/W	CH0_TRIG_TYP	DMA channel 0 trigger type − 0 = SREQ, 1 = REQ
	[4:0]	0x00	R/W	CH0_TRIG_SEL	DMA channel 0 trigger select
XBAR ADC TRIGGER SELECT REGISTER: 0x40500034 The XBAR ADC Trigger Select Register is used to select the triggers for an ADC conversion. To avoid spurious ADC triggers, clear the ADC_HDW_TRIG bit in the ADC Control register while changing the trigger selection.					
IN_ADC_CFG	[16]	0x0	R/W	TRIG_TYP	0 = PIN 1 = TIMER
	[15:8]	0x0	R/W	PIN_SEL	Select the ADC Pin Trigger: 0x0 = Not Available 0x1 = PB0 0x2 = PC0 0x3 = PC4 0x4−0x7 = Reserved
	[7:0]	0x0	R/W	TIM_SEL	Select the ADC Timer Trigger: 0x0 = TIMER0 0x1 = TIMER1 0x2 = TIMER2 0x3 = Reserved 0x4 = PWM0 0x5 = PWM1 0x6 = PWM2 0x7 = PWM3
XBAR TIMER0 INPUT CONFIGURATION REGISTER: 0x40500038 The Timer0 Input Configuration Register connects the TIMER 0 external count input to the selected input pins. The pin should normally be un−driven in the GPIO and peripherals.					
IN_TIM0_CFG	[1:0]	0x0	R/W	SEL	b00 − TCLK = 0 b01 − TCLK = PA1 b10 − TCLK = PB1 b11 − TCLK = PC1

Table 29. CROSSBAR INPUT FUNCTION CONFIGURATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
XBAR TIMER1 INPUT CONFIGURATION REGISTER: 0x4050003C The Timer1 Input Configuration Register connects the TIMER 1 external count input to the selected input pins. The pin should normally be un−driven in the GPIO and peripherals.					
IN_TIM1_CFG	[1:0]	0x0	R/W	SEL	b00 − TCLK = 0 b01 − TCLK = PA4 b10 − TCLK = PB4 b11 − TCLK = PC4
XBAR TIMER2 INPUT CONFIGURATION REGISTER: 0x40500040 The Timer2 Input Configuration Register connects the TIMER 2 external count input to the selected input pins. The GPI or alternate function is still able to drive this pin as well, so the pin should normally be un−driven in the GPIO, and the weak pull−up disabled in this block.					
IN_TIM2_CFG	[1:0]	0x0	R/W	SEL	b00 − TCLK = 0 b01 = Not Available b10 − TCLK = PB3 b11 = Reserved
XBAR CAPTURE0 INPUT CONFIGURATION REGISTER: 0x40500044 The Capture0 Input Configuration Register connects the PWM CAPTURE0 input to selected input pins. The GPI or alternate function is still able to drive this pin as well, so the pin should normally be un−driven in the GPIO, and the weak pull−up disabled in this block.					
IN_CAPT0_CFG	[1:0]	0x0	R/W	SEL	b00 − CAPT = 0 b01 − CAPT = PA5 b10 − CAPT = PB2 b11 = Not Available
XBAR CAPTURE1 INPUT CONFIGURATION REGISTER: 0x40500048 The Cpture1 Input Configuration Register connects the PWM CAPTURE1 input to selected input pins. The GPI or alternate function is still able to drive this pin as well, so the pin should normally be un−driven in the GPIO, and the weak pull−up disabled in this block.					
IN_CAPT1_CFG	[1:0]	0x0	R/W	SEL	b00 − CAPT = 0 b01 − CAPT = PA0 b10 − CAPT = PB0 b11 = Not Available
XBAR CAPTURE2 INPUT CONFIGURATION REGISTER: 0x4050004C The Capture2 Input Configuration Register connects the PWM CAPTURE2 input to selected input pins. The GPI or alternate function is still able to drive this pin as well, so the pin should normally be un−driven in the GPIO, and the weak pull−up disabled in this block.					
IN_CAPT2_CFG	[1:0]	0x0	R/W	SEL	b00 − CAPT = 0 b01 − CAPT = PC2 b10 = Reserved b11 − CAPT = PA2
XBAR CAPTURE3 INPUT CONFIGURATION REGISTER: 0x40500050 The Capture3 Input Configuration Register connects the PWM CAPTURE3 inputs to selected input pins. The GPI or alternate function is still able to drive this pin as well, so the pin should normally be un−driven in the GPIO, and the weak pull−up disabled in this block.					
IN_CAPT3_CFG	[1:0]	0x0	R/W	SEL	b00 − CAPT = 0 b01 − CAPT = PC0 b10 = Reserved b11 = Not Available
XBAR CLOCK IN SELECT REGISTER: 0x40500054 The Clock In Select Register connects a GPIO to the clock block as a source for the system clock					
IN_EXTCLK_CFG	[2:0]	0x0	R/W	SEL	b000 − CLK_IN = 0 b001 − CLK_IN = PA3 b010 − CLK_IN = PC1 b011 = Reserved b100 = Not Available
XBAR MICRTRACE CONTROL REGISTER: 0x4050005C The MICTROACE Control Register connects GPIOs to the trace start (TSTART) and trace stop (TSTOP) signals, as well as setting the debug enable (NIDEN) signal. One setting allows a single pin to control the start/stop control, with start enabled with the pin high, and stop enabled with the pin low.					
IN_MTB_CFG	[2]	0x0	R/W	NIDEN	1 = Debug enabled, 0 = Debug disables
	[1:0]	0x0	R/W	SEL	b00 − TSTART = 0, TSTOP = 0 b01 − TSTART = PA1, TSTOP = PA2 b10 − TSTART = PB4, TSTOP = PB5 b11 − TSTART = PC0, TSTOP = !PC0

Table 29. CROSSBAR INPUT FUNCTION CONFIGURATION REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
XBAR EXTERNAL INTERRUPT SELECT REGISTER: 0x40500060 The External−Interrupt Select Register connects a GPIO to the interrupt controller as a source for an external interrupt.					
EXT_INT_SEL	[4]	0x1	R/W	INT_POL	0 = Interrupt on pin low 1 = Interrupt on pin high
	[2:0]	0x0	R/W	INT_IN	0x0 = INT_IN = 0 0x1 = INT_IN = PA4 0x2 = INT_IN = PB7 0x3 = INT_IN = PC2 0x4 = Reserved 0x5 = Reserved
XBAR INTERRUPT ENABLE REGISTER: 0x40500064 The Interrupt Enable Register enables or disables interrupt generation from the selected external interrupt pin.					
EXT_INT_EN	[0]	0x0	R/W	EN	External interrupt enable 1 = enabled, 0 = disabled
XBAR INTERRUPT STATUS REGISTER: 0x40500068 The bit in the Interrupt Status Register is set on the selected edge of the pin signal. If the interrupt is also enabled, an interrupt is sent to the processor. Reading this register will not clear the bit. Writing a 1 to bit 0 of this register will clear the bit.					
EXT_INT_STS	[0]	0x0	R/W	PEND	Error interrupt pending 1 = pending, 0 = not pending
XBAR NMI CONFIGURATION REGISTER: 0x4050006C The NMI configuration register selects the source for the Non−Maskable Interrupt to the Cortex M0+. The NMI interrupt is enabled when one or more sources are selected. If multiple sources are selected any of the enabled sources can trigger an NMI interrupt and the individual status registers for each source should be used to determine which triggered the NMI. If no source is enabled, the NMI is inactive.					
NMI_CFG	[3]	0x0	R/W	EXT_INT	0 = External interrupt does not generate an NMI 1 = External interrupt generates an NMI See External Interrupt Select and Status Registers for configuration and interrupt source status.
	[2]	0x0	R/W	CLK_LOSS	0 = Clock loss signal does not generate an NMI 1 = Clock loss signal generates an NMI See Clock Status Register or Oscillator Interrupt Register for interrupt source status.
	[1:0]	0x0	R/W	BRO	0x0 = Brown Out does not generate an NMI 0x1 = Brown Out high generates NMI 0x2 = Reserved (no NMI generation) 0x3 = Brown Out rising edge generates NMI See Brownout Status Register for interrupt source status.
XBAR BROWNOUT STATUS REGISTER: 0x40500070 The bit in the Brownout Status Register is high if the brownout detector is currently indicating a brownout condition.					
BRO_STS	[0]	0x0	R/W	BRO_DET	Brownout detected
AXM0F343−64 MCU: XBAR PWM EXT IN REGISTER: 0x40500074 The PWM Ext In register selects the source of external pin to reset timer or PWM Shutdown signal.					
PWM_EXT_IN	[1:0]	0x0	R/W	EXT_IN	Not Available

DMA (DMA)

The DMA is an implementation of the Arm PL230 µDMA configured with 3 DMA channels. The DMA acts as another bus master on the AHB Bus to facilitate data transfers. Some key features of the DMA controller are:

- Each DMA channel has dedicated handshake signals
- Each DMA channel has a programmable priority level
- Each priority level arbitrates using a fixed priority that is determined by the DMA channel number
- It supports multiple transfer types:
  - memory−to−memory
  - memory−to−peripheral
  - peripheral−to−memory
- It supports multiple DMA cycle types
- It supports multiple DMA transfer data widths
- Each DMA channel can access a primary, and alternate, channel control data structure
- All the channel control data is stored in system memory in little−endian format
- It performs all DMA transfers using the SINGLE AHB−Lite burst type
- The destination data width is equal to the source data width
- The number of transfers in a single DMA cycle can be programmed from 1 to 1024
- The transfer address increment can be greater than the data width
- It has a single output to indicate when an ERROR condition occurs on the AHB bus
- The DMA only works on the AHB and APB busses and thus the destination or source address cannot be to the GPIO which is on the IOP bus

See the "PrimeCell® µDMA Controller (PL230) Revision: r0p0 Technical Reference Manual" for a full description of all features and operation.

DMA Channel Control Data Structure

A 128 byte (32 word) region in the SRAM must be allocated for the DMA Channel Control Data Structure. Each DMA Channel has a Primary Channel Control Structure and an Alternate Channel Control Structure as illustrated below.

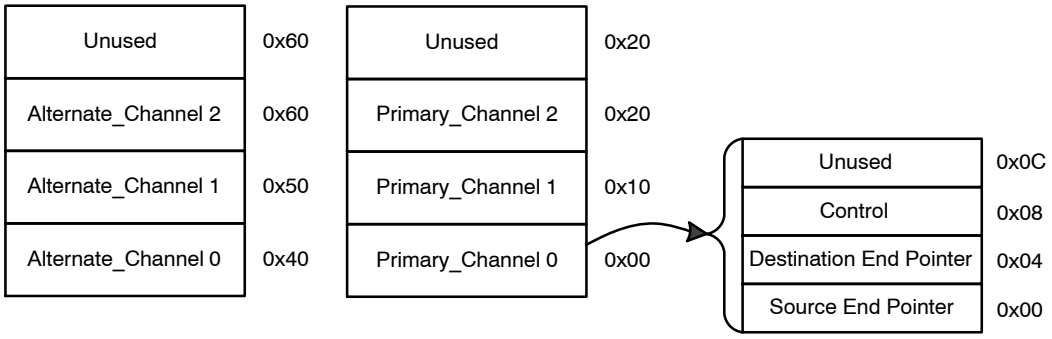


Figure 5. Memory Map for DMA Channel Data Control

DMA Channel Memory Region Table

Table 30. DMA CHANNEL MEMORY REGION TABLE

Function	Bits	Default	Type	Symbol	Description
DMA SOURCE DATA END POINTER: 0x00 The src_data_end_ptr memory location contains a pointer to the end address of the source data.					
SRC_DATA_END_PTR	[31:0]	0x0	R/W	SRC_DATA_END_PTR	Pointer to the end address of the source data
DMA DESTINATION DATA END POINTER: 0x04 The dst_data_end_ptr memory location contains a pointer to the end address of the destination data.					
DST_DATA_END_PTR	[31:0]	0x0	R/W	DST_DATA_END_PTR	Pointer to the end address of the destination data
DMA CONTROL DATA CONFIGURATION: 0x08 For each DMA transfer, the channel_cfg memory location provides the control information for the controller.					

Table 30. DMA CHANNEL MEMORY REGION TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CONTROL DATA CONFIGURATION: 0x08 For each DMA transfer, the channel_cfg memory location provides the control information for the controller.					
CTL	[31:30]	0x0	R/W	DST_INC	Destination address increment. The address increment depends on the source data width as follows: **Source data width = byte** b00 = byte. b01 = halfword. b10 = word. b11 = no increment. Address remains set to the value that the DST_DATA_END_PTR memory location contains. **Source data width = halfword** b00 = reserved. b01 = halfword. b10 = word. b11 = no increment. Address remains set to the value that the DST_DATA_END_PTR memory location contains. **Source data width = word** b00 = reserved. b01 = reserved. b10 = word. b11 = no increment. Address remains set to the value that the DST_DATA_END_PTR memory location contains.
	[29:28]	0x0	R/W	DST_SIZE	Destination data size. **Note:** You must set dst_size to contain the same value that src_size contains.
	[27:26]	0x0	R/W	SRC_INC	Set the bits to control the source address increment. The address increment depends on the source data width as follows: **Source data width = byte** b00 = byte. b01 = halfword. b10 = word. b11 = no increment. Address remains set to the value that the SRC_DATA_END_PTR memory location contains. **Source data width = halfword** b00 = reserved. b01 = halfword. b10 = word. b11 = no increment. Address remains set to the value that the SRC_DATA_END_PTR memory location contains. **Source data width = word** b00 = reserved. b01 = reserved. b10 = word. b11 = no increment. Address remains set to the value that the SRC_DATA_END_PTR memory location contains.
	[25:24]	0x0	R/W	SRC_SIZE	Set the bits to match the size of the source data: b00 = byte b01 = halfword b10 = word b11 = reserved.
	[23:21]	0x0	R/W	DST_PROT_CTL	Set the bits to control the state of HPROT[3:1] when the controller writes the destination data. **Bit [23]** Controls the state of HPROT[3] as follows: 0 = HPROT[3] is LOW and the access is non−cacheable. 1 = HPROT[3] is HIGH and the access is cacheable. **Bit [22]** Controls the state of HPROT[2] as follows: 0 = HPROT[2] is LOW and the access is non−bufferable. 1 = HPROT[2] is HIGH and the access is bufferable. **Bit [21]** Controls the state of HPROT[1] as follows: 0 = HPROT[1] is LOW and the access is non−privileged. 1 = HPROT[1] is HIGH and the access is privileged.

Table 30. DMA CHANNEL MEMORY REGION TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CONTROL DATA CONFIGURATION: 0x08 For each DMA transfer, the channel_cfg memory location provides the control information for the controller.					
	[20:18]	0x0	R/W	SRC_PROT_CTL	Set the bits to control the state of HPROT[3:1] when the controller reads the source data. **Bit [20]** Controls the state of HPROT[3] as follows: 0 = HPROT[3] is LOW and the access is non−cacheable. 1 = HPROT[3] is HIGH and the access is cacheable. **Bit [19]** Controls the state of HPROT[2] as follows: 0 = HPROT[2] is LOW and the access is non−bufferable. 1 = HPROT[2] is HIGH and the access is bufferable. **Bit [18]** Controls the state of HPROT[1] as follows: 0 = HPROT[1] is LOW and the access is non−privileged. 1 = HPROT[1] is HIGH and the access is privileged.
	[17:14]	0x0	R/W	R_POWER	Set these bits to control how many DMA transfers can occur before the controller rearbitrates. The possible arbitration rate settings are: **b0000** Arbitrates after each DMA transfer. **b0001** Arbitrates after 2 DMA transfers. **b0010** Arbitrates after 4 DMA transfers. **b0011** Arbitrates after 8 DMA transfers. **b0100** Arbitrates after 16 DMA transfers. **b0101** Arbitrates after 32 DMA transfers. **b0110** Arbitrates after 64 DMA transfers. **b0111** Arbitrates after 128 DMA transfers. **b1000** Arbitrates after 256 DMA transfers. **b1001** Arbitrates after 512 DMA transfers. **b1010 – b1111** Arbitrates after 1024 DMA transfers. This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.
	[13:4]	0x0	R/W	N_MINUS_1	Prior to the DMA cycle commencing, these bits represent the total number of DMA transfers that the DMA cycle contains. You must set these bits according to the size of DMA cycle that you require. The 10−bit value indicates the number of DMA transfers, minus one. The possible values are: B0000000000 = 1 DMA transfer b0000000001 = 2 DMA transfers b0000000010 = 3 DMA transfers b0000000011 = 4 DMA transfers b0000000100 = 5 DMA transfers . . . b1111111111 = 1024 DMA transfers. The controller updates this field immediately prior to it entering the arbitration process. This enables the controller to store the number of outstanding DMA transfers that are necessary to complete the DMA cycle.

Table 30. DMA CHANNEL MEMORY REGION TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CONTROL DATA CONFIGURATION: 0x08 For each DMA transfer, the channel_cfg memory location provides the control information for the controller.					
	[3]	0x0	R/W	NEXT_USEBURST	Controls if the USE_BURST_SET [C] bit is set to a 1, when the controller is performing a peripheral scatter−gather and is completing a DMA cycle that uses the alternate data structure. **Note:** Immediately prior to completion of the DMA cycle that the alternate data structure specifies, the controller sets the USE_BURST_SET [C] bit to 0 if the number of remaining transfers is less than 2^R. The setting of the next_useburst bit controls if the controller performs an additional modification of the USE_BURST_SET [C] bit. In peripheral scatter−gather DMA cycle then after the DMA cycle that uses the alternate data structure completes, either: **0** = the controller does not change the value of the USE_BURST_SET [C] bit. If the USE_BURST_SET [C] bit is 0 then for all the remaining DMA cycles in the peripheral scatter−gather transaction, the controller responds to requests on **dma_req[]** and **dma_sreq[]**, when it performs a DMA cycle that uses an alternate data structure. **1** = the controller sets the USE_BURST_SET [C] bit to a 1. Therefore, for the remaining DMA cycles in the peripheral scatter−gather transaction, the controller only responds to requests on **dma_req[]**, when it performs a DMA cycle that uses an alternate data structure.
	[2:0]	0x0	R/W	CYCLE_CTRL	The operating mode of the DMA cycle. The modes are: **b000** Stop. Indicates that the data structure is invalid. **b001** Basic. The controller must receive a new request, prior to it entering the arbitration process, to enable the DMA cycle to complete. **b010** Auto−request. The controller automatically inserts a request for the appropriate channel during the arbitration process. This means that the initial request is sufficient to enable the DMA cycle to complete. **b011** Ping−pong. The controller performs a DMA cycle using one of the data structures. After the DMA cycle completes, it performs a DMA cycle using the other data structure. After the DMA cycle completes and provided that the host processor has updated the original data structure, it performs a DMA cycle using the original data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes the cycle_ctrl bits to b001 or b010. **b100** Memory scatter/gather. When the controller operates in memory scatter−gather mode, you must only use this value in the primary data structure. **b101** Memory scatter/gather. When the controller operates in memory scatter−gather mode, you must only use this value in the alternate data structure. **b110** Peripheral scatter/gather. When the controller operates in peripheral scatter−gather mode, you must only use this value in the primary data structure. **b111** Peripheral scatter/gather. When the controller operates in peripheral scatter−gather mode, you must only use this value in the alternate data structure.

DMA Register Table

Table 31. DMA REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
DMA STATUS REGISTER: 0x40800000 The read−only DMA Status Register returns the status of the controller.					
STS	[31:28]	0x00	RO	TEST	To reduce the gate count you can configure the controller, to exclude the integration test logic. Read as: 0x0 = controller does not include the integration test logic 0x1 = controller includes the integration test logic 0x2 − 0xF = undefined.
	[27:21]	−	−	−	Undefined
	[20:16]	0x02	RO	CH_NUM	Number of available DMA channels minus one = 0x02
	[15:8]	−	−	−	Undefined
	[7:4]	0x00	RO	STATE	Current state of the control state machine. State can be one of the following: 0000 = idle 0001 = reading channel controller data 0010 = reading source data end pointer 0011 = reading destination data end pointer 0100 = reading source data 0101 = writing destination data 0110 = waiting for DMA request to clear 0111 = writing channel controller data 1000 = stalled 1001 = done 1010 = peripheral scatter−gather transition 1011 − 1111 = undefined.
	[3:1]	−	−	−	Undefined
	[0]	−	RO	EN	Enable status of the controller: 0 = controller is disabled 1 = controller is enabled.
DMA CONFIGURATION REGISTER: 0x40800004 The write−only dma_cfg Register controls the configuration of the controller.					
CFG	[7:5]	0x0	WO	CH_PROT_CTL	Sets the AHB−Lite protection by controlling the HPROT[3:1] signal levels as follows: **Bit [7]** Controls HPROT [3] to indicate if a cacheable access is occurring. **Bit [6]** Controls HPROT [2] to indicate if a bufferable access is occurring. **Bit [5]** Controls HPROT [1] to indicate if a privileged access is occurring. **Note:** When bit [*n*] = 1 then the corresponding HPROT is HIGH. When bit [*n*] = 0 then the corresponding HPROT is LOW.
	[4:1]	0x0	−	−	Undefined. Write as zero.
	[0]	0x0	WO	EN	Enable for the controller: 0 = disables the controller 1 = enables the controller.
DMA CHANNEL CONTROL DATA BASE POINTER REGISTER: 0x40800008 The ctrl_base_ptr Register is a read/write register. You must configure this register so that the base pointer points to a location in the SRAM.					
CTL_BASE_PTR	[31:7]	0x0	R/W	CTL_BASE_PTR	Pointer to the base address of the primary data structure.
	[6:0]	−	R/W	−	Undefined. Write as zero.
DMA CHANNEL ALTERNATE CONTROL DATA BASE POINTER REGISTER: 0x4080000C The read−only alt_ctrl_base_ptr Register returns the base address of the alternate data structure.					
ALT_CTL_BASE_PTR	[31:0]	0x0	RO	ALT_CTL_BASE_PTR	Base address of the alternate data structure

Table 31. DMA REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CHANNEL WAIT ON REQUEST STATUS REGISTER: 0x40800010 The read−only dma_waitonreq_status Register returns the status of DMA wait on request signal.					
WAIT_ON_REQ_STS	[2:0]	0x0	RO	WAIT_ON_REQ_STS	Channel wait on request status. [0] = DMA Channel 0 wait on request [1] = DMA Channel 1 wait on request [2] = DMA Channel 2 wait on request
DMA CHANNEL SOFTWARE REQUEST REGISTER: 0x40800014 The write−only chnl_sw_request Register enables you to generate a software DMA request.					
SW_REQ	[2:0]	0x0	WO	SW_REQ	Channel wait on request status. [0] = Create DMA Channel 0 request [1] = Create DMA Channel 1 request [2] = Create DMA Channel 2 request
DMA CHANNEL USEBURST SET REGISTER: 0x40800018 The read/write chnl_useburst_set Register disables the single request **dma_sreq** input from generating requests, and therefore only the request, **dma_req**, generates requests. Reading the register returns the useburst status.					
USE_BURST_SET	[2:0]	0x0	R/W	USE_BURST_SET	Returns the useburst status, or disables **dma_sreq[**C**]** from generating DMA requests. Channel Useburst bit assignments: [0] = DMA Channel 0 Useburst [1] = DMA Channel 1 Useburst [2] = DMA Channel 2 Useburst **Read:** 0 = DMA channel responds to requests that it receives on its dma_req[C] 1 = DMA channel does not respond to requests that it receives on its dma_sreq[C]. The controller only responds to dma_req[C] requests and performs 2^R transfers. **Write:** 0 = No effect. Use the chnl_useburst_clr Register to clear bit. 1 = Disables dma_sreq[C] from generating DMA requests. The controller performs 2^R transfers. After the penultimate 2R transfer completes, if the number of remaining transfers, N, is less than 2^R then the controller resets the chnl_useburst_set bit to 0. This enables you to complete the remaining transfers using **dma_req** or **dma_sreq**. **Note:** If you program channel_cfg with a value of N less than 2^R then you must not set the corresponding chnl_useburst_set bit, if the peripheral does not assert **dma_req**. In peripheral scatter−gather mode, if the next_useburst bit is set in channel_cfg then the controller sets the chnl_useburst_set [C] bit to a 1, when it completes the DMA cycle that uses the alternate data structure.
DMA CHANNEL USEBURST CLEAR REGISTER: 0x4080001C The write−only chnl_useburst_clr Register enables **dma_sreq** to generate requests.					
USE_BURST_CLR	[2:0]	0x0	WO	USE_BURST_CLR	Set the appropriate bit to enable **dma_sreq** to generate requests. Channel Useburst bit assignments: [0] = DMA Channel 0 Useburst [1] = DMA Channel 1 Useburst [2] = DMA Channel 2 Useburst

Table 31. DMA REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CHANNEL REQUEST MASK SET REGISTER: 0x40800020 The read/write chnl_req_mask_set Register disables a HIGH on **dma_req**, or **dma_sreq**, from generating a request. Reading the register returns the request mask status for **dma_req** and **dma_sreq**.					
REQ_MASK_SET	[2:0]	0x0	R/W	REQ_MASK_SET	Returns the request mask status of **dma_req** and **dma_sreq**, or disables the corresponding channel from generating DMA requests. Channel Request Mask bit assignments: [0] = DMA Channel 0 Request Mask [1] = DMA Channel 1 Request Mask [2] = DMA Channel 2 Request Mask **Read:** 0 = External Requests are enabled for channel 1 = External Requests are disabled for channel **Write:** 0 = No effect. Use the chnl_req_mask_clr Register to enable DMA requests. 1 = Disables **dma_req[C]** and **dma_sreq[C]** from generating DMA requests.
DMA CHANNEL REQUEST MASK CLEAR REGISTER: 0x40800024 The write−only chnl_req_mask_clr Register enables a HIGH on **dma_req**, or **dma_sreq**, to generate a request.					
REQ_MASK_CLR	[2:0]	0x0	WO	REQ_MASK_CLR	Set the appropriate bit to enable DMA requests for the channel corresponding to **dma_req** and **dma_sreq**. Channel Request Mask bit assignments: [0] = DMA Channel 0 Request Mask [1] = DMA Channel 1 Request Mask [2] = DMA Channel 2 Request Mask **Write:** 0 = No effect. Use the chnl_req_mask_set Register to disable **dma_req** and **dma_sreq** from generating requests. 1 = Enables **dma_req[C]** or **dma_sreq[C]** to generate DMA requests.
DMA CHANNEL ENABLE SET REGISTER: 0x40800028 The read/write chnl_enable_set Register enables you to enable a DMA channel. Reading the register returns the enable status of the channels.					
EN_SET	[2:0]	0x0	R/W	EN_SET	Returns the enable status of the channels, or enables the corresponding channels. Channel Enable bit assignments: [0] = DMA Channel 0 Enable [1] = DMA Channel 1 Enable [2] = DMA Channel 2 Enable **Read:** 0 = Channel is disabled 1 = Channel is enabled **Write:** 0 = No effect. Use the chnl_enable_clr Register to disable a channel. 1 = Enables channel.

Table 31. DMA REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CHANNEL ENABLE CLEAR REGISTER: 0x4080002C The write−only chnl_enable_clr Register enables you to disable a DMA channel.					
EN_CLR	[2:0]	0x0	WO	EN_CLR	Set the appropriate bit to disable the corresponding DMA channel. Channel Enable bit assignments: [0] = DMA Channel 0 Enable [1] = DMA Channel 1 Enable [2] = DMA Channel 2 Enable **Write:** 0 = No effect. Use the chnl_enable_set Register to enable DMA channels. 1 = Disables channel
DMA CHANNEL PRIMARY−ALTERNATE SET REGISTER: 0x40800030 The read/write chnl_pri_alt_set Register enables you to configure a DMA channel to use the alternate data structure. Reading the register returns the status of which data structure is in use for the corresponding DMA channel.					
PRI_ALT_SEL_SET	[2:0]	0x0	R/W	PRI_ALT_SEL_SET	Returns the channel control data structure status, or selects the alternate data structure for the corresponding DMA channel. Channel Primary−Alternate bit assignments: [0] = DMA Channel 0 Primary−Alternate [1] = DMA Channel 1 Primary−Alternate [2] = DMA Channel 2 Primary−Alternate **Read:** 0 = DMA channel is using the primary data structure. 1 = DMA channel is using the alternate data structure. **Write:** 0 = No effect. Use the chnl_pri_alt_clr Register to set bit [C] to 0. 1 = Selects the alternate data structure for channel. **Note:** The controller toggles the value of the chnl_pri_alt_set [C] bit after it completes: • the four transfers that the primary data structure specifies for a memory scatter−gather, or peripheral scatter−gather, DMA cycle • all the transfers that the primary data structure specifies for a ping−pong DMA cycle • all the transfers that the alternate data structure specifies for the following DMA cycle types: — Ping−pong — Memory scatter−gather — Peripheral scatter−gather.

Table 31. DMA REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA CHANNEL PRIMARY−ALTERNATE CLEAR REGISTER: 0x40800034 The write−only chnl_pri_alt_clr Register enables you to configure a DMA channel to use the primary data structure.					
PRI_ALT_CLR	[2:0]	0x0	WO	PRI_ALT_CLR	Set the appropriate bit to select the primary data structure for the corresponding DMA channel. Channel Enable bit assignments: [0] = DMA Channel 0 Primary−Alternate [1] = DMA Channel 1 Primary−Alternate [2] = DMA Channel 2 Primary−Alternate **Write:** 0 = No effect. Use the chnl_pri_alt_set Register to select the alternate data structure. 1 = Selects the primary data structure for channel. **Note:** The controller toggles the value of the chnl_pri_alt_set [C] bit after it completes: • the four transfers that the primary data structure specifies for a memory scatter−gather, or peripheral scatter−gather, DMA cycle • all the transfers that the primary data structure specifies for a ping−pong DMA cycle • all the transfers that the alternate data structure specifies for the following DMA cycle types: — Ping−pong — Memory scatter−gather — Peripheral scatter−gather.
DMA CHANNEL PRIORITY SET REGISTER: 0x40800038 The read/write chnl_priority_set Register enables you to configure a DMA channel to use the high priority level. Reading the register returns the status of the channel priority mask.					
PRIO_SET	[2:0]	0x0	R/W	PRIO_SET	Returns the channel priority mask status, or sets the channel priority to high. Channel Priority bit assignments: [0] = DMA Channel 0 Priority [1] = DMA Channel 1 Priority [2] = DMA Channel 2 Priority **Read:** 0 = DMA channel C is using the default priority level. 1 = DMA channel C is using a high priority level. **Write:** 0 = No effect. Use the chnl_priority_clr Register to set channel C to the default priority level. 1 = Channel C uses the high priority level.
DMA CHANNEL PRIORITY CLEAR REGISTER: 0x4080003C The write−only chnl_priority_clr Register enables you to configure a DMA channel to use the default priority level.					
PRIO_CLR	[2:0]	0x0	WO	PRIO_CLR	Set the appropriate bit to select the default priority level for the specified DMA channel. Channel Enable bit assignments: [0] = DMA Channel 0 Priority [1] = DMA Channel 1 Priority [2] = DMA Channel 2 Priority **Write:** 0 = No effect. Use the chnl_priority_set Register to set channel to the high priority level. 1 = Channel uses the default priority level.

Table 31. DMA REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
DMA BUS ERROR CLEAR REGISTER: 0x4080004C The read/write err_clr Register returns the status of dma_err, and enables you to set dma_err LOW.					
ERR_CLR	[0]	0x0	R/W	ERR_CLR	Returns the status of **dma_err**, or sets the signal LOW. **Read:** 0 = **dma_err** is LOW 1 = **dma_err** is HIGH. **Write:** 0 = No effect, status of **dma_err** is unchanged. 1 = Sets **dma_err** LOW. **Note:** If you de−assert **dma_err** at the same time as an ERROR occurs on the AHB−Lite bus, then the ERROR condition takes precedence and **dma_err** remains asserted.
DMA INTERRUPT ENABLE REGISTER: 0x40800050 The Interrupt Enable Register is a read/write register. This enables or disables interrupt generation from each of the three channels, or on error.					
INT_EN	[3]	0x0	R/W	ERR	Error interrupt enable 1: enabled, 0: disabled
	[2:0]	0x0	R/W	DONE	Enable done interrupts for each channel 1: enabled, 0: disabled
DMA INTERRUPT STATUS REGISTER: 0x40800054 Bits in the Interrupt Status Register is set on the rising edge of the error signal, or the done signals from each channel. If the associated interrupt is also enabled, an interrupt is sent to the processor. Reading this register will not clear the bits, as that would not allow individual bits to be cleared independently. Writing to this register will clear any bits that have a '1' written to them.					
INT_STS	[3]	0x0	R/W	ERR	Error interrupt pending 1: pending, 0: not pending
	[2:0]	0x0	R/W	DONE	Done interrupts for each channel 1: pending, 0: not pending

EXTERNAL COMMUNICATION INTERFACES

Universal Synchronous/Asynchronous Receiver/Transmitter (USART0 and USART1)

There are two Universal Synchronous/Asynchronous Receiver/Transmitter modules (USART0 and USART1) with the following features:

- Flexible RX/TX connections through the crossbar configuration
- Synchronous and asynchronous modes
- Variable word lengths: 5−9 bits
- Variable stop bits: 1−2 bits
- Timer generated (TIM0, TIM1, and TIM2) arbitrary baud rates for asynchronous mode
- Parity generation and parity check support through software
- RX break detection and TX break initiation
- Optional receiver deglitching

The USART includes a single byte buffer. This is sufficient for most simple embedded applications. For example, a processor running at 40 MHz with a baud rate of 115200, 8−bit data with 1 stop bit, and no parity means a character transfer every 40e6 x (1 + 8 + 1) / 115200 = 3472 cycles. For duplex communication, the processor might receive an interrupt every 868 clock cycles. Because the interrupt response time and the handler execution time are usually quite short, this leaves sufficient processing time for the thread.

Configuration and Control

Asynchronous/Synchronous Mode

The USART supports use of both asynchronous and synchronous modes. The SYNC bit is used to configure modes. In asynchronous mode the data rate is controlled by an internally generated baud rate clock. Both ends of the transmission must have aligned baud rates (within the error tolerance) or data will be corrupted. This configuration requires the least amount of interconnect (RX and TX wires only).

In synchronous mode a clock is provided to align data between sender and receiver. The clock determines the baud rate so baud rate variation between sender and receiver is not an issue. There is one clock per bit. The clock edge used for sampling the RX pin and changing the TX pin are independently controlled with the EDGE configuration bits.

Baud Rate Generation (Asynchronous Mode)

The baud rate generation clock comes from the selected 16b general purpose timer. Both USART can simultaneously use the same timer if the baud rates match. The timer must be configured to produce oscillation on the TimxOUT signal at a rate that is 16X the desired baud rate. The BRG configuration bits are used to select the source timer for the baud rate generation. For lowest power, BRG should be kept at the disabled state ‘000’ (and SYNC should be low) when USART functionality is disabled.

For tightest accuracy baud rate generation it is recommended to use up count mode with programmable step size (saw tooth up multiply mode − See section on 16−bit General Purpose Timers). The period/step size register would be set as follows:

$$\text{Step Size} = \frac{65536 \cdot \text{BaudRate} \cdot 16}{F_{\text{TimerClock}}} \quad \text{(eq. 6)}$$

For example with a 40 MHz timer clock frequency and a baud rate of 115200 you would have a step size of (65536 x 115200 x 16) / 40 MHz = 3020. With this step size your actual baud rate (ignoring timer clock source error) will be (3020 x 40M / (65536 x 16)) = 115204 which is only 0.0035% error.

The RX data is sampled in the middle of the expected bit width (according to the 16X baud rate clock referenced to the falling edge of the start bit). This allows roughly half a bit of accumulative error allowed across the whole transaction due to clock rate inaccuracy.

Configuration and Control

Transaction Enable and Configuration

The transmitter and receiver are enabled individually by setting the TX_EN and RX_EN bits.

The WORD_LEN bits are used to set the word length of the send and receive data. Values from 5−9 bits are possible. The STOP_BITS bit configures the number of stop bits. This setting only affects the number of stop bits put out by the transmitter. The receiver will always accept one stop bit.

Parity Generation and Parity Checks

Parity is not supported natively in hardware. However, polarity support can be achieved by increasing the data word length by one bit and having software generate or check the last bit as a parity bit. For example an 8b data transaction would use 9b word length with the extra bit being interpreted as the parity bit. A simple parity calculation can be performed on a data word and the resulting parity bit concatenated to the data word on outgoing transactions. Incoming transactions could be checked for proper parity before extracting the data word.

Break Transmission

Setting TX_BRK_EN bit to a ‘1’ forces the TX line low (after any pending TX transaction completes) until released (by writing a ‘0’) to indicate a break condition on the line. Breaks are typically used to get attention of the other party when proper communication isn’t established.

Receiver De−glitch

Improved noise immunity can be achieved in a noisy system by setting the RX_DGL_EN bit. When deglitching is enabled the RX signal must have a stable value for 3 baud rate clocks (16X the source clock) clocks before a transition is recognized. This will filter out noise (pulse rejection and noisy transitions) on the order of 3/16 of a bit period.

Multi−processor Mode (MCE)

The MCE bit can be used to help support a multi−processor shared communication line mode. When enabled the received data word is ignored if the last bit of the received word is not ‘1’. In this case a configured data width would be increased by one and the external controller would append the select bit to the end of the regular data word.

Status and Interrupts

There are 7 bits of status (4 for RX and 3 for TX) and up to 5 configurable interrupts (3 for RX and 2 for TX).

Receiver Break Detection (RX_BRK)

A break condition occurs when the receiver input is at the logic low level for longer than some duration of time, typically, for more than a character time. This is not necessarily an error, but appears to the receiver as a character of all zero bits with a framing error. The break signal can be used as an attention signal when normal communication is not possible because of things like baud rate mismatch. When a RX line break (low for the whole RX frame) is detected RX_BRK goes high. The RX_BRK interrupt (if enabled by INT_EN_RX_BRK) is cleared when break condition is removed.

Receiver Frame Error (RX_FE)

A frame error occurs when the designated start and stop bits are not found. As the start bit is used to identify the beginning of an incoming character, it acts as a reference for the remaining bits. If the data line is not in the expected state when the “stop” bit is expected, a framing error will occur. RX_FE indicates that a frame error was detected. The RX_FE interrupt (if enabled by INT_EN_RX_FE) is cleared by writing a ‘1’ to the RX_FE bit in the status register.

Receiver Full and Overrun (RX_FULL/RX_OVER)

When a character is received the RX_FULL bit it set to ‘1’. If the INT_EN_RX bit is set it will also trigger an interrupt indicating to the micro that RX data is ready for processing. Reading the received data will clear the status bit and the interrupt.

Overrun occurs when a new character is received before the previous character has been read out of the data buffer by the micro. It is an indicator that data has been lost. RX_OVER goes high on this condition. RX_OVER is a status bit only that can be checked as part of a RX_FULL interrupt service routine.

Transmitter Idle (TX_IDLE)

The TX_IDLE bit tells whether the USART is busy or idle. It is high when the transmit buffer is empty and a transmit transaction isn’t in progress. If the INT_EN_TX_IDLE configuration bit is set then an interrupt will be issued when the transmit module goes idle and there isn’t new data to send. TX_IDLE status and the corresponding interrupt will be cleared when new data is written to the transmit buffer.

Transmitter Empty (TX_EMPTY)

TX_EMPTY indicates that the transmit buffer is empty and is ready for new data. The TX_EMPTY bit is cleared when data is written to the transmit buffer. It is set when the transmit data has been copied into the outgoing shift register and the transmit buffer is once again ready for a new value. The associated interrupt is enabled by the INT_EN_TX configuration bit. This interrupt can be used to keep ahead of the transmitter by filling the transmit buffer with the next character to be sent before the transmitter goes idle.

USART Register Table

Table 32. USART REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
USART0 DATA REGISTER: 0X41B00000 **USART1 DATA REGISTER: 0X41C00000**					
DATA	[8:0]	0x000	R/W	DATA	Data register to read the receive data and write the transmit data
USART0 CONFIGURATION REGISTER: 0x41B00004 **USART1 CONFIGURATION REGISTER: 0x41C00004**					
CFG	[16]	0x0	R/W	TX_IDLE_INT_EN	Transmitter Idle interrupt enable
	[15]	0x0	R/W	SYNC	0 = Asynchronous mode, 1 = Synchronous mode
	[14]	0x0	R/W	MCE	Multiprocessor Communication Enable; If set, the received byte is only stored in the receive register if the top most bit is set
	[13]	0x0	R/W	RX_BRK _INT_EN	Receiver Break detect change interrupt enable
	[12]	0x0	R/W	RX_FE_INT_EN	Receiver Framing error interrupt enable
	[11]	0x0	R/W	TS_INT_EN	Transmitter interrupt enable
	[10]	0x0	R/W	RX_ INT_EN	Receiver interrupt enable
	[9]	0x0	R/W	TX_EN	Transmitter enable
	[8]	0x0	R/W	RX_EN	Receiver enable
	[7]	0x0	R/W	TX_BRK_EN	Transmit break enable. Setting this bit to a ‘1’ forces the TX line low (after any pending TX transaction completes) until released (by writing a ‘0’) to indicate a break condition on the line.
	[6]	0x0	R/W	RX_DGL_EN	Receiver deglitch enable. If enabled additional filtering (a check for stable transition data over 3 consecutive USART clock cycles) is added to the RX input stream. Use in noisy environments.
	[5]	0x0	R/W	STOP_BITS	Stop Bits; 0 = 1 Stop Bit, 1 = 2 Stop Bits
	[4:2]	0x0	R/W	WORD_LEN	Word Length 000: 8−Bits 001: 9−Bits 010−100: Reserved 101: 5−Bits 110: 6−Bits 111: 7−Bits
	[1:0]	0x0	R/W	BRG_EDGE	BRG in ASYNC mode: Baud Rate Generator (TxOUT = 16 × baud rate) 00: Off 01: Timer 0 10: Timer 1 11: Timer 2 EDGE In SYNC mode: −0: receiver samples on falling edge −1: receiver samples on rising edge 0−: transmitter changes on rising edge 1−: transmitter changes on falling edge
USART0 STATUS REGISTER: 0x401B0008 **USART1 STATUS REGISTER: 0x401C0008**					
STS	[6]	0x0	RO	TX_IDLE	Tx Idl (‘1’ = IDLE, ‘0’ = BUSY)
	[5]	0x0	RW	RX_BRK	Rx Break Detected interrupt request (automatically clears when break condition is removed)
	[4]	0x0	RW	RX_FE	Rx Framing Error interrupt request (write 1 to clear)
	[3]	0x0	Unused	Unused	Unused
	[2]	0x0	RO	TX_EMPTY	Tx Empty interrupt request (write TX data to clear)
	[1]	0x0	RW	RX_OVER	Rx Overrun status bit. This bit does not raise an interrupt request. (write 1 to clear)
	[0]	0x0	RO	RX_FULL	Rx Full interrupt request (read RX data to clear)

Master/Slave SPI Controller (SPI)

The Master/Slave Serial Peripheral Interface Controller (SPI) is a synchronous serial data link controller that can be configured as either a SPI slave peripheral or a SPI master controller. A flexible clocking scheme allows it to connect to a variety of external SPI compatible peripherals or master controllers. The SPI controller has configuration options for:

- Data frame width (8, 16, 24, and 32−bit modes)
- Clock phase (CPHA) and polarity (CPOL)
- 4 or 3 wire mode SPI slave (with or without slave select functionality)
- Master clock source and clock pre−scale
- Data direction (MSB or LSB first)
- Interrupt generation for RX/TX events and slave select (SS) events

The SPI peripheral can also be used for additional serial data link protocols. It supports the 2−channel audio optimized protocol I^2S including the standard and left justified framing modes. It also supports PCM style data link protocols with both early and late frame syncing.

Master/Slave SPI Block Diagram (SPI)

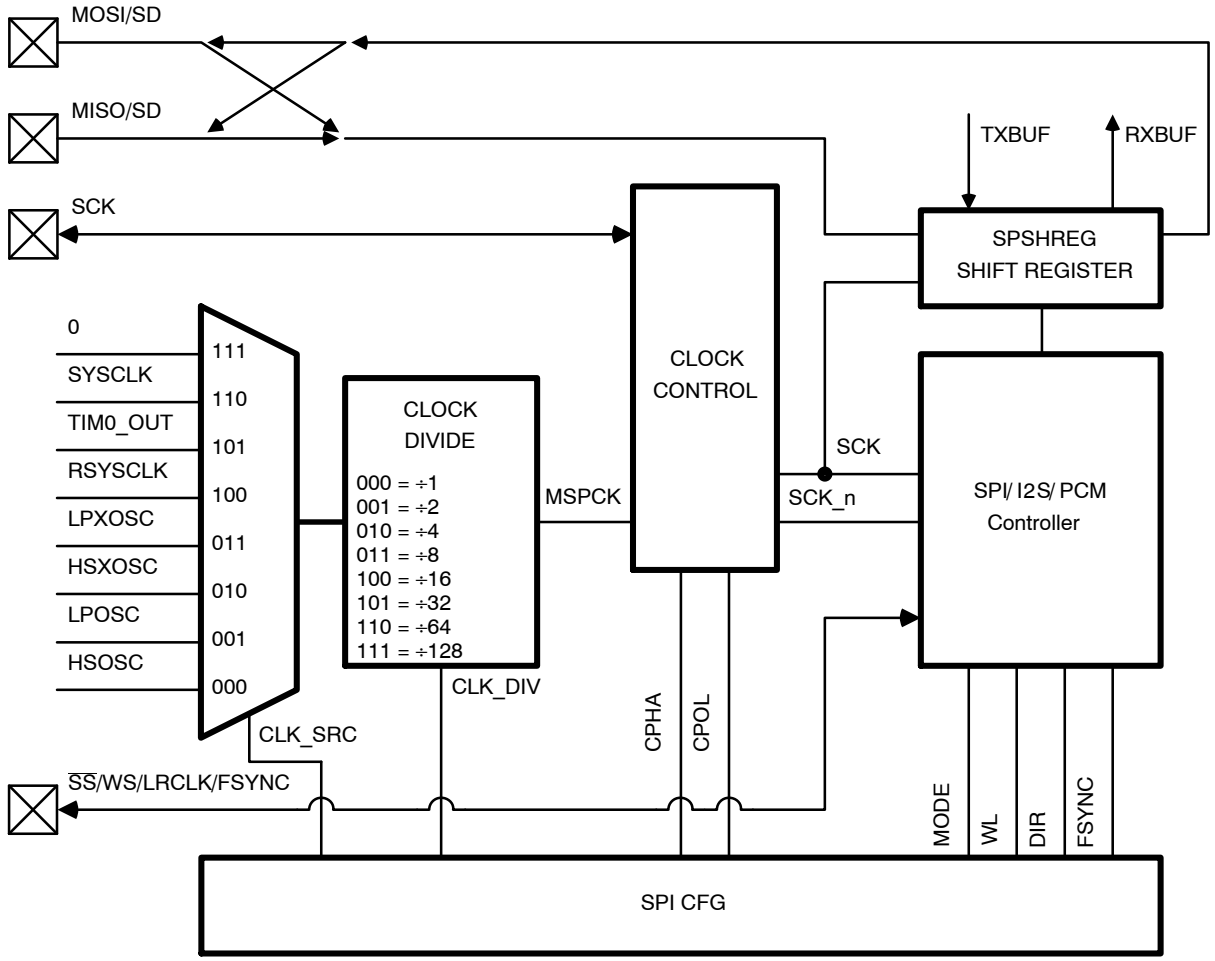


Figure 6. Master/Slave SPI Block Diagram (SPI)

Master/Slave SPI Framing

The following waveforms show options for SPI style data framing.

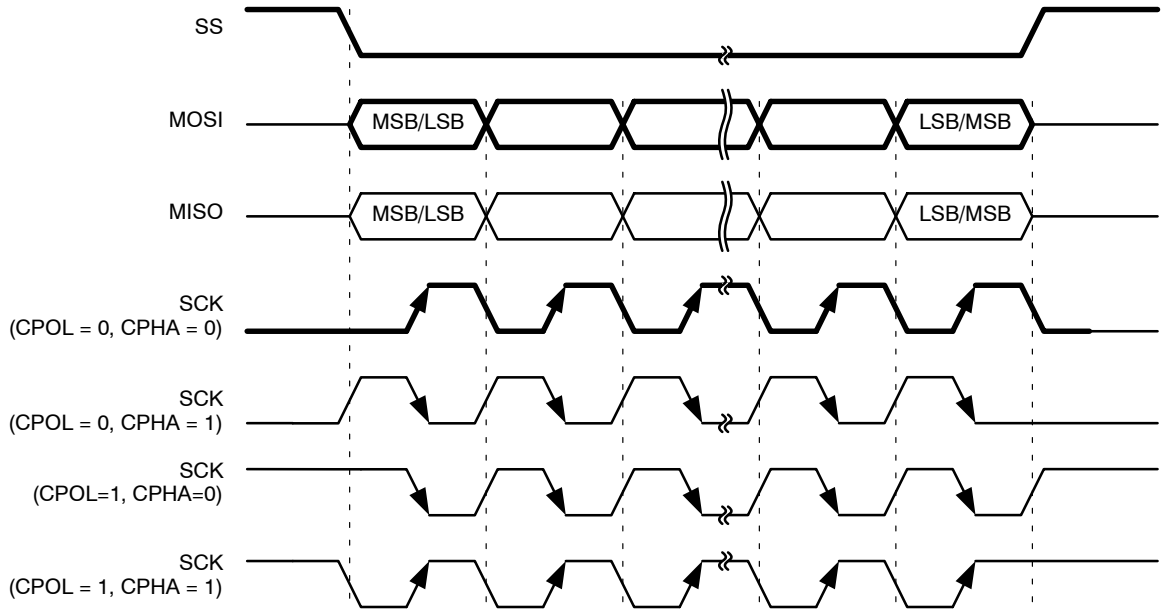


Figure 7.

Master/Slave SPI I²S Framing

The following waveforms show options for I²S style data framing. Standard I²S framing drops LRCLK low one cycle before the MSB of the left channel and sets it high one cycle before the MSB of the right channel. Left justified mode reverses the polarity of the left/right indicator (LRCLK) and removes the clock delay for the first bit in relation to the frame sync signal.

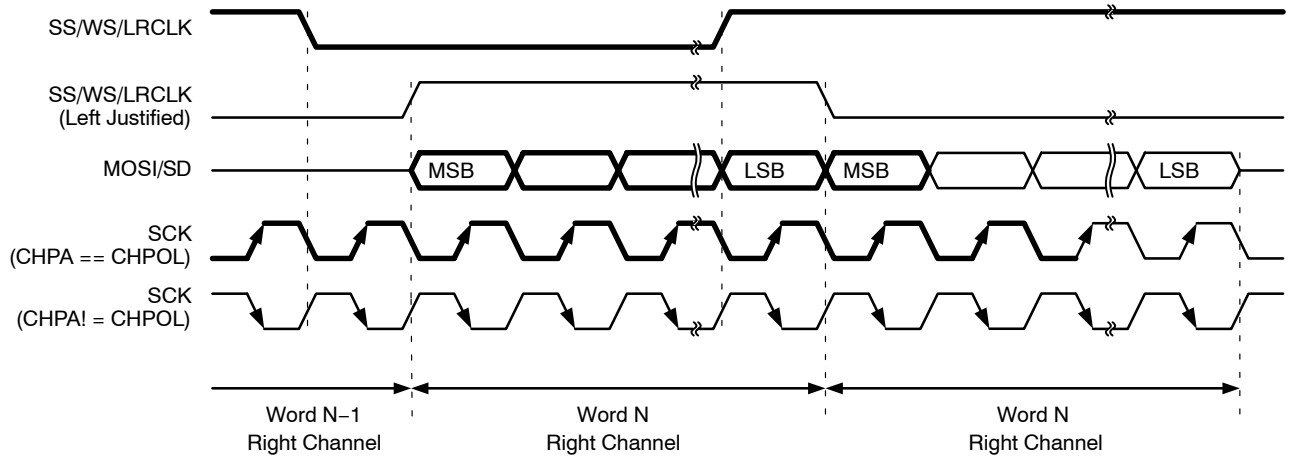


Figure 8.

Illegal combinations of controller options are possible and should be avoided. For example, I²S is defined to be MSB first. The SPI peripheral will allow I²S framing with LSB first data without complaint.

Master/Slave SPI PCM Style Framing

The following waveforms show options for PCM style data framing. Long word FSYNC is not supported. Left and right justified data is supported through software.

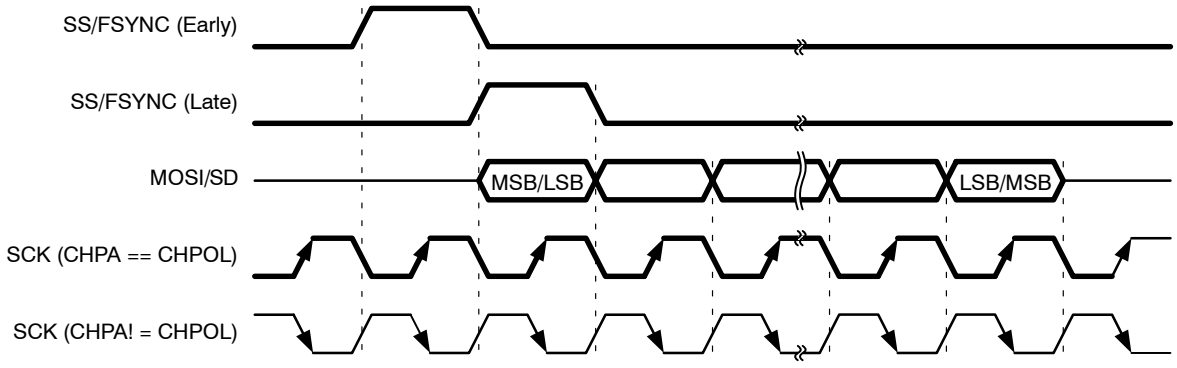


Figure 9.

Master/Slave SPI (SPI) Register Table

Table 33. MASTER/SLAVE SPI (SPI) REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
SPI SHIFT REGISTER: 0x41A00000 The SPI Shift Register contains the transmit (TX) and receive (RX) data for the SPI transfer. Writes to the shift register are shifted out of the part (TXBUF). Reading the shift register returns data that was shifted into the part (RXBUF).					
DATA	[31:0]	0x00000000	R/W	DATA	SPI shift data register R = RXBUF W = TXBUF
SPI CONFIGURATION REGISTER: 0x41A00004 The MS_SPI Configuration Register is used to configure the transaction parameters. The System Clock may stop if the processor enters standby mode. Therefore, if System Clock is selected as SPI clock source, an ongoing SPI transaction may be halted while the processor is in standby mode.					
CFG	[18]	0x0	R/W	SS_CHANGE_INT_EN	SPI SS change interrupt enable
	[17]	0x0	R/W	TX_INT_EN	SPI Transmit interrupt enable
	[16]	0x0	R/W	RX_INT_EN	SPI Receive interrupt enable
	[15]	0	R/W	CLK_PHASE	SPI Clock Phase
	[14]	0	R/W	CLK_POL	SPI Clock Polarity
	[13:11]	000	R/W	CLK_DIV	SPI Clock pre−scaler 000: ÷1 001: ÷2 010: ÷4 011: ÷8 100: ÷16 101: ÷32 110: ÷64 111: ÷128
	[10:8]	111	R/W	CLK_SRC	Clock Source 000: HSOSC 001: LPOSC 010: XOSC 011: LPXOSC 100: EXTCLK 101: T0OUT 110: System Clock (1) 111: Off
	[7]	0x0	R/W	DIR	Shift Direction, 0 = MSB first, 1 = LSB first

Table 33. MASTER/SLAVE SPI (SPI) REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
SPI CONFIGURATION REGISTER: 0x41A00004 The MS_SPI Configuration Register is used to configure the transaction parameters. The System Clock may stop if the processor enters standby mode. Therefore, if System Clock is selected as SPI clock source, an ongoing SPI transaction may be halted while the processor is in standby mode.					
	[6:4]	000	R/W	FSYNC	Frame Sync Mode: 000 = SPI SEL 0 001 = SPI SEL 1 010 = SPI SEL2 011 = SPI no SEL 100 = I2S Standard 101 = I2S Left Justified 110 = PCM early FSYNC 111 = PCM late FSYNC
	[3:2]	00	R/W	WL	Word Length: 00 = 8 bits 01 = 16 bits 10 = 24 bits 11 = 32 bits
	[1:0]	00	R/W	MODE	Mode: 00 = Off 01 = Do not use 10 = Master 11 = Slave
SPI STATUS REGISTER 0x41A00008 The SPI Status Register is a read only register that allows the software to observe status of various SPI state and events.					
STS	[6]	0x0	RO	FIRST	First Word
	[5]	0x1	RO	SSSTAT	Current SS status
	[4]	0x0	R/W	SSCHG	SS change interrupt request (write 1 to clear)
	[3]	0x0	R/W	TX_UNDER	TX Underrun interrupt request (write 1 to clear)
	[2]	0x1	RO	TX_EMPTY	TX Empty interrupt request
	[1]	0x0	R/W	RX_OVER	RX Overrun interrupt request (write 1 to clear)
	[0]	0x0	RO	RX_FULL	RX Full interrupt request

I2C Controller (I2C)

The I2C interface is compatible with the Inter−IC Bus Specification from Philips Semiconductors®. The I2C interface uses a two−wire interface including a bidirectional clock line (SCL) and bidirectional data line (SDA). These pins are connected to the pins through the crossbar. This interface is designed for communications with external devices. The I2C interface will support both master and slave mode operation. This module connects to the DMA controller through a DMA channel to support moving data without intervention by the processor.

Master and slave modes can be enabled at the same time. The SLAVE_EN register effectively only enables slave address recognition, including recognition of the general call address (0x00).

To help maximize throughput the I2C interface makes use of a single byte buffer. Depending on the configuration modes the buffer is used to automatically start a new transmission or copy a received byte. The buffer will be used for both Slave and Master modes. The buffer is hidden and is automatically accessed through the regular I2C data register.

I2C Slave Mode

1. For I2C slave mode, the device receives the clock (SCL) from an external master device.
2. The I2C slave interface uses a configurable address that identifies the device.
3. Both pins must be connected to a positive supply voltage using a pull−up resistor.
4. It responds to communications under two conditions:
   a. With the configured address or
   b. The general call address (0x00).
5. The system clock must be set to at least 2.5x faster than the SCL clock rate.
6. A maximum transmission rate of 400 kbps (Fast−mode) is fully supported.
7. A maximum transmission rate of 1 Mbps (Fast−mode plus) is supported with the exception of the pad drive strength required by the I2C standard (20 mA at 0.4 V). Multiple DIO pads can be used in parallel to increase the drive strength, but this may not be sufficient to meet the specifications, especially at low VDDO.

I2C Slave Mode Behavior Table

Table 34. I2C SLAVE MODE BEHAVIOR TABLE

Field	Behavior			
Mode	**Auto ACK Off, Write Frame**	**Auto ACK Off, Read Frame**	**Auto ACK On, Write Frame**	**Auto ACK On, Read Frame**
Start	RX	RX	RX	RX
Address	RX	RX	RX	RX
ACK	Interrupt if address matched, stretch clock until ACK/NACK	Interrupt if address matched, stretch clock until ACK/NACK	ACK if address matched	ACK if address matched
Data 0	RX	Interrupt if buffer empty, stretch until data available, TX	RX	Interrupt if buffer empty, stretch until data available, TX, Interrupt
ACK	Interrupt, stretch until clock until ACK /NACK	Get ACK/NACK from remote	Interrupt, ACK if buffer free else stretch clock until free	Get ACK/NACK from remote
Data N	RX	Interrupt if buffer empty, stretch until data available, TX	RX	Interrupt if buffer empty, stretch until data available, TX, Interrupt
ACK	Interrupt, stretch until clock until ACK /NACK	Get ACK/NACK from remote	Interrupt, ACK if buffer free else stretch clock until free	Get ACK/NACK from remote
Stop	RX, Interrupt if stop interrupt enabled	RX, Interrupt if stop interrupt enabled	RX, Interrupt if stop interrupt enabled	RX, Interrupt if stop interrupt enabled

I2C Master Mode

1. For I2C master mode, the device generates the clock (SCL) to communicate with an external slave (or multiple slave devices).
2. Maximum supported transmission rate is the same as in slave mode, and the clock provided to the SCL line is configurable.
3. Multi−master mode is also supported and has to be handled by the firmware in case of conflict through the bus−error control (Section 10.1.18.6).
4. In order to issue a start condition and address a slave device the interface makes use of a start and address register that is separate from the I2C data register. This register can also be used after an acknowledgement to produce a repeated start condition.
5. When using the I2C interface in master mode, a write to the I2C_ADDR_START register is used to initiate a transaction.
6. The data written to this register is interpreted as the direction (READ_WRITE field) and address (ADDRESS field) to be sent in the addressing byte of the transaction.
7. If a transaction is currently in progress, writing to the I2C_ADDR_START register causes a repeated start condition terminating the previous transaction.

I²C Master Mode Behavior Table

Table 35. I²C SLAVE MODE BEHAVIOR TABLE

Field	Behavior			
Mode	**Auto ACK Off, Read Frame**	**Auto ACK Off, Write Frame**	**Auto ACK On, Read Frame**	**Auto ACK On, Write Frame**
Start	TX	TX	TX	TX
Address	TX	TX	TX	TX
ACK	Get ACK/NACK from remote	Get ACK/NACK from remote	Get ACK/NACK from remote	Get ACK/NACK from remote
Data 0	Interrupt, stretch clock until ACK/NACK RX	Interrupt if buffer empty, stretch until data available, TX	RX	Interrupt if buffer empty, stretch until data available, TX, Interrupt
ACK	Interrupt, stretch until clock until ACK/NACK	Get ACK/NACK from remote	Interrupt, ACK if buffer free else stretch clock until free	Get ACK/NACK from remote
Data N	RX	Interrupt if buffer empty, stretch until data available, TX	RX	Interrupt if buffer empty, stretch until data available, TX, Interrupt
ACK	Interrupt, stretch until clock until ACK/NACK	Get ACK/NACK from remote	Interrupt, ACK if buffer free else stretch clock until free	Get ACK/NACK from remote
Stop	TX, Interrupt if stop interrupt enabled	TX, Interrupt if stop interrupt enabled	TX, Interrupt if stop interrupt enabled	TX, Interrupt if stop interrupt enabled

I²C Operation

Non−Auto Acknowledge Mode

The attributes of the Non−Auto Acknowledge mode are:

1. When auto acknowledge mode is disabled, the firmware controls the clock stretching and acknowledgement of all bytes. The firmware is also aware of if the port was addressed on the general call address or the programmed address, the I²C lines state, and the acknowledgement status of the previous byte.
2. Interrupts to the firmware are issued as follows:
   a. Every time a data or acknowledge is required.
   b. When a stop condition is detected provided that the stop condition interrupt is enabled when in Slave mode, Master mode is required to not be enabled.
   c. In master read frame non−auto acknowledge mode, an additional interrupt is generated once the address acknowledge bit has been received.
      i. No data from the firmware is required in that special case but it helps to control the state of the interface by allowing checking if the slave device has acknowledged or not at that point.
      ii. Issuing an ACK (or NACK) command allows the interface to continue the transaction if the slave responded.
3. When the interface is waiting for a data, SCL is automatically stretched until the data has been made available.
4. When performing a master mode write, a stop condition is automatically generated if an address or a data has not been acknowledged by the slave.

Supporting non−auto acknowledge mode is not a requirement for the DMA.

Auto Acknowledge Mode

The attributes of the Auto Acknowledge mode are:

1. The transfer is fully controlled by the data availability when transmitting bytes or by the buffer emptiness when receiving data.
2. In both master and slave modes and on read or write frames, transfers can occur without any clock stretching provided that the firmware is able to fill and empty the data buffer fast enough.
3. The buffer full/empty values can be overridden using the ACK/NACK bits.
4. Interrupts to the firmware or DMA requests are issued every time a new data can be written to the buffer when transmitting data and when new data have been received.
5. An additional LAST_DATA control bit is provided to automatically stop a transaction. When set, the following rules apply:
   a. During Read frame in master mode, the next data will be NACKed automatically and a stop condition will be generated.
   b. In data transmit master mode (i.e. write frame in master), a stop condition will be generated after the next acknowledge bit.
   c. The LAST_DATA control bit is reset when a stop condition is received.
6. In master mode write frame, a stop condition is automatically generated if an address or a data has not been acknowledged by the slave.

7. In DMA master mode the recommended steps are:
   a. Configure and start the DMA and enable DMA interrupt in the NVIC,
   b. Configure the I^2C as follows:
      i. Enabling the I^2C block
      ii. Enable master mode
      iii. Set the appropriate Pre−scale factor
      iv. Set DMA mode
      v. Enable auto ACK
      vi. Set the slave address
   c. Start the communication by sending the I2C_ADDR_STRT command.
8. In DMA slave mode the recommended steps are:
   a. Configure and start the DMA and enable DMA interrupt in the NVIC,
   b. Configure the I^2C block as follows:
      i. Enable the I^2C interface
      ii. Enable slave mode
      iii. Set DMA mode
      iv. Enable auto ACK
      v. Set the slave address
   c. By setting it to DMA mode, the I^2C block sends a request the DMA for each new TX data sample.

Interrupt and DMA Driven Data Transfers

1. The I^2C interface allows for an interrupt to be generated under the following conditions:
   a. Start or restart condition with a recognized address
   b. Acknowledgement or other data handling required
   c. Stop condition (required for only acknowledged transactions)
   d. Bus error detected
2. Stop condition interrupts can be disabled in slave mode.
3. Acknowledgement interrupts can be automated using an auto−acknowledge feature.
4. The I^2C interface can be configured to produce a DMA data handling request rather than data interrupts to support data transfers that are not controlled by the CORTEX−M0+.
5. The types of interrupts received are identifiable through the following status bits that will be available through the I^2C control register:
   a. Address byte or data byte
   b. General call address or defined slave address
   c. Note that if the defined slave address is the general call address than this will be identified as not being an access at the general call address.
   d. Acknowledgement state for the previous data byte
   e. Data buffer full (indicates that data is needed for transmission or that data needs to be read when receiving).
   f. Stop received
   g. Bus error

Bus Error

1. A BUS_ERROR is detected when the bit driven onto the serial data line is different from the data read from the serial data line.
2. As soon as a BUS_ERROR is detected, the I^2C interface stops driving both SCL and SDA lines and an interrupt is generated.
3. The BUS_ERROR bit remains asserted until the SDA line is released by the other devices on the bus.

Interface Reset

An interface reset can be issued by writing the I^2C control register. This command immediately stops any transfer (master or slave mode), releases the I^2C lines and puts the state machine in idle mode. An interface reset should be requested by the firmware whenever a bus error is detected.

I^2C Register Table

Table 36. I^2C REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
I^2C CONFIGURATION REGISTER: 0x41D00000					
CFG	[26]	0x1	R/W	ERROR_INT_EN	Enable interrupts on command/bus errors 0: Error interrupts will not be generated 1: An Error interrupt will be generated on a command error (such as a transmit command after receiving a NACK) or a bus error. (default)
	[25]	0x1	R/W	TSTART_INT_EN	Enable interrupts on transmission start 0: TSTART interrupts will not be generated 1: A buffer interrupt will be generated when the transmission starts on an active transaction (default)
	[24]	0x1	R/W	CMD_INT_EN	Enable interrupts on command waiting 0: Command interrupts will not be generated 1: A command interrupt will be generated when a new command is required for an active transaction (default)

Table 36. I²C REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
I²C CONFIGURATION REGISTER: 0x41D00000					
	[23:16]	0x00	R/W	PRESCALE	Pre−scaler used to divide SYSCLK to the correct SCL frequency when operating in I2C interface master mode. SCL is pre−scaled by (PRESCALE + 1) x 3 0x00: Pre−scale SCL from SYSCLK by 3 (default) 0x01: Pre−scale SCL from SYSCLK by 6 0x02: Pre−scale SCL from SYSCLK by 9 0x03: Pre−scale SCL from SYSCLK by 12 … 0xFF: Pre−scale SCL from SYSCLK by 768
	[14:8]	0x00	R/W	SLAVE_ADDR	Set the I²C slave address for this device
	[4]	0x0	R/W	DMA_CTL_EN	Select whether data transfer will be controlled by the CORTEX−M0+ or the DMA for I²C 0: The CORTEX−M0+ controls data transfers using I²C (default) 1: The DMA controls data transfers using I²C
	[3]	0x0	R/W	STOP_INT_EN	Configure whether stop interrupts will be generated by the I²C interface 0: Stop interrupts will not be generated (default) 1: A stop interrupt will be generated when a stop condition occurs for an active transaction
	[2]	0x0	R/W	AUTO_ACK_EN	Select whether acknowledgement is automatically generated or not 0: Require manual acknowledgement of all I²C interface transfers (default) 1: Use automatic acknowledgement for I²C interface transfers
	[1]	0x0	R/W	I2C_EN	Enable/disable the I²C 0: Disable the I²C (default) 1: Enable the I²C
	[0]	0x0	R/W	SLAVE_EN	Select whether the I²C interface will be enabled for slave mode or not 0: Disable I²C interface slave mode operation (default) 1: Enable I²C interface slave mode operation
I²C CONTROL REGISTER: 0x41D00004					
CTL	[5]	0x0	WO	RST	Reset the I²C interface 1: Reset the I²C interface. This synchronous reset will automatically clear itself, there is no need to write it back to ‘0’.
	[4]	0x0	WO	LAST_DATA	Indicate that the current data is the last byte of a data transfer 1: Indicate that the current data is the last byte of a data transfer
	[3]	0x0	WO	STOP	Issue a stop condition the I²C interface bus 1: Issue a stop condition the I²C interface bus
	[2]	0x0	WO	−	Reserved
	[1]	0x0	WO	NACK	Issue a not acknowledge on the I²C interface bus 1: Issue a not acknowledge on the I²C interface bus
	[0]	0x0	WO	ACK	Issue an acknowledge on the I²C interface bus 1: Issue an acknowledge on the I²C interface bus
I²C DATA REGISTER: 0x41D00008					
DATA	[7:0]	0x00	R/W	DATA	Single byte buffer for data transmitted and received over the I²C interface
I²C DATA (MIRROR) REGISTER: 0x41D0000C					
DATA_M	[7:0]	0x00	R/W	DATA_M	Mirror of the single byte buffer for data transmitted and received over the I²C interface
I²C MASTER ADDRESS AND START REGISTER: 0x41D00010					
ADDR_START	[7:1]	0x0	WO	ADDR	I²C address to use for the transaction
	[0]	0x0	WO	READ_WRITE	Select whether a read or a write transaction is started 0: Start a I²C write transaction 1: Start a I²C read transaction

Table 36. I²C REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
I²C STATUS REGISTER: 0x41D00014					
STS	[15]	0x0	RO	ERROR_S	Error status bit (sticky) – This status bit is automatically reset when the I2C_STATUS register is read. 0: I²C interface is not and has not been in an error state (default) 1: I²C interface is or has been in an error state
	[14]	0x0	RO	BUS_ERROR_S	Bus error status bit (sticky) – This status bit is automatically reset when the I2C_STATUS register is read. 0: I²C interface is not and has not been in the bus error state (default) 1: I²C interface is or has been in the bus error state
	[13]	0x0	RO	START_PEND	Master frame start pending status bit 0: No pending master start frame (default) 1: A master frame is pending to start (bit is set when I2C_ADDR_START is written)
	[12]	0x0	RO	MASTER_MODE	Master mode status bit 0: I²C interface is not operating in master mode (default) } 1: I²C interface is operating in master mode
	[11]	0x0	RO	DMA_REQ	Indicate if the I²C interface is currently requesting DMA data 0: The I²C interface is not requesting a DMA action (default) 1: The I²C interface is requesting a DMA action
	[10]	0x0	RO	STOP_DETECT	Indicate if an I²C stop bit has been detected 0: No stop condition has been detected on the I²C bus (default) 1: A stop condition has been detected on the I²C bus
	[9]	0x0	RO	DATA_EVENT	Indicate that I²C interface either needs data to transmit or has received data 0: No I²C data is needed or available (default) 1: I²C data is needed or is available
	[8]	0x0	RO	ERROR	Indicate if the I²C interface has detected a bus error (automatically cleared when a stop condition is detected) 0: I²C interface is not in an error state (default) 1: I²C interface is in an error state
	[7]	0x0	RO	BUS_ERROR	Indicate if the I²C interface has detected a bus error (automatically cleared when a stop condition is detected) 0: I²C interface is not in the bus error state (default) 1: I²C interface is in the bus error state
	[6]	0x0	RO	BUFFER_FULL	Indicate if the I²C data buffer is full 0: The I²C interface buffer is empty (default) 1: The I²C interface buffer is full
	[5]	0x0	RO	CLK_STRETCH	Indicate if the I²C interface is holding the clock signal 0: The I²C clock line is not being stretched (default) 1: The I²C SCL line is being held low
	[4]	0x1	RO	BUS_FREE	Indicate if the I²C interface bus is free 0: One or both of the I²C bus lines is currently 0 1: Both I²C bus lines are currently free (default)
	[3]	0x0	RO	ADDR_DATA	Indicate if the I²C data register holds an address or data byte 0: The I²C data register holds data (default) 1: The I²C data register holds an address
	[2]	0x0	RO	READ_WRITE	Indicate whether the I²C bus transfer is a read or a write 0: The current I²C transfer is a write (default) 1: The current I²C transfer is a read
	[1]	0x0	RO	GEN_CALL	Indicate whether the I²C bus transfer is using the general call address or another address 0: The address used for the current I²C transfer is not the general call address (default) 1: The address used for the current I²C transfer is the general call address
	[0]	0x0	RO	ACK	Indicate whether an acknowledge or a not acknowledge has been received 0: Indicate that the last I²C byte was acknowledged (default) 1: Indicate that the last I²C byte was not acknowledged

Table 36. I²C REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
I²C INTERRUPT STATUS REGISTER: 0x41D00018 Reading the Interrupt Status Register does not change the contents of the register. Writing the register will clear any bits that are written with a '1', rather than a pure write of the data. Writing to the register will not set any bits to a 1.					
INT_STS	[3]	0x0	R/W	ERR_DET	An interrupt has been sent due to a command error (such as sending new data after a NACK), or a bus error. Cleared by writing a 1.
	[2]	0x0	R/W	STOP_DET	An interrupt has been sent because a stop bit has been received. Cleared by writing a 1.
	[1]	0x0	R/W	START_SYNC	An interrupt has been sent because a transmission has started. Cleared by writing a 1.
	[0]	0x0	R/W	CMD_READY	An interrupt has been sent because the I2C was awaiting a command. Cleared by writing a 1.

TIMERS

There are 7 functional timers available for various functions including the built−in Cortex−M0+ SYSTICK timer. The timers are summarized as follows:

- 1 – 24 bit System Tick Timer (SYSTICK)
- 1 – 32−bit wakeup timer used for scheduling wakeup events from low power modes
- 1 – 32−bit ticker timer used for non−wakeup event scheduling
- 3 – 16−bit general purpose, up−down, programmable step timers used for:
- USART baud rate generation (USART0, 1)
- PWM generation (CPWM0, 1, 2, 3)
- 1 – 24−bit watchdog timer used to reset the part during run−away code

The software packages related to radio application included in the Semiconductor Integrated Development Environment (IDE) are using wakeup timer, ticker timer and general purpose timer 0.

SysTick Timer (SysTICK)

The SYSTICK timer is used for scheduling events on regular intervals such as might be done in a real−time operating system or a scheduler. See section 4.4 of the Cortex−M0+UserGuideReferenceMaterial document for usage model of the SYSTICK timer. Multiple clock sources are available for this timer. See clock control register for these controls. Asynchronous clock sources are synchronized to SCLK to create the STCLKEN register. Because of the multiple clock sources and frequencies it was not possible to provide a calibration hint to easily map the time requirements to a number of cycles. Software will need to be aware of the given clock source and frequency to set the ticker reload value for a particular time period. Because it runs off of SCLK the Systick timer will continue to run in SLEEP mode but will stall in lower power modes. The separate ticker timer can be used in a similar fashion.

SysTick Register Table

Table 37. SYSTICK REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
SysTick CONTROL AND STATUS REGISTER: 0xE000_E010					
CTL	[31:17, [15:3]	N/A	N/A		Reserved, don't use.
	[16]	0	RO	COUNTFLAG	Goes high on transition from 1 to 0. Cleared on read
	[2]	0	R/W	CLKSOURCE	0 = External reference clock, 1 = Processor clock
	[1]	0	R/W	TICKINT	0 = Interrupt disabled, 1 = Interrupt enabled
	[0]	0	R/W	ENABLE	0 = Timer disabled, 1 = Timer enabled
SysTick RELOAD VALUE REGISTER: 0xE000_E014					
LOAD	[23:0]	0x000000	R/W	LOAD	The value loaded in the timer after it decrements to 0
SysTick CURRENT VALUE REGISTER: 0xE000_E018					
CNT	[23:0]	0x000000	RO	CNT	The current value of the timer
SysTick CALIBRATION VALUE REGISTER: 0xE000_E01C					
CALIB	[31]	0	RO	NOREF	0 – External reference clock is available
	[30]	1	RO	SKEW	1 – Because of multiple clock sources and frequencies a valid TENMS number is not available
	[29:24]	0	RO	–	Reserved
	[23:0]	0	RO	TENMS	0 – No calibration value provided

Wakeup Timer (WUT)

The main purpose of the wakeup timer is to facilitate scheduled exit from low power modes. It can also be used for general purpose event timing. The wakeup timing resolution depends on the programmable clock source and pre−scale ratios. Typically the slower (low power) clock sources are used to minimize power. The wakeup timer enables wakeup capability in all power modes except shutdown mode.

The wakeup timer features a 32−bit free running increment counter and a 32−bit event register. When the counter increment matches the event register, the match status bit is set and an interrupt (if enabled) is issued. The match status bit and corresponding interrupt is cleared by writing a '1' to the status bit.

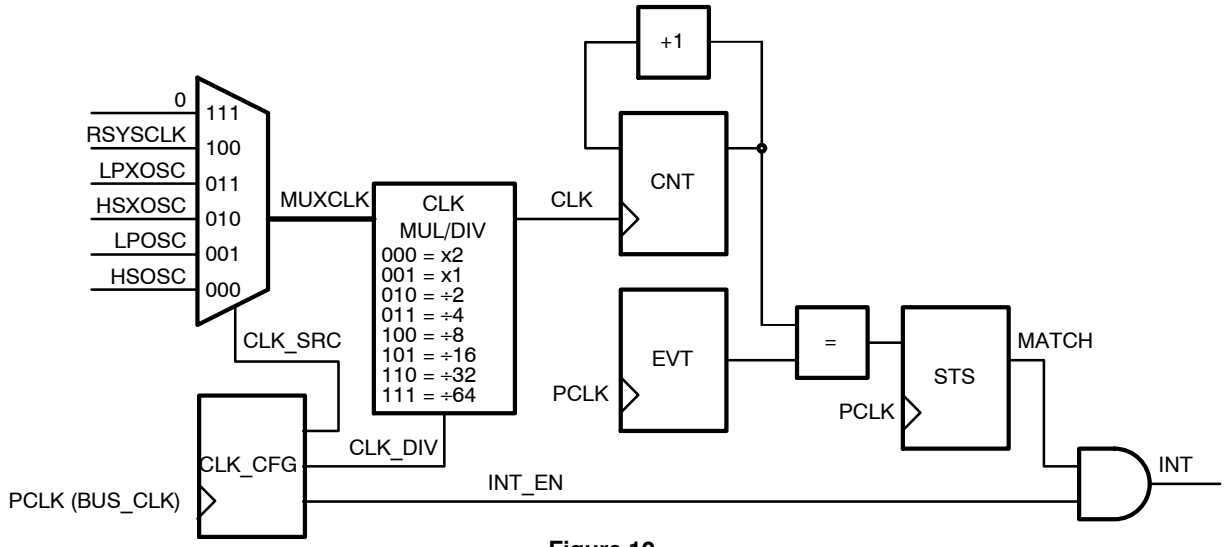


Figure 10.

Typical use model of the wakeup timer is as follows:

1. Enable the wakeup timer peripheral clock and the desired wakeup timer source clock
2. Configure the wakeup timer clock source and clock pre−scale value for the desired timing resolution
3. Read the current value of the wakeup timer
4. Add the desired wait time increment value to the current wakeup timer value and write the value to the event register (WTEVT = WTCNT + t_{wakeup} x f_{wakeup})
5. Enable wakeup timer interrupt and enter low power mode

Periodic events can be scheduled with the wakeup timer by repeating steps 3 − 5. Multiple event scheduling can be accomplished with a queuing approach.

Wakeup Timer (WUT) Register Table

Table 38. WAKEUP TIMER (WUT) REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
WAKEUP TIMER CONFIGURATION REGISTER: 0x40100000 The wakeup timer configuration register controls event scheduling resolution through clock source and clock pre−scale settings. It is also used to configure interrupt operation.					
CFG	[8]	0	R/W	INT_EN	Wakeup Timer Interrupt Enable
	[5:3]	001	R/W	CLK_DIV	Wakeup Timer Clock Divider 000: x2 001: x1 010: ÷2 011: ÷4 100: ÷8 101: ÷16 110: ÷32 111: ÷64
	[2:0]	111	R/W	CLK_SRC	Wakeup Timer Clock Source 000: HSOSC 001: LPOSC 010: XOSC 011: LPXOSC 100: EXTCLK 101: invalid 110: invalid 111: Off

Table 38. WAKEUP TIMER (WUT) REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
WAKEUP TIMER STATUS REGISTER: 0x40100004 The status register indicates if the wakeup timers has matched the event register. It is set when the counter value matches the event register and remains set until cleared by writing a '1' to the status bit. The match flag will persists despite clear operation if the match condition persists. Thus the recommended sequence of events upon a wakeup timer match are: 1. Read the STS register to determine event match 2. Update the CNT register 3. Write the STS register to clear the match bit					
STS	[0]	0	R/W	STS	Wakeup Timer matched with event; write 1 to clear
WAKEUP TIMER COUNTER REGISTER: 0x40100008 This register contains the wakeup counter value. It is read as a current time value to calculate new event timing.					
CNT	[31:0]	0x00000000	RO	CNT	Wakeup Counter
WAKEUP TIMER EVENT MATCH REGISTER: 0x4010000C The wakeup timer contains one event register. The event register is continuously compared to the counter register, and match events are generated if a match is detected.					
EVT	[31:0]	0x00000000	R/W	EVT	Wakeup Timer Event Register

TICK Timer (TICK)

The TICK timer is targeted for general purpose event scheduling. Like the SYSTICK timer it is meant to support basic scheduler or RTOS functions during run mode. Unlike SYSTICK it can also be used in sleep mode. The TICK timing resolution depends on the programmable clock source and pre−scale ratios. Typically the high speed clock sources are used for tighter timing resolution.

The TICK timer features a 32−bit free running increment counter and a 32−bit event register. When the counter increment matches the event register the match status bit is set and an interrupt (if enabled) is issued. The match status bit and corresponding interrupt is cleared by writing a '1' to the status bit.

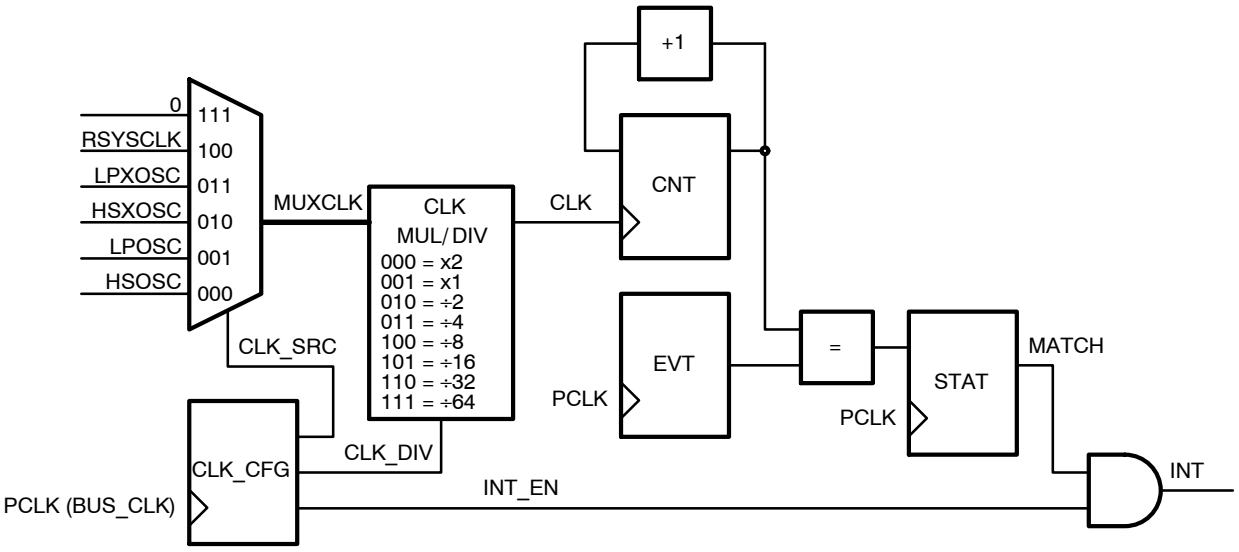


Figure 11.

TICK Timer Register Table

Table 39. TICK TIMER REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
TICK TIMER CONFIGURATION REGISTER: 0x41000000 The TICK timer configuration register controls event scheduling resolution through clock source and clock pre−scale settings. It is also used to configure interrupt operation.					
CFG	[8]	0	R/W	INT_EN	TICK Timer Interrupt Enable
	[5:3]	001	R/W	CLK_DIV	TICK Timer Clock Divider 000: x2 001: x1 010: ÷2 011: ÷4 100: ÷8 101: ÷16 110: ÷32 111: ÷64
	[2:0]	111	R/W	CLK_SRC	TICK Timer Clock Source 000: HSOSC 001: LPOSC 010: HSXOSC 011: LPXOSC 100: EXTCLK 101: invalid 110: invalid 111: Off
TICK TIMER STATUS REGISTER: 0x41000004 The status register indicates if the ticker timer has matched the event register. It is set when the counter value matches the event register and remains set until cleared by writing a '1' to the status bit. The match flag will persists despite clear operation if the match condition persists. Thus the recommended sequence of events upon a wakeup timer match are: 1. Read the STS register to determine event match 2. Update the CNT register 3. Write the STS register to clear the match bit					
STS	[0]	0	RO	STS	TICK Timer Event matched with Counter; cleared on read
TICK TIMER COUNTER REGISTER: 0x41000008 This register contains the TICK counter value. It is read as a current time value to calculate new event timing.					
CNT	[31:0]	0x00000000	RO	CNT	TICK Counter
TICK TIMER EVENT MATCH REGISTER: 0x4100000C The TICK timer contains one event register. The event register is continuously compared to the counter register, and match events are generated if a match is detected.					
EVT	[31:0]	0x00000000	R/W	EVT	TICK Timer Event Register

16−bit General Purpose Timers (TIM0, TIM1, or TIM3)

The 16−bit general purpose timers (TIM0/1/2) have the following available functions:

- Count up, count down, count up/down
- Event timer
- Event counter (pulse counter)
- Event multichannel capture/compare using the CPWM modules
- Pulse Width Modulation using the CPWM modules
- Increment by N capability
- Sigma−Delta DAC modulation mode
- 2X frequency to enable PWM up to 80 MHz Clock
- AXM0F343−64 MCU Asynchronous reset for digital power applications

General Purpose Timer Block Diagram

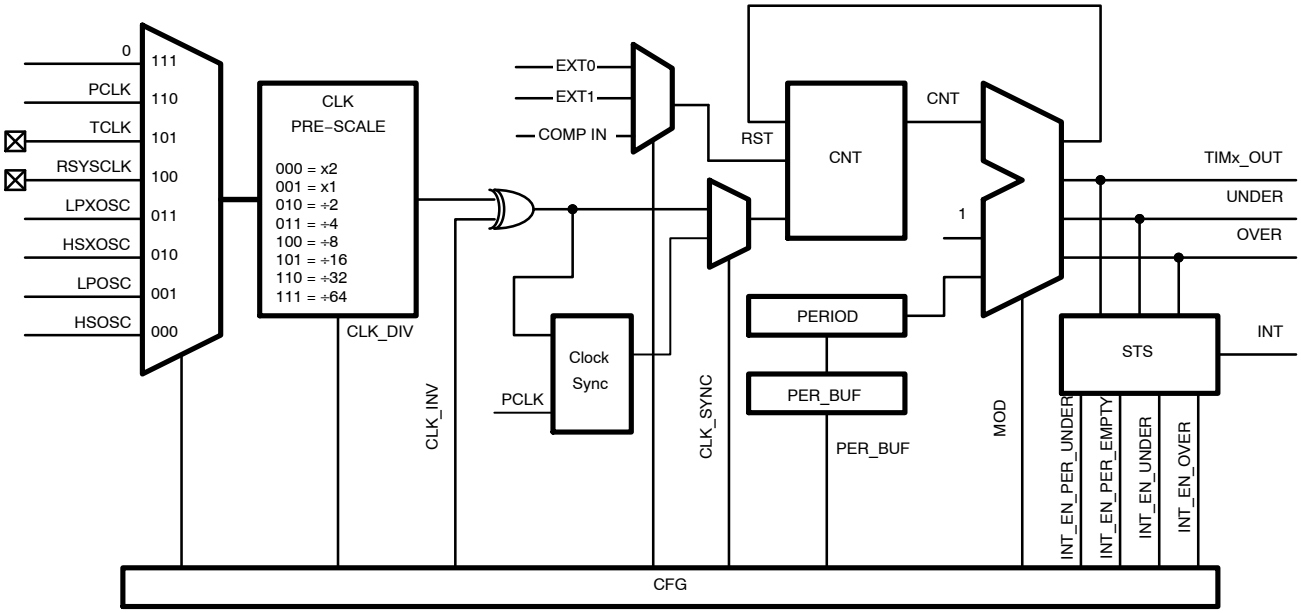


Figure 12. General Purpose Timer Block Diagram

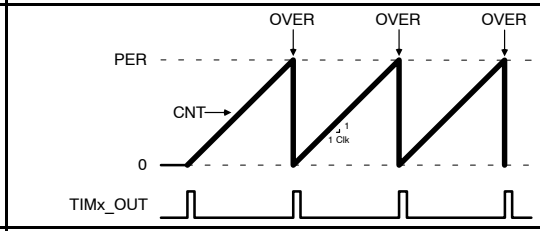
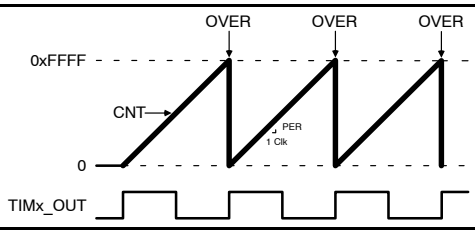
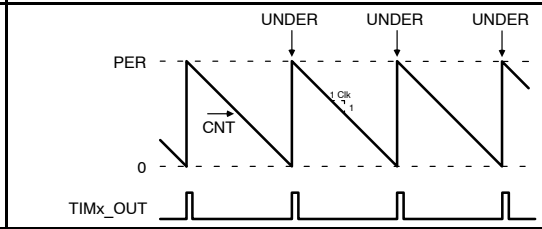
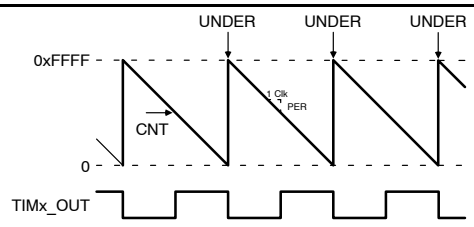
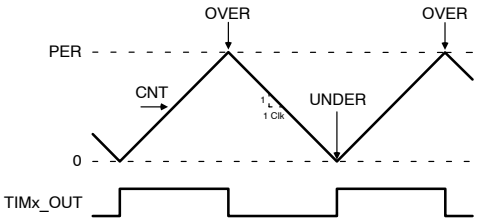
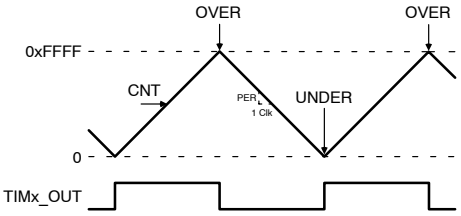
Timer Modes

The general purpose timers support count up (sawtooth up), count down (sawtooth down), and count up/down (triangle) modes. Up count and down count modes help align events to the starting/ending edge while up/down counting helps align events around the center of the counting sequence or the peak of the triangle. This is valuable for PWM alignments depending on the need to edge or center align multiple PWM.

Counting Mode

The timers also support both divide mode and multiply mode increment/decrement. In divide mode the timer counter increment/decrement is a fixed ‘1’ value and the rollover value is programmable. This is useful in creating a fixed period time of exactly Period (PER) cycles (period is 2X for triangle mode). In multiply mode the overflow period is fixed at the max value (0xFFFF) while the increment/decrement step size is variable. This allows more resolution in the average period of the timer but can result in variable timing from cycle to cycle if the step size is not an integer division of the max value. This mode is particularly useful in baud rate generation.

Table 40.

	Div	Multiply
Sawtooth Up		
Sawtooth Down		
Triangle		

- Sawtooth up (count up)
  - Divide Mode: Timer increments the count up, on the rising edge of timer clock until the value in the Period register is reached. The Timer flags an OVER Interrupt. The Count is reset to 0 and starts counting at the next rising edge of the timer clock. TXOUT:
  - Multiply Mode: Timer increments the count up by the value of the Period register on the rising edge of the timer clock until the value in the count register is full 0xFFFF. The Timer flags an OVER Interrupt. The Count is reset to 0 and starts counting at the next rising edge of the timer clock. TXOUT:
- Sawtooth down (count down)
  - Divide Mode: Timer decrements the count on the rising edge of timer clock until the value reaches 0. The Timer flags an UNDER Interrupt. The Count is reset to the value of the Period Register and starts counting at the next rising edge of the timer clock. TXOUT:
  - Multiply Mode: The Count Register starts at full 0xFFFF then the Timer decrements the count down by the value of the Period register on the rising edge of the timer clock until the 0 is reached in the Count Register. The Timer flags an UNDER Interrupt. The Count is reset to the value of the Period Register and starts counting at the next rising edge of the timer clock. TXOUT:
- Triangle
  - Divide Mode: The Count Register starts at 0. Timer increments the count up, on the rising edge of timer clock until the value in the Period register is reached. The Timer flags an OVER Interrupt. Timer decrements the count on the rising edge of timer clock until the value reaches 0. The Timer flags an UNDER Interrupt. TXOUT: Rise on an OVER event and Fall on and UNER event.
  - Multiply Mode: The Count Register starts at 0. The Timer decrements the count down by the value of the Period register on the rising edge of the timer clock until the 0 is reached in the Count Register. The Timer flags an OVER Interrupt. Timer increments the count up by the value of the Period register on the rising edge of the timer clock until the value in the count register is full 0xFFFF. The Timer flags an OVER Interrupt. TXOUT: Rise on an OVER event and Fall on and UNER event.

ΣΔ Mode

In ΣΔ mode, the timer implements a second order modulator with STF z^{-1} and NTF $1 - 2z^{-1} + z^{-2}$. The transfer functions are depicted below.

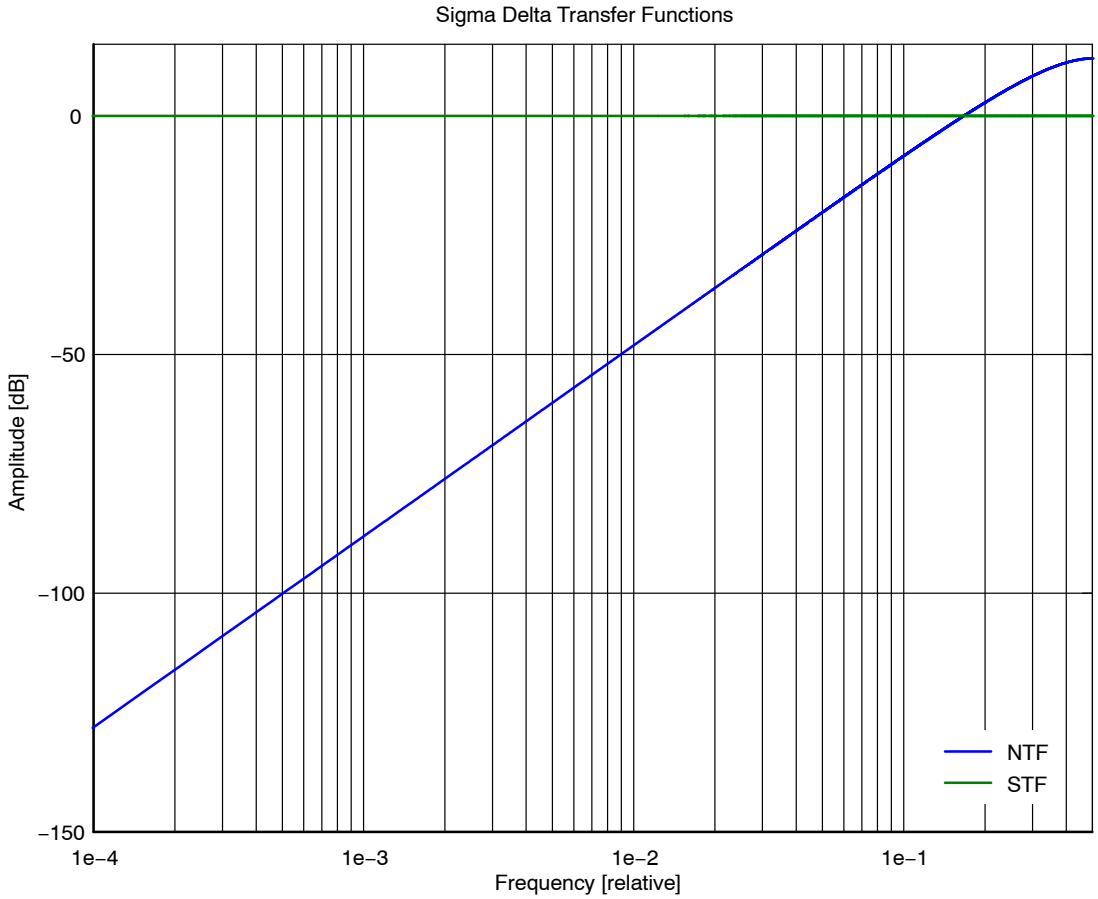


Figure 13.

Values to be converted to an analog voltage must be loaded into the TPERIOD register. The register is treated as a signed value. Its scaling is as follows:

Table 41.

TPERIOD	Voltage
0x8000 / −32768	$^1/_4$·VDD
0x0000 / 0	$^1/_2$·VDD
0x7FFF / 32767	$^3/_4$·VDD

General Purpose Timer (TIM0, TIM1, and TIM2) Register Table

Table 42. GENERAL PURPOSE TIMER (TIM0, TIM1, AND TIM2) REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
TIM0 COUNTER REGISTER: 0x41100000 TIM1 COUNTER REGISTER: 0x41200000 TIM2 COUNTER REGISTER: 0x41300000					
CNT	[15:0]	0x0000	R/W	CNT	TCOUNT register (counter register, unsigned)
TIM0 PERIOD REGISTER: 0x41100004 TIM1 PERIOD REGISTER: 0x41200004 TIM2 PERIOD REGISTER: 0x41300004					
PER	[15:0]	0x0000	R/W	PER	TPERIOD register (counter period register, unsigned)
TIM0 CONFIGURATION REGISTER: 0x41100008 TIM1 CONFIGURATION REGISTER: 0x41200008 TIM2 CONFIGURATION REGISTER: 0x41300008					
CFG	[24]	0x0	R/W	ASYNC_RESET_EN AXM0F343−64 MCU Reserved AXM0F343−256 MCU	Write 1 to enable asynchronous reset function
	[23:22]	0x0	R/W	RESET_TRIGGER AXM0F343−64 MCU Reserved AXM0F343−256 MCU	Selects active level of trigger 0x0 = Low level 0x1 = High Level 0x2 = Negative edge triggered 0x3 = Positive edge triggered
	[21:20]	0x0	R/W	RESET_SEL AXM0F343−64 MCU Reserved AXM0F343−256 MCU	Asynchronous reset signal source selection 0x0 = Not Available 0x1 = Not Available 0x2 = Comparator input[1] 0x3 = Comparator input[0]
	[19]	0x0	R/W	PER_UNDER_INT_EN	TPERIOD underrun interrupt enable
	[18]	0x0	R/W	PER_EMPTY_INT_EN	TPERIOD empty interrupt enable
	[17]	0x0	R/W	UNDER _INT_EN	Interrupt on TUNDER enable
	[16]	0x0	R/W	OVER _INT_EN	Interrupt on TOVER enable
	[15]	0x0	R/W	CLK_SYNC	Timer clock synchronization to system clock; if set, the timer input signal TCLK is synchronized to PCLK. If cleared, TCLK clocks the timer asynchronously if selected as clock source
	[14]	0x0	R/W	CLK_INV	Timer Clock Invert. If set, the selected clock is inverted before being fed into the timer core.
	[13:11]	0x1	R/W	CLK_DIV	Clock Divider 0x0 = ×2 0x1 = ×1 0x2 = ÷2 0x3 = ÷4 0x4 = ÷8 0x5 = ÷16 0x6 = ÷32 0x7 = ÷64
	[10:8]	0x7	R/W	CLK_SRC	Clock Source 0x0 = HSOSC 0x1 = LPOSC 0x2 = HSXOSC 0x3 = LPXOSC 0x4 = EXTCLK 0x5 = TIMxCLK 0x6 = System Clock 0x7 = Off

Table 42. GENERAL PURPOSE TIMER (TIM0, TIM1, AND TIM2) REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
	[5:4]	0x0	R/W	PER_BUF	If period buffering is enabled, the architectural period register is backed by a shadow period register that drives the timer core. That shadow register is updated on the events configured below. Period Buffering in Divide and Multiply Modes: 0x0 = No Double Buffering 0x1 = Internal Buffer updated on OVER 0x2 = Internal Buffer updated on UNDER 0x3 = Internal Buffer updated on OVER and UNDER Period Buffering in ΣΔ Mode: 0x0 = No Double Buffering 0x1 = Internal Buffer updated on TIM0OUT 0x2 = Internal Buffer updated on TIM1OUT 0x3 = Internal Buffer updated on TIM2OUT
	[3:0]	0x0	R/W	MOD	Operating Mode 0x0 = Off 0x1 = ΣΔ 0x2 = Divide Sawtooth Up 0x3 = Multiply Sawtooth Up 0x4 = Divide Sawtooth Down 0x5 = Multiply Sawtooth Down 0x6 = Divide Triangle 0x7 = Multiply Triangle

TIM0 INTERRUPT STATUS REGISTER: 0x4110000C
TIM1 INTERRUPT STATUS REGISTER: 0x4120000C
TIM2 INTERRUPT STATUS REGISTER: 0x4130000C

INT_STS	[5]	0x0	R/W	PER_UNDER	Period underrun interrupt request (write 1 to clear) The interrupt only occurs when timer is configured with PER_BUF having a non−zero value in DIV/MULTIPLY mode
	[4]	0x0	RO	PER_EMPTY	Period empty interrupt request (cleared by writing a new value to the TPERIOD register)
	[3]	0x0	R/W	UNDER_MISSED	Underflow interrupt missed (write 1 to clear)
	[2]	0x1	R/W	OVER_MISSED	Overflow interrupt missed (write 1 to clear)
	[1]	0x0	R/W	UNDER	Underflow interrupt request (write 1 to clear)
	[0]	0x0	R/W	OVER	Overflow interrupt request (write 1 to clear)

Watchdog Timer (WDOG)

The watchdog timer applies a reset to the system in the event of a software failure, providing a way to recover from software crashes. The watchdog timer is disabled by default and must be enabled through software.

The watchdog module is based on a 32−bit down−counter that is initialized from a reload register, WDOGLOAD. The watchdog module generates a regular interrupt, WDOGINT, depending on a programmed value.

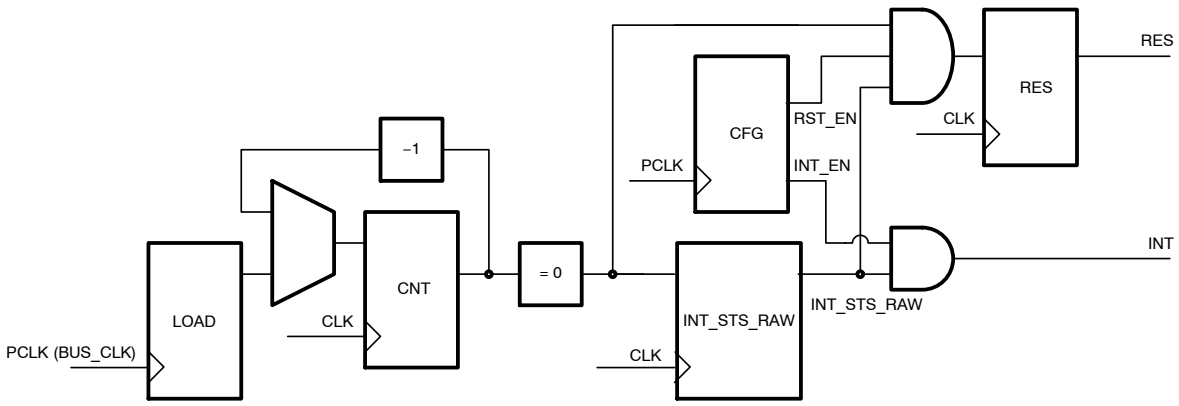


Figure 14.

The counter decrements by one on each positive clock edge of WDOGCLK. The watchdog monitors the interrupt and asserts a reset request WDOGRES signal when the counter reaches 0, and the counter is stopped. On the next enabled WDOGCLK clock edge, the counter is reloaded from the LOAD register and the countdown sequence continues. If the interrupt is not cleared by the time the counter next reaches 0, the watchdog module reasserts the reset signal.

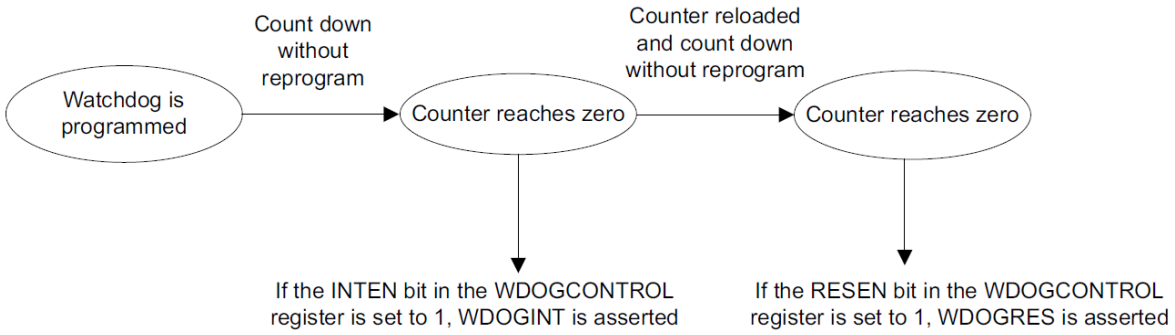


Figure 15.

The watchdog is protected with a lock mechanism to prevent rogue software from disabling the watchdog functionality. A special value has to be written to the lock register to access watchdog control.

The watchdog includes an integration test mode that allows you to test the watchdog interrupt and/or reset via software without waiting for the watchdog timer to expire.

Watchdog Register Table Summary (Base Address 0x40F00000)

Table 43.

Function	Bits	Default	Type	Symbol	Description
WATCHDOG LOAD REGISTER: 0x40F00000 The LOAD Register is a 32−bit register containing the value from which the counter is to decrement. When this register is written to, the count is immediately restarted from the new value. The minimum valid value for LOAD is 1.					
LOAD	[31:0]	0xFFFF_FFFF	R/W	LOAD	Watchdog Load Register
WATCHDOG VALUE REGISTER: 0x40F00004 The WDOGVALUE Register gives the current value of the decrementing counter.					
VAL	[31:0]	0xFFFF_FFFF	RO	VAL	Watchdog Value Register
WATCHDOG CONTROL REGISTER: 0x40F00008 The WDOGCONTROL Register is a read/write register that enables the software to control the watchdog unit.					
CFG	[1]	0x0	R/W	RST_EN	Enable Watchdog reset output, WDOGRES. Acts as a mask for the reset output. Set HIGH to enable the reset, and LOW to disable the reset.
	[0]	0x0	R/W	INT_EN	Enable the interrupt event, WDOGINT. Set High to enable the counter and the interrupt, and set LOW to disable the counter and interrupt. Reloads the counter from the value in WDOGLOAD when the interrupt is enabled, and was previously disabled.
WATCHDOG RAW INTERRUPT STATUS REGISTER: 0x40F00010 The INT_STS_RAW Register is read−only. It indicates the raw interrupt status from the counter. This value is ANDed with the interrupt enable bit from the control register to create the masked interrupt that is passed to the interrupt output pin.					
INT_STS_RAW	[0]	0x0	RO	INT_STS_RAW	Raw interrupt status from counter
WATCHDOG INTERRUPT STATUS REGISTER: 0x40F00014 The INT_STS Register is read to check the interrupt status. It indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the INT_EN bit from the control register, and is the same value that is passed to the interrupt output pin. This register is written to clear the watchdog interrupt and to restart the watchdog counter. Bit 0 of the write must be one to perform the clear/restart.					
INT_STS	[0]	0x0	R/W	INT_STS	Read: Enabled interrupt status from counter Write: Clear existing interrupt and restart watchdog count to load value if enabled.

Table 43. (continued)

Function	Bits	Default	Type	Symbol	Description
WATCHDOG LOCK REGISTER (Offset 0xC00) The LOCK Register is write−only. Use of this register causes write−access to all other registers to be disabled. This is to prevent rogue software from disabling the watchdog functionality. Writing a value of 0x1ACCE551 enables write access to all other registers. Writing any other value disables write accesses. A read from this register returns bit[0] only: • '0' indicates that write access is enabled, not locked. • '1' indicates that write access is disabled, locked.					
LOCK	[31:0]	−	WO	EN_WRITES	Enable write access to all other registers by writing 0x1ACCE551. Disable write access by writing any other value.
	[0]	0x0	RO	WR_STS	Register write lock status: 0 = write access to all other registers is enabled 1 = write access to all other registers is disabled

Capture/PWM (CPWM0/1/2/3)

There are 4 16−bit compare/capture/PWM units (CPWM). Each can be independently used to

- Capture the value of the timer at various internal and external events
- Compare the timer value to a specific value
- Generate pulse width modulated (PWM) outputs
- AXM0F343−64 MCU Shutdown Operation
- AXM0F343−64 MCU Variable Frequency Operation

Each CPWM unit can be independently tied to any of the 4 16−bit general purpose timers. Multiple CPWM units can be tied to the same timer. Alignment of multiple PWM signals can be achieved by connecting multiple CPWM to the same timer. If the timer is configured to up or down count mode you get edge aligned PWM signals. If the timer is configured to up/down mode you get center aligned PWM signals. Each CPWM unit can generate differential PWM signals with a programmable dead time in between transitions. This is useful in generation of non−overlapping (break before make) control signals.

PWM Operation

The PWM output value must be stored in the DATA [16:0] register. It is treated as a signed register. Its value range is −1…65535. −1 outputs a constant zero on PWM, and 65535 outputs a constant one.

The PWM unit compares the DATA value to the zero extended counter register of the associated timer. Any of the divide/multiply saw tooth/triangle modes can be used, depending on the alignment requirements if multiple PWM signals should be generated from the same timer. If the zero extended timer counter value is less or equal to the (signed) DATA value, a one is output on PWMH and the inverse on PWML, provided no gap and inversion is programmed.

In order to provide guaranteed non−overlapping control signals (for example for H bridge control), the onset of the rising edge of both PWMH and PWML may be delayed by up to 15 timer clock cycles. This is programmed into the PWMGAP field.

Finally, both the PWMH and PWML signals may be inverted independently using the SRC field.

The figure below illustrates the PWM waveforms from an up count saw tooth enabled timer. Both PWMH and PWML are shown non−inverted. The first two waveforms show the PWM pair with no dead time gap. The second two waveforms show with the gap. Notice that all PWM based on this type of timer will be edge aligned.

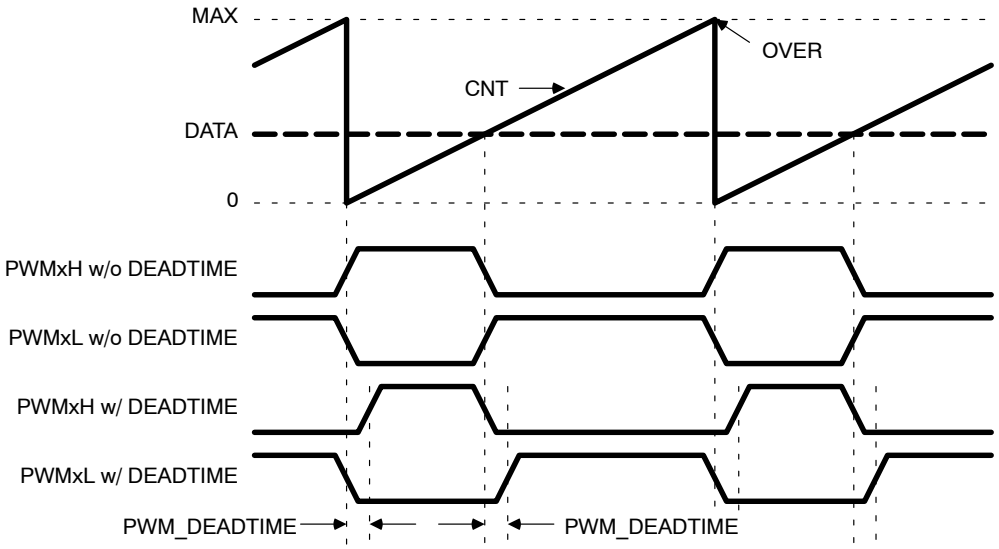


Figure 16. Differential PWM with Saw Tooth Timer 1

The next figure shows the same PWM differential signals (with and without gap) with a up/down (triangle) enabled timer. Notice that all PWM signals based on this type of timer will be center aligned.

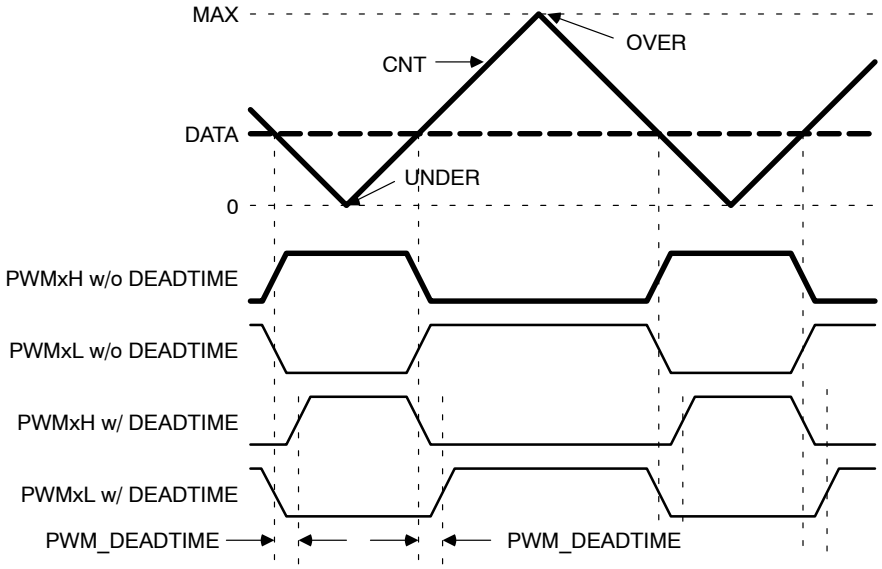


Figure 17. Differential PWM with Triangle Timer 2

Capture Operation

In capture mode, upon occurrence of an event configured in EDGE on the capture input, the value of the timer counter register TCNT of the timer selected in CPSOURCE is copied to the lower 16 bits of the DATA register. Bit 16 of the DATA register indicates the direction of the edge that was captured.

AXM0F343−64 MCU Shutdown Operation

Permanent shutdown is intended to be used for overcurrent protection and will stop any further PWM outputs until released explicitly by clearing the PWM_SHUTDOWN bit in the STS register.

An automatic restart shutdown is used for Current or Voltage limiting such as a constant current supply. In this case the current can be limited at each PWM cycle and will automatically restart at the next PWM boundary such that the PWM duty cycle can be held to a maximum average width, thus limiting the average current available at the output of a regulator.

Shutdown signal can come from external input or an internal comparator and has flexibility allowing any edge or any level to trigger shutdown operation.

Shutdown can be masked off to eliminate termination during the large currents that normally occur in an inductive circuit drive. Delay can be programmed up to 255 clock cycles.

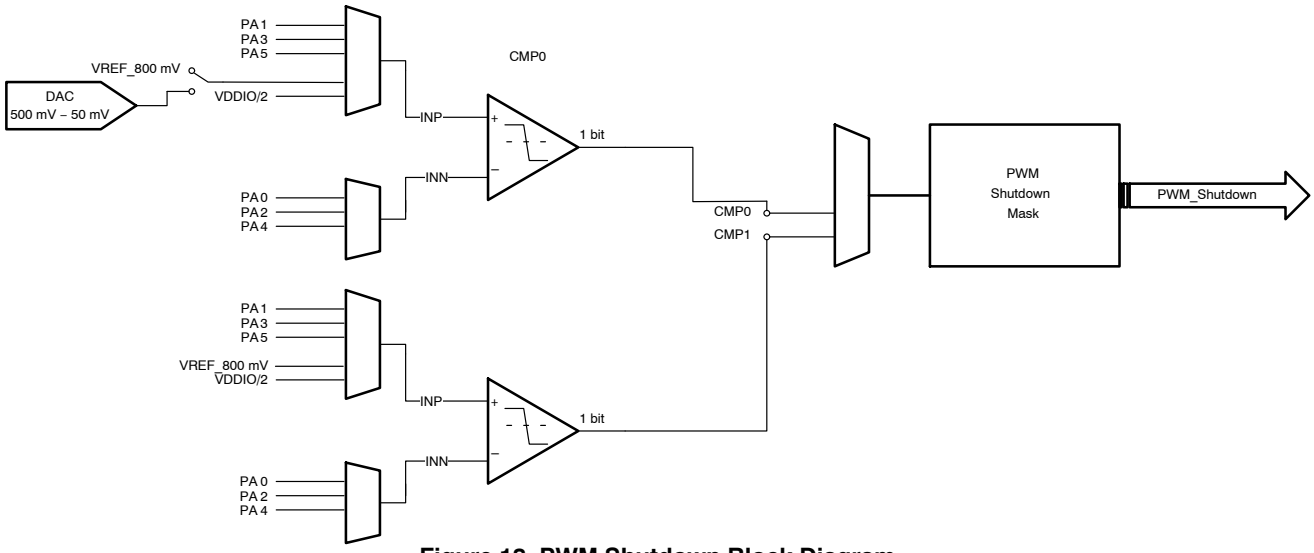


Figure 18. PWM Shutdown Block Diagram

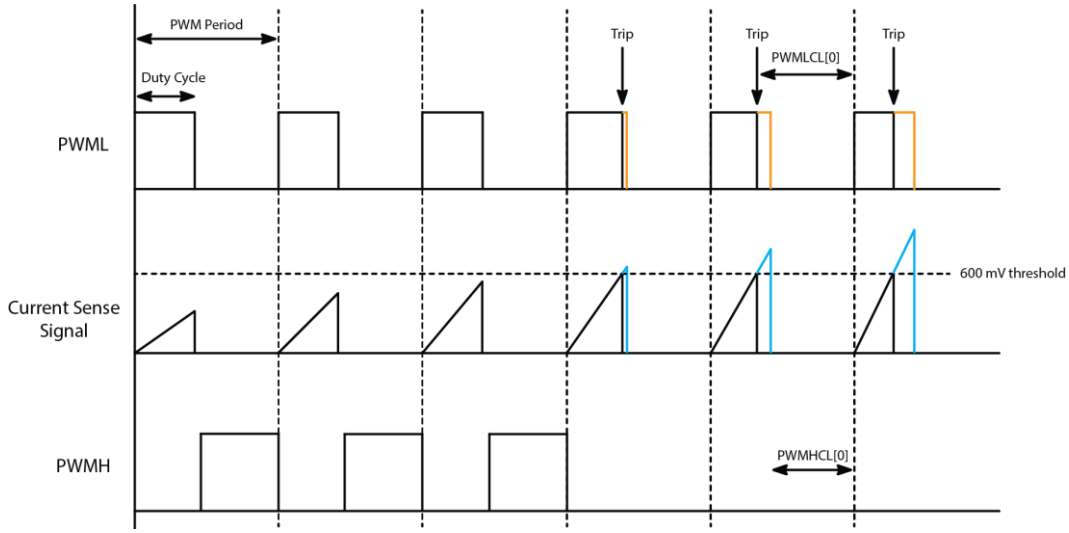


Figure 19. PWM Shutdown Diagram

AXM0F343−64 MCU Variable Frequency Operation

Variable frequency PWM can be accomplished by using the external timer reset functionality. AXM0F343−64 MCU provides option to restart PWM cycle based on external condition. This allows the PWM cycle to start based on an external trigger such as a compactor trip. The reset trigger can be selected from comparator output. This is defined by the RESET_SEL bits in the Timer CFG register. All configuration selections are described in timer block specification.

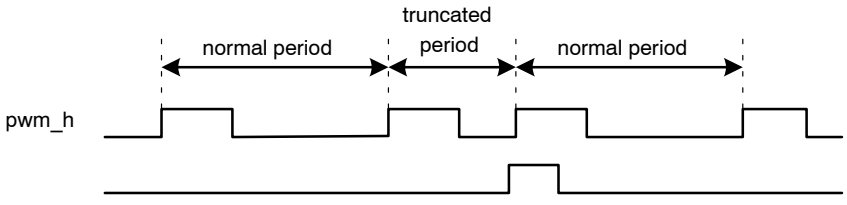


Figure 20. PWM Variable Frequency Diagram

Compare/Capture/PWM Registers Table

Table 44. COMPARE/CAPTURE/PWM REGISTERS TABLE

Function	Bits	Default	Type	Symbol	Description
CPWM0 CONFIGURATION REGISTER: 0x41400000 **CPWM1 CONFIGURATION REGISTER: 0x41500000** **CPWM2 CONFIGURATION REGISTER: 0x41600000** **CPWM3 CONFIGURATION REGISTER: 0x41700000**					
CFG	[23:16]	0x0	R/W	PWM_DEADTIME	Gap in timer clock cycles between PWMxH and PWMxL high 0x00 = no gap 0x01 = 1 cycle gap 0x02 = 2 cycles gap … … 0xFF = 255 cycles gap
	[11]	0x0	R/W	PWM_UNDER_INT_EN	PWM underrun interrupt enable
	[10]	0x0	R/W	PWM_EMPTY_INT_EN	PWM empty interrupt enable
	[9]	0x0	R/W	CAPT_OVER_INT_EN	Interrupt on Capture Overflow enable
	[8]	0x0	R/W	CAPT_FULL_INT_EN	Interrupt on Capture Full enable
	[7:5]	0x0	R/W	SRC_POL	Capture mode: 0x0 = CAPTURE Pin 0x1 = Invalid 0x2 = TIM0OUT 0x3 = TIM0CLK 0x4 = TIM1OUT 0x5 = TIM1CLK 0x6 = TIM2OUT 0x7 = TIM2CLK PWM mode: XX0 Pin PWMxH non−inverted XX1 Pin PWMxH inverted X0X Pin PWMxL non−inverted X1X Pin PWMxL inverted CAPTURE Pin: XBAR Capture0/Capture1/Capture2/ Capture3
	[4]	0x0	R/W	MOD	0 PWM mode 1 Capture mode
	[3:2]	0x0	R/W	EDGE_HOLDING	Capture: 0x0 = Capture Disabled 0x1 = Capture Rising Edge 0x2 = Capture Falling Edge 0x3 = Capture Both Edges PWM: 0x0 = Update PWM holding register immediately after writing DATA 0x1 = Update PWM holding register on TxUNDER 0x2 = Update PWM holding register on TxOVER 0x3 = Update PWM holding register on TxOVER and TxUNDER
	[1:0]	0x0	R/W	TIM_SEL	Operating Mode 0x0 = Off 0x1 = TIM0 0x2 = TIM1 0x3 = TIM2

Table 44. COMPARE/CAPTURE/PWM REGISTERS TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
CPWM0 STATUS REGISTER: 0x41400004 **CPWM1 STATUS REGISTER: 0x41500004** **CPWM2 STATUS REGISTER: 0x41600004** **CPWM3 STATUS REGISTER: 0x41700004**					
STS	[4]	0x0	RW	PWM_SHUTDOWN	PWM shutdown interrupt request (write 1 to clear)
	[3]	0x0	R/W	PWM_UNDER	PWM underrun interrupt request (write 1 to clear)
	[2]	0x1	RO	PWM_EMPTY	PWM empty interrupt request (cleared by writing to the CPDATA register)
	[1]	0x0	R/W	CAPT_UNDER	Capture overrun interrupt request (write 1 to clear)
	[0]	0x0	RO	CAPT_FULL	Capture full interrupt request (cleared by reading from the CPDATA register)
CPWM0 DATA REGISTER: 0x41400008 **CPWM1DATA REGISTER: 0x41500008** **CPWM2 DATA REGISTER: 0x41600008** **CPWM3 DATA REGISTER: 0x41700008**					
CDATA	[16]	0x0	R/W	EDGE_SIGN	Capture mode: 0 Rising Edge 1 Falling Edge PWM mode: PWM threshold sign bit
	[15:0]	0x0000	R/W	VAL_THRESHOLD	Capture mode: Capture Data PWM mode: PWM threshold (twos complement signed, sign bit CPDSIGN)
AXM0F343−64 MCU CPWM0 SHUTDOWN CONFIGURATION REGISTER: 0x4140000C **AXM0F343−64 MCU CPWM1 SHUTDOWN CONFIGURATION REGISTER: 0x4150000C** **AXM0F343−64 MCU CPWM2 SHUTDOWN CONFIGURATION REGISTER: 0x4160000C** **AXM0F343−64 MCU CPWM3 SHUTDOWN CONFIGURATION REGISTER: 0x4170000C**					
AXM0F343−64 MCU SHUTDOWN CFG	[15:8]	0x0	R/W	PWM_SHUTDOWN_MASK	Define number of clock cycle in starting phase of PWM to mask shutdown input 0x00 no mask 0x01 1 cycle mask 0x02 2 cycle mask −−− −−−− 0xFF 255 cycle mask
	[7:6]	0x0	R/W	SHUTDOWN_TRIG_SEL	PWM Shutdown trigger sel 0x0 = Low Level sensitive 0x1 = High Level sensitive 0x2 = Negative edge sensitive 0x3 = Positive edge sensitive
	[5:4]	0x0	R/W	SHUTDWON_SEL	PWM Asynchronous Shutdown signal source selection 0x0 = Ext1 0x1 = Ext2 0x2 = COMP IN[1] 0x3 = COMP IN[0]
	[3]	0x0	R/W	PWM_RESTART	Control PWM restart mode from shutdown 0 = PWM will restart when the PWM_SHUTDOWN bit is cleared by writing a 1 1 = PWM will restart automatically upon the next PWM cycle. The STS register will automatically clear.
	[2]	0x0	R/W	PWM_LOW_VAL	Programmable shutdown value for PWM Low level
	[1]	0x0	R/W	PWM_HIGH_VAL	Programmable shutdown value for PWM High level
	[0]	0x0	R/W	PWM_SHUTDOWN_EN	PWM Shutdown mode enabled 0 = Disable 1 = Enable

SECURITY FUNCTIONS

True Random Number Generator

The True Random Number Generator is able to produce true 8−bit random numbers and can be used multiple times to generate 32−bit random numbers. The TRNG uses on−chip sources to generate a string of random bits. This is in contrast to pseudo−random number generators often used, which only look random but are in fact generated by a deterministic algorithm.

The output of the TRNG passes the FIPS Test Suite. For high security applications, bits from the TRNGBYTE should not be used directly, however, because each bit provides slightly less than one bit entropy. Rather TRNG bits should be fed into an entropy pool before use. The AES block can be used for this.

The TRNGMODE Register is used to configure the true random number generator. The recommended setting is to have CLK_SEL0 set to a slower clock when compared to clock selected using CLK_SEL1. It is recommended to use two distinct relatively asynchronous clock sources. A good configuration with on−chip clock sources is selecting LPOSC with CLK_SEL0 and HSOSC with CLK_SEL1. Any entropy value can be used, however we recommend setting ENTROPY to 0x7. Read value when AVAIL bit is set to 1.

True Random Number Generator (TRNG) Registers Table

Table 45. TRUE RANDOM NUMBER GENERATOR (TRNG) REGISTERS TABLE

Function	Bits	Default	Type	Symbol	Description
TRNGCONFIG REGISTER: 0x40A00000 The TRNGMODE Register is used to configure the true random number generator. The recommended settings are TBD.					
CFG	[24]	0x0	RO	AVAIL	When ‘1’, a random byte is available in TRNGBYTE; this bit is cleared by reading TRNGBYTE
	[21:19]	0x7	R/W	CLK_DIV1	Clock pre−scaler 000 = x2 001 = x1 010 = ÷2 011 = ÷4 100 = ÷8 101 = ÷16 110 = ÷32 111 = ÷64
	[18:16]	0x1	R/W	CLK_SEL1	Clock Source 000 = HSOSC 001 = LPOSC 010 = XOSC 011 = LPXOSC 100 = EXTCLK 101 = invalid 110 = System Clock 111 = Off
	[13:11]	0x7	R/W	CLK_DIV0	Clock pre−scaler 000 = x2 001 = x1 010 = ÷2 011 = ÷4 100 = ÷8 101 = ÷16 110 = ÷32 111 = ÷64
	[10:8]	0x1	R/W	CLK_SEL0	Clock source: 000 = HSOSC 001 = LPOSC 010 = XOSC 011 = LPXOSC 100 = EXTCLK 101 = invalid 110 = System Clock 111 = Off

Table 45. TRUE RANDOM NUMBER GENERATOR (TRNG) REGISTERS TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
TRNGCONFIG REGISTER: 0x40A00000 The TRNGMODE Register is used to configure the true random number generator. The recommended settings are TBD.					
	[3:1]	0x7	R/W	ENTROPY	Entropy (assumed to be within one input bit): 000 = 1 Bit 001 = 1/2 Bit 010 = 1/4 Bit 011 = 1/8 Bit 100 = 1/16 Bit 101 = 1/32 Bit 110 = 1/64 Bit 111 = 1/128 Bit
	[0]	0x0	R/W	INT_EN	Random Number Generator Interrupt Enable
TRNGBYTE REGISTER: 0x40A00004 The TRNGBYTE Register contains the generated random number.					
DATA	[7:0]	0x0	RO	DATA	Random Byte

CRC Engine (CRC)

The CRC (cyclic redundancy check) Module is a peripheral that calculates a 32 bit CRC using the standard Ethernet polynomial 0x04C11DB7. This is the CRC used to verify data transmission or storage integrity. In the EN/IEC 60335−1 standard, this unit provides a means to verify the integrity of the Flash memory. The CRC is computed 32 bits at a time. The CRC must be computed on data that is an integer number of 32 bit words. If a half word or byte is written to the CRC input, it will be zero padded to a full 32 bits and the CRC will be calculated using those extra zeros.

To compute the CRC on a file or data stream, start by writing the CRC Data Register to all 1s (0xFFFF_FFFF). Then write each word of the file or data stream to the CRC Input Register in order. Once all words have been written to the CRC Input Register, read the result from the CRC Data Register. If the expected CRC code is placed as the last word of the address range the CRC result will be all zeros indicating a match over the given range.

CRC Register Table

Table 46. CRC REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
CRC DATA REGISTER: 0x40D00000 The CRC Data Register is a read/write register that contains the CRC computation. At reset, the register will contain 0xFFFF_FFFF. If a new CRC calculation is to be started, it should be written back to 0xFFFF_FFFF. If you want a different starting value then you would also write it here. Writing to this register will not start a new CRC calculation.					
RESULT	[31:0]	0xFFFF_FFFF	R/W	RESULT	Contains the current CRC calculation result.
CRC INPUT REGISTER: 0x40D00004 The CRC Input Register is a write only register and is used as the input data to compute a new intermediate (or final) CRC value. Writing to this register will automatically cause an update to the CRC Data register.					
INPUT	[31:0]	N/A	WO	INPUT	CRC update data source.

AES Accelerator (AES)

Hardware accelerated AES encryption and decryption is enabled within AES peripherals. In conjunction with software, the AES block allows a significant reduction in clocks cycles for AES encryption and decryption compared with a software only solution. The AES hardware acceleration supports both 128−bit, 192−bit, and 256−bit encryption/decryption.

Four functions are available to accelerate AES encryption and decryption.

- SubByte
- InvSubByte
- MixColumns
- InvMixColumns

The SubByte and InvSubByte each take four clocks to complete, the MixColumns and InvMixColumns functions are each performed in a single clock cycle, thus dramatically improving the speed of and AES encryption/decryption. The AES operation is triggered by writing a value to the AESDIN register.

It is recommended that the user use the provide software API for the AES function.

AES Performance Comparison Table

Table 47. AES PERFORMANCE COMPARISON TABLE

AES Operation	AXM0F343 with Hardware Assist (Cycles)	Industry Leading Software−Only Solution (Cycles)	Acceleration Improvement Factor
128 bit key expansion	3083	8976	2.9X
128 bit encryption	2467	8928	3.6X
128 bit decryption	1948	16848	8.6X
256 bit key expansion	2055	12672	6.2X
256 bit encryption	3325	12624	3.8X
256 bit decryption	2708	24048	8.9X

AES Register Table

Table 48. AES REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
AES CONTROL REGISTER: 0x40900000					
CTL	[1:0]	00	R/W	CTL	00 = SubByte 01 = MixColumns 10 = InvSubBtye 11 = InvMixColumns
AES INPUT REGISTER: 0x40900004					
INPUT	[31:0]	N/A	WO	INPUT	AES data to operate on. For MixColumns/InvMixColumns, The bits are interpreted as follows: bits 31:24 – top row of the State bits 23:16 – second row of the State bits 15:8 – third row of the State bits 7:0 – bottom row of the State
AES OUTPUT REGISTER: 0x40900008					
OUTPUT	[31:0]	N/A	RO	OUTPUT	AES Result data (same format as the input data)

ANALOG FUNCTIONS

Analog Comparators (CMP)

The AXM0F343 MCU family of products includes two Comparators that are latched comparators. There is a one clock latency on the output of the comparators relative to the input. Input can be selected from a subset of GPIO pins or internal signals. The Comparators are sampled on the System Clock. The comparators are reset and resolve during each clock cycle, thus in order to resolve they will require a larger overdrive as the system clock frequency is increased. To accommodate more accurate comparisons the system clock must be reduced for the compare cycles. When the inputs are disabled the out is 0.

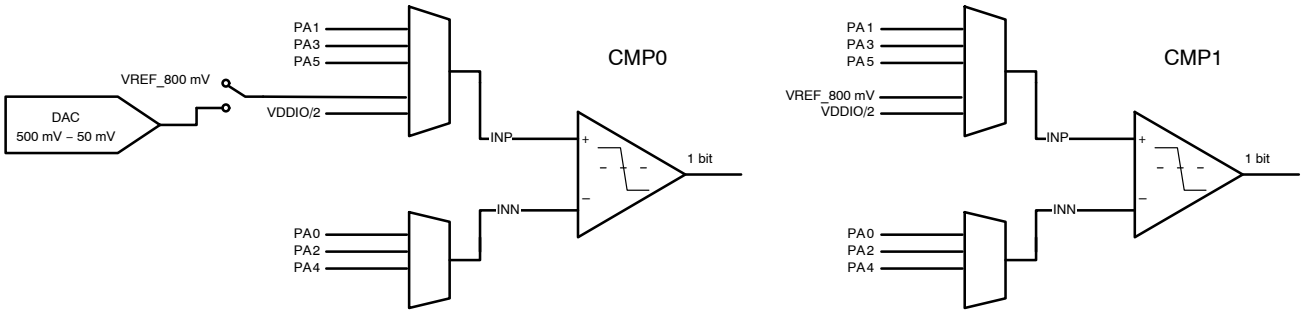


Figure 21. AXM0F343−64 MCU Comparator Input Diagram

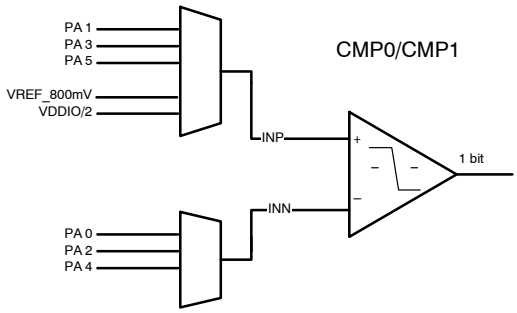


Figure 22. AXM0F343−256 MCU Comparator Input Diagram

Analog Comparator Interface Register Table

Table 49. ANALOG COMPARATOR INTERFACE REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
ANALOG COMPARATOR OUTPUT REGISTER: 0x40600000 The Analog Comparator Output Register is a read only register that contains the result of the Analog Comparator output.					
OUT	[1]	0x0	RO	CMP1	Output of the analog comparator. 1 = Positive Terminal Voltage is higher than Negative Terminal 0 = Positive Terminal Voltage is less than Negative Terminal
	[0]	0x0	RO	CMP0	Output of the analog comparator. 1 = Positive Terminal Voltage is higher than Negative Terminal 0 = Positive Terminal Voltage is less than Negative Terminal

Table 49. ANALOG COMPARATOR INTERFACE REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
ANALOG COMPARATOR CONTROL REGISTER: 0x40600004 The Analog Comparator Control Register is a read/write register that enables and sets−up the inputs for the Analog Comparators.					
CFG	[18]	0x0	R/W	COMP_DACSEL AXM0F343−64 MCU Reserved AXM0F343−256 MCU	Selects if internal Comparator reference (Internal Ref) is the Band Gap or DAC Output 0x1 = DAC Output 0x0 = Internal Ref = Bandgap
	[17:15]	0x00	R/W	COMP_DAC AXM0F343−64 MCU Reserved AXM0F343−256 MCU	DAC 3 bit voltage control. CMP0 Only 0x0 = 50 mV 0x1 = 100 mV 0x2 = 150 mV 0x3 = 200 mV 0x4 = 250 mV 0x5 = 300 mV 0x6 = 400 mV 0x7 = 500 mV
	[14:12]	0x0	R/W	CMP1_PLUS	Select the Positive Terminal Input of the Analog Comparator 1: 0x0 = Disabled 0x1 = PA1 0x2 = PA3 0x3 = PA5 0x4 = Not Available 0x5 = Internal Ref 0x6 = Internal VDD/2 0x7 = Undefined
	[11:9]	0x0	R/W	CMP1_MINUS	Select the Negative Terminal Input of the Analog Comparator 1: 0x0 = Disabled 0x1 = PA0 0x2 = PA2 0x3 = PA4 0x4 = Not Available 0x5 − 0x7 Undefined
	[8]	0x0	R/W	CMP1_EN	Comparator 1 enable (1 = Enabled)
	[7]	−	−	−	Reserved
	[6:4]	0x0	R/W	CMP0_PLUS	Select the Positive Terminal Input of the Analog Comparator 0: 0x0 = Disabled 0x1 = PA1 0x2 = PA3 0x3 = PA5 0x4 = Not Available 0x5 = Internal Ref (or DAC AXM0F343−64 MCU) 0x6 = Internal VDD/2 0x7 = Undefined
	[3:1]	0x0	R/W	CMP0_MINUS	Select the Negative Terminal Input of the Analog Comparator 0: 0x0 = Disabled 0x1 = PA0 0x2 = PA2 0x3 = PA4 0x4 = Not Available 0x5 − 0x7 Undefined
	[0]	0x0	R/W	CMP0_EN	Comparator 0 enable (1 = Enabled)
ANALOG COMPARATOR INTERRUPT ENABLE REGISTER: 0x40600008 The Analog Comparator Interrupt Enable Register is a read/write register that allows the software to enable Analog Comparator interrupts.					
INT_EN	[3]	0x0	R/W	CMP1_FALL	Enable the Analog Comparator 1 Falling Edge Interrupt
	[2]	0x0	R/W	CMP1_RISE	Enable the Analog Comparator 1 Rising Edge Interrupt
	[1]	0x0	R/W	CMP0_FALL	Enable the Analog Comparator 0 Falling Edge Interrupt
	[0]	0x0	R/W	CMP0_RISE	Enable the Analog Comparator 0 Rising Edge Interrupt

Table 49. ANALOG COMPARATOR INTERFACE REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
ANALOG COMPARATOR INTERRUPT STATUS REGISTER: 0x4060000C The Analog Comparator Interrupt Status Register contains the status of the Analog Comparator interrupts. It is written to clear the interrupts.					
INT_STS	[3]	0x0	R/W	CMP1_FALL	Analog Comparator 1 Falling Edge Interrupt status. Write 1 to clear. Will be set when the Analog Comparator 1Output transitions from 1 to 0.
	[2]	0x0	R/W	CMP1_RISE	Analog Comparator 1 Rising Edge Interrupt status. Write 1 to clear. Will be set when the Analog Comparator 1 Output transitions from 0 to 1.
	[1]	0x0	R/W	CMP0_FALL	Analog Comparator 0 Falling Edge Interrupt status. Write 1 to clear. Will be set when the Analog Comparator 0 Output transitions from 1 to 0.
	[0]	0x0	R/W	CMP0_RISE	Analog Comparator 0 Rising Edge Interrupt status. Write 1 to clear. Will be set when the Analog Comparator 0 Output transitions from 0 to 1.

12−bit SAR ADC (ADC)

The ADC Interface contains the registers used to enable and configure the ADC and to retrieve ADC status and conversion data. The 12−bit Differential ADC takes the output from the GPIO or internally generated signal and converts it to a 12−bit digital word. The ADC can operate up to 1MSPS but the performance may degrade even at lower sample rates when using the gain block if improperly configured. The reference voltage for the ADC is programmable to be either the 1.0 V reference voltage, the AVDD regulator, or an external reference. The common mode input voltage is approximately AVDD / 2 (1.8 V / 2 = 0.9 V).

The ADC Gain Amplifier has a 10x gain mode, 1x(unity) gain, and 1/4x gain, all user programmable. The maximum input signal is the chip supply. The input range of the ADC is limited to the lesser of VDDIO or 3.3 V.

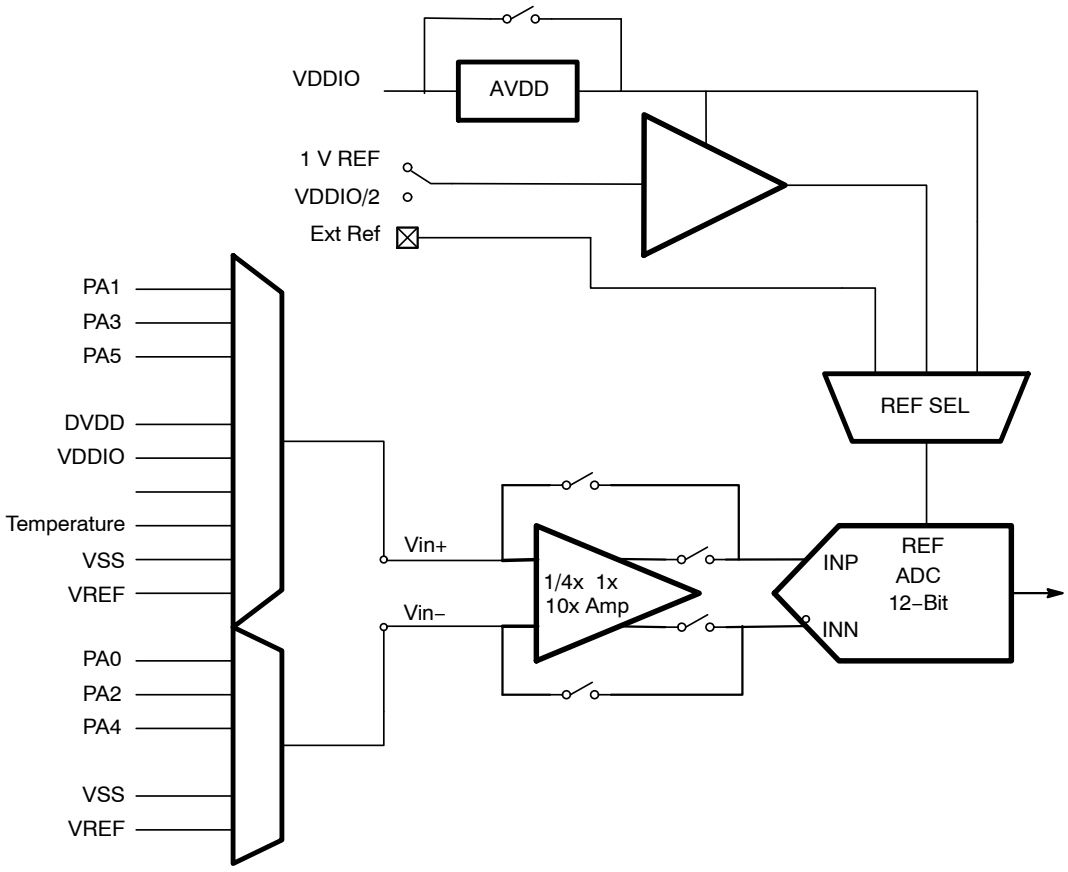


Figure 23. ADC Block Diagram

A programmable gain stage can be used to drive the input of the ADC.

In differential mode the gain block can be used for differential 1x (unity) gain, 10x gain, and 1/4x gain.

For single−ended use it is required to use the 1x (unity) gain mode. Since the ADC is internally differential, the opposite ADC input is driven by the gain buffer. The SINGLE_RANGE_SEL bit allows the input range to be either 0 to Vref (bit = 0) or 0 to 2xVref (bit = 1). It is recommended to use SINGLE_RANGE_SEL set to 1 as it has improved gain amp performance. When SINGLE_RANGE_SEL = 0, the internal ADC input is driven to VSS and the buffer gain is 2; this allows for the ADC to see a 2xVref signal for a full range conversion.

Gain block input range is 250 mV to AVDD – 250 mV.

The PGA is a switched−cap gain stage with two−phase operation. On the first phase, the input capacitors are sampled. During the second phase, the amplification occurs and drives the input of the ADC. The ADC samples the output of the gain block during the last N ADC clock cycles of the gain stage amplify phase, where N is the number of ADC sample clocks as described in the ADC Interface.

AVDD regulator is designed to support internal or external compensation. If the AVDD pin is connected to a 10 μF capacitor then the REG_EXT bit needs to be set. If the AVDD pin is left floating, then the REG_EXT bit needs to be not set. The AVDD reference/supply will be unstable with an alternate setup. Having the AVDD properly external compensated will improve ADC performance.

Internal Temperature Sensor

To utilize the internal temperature sensor the ADC must be set with the 1.0 V internal reference, Single_Range_SEL set to 0 to VREF, 10X Gain mode, and CH_Temp_EN. It is recommended to set the samples clock width (SAMPL_CLK) to 8 sample clocks and slow down the ADC clock when measuring the temperature sensor.

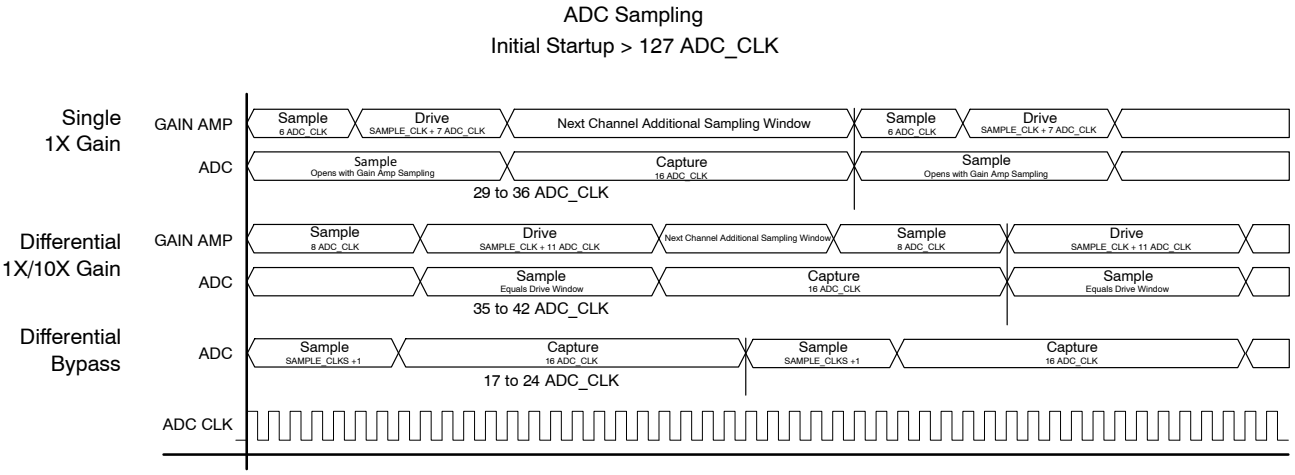


Figure 24. ADC Timing Block Diagram

12−bit SAR ADC Register Table

Table 50. 12−bit SAR ADC REGISTER TABLE

Function	Bits	Default	Type	Symbol	Description
ADC DATA REGISTER: 0x40700000 The ADC Data Register is a read only register that contains the ADC Data.					
DATA	[11:0]	0x000	RO	DATA	Contains the ADC data output.
ADC CONFIGURATION REGISTER: 0x40700004 The ADC Configuration Register is a read/write register that sets−up the ADC. Writing this register will reset the channel sequence to start at the lowest enabled channel. Setting the ADC gain to something other than bypass, or enabling channel 12 (the temperature sensor) enables a charge pump for the gain block. There is a start−up delay of 128 ADC clocks after the charge pump is first enabled before the ADC will allow any conversions to occur. **Note:** ADC Conversions will cycle through the enabled channels, starting at channel 0. The next ADC conversion in single or continuous mode will be the next enabled channel in the cycle.					
CFG	[30:29]	0x00	R/W	Reserved	
	[28]	0x0	R/W	Reserved	
	[27]	0x0		Reserved	
	[26]	0x0	R/W	REG_EXT	Select ADC regulator compensation − must match board configuration. 0 = Internal compensation − Use for lower component count. 1 = External capacitor − Use external capacitor for improved supply rejection.
	[25]	0x0	R/W	SINGLE_RANGE_SEL	0 = Range equals 0 to VREF 1 = Range equals 0 to 2x VREF Changes the single ended conversion range. 0 to 2xVREF is the recommended range selection.
	[24]	0x0	R/W	INT_REF_SEL	Select ADC internal reference source (only asserted when bits [23:22] = ‘b00 1 = VDDIO/2 0 = 1V internal reference
	[23:22]	0x0	R/W	REF_SEL	Select ADC reference voltage 0x0 = internal reference 0x1 = external reference (routed through PA5) Not Buffered and through 1 kΩ internal resistor. 0x2 = internal regulator output (AVDD) 0x3 reserved
	[21]	0x0	R/W	REG_BYPASS	Bypass ADC regulator (AVDD) with VDDIO presented on AVDD. Note that when selected this voltage must not exceed 2.1 V on VDDIO/AVDD.
	[20:19]	0x0	R/W	GAIN	Configures ADC Gain Control 0x0 = Bypass gain block: Differential Conversions Only 0x1 = 0.25x: Internal Supply Measurements only 0x2 = 1x: Required for Single Ended Conversions 0x3 = 10x: Use only for Differential Conversions
	[18:16]	0x0	R/W	SAMP_CLK	Select number of clocks for ADC sample phase. Number of sample clocks = ADC_SAMP_CLKS + 1. See the timing diagram above.
	[15]	0x0	R/W	CH_CORE_VOLT_EN	ADC Internal Channel 3 (Digital Core Voltage)
	[14]	0x0	R/W	CH_CHIP_VOLT_EN	ADC Internal Channel 2 (Chip Supply Voltage)
	[13]	0x0	R/W	CH_CMP_IN_EN	ADC Internal Channel 1 (Comparator Input)
	[12]	0x0	R/W	CH_TEMP_EN	ADC Internal Channel 0 (Temp Sensor)
	[11]	0x0	R/W	CH_DIFF_6_7_EN	Analog External Differential Channels 7&6
	[10]	0x0	R/W	CH_DIFF_4_5_EN	Analog External Differential Channels 5&4

Table 50. 12−bit SAR ADC REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
ADC CONFIGURATION REGISTER: 0x40700004 The ADC Configuration Register is a read/write register that sets−up the ADC. Writing this register will reset the channel sequence to start at the lowest enabled channel. Setting the ADC gain to something other than bypass, or enabling channel 12 (the temperature sensor) enables a charge pump for the gain block. There is a start−up delay of 128 ADC clocks after the charge pump is first enabled before the ADC will allow any conversions to occur. **Note:** ADC Conversions will cycle through the enabled channels, starting at channel 0. The next ADC conversion in single or continuous mode will be the next enabled channel in the cycle.					
	[9]	0x0	R/W	CH_DIFF_2_3_EN	Analog External Differential Channels 3&2
	[8]	0x0	R/W	CH_DIFF_0_1_EN	Analog External Differential Channels 1&0
	[7]	0x0	R/W	CH_SINGLE_7_EN	Analog External Channel 7 (single ended)
	[6]	0x0	R/W	CH_SINGLE_6_EN	Analog External Channel 6 (single ended)
	[5]	0x0	R/W	CH_SINGLE_5_EN	Analog External Channel 5 (single ended)
	[4]	0x0	R/W	CH_SINGLE_4_EN	Analog External Channel 4 (single ended)
	[3]	0x0	R/W	CH_SINGLE_3_EN	Analog External Channel 3 (single ended)
	[2]	0x0	R/W	CH_SINGLE_2_EN	Analog External Channel 2 single ended)
	[1]	0x0	R/W	CH_SINGLE_1_EN	Analog External Channel 1 (single ended)
	[0]	0x0	R/W	CH_SINGLE_0_EN	Analog External Channel 0 (single ended)
ADC CONTROL REGISTER: 0x40700008 The ADC Control Register is a read/write register that starts the ADC conversion. To avoid spurious ADC triggers when changing the hardware trigger selection in the XBAR ADC Trigger Select Register, clear the ADC_HDW_TRIG bit while changing the trigger selection. The default CNT_LIMIT is 0, causing the ADC to run a single conversion and then stop when CNT_EN is set. The maximum value is 2047 (0x7FF) causing the ADC to run 2048 conversions and then stop. Setting the SEQ_RST bit may delay the start of the first conversion while the internal state machine steps through the channels until a selected channel is found (maximum of 16 ADC clocks).					
CTL	[18:8]	0x0	R/W	CNT_LIMIT	When the CNT_EN bit is set, CNT_LIMIT + 1 conversions (max 2047) will be run sequentially. Once CNT_LIMIT+1 is reached the CNT_EN bit clears and the ADC returns to idle.
	[7]	0x0	−	Reserved	
	[6]	0x0	R/W	SEQ_RST	1 = Reset the channel sequence to start at the lowest enabled channel. Self−clearing.
	[5:4]	0x0	R/W	EDGE_TRIG	Set to 0x0 if the hardware trigger is not a pin. 0x0 = Trigger on rising edge 0x1 = Trigger on falling edge 0x2 = Trigger on both edges 0x3 reserved
	[3]	0x0	R/W	AHW_TRIG	In Continuous Conversion Mode, trigger ADC Start Conversion on rising edge of hardware signal. See IN_ADC_CFG in XBAR for trigger assignment. Ignored if not in Continuous Conversion Mode. When ADC_EXT_TRIGGER_PWM = 0.
	[2]	0x0	R/W	OVER_EN	ADC over−write enable 1 = Allow new ADC data to overwrite un−read previous data and channel number 0 = Discard new ADC data and channel number if previous data un−read.
	[1]	0x0	R/W	CONT_EN	Start ADC in Continuous Conversion Mode If a hardware trigger is not enabled, ADC will start next conversion immediately after previous conversion is finished. Else if AHW_TRIG or ADC_EXT_TRIGGER_PWM are set, ADC will start a conversion on each trigger received.
	[0]	0x0	R/W	CNT_EN	Start ADC in Count Conversion Mode CNT_LIMIT + 1 ADC conversions will be taken starting on the next enabled ADC channel. Cleared after CNT_LIMIT + 1 conversions complete. Writing this bit to 0 while the sequence is running will abort the sequence.

Table 50. 12−bit SAR ADC REGISTER TABLE (continued)

Function	Bits	Default	Type	Symbol	Description
ADC CALIBRATE REGISTER (Offset 0x0C) The ADC Calibrate Register is a read/write register to run an ADC calibration.					
CAL	[0]	0x0	R/W	CAL	Write 1 to Start an ADC Calibration Run. Will read as 1 while calibration is in progress which blocks other ADC_CTL conversions. Will read 0 when Calibration is completed. **Note:** Calibration takes about 650 ADC clock cycles.
ADC INTERRUPT ENABLE REGISTER (Offset 0x10) The ADC Interrupt Enable Register is a read/write register that allows the software to enable ADC interrupts.					
INT_EN	[1]	0x0	R/W	OVER	Enable the ADC Over−write Interrupt
	[0]	0x0	R/W	DATA	Enable the ADC Data Ready Interrupt
ADC INTERRUPT STATUS REGISTER (Offset 0x14) The ADC Interrupt Status Register contains the status of the ADC interrupts. It is written to clear the interrupts.					
INT_STS	[1]	0x0	R/W	OVER	ADC Over−write Interrupt status. Write 1 to clear. Will be set when next ADC_DATA is written to ADC Data Register before previous data has been read Or ADC_DATA is not updated due to the previous data not having been read.
	[0]	0x0	R/W	DATA	ADC Data Ready Interrupt status. Write 1 to clear. Will be sent when new ADC_DATA is ready.
ADC CHANNEL (Offset 0x1C) The ADC Channel Register is a read only register to report the most recently read ADC mux channel. It is recommended that this register be read before the ADC Data register to ensure that the channel information is in synchronization with the data (reading the data register allows another conversion to overwrite the existing channel and data values if ADC_OVER is not set).					
CH_STS	[3:0]	0x0	RO	CH_STS	The channel (0−F) used for the most recent conversion.
ADC STATUS REGISTER (Offset 0x20) The ADC Status Register is a read only register to determine the number of conversions that have completed since the START_ADC_COUNT bit was set in the ADC Control Register, and if the ADC is currently running. The count is reset to 0 every time the START_ADC_COUNT bit is set, even if that bit was cleared by software before the full count of conversions were completed.					
STS	[15]	0x1	RO	IDLE	1 = ADC idle 0 = ADC running
	[14:11]		RO	Reserved	
	[10:0]	0x0	RO	CNT	The number of conversions completed since the CNT_EN bit was set

DEBUG FUNCTIONS

Revision ID and Scratch Register

The Revision ID register is used to read the device revision number. Bits 31:4 are the same as the ECO revision code of the Arm Cortex M0+.

The SCRATCH register can be used by software for code storage used in debug. Unlike the AO_SCRATCH register, the debug SCRATCH register is cleared on all forms of reset.

Table 51.

Function	Bits	Default	Type	Symbol	Description
REVISION ID REGISTER: 0x41F00050 – AXM0F343−64 MCU					
REV_ID	[31:4]	0x0000020	RO	ARM_REV_ECO	Same as ARM ECO revision code
	[3:0]	0x1	RO	ON_REV_ID	AXM0F343−64 MCU revision ID
REVISION ID REGISTER: 0x41F00050 – AXM0F343−256 MCU					
REV_ID	[31:4]	0x0001010	RO	ARM_REV_ECO	Same as ARM ECO revision code
	[3:0]	0x8	RO	ON_REV_ID	AXM0F343−256 MCU revision ID
REVISION ID REGISTER: 0x41F00044					
SCRATCH	[31:0]	0x00000000	R/W	SCRATCH	Hardware scratch register for software debug

Debug Port Lockout

The debug port lock is used to disable the serial wire debug port to prevent access to the internal buses and memory for security sensitive applications. At power−up, before the main system reset is released, the LOCK word at address AXM0F343−64 MCU: 0x0000FFFC/ AXM0F343−256 MCU: 0x0003FFFC in the Program Flash memory will be read by an internal state machine. If the value read is set to anything other than 0xFFFF, the LOCK bit will be set in the lock debug port register and the debug port will be disabled. If the LOCK word in the Flash is not set, the debug port will be enabled for external access. For part reprogramming and testing, a lock−override method is implemented in the design such that if the DBG_EN pin is driven high and the eight bit value 0xA5 is clocked in using the debug clock and data pins after the LOCK bit has been set, built in logic will erase the program space in the Program Flash AXM0F343−64 MCU: 64 kB/AXM0F343−256 MCU: 256 kB before clearing the LOCK bit to enable the debug port. When locked, the firmware may implement other methods to access the internal memory map or unlock the debug port via one of the standard interfaces as desired by the application.

Table 52.

Function	Bits	Default	Type	Symbol	Description
DEBUG PORT LOCK REGISTER: 0x40F00060					
LOCK	[1]	0x0	R/W	OVER	Debug Port Lock Override 1 = Lock overridden 0 = No override
	[0]	0x0	RO	STS	Debug Port Locked, detected at power−up.: 1 = Debug Port Locked 0 = Debug Port Unlocked

Micro Trace Buffer (MTB)

When enabled, the MTB records changes in program flow, reported by the processor over the execution trace interface. This information is stored as trace packets in the SRAM. An off−chip debugger can extract the trace information using the DAP to read the trace information from the SRAM. The debugger can then reconstruct the program flow from this information. The MTB simultaneously stores trace information into the SRAM, and gives the processor access to the SRAM. The MTB ensures that trace write accesses have priority over processor accesses.

The MTB does not:

- Include any form of load/store data trace capability.
- Include tracing of any other information.

SRAM accesses occur with zero wait states when there is no trace data being written to the SRAM. Trace packet write access to the SRAM take priority over other accesses. Therefore, one or more wait states can be inserted into other accesses if trace information is simultaneously written to the SRAM.

See CoreSight MTB−M0+ Technical Reference Manual for full details.

MTB Execution Trace Packet Format

The execution trace packet consists of a pair of 32−bit words that the MTB generates when it detects the processor PC value changes non−sequentially. A non−sequential PC change can occur during branch instructions or during exception entry.

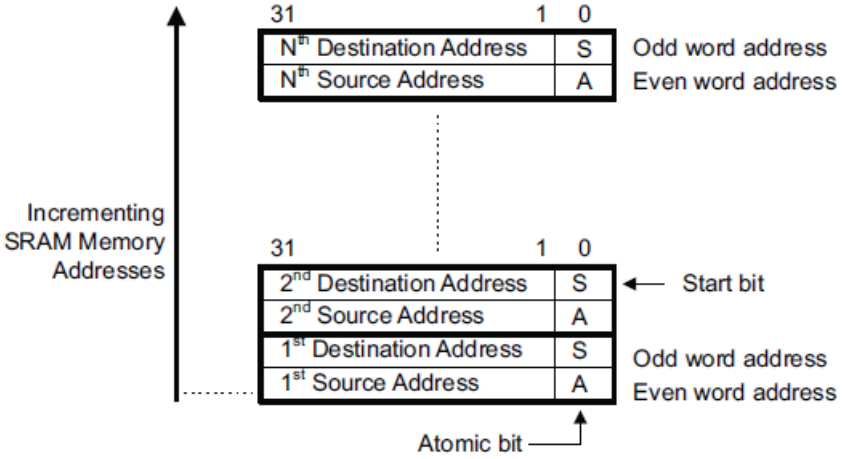


Figure 25. MTB Execution Trace Storage Format

The first, lower addressed, word contains the source of the branch, the address it branched from. The value stored only records bits [31:1] of the source address, because Thumb instructions are at least halfword aligned. The least significant bit of the value is the A−bit. The A−bit indicates the atomic state of the processor at the time of the branch, and can differentiate whether the branch originated from an instruction in a program, an exception, or a PC update in Debug state. When it is zero the branch originated from an instruction, when it is one the branch originated from an exception or PC update in Debug state. This word is always stored at an even word location.

The second, higher addressed word contains the destination of the branch, the address it branched to. The value stored only records bits [31:1] of the branch address. The least significant bit of the value is the S−bit. The S−bit indicates where the trace started. An S−bit value of 1 indicates where the first packet after the trace started and a value of 0 is used for other packets.

Because it is possible to start and stop tracing multiple times in a trace session, the memory might contain several packets with the S−bit set to 1. This word is always stored in the next higher word in memory, an odd word address. When the A−bit is set to 1, the source address field contains the architecturally−preferred return address for the exception. For example, if an exception was caused by an SVC instruction, then the source address field contains the address of the following instruction. This is different from the case where the A−bit is set to 0. In this case, the source address contains the address of the branch instruction.

For an exception return operation, two packets are generated:

- The first packet has the:
  - Source address field set to the address of the instruction that causes the exception return, BX or POP.
  - Destination address field set to bits [31:1] of the EXC_RETURN value.
  - The A−bit set to 0.
- The second packet has the:
  - Source address field set to bits [31:1] of the EXC_RETURN value.
  - Destination address field set to the address of the instruction where execution commences.
  - A−bit set to 1.

MTB Register Table (Base Address 0x30000000)

Table 53. MTB REGISTER TABLE (BASE ADDRESS 0X30000000)

Function	Bits	Default	Type	Symbol	Description
MTB POSITION REGISTER: 0x30000000 The MTB Position Register contains the trace write pointer and the wrap bit. A debug agent might use the WRAP bit to determine whether the trace information above and below the pointer address is valid. The behavior of the trace functionality is **UNPREDICTABLE** if the position register is not programmed prior to enabling trace.					
POSITION	[31:3]	–	R/W	POINTER	Trace packet location pointer. Because a packet consists of two words, the POINTER field is the location of the first word of a packet. This field contains bits [31:3] of the address, in the SRAM, where the next trace packet will be written. The field points to an unused location and is automatically incremented. A debug agent can calculate the system address of the SRAM location pointed to by the POSITION register using the following equation: system address = BASE + ((P + (2^{AWIDTH} – (BASE MOD 2^{AWIDTH}))) MOD 2^{AWIDTH}). Where P = POSITION AND 0xFFFF_FFF8 amd BASE is the BASE register value.
MTB POSITION REGISTER: 0x30000000 The MTB Position Register contains the trace write pointer and the wrap bit. A debug agent might use the WRAP bit to determine whether the trace information above and below the pointer address is valid. The behavior of the trace functionality is **UNPREDICTABLE** if the position register is not programmed prior to enabling trace.					
	[2]	–	R/W	WRAP	This bit is set to 1 automatically when the POINTER value wraps as determined by the MASTER.MASK field in the MASTER Trace Control Register.
	[1:0]	–	–	–	Reserved

Table 53. MTB REGISTER TABLE (BASE ADDRESS 0X30000000) (continued)

Function	Bits	Default	Type	Symbol	Description
MTB MASTER REGISTER: 0x30000004 The MTB Master Register contains the main trace enable bit along with other trace control fields.					
MASTER	[31]	0x0	R/W	EN	Main trace enable bit. When this bit is 1 trace data is written into the SRAM memory location addressed by POSITION.POINTER. The POSITION.POINTER value auto increments after the trace data packet is written. The EN bit can be automatically set to 0 using the FLOW.WATERMARK field and the FLOW.AUTOSTOP bit. The EN bit is automatically set to 1 if the TSTARTEN bit is 1 and the TSTART signal is HIGH. The EN bit is automatically set to 0 if TSTOPEN bit is 1 and the TSTOP signal is HIGH. **Note:** If the EN bit is set to 0 because the FLOW.WATERMARK field is set, then it is not automatically set to 1 if the TSTARTEN bit is 1 and the TSTART input is HIGH. In this case tracing can only be restarted if the FLOW.WATERMARK or POSITION.POINTER value is changed by software.
	[30:10]	–	–	–	Reserved
	[9]	0x0	R/W	HALT_REQ	Halt request bit. This bit is connected to the halt request signal of the trace logic, EDBGRQ. When HALTREQ is set to 1, EDBGRQ is asserted if DBGEN is also HIGH. The HALTREQ bit can be automatically set to 1 using the FLOW.WATERMARK field.
	[8:7]	–	–	–	Reserved
	[6]	0x0	R/W	TSTOP_EN	Trace stop input enable. If this bit is 1 and the TSTOP signal is HIGH, then the EN bit is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped.
	[5]	0x0	R/W	TSTART_EN	Trace start input enable. If this bit is 1 and the TSTART signal is HIGH, then the EN bit is set to 1. Tracing continues until a stop condition occurs.
	[4:0]	Unknown	R/W	MASK	This value determines the maximum size of the trace buffer in SRAM. It specifies the most−significant bit of the POSITION.POINTER field that can be updated by automatic increment. If the trace tries to advance past this power of two, the POSITION.WRAP bit is set to 1, the POSITION. POINTER [MASK:0] bits are set to zero, and the POSITION.POINTER [AWIDTH−4:MASK+1] bits remain unchanged. This field causes the trace packet information to be stored in a circular buffer of size $2^{(MASK+4)}$ bytes, that can be positioned in memory at multiples of this size. Valid values of this field are zero to AWIDTH−4. Values greater than the maximum have the same effect as the maximum.

Table 53. MTB REGISTER TABLE (BASE ADDRESS 0X30000000) (continued)

Function	Bits	Default	Type	Symbol	Description
MTB FLOW REGISTER: 0x30000008 The behavior of the trace functionality is **UNPREDICTABLE** if the flow registers is not programmed prior to enabling trace.					
FLOW	[31:3]	Unknown	R/W	WATERMARK	This field contains an address in the same format as the |POSITION.POINTER field. When the POSITION.POINTER matches the WATERMARK field value, actions defined by the AUTOHALT and AUTOSTOP bits are performed.
	[2]	–	–	–	Reserved
	[1]	0x0	R/W	AUTOHALT	If this bit is 1 and WATERMARK is equal to POSITION.POINTER, then the MASTER.HALTREQ bit is automatically set to 1. If the DBGEN signal is HIGH, the MTB asserts this halt request to the Cortex−M0+ processor by asserting the EDBGRQ signal.
	[0]	0x0	R/W	AUTOSTOP	If this bit is 1 and WATERMARK is equal to POSITION.POINTER, then the MASTER.EN bit is automatically set to 0. This stops tracing.
MTB BASE REGISTER: 0x3000000C The MTB Base Register indicates where the SRAM is located in the processor memory map. This register is provided to enable auto discovery of the MTB SRAM location, by a debug agent.					
BASE	[31:0]	0xFFFFFFFF	RO	BASE	The value provided is the value of the Base address.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

ON Semiconductor Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative