One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Using the ADuCM360/ADuCM361 Low Power, Precision Analog Microcontroller with Dual Sigma-Delta ADCs, ARM Cortex-M3

SCOPE

This user guide provides a detailed description of the ADuCM360/ADuCM361 functionality and features.

DISCLAIMER

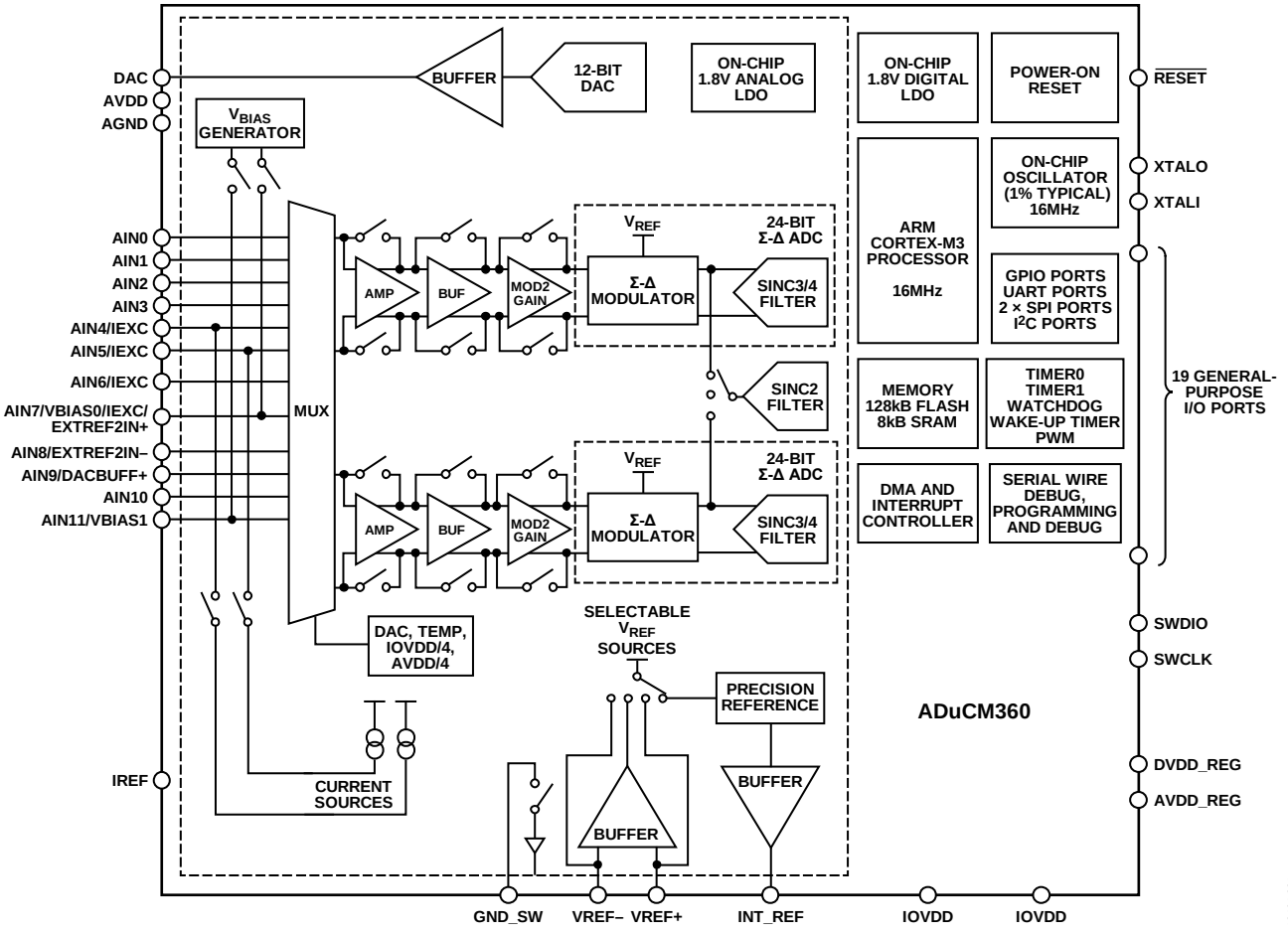

Figure 1. ADuCM360 Block Diagram

TABLE OF CONTENTS

REVISION HISTORY

5/14—Rev. B to Rev. C

Change to Page 61 61

7/13—Rev. A to Rev. B

Changes to Figure 1 1
Moved Revision History Section 4
Added Figure 2, Renumbered Sequentially; Changes to Single 12-Bit Voltage Output DAC Section 7
Changes to Memory Section 8
Changes to Figure 3 10
Change to Table 4 11
Changes to Clocking Architecture Control Register 0 Section and Table 5 12
Changes to Power Management Unit Operation Section 13
Changes to ADC Circuit Overview Section and Figure 4 20
Changes to Analog Input Channel Configuration Section 21
Removed Step Detection Circuit—Mode 1 Operation, DETCON[3] = 1 Section 24
Changed Step Detection Circuit—Mode 0 Operation, DETCON[3] = 0 Section to Step Detection Circuit Section; Changes to Figure 7 Caption 26
Changes to Bias Voltage (V_{BIAS}) Generator Circuit Section, Digital Filter Option Section, and Figure 10 28
Changes to Table 16 and Table 17; Added Simultaneous Sampling Section 29
Changes to ADC Data Register Section and Figure 11 Caption 34
Changes to Bit 2, Table 22 37
Changes to Offset Calibration Registers Section and Table 25 39
Changes to Bit 15 and Bit 14, Table 29 40
Changes to Table 42 and Table 43 47
Change to Bit 6, Table 48 49
Changes to Figure 17 52
Change to DAC NPN Transistor Driver Mode, DACCON[8] = 1 Section 53
Changes to Table 55 56
Changes to Flash Protection: User Read Protection Section 88
Changes to Flash Controller System IRQ Abort Enable Register Section and Table 109 95
Changes to Flash Controller System IRQ Abort Enable Register Section and Table 110 97
Changes to Flash Controller System IRQ Abort Enable Register Section and Table 111 98
Changes to I/O Pull-Up Enable Section 101
Changes to I^2C Overview Section 107
Changes to DMA Mode Section 139
Changes to Bit 15, Table 169 149
Changes to Interrupts/Wake-up Signals Section 155
Changes to Wake-Up Timer Compare Register A Section 159
Changes to Watchdog Timer Clear Interrupt Register Section 162
Changes to PWM Section and Table 200 163
Changes to Figure 33 and Table 201 164
Changes to Table 202 165
Changes to H-Bridge Mode Section, and Table 203 166
Changes to PWM Trip Function Interrupt Section and PWM Output Pairs Interrupts Section 167
Changes to Table 205 168
Changes to Low Dropout Regulators Section 170
Changes to Table 208 171
Changes to Figure 38 174
Added Related Links Section 176

11/12—Rev. 0 to Rev. A

Changes to Pin 35 and Pin 36 Description 169

10/12—Revision 0: Initial Version

USING THE ADuCM360/ADuCM361 HARDWARE USER GUIDE

NUMBER NOTATIONS

Table 1. Number Notations

Notation	Description
Bit N	Bits are numbered in little endian format, that is, the least significant bit of a number is referred to as Bit 0.
V[X:Y]	Bit field representation covering Bit X to Bit Y of a value or a field (V).
0xNN	Hexadecimal (Base 16) numbers are preceded by the prefix 0x.
0bNN	Binary (Base 2) numbers are preceded by the prefix 0b'.
NN	Decimal (Base 10) are represented using no additional prefixes or suffixes.

REGISTER ACCESS CONVENTIONS

Table 2. Register Access Conventions

Access	Description
RW	Memory location has read and write access.
R	Memory location is read access only. A read always returns 0, unless otherwise specified.
W	Memory location is write access only.

ACRONYMS AND ABBREVIATIONS

Table 3. Acronyms and Abbreviations

Acronym/Abbreviation	Description
Σ-Δ	Sigma delta
ADC	Analog-to-digital converter
AF	Averaging factor
AFE	Analog front end
ARM	Advanced RISC machine
CD	Clock divider
DMA	Direct memory access
DSB	Data synchronization barrier
FSE	Full-scale error: gain error plus offset error
JTAG	Joint test action group
LSB	Least significant byte/bit
MMR	Memory mapped register
MSB	Most significant byte/bit
NMI	Nonmaskable interrupt
NVIC	Nested vectored interrupt controller
PGA	Programmable gain amplifier
PMU	Power management unit
POR	Power-on reset
PSM	Power supply monitor
PWM	Pulse-width modulator
RMS	Root mean square
Rx	Receive
SF	Sinc3/sinc4 filter
SIL	Safety integrity level
SPI	Serial peripheral interface
Tx	Transmit
UART	Universal asynchronous transmitter

INTRODUCTION TO THE ADuCM360/ADuCM361

The ADuCM360 is a fully integrated, 4 kSPS, 24-bit data acquisition system incorporating dual, high performance, multichannel sigma-delta (Σ-Δ) analog-to-digital converters (ADCs), 32-bit ARM Cortex-M3® processor, and Flash/EE memory on a single chip. The part is designed for direct interfacing to external precision sensors in both wired and battery-powered applications.

The ADuCM361 contains all the features of the ADuCM360 except ADC0. ADuCM361 only has one ADC, ADC1.

The device contains an on-chip 32 kHz oscillator and an internal 16 MHz high frequency oscillator. This clock is routed through a programmable clock divider from which the processor clock operating frequency is generated. The maximum clock speed is 16 MHz and this is not limited by operating voltage or temperature.

The microcontroller is a low power Cortex-M3 processor from ARM. It is a 32-bit RISC machine, offering up to 20 MIPS peak performance. The Cortex-M3 processor incorporates a flexible 11-channel DMA controller supporting the following: 2 SPI, UART, ADC and DAC. There are 128 kB of nonvolatile Flash/EE and 8 kB of SRAM integrated on chip.

The analog subsystem consists of dual ADCs, each connected to a flexible input multiplexer. Both ADCs can operate in fully differential and single-ended modes. Other on-chip ADC features include dual programmable excitation current sources, diagnostic current sources, and a bias voltage generator of AVDD_REG/2 (900 mV) to set the common-mode voltage of an input channel. A low-side internal ground switch is provided to allow powering down of a bridge between conversions. The ADCs contain two parallel filters—a sinc3 or sinc4 in parallel with a sinc2. The sinc3 or sinc4 filter is for precision measurements. The sinc2 filter is for fast measurements and for detection of step changes in the input signal. The device also contains a low noise, low drift internal band gap reference or can be configured to accept up to two external reference sources in ratiometric measurement configurations. An option to buffer the external reference inputs is also provided on chip. A single-channel buffered voltage output DAC is also provided on chip. The ADuCM360/ADuCM361 also integrates a range of on-chip peripherals that can be configured under microcontroller software control as required in the application. These peripherals include UART, I^2C, and dual SPI serial I/O communication controllers, 19-pin GPIO ports, two general-purpose timers, wake-up timer, and system watchdog timer. A 16-bit PWM with six output channels is also provided.

The ADuCM360/ADuCM361 is specifically designed to operate in battery-powered applications where low power operation is critical. The microcontroller can be configured in a normal operating mode that consumes 290 μA/MHz (including flash/SRAM I_{DD}), resulting in an overall system current consumption of 1 mA when all peripherals are active.

The part can also be configured in a number of low power operating modes under direct program control, including hibernate mode (internal wake-up timer active), which consumes only 4 μA. In hibernate mode, peripherals such as external interrupts or the internal wake up timer can wake up the device. This allows the part to operate in an ultralow power operating mode and still respond to asynchronous external or periodic events.

On-chip factory firmware supports in-circuit serial download via a serial wire interface (2-pin JTAG system) and UART while nonintrusive emulation is also supported via the serial wire interface. These features are incorporated into a low cost QuickStart™ development system supporting this precision analog microcontroller family.

The part operates from an external 1.8 V to 3.6 V voltage supply and is specified over an industrial temperature range of −40°C to +125°C.

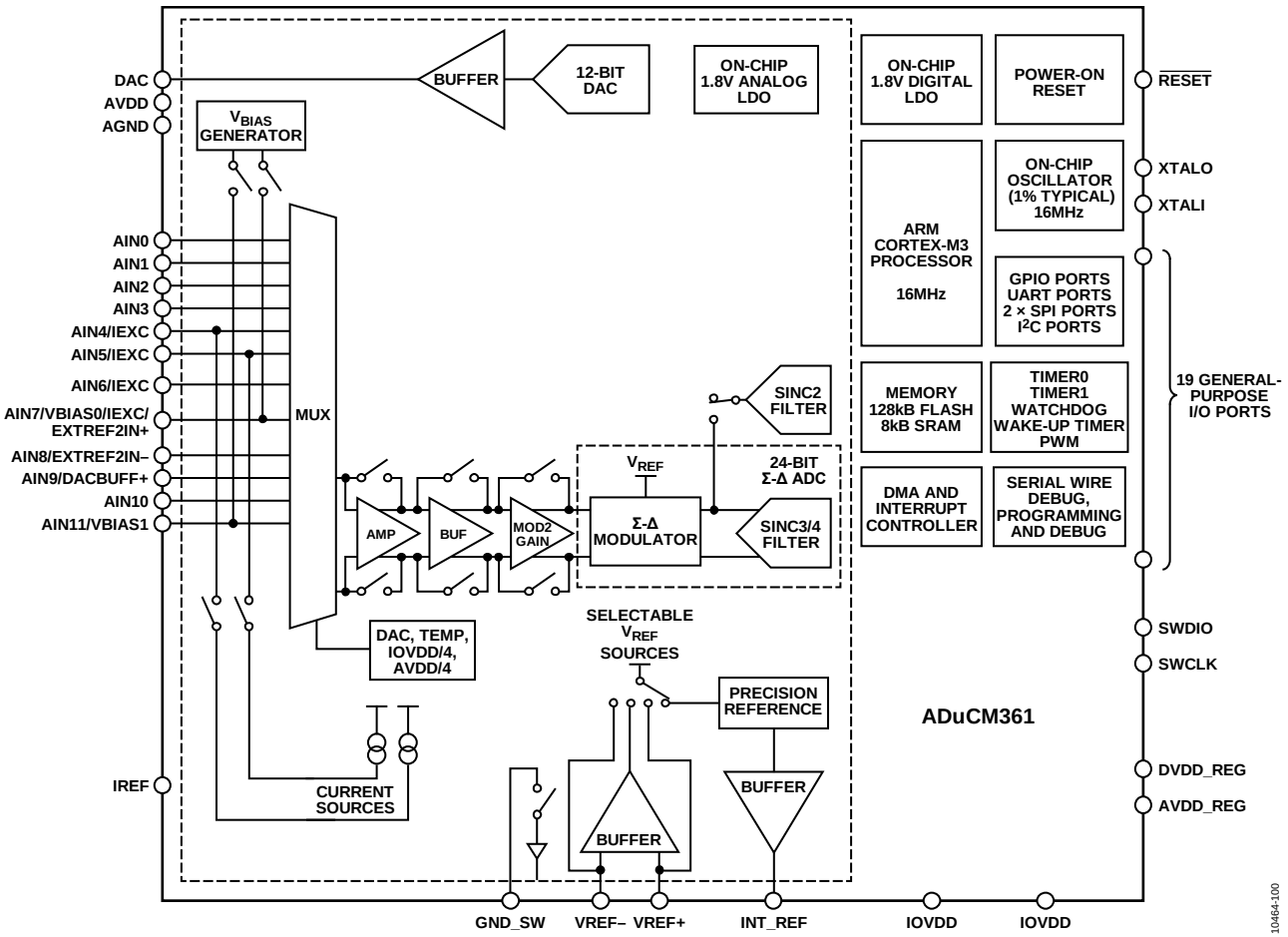

Figure 2. ADuCM361 Block Diagram

MAIN FEATURES OF ADuCM360/ADuCM361

Analog I/O

- Ultrahigh precision, multichannel, dual 24-bit ADCs
- Single-ended and fully differential inputs
- Independently programmable ADC output rate (3.5 Hz to 3.906 kHz)
- Simultaneous 50 Hz/60 Hz rejection:
  - 50 SPS continuous conversion mode
  - 16.67 SPS single conversion mode
- Flexible input multiplexer for input channel selection to both ADCs
- ADC0 and ADC1 (24-bit) ADC channel
  - 6 differential or 11 single-ended input channels
  - 4 internal channels for monitoring DAC, temperature sensor, IOVDD, and AVDD (ADC1 only)
  - Programmable gain (1 to 128)
  - Selectable input range (see the ADuCM360/ADuCM361 data sheet for more details)
  - RMS noise (see ADuCM360/ADuCM361 data sheet for more details)
- Programmable sensor excitation current sources
  - 10 µA/50 µA/100 µA/150 µA/200 µA/250 µA/300 µA/400 µA/450 µA/500 µA/600 µA/750 µA/800 µA and 1 mA current source options
- On-chip precision voltage reference (see the ADuCM360/ADuCM361 data sheet for more details)

Single 12-Bit Voltage Output DAC

- Voltage output DAC
- Can also be configured for an NPN-transistor driver mode and interpolation mode

Microcontroller

- ARM Cortex-M3 32-bit processor
- Serial wire download and debug
- Internal watch crystal for wake-up timer
- 16 MHz oscillator with 8-way programmable divider

Memory

- 128 kB Flash/EE memory, 8 kB SRAM
- 10000 cycle Flash/EE endurance
- 10 year Flash/EE retention
- In-circuit download via serial wire and UART

Power

- Operates directly from a 3.0 V battery
- Supply range: 1.8 V to 3.6 V
- Flexible operating modes for low power applications
- Power consumption:
  - Processor active mode: core consumes 290 μA/MHz
  - Active mode: 1 mA (all peripherals active) processor operating at 500 kHz
  - Power-down mode: 4 μA (WU timer active)

On-Chip Peripherals

- UART, I^2C, and two SPI serial I/O communication controllers
- 19-pin GPIO port
- Two general-purpose timers
- Wake-up timer
- Watchdog timer
- 16-bit PWM controller
- Multichannel DMA and interrupt controller

Packages and Temperature Range

- 48-lead LFCSP (7 mm × 7 mm) package −40°C to +125°C

Tools

- Low cost QuickStart development system
- Third-party compiler and emulator tool support

Applications

- Industrial automation and process control
- Intelligent, precision sensing systems
- 4 mA to 20 mA loop powered smart sensor systems

MEMORY ORGANIZATION

The ADuCM360/ADuCM361 memory organization is described in this section.

Three separate blocks of memory are accessible to the user:

- 8 kB of SRAM from 0x20000000 to 0x20001FFF
- 128 kB of on-chip Flash/EE memory available to the user from 0x00000 to 0x1FFFF
- An additional 2 kB reserved for the kernel space from 0x20000 to 0x207FF

These blocks are mapped according to the Cortex-M3 memory map as shown in Figure 3. All on-chip peripherals are accessed via memory mapped registers, situated in the bit band region.

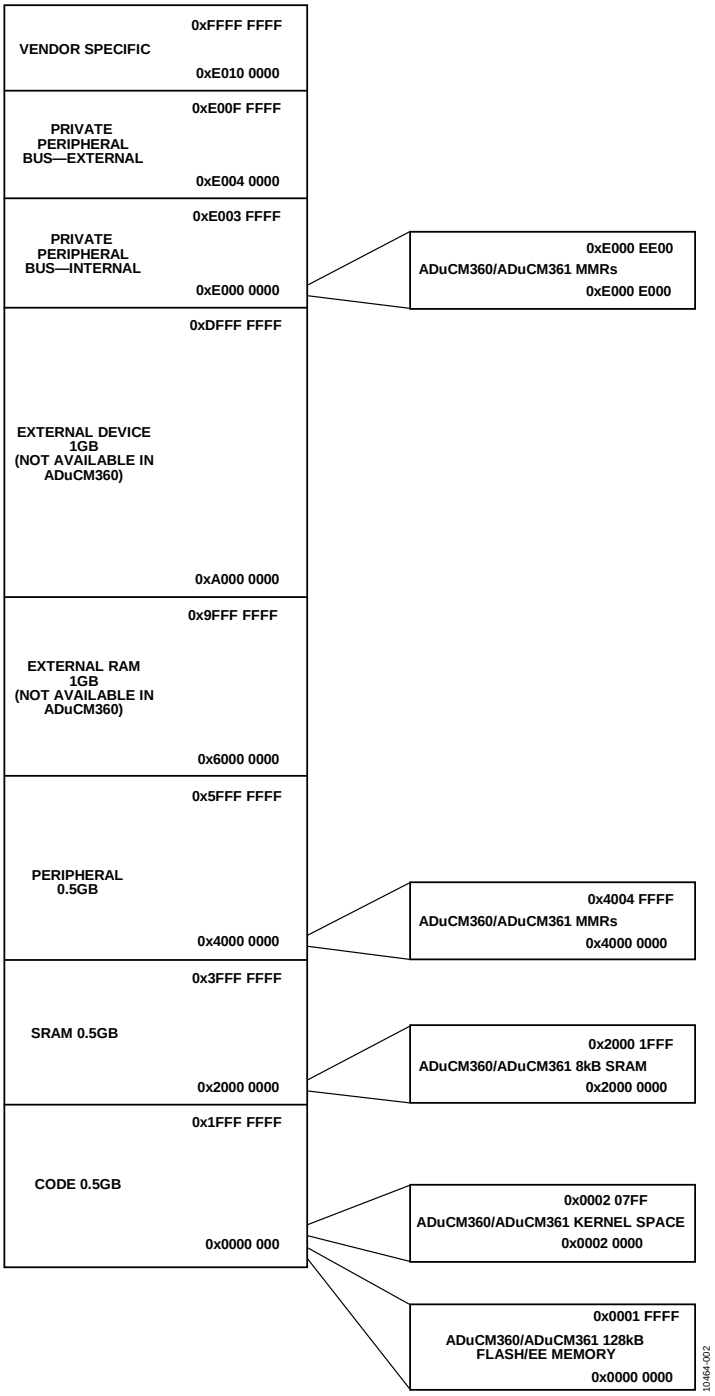

Figure 3. ADuCM360/ADuCM361 Cortex-M3 Memory Map Diagram

CLOCKING ARCHITECTURE

CLOCKING ARCHITECTURE FEATURES

The ADuCM360/ADuCM361 integrates two on-chip oscillators and circuitry for an external crystal:

- LFOSC is a 32 kHz low power internal oscillator, used in low power modes.
- HFOSC is a 16 MHz internal oscillator that is used in active mode.
- LFXTAL is a 32 kHz external crystal.
- Power saving clock mechanism; enable and disable per peripheral.

CLOCKING ARCHITECTURE BLOCK DIAGRAM

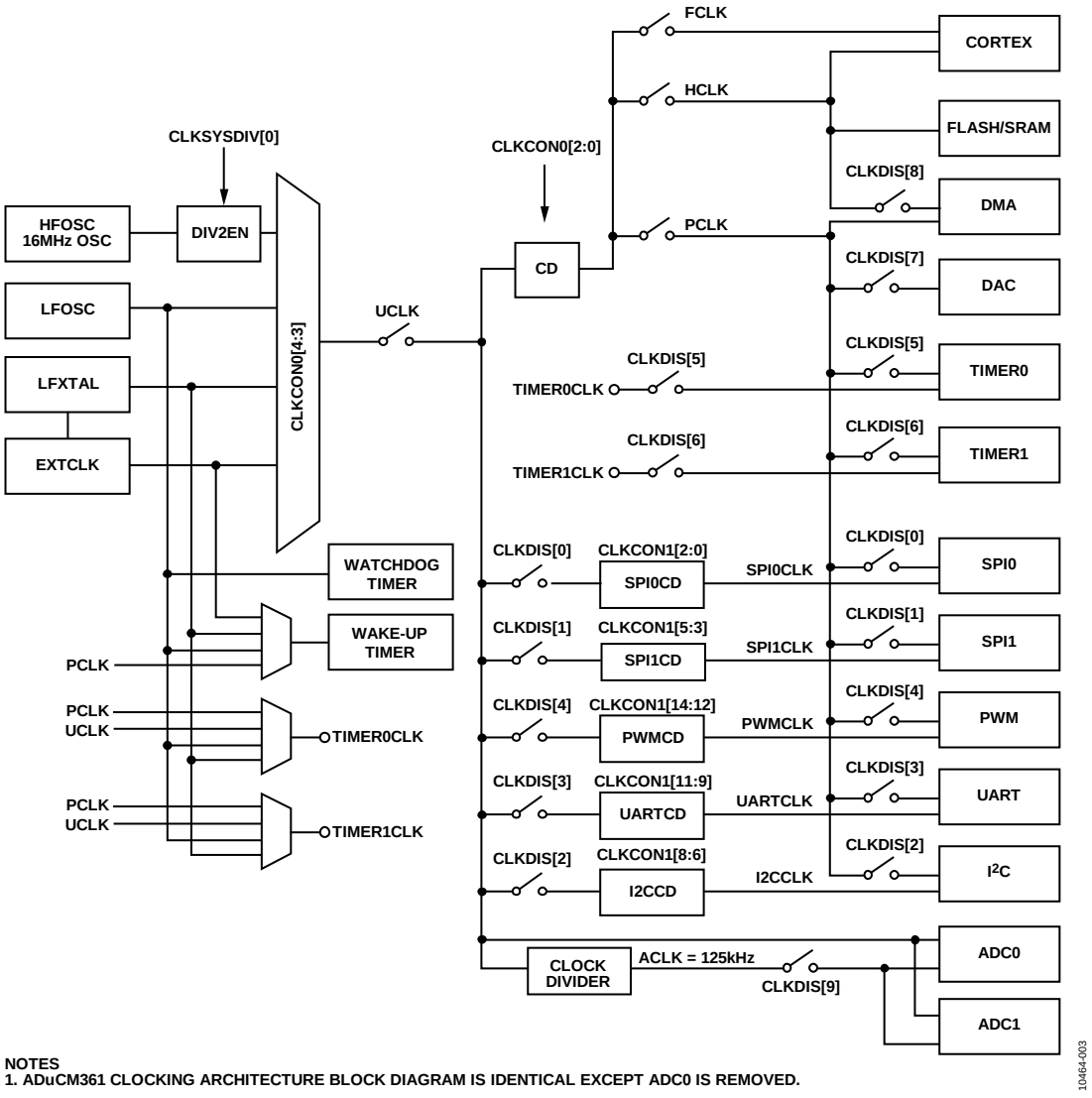

Figure 4. ADuCM360 Clocking Architecture Block Diagram
Note that the ADuCM361 Clocking Architecture Block Diagram is Identical Except ADC0 Is Removed

CLOCKING ARCHITECTURE OVERVIEW

Three of the clock sources—LFOSC, HFOSC and LFXTAL—can be used as system clocks. An external clock on P1.0 can also be used for test purposes.

Internally, the system clock is divided into five clocks:

- FCLK for the cortex processor
- UCLK system clock
- HCLK for the flash, SRAM, DMA
- ACLK for the analog front end and ADC clock
- PCLK for the other peripherals

Figure 4 shows all the clocks available and includes clock gates for power management. For more information about the clock gates, see the Power Management Unit section.

CLOCKING ARCHITECTURE OPERATION

At power-up, the processor executes from the 16 MHz internal oscillator divided down to 1 MHz. User code can select the clock source for the system clock and can divide the clock by a factor of 2^{CD}, where CD is the clock divider bits (CLKCON0[2:0]). This allows slower code execution and reduced power consumption.

Note that P1.0 must be configured first as a clock input before the clock source is switched in the clock control register.

CLOCKING ARCHITECTURE MEMORY MAPPED REGISTERS

Table 4. Clock Control Memory Mapped Registers Address Table

Address	Name	Description	Access	Default
0x40002000	CLKCON0	System Clocks Control Register 0	RW	0x0004
0x40002004	CLKCON1	System Clocks Control Register 1	RW	0x0000
0x4000202C	CLKDIS	System clock to peripherals enable/disable	RW	0x01FF
0x40002410	XOSCCON	Crystal oscillator control register	RW	0x0000
0x40002444	CLKSYSDIV	System clock divide-by-2 control register	RW	0x0001

Clocking Architecture Control Register 0

Address: 0x40002000, Reset: 0x0004, Name: CLKCON0

Table 5. CLKCON0 Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved
7:5	CLKOUT	Clock out multiplexer selection bits (use for test purposes only) 000: UCLK (default) 001: UCLK 010: PCLK 011: Reserved 100: Reserved 101: HFOSC 110: LFOSC 111: LFXTAL
4:3	CLKMUX	Clock in multiplexer selection bits 00: HFOSC (default) 01: LFXTAL 10: LFOSC 11: EXTCLK
2:0	CD	Clock divide bits[1] 000: DIV1: UCLK/1 001: DIV2: UCLK/2 010: DIV4: UCLK/4 011: DIV8: UCLK/8 100: DIV16: UCLK/16 (default) 101: DIV32: UCLK/32 110: DIV64: UCLK/64 111: DIV128: UCLK/128

[1] An additional divide-by-2 is required if CLKSYSDIV[0] is set to 1.

Clocking Architecture Control Register 1

Address: 0x40002004, Reset: 0x0000, Name: CLKCON1[1]

Table 6. CLKCON1 Register Bit Descriptions

Bits	Name	Description
15	Reserved	
14:12	PWMCD	Clock divide bits for PWM system clock[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
11:9	UARTCD	Clock divide bits for UART system clock[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
8:6	I2CCD	Clock divide bits for I²C system clock[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz (minimum value to support 400 kHz I²C baud rate) 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz (minimum value to support 100 kHz I²C baud rate) 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
5:3	SPI1CD	Clock divide bits for SPI1 system clock[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
2:0	SPI0CD	Clock divide bits for SPI0 system clock[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz

[1] Peripheral clock must be greater than or equal to the FCLK. FCLK is set by CLKCON0[2:0].
[2] Calculations are for UCLK = 16 MHz with CLKSYSDIV[0] set to 0. An additional divide-by-2 is required if CLKSYSDIV[0] is set to 1.

Clocking Architecture Disable Register

Address: 0x4000202C, Reset: 0x01FF, Name: CLKDIS

Table 7. CLKDIS Register Bit Descriptions

Bits	Name	Description
15:10	Reserved	These bits are reserved and should be written 0.
9	DISADCCLK	0: Enable ADC system clock (default). 1: Disable ADC system clock.
8	DISDMACLK	0: Enable DMA system clock. 1: Disable DMA system clock.
7	DISDACCLK	0: Enable DAC system clock. 1: Disable DAC system clock.
6	DIST1CLK	0: Enable Timer1 system clock. 1: Disable Timer1 system clock.
5	DIST0CLK	0: Enable Timer0 system clock. 1: Disable Timer0 system clock.
4	DISPWMCLK	0: Enable PWM system clock. 1: Disable PWM system clock.
3	DISUARTCLK	0: Enable UART system clock. 1: Disable UART clock.
2	DISI2CCLK	0: Enable I^2C system clock. 1: Disable I^2C system clock.
1	DISSPI1CLK	0: Enable SPI1 system clock. 1: Disable SPI1 system clock.
0	DISSPI0CLK	0: Enable SPI0 system clock. 1: Disable SPI0 system clock.

External Crystal Oscillator Control Register

Address: 0x40002410, Reset: 0x0000, Name: XOSCCON

Table 8. XOSCCON Register Bit Descriptions

Bits	Name	Description
7:3	Reserved	These bits are reserved and should be written 0.
2	DIV2	Divide-by-2 enable bit. 0: Disable the clock divider. 1: Enable the clock divider.
1	Reserved	This bit is reserved and should be written 0.
0	ENABLE	Crystal oscillator circuit enable. 0: Disable the oscillator circuitry. 1: Enable the oscillator circuitry.

Clocking Architecture Divide Register

Address: 0x40002444, Reset: 0x0001, Name: CLKSYSDIV

Table 9. CLKSYSDIV Register Bit Descriptions

Bits	Name	Description
7:1	Reserved	These bits are reserved and should be written 0.
0	DIV2EN	Divide-by-2 enable bit. By default, this bit is 1, meaning that the system clock is 8 MHz. Enable this in low power systems. 0: Disable the clock divider; the system clock is 16 MHz. 1: Enable the clock divider; the system clock is 8 MHz (default).

POWER MANAGEMENT UNIT

POWER MANAGEMENT UNIT FEATURES

The power management unit (PMU) controls the different power modes of the ADuCM360/ADuCM361.

Six power modes are available:

- Active
- MCUHALT
- PERHALT
- SYSHALT
- TOTALHALT
- Hibernate

POWER MANAGEMENT UNIT OVERVIEW

The PMU controls the power modes of the ADuCM360/ADuCM361.

The Cortex-M3 sleep modes are linked to the PMU modes and are described in this section. The PMU is in the always on section. Each mode gives a power reduction benefit with a corresponding reduction in functionality. The most restrictive mode gives a power-down figure of ~4 µA, and the least restrictive mode gives a power-down figure that is dependent on the CD clock rate chosen by the user, ~290 µA/MHz.

All current values are available in the ADuCM360/ADuCM361 data sheet.

Table 10. Power Modes Summary

Mode	Name	Description	Estimated Current[1] CLKSYSDIV = 0	Estimated Current CLKSYSDIV = 1	Wake-Up Time
0	Active	Full active mode.	250 µA + 290 µA/MHz	145 µA + 145 µA/MHz	
1	MCUHALT	Gate HCLK when Cortex-M3 is also in sleep modes.	435 µA + 50 µA/MHz	355 µA + 50 µA/MHz	3 × FCLK to 5 × FCLK
2	PERHALT	Gate PCLK when Cortex-M3 also is in sleep modes.	425 µA + 60 µA/MHz	345 µA + 60 µA/MHz	3 × FCLK to 5 × FCLK
3	SYSHALT	Gate both HCLK, ACLK & PCLK when Cortex-M3 is in sleep modes.	420 µA + 16 µA/MHz	345 µA + 16 µA/MHz	3 × FCLK to 5 × FCLK
4	TOTALHALT	Gate FCLK when Cortex-M3 is in deep sleep mode.	~4 µA	~4 µA	30.8 µs
5	Hibernate	Turns all digital blocks off except always on blocks.	~4 µA	~4 µA	30.8 µs

[1] Estimated current = static current + dynamic current, where dynamic current is frequency. The values given are typical values at T_A = 25°C.

POWER MANAGEMENT UNIT OPERATION

The debug tools can prevent the Cortex-M3 from fully entering its power saving modes by setting bits in the debug logic. Only a power-on-reset resets the debug logic. Therefore, the device should be power cycled after using serial wire debug with application code containing the WFI instruction.

Power Mode: Active Mode, Mode 0

The system is fully active. Memories and all user enabled peripherals are clocked, and the Cortex-M3 processor is executing instructions. Note that the Cortex-M3 processor manages its internal clocks and can be in a partial clock gated state. This clock gating affects only the internal Cortex-M3 processing core. Automatic clock gating is used on all blocks and is transparent to the user. User code can use a WFI command to put the Cortex-M3 processor into sleep mode; it is independent of the power mode settings of the PMU.

When the ADuCM360/ADuCM361 wake up from any of the low power mode, the devices return to Mode 0.

Power Mode: MCUHALT Mode, Mode 1

The system gates HCLK at an early stage after the Cortex-M3 processor has entered sleep mode. The HCLK is the clock to the SRAM, flash, and DMA blocks. The gating of this clock stops all these peripherals. The Cortex-M3 processor FCLK is active, and the device wakes up using the NVIC. If ADC0, ADC1, or sinc2 DMA channels are enabled in the DMAENSET register, the clock remains active to the DMA block and the SRAM.

Power Mode: PERHALT Mode, Mode 2

The system gates PCLK at an early stage after the Cortex-M3 processor has entered sleep mode. The PCLK is the clock to user peripherals. The gating of this clock stops all clocks to these peripherals. The Cortex-M3 processor FCLK is active and the device wakes up using the NVIC. If ADC0, ADC1, or sinc2 DMA channels are enabled in the DAMENSET register, the clock remains active to the DMA block and the SRAM.

Power Mode: SYSHALT Mode, Mode 3

The system gates HCLK and PCLK at an early stage after the Cortex-M3 processor has entered sleep mode. The Cortex-M3 processor FCLK is active and the device wakes up using the NVIC.

Power Mode: TOTALHALT Mode, Mode 4

The system gates FCLK, HCLK, and PCLK at an early stage after the Cortex-M3 processor has entered deep sleep mode. The FCLK is the free running clock of the Cortex-M3 processor. The gating of FCLK, HCLK, and PCLK clocks stops some of the NVIC functionality. The Cortex-M3 processor FCLK is, therefore, stopped and only the peripherals listed in Table 55 can wake up the Cortex-M3 processor. The low power LDO is turned on and the high power LDO is turned off automatically when entering Mode 4.

Power Mode: Hibernate Mode, Mode 5

The system gates power to the digital core. All states are retained during this power gating, which appears to the user as a clock gating of FCLK, HCLK, and PCLK at an early stage after the Cortex-M3 processor has entered deep sleep mode; however, the leakage current is lower and the response time is different (that is, the device is slower to come up). The Cortex-M3 processor FCLK is, therefore, stopped and only the peripherals listed in Table 55 can wake up the Cortex-M3 processor.

The low power LDO enabled in Power Mode 4 and Power Mode 5 is only capable of sourcing 100 μA typically. Therefore, users need to be careful that no high frequency clock signals (that is, signals greater than 1 MHz) are provided on digital inputs because such signals can cause internal circuits to consume power that exceeds this 100 μA limit. Because of this, Power Mode 4 and Power Mode 5 are not supported when an external clock is present on P1.0. These modes are supported, however, when digital pins are sourcing current as digital outputs because this current is sourced from the IOVDD supply and not the low power LDO.

If Power Mode 1 to Power Mode 5 is entered when the processor is in an interrupt handler, the power-down mode can be exited only by a reset or a higher priority interrupt source.

POWER MANAGEMENT UNIT MEMORY MAPPED REGISTERS

The power modes are controlled by a single register based in the section that is always on at Address 0x40002400.

Table 11. System Clocks Memory Mapped Register Address (Base Address: 0x40002400)

Offset	Name	Description	Access	Default
0x0000	PWRMOD	Power modes register	RW	0x00
0x0004	PWRKEY	Key protection for PWRMOD	RW	N/A

Power Modes Control Register

Address: 0x40002400, Reset: 0x0000, Name: PWRMOD

Table 12. PWRMOD Register Bit Descriptions

Bits	Name	Description
7:3	Reserved	Reserved. These bits should be written 0 by user code.
2:0	MOD	Power modes control bits. Selects the power mode to enter. When read, these bits contain the last power mode value entered by user code. 000: Full active. 001: MCUHALT. 010: PERHALT. 011: SYSHALT. 100: TOTALHALT. 101: Hibernate. Other: Reserved.

To place the cortex in deep sleep mode when the device is in Mode 4 or Mode 5, the Cortex-M3 processor system control register (Address 0xE000ED10) must be configured to 1 in Bit 2 (SCR[2]).

Power Modes Key Register

Address: 0x40002400, Reset: N/A, Name: PWRKEY

Table 13. PWRKEY Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Power control key register. The PWRMOD register is key protected. Two writes to the key are necessary to change the value in the PWRMOD register: first write 0x4859, and then write 0xF27B. The PWRMOD register should then be written. A write to another register before writing to PWRMOD returns the protection to the lock state.

Code Example to Enter Power Saving Modes

```
SCB->SCR = 0x04;                  // sleepdeep mode
pADI_PWRCTL->PWRKEY = 0x4859;     // key1
pADI_PWRCTL->PWRKEY = 0xF27B;     // key2
pADI_PWRCTL->PWRMOD = 0x4;
asmwfi();
__asm void asmwfi()
{
                      nop
                      nop
                      nop
                      WFI
                      nop
                      nop
                      BX LR
}
```

CORTEX-M3 PROCESSOR

CORTEX-M3 PROCESSOR FEATURES

High Performance

- 1.25 DMIPS/MHz.
- Many instructions, including multiply, are single cycle.
- Separate data and instruction buses allow simultaneous data and instruction accesses to be performed.
- Optimized for single-cycle flash usage.

Low Power

- Low standby current.
- Core implemented using advanced clock gating so that only the actively used logic consumes dynamic power.
- Power saving mode support (sleep and deep sleep modes). The design has separate clocks to allow unused parts of the processor to be stopped.

Advanced Interrupt Handling

- The nested vectored interrupt controller (NVIC) supports up to 240 interrupts. The ADuCM360/ADuCM361 supports 38 of these. The vectored interrupt feature greatly reduces interrupt latency because there is no need for software to determine which interrupt handler to serve. In addition, there is no need to have software to set up nested interrupt support.
- The Cortex-M3 processor automatically pushes registers onto the stack at the entry interrupt and pops them back at the exit interrupt. This reduces interrupt handling latency and allows interrupt handlers to be normal C functions.
- Dynamic priority control for each interrupt.
- Latency reduction using late arrival interrupt acceptance and tail-chain interrupt entry.
- Immediate execution of a nonmaskable interrupt request for safety-critical applications.

System Features

- Support for bit-band operation and unaligned data access.
- Advanced fault handling features include various exception types and fault status registers.

Debug Support

- Serial wire debug interfaces (SW-DP).
- Flash patch and breakpoint (FPB) unit for implementing breakpoints. Limited to six active breakpoints.
- Data watchpoint and trigger (DWT) unit for implementing watchpoints, trigger resources, and system profiling. Limited to two active watchpoints.

CORTEX-M3 PROCESSOR OVERVIEW

The ADuCM360/ADuCM361contains an embedded ARM Cortex-M3 processor, Revision r2p0 (AT420). The ARM Cortex-M3 provides a high performance, low cost platform that meets the system requirements of minimal memory implementation, reduced pin count, and low power consumption while delivering outstanding computational performance and exceptional system response to interrupts.

CORTEX-M3 PROCESSOR OPERATION

Several Cortex-M3 components are flexible in their implementation. This section details the actual implementation of these components in the ADuCM360/ADuCM361.

Serial Wire Debug (SW/JTAG-DP)

The ADuCM360/ADuCM361 only supports the serial wire interface via the SWCLK and SWDIO pins. It does not support the 5-wire JTAG interface.

ROM Table

The ADuCM360/ADuCM361 implements the default ROM table.

Nested Vectored Interrupt Controller Interrupts (NVIC)

The Cortex-M3 processor includes a nested vectored interrupt controller (NVIC), which offers several features:

- Nested interrupt support
- Vectored interrupt support

- Dynamic priority changes support
- Interrupt masking

In addition, the NVIC has a nonmaskable interrupt (NMI) input.

The NVIC is implemented on the ADuCM360/ADuCM361, and more details are available in the System Exceptions and Peripheral Interrupts section.

Wake-Up Interrupt Controller (WIC)

The ADuCM360/ADuCM361 has a modified WIC, which provides the lowest possible power-down current. This feature is transparent to the user, and more details are available in the Power Management Unit section.

Note that if the part enters a power saving mode when servicing an interrupt, it can be woken up only by a higher priority interrupt source.

µDMA

The ADuCM360/ADuCM361 implements the ARM µDMA. More details are available in the DMA Controller section.

RELATED DOCUMENTS

- Cortex-M3 Revision r2p0 Technical Reference Manual (DDI0337)
- Cortex-M3 Errata Notice: Cortex-M3/Cortex-M3 with ETM (AT420/AT425)
- ARMv7-M Architecture Reference Manual (DDI0403)
- ARMv7-M Architecture Reference Manual Errata markup
- ARM Debug Interface v5 (IHI0031)
- PrimeCell µDMA Controller (PL230) Technical Reference Manual Revision r0p0 (DDI0417)

ADC CIRCUIT

ADC CIRCUIT FEATURES

- ADuCM360: two Σ-Δ ADCs (ADC0 and ADC1)
- ADuCM361: one Σ-Δ ADC (ADC1)
- Six fully differential inputs
- Eleven single-ended inputs

ADC CIRCUIT BLOCK DIAGRAM

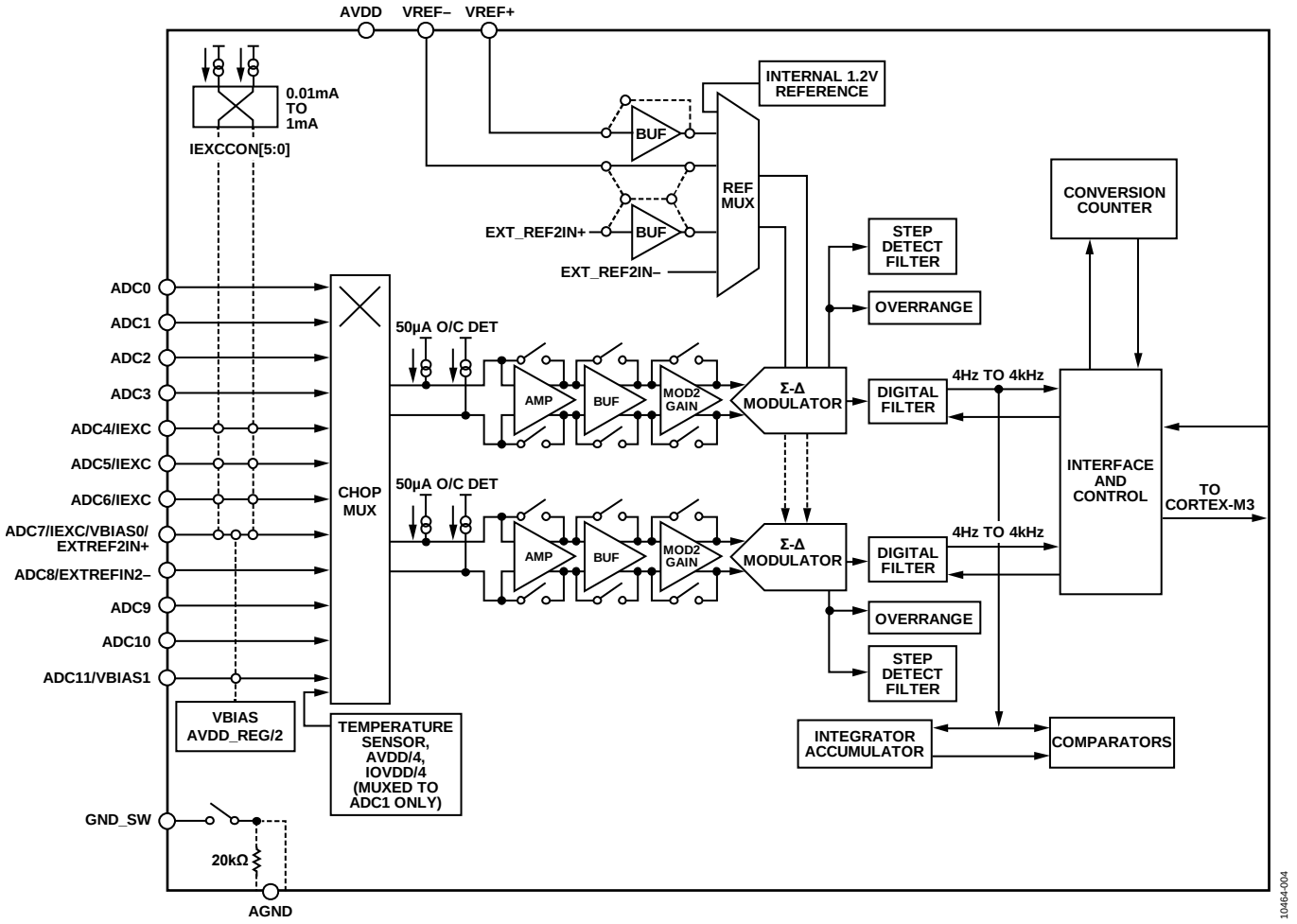

Figure 5. ADC Block Diagram

ADC CIRCUIT OVERVIEW

The ADuCM360 incorporates two independent multichannel Σ-Δ ADCs (ADC0 and ADC1).

The ADuCM361 incorporates one multichannel Σ-Δ ADC (ADC1).

Both ADCs are connected to a common input multiplexer of six fully differential inputs or 11 single-ended inputs. The device also contains a low noise, low drift internal band gap reference. The part accepts two external reference sources and provides the option of buffering these inputs. Other on-chip ADC features include dual programmable excitation current sources, diagnostic current sources, and a bias voltage generator of AVDD_REG/2 to set the common-mode voltage of an input channel. A low-side internal ground switch is provided that allows powering down an external sensor between conversions.

Both ADCs share the same input multiplexer. Each ADC has a separate input buffer and a low noise programmable gain amplifier (PGA) that allows small amplitude signals to be connected directly to the Σ-Δ modulator of each ADC. The analog input pins can be configured as fully differential input pairs or as single-ended pairs. In single-ended mode, the channel selected by the ADCCN in the ADCxCON register acts as the negative input channel reference to the ADC. The same channel can be used as a negative input for all other single-ended positive

inputs. These can be connected to ground. ADC1 also supports four internal channels: AVDD/4, IOVDD/4, temperature sensor, and DAC. The ADCs operate independently of each other, and each has its own individual Cortex-M3 processor interrupt sources and configuration register set.

For fast processing of the ADC inputs, it is possible to select a faster sinc2 filter to operate in parallel with the normal sinc3 or sinc4 filter on either ADC0 or ADC1. This filter outputs data every 2 ms and can be read continuously. Alternatively, the sinc2 filter can be used for fast detection of step changes on an ADC input channel. In addition, for higher precision at high sampling rates (500 Hz to 4 kHz), a sinc4 filter is provided.

If an error occurs during an ADC conversion or calibration routine (for example, a missing reference), the ADCxDAT registers read full scale for the particular gain setting and ADCxSTA[4] is set to 1 for an overflow error. ADCxDAT reports negative full scale for an underflow error, and again ADCxSTA[4] is set to 1.

On the ADuCM361, ADC0 is not available.

On the ADuCM361, ADC1 is identical to ADC1 of the ADuCM360.

ADC CIRCUIT OPERATION

Analog Input Channel Configuration

Both ADCs are connected to a common input multiplexer.

The ADCCP (ADCxCON[9:5]) and ADCCN (ADCxCON[4:0]) bits of the ADC control registers are used to configure the input signals to each ADC as follows:

- ADC0CON[9:5]: These bits select the positive input channel to ADC0. The bits can select any of the following input channels as the source: AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11 or AGND.
- ADC0CON[4:0]: These bits select the negative input channel to ADC0. The bits can select any of the following input channels as the source: AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11 or AGND.
- ADC1CON[9:5]: These bits select the positive input channel to ADC1. The bits can select any of the following input channels as the source: AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11 or any of the internal channels (that is, DAC output, temperature sensor channel, AVDD ÷ 4, or IOVDD ÷ 4).
- ADC1CON[4:0]: These bits select the negative input channel to ADC1. The bits can select any of the following input channels as the source: AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11 or any of the internal channels (that is, DAC output, temperature sensor channel, AVDD, or IOVDD).

The input channels can be buffered or unbuffered. In buffered mode, the selected input channel feeds into a high impedance input stage of the amplifier. This allows the ADC to tolerate significant source impedances and allows direct connection to resistive type sensors, such as RTDs and strain gauges. In unbuffered mode, this input buffer is bypassed and affords a wider input voltage range. The buffer is controlled by ADCxCON[17:14].

See the specifications in the ADuCM360/ADuCM361 data sheet for further details.

The ADuCM360 supports a mode of operation in which the PGA on both ADCs can drive the ADC modulator directly regardless of the gain setting, allowing the input buffers to be bypassed and powered down. Because each positive and negative input buffer consumes typically 35μA, this represents an I_{DD} saving of 140 μA if both ADCs are enabled. The ADuCM361 also supports such a mode of operation, but on ADC1 only. The time to a first valid (fully settled) result after an input channel switch is shown in Table 16, $t_{SETTLING}$.

Connecting the Same Input Channel to the Positive Input of Both ADCs (ADuCM360 Only)

The ADuCM360 has two independent ADCs and a fully flexible input multiplexer. This allows the positive inputs to both ADCs to be connected to the same input channel.

Most applications separate the input channels to each ADC. However, in some cases—for example, in industrial applications—it may be desirable to measure the same input on both ADCs to improve the safety integrity level (SIL) ratings.

If the PGA is enabled (gain ≥ 2), the ADC output value when both ADCs are sampling the same input is different compared with when just one ADC is sampling this input. The reason for the difference is due to the change in input current when the analog input is connected to both PGAs compared with when it is connected to just one.

When measuring an input channel with both ADCs, it is recommend that separate zero-scale and full-scale calibration values be used for this condition. These should be generated via the system calibration mode of the ADC.

PGA Gain Configuration

Both ADCs incorporate an on-chip programmable gain amplifier (PGA). The PGA can be programmed through eight different settings, resulting in a range of 1 to 128. The gain is controlled by the ADC0MDE[7:4] and ADC1MDE[7:4] control registers. The PGA allows signals of very small amplitude to be gained up while still maintaining low noise performance. In unipolar mode (based on a 1.2 V ADC reference), the input range varies from 0 V to 1.2 V using a gain of 1 or from 0 mV to 7.8125 mV using a gain of 128. In bipolar mode, this changes to 0 V to ±1.2 V with a gain of 1 and to 0 mV to ±7.8125 mV with a gain of 128.

When a gain of 1 is selected, the PGA is not enabled. The PGA is enabled only for gains of 2, 4, 8, 16, 32, 64, and 128. When the PGA is enabled, the PGA output voltage must not exceed 1 V. Therefore, when using a gain of 4, the maximum input voltage that can be applied is ±250 mV; when using a gain of 8, the maximum input voltage that can be applied is ±125 mV. For the full list of valid input voltage ranges for all ADCs, refer to the ADuCM360/ADuCM361 data sheet. To determine if the ADC input is about to exceed the 1 V PGA output threshold, the ADC comparator can be used. The following code example shows how to set the comparator threshold to 1 V. An interrupt is asserted if the threshold is exceeded. This code is valid for all ADC0 PGA gain settings from 2 to 128.

Code Example

```
void ADC0INIT(void)
{
        pADI_ADC0->MSKI = 0x5;                  // Enable ADC0 /rdy IRQ          + ADCxTHEX interrupt
          pADI_ADC0->PRO = 0x4;                 // Enable threshold detection
         pADI_ADC0->TH = 0x6AA2;                // Set threshold to 1 V max PGA output
                                                // Add rest of ADC init code after this
}
In the interrupt handler, add the following:
 uiADCSTA = pADI_ADC0->STA;                     // read ADC status register
   if ((uiADCSTA & 0x4) == 0x4)                 // Check for ADC0TH exceeded condition
  {
                ucADC0ERR = 4;                  // 1 V threshold exceeded
               pADI_ADC0->MSKI &= 0xFB;         // Disable threshold detection to clear this interrupt source
                pADI_ADC0->PRO = 0;             // Disable threshold detection to clear this interrupt source
 }

Optional—reenable in main loop:
                                                // Disable threshold detection to clear this interrupt source
                if (ucADC0ERR == 0x4)
                {
                        ucADC0ERR = 0;
                       pADI_ADC0->MSKI |= 0x4;
                       pADI_ADC0->PRO = 4;

                }
```

An optional extra gain of 2 can be added in the ADC modulator; that is, a gain of 2 can be added to the output of the PGA. The 1 V PGA output limitation remains, but this extra feature results in an additional 0.5 LSB rms bits of resolution.

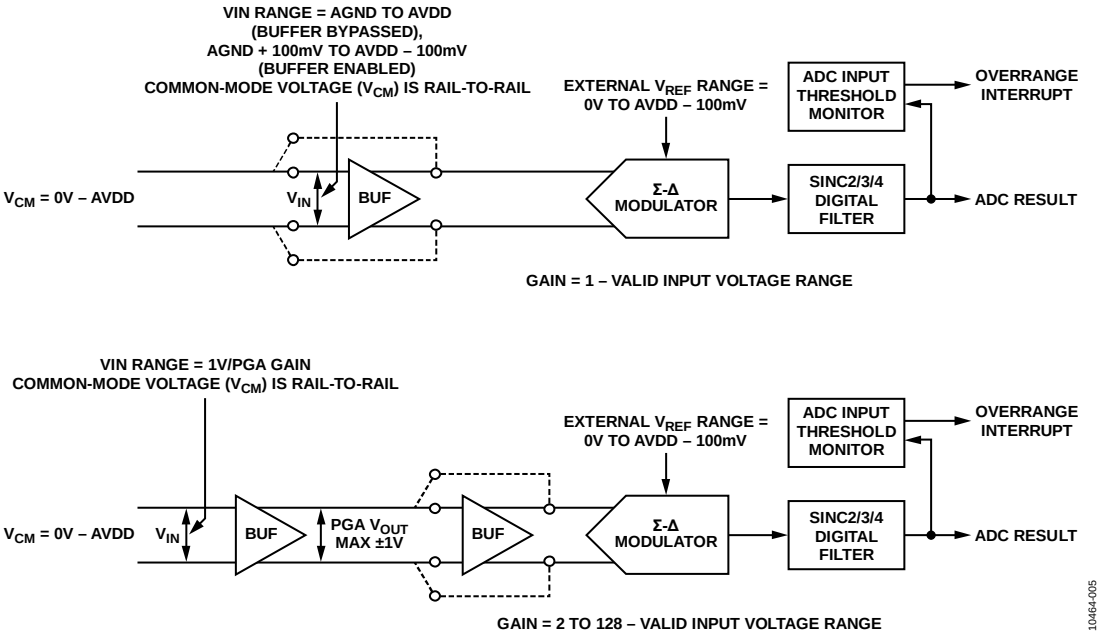

Figure 6. ADC Inputs

Table 14. Absolute Input Voltage and Common-Mode Voltage Ranges for All Gains

Gain	Absolute Input Voltage Range[1] (Buffers Off)	Absolute Input Voltage Range[1] (Buffers On)	V_{CM} Range (Input Buffers On)	V_{CM} Range (Input Buffers Off)
1	AGND to AVDD	AGND + 100 mV to AVDD – 100 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
2	0 V to 500 mV	0 V to 500 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
4	0 V to 250 mV	0 V to 250 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
8	0 V to 125 mV	0 V to 125 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
16	0 V to 62.5 mV	0 V to 62.5 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
16 + MOD2	0 V to 62.5 mV	0 V to 62.5 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
32	0 V to 22.18 mV	22.18 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
32 + MOD2	0 V to 15.625 mV	0 V to 15.625 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
64	0 V to 15.625 mV	0 V to 10.3125 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
64 + MOD2	0 V to 7.8125 mV	0 V to 7.8125 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]
128	0 V to 7.8125 mV	0 V to 3.98 mV	AGND + 100 mV to AVDD – 100 mV	AGND to AVDD[2]

[1] Absolute input voltage is the differential voltage range AIN+ to AIN–.
[2] If the ADC input buffers are bypassed, the common-mode voltage (V_{CM}) is configurable rail-to-rail. However, within 200 mV of the supply rails, the input current increases.

Bipolar/Unipolar Configuration

The analog inputs to the ADuCM360/ADuCM361 can accept either unipolar or bipolar input voltage ranges.

A bipolar voltage does not imply that the part can handle negative voltages with respect to ground. It simply means that the input range can vary above or below the common-mode voltage by the value of V_{REF} as long as the absolute input voltage range is not exceeded.

In unipolar mode, the analog input voltage range is reduced to the sum of the common-mode voltage and the positive input level; therefore, the positive input cannot go below the common-mode voltage.

For example, if ADC0 is configured for differential mode using an external 1.0 V reference source and the ADC negative input is set to 1.5 V (common mode = 1.5 V), the resulting valid input voltage range is 0.5 V to 2.5 V in bipolar mode and 1.5 V to 2.5 V in unipoloar mode. Furthermore, in unipolar mode, if the input voltage is a negative value (AIN– > AIN+), the ADCxDAT is clamped to 0x00000000 and ADCxSTA[4] (error bit) is set to 1.

Fully Differential and Single-Ended Inputs

Both ADCs support two voltage input types. In fully differential mode, the user can select from any input pair to both ADCs. The external circuit determines the common-mode voltage; therefore, the external circuit must be configured to ensure that the differential signal does not exceed the minimum/maximum absolute input voltage for that input. Alternatively, the user can configure either AIN7 or AIN11 as a common-mode output pin (V_{BIAS} mode) and then use the pin to bias the differential pair.

In single-ended mode, the negative input is selected by the ADCxCON[4:0] control bits. If the negative input pin is connected to ground and the PGA gain setting is 1, the input buffer must be bypassed. Otherwise, the negative input must be biased above the minimum input voltage value.

The PGA on both ADCs can drive the ADC modulator directly, regardless of the gain setting. Therefore, negative inputs can be connected directly to AGND.

Typical ADC Modes of Operation

The ADC can be configured to operate in one of three general modes of operation: conversion mode, calibration mode, or power-down mode.

ADC Conversion Mode

The ADC conversion mode can be used to initiate continuous conversions at a fixed rate or single conversions triggered by software.

- Single conversion: A single ADC conversion can be initiated in software by setting Bit 1 of the ADC0MDE/ADC1MDE registers. (ADCxMDE[2:0] = 010). After a single conversion is completed, the ADC returns to idle mode.
- Continuous conversion: Continuous conversion can be initiated in software by setting Bit 0 of the ADC0MDE/ADC1MDE registers. (ADCxMDE[2:0] = 001). Continuous conversion mode results in the ADCxDAT register being updated at the sampling rate selected by the ADCxFLT register.

When a conversion has completed in either mode, the ADCxRDY bit (ADCxSTA[0]) is asserted, indicating that the ADC result is present in the ADC0DAT, ADC1DAT, or STEPDAT register and is available for reading. ADCxRDY and the output of the sinc2 filter can be configured to flag an interrupt to the ARM Cortex-M3 processor. If an error occurs in the conversion due to an underrange or overrange error in the input voltage of either ADC, the error is set in the appropriate ADCxSTA[4] register.

The ADC comparator threshold bit (ADCxSTA[2]) should also be checked as part of the ADC interrupt handler to ensure that the 1 V PGA output limit has not been exceeded (see the PGA Gain Configuration section for more information).

ADC Calibration Mode

The ADC calibration mode removes ADC and system error where possible.

It is recommended that a system zero-scale and system full-scale calibration be performed using the required system PGA gain setting.

Before entering any calibration mode, the ADC being calibrated must first be in idle mode (ADCxMDE[2:0] = 011). In addition, the ADCxOF and ADCxGN registers can be written to only when the ADC is in idle mode.

- Internal zero-scale calibration.
  In this mode, an offset calibration is performed on any enabled ADC using an internally generated 0 V signal. The calibration is carried out at the user-programmed ADC settings; therefore, as with a normal single ADC conversion, it takes two to three ADC conversion cycles before a fully settled calibration result is ready. The calibration result is automatically written to the ADCxOF register of the respective ADC and ADCxSTA[5] is set to 1 to indicate the completion of the calibration command. After a reset of the part, the ADCxOF register is reloaded with the factory calibration value.
- Internal full-scale calibration (available only when gain = 1).
  In this mode, a gain calibration against an internal full-scale reference voltage is performed on all enabled ADCs. A gain calibration is a two-stage process and takes twice the time of an offset calibration. The calibration result is automatically written to the ADC gain register of the respective ADC, and ADCxSTA[5] is set to 1 to indicate the completion of the calibration command. The internal full-scale calibration works for gain = 1 only. After a reset of the part, the ADC gain register is reloaded with the factory calibration value.
- System zero-scale calibration.
  In this mode, a zero-scale calibration is performed on enabled ADC channels against an external zero-scale voltage driven at the ADC input pins. Usually, the selected channel is shorted externally. The calibration result is automatically written to the ADCxOF MMR of the respective ADC, and ADCxSTA[5] is set to 1 to indicate the completion of the calibration command. After a reset of the part, the ADCxOF register is reloaded with the factory calibration value.
- System full-scale calibration.
  In this mode, a full-scale calibration is performed using the enabled ADC channels against an external full-scale voltage driven at the ADC input pins. The ADC gain register is updated after a full-scale calibration sequence, and ADCxSTA[5] is set to 1 to indicate the completion of the calibration command. After a reset of the part, the ADC gain register is reloaded with the factory calibration value.

Note that the ADC gain and ADCOF registers can be modified by the user only when the ADC is in idle mode (ADCxMDE[2:0] = 011). The following separate ADC gain registers are provided for the ADCs when the reference source is the internal reference, external reference, or AVDD supply: ADCxINTGN, ADCxEXTGN, and ADCxVDDGN. Only the ADCxINTGN is loaded with a factory calibration value. The ADCxEXTGN and ADCxVDDGN registers must be updated by the user after a reset sequence.

ADC Power-Down Mode

Particularly between single conversions, the ADC should be powered down to reduce overall system power consumption.

In full power-down mode, the ADCs and the input amplifiers are fully powered off for maximum power reduction. In idle mode, the ADC is fully powered on but is held in reset. The part enters this mode after calibration or between single conversions. The current consumption is reduced relative to fully active mode.

OTHER ADC SUPPORT CIRCUITS

Voltage Reference Sources

Four reference sources are supported by the ADuCM360/ADuCM361: internal reference, External Reference 1, External Reference 2, and AVDD.

Internal Reference

The internal reference is a precision 1.2 V band gap, low noise, low drift reference. See the ADuCM360/ADuCM361 data sheet for more information.

External Reference 1

External Reference 1 is connected to the VREF+ and VREF− pins. An option to internally buffer the external reference is also provided. A separate buffer is provided for the positive and negative external reference inputs. If the negative input is ground, the negative input buffer should be bypassed.

External Reference 2

External Reference 2 can be connected to the AIN7/VBIAS0/IEXC/EXTREF2IN+ and AIN8/EXTREF2IN− pins. External Reference 2 offers the option of buffering the positive input to EXTREF2IN+ (see the description of ADCxCFG[1:0] in Table 29 for more details).

AVDD

The AVDD of the part can be selected as a reference source.

Reference Detection Circuit

An internal circuit for detecting an error on the external reference pins is available. The detection circuit checks for open-circuit conditions on the VREF+ and VREF− pins. It also checks that the differential voltage between VREF+ and VREF− is greater than 400 mV. If an error condition is detected, the error flag in the DETSTA[3] register is set.

If an external reference is selected, the on-chip reference voltage detection circuit flags an error if an open is detected on the VREF+ or VREF− pin or if the differential input reference voltage across VREF+/VREF− is below 400 mV. In normal operation, the sampling rate is configured via the ADCxFLT registers.

No reference detection circuit is available for the EXTREF2IN± pins.

Fast Response Output/Input Step Detection Circuit

On both ADCs, the output of the Σ-Δ modulator is fed into a sinc3 or sinc4 digital filter. However, for faster response times, a sinc2 filter can be selected to operate in parallel with the sinc3 or sinc4 filter of both ADCs, but the sinc3 and sinc4 filters cannot operate simultaneously. This sinc2 filter has a configurable update rate with a fastest speed of 2 ms and with 14 effective bits resolution. The output of the sinc2 filter is fed to a separate result register, STEPDAT.

The sinc2 filter can also be used in step detection mode for detecting step changes in the input signal. In this mode, the output of the sinc2 filter is compared with the two previous output values. If the difference between the latest value and the two previous values is greater than a predefined step threshold, a step detected flag is asserted in the DETSTA register and an interrupt is triggered. Alternatively, a step can be triggered by comparing the output of the sinc2 filter with the last output of the sinc3 or sinc4 filter. If the difference is greater than the configured threshold level, the step detected flag is asserted.

Two possible interrupt sources are provided with the sinc2 filter:

- Using the DMA controller and the associated ADC DMA interrupt; when a preconfigured number of sinc2 samples has been stored to memory, this interrupt flag is set.
- A separate interrupt flag is provided for the step detection filter (DETSTA[1]).

The DETCON register configures the sinc2 filter and step detection circuit.

Every output from the sinc2 filter can be read by polling DETSTA[0].

Step Detection Circuit

As Figure 7 shows, the step detection filter uses 256 × ADC clocks for each sample period. The output at the end of each filter period is the average value of each of the 256 samples. The step detection hardware compares each output with the three previous outputs. If the difference between the current output and any of the previous outputs is greater than the selected threshold level, a step is detected. The oversampling rate of the sinc2 filter is configurable for 256/512/768/1024 × ADC clocks, resulting in sampling periods of 2 ms/4 ms/6 ms/8 ms, respectively, by setting DETCON[1:0].

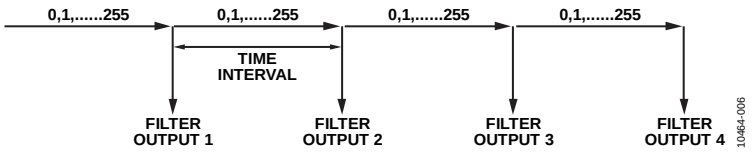

Figure 7. ADuCM360/ADuCM361 Step Detection Timing Diagram

As Figure 8 shows, the difference between Level 1 and Level 2 is greater than the selected threshold level:

- If Output 4 − Output 3 = Result < Threshold, then no step detected.
- If Output 4 − Output 2 = Result > Threshold, then step detected.
- If Output 4 − Output 1 = Result > Threshold, then step detected.

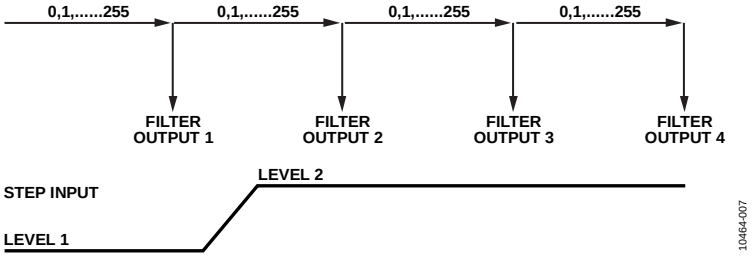

Figure 8. Step Detection Circuit

The key features of the step detection circuit are as follows:

- This step detection filter operates at a sampling rate of 500 Hz by default, resulting in a 2 ms output rate. This can be extended to 4 ms, 6 ms, or 8 ms by using the filter control register.
- The step threshold is programmable using a 10-bit register, STEPTH.
- The minimum threshold setting is 0.05% of full scale.
- The output resolution is 14 bits noise free.
- A separate step detection interrupt and STEPDAT register are provided for the output of this filter.
- The minimum delay between a step condition and the assertion of the step flag by the filter is 6 ms, assuming that the sampling period is 2 ms. This delay is calculated as 3 × sampling period.
- The slowest slew rate that the filter can handle is the sampling period for one output. For example, if the step detection range is selected as a 0.25% or greater step, the slew rate of the step must not be greater than 2 ms when the sampling rate is selected as 500 Hz.

SINC2 Filter Data Output Register—STEPDAT

The output of the sinc2 filter is provided by the STEPDAT register. The gain is automatically accounted for in the STEPDAT output value.

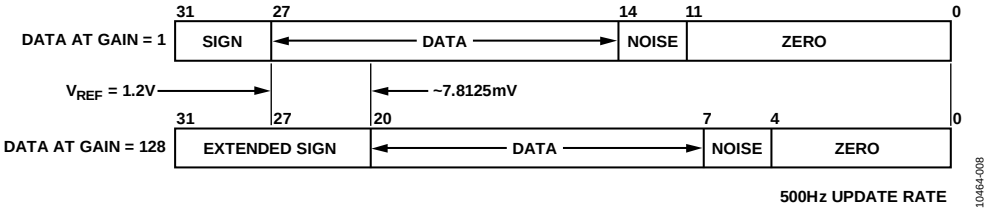

Figure 9. ADC Sinc2 Data Register 500 Hz Update Rate

ADC Comparator and Accumulator

Both ADCs have a digital comparator and accumulator for postprocessing ADC result values.

Every ADC result can be compared with a preset threshold level (ADCxTH) as configured via ADCxPRO[3:2]. An interrupt is generated if the absolute (sign independent) value of the ADC result is greater than the preprogrammed comparator threshold level. An extended function of this comparator function allows user code to configure a threshold counter (ADCxTHV) to monitor the number of ADC results that has occurred above or below the preset threshold level. Again, an ADC interrupt is generated when the threshold counter reaches a preset value (ADCxRCR).

A 32-bit accumulator (ADCxACC) function can be configured (ADCxPRO[5:4]), allowing the ADC to add (or subtract) multiple ADC sample results. User code can read the accumulated value directly (ADCxACC) without any further software processing. The accumulator also comes with a comparator option, allowing a separate interrupt to be generated if the accumulator value exceeds a user-defined threshold (ADCxACC > ADCxATH).

Diagnostic Current Sources

To detect a connection failure to an external sensor, the ADuCM360/ADuCM361 incorporates a 50 A constant (burnout) current source on the selected analog input channels to both ADCs. These can be switched on or off via the ADCxCON[11:10] bits. Diagnostic current sources are accurate to ±10%.

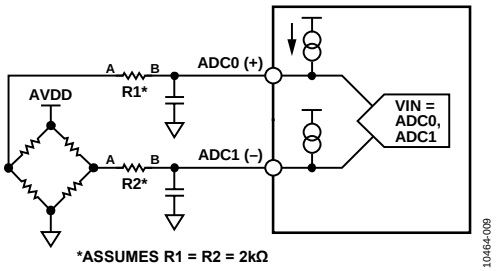

Figure 10. Example Circuit Using Diagnostic Current Sources

Table 15. Example Scenarios for Using Diagnostic Current Sources

Diagnostic Test		Normal Result	Fault Result	Detected Measurement for Fault
Register Setting	**Description**			
ADCxCON[11:10] = 00	Convert AINx/AINy as normal with diagnostic currents disabled.			
ADCxCON[11:10] = 01	Enable a 50 µA diagnostic current source on AINx by setting ADCxCON[11:10] = 01. Convert AINx and AINy.	ADC changes by ΔV = +50 µA × R1. For example, ~100 mV for R1 = 2 kΩ.	Short circuit between AINx and AINy. Short circuit between R1_a and R1_b.	ADC reading ≈ 0 V, regardless of PGA setting.
	Convert AINx in single-ended mode with diagnostic currents enabled.	Expected voltage on AINx.	AINx open circuit or R1 open circuit.	ADC reading = positive full scale, even on the lowest PGA setting.
ADCxCON[11:10] = 11	Enable a 50 µA diagnostic current source on both AINx and AINy by setting ADCxCON[11:10] = 11. Convert AINx and AINy.	ADC reading changes by ΔV = 50 µA × (R1 − R2), that is, ~10 mV for 10% tolerance.	R1 does not match R2.	ADC reading > 10 mV of expected value.

Bias Voltage (V_{BIAS}) Generator Circuit

The ADuCM360/ADuCM361 provides a circuit for generating a bias voltage or common-mode voltage of AVDD_REG/2. The AVDD_REG/2 common-mode voltage can be connected to one of two analog pins, AIN7 or AIN11, using ADCxCFG[10:8]. Therefore, instead of connecting it internally to the output of the PGA, the user can connect this voltage externally.

This functionality is particularly useful for thermocouple setups. If AIN7 is used as the V_{BIAS} generator, it should be connected to the thermocouple side of any input filtering.

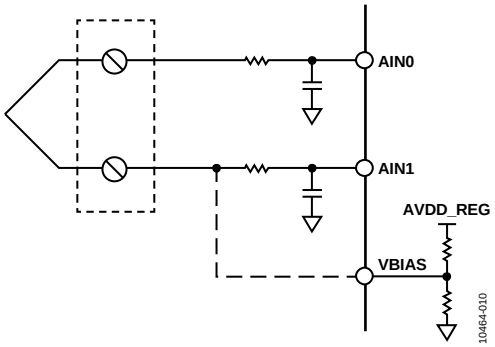

Figure 11. Using V_{BIAS} Voltage to Set Thermocouple Common-Mode Voltage

Note that only one analog pin can be configured as a V_{BIAS} output. ADCxCFG[10:8] are used to select the VBIAS output pin.

The power-up time of the V_{BIAS} output is dependent on the load capacitance placed on the pin. To reduce power-up times, a V_{BIAS} boost mode is provided that allows the V_{BIAS} generator circuit to output more current to the VBIAS pin, thus reducing the power-up time. ADCxCFG[13] increases the default output current by a factor of 30. When the boost bit is enabled, the V_{BIAS} generator circuit consumes up to 150 µA of extra current. When charging an external 10 nF load capacitor, the power-up time is 30 ms with boost mode disabled and is 1 ms with boost mode enabled.

OTHER ADC DETAILS

Digital Filter Option

The ADuCM360/ADuCM361 main ADC digital filter settings are controlled by the ADC0FLT and ADC1FLT registers. The decimation factor of the sinc3 or sinc4 filter is set using ADCxFLT[6:0]. See Table 16 and Table 17 for further details on the decimation factor values. The range of operation of the sinc3 or sinc4 filter (SF) word depends on whether the chop function is enabled. With chopping disabled, the minimum SF word allowed is 1 and the maximum SF word allowed is 127, resulting in an ADC throughput range of 3.5 Hz to 3.906 kHz. With chopping enabled, the top frequency is 1.262 kHz. For more information about how to calculate the ADC sampling frequency based on the value programmed in ADCxFLT[6:0], refer to the description of the ADCxFLT register in the ADC Circuit Memory Mapped Registers section.

For sampling frequencies of 500 Hz or greater, the sinc4 filter option should be enabled by setting ADCxFLT[12] = 1.

For sampling frequencies of less than 500 Hz, the sinc3 option should be enabled by clearing ADCxFLT[12] = 0.

ADCxFLT[7] provides the option of adding an extra notch at 1.2 × frequency of the main notch. This feature is most commonly used for simultaneous 50 Hz/60 Hz rejection, where 50 Hz is the frequency of the main filter notch.

A sinc2 filter option that can operate parallel to the sinc3 or sinc4 filter on ADC0 or ADC1 is also available. The sinc2 filter can be used as a step detection filter or for faster output measurements (but with lower resolution).

To use the sinc2 as a faster filter for ADC0 or ADC1, set DETCON[7] = 1.

Table 16. ADC Conversion Rates and Settling Times

Sinc4 Enabled ADCxFLT[12]	Chop Enabled ADCxFLT[15]	Averaging Factor (AF) ADCxFLT[11:8]	Running Average ADCxFLT[14]	f_{ADC} Normal Mode	$t_{SETTLING}$[1]
No (0)	No (0)	0	No (0)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{3}{f_{ADC}}$
No (0)	No (0)	0	Yes (1)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{4}{f_{ADC}}$
No (0)	No (0)	>0	No (0)	$\frac{125{,}000}{[SF+1]\times 16\times[3+AF]}$	$\frac{1}{f_{ADC}}$
No (0)	No (0)	>0	Yes (1)	$\frac{125{,}000}{[SF+1]\times 16\times[3+AF]}$	$\frac{2}{f_{ADC}}$
No (0)	Yes (1)	Any value	Any value	$\frac{125{,}000}{[SF+1]\times 16\times[3+AF]+3}$	$\frac{2}{f_{ADC}}$
Yes (1)	No (0)	0	No (0)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{4}{f_{ADC}}$
Yes (1)	No (0)	0	Yes (1)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{5}{f_{ADC}}$
Yes (1)	Yes (1)	0	Any value	$\frac{125{,}000}{[SF+1]\times 16\times 4\times+3}$	$\frac{2}{f_{ADC}}$

[1] An additional time of approximately 256 µs per ADC is required before the first ADC result is available.

Table 17. Allowable Combinations of SF and AF

	AF					
SF	**0**	**1:2**	**3:7**	**8:9**	**10:14**	**15**
0:123	Yes	Yes	Yes	Yes	Yes	Yes
124	Yes	Yes	Yes	Yes	Yes	Only if notch 2 on
125	Yes	Yes	Yes	Only if notch 2 on	No	No
126	Yes	Only if notch 2 on	No	No	No	No
127	Yes	No	No	No	No	No

ADC Chopping

The ADCs on the ADuCM360/ADuCM361 implement a chopping scheme whereby the ADC repeatedly reverses its inputs. Therefore, the decimated digital output values from the sinc3 or sinc4 filter have a positive and negative offset term associated with them. This results in the ADC including a final summing stage that sums and averages each value from the filter with previous filter output values. This new value is then sent to the ADC data MMR. This chopping scheme results in excellent dc offset and offset drift specifications and is extremely beneficial in applications where drift and noise rejection are required.

Noise/Resolution Table

The noise resolution figures of both ADCs are available in the ADuCM360/ADuCM361 data sheet.

Simultaneous Sampling

The ADuCM360 has two identical ADCs. These can be configured for simultaneous sampling. The filter settings of ADC1 (in ADC1FLT) are then automatically applied to ADC0.

To enable simultaneous sampling, follow these configuration steps:

- Configure ADC0MDE and ADC1MDE.
- Configure ADC1FLT.
- Configure ADC0CON and ADC1CON as required but, ensure Bit 19 in both cases = 0
- Set ADCCFG[15] = 1.

Both ADCs will then start converting at the same time and output data at the same rate.

Internal Channels

The ADuCM360/ADuCM361 has four internal channel options:

- Temperature sensor
- DAC monitor
- AVDD ÷ 4
- IOVDD ÷ 4

Temperature Sensor Settings

The ADuCM360/ADuCM361 provides a voltage output from an on-chip band gap reference that is proportional to the absolute temperature on ADC1. This voltage output can also be routed through the front end of the ADC1 multiplexer (effectively, an additional ADC channel input), facilitating an internal temperature sensor channel that measures die temperature. The internal temperature sensor is not designed for use as an absolute ambient temperature calculator. Its intended use is as an approximate indicator of the temperature of the ADuCM360/ADuCM361 die.

An ADC temperature sensor conversion differs from a standard ADC voltage. The ADC performance specifications do not apply to the temperature sensor.

The temperature sensor settings are as follows:

- Enable the temperature sensor on ADC1; set ADC1CON[9:0] = 0xBC211.
- PGA enabled. It is suggested to use gain = 2.
- ADCMDE = 0x11.

To calculate the die temperature, use the following formula:

$$T - T_{REF} = (V_{ADC} - V_{TREF}) \times K$$

where:
T is the temperature result.
T_{REF} is 25°C.
V_{ADC} is the average ADC result from two consecutive conversions.
V_{TREF} is 82.1 mV, which corresponds to T_{REF} = 25°C as described in the ADuCM360/ADuCM361 data sheet.
K is the gain of the ADC in temperature sensor mode as determined by characterization data, K = 4 °C/mV.

This corresponds to 1/V TC as described in the ADuCM360/ADuCM361 data sheet.

Using the default values from the ADuCM360/ADuCM361 data sheet without any calibration, the equation becomes

$$T - 25°C = (V_{ADC} - 82.1) \times 4$$

where:
V_{ADC} is in millivolts.

Check the latest version of the ADuCM360/ADuCM361 data sheet for the most up to date figures.

For increased accuracy, perform a single-point calibration at a controlled temperature value. For the calculation shown without calibration, (T_{REF}, V_{TREF}) = (25°C, 82.1 mV). The idea of a single-point calibration is to use other known values (T_{REF}, V_{TREF}) to replace the common values (25°C, 82.1 mV) for every part. For some users, it is not possible to get such a known pair.

DAC Monitor Settings

The DAC monitor settings are as follows:

- Gain = 1
- 0x1 pADI_ADC1->MDE = 0x1
- pADI_ADC1->CON = 0xBC18F, AIN+ = DAC, AIN− = AGND

AVDD/4 Settings

The AVDD/4 settings are as follows:

- Select DAC as AIN− and set the DAC output to 600 mV
- pADI_DAC->DACCON = 0x410
- pADI_ADC1->DAT = 0x8000000
- pADI_ADC1->CON = 0xBC1AC

- pADI_ADC1->MDE = 0x1
- AVDD supply = (ADC voltage + DAC) × 4

Note that the output must be measured relative to the DAC output and not to AGND for the channel to work properly.

IOVDD/4 Settings

The IOVDD/4 settings are as follows:

- Select DAC as AIN− and set the DAC output to 600 mV
- pADI_DAC->DACCON = 0x410
- pADI_ADC1->DAT = 0x8000000
- pADI_ADC1->CON = 0xBC1CC
- pADI_ADC1->MDE = 0x1
- AVDD supply = (ADC voltage + DAC) × 4

Note that the output must be measured relative to the DAC output and not to AGND for the channel to work properly.

ADC DMA Details

A direct memory access function is available for the ADC0, ADC1, and sinc2 outputs. This function helps to reduce the processing overhead of handling multiple ADC conversion interrupts.

Two separate DMA functions are provided for the ADCs:

- A DMA write from memory to the ADC control registers. Note that this function is available only for ADC0 and ADC1 and is not available for the sinc2 output.
- A DMA write from the ADC0DAT, ADC1DAT, or STEPDAT register directly into memory.

See the Example Code: DMA Configuration for Moving ADC0 Results Directly to Memory section and the Example Code: DMA Configuration for Setting the ADC0 Control Registers Using DMA section for a list of steps and example code for setting up the DMA controller.

Example Code: DMA Configuration for Moving ADC0 Results Directly to Memory

```
pADI_ADC0->MDE = 0x00000003;                                    // Allow writing to the calibration
registers by placing ADC in idle mode
pADI_ADC0->CON = 0x80001;                                       // Enable ADC0, AIN0/AIN1 input
channels, Internal reference
pADI_ADC0-> ADCCFG = 0x00;                                      // Enable input buffers, disable
ground switch
pADI_ADC0->FLT = 0x7d;                                          // 50Hz sampling rate, chop off
Dma_Init();                                                     // Init DMA structures
pADI_ADCDMA-> ADCDMACON = 0x0000003;                            // Enable ADC0 DMA channel for ADC
results to memory
ADC0DMAREAD(uxADC0Data, 32);                                    // Setup DMA control registers
NVIC_EnableIRQ(DMA_ADC0_IRQn);                                  // Enable DMA ADC0 Interrupt
pADI_DMA->DMAENSET = 0x200;                                     // Select ADC0 channel in DMA
controller
pADI_ADC0->MDE = 0x21;                                          // G = 4, continuous conversions


void ADC0DMAREAD(unsigned long *pucRX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;                                                // Common configuration of all the
descriptors used here
    Desc.ctrlCfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
```

```
    Desc.ctrlCfg.Bits.dst_prot_ctrl     = 0x0;
    Desc.ctrlCfg.Bits.src_size          = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size          = DSTSIZE_WORD;

                                                        // RX Primary Descriptor
     Desc.srcEndPtr                     = (unsigned int)&ADC0DAT;
    Desc.destEndPtr                     = (unsigned int)(pucRX_DMA + iNumRX - 0x1); //
    Desc.ctrlCfg.Bits.n_minus_1         = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc           = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc           = INC_WORD;
    *Dma_GetDescriptor(DMA_REQ_ADC0,FALSE) = Desc;

}
void Dma_Init(void)
{
                                                        // Clear all the DMA descriptors
(individual blocks will update their descriptors
   memset(dmaChanDesc,0x0,sizeof(dmaChanDesc));

                                                        // Set up the DMA base pointer.
pADI_DMA->DMAPDBPTR = (unsigned int) &dmaChanDesc;

                                                        // Enable the  µDMA (Write only reg)
pADI_DMA->DMACFG = 0x1;
}
DmaDesc * Dma_GetDescriptor(unsigned int iChan,boolean bAlternate)
{
    if (iChan < DMACHAN_NUM)
        {
        if (bAlternate)
            return &(dmaChanDesc[iChan + 12]); //DMA_ALT_OFFSET]);
        else
            return &(dmaChanDesc[iChan ]);
        }
    else
        return 0x0;
}
void DMA_ADC0_Int_Handler()
{
                      unsigned char n = 0;
                      NVIC_DisableIRQ(DMA_ADC0_IRQn);      // Clear Interupt source
   ucDMA_ADC0_COMPLETE = 1;
}
```

Example Code: DMA Configuration for Setting the ADC0 Control Registers Using DMA

```
unsigned long uxADC0SETUP[] = {0x1,                     // ADC0MSKI
0x80001,                                                // ADC0CON
0x0000,                                                 // ADC0OF
```

```
0x5555,                                                 // ADC0INTGN
0x5555,                                                 // ADC0EXTGN
0x5555,                                                 // ADC0VDDGN
0x00,                                                   // ADCxCFG
0x3,                                                    // ADC0FLT
0x113};                                                 // ADC0MDE    - G = 4 Not sure what
you are asking here (Just wanted to make sure that the placement was okay. Also can we add a
space before and after the equal sign?) OK


ucDMA_ADC0_COMPLETE = 0;                                // Flag to indicate DMA complete -
set in IRQ handler
pADI_ADC0->MDE;                                         // Allow writing to the calibration
registers by placing ADC in idle mode
Dma_Init();                                             // Init DMA structures
pADI_ADCDMA-> ADCDMACON = 0x0000002;                            // Enable ADC0 DMA channel
for moving array contents in memory into ADC0 control registers
ADC0DMAWRITE(uxADC0SETUP, 9);                                   // Setup DMA control
registers
NVIC_EnableIRQ(DMA_ADC0_IRQn);                                  // Enable DMA ADC0 Interrupt
pADI_DMA->DMAENSET = 0x200;                                     // Select ADC0 channel in DMA
controller
pADI_TM1->LD = 0x28;                                            // On pre-release silicon,
timer overflow required to trigger DMA write
pADI_TM1->CON = 0x18;


void ADC0DMAWRITE(unsigned long *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                                // Common configuration of
all the descriptors used here
    Desc.ctrlCfg.bits.cycle_ctrl        = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst     = 0x0;
    desc.ctrlcfg.bits.r_power           = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl     = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl     = 0x0;
    Desc.ctrlCfg.Bits.src_size          = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size          = DSTSIZE_WORD;
                                                                // TX Primary Descriptor
Desc.srcEndPtr                          = (unsigned long)(pucTX_DMA + iNumRX - 0x1);
Desc.destEndPtr                         =  (unsigned long)&ADC0MDE;
Desc.ctrlCfg.Bits.n_minus_1             = iNumRX - 0x1;
Desc.ctrlCfg.Bits.src_inc               = INC_WORD;
Desc.ctrlCfg.Bits.dst_inc               = INC_WORD;
*Dma_GetDescriptor(DMA_REQ_ADC0,FALSE) = Desc;
}
```

For further details, see the DMA Controller section of this document.

ADC Data Register

The ADCxDAT register output format allows the user to easily calculate the voltage measured by the ADC.

In bipolar mode, the equation is as follows:

$(V_{REF})/2^{28}) \times ADCxDAT$ (signed results)

In unipolar mode, the equation is as follows:

$(V_{REF})/2^{28}) \times ADCxDAT$ (unsigned results, positive values only)

The gain value is automatically adjusted by the ADC hardware.

Each ADC gives a 24-bit conversion result. The data is twos complement or unipolar, based on the ADCCODE bit (ADCxCON[18]). Reading the ADCxDAT MMR clears any RDY flags that are active.

Although the ADC is nominally 24 bits, the noise places a limit of about 1.05 μV rms at a 3.53 Hz update and a gain of 1. This results in a useful signal range of ±1.2 V to 1.05 μV, which equates to approximately 2,285,000:1, or 21 bits. The result is placed in Bits[27:7] of ADCxDAT.

At a gain of 128, the rms noise is given as 52 nV at a 3.53 Hz update rate. This results in a useful signal range of about 7.8125 mV to 42 nV, which equates to approximately 300,000:1, or 18.5 bits. The result is placed in Bits[20:3] of ADCxDAT.

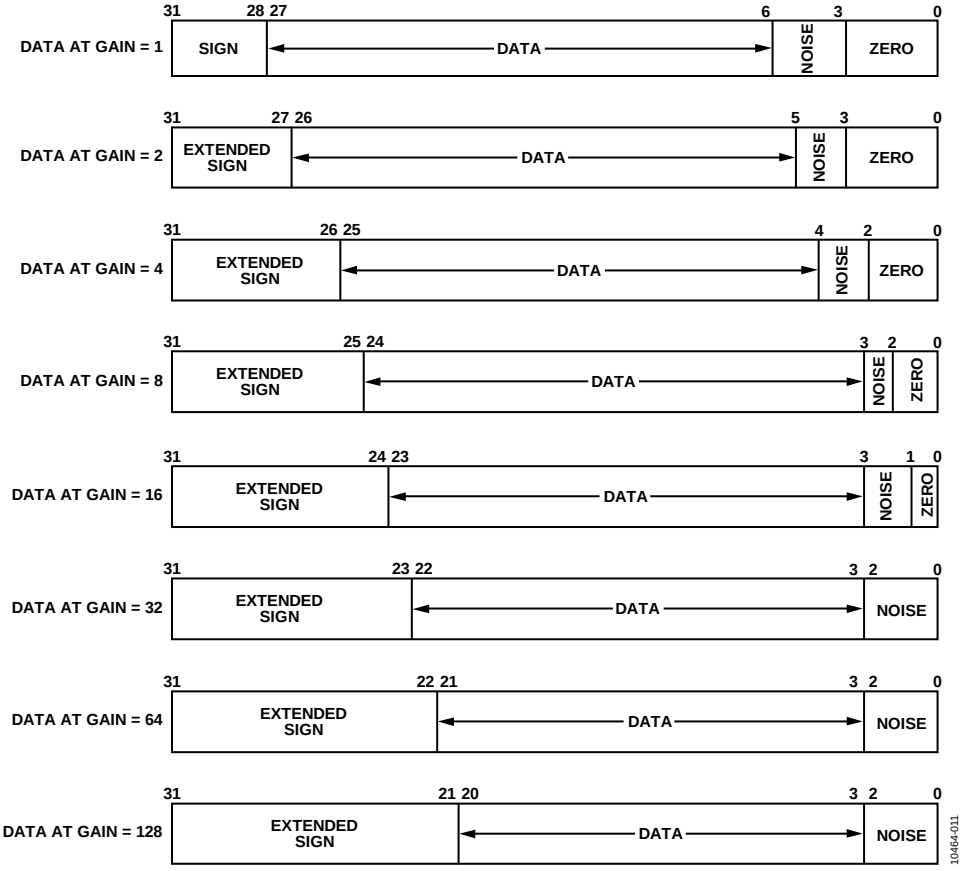

Figure 12. ADC Data Register 3.53 Hz Update Rate

For a faster update rate, such as 50 Hz, the noise figure is worse by nearly two bits (see Figure 13).

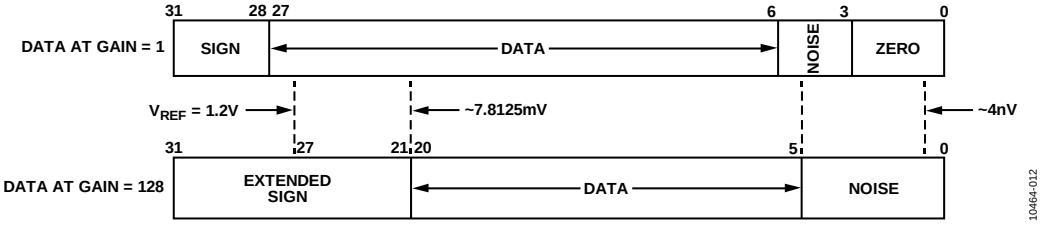

Figure 13. ADC Data Register 50 Hz Update Rate

There is still significant space at the bottom end if the noise performance improves or if additional filtering is applied.

The ADC does not write new data into the ADCxDAT MMRs if the relevant RDY bits are set. However, this does not apply if the ADuCM360/ADuCM361 is in power-down mode with DMA enabled, in which case ADCxDAT always contains the most recent ADC data.

The following should be noted:

- The error bit, ADCxSTA[4], can be used for detecting gross (that is, >30%) ADC overflow or underflow conditions.
- See the PGA Gain Configuration section for details about the 1 V output voltage limit when the PGA is enabled.
- If RDY is low, there is no guarantee that the ADCxDAT MMRs will be stable if they are read.
- If the ADCxRCR counter is active, the data registers are not updated until RDY is set (that is, when the ADCxRCR counter reaches the programmed limit). This is not true if the processor is off, in which case the ADCxDAT MMR(s) contains the most recent ADC result(s) when the processor wakes up.

Excitation Current Source

The ADuCM360/ADuCM361 contains two matched software configurable current sources. These excitation currents are sourced from AVDD. They are individually configurable to give a current range of 10 µA to 1 mA. The current output value is configured via IEXCDAT. These current sources can be used to excite an external resistive bridge or RTD sensors. The IEXCCON MMR controls the excitation current sources. IEXCCON[2:0] configure the output pin for Excitation Current Source 0. Similarly, IEXCCON[5:3] configure the output pin for Excitation Current Source 1.

It is also possible to configure both excitation current sources to output current to a single output pin, effectively doubling the output current by configuring both excitation current sources for the same pin. This allows up to 2 mA of output current on a single excitation pin. For example, setting IEXCCON = 0x64 and IEXCDAT = 0x3E results in a 2 mA output current on the AIN4 pin.

The excitation current source design by default uses an internal reference resistor. When using the internal reference resistor, the typical output current drift performance is 200 ppm/°C. By using a low drift external 150 kΩ reference resistor and setting IEXCCON[6] = 0, the drift in the output current value can be reduced.

ADC CIRCUIT MEMORY MAPPED REGISTERS

Table 18. ADC0 Register Summary (ADuCM360 Only)

Address	Name	Description	Access	Default
0x40030000	ADC0STA	ADC status register	R	0x00
0x40030004	ADC0MSKI	Interrupt control register	RW	0x00
0x40030008	ADC0CON	Main control register	RW	0x3038C
0x4003000C	ADC0OF	Offset calibration register	RW	Set in production testing
0x40030010	ADC0INTGN	Gain calibration register when using internal reference	RW	Set in production testing
0x40030014	ADC0EXTGN	Gain calibration register when using external reference	RW	Set in production testing
0x40030018	ADC0VDDGN	Gain calibration register when using AVDD as the ADC reference	RW	0x5555
0x4003001C	ADC0CFG	Control register for the V_{BIAS} voltage generator, ground switch, and external reference buffer	RW	0x0000
0x40030020	ADC0FLT	Filter configuration register	RW	0x007D
0x40030024	ADC0MDE	Mode control register	RW	0x0003
0x40030028	ADC0RCR	Number of ADC0 conversions before an ADC interrupt is generated	RW	001
0x4003002C	ADC0RCV	Current number of ADC0 conversion results	R	0x0000
0x40030030	ADC0TH	Sets the threshold	RW	0x0000
0x40030034	ADC0THC	Number of cumulative ADC0 conversion result readings above ADC0TH that must occur	RW	0x01
0x40030038	ADC0THV	8-bit threshold exceeded counter register	R	0x00
0x4003003C	ADC0ACC	32-bit accumulator register	R	0x00000000
0x40030040	ADC0ATH	Holds the threshold value for the accumulator comparator	RW	0x00000000
0x40030044	ADC0PRO	Configuration register for postprocessing of ADC0 results	RW	0x00
0x40030048	ADC0DAT	Conversion result register	R	0x00000000

Table 19. ADC1 Register Summary (ADuCM360 and ADuCM361)

Address	Name	Description	Access	Default
0x40030080	ADC1STA	ADC status register	R	0x00
0x40030084	ADC1MSKI	Interrupt control register	RW	0x00
0x40030088	ADC1CON	Main control register	RW	0x303FF
0x4003008C	ADC1OF	Offset calibration register	RW	Set in production testing
0x40030090	ADC1INTGN	Gain calibration register when using internal reference	RW	Set in production testing
0x40030094	ADC1EXTGN	Gain calibration register when using external reference	RW	Set in production testing
0x40030098	ADC1VDDGN	Gain calibration register when using AVDD as the ADC reference	RW	0x5555
0x4003009C	ADC1CFG	Control register for the V_{BIAS} voltage generator, ground switch, and external reference buffer	RW	0x0000
0x400300A0	ADC1FLT	Filter configuration register	RW	0x007D
0x400300A4	ADC1MDE	Mode control register	RW	0x0003
0x400300A8	ADC1RCR	Number of ADC1 conversions before an ADC interrupt is generated	RW	0x0001
0x400300AC	ADC1RCV	Current number of ADC1 conversion results	R	0x0000
0x400300B0	ADC1TH	Sets the threshold	RW	0x0000
0x400300B4	ADC1THC	Number of cumulative ADC1 conversion result readings above ADC1TH that must occur	RW	0x01
0x400300B8	ADC1THV	8-bit threshold exceeded counter register	R	0x00
0x400300BC	ADC1ACC	32-bit accumulator register	R	0x00000000
0x400300C0	ADC1ATH	Holds the threshold value for the accumulator comparator	RW	0x00000000
0x400300C4	ADC1PRO	Configuration register for postprocessing of ADC1 results	RW	0x00
0x400300C8	ADC1DAT	Conversion result register	R	0x00000000

Table 20. ADCSTEP Register Summary

Address	Name	Description	Access	Default
0x400300E0	DETCON	Control register for reference detection and the step detection filter	RW	0x0000
0x400300E4	DETSTA	Status register for detection	R	0x00
0x400300E8	STEPTH	Threshold for step detection filter	RW	0x0000
0x400300EC	STEPDAT	Offers coarse data from the output of the step detection filter	R	0x00000000
0x400300F8	ADCDMACON	ADC DMA mode configuration register	RW	0x0000

Table 21. Other ADC Register Summary

Address	Name	Description	Access	Default
0x40008840	REFCTRL	Internal reference control register	RW	0x0000
0x40008850	IEXCCON	Controls the on-chip excitation current sources	RW	0xC0
0x40008854	IEXCDAT	Sets the output current setting for both excitation current sources	RW	0x06

ADC Registers

ADC Status Registers

Address: 0x40030000, Reset: 0x00, Name: ADC0STA (ADuCM360 Only)

Address: 0x40030080, Reset: 0x00, Name: ADC1STA (ADuCM360 and ADuCM361)

Table 22. Bit Descriptions for ADCxSTA

Bits	Name	Description
7:6	Reserved	
5	ADCxCAL	ADC calibration status bit. Set to 1 when an ADC calibration is complete. Cleared to 0 after reading ADCxSTA.
4	ADCxERR	ADC conversion error status bit. Set to 1 when an underrange or overrange error occurs. Cleared to 0 after reading ADCxSTA.
3	ADCxATHEX	ADC accumulator comparator threshold status bit. Valid only if the ADC accumulator is enabled via the ADCxPRO register. Set to 1 when the ADC accumulator value in ADCxACC exceeds the threshold value programmed in the ADC comparator threshold register, ADCxATH. Cleared to 0 when the value in ADCxACC does not exceed the value in ADCxATH.
2	ADCxTHEX	ADC comparator threshold. Valid only if the ADC comparator is enabled via the ADCxPRO register. Set to 1 if the absolute value of the ADC conversion result exceeds the value written in the ADCxTH register. If the ADC threshold counter is used (ADCxRCR), this bit is set only when the specified number of ADC conversions equals the value in the ADCxTHV register. Otherwise, this bit is cleared. This bit is cleared by hardware when reconfiguring the ADC. See Code Example for more details
1	ADCxOVR	DC overrange bit. Set to 1 if the overrange detection function is enabled via the ADCxPRO register. This bit is set by hardware if the ADC input is grossly (>30% approximate) overrange. This bit is updated every 500 µs. Cleared to 0 by software when ADCxPRO[1] is cleared or if the ADC gain is changed in ADCxMDE.
0	ADCxRDY	Valid conversion result. Set to 1 by hardware as soon as a valid conversion result is written in the ADCDAT register if the ADC channel is enabled. If the ADC counter register is used (ADCxRCR), this bit is set only when the specified number of ADC conversions equals the value in the ADCxRCR register. Cleared to 0 after reading ADCDAT.

Interrupt Control Register

Address: 0x40030004, Reset: 0x00, Name: ADC0MSKI (ADuCM360 Only)

Address: 0x40030084, Reset: 0x00, Name: ADC1MSKI (ADuCM360 and ADuCM361)

Table 23. Bit Descriptions for ADCxMSKI

Bits	Name	Description
7:4	Reserved	
3	ATHEX	ADC accumulator comparator threshold status bit mask. 0: Disable the interrupt (default). 1: Enable an interrupt when the ADCxATHEX bit in the ADCxSTA register is set.
2	THEX	ADC comparator threshold mask. 0: Disable the interrupt (default). 1: Enable an interrupt when the ADCxTHEX bit in the ADCxSTA register is set.
1	OVR	ADC overrange bit mask. 0: Disable the interrupt (default). 1: Enable an interrupt when the ADCxOVR bit in the ADCxSTA register is set.
0	RDY	Valid conversion result mask. 0: Disable the interrupt (default). 1: Enable an interrupt when the ADCxRDY bit in the ADCxSTA register is set.

Control Registers

Address: 0x40030008, Reset: 0x3038C, Name: ADC0CON (ADuCM360 Only)

Address: 0x40030088, Reset: 0x303FF, Name: ADC1CON (ADuCM360 and ADuCM361)

Table 24. Bit Descriptions for ADCxCON

Bits	Name	Description
31:20	Reserved	
19	ADCEN	Enable bit. 0: Power down the ADC, and clear the ADCxRDY bit. 1: Enable the ADC.
18	ADCCODE	ADC output coding. This bit also affects the coding of the accumulator (ADCxACC) and ADCxTH. 0: Twos complement (bipolar); note that ADCDAT >> 1. 1: Unipolar.
17	BUFPOWN	Negative buffer power. 0: Negative buffer enabled. 1: Negative buffer powered down.
16	BUFPOWP	Positive buffer power. 0: Positive buffer enabled. 1: Positive buffer powered down.
15	BUFBYPP	Positive buffer bypass. 0: Positive buffer not bypassed. 1: Positive buffer bypassed.
14	BUFBYPN	Negative buffer bypass. 0: Negative buffer not bypassed. 1: Negative buffer bypassed.
13:12	ADCREF	Reference selection. 00: INTREF-AGND. 01: EXTREF. The external buffer mode is set in the ADCxCFG register. 10: EXTREF2IN (valid only for ADC1). EXTREF2IN+ buffer controlled via ADCxCFG. 11: AVDD-AGND.
11:10	ADCDIAG	Diagnostic current bits. 00: Current source off. 01: Enable current (50 µA) on selected positive input (for example, AIN0). 10: Enable current (50 µA) on selected negative input (for example, AIN1). 11: Enable current (50 µA) on selected input (for example, AIN0 and AIN1).
9:5	ADCCP	Positive input channel selection. 00000: AIN0. 00001: AIN1. ... 01011: AIN11. 01111: AGND. 11100: All channels off (default). 01100: DAC output (valid only for ADC1). 01101: AVDD/4 (valid only for ADC1). 01110: IOVDD/4 (valid only for ADC1). 10000: Temperature sensor (valid only for ADC1). 11111: All channels off (default) (valid only for ADC1).
4:0	ADCCN	Negative input channel selection. 00000: AIN0. 00001: AIN1. ... ... 01011: AIN11. 01111: AGND. 01100: DAC output. 11111: Reserved. 10001: Temperature sensor (valid only for ADC1). For all other internal channel measurements, select AGND.

Offset Calibration Registers

The offset registers are signed, in twos complement form. A code of 0000 means subtract 0 offset. This does not change to unipolar if the ADCCODE bit in ADCxCON is set.

The user can only write to the offset registers when the ADC is at idle. But a software delay of 6 µs is required between writing to ADCxMDE and writing to the offset or gain register.

The valid range of the ADCxOF register is 0xA000 to 0x5FFF. Using values outside this range generates an under/over range error. The offset is compensated for before shifting the ADC results, depending on the gain setting. Therefore, the LSB weight of the ADCxOF register is independent from the gain settings (LSB weight = $2 \times V_{REF}/49150 = 48.8$ µV) and the offset must be adjusted when changing gain settings.

Address: 0x4003000C, Reset: Factory defined, Name: ADC0OF (ADuCM360 Only)

Address: 0x4003008C, Reset: Factory defined, Name: ADC1OF (ADuCM360 and ADuCM361)

Table 25. Bit Descriptions for ADCxOF

Bits	Name	Description
15:0	VALUE	ADC offset calibration coefficient. Valid range: 0xA000 to 0x5FFF.

Gain Calibration Registers

There are three types of gain registers:

- One when the internal reference configuration is used
- One when the external reference configuration is used
- One when the AVDD reference configuration is used

The corresponding ADCGN is applied according to the reference configuration.

The gain register is an unsigned number, representing a scaling factor. The maximum value is 0xFFFF, and the nominal value is 0x5555.

The gain calibration registers are written with factory calibrated values at power-on or reset. The register is overwritten if a gain calibration is initiated by the user via the ADCxMDE register.

ADC internal full-scale calibration is only intended to work when gain = 1.User can write to gain registers only when the ADC is in idle or power-down mode or when ADCEN = 0 (ADCxCON[19]). However, a software delay is required between writing to ADCxMDE and writing to the offset or gain register. A delay of 6 µs is recommended.

Factory calibration is performed for the internal reference only and is stored in ADCxINTGN. The user must populate ADCxEXTGN and ADCxVDDGN before using the part.

Gain Calibration Registers When Using Internal Reference Register

Address: 0x40030010, Default: Factory Defined, Name: ADC0INTGN (ADuCM360 Only)

Address: 0x40030090, Default: Factory Defined, Name: ADC1INTGN (ADuCM360 and ADuCM361)

Table 26. Bit Descriptions for ADCxINTGN

Bits	Name	Description
15:0	VALUE	ADC 16-bit gain calibration coefficient. Use when the internal reference is selected as the reference.

Gain Calibration Register When Using External Reference Register

Address: 0x40030014, Default: Factory Defined, Name: ADC0EXTGN (ADuCM360 Only)

Address: 0x40030094, Default: Factory Defined, Name: ADC1EXTGN (ADuCM360 and ADuCM361)

Table 27. Bit Descriptions for ADCxEXTGN

Bits	Name	Description
15:0	VALUE	ADC 16-bit gain calibration coefficient. Use when the external reference is selected as the reference.

Gain Calibration Register When Using AVDD as the ADC Reference Register

Address: 0x40030018, Reset: 0x5555, Name: ADC0VDDGN (ADuCM360 Only)

Address: 0x40030098, Reset: 0x5555, Name: ADC1VDDGN (ADuCM360 and ADuCM361)

Table 28. Bit Descriptions for ADCxVDDGN

Bits	Name	Description
15:0	VALUE	ADC 16-bit gain calibration coefficient. Use when AVDD is selected as the reference.

Configuration Register

Address: 0x4003001C, Reset: 0x0000, Name: ADC0CFG (ADuCM360 Only)

Address: 0x4003009C, Reset: 0x0000, Name: ADC1CFG (ADuCM360 and ADuCM361)

ADC0CFG and ADC1CFG are the same physical register, which is shared between both ADCs, but the register has a dual address for ADC DMA writes.

Table 29. Bit Descriptions for ADCxCFG

Bits	Name	Description
15	SIMU (valid for ADuCM360 only).	0: Enable simultaneous sampling. ADC1FLT settings are used for both ADCs (valid for ADuCM360 only). 1: Independent ADC sampling. (Valid for ADuCM360 only). Reserved for ADuCM361.
14	Reserved	
13	BOOST30	Boost the V_{BIAS} current source ability by 30 times. Note that this increases the current consumption. 0: Disable the boost current. 1: Enable the boost current.
12:11	Reserved	
10:8	PINSEL	Enable the V_{BIAS} generator, and send V_{BIAS} to a selected AIN pin. 000: Disable. 001: Reserved. 010: Reserved. 011: Reserved. 100: Send to AIN7. 101: Reserved. 110: Send to AIN11. 111: Disable.
7	GNDSWON	GND_SW. 0: Analog ground switch is disconnected from the external pin. 1: Connect the external GND_SW pin to an internal analog ground reference point. This bit can be used to connect and disconnect external circuits and components to ground under program control and, thereby, minimize dc current consumption when the external circuit or component is not being used.
6	GNDSWRESEN	20 kΩ resistor in series with GND_SW. 0: Disable a 20 kΩ resistor in series with the ground switch. 1: Enable a 20 kΩ resistor in series with the ground switch.
5:2	Reserved	
1:0	EXTBUF	Enable external buffers. 00: Both external reference buffers are powered down and bypassed. 01: External buffers are enabled. The VREF+ and VREF− inputs are selected as buffer inputs. 10: External buffers are enabled. The VREF+ and EXTREF2IN+ inputs are selected as buffer inputs. 11: External buffer on VREF+ is enabled. External buffer on VREF− is powered down and bypassed.

Filter Configuration Registers

Address: 0x40030020, Reset: 0x007D, Name: ADC0FLT (ADuCM360 Only)

Address: 0x400300A0, Reset: 0x007D, Name: ADC1FLT (ADuCM360 and ADuCM361)

Table 30. Bit Descriptions for ADCxFLT

Bits	Name	Description
15	CHOP	Enables system chopping. Enabling system chopping results in very low offset errors and drift. 0: Disable system chopping. 1: Enable system chopping.
14	RAVG2	Enables a running average-by-2. Enabling the average-by-2 function reduces the ADC noise. This feature is automatically active when chopping is enabled and is an optional feature when chopping is inactive. It does not reduce the output rate with chop-off, but does increase the settling-time by one conversion. 0: Disable running average-by-2 function. 1: Enable running average-by-2 function.
13	Reserved	
12	SINC4EN	Enable the sinc4 filter instead of the sinc3 filter. Note that when the output rate is less than 500 Hz, this bit must not be set. When the output rate is greater than or equal to 500 Hz, setting this bit enables the sinc4 filter. The sinc4 filter is recommended when the output rate is greater than 500 Hz; in such cases, the sinc3 filter is insufficient to filter out the quantization noise. 0: Disable sinc4 filter and enable sinc3 filter. 1: Enable sinc4 filter and disable sinc3 filter.
11:8	AF	Averaging factor, #Avgs = AF + 1, where AF implements a programmable first-order sinc postfilter. This additional averaging factor (AF) further reduces the ADC output rate. There are restrictions on the maximum allowable values of SF and AF (see Table 16 and Table 17).
7	NOTCH2	Sinc3/Sinc 4 modify. 0: Disable filter notch. 1: Set by user to modify the standard sinc3/sinc 4 frequency response to increase the filter stop-band rejection by approximately 5 dB. This is achieved by inserting a second notch (NOTCH2) at $f_{NOTCH2} = 1.2 \times f_{NOTCH}$, where f_{NOTCH} is the location of the first notch in the response. The second notch is generally at (6/5) × the main sinc notch to generate notches at both 50 Hz and 60 Hz.
6:0	SF	Sinc3/Sinc4 filter decimation factor. SF controls the oversampling rate/decimation factor of the sinc3/sinc4 filter. The default value for these bits is 0x7D. The conversion rate from the sinc3/sinc4 filter is given in Table 16, and the allowable combination of SF and AF are available in Table 17.

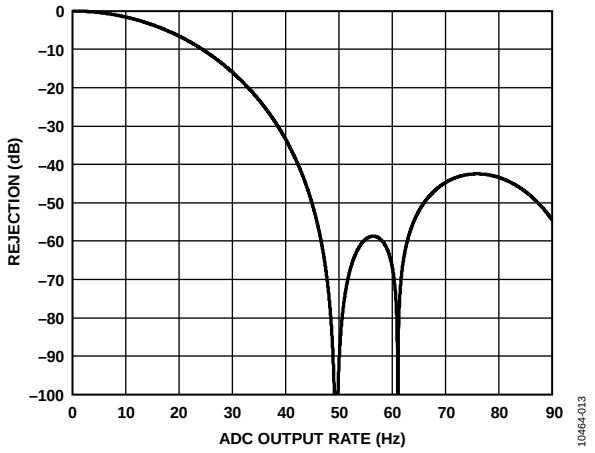

Figure 14. Digital Filter Frequency Response: Notch 2 Filter On

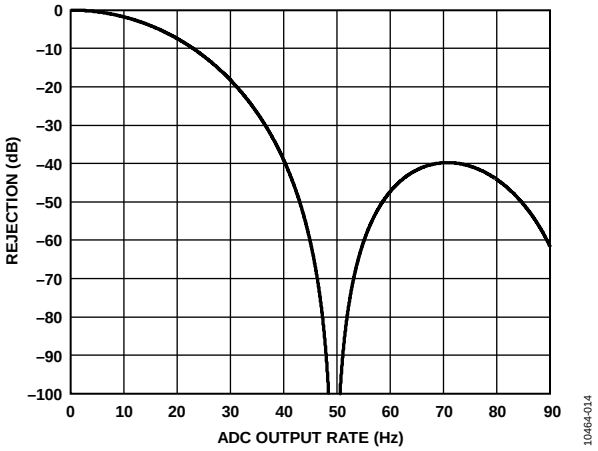

Figure 15. Digital Filter Frequency Response: Notch 2 Filter Off

Mode Control Register

A write to ADCxMDE resets the corresponding ADC, including the RDY bits and other ADCxSTA flags. Depending on the clock used by the ADC, there may be a delay between when the register is written and when the setting takes effect for the ADC logic. Therefore, the more reliable method to reset the RDY bit is to read ADCxDAT.

After each calibration, the ADC returns to idle mode, with the RDY and CAL bits in ADCxSTA set.

The user can write to the offset and gain registers only when the ADC is in idle or power-down mode or when ADCEN = 0 (ADCxCON[19]). However, a software delay is required between writing to ADCMDE and writing to the offset or gain register.

Address: 0x40030024, Reset: 0x0003, Name: ADC0MDE (ADuCM360 Only)

Address: 0x400300A4, Reset: 0x0003, Name: ADC1MDE (ADuCM360 and ADuCM361)

Table 31. Bit Descriptions for ADCxMDE

Bits	Name	Description	
15:8	Reserved		
7:4	ADCxPGA	PGA gain select bit. 0000 PGA gain = 1. 0001 PGA gain = 2. 0010 PGA gain = 4. 0011 PGA gain = 8. 0100 PGA gain = 16. 0101 PGA gain = 32. 0110 PGA gain = 64. 0111 PGA gain = 128. 1000 to 1111 = reserved.	
3	ADCMOD2	Modulator gain of 2. This bit amplifies the output of the PGA by 2 for better resolution. 0: modulator gain of 2 is disabled. 1: modulator gain of 2 is enabled.	
2:0	ADCMD	ADC mode bits.	
		Setting	**Function**
		000	ADC power-down mode. The ADC circuitry is powered off. This powers down the ADC and PGA. Be sure to allow a 7 µs software delay before powering down the processor.
		001	Continuous convert mode. The enabled ADC(s) continuously produce conversions at F_{ADC}. RDY must be cleared to enable new data to be written into ADCxDAT.
		010	Single convert mode. This performs a single-shot conversion on the enabled ADC(s). The ADC enters idle mode after RDY is set.
		011	Idle mode. The ADC is powered up but held in reset. Entered after calibration.
		100	Internal zero-scale calibration. Performed with an internally generated 0 V. The input of the ADC is a short to the negative pin of the selected channel. The result is written to the ADCxOF register of each enabled ADC. This can be used when chopping is off to calibrate the ADC offset. Note that the part must be in idle mode before entering any calibration mode.
		101	Internal full-scale calibration. Performed with an internal/external V_{REF}. Gain = 1. Refer to the ADC Calibration Mode section for more details. Note that the part must be in idle mode before entering any calibration mode.
		110	System zero-scale calibration. Performed on the selected channel; the channel should be shorted externally. Note that the part must be in idle mode before entering any calibration mode.
		111	System full-scale calibration. User to add correct FS voltage to input pins. After each calibration, the ADC returns to idle mode, with the RDY and CAL bits in ADCxSTA set. Note that the part must be in idle mode before entering any calibration mode.

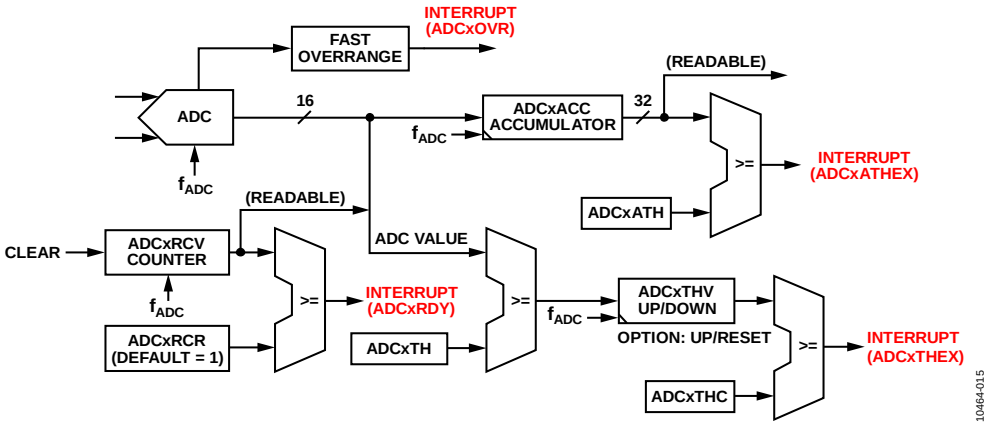

Figure 16. ADC Accumulator/Comparator/Counter Functional Diagram

Note that the result counter does not directly cause an interrupt; instead, this counter masks ADCxRDY interrupts.

Counter Registers: ADCxRCR and ADCxRCV

Address: 0x40030028, Reset: 0x0001, Name: ADC0RCR (ADuCM360 Only)

Address: 0x400300A8, Reset: 0x0001, Name: ADC1RCR (ADuCM360 and ADuCM361)

ADCxRCR sets the number of conversions required before an ADC interrupt is generated. This feature is enabled in ADCxPRO.

Table 32. Bit Descriptions for ADCxRCR

Bits	Name	Description
15:0	VALUE	ADC counter register. The ADCxRDY interrupt is only asserted after the number of ADC samples specified in this register are completed. By default, the value of this register is 1, meaning that an interrupt is generated for every sample. Only valid if ADCxPRO[0] is set to 1.

Address: 0x4003002C, Reset: 0x0000, Name: ADC0RCV (ADuCM360 Only)

Address: 0x400300AC, Reset: 0x0000, Name: ADC1RCV (ADuCM360 and ADuCM361)

ADCxRCV holds the current number of ADC conversion results. The value of the counter is used to mask ADC interrupts, which results in lower frequency interrupts to the ADuCM360/ADuCM361. This value is also important where a number of samples must be accumulated in the ADCxACC. The number of samples accumulated must be read from this MMR to calculate the average result.

Table 33. Bit Descriptions for ADCxRCV

Bits	Name	Description
15:0	VALUE	ADC counter value register. This read only register indicates the number of ADC conversions completed since the last interrupt.

Postprocessing Register: ADCxTH, ADCxTHC, ADCxTHV, ADCxACC, ADCxACC, ADCxATH

Address: 0x40030030, Reset: 0x0000, Name: ADC0TH (ADuCM360 Only)

Address: 0x400300B0, Reset: 0x0000, Name: ADC1TH (ADuCM360 and ADuCM361)

ADCxTH controls the comparator on the ADC output. The MMR holds the threshold value against which the absolute value of the ADC conversion result is compared. The comparison is performed at F_{ADC}, allowing the ADuCM360/ADuCM361 to be interrupted during a high input event, even if the ADC RDY interrupts are masked using the ADCxRC counters.

For twos complement coding, the MSB of this register is ignored because an absolute value comparison is performed (ADCxTH[14:0]). Note that 0x8000 is mapped to 0x7FFF.

For unipolar coding, a 16-bit comparison is performed (ADCxTH[15:0]).

Table 34. Bit Descriptions for ADCxTH

Bits	Name	Description
15:0	VALUE	ADC comparator threshold value. In bipolar mode, Bits[14:0] are compared to ADCxDAT[27:13]. Bit 15 is the sign bit. In unipolar mode, Bits[15:0] are compared to ADCxDAT[27:12]. Bit 15 is the sign bit.

Address: 0x40030034, Reset: 0x01, Name: ADC0THC (ADuCM360 Only)

Address: 0x400300B4, Reset: 0x01, Name: ADC1THC (ADuCM360 and ADuCM361)

ADCxTHC determines how many cumulative conversion result readings above the threshold value (ADCxTH) must occur before the ADC comparator threshold bit is set in ADCxSTA, with ADCxSTA[2] generating an interrupt. Values below the threshold decrement or reset the count to 0.

Table 35. Bit Descriptions for ADC1THC

Bits	Name	Description
7:0	VALUE	ADC comparator count setting, which is only valid if ADCxPRO[3:2] = 10 or 11. An ADCxTH interrupt is asserted if the comparator threshold is exceeded for the number of ADC samples selected in this register. For example, if ADC0PRO[3:2] = 10 and ADC0THC = 30, the ADC0DAT register must be higher than the threshold stored in ADC0TH for 30 consecutive samples.

Address: 0x40030038, Reset: 0x00, Name: ADC0THV (ADuCM360 Only)

Address: 0x400300B8, Reset: 0x00, Name: ADC1THV (ADuCM360 and ADuCM361)

ADC0THV increments every time the absolute value of an ADC conversion result |Result| ≥ ADCxTH. This register is decremented or reset to 0 every time the absolute value of an ADC conversion result |Result| < ADC0TH. The configuration of this function is enabled via the ADC comparator bits in the ADCxPRO[3:2] MMR.

Table 36. Bit Descriptions for ADCxTHV

Bits	Name	Description
7:0	VALUE	ADC comparator count setting, which is only valid if ADCxPRO[3:2] = 10 or 11. This read only register returns the number of ADC0DAT values that have exceeded the threshold set in ADCxTH. The bits are reset to 0 when ADCxTHC is reset to 0.

Address: 0x4003003C, Reset: 0x00000000, Name: ADC0ACC (ADuCM360 Only)

Address: 0x400300BC, Reset: 0x00000000, Name: ADC1ACC (ADuCM360 and ADuCM361)

ADCxACC returns the status of the accumulator. The ADCxACC register updates one or two ADC clocks earlier than ADCxDAT. There is no warning if the accumulator overflows. The ADCxRCV counter can be used to reset the ADCxACC register after 65,535 samples, which is the maximum number of samples possible without the risk of an overflow. The accumulator is a signed, twos complement/unipolar register. There is a mode available that clamps the result to a minimum value of 0; alternatively, the accumulator can be configured to add negative values. The accumulator is reset by disabling the accumulator in ADCxPRO or by reconfiguring the ADC.

Note that the ADCxDAT value is shifted to the right by 12 bits for all gain settings.

Table 37. Bit Descriptions for ADCxACC

Bits	Name	Description
31:0	VALUE	ADC accumulator. ADCxPRO[5:4] enables the accumulator mode.

Address: 0x40030040, Reset: 0x00000000, Name: ADC0ATH (ADuCM360 Only)

Address: 0x400300C0, Reset: 0x00000000, Name: ADC1ATH (ADuCM360 and ADuCM361)

ADCxATH controls the accumulator comparator on the ADC output. The MMR holds the threshold value against which the absolute value of the accumulated ADC conversion result is compared. For twos complement coding, the MSB of this register is ignored because an absolute value comparison is performed (ADCxTH[14:0]). For unipolar coding, a 16-bit comparison is performed (ADCxTH[15:0]).

Table 38. Bit Descriptions for ADCxATH

Bits	Name	Description
31:0	VALUE	ADC accumulator threshold value. The value in the ADCxACC register is compared with the value stored in this register. If ADCxACC > ADCxATH and ADCxPRO[5:4] = 11, the ADCxATH interrupt source is asserted.

Postprocessing Control Registers

Address: 0x40030044, Reset: 0x00, Name: ADC0PRO (ADuCM360 Only)

Address: 0x400300C4, Reset: 0x00, Name: ADC1PRO (ADuCM360 and ADuCM361)

A write to ADCxPRO does not reset the corresponding ADC.

If the overrange or comparator interrupts are set, they are cleared by clearing interrupt mask, disabling the comparator (in the case of the accumulator this clears ACC), or reconfiguring the ADC.

Table 39. Bit Descriptions for ADCxPRO

Bits	Name	Description	
7:6	Reserved		
5:4	ADCxACCEN	ADC accumulator enable.	
		Setting	**Function**
		00	Accumulator disabled.
		01	Accumulator enabled. A negative ADC result decrements the accumulator with a lowest limit of 0.
		10	Accumulator enabled. A negative ADC result decrements the accumulator with no lower limit.
		11	Accumulator and comparator enabled. An interrupt is generated if the value in ADCxACC exceeds ADCxATH. To enable this interrupt, ADCxMSKI[3] must be set to 1. When this interrupt occurs, ADCxSTA[3] is set to 1, indicating that an ADC accumulator interrupt has occurred.
3:2	ADCxCMPEN	ADC comparator enable. To enable the ADC comparator interrupt, ADCxMSKI[2] must be set to 1. When this interrupt occurs, ADCxSTA[2] is set to 1, indicating that an ADC comparator interrupt has occurred.	
		Setting	**Function**
		00	Comparator disabled.
		01	Comparator enabled. Interrupt if ADCxDAT > ADCxTH.
		10	Comparator with counter enabled. Interrupt if ADCxDAT > ADCxTH for the number of samples specified in ADCxTHC. A sample result of ADCxDAT < ADCxTH resets the counter to 0.
		11	Comparator with counter enabled. Interrupt if ADCxDAT > ADCxTH for the number of samples specified in ADCxTHC. A sample result of ADCxDAT < ADCxTH decrements the counter with a lowest limit of 0.
1	ADCxOREN	ADC overrange enable.	
0	ADCxRCEN	ADC result counter enable.	

Data Registers

Address: 0x40030048, Reset: 0x00000000, Name: ADC0DAT (ADuCM360 Only)

Table 40. Bit Descriptions for ADC0DAT

Bits	Name	Description
31:0	VALUE	ADC0 data output register. See the ADC Data Register section for more details.

Address: 0x400300C8, Reset: 0x00000000 Name: ADC1DAT, (ADuCM360 and ADuCM361)

Table 41. Bit Descriptions for ADC1DAT

Bits	Name	Description
31:0	VALUE	ADC1 data output register. See the ADC Data Register section for more details.

Step Detection MMRs

Step Detection Control Register

Address: 0x400300E0, Reset: 0x0000, Name: DETCON

Control register for reference detection and the step detection filter.

Table 42. Bit Descriptions for DETCON

Bits	Name	Description
15:9	Reserved	Reserved. Always set to 0.
8	REFDET	Enable external reference detection circuit. 0: Disable external reference detection circuit. 1: Enable external reference detection circuit.
7	SINC2	Enable sinc2 filter. 0: Disable sinc2 filter. 1: Enable sinc2 filter. The latest sinc2 data is compared with three previous outputs. If the difference exceeds the predefined threshold, the flag (DETSTA[1]) is generated.
6:3	Reserved	Reserved. Always set to 0.
2	ADCSEL	Select ADC for the ADuCM360. This bit allows the user to choose the step detection filter that is enabled for ADC0 or ADC1. For the ADuCM361, this bit is reserved and must be set to 1. 0: The sinc2 filter is enabled for ADC0 (valid for ADuCM360 only). 1: The sinc2 filter is enabled for ADC1 (valid for ADuCM360 only).
1:0	RATE	Control the time interval of the sinc2 filter. 00: The sinc2 filter outputs every 2 ms. 01: The sinc2 filter outputs every 4 ms. 10: The sinc2 filter outputs every 6 ms. 11: The sinc2 filter outputs every 8 ms.

Step Detection Status Register

Address: 0x400300E4, Reset: 0x00, Name: DETSTA

Table 43. Bit Descriptions for DETSTA

Bits	Name	Description
7:5	Reserved	
4	REFSTA	Set to 1 to indicate an external reference lower than 0.4 V or an open circuit. Cleared to 0 by writing 1 to DETSTA[3]. Cleared by default.
3	DATOF	Data overflow status bit. Set to 1 to indicate STEPDAT has been written to again before previous value was read. Cleared to 0 when user reads STEPDAT on time.
2	STEPERR	Set to 1 to indicate that the sinc2 filter conversion has been clamped to +FS/−FS due to such events as overrange or underrange. Cleared to 0.
1	STEPFLAG	Set to 1 to indicate that a step change greater than the threshold has been detected. Cleared to 0 by reading STEPDAT.
0	STEPDATRDY	Set to 1 if a step change greater than threshold is detected; a coarse data in STEPDAT is then offered, which is the output of the step detection filter. STEPDATRDY indicates STEPDAT is available. STEPDAT is offered every 2 ms/4 ms/6 ms/8 ms and is configured by DETCON[1:0].

Step Detection Filter Register

Address: 0x400300E8, Reset: 0x0000, Name: STEPTH

Table 44. Bit Descriptions for STEPTH

Bits	Name	Description
15:10	Reserved	
9:0	VALUE	10-bit threshold register for step detection filter; *Threshold* = (*Value* + 1)/2048 × V_{REF} 0000000000 = (0 + 1)/2048 × V_{REF} ≈ 0.05% × V_{REF} 0000000001 = (1 + 1)/2048 × V_{REF} ≈ 0.1% × V_{REF} ... 0000000011 = (3 + 1)/2048 × V_{REF} ≈ 0.2% × V_{REF} ... 1111111111 = (1023 + 1)/2048 × V_{REF} ≈ 50% × V_{REF}

Step Detection Data Register

Address: 0x400300EC, Reset: 0x00000000, Name: STEPDAT

Table 45. Bit Descriptions for STEPDAT

Bits	Name	Description
31:0	VALUE	16-bit conversion result for ADC0 or ADC1, configured by DETCON[2].

ADC DMA Mode Configuration Register

Address: 0x400300F8, Reset: 0x0000, Name: ADCDMACON

When using the DMA controller to write to the ADC MMR registers, ensure that the write to the ADC0 registers is complete before enabling the write to the ADC1 registers.

Enabling ADC0 and ADC1 value DMA reads at the same time is fully supported.

Table 46. Bit Descriptions for ADCDMACON

Bits	Name	Description
15:5	Reserved	
4	SINC2DMAEN	DMA enable for sinc2 channel. 0: Disable sinc2 DMA accesses. 1: Enable sinc2 DMA accesses.
3	ADC1DMAEN	DMA enable for ADC1 registers—ADC1CON, ADC1FLT, ADC1CP, ADC1CN, ADC1GN, and ADC1OF. For this bit to take effect, ADC1DMAEN must be set to 1. 0: The ADC1 registers are written from user memory with DMA. 1: ADC1DAT is read to user memory with DMA.
2	ADC1CTRL	DMA enable for ADC1 channel. 0: Disable ADC1 DMA accesses. 1: Enable ADC1 DMA accesses.
1	ADC0DMAEN	DMA enable for ADC0 registers—ADC0CON, ADC0FLT, ADC0CP, ADC0CN, ADC0GN, and ADC0OF. For this bit to take effect, ADC0DMAEN must be set to 1. 0: The ADC0 registers are written from user memory with DMA (valid for ADuCM360 only). 1: ADC0DAT will be read to user memory with DMA (valid for ADuCM360 only).
0	ADC0CTRL	DMA enable for ADC0 channel. 0: Disable ADC0 DMA accesses (valid for ADuCM360 only). 1: Enable ADC0 DMA accesses (valid for ADuCM360 only).

Internal Reference Control Register

Address: 0x40008840, Reset: 0x0000, Name: REFCTRL

Table 47. Bit Descriptions for REFCTRL

Bits	Name	Description
15:1	Reserved	
0	REFPD	Power-down reference. This bit must be cleared for the ADCs to work, regardless of if an external reference is selected. 0: Enable internal reference block. 1: Power down internal reference block.

Excitation Current Sources Control Register

Address: 0x40008850, Reset: 0xC0, Name: IEXCCON

Table 48. Bit Descriptions for IEXCCON

Bits	Name	Description
7	PD	IEXC power-down. 0: Enable excitation current source block. 1: Power down excitation current source block.
6	REFSEL	IREF source. 0: Select external current reference resistor on IREF pin. 1: Select internal current reference resistor source.
5:3	IPSEL1	Select IEXC1 output pin. Setting IEXCCON[5] to 1 enables Excitation Current Source 1. When IEXCCON[5] is cleared to 0, Excitation Current Source 1 is disabled. 000: Reserved. … 011: Reserved. 100: IEXC1 output on AIN4. 101: IEXC1 output on AIN5. 110: IEXC1 output on AIN6. 111: IEXC1 output on AIN7.
2:0	IPSEL0	Select IEXC0 output pin. Setting IEXCCON[2] to 1 enables Excitation Current Source 0. When IEXCCON[2] is cleared to 0, Excitation Current Source 0 is disabled. 000: Reserved. … 011: Reserved. 100: IEXC0 output on AIN4. 101: IEXC0 output on AIN5. 110: IEXC0 output on AIN6. 111: IEXC0 output on AIN7.

Excitation Current Data Register

Address: 0x40008854, Reset: 0x06, Name: IEXCDAT

Table 49. Bit Descriptions for IEXCDAT

Bits	Name	Description
7:6	Reserved	
5:1	IDAT	Output current. These five bits set the output current on each pin. See Table 50 for further details.
0	IDAT0	10 µA enable. 0: Disable 10 µA output current. 1: Enable 10 µA output current.

Table 50. Output Current Values Based on IEXCDAT[5:0]

IEXCDAT[5:3]	IEXCDAT[2:1] = 11	IEXCDAT[2:1] = 10	IEXCDAT[2:1] = 01	IEXCDAT[2:1] = 00	IEXCDAT[0] = 1
000	0 µA	0 µA	0 µA	0 µA	Add 10 µA to values on left if this bit is set to 1.
001	200 µA	150 µA	100 µA	50 µA	
010	400 µA	300 µA	200 µA	100 µA	
011	600 µA	450 µA	300 µA	150 µA	
100	800 µA	600 µA	400 µA	200 µA	
101	1000 µA	750 µA	500 µA	250 µA	
110	1000 µA	750 µA	500 µA	250 µA	
111	1000 µA	750 µA	500 µA	250 µA	

DAC

DAC FEATURES

- 12-bit voltage output DAC
- Two selectable ranges:
  - 0 V to V_{REF} (internal band gap 1.2 V reference)
  - 0 V to AVDD_REG (1.8 V)

DAC OVERVIEW

The ADuCM360/ADuCM361 incorporates a voltage output DAC. In normal mode, the DAC resolution is 12-bits. In interpolation mode, the DAC resolution is 16 bits with 14 effective bits. The DAC has a rail-to-rail voltage output buffer capable of driving a 5 kΩ||100 pF load. The DAC can also be configured for an NPN-transistor driver mode. In this mode of operation, the DAC output buffer can be configured to control a 2-wire, 4 mA to 20 mA loop interface with minimal external components. When the DAC is in NPN transistor driver mode, DAC interpolation is not supported.

The DAC has two selectable ranges:

- 0 V to V_{REF} (internal band gap 1.2 V reference)
- 0 V to AVDD_REG (1.8 V)

Note that CLKDIS[7] must be cleared to 0 to enable the system clock to the DAC block. This bit must be cleared to 0 to enable the DAC.

DAC OPERATION

The DAC is configurable through a control register and a data register. The on-chip DAC architecture consists of a resistor string DAC followed by an output buffer amplifier.

The reference source for the DAC is user selectable in software:

- 0-to-V_{REF} mode: The DAC output transfer function spans from 0 V to the internal 1.2 V reference, V_{REF}.
- 0-to-AVDD_REG mode: The DAC output transfer function spans from 0 V to 1.8 V.

The DAC can be configured in three user modes: normal mode, DAC interpolation mode, and NPN transistor driver mode.

Normal DAC Mode, DACCON[3] = 0

In this mode of operation, the DAC is configured as a 12-bit voltage output DAC. By default, the DAC buffer is enabled, but the output buffer can be disabled. If the DAC output buffer is disabled, the DAC is capable of driving a capacitive load of only 10 pF. The DAC buffer is disabled by setting DACCON[6] (DACBUFBYPASS).

The DAC output buffer amplifier features a true rail-to-rail output stage implementation. This means that when unloaded, each output is typically capable of swinging to within less than 5 mV of AGND and AVDD_REG. The linearity specification of the DAC when driving a 5 kΩ resistive load to ground is guaranteed through the full transfer function except for Code 0 to Code 100 and, in 0-to- AVDD_REG mode only, Code 3995 to Code 4095. Linearity degradation near ground and AVDD_REG is caused by saturation of the output amplifier, and a general representation of its effects (neglecting offset and gain error) is illustrated in Figure 17.

The dotted line in Figure 17 indicates the ideal transfer function, and the solid line represents what the transfer function may look like with endpoint nonlinearities due to saturation of the output amplifier. Note that Figure 17 represents a transfer function in 0-to-AVDD_REG mode only. In 0-to-V_{REF} mode, the lower nonlinearity is similar. However, the upper portion of the transfer function follows the ideal line all the way to the end showing no signs of endpoint linearity errors.

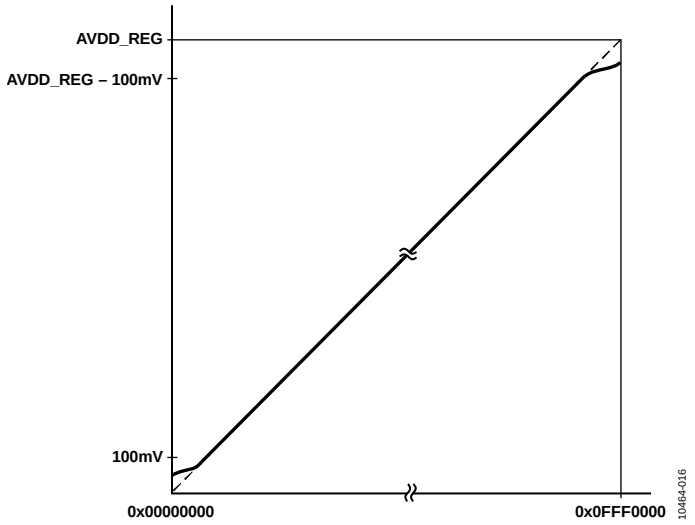

Figure 17. DAC Endpoint Nonlinearities Due to Amplifier Saturation

The endpoint nonlinearities conceptually illustrated in Figure 17 worsen as a function of output loading. Most of the ADuCM360/ADuCM361 data sheet specifications in normal mode assume a 5 kΩ resistive load to ground at the DAC output. As the output is forced to source or sink more current, the nonlinear regions at the top or bottom, respectively, become larger. With larger current demands, this can significantly limit the output voltage swing.

DAC Interpolation Mode, DACCON[3] = 1

In interpolation mode, a higher DAC output resolution of 16 bits, with approximately 14 effective bits, is achieved with a longer update rate than normal mode. The update rate is controlled by the interpolation clock rate and is set in DACCON[2]. In this mode, an external RC filter is required to create a constant voltage. Due to the external filter and slower update rate, the bandwidth of the DAC in this mode is lower.

DAC NPN Transistor Driver Mode, DACCON[8] = 1

In DAC NPN transistor driver mode or 4 mA to 20 mA driver mode, the DAC output is used to control the base of an NPN transistor in a 2-wire, 4 mA to 20 mA loop network. The DAC output buffer acts as a normal op amp. This is shown in Figure 18.

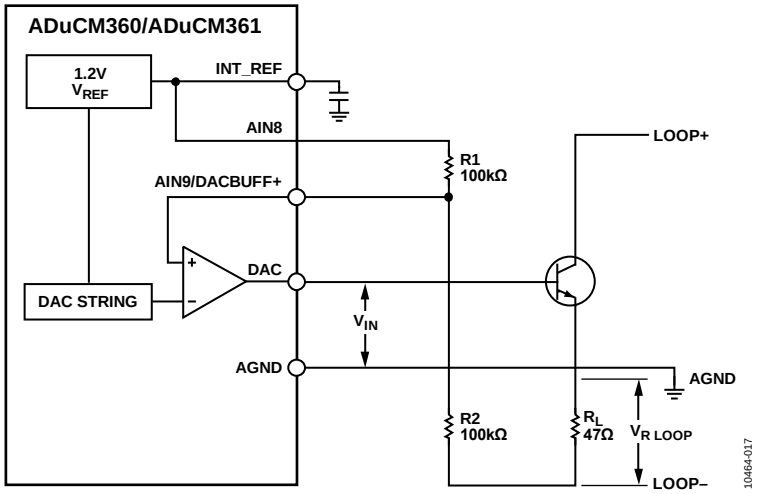

Figure 18. DAC NPN Mode: Typical Operation

The 4 mA to 20 mA feedback circuit is controlled by the ADuCM360/ADuCM361 12-bit DAC output. The DAC output controls the voltage across the 47 Ω resistor marked as RLoop in the circuit. By controlling the voltage across this resistor, the feedback current to the 4 mA to 20 mA host can be accurately set.

Note the top of RLoop connects to the ADuCM360/ADuCM361 ground. The bottom of RLoop connects to the Loop ground. Because of this, the ADuCM360/ADuCM361 DAC output buffer and regulator current, plus the current set by the DAC output flows across RLoop.

When DACCON[8] = 1, V_{REF} is present on AIN8. The on-chip 1.2 V voltage reference is connected externally to R1. The voltage between R1 and R2 (VR12) is connected to the noninverting input of the DAC output buffer. VR12 sets the output voltage range of the op amp. If R1 = R2, this range is 0 mV to 600 mV.

The voltage at the junction of R1 and R2 can be expressed as

$$Vr12 = (VRLoop + Int\_Vref) \times R2/(R1 + R2) - VRLoop$$

With the loop settled, Vin = Vr12.

Because R1 = R2, we get

$$Vin = (VRLoop + Int\_Vref)/2 - VRLoop = Int\_Vref/2 - VRLoop/2$$

or

$$VRLoop = Int\_Vref - 2Vin$$

Full-scale current flows when Vin = 0 (DACDAT = 0), at which point VRLoop = Int_Vref/2.

Therefore, full-scale current is Int_Vref/RLoop, or ~24 mA.

When Vin = Int_Vref/2 (DAC = 600 mV), where Int_Vref is the 1.2 V internal reference voltage, no current flows across RLoop.

DAC DMA OPERATION

DMA Channel 8 is the DAC DMA channel. This is a useful feature if a user wants to configure the DAC as a waveform generator and does not want to interrupt the processor each time the DAC output voltage is updated. Timer 1 must be set up to trigger the DMA write to the DACDAT register.

Example Code to Configure the DAC for DMA Operation

```
void DACDMAINIT( void)
{
pADI_DAC->DACCON = 0x410;                    // Enable DAC, 0-INT_VREF range, timer 1
triggered
Dma_Init();                                  // Init DMA structures
DACDMAWRITE(uxDACDMA, 16);                   // Setup DMA control registers
NVIC_EnableIRQ(DMA_DAC_IRQn);                // Enable DMA DAC Interrupt
pADI_DMA->DMACFG = 0x1;                      // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x100;                  // Select DAC channel in DMA controller
pADI_TM1->LD = 0x00000088;                   // the min clock number is 0x17;
pADI_TM1->CON = 0x00000018;
}

void DACDMAWRITE(unsigned long *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                             // Common configuration of all the descriptors
used here
    Desc.ctrlCfg.Bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_WORD;

                                                  // TX Primary Descriptor
```

```
Desc.srcEndPtr                              = (unsigned long)(pucTX_DMA + iNumRX - 0x1);
Desc.destEndPtr                             = (unsigned long)&DACDAT;
Desc.ctrlCfg.Bits.n_minus_1                 = iNumRX - 0x1;
Desc.ctrlCfg.Bits.src_inc                   = INC_WORD;
Desc.ctrlCfg.Bits.dst_inc                   = INC_NO;
*Dma_GetDescriptor(DMA_DAC_REQ_OUT,FALSE)   = Desc;
}

void DMA_DAC_Out_Int_Handler()
{
NVIC_DisableIRQ(DMA_DAC_IRQn);                    // Clear Interrupt source
}
```

DAC MEMORY MAPPED REGISTERS

Table 51. DAC Memory Mapped Registers Address (Base Address: 0x40020000)

Offset	Name	Description	Access	Default
0x0000	DACCON	Control register	RW	0x0200
0x0004	DACDAT	Data register	R	0x00000000

DAC Control Register

Address: 0x40020000, Reset: 0x200, Name: DACCON

Table 52. DACCON Register Bit Description

Bit	Name	Description
15:11	Reserved	Reserved.
10	DMAEN	0: Normal DAC mode. 1: DAC DMA mode enabled.
9	PN	0: Enable the DAC. 1: Power down DAC output (DAC output is tristate). (Default.)
8	NPN	0: Enable the DAC output buffer in normal mode. 1: Enable the DAC output buffer in NPN transistor driver mode.
7	Reserved	Reserved.
6	BUFBYP	0: Buffer the DAC output. 1: Bypass the output buffer and send the DAC output directly to the output pin.
5	CLK	0: Update the DAC on the negative edge of UCLK. 1: Update the DAC on the negative edge of Timer1. This mode is ideally suited for waveform generation where the next value in the waveform is written to DACDAT at regular intervals of Timer1.
4	CLR	Writing to this bit has an immediate effect on the DAC output. 0: Clear the DAC output and set DACDAT to 0. 1: Normal DAC operation.
3	MDE	0: Enable the DAC in normal 12-bit mode. 1: Enable the DAC in 16-bit interpolation mode.
2	RATE	This bit is used with interpolation mode. 0: Configure the interpolation clock as UCLK/32. 1: Configure the interpolation clock as UCLK/16.
1:0	RNG	DAC range. 00: 0 V to V_{REF} (1.2 V) (internal reference source). 01: Reserved. 10: Reserved. 11: 0 V to AVDD_REG (1.8 V).

DAC DATA Register

Address: 0x40020004, Reset: 0x00000000, Name: DACDAT

Table 53. DACDAT Register Bit Description

Bit	Name	Description
31:28	Reserved	
27:16	DAT12	12-bit data for DAC
15:12	DAT16	Extra four bits used in interpolation mode
11:0	Reserved	

SYSTEM EXCEPTIONS AND PERIPHERAL INTERRUPTS

CORTEX-M3 AND FAULT MANAGEMENT

The ADuCM360/ADuCM361 integrates a Cortex-M3 processor, which supports a number of system exceptions and interrupts generated by peripherals. Table 54 lists the Cortex-M3 processor system exceptions.

Table 54. System Exceptions

Number	Type	Priority	Description
1	Reset	−3 (highest)	Any reset.
2	NMI	−2	Nonmaskable interrupt connected to power supply monitor of ADuCM360/ADuCM361.
3	Hard fault	−1	All fault conditions if the corresponding fault handler is not enabled.
4	Memory management fault	Programmable	Memory management fault; access to illegal locations.
5	Bus fault	Programmable	Prefetch fault, memory access fault, data abort, and other address/memory related faults.
6	Usage fault	Programmable	Same as undefined instruction executed or illegal state transition attempt.
7 to 10	Reserved	N/A	
11	SVCall	Programmable	System service call with SVC instruction. Used for system function calls.
12	Debug monitor	Programmable	Debug monitor (breakpoint, watchpoint, or external debug requests).
13	Reserved	N/A	
14	PendSV	Programmable	Pendable request for system service. Used for queuing system calls until other tasks and interrupts are serviced.
15	SYSTICK	Programmable	System tick timer.

The peripheral interrupts are controlled by the NVIC and are listed in Table 55. All interrupt sources can wake the part up from Mode 1, Mode 2, or Mode 3. Only a limited number of interrupts can wake up the processor from the low power modes (Mode 4 and Mode 5), as shown in Table 55. When the device is woken up from Mode 4 or Mode 5, it returns to Mode 0. If the processor enters any power mode from Mode 1 to Mode 5 while the processor is in an interrupt handler, only an interrupt source with a higher priority than the current interrupt can wake the part up.

Two steps are normally required to configure an interrupt

- Configuring a peripheral to generate an interrupt request to the NVIC.
- Configuring the NVIC for that peripheral request.

Table 55. Interrupt Vector Table

Position Number	Vector	Wake-Up Processor from Mode 1	Wake-Up Processor from Mode 2 or Mode 3	Wake-Up Processor from Mode 4 or Mode 5
0	Wake-up timer	Yes	Yes	Yes
1	External Interrupt 0	Yes	Yes	Yes
2	External Interrupt 1	Yes	Yes	Yes
3	External Interrupt 2	Yes	Yes	Yes
4	External Interrupt 3	Yes	Yes	Yes
5	External Interrupt 4	Yes	Yes	Yes
6	External Interrupt 5	Yes	Yes	Yes
7	External Interrupt 6	Yes	Yes	Yes
8	External Interrupt 7	Yes	Yes	Yes
9	Watchdog timer	Yes	Yes	Yes
10	Reserved	N/A	N/A	N/A
11	Timer0	Yes	No	No
12	Timer1	Yes	No	No
13	ADC0	Yes	No	No
14	ADC1	Yes	No	No

Position Number	Vector	Wake-Up Processor from Mode 1	Wake-Up Processor from Mode 2 or Mode 3	Wake-Up Processor from Mode 4 or Mode 5
15	sinc2	Yes	No	No
16	Flash controller	Yes	No	No
17	UART	Yes	No	No
18	SPI0	Yes	No	No
19	SPI1	Yes	No	No
20	I^2C slave	Yes	No	No
21	I^2C master	Yes	No	No
22	DMA error	Yes	No	No
23	DMA SPI_1 Tx IRQ	Yes	No	No
24	DMA SPI_1 Rx IRQ	Yes	No	No
25	DMA UART Tx IRQ	Yes	No	No
26	DMA UART Rx IRQ	Yes	No	No
27	DMA I^2C slave Tx	Yes	No	No
28	DMA I^2C slave Rx	Yes	No	No
29	DMA I^2C master Tx	Yes	No	No
30	DMA I^2C master Rx	Yes	No	No
31	DMA DAC output	Yes	No	No
32	DMA ADC0	Yes	No	No
33	DMA ADC1	Yes	No	No
34	DMA sinc2 output/step detected	Yes	No	No
35	PWM_TRIP	Yes	No	No
36	PWM0 interrupt	Yes	No	No
37	PWM1 interrupt	Yes	No	No
38	PWM2 interrupt	Yes	No	No

Internally to the Cortex-M3 processor, the highest user-programmable priority (0) is treated as fourth priority—after a reset, an NMI, and a hard fault. Note that 0 is the default priority for all the programmable priorities. If the same priority level is assigned to two or more interrupts, their hardware priority (the lower the position number) determines the order in which the processor activates them. For example, if both SPI0 and SPI1 are Priority Level 1, then SPI0 has higher priority.

To enable an interrupt for any peripheral listed from 0 to 31 in Table 55, set the appropriate bit in the ISER0 register—ISER0 is a 32-bit register and each bit corresponds to the first 32 entries of Table 55.

For example, to enable ADC0 interrupt source in the NVIC, set ISER0[13] = 1. Similarly, to disable ADC0 interrupt, set ICER0[13] = 1.

To enable an interrupt for any peripheral listed from 32 to 38 in Table 55, set the appropriate bit in the ISER1 register—ISER1 is a 32-bit register and ISER1 Bit 0 to Bit 6 correspond to the entries 32 to 38 of Table 55.

For example, to enable the PWM0 interrupt source in the NVIC, set ISER1[4] = 1. Similarly, to disable the PWM0 interrupt, set ICER1[4] = 1.

Table 56 lists the registers to enable and disable relevant interrupts and set the priority levels.

Table 56. NVIC Registers

Address	Analog Devices Header File Name	Description	Access
0xE000E004	ICTR	Shows the number of interrupt lines that the NVIC supports.	R
0xE000E010	STCSR	SYSTICK control and status register.	RW
0xE000E014	STRVR	SYSTICK reload value register.	RW
0xE000E018	STCVR	SYSTICK current value register.	RW
0xE000E01C	STCR	SYSTICK calibration value register.	R
0xE000E100	ISER0	Set IRQ0 to IRQ31 enable. Each bit corresponds to Interrupt 0 to Interrupt 31 in Table 55.	RW
0xE000E104	ISER1	Set IRQ32 to IRQ38 enable. Each bit corresponds to interrupt 32 to Interrupt 38 in Table 55.	RW
0xE000E180	ICER0	Clear IRQ0 to IRQ31 by setting appropriate bit. Each bit corresponds to Interrupt 0 to Interrupt 31 in Table 55.	RW
0xE000E184	ICER1	Clear IRQ32 to IRQ38 by setting appropriate bit. Each bit corresponds to Interrupt 32 to Interrupt 38 in Table 55.	RW
0xE000E200	ISPR0	Set IRQ0 to IRQ31 pending. Each bit corresponds to Interrupt 32 to Interrupt 38 in Table 55.	RW
0xE000E204	ISPR1	Set IRQ32 to IRQ38 pending. Each bit corresponds to Interrupt 32 to Interrupt 38 in Table 55.	RW
0xE000E280	ICPR0	Clear IRQ0 to IRQ31 pending. Each bit corresponds to Interrupt 32 to Interrupt 38 in Table 55.	RW
0xE000E284	ICPR1	Clear IRQ32 to IRQ38 pending. Each bit corresponds to Interrupt 32 to Interrupt 38 in Table 55.	RW
0xE000E300	IABR0	IRQ0 to IRQ31 active bits.	RW
0xE000E304	IABR1	IRQ32 to IRQ38 active bits.	RW
0xE000E400	IPR0	IRQ0 to IRQ3 priority.	RW
0xE000E404	IPR1	IRQ4 to IRQ7 priority.	RW
0xE000E408	IPR2	IRQ8 to IRQ11 priority.	RW
0xE000E40C	IPR3	IRQ12 to IRQ15 priority.	RW
0xE000E410	IPR4	IRQ16 to IRQ19 priority.	RW
0xE000E414	IPR5	IRQ20 to IRQ23 priority.	RW
0xE000E418	IPR6	IRQ24 to IRQ27 priority.	RW
0xE000E41C	IPR7	IRQ28 to IRQ31 priority.	RW
0xE000E420	IPR8	IRQ32 to IRQ35 priority.	RW
0xE000E424	IPR9	IRQ36 to IRQ38 priority.	RW
0xE000ED00	CPUID	CPUID base register.	R
0xE000ED04	ICSR	Interrupt control and status register.	RW
0xE000ED08	VTOR	Vector table offset register.	RW
0xE000ED0C	AIRCR	Application interrupt/reset control register.	RW
0xE000ED10	SCR	System control register.	RW
0xE000ED14	CCR	Configuration control register.	RW
0xE000ED18	SHPR1	System Handlers Register 1.	RW
0xE000ED1C	SHPR2	System Handlers Register 2.	RW
0xE000ED20	SHPR3	System Handlers Register 3.	RW
0xE000ED24	SHCRS	System handler control and state.	RW
0xE000ED28	CFSR	Configurable fault status.	RW
0xE000ED2C	HFSR	Hard fault status.	RW
0xE000ED34	MMAR	MemManage fault address register.	RW
0xE000ED38	BFAR	Bus fault address.	RW
0xE000EF00	STIR	Software trigger interrupt register.	W

EXTERNAL INTERRUPT CONFIGURATION

Eight external interrupts are implemented. These eight external interrupts can be separately configured to detect any combination of the following type of events:

- Edge: rising edge, falling edge, or both rising and falling edges. An interrupt signal (pulse) is sent to the NVIC upon detecting a transition from low to high, high to low, or on either high to low or low to high.
- Level: high or low. An interrupt signal is generated and remains asserted in the NVIC until the conditions generating the interrupt deassert. The level needs to be maintained for a minimum of one core clock cycle to be detected.

The external interrupt detection unit block is in the always on section and allows external interrupt to wake up the device when in hibernate mode.

INTERRUPT MEMORY MAPPED REGISTERS

The interrupt detection unit consists of MMRs contained in the always on section and are based at Address 0x40002400.

Table 57. Interrupt Detection Unit Memory Mapped Registers Address (Base Address: 0x40002400)

Offset	Name	Description	Access	Default
0x0020	EI0CFG	External Interrupt Configuration Register 0	RW	0x0000
0x0024	EI1CFG	External Interrupt Configuration Register 1	RW	0x0000
0x0030	EICLR	External interrupt clear register	RW	0x0000
0x0034	NMICLR	Nonmaskable interrupt clear	RW	0x0000

External Interrupt Configuration Register 0

Address: 0x40002420, Reset: 0x0000, Name: EI0CFG

Table 58. EI0CFG Register Bit Descriptions

Bits	Name	Description
15	IRQ3EN	External Interrupt 3 enable bit 0: External Interrupt 3 disabled 1: External Interrupt 3 enabled
14:12	IRQ3MDE	External Interrupt 3 mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)
11	IRQ2EN	External Interrupt 2 enable bit 0: External Interrupt 2 disabled 1: External Interrupt 2 enabled
10:8	IRQ2MDE	External Interrupt 2 mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)
7	IRQ1EN	External Interrupt 1 enable bit 0: External Interrupt 1 disabled 1: External Interrupt 1 enabled
6:4	IRQ1MDE	External Interrupt 1 mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)
3	IRQ0EN	External Interrupt 0 enable bit 0: External Interrupt 0 disabled 1: External Interrupt 0 enabled
2:0	IRQ0MDE	External Interrupt 0 mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)

External Interrupt Configuration Registers 1

Address: 0x40002424, Reset: 0x0000, Name: EI1CFG

Table 59. EI1CFG Register Bit Descriptions

Bits	Name	Description
15	IRQ7EN	External Interrupt 7 enable bit 0: External Interrupt 7 disabled 1: External Interrupt 7 enabled
14:12	IRQ7MDE	External Interrupt 7 mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)
11	IRQ6EN	External Interrupt 6 enable bit 0: External Interrupt 6 disabled 1: External Interrupt 6 enabled
10:8	IRQ6MDE	External Interrupt 6 Mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)
7	IRQ5EN	External Interrupt 5 enable bit 0: External Interrupt 5 disabled 1: External Interrupt 5 enabled
6:4	IRQ5MDE	External Interrupt 5 mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)
3	IRQ4EN	External Interrupt 4 enable bit 0: External Interrupt 4 disabled 1: External Interrupt 4 enabled
2:0	IRQ4MDE	External Interrupt 4 Mode registers 000: Rising edge 001: Falling edge 010: Rising and falling edges 011: High level 100: Low level 101: Falling edge (same as 001) 110: Rising and falling edges (same as 010) 111: High level (same as 011)

External Interrupt Clear Register

Address: 0x40002430, Reset: 0x0000, Name: EICLR

Table 60. EICLR Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved.
7	IRQ7	External Interrupt 7 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
6	IRQ6	External Interrupt 6 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
5	IRQ5	External Interrupt 5 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
4	IRQ4	External Interrupt 4 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
3	IRQ3	External Interrupt 3 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
2	IRQ2	External Interrupt 2 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
1	IRQ1	External Interrupt 1 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.
0	IRQ0	External Interrupt 0 clear bit. 1: Clear an internal interrupt flag. Cleared automatically by hardware.

Ensure that the register write has fully completed before returning from the interrupt handler. Use the DSB instruction, for example:

```
void Ext_Int4_Handler ()
{
 EiClr(EXTINT4);
 __DSB();
}
```

This ensures that outstanding memory operations complete before new instructions start executing.

Nonmaskable Interrupt Clear Register

Address: 0x40002434, Reset: 0x0000, Name: NMICLR

Table 61. NMICLR Register Bit Descriptions

Bits	Name	Description
15:1	Reserved	Reserved.
0	CLEAR	NMI clear bit. Set by the user to clear the internal flag of the NMI. (The internal flag is not visible to the user.) After an external interrupt has been asserted by the IDU, an interrupt pulse to the Cortex-M3 is generated. Unless this external interrupt is cleared, subsequent interrupts on the same port are ignored by the IDU, and no further pulses are generated. Cleared automatically by hardware.

Ensure that the register write has fully completed before returning from the interrupt handler. Use the DSB instruction, for example:

```
void Ext_Int4_Handler ()
{
 EiClr(EXTINT4);
 __DSB();
}
```

DMA CONTROLLER

DMA FEATURES

- 12 dedicated DMA channels

DMA OVERVIEW

Direct memory access (DMA) is used to provide high speed data transfer between peripherals and memory. Data can be moved quickly by DMA without any processor actions. This keeps processor resources free for other operations.

The DMA controller has 12 channels in total. The 12 channels used are dedicated to managing DMA requests from specific peripherals. Channels are assigned as shown in Table 62.

Note that CLKDIS[8] must be cleared to 0 to enable the system clock to the DMA block, thus enabling any DMA operations.

Table 62. DMA Channel Assignment

Channel	Peripheral
0	SPI1 Tx
1	SPI1 Rx
2	UART Tx
3	UART Rx
4	I^2C slave Tx
5	I^2C slave Rx
6	I^2C master Tx
7	I^2C master Rx
8	DAC DMA output
9	ADC0
10	ADC1
11	SINC2 output/step detection

The channels are connected to dedicated hardware DMA requests; a software trigger is also supported on each channel. This configuration is done by software.

Each DMA channel has a programmable priority level default or high. Within a priority level, arbitration is done using a fixed priority that is determined by the DMA channel number.

The DMA controller supports multiple DMA transfer data widths: independent source and destination transfer size (byte, half word, and word). Source/destination addresses must be aligned on the data size.

The DMA transfer error interrupt is generated when a bus error condition occurs during a DMA transfer.

The DMA controller supports peripheral-to-memory, memory-to-peripheral, and memory-to-memory transfers and access to flash or SRAM, as source and destination.

DMA OPERATION

The DMA controller performs direct memory transfer by sharing the system bus with the Cortex-M3 processor. The DMA request may stall the processor access to the system bus for some bus cycles when the processor and DMA are targeting the same destination (memory or peripheral).

ERROR MANAGEMENT

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel is automatically disabled. If the DMA error interrupt is enabled in the NVIC, the error also generates an interrupt.

INTERRUPTS

An interrupt can be produced when a transfer is complete for each DMA channel. Separate interrupt enable bits are available in the NVIC for each of the DMA channels.

The DMA controller fetches channel control data structures located in the SRAM memory to perform data transfers. When enabled to use DMA operation, the DMA-capable peripherals request the DMA controller for transfer. At the end of the programmed number of DMA transfers for a channel, the DMA controller generates an interrupt corresponding to that channel. This interrupt indicates the completion of the DMA transfer.

DMA PRIORITY

The priority of a channel is determined by its number and priority level. Each channel can have two priority levels: default or high. All channels at high priority level have higher priority than all channels at default priority level. At the same priority level, a channel with a lower channel number has higher priority than a channel with a higher channel number. The DMA channel priority levels can be changed by writing into the appropriate bit in the DMAPRISET register.

CHANNEL CONTROL DATA STRUCTURE

Every channel has two control data structures associated with it: primary data structure and an alternate data structure. For simple transfer modes, the DMA controller uses either the primary or the alternate data structure. For more complex data transfer modes such as ping-pong or scatter-gather, the DMA controller uses both the primary and alternate data structures. Each control data structure (primary or alternate) occupies four 32-bit locations in the memory as shown in Table 63. The entire channel control data structure is shown in Table 64.

Table 63. Channel Control Data Structure

Offset	Name	Description
0x00	SRC_END_PTR	Source end pointer
0x04	DST_END_PTR	Destination end pointer
0x08	CHNL_CFG	Control data configuration
0x0C	Reserved	Reserved

Before the controller can perform a DMA transfer, the data structure related to the DMA channel must be programmed at the designated location in system memory, SRAM.

- The source end pointer memory location contains the end address of the source data.
- The destination end pointer memory location contains the end address of the destination data.
- The control data configuration memory location contains the channel configuration control data.

The programming determines the source and destination data size, number of transfers, and the number of arbitrations.

Table 64. Memory Map of Primary and Alternate DMA Structures

Channel	Primary Structures		Alternate Structures	
Channel 11	Reserved; set to 0	0x0BC	Reserved; set to 0	0x1BC
	Control	0x0B8	Control	0x1B8
	Destination end pointer	0x0B4	Destination end pointer	0x1B4
	Source end pointer	0x0B0	Source end pointer	0x1B0
…	…	…	…	…
Channel 1	Reserved; set to 0	0x01C	Reserved; set to 0	0x11C
	Control	0x018	Control	0x118
	Destination end pointer	0x014	Destination end pointer	0x114
	Source end pointer	0x010	Source end pointer	0x110
Channel 0	Reserved; set to 0	0x00C	Reserved; set to 0	0x10C
	Control	0x008	Control	0x108
	Destination end pointer	0x004	Destination end pointer	0x104
	Source end pointer	0x000	Source end pointer	0x100

The user must define DMA structures in their source code as shown in the examples in the Example Code: Define DMA Structures section. After the structure has been defined, its start address must be assigned to the DMA base address pointer register, DMAPDBPTR.

Each register for each DMA channel is then at the offset address, as specified in Table 64, plus the value in the DMAPDBPTR register.

Example Code: Define DMA Structures

```
memset(dmaChanDesc,0x0,sizeof(dmaChanDesc));
                                            // Setup the DMA base address pointer register.
pADI_DMA->DMAPDBPTR = (unsigned int) &dmaChanDesc;
```

CONTROL DATA CONFIGURATION

For each DMA transfer, the CHNL_CFG memory location provides the control information for the DMA transfer to the controller.

Table 65. Control Data Configuration

Bits	Name	Description		
31:30	DST_INC	Destination address increment. The address increment depends on the source data width as follows:		
		Source Data Width	**DST_INC**	**Destination Address Increment**
		Byte	00	Byte.
			01	Half word.
			10	Word.
			11	No increment. Address remains set to the value that the DST_END_PTR memory location contains.
		Half Word	00	Reserved.
			01	Half word.
			10	Word.
			11	No increment. Address remains set to the value that the DST_END_PTR memory location contains.
		Word	00	Reserved.
			01	Reserved.
			10	Word.
			11	No increment. Address remains set to the value that the DST_END_PTR memory location contains.
29:28	Reserved	Undefined. Write as 0.		
27:26	SRC_INC	Source address increment. The address increment depends on the source data width as follows:		
		Source Data Width	**DST_INC**	**Source Address Increment**
		Byte	00	Byte.
			01	Half word.
			10	Word.
			11	No increment. Address remains set to the value that the SRC_END_PTR memory location contains.
		Half Word	00	Reserved.
			01	Half word.
			10	Word.
			11	No increment. Address remains set to the value that the SRC_END_PTR memory location contains.
		Word	00	Reserved.
			01	Reserved.
			10	Word.
			11	No increment. Address remains set to the value that the SRC_END_PTR memory location contains.
25:24	SRC_SIZE	Size of the source data. 00: byte. 01: half word. 10: word. 11: Reserved.		
23:18	Reserved	Undefined. Write as 0.		
17:14	R_POWER	Set these bits to control how many DMA transfers can occur before the controller rearbitrates. Must be set to 0000 for all DMA transfers involving peripherals. Note that the operation of the DMA is indeterminate if a value other than 0000 is programmed in this location for DMA transfers involving peripherals.		
13:4	N_MINUS_1	The number of configured transfers minus 1 for that channel. The 10-bit value indicates the number of DMA transfers (not the total number of bytes) minus one. The possible values are 0x000: 1 DMA transfer. 0x001: 2 DMA transfers. 0x002: 3 DMA transfers. … 0x3FF: 1024 DMA transfers.		

Bits	Name	Description
3	Reserved	Undefined. Write as 0.
2:0	CYCLE_CTRL	The transfer types of the DMA cycle. 000: Stop (invalid). 001: Basic. 010: Autorequest. 011: Ping-pong. 100: Memory scatter-gather primary. 101: Memory scatter-gather alternate. 110: Peripheral scatter-gather primary. 111: Peripheral scatter-gather alternate.

During the DMA transfer process, but before arbitration, CHNL_CFG is written back to system memory with the N_MINUS_1 field changed to reflect the number of transfers yet to be completed.

At the end when the whole DMA cycle is complete, the CYCLE_CTRL bits are made invalid to indicate the completion of the transfer.

DMA TRANSFER TYPES (CHNL_CFG[2:0])

The DMA controller supports five types of DMA transfer. The various types are selected by programming the appropriate values into the CYCLE_CTRL bits (Bits[2:0]) in the CHNL_CFG location of the control data structure.

Invalid (CHNL_CFG[2:0] = 000)

This means no DMA transfer is enabled for the channel. After the controller completes a DMA cycle, it sets the cycle type to invalid to prevent it from repeating the same DMA cycle.

Basic (CHNL_ CFG[2:0] = 001)

In this mode, the controller can be configured to use either the primary or alternate data structure. The peripheral must present a request for every data transfer. After the channel is enabled, when the controller receives a request, it performs the following operations:

1. The controller performs a transfer. If the number of transfers remaining is 0, the flow continues at Step 3.
2. The controller arbitrates
   a. If a higher priority channel is requesting service, then the controller services that channel.
   b. If the peripheral or software signals a request to the controller, then it continues at Step 1.
3. At the end of the transfer, the controller generates the corresponding DMA channel interrupt in the NVIC.

Autorequest (CHNL_CFG[2:0] = 010)

When the controller operates in this mode, it is only necessary for the controller to receive a single request to enable it to complete the entire DMA cycle. This allows a large data transfer to occur without significantly increasing the latency for servicing higher priority requests or requiring multiple requests from the processor or peripheral. This mode is very useful for a memory-to-memory copy application.

Autorequest is not suitable for peripheral use.

In this mode, the controller can be configured to use either the primary or alternate data structure. After the channel is enabled, when the controller receives a request, it performs the following operations:

1. The controller performs min(2^{R_POWER}, N) transfers for the channel, where R_POWER is Bits[17:14] of the control data configuration register and N is the number of transfers. If the number of transfers remaining is 0, the flow continues at Step 3.
2. A request for the channel is automatically generated. The controller arbitrates. If the channel has the highest priority, the DMA cycle continues at Step 1.
3. At the end of the transfer, the controller generates an interrupt for the corresponding DMA channel.

Ping-Pong (CHNL_CFG[2:0] = 011)

In ping-pong mode, the controller performs a DMA cycle using one of the data structures and then performs a DMA cycle using the other data structure. The controller continues to switch from primary to alternate to primary until it reads a data structure that is invalid, or until the host processor disables the channel.

This mode is useful for transferring data from peripheral to memory using different buffers in the memory. In a typical application, the host must configure both primary and alternate data structures before starting the transfer. As the transfer progresses, the host can subsequently configure primary or alternate control data structures in the interrupt service routine when the corresponding transfer ends.

The DMA controller interrupts the processor after the completion of transfers associated with each control data structure. The individual transfers using either the primary or alternate control data structure work exactly the same as a basic DMA transfer.

Memory Scatter-Gather (CHNL_CFG[2:0] = 100 or 101)

In memory scatter-gather mode, the controller must be configured to use both the primary and alternate data structures. The controller uses the primary data structure to program the control configuration for alternate data structure. The alternate data structure is used for actual data transfers, which are similar to an autorequest DMA transfer. The controller arbitrates after every primary transfer. The controller only needs one request to complete the entire transfer. This mode is used when performing multiple memory-to-memory copy tasks. The processor can configure all of the tasks simultaneously and does not need to intervene in between each task. The controller generates the corresponding DMA channel interrupt in the NVIC when the entire scatter-gather transaction completes using a basic cycle.

In this mode, the controller receives an initial request and then performs four DMA transfers using the primary data structure to program the control structure of the alternate data structure. After this transfer completes, the controller starts a DMA cycle using the alternate data structure. After the cycle completes, the controller performs another four DMA transfers using the primary data structure. The controller continues to switch from primary to alternate to primary until the processor configures the alternate data structure for a basic cycle or the DMA reads an invalid data structure.

Table 66 lists the fields of the CHNL_CFG memory location for the primary data structure, which must be programmed with constant values for the memory scatter-gather mode.

Table 66. CHNL_CFG for Primary Data Structure in Memory Scatter-Gather Mode, CHNL_CFG[2:0] = 100

Bits	Name	Description
31:30	DST_INC	10: Configures the controller to use word increments for the address.
29:28	Reserved	Undefined. Write as 0.
27:26	SRC_INC	10: Configures the controller to use word increments for the address.
25:24	SRC_SIZE	10: Configures the controller to use word transfers.
23:18	Reserved	Undefined. Write as 0.
17:14	R_POWER	0010: Indicates that the DMA controller is to perform four transfers.
13: 4	N_MINUS_1	Configures the controller to perform N DMA transfers, where N is a multiple of 4.
3	Reserved	Undefined. Write as 0.
2:0	CYCLE_CTRL	100: Configures the controller to perform a memory scatter-gather DMA cycle.

Peripheral Scatter-Gather (CHNL_CFG[2:0] = 110 or 111)

In peripheral scatter-gather mode, the controller must be configured to use both the primary and alternate data structure. The controller uses the primary data structure to program the control structure of the alternate data structure. The alternate data structure is used for actual data transfers, and each transfer takes place using the alternate data structure with a basic DMA transfer. The controller does not arbitrate after every primary transfer. This mode is used when there are multiple peripheral-to-memory DMA tasks to be performed. The Cortex-M3 can configure all of the tasks simultaneously and does not need to intervene in between each task. This is very similar to memory scatter-gather mode except for arbitration and request requirements. The controller generates the corresponding DMA channel interrupt in the NVIC when the entire scatter-gather transaction completes using a basic cycle.

In peripheral scatter-gather mode, the controller receives an initial request from a peripheral and then performs four DMA transfers using the primary data structure to program the alternate control data structure. The controller then immediately starts a DMA cycle using the alternate data structure, without rearbitrating.

After this cycle completes, the controller rearbitrates, and if it receives a request from the peripheral that has the highest priority, it performs another four DMA transfers using the primary data structure. It then immediately starts a DMA cycle using the alternate data structure without rearbitrating. The controller continues to switch from primary to alternate to primary until the processor configures the alternate data structure for a basic cycle or the DMA reads an invalid data structure.

Table 67 lists the fields of the CHNL_CFG memory location for the primary data structure, which must be programmed with constant values for the peripheral scatter-gather mode.

Table 67. CHNL_CFG For Primary Data Structure in Peripheral Scatter Gather Mode, CHNL_CFG[2:0] = 110

Bits	Name	Description
31:30	DST_INC	10: Configures the controller to use word increments for the address.
29:28	Reserved	Undefined. Write as 0.
27:26	SRC_INC	10: Configures the controller to use word increments for the address.
25:24	SRC_SIZE	10: Configures the controller to use word transfers.
23:18	Reserved	Undefined. Write as 0.
17:14	R_POWER	0010: Indicates that the DMA controller performed four transfers without rearbitration.
13:4	N_MINUS_1	Configures the controller to perform N DMA transfers, where N is a multiple of 4.
3	Reserved	Undefined. Write as 0.
2:0	CYCLE_CTRL	110: Configures the controller to perform a memory scatter-gather DMA cycle.

ADDRESS CALCULATION

The DMA controller calculates the source read address based on the content of SRC_END_PTR, the source address increment setting in CHNL_CFG, and the current value of the N_MINUS_1 (CHNL_CFG[13:4]).

Similarly, the destination write address is calculated based on the content of DST_END_PTR, the destination address increment setting in CHNL_CFG, and the current value of the N_MINUS_1 (CHNL_CFG[13:4]).

Source Read Address = *SRC_END_PTR* − (*N_MINUS_1* << (*SRC_INC*)) for *SRC_INC* = 0, 1, 2

Source Read Address = *SRC_END_PTR* for *SRC_INC* = 3

Destination Write Address = *DST_END_PTR* − (*N_MINUS_1* << (*DST_INC*)) for *DST_INC* = 0, 1, 2

Destination Write Address = *DST_END_PTR* for *DST_INC* = 3

where *N_MINUS_1* is the number of configured transfers minus 1 for that channel.

Example Code to Configure Setup the DMA Controller

```
typedef struct
{
   unsigned int cycle_ctrl    :3;
   unsigned int next_useburst :1;
   unsigned int n_minus_1     :10;
   unsigned int r_power       :4;
   unsigned int src_prot_ctrl :3;
   unsigned int dst_prot_ctrl :3;
   unsigned int src_size      :2;
   unsigned int src_inc       :2;
   unsigned int dst_size      :2;
   unsigned int dst_inc       :2;
} CtrlCfgBits;

                                                    // Define the structure of a DMA
descriptor.
typedef struct dmaDesc
{
   unsigned int srcEndPtr;
   unsigned int destEndPtr;
   union
   {
      unsigned int   ctrlCfgVal;
      CtrlCfgBits    Bits;
   } ctrlCfg ;
   unsigned int reserved4Bytes;
} DmaDesc;

DmaDesc * Dma_GetDescriptor(unsigned int iChan,boolean bAlternate)
{
    if (iChan < DMACHAN_NUM)
        {
        if (bAlternate)
            return &(dmaChanDesc[iChan + 12]);       //DMA_ALT_OFFSET]);
        else
            return &(dmaChanDesc[iChan ]);
        }
    else
        return 0x0;
}
void Dma_Init(void)
{
                                                    // Clear all the DMA descriptors
(individual blocks will update their
                                                    // descriptors
   memset(dmaChanDesc,0x0,sizeof(dmaChanDesc));
```

```
                                                    // Setup the DMA base address pointer
register.
   pADI_DMA->DMAPDBPTR = (unsigned int) &dmaChanDesc;

                                                    // Enable the  uDMA (WO reg)
   pADI_DMA->DMACFG = 0x1;
}
void DACDMAINIT( void)
{
pADI_DAC->DACCON = 0x410;                           // Enable DAC, 0-INT_VREF range, timer 1
triggered
Dma_Init();                                         // Init DMA structures
DACDMAWRITE(uxDACDMA, 16)                           // Setup DMA control registers
NVIC_EnableIRQ(DMA_DAC_IRQn);                       // Enable DMA DAC Interrupt
pADI_DMA->DMACFG = 0x1;                             // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x100;                         // Select DAC channel in DMA controller
pADI_TM1->LD = 0x00000088;                          // the min clock number are 0x17 – Timer 1
overflow to trigger DAC write
pADI_TM1->CON = 0x00000018;
}

void DACDMAWRITE(unsigned long *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                                    // Common configuration of all the
descriptors used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_WORD;

                                                    // TX Primary Descriptor
Desc.srcEndPtr                          = (unsigned long)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                     = (unsigned long)&DACDAT;
    Desc.ctrlCfg.Bits.n_minus_1         = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc           = INC_WORD;
    Desc.ctrlCfg.Bits.dst_inc           = INC_NO;
    *Dma_GetDescriptor(DMA_DAC_REQ_OUT,FALSE) = Desc;
}
```

DMA MEMORY MAPPED REGISTERS

Table 68. DMA Controller Memory Mapped Registers Address Table (Base Address: 0x40010000)

Address	Name	Description	Access	Default
0x40010000	DMASTA	Status register	R	0x000B0000
0x40010004	DMACFG	Configuration register	RW	0x00000000
0x40010008	DMAPDBPTR	Primary control data base address pointer register	RW	0x00000000
0x4001000C	DMAADBPTR	Alternate control data base address pointer register	R	0x00000100
0x40010014	DMASWREQ	Software request register	R	0x00000000
0x40010020	DMARMSKSET	Request mask set register	RW	0x00000000
0x40010024	DMARMSKCLR	Request mask clear register	R	0x00000000
0x40010028	DMAENSET	Enable set register	RW	0x00000000
0x4001002C	DMAENCLR	Enable clear register	W	0x00000000
0x40010030	DMAALTSET	Primary-alternate set register	RW	0x00000000
0x40010034	DMAALTCLR	Primary-alternate clear register	W	0x00000000
0x40010038	DMAPRISET	Priority set register	RW	0x00000000
0x4001003C	DMAPRICLR	Priority clear register	W	0x00000000
0x4001004C	DMAERRCLR	Bus error clear register	RW	0x00000000
0x40010FD0	DMAPERID4	DMA Peripheral ID 4 register	R	0x00000004
0x40010FE0	DMAPERID0	DMA Peripheral ID 0 register	R	0x00000030
0x40010FE4	DMAPERID1	DMA Peripheral ID 1 register	R	0x000000B2
0x40010FE8	DMAPERID2	DMA Peripheral ID 2 register	R	0x0000000B
0x40010FEC	DMAPERID3	DMA Peripheral ID 3 register	R	0x00000000
0x40010FF0	DMAPCELLID0	DMA PrimeCell ID 0 register	R	0x0000000D
0x40010FF4	DMAPCELLID1	DMA PrimeCell ID 1 register	R	0x000000F0
0x40010FF8	DMAPCELLID2	DMA PrimeCell ID 2 register	R	0x00000005
0x40010FFC	DMAPCELLID3	DMA PrimeCell ID 3 register	R	0x0000000B1

DMA Status Register

Address: 0x40010000, Reset: 0x000B0000, Name: DMASTA

Table 69. DMASTA Register Bit Descriptions

Bits	Name	Description
31:21	Reserved	Reserved.
20:16	CHNLSMINUS1	Number of available DMA channels minus 1. For example, if there are 12 channels available, the register reads back 0xB for these bits.
15:8	Reserved	Reserved. Undefined.
7:4	STATE	Current state of DMA controller state machine. Provides insight into the operation performed by the DMA at the time this register is read. 0000: IDLE: idle. 0001: RDCHNLDATA: reading channel controller data. 0010: RDSRCENDPTR: reading source data end pointer. 0011: RDDSTENDPTR: reading destination end pointer. 0100: RDSRCDATA: reading source data. 0101: WRDSTDATA: writing destination data. 0110: WAITDMAREQCLR: waiting for DMA request to clear. 0111: WRCHNLDATA: writing channel controller data. 1000: STALLED: stalled. 1001: DONE: done. 1010: SCATRGATHR: peripheral scatter-gather transition. 1011 to 1111: reserved.
3:1	Reserved	Reserved. Undefined.
0	ENABLE	Enable status of the controller. 0: Controller is disabled. 1: Controller is enabled.

DMA Configuration Register

Address: 0x40010004, Reset: 0x00000000, Name: DMACFG

Table 70. DMACFG Register Bit Descriptions

Bits	Name	Description
31:1	Reserved	Reserved.
0	ENABLE	Controller enable. 0: Controller is disabled. 1: Controller is enabled.

DMA Primary Control Data Base Address Pointer Register

Address: 0x40010008, Reset: 0x00000000, Name: DMAPDBPTR

Table 71. DMAPDBPTR Register Bit Descriptions

Bits	Name	Description
31:0	CTRLBASEPTR	Pointer to the base address of the primary data structure. Note that 5 + log(2)M LSBs are reserved and must be written 0, where M is the number of channels. The DMAPDBPTR register must be programmed to point to the primary channel control base address pointer in the system memory. The amount of system memory that must be assigned to the DMA controller depends on the number of DMA channels used and whether the alternate channel control data structure is used. This register cannot be read when the DMA controller is in the reset state.

DMA Alternate Control Data Base Address Pointer Register

Address: 0x4001000C, Reset: 0x00000100, Name: DMAADBPTR

Table 72. DMAADBPTR Register Bit Descriptions

Bits	Name	Description
31:0	ALTCBPTR	Base address of the alternate data structure. The DMAADBPTR read only register returns the base address of the alternate channel control data structure. This register removes the necessity for application software to calculate the base address of the alternate data structure. This register cannot be read when the DMA controller is in the reset state.

DMA Software Request Register

Address: 0x40010014, Reset: 0x00000000, Name: DMASWREQ

The DMA software request register is used for memory-to-memory DMA transfers when a DMA channel is not used by a peripheral. It is used to trigger the DMA transfer. Set the appropriate bit to generate a software DMA request on the corresponding DMA channel. These bits are automatically cleared by the hardware after the corresponding software request completes.

Table 73. DMASWREQ Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	Reserved. Reads back 0.
11	SINC	0: Does not create a DMA request for sinc output step detection. 1: Generates a DMA request for sinc output step detection.
10	ADC1	0: Does not create a DMA request for ADC1. 1: Generates a DMA request for ADC1.
9	ADC0	0: Does not create a DMA request for ADC0 (valid for ADuCM360 only). 1: Generates a DMA request for ADC0 (valid for ADuCM360 only).
8	DAC	0: Does not create a DMA request for DAC DMA output. 1: Generates a DMA request for DAC DMA output.
7	I2CMRX	0: Does not create a DMA request for I2CMRX. 1: Generates a DMA request for I2CMRX.
6	I2CMTX	0: Does not create a DMA request for I2CMTX. 1: Generates a DMA request for I2CMTX.
5	I2CSRX	0: Does not create a DMA request for I2CSRX. 1: Generates a DMA request for I2CSRX.
4	I2CSTX	0: Does not create a DMA request for I2CSTX. 1: Generates a DMA request for I2CSTX.
3	UARTRX	0: Does not create a DMA request for UARTRX. 1: Generates a DMA request for UARTRX.
2	UARTTX	0: Does not create a DMA request for UARTTX. 1: Generates a DMA request for UARTTX.
1	SPI1RX	0: Does not create a DMA request for SPI1RX. 1: Generates a DMA request for SPI1RX.
0	SPI1TX	0: Does not create a DMA request for SPI1TX. 1: Generates a DMA request for SPI1TX.

DMA Request Mask Set Register

Address: 0x40010020, Reset: 0x00000000, Name: DMARMSKSET

This register disables DMA requests from peripherals. Each bit of the register represents the corresponding channel number in the DMA controller. Set the appropriate bit to mask the request from the corresponding DMA channel.

Table 74. DMARMSKSET Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	Reserved. Reads back 0.
11	SINC2	When read, 0: Requests are enabled for sinc2. 1: Requests are disabled for sinc2. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with I2CMRX from generating DMA requests.
10	ADC1	When read, 0: Requests are enabled for ADC1. 1: Requests are disabled for ADC1. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with ADC1from generating DMA requests.
9	ADC0	When read, 0: Requests are enabled for ADC0 (valid for ADuCM360 only). 1: Requests are disabled for ADC0 (valid for ADuCM360 only). When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests (valid for ADuCM360 only). 1: Disables peripheral associated with ADC0 from generating DMA requests (valid for ADuCM360 only).
8	DAC	When read, 0: Requests are enabled for DAC. 1: Requests are disabled for DAC. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with DAC from generating DMA requests.
7	I2CMRX	When read, 0: Requests are enabled for I2CMRX. 1: Requests are disabled for I2CMRX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with I2CMRX from generating DMA requests.
6	I2CMTX	When read, 0: Requests are enabled for I2CMTX. 1: Requests are disabled for I2CMTX. When written, 1: Disables peripheral associated with I2CMTX from generating DMA requests. 0: No effect. Use the DMARMSKCLR register to enable DMA requests.
5	I2CSRX	When read, 0: Requests are enabled for I2CSRX. 1: Requests are disabled for I2CSRX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with I2CSRX from generating DMA requests.

Bits	Name	Description
4	I2CSTX	When read, 0: Requests are enabled for I2CSTX. 1: Requests are disabled for I2CSTX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with I2CSTX from generating DMA requests.
3	UARTRX	When read, 0: Requests are enabled for UARTRX. 1: Requests are disabled for UARTRX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with UARTRX from generating DMA requests.
2	UARTTX	When read, 0: Requests are enabled for UARTTX. 1: Requests are disabled for UARTTX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with UARTTX from generating DMA requests.
1	SPI1RX	When read, 0: Requests are enabled for SPI1RX. 1: Requests are disabled for SPI1RX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with SPI1RX from generating DMA requests.
0	SPI1TX	When read, 0: Requests are enabled for SPI1TX. 1: Requests are disabled for SPI1TX. When written, 0: No effect. Use the DMARMSKCLR register to enable DMA requests. 1: Disables peripheral associated with SPI1TX from generating DMA requests.

DMA Request Mask Clear Register

Address: 0x40010024, Reset: 0x00000000, Name: DMARMSKCLR

This register enables DMA requests from peripherals by clearing the mask set in the DMARMSKSET register. Each bit of the register represents the corresponding channel number in the DMA controller.

Set the appropriate bit to clear the corresponding bits in the DMARMSKSET register.

Table 75. DMARMSKCLR Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	Reserved.
11	SINC2	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with sinc2 to generate DMA requests.
10	ADC1	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with ADC1 to generate DMA requests.
9	ADC0	0: No effect. Use the DMARMSKSET register to disable DMA requests (valid for ADuCM360 only). 1: Enables peripheral associated with ADC0 to generate DMA requests (valid for ADuCM360 only).
8	DAC	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with DAC to generate DMA requests.
7	I2CMRX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with I2CMRX to generate DMA requests.
6	I2CMTX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with I2CMTX to generate DMA requests.

Bits	Name	Description
5	I2CSRX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with I2CSRX to generate DMA requests.
4	I2CSTX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with I2CSTX to generate DMA requests.
3	UARTRX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with UARTRX to generate DMA requests.
2	UARTTX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with UARTTX to generate DMA requests.
1	SPI1RX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with SPI1RX to generate DMA requests.
0	SPI1TX	0: No effect. Use the DMARMSKSET register to disable DMA requests. 1: Enables peripheral associated with SPI1TX to generate DMA requests.

DMA Enable Set Register

Address: 0x40010028, Reset: 0x00000000, Name: DMAENSET

Enable DMA channels.

This register allows for the enabling of DMA channels. Reading the register returns the enable status of the channels. Each bit of the register represents the corresponding channel number in the DMA controller. Set the appropriate bit to enable the corresponding channel.

Table 76. DMAENSET Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	Reserved
11	SINC2	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables sinc2 DMA channel.
10	ADC1	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables ADC1 DMA channel.
9	ADC0	0: No effect. Use the DMAENCLR register to disable the channel (valid for ADuCM360 only). 1: Enables ADC0 DMA channel (valid for ADuCM360 only).
8	DAC	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables DAC DMA channel.
7	I2CMRX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables I2CMRX DMA channel.
6	I2CMTX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables I2CMTX DMA channel.
5	I2CSRX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables I2CSRX DMA channel.
4	I2CSTX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables I2CSTX DMA channel.
3	UARTRX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables UARTRX DMA channel.
2	UARTTX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables UARTTX DMA channel.
1	SPI1RX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables SPI1RX DMA channel.
0	SPI1TX	0: No effect. Use the DMAENCLR register to disable the channel. 1: Enables SPI1TX DMA channel.

DMA Enable Clear Register

Address: 0x4001002C, Reset: 0x00000000, Name: DMAENCLR

Disable DMA channels.

This register allows for the disabling of DMA channels. Reading the register returns the enable status of the channels. Each bit of the register represents the corresponding channel number in the DMA controller.

Note that the controller disables a channel automatically by setting the appropriate bit when it completes the DMA cycle. Set the appropriate bit to disable the corresponding channel.

Table 77. DMAENCLR Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	
11	SINC2	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables sinc2 DMA channel.
10	ADC1	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables ADC1DMA channel.
9	ADC0	0: No effect. Use the DMAENSET register to enable the channel (valid for ADuCM360 only). 1: Disables ADC0DMA channel(valid for ADuCM360 only).
8	DAC	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables DAC DMA channel.
7	I2CMRX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables I2CMRX DMA channel.
6	I2CMTX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables I2CMTX DMA channel.
5	I2CSRX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables I2CSRX DMA channel.
4	I2CSTX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables I2CSTX DMA channel.
3	UARTRX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables UARTRX DMA channel.
2	UARTTX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables UARTTX DMA channel.
1	SPI1RX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables SPI1RX DMA channel.
0	SPI1TX	0: No effect. Use the DMAENSET register to enable the channel. 1: Disables SPI1TX DMA channel.

DMA Primary/Alternate Set Register

Address: 0x40010030, Reset: 0x00000000, Name: DMAALTSET

Control structure status/select alternate structure.

Returns the channel control data structure status or selects the alternate data structure for the corresponding DMA channel.

Note that the DMA controller sets/clears these bits automatically as necessary for ping-pong, memory scatter-gather, and peripheral scatter-gather transfers.

Table 78. DMAALTSET Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	
11	SINC2	When read, 0: DMA sinc2 is using the primary data structure. 1: DMA sinc2 is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set sinc2 to 0. 1: Selects the alternate data structure for sinc2.
10	ADC1	When read, 0: DMA ADC1 is using the primary data structure. 1: DMA ADC1 is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set ADC1 to 0. 1: Selects the alternate data structure for ADC1.
9	ADC0	When read, 0: DMA ADC0 is using the primary data structure (valid for ADuCM360 only). 1: DMA ADC0 is using the alternate data structure (valid for ADuCM360 only). When written, 0: No effect. Use the DMAALTCLR register to set ADC0 to 0 (valid for ADuCM360 only). 1: Selects the alternate data structure for ADC0 (valid for ADuCM360 only).
8	DAC	When read, 0: DMA DAC is using the primary data structure. 1: DMA DAC is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set DAC to 0. 1: Selects the alternate data structure for DAC.
7	I2CMRX	When read, 0: DMA I2CMRX is using the primary data structure. 1: DMA I2CMRX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set I2CMRX to 0. 1: Selects the alternate data structure for I2CMRX.
6	I2CMTX	When read, 0: DMA I2CMTX is using the primary data structure. 1: DMA I2CMTX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set I2CMTX to 0. 1: Selects the alternate data structure for I2CMTX
5	I2CSRX	When read, 0: DMA I2CSRX is using the primary data structure. 1: DMA I2CSRX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set I2CSRX to 0. 1: Selects the alternate data structure for I2CSRX.

Bits	Name	Description
4	I2CSTX	When read, 0: DMA I2CSTX is using the primary data structure. 1: DMA I2CSTX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set I2CSTX to 0. 1: Selects the alternate data structure for I2CSTX.
3	UARTRX	When read, 0: DMA UARTRX is using the primary data structure. 1: DMA UARTRX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set UARTRX to 0. 1: Selects the alternate data structure for UARTRX.
2	UARTTX	When read, 0: DMA UARTTX is using the primary data structure. 1: DMA UARTTX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set UARTTX to 0. 1: Selects the alternate data structure for UARTTX.
1	SPI1RX	When read, 0: DMA SPI1RX is using the primary data structure. 1: DMA SPI1RX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set SPI1RX to 0. 1: Selects the alternate data structure for SPI1RX.
0	SPI1TX	When read, 0: DMA SPI1TX is using the primary data structure. 1: DMA SPI1TX is using the alternate data structure. When written, 0: No effect. Use the DMAALTCLR register to set SPI1TX to 0. 1: Selects the alternate data structure for SPI1TX.

DMA Primary/Alternate Clear Register

Address: 0x40010034, Reset: 0x00000000, Name: DMAALTCLR

Select primary data structure.

Set the appropriate bit to select the primary data structure for the corresponding DMA channel.

Note that the DMA controller sets/clears these bits automatically as necessary for ping-pong, memory scatter-gather, and peripheral scatter-gather transfers.

Table 79. DMAALTCLR Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	
11	SINC2	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for sinc2.
10	ADC1	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for ADC1.
9	ADC0	0: No effect. Use the DMAALTSET register to select the alternate data structure (valid for ADuCM360 only). 1: Selects the primary data structure for ADC0 (valid for ADuCM360 only).
8	DAC	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for DAC.
7	I2CMRX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for I2CMRX.

Bits	Name	Description
6	I2CMTX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for I2CMTX.
5	I2CSRX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for I2CSRX.
4	I2CSTX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for I2CSTX.
3	UARTRX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for UARTRX.
2	UARTTX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for UARTTX.
1	SPI1RX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for SPI1RX.
0	SPI1TX	0: No effect. Use the DMAALTSET register to select the alternate data structure. 1: Selects the primary data structure for SPI1TX.

DMA Priority Set Register

Address: 0x40010038, Reset: 0x00000000, Name: DMAPRISET

Configure channel for high priority.

This register enables the user to configure a DMA channel to use the high priority level.

Reading the register returns the status of the channel priority mask. Each bit of the register represents the corresponding channel number in the DMA controller.

Returns the channel priority mask status or sets the channel priority to high.

Table 80. DMAPRISET Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	
11	SINC2	When read, 0: DMA sinc2 is using the default priority level. 1: DMA sinc2 is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set sinc2 to the default priority level. 1: sinc2 uses the high priority level.
10	ADC1	When read, 0: DMA ADC1 is using the default priority level. 1: DMA ADC1 is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set ADC1 to the default priority level. 1: ADC1 uses the high priority level.
9	ADC0	When read, 0: DMA ADC0 is using the default priority level (valid for ADuCM360 only). 1: DMA ADC0 is using a high priority level (valid for ADuCM360 only). When written, 0: No effect. Use the DMAPRICLR register to set ADC0 to the default priority level (valid for ADuCM360 only). 1: ADC0 uses the high priority level (valid for ADuCM360 only).
8	DAC	When read, 0: DMA DAC is using the default priority level. 1: DMA DAC is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set DAC to the default priority level. 1: DAC uses the high priority level.

Bits	Name	Description
7	I2CMRX	When read, 0: DMA I2CMRX is using the default priority level. 1: DMA I2CMRX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set I2CMRX to the default priority level. 1: I2CMRX uses the high priority level.
6	I2CMTX	When read, 0: DMA I2CMTX is using the default priority level. 1: DMA I2CMTX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set I2CMTX to the default priority level. 1: I2CMTX uses the high priority level.
5	I2CSRX	When read, 0: DMA I2CSRX is using the default priority level. 1: DMA I2CSRX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set I2CSRX to the default priority level. 1: I2CSRX uses the high priority level.
4	I2CSTX	When read, 0: DMA I2CSTX is using the default priority level. 1: DMA I2CSTX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set I2CSTX to the default priority level. 1: I2CSTX uses the high priority level.
3	UARTRX	When read, 0: DMA UARTRX is using the default priority level. 1: DMA UARTRX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set UARTRX to the default priority level. 1: UARTRX uses the high priority level.
2	UARTTX	When read, 0: DMA UARTTX is using the default priority level. 1: DMA UARTTX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set UARTTX to the default priority level. 1: UARTTX uses the high priority level.
1	SPI1RX	When read, 0: DMA SPI1RX is using the default priority level. 1: DMA SPI1RX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set SPI1RX to the default priority level. 1: SPI1RX uses the high priority level.
0	SPI1TX	When read, 0: DMA SPI1TX is using the default priority level. 1: DMA SPI1TX is using a high priority level. When written, 0: No effect. Use the DMAPRICLR register to set SPI1TX to the default priority level. 1: SPI1TX uses the high priority level.

DMA Priority Clear Register

Address: 0x4001003C, Reset: 0x00000000, Name: DMAPRICLR

Configure channel for default priority level. The DMAPRICLR write only register enables the user to configure a DMA channel to use the default priority level. Each bit of the register represents the corresponding channel number in the DMA controller. Set the appropriate bit to select the default priority level for the specified DMA channel.

Table 81. DMAPRICLR Register Bit Descriptions

Bits	Name	Description
31:12	Reserved	
11	SINC2	0: No effect. Use the DMAPRISET register to set sinc2 to the high priority level. 1: sinc2 uses the default priority level.
10	ADC1	0: No effect. Use the DMAPRISET register to set ADC1 to the high priority level. 1: ADC1 uses the default priority level.
9	ADC0	0: No effect. Use the DMAPRISET register to set ADC0 to the high priority level (valid for ADuCM360 only). 1: ADC0 uses the default priority level (valid for ADuCM360 only).
8	DAC	0: No effect. Use the DMAPRISET register to set DAC to the high priority level. 1: DAC uses the default priority level.
7	I2CMRX	0: No effect. Use the DMAPRISET register to set I2CMRX to the high priority level. 1: I2CMRX uses the default priority level.
6	I2CMTX	0: No effect. Use the DMAPRISET register to set I2CMTX to the high priority level. 1: I2CMTX uses the default priority level.
5	I2CSRX	0: No effect. Use the DMAPRISET register to set I2CSRX to the high priority level. 1: I2CSRX uses the default priority level.
4	I2CSTX	0: No effect. Use the DMAPRISET register to set I2CSTX to the high priority level. 1: I2CSTX uses the default priority level.
3	UARTRX	0: No effect. Use the DMAPRISET register to set UARTRX to the high priority level. 1: UARTRX uses the default priority level.
2	UARTTX	0: No effect. Use the DMAPRISET register to set UARTTX to the high priority level. 1: UARTTX uses the default priority level.
1	SPI1RX	0: No effect. Use the DMAPRISET register to set SPI1RX to the high priority level. 1: SPI1RX uses the default priority level.
0	SPI1TX	0: No effect. Use the DMAPRISET register to set SPI1TX to the high priority level. 1: SPI1TX uses the default priority level.

DMA Bus Error Clear Register

Address: 0x4001004C, Reset: 0x00000000, Name: DMAERRCLR

Table 82. DMAERRCLR Register Bit Descriptions

Bits	Name	Description
31:1	Reserved	Reserved.
0	ERROR	Bus error status. This register is used to read and clear the DMA bus error status. The error status is set if the controller encounters a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the controller. The other channels are unaffected. Write 1 to clear bits. When read, 0: No bus error occurred. 1: A bus error is pending. When written, 0: No effect. 1: Bit is cleared.

DMA Peripheral ID 4 Register

Address: 0x40001FD0, Reset: 0x00000004, Name: DMAPERID4

Table 83. DMAPERID4 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PID4	DMA Peripheral ID 4. Default value 0x04.

DMA Peripheral ID 0 Register

Address: 0x40001FE0, Reset: 0x00000030, Name: DMAPERID0

Table 84. DMAPERID0 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PID0	DMA Peripheral ID 0. Default value is 0x30.

DMA Peripheral ID 1 Register

Address: 0x40001FE4, Reset: 0x000000B2, Name: DMAPERID1

Table 85. DMAPERID1 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PID1	DMA Peripheral ID 1. Default value is 0xB2.

DMA Peripheral ID 2 Register

Address: 0x40001FE8, Reset: 0x0000000B, Name: DMAPERID2

Table 86. DMAPERID2 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PID2	DMA Peripheral ID 2 Default value is 0x0B.

DMA Peripheral ID 3 Register

Address: 0x40001FEC, Reset: 0x00000000, Name: DMAPERID3

Table 87. DMAPERID3 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PID3	DMA Peripheral ID 3. Default value is 0x00.

DMA PrimeCell ID 0 Register

Address: 0x40001FF0, Reset: 0x0000000D, Name: DMAPCELLID0

Table 88. DMAPCELLID0 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PCELLID0	DMA PrimeCell Identification Register 0. Default value is 0x0D.

DMA PrimeCell ID 1 Register

Address: 0x40001FF4, Reset: 0x000000F0, Name: DMAPCELLID1

Table 89. DMAPCELLID1 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PCELLID1	DMA PrimeCell Identification Register 1. Default value is 0xF0.

DMA PrimeCell ID 2 Register

Address: 0x40001FF8, Reset: 0x00000005, Name: DMAPCELLID2

Table 90. DMAPCELLID2 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PCELLID2	DMA PrimeCell Identification Register 2. Default value is 0x05.

DMA PrimeCell ID 3 Register

Address: 0x40001FFC, Reset: 0x000000B1, Name: DMAPCELLID3

Table 91. DMAPCELLID3 Register Bit Descriptions

Bits	Name	Description
31:14	Reserved	Reserved.
13:0	PCELLID3	DMA PrimeCell Identification Register 3. Default value is 0xB1.

FLASH CONTROLLER

FLASH CONTROLLER FEATURES

- 128 kB Flash/EE
- 2 kB information space
- Flash controller

FLASH CONTROLLER OVERVIEW

The ADuCM360/ADuCM361 includes 128 kB of Flash/EE memory, 2 kB of information space, and a flash controller.

Read and write to flash are executed by direct access.

Commands Supported

- Mass erase and page erase.
- Generation of signatures for single or multiple pages.
- Command abort.

Any access that the processor makes to the flash memory while a command is in progress is stalled until that command completes.

Flash Protection

- Write protection for user space.
- Ability to lock serial wire interface for read protection.

Flash Integrity

- Automatic signature check of kernel space on reset.
- User signature for application code.

FLASH MEMORY ORGANIZATION

The ADuCM360/ADuCM361 controller supports 128 kB of user flash and 2 kB of information space. The information space is memory mapped above user flash space, as shown in Figure 19. The main 128 kB space is organized into 32 × flash blocks. Each flash block contains eight flash pages, and each page is 512 bytes in size. The flash is protected in block segments, but it can be erased/written in page segments.

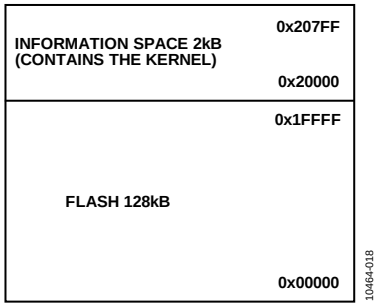

Figure 19. Information and User Space Memory Map on ADuCM360/ADuCM361

User Space

The top 20 bytes of the flash are shown in Figure 20. If a user tries to read to or write from a portion of memory that is not available, a bus error is returned.

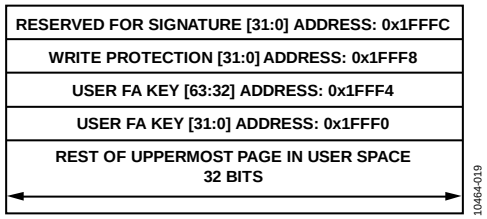

Figure 20. Uppermost Page of User Memory

Kernel Space

The kernel space is reserved for test data and the serial downloader and is read protected.

WRITING TO FLASH/EE MEMORY

Flash is written directly—similarly to SRAM. To enable writes to the flash memory, user code must first set FEECON0[2] to 1. When the flash write sequence is complete, as indicated by FEESTA[3], user code should then clear FEECON0[2]. After these bits are configured and flash protection is disabled, user code can then write to the flash. When a flash erase or write operation is in progress, the Cortex-M3 processor is disconnected from the flash until the erase/write operation is completed.

Note that only 32-bit writes are supported; 16-bit and 8-bit writes are not supported. Burst writes to flash are supported. Because only 0s are written to flash, masking can be used to write individual bits or bytes if necessary. A single location can only be written twice without an erase. If the addressed location is write protected, then the controller responds with a hard fault system exception error.

If a write is followed immediately by a second write to the next location in flash and this is in the same row as the previous write, then the flash write time is faster because the controller does not need to change the row address. Note that only a store multiple assembly instruction takes advantage of this flash feature.

Before performing a write to Flash/EE memory, the FEESTA register should be read and checked to ensure that no other commands or writes are being executed.

Example Code to Write to Flash Memory

```
void WriteToFlash(unsigned long *pArray, unsigned long ulFlashPage, unsigned int uiSize)
{
 unsigned int uiPollFEESTA = 0;
 volatile unsigned long *ulStartAddress;
 unsigned int Size = 0;
 unsigned int n = 0;

 Size = uiSize;
 ulStartAddress = ( unsigned long      *)ulFlashPage;
pADI_FEE->FEECON0 |= 0x4;                                  // Enable a write to Flash memory
 uiPollFEESTA = pADI_FEE->FEESTA;                          // Read Status to ensure it is clear
 uiPollFEESTA = 0;
 for (n = 0; n < Size; n=n+4)
 {
  uiPollFEESTA = 0;
  *ulStartAddress  = *pArray;
  while ((uiPollFEESTA & 0x8) == 0x0)                      // Wait for Command to complete
  {
  uiPollFEESTA = pADI_FEE->FEESTA;
  }
  ulStartAddress = ulStartAddress++;
  pArray++;
 }
 pADI_FEE->FEECON0 &= 0xFB;                                // disable a write to Flash memory
}
```

ERASING FLASH/EE MEMORY

User code can call two flash erase commands,

- Mass erase. This command erases the entire user flash memory. After entering the user protection key into the FEEKEY register, write the mass erase command to FEECMD.
- Page erase. This command erases the 512-byte flash page selected by FEEADR0L/FEEADR0H. After entering the user protection key into the FEEKEY register, load the FEEADR0L/FEEADR0H registers with the page address to be erased. Finally, write the page erase command to FEECMD. CMDDONE (FEESTA[2]) indicates that the page erase command is completed.

During a page or mass erase sequence, the flash controller and flash block consume extra current for the duration of the flash erase sequence.

Example Code to Mass Erase Flash Memory

```
pADI_FEE->FEEKEY = 0xF456;                                 // Enter Flash User Protection key

pADI_FEE->FEEKEY = 0xF123;
pADI_FEE->FEECMD = 0x3;                                    // Load Mass Erase value
```

FLASH CONTROLLER PERFORMANCE AND COMMAND DURATION

The flash controller performance and command duration are as follows. These flash timings are not dependent on the clock dividers.

- Direct single write access (32-bit location): 46 µs
- Mass erase: 21 ms
- Page erase: 21 ms
- Direct write of a page (128 32-bit locations): 3.04 ms
- Mass verify for all of user space: 2.05 ms (128 kB and 16 MHz HCLK)
- Sign for all of user space: 2.05 ms (128 kB and 16 MHz HCLK)

FLASH PROTECTION

There are three types of protection implemented:

- Key protection
- Read protection
- Write protection

Flash Protection: Key Protection

Some of the flash controller registers are key protected to avoid accidental writes to these registers.

The user key is 0xF123F456. It is entered via the 16-bit Key Register 0xF456 first, followed by 0xF123. This key must be entered to run certain user commands, to write to certain locations in flash, or to enable write access to the setup register. Once entered, the key remains asserted unless a command is written to the FEECMD register. When the command completes, the key clears automatically. If this key is entered to enable write access to the setup register or to enable writes to certain locations in flash, it needs to be cleared by user code afterwards. To clear it, write any 16-bit value to the key register.

Flash Protection: User Read Protection

User space read protection is provided by disabling serial wire access to ensure that the flash contents cannot be read via the debug or download interfaces. A user can disable serial wire access by writing 0 to DBG (FEECON1[0]). Note that the FEEKEY register must be loaded prior to writing to FEECON1.

It is also possible to disable the serial wire interface via the downloader.

Serial wire access is disabled while the kernel is running; otherwise, serial wire access can prevent the kernel from running to completion. When the kernel has completed, it enables serial wire access by writing 1 to DBG in the flash FEECON1 register.

Flash Protection: User Write Protection

User write protection is provided to prevent accidental writes to pages in user space and to protect blocks of user code when downloading extra code to flash. If a write or erase of a protected location is detected, the ADuCM360/ADuCM361 flash controller generates an exception.

The write protection is stored at the top of the user space. The top four bytes are reserved for a signature, and the next four bytes are for write protection. The write protection is uploaded by the flash controller into local registers after a reset. After uploading, the 32 write protection bits can be read via memory mapped Register FEEPRO[31:0]. To write to the write protection bits, a user must first write 0xF456, followed by 0xF123 to the key register. After the write protection has been written, it cannot be rewritten without a full erase of user space. After a mass erase, the device must be reset to deassert the uploaded copy of the write protection bits.

The following is the sequence to program the Write Protection FEEPRO[31:0] bits:

1. Write 0xF456 followed by 0xF123 to the key register.
2. Write the needed write protection directly to flash. Write 0 to enable protection. The write protection address is 0x1FFF8 for 128 kB of flash.
3. Verify that the write completed successfully by polling the status register, FEESTA[3], or by enabling a write complete interrupt.
4. Reset the device and the write protection field is uploaded from the kernel space and activated by the flash controller.

If the write protection in flash has not been programmed, that is, 0xFFFFFFFF is uploaded from flash on power-up, then the FEEPRO register can be written to directly from user code. This allows a user to verify the write protection before committing it to flash.

If the write protection in flash has been programmed, the MSB of the write protection must be programmed to 0 to prevent erasing the write protection block. For write protection, memory is split into 32 blocks of eight pages each, or 32 blocks of 4 kB (with one page being 512 bytes). If an attempt is made to write to the write protection word in flash without setting the FEEKEY first, a bus error is generated (hard fault system exception error).

FLASH CONTROLLER FAILURE ANALYSIS KEY

It may be necessary to perform failure analysis on parts that are returned by a user even though read protection is enabled. A method has been provided—a user failure analysis allow key—to allow failure analysis of protected memory.

This key is a 64-bit key that is stored at the top of the user space in flash, as shown in Figure 20. It is used to gain access to user code if the serial wire interface has been locked. It is the user's responsibility to program this key to a value. The key must be given to Analog Devices to enable access to user code.

FLASH INTEGRITY SIGNATURE FEATURE

The signature is used to check the integrity of the flash device. Software can call a signature check command occasionally or whenever a new block of code is about to be executed. The signature is a 24-bit CRC with the polynomial $x^{24} + x^{23} + x^6 + x^5 + x + 1$.

The sign command can be used to generate a signature and check the signature of a block of code, where a block can be a single page or multiple pages. A 24-bit LFSR is used to generate the signature. The hardware assumes that the signature for a block is stored in the upper four bytes of the most significant page of a block; therefore, these four bytes are not included when generating the signature.

Use the following procedure to generate a signature:

1. Write the start address of the block to the FEEADR0L/FEEADR0L H register.
2. Write the end address of the block to the FEEADR1L/ FEEADR1LH register.
3. Write the sign command to the command register (FEECMD = 10).

When the command has completed, the signature is available in the sign register. The signature is compared with the data stored in the upper four bytes of the uppermost page of the block. If the data does not match the signature, a fail status is returned in the status register (FEESTA[5:4] = 10).

While the signature is being computed, all other accesses to flash are stalled. For a 128 kB block, that is 32k reads.

Note that the FEEADR0L/FEEADR0H and FEEADR1L/FEEADR1H addresses are byte addresses, but only pages need to be identified; that is, the lower nine bits are ignored by the hardware.

Note that the user must run the CRC polynomial in user code first to generate the CRC value and must write this to the upper four bytes of the uppermost page of a block. When this operation is complete, any call of the signature feature compares this 4-byte value to the result of the signature check function.

INTEGRITY OF THE KERNEL

The hardware automatically checks the integrity of the kernel after reset. In the event of a failure, FEESTA[6] is set and user code cannot run. This bit can only be read via a serial wire read if the serial wire interface is enabled.

ABORT USING INTERRUPTS

Commands (erase, sign, or mass verify) and writes can be aborted on receipt of an interrupt as listed in Table 55. Aborts are also possible by writing an abort command to the FEECMD register. However, if flash is being programmed and the routine controlling the programming is in flash, it is not possible to use the abort command to abort the cycle because instructions cannot be read. Therefore, the ability to abort a cycle on the assertion of any system interrupt is provided. The FEEAENx register is used to enable aborts on receipt of an interrupt.

When a command or write is aborted via a system interrupt, FEESTA[5:4] indicates an abort (FEESTA[5:4] =11).

It is not possible to abort a flash write or erase cycle without waiting for the high voltage in the flash core to discharge first; that is, a write cycle must finish with a waiting time of 6.6 µs for the high voltage to discharge. An erase cycle must finish with a waiting time of 6.6 µs. A mass erase cycle must finish with a waiting time of 121 µs.

Depending on the state that a write cycle is in when the abort asserts, the write cycle may or may not complete. If the write or erase cycle did not complete successfully, a fail status of aborted can be read in the status register.

If an immediate response to an interrupt is required during an erase or program cycle, then the interrupt service routine and the interrupt vector table must be moved to SRAM for the duration of the cycle.

If the DMA engine is set up to write a block of data to flash, then an interrupt can be set up to abort the current write; however, the DMA engine starts the next write immediately. The interrupt causing the abort stays asserted so that there is a number of aborted write cycles in this case before the processor gains access to flash.

When an abort is triggered by an interrupt, all commands are repeatedly aborted until the appropriate FEEAENx bit is cleared or the interrupt source is cleared.

FLASH CONTROLLER MEMORY MAPPED REGISTERS

Table 92. Flash Controller Memory Mapped Registers Address Table (Base Address: 0x40002800)

Offset	Name	Description	Access	Default
0x0000	FEESTA	Status register.	R	0x0000
0x0004	FEECON0	Command control register.	RW	0x0000
0x0008	FEECMD	Command register.	RW	0x0000
0x0010	FEEADR0L	Lower page address register.	RW	0x0000
0x0014	FEEADR0H	Upper page address register.	RW	0x0000
0x0018	FEEADR1L	Lower page address register.	RW	0x0000
0x001C	FEEADR1H	Upper page address register.	RW	0x0000
0x0020	FEEKEY	The key register.	W	0x0000
0x0028	FEEPROL	Lower 16 bits of the write protection register.	RW	0xFFFF
0x002C	FEEPROH	Upper 16 bits of the write protection register.	RW	0xFFFF
0x0030	FEESIGL	Lower 16 bits of the signature.	R	Modified by the kernel
0x0034	FEESIGH	Upper 16 bits of the signature.	R	Modified by the kernel
0x0038	FEECON1	User setup register.	RW	Modified by the kernel
0x0048	FEEADRAL	Lower 16 bits of the write abort address register.	R	0x0800
0x004C	FEEADRAH	Upper 16 bits of the write abort address register.	R	0x0002
0x0078	FEEAEN0	Interrupt abort enable register. Interrupt 15 to Interrupt 0.	RW	0x0000
0x007C	FEEAEN1	Interrupt abort enable register. Interrupt 31 to Interrupt 16.	RW	0x0000
0x0080	FEEAEN2	Interrupt abort enable register. Interrupt 32 to Interrupt 38.	RW	0x0000

Flash Memory Status Register

Address: 40002800, Reset: 0x0000, Name: FEESTA

Table 93. FEESTA Register Bit Descriptions

Bits	Name	Description		
15:7	Reserved	Return 0 when read.		
6	SIGNERR	Kernel space signature check on reset error. Set to 1 when the signature check fails. After reset, the flash controller automatically checks the information space signature. If the signature check fails, this bit is asserted. The user can check if this bit is set via serial wire only. User code does not execute if this bit is set.		
5:4	CMDRES	These two bits indicate the status of a command on completion or the status of a write. If multiple commands are executed or there are multiple writes without a read of the status register, then the first error encountered is stored. These bits clear to 00 when read.		
		Setting	**Status**	**Function**
		00	SUCCESS	Indicates a successful completion of a command or a write.
		01	PROTECTED	Indicates an attempted erase of a protected location.
		10	VERIFYERR	Indicates a read verify error. After an erase, the controller reads the corresponding word(s) to verify that the transaction completed successfully. If data read is not all F's, this is the resulting status. If the sign command is executed and the resulting signature does not match the data in the upper four bytes of the upper page in a block, this is the resulting status.
		11	ABORT	Indicates that a command or a write was aborted by an abort command or that a system interrupt has caused an abort.
3	WRDONE	Write complete. If there are multiple writes (or a burst write), this status bit asserts after the first long word written and stays asserted until read. If there is a burst write to flash, this bit asserts after every long word written (assuming the bit is cleared by a read after every long word written). Set to 1 when a write completes. Cleared to 0 when read.		
2	CMDDONE	Command complete. If there are multiple commands, this status bit asserts after the first command completes and stays asserted until read. Set to 1 when a command completes. Cleared to 0 when read.		
1	WRBUSY	Write busy. Set to 1 when the flash block is executing a write.		
0	CMDBUSY	Command busy. Set to 1 when the flash block is executing any command entered via the command register.		

Flash Memory Control Register

Address: 0x40002804, Reset: 0x0000, Name: FEECON0

Table 94. FEECON0 Register Bit Descriptions

Bits	Name	Description
15:3	Reserved	Return 0 when read.
2	WREN	Write enable. A flash write when this bit is set to 0 results in a hard fault system exception error, and the write does not take place. 0: Disable writing to the flash. 1: Enable writing to the flash.
1	IENERR	Error interrupt enable. An interrupt is generated when a command or flash write completes with an error status. 0: Disable. 1: Enable.
0	IENCMD	Command complete interrupt enable. An interrupt is generated when a command or flash write completes. 0: Disable. 1: Enable.

Flash Memory Command Register

Address: 0x40002808, Reset: 0x0000, Name: FEECMD

Table 95. FEECMD Register Bit Descriptions

Bits	Name	Description
15:4	Reserved	Return 0 when read.
3:0	CMD	Commands.

The following commands are supported by the flash block.

For repeated page erase commands, the key must be entered before each command. If a command is entered without entering the key first, no action is taken and CMDDONE does not assert.

Any core access to the flash is stalled until the command completes.

Table 96. Flash Controller Commands (FEECMD[3:0])

FEECMD[3:0]	Name	Description
0000	IDLE	No command executed.
0001	ERASEPAGE	Write the address of the page to be erased to FEEADR0L/H and then write this code to the FEECMD register, and the flash erases the page. When the erase has completed, the flash reads every location in the page to verify that all words in the page are erased. If there is a read verify error, this is indicated in FEESTA. To erase multiple pages, wait until a previous page erase has completed. Check the status, and then issue a command to start the next page erase. Before entering this command, 0xF456 followed by 0xF123 must be written to the key register.
0010	SIGN	Use this command to generate a signature for a block of data. The signature is generated on a page-by-page basis. To generate a signature, the address of the first page of the block is entered in FEEADR0L/FEEADR0H. The address of the last page is written to FEEADR1L/FEEADR1H. Then write this code to the FEECMD register. When the command has completed, the signature is available for reading in FEESIGL/FEESIGH. The last four bytes of the last page in a block are reserved for storing the signature. Before entering this command, 0xF456 followed 0xF123 must be written to the key register.
0011	MASSERASE	Erase all of user space. To enable this operation, 0xF456 followed by 0xF123 must first be written to FEEKEY (this is to prevent accidental erases). When the mass erase has completed, the controller reads every location to verify that all locations are 0xFFFFFFFF. If there is a read verify error, this is indicated by FEESTA[5:4] = 0x2.
0100	ABORT	If this command is issued, any command currently in progress is stopped. The status indicates command completed with an error status as indicated by FEESTA[5:4] = 0x3. Note that this is the only command that can be issued while another command is already in progress. This command can also be used to stop a write that is in progress. If a write is aborted, the address of the location being written can be read via the FEEADRAL/FEEADRAH register. While the flash controller is writing one long word, another long word write may be in the pipeline from the Cortex-M3 or DMA engine (depending on how the software implements writes). Therefore, both writes may need to be aborted. If a write or erase is aborted, then the flash timing is violated and it is not possible to determine whether the write or erase completed successfully. To enable this operation, 0xF456 followed by 0xF123 must first be written to FEEKEY (this is to prevent accidental aborts).

Flash Controller Page Address 0 Registers

For an erase command, FEEADR0x indicates the page address to be erased.

For a sign command, FEEADR0x indicates the start page for calculating the signature.

Lower Page Address Register

Address: 0x40002810, Reset: 0x0000, Name: FEEADR0L

Table 97. FEEADR0L Register Bit Descriptions

Bits	Name	Description
15:9	VALUE	Bits[15:9] of the address of a page in flash. Used by the erase and sign commands.
8:0	Reserved	Nine reserved bits for byte addresses. The lower nine bits of a byte address are ignored because erase and sign commands use page address. Returns 0x0 if read.

Upper Page Address Register

Address: 0x40002814, Reset: 0x0000, Name: FEEADR0H

FEEADR0H is two bits wide to allow access to the information space on the ADuCM360/ADuCM361.

Table 98. FEEADR0H Register Bit Descriptions

Bits	Name	Description
15:1	Reserved	Set to 0.
0	VALUE	Bit 16 of the page address.

Flash Controller Page Address 1 Registers

For a sign command, FEEADR1x indicates the end page for calculating the signature.

Lower Page Address Register

Address: 0x40002818, Reset: 0x0000, Name: FEEADR1L

Table 99. FEEADR1L Register Bit Descriptions

Bits	Name	Description
15:9	VALUE	Bits[15:9] of the address of a page in flash. Used to identify the last page used by the sign command.
8:0	Reserved	Nine reserved bits for byte addresses. The lower nine bits of a byte address are ignored because the sign command uses page address. Returns 0x0 if read.

Upper Page Address Register

Address: 0x4000281C, Reset: 0x0000, Name: FEEADR1H

Table 100. FEEADR1H Register Bit Descriptions

Bits	Name	Description
15:1	Reserved	Set to 0.
0	VALUE	Bit 16 of the page address.

Flash Controller Key Register

Address: 0x40002820, Reset: 0x0000, Name: FEEKEY

Table 101. FEEKEY Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Enter 0xF456 followed by 0xF123. Returns 0x0 if read.

Flash Controller Write Protection Registers

Lower 16 Bits

Address: 0x40002828, Reset: 0xFFFF, Name: FEEPROL

Table 102. FEEPROL Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of the write protection. This register is read only if the write protection in flash has been programmed. 0: Protect a section of flash. 1: Leave a flash block unprotected.

Upper 16 Bits

Address: 0x4000282C, Reset: 0xFFFF, Name: FEEPROH

Table 103. FEEPROH Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Upper 16 bits of the write protection. This register is read only if the write protection in flash has been programmed. 0: Protect a section of flash. 1: Leave a flash block unprotected.

Flash Controller Signature Registers

Lower 16 Bits

Address: 0x40002830, Reset: Modified by the kernel, Name: FEESIGL

Table 104. FEESIGL Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of the signature. Signature[15:0].

Upper 16 Bits

Address: 0x40002834, Reset: Modified by the kernel, Name: FEESIGH

Table 105. FEESIGH Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved.
7:0	VALUE	Upper eight bits of the signature. Signature[23:16].

User Setup Register

Address: 0x40002838, Reset: Modified by the kernel, Name: FEECON1

FEECON1 register is key protected. The key must be entered in FEEKEY. After writing to FEECON1, a 16-bit value must be written again to FEEKEY, to reassert the key protection.

Table 106. FEECON1 Register Bit Descriptions

Bits	Name	Description
15:1	Reserved	Returns 0 when read.
0	DBG	Serial wire debug enable. The kernel set this bit to 1 when it has finished executing, thus enabling debug access to a user. 0: Disable access via the serial wire debug interface. 1: Enable access via the serial wire debug interface.

Flash Controller Write Abort Address Registers Lower 16 Bits

Address: 0x40002848, Reset: 0x0800, Name: FEEADRAL

Table 107. FEEADRAL Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of the FEEADRA register. If a write is aborted, these bits contain the address of the location written when the write was aborted.

Flash Controller Write Abort Address Registers Upper 16 Bits

Address: 0x4000284C, Reset: 0x0002, Name: FEEADRAH

Table 108. FEEADRAH Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Upper 16 bits of the FEEADRA register.

Flash Controller System IRQ Abort Enable Register

Address: 0x40002878, Reset: 0x0000, Name: FEEAEN0

To allow a system interrupt to abort a write or a command (erase, sign, or mass verify), write a 1 to the appropriate bit in this register. For example, if external IRQ2 is required to abort a flash command, set FEEAEN0 = 0x8.

FEEAEN0 applies to Interrupt Vector Number 0 to Interrupt Vector Number 15 in Table 55.

Table 109. FEEAEN0 Register Bit Descriptions

Bits	Bit Name	Description	Reset	Access
15	SINC2	SINC2 interrupt abort enable bit. 0: DIS. SINC2 interrupt abort disabled. 1: EN. SINC2 interrupt abort enabled.	0x0	RW
14	ADC1	ADC1 interrupt abort enable bit. 0: DIS. ADC1 interrupt abort disabled. 1: EN. ADC1 interrupt abort enabled.	0x0	RW
13	ADC0 (ADuCM360) Reserved (ADuCM361)	ADC0 interrupt abort enable bit. 0: DIS. ADC0 interrupt abort disabled. 1: EN. ADC0 interrupt abort enabled.	0x0	RW
12	T1	Timer1 (general purpose timer 1) interrupt abort enable bit. 0: DIS. Timer1 interrupt abort disabled. 1: EN. Timer1 interrupt abort enabled.	0x0	RW
11	T0	Timer0 (general purpose timer 0) interrupt abort enable bit. 0: DIS. Timer0 interrupt abort disabled. 1: EN. Timer0 interrupt abort enabled.	0x0	RW
10	Reserved			
9	T3	Timer3 (watchdog timer) interrupt abort enable bit. 0: DIS. Timer3 interrupt abort disabled. 1: EN. Timer3 interrupt abort enabled.	0x0	RW
8	EXTINT7	External interrupt 7 abort enable bit. 0: DIS. External interrupt 7 abort disabled. 1: EN. External interrupt 7 abort enabled.	0x0	RW
7	EXTINT6	External interrupt 6 abort enable bit. 0: DIS. External interrupt 6 abort disabled. 1: EN. External interrupt 6 abort enabled.	0x0	RW
6	EXTINT5	External interrupt 5 abort enable bit. 0: DIS. External interrupt 5 abort disabled. 1: EN. External interrupt 5 abort enabled.	0x0	RW

Bits	Bit Name	Description	Reset	Access
5	EXTINT4	External interrupt 4 abort enable bit. 0: DIS. External interrupt 4 abort disabled. 1: EN. External interrupt 4 abort enabled.	0x0	RW
4	EXTINT3	External interrupt 3 abort enable bit. 0: DIS. External interrupt 3 abort disabled. 1: EN. External interrupt 3 abort enabled.	0x0	RW
3	EXTINT2	External interrupt 2 abort enable bit. 0: DIS. External interrupt 2 abort disabled. 1: EN. External interrupt 2 abort enabled.	0x0	RW
2	EXTINT1	External interrupt 1 abort enable bit. 0: DIS. External interrupt 1 abort disabled. 1: EN. External interrupt 1 abort enabled.	0x0	RW
1	EXTINT0	External interrupt 0 abort enable bit. 0: DIS. External interrupt 0 abort disabled. 1: EN. External interrupt 0 abort enabled.	0x0	RW
0	T2	Timer 2 (wake up timer) interrupt abort enable bit. 0: DIS. Timer 2 interrupt abort disabled. 1: EN. Timer 2 interrupt abort enabled	0x0	RW

Flash Controller System IRQ Abort Enable Register

Address: 0x4000287C, Reset: 0x0000, Name: FEEAEN1

To allow a system interrupt to abort a write or a command (erase, sign, or mass verify), write a 1 to the appropriate bit in this register. The appropriate bit is determined by the interrupt required to abort the flash command. FEEAEN1 applies to Interrupt Vector Number 16 to Interrupt Vector Number 31 in Table 55.

Table 110. FEEAEN1 Register Bit Descriptions

Bits	Bit Name	Description	Reset	Access
15	DMADAC	DAC output DMA interrupt abort enable bit. 0: DIS. DAC DMA interrupt abort disabled. 1: EN. DAC DMA interrupt abort enabled.	0x0	RW
14	DMAI2CMRX	I^2C master Rx DMA interrupt abort enable bit. 0: DIS. I^2C master Rx DMA interrupt abort disabled. 1: EN. I^2C master Rx DMA interrupt abort enabled.	0x0	RW
13	DMAI2CMTX	I^2C master Tx DMA interrupt abort enable bit. 0: DIS. I^2C master Tx DMA interrupt abort disabled. 1: EN. I^2C master Tx DMA interrupt abort enabled.	0x0	RW
12	DMAI2CSRX	I^2C slave Rx DMA interrupt abort enable bit. 0: DIS. I^2C slave Rx DMA interrupt abort disabled. 1: EN. I^2C slave Rx DMA interrupt abort enabled.	0x0	RW
11	DMAI2CSTX	I^2C slave Tx DMA interrupt abort enable bit. 0: DIS. I^2C slave Tx DMA interrupt abort disabled. 1: EN. I^2C slave Tx DMA interrupt abort enabled.	0x0	RW
10	DMAUARTRX	UARTRx DMA interrupt abort enable bit. 0: DIS. UARTRx DMA interrupt abort disabled. 1: EN. UARTRx DMA interrupt abort enabled.	0x0	RW
9	DMAUARTTX	UARTTx DMA interrupt abort enable bit. 0: DIS. UARTTx DMA interrupt abort disabled. 1: EN. UARTTx DMA interrupt abort enabled.	0x0	RW
8	DMASPI1RX	SPI1RXx DMA interrupt abort enable bit. 0: DIS. SPI1Rx DMA interrupt abort disabled. 1: EN. SPI1Rx DMA interrupt abort enabled.	0x0	RW
7	DMASPI1TX	SPI1Tx DMA interrupt abort enable bit. 0: DIS. SPI1Tx DMA interrupt abort disabled. 1: EN. SPI1Tx DMA interrupt abort enabled.	0x0	RW
6	DMAERROR	DMA error interrupt abort enable bit. 0: DIS. DMA error interrupt abort disabled. 1: EN. DMA error interrupt abort enabled.	0x0	RW
5	I2CM	I^2C master interrupt abort enable bit. 0: DIS. I^2C slave interrupt abort disabled. 1: EN. I^2C master interrupt abort enabled.	0x0	RW
4	I2CS	I^2C slave interrupt abort enable bit. 0: DIS. I^2C slave interrupt abort disabled. 1: EN. I^2C slave interrupt abort enabled.	0x0	RW
3	SPI1	SPI1 interrupt abort enable bit. 0: DIS. SPI1 interrupt abort disabled. 1: EN. SPI1 interrupt abort enabled.	0x0	RW
2	SPI0	SPI0 interrupt abort enable bit. 0: DIS. SPI0 interrupt abort disabled. 1: EN. SPI0 interrupt abort enabled.	0x0	RW
1	UART	UART interrupt abort enable bit. 0: DIS. UART interrupt abort disabled. 1: EN. UART interrupt abort enabled.	0x0	RW

Bits	Bit Name	Description	Reset	Access
0	FEE	Flash controller interrupt abort enable bit. 0: DIS. Flash controller interrupt abort disabled. 1: EN. Flash controller interrupt abort enabled.		

Flash Controller System IRQ Abort Enable Register

Address: 0x40002880, Reset: 0x0000, Name: FEEAEN2

To allow a system interrupt to abort a write or a command (erase, sign, or mass verify), write a 1 to the appropriate bit in this register. The appropriate bit is determined by the interrupt required to abort the flash command. FEEAEN2 applies to Interrupt Vector Number 32 to Interrupt Vector Number 38 in Table 55.

Table 111. FEEAEN2 Register Bit Descriptions

Bits	Bit Name	Description	Reset	Access
[15:7]	RESERVED	Reserved.	0x00	R
6	PWM2	PWM2 interrupt abort enable bit. 0: DIS. PWM2 interrupt abort disabled. 1: EN. PWM2 interrupt abort enabled.	0x0	RW
5	PWM1	PWM1 interrupt abort enable bit. 0: DIS. PWM1 interrupt abort disabled. 1: EN. PWM1 interrupt abort enabled.	0x0	RW
4	PWM0	PWM0 interrupt abort enable bit. 0: DIS. PWM0 interrupt abort disabled. 1: EN. PWM0 interrupt abort enabled.	0x0	RW
3	PWMTRIP	PWMTRIP interrupt abort enable bit. 0: DIS. PWMTRIP interrupt abort disabled. 1: EN. PWMTRIP interrupt abort enabled.	0x0	RW
2	DMASINC2	SINC2 DMA interrupt abort enable bit. 0: DIS. SINC2 DMA interrupt abort disabled. 1: EN. SINC2 DMA interrupt abort enabled.	0x0	RW
1	DMAADC1	ADC1 DMA interrupt abort enable bit. 0: DIS. ADC1 DMA interrupt abort disabled. 1: EN. ADC1 DMA interrupt abort enabled.	0x0	RW
0	DMAADC0(ADuCM360) Reserved (ADuCM361).	ADC0 DMA interrupt abort enable bit. 0: DIS. ADC0 DMA interrupt abort disabled. 1: EN. ADC0 DMA interrupt abort enabled.	0x0	RW

RESET

RESET FEATURES

There are four kinds of resets:

- External reset
- Power-on reset
- Watchdog timeout
- Software system reset

RESET OPERATION

The software system reset is provided as part of the Cortex-M3 processor. To generate a software system reset, the application interrupt/reset control register must be written to 0x05FA0004. This register is part of the NVIC register and is located at Address 0xE000ED0C.

The RSTSTA register stores the cause for the reset until it is cleared by writing the RSTCLR register. RSTSTA and RSTCLR can be used during a reset exception service routine to identify the source of the reset.

Note that the watchdog timer is enabled by default after a reset. The default timeout period is approximately 32 sec.

User code should disable the watchdog timer at the start of user code when debugging or if the watchdog timer is not required.

```
pADI_WDT->T3CON = 0x00 ;                    // Disable watchdog timer
```

Table 112. Device Reset Implications

	Impact					
Reset	**Reset External Pins to Default State**	**Execute Kernel**	**Reset All MMRs Except RSTSTA**	**Reset All Peripherals**	**Valid SRAM**	**RSTSTA After Reset Event**
SWRST	Yes[1]	Yes	Yes	Yes	Yes/No[2]	RSTSTA[3] = 1
WDRST	Yes[1]	Yes	Yes	Yes	Yes/No[2]	RSTSTA[2] = 1
External Reset Pin	Yes[1]	Yes	Yes	Yes	Yes/No[2]	RSTSTA[1] = 1
POR	Yes[1]	Yes	Yes	Yes	No	RSTSTA[0] = 1

[1] P0.7 returns to its default state, that is, POR output. It is low only in the case of a POR event; in all other cases, it remains high.
[2] RAM is not valid in the case of a reset following a UART download.

RESET MEMORY MAPPED REGISTERS

Table 113. Reset Memory Mapped Register Addresses (Base Address: 0x40002400)

Offset	Name	Description	Access	Default
0x0040	RSTSTA	Status register, read only.	R	0x01, depends on the type of reset
0x0040	RSTCLR	Clear status register, write only.	W	N/A

Reset Status/Status Clear Registers

Address: 0x40002440, Reset: 0x01, Depends on the Type of Reset/N/A, Name: RSTSTA/RSTCLR

Table 114. RSTSTA/RSTCLR Register Bit Descriptions

Bits	Name	Description
7:4	Reserved	Reserved.
3	SWRST	Software reset. 0: Cleared by setting the corresponding bit in RSTCLR. 1: Set automatically to 1 when the Cortex-M3 system reset is generated.
2	WDRST	Watchdog timeout. 0: Cleared by setting the corresponding bit in RSTCLR. 1: Set automatically to 1 when a watchdog timeout occurs.
1	EXTRST	External reset. 0: Cleared by setting the corresponding bit in RSTCLR. 1: Set automatically to 1 when an external reset occurs.
0	POR	Power-on reset. 0: Cleared by setting the corresponding bit in RSTCLR. 1: Set automatically when a power-on reset occurs.

DIGITAL I/Os

DIGITAL I/Os FEATURES

The ADuCM360/ADuCM361 features a number of general-purpose bidirectional input/output (GPIO) pins. Most of the GPIO pins have multiple functions, configurable by user code. At power up, all but three of these pins are configured as GPIOs; one pin reflects the state of the POR, and the last two digital pins are configured for serial wire interface. These three pins can be configured by user code to be used as GPIOs (Mode 01).

DIGITAL I/Os BLOCK DIAGRAM

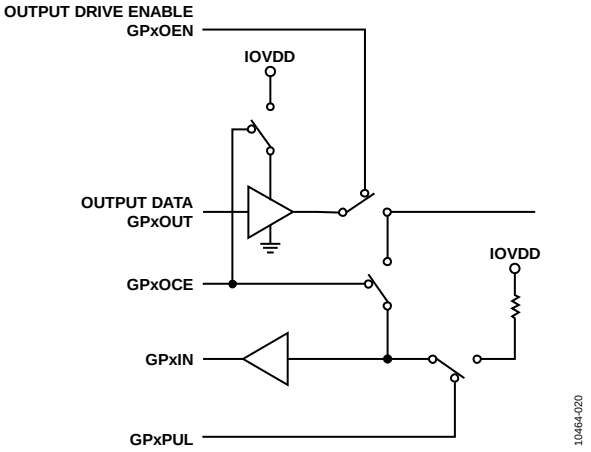

Figure 21. GPIO Structure

DIGITAL I/Os OVERVIEW

The GPIOs are grouped into three ports: Port 0 contains eight GPIOs, Port 1 contains eight GPIOs, and Port 2 contains three GPIOs. Each GPIO can be configured as input, output, or fully open circuit and has an internal pull-up programmable resistor with a drive capability of 1 mA. All I/O pins are functional over the full supply range (IOVDD = 1.8 V to 3.6 V (maximum)), and the logic input voltages are specified as percentages of the supply as follows:

$V_{INL} = 0.2 \times IOVDD\ max$

$V_{INH} = 0.7 \times IOVDD\ min$

The absolute maximum input voltage is IOVDD + 0.3 V, and the typical leakage current of the GPIOs configured as input or open circuit is 50 nA per GPIO. When the ADuCM360/ADuCM361 enters a power saving mode, the GPIO pins retain their states. Note that a driving peripheral cannot drive the pin. That is, if the UART is driving the pin upon entry to deep sleep, it is isolated from the pin and power is gated. Its state and control are restored upon wake up.

DIGITAL I/Os OPERATION

I/O Pull-Up Enable

All GPIO pins have an internal pull-up resistor with a drive capability of 1 mA. Using the GPxPUL register, it is possible to enable/disable pull-up registers on the pins when they are configured as inputs. The pull-ups are automatically disabled when the GPIO pin is set as an output or open circuit is enabled.

The pull-up is implemented as a MOS device; therefore, the pull-up resistor value varies with the voltage on the pin.

For low power operation, ensure the digital I/O's are not floating when configured as inputs. Ensure pull-up resistors are enabled to ensure a defined logic state.

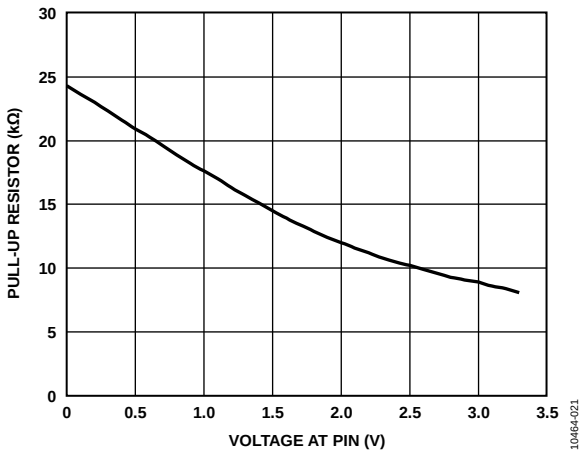

Figure 22. GPIO Pull-Up Resistor Value

I/O Data In

When configured as an input (by default), the GPIO input levels are available in GPxIN.

Open-Circuit Enable

This disables the input paths if the pin is set as an output. To disable the input and not drive the pin, set the open circuit and drive Logic 1. External interrupts are not available when open circuit is enabled.

I/O Data Out

When the GPIOs are configured as outputs, the values in GPxOUT are reflected on the GPIOs.

Bit Set

Bit set mode is used to set one or more GPIO data out without affecting others within a port. Only the GPIO corresponding with the write data bit equal to 1 is set; the remaining GPIO are unaffected.

Bit Clear

Bit clear mode is used to clear one or more GPIO data out without affecting others within a port. Only the GPIO corresponding with write data bit equal to 1 is cleared; the remaining GPIOs are unaffected.

Bit Toggle

Bit toggle mode is used to toggle one or more GPIO data outputs without affecting others within a port. Only the GPIO corresponding to write data bit equal to 1 is toggled; the remaining GPIOx are unaffected.

I/O Data Output Enable

The data output path is enabled; the values in GPxOUT are reflected on the GPIOs.

Table 115. GPIO States

GPxOCE	GPxOEN	GPxOUT[1]	GPIO Input States/Interrupts	GPIO Output States
0	0	X	Always available	Not available. Configured as input.
0	1	X	Always available	Output level depends on OUT (drive Logic 0 or Logic 1).
1	0	X	Always available	Not available. Configured as input.
1	1	0	Not available	Drive 0 out (open drain).
1	1	1	Not available	Output floating (open drain).

[1] X = don't care.

DIGITAL PORT MULTIPLEX

This block provides control over the GPIO functionality of specified pins because some of the pins have a choice to work as GPIO or to have other specific functions.

Table 116. GPIO Multiplex Table

	Configuration Modes			
GPIO	**00**	**01**	**10**	**11**
GP0—GP0CON Controls These Bits				
P0.0	GPIO	SPI1 MISO (GP0CON |= 0x1)		
P0.1	GPIO	SPI1 SCLK (GP0CON |= 0x4)	SCL (GP0CON |= 0x8)	UART RxD (GP0CON |= 0xC)
P0.2	GPIO	SPI1 MOSI (GP0CON |= 0x10)	SDA (GP0CON |= 0x20)	UART TxD (GP0CON |= 0x30)
P0.3	GPIO/IRQ0	SPI1 CS (GP0CON |= 0x40)		
P0.4	GPIO	UART RTS (GP0CON |= 0x100)	ECLK OUT (GP0CON |= 0x200)	
P0.5	GPIO/IRQ1	UART CTS (GP0CON |= 0x400)		
P0.6	GPIO/IRQ2	UART RXD (GP0CON |= 0x1000)		
P0.7	$\overline{\text{POR}}$	GPIO (GP0CON |= 0x4000)	UART TXD (GP0CON |= 0x8000)	
GP1—GP1CON Controls These Bits				
P1.0	GPIO/IRQ3	PWM SYNC (GP1CON |= 0x1)	EXT CLK IN (EXTCLK) (GP1CON |= 0x2)	
P1.1	GPIO/IRQ4	PWM TRIP (GP1CON |= 0x4)		UART DTR (GP1CON |= 0xC)
P1.2	GPIO	PWM0 (GP1CON |= 0x10)		UART RI (GP1CON |= 0x30)
P1.3	GPIO	PWM1 (GP1CON |= 0x40)		UART DSR (GP1CON |= 0xC0)
P1.4	GPIO	PWM2 (GP1CON |= 0x100)	SPI0 MISO (GP1CON |= 0x200)	
P1.5	GPIO/IRQ5	PWM3 (GP1CON |= 0x400)	SPI0 SCLK (GP1CON |= 0x800)	
P1.6	GPIO/IRQ6	PWM4 (GP1CON |= 0x1000)	SPI0 MOSI (GP1CON |= 0x2000)	
P1.7	GPIO/IRQ7	PWM5 (GP1CON |= 0x4000)	SPI0 CS (GP1CON |= 0x8000)	
GP2—GP2CON Controls These Bits				
P2.0	GPIO	SCL (GP2CON |= 0x1)		UART CLKIN (GP2CON |= 0x3)
P2.1	GPIO	SDA (GP2CON |= 0x4)		UART DCD (GP2CON |= 0xC)
P2.2	GPIO			
P2.3	SWCLK (default)[1]			
P2.4	SWD (default)[2]			
P2.5				
P2.6				
P2.7				

[1] P2.3 defaults as serial wire interface pin.
[2] P2.4 defaults as serial wire interface pin.

GPIO MEMORY MAPPED REGISTERS

Table 117. GPIO Interface Memory Address Table Base (Address: 0x40006000)

Offset	Name	Description	Access	Default
0x0000	GP0CON	GPIO Port 0 configuration	RW	0x0000
0x0004	GP0OEN	GPIO Port 0 output enable	RW	0x00
0x0008	GP0PUL	GPIO Port 0 output pull-up enable	RW	0xFF
0x000C	GP0OCE	GPIO Port 0 open-circuit enable	RW	0x00
0x0014	GP0IN[1]	GPIO Port 0 data input	R	N/A
0x0018	GP0OUT	GPIO Port 0 data out	RW	0x00
0x001C	GP0SET	GPIO Port 0 data out set	W	0x00
0x0020	GP0CLR	GPIO Port 0 data out clear	W	0x00
0x0024	GP0 TGL	GPIO Port 0 data out toggle	W	0x00
0x0030	GP1CON	GPIO Port 1 configuration	RW	0x0000
0x0034	GP1OEN	GPIO Port 1 output enable	RW	0x00
0x0038	GP1PUL	GPIO Port 1 output pull-up enable	RW	0xFF
0x003C	GP1OCE	GPIO Port 1 open-circuit enable	RW	0x00
0x0044	GP1IN[1]	GPIO Port 1 data input	R	N/A
0x0048	GP1OUT	GPIO Port 1 data out	RW	0x00
0x004C	GP1SET	GPIO Port 1 data out set	W	0x00
0x0050	GP1CLR	GPIO Port 1 data out clear	W	0x00
0x0054	GP1TGL	GPIO Port 1 pin toggle	W	0x00
0x0060	GP2CON	GPIO Port 2 configuration	RW	0x00
0x0064	GP2OEN	GPIO Port 2 output enable	RW	0x00
0x0068	GP2PUL	GPIO Port 2 output pull-up enable	RW	0xFF
0x006C	GP2OCE	GPIO Port 2 open-circuit enable	RW	0x00
0x0074	GP2IN[1]	GPIO Port 2 data input	R	N/A
0x0078	GP2OUT	GPIO Port 2 data out	RW	0x00
0x007C	GP2SET	GPIO Port 2 data out set	W	0x00
0x0080	GP2CLR	GPIO Port 2 data out clear	W	0x00
0x0084	GP2TGL	GPIO Port 2 pin toggle	W	0x00

[1] Contents of the GPxIN register depend on the digital level on the corresponding pins.

GPIO Configurations Register

Table 118. GPxCON Register Bit Descriptions (GP0CON Address: 0x40006000, GP1CON Address: 0x40006030, GP2CON Address: 0x40006060)

Bits	Name	Description
15:14	CON7	Configuration bits for Px.7 as per Table 116.
13:12	CON6	Configuration bits for Px.6 as per Table 116.
11:10	CON5	Configuration bits for Px.5 as per Table 116.
9:8	CON4	Configuration bits for Px.4 as per Table 116.
7:6	CON3	Configuration bits for Px.3 as per Table 116.
5:4	CON2	Configuration bits for Px.2 as per Table 116.
3:2	CON1	Configuration bits for Px.1 as per Table 116.
1:0	CON0	Configuration bits for Px.0 as per Table 116.

GPIO Output Enable Register

Table 119. GPxOEN Register Bit Descriptions (GP0OEN Address: 0x40006004, GP1OEN Address: 0x40006034, GP2OEN Address: 0x40006064)

Bits	Name	Description
7:0	OEN[7:0]	Output enable. 0: Disables the output on corresponding GPIO on Port X. 1: Enables the output on corresponding GPIO on Port X.

GPIO Pull-Up Register

Table 120. GPxPUL Register Bit Descriptions (GP0PUL Address: 0x40006008, GP1PUL Address: 0x40006038, GP2PUL Address: 0x40006068)

Bits	Name	Description
7:0	PUL[7:0]	Output enable. 0: Disables the internal pull-up on corresponding GPIO on Port X. 1: Enables the internal pull-up on corresponding GPIO on Port X.

GPIO Open-Circuit Enable Register

Table 121. GPxOCE Register Bit Descriptions (GP0OCE Address: 0x4000600C, GP1OCE Address: 0x4000603C, GP2OCE Address: 0x4000606C)

Bits	Name	Description
7:0	OCE[7:0]	Output enable. Sets the GPIO pads on Port X to open-circuit mode.

GPIO Data Input Register

Table 122. GPxIN Register Bit Descriptions (GP0IN Address: 0x40006014, GP1IN Address: 0x40006044, GP2IN Address: 0x40006074)

Bits	Name	Description
7:0	IN[7:0]	Reflects the level on the GPIO pins except when in configured in open circuit.

GPIO Data Out Register

Table 123. GPxOUT Register Bit Descriptions (GP0OUT Address: 0x40006018, GP1OUT Address: 0x40006048, GP2OUT Address: 0x40006078)

Bits	Name	Description
7:0	OUT[7:0]	Data out register. Reads back the value on GPIO outputs. For example, writing GP0OUT = 0x12 drives 1 on P0.1 and P0.4, and the remaining GPIO are low, assuming these pins are configured as outputs. 0: Cleared by user to drive the corresponding GPIO low. 1: Set by user code to drive the corresponding GPIO high.

GPIO Bit Set Register

Table 124. GPxSET Register Bit Descriptions (GP0SET Address: 0x4000601C, GP1SET Address: 0x4000604C, GP2SET Address: 0x4000607)

Bits	Name	Description
7:0	SET[7:0]	Data out register. 0: No action. 1: Set by user code to drive the corresponding GPIO high.

GPIO Bit Clear Register

Table 125. GPxCLR Register Bit Descriptions (GP0CLR Address: 0x40006020, GP1CLR Address: 0x40006050, GP2CLR Address: 0x40006080)

Bits	Name	Description
7:0	CLR[7:0]	Data out register. 0: Cleared by user code; has no effect. 1: Set by user code to drive the corresponding GPIO low.

GPIO Toggle Pin Register

Table 126. GPxTGL Register Bit Descriptions (GP0TGL Address: 0x40006024, GP1TGL Address: 0x40006054, GP2TGL Address: 0x40006084)

Bits	Name	Description
7:0	TGL[7:0]	Toggle pin. 0: Cleared by user code; has no effect. 1: Set by user code to invert the corresponding GPIO.

I²C SERIAL INTERFACE

I²C FEATURES

- Master or slave mode with 2-byte transmit and receive FIFOs
- Supports
  - 7-bit and 10-bit addressing modes
  - Four 7-bit device addresses or one 10-bit address and two 7-bit addresses in the slave
  - Repeated starts in master and slave modes
  - Clock stretching supported for the slave and master
  - Master arbitration
  - Continuous read mode for the master or up to 512 bytes fixed read
  - Internal and external loopback
- Support for DMA in master and slave modes
- Software control on the slave of NACK signal

I²C refers to a communications protocol originally developed by Philips Semiconductors (now NXP semiconductors).

I²C OVERVIEW

The I²C data transfer uses a serial clock pin (SCL) and a serial data pin (SDA). The pins are configured in a wired-AND'ed format that allows arbitration in a multimaster system. Both SCL and SDA are bidirectional and must be connected to a positive supply voltage using a pull-up resistor.

The transfer sequence of an I²C system consists of a master device initiating a transfer by generating a start condition while the bus is idle. The master transmits the slave device address and the direction of the data transfer during the initial address transfer. If the master does not lose arbitration and the slave acknowledges the initial address transfer, the data transfer is initiated. This continues until the master issues a stop condition and the bus becomes idle. Figure 23 shows a typical I²C transfer.

A master device can be configured to generate the serial clock. The frequency is programmed by the user in the serial clock divisor register, I2CDIV. The master channel can be set to operate in fast mode (400 kHz) or standard mode (100 kHz).

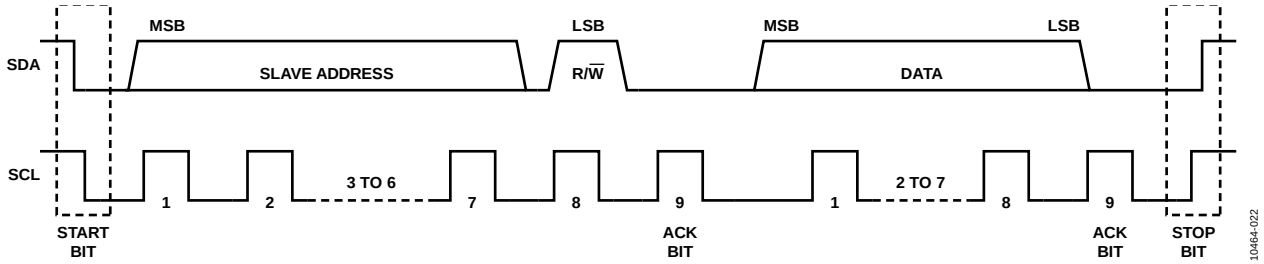

Figure 23. Typical I²C Transfer Sequence

The I²C bus peripheral address in the I²C bus system is programmed by the user. This ID can be modified any time a transfer is not in progress. The user can set up to four slave addresses that are recognized by the peripheral. The peripheral is implemented with a 2-byte FIFO for each transmit and receive shift register. The IRQ and status bits in the control registers are available to signal to the processor core when the FIFOs need to be serviced.

I²C OPERATION

I²C Startup

The following steps are required to set the I²C peripheral running:

1. Configure I²C clock in CLKDIS and CLKCON1 registers.
2. Configure digital pins for I²C operation (P0.1/P0.2, P2.0/P2.1).
3. Configure I²C registers as required for slave or master operation.
4. Enable the I²C slave or master interrupt source as required.

Note that the user should disable the internal pull-up resistors on the I²C pins via the GP0PUL register when using I²C.

Table 127. GPIO Multiplex

GPIO	Configuration Mode (10)
P0.1, P2.0	SCL
P0.2, P2.1	SDA

Addressing Modes

7-Bit Addressing

The I2CID0, I2CID1, I2CID2, and I2CID3 registers contain the slave device IDs. The device compares the four I2CIDx registers to the address byte. To be correctly addressed, the seven MSBs of either ID register must be identical to that of the seven MSBs of the first received address byte. The LSB of the ID registers (the transfer direction bit) is ignored in the process of address recognition.

The master addresses a device using the I2CADR0 register.

10-Bit Addressing

This feature is enabled by setting I2CSCON[1] for master and slave mode.

The 10-bit address of the slave is stored in I2CID0 and I2CID1, where I2CID0 contains the first byte of the address, and the R/W bit and the upper five bits must be programmed to 11110 as shown in Figure 24. I2CID1 contains the remaining eight bits of the 10-bit address. I2CID2 and I2CID3 can still be programmed with 7-bit addresses.

The master communicates to a 10-bit address slave using the I2CADR0/1 registers. The format is described in Figure 24.

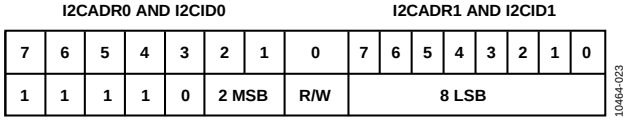

Figure 24. 10-Bit Address Format

A repeated start condition occurs when a second start condition is sent to a slave without a stop condition being sent in between. This allows the master to reverse the direction of the transfer by changing the R/$\overline{W}$ bit without having to give up control of the bus.

An example of a transfer sequence is shown in Figure 25. This is generally used where the first data sent to the part sets up the register address to be read from.

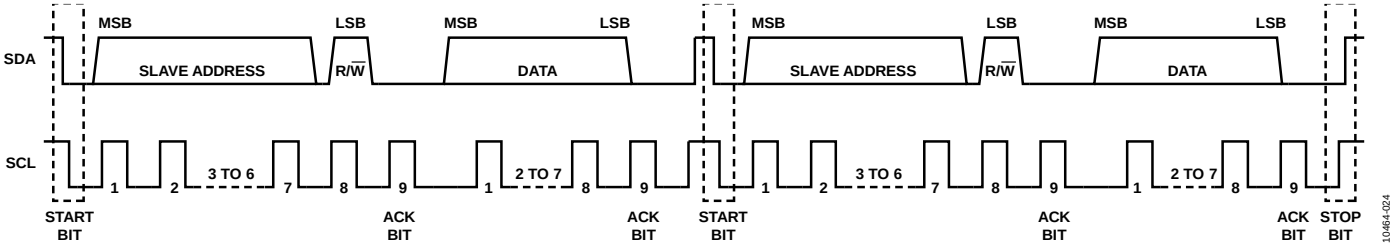

Figure 25. I²C Repeated Start Sequence

On the slave side, an interrupt is generated (if enabled in the I2CSCON register) when a repeated start and a slave address are received. This can be differentiated from receiving a start and slave address by using the status bits START and REPSTART in the I2CSSTA MMR.

On the master side, the master generates a repeated start if the I2CADR0 register is written while the master is still busy with a transaction. After the state machine has started to transmit the device address, it is safe to write to the I2CADR0 register.

For example, if a transaction involving a write, repeated start, and then read/write is required, write to the I2CADR0 register either after the state machine starts to transmit the device address or after the first TXREQ interrupt is received. When the transmit FIFO empties, a repeated start is generated.

Similarly, if a transaction involving a read, repeated start, and then read/write is required, write to the first master address byte register, I2CADR1, either after the state machine starts to transmit the device address or after the first RXREQ interrupt is received. When the requested receive count is reached, a repeated start is generated.

I²C Clock Control

The I²C peripheral is clocked by a gated 16 MHz system clock (UCLK). The CLKCON1[8:6] bits allow the I²C block to be clocked with a slower clock by allowing the 16 MHz clock to be divided—this helps to reduce power.

Note that the minimum I²C clock required for a 100 kHz I²C master clock is 2 MHz.

The minimum I²C clock required for a 400 kHz I²C master clock is 8 MHz.

The I²C master in the system generates the serial clock for a transfer. The master channel can be configured to operate in fast mode (400 kHz) or standard mode (100 kHz).

The bit rate is defined in the I2CDIV MMR as follows:

$$f_{SCL} = f_{I2CCLK}/(LOW + HIGH + 3)$$

where:
$f_{I2CCLK} = f_{UCLK}/(CLKSYSDIV \times I2CCD)$.
f_{UCLK} is the system clock, 16 MHz.
CLKSYSDIV is 1 or 2, depending on the CLKSYSDIV[0] bit setting.
I2CCD is the clock divide value and is set by the CLKCON1[8:6].
HIGH is the high period of the clock, I2CDIV [15:8] = (REQD_HIGH_TIME/UCLK_PERIOD) – 2.
LOW is the low period of the clock, I2CDIV [7:0] = (REQD_LOW_TIME/UCLK_PERIOD) – 1.

For 100 kHz SCL operation, with a low time of 5000 ns and a high time of 5000 ns, and a UCLK frequency of 16 MHz,

HIGH = (5000 ns/(1/16000000)) − 2 = 78 = 0x4E

LOW = (5000 ns/(1/16000000)) − 1 = 79 = 0x4F

f_{SCL} = 16000000/(79 + 78 + 3) = 100 kHz

For 400 kHz SCL operation, with a low time of 1250 ns and a high time of 1250 ns, and a UCLK frequency of 16 MHz,

HIGH = (1250 ns/(1/16000000)) − 2 = 18 = 0x12

LOW = (1250 ns/(1/16000000)) − 1 = 19 = 0x13

f_{SCL} = 16000000/(18 + 19 + 3) = 400 kHz

I²C OPERATING MODES

Master Transfer Initiation

If the master enable bit (I2CMON[0], MASEN) is set, a master transfer sequence is initiated by writing a value to the I2CADRx register. If there is valid data in the I2CMTX register, it is the first byte transferred in the sequence after the address byte during a write sequence.

Slave Transfer Initiation

If the slave enable bit (I2CSCON[0], SLVEN) is set, a slave transfer sequence is monitored for the device address in Register I2CID0, Register I2CID1, Register I2CID2, or Register I2CID3. If the device address is recognized, the part participates in the slave transfer sequence.

Note that a slave operation always starts with the assertion of one of three interrupt sources—read request (RXREQ), write request (TXREQ), or general call (GCINT) interrupt—and the software should always look for a stop interrupt to ensure that the transaction has completed correctly and to deassert the stop interrupt status bit.

Rx/Tx Data FIFOs

The transmit data path consists of a master and slave Tx FIFO, each two bytes deep, I2CMTX and I2CSTX, and a transmit shifter. The transmit status bits in I2CMSTA[1:0] and I2CSSTA[0] denote whether there is valid data in the Tx FIFO. Data from the Tx FIFO is loaded into the Tx shifter when a serial byte begins transmission. If the Tx FIFO is not full during an active transfer sequence, the transmit request bit (TXREQ) in I2CMSTA or I2CSSTA asserts.

In the slave, if there is no valid data to transmit when the Tx shifter is loaded, the transmit underrun status bit (TXUR) asserts (I2CMSTA[12], ISCSSTA[1]).

The master generates a stop condition if there is no data in the transmit FIFO and the master is writing data.

The receive data path consists of a master and slave Rx FIFO, each two bytes deep, I2CMRX and I2CSRX. The receive request interrupt bit (RXREQ) in I2CMSTA or I2CSSTA indicates whether there is valid data in the Rx FIFO. Data is loaded into the Rx FIFO after each byte is received. If valid data in the Rx FIFO is overwritten by the Rx shifter, the receive overflow status bit (RXOF) is asserted (I2CMSTA[9] or I2CSSTA[4]).

Master NACK

When receiving data, the master responds with a NACK if its FIFO is full and an attempt is made to write another byte to the FIFO. This last byte received is not written to the FIFO and is lost.

No Acknowledge from the Slave

If the slave does not want to acknowledge a read access, then simply not writing data into the slave transmit FIFO results in a NACK.

If the slave does not want to acknowledge a master write, assert the NACK bit in the slave control register, I2CSCCON[7].

Normally, the slave will ACK all bytes that are written into the receive FIFO. If the receive FIFO fills up, the slave cannot write further bytes to it, and it will not acknowledge the byte that was not written to the FIFO. The master should then stop the transaction.

The slave does not acknowledge a matching device address if the read/write bit is set and the transmit FIFO is empty. Therefore, there is very little time for the microcontroller to respond to a slave transmit request and the assertion of ACK. It is recommended that EARLYTXR, I2CSCON[5] be asserted for this reason.

General Call

An I^2C general call is for addressing every device on the I^2C bus. A general call address is 0x00 or 0x01. The first byte, address byte is followed by a command byte.

If the address byte is 0x00, then Byte 2, the command byte, can be one of the following:

- 0x6: the I^2C interface (master and slave) is reset. The general call interrupt status asserts, and the general call ID bits, GCID (I2CSSTA[9:8]), are 0x1. User code should take corrective action to reset the entire system or simply reenable the I^2C interface.
- 0x4: the general call interrupt status bit is asserted, and the general call ID bits (GCID) are 0x2.

If the address byte is 0x01, a hardware general call is issued.

- Byte 2 in this case is the hardware master address.

The general call interrupt status bit is set on any general call after the second byte is received, and user code should take corrective action to reprogram the device address.

If GCEN is asserted, the slave always acknowledges the first byte of a general call. It acknowledges the second byte of a general call if the second byte is 0x04 or 0x06 or if the second byte is a hardware general call and HGCEN (I2CSCON[3]) is asserted.

The I2CALT register contains the alternate device ID for a hardware general call sequence. If the hardware general call enable bit, HGCEN, GCEN, and SLVEN are all set, the device recognizes a hardware general call. When a general call sequence is issued and the second byte of the sequence is identical to ALT, the hardware call sequence is recognized for the device.

I^2C Reset Mode

The slave state machine is reset when SLVEN is written to 0.

The master state machine is reset when MASEN is written to 0.

I^2C Test Modes

The device can be placed in an internal loopback mode by setting the LOOPBACK bit (I2CMCON[2]). There are four FIFOs (master Tx and Rx, and slave Tx and Rx), so, in effect, the I^2C peripheral can be set up to talk to itself. External loopback can be performed if the master is set up to address the slave's address.

I^2C Low Power Mode

If the master and slave are both disabled (MASEN = SLVEN = 0), the I^2C section is off. To fully power down the I^2C block, the clock to the I^2C section of the chip should be disabled by setting CLKCON1[8:6] = 111 and CLKDIS[2] = 1.

DMA Requests

Four DMA channels are required to service the I^2C master and slave. DMA enable bits are provided in the slave control register and in the master control register. The following is example code showing how to configure the DMA controller for the four different configurations.

Example Code: I²C Slave Write DMA Cycles

```
void I2CSLAVETXDMAINIT(void)
{
pADI_GP0->GPCON = 0x28;                                    // Configure P0.[1:2] as I2C pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                            // Enable clock to I2C block
pADI_I2C->I2CSCON = I2CSCON_SLVEN                          // Enable Slave mode
+ I2CSCON_EARLYTXR                                         // Enable early Tx request interrupt
source
+ I2CSCON_IENSTOP    + I2CSCON_IENRX                       // Enable stop and Rx interrupts
+ I2CSCON_IENTX    + I2CSCON_IENREPST                      // Enable Tx and repeated Start
interrupts
+ 0x4000;                                                  // Enable Tx DMA
pADI_I2C->I2CID0 = 0xA2;                                   // Set Slave ID0 register
pADI_I2C->I2CID1 = 0xA4;                                   // Set Slave ID1 register
pADI_I2C->I2CID2 = 0xA6;                                   // Set Slave ID2 register
pADI_I2C->I2CID3 = 0xA0;                                   // Set Slave ID3 register
pADI_I2C->I2CFSTA = 0x100;                                 // Flush Slave Tx FIFO
pADI_I2C->I2CFSTA &= ~0x100;
                                                           // Enable I2C Slave,
Dma_Init();                                                // See ADC DMA example for this
function

I2CSLAVEDMAWRITE(uxI2CTXData, 16);
pADI_DMA->DMACFG = 0x1;                                    // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x10;                                 // Enable I2C_TX_DMA Channel
NVIC_EnableIRQ(DMA_I2CS_TX_IRQn);                          // I2C Tx DMA interrupt sources  -
M360
}

void I2CSLAVEDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                                           // Common configuration of all the
descriptors used here
    Desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;   //cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x1;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                           //  TX Primary Descriptor
Desc.srcEndPtr                     = (unsigned int)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                    =  (unsigned int)&I2CSTX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
```

```
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_I2C0_SLV_TX,FALSE) = Desc;
}
                                                          // Slave DMA Tx IRQ handler
void DMA_I2C0_STX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CS_TX_IRQn);                        // Clear Interupt source
}
```

Example Code: I²C Slave Read DMA Cycles

```
void I2CSLAVERXDMAINIT(void)
{
pADI_GP0->GPCON = 0x28;                                   // Configure P0.[1:2] as I2C pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                           // Enable clock to I2C block
pADI_I2C->I2CSCON = I2CSCON_SLV                           // Enable Slave mode
+ I2CSCON_EARLYTXR                                        // Enable early Tx request interrupt
source
+ I2CSCON_IENSTOP    + I2CSCON_IENRX                      // Enable stop and Rx interrupts
+ I2CSCON_IENTX    + I2CSCON_IENREPST                     // Enable Tx and repeated Start
interrupts
+ 0x2000;                                                 // Enable Rx DMA
pADI_I2C->I2CID0 = 0xA2;                                  // Set Slave ID0 register
pADI_I2C->I2CID1 = 0xA4;                                  // Set Slave ID1 register
pADI_I2C->I2CID2 = 0xA6;                                  // Set Slave ID2 register
pADI_I2C->I2CID3 = 0xA0;                                  // Set Slave ID3 register
                                                          // Enable I2C Slave,
Dma_Init();

I2CSLAVEDMAREAD(uxI2CRXData, 16);
pADI_DMA->DMACFG = 0x1;                                   // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x20;                                // Enable I2C_RX_DMA Channel
NVIC_EnableIRQ(DMA_I2CS_RX_IRQn);                         // I2C Rx DMA interrupt sources  -
M360
}

void I2CSLAVEDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                                          // Common configuration of all the
descriptors used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    desc.ctrlcfg.bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;
```

```
                                                                    // RX Primary Descriptor
Desc.srcEndPtr                      = (unsigned int)&I2CSRX;
    Desc.destEndPtr                     = (unsigned int)(pucRX_DMA + iNumRX - 0x1); //
    Desc.ctrlCfg.Bits.n_minus_1         = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc           = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc           = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_I2C0_SLV_RX,FALSE) = Desc;
}
                                                                    // I2C DMA Rx IRQ handler
void DMA_I2C0_SRX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CS_RX_IRQn);                                  // Clear Interupt source
}
```

Example Code: I²C Master Write DMA Cycles

```
void I2CMASTERTXDMAINIT(void)
{
pADI_GP0->GPCON = 0x28;                                             // Configure P0.[1:2] as I2C
pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                                     // Enable clock to I2C block
                                                                    // Enable I2C master, IRQ on
byte Received
                                                                    // Enable IRQ on Transmit
complete, Arbitration lost, NACK received, Complete transaction (Stop detected)
                                                                    // Decrement Tx FIFO status
on byte full transmitted.
pADI_I2C->I2CMCON = I2CMCON_MASEN + I2CMCON_IENRX
+ I2CMCON_IENTX        + I2CMCON_IENALOST   + I2CMCON_IENNACK
+ I2CMCON_IENCMP
+ 0x800;                                                           // Enable Rx DMA
pADI_I2C->I2CFSTA = 0x200;                                          // Flush Master Tx FIFO
pADI_I2C->I2CFSTA &= ~0x200;


                                                                    // Enable I2C Master,
Dma_Init();


I2CMASTERDMAWRITE(uxI2CTXData, 16);
pADI_DMA->DMACFG = 0x1;                                             // Enable DMA mode in DMA
controller
pADI_DMA->DMAENSET = 0x40;                                          // Enable I2C_TX_DMA Channel
NVIC_EnableIRQ(DMA_I2CM_TX_IRQn);                                   // I2C Tx DMA interrupt
sources- M360
pADI_I2C->I2CADR0 =    0xA0;
}

void I2CMASTERDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                                    // Common configuration of
all the descriptors used here
```

```
    Desc.ctrlCfg.bits.cycle_ctrl       = cyclectrl_basic;   //cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x1;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                          // TX Primary Descriptor
Desc.srcEndPtr                     = (unsigned int)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                    =  (unsigned int)&I2CMTX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_I2C0_MST_TX,FALSE) = Desc;
}
                                                          // Master DMA Tx IRQ handler
void DMA_I2C0_MTX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CM_TX_IRQn);                        // Clear Interupt source
}
```

Example Code: I²C Master Read DMA Cycles

```
void I2CMASTERRXDMAINIT(unsigned char ucNumBytes)
{
pADI_GP0->GPCON = 0x28;                                   // Configure P0.[1:2] as I2C pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                           // Enable clock to I2C block
                                                          // Enable I2C master, IRQ on byte
Received
                                                          // Enable IRQ on Transmit complete,
Arbitration lost, NACK received, Complete transaction (Stop detected)
                                                          // Decrement Tx FIFO status on byte
full transmitted.
pADI_I2C->I2CMCON = I2CMCON_MASEN + I2CMCON_IENRX
+ I2CMCON_IENTX        + I2CMCON_IENALOST   + I2CMCON_IENNACK
+ I2CMCON_IENCMP
+ 0x400;                                                  // Enable Rx DMA
pADI_I2C->I2CMRXCNT = ucNumBytes;

Dma_Init();

I2CMASTERDMAREAD(uxI2CRXData, ucNumBytes);
pADI_DMA->DMACFG = 0x1;                                   // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x80;                                // Enable I2C_RX_DMA Channel
NVIC_EnableIRQ(DMA_I2CM_RX_IRQn);                         // I2C Rx DMA interrupt sources-M360
pADI_I2C->I2CADR0 =    0xA1;
}

void I2CMASTERDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
```

```
{
DmaDesc Desc;

                                                                    // Common configuration of
all the descriptors used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                                    //  RX Primary   Descriptor
Desc.srcEndPtr                     = (unsigned int)&I2CMRX;
    Desc.destEndPtr                    = (unsigned int)(pucRX_DMA + iNumRX - 0x1);
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc          = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_I2C0_MST_RX,FALSE) = Desc;
}
                                                                    // I2C DMA Rx IRQ handler
void DMA_I2C0_MRX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CM_RX_IRQn);                                  // Clear Interupt source
}
```

I²C MEMORY MAPPED REGISTERS

The I²C interface MMRs are based at Address 0x40003000.

Table 128. I²C Interface Memory Address Table Master Registers (Base Address: 0x40003000)

Offset	Name	Description	Access	Default
0x0000	I2CMCON	Master control register. Read/write register.	RW	0x0000
0x0004	I2CMSTA	Master status, error, and IRQ register.	R	0x0000
0x0008	I2CMRX	Master receive data register.	R	0x0000
0x000C	I2CMTX	Master transmit data register.	W	0x0000
0x0010	I2CMRXCNT	Master receive data count register.	RW	0x0000
0x0014	I2CMCRXCNT	Master current receive data count register.	R	0x0000
0x0018	I2CADR0	First master address byte register.	RW	0x0000
0x001C	I2CADR1	Second master address byte register (10-bit addresses only).	RW	0x0000
0x0024	I2CDIV	Serial clock period divisor register.	RW	0x1F1F

I²C Register Map Slave Registers

Table 129. I²C Interface Memory Address Table Slave Registers (Base Address: 0x40003000)

Offset	Name	Description	Access	Default
0x0028	I2CSCON	Slave control register.	RW	0x0000
0x002C	I2CSSTA	Slave status, error, and IRQ register.	R	0x0001
0x0030	I2CSRX	Slave receive data register.	R	0x0000
0x0034	I2CSTX	Slave transmit data register.	W	0x0000
0x0038	I2CALT	Hardware general call ID register.	RW	0x0000
0x003C	I2CID0	First slave address device ID.	RW	0x0000
0x0040	I2CID1	Second slave address device ID.	RW	0x0000
0x0044	I2CID2	Third slave address device ID.	RW	0x0000
0x0048	I2CID3	Fourth slave address device ID.	RW	0x0000

I²C Register Map Shared Registers

Table 130. I²C Interface Memory Address Table Shared Registers (Base Address: 0x40003000)

Offset	Name	Description	Access	Default
0x004C	I2CFSTA	Master and slave Rx and Tx FIFO status register.	RW	0x0000

I²C Master Control Register

Address: 0x40003000, Reset: 0x0000, Name: I2CMCON

Table 131. I2CMCON Register Bit Descriptions

Bits	Name	Description
15:12	Reserved	Reserved bits.
11	TXDMA	Enable master Tx DMA request. 0: Disable Tx DMA mode. 1: Enable I²C master Tx DMA requests.
10	RXDMA	Enable master Rx DMA request. 0: Disable Rx DMA mode. 1: Enable I²C master Rx DMA requests.
9	Reserved	Always clear to 0.
8	IENCMP	Transactions completed (or stop detected) interrupt enable. 0: Disable. 1: Enable a transaction completed interrupt. If this bit is asserted, an interrupt is generated when a stop is detected.
7	IENNACK	NACK received interrupt enable. 0: Disable. 1: Enable.
6	IENALOST	Arbitration lost interrupt enable. 0: Disable. 1: Enable.
5	IENTX	Transmit request interrupt enable. 0: Disable. 1: Enable.
4	IENRX	Receive request interrupt enable. 0: Disable. 1: Enable.
3	STRETCH	Stretch SCL enable. Setting this bit instructs the device that if SCL is 0, hold it at 0; or, if SCL is 1, then when it next goes to 0, hold it at 0. 0: Disable. 1: Enable.
2	LOOPBACK	Internal loopback enable. SCL and SDA out of the device are multiplexed onto their corresponding inputs. Note that it is also possible for the master to loopback a transfer to the slave as long as the device address corresponds, that is, external loopback. 0: Disable. 1: Enable.
1	COMPETE	Start back off disable. 0: Disable. 1: Enables the device to compete for ownership even if another device is currently driving a start condition.
0	MASEN	Master enable. The master should be disabled when not in use because this gates the clock to the master and saves power. This bit should not be cleared until a transaction has completed (see the TCOMP bit in the master status register (Table 132)). Note that writable register bits are not reset by this bit. 0: Disable master and the master state machine is held in reset. 1: Enable master.

I²C Master Status Register

Address: 0x40003004, Reset: 0x0000, Name: I2CMSTA

Table 132. I2CMSTA Register Bit Descriptions

Bits	Name	Description
15:13	Reserved	Reserved bits. Returns 0 when read.
12	TXUR	Transmit FIFO underrun. Set to 1 when the I²C master ends the transaction due to a Tx FIFO empty condition. This bit is only set when IENTX (I2CMCON[10]) is set. Cleared to 0 on a read of the I2CMSTA register.
11	MSTOP	Stop driven by I²C master. Set to 1 when the I²C master drives a stop condition on the I²C bus, which indicates a transaction completion, Tx underrun, Rx overflow, or a NACK by the slave. It is different from TCOMP because it is not set when the stop condition occurs due to any other master on the I²C bus. This bit does not generate an interrupt. See the TCOMP description for available interrupts related to the stop condition. Cleared to 0 on a read of the I2CMSTA register.
10	LINEBUSY	Line is busy. Set to 1 when a start is detected on the I²C bus. Cleared to 0 when a stop is detected on the I²C bus.
9	RXOF	Receive FIFO overflow. Set to 1 when a byte is written to the receive FIFO if the FIFO is already full. Cleared to 0 on a read of the I2CMSTA register.
8	TCOMP	Transaction complete (or stop detected). (Can drive an interrupt.) This bit only asserts if the master is enabled (MASEN = 1). This bit should be used to determine when it is safe to disable the master. It can also be used to wait for another master's transaction to complete on the I²C bus when this master loses arbitration. Set to 1 when a stop condition is detected on the I²C bus. If IENCMP is 1 (I2CMCON [8]), an interrupt is generated when this bit asserts. Cleared to 0 on a read of the I2CMSTA register.
7	NACKDATA	NACK received in response to data write. (Can drive an interrupt.) Set to 1 when a NACK is received in response to a data write transfer. If IENNACK is 1 (I2CMCON [7]), an interrupt is generated when this bit asserts. Cleared to 0 on a read of the I2CMSTA register.
6	BUSY	Master busy. Set to 1 when the master state machine is servicing a transaction. Cleared to 0 if the state machine is idle or another device has control of the I²C bus.
5	ALOST	Arbitration lost. (Can drive an interrupt.) Set to 1 if the master loses arbitration. If IENALOST is 1 (I2CMCON [6]), an interrupt is generated when this bit asserts. Cleared to 0 on a read of the I2CMSTA register.
4	NACKADDR	NACK received in response to an address. (Can drive an interrupt.) Set to 1 if a NACK is received in response to an address. If IENNACK is 1 (I2CMCON [7]), an interrupt is generated when this bit asserts. Cleared to 0 on a read of the I2CMSTA register.
3	RXREQ	Receive request. (Can drive an interrupt.) Set to 1 when there is data in the receive FIFO. If IENRX is 1 (I2CMCON [4]), an interrupt is generated when this bit asserts. Cleared to 0 on a read of the I2CMRX register.
2	TXREQ	Transmit request. (Can drive an interrupt.) Set to 1 when the direction bit is 0 and the transmit FIFO is either empty or not full. If IENTX is 1 (I2CMCON [5]), an interrupt is generated when this bit asserts. Cleared to 0 on a write to the I2CMTX register.
1:0	TXFSTA	Transmit FIFO status. 00: EMPTY: FIFO empty. 01: Reserved. 10: ONEBYTE: one byte in FIFO. 11: FULL: FIFO full.

I²C Master Receive Register

Address: 0x40003008, Reset: 0x0000, Name: I2CMRX

Table 133. I2CMRX Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bits. Returns 0 when read.
7:0	VALUE	Receive register. Read only. Set to 0 by default. This register allows access to the receive data FIFO. The FIFO can hold two bytes.

I²C Master Transmit Register

Address: 0x4000300C, Reset: 0x0000, Name: I2CMTX

I2CMTX is a write only register. If read, the resulting values are undefined.

Table 134. I2CMTX Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bits. Returns 0 when read.
7:0	VALUE	Transmit register. Set to 0 by default. Writing to this register loads a byte to the Tx FIFO. The FIFO can hold two bytes.

I²C Master Receive Data Count Register

Address: 0x400030010, Reset: 0x0000, Name: I2CMRXCNT

Table 135. I2CMRXCNT Register Bit Descriptions

Bits	Name	Description
15:9	Reserved	Reserved bits. Returns 0 when read.
8	EXTEND	Extended read: Use this bit if greater than 256 bytes are required on a read. 0: Disable extended read. 1: Enable extended read. For example, to receive 412 bytes, write 0x100 (EXTEND = 1) to this register (I2CMRXCNT). Wait for the first byte to be received, and then check the I2CMCRXCNT register for every byte received thereafter. When I2CMCRXCNT returns to 0, 256 bytes have been received. Next, write 0x09C (412 − 256 = 156 decimal (equal to 0x9C), with the EXTEND bit set to 0) to this register (I2CMRXCNT).
7:0	COUNT	Receive count. Program the number of bytes required minus one to this register. If just one byte is required, write 0 to this register. If greater than 256 bytes are required, use EXTEND.

I²C Master Current Receive Count Register

Address: 0x40003014, Reset: 0x0000, Name: I2CMCRXCNT

Table 136. I2CMCRXCNT Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bits. Returns 0 when read.
7:0	VALUE	Current receive count. This register gives the total number of bytes received so far. If 256 bytes are requested, this register reads 0 when the transaction has completed.

I²C Master First Address Byte Register

Address: 0x40003018, Reset: 0x0000, Name: I2CADR0

Table 137. I2CADR0 Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bits. Returns 0 when read.
7:0	VALUE	Address byte. If a 7-bit address is required, I2CADR0[7:1] is programmed with the address and I2CADR0[0] is programmed with the direction (read or write). If a 10-bit address is required, I2CADR0[7:3] is programmed with 11110, I2CADR0[2:1] is programmed with the two MSBs of the address, and I2CADR0[0] is programmed with the direction (read or write).

I²C Master Second Address Byte Register

Address: 0x4000301C, Reset: 0x0000, Name: I2CADR1

Table 138. I2CADR1 Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bits. Returns 0 when read.
7:0	VALUE	Address byte. This register is only required when addressing a slave with 10-bit addressing. I2CADR1[7:0] is programmed with the lower eight bits of the address.

I²C Serial Clock Period Divider Register

Address: 0x40003024, Reset: 0x1F1F, Name: I2CDIV

Table 139. I2CDIV Register Bit Descriptions

Bits	Name	Description
15:8	HIGH	Serial clock high time. This register controls the clock high time. See the I²C Clock Control section for more details.
7:0	LOW	Serial clock low time. This register controls the clock low time. See the I²C Clock Control section for more details.

I²C Slave Control Register

Address: 0x40003028, Reset: 0x0000, Name: I2CSCON

Table 140. I2CSCON Register Bit Descriptions

Bits	Name	Description
15	Reserved	Reserved bit.
14	TXDMA	Enable slave Tx DMA request. 0: Disable DMA mode. 1: Enable I²C slave DMA requests.
13	RXDMA	Enable slave Rx DMA request. 0: Disable DMA mode. 1: Enable I²C slave DMA requests.
12	IENREPST	Repeated start interrupt enable. 0: Disable an interrupt when the REPSTART status bit asserts. 1: Generate an interrupt when the REPSTART status bit asserts.
11	Reserved	Reserved.
10	IENTX	Transmit request interrupt enable. 0: Disable transmit request interrupt. 1: Enable transmit request interrupt.
9	IENRX	Receive request interrupt enable. 0: Disable receive request interrupt. 1: Enable receive request interrupt.

Bits	Name	Description
8	IENSTOP	Stop condition detected interrupt enable. 0: Disable stop condition detect interrupt. 1: Enable stop condition detect interrupt.
7	NACK	NACK next communication. 0: By default. 1: Allow the next communication to be NACK'ed. This can be used, for example, if during a 24xx I²C serial EEPROM style access, an attempt was made to write to a read only or nonexistent location in system memory. That is the indirect address in a 24xx I²C serial EEPROM style write pointed to an unwritable memory location.
6	STRETCH	Stretch SCL enable. 0: Disable clock stretching. 1: Tell the device that if SCL is 0, hold it at 0; or, if SCL is 1, when it next goes to 0, hold it at 0.
5	EARLYTXR	Early transmit request mode. 0: Enable a transmit request just after the negative edge of the direction bit (R/W) SCL clock pulse. 1: Enable a transmit request just after the positive edge of the direction bit (R/W) SCL clock pulse.
4	GCSBCLR	General call status bit clear. The general call status and general call ID bits can only be reset by a write to this bit or a full reset. General call ID = I2CSSTA[9:8]. 0: Disable general call status clear. 1: Clear the general call status and general call ID bits.
3	HGCEN	Hardware general call enable. 0: Disable hardware general call. 1: When this bit and the general call enable bit are set after the device receives a general call, Address 0x00 and a data byte check the contents of the I2CALT against the receive shift register. If the contents match, the device has received a hardware general call. This is used if a device needs urgent attention from a master device without knowing which master it needs to turn to. This is a broadcast message to all master devices in the bus. The device that requires attention embeds its own address into the message. The LSB of the I2CALT register should always be written to a 1.
2	GCEN	General call enable. 0: Disable the I²C slave to ACK an I²C general call, Address 0x00 (write). 1: Enable the I²C slave to ACK an I²C general call, Address 0x00 (write).
1	ADR10EN	Enable 10-bit addressing. One 10-bit address is supported by the slave and is stored in I2CID0 and I2CID1, where I2CID0 contains the first byte of the address and the upper five bits must be programmed to 11110. I2CID2 and I2CID3 can be programmed with 7-bit addresses at the same time. 0: If this bit is clear, the slave can support four slave addresses, programmed in Register I2CID0 to Register I2CID3. 1: Enable 10-bit addressing.
0	SLVEN	Slave enable. Note that writable register bits are not reset. 0: Disable the slave, and all slave state machine flops are held in reset. 1: Enable slave.

I²C Slave Status Register

Address: 0x4000302C, Reset: 0x0001, Name: I2CSSTA

Table 141. I2CSSTA Register Bit Descriptions

Bits	Name	Description
15	Reserved	Reserved bit. Returns 0 when read.
14	START	Start and matching address. Set to 1 if a start is detected on SCL/SDA and one of the following is true: the device address is matched, a general call (GC = 0000_0000) code is received while GC is enabled, a high speed (HS = 0000_1XXX) code is received. Cleared to 0 on receipt of either a stop or start condition.
13	REPSTART	Repeated start and matching address. (Can drive an interrupt.) Set to 1 if START (I2CSSTA[14]) is already asserted and then a repeated start is detected. Cleared to 0 when read or on receipt of a stop condition.

Bits	Name	Description
12:11	IDMAT	Device ID matched. Set to 00 when received address matched ID Register 0. Set to 01 when received address matched ID Register 1. Set to 10 when received address matched ID Register 2. Set to 11 when received address matched ID Register 3.
10	STOP	Stop after start and matching address. (Can drive an interrupt.) Set to 1 if the slave device received a stop condition after a previous start condition and a matching address. If IENSTOP (I2CSCON[8]) is set, the slave interrupt request is asserted when this bit is set. Cleared to 0 by a read of the status register.
9:8	GCID	General call ID. These status bits are not cleared by a general call reset. These bits are cleared to 0 by writing 1 to GCSBCLR (I2CSCON[4]). 00: No general call. 01: General call reset and program address. 10: General call program address. 11: General call matching alternative ID.
7	GCINT	General call interrupt. (Always drives an interrupt.) If the interrupt was a general call reset, all registers are at their default values. If the interrupt was a hardware general call, the Rx FIFO holds the second byte of the general call, and this can be compared with the I2CALT register. Set to 1 if the slave device receives a general call of any type. Cleared to 0 by writing 1 to GCSBCLR (I2CSCON[4]).
6	BUSY	Slave busy. Set to 1 if the slave device receives an I^2C start condition. Cleared to 0 by hardware on any of the following conditions: the address does not match an ID register, the slave device receives an I^2C stop condition, or if a repeated start address doesn't match.
5	NACK	NACK generated by the slave. Used to indicate that the slave responded to its device address with a NACK. Set to 1 under any of the following conditions: if there was no data to transmit and the sequence was a slave read, the device address is NACK'ed; if the NACK bit was set in the slave control register and the device was addressed. Cleared to 0 on a read of the I2CSSTA register.
4	RXOF	Receive FIFO overflow. Set to 1 when a byte is written to the receive FIFO if the FIFO is already full. Cleared to 0 on a read of the I2CSSTA register.
3	RXREQ	Receive request. (Can drive an interrupt.) Set on the falling edge of the SCL clock pulse that clocks in the last data bit of a byte. Set to 1 when the receive FIFO is not empty. Cleared to 0 when the receive FIFO is read or flushed.
2	TXREQ	Transmit request. (Can drive an interrupt.) If EARLYTXR = 0, this bit is set when the direction bit for a transfer is received high. Thereafter, as long as the transmit FIFO is not full, this bit remains asserted. Initially, it is asserted on the negative edge of the SCL pulse that clocks in the direction bit (if the device address also matched). If EARLYTXR = 1, this bit is set when the direction bit for a transfer is received high. Thereafter, as long as the transmit FIFO is not full, this bit remains asserted. Initially, it is asserted after the positive edge of the SCL pulse that clocks in the direction bit (if the device address also matched). Cleared to 0 on a read of the I2CSSTA register.
1	TXUR	Transmit FIFO underrun. Set to 1 if a master requests data from the device and the Tx FIFO is empty for the rising edge of SCL. Cleared to 0 on a write of the I2CSTX register.
0	TXFSEREQ	Tx FIFO status. Not valid if EARLYTXR = 1. Set to 1 whenever the slave Tx FIFO is empty (default). Cleared to 0 when the slave Tx FIFO is not empty.

I²C Slave Receive Data Register

Address: 0x40003030, Reset: 0x0000, Name: I2CSRX

Table 142. I2CSRX Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bit. Returns 0 when read.
7:0	VALUE	Receive register.

I²C Slave Transmit Data Register

Address: 0x40003034, Reset: 0x0000, Name: I2CSTX

I2CSTX is a write only register. If this register is read, the resulting values are undefined.

Table 143. I2CSTX Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bit. Returns 0 when read.
7:0	VALUE	Transmit register.

I²C Slave Alt Register

Address: 0x40003038, Reset: 0x0000, Name: I2CALT

Table 144. I2CALT Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved bit. Returns 0 when read.
7:0	VALUE	This register is used in conjunction with HGCEN (I2CSCON[3]) to match a master generating a hardware general call. It is used in the case where a master device cannot be programmed with a slave's address and, instead, the slave must recognize the master's address.

I²C Slave ID Registers

Table 145. I2CIDx Register Bit Descriptions (I2CID0 Address: 0x4000303C, I2CID1 Address: 0x40003040, I2CID2 Address: 0x40003044, I2CID3 Address: 0x40003048)

Bits	Name	Description
15:8	Reserved	Reserved bit. Returns 0 when read.
7:0	VALUE	There are four of these registers: I2CID0 to I2CID3. I2CID[7:1] is programmed with the device ID. I2CID[0] is don't care. See I2CSCON[1] in Table 140 for a description of how these registers are programmed with a 10-bit address.

I²C FIFO Status Register

Address: 0x4000304C, Reset: 0x0000, Name: I2CFSTA

Table 146. I2CFSTA Register Bit Descriptions

Bits	Name	Description
15:10	Reserved	Reserved bit. Returns 0 when read.
9	MFLUSH	0: Normal operation. 1: Flush the master transmit FIFO. The master transmit FIFO must be flushed if arbitration is lost or a slave responds with a NACK.
8	SFLUSH	0: Normal operation 1: Flush the slave transmit FIFO.
7:6	MRXFSTA	Master receive FIFO status. The status is a count of the number of bytes in a FIFO. 00: FIFO empty. 01: one byte in the FIFO. 10: two bytes in the FIFO. 11: Reserved.

Bits	Name	Description
5:4	MTXFSTA	Master transmit FIFO status. The status is a count of the number of bytes in a FIFO. 00: FIFO empty. 01: one byte in the FIFO. 10: two bytes in the FIFO. 11: Reserved.
3:2	SRXFSTA	Slave receive FIFO status. The status is a count of the number of bytes in a FIFO. 00: FIFO empty. 01: one byte in the FIFO. 10: two bytes in the FIFO. 11: Reserved.
1:0	STXFSTA	Slave transmit FIFO status. The status is a count of the number of bytes in a FIFO. 00: FIFO empty. 01: one byte in the FIFO. 10: two bytes in the FIFO. 11: Reserved.

SERIAL PERIPHERAL INTERFACES

SPI FEATURES

Two complete hardware serial peripheral interfaces with the following standard SPI features:

- Serial clock phase mode and serial clock polarity mode
- LSB first transfer option
- Loopback mode
- Master or slave mode
- Transfer and interrupt mode
- Continuous transfer mode
- Tx/Rx FIFO
- Interrupt mode, interrupt after one, two, three, or four bytes
- Rx overflow mode and Tx underrun mode
- Open-circuit data output mode
- Full-duplex communications supported (simultaneous transmit/receive)

SPI OVERVIEW

The ADuCM360/ADuCM361 integrates two complete hardware serial peripheral interfaces (SPI). SPI is an industry standard, synchronous serial interface that allows eight bits of data to be synchronously transmitted and simultaneously received, that is, full duplex. The two SPIs implemented on the ADuCM360/ADuCM361 can operate to a maximum bit rate of 8 Mbps in both master and slave mode.

SPI1 has an additional DMA feature. It has two DMA channels that interface with a µDMA controller of the ARM Cortex-M3 processor. One DMA channel is used for transmit and the other for receive.

Note that SPI0 does not support DMA.

SPI OPERATION

The SPI port can be configured for master or slave operation and consists of four pins: MISO, MOSI, SCLK, and $\overline{CS}$.

Note that the GPIOs used for SPI communication must be configured in SPI mode before enabling the SPI peripheral and that the internal pull-up resistors on the SPI pins should be disabled via the GPxPUL registers when using the SPI.

MISO (Master In, Slave Out) Pin

The MISO pin is configured as an input line in master mode and an output line in slave mode. The MISO line on the master (data in) should be connected to the MISO line in the slave device (data out). The data is transferred as byte-wide (8-bit) serial data, MSB first.

MOSI (Master Out, Slave In) Pin

The MOSI pin is configured as an output line in master mode and an input line in slave mode. The MOSI line on the master (data out) should be connected to the MOSI line in the slave device (data in). The data is transferred as byte-wide (8-bit) serial data, MSB first.

SCLK (Serial Clock I/O) Pin

The master serial clock (SCLK) synchronizes the data being transmitted and received through the MOSI SCLK period. Therefore, a byte is transmitted/received after eight SCLK periods. The SCLK pin is configured as an output in master mode and as an input in slave mode.

In master mode, the polarity and phase of the clock are controlled by the SPIxCON register, and the bit rate is defined in the SPIxDIV register as follows:

$$f_{SERIALCLOCK} = \frac{UCLK / DIV}{2 \times (1 + SPIxDIV)}$$

Where *UCLK/DIV* is the 16 MHz system clock divided by the factor set in CLKCON1 and CLKSYSDIV registers.

It is possible to divide the clocks to SPI0 and SPI1 separately:

- CLKCON1[2:0] and CLKDIS[0] control UCLK/DIV to SPI0.
- CLKCON1[5:3] and CLKDIS[1] control UCLK/DIV to SPI1.

By reducing the clock rate to the SPI blocks, it is possible to reduce the power consumption of the SPI block.

The maximum data rate is 8 Mbps.

In slave mode, the SPIxCON register must be configured with the phase and polarity of the expected input clock. The slave accepts data from an external master up to 8 Mbps.

In both master and slave mode, data is transmitted on one edge of the SCLK signal and sampled on the other. Therefore, it is important that the polarity and phase be configured the same for the master and slave devices.

Chip Select ($\overline{CS}$ Input) Pin

In SPI slave mode, a transfer is initiated by the assertion of $\overline{CS}$, which is an active low input signal. The SPI port then transmits and receives 8-bit data until the transfer is concluded by deassertion of $\overline{CS}$. In slave mode, $\overline{CS}$ is always an input.

In SPI master mode, the $\overline{CS}$ is an active low output signal. It asserts itself automatically at the beginning of a transfer and deasserts itself upon completion.

SPI TRANSFER INITIATION

In master mode, the transfer and interrupt mode bit, SPIxCON[6] determines the manner in which an SPI serial transfer is initiated. If the mode bit is set, a serial transfer is initiated after a write to the Tx FIFO. If the mode bit is cleared, a serial transfer is initiated after a read of the Rx FIFO; the read must be done while the SPI interface is idle. A read done during an active transfer does not initiate another transfer.

For any setting of SPIxCON[1] and SPIxCON[6], the SPI simultaneously receives and transmits data. Therefore, during data transmission, the SPI is also receiving data and filling up the Rx FIFO. If the data is not read from the Rx FIFO, the overflow interrupt occurs when the FIFO starts to overflow. If the user does not want to read the Rx data or receive overflow interrupts, SPIxCON[12] can be set and the receive data is not saved to the Rx FIFO.

Similarly, when the user wants to only receive data and does not want to write data to the Tx FIFO, SPIxCON[13] can be set to avoid receiving underrun interrupts from the Tx FIFO.

Tx Initiated Transfer

For transfers initiated by a write to the Tx FIFO, the SPI starts transmitting as soon as the first byte is written to the FIFO, irrespective of the configuration in SPIxCON[15:14]. The first byte is immediately read from the FIFO, written to the Tx shift register, and the transfer commences.

If the continuous transfer enable bit, SPIxCON[11], is set, the transfer continues until no valid data is available in the Tx FIFO. There is no stall period between transfers where $\overline{CS}$ is deasserted; $\overline{CS}$ is asserted and remains asserted for the duration of the transfer until the Tx FIFO is empty. Determining when the transfer stops does not depend on SPIxCON[15:14]; the transfer stops when there is no valid data left in the FIFO. Conversely, the transfer continues while there is valid data in the FIFO.

If the continuous transfer enable bit, SPIxCON[11], is cleared, each transfer consists of a single 8-bit serial transfer. If valid data exists in the Tx FIFO, a new transfer is initiated after a stall period where $\overline{CS}$ is deasserted.

Rx Initiated Transfer

Transfers initiated by a read of the Rx FIFO depend on the number of bytes to be received in the FIFO. If SPIxCON[15:14] = 11 and a read to the Rx FIFO occurs, the SPI initiates a 4-byte transfer. If continuous mode is set, the four bytes occur continuously with no deassertion of $\overline{CS}$ between bytes. If continuous mode is not set, the four bytes occur with stall periods between transfers where the $\overline{CS}$ is deasserted. A read of the Rx FIFO while the SPI is receiving data does not initiate another transfer after the present transfer is complete.

In slave mode, a transfer is initiated by the assertion of $\overline{CS}$.

Note that only $\overline{CS}$ is active in slave mode.

The device as a slave transmits and receives 8-bit data until the transfer is concluded by the deassertion of $\overline{CS}$.

The SPI transfer protocol diagrams (see Figure 26 and Figure 27) illustrate the data transfer protocol for the SPI and the effects of CPHA and CPOL bits in the control register on that protocol.

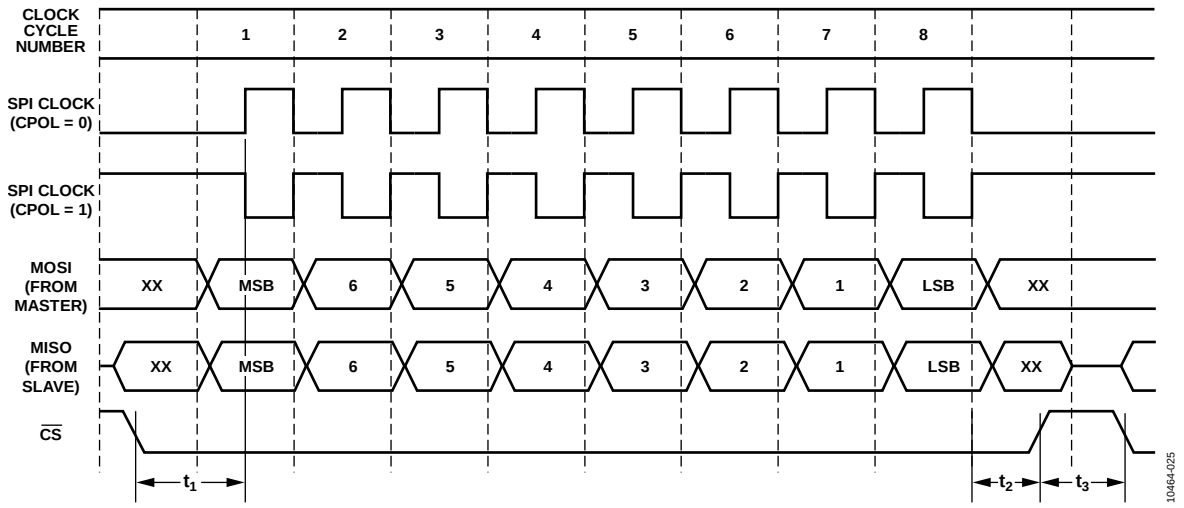

Figure 26. SPI Transfer Protocol CPHA = 0

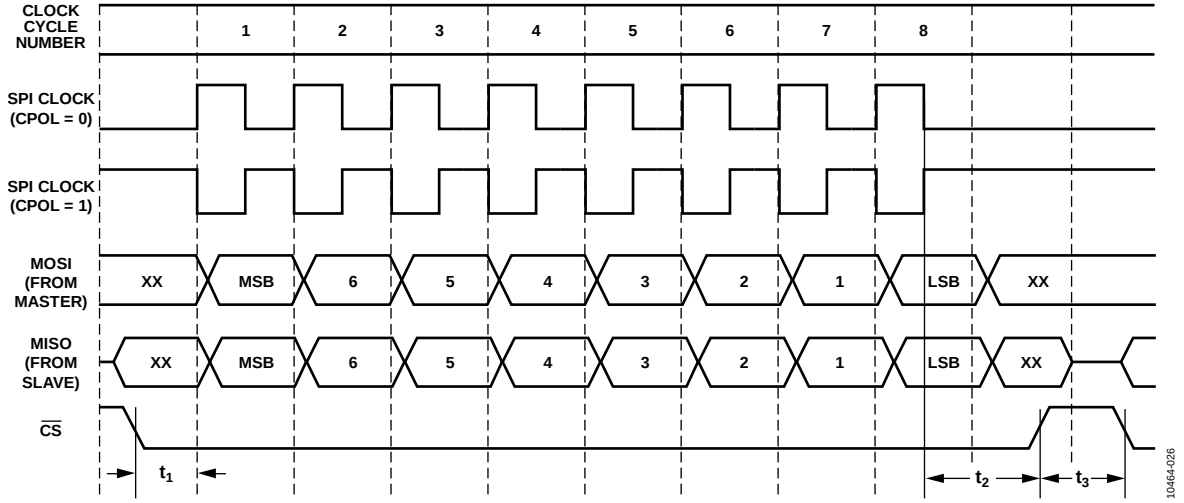

Figure 27. SPI Transfer Protocol CPHA = 1

SPI Data Underrun and Overflow

If the Tx underrun mode bit, ZEN (SPIxCON[7]), is cleared, the last byte from the previous transmission is shifted out when a transfer is initiated with no valid data in the FIFO. If ZEN is set, 0s are transmitted when a transfer is initiated with no valid data in the FIFO.

If the Rx overflow overwrite enable bit, RXOF(SPIxCON[8]), is set, the valid data in the Rx FIFO is overwritten by the new serial byte received if there is no space left in the FIFO. If RXOF is cleared, the new serial byte received is discarded if there is no space left in the FIFO.

When valid data is being overwritten in the Rx FIFO, the oldest byte is overwritten first, followed by the next oldest byte, and so on.

Full Duplex Operation

Simultaneous read/writes are supported on the SPI.

When implementing full duplex transfers in master mode, use the following procedure:

1. Initiate a transfer sequence via a transmit on the MOSI pin. Set SPIxCON[6] = 1. If interrupts are enabled, interrupts are triggered when a transmit interrupt occurs but not when a byte is received.
2. If you are using interrupts, the SPI Tx interrupt indicated by SPIxSTA[5] or the Tx FIFO underrun interrupt (SPIxSTA[4]) is asserted approximately 3 × SPICLK to 4 × SPICLK periods into the transfer of the first byte. Reload a byte into the Tx FIFO, if necessary, by writing to SPIxTX.
3. The first byte received via the MISO pin does not update the Rx FIFO status bits (SPIxSTA[10:8]) until 12 × SPICLK periods after $\overline{CS}$ has gone low. Therefore, two transmit interrupts may occur before the first receive byte is ready to be handled.
4. After the last transmit interrupt has occurred, it may be necessary to read two more bytes. It is recommended that SPIxSTA[10:8] are polled outside of the SPI interrupt handler after the last transmit interrupt is handled.

SPI INTERRUPTS

There is one interrupt line per SPI and four sources of interrupts. SPIxSTA[0] reflects the state of the interrupt line, and SPIxSTA[7:4] reflects the state of the four sources.

The SPI generates either TIRQ or RIRQ. Both interrupts cannot be enabled at the same time. The appropriate interrupt is enabled using the TIM bit, SPIxCON[6]. If TIM = 1, TIRQ is enabled. If TIM = 0, RIRQ is enabled.

Also note that the SPI0 and SPI1 interrupt source must be enabled in the NVIC Register ISER0 (ISER0[18] = SPI0, ISER0[19] = SPI1).

Tx Interrupt

If TIM is set, the Tx FIFO status causes the interrupt. SPIxCON[15:14] control when the interrupt occurs, as shown in Table 147.

Table 147. SPIxCON[15:14] IRQ Mode Bits

SPIxCON[15:14]	Interrupt Condition
00	An interrupt is generated after each byte that is transmitted. The interrupt occurs when the byte is read from the FIFO and written to the shift register.
01	An interrupt is generated after every two bytes that are transmitted.
10	An interrupt occurs after every third byte that is transmitted.
11	An interrupt occurs after every fourth byte that is transmitted.

The interrupts are generated depending on the number of bytes transmitted and not on the number of bytes in the FIFO. This is unlike the Rx interrupt, which depends on the number of bytes in the Rx FIFO and not the number of bytes received.

The transmit interrupt is cleared by a read to the status register. The status of this interrupt can be read by reading SPIxSTA[5]. The interrupt is disabled if SPIxCON[13] is left high.

A write to the control register, SPIxCON, resets the transmitted byte counter back to 0. For example, in a case where SPIxCON[15:14] is set to 11 and SPIxCON is written to after three bytes have been transmitted, the Tx interrupt does not occur until another four bytes have been transmitted.

Rx Interrupt

If SPIxCON[6] is cleared, the Rx FIFO status causes the interrupt. SPIxCON[15:14] control when the interrupt occurs. The interrupt is cleared by a read of SPIxSTA. The status of this interrupt can be read by reading SPIxSTA[6].

Interrupts are only generated when data is written to the FIFO. For example, if SPIxCON[15:14] are set to 0x00, an interrupt is generated after the first byte is received. When the status register is read, the interrupt is deactivated. If the byte is not read from the FIFO, the interrupt is not regenerated. Another interrupt is not generated until another byte is received into the FIFO.

The interrupt depends on the number of valid bytes in FIFO and not the number of bytes received. For example, when SPIxCON[15:14] are set to 0x01, an interrupt is generated after a byte is received if there are two or more bytes in the FIFO. The interrupt is not generated after every two bytes received.

The interrupt is disabled if SPIxCON[12] is left high.

Underrun/Overflow Interrupts

SPIxSTA[7] and SPIxSTA[4] generate SPI interrupts.

When a transfer starts with no data in the Tx FIFO, SPIxSTA[4] is set to indicate an underrun condition. This causes an interrupt. The interrupt (and status bit) are cleared on a read of the status register. This interrupt occurs irrespective of SPIxCON[15:14]. This interrupt is disabled if SPIxCON[13] is set.

When data is received and the Rx FIFO is already full, SPIxSTA[7] is set to 1, indicating an overflow condition. This causes an interrupt. The interrupt (and status bit) are cleared on a read of the status register. This interrupt occurs irrespective of SPIxCON[15:14]. This interrupt is disabled if SPIxCON[12] is set.

All interrupts are cleared either by a read of the status register or if SPIxCON[0] is deasserted. The Rx and Tx interrupts are also cleared if the relevant flush bits are asserted. Otherwise, the interrupts stay active even if the SPI is reconfigured.

WIRE-OR'ED MODE (WOM)

To prevent contention when the SPI is used in a multimaster or multislave system, the data output pins, MOSI and MISO, can be configured to behave as open-circuit drivers. An external pull-up resistor is required when this feature is selected. The WOM bit (SPIxCON[4]) controls the pad enable outputs for the data lines.

CSERR CONDITION

The CSERR bit (SPIxSTA[12]) indicates if an erroneous deassertion of the $\overline{CS}$ signal has been detected before the completion of all the eight SCLK cycles. This bit generates an interrupt and is available in all modes of operation: slave, master, and during DMA transfers.

If an interrupt occurs, generated by the CSERR bit (SPIxSTA[12]), then the SPI ENABLE bit (SPIxCON[0]) should be disabled and restarted to enable a clean recovery. This ensures that subsequent transfers are error free. The BCRST bit (SPIxDIV[7]) should be set at all times in both slave and master mode except when a midbyte stall in SPI communication is required. In this case, the CSERR flag is set but can be ignored.

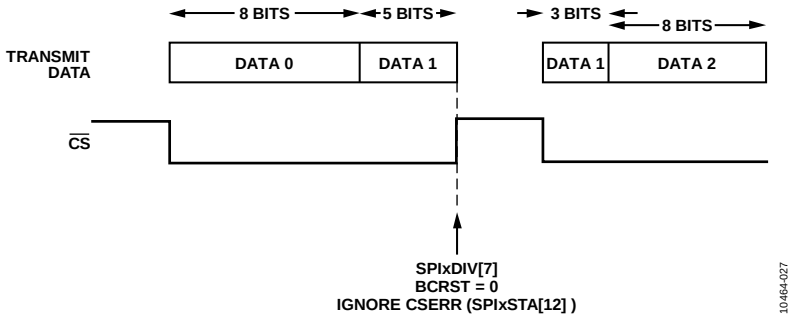

Figure 28. SPI Communication: Midbyte Stall

Note the SPI should only be reenabled when the $\overline{CS}$ signal is high.

SPI1 DMA

DMA operation is provided on the SPI1 channel only. Two DMA channels are dedicated to transmit and receive. The SPI1 DMA channels should be configured in the µDMA controller of the ARM Cortex-M3 processor.

It is possible to enable a DMA request on one or two channels at the same time by setting the DMA request bits for receive or transmit in the SPI1DMA register. If only the DMA transmit request (SPI1DMA[1]) is enabled, the Rx FIFO overflows during a SPI transfer, unless the received data is read by user code, and an overflow interrupt is generated. To avoid generating overflow interrupts, the Rx FIFO flush bit should be set, or the SPI interrupt should be disabled in the NVIC. If only the DMA receive request (SPI1DMA[2]) is enabled, the Tx FIFO is underrun. To avoid an underrun interrupt, the SPI interrupt should be disabled.

The SPI Tx (SPI1STA[5]) and SPI Rx (SPI1STA[6]) interrupts are not generated when using DMA. The SPI TXUR (SPI1STA[4]) and RXOF (SPI1STA[7]) interrupts are generated when using DMA. SPI1CON[15:14] are not used in transmit mode and should be set to 0x00 in receive mode.

The ENABLE bit (SPI1DMA[0]) controls the start of a DMA transfer. DMA requests are only generated when ENABLE = 1. At the end of a DMA transfer, that is, when receiving a DMA SPI transfer interrupt, this bit needs to be cleared to prevent extra DMA requests to the µDMA controller. The data still present in the Tx FIFO is transmitted if in Tx mode.

DMA Master Transmit Configuration

The DMA SPI Tx channel should be configured.

The NVIC should be configured to enable DMA Tx master interrupt (ISER0[23]).

The SPI block should be configured as follow:

```
pADI_SPI1->SPIDIV = SPI_serial_freq;                //Configures serial clock frequency
```

where $F_{serial\ clock} = f_{uclk}/(2*(1+SPIDIV))$

```
pADI_SPI1->SPIDCON = 0x1043;                        //Enable SPI in master mode and transmit
mode, Rx FIFO flush enabled.
pADI_SPI1->SPIDMA = 0x3;                            //Enable DMA mode, enable Tx DMA request
```

When all data present in the DMA buffer are transmitted, the DMA generates an interrupt. User code should disable the DMA request. Data is still in the Tx FIFO because the DMA request is generated each time there is free space in the Tx FIFO to always keep the FIFO full. User code can check how many bytes are still present in the FIFO in the FIFO status register.

Example Code: Initialize SPI1 DMA Transmit Sequence

```
void SPIDMAINIT(unsigned char ucMasterHSlaveL, unsigned int uiBaudrate)
{
pADI_GP0->GPCON = 0x55;                         // Configure P0.[3:0] as SPI pins
                                                // Slave only implemented
if ( ucMasterHSlaveL == 0)
{
pADI_SPI1->SPICON = 0xB85;
}
else
pADI_SPI1->SPICON = 0xBC7;
pADI_SPI1->SPIDMA = 0x7;                        // Enable DMA + SPI Rx/Tx DMA
pADI_DMA->DMACFG = 0x1;                         // Enable DMA mode in DMA controller

Dma_Init();
pADI_SPI1->SPICON |=0x3000;                     // Flush Rx/Tx FIFOs
pADI_SPI1->SPICON &=0xCFFF;                     // Flush Rx/Tx FIFOs
SPIDMAWRITE(uxSPITXData, 16);
pADI_DMA->DMAENSET = 0x103;
NVIC_EnableIRQ(DMA_SPI1_RX_IRQn);
NVIC_EnableIRQ(DMA_SPI1_TX_IRQn);

                                                // Enable DMA SPI1_RX/SPI1Tx Interrupt
}
void SPIDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                // Common configuration of all the descriptors
used here
    Desc.ctrlCfg.Bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    desc.ctrlcfg.bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                               //  TX Primary   Descriptor
Desc.srcEndPtr                         = (unsigned int)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                    = (unsigned int)&SPI1TX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_SPI1_TX,FALSE) = Desc;
                      pADI_SPI1->SPITX = *pucTX_DMA;
}
```

DMA Master Receive Configuration

The SPI1CNT register is available in DMA receive master mode only. It sets the number of receive bytes required by the SPI master, or the number of clocks that the master needs to generate. When the required number of bytes is received, no more transfers are initiated. To initiate a DMA master receive transfer, a dummy read should be completed by user code. This dummy read should be added to the SPI1CNT number.

The counter counting the bytes as they are received is reset either when SPI is disabled in SPI1CON[0] or if the SPI1CNT register is modified by user code.

Performing SPI1 DMA Master Receive

The DMA SPI Rx channel should be configured.

The NVIC should be configured to enable DMA Rx master interrupt (ISER0[24]).

The SPI block should be configured as follow:

```
pADI_SPI1->SPIDIV = SPI_serial_freq;                // Configures serial clock frequency where
F_serial clock = f_uclk / (2*(1+SPIDIV))
pADI_SPI1->SPICON = 0x2003;                         // Enable SPI in master mode and receive
mode, 1 byte transfer.
pADI_SPI1->SPIDMA = 0x5;                            // Enable DMA mode, enable Rx DMA request
pADI_SPI1->SPICNT = XXX;                            // Number of bytes to transferred + 1
A =;                                                // Dummy read
```

The DMA transfer stops when the number of bytes has been transferred. Note that the DMA buffer must be of the same size as SPI1CNT to generate a DMA interrupt when the transfer is complete.

Example Code: Initialize SPI1 DMA Receive Sequence

```
void SPIDMAINIT(unsigned char ucMasterHSlaveL, unsigned int uiBaudrate)
{
pADI_GP0->GPCON = 0x55;                                 // Configure P0.[3:0] as SPI pins
                                                        // Slave only implemented
if ( ucMasterHSlaveL == 0)
{
= 0xB85;
}
else
pADI_SPI1->SPICON = 0xBC7;
= 0x7;                                                  // Enable DMA + SPI Rx/Tx DMA
pADI_DMA->DMACFG = 0x1;                                 // Enable DMA mode in DMA controller

Dma_Init();
pADI_SPI1->SPICON |=0x3000;                             // Flush Rx/Tx FIFOs
&=0xCFFF;                                               // Flush Rx/Tx FIFOs
SPIDMAREAD (uxSPIRXData, 16);
pADI_DMA->DMAENSET = 0x103;
                                                        // Enable DMA SPI1_RX/SPI1Tx Interrupt
NVIC_EnableIRQ(DMA_SPI1_RX_IRQn);
NVIC_EnableIRQ(DMA_SPI1_TX_IRQn);

}
void SPIDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                // Common configuration of all the descriptors
used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                               //  RX Primary   Descriptor
    Desc.srcEndPtr                     = (unsigned int)&SPI1RX;
    Desc.destEndPtr                    = (unsigned int)(pucRX_DMA + iNumRX - 0x1);
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc          = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_SPI1_RX,FALSE) = Desc;
}
```

SPI AND POWER-DOWN MODES

In master mode, before entering power-down mode it is recommended to disable the SPI block in SPIxCON[0]. In slave mode, in either mode of operation, interrupt driven or DMA, the $\overline{CS}$ line level should be checked via the GPIO registers to ensure that the SPI is not communicating and that the SPI block is disabled while the $\overline{CS}$ line is high. At power-up, the SPI block can be reenabled.

SPI MEMORY MAPPED REGISTERS

The SPI0 interface consists of MMRs based at Address 0x40004000. The SPI1 interface consists of MMRs based at Address 0x40004400.

Table 148. SPI Peripheral Memory Address Table (SPI0 Base Address 0x40004000, SPI1 Base Address 0x40004400)

Offset	Name[1]	Description	Access	Default
0x0000	SPIxSTA	Status register	R	0x0000
0x0004	SPIxRX	8-bit receive register	R	0x0000
0x0008	SPIxTX	8-bit transmit register	W	0x0000
0x000C	SPIxDIV	16-bit bit rate selection register	RW	0x0000
0x0010	SPIxCON	16-bit configuration register	RW	0x0000
0x0014	SPI1DMA	DMA enable register (DMA provided on channel SPI1 only)	RW	0x0000
0x0018	SPIxCNT	8-bit received byte count register	R	0x0000

[1] Where x is 0 or 1 for SPI0 or SPI1.

SPI Status Registers

Address: 0x40004000, Reset: 0x0000, Name: SPI0STA

Address: 0x40004400, Reset: 0x0000, Name: SPI1STA

Table 149. SPIxSTA Register Bit Descriptions

Bits	Name	Description	
15:13	Reserved	Reserved bits.	
12	CSERR	Detected an abrupt $\overline{CS}$ deassertion. Set to 1 when the $\overline{CS}$ line is deasserted abruptly, even before the full byte of data was transmitted completely. This bit causes an interrupt. If the CSERR bit is set, it is recommended to clear the ENABLE bit in the SPIxCON register to ensure a clean recovery. Cleared to 0 when the SPISTA register is read.	
11	RXS	SPI Rx FIFO excess bytes present. Indicates when there are more bytes in the Rx FIFO than the Rx interrupt indicated. It depends on SPIxCON[15:14]. Cleared to 0 when the number of bytes in the FIFO is equal or less than the number in SPIxCON[15:14]. This bit is not cleared when the SPISTA register is read. This bit is not dependent on SPIxCON[6] and does not cause an interrupt.	
		SPIxCON[15:14]	**RXS**
		00	RXS is set if there are two or more bytes in the Rx FIFO.
		01	RXS is set if there are three or more bytes in the Rx FIFO.
		10	RXS is set if there are four or more bytes in the Rx FIFO.
		11	RXS is not set.
10:8	RXFSTA	SPI Rx FIFO status bits. 000: EMPTY: Set to 000 when Rx FIFO is empty. 001: ONEBYTE: Set to 001 when there is one valid byte in the FIFO. 010: TWOBYTES: Set to 010 when there are two valid bytes in the FIFO. 011: THREEBYTES: Set to 011 when there are three valid bytes in the FIFO. 100: FOURBYTES: Set to 100 when there are four valid bytes in the FIFO.	
7	RXOF	SPI Rx FIFO overflow status bit (interrupt). Set to 1 if the Rx FIFO is already full when new data is loaded to the FIFO. This bit generates an interrupt except when RFLUSH is set in SPIxCON. Cleared to 0 when the SPIxSTA register is read.	
6	RX	SPI Rx IRQ status bit. Not available in DMA mode. Set to 1 when a receive interrupt occurs. This bit is set when TIM in SPIxCON is cleared and the required number of bytes has been received. Cleared to 0 when the SPISTA register is read.	

Bits	Name	Description
5	TX	SPI Tx IRQ status bit. Not available in DMA mode. Set to 1 when a transmit interrupt occurs. This bit is set when TIM in SPIxCON is set and the required number of bytes has been transmitted. Cleared to 0 when the SPIxSTA register is read.
4	TXUR	SPI Tx FIFO underrun (interrupt). Set to 1 when a transmit is initiated without any valid data in the Tx FIFO. This bit generates an interrupt except when TFLUSH is set in SPIxCON. Cleared to 0 when the SPIxSTA register is read.
3:1	TXFSTA	Indicated how many valid bytes are in the SPI Tx FIFO. 000: EMPTY: Set to 000 when Tx FIFO is empty. 001: ONEBYTE: Set to 001 when there is one valid byte in the FIFO. 010: TWOBYTES: Set to 010 when there are two valid bytes in the FIFO. 011: THREEBYTES: Set to 011 when there are three valid bytes in the FIFO. 100: FOURBYTES: Set to 100 when there are four valid bytes in the FIFO.
0	IRQ	SPI interrupt status bit. Set to 1 when an SPI-based interrupt occurs. Cleared to 0 after reading SPIxSTA.

SPI Receive Registers

Address: 0x40004004, Reset: 0x0000, Name: SPI0RX

Address: 0x40004404, Reset: 0x0000, Name: SPI1RX

Table 150. SPIxRX Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	These bits are reserved and should be written 0 by user code.
7:0	VALUE	8-bit receive register. A read of the Rx FIFO returns the next byte to be read from the FIFO. A read of the FIFO when it is empty returns 0s.

SPI Transmit Registers

Address: 0x40004008, Reset: 0x0000, Name: SPI0TX

Address: 0x40004408, Reset: 0x0000, Name: SPI1TX

Table 151. SPIxTX Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	These bits are reserved and should be written 0 by user code.
7:0	VALUE	8-bit transmit register. A write to the Tx FIFO address space writes data to the next available location in the Tx FIFO. If the FIFO is full, the oldest byte of data in the FIFO is overwritten. A read from this address location return 0s.

SPI Clock Divider Registers

Address: 0x4000400C, Reset: 0x0000, Name: SPI0DIV

Address: 0x4000440C, Reset: 0x0000, Name: SPI1DIV

Table 152. SPIxDIV Register Bit Descriptions

Bits	Name	Description
15:7	Reserved	These bits are reserved and should be written 0 by user code.
7	BCRST	Reset mode for CSERR. This bit is used to configure the expected behavior of the SPI Interface logic after an abrupt deassertion of $\overline{CS}$. 0: SPI interface logic continues from where it stopped. The SPI can receive the remaining bits when $\overline{CS}$ is asserted, and user code must ignore the CSERR interrupt. 1: SPI interface logic is reset after a CSERR condition, and the user code is expected to clear the SPI ENABLE bit in SPIxCON.
6	Reserved	Reserved.

Bits	Name	Description
5:0	DIV	Factor used to divide UCLK to generate the serial clock: $f_{SERIAL\ CLOCK} = f_{UCLK/DIV}/(2 \times (1 + SPIDIV))$. The maximum frequency for the serial clock is ½ UCLK/DIV. These bits are only used for master mode. In slave mode, there is no need to set the serial clock frequency. The SPI receives the clock from the master. For SPI0, UCLK/DIV is set by CLKCON1[2:0] and CLKDIS[0]. For SPI1, UCLK/DIV is set by CLKCON1[5:3] and CLKDIS[1].

Note that when setting the SPI serial clock, the FCLK frequency must be taken into account. The FCLK frequency can be no less than half the SPI serial clock frequency.

For example, if the SPI clock divide register is set to 0x0000 (SCLK frequency = ½ UCLK/DIV frequency), the maximum that the CD bits in CLKCON0 can be set to is two (FCLK frequency = ¼ UCLK/DIV frequency).

SPI Control Registers

Address: 0x40004010, Reset: 0x0000, Name: SPI0CON

Address: 0x40004410, Reset: 0x0000, Name: SPI1CON

Table 153. SPIxCON Register Bit Descriptions

Bits	Name	Description	
15:14	MOD	SPI IRQ mode bits. If TIM is set, these bits are configured when the Tx/Rx interrupts occur in a transfer. If SPI1 DMA mode is used, these bits must be 00.	
		MOD	**Function**
		00	TX1RX1: Tx interrupt occurs when one byte has been transferred; Rx interrupt occurs when one or more bytes have been received into the FIFO.
		01	TX2RX2: Tx interrupt occurs when two bytes have been transferred; Rx interrupt occurs when two or more bytes have been received into the FIFO.
		10	TX3RX3: Tx interrupt occurs when three bytes have been transferred; Rx interrupt occurs when three or more bytes have been received into the FIFO.
		11	TX4RX4: Tx interrupt occurs when four bytes have been transferred; Rx interrupt occurs when the Rx FIFO is full, or when four bytes are present.
13	TFLUSH	SPI Tx FIFO flush enable bit. 0: Disable Tx FIFO flushing. 1: Flush the Tx FIFO. This bit does not clear itself and should be toggled if a single flush is required. If this bit is left high, then either the last transmitted value or 0x00 is transmitted, depending on the ZEN bit (SPIxCON[7]). Any writes to the Tx FIFO are ignored while this bit is set.	
12	RFLUSH	SPI Rx FIFO flush enable bit. If this bit is set, all incoming data is ignored and no interrupts are generated. If this bit is cleared and TIM = 0, a read of the Rx FIFO initiates a transfer. 0: Disable Rx FIFO flushing. 1: Flush the Rx FIFO. This bit does not clear itself and should be toggled if a single flush is required.	
11	CON	Continuous transfer enable. 0: Disable continuous transfer. Each transfer consists of a single 8-bit serial transfer. If valid data exists in the SPITX register, a new transfer is initiated after a stall period of one serial clock cycle. The $\overline{CS}$ line is deactivated for this one serial clock cycle. 1: Enable continuous transfer. In master mode, the transfer continues until no valid data is available in the Tx register. $\overline{CS}$ is asserted and remains asserted for the duration of each 8-bit serial transfer until Tx is empty.	
10	LOOPBACK	Loopback enable bit. 0: Normal mode. 1: Connect MISO to MOSI; therefore, data transmitted from the Tx register is looped back to the Rx register. The MASEN bit (SPIxCON [1]) must be set for loopback mode to work properly.	
9	OEN	Slave MISO output enable bit. 0: Disable the output driver on the MISO pin. The MISO pin is open-circuit when this bit is clear. 1: MISO to operate as normal.	
8	RXOF	SPI Rx overflow overwrite enable. 0: The new serial byte received is discarded. 1: The valid data in the Rx register is overwritten by the new serial byte received.	

Bits	Name	Description
7	ZEN	SPI transmit 0s when Tx FIFO is empty. 0: Transmit the last transmitted value when there is no valid data in the Tx FIFO. 1: Transmit 0x00 when there is no valid data in the Tx FIFO.
6	TIM	SPI transfer and interrupt mode. 0: RXRD: Initiate transfer with a read of the SPI Rx register. The read must be done while the SPI interface is idle mode to initiate a transfer when TIM = 0. Interrupt only occurs when Rx is full. 1: TXWR: Initiate transfer with a write to the SPI Tx register. Interrupt only occurs when Tx is empty.
5	LSB	LSB first transfer enable bit. 0: MSB is transmitted first. 1: LSB is transmitted first.
4	WOM	SPI wired OR mode enable bit. 0: Normal output levels. 1: Open-circuit data output enable. In master mode, When a 0 is being transmitted on the MOSI pin, the output driver is enabled. When a 1 is being transmitted on the MOSI pin, the output driver is disabled and an external pull-up resister is required to pull the pin high. The typical resistor value is 1 kΩ. In slave mode, When a 0 is being transmitted on the MISO pin, the output driver is enabled. When a 1 is being transmitted on the MISO pin, the output driver is disabled and an external pull-up resister is required to pull the pin high. The typical resistor value is 1 kΩ.
3	CPOL	Serial clock polarity mode bit. 0: Serial clock idles low. 1: Serial clock idles high.
2	CPHA	Serial clock phase mode bit. 0: Serial clock pulses at the middle of the first data bit transfer. 1: Serial clock pulses at the start of the first data bit.
1	MASEN	Master mode enable bit 0: Enable slave mode. 1: Enable master mode.
0	ENABLE	SPI enable bit. 0: Disable the SPI. Clearing this bit also resets all the FIFO related logic and the output bit shift counter to enable a clean start. 1: Enable the SPI.

Note that

- When changing the configuration, care must be taken not to change it during a data transfer to avoid corrupting the data. It is recommended to change configuration when the module is disabled (disable the SPI, ENABLE = 0) and then to reconfigure and reenable the SPI (ENABLE = 1).
- When reconfiguring from slave mode to master mode, or vice versa, both FIFOs must be empty.
- When using DMA, MOD (SPIxCON[15:14]) settings become irrelevant.
- Although the interrupt generation logic is independent of MOD (SPIxCON[15:14]), the master transfer initiation logic still depends on MOD. However, the DMA service happens only in a byte-by-byte basis and not in bursts. Therefore, only the value of MOD = 0 should be used in the case of DMA.

SPI1 DMA Enable Registers (SPI1 Only)

Address: 0x40004414, Reset: 0x0000, Name: SPI1DMA

Table 154. SPI1DMA Register Bit Descriptions

Bits	Name	Description
15:3	Reserved	These bits are reserved and should be written 0 by user code.
2	IENRXDMA	0: Cleared by default. 1: Enable receive DMA request.
1	IENTXDMA	0: Cleared by default. 1: Enable transmit DMA request.
0	ENABLE	0: Cleared by user code at the end of DMA transfer. 1: Start a DMA transfer This bit needs to be cleared to prevent extra DMA request to the µDMA controller.

Note that

- DMA requests are not generated when DMA ENABLE = 0 (SPI1DMA[0]).
- At the end of a DMA transfer, this bit must be reset to prevent extra DMA requests to the µDMA controller.
- When ENABLE (SPI1DMA[0]) = 1, Tx (SPI1STA[5]) and Rx (SPI1STA[6]) interrupts are automatically disabled. However, the TXUR (SPI1STA[4]) and RXOF (SPI1STA[7]) interrupts are still available to indicate a Tx FIFO underrun and a Rx FIFO overflow, respectively.
- When using SPI1 DMA mode, SPI1CON[15:14] must be cleared to 0x00.

SPI Received Byte Count Registers

Address: 0x40004018, Reset: 0x0000, Name: SPI0CNT

Address: 0x40004418, Reset: 0x0000, Name: SPI1CNT

Table 155. SPIxCNT Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	These bits are reserved and should be written 0 by user code.
7:0	VALUE	Number of Rx bytes required in master mode. Disabled in Tx mode. When the required number of bytes is received, no more transfers are initiated. The counter counting the bytes received is reset if the SPI is disabled or if SPIxCON[0] or SPI1CNT is updated.

UART SERIAL INTERFACE

UART FEATURES

- Industry standard, 16450 UART peripheral
- Support for DMA

UART OVERVIEW

The UART peripheral is a full-duplex universal asynchronous receiver/transmitter (UART), compatible with the industry standard, 16450. The UART is responsible for converting data between serial and parallel formats. The serial communication follows an asynchronous protocol, supporting various word length, stop bits, and parity generation options.

This UART also contains modem control and interrupt handling hardware. The UART features a fractional divider that facilitates high accuracy baud rate generation.

Interrupts can be generated from a number of unique events, such as full/empty data buffer, transfer error detection, and break detection.

UART OPERATION

Serial Communications

An asynchronous serial communication protocol is followed with these options:

- Five to eight data bits
- One, two, or 1½ stop bits
- None, even, or odd parity
- Baud rate = UCLK/DIV ÷ (2 × 16 × COMDIV) ÷ (M + N ÷ 2048), where COMDIV = 1 to 65536, M = 1 to 3, and N = 0 to 2047
- Where UCLK/DIV is the divided 16 MHz clock as configured via CLKCON1 and CLSYSDIV

All data-words require a start bit and at least one stop bit. This creates a range from seven bits to 12 bits for each word. Transmit operation is initiated by writing to the transmit holding register (COMTX). After a synchronization delay, the data is moved to the internal transmit shift register (TSR), where it is shifted out at a baud (bit) rate equal to UCLK ÷ (2 × 16 × COMDIV) ÷ (M + N ÷ 2048) with start, stop, and parity bits appended as required. All data-words begin with a low going start bit. The transfer of COMTX to the TSR causes the transmit register empty status flag to be set.

Receive operation uses the same data format as the transmit configuration except for the number of stop bits, which is always one. After detection of the start bit, the received word is shifted in the internal receive shift register (RSR). After the appropriate number of bits (including stop bits) is received, the data and any status is updated, and the RSR is transferred to the receive buffer register (COMRX). The receive buffer register full status flag is updated upon the transfer of the received word to this buffer and the appropriate synchronization delay.

A sampling clock equal to 16 times the baud rate is used to sample the data as close to the midpoint of the bit as possible. A receive filter is also present that removes spurious pulses of less than two times the sampling clock period.

Note that data is transmitted and received least significant bit first. This is often not the assumed case by the user. However, it is standard for the protocol.

For power saving purposes, it is possible to reduce the clock to the UART block via the CLKCON1 register (CLKCON1[11:9] and CLKDIS[3]). By default, the clock to the UART is disabled (CLKDIS[3] = 1).

PROGRAMMED I/O MODE

In this mode, the software is responsible for moving data to and from the UART. This is typically accomplished by interrupt service routines that respond to the transmit and receive interrupts by either reading or writing data as appropriate. This mode puts certain constraints on the software itself in that the software must respond within a certain time to prevent overflow errors from occurring in the receive channel.

Polling the status flag is processor intensive and not typically used unless the system can tolerate the overhead. Interrupts can be disabled using the COMIEN register.

Writing the COMTX when it is not empty or reading the COMRX when it is not full produces incorrect results and should not be done. In the former case, the COMTX is overwritten by the new word, and the previous word is never transmitted. In the latter case, the previously received word is read again. Both of these errors must be avoided in software by correctly using either interrupts or the status register polling. These errors are not detected in hardware.

ENABLE/DISABLE BIT

Before the ADuCM360/ADuCM361 enters power-down mode, it is recommended to disable the serial interfaces. A bit is provided in the UART control register to disable the UART serial peripheral. This bit disables the clock to the peripheral. When setting this bit, care must be taken in software that no data is being transmitted or received. If set during communication, the data transfer does not complete; the receive or transmit register contains only part of the data. It is also recommended to set CLKCON1[11:9] = 111 to minimize power consumption in power-down mode.

INTERRUPTS

The UART peripheral has one interrupt output to the interrupt controller for both Rx and Tx interrupts. The COMIIR register must be read by software to determine the cause of the interrupt. Note that in DMA mode the break and modem status interrupts are not available.

In I/O mode, when receiving, the interrupt is generated for the following cases:

- COMRX full
- Receive overflow error
- Receive parity error
- Receive framing error
- Break interrupt (UART RxD held low)
- Modem status interrupt (changes to CTS)
- COMTX empty

BUFFER REQUIREMENTS

This UART is double buffered (holding register and shift register).

DMA MODE

In this mode, user code does not move data to and from the UART. DMA request signals going to the external DMA block indicate that the UART is ready to transmit or receive data. These DMA request signals can be disabled in the COMIEN register.

In DMA mode, modem functionality is not available.

Example Code to Set Up UART Receive DMA Channel

```
void UARTRXDMAINIT(void)
{
pADI_UART->COMCON = 0;
pADI_UART->COMIEN = 0x20;
                                                    // Enable DMA Tx transfers
pADI_UART->COMLCR = COMLCR_WLS_8BITS + COMLCR_STOP;  // 8-data bits + 1 Stop bit.
pADI_UART->COMDIV = 0x11;
                                                    // Set UART Baud rate
pADI_UART->COMDIV = 0x11;
                                                    // COMDIV = 17
pADI_UART->COMFBR = COMFBR_ENABLE_EN + 0x1883;      // DIVM = 3, DIVN = 131.
                      |= 0x9000;
                                                    // Configure P0.7/P0.6 for UART
                     Dma_Init();

                     UARTDMAREAD(uxUARTRXData, 4);
                     pADI_DMA->DMACFG = 0x1;
                                                    // Enable DMA mode in DMA controller
                     pADI_DMA->DMAENSET = 0x8;
                                                    // Enable UART_RX_DMA Channel
                                                    // UART Rx DMA interrupt sources -
M360
}
void UARTDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
```

```
{
   DmaDesc Desc;
                                                        // Common configuration of all the
descriptors used here
    Desc.ctrlCfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;
                                                        //  RX Primary   Descriptor
    Desc.srcEndPtr                     = (unsigned int)&COMRX;
    Desc.destEndPtr                    = (unsigned int)(pucRX_DMA + iNumRX - 0x1);
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc          = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_UART_RX,FALSE) = Desc;
}
                                                        // UART DMA Rx IRQ handler
void DMA_UART_RX_Int_Handler()
{
NVIC_DisableIRQ(DMA_UART_RX_IRQn);                      // Clear Interrupt source
}
```

Example Code to Set Up UART Transmit DMA Channel

```
void UARTTXDMAINIT(void)
{
  pADI_UART->COMCON = 0;
  pADI_UART->COMIEN = 0x10;                             // Enable DMA Tx transfers
  pADI_UART->COMLCR = COMLCR_WLS_8BITS + COMLCR_STOP;   // 8-data bits + 1 Stop bit.
  pADI_UART->COMDIV = 0x11;
  // Set UART Baud rate
  pADI_UART->COMFBR = COMFBR_ENABLE_EN + 0x1883;        // DIVM = 3, DIVN = 131.
  pADI_GP0->GPCON |= 0x3C;                              // Configure P0.1/P0.2 for UART
Dma_Init();
UARTDMAWRITE(uxUARTTXData, 16);
pADI_DMA->DMACFG = 0x1;                                 // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x4;                               // Enable UART_TX_DMA Channel
NVIC_EnableIRQ(DMA_UART_TX_IRQn);                       // UART Tx DMA interrupt sources -
M360
}



void UARTDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
```

```
    // Common configuration of all the descriptors used here
    Desc.ctrlCfg.Bits.cycle_ctrl       = CYCLECTRL_BASIC;   //CYCLECTRL_BASIC;
    Desc.ctrlCfg.Bits.next_useburst    = 0x1;
    Desc.ctrlCfg.Bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                        //  TX Primary   Descriptor
                        Desc.srcEndPtr                     = (unsigned int)(pucTX_DMA + iNumRX -
0x1);
    Desc.destEndPtr                    =  (unsigned int)&COMTX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_UART_TX,FALSE) = Desc;

}

                                                        // UART DMA Tx IRQ handler
void DMA_UART_TX_Int_Handler()
{
                        NVIC_DisableIRQ(DMA_UART_TX_IRQn);
                                                        // Clear Interrupt source
}
```

UART MEMORY MAPPED REGISTERS

The UART interface consists of MMRs based at Address 0x40005000.

Table 156. UART Interface Memory Address Table (Base Address: 0x40005000)

Offset	Name	Description	Access	Default
0x0000	COMTX	Transmit holding register	W	0x0000
0x0000	COMRX	Receive buffer register	R	0x0000
0x0004	COMIEN	Interrupt enable register	RW	0x0000
0x0008	COMIIR	Interrupt identification register	R	0x0000
0x000C	COMLCR	Line control register	RW	0x0000
0x0010	COMMCR	Modem control register	RW	0x0000
0x0014	COMLSR	Line status register	R	0x0000
0x0018	COMMSR	Modem status register	R	0x0000
0x0024	COMFBR	Fractional baud rate register	RW	0x0000
0x0028	COMDIV	Baud rate divider register	RW	0x0000
0x0030	COMCON	UART control register	RW	0x0000

UART Transmit and Receive Registers

Address: 0x40005000, Reset: 0x0000, Name: COMRX/COMTX

COMRX and COMTX share the same address while they are implemented as different registers. If these registers are written to, the user accesses the transmit holding register (COMTX). If these registers are read from, the user accesses the receive buffer register (COMRX).

COMRX

This is an 8-bit register from which the user can read received data. If the ERBFI bit is set in the COMIEN register, an interrupt is generated when this register is fully loaded with the received data via serial input port.

Note that when the user sets the ERBFI bit while COMRX is already full, an interrupt is generated immediately.

COMTX

This is an 8-bit register to which the user can write the data to be sent. If the ETBEI bit is set in the COMIEN register, an interrupt is generated when COMTX is empty.

Note that if the user sets ETBEI while COMTX is already empty, an interrupt is generated immediately.

Table 157. COMRX/COMTX Register Bit Descriptions

Bits	Name	Description
7:0	VALUE	Receive buffer register/transmit holding register.

UART Interrupt Enable Register

Address: 0x40005004, Reset: 0x0000, Name: COMIEN

COMIEN is the interrupt enable register that is used to configure which interrupt source generates the interrupt. Only the lowest four bits in this register enable interrupts. Bit 4 and Bit 5 enable UART DMA signals. The UART DMA channel and interrupt must be configured in the DMA block.

Table 158. COMIEN Register Bit Descriptions

Bits	Name	Description
7:6	Reserved	Reserved.
5	EDMAR	DMA requests in transmit mode. 0: Disable. 1: Enable.
4	EDMAT	DMA requests in receive mode. 0: Disable. 1: Enable.
3	EDSSI	Modem status interrupt. (This interrupt is generated when any bit of COMMSR[3:0] is set.) 0: Disable. 1: Enable.
2	ELSI	Rx status interrupt. (This interrupt is generated when any bit of COMLSR[4:1] is set.) 0: Disable. 1: Enable.
1	ETBEI	Transmit buffer empty interrupt. 0: Disable. 1: Enable.
0	ERBFI	Receive buffer full interrupt. 0: Disable. 1: Enable.

UART Interrupt Identification Register

Address: 0x40005008, Reset: 0x0000, Name: COMIIR

Table 159. COMIIR Register Bit Descriptions

Bits	Name	Description
7:3	Reserved	Reserved.
2:1	STA	Status bits. Status bits are used to encode the interrupt status when NIRQ is low. See Table 160 for more details.
0	NINT	Interrupt flag. 0: Indicates any of the following: receive buffer full, transmit buffer empty, line status, or modem status interrupt occurred. 1: There is no interrupt (default).

Table 160. Interrupt Identification Table

Bits[2:1], STA	Bit 0, NIRQ	Priority	Definition	Clearing Operation
00	1	N/A	No interrupt	N/A
11	0	1	Receive line status interrupt	Read COMLSR register
10	0	2	Receive buffer full interrupt	Read COMRX register
01	0	3	Transmit buffer empty interrupt	Write data to COMTX or read COMIIR
00	0	4	Modem status interrupt	Read COMMSR register

UART Line Control Register

Address: 0x4000500C, Reset: 0x0000, Name: COMLCR

Table 161. COMLCR Register Bit Descriptions

Bits	Name	Description		
7	Reserved	Reserved.		
6	BRK	Set Break. 1: Force TxD to 0. 0: Operate in normal mode.		
5	SP	Stick parity. 0: Parity is not forced based on EPS and PEN values. 1: To force parity to defined values based on EPS and PEN values as follows:		
		EPS	**PEN**	**Function When SP = 1**
		1	1	Parity forced to 1.
		0	1	Parity forced to 0.
		X	0	No parity transmitted.
4	EPS	Even parity select bit. 0: Odd parity. 1: Even parity.		
3	PEN	Parity enable bit. 0: No parity transmission or checking. 1: transmit and check the parity bit.		
2	STOP	Stop bit. 0: Generate one stop bit in the transmitted data. 1: Transmit 1.5 stop bits if the word length is 5 bits, or 2 stop bits if the word length is 6, 7, or 8 bits. The receiver checks the first stop bit only, regardless of the number of stop bits selected.		
1:0	WLS	Word length select bits. 00: 5 bits. 01: 6 bits. 10: 7 bits. 11: 8 bits.		

UART Modem Control Register

Address: 0x40005010, Reset: 0x0000, Name: COMMCR

Table 162. COMMCR Register Bit Descriptions

Bits	Name	Description
7:5	Reserved	Reserved.
4	LOOPBACK	Loop back. 0: Normal mode. 1: Enable loopback mode. In loopback mode, the UART TXD is forced high. The modem signals are also directly connected to the status inputs (RTS to CTS).
3	OUT1	Reserved.
2	OUT2	Reserved.
1	RTS	Request to send. 0: Force the RTS output to 1. 1: Force the RTS output to 0.
0	DTR	Data terminal ready. 0: Force the DTR output to 1. 1: Force the DTR output to 0.

UART Line Status Register

Address: 0x40005014, Reset: 0x0000, Name: COMLSR

Table 163. COMLSR Register Bit Descriptions

Bits	Name	Description
7	Reserved	Reserved.
6	TEMT	COMTX and shift register empty status bit. 1: If COMTX and the shift register are empty, this bit indicates that the data has been transmitted; that is, it is no longer present in the shift register (default). Cleared to 0 when writing to COMTX.
5	THRE	COMTX empty status bit. 1: If COMTX is empty, COMTX can be written as soon as this bit is set; the previous data may not have been transmitted yet and can still be present in the shift register (default). Cleared to 0 when writing to COMTX.
4	BI	Break indicator. 1: When UART RXD is held low for more than the maximum word length. Cleared to 0 automatically.
3	FE	Framing error. 1: When the stop bit is invalid. Cleared to 0 automatically.
2	PE	Parity error. 1: When a parity error occurs. Cleared to 0 automatically.
1	OE	Overrun error. 1: Automatically if data is overwritten before being read. Cleared to 0 automatically.
0	DR	Data ready. 1: Automatically when COMRX is full. Cleared to 0 by reading COMRX.

UART Modem Status Register

Address: 0x40005018, Reset: 0x0000, Name: COMMSR

Table 164. COMMSR Register Bit Descriptions

Bits	Name	Description
7	DCD	Data carrier detect. Set to 1 if DCD is currently logic low. Cleared to 0 if DCD is currently logic high.
6	RI	Ring indicator. Set to 1 if RI is currently logic low. Cleared to 0 if RI is currently logic high.
5	DSR	Data set ready. Set to 1 if DSR is currently logic low. Cleared to 0 if DSR is currently logic high.
4	CTS	Clear to send. Set to 1 if CTS is currently logic low. Cleared to 0 if CTS is currently logic high.
3	DDCD	Delta DCD. Set to 1 if DCD changed state since COMMSR was last read. Cleared to 0 if DCD did not change state since COMMSR was last read. Cleared by reading COMMSR.
2	TERI	Trailing edge RI. Set to 1 if NRI changed from 0 to 1 since COMMSR was last read. Cleared by reading COMMSR.
1	DDSR	Delta DSR. Set to 1 if DSR changed state since COMMSR was last read. Cleared by reading COMMSR.
0	DCTS	Delta CTS. Set to 1 if CTS changed state since COMMSR was last read. Cleared by reading COMMSR.

UART Fractional Baud Rate Divider Register

Address: 0x40005024, Reset: 0x0000, Name: COMFBR

Table 165. COMFBR Register Bit Descriptions

Bits	Name	Description
15	ENABLE	Fractional baud rate generator enable bit for more accurate baud rate generation. The generating of a fractional baud rate can be described by the following formula, where *UCLK/DIV* is the divided 16 MHz clock as configured via CLKCON1 and CLKSYSDIV. CLKDIS[3] should be set to 0 to enable the clock. The final baud rate of UART operation is shown in Figure 29. $$Baud\ Rate = \frac{UCLK / DIV}{2 \times (M + N/2048)}{16 \times COMDIV}$$ 1: Enabled. 0: Disabled.
14:13	Reserved	Reserved.
12:11	DIVM	Fractional baud rate M divide bits (1 to 3). This bit should not be 0.
10:0	DIVN	Fractional baud rate N divide bits (0 to 2047).

Table 166. Baud Rate Examples—UCLK/DIV = 16 MHz for All Examples

Baud Rates	COMDIV	DIVM	DIVN	Actual	% Error
9600	17	3	131	9599.25	−0.0078%
19,200	8	3	523	19,199.04	−0.0050%
38,400	4	3	523	38,398.08	−0.0050%
57,600	8	1	174	57,605.76	+0.0100%
115,200	2	2	348	115,211.5	+0.0100%
230,400	2	1	174	230,423	+0.0100%
460,800	1	1	174	460,846.1	+0.0100%

UART Divider Register

Address: 0x40005028, Reset: 0x0000, Name: COMDIV

The baud rate divider register is a 16-bit register used to generate the baud rate for UART data transfer. The baud rate without fractional divider is a divided down version of the master clock, as shown in Figure 29.

$$Baud\ Rate = UCLK/DIV \div (2 \times 16 \times COMDIV) \div (M + N \div 2048)$$

where *UCLK/DIV* is the divided 16 MHz clock as configured via CLKCON1[11:9] and CLKDIS[3].

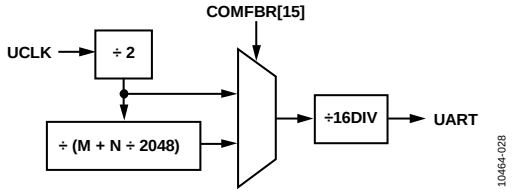

Figure 29. Baud Rate Generation

Table 167. COMDIV Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Set the baud rate. The COMDIV register should not be 0.

UART Control Register

Address: 0x40005030, Reset: 0x0000, Name: COMCON

Table 168. COMCON Register Bit Descriptions

Bits	Name	Description
8:1	Reserved	Reserved.
0	DISABLE	UART disable bit. It is recommended to disable the UART before entering power-down mode. 0: Enable the UART. 1: Disable the UART. This resets the state machine and internal counters, but the contents of the MMRs remain unchanged.

GENERAL-PURPOSE TIMERS

GENERAL-PURPOSE TIMERS FEATURES

- Two identical, general-purpose, 16-bit count-up/count-down timers
  - Timer0 and Timer1
- Clocked from four different clock sources (can be scaled down using a prescalar of 1, 16, 256, or 32,768)
  - 16 MHz system clock or 8 MHz system clock if CLKSYSDIV = 1 (UCLK)
  - Peripheral clock (PCLK)
  - 32 kHz internal oscillator (LFOSC)
  - 32 kHz external crystal (LFXTAL)
- Two modes
  - Free running
  - Periodic
- Capture events feature
  - Capability to capture 15 different events on each timer

GENERAL-PURPOSE TIMERS BLOCK DIAGRAM

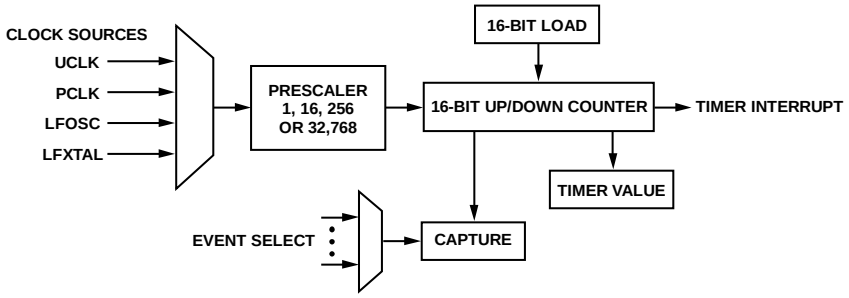

Figure 30. General-Purpose Timers Block Diagram

GENERAL-PURPOSE TIMERS OVERVIEW

Timer0 and Timer1 are two identical general-purpose, 16-bit count-up/count-down timers. They can be clocked from four different clock sources: UCLK, PCLK, 32 kHz internal oscillator (LFOSC), and 32 kHz external crystal (LFXTAL). This clock source can be scaled down using a prescalar of 1, 16, 256, or 32,768.

The timers can be either free running or periodic.

- In free running mode, the counter decrements from full scale to zero scale or increments from zero scale to full scale and then restarts.
- In periodic mode, the counter decrements or increments from the value in the load register (TxLD MMR) until zero scale or full scale is reached and then restarts at the value stored in the load register.

The value of a counter can be read at any time by accessing its value register (TxVAL).

This register is synchronous to UCLK. Therefore, when a timer is clocked from a clock other than UCLK, TxVAL reflects the latest timer value.

The TxCON register selects the timer mode, configures the clock source, selects count-up/count-down, starts the counter, and controls the event capture function.

An interrupt signal is generated each time the value of the counter reaches 0 when counting down, or each time the counter value reaches the maximum value when counting up. An IRQ can be cleared by writing 1 to the time clear interrupt register of that particular timer (TxCLRI).

In addition, Timer0 and Timer1 have a capture register (TxCAP) that is triggered by a selected IRQ source initial assertion. When triggered, the current timer value is copied to TxCAP, and the timer continues to run. This feature can be used to determine the assertion of an event with increased accuracy. Each timer can capture 16 different events, listed in Table 169.

GENERAL-PURPOSE TIMERS OPERATION

Free Running Mode

In free running mode, the timer is started by setting the enable bit (TxCON[4]) to 1 and the MOD bit (TxCON[3]) to 0. The timer increments from zero scale/full scale to full scale/zero scale if counting up/down. Full scale is 2^{16} – 1 or 0xFFFF in binary format. On

reaching full scale (or zero scale), a timeout interrupt occurs and TxSTA[0] is set. To clear it, user code must write 1 to TxCLRI[0]. If TxCON[7] is set, the timer keeps counting and reloads when the TxCLRI register is written.

Periodic Mode

In periodic mode, the initial TxLD value should be loaded before starting the timer by setting the enable bit (TxCON[4]) to 1. The timer value either increments from the value in TxLD to full scale or decrements from the value in TxLD to zero scale, depending on the TxCON[2] settings (count up/down). Upon reaching full scale or zero scale, the timer generates an interrupt. The TxLD is reloaded into TxVAL, and the timer continues counting up or down. The timer should be disabled prior to changing the TxCON or TxLD register. If the TxLD register is changed while the timer is being loaded, undefined results can occur. By default, the counter is reloaded automatically when generating the interrupt signal. If TxCON[7] is set to 1, the counter is also reloaded when user code writes TxCLRI. This allows user changes to the TxLD to take effect immediately and not on the next timeout.

The timer interval is calculated as follows:

If the timer is set to count down, then

$$Interval = (TxLD \times Prescaler)/Source\ Clock$$

For example, if TxLD = 0x100, prescaler = 4, and clock source = UCLK, the interval is 64 μs (where UCLK = 16 MHz).

If the timer is set to count up, then

$$Interval = ((Full\ Scale - TxLD) \times Prescaler)/Source\ Clock$$

Asynchronous Clock Source

Timers are started by setting the enable bit (TxCON[4]) to 1 in the control register of the corresponding timer.

However, when the timer clock source is LFXTAL or LFOSC, some precautions must be taken:

- The control register (TxCON) should not be written if TxSTA[6] is set. Therefore, TxSTA should be read prior to configuring the control register (TxCON). When TxSTA[6] is cleared, the register can be modified. This ensures that synchronizing the timer control between the processor and timer clock domains is complete.
- After clearing the interrupt in TxCLRI, it must be ensured that the register write has fully completed before returning from the interrupt handler. Use the data synchronization barrier (DSB) instruction if necessary and check that TxSTA[7] = 0.

```
__asm void asmDSB()
{
nop
DSB
BX LR
}
```

- The value of a counter can be read at any time by accessing its value register (TxVAL). In an asynchronous configuration, TxVAL should always be read twice. If the two readings are different, it should be read a third time to get the correct value.

At any time, TxVAL contains a valid value to be read, synchronized to PCLK.

TxSTA should be read prior to writing to any timer registers following the set or clear of the enable bit. When TxSTA[7] is cleared, registers can be modified. This ensures that the timer has completed synchronization between the processor and timer clock domains. The typical synchronization time is two timer clock periods.

At any time, TxVAL contains a valid value to be read, synchronized to the PCLK clock. The TxCON register enables the counter, selects the mode, selects the prescale value, and controls the event capture function.

Note that CLKDIS[6] must be cleared to 0 to enable the system clock to Timer1. This bit must be cleared to 0 to enable Timer 1. Similarly, CLKDIS[5] must be cleared to 0 to enable the system clock to Timer0. This bit must be cleared to 0 to enable Timer0.

Capture Event Function

There are 30 interrupt events that can be captured by the general-purpose timers. These events are divided into two groups of 16 inputs for each of the timers, as shown in Table 169. Any one of the 16 events associated with a general-purpose timer can cause a capture of the 16-bit TxVAL register into the 16-bit TxCAP register. TxCON has a 4-bit field selecting which of the 16 events to capture.

When the selected interrupt event occurs, the TxVAL register is copied into the TxCAP register. TxSTA[1] is set, indicating that a capture event is pending. The bit is cleared by writing 1 to TxCLRI[1]. The TxCAP register also holds its value and cannot be overwritten until a 1 is written to TxCLRI[1].

Table 169. Timer Capture Event

Event Select Bits (TxCON[11:8])	Timer0 Capture Source	Timer1 Capture Source
0	Wake-up timer	UART
1	External Interrupt 0	Timer0
2	External Interrupt 1	SPI0
3	External Interrupt 2	SPI1
4	External Interrupt 3	I^2C slave
5	External Interrupt 4	I^2C master
6	External Interrupt 5	DMA error
7	External Interrupt 6	DMA done, any of the DMA channels
8	External Interrupt 7	External Interrupt 1
9	Watchdog timer	External Interrupt 2
10	Timer1	External Interrupt 3
11	ADC0 (ADuCM360)/reserved (ADuCM361)	PWMTRIP
12	ADC1	PWM0
13	Step detection	PWM1
14	DMA done—any of the DMA channels	PWM2
15	Flash controller	Reserved.

GENERAL-PURPOSE TIMERS MEMORY MAPPED REGISTERS

General-Purpose Timer0 Register Map

Table 170. General-Purpose Timer0 Memory Mapped Registers Address Table (Timer0 Base Address: 0x40000000)

Offset	Name	Description	Access	Default
0x0000	T0LD	16-bit load value	RW	0x0000
0x0004	T0VAL	16-bit timer value, read only	R	0x0000
0x0008	T0CON	Control register	RW	0x000A
0x000C	T0CLRI	Clear interrupt register	RW	0x0000
0x0010	T0CAP	Capture register	R	0x0000
0x001C	T0STA	Status register	R	0x0000

General-Purpose Timer1 Register Map

Table 171. General-Purpose Timer1 Memory Mapped Registers Address Table (Timer1 Base Address: 0x40000400)

Offset	Name	Description	Access	Default
0x0000	T1LD	16-bit load value	RW	0x0000
0x0004	T1VAL	16-bit timer value, read only	R	0x0000
0x0008	T1CON	Control register	RW	0x000A
0x000C	T1CLRI	Clear interrupt register	RW	0x0000
0x0010	T1CAP	Capture register	R	0x0000
0x001C	T1STA	Status register	R	0x0000

General-Purpose Timers Load Registers

Address: 0x40000000, Reset: 0x0000, Name: T0LD

Address: 0x40000400, Reset: 0x0000, Name: T1LD

Table 172. T0LD and T1LD Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Load value, 0 by default.

General-Purpose Timers Value Registers

Address: 0x40000004, Reset: 0x0000, Name: T0VAL

Address: 0x40000404, Reset: 0x0000, Name: T1VAL

Table 173. T0VAL and T1VAL Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Current counter value, read only.

General-Purpose Timers Control Registers

Address: 0x40000008, Reset: 0x000A, Name: T0CON

Address: 0x40000408, Reset: 0x000A, Name: T1CON

The TxCON registers should not be written if the corresponding TxSTA[6] or TxSTA[7] is set. Note that the timer clock must be four times slower than FCLK (the system clock); therefore, care must be taken when selecting the prescaler at different CD values.

Table 174. T0CON and T1CON Register Bit Descriptions

Bits	Name	Description
15:13	Reserved	Reserved. These bits should be written 0.
12	EVENTEN	Event select bit. 0: Disable time capture of an event. 1: Enable time capture of an event.
11:8	EVENT	Event select range (0 to 15). The events are described in Table 169.
7	RLD	Reload control bit for periodic mode. 0: Reload only on a timeout. 1: Change the load value on a write to TxCLRI.
6:5	CLK	Clock select. 00: UCLK. 01: PCLK. 10: LFOSC (32 kHz internal oscillator). 11: LFXTAL (external crystal, 32 kHz).
4	ENABLE	Timer enable bit. The timer starts counting from its initial value (0 if count-up mode or 0xFFFF if count-down mode). Clearing this bit resets the timer, including the TxVAL register. 0: Disable the timer. 1: Enable the timer.
3	MOD	Timer mode. 0: Operate in free running mode. 1: Operate in periodic mode.
2	UP	Count up. 0: Timer to count down. 1: Timer to count up.
1:0	PRE	Prescaler. If the selected clock source is UCLK, the selected prescaler is further divided by a factor of 4. 00: DIV1: source clock/1. 01: DIV16: source clock/16. 10: DIV256: source clock/256. 11: DIV32768: source clock/32,768.

General-Purpose Timers Clear Interrupt Registers

Address: 0x4000000C, Reset: 0x0000, Name: T0CLRI

Address: 0x4000040C, Reset: 0x0000, Name: T1CLRI

Table 175. T0CLRI and T1CLRI Register Bit Descriptions

Bits	Name	Description
15:2	Reserved	Reserved. These bits should be written 0.
1	CAP	Clear captured event interrupt. 1: Clear a capture event interrupt. This bit always reads 0.
0	TMOUT	Clear timeout interrupt. 1: Clear a timeout interrupt. This bit always reads 0.

Ensure that the register write has fully completed before returning from the interrupt handler. Use the data synchronization barrier (DSB) instruction if necessary. The following is a code example showing how to implement the DSB ARM Cortex-M3 instruction in a C program.

```
__asm void asmDSB()
{
       nop
       DSB
BX LR
}
void GP_Tmr0_Int_Handler()
{
  T0CLRI = 0x1;
  asmDSB();
}
```

General-Purpose Timers Capture Registers

Address: 0x40000010, Reset: 0x0000, Name: T0CAP

Address: 0x40000410, Reset: 0x0000, Name: T1CAP

Table 176. T0CAP and T1CAP Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	16-bit captured value. Read only. TxCAP holds its value until TxCLRI[1] is set by user code. If the same event occurs again, TxCAP is not overwritten.

General-Purpose Timers Status Registers

Address: 0x4000001C, Reset: 0x0000, Name: T0STA

Address: 0x4000041C, Reset: 0x0000, Name: T1STA

Table 177. T0STA and T1STA Register Bit Descriptions

Bits	Name	Description
15:8	Reserved	Reserved. These bits should be written 0.
7	PDOK	T0CLRI/T1CLRI synchronization. Set to 1 automatically when the T0CLRI/T1CLRI value is being updated in the timer clock domain, indicating that the timer's configuration is not yet valid. Cleared to 0 when the interrupt is cleared in the timer's clock domain.
6	BUSY	Timer busy. Set to 1 to indicate that the timer is not ready to receive commands to TxCON. Previous change of the TxCON value has not been synchronized in the timer clock domain. Cleared to 0 to indicate that the timer is ready to receive commands to TxCON. The previous change of TxCON has been synchronized in the timer clock domain.
5:2	Reserved	Reserved. These bits should be written 0.
1	CAP	Capture event pending. Set to 1 to indicate a capture event is pending. Cleared to 0 by writing 1 to TxCLRI[1].
0	TMOUT	Timeout event occurred. Set to 1 to indicate a timeout event has occurred. For count-up mode, this is when the counter reaches full scale. For count-down mode, this is when the counter reaches 0. Cleared to 0 by writing 1 to TxCLRI[0].

WAKE-UP TIMER

WAKE-UP TIMER FEATURES

- 32-bit counter (count down or count up)
- Four clock sources with programmable prescaler (1, 16, 256, or 32,768)
  - Peripheral clock (PCLK)
  - 32 kHz external crystal (LFXTAL)
  - 32 kHz internal oscillator (LFOSC)
  - External clock applied on P1.0 (EXTCLK)
- Four compare points, one automatic increment

WAKE-UP TIMER BLOCK DIAGRAM

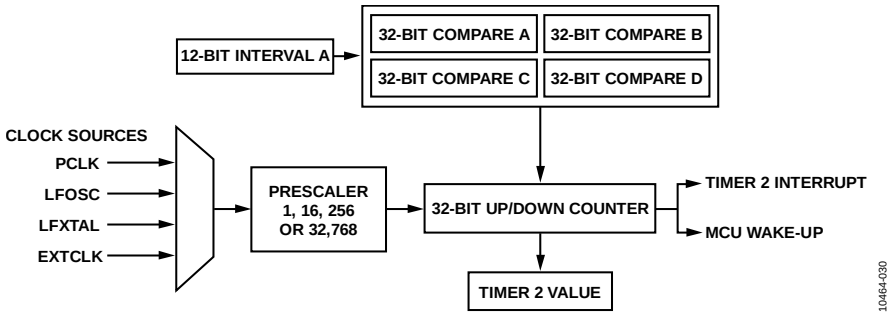

Figure 31. Wake-Up Timer Block Diagram

WAKE-UP TIMER OVERVIEW

The wake-up timer (Timer2) block consists of a 32-bit counter clocked from one of four different sources: system clock (PCLK), external crystal (LFXTAL), internal oscillator (LFOSC), or an external clock applied on P1.0 (EXTCLK).The selected clock source can be scaled down using a prescaler of 1, 16, 256, or 32,768. The wake-up timer continues to run independent of the clock source used when the PCLK clock is disabled.

The timer can be used in free running or periodic mode. In free running mode, the timer counts from 0x00000000 to 0xFFFFFFFF and then restarts at 0x00000000. In periodic mode, the timer counts from 0x00000000 to T2WUFD (T2WUFD0 and T2WUFD1).

In addition, the wake-up timer has four specific time fields to compare with the wake-up counter: T2WUFA, T2WUFB, T2WUFC, and T2WUFD. All four wake-up compare points can generate interrupts or wake-up signals. When in free running mode, T2WUFA, T2WUFB, T2WUFC, andT2WUFD must be reconfigured in software to generate a periodic interrupt.

WAKE-UP TIMER OPERATION

The wake-up timer comparator registers must be configured before starting the timer. The timer is started by writing the control enable bit (T2CON[7]). The timer increments until the value reaches full scale in free running mode or when T2WUFD matches the wake-up value, T2VAL.

The wake-up timer is a 32-bit timer. Its current value is stored in two 16-bit registers: T2VAL1 stores the upper 16 bits, and T2VAL0 stores the lower 16 bits.

When T2VAL0 is read, T2VAL1 is frozen at its current value until it is subsequently read. The control bit FREEZE (T2CON[3]) must be set to freeze the T2VAL register between the lower and upper reads.

Clock Selection

Clock selection is made by setting T2CON[10:9].

If LFXTAL is selected (T2CON[10:9] = 01), it must be ensured that the clock circuit is on (XOSCCON[0] = 1).

If PCLK is selected (T2CON[10:9] = 00), configuring T2CON[1:0] = 00 results in a prescaler of 4.

Synchronization to the LFOSC and LFXTAL clock domain is done automatically by hardware, and precautions concerning asynchronous clocks as described in Timer0, Timer1, and Timer3 do not apply.

Compare Field Registers

Hardware Updated Field

T2INC is a 12-bit interval register that is used to update the compare value in T2WUFAx by hardware. When a new value is written in T2INC, Bits[16:5] of the internal 32-bit compare register (T2WUFAx) are loaded with the new T2INC value. If the new compare value is less than the T2WUFD value in periodic mode or less than 0xFFFFFFFF in free running mode, this 32-bit compare register is automatically incremented with the contents of T2INC (shifted by five) each time the wake-up counter reaches the value in this compare register. If the new compare value is greater than these limits, it is recalculated as follows:

In free running mode, the new T2WUFA = old T2WUFA + (32 × T2INC) − 0xFFFFFFFF.

In periodic mode, the new T2WUFA = old T2WUFA + (32 × T2INC) − T2WUFD.

The maximum programmable interval is just above 4 sec.

T2INC is compared with Bits[16:5] of the timer value. Because it is shifted left by five bits, its value must be multiplied by 32 to obtain the compare value.

With the default value of 0xC8 (where for calculation purposes 0xC8 = 200 in decimal), a prescaler = 1, and 32 kHz clock selected,

Interval = ((200 × 32) + 1) × 1/32,768 = 195.3155 ms

To modify the interval value, the timer must be stopped so that the interval register can be loaded in the compare register if T2CON[11] = 0.

To modify the interval value, set STOPINC (T2CON[11] = 1) while the timer is running.

The new T2INC value takes effect after the next Wake-Up Field A interrupt. If the user is writing to this register while the timer is enabled, the STOPINC bit should be set before writing to it, and then STOPINC should be cleared after the update.

Software Updated Field

T2WUFB, T2WUFC, and T2WUFD are 32-bit values programmed by the user in the T2WUFx0 and T2WUFx1 registers (x = B, C, or D). T2WUFD contains the load value when the wake-up timer is configured in periodic mode.

The T2WUFBx and T2WUFCx registers can be written to at any time, but the corresponding interrupt enable—T2IEN[1] or T2IEN[2]—must be disabled. After the register is updated, the interrupt can be reenabled.

In periodic mode, the T2WUFDx registers can be written to only when the timer is disabled. In free running mode, the T2WUFDx registers can be written to while the timer is running. Before doing so, the corresponding interrupt enable (T2IEN[3]) must be disabled. After the register is updated, the interrupt can be reenabled.

In free running mode, T2WUFB, T2WUFC, and T2WUFD can be written to at any time, but the corresponding interrupt enable in the T2IEN register must be disabled. After the register is updated, the interrupt can be reenabled. In periodic mode, this is only applicable to T2WUFB and T2WUFC.

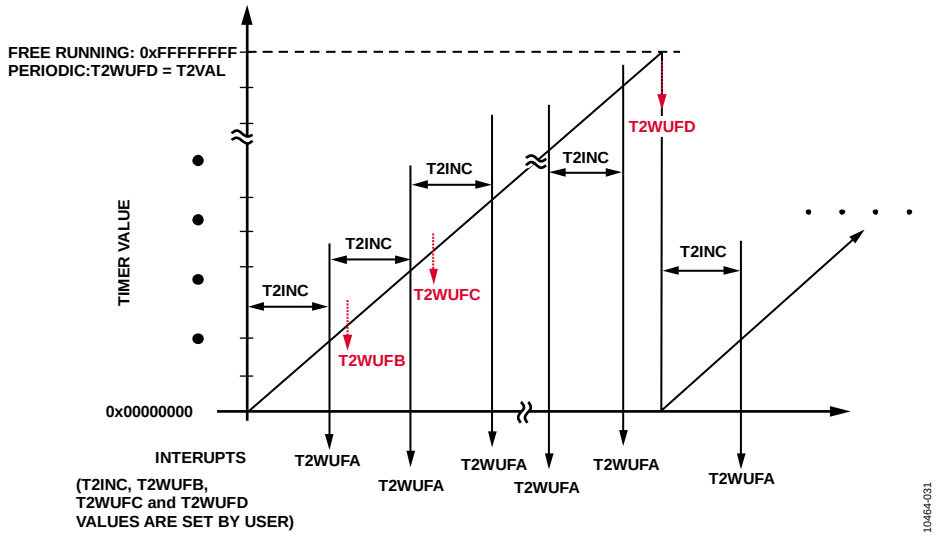

Figure 32. Wake-Up Timer Fields Action

Interrupts/Wake-Up Signals

An interrupt is generated when the counter value corresponds to any of the compare points or full scale in free running mode. The timer continues counting or is reset to 0.

The wake-up timer generates five maskable interrupts. They are enabled in the T2IEN register. Interrupts can be cleared by setting the corresponding bit in the T2CLRI register.

Note that it takes two 32 kHz clock cycles for the interrupt clear to take effect when a 32 kHz clock is used.

Ensure that the register write has fully completed before returning from the interrupt handler. Use the data synchronization barrier (DSB) instruction if necessary.

The following is a code example showing how to implement the DSB ARM Cortex-M3 instruction in a C program.

```
void WakeUp_Int_Handler()
{
WutClrInt(iSource);
__DSB();
}
```

During that time, the part should not be placed in any of the power-down modes. IRQCRY (T2STA[6]) indicates when the device can be placed in power-down mode.

The timer is stopped and reset when clearing the timer enable bit in the T2CON register (T2CON[7]).

WAKE-UP TIMER MEMORY MAPPED REGISTERS

Table 178. Wake-Up Timer Memory Mapped Registers Address Table (Base Address: 0x40002500)

Offset	Name	Description	Access	Default
0x0000	T2VAL0	Current count value LSB	R	0x0000
0x0004	T2VAL1	Current count value MSB	R	0x0000
0x0008	T2CON	Control register	RW	0x0040
0x000C	T2INC	12-bit interval register for Wake-Up Field A	RW	0x00C8
0x0010	T2WUFB0	Wake-Up Field B LSB	RW	0x1FFF
0x0014	T2WUFB1	Wake-Up Field B MSB	RW	0x0000
0x0018	T2WUFC0	Wake-Up Field C LSB	RW	0x2FFF
0x001C	T2WUFC1	Wake-Up Field C MSB	RW	0x0000
0x0020	T2WUFD0	Wake-Up Field D LSB	RW	0x3FFF
0x0024	T2WUFD1	Wake-Up Field D MSB	RW	0x0000
0x0028	T2IEN	Interrupt enable	RW	0x0000
0x002C	T2STA	Status	R	0x0000
0x0030	T2CLRI	Clear interrupts	W	N/A
0x003C	T2WUFA0	Wake-Up Field A LSB	RW	0x1900
0x0040	T2WUFA1	Wake-Up Field A MSB	RW	0x0000

Wake-Up Timer Count Value Registers

Address: 0x40002500, Reset: 0x0000, Name: T2VAL0

Address: 0x40002504, Reset: 0x0000, Name: T2VAL1

Table 179. T2VAL0 Register Bit Descriptions (T2VAL0 Address: 0x40002500)

Bits	Name	Description
15:0	VALUE	Lower 16 bits of current wake-up timer value

Table 180. T2VAL1 Register Bit Descriptions (T2VAL1 Address: 0x40002504)

Bits	Name	Description
15:0	VALUE	Upper 16 bits of current wake-up timer value

Wake-Up Timer Control Register

Address: 0x40002508, Reset: 0x0040, Name: T2CON

Table 181. T2CON Register Bit Descriptions

Bits	Name	Description
15:12	Reserved	Unused bit locations.
11	STOPINC	This bit allows the user to update the interval register safely. 0: Allows the Wake-Up Field A to be updated. 1: Stops Wake-Up Field A from being updated with the interval register value.
10:9	CLK	Clock select: used to select clock source for timer. 00: PCLK (default). 01: LFXTAL: 32 kHz external crystal. 10: LFOSC: 32 kHz internal oscillator. 11: EXTCLK: External clock, from P1.0.
8	WUEN	Wake-up enable bits for time field values. This bit must be set to 1. 0: Disable asynchronous wake-up timer. Interrupt conditions do not wake up the part from sleep mode. 1: Enable asynchronous wake-up timer even when the processor clock is off. When one of the time values equals the T2WUFA, the wake-up output signal is generated.
7	ENABLE	Timer enable bit. This bit must be 0 (low) when changing any of the control information or timer field values. 0: Disable the timer (default). 1: Enable the timer.
6	MOD	Timer free run enable. 0: PERIODIC: Operate in periodic mode; that is, counts up to the value in T2WUFD. 1: FREERUN: Operate in free running mode (default); that is, counts from 0 to FFFF FFFF and then restarts at 0.
5:4	Reserved	Reserved.
3	FREEZE	Freeze enable bit. 0: Disable the freeze enable bit (default). 1: Enable the freeze of the high 16 bits after the lower bits have been read from T2VAL0. This ensures that the software reads an atomic shot of the timer. The entire T2VAL register unfreezes after the high bits (T2VAL1) have been read.
2	Reserved	Reserved.
1:0	PRE	Clock prescaler select. If clock prescaler select = 00 (T2CON[1:0] = 00) and the selected clock source is PCLK (T2CON[10:9] = 00), this setting results in a prescaler of 4. 00: DIV1: source clock/1 (default). 01: DIV16: source clock/16. 10: DIV256: source clock/256. 11: DIV32768: source clock/32,768.

Note that the timer clock needs to be four times slower than PCLK (the system clock); therefore, care needs to be taken when selecting the prescaler at different CD values.

Wake-Up Timer Interval Register: Interval Register

Address: 0x4000250C, Reset: 0x00C8, Name: T2INC

Table 182. T2INC Register Bit Descriptions

Bits	Name	Description
15:12	Reserved	Reserved
11:0	VALUE	Wake-up interval

Wake-Up Timer Wake-Up Field Register B

Address: 0x40002510, Reset: 0x1FFF, Name: T2WUFB0

Address: 0x40002514, Reset: 0x0000, Name: T2WUFB1

The T2WUFBx registers can be written to at any time, but the corresponding interrupt enable (T2IEN[1]) must be disabled. After the register is updated, the interrupt can be reenabled.

Table 183. T2WUFB0 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of Wake-Up Field B

Table 184. T2WUFB1 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Upper 16 bits of Wake-Up Field B

Wake-Up Timer Wake-Up Field Register C

Address: 0x40002518, Reset: 0x2FFF, Name: T2WUFC0

Address: 0x4000251C, Reset: 0x0000, Name: T2WUFC1

T2WUFCx registers can be written to at any time, but the corresponding interrupt enable (T2IEN[2]) must be disabled. After the register is updated, the interrupt can be reenabled.

Table 185. T2WUFC0 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of Wake-Up Field C

Table 186. T2WUFC1 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Upper 16 bits of Wake-Up Field C

Wake-Up Timer Wake-Up Field Register D

Address: 0x40002520, Reset: 0x3FFF, Name: T2WUFD0

Address: 0x40002524, Reset: 0x0000, Name: T2WUFD1

In periodic mode, T2WUFDx registers can be written to only when the timer is disabled. In free running mode, T2WUFDx registers can be written to while the timer is running. Before doing so, the corresponding interrupt enable (T2IEN[3]) must be disabled. After the register is updated, the interrupt can be reenabled.

Table 187. T2WUFD0 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of Wake-Up Field D

Table 188. T2WUFD1 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Upper 16 bits of Wake-Up Field D

Wake-Up Timer Interrupt Enable Register

Address: 0x40002528, Reset: 0x0000, Name: T2IEN

Table 189. T2IEN Register Bit Descriptions

Bits	Name	Description
15:5	Reserved	Unused bit locations.
4	ROLL	Interrupt enable bit when the counter rolls over. Only occurs in free running mode. 0: Disable the rollover interrupt (default). 1: Generate an interrupt when Timer2 rolls over.
3	WUFD	T2WUFD interrupt enable. 0: Disable T2WUFD interrupt (default). 1: Generate an interrupt when T2VAL reaches T2WUFD.
2	WUFC	T2WUFC interrupt enable. 0: Disable T2WUFC interrupt (default). 1: Generate an interrupt when T2VAL reaches T2WUFC.
1	WUFB	T2WUFB interrupt enable. 0: Disable T2WUFB interrupt (default). 1: Generate an interrupt when T2VAL reaches T2WUFB.
0	WUFA	T2WUFA interrupt enable. 0: Disable T2WUFA interrupt (default). 1: Generate an interrupt when T2VAL reaches T2WUFA.

Wake-Up Timer Status Register

Address: 0x4000252C, Reset: 0x0000, Name: T2STA

Table 190. T2STA Register Bit Descriptions

Bits	Name	Description
15:9	Reserved	Unused bit locations.
8	PDOK	Indicates when a change in the enable bit is synchronized to the 32 kHz clock domain. Set to 1 when the enable bit in the control register (T2CON[7]) is set or cleared. Cleared to 0 when the change in the enable bit has been synchronized to the 32 kHz clock domain.
7	FREEZE	Status of T2VAL freeze. Set to 1 when the T2VAL0 is read, indicating T2VAL is frozen. Cleared to 0 when T2VAL1 is read, indicating T2VAL is unfrozen.
6	IRQCRY	Status of wake-up signal to power-down control. Set to 1when any of the interrupts is still set in the external crystal clock domain. Cleared to 0 automatically when the interrupts are cleared.
5	Reserved	Unused bit locations.
4	ROLL	Interrupt status bit for instances when the counter rolls over. Only occurs in free running mode. Set to 1when enabled in the interrupt enable register and the T2VALS counter register is equal to all 1s. Cleared to 0.
3	WUFD	T2WUFD interrupt flag. Set to 1 to indicate that a comparator interrupt has occurred if the corresponding bit is enabled in the interrupt enable register. Cleared to 0 after a write to the corresponding bit in T2CLRI.
2	WUFC	T2WUFC interrupt flag. Set to 1 to indicate that a comparator interrupt has occurred if the corresponding bit is enabled in the interrupt enable register. Cleared to 0 after a write to the corresponding bit in T2CLRI.
1	WUFB	T2WUFB interrupt flag. Set to 1 to indicate that a comparator interrupt has occurred if the corresponding bit is enabled in the interrupt enable register. Cleared to 0 after a write to the corresponding bit in T2CLRI.
0	WUFA	T2WUFA interrupt flag. Set to 1 to indicate that a comparator interrupt has occurred if the corresponding bit is enabled in the interrupt enable register. Cleared to 0 after a write to the corresponding bit in T2CLRI.

Wake-Up Timer Clear Interrupt Register

Address: 0x40002530, Reset: N/A, Name: T2CLRI

Table 191. T2CLRI Register Bit Descriptions

Bits	Name	Description
15:5	Reserved	Unused bit locations.
4	ROLL	Interrupt clear bit for when counter rolls over. Only occurs in free running mode. 0: Cleared automatically after synchronization. 1: Clear a ROLL interrupt flag.
3	WUFD	T2WUFD interrupt flag. 0: Cleared automatically after synchronization. 1: Clear a T2WUFD interrupt flag.
2	WUFC	T2WUFC interrupt flag. 0: Cleared automatically after synchronization. 1: Clear a T2WUFC interrupt flag.
1	WUFB	T2WUFB interrupt flag. 0: Cleared automatically after synchronization. 1: Clear a T2WUFB interrupt flag.
0	WUFA	T2WUFA interrupt flag. 0: Cleared automatically after synchronization. 1: Clear a T2WUFA interrupt flag.

Ensure that the register write has fully completed before returning from the interrupt handler. Use the data synchronization barrier (DSB) instruction if necessary. See the Interrupts/Wake-Up Signals section for a DSB code example.

Wake-Up Timer Compare Register A

Address: 0x4000253C, Reset: 0x1900, Name: T2WUFA0

Address: 0x40002540, Reset: 0x0000, Name: T2WUFA1

T2WUFAx registers can be written to while the timer is enabled. T2WUFA0 must be written to first and when T2WUFA1 is written to both registers values are changed. If a small value is required the a write of 0x0000 to T2WUFA1is sufficient for T2WUFA0 to get updated. T2WUFAx can be written to whether or not the STOPINC (T2CON[11]) bit is set. If set to 1, the STOPINC (T2CON[11])bit stops T2WUFAx from being incremented by hardware.

T2WUFAx is an asynchronous register in the 32 kHz clock domain. Therefore, it should be read twice to confirm that the correct value has been read.

Table 192. T2WUFA0 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Lower 16 bits of Compare Register A

Table 193. T2WUFA1 Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Upper 16 bits of Compare Register A

WATCHDOG TIMER

WATCHDOG TIMER FEATURES

- 16-bit count-down timer, which can be used to recover from an invalid software state.
- Clocked by the 32 kHz internal oscillator (LFOSC) with a programmable prescaler (1, 16, 256, or 4096).

WATCHDOG TIMER BLOCK DIAGRAM

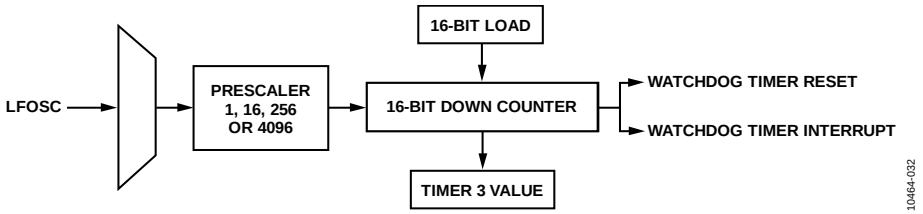

Figure 33. Watchdog Timer Block Diagram

WATCHDOG TIMER OVERVIEW

The watchdog timer (Timer3) is used to recover from an invalid software state. When enabled, it requires periodic servicing to prevent it from forcing a reset of the device. For debug purposes, the timer can be configured to generate an interrupt instead of a reset.

The watchdog timer is clocked by the internal 32.768 kHz oscillator, LFOSC. It is clocked at all times except during a reset.

The watchdog timer is a 16-bit count-down timer with a programmable prescaler. The prescaler is selectable and can divide LFOSC by a factor of 1, 16, 256, or 4096.

WATCHDOG TIMER OPERATION

Enabling the Timer

The watchdog timer is enabled by default after a reset.

User code should disable the watchdog timer at the start of user code when debugging or if the watchdog timer is not required.

```
T3CON = 0x00;                                   // Disable watchdog timer
```

Enabling the watchdog timer (set T3CON[5] = 1) also write protects T3CON and T3LD. This means that after kernel execution, user code can disable the timer and then reconfigure it with T3CON[5] = 1 only once. Then T3CON and T3LD are write protected. T3STA[5] indicates if the timer configuration has been locked. Only a reset clears T3CON[5], unlocking T3CON and T3LD, and allows reconfiguration of the timer.

If T3CON is not modified, user code can change T3LD at any time. If T3CON[5] is cleared to 0, the timer is disabled. Settings can be modified, and the timer can be reenabled.

Programmable Timeout

If the watchdog timer is set to free running (T3CON[6] = 0), the watchdog timer value decrements from 0x1000 to 0, wraps around to 0x1000, and continues to decrement. To achieve a timeout value greater or less than 0x1000 (~32 sec with default prescale T3CON[3:2] = 10, prescale = 256), periodic mode should be used (T3CON[6] = 1) and T3LD and T3CON[3:2] (prescale) should be written with the values corresponding to the desired timeout period. The maximum timeout is ~8192 sec (T3LD = 0xFFFF, T3CON[3:2] = 11, prescale = 4096).

Reset/Interrupt or Refreshing the Timer

The default timeout period is approximately 32 sec. A watchdog timer timeout can generate a reset or an interrupt when the watchdog timer decrements to 0. The T3CON[1] bit is added to allow selection of an interrupt or a reset. This can be used for debug. The interrupt can be cleared by writing 0xCCCC to the T3CLRI write only register.

The interrupt or reset can be prevented by writing T3CLRI with 0xCCCC before the expiration period. A write to T3CLRI causes the watchdog timer to reload with the T3LD (or 0x1000 if in free running mode) immediately to begin a new timeout period and restart counting. If any value other than 0xCCCC is written, an interrupt or reset is generated (as selected by T3CON[1]).

Asynchronous Clock Source

The watchdog timer is clocked by the internal 32.768 kHz oscillator, LFOSC. Three bits are provided in the T3STA register to ensure the correct synchronization of the timer clock and processor clock domains. After a write to T3CON, T3LD, or T3CLRI, user code should wait until the corresponding status bit is cleared.

The T3VAL register always contains the timer value synchronized to the core clock domain.

WATCHDOG TIMER MEMORY MAPPED REGISTERS

Table 194. Watchdog Timer Memory Mapped Registers Address Table (Base Address: 0x40002580)

Offset	Name	Description	Access	Default
0x0000	T3LD	Load value.	RW	0x1000
0x0004	T3VAL	Current count value, read only.	R	0x1000
0x0008	T3CON	Control register.	RW	0x00E9
0x000C	T3CLRI	Clear interrupt.	W	N/A
0x0018	T3STA	Status register.	R	0x0000

Watchdog Timer Load Register

Address: 0x40002580, Reset: 0x1000, Name: T3LD

Table 195. T3LD Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Load value.

Watchdog Timer Current Value Register

Address: 0x40002584, Reset: 0x1000, Name: T3VAL

Table 196. T3VAL Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Current count value. Read only register.

Watchdog Timer Control Register

Address: 0x40002588, Reset: 0x00E9, Name: T3CON

Table 197. T3CON Register Bit Descriptions

Bits	Name	Description
15:7	Reserved	Reserved. These bits should be written 0.
6	MOD	Timer mode. 0: FREERUN: Operate in free running mode. Note that in free running, it wraps around at 0x1000. 1: PERIODIC: Operate in periodic mode (default).
5	ENABLE	Timer enable. 0: Disable the timer. Can only be done once. 1: Enable the timer (default).
4	Reserved	Reserved
3:2	PRE	Prescaler. 00: DIV1: source clock/1. 01: DIV16: source clock/16. 10: DIV256: source clock/256 (default). 11: DIV4096: source clock/4096.
1	IRQ	Timer interrupt. 0: Generate a reset on a timeout (default). 1: Generate an interrupt when the timer times out. This feature is provided for debug purposes and is only available in active mode.
0	PD	Power-down clear. 0: Continue the count upon exit from power-down modes. 1: Reset, reload, and restart the timer count upon exit from power-down modes (default).

Watchdog Timer Clear Interrupt Register

Address: 0x4000258C, Reset: N/A, Name: T3CLRI

Table 198. T3CLRI Register Bit Descriptions

Bits	Name	Description
15:0	VALUE	Clear watchdog. User writes 0xCCCC to reset/reload/restart T3 or clear interrupt. A write of any other value causes a watchdog reset/interrupt. Write only, reads 0.

Ensure that the register write has fully completed before returning from the interrupt handler. Use the data synchronization barrier (DSB) instruction if necessary. The following is a code example showing how to implement the DSB ARM Cortex-M3 instruction in a C program.

```
void WDog_Tmr_Int_Handler()
{
 WdtClrInt();        // clear watchdog timer interrupt
__DSB();
}
```

Watchdog Timer Status Register

Address: 0x40002598, Reset: 0x0000, Name: T3STA

Table 199. T3STA Register Bit Descriptions

Bits	Name	Description
15:5	Reserved	Reserved.
4	LOCK	Lock status bit. Set to 1 automatically in hardware when user code sets T3CON[5]. Cleared to 0 after any reset and until user code sets T3CON[5].
3	CON	T3CON write sync in progress. Set to 1 with the timer not ready to receive commands to T3CON. The previous change of the T3CON value has not been synchronized in the timer clock domain. Cleared to 0 when the timer is ready to receive commands to T3CON. The previous change of the T3CON value has been synchronized in the timer clock domain.
2	LD	T3LD write sync in progress. Set to 1 when the previous change of the T3LD value has not been synchronized in the timer clock domain. Cleared to 0 when the previous change of T3LD has been synchronized in the timer clock domain.
1	CLRI	T3CLRI write sync in progress. Set to 1 automatically when the T3CLRI value is being updated in the timer clock domain, indicating that the timer's configuration is not yet valid. Cleared to 0 when the interrupt is cleared in the timer clock domain.
0	IRQ	WD_IRQ. Set to 1 when a T3 interrupt is pending. Cleared to 0 when a T3 interrupt is not pending.

PWM

PWM FEATURES

- 3 pairs of PWM outputs individually controlled
- H-Bridge mode supported on 2 pairs

PWM OVERVIEW

The ADuCM360/ADuCM361 integrates a 6-channel PWM interface. The six outputs are grouped as three pairs (0 to 2). Pair 0 and Pair 1 can be configured in standard mode or to drive an H-bridge. Pair 2 and Pair 3 can be configured in standard mode only. The PWM pairs and modes are summarized in Table 200.

Table 200: PWM Channel Grouping

Pair	Port Name	Description	PWM Mode Available
0	PWM0	High-side PWM output	H-Bridge and standard
	PWM1	Low-side PWM output	H-Bridge and standard
1	PWM2	High-side PWM output	H-Bridge and standard
	PWM3	Low-side PWM output	H-Bridge and standard
2	PWM4	High-side PWM output	Standard
	PWM5	Low-side PWM output	Standard

On power-up, the PWM outputs default to H-bridge mode for pair 0 and pair 1. In all modes, users have control over the period of each pair of outputs and over the duty cycle of each individual output.

In the event of external fault conditions, a falling edge on the PWM_{TRIP} pin provides an instantaneous shutdown of the PWM controller. All PWM outputs are placed in the off state, that is, in low state for the low side and high state for the high side, and a PWM_{TRIP} interrupt can be generated.

PWM OPERATION

The PWM clock is selectable via PWMCON0 with one of the following values: UCLK divided by 2, 4, 8, 16, 32, 64, 128, or 256. In all modes, the PWMxCOMx MMRs controls the point at which the PWM output changes state. An example is shown in Figure 34.

Each pair has an associated counter. The length of the PWM period is defined by PWMxLEN.

The PWM waveforms are set by the count value of the 16-bit timer and the compare register contents.

The low-side waveform, PWM1, goes high when the timer count reaches PWM0LEN, and it goes low when the timer count reaches the value held in PWM0COM2 or when the high-side waveform PWM0 goes low.

The high-side waveform, PWM0, goes high when the timer count reaches the value held in PWM0COM0, and it goes low when the timer count reaches the value held in PWM0COM1.

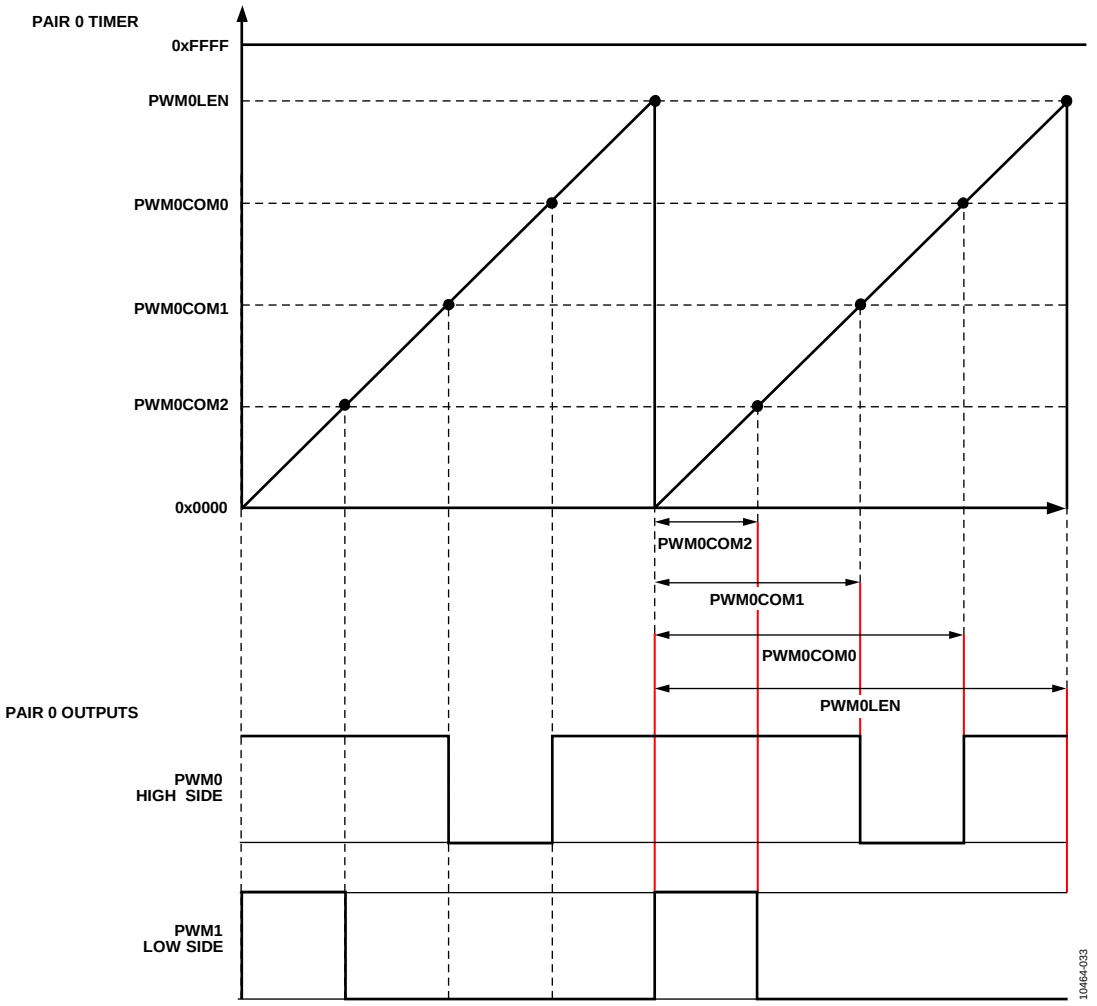

Figure 34. Waveform of PWM Channel Pair in Standard Mode

Note that the high-side PWM output for each channel must have a high duration period greater than or equal to the high period duration of the low-side output. For example, the high period for PWM0 must be equal to or greater than the high period of PWM1.

Table 201 lists equations for the period and duration for both the outputs of a PWM channel.

Table 201. PWM Equations

PWM	Period	Duration

Low Side (PWM1)	$t_{UCLK/DIV} \times (PWM0LEN + 1) \times N_{PRESCALE}$	High Duration
		If (PWM0COM2 < PWM0COM1), then
		$t_{UCLK/DIV} \times (PWM0LEN - PWM0COM2) \times N_{PRESCALE}$
		Else
		$t_{UCLK/DIV} \times (PWM0LEN - PWM0COM1) \times N_{PRESCALE}$

High Side (PWM0)	$t_{UCLK/DIV} \times (PWM0LEN + 1) \times N_{PRESCALE}$	Low Duration $t_{UCLK/DIV} \times (PWM0COM0 - PWM0COM1) \times N_{PRESCALE}$

Note that:

- $t_{UCLK/DIV}$ is the PWM clock frequency selected by CLKCON1[14:12] and CLKSYSDIV[0].
- $N_{PRESCALE}$ = prescalar value as determined by PWMCON0[8:6].

Standard Mode

In standard mode, each pair is individually controlled by a selection of registers, as shown in Table 202.

Table 202. Compare Register Descriptions in Standard Mode (Base Address: 0x40001000)

Pair	Name	Description
0	PWM0COM0	PWM0 output goes high when the PWM timer reaches the count value stored in this register.
	PWM0COM1	PWM0 output goes low when the PWM timer reaches the count value stored in this register.
	PWM0COM2	PWM1 output goes low when the PWM timer reaches the count value stored in this register.
	PWM0LEN	PWM1 output goes high when the PWM timer reaches the count value stored in this register.
1	PWM1COM0	PWM2 output goes high when the PWM timer reaches the count value stored in this register.
	PWM1COM1	PWM2 output goes low when the PWM timer reaches the count value stored in this register.
	PWM1COM2	PWM3 output goes low when the PWM timer reaches the count value stored in this register.
	PWM1LEN	PWM3 output goes high when the PWM timer reaches the count value stored in this register.
2	PWM2COM0	PWM4 output goes high when the PWM timer reaches the count value stored in this register.
	PWM2COM1	PWM4 output goes low when the PWM timer reaches the count value stored in this register.
	PWM2COM2	PWM5 output goes low when the PWM timer reaches the count value stored in this register.
	PWM2LEN	PWM5 output goes high when the PWM timer reaches the count value stored in this register.

The following code configures the duty cycle of the PWM0/PWM1 outputs to PWM0 high for 30% of the period and to PWM1 high for 20% of the period:

```
pADI_CLKCTL-> CLKCON0 = 0;
pADI_CLKCTL-> CLKCON1= 0;
pADI_PWM->PWM0LEN = 0x50;                       // Set PWM period to 21.9uS (456 kHz)
ucValid = PWM0_1DutyCycle(30,20);
                                                    // Pass values between 0-99 for duty cycle
of PWM0/1    - PWM 1 must have a duty cycle <=PWM0
unsigned char  PWM0_1DutyCycle( int iPWM0High, int iPWM1High)
{
   if ((iPWM1High >=100) ||(iPWM0High >=100) )
         return 1;                              // Error
   if  (iPWM0High >= iPWM1High)
   {
          pADI_PWM->PWM0COM0      =  PWM0LEN;
          pADI_PWM->PWM0COM1      =  ( int)((PWM0LEN * (iPWM0High)/100));
       pADI_PWM->PWM0COM2 = ( int)((PWM0LEN * iPWM1High)/100);
              return 0;
   }
   else
   {
         return 1;                              // PWM0 High period must be > PWM1 High period
   }
}
```

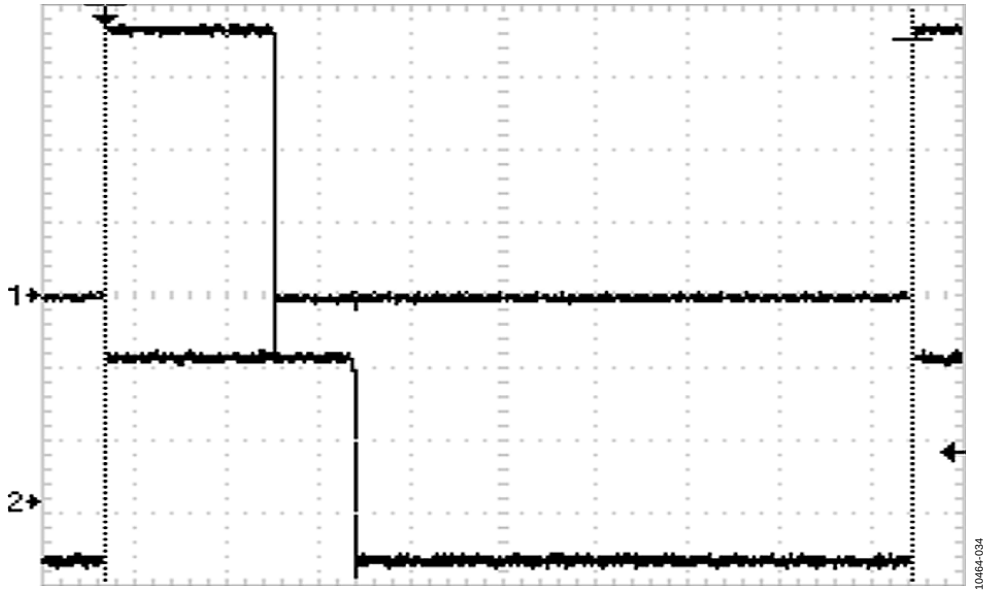

Figure 35. PWM Output on PWM0 and PWM1 Pins Generated by Above Code; PWM0 is Channel 2

H-Bridge Mode

In H-bridge mode, the period and duty cycle of the 4 outputs are controlled using the Pair 0 registers: PWM0COM0, PWM0COM1, PWM0COM2, and PWM0LEN. In addition, the state of the output is controlled by PWMCON0 Bit 9, Bit 5, Bit 4, and Bit 2 as summarized in Table 203.

An example of H-bridge configuration is shown in Figure 36. Note that only PWM0 to PWM3 participate in H-bridge mode; other outputs (PWM4 and PWM5) do not and continue to generate standard mode output.

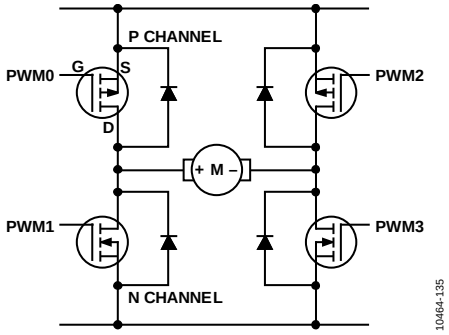

Figure 36. Example H-Bridge Configuration

Table 203. PWM Output in H-Bridge Mode

PWM Control Bits				PWM Outputs[1]				
ENA PWMCON0[9]	POINV PWMCON0[5]	HOFF PWMCON0[4]	DIR PWMCON0[2]	PWM0	PWM1	PWM2	PWM3	State of Motor
0	X	0	X	1 (Disable)	1 (Enable)	1 (Disable)	1 (Enable)	Brake
X	X	1	X	1 (Disable)	0 (Disable)	1 (Disable)	0 (Disable)	Free run
1	0	0	0	0 (Enable)	0 (Disable)	HS	LS	Move controlled by LS on PWM3
1	0	0	1	HS	LS	0 (Enable)	0 (Disable)	Move controlled by LS on PWM1
1	1	0	0	$\overline{LS}$	$\overline{HS}$	1 (Disable)	1 (Enable)	Move controlled by $\overline{LS}$ on PWM0
1	1	0	1	1 (Disable)	1 (Enable)	$\overline{LS}$	$\overline{HS}$	Move controlled by $\overline{LS}$ on PWM2

[1] HS = high side, LS = low side, $\overline{HS}$ = inverse of high side, $\overline{LS}$ = inverse of low side, as programmed in PWM0 registers.

PWM INTERRUPT GENERATION

PWM Trip Function Interrupt

When the PWM trip function is enabled (TRIPEN, PWMCON1[6]) and the PWM trip input signal goes low (falling edge), the PWM peripheral disables itself (PWMCON0[0] = 0). It also generates the PWM trip interrupt. The interrupt is cleared by setting PWMCLRI[4].

When using the PWM trip interrupt, the PWM interrupt should be cleared before exiting the ISR. This prevents the generation of multiple interrupts.

PWM Output Pairs Interrupts

In standard mode, each PWM pair has a dedicated interrupt: IRQPWM0, IRQPWM1, IRQPWM2.

When the interrupt generation is enabled (PWMCON0[10]) and the counter value for Pair 0 changes from PWM0LEN to 0, it also generates the IRQPWM0 interrupt.

The interrupt is cleared by setting PWMCLRI[0].

When the interrupt generation is enabled (PWMCON0[10]) and the counter value for Pair 1 changes from PWM1LEN to 0, it also generates the IRQPWM1 interrupt.

The interrupt is cleared by setting PWMCLRI[1].

When the interrupt generation is enabled (PWMCON0[10]) and the counter value for Pair 2 changes from PWM2LEN to 0, it also generates the IRQPWM2 interrupt.

The interrupt is cleared by setting PWMCLRI[2].

In H-bridge mode, Pair 0 and Pair 1 are used in the bridge configuration and generate 1 interrupt only, IRQPWM0. While Pair 0 and Pair 1 are in H-bridge mode, Pair 2 can be used in standard mode and it generates the IRQPWM2 interrupt.

PWM MEMORY MAPPED REGISTERS

Table 204. PWM Memory Mapped Registers Address Table (Base Address: 0x40001000)

Offset	Name	Description	Access	Default
0x000	PWMCON0	PWM Control Register 0	RW	0x0012
0x004	PWMCON1	Trip control register	RW8[1]	0x0000
0x008	PWMCLRI	PWM interrupt clear	W	0x0000
0x010	PWM0COM0	Compare Register 0 for PWM0 and PWM1	RW	0x0000
0x014	PWM0COM1	Compare Register 1 for PWM0 and PWM1	RW	0x0000
0x018	PWM0COM2	Compare Register 2 for PWM0 and PWM1	RW	0x0000
0x01C	PWM0LEN	Period value register for PWM0 and PWM1	RW	0x0000
0x020	PWM1COM0	Compare Register 0 for PWM2 and PWM3	RW	0x0000
0x024	PWM1COM1	Compare Register 1 for PWM2 and PWM3	RW	0x0000
0x028	PWM1COM2	Compare Register 2 for PWM2 and PWM3	RW	0x0000
0x02C	PWM1LEN	Period value register for PWM2 and PWM3	RW	0x0000
0x030	PWM2COM0	Compare Register 0 for PWM4 and PWM5	RW	0x0000
0x034	PWM2COM1	Compare Register 1 for PWM4 and PWM5	RW	0x0000
0x038	PWM2COM2	Compare Register 2 for PWM4 and PWM5	RW	0x0000
0x03C	PWM2LEN	Period value register for PWM4 and PWM5	RW	0x0000

[1] RW8 is eight bits, read or write.

PWM Control Register 0

Address: 0x40001000, Reset: 0x0020, Name: PWMCON0

Table 205. PWMCON0 Register Bit Descriptions

Bit Position	Name	Description
15	SYNC	PWM synchronization. 0: Ignore transitions on the PWMSYNC pin. 1: All PWM counters are reset on the next clock edge after the detection of a falling edge on the PWNSYNC pin.
14	Reserved	
13	PWM5INV	Pair 2 low-side polarity (PWM5). Available in standard mode only. 0: PWM5 is normal. 1: Invert PWM5.
12	PWM3INV	Pair 1 low-side polarity (PWM3). Available in standard mode only. 0: PWM3 is normal. 1: Invert PWM3.
11	PWM1INV	Pair 0 low-side polarity (PWM1). Available in standard mode only. 0: PWM1 is normal. 1: Invert PWM1.
10	PWMIEN	0: Disables the PWM interrupts. 1: Enables PWM interrupts.
9	ENA	If HOFF = 0 and HMODE = 1. Available in H-bridge mode only. 0: Disable PWM outputs. 1: Enable PWM outputs.
8:6	PRE	PWM clock prescaler. Sets UCLK divider. 000: UCLK/2. 001: UCLK/4. 010: UCLK/8. 011: UCLK/16. 100: UCLK/32. 101: UCLK/64. 110: UCLK/128. 111: UCLK/256.
5	POINV	PWM polarity. Available in H-bridge mode only. 0: PWM outputs as normal. 1: Invert all PWM outputs.
4	HOFF	High-side off. Available in H-bridge mode only. 0: PWM outputs as normal. 1: Forces PWM0 and PWM2 outputs high and PWM1 and PWM3 low (default).
3	LCOMP	Load compare registers. Available in H-bridge and standard modes. 0: Use the values previously stored in the internal compare registers. 1: Load the internal compare registers with the values in PWMxCOMx on the next transition of the PWM timer from 0x00 to 0x01 (PWM timer overflow).
2	DIR	Direction control. Available in H-bridge mode only. 0: Enable PWM2 and PWM3 as the output signals while PWM0 and PWM1 are held low. 1: Enables PWM0 and PWM1 as the output signals while PWM2 and PWM3 are held low.
1	MOD	PWM mode of operation. 0: PWMs in standard mode. 1: Enables H-bridge mode and Bits[5:2] of PWMCON0 (default).
0	ENABLE	PWM output enable. 0: Disables all PWM outputs. 1: Enables all PWM outputs.

Note that

- Except for LCOMP, all other bits of PWMCON0 register can be changed only when ENABLE is low.
- When LCOMP is written with Value 1, it stays at that value until the new value is loaded in the compare registers for all the channels.

PWM1 Control Register

Address: 0x40001004, Reset: 0x0020, Name: PWMCON1

Table 206. PWMCON1 Register Bit Descriptions

Bits	Name	Description
15:7	Reserved	Reserved. Reads 0.
6	TRIPEN	0: Disable PWM trip functionality. 1: Enable PWM trip functionality.
5:0	Reserved	Reserved.

PWMCLRI Register

Address: 0x40001008, Reset: 0x0020, Name: PWMCLRI

Table 207. PWMCLRI Register Bit Descriptions

Bits	Name	Description
15:5	Reserved	Reserved. These bits always read 0.
4	TRIP	1: Clear the latched TRIPIRQPWM interrupt. This bit always reads 0.
3	Reserved	This bit always reads 0.
2	PWM2	1: Clear the latched IRQPWM2 interrupt. This bit always reads 0.
1	PWM1	1: Clear the latched IRQPWM1 interrupt. This bit always reads 0.
0	PWM0	1: Clear the latched IRQPWM0 interrupt. This bit always reads 0.

POWER SUPPLY SUPPORT CIRCUITS

POWER SUPPLY SUPPORT CIRCUITS FEATURES

Low Dropout Regulators

The ADuCM360/ADuCM361 integrates an on-chip regulator (AVDD_REG-based LDO) that is driven directly from the AVDD supply to generate a 1.8 V internal supply.

This regulated supply is then used as the supply voltage for the ARM Cortex-M3 and peripherals, including the precision analog circuits on chip. The LDO has its own output pin, AVDD_REG. It needs an external capacitor of 0.47 µF for stability. Turning on and off the LDO is done automatically when selecting the operation mode of the device. It is not possible to overdrive the LDO.

Pin 7, DVDD_REG must be connected to DGND via a 470 nF capacitor. Pin 18, AVDD_REG, must be connected to AGND via a 470 nF capacitor. Pin 7 , DVDD_REG, and Pin 18, AVDD_REG, must also be connected together. See Figure 39.

Power-On Reset

The power-on reset (POR) function is also integrated to ensure safe operation of the processor. The POR circuit is designed to guarantee full functional operation of the Flash/EE memory-based processor during power-on and power-down cycles.

As shown in Figure 37, when the supply voltage on AVDD reaches a minimum operating voltage of 1.6 V and the LDO output is 1.65 V, a POR signal keeps the processor in reset for 50 ms. The output of the POR is available on P0.6.

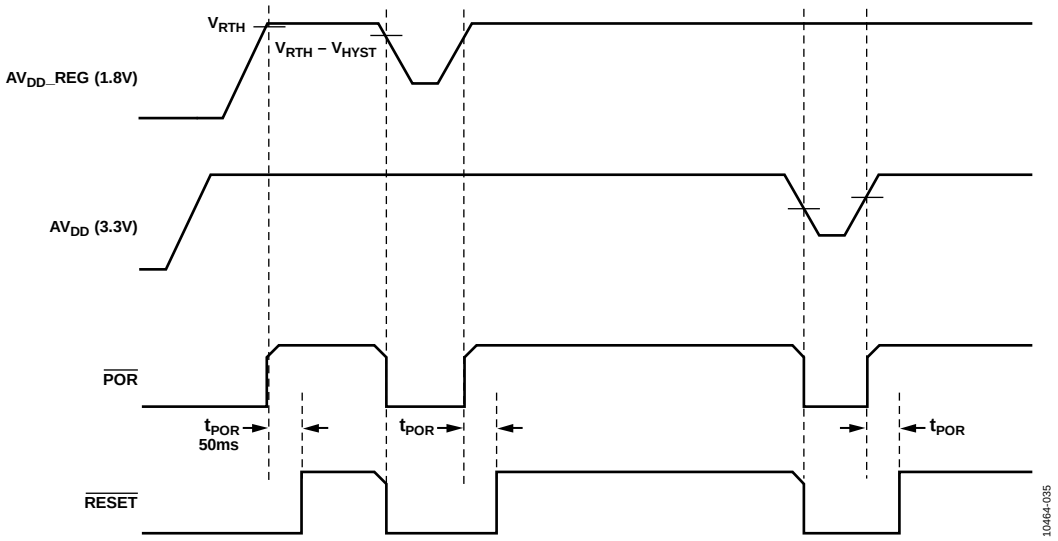

Figure 37. Typical Power-On Cycle

HARDWARE DESIGN CONSIDERATIONS

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

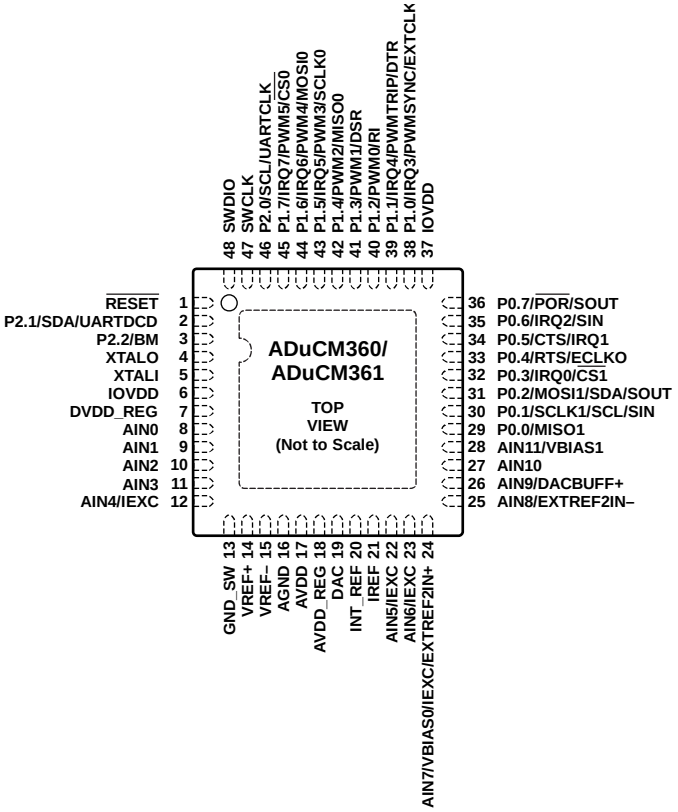

Figure 38. Pin Configuration

Table 208. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$\overline{\text{RESET}}$	Reset Pin, Active Low Input. An internal pull-up is provided.
2	P2.1/SDA/UARTDCD	General-Purpose Input/Output P2.1/I²C Serial Data Pin/UART Data Carrier Detect Pin.
3	P2.2/BM	General-Purpose Input/Output P2.2/Boot Mode Input Select Pin. When this pin is held low during any reset sequence and for a short time after, the part enters UART download mode.
4	XTALO	External Crystal Oscillator Output Pin. Optional 32.768 kHz source for real-time clock.
5	XTALI	External Crystal Oscillator Input Pin. Optional 32.768 kHz source for real-time clock.
6	IOVDD	Digital System Supply Pin. This pin must be connected to DGND via a 0.1 µF capacitor.
7	DVDD_REG	This pin must be connected to DGND via a 470 nF capacitor and to Pin 18, AVDD_REG, as shown in Figure 39.
8	AIN0	ADC Analog Input 0. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode.
9	AIN1	ADC Analog Input 1. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode.
10	AIN2	ADC Analog Input 2. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode.
11	AIN3	ADC Analog Input 3. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode.
12	AIN4/IEXC	ADC Analog Input 4/Excitation Current Source. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN4). This pin can also be configured as the output pin for Excitation Current Source 0 or Excitation Current Source 1 (IEXC).
13	GND_SW	Sensor Power Switch to Analog Ground Reference.

Pin No.	Mnemonic	Description
14	VREF+	External Reference Positive Input. An external reference can be applied between the VREF+ and VREF− pins.
15	VREF−	External Reference Negative Input. An external reference can be applied between the VREF+ and VREF− pins.
16	AGND	Analog System Ground Reference Pin.
17	AVDD	Analog System Supply Pin. This pin must be connected to AGND via a 0.1 µF capacitor.
18	AVDD_REG	Internal Analog Regulator Supply Output. This pin must be connected to AGND via a 470 nF capacitor and to Pin 7, DVDD_REG, as shown in Figure 39..
19	DAC	DAC Voltage Output.
20	INT_REF	Internal Reference. This pin must be connected to ground via a 470 nF decoupling capacitor.
21	IREF	Optional Reference Current Resistor Connection for the Excitation Current Sources. The reference current used for the excitation current sources is set by a low drift (5 ppm/°C) external resistor connected to this pin.
22	AIN5/IEXC	ADC Analog Input 5/Excitation Current Source. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN5). This pin can also be configured as the output pin for Excitation Current Source 0 or Excitation Current Source 1 (IEXC).
23	AIN6/IEXC	ADC Analog Input 6/Excitation Current Source. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN6). This pin can also be configured as the output pin for Excitation Current Source 0 or Excitation Current Source 1 (IEXC).
24	AIN7/VBIAS0/IEXC/EXTREF2IN+	ADC Analog Input 7/Bias Voltage Output/Excitation Current Source/External Reference 2 Positive Input. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN7). This pin can also be configured as an analog output pin to generate a bias voltage, VBIAS0 of AVDD_REG/2 (VBIAS0); as the output pin for Excitation Current Source 0 or Excitation Current Source 1 (IEXC); or as the positive input for External Reference 2 (EXTREF2IN+).
25	AIN8/EXTREF2IN−	ADC Analog Input 8/External Reference 2 Negative Input. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN8). This pin can also be configured as the negative input for External Reference 2 (EXTREF2IN−). When DACCON[8] = 1, V_{REF} is present on AIN8.
26	AIN9/DACBUFF+	ADC Analog Input 9/Noninverting Input to the DAC Output Buffer. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN9). This pin can also be configured as the noninverting input to the DAC output buffer when the DAC is configured for NPN mode (DACBUFF+).
27	AIN10	ADC Analog Input 10. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode.
28	AIN11/VBIAS1	ADC Analog Input 11/Bias Voltage Output. This pin can be configured as a positive or negative input to either ADC in differential or single-ended mode (AIN11). This pin can also be configured as an analog output pin to generate a bias voltage, VBIAS1 of AVDD_REG/2 (VBIAS1).
29	P0.0/MISO1	General-Purpose Input/Output P0.0/SPI1 Master Input, Slave Output Pin.
30	P0.1/SCLK1/SCL/SIN	General-Purpose Input/Output P0.1/SPI1 Serial Clock Pin/I²C Serial Clock Pin/UART Serial Input (Data Input for the UART Downloader).
31	P0.2/MOSI1/SDA/SOUT	General-Purpose Input/Output P0.2/SPI1 Master Output, Slave Input Pin/I²C Serial Data Pin/ UART Serial Output (Data Output for the UART Downloader).
32	P0.3/IRQ0/$\overline{CS1}$	General-Purpose Input/Output P0.3/External Interrupt Request 0/SPI1 Chip Select Pin (Active Low).
33	P0.4/RTS/ECLKO	General-Purpose Input/Output P0.4/UART Request-to-Send Signal/External Clock Output Pin for Test Purposes.
34	P0.5/CTS/IRQ1	General-Purpose Input/Output P0.5/UART Clear-to-Send Signal/External Interrupt Request 1.
35	P0.6/IRQ2/SIN	General-Purpose Input/Output P0.6/External Interrupt Request 2/UART Serial Input.
36	P0.7/$\overline{POR}$/SOUT	General-Purpose Input/Output P0.7/Power-On Reset Pin (Active Low)/UART Serial Output.
37	IOVDD	Digital System Supply Pin. This pin must be connected to DGND via a 0.1 µF capacitor.
38	P1.0/IRQ3/PWMSYNC/EXTCLK	General-Purpose Input/Output P1.0/External Interrupt Request 3/PWM External Synchronization Input/External Clock Input Pin.
39	P1.1/IRQ4/PWMTRIP/DTR	General-Purpose Input/Output P1.1/External Interrupt Request 4/PWM External Trip Input/ UART Data Terminal Ready Pin.
40	P1.2/PWM0/RI	General-Purpose Input/Output P1.2/PWM0 Output/UART Ring Indicator Pin.
41	P1.3/PWM1/DSR	General-Purpose Input/Output P1.3/PWM1 Output/UART Data Set Ready Pin.
42	P1.4/PWM2/MISO0	General-Purpose Input/Output P1.4/PWM2 Output/SPI0 Master Input, Slave Output Pin.

Pin No.	Mnemonic	Description
43	P1.5/IRQ5/PWM3/SCLK0	General-Purpose Input/Output P1.5/External Interrupt Request 5/PWM3 Output/SPI0 Serial Clock Pin.
44	P1.6/IRQ6/PWM4/MOSI0	General-Purpose Input/Output P1.6/External Interrupt Request 6/PWM4 Output/SPI0 Master Output, Slave Input Pin.
45	P1.7/IRQ7/PWM5/$\overline{CS0}$	General-Purpose Input/Output P1.7/External Interrupt Request 7/PWM5 Output/SPI0 Chip Select Pin (Active Low).
46	P2.0/SCL/UARTCLK	General-Purpose Input/Output P2.0/I^2C Serial Clock Pin/Input Clock Pin for UART Block Only.
47	SWCLK	Serial Wire Debug Clock Input Pin.
48	SWDIO	Serial Wire Debug Data Input/Output Pin.
	EP	Exposed Pad. The LFCSP has an exposed pad that must be soldered to a metal plate on the PCB for mechanical reasons and to DGND.

TYPICAL SYSTEM CONFIGURATION

Figure 39 shows a typical ADuCM360/ADuCM361 configuration. This figure illustrates some of the hardware considerations. The bottom of the LFCSP package has an exposed pad that must be soldered to a metal plate on the PCB for mechanical reasons and to DGND. The metal plate of the PCB can be connected to ground. The 0.47 µF capacitor on the AVDD_REG and DVDD_REG pins should be placed as close to the pins as possible. In noisy environments, an additional 1 nF capacitor can be added to IOVDD and AVDD.

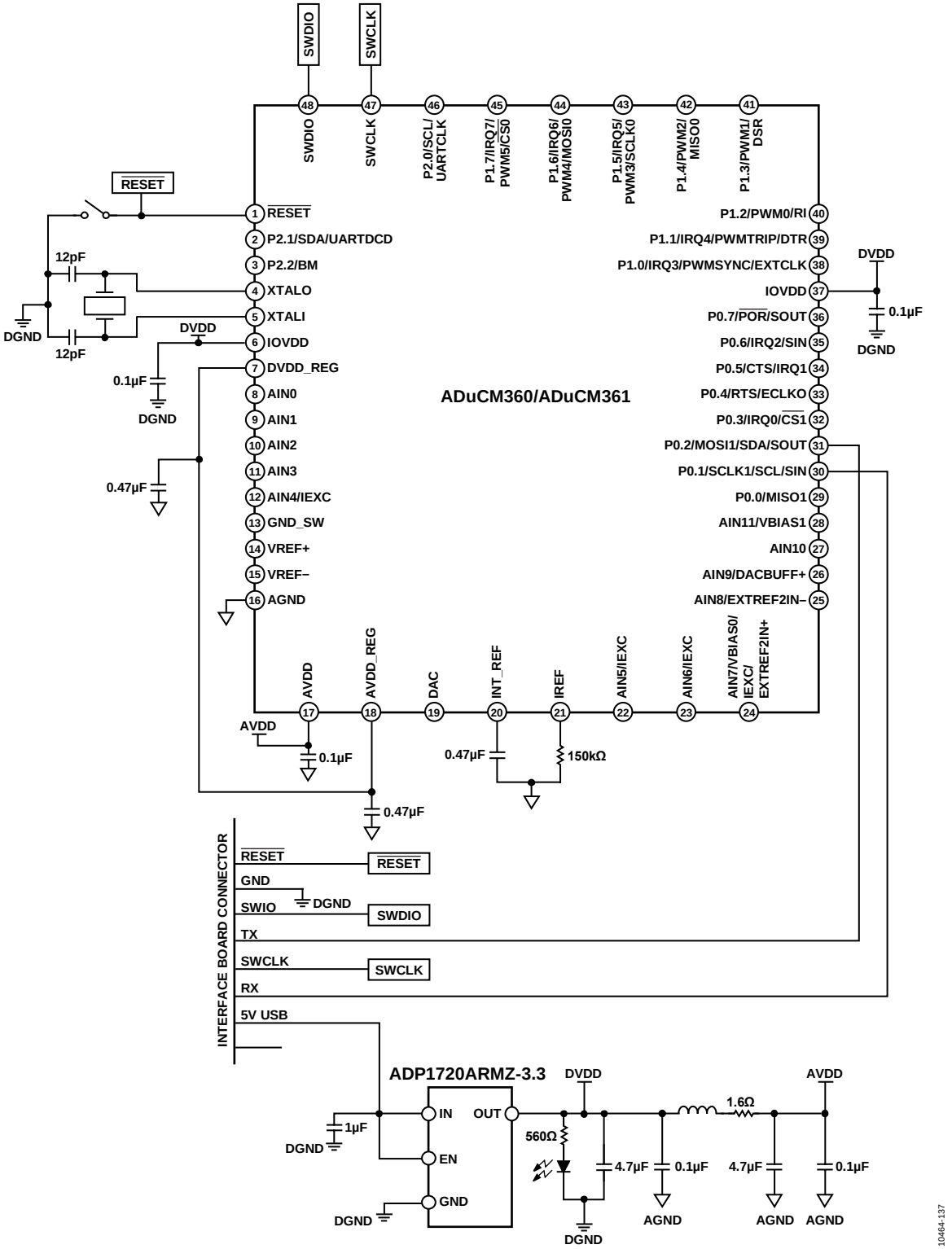

Figure 39. Typical System Configuration

SERIAL WIRE DEBUG INTERFACE

Serial wire debug (SWD) provides a debug port for pin limited packages. SWD replaces the 5-pin JTAG port with a clock (SWDCLK) and a single bidirectional data pin (SWDIO), providing all the normal JTAG debug and test functionality. SWDIO and SWCLK are overlaid on the TMS and TCK pins on the ARM 20-pin JTAG interface.

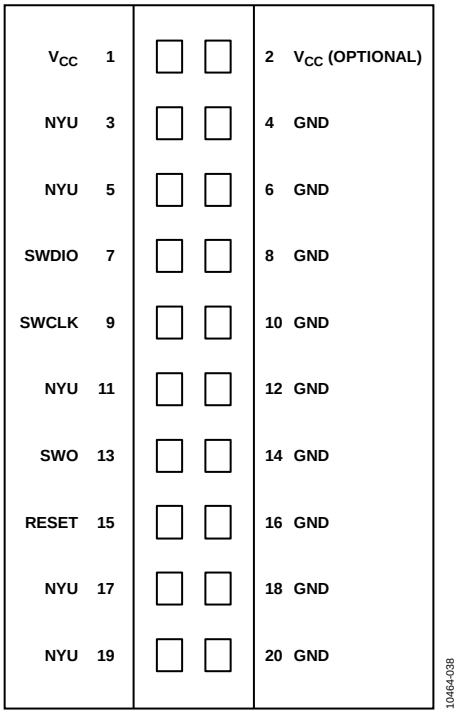

Figure 40. SWD 20-Pin Connector Pinout

Table 209. SWD Connections

Signal	Connect to
SWDIO	Data I/O pin. Use a 100 kΩ pull-up resistor to VCC.
SWO	No connect.
SWCLK	Clock pin. Use a 100 kΩ pull-up resistor to VCC.
VCC	Positive supply voltage—power supply for JTAG interface drivers.
GND	Digital ground.
RESET	No connect.

RELATED LINKS

Resource	Description
ADuCM360	Product Page, ADuCM360
ADuCM361	Product Page, ADuCM361
ADuCM360/ADuCM361 Data sheet	ADuCM360/ADuCM361 data sheet

I²C refers to a communications protocol originally developed by Philips Semiconductors (now NXP semiconductors).

UG-367 **UG10464-0-5/14(C)**