QADC
QUEUED ANALOG-TO-DIGITAL CONVERTER
REFERENCE MANUAL

PREFACE

This manual describes the capabilities, operation, and functions of the queued analog-to-digital converter (QADC). The following conventions are used throughout the manual.

Logic level one is the voltage that corresponds to Boolean true (1) state.

Logic level zero is the voltage that corresponds to Boolean false (0) state.

To **set** a bit or bits means to establish logic level one on the bit or bits.

To **clear** a bit or bits means to establish logic level zero on the bit or bits.

A signal that is **asserted** is in its active logic state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.

A signal that is **negated** is in its inactive logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

A **specific bit or signal** within a range is referred to by mnemonic and number. For example, ADDR15 is bit 15 of the address bus. **A range of bits or signals** is referred to by mnemonic and the numbers that define the range. For example, DATA[7:0] form the low byte of the data bus.

TABLE OF CONTENTS

Paragraph **Title** **Page**

SECTION 1 OVERVIEW

SECTION 2 SIGNAL DESCRIPTIONS

SECTION 3 CONFIGURATION AND CONTROL

SECTION 4 EXTERNAL MULTIPLEXING

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 5 PIN CONNECTION CONSIDERATIONS

SECTION 6 ANALOG SUBSYSTEM

SECTION 7 DIGITAL CONTROL

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 8 INTERRUPTS

SECTION 9 QUEUE PRIORITY EXAMPLES

APPENDIX A ELECTRICAL CHARACTERISTICS

APPENDIX B REGISTER SUMMARY

APPENDIX C CONVERSION ACCURACY DEFINITIONS

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

INDEX

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

SECTION 1 OVERVIEW

The queued analog-to-digital converter (QADC) is a 10-bit, unipolar, successive approximation converter. A maximum of 16 analog input channels can be supported using internal multiplexing. A maximum of 44 input channels can be supported in the expanded, externally multiplexed mode. The actual number of channels depends upon the number of pins available to the QADC module.

1.1 Block Diagram

Figure 1-1 displays the major components of the QADC module. The QADC consists of an analog front-end and a digital control subsystem, which includes an intermodule bus (IMB) interface block.

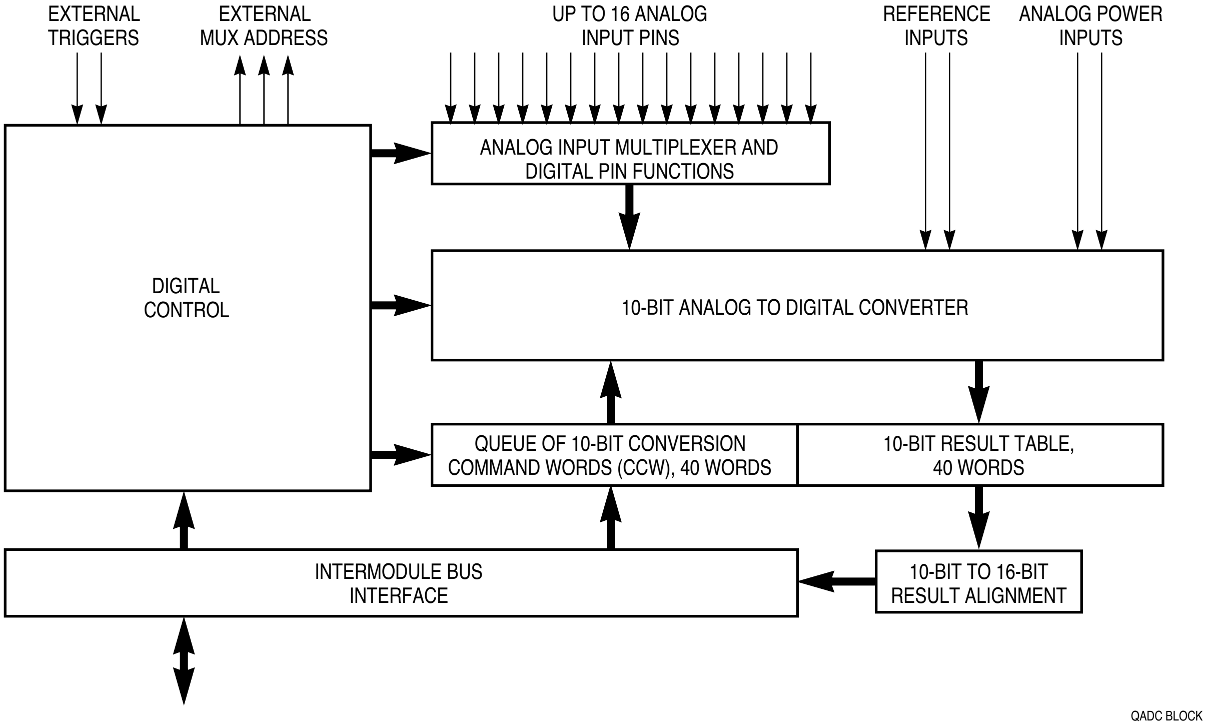

EXTERNAL TRIGGERS
EXTERNAL MUX ADDRESS
UP TO 16 ANALOG INPUT PINS
REFERENCE INPUTS
ANALOG POWER INPUTS
DIGITAL CONTROL
ANALOG INPUT MULTIPLEXER AND DIGITAL PIN FUNCTIONS
10-BIT ANALOG TO DIGITAL CONVERTER
QUEUE OF 10-BIT CONVERSION COMMAND WORDS (CCW), 40 WORDS
10-BIT RESULT TABLE, 40 WORDS
INTERMODULE BUS INTERFACE
10-BIT TO 16-BIT RESULT ALIGNMENT
QADC BLOCK

Figure 1-1 QADC Block Diagram

The analog section includes input pins, an analog multiplexer, and two sample and hold analog circuits. The analog conversion is performed by the digital-to-analog converter (DAC) resistor-capacitor array and a high-gain comparator.

The digital control section contains control logic to sequence the conversion process, channel select logic, and a successive approximation register (SAR). Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table RAM, and the result word table RAM.

The bus interface unit (BIU) allows the QADC to operate with the applications software through the IMB environment.

1.2 QADC Features

- Sample and hold
- Up to 16 analog input channels using internal multiplexing
- Directly supports up to four external multiplexers (for example, the MC14051)
- Up to 44 total input channels with internal and external multiplexing
- Programmable input sample time for various source impedances
- Two conversion command queues of variable length
- Sub-queues possible using pause mechanism
- Queue complete and pause software interrupts available on both queues
- Automated queue modes initiated by:
  - External trigger
  - Periodic/interval timer, within QADC module
  - Software command
- Single-scan or continuous-scan of queues
- 40 result registers
- Output data readable in three formats:
  - Right-justified unsigned
  - Left-justified signed
  - Left-justified unsigned
- Unused analog channels can be used as digital ports

1.3 Memory Map

The QADC occupies 512 bytes, or 256 words, of address space. Nine words are control, port, and status registers, 40 words are the CCW table, and 40 words are the result word table which occupy 120 address locations because the result data is readable in three data alignment formats. The remaining words are reserved for expansion. **Table 1-1** displays the QADC memory map.

Each register address in **Table 1-1** consists of a 15-bit base address plus a 9-bit offset. The offset represents the nine low order bits of register address. "$####" represents the 15-bit base address plus the high order bit of the offset. (For example, both LJUR and RJUR registers are represented by $####B0–$####FE, even though they represent different addresses since each register has a different most significant bit in their offset). For the precise locations of these registers, refer to the appropriate microcontroller unit (MCU) manual. The column labeled "Access" specifies which address space is designated supervisor data space or unrestricted data space. Remember that the MSB is determined by the MM bit.

Table 1-1 QADC Address Map

Access	Address	Offset	15 8	7 0
S[1]	$####00	$000	MODULE CONFIGURATION REGISTER (QADCMCR)	
S	$####02	$002	TEST REGISTER (QADCTEST)	
S	$####04	$004	INTERRUPT REGISTER (QADCINT)	
S/U[2]	$####06	$006	PORT A DATA (PORTQA)	PORT B DATA (PORTQB)
S/U	$####08	$008	PORT DATA DIRECTION REGISTER (DDRQA)	
S/U	$####0A	$00A	CONTROL REGISTER 0 (QACR0)	
S/U	$####0C	$00C	CONTROL REGISTER 1 (QACR1)	
S/U	$####0E	$00E	CONTROL REGISTER 2 (QACR2)	
S/U	$####10	$010	STATUS REGISTER (QASR)	
—	$####12–$####2E	$012–$02E	RESERVED	
S/U	$####30–$####7E	$030–$07E	CONVERSION COMMAND WORD (CCW) TABLE	
—	$####80–$####AE	$080–$0AE	RESERVED	
S/U	$####B0–$####FE	$0B0–$0FE	RESULT WORD TABLE RIGHT JUSTIFIED, UNSIGNED RESULT REGISTER (RJURR)	
—	$####00–$####2E	$100–$12E	RESERVED	
S/U	$####30–$####7E	$130–$17E	RESULT WORD TABLE LEFT JUSTIFIED, SIGNED RESULT REGISTER (LJSRR)	
—	$####80–$####AE	$180–$1AE	RESERVED	
S/U	$####B0–$####FE	$1B0–$1FE	RESULT WORD TABLE LEFT JUSTIFIED, UNSIGNED RESULT REGISTER (LJURR)	

NOTES:
1. S = Supervisor only
2. S/U = Unrestricted or supervisor depending on the state of the SUPV bit in the QADCMCR.

Access to supervisor-only data space is permitted only when the software is operating in supervisor access mode. Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user data space addresses. The SUPV bit in the module configuration register designates the assignable space as supervisor or unrestricted.

Attempts to read a supervisor-only data space when not in the supervisor access mode causes a value of $0000 to be returned. Attempts to read assignable data space in the unrestricted access mode when the space is programmed as supervisor space causes a value of $FFFF to be returned. Attempts to write supervisor-only or supervisor-assigned data space when in the unrestricted access mode has no effect.

The CPU32 indicates the supervisor and user space access with the function code signal FC2 on the IMB bus. CPU16 does not support supervisor/user space selection, and is always in the supervisor access mode. In such cases, the SUPV bit has no meaning or effect.

The first block of the address map contains the nine words used for control, status, port, and test information. The control registers permit the software to initialize the QADC for the desired configuration and queue operating mode. Also included are the status bits that the software may read to identify an interrupt source and to determine other information about the operation of the QADC. Refer to **APPENDIX B REGISTER SUMMARY** for more information.

The QADC has three global registers for configuring module operation: the module configuration register (QADCMCR), the interrupt register (QADCINT), and a test register (QADCTEST). The global registers are always defined to be in supervisor-only data space. When the CPU supports the supervisor/user address data space designations, software can establish the global registers to be in supervisor data space and the remaining registers and tables to be in user space.

All QADC analog channel/port pins that are not used for analog input channels can be used as digital port pins. Port values are read/written by accessing the port A and B data registers (PORTQA and PORTQB). The digital port pins are specified as inputs or outputs by programming the port data direction register (DDRQA). Only port A uses open drain pull-down output drivers.

The remaining four registers in the control register block control the operation of the queuing mechanism, and provide a means of monitoring the operation of the QADC. Control register 0 (QACR0) contains hardware configuration information. Control register 1 (QACR1) is associated with queue 1, and control register 2 (QACR2) is associated with queue 2. The status register (QASR) provides visibility on the status of each queue and the particular conversion that is in progress.

Following the register block in the address map is the CCW table. There are 40 words to hold the desired analog conversion sequences. Each CCW is a 16-bit word, with ten implemented bits in four fields. Refer to **APPENDIX B REGISTER SUMMARY** for more information.

The final block of address space belongs to the result word table, which appears in three places in the memory map. Each result word table location holds one 10-bit conversion value. The software selects one of three data formats, which map the 10-bit result onto the 16-bit data bus by reading the address which produces the desired alignment. The first address block presents the result data in right justified format, the second block is presented in left justified signed format, and the third is presented in left justified unsigned format. Refer to **APPENDIX B REGISTER SUMMARY** for more information.

SECTION 2 SIGNAL DESCRIPTIONS

The QADC uses a maximum of 21 external pins. There are 16 channel/port pins that can support up to 44 channels when external multiplexing is used (including internal channels). All of the channel pins can also be used as general-purpose digital port pins. In addition, there are also two analog reference pins, two analog submodule power pins, and one V_{SSE} pin for the open drain output drivers on port A.

The QADC allows external trigger inputs and the multiplexer outputs to be combined onto some of the channel pins. All of the channel pins are used for at least two functions, depending on the modes in use.

The following paragraphs describe QADC pin functions. **Figure 2-1** displays the QADC module pins.

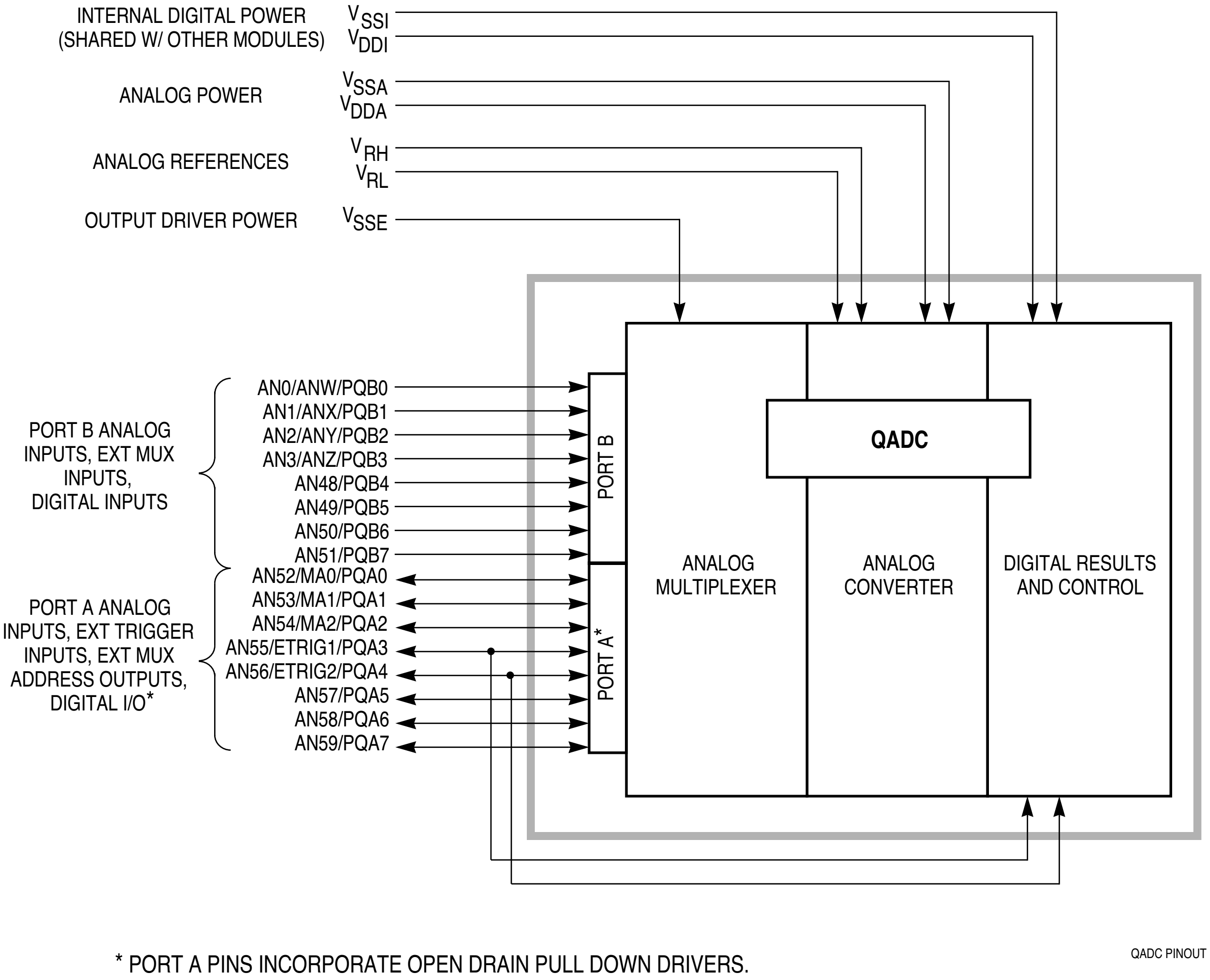


Figure 2-1 QADC Input and Output Signals

2.1 Port A Pin Functions

The eight port A pins can be used as analog inputs, or as a bidirectional 8-bit digital input/output port. Refer to the following paragraphs for more information.

2.1.1 Port A Analog Input Pins

When used as analog inputs, the eight port A pins are referred to as AN[59:52]. Due to the digital output drivers associated with port A, the analog characteristics of port A are different from those of port B. Refer to **APPENDIX B REGISTER SUMMARY** for more information on analog signal characteristics. All of the analog signal input pins may be used for at least one other purpose.

2.1.2 Port A Digital Input/Output Pins

Port A pins are referred to as PQA[7:0] when used as a bidirectional 8-bit digital input/output port. These eight pins may be used for general-purpose digital input signals or digital open drain pull-down output signals.

Port A pins are connected to a digital input synchronizer during reads and may be used as general purpose digital inputs when the applied voltages meet high voltage input (V_{IH}) and low voltage input (V_{IL}) requirements. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information on voltage requirements.

Each port A pin is configured as an input or output by programming the upper half of the port data direction register (DDRQA). The digital input signal states are read by the software in the upper half of the port data register when the port data direction register specifies that the pins are inputs. The digital data in the port data register is driven onto the port A pins when the corresponding bit in the port data direction register specifies output. Refer to **APPENDIX B REGISTER SUMMARY** for more information. Since the outputs are open drain drivers (so as to minimize the effects to the analog function of the pins), active external pull-up provisions must be made when the output is used to drive another integrated circuit.

2.2 Port B Pin Functions

The eight port B pins can be used as analog inputs, or as an 8-bit digital input only port. Refer to the following paragraphs for more information.

2.2.1 Port B Analog Input Pins

When used as analog inputs, the eight port B pins are referred to as AN[51:48]/AN[3:0]. Since port B functions as analog and digital input only, the analog characteristics are different from those of port A. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information on analog signal characteristics. All of the analog signal input pins may be used for at least one other purpose.

2.2.2 Port B Digital Input Pins

Port B pins are referred to as PQB[7:0] when used as an 8-bit digital input only port. In addition to functioning as analog input pins, the port B pins are also connected to the input of a synchronizer during reads and may be used as general-purpose digital inputs when the applied voltages meet V_{IH} and V_{IL} requirements. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information on voltage requirements.

Since port B pins are input only, a data direction register is not necessary. The digital input signal states are read by the software in the lower half of the port data register. Refer to **APPENDIX B REGISTER SUMMARY** for more information.

2.3 External Trigger Input Pins

The QADC uses two external trigger pins (ETRIG[2:1]). The external trigger inputs are included in two multifunction port A pins (PQA[4:3]), which are normally used as analog channel input pins. Each of the two input external trigger pins is associated with one of the scan queues, queue 1 and queue 2. When a queue is in an external trigger mode, the corresponding external trigger pin is configured as a digital input and the software programmed input/output direction for the external trigger pins in the data direction register (DDRQA) is ignored. Refer to paragraph 7.7 in **SECTION 7 DIGITAL CONTROL** for more information.

2.4 Multiplexed Address Output Pins

In the non-multiplexed mode, the 16 channel pins are connected to an internal multiplexer which routes the analog signals into the internal A/D converter.

In the externally multiplexed mode, the QADC allows automatic channel selection through up to four external 8-to-1 selector chips. The QADC provides a 3-bit multiplexed address output to the external multiplex chips to allow selection of one of eight inputs. The multiplexed address output signals (MA[2:0]) can be used as multiplex address output bits, or as general I/O.

MA[2:0] are used as the address inputs for up to four 8-channel multiplexer chips (for example, the MC14051 and the MC74HC4051). Since the MA[2:0] pins are digital outputs in the multiplexed mode, the software programmed input/output direction for the multiplex address pins in the data direction register is ignored.

Refer to paragraph 7.7 in **SECTION 7 DIGITAL CONTROL** for more information on the use of multiplexed address output pins in the external multiplexed mode.

2.5 Multiplexed Analog Input Pins

In the external multiplexed mode, four of the port B pins are redefined to each represent eight input channels. Refer to **Table 2-1**.

Table 2-1 Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw	Even numbered channels from 0 to 14
ANx	Odd numbered channels from 1 to 15
ANy	Even channels from 16 to 30
ANz	Odd channels from 17 to 31

Refer to paragraph 7.7 in **SECTION 7 DIGITAL CONTROL** for more information.

2.6 Voltage Reference Pins

V_{RH} and V_{RL} are the dedicated input pins for the high and low reference voltages. Separating the reference inputs from the power supply pins allows for additional external filtering, which increases reference voltage precision and stability, and subsequently contributes to a higher degree of conversion accuracy. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information.

2.7 Dedicated Analog Supply Pins

V_{DDA} and V_{SSA} pins supply power to the analog subsystems of the QADC module. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on the digital power supply. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information.

2.8 External Digital Supply Pin

Each port A pin includes a digital open drain output driver, as well as an analog input signal path and a digital input synchronizer. The V_{SSE} pin provides the ground level for the drivers on the port A pins. Since the QADC output pins have open drain type drivers, a dedicated V_{DDE} pin is not needed. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information.

2.9 Internal Digital Supply Pins

V_{DDI} and V_{SSI} pins provide the power for the digital portions of the QADC, and for all other digital modules on the microcontroller chip. Since these pins are common to all modules, they are not counted as QADC pins. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information.

SECTION 3 CONFIGURATION AND CONTROL

The QADC module communicates with other microcontroller modules via the intermodule bus (IMB). The QADC bus interface unit (BIU) coordinates IMB activity with internal QADC bus activity. This section describes the operation of the BIU, IMB read/ write accesses to QADC memory locations, module configuration, and general-purpose I/O operation.

3.1 QADC Bus Interface Unit

The BIU is designed to act as a slave device on the IMB. The BIU has the following functions: to respond with the appropriate bus cycle termination, and to supply IMB interface timing to all internal module signals.

BIU components consist of IMB buffers, address match and module select logic, interrupt and arbitration logic, the BIU state machine, clock prescaler logic, data bus routing logic, and the interface to the internal module data bus.

The QADC responds to all IMB operations and signals, allowing byte, word, and long word addressable read and write operations in any addressable space.

NOTE

Normal accesses to the QADC require two clocks. However, if the CPU tries to access locations that are also accessible by the QADC while the QADC is accessing them, the QADC produces wait states. From one to four CPU wait states may be inserted by the QADC in the process of reading and writing.

3.2 QADC Bus Accessing

The QADC permits software access to 8-bit, 16-bit words, and 32-bit long words, at even and odd addresses, however, coherency (ensuring that all samples are taken consecutively in one scan) is not provided for accesses that require more than one bus cycle. For example, if a read of two consecutive word locations in a result area are made, the QADC could change one word in the result area between the bus cycles. There is no holding register for the second word. Refer to paragraph 7.6.3 in **SECTION 7 DIGITAL CONTROL** for more information on coherency. All read and write accesses that require more than one 16-bit access to complete occur as two or more independent bus cycles. These accesses include misaligned and long word accesses.

NOTE

CPU32 does not support word access or long word access to an odd address. Both of these are considered misaligned accesses. The CPU16 supports misaligned and long word accesses.

Figure 3-1 shows the three bus cycles which are implemented by the QADC. The following paragraphs describe how the three types of accesses are used, including misaligned and long word accesses.

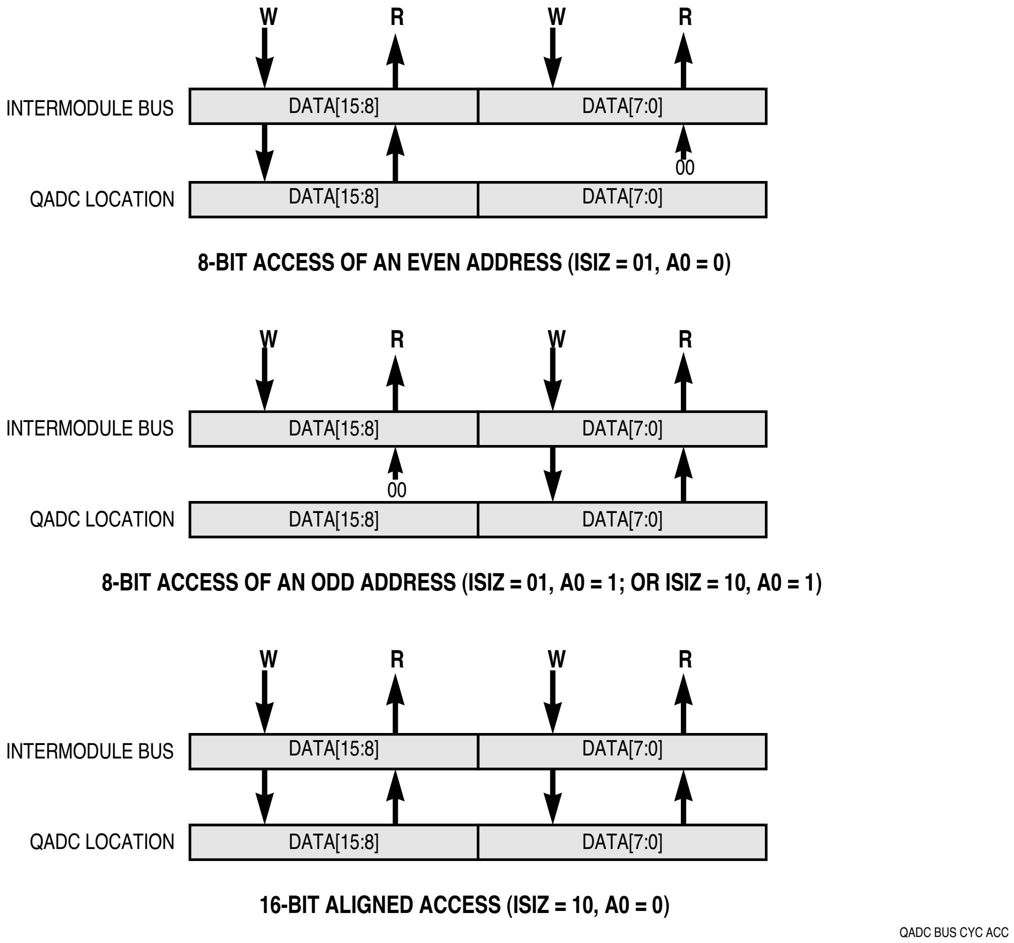


Figure 3-1 Bus Cycle Accesses

Byte accesses to an even address read or write bits 15 through 8 of a QADC location, as shown in the top illustration of **Figure 3-1**. In the case of write cycles, bits 7 through 0 of the addressed QADC register are not disturbed. In the case of read cycles, the QADC provides zeros to the IMB for bits 7 through 0.

Byte accesses to an odd address read or write bits 7 through 0 of a QADC location, as shown in the center illustration of **Figure 3-1**. In the case of write cycles, bits 15 through 8 of the addressed QADC location are not disturbed. In the case of read cycles, the QADC provides zeros to the IMB for bits 15 through 8.

Word accesses to an even address read or write bits 15 through 0 of a QADC location, as shown in the lower illustration of **Figure 3-1**. A full 16 bits of data is written to and read from the QADC location with each access.

Word accesses to an odd address require two bus cycles; half of two different 16-bit QADC locations are accessed. The first bus cycle is treated by the QADC as an 8-bit read or write of an odd address. The second cycle is an 8-bit read or write of an even address. The QADC address space is organized into 16-bit even address locations, so a 16-bit read or write of an odd address obtains or provides the lower half of one QADC location, and the upper half of the following QADC location.

Long word (32-bit) accesses to an even address require two bus cycles to complete the access, and two full QADC locations are accessed. The first bus cycle reads or writes the addressed 16-bit QADC location and the second cycle reads or writes the following 16-bit location.

Long word accesses to an odd address require three bus cycles. Portions of three different QADC locations are accessed. The first bus cycle is treated by the QADC as an 8-bit access of an odd address, the second cycle is a 16-bit aligned access, and the third cycle is an 8-bit access of an even address. The QADC address space is organized into 16-bit even address locations, so a 32-bit read or write of an odd address provides the lower half of one QADC location, the full 16-bit content of the following QADC location, and the upper half of the third QADC location.

3.3 Module Configuration

The module configuration register (QADCMCR) contains parameters which allow the QADC to interface with other MCU modules. The register defines freeze and stop mode operation, supervisor-only access protection, and the QADC software interrupt arbitration priority number. The implemented fields can be read and written. Unimplemented locations read zero and writes have no effect. They are typically written once when the software initializes the QADC, and not changed afterwards.

3.3.1 Low Power Stop Mode

When the STOP bit in the QADCMCR is set, the clock signal to the A/D converter is disabled, effectively turning off the analog circuitry. This results in a static, low power consumption, idle condition. The stop mode aborts any conversion sequence in progress. Because the bias currents to the analog circuits are turned off in stop mode, the QADC requires some recovery time (T_{SR} in **APPENDIX A ELECTRICAL CHARACTERISTICS**) to stabilize the analog circuits after the stop enable bit is cleared.

In the stop mode, the BIU state machine and logic do not shut down, and the QADCMCR, the interrupt register (QADCINT), and the test register (QADCTEST) are fully accessible and are not reset. The data direction register (DDRQA), port data register (PORTQA/PORTQB), and control register 0 (QACR0) are not reset and are read-only accessible. The RAM is not reset and is not accessible. Control register 1 (QACR1), control register 2 (QACR2), and the status register (QASR) are reset and are read-only accessible. In addition, the QADC clock (QCLK) and the periodic/interval timer are held in reset during stop mode.

If the STOP bit is clear, stop mode is disabled.

3.3.2 Freeze Mode

Freeze mode occurs when the background debug mode is enabled in the device integration module and a breakpoint is encountered. This is indicated by the assertion of the internal FREEZE line on the IMB. The FRZ bit in the QADCMCR determines whether or not the QADC responds to an IMB internal FREEZE signal assertion. Freeze is very useful when debugging an application.

When the FRZ bit is set, the QADC finishes any current conversion then freezes. Depending on when the FRZ bit is asserted, there are three possible queue “freeze” scenarios:

- When a queue is not executing, the QADC freezes immediately.
- When a queue is executing, the QADC completes the current conversion and then freezes.
- If during the execution of the current conversion, the queue operating mode for the active queue is changed, or a queue 2 abort occurs, the QADC freezes immediately.

When the QADC enters the freeze mode while a queue is active, the current CCW location of the queue pointer is saved.

During freeze, the analog clock is held in reset and is not clocked. Although QCLK is unaffected, the periodic/interval timer is held in reset. External trigger events that occur during the freeze mode are not recorded. The BIU remains active to allow IMB access to all QADC registers and RAM. Although the QADC saves a pointer to the next CCW in the current queue, the software can force the QADC to execute a different CCW by writing new queue operating modes for normal operation. The QADC looks at the queue operating modes, the current queue pointer, and any pending trigger events to decide which CCW to execute.

If the FRZ bit is clear, the internal FREEZE signal is ignored.

3.3.3 Supervisor/Unrestricted Address Space

The QADC memory map is divided into two segments: supervisor-only data space and assignable data space. Access to supervisor-only data space is permitted only when the software is operating in supervisor access mode. Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user data space accesses. The SUPV bit in the QADCMCR designates the assignable space as supervisor or unrestricted.

Attempts to read a supervisor-only data space when not in the supervisor access mode causes a value of $0000 to be returned. Attempts to read assignable data space in the unrestricted access mode when the space is programmed as supervisor space causes a value of $FFFF to be returned. Attempts to write supervisor-only or supervisor-assigned data space when in the unrestricted access mode has no effect.

The CPU indicates the supervisor and user space access with the function code bits (FC[2:0]) on the IMB bus.

NOTE

> Some CPUs do not support supervisor/user space selection, and are always in the supervisor access mode. In such cases, the SUPV bit has no meaning or effect.

The supervisor-only data space segment contains the QADC global registers, which include the QADCMCR, the QADCTEST, and the QADCINT. The supervisor/unrestricted space designation for the CCW table, the result word table, and the remaining QADC registers are programmable.

3.3.4 Interrupt Arbitration Priority

Each module that can generate interrupts, including the QADC, has an IARB (interrupt arbitration number) field in the QADCMCR. Each IARB field must have a different value. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level.

The reset value of IARB in the QADCMCR is $F. This prevents QADC interrupts from being discarded. Initialization software must set the IARB field to a lower value in the range $F (highest priority) to $1 (lowest priority) if lower priority interrupts are to be arbitrated.

Refer to **SECTION 8 INTERRUPTS** for more information.

3.3.5 QADC Module Configuration Register

The QADCMCR contains fields and bits that control freeze and stop modes and determines the privilege level required to access most registers. It also contains the IARB field.

QADCMCR — Module Configuration Register **$####00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	NOT USED						SUPV	NOT USED			IARB			
RESET:															
0	0							1				0	0	0	0

STOP — Stop Enable

0 = Disable stop mode.
1 = Enable stop mode.

FRZ — Freeze Enable

0 = Ignores the IMB internal FREEZE signal.
1 = Finish any current conversion, then freeze

SUPV — Supervisor/Unrestricted Data Space

0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted.
1 = All QADC registers and tables are designated as supervisor-only data space.

IARB[3:0] — Interrupt Arbitration Number

IARB determines QADC interrupt arbitration priority. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest value). Refer to **SECTION 8 INTERRUPTS** for more information.

3.4 QADC Test Register

QADCTEST — QADC Test Register **$####02**

QADCTEST is used only during factory testing of the MCU.

3.5 General-Purpose I/O Port Operation

Each of the port pins, when used as a general-purpose input, is conditioned by a synchronizer with an enable feature. The synchronizer is not enabled until the QADC decodes an IMB bus cycle which addresses the port data register to minimize the high-current effect of mid-level signals on the inputs used for analog signals. Digital input signals must meet the input low voltage (V_{IL}) or input high voltage (V_{IH}) specifications in **APPENDIX A ELECTRICAL CHARACTERISTICS**. If an analog input pin does not meet the digital input pin specifications when a digital port read operation occurs, an indeterminate state is read.

During a port data register read, the actual value of the pin is reported when its corresponding bit in the data direction register defines the pin to be an input (port A only). When the data direction bit specifies the pin to be an output, the content of the port data register is read. By reading the latch which drives the output pin, software instructions that read data, modify it, and write the result, like bit manipulation instructions, work correctly. When a reduced number of digital port pins are implemented on a particular microcontroller version, the unused bit positions are read as a zero and write operations do not have any effect.

There are two special cases to consider for the digital I/O port operation. When the MUX (externally multiplexed) bit is set in QACR0, the data direction register settings are ignored for the bits corresponding to PQA[2:0], the three multiplexed address (MA[2:0]) output pins. The MA[2:0] pins are forced to be digital outputs, regardless of the data direction setting, and the multiplexed address outputs are driven. The data returned during a port data register read is the value of the multiplexed address latches which drive MA[2:0], regardless of the data direction setting.

Similarly, when an external trigger queue operating mode is selected, the data direction setting for the corresponding pins, PQA3 or PQA4, is ignored. The port pins are forced to be digital inputs for ETRIG1 and/or ETRIG2. The data driven during a port data register read is the actual value of the pin, regardless of the data direction setting.

3.5.1 Port Data Register

QADC ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB).

Port A pins are referred to as PQA[7:0] when used as a bidirectional 8-bit input/output port that may be used for general-purpose digital input signals or digital open drain pull-down output signals. Port A can also be used for analog inputs (AN[59:52], external trigger inputs (ETRIG[2:1]), and external multiplexer address outputs (MA[2:0]).

Port B pins are referred to as PQB[7:0] when used as an input-only 8-bit digital port that may be used for general-purpose digital input signals. Data for PQB[7:0] is accessed in the lower half of the QPDR. Port B can also be used for nonmultiplexed (AN[51:48]/AN[3:0]) and multiplexed (ANz, ANy, ANx, ANw) analog inputs.

PORTQA and PORTQB are unaffected by reset.

PORTQA — Port A Data Register **$####06**
PORTQB — Port B Data Register **$####07**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PQA7	PQA6	PQA5	PQA4	PQA3	PQA2	PQA1	PQA0	PQB7	PQB6	PQB5	PQB4	PQB3	PQB2	PQB1	PQB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN3	AN2	AN1	AN0
EXTERNAL TRIGGER INPUTS:															
			ETRIG2	ETRIG1											
MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

3.5.2 Port Data Direction Register

The port data direction register (DDRQA) is associated with the port A digital I/O pins. The bidirectional pins have somewhat higher leakage and capacitance specifications. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. The software is responsible for ensuring that DDR bits are not set to one on pins used for analog inputs. When the DDR bit is set to one and the pin is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

NOTE

> Caution should be exercised when mixing digital and analog inputs. This should be isolated as much as possible. Rise and fall times should be as large as possible to minimize AC coupling effects.

Since port B is input-only, a data direction register is not needed. Therefore, the lower byte of the port data direction register is not implemented. Read operations on the reserved bits return zeros, and writes have no effect.

DDRQA — Port Data Direction Register **$####08**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDQA7	DDQA6	DDQA5	DDQA4	DDQA3	DDQA2	DDQA1	DDQA0	RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECTION 4 EXTERNAL MULTIPLEXING

External multiplexer chips concentrate a number of analog signals onto a few inputs to the analog converter. This is helpful in applications that need to convert more analog signals than the A/D converter can normally provide. External multiplexing also puts the multiplex chip closer to the signal source. This minimizes the number of analog signals that need to be shielded due to the close proximity to noisy high speed digital signals at the microcontroller chip.

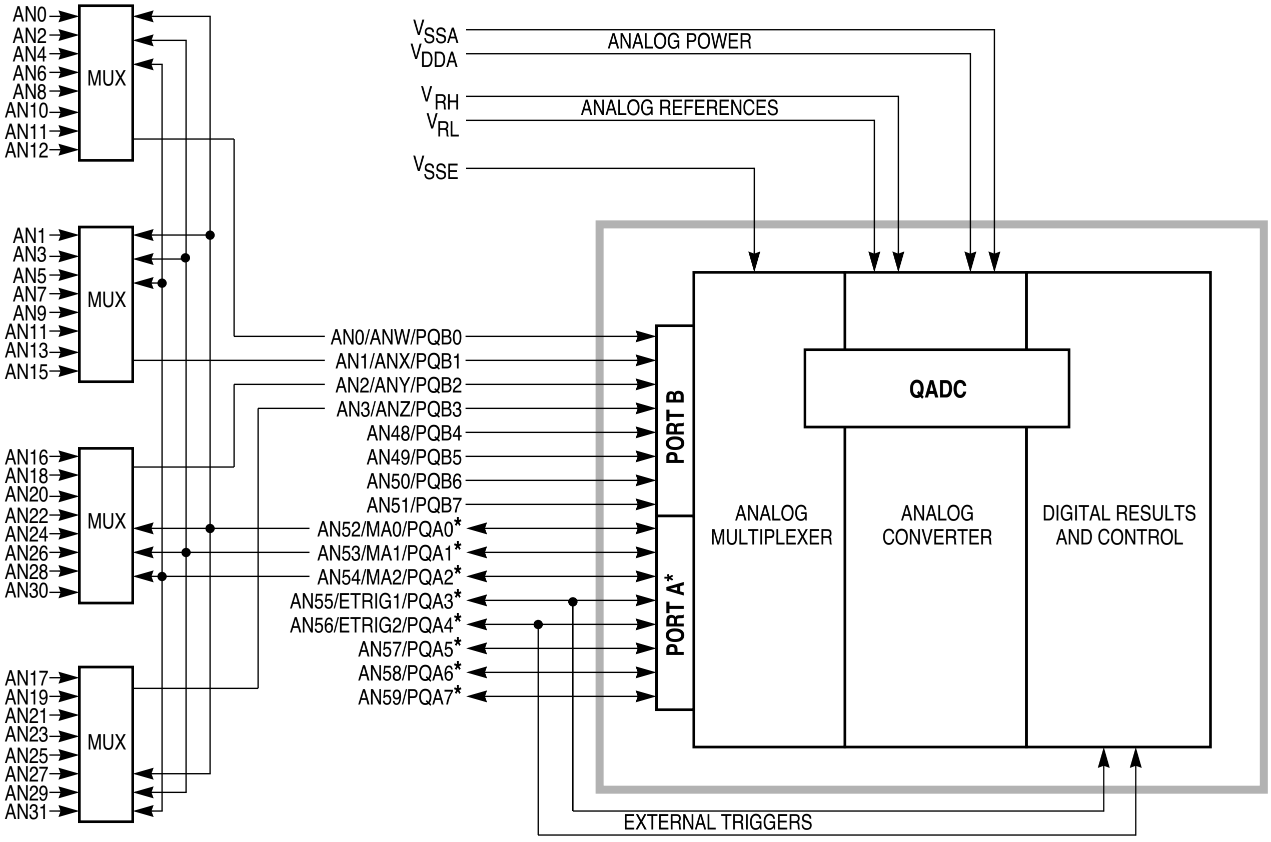
For example, four 8-input multiplexer chips can be put at the connector where the analog signals first arrive on the computer board. As a result, only four analog signals need to be shielded from noise as they approach the microcontroller chip, rather than having to protect 32 analog signals. However, external multiplexer chips may introduce additional noise and errors if not properly utilized. Therefore, it is necessary to maintain low “on” resistance (the impedance of an analog switch when active within a multiplex chip) and insert a low pass filter (R/C) on the input side of the multiplex chip.

4.1 External Multiplexing Operation

The QADC can use from one to four external multiplexer chips to expand the number of analog signals that may be converted. Up to 32 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are automatically selected from the channel field of the conversion command word (CCW) table, the same as internally multiplexed channels. Refer to paragraph 7.7 in **SECTION 7 DIGITAL CONTROL** for additional information on channel number assignments.

All of the automatic queue features are available for externally and internally multiplexed channels. The software selects the external multiplexed mode by setting the MUX bit in control register 0 (QACR0).

Figure 4-1 shows the maximum configuration of four external multiplexer chips connected to the QADC. The external multiplexer chips select one of eight analog inputs and connect it to one analog output, which becomes an input to the QADC. The QADC provides three multiplexed address signals — MA0, MA1, and MA2, to select one of eight inputs. These inputs are connected to all four external multiplexer chips. The analog output of the four multiplex chips are each connected to four separate QADC inputs — AN_W, AN_X, AN_Y, and AN_Z.



*** PORT A PINS INCORPORATE OPEN DRAIN PULL DOWN DRIVERS.**

Figure 4-1 Example of External Multiplexing

When the external multiplexed mode is selected, the QADC automatically creates the MA open drain output signals from the channel number in each CCW. The QADC also converts the proper input channel (AN_W, AN_X, AN_Y, and AN_Z) by interpreting the CCW channel number. As a result, up to 32 externally multiplexed channels appear to the conversion queues as directly connected signals. The software simply puts the channel number of externally multiplexed channels into CCWs. Refer to **Table 7-9** in **SECTION 7 DIGITAL CONTROL** which shows the channel numbers for the externally multiplexed channels that are assigned the range of channel 0 to channel 31.

Figure 4-1 shows that the three MA signals may also be analog input pins. When external multiplexing is selected, none of the MA pins can be used for analog or digital inputs. They become multiplexed address outputs.

4.2 Module Version Options

The number of available analog channels varies, depending on whether or not external multiplexing is used. A maximum of 16 analog channels are supported by the internal multiplexing circuitry of the converter. **Table 4-1** shows the total number of analog input channels supported with zero to four external multiplexer chips. The QADC uses 21 pins which allow a maximum of 41 analog channels to be converted.

Table 4-1 Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels[1, 2]				
No External Mux Chips	**One External Mux Chip**	**Two External Mux Chips**	**Three External Mux Chips**	**Four External Mux Chips**
16	12 + 8 = 20	11 + 16 = 27	10 + 24 = 34	9 + 32 = 41

NOTES:

1. The above assumes that the external trigger inputs are shared with two analog input pins.
2. When external multiplexing is used, three input channels become multiplexed address outputs, and for each external multiplexer chip, one input channel becomes a multiplexed analog input.

SECTION 5 PIN CONNECTION CONSIDERATIONS

The QADC requires accurate, noise-free input signals for proper operation. This section discusses the design of external circuitry to maximize QADC performance.

5.1 Analog Reference Pins

No A/D converter can be more accurate than its analog reference. Any noise in the reference can result in at least that much error in a conversion. The reference for the QADC, supplied by pins V_{RH} and V_{RL}, should be low-pass filtered from its source to obtain a noise-free, clean signal. In many cases, simple capacitive bypassing may suffice. In extreme cases, inductors or ferrite beads may be necessary if noise or RF energy is present. Series resistance is not advisable since there is an effective DC current requirement from the reference voltage by the internal resistor string in the RC DAC array. External resistance may introduce error in this architecture under certain conditions. Any series devices in the filter network should contain a minimum amount of DC resistance.

For accurate conversion results, the analog reference voltages must be within the limits defined by V_{DDA} and V_{SSA}, as explained in the following subsection.

5.2 Analog Power Pins

The analog supply pins (V_{DDA} and V_{SSA}) define the limits of the analog reference voltages (V_{RH} and V_{RL}) and of the analog multiplexer inputs. **Figure 5-1** is a diagram of the analog input circuitry.

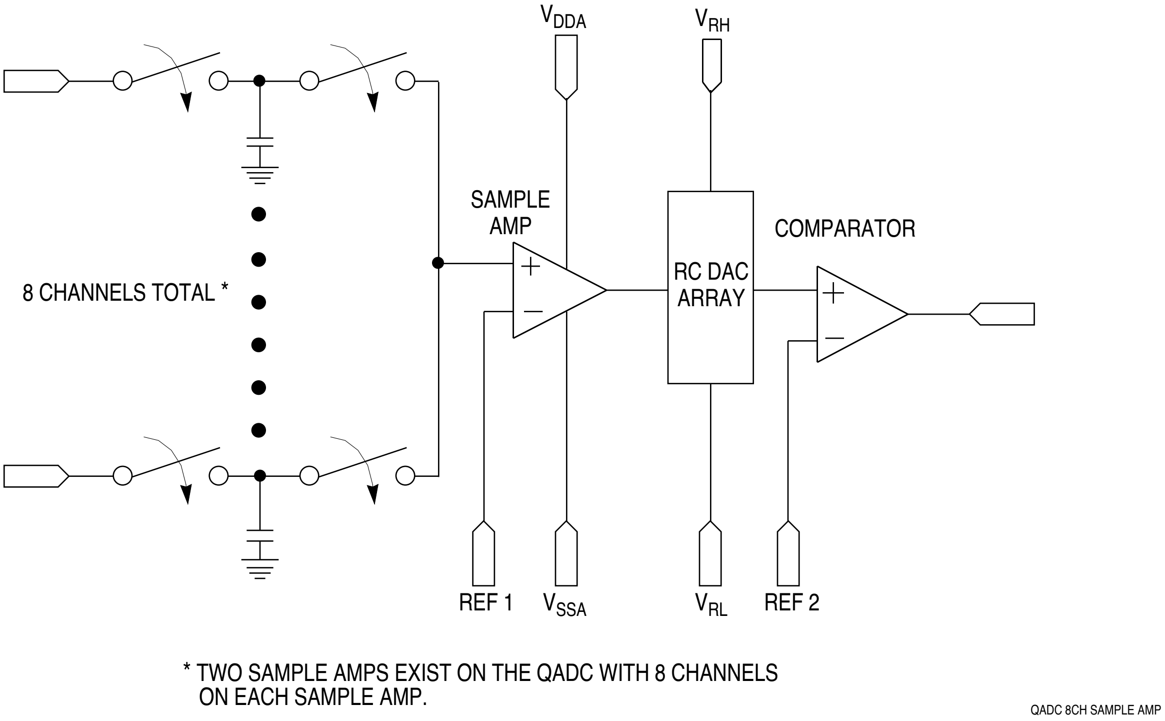


Figure 5-1 Analog Input Circuitry

Since the sample amplifier is powered by V_{DDA} and V_{SSA}, it can accurately transfer input signal levels up to but not exceeding V_{DDA} and down to but not below V_{SSA}. If the input signal is outside of this range, the output from the sample amplifier is clipped.

In addition, V_{RH} and V_{RL} must be within the range defined by V_{DDA} and V_{SSA}. As long as V_{RH} is less than or equal to V_{DDA} and V_{RL} is greater than or equal to V_{SSA} and the sample amplifier has accurately transferred the input signal, resolution is ratiometric within the limits defined by V_{RL} and V_{RH}. If V_{RH} is greater than V_{DDA}, the sample amplifier can never transfer a full-scale value. If V_{RL} is less than V_{SSA}, the sample amplifier can never transfer a zero value.

Figure 5-2 shows the results of reference voltages outside the range defined by V_{DDA} and V_{SSA}. At the top of the input signal range, V_{DDA} is 10 mV lower than V_{RH}. This results in a maximum obtainable 10-bit conversion value of 3FE. At the bottom of the signal range, V_{SSA} is 15 mV higher than V_{RL}, resulting in a minimum obtainable 10-bit conversion value of three.

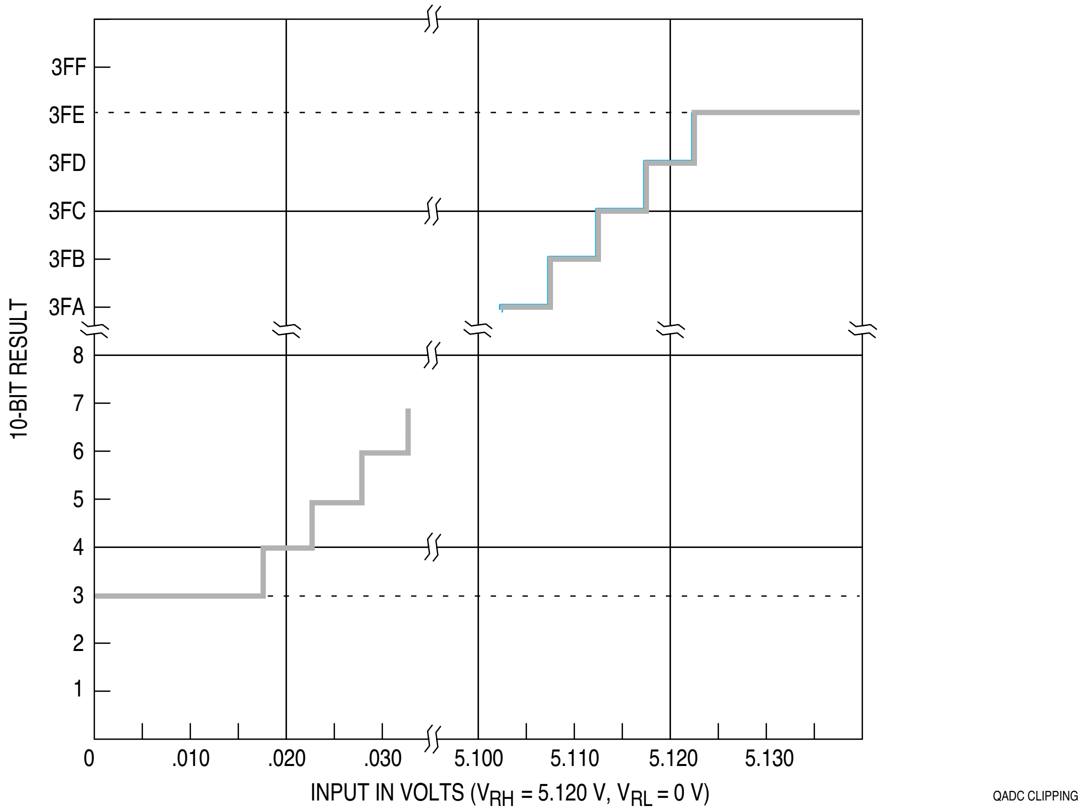


Figure 5-2 Errors Resulting from Clipping

5.3 Analog Supply Filtering and Grounding

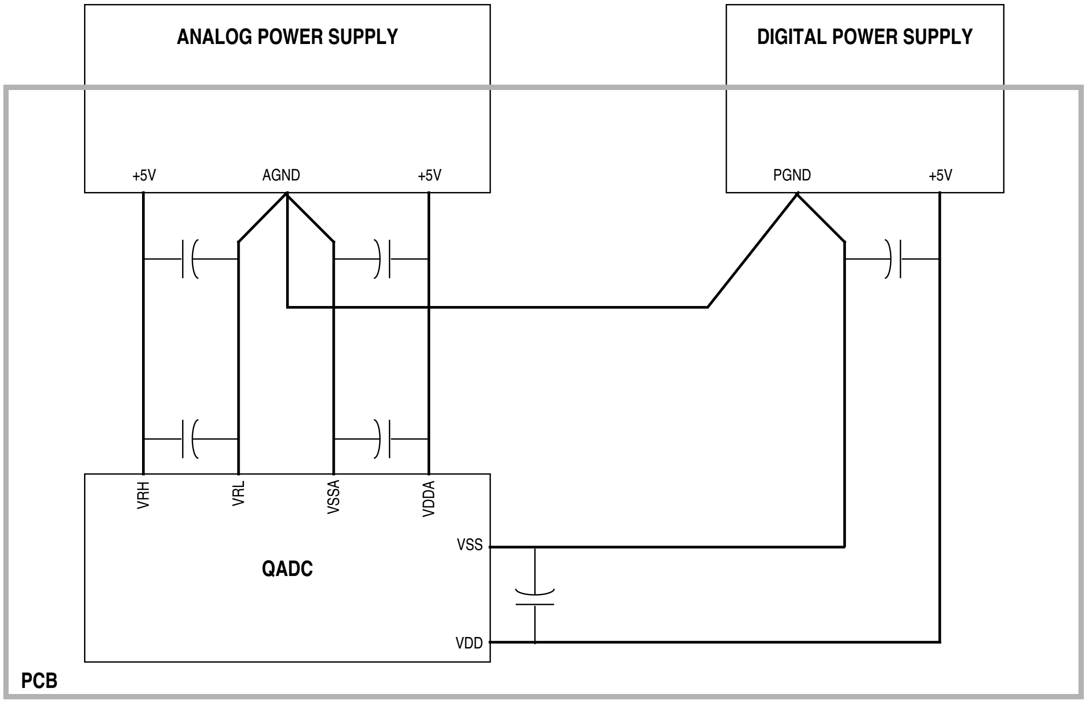
Two important factors influencing performance in analog integrated circuits are supply filtering and grounding. Generally, digital circuits use bypass capacitors on every V_{DD}/ V_{SS} pin pair. This applies to analog subsystems or submodules also. Equally important as bypassing is the distribution of power and ground.

Analog supplies should be isolated from digital supplies as much as possible. This necessity stems from the higher performance requirements often associated with analog circuits. Therefore, deriving an analog supply from a local digital supply is not recommended. However, if for economic reasons digital and analog power are derived from a common regulator, filtering of the analog power is recommended in addition to the bypassing of the supplies already mentioned. For example, an RC low pass filter could be used to isolate the digital and analog supplies when generated by a common regulator. If multiple high precision analog circuits are locally employed (i.e. two A/D converters), the analog supplies should be isolated from each other as sharing supplies introduces the potential for interference between analog circuits.

Grounding is the most important factor influencing analog circuit performance in mixed signal systems (or in standalone analog systems). Close attention must be paid not to introduce additional sources of noise into the analog circuitry. Common sources of noise include ground loops, inductive coupling, and combining digital and analog grounds together inappropriately.

The problem of how and when to combine digital and analog grounds arises from the large transients which the digital ground must handle. If the digital ground is not able to handle the large transients, the current from the large transients can return to ground through the analog ground. It is the excess current overflowing into the analog ground which causes performance degradation by developing a differential voltage between the true analog ground and the microcontroller's ground pin. The end result is that the ground observed by the analog circuit is no longer true ground and often ends in skewed results.

Two similar approaches designed to improve or eliminate the problems associated with grounding excess transient currents involve star-point ground systems. One approach is to star-point the different grounds at the power supply origin, thus keeping the ground isolated. Refer to **Figure 5-3**.



QADC POWER SCHEM

Figure 5-3 Star-Ground at the Point of Power Supply Origin

Another approach is to star-point the different grounds near the analog ground pin on the microcontroller by using small traces for connecting the non-analog grounds to the analog ground. The small traces are meant only to accommodate DC differences, not AC transients.

NOTE

This star-point scheme still requires adequate grounding for digital and analog subsystems in addition to the star-point ground.

Other suggestions for PCB layout in which the QADC is employed include:

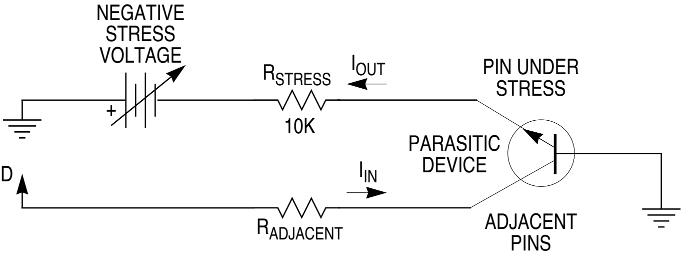
- Analog ground must be low impedance to all analog ground points in the circuit.
- Bypass capacitors should be as close to the power pins as possible.
- The analog ground should be isolated from the digital ground. This can be done by cutting a separate ground plane for the analog ground.
- Non-minimum traces should be utilized for connecting bypass capacitors and filters to their corresponding ground/power points.
- Minimum distance for trace runs when possible.

5.4 Accommodating Positive/Negative Stress Conditions

Positive or negative stress refers to conditions which exceed nominally defined operating limits. Examples include applying a voltage exceeding the normal limit on an input (for example, voltages outside of the suggested supply/reference ranges) or causing currents into or out of the pin which exceed normal limits. QADC specific considerations are voltages greater than V_{DDA}, V_{RH} or less than V_{SSA} applied to an analog input which cause excessive currents into or out of the input. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information on exact magnitudes.

Both stress conditions can potentially disrupt conversion results on neighboring inputs. Parasitic devices, associated with CMOS processes, can cause an immediate disruptive influence on neighboring pins. Common examples of parasitic devices are diodes to substrate and bipolar devices with the base terminal tied to substrate (V_{SSI}/V_{SSA} ground). Under stress conditions, current introduced on an adjacent pin can cause errors on adjacent channels by developing a voltage drop across the adjacent external channel source impedances.

Figure 5-4 shows an active parasitic bipolar when an input pin is subjected to negative stress conditions. Positive stress conditions do not activate a similar parasitic device.



QADC PAR STRESS CONN

Figure 5-4 Input Pin Subjected to Negative Stress

The current out of the pin (I_{OUT}) under negative stress is determined by the following equation:

$$I_{OUT} = \frac{|V_{STRESS} - V_{BE}|}{R_{STRESS}}$$

where:

V_{STRESS} = Adjustable voltage source

V_{BE} = Parasitic bipolar base/emitter voltage (refer to $V_{NEGCLAMP}$ in **APPENDIX A ELECTRICAL CHARACTERISTICS**)

R_{STRESS} = Source impedance (10K resistor in **Figure 5-4** on stressed channel)

The current into (I_{IN}) the neighboring pin is determined by the $1/K_N$ (Gain) of the parasitic bipolar transistor ($1/K_N \ll 1$).

One way to minimize the impact of stress conditions on the QADC is to apply voltage limiting circuits such as diodes to supply and ground. However, leakage from such circuits and the potential influence on the sampled voltage to be converted must be considered. Refer to **Figure 5-5**.

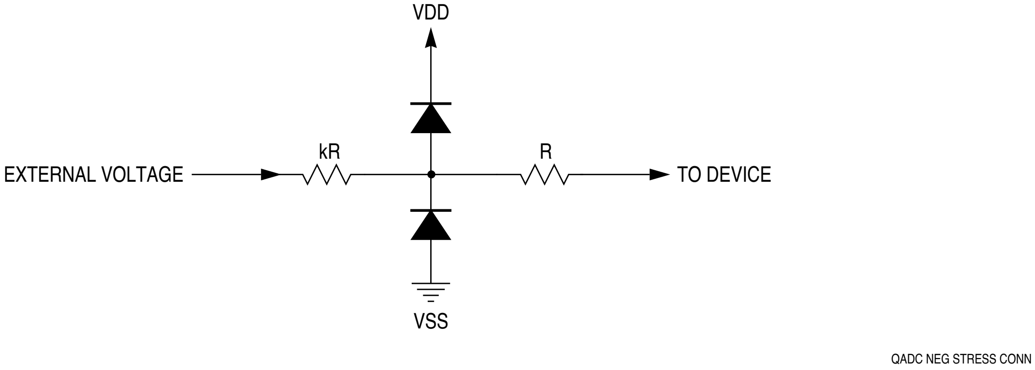


Figure 5-5 Voltage Limiting Diodes in a Negative Stress Circuit

Another method for minimizing the impact of stress conditions on the QADC is to strategically allocate QADC inputs so that the lower accuracy inputs are adjacent to the inputs most likely to see stress conditions.

Finally, suitable source impedances should be selected to meet design goals and minimize the effect of stress conditions.

5.5 Analog Input Considerations

The source impedance of the analog signal to be measured and any intermediate filtering should be considered whether external multiplexing is used or not. **Figure 5-6** shows the connection of eight typical analog signal sources to one QADC analog input pin through a separate multiplexer chip. Also, an example of an analog signal source connected directly to a QADC analog input channel is displayed.

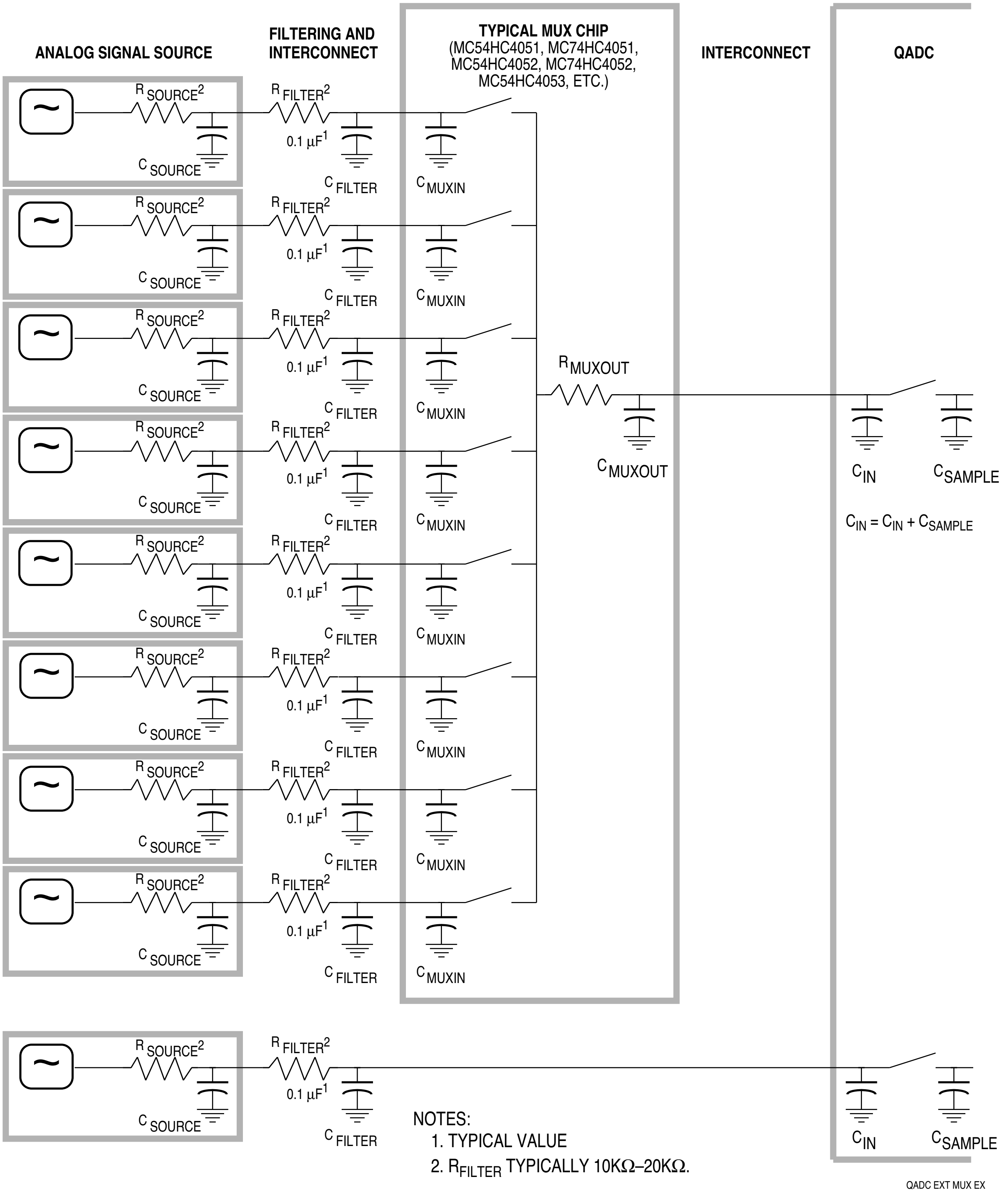


Figure 5-6 External Multiplexing Of Analog Signal Sources

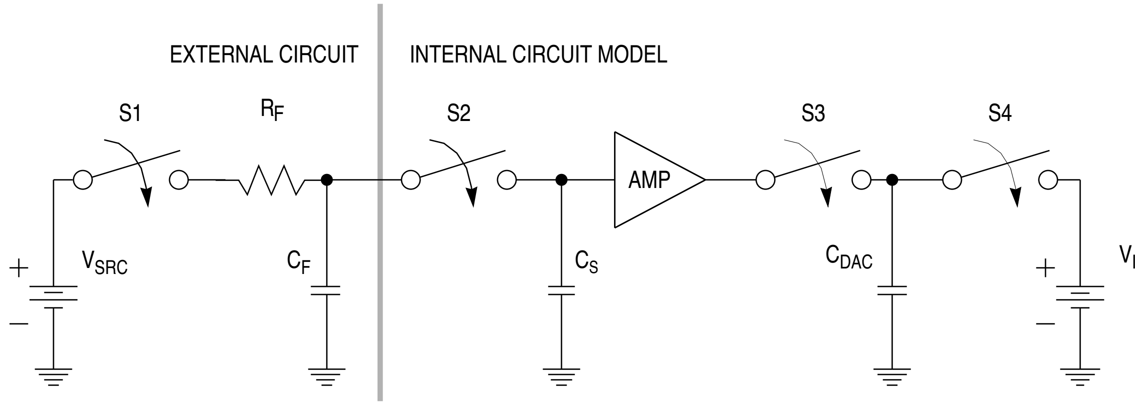
5.6 Analog Input Pins

Analog inputs should have low AC impedance at the pins. Low AC impedance can be realized by placing a capacitor with good high frequency characteristics at the input pin of the part. Ideally, that capacitor should be as large as possible (within the practical range of capacitors that still have good high frequency characteristics). This capacitor has two effects:

- It helps attenuate any noise that may exist on the input.
- It sources charge during the sample period when the analog signal source is a high-impedance source.

Series resistance can be used with the capacitor on an input pin to implement a simple RC filter. The maximum level of filtering at the input pins is application dependent and is based on the bandpass characteristics required to accurately track the dynamic characteristics of an input. Simple RC filtering at the pin may be limited by the source impedance of the transducer or circuit supplying the analog signal to be measured. (refer to **5.6.2 Error Resulting from Leakage**). In some cases, the size of the capacitor at the pin may be very small.

Figure 5-7 is a simplified model of an input channel. Refer to this model in the following discussion of the interaction between the external circuitry and the circuitry inside the QADC.



V_{SRC} = SOURCE VOLTAGE
R_F = FILTER IMPEDANCE (SOURCE IMPEDANCE INCLUDED)
C_F = FILTER CAPACITOR
C_S = INTERNAL CAPACITANCE (FOR A BYPASSED CHANNEL, THIS IS THE C_{DAC} CAPACITANCE)
C_{DAC} = DAC CAPACITOR ARRAY
V_I = INTERNAL VOLTAGE SOURCE FOR PRECHARGE (V_{DDA}/2)

QADC SAMPLE AMP MODEL

Figure 5-7 Electrical Model of an A/D Input Pin

In **Figure 5-7**, R_F and C_F comprise the external filter circuit. C_S is the internal sample capacitor. Each channel has its own capacitor. C_S is never precharged; it retains the value of the last sample. V_I is an internal voltage source used to precharge the DAC capacitor array (C_{DAC}) before each sample. The value of this supply is $V_{DDA}/2$, or 2.5 volts for 5-volt operation.

The following paragraphs provide a simplified description of the interaction between the QADC and the user's external circuitry. This circuitry is assumed to be a simple RC low-pass filter passing a signal from a source to the QADC input pin. The following simplifying assumptions are made:

- The source impedance is included with the series resistor of the RC filter.
- The external capacitor is perfect (no leakage, no significant dielectric absorption characteristics, etc.)
- All parasitic capacitance associated with the input pin is included in the value of the external capacitor.
- Inductance is ignored.
- The "on" resistance of the internal switches is zero ohms and the "off" resistance is infinite.

5.6.1 Settling Time for the External Circuit

The values for R_F and C_F in the user's external circuitry determine the length of time required to charge C_F to the source voltage level (V_{SRC}). At time t = 0, S1 in **Figure 5-7** closes. S2 is open, disconnecting the internal circuitry from the external circuitry. Assume that the initial voltage across C_F is zero. As C_F charges, the voltage across it is determined by the following equation, where t is the total charge time:

$$V_{CF} = V_{SRC}(1 - e^{-t/R_F C_F})$$

When t = 0, the voltage across $C_F = 0$. As t approaches infinity, V_{CF} will equal V_{SRC}. (This assumes no internal leakage.) With 10-bit resolution, 1/2 of a count is equal to 1/2048 full-scale value. Assuming worst case (V_{SRC} = full scale), **Table 5-1** shows the required time for C_F to charge to within 1/2 of a count of the actual source voltage during 10-bit conversions. **Table 5-1** is based on the RC network in **Figure 5-7**.

NOTE

The following times are completely independent of the A/D converter architecture (assuming the QADC is not affecting the charging).

Table 5-1 External Circuit Settling Time (10-Bit Conversions)

Filter Capacitor (CF)	**Source Resistance (R_F)**			
	100 Ω	**1 kΩ**	**10 kΩ**	**100 kΩ**
1 µF	760 µs	7.6 ms	76 ms	760 ms
.1 µF	76 µs	760 µs	7.6 ms	76 ms
.01 µF	7.6 µs	76 µs	760 µs	7.6 ms
.001 µF	760 ns	7.6 µs	76 µs	760 µs
100 pF	76 ns	760 ns	7.6 µs	76 µs

The external circuit described in **Table 5-1** is a low-pass filter. A user interested in measuring an AC component of the external signal must take the characteristics of this filter into account.

5.6.2 Error Resulting from Leakage

A series resistor limits the current to a pin, therefore input leakage acting through a large source impedance can degrade A/D accuracy. The maximum input leakage current is specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Input leakage is greatest at high operating temperatures and as a general rule decreases by one half for each 10° C decrease in temperature.

Assuming $V_{RH} – V_{RL}$ = 5.12 V, 1 count (assuming 10-bit resolution) corresponds to 5 mV of input voltage. A typical input leakage of 50 nA acting through 100 kΩ of external series resistance results in an error of less than 1 count (5.0 mV). If the source impedance is 1 MΩ and a typical leakage of 50 nA is present, an error of 10 counts (50 mV) is introduced.

In addition to internal junction leakage, external leakage (e.g., if external clamping diodes are used) and charge sharing effects with internal capacitors also contribute to the total leakage current. **Table 5-2** illustrates the effect of different levels of total leakage on accuracy for different values of source impedance. The error is listed in terms of 10-bit counts.

CAUTION

Leakage from the part of 10 nA is obtainable only within a limited temperature range.

Table 5-2 Error Resulting From Input Leakage (IOFF)

Source Impedance	Leakage Value (10-bit Conversions)			
	10 nA	**50 nA**	**100 nA**	**1000 nA**
1 kΩ	—	—	—	0.2 counts
10 kΩ	—	0.1 counts	0.2 counts	2 counts
100 kΩ	0.2 counts	1 count	2 counts	20 counts

SECTION 6 ANALOG SUBSYSTEM

This section describes the QADC analog subsystem, which includes the front-end analog multiplexer, digital to analog converter (DAC) array, the comparator, and the successive approximation register (SAR).

6.1 Analog-to-Digital Converter Operation

The analog subsystem consists of the path from the input pins through the input multiplexing circuitry, into the DAC array, and through the analog comparator. The output of the comparator feeds into the SAR and is considered the boundary between the analog and digital subsystems of the QADC. **Figure 6-1** shows a block diagram of the QADC analog submodule.

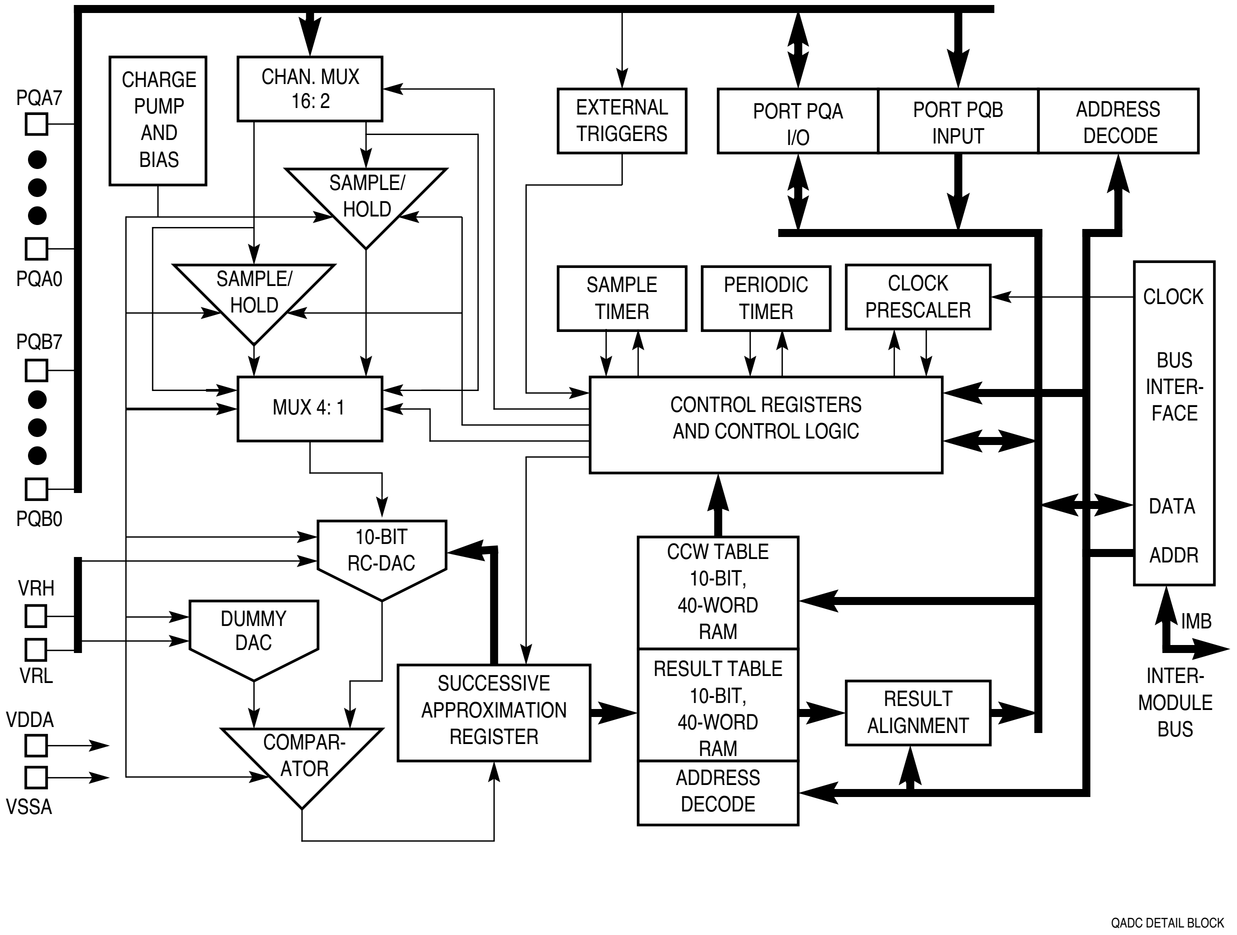


Figure 6-1 QADC Module Block Diagram

6.1.1 Conversion Cycle Times

Total conversion time is made up of initial sample time, transfer time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is connected to the sample capacitor at the input of the sample buffer amplifier. During the transfer period, the sample capacitor is disconnected from the multiplexer, and the stored voltage is buffered and transferred to the RC DAC array. During the final sampling period, the sample capacitor and amplifier are bypassed, and the multiplexer input charges the RC DAC array directly. During the resolution period, the voltage in the RC DAC array is converted to a digital value and stored in the SAR.

Initial sample time is fixed at two QCLKs and the transfer time at four QCLKs. Final sample time can be 2, 4, 8, or 16 ADC clock cycles, depending on the value of the IST field in the CCW. Resolution time is ten cycles.

Transfer and resolution require a minimum of 18 QCLK clocks (8.6 µs with a 2.1-MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 15.2 µs (with a 2.1-MHz QCLK).

Figure 6-2 illustrates the timing for conversions. This diagram assumes a final sampling period of two QCLKs.

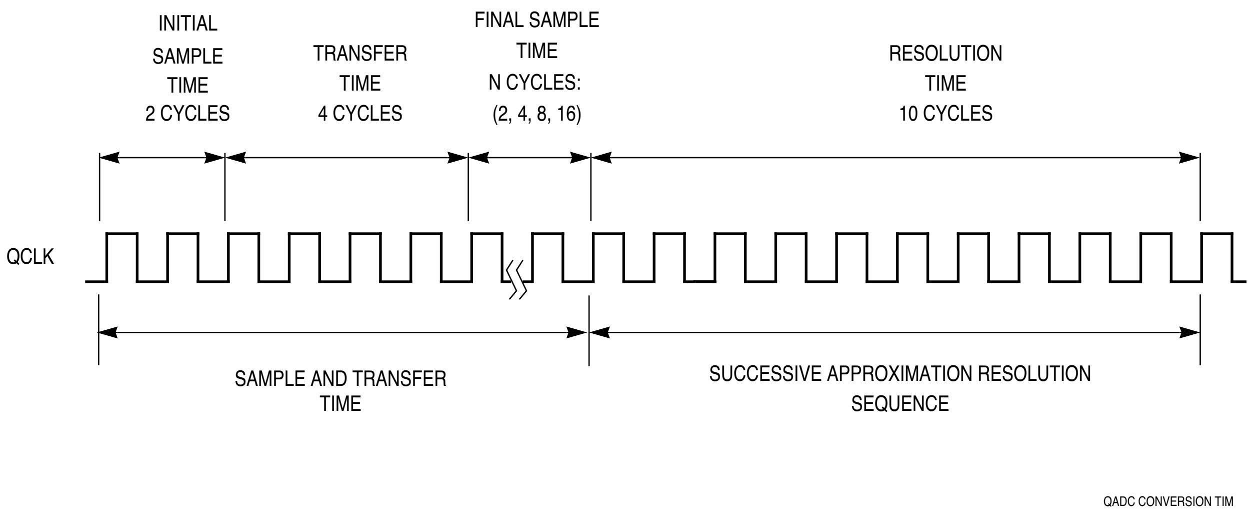


Figure 6-2 Conversion Timing

6.1.1.1 Amplifier Bypass Mode Conversion Timing

If the amplifier bypass mode is enabled for a conversion by setting the amplifier bypass (BYP) field in the CCW, the timing changes to that shown in **Figure 6-3**. Refer to paragraph 7.7 in **SECTION 7 DIGITAL CONTROL** for more information on the BYP field. The initial sample time and the transfer time are eliminated reducing the potential conversion time by six QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKs must be allowed. This results in a savings of four QCLKs. When using the bypass mode, the external circuit should be of low source impedance (typically less than 10 kΩ). Also, the loading effects of the external circuitry by the QADC need to be considered, since the benefits of the sample amplifier are not present.

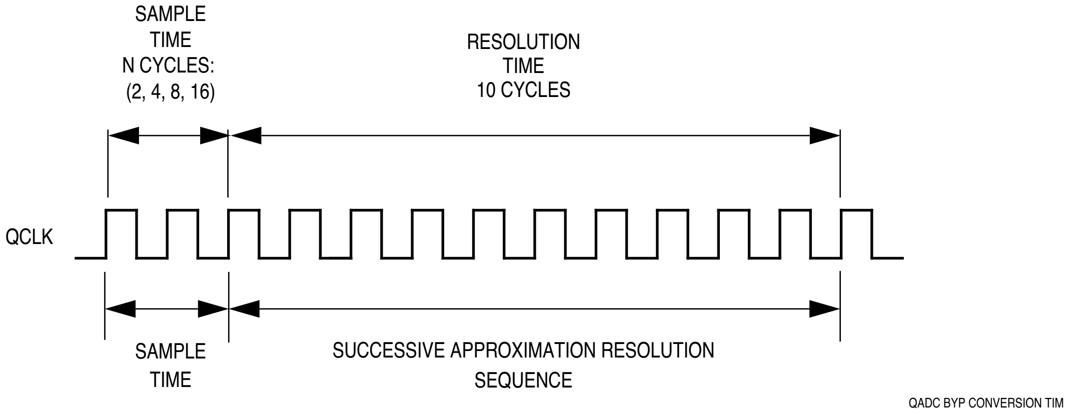


Figure 6-3 Bypass Mode Conversion Timing

6.2 Front-End Analog Multiplexer

The internal multiplexer selects one of the 16 analog input pins or one of three special internal reference channels for conversion. The following are the three special channels:

- V_{RH} — Reference Voltage High
- V_{RL} — Reference Voltage Low
- $V_{DDA}/2$ — Mid-Analog Supply Voltage

The selected input is connected to one side of the DAC capacitor array. The other side of the DAC array is connected to the comparator input. The multiplexer also includes positive and negative stress protection circuitry, which prevents other channels from affecting the current conversion when voltage levels are applied to the other channels. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for specific voltage level limits.

6.3 Digital to Analog Converter Array

The digital to analog converter (DAC) array consists of binary-weighted capacitors and a resistor-divider chain. The array serves two purposes:

- The array holds the sampled input voltage during conversion.
- The resistor-capacitor array provides the mechanism for the successive approximation A/D conversion.
- Resolution begins with the most significant bit (MSB) and works down to the least significant bit (LSB). The switching sequence is controlled by the digital logic.

6.4 Comparator

The comparator is used during the approximation process to sense whether the digitally selected arrangement of the DAC array produces a voltage level higher or lower than the sampled input. The comparator output feeds into the SAR which accumulates the A/D conversion result sequentially, starting with the MSB.

6.5 Successive Approximation Register

The input of the successive approximation register (SAR) is connected to the comparator output. The SAR sequentially receives the conversion value one bit at a time, starting with the MSB. After accumulating the 10 bits of the conversion result, the SAR data is transferred to the appropriate result location, where it may be read from the IMB by user software.

SECTION 7 DIGITAL CONTROL

The digital control subsystem includes the control logic to sequence the conversion activity, channel select logic, the clock and periodic/interval timer, control and status registers, the conversion command word table RAM, and the result word table RAM.

The central element for control of the QADC conversions is the 40-entry conversion command word (CCW) table. Each CCW specifies the conversion of one input channel. Depending on the application, one or two queues can be established in the CCW table. A queue is a scan sequence of one or more input channels. By using a pause mechanism, subqueues can be created in the two queues. Each queue can be operated using several different scan modes. The scan modes for queue 1 and queue 2 are programmed in QACR1 and QACR2 (control registers 1 and 2). Once a queue has been started by a trigger event (any of the ways to cause the QADC to begin executing the CCWs in a queue or subqueue), the QADC performs a sequence of conversions and places the results in the result word table.

7.1 Queue Priority

Queue 1 has priority over queue 2 execution. The following cases show the conditions under which queue 1 asserts its priority:

- When a queue is not active, a trigger event for queue 1 or queue 2 causes the corresponding queue execution to begin.
- When queue 1 is active and a trigger event occurs for queue 2, queue 2 cannot begin execution until queue 1 reaches completion or the paused state. The status register records the trigger event by reporting the queue 2 status as trigger pending. Additional trigger events for queue 2, which occur before execution can begin, are recorded as trigger overruns.
- When queue 2 is active and a trigger event occurs for queue 1, the current queue 2 conversion is aborted. The status register reports the queue 2 status as suspended. Any trigger events occurring for queue 2 while queue 2 is suspended are recorded as trigger overruns. Once queue 1 reaches the completion or the paused state, queue 2 begins executing again. The programming of the resume bit in QACR2 determines which CCW is executed in queue 2. Refer to **7.6.3 Control Register 2** for more information.
- When simultaneous trigger events occur for queue 1 and queue 2, queue 1 begins execution and the queue 2 status is changed to trigger pending.
- Subqueues that are paused.

The pause feature can be used to divide queue 1 and/or queue 2 into multiple subqueues. A subqueue is defined by setting the pause bit in the last CCW of the subqueue.

Figure 7-1 shows the CCW format and an example of using pause to create subqueues. Queue 1 is shown with four CCWs in each subqueue and queue 2 has two CCWs in each subqueue.

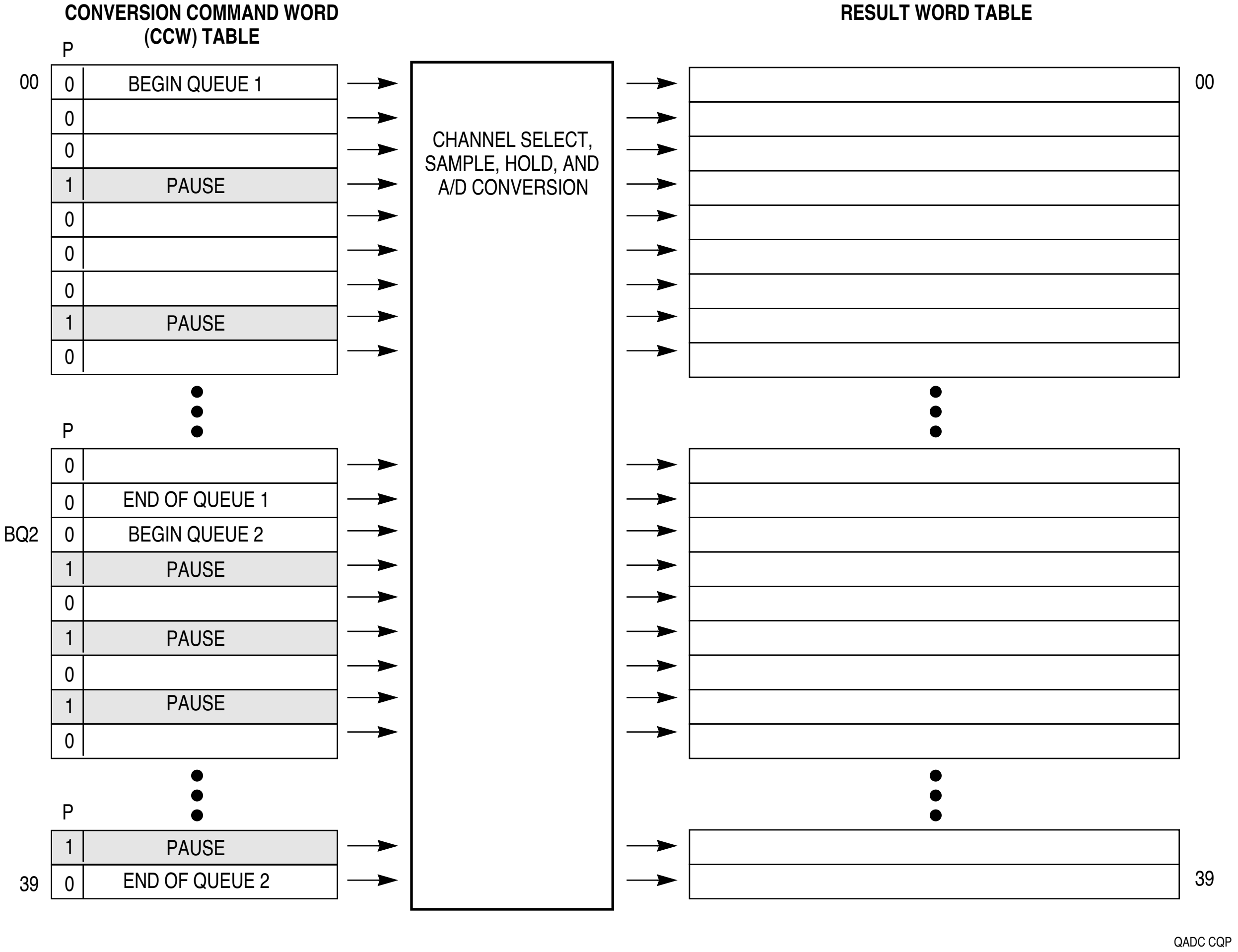


Figure 7-1 QADC Queue Operation With Pause

The queue operating mode selected for queue 1 determines what type of trigger event causes the execution of each of the subqueues within queue 1. Similarly, the queue operating mode for queue 2 determines the type of trigger event required to execute each of the subqueues within queue 2.

For example, when the external trigger rising edge continuous-scan mode is selected for queue 1, and there are six subqueues within queue 1, a separate rising edge is required on the external trigger pin after every pause to begin the execution of each subqueue (refer to **Figure 7-1**). Refer to **7.3 Scan Modes** for more information on different scan modes.

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each subqueue. Once a subqueue is initiated, each CCW is executed sequentially until the last CCW in the subqueue is executed and the pause state is entered.

Execution can only continue with the next CCW, which is the beginning of the next subqueue. A subqueue cannot be executed a second time before the overall queue execution has been completed. Refer to **7.6.3 Control Register 2** for more information.

Trigger events which occur during the execution of a subqueue are ignored, except that the trigger overrun flag is set. When a continuous-scan mode is selected, a trigger event occurring after the completion of the last subqueue (after the queue completion flag is set), causes the execution to continue with the first subqueue, starting with the first CCW in the queue.

When the QADC encounters a CCW with the pause bit set, the queue enters the paused state after completing the conversion specified in the CCW with the pause bit. The pause flag is set and a pause software interrupt may optionally be issued. The status of the queue is shown to be paused, indicating completion of a subqueue. The QADC then waits for another trigger event to again begin execution of the next subqueue.

7.2 Boundary Conditions

The following are queue operation boundary conditions:

- The first CCW in a queue contains channel 63, the end-of-queue (EOQ) code. The queue becomes active and the first CCW is read. The end-of-queue is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set beyond the end of the CCW table (40–63) and a trigger event occurs on queue 2. Refer to **7.6.3 Control Register 2** for information on BQ2. The end-of-queue condition is recognized immediately, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 is set to CCW0 and a trigger event occurs on queue 1. After reading CCW0, the end-of-queue condition is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.

NOTE

Multiple end-of-queue conditions may be recognized simultaneously, although there is no change in the QADC behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC reads CCW0 and detects both end-of-queue conditions. The completion flag is set and queue 1 becomes idle.

Boundary conditions also exist for combinations of pause and end-of-queue. One case is when a pause bit is in one CCW and an end-of-queue condition is in the next CCW. The conversion specified by the CCW with the pause bit set completes normally. The pause flag is set. However, since the end-of-queue condition is recognized, the completion flag is also set and the queue status becomes idle, not paused. Examples of this situation include:

- The pause bit is set in CCW5 and the channel 63 (EOQ) code is in CCW6.
- The pause is in CCW39.
- During queue 1 operation, the pause bit is set in CCW20 and BQ2 points to CCW21.

Another pause and end-of-queue boundary condition occurs when the pause and an end-of-queue condition occur in the same CCW. Both the pause and end-of-queue conditions are recognized simultaneously. The end-of-queue condition has precedence so a conversion is not performed for the CCW and the pause flag is not set. The QADC sets the completion flag and the queue status becomes idle. Examples of this situation are:

- The pause bit is set in CCW10 and EOQ is programmed into CCW10.
- During queue 1 operation, the pause bit set in CCW32, which is also BQ2.

7.3 Scan Modes

The QADC queuing mechanism allows the application to utilize different requirements for automatically scanning input channels.

In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed.

In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue is executed. The following paragraphs describe single-scan and continuous-scan operations.

7.3.1 Disabled Mode and Reserved Mode

When the disabled mode or a reserved mode is selected, the queue is not active. Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, wait states are not encountered for IMB accesses of the RAM. When both queues are disabled, it is safe to change the prescaler values.

7.3.2 Single-Scan Modes

When the application software wants to execute a single pass through a sequence of conversions defined by a queue, a single-scan queue operating mode is selected. By programming the MQ1 field in QACR1, the following modes can be selected for queue 1:

- Software initiated single-scan mode
- External trigger rising edge single-scan mode
- External trigger falling edge single-scan mode

In addition to the above modes, queue 2 can also be programmed for the interval timer single-scan mode, using the periodic/interval timer. The queue operating mode for queue 2 is selected by the MQ2 field in QACR2.

In all single-scan queue operating modes, the software must also enable the queue to begin execution by writing the single-scan enable bit to a one in the queue’s control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2 respectively.

Until the single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to a one during the write cycle, which selects the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The completion flag, completion interrupt, or queue status are used to determine when the queue has completed.

After the single-scan enable bit is set, a trigger event causes the QADC to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue is completed. After the queue reaches completion, the QADC resets the single-scan enable bit to zero. If the single-scan enable bit is written to a one or a zero by the software before the queue scan is complete, the queue is not affected. However, if the software changes the queue operating mode, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The current conversion is aborted and the new queue operating mode takes effect.

In the software initiated single-scan mode, the writing of a one to the single-scan enable bit causes the QADC to internally generate a trigger event and the queue execution begins immediately. In the other single-scan queue operating modes, once the single-scan enable bit is written, the selected trigger event must occur before the queue can start. A trigger overrun is recorded if a trigger event occurs during queue execution in the external trigger single-scan mode and the interval timer single-scan mode.

When a pause bit is encountered during the queue scan in a single-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state. The single-scan enable bit allows the entire queue to be scanned once.

In the software initiated single-scan mode, the trigger event is generated internally as soon as the conversion is complete for the CCW with the pause bit set. The pause is two QCLKs. In the external trigger single-scan mode, the queue remains paused until another trigger edge is received on the external trigger pin.

In the interval timer single-scan mode, the next expiration of the timer is the trigger event for the queue. After the queue execution is complete, the queue status is shown as idle. The software can restart the queue by setting the single-scan enable bit to a one. Queue execution begins with the first CCW in the queue.

7.3.2.1 Software Initiated Single-Scan Mode

Software can initiate the execution of a scan sequence for queue 1 or 2 by selecting the software initiated single-scan mode, and writing the single-scan enable bit in QACR1 or QACR2. A trigger event is generated internally and the QADC immediately begins execution of the first CCW in the queue. If a pause is encountered, queue execution ceases momentarily while another trigger event is generated internally, and then execution continues.

The QADC automatically performs the conversions in the queue until an end-of-queue condition is encountered. The queue remains idle until the software again sets the single-scan enable bit. While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is paused. The trigger overrun flag is never set while in the software initiated single-scan mode.

The software initiated single-scan mode is useful in the following applications:

- Allows software complete control of the queue execution.
- Allows the software to easily alternate between several queue sequences.

7.3.2.2 External Trigger Single-Scan Mode

The external trigger single-scan mode is a variation of the external trigger continuous-scan mode, and is also available with both queue 1 and queue 2. The software programs the polarity of the external trigger edge that is to be detected, either a rising or a falling edge. The software must enable the scan to occur by setting the single-scan enable bit for the queue.

The first external trigger edge causes the queue to be executed one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. After the queue is completed, the QADC clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of the queue to be initiated by the next external trigger edge.

The external trigger single-scan mode is useful when the input trigger rate can exceed the queue execution rate. Analog samples can be taken in sync with an external event, even though the software is not interested in data taken from every edge. The software can start the external trigger single-scan mode and get one set of data, and at a later time, start the queue again for the next set of samples.

The external trigger single-scan mode is also useful when the software needs to change the polarity of the external trigger so that both the rising and falling edges cause queue execution.

7.3.2.3 Interval Timer Single-Scan Mode

Queue 2 can also use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 128 to 128K QCLK cycles in binary multiples. When the interval timer single-scan mode is selected and the software sets the single-scan enable bit in QACR2, the timer begins counting. When the time interval expires, an internal trigger event is created to start the queue. The timer is reloaded and begins counting again. Meanwhile, the QADC begins execution with the first CCW in queue 2.

The QADC automatically performs the conversions in the queue until a pause or an end-of-queue condition is encountered. When a pause occurs, queue execution stops until the timer interval expires again, and queue execution continues. When the queue execution reaches an end of queue situation, the timer is held in reset and the single-scan enable bit is cleared. Software may set the single-scan enable bit again, allowing another scan of the queue to be initiated by the interval timer.

The interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause, or may be considered a trigger overrun. Once the queue execution is completed, the single-scan enable bit must be set again to enable the timer to count again.

The interval timer single-scan mode can be used in applications which need coherent results, for example:

- When it is necessary that all samples are guaranteed to be taken during the same scan of the analog pins.
- When the interrupt rate in the periodic timer continuous-scan mode would be too high.
- In sensitive battery applications, where the single-scan mode uses less power than the software initiated continuous-scan mode.

7.3.3 Continuous-Scan Modes

When the application software wants to execute multiple passes through a sequence of conversions defined by a queue, a continuous-scan queue operating mode is selected. By programming the MQ1 field in QACR1, the following software initiated modes can be selected for queue 1:

- Software initiated continuous-scan mode
- External trigger rising edge continuous-scan mode
- External trigger falling edge continuous-scan mode

In addition to the above modes, queue 2 can also be programmed for the periodic timer continuous-scan mode, where a scan is initiated at a selectable time interval using the on-chip periodic/interval timer. The queue operating mode for queue 2 is selected by the MQ2 field in QACR2.

When a queue is programmed for a continuous-scan mode, the single-scan enable bit in the queue control register does not have any meaning or effect. As soon as the queue operating mode is programmed, the selected trigger event can initiate queue execution.

In the case of the software initiated continuous-scan mode, the trigger event is generated internally and queue execution begins immediately. In the other single-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is recorded if a trigger event occurs during queue execution in the external trigger continuous-scan mode and the periodic timer continuous-scan mode. When a pause bit is encountered during the scan, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

In the software initiated continuous-scan mode, the trigger event is generated internally as soon as the conversion is complete for the CCW with the pause bit set. The pause is two QCLKs. In the external trigger continuous-scan mode, the queue remains paused until another trigger edge is received on the external trigger pin. In the periodic timer continuous-scan mode, the next expiration of the timer is the trigger event for the queue.

After the queue execution is complete, the queue status is shown as idle. Since the continuous-scan queue operating modes allow the entire queue to be scanned multiple times, software involvement is not needed to enable queue execution to continue from the idle state. The next trigger event causes queue execution to begin again, starting with the first CCW in the queue.

NOTE

It may not be possible to guarantee coherent samples since the relationship between any conversion in time and any other conversion may be variable (due to programmable trigger events, queue priorities, and so on).

7.3.3.1 Software Initiated Continuous-Scan Mode

When the software initiated continuous-scan mode is programmed, the trigger event is generated automatically by the QADC. Queue execution begins immediately. If a pause is encountered, queue execution ceases for two QCLKs, while another trigger event is generated internally, and then execution continues. When the end-of-queue is reached, another internal trigger event is generated, and queue execution begins again from the beginning of the queue.

While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is paused or idle. The trigger overrun flag is never set while in the software initiated continuous-scan mode.

The software initiated continuous-scan mode keeps the result registers updated more frequently than any of the other queue operating modes. The software can always read the result table to get the latest converted value for each channel. The channels scanned are kept up to date by the QADC without software involvement. Software can read a result value at any time.

NOTE

It may not be possible to guarantee coherent samples since the relationship between any conversion in time and any other conversion may be variable (due to programmable trigger events, queue priorities, and so on).

The software initiated continuous-scan mode may be chosen for either queue, but is normally used only with queue 2. When the software initiated continuous-scan mode is chosen for queue 1, that queue operates continuously and queue 2, being lower in priority, never gets executed. The short interval of time between a queue 1 pause and the internally generated trigger event, or between a queue 1 completion and the subsequent trigger event is not sufficient to allow queue 2 execution to begin.

The software initiated continuous-scan mode is a useful choice with queue 2 for converting channels that do not need to be synchronized to anything, or for the slow-to-change analog channels. Interrupts are normally not used with the software initiated continuous-scan mode. Rather, the software reads the latest conversion result from the result table at any time. Once initiated, software action is not needed to sustain conversions of channel.

7.3.3.2 External Trigger Continuous-Scan Mode

The QADC provides external trigger pins for both queues. When the external trigger software initiated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external trigger signal is programmable, so that the software can choose to begin queue execution on the rising or falling edge. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When the next external trigger edge is detected, the queue execution begins again automatically. Software initialization is not needed between trigger events.

Some applications need to synchronize the sampling of analog channels to external events. There are cases when it is not possible to use software initiation of the queue scan sequence, since interrupt response times vary.

7.3.3.3 Periodic Timer Continuous-Scan Mode

The QADC includes a dedicated periodic/interval timer for initiating a scan sequence for queue 2 only. Software selects a programmable timer interval ranging from 128 to 128K times the QCLK period in binary multiples. The QCLK period is prescaled down from the intermodule bus (IMB) MCU clock.

When the periodic timer continuous-scan mode is selected, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the queue. The timer is then reloaded and begins counting again. Meanwhile, the QADC automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC waits for the periodic interval to expire again, then continues with the queue. After end-of-queue has been detected, the next trigger event causes queue execution to begin again with the first CCW in queue 2.

The periodic timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause or queue completion, or may be considered a trigger overrun. As with all continuous-scan queue operating modes, software action is not needed between trigger events.

Software enables the completion interrupt when using the periodic timer continuous-scan mode. When the interrupt occurs, the software knows that the periodically collected analog results have just been taken. The software can use the periodic interrupt to obtain non-analog inputs as well, such as contact closures, as part of a periodic look at all inputs.

7.4 QADC Clock (QCLK) Generation

Figure 7-2 is a block diagram of the clock subsystem. The QCLK provides the timing for the A/D converter state machine which controls the timing of the conversion. The QCLK is also the input to a 17-stage binary divider which implements the periodic/interval timer. To retain the specified analog conversion accuracy, the QCLK frequency (F_{QCLK}) must be within the tolerance specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**.

Before using the QADC, the software must initialize the prescaler with values that put the QCLK within the specified range. Though most software applications initialize the prescaler once and do not change it, write operations to the prescaler fields are permitted.

CAUTION

> A change in the prescaler value while a conversion is in progress is likely to corrupt the result from any conversion in progress. Therefore, any prescaler write operation should be done only when both queues are in the disabled modes.

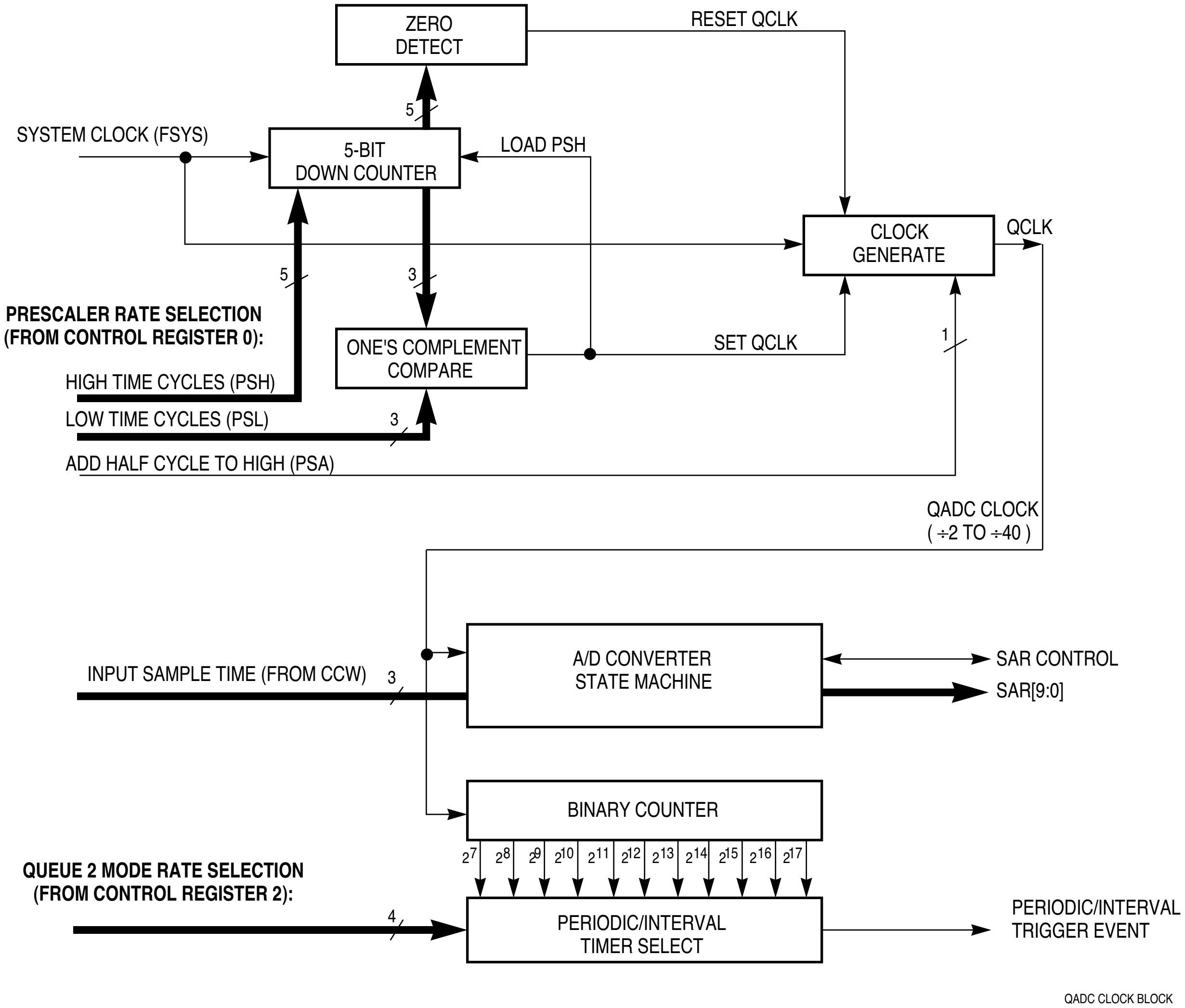


Figure 7-2 QADC Clock Subsystem Functions

To accommodate wide variations of the main MCU clock frequency (IMB system clock – F_{SYS}), QCLK is generated by a programmable prescaler which divides the MCU system clock to a frequency within the specified QCLK tolerance range. To allow the A/D conversion time to be maximized across the spectrum of system clock frequencies, the QADC prescaler permits the frequency of QCLK to be software selectable. It also allows the duty cycle of the QCLK waveform to be programmable.

The software establishes the basic high phase of the QCLK waveform with the PSH (prescaler clock high time) field in QACR0, and selects the basic low phase of QCLK with the PSL (prescaler clock low time) field. The duty cycle of the QCLK can be further modified with the PSA (prescaler add a clock tic) bit in QACR0. The combination of the PSH and PSL parameters establishes the frequency of the QCLK.

Figure 7-2 shows that the prescaler is essentially a variable pulse width signal generator. A 5-bit down counter, clocked at the system clock rate, is used to create both the high phase and the low phase of the QCLK signal. At the beginning of the high phase, the 5-bit counter is loaded with the 5-bit PSH value. When the zero detector finds that the high phase is finished, the QCLK is reset. A 3-bit comparator looks for a one's complement match with the 3-bit PSL value, which is the end of the low phase of the QCLK. The PSA bit allows the QCLK high-to-low transition to be delayed by a half cycle of the input clock.

The following sequence summarizes the process of determining what values are to be put into the prescaler fields in QACR0:

1. Choose the system clock frequency (F_{SYS}).
2. Choose first-try values for PSH, PSL, and PSA, then skip to step 4.
3. Choose different values for PSH, PSL, and PSA.
4. If High QCLK Time is less than T_{PSH} (QADC Clock Duty Cycle – Minimum High Phase Time), return to step 3. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information on T_{PSH}.

NOTE

High QCLK Time (in ns) = 1000 (PSH + 1 + 0.5 PSA) ÷ F_{SYS} (in MHz)

5. If Low QCLK Time is less than T_{PSL} (QADC Clock Duty Cycle – Minimum Low Phase Time), return to step 3. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information on T_{PSL}.

NOTE

Low QCLK Time (in ns) = 1000 (PSL + 1 – 0.5 PSA) ÷ F_{SYS} (in MHz)

6. Calculate the QCLK frequency (F_{QCLK}).

NOTE

F_{QCLK} (in MHz) = 1000 ÷ (High QCLK Time + Low QCLK Time)

7. Choose the number of input sample cycles for a typical input channel.
8. If the calculated conversion times are not sufficient, return to step 3.

9. Code the selected PSH, PSL, and PSA values into the prescaler fields of QACR0.

Figure 7-3 and **Table 7-1** show examples of QCLK programmability. The examples include conversion times based on the following assumptions:

- System clock (F_{SYS}) = 20.97 MHz.
- Input sample time is as fast as possible (IST = 0, 2 QCLK cycles).

Figure 7-3 and **Table 7-1** also show the conversion time calculated for a single conversion in a queue. For other MCU system clock frequencies and other input sample times, the same calculations can be made.

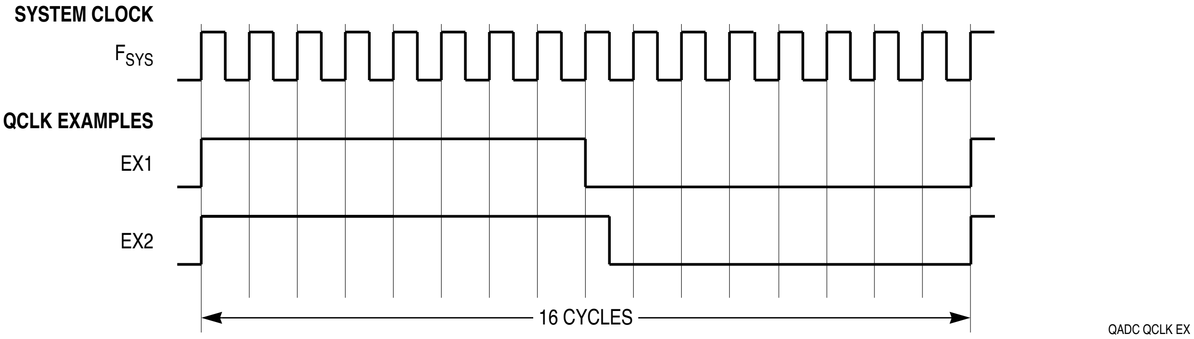


Figure 7-3 QADC Clock Programmability Examples

Table 7-1 QADC Clock Programmability

Control Register 0 Information				**F_{SYS} = 20.97 Input Sample Time (IST) = %00**	
Example Number	**PSH**	**PSA**	**PSL**	**QCLK (MHz)**	**Conversion Time (µs)**
1	7	0	7	1.0	18.0
2	7	1	7	1.0	18.0

The MCU system clock frequency is the basis of the QADC timing. The QADC requires that the system clock frequency be at least twice the QCLK frequency. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information on the minimum and maximum allowable QCLK frequencies.

The QCLK frequency is established by the combination of the PSH and PSL parameters in QACR0. The 5-bit PSH field selects the number of system clock cycles in the high phase of the QCLK wave. The 3-bit PSL field selects the number of system clock cycles in the low phase of the QCLK wave.

Example 1 in **Figure 7-3** shows that when PSH = 3, the QCLK remains high for four cycles of the system clock. It also shows that when PSL = 3, the QCLK remains low for four system clock cycles. In example 1, the equation for QCLK high cycles is the PSH value plus one, and the equation for the QCLK low cycles is the PSL value plus one.

Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information on minimum QCLK high time (T_{PSH}) and the minimum QCLK low time (T_{PSL}).

In order to be able to tune the QCLK as close as possible to the fastest conversion time for any given system clock frequency, the QADC permits one more programmable control of the QCLK high and low time. The PSA parameter in QACR0 allows the QCLK high phase to be stretched for a half cycle of the system clock, and correspondingly, the QCLK low phase is shortened by a half cycle of the system clock.

Example 2 in **Figure 7-3** is the same as Example 1, except that the PSA bit is set. The QCLK high phase has 4.5 system clock cycles; the QCLK low phase has 3.5 system clock cycles.

The following are equations for calculating the QCLK high and low phases:

- High QCLK Time = 1000 (PSH + 1 + 0.5 PSA) ÷ F_{SYS} (in ns)
- Low QCLK Time = 1000 (PSL + 1 – 0.5 PSA) ÷ F_{SYS} (in ns)
- F_{QCLK} = 1000 ÷ (High QCLK Time + Low QCLK Time) (in MHz)

Where:

- PSH = 0 to 31, the prescaler QCLK high cycles in QACR0
- PSL = 0 to 7, the prescaler QCLK low cycles in QACR0
- PSA = 0 to 1, the prescaler QCLK add half cycle in QACR0
- F_{SYS} = System clock frequency in MHz
- F_{QCLK} = QCLK frequency in MHz

7.5 Periodic/Interval Timer

The on-chip periodic/interval timer is enabled to generate trigger events at a programmable interval, initiating execution of queue 2. The periodic/interval timer stays reset under the following conditions:

- Queue 2 is programmed to any queue operating mode which does not use the periodic/interval timer
- Interval timer single-scan mode is selected, but the single-scan enable bit is set to zero
- IMB system reset or the master reset is asserted
- Stop mode is selected
- Freeze mode is selected

Two other conditions which cause a pulsed reset of the timer are:

- Roll over of the timer counter
- A queue operating mode change from one periodic/interval timer mode to another periodic/interval timer mode

During the stop mode, the periodic/interval timer is held in reset. Since stop mode causes QACR2 to be initialized to zero, a valid periodic or interval timer mode must be written to QACR2 after stop mode is exited to release the timer from reset.

When the IMB internal FREEZE line is asserted and a periodic or interval timer mode is selected, the timer counter is reset after the current conversion completes. When the periodic or interval timer mode has been enabled (the timer is counting), but a trigger event has not been issued, the freeze mode takes effect immediately, and the timer is held in reset. When the internal FREEZE line is negated, the timer counter starts counting from the beginning. Refer to paragraph 3.3.2 in **SECTION 3 CONFIGURATION AND CONTROL** for more information.

7.6 Control and Status Registers

The following paragraphs describe the control and status registers. The QADC has three control registers and one status register.

7.6.1 Control Register 0

Control register 0 establishes the QCLK with prescaler parameter fields and defines whether external multiplexing is enabled. All of the implemented control register fields can be read or written, reserved locations read zero and writes have no effect. They are typically written once when the software initializes the QADC, and not changed afterwards.

QACR0 — Control Register 0 **$####0A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX	RESERVED						PSH					PSA	PSL		
RESET:															
0							0	0	0	1	1	0	0	1	1

MUX — Externally Multiplexed Mode

The MUX bit allows the software to select the externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA0, MA1 and MA2 pins to be outputs.

0 = Internally multiplexed, 16 possible channels.
1 = Externally multiplexed, 44 possible channels.

PSH[8:4] — Prescaler Clock High Time

The PSH field selects the QCLK high time in the prescaler. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information on QADC operating clock frequency (F_{QCLK}) values. To keep the QCLK within the specified range, the PSH field selects the high time of the QCLK, which can range from 1 to 32 system clock cycles. The minimum high time for the QCLK is specified as T_{PSH}.

Table 7-2 displays the bits in PSH field which enable a range of QCLK high times.

Table 7-2 Prescaler Clock High Times

PSH[8:4]	QCLK High Time
00000	1 System Clock Cycle
00001	2 System Clock Cycles
00010	3 System Clock Cycles
00011	4 System Clock Cycles
00100	5 System Clock Cycles
00101	6 System Clock Cycles
00110	7 System Clock Cycles
00111	8 System Clock Cycles
01000	9 System Clock Cycles
01001	10 System Clock Cycles
01010	11 System Clock Cycles
01011	12 System Clock Cycles
01100	13 System Clock Cycles
01101	14 System Clock Cycles
01110	15 System Clock Cycles
01111	16 System Clock Cycles
10000	17 System Clock Cycles
10001	18 System Clock Cycles
10010	19 System Clock Cycles
10011	20 System Clock Cycles
10100	21 System Clock Cycles
10101	22 System Clock Cycles
10110	23 System Clock Cycles
10111	24 System Clock Cycles
11000	25 System Clock Cycles
11001	26 System Clock Cycles
11010	27 System Clock Cycles
11011	28 System Clock Cycles
11100	29 System Clock Cycles
11101	30 System Clock Cycles
11110	31 System Clock Cycles
11111	32 System Clock Cycles

PSA — Prescaler Add a Clock Tic

PSA adds a system clock tic to the QCLK high time in the prescaler. The PSH and PSL fields establish F_{QCLK}. The PSA field does not change the frequency of the clock, however, it does change the duty cycle by one tic, or one half of the MCU system clock period. The PSA bit modifies the QCLK duty cycle by adding one system clock tic to the high time and subtracting one system clock tic from the low time.

0 = QCLK high and low times are not modified.

1 = Add one system clock half cycle to the high time of the QCLK and subtract one system clock half cycle from the low time.

PSL[2:0] — Prescaler Clock Low Time

The PSL field selects the QCLK low time in the prescaler. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information on F_{QCLK} values. To keep the QCLK within the specified range, the PSL field selects the low time of the QCLK, which can range from one to eight system clock cycles. The minimum low time for the clock is specified as T_{PSL}.

Table 7-2 displays the bits in PSL field which enable a range of QCLK low times.

Table 7-3 Prescaler Clock Low Times

PSL[2:0]	QCLK Low Time
000	1 System Clock Cycle
001	2 System Clock Cycles
010	3 System Clock Cycles
011	4 System Clock Cycles
100	5 System Clock Cycles
101	6 System Clock Cycles
110	7 System Clock Cycles
111	8 System Clock Cycles

7.6.2 Control Register 1

Control register 1 is the mode control register for the operation of queue 1. The applications software defines the queue operating mode for the queue, and may enable a completion and/or pause interrupt. All of the control register fields are read/write data. However, the SSE1 bit always reads as zero unless the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC, and not changed afterwards.

QACR1 — Control Register 1 **$####0C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIE1	PIE1	SSE1	NOT USED		MQ1			NOT USED							
RESET:															
0	0	0			0	0	0								

CIE1 — Queue 1 Completion Interrupt Enable

CIE1 enables an interrupt upon completion of queue 1. The interrupt request is initiated when the conversion is complete for the CCW in queue 1.

0 = Disable the queue completion interrupt associated with queue 1.

1 = Enable an interrupt after the conversion of the sample requested by the last CCW in queue 1.

PIE1 — Queue 1 Pause Interrupt Enable

PIE1 enables an interrupt when queue 1 enters the pause state. The interrupt request is initiated when conversion is complete for a CCW that has the pause bit set.

0 = Disable the pause interrupt associated with queue 1.

1 = Enable an interrupt after the conversion of the sample requested by a CCW in queue 1 which has the pause bit set.

SSE1 — Queue 1 Single-Scan Enable Bit

SSE1 enables a single-scan of queue 1 to start after a trigger event occurs. The SSE1 bit may be set to a one during the same write cycle when the MQ1 bits are set for one of the single-scan queue operating modes. The single-scan enable bit can be written as a one or a zero, but is always read as a zero, unless a test mode is selected. The SSE1 bit enables a trigger event to initiate queue execution for any single-scan operation on queue 1. The QADC clears the SSE1 bit when the single-scan is complete.

0 = Trigger events are not accepted for single-scan modes.
1 = Accept a trigger event to start queue 1 in a single-scan mode.

MQ1[10:8] — Queue 1 Operating Mode

The MQ1 field selects the queue operating mode for queue 1.

Table 7-4 shows the bits in the MQ1 field which enable different queue 1 operating modes.

Table 7-4 Queue 1 Operating Modes

MQ1[10:8]	Operating Modes
000	Disabled mode, conversions do not occur
001	Software triggered single-scan mode (started with SSE1)
010	External trigger rising edge single-scan mode (on ETRIG1 pin)
011	External trigger falling edge single-scan mode (on ETRIG1 pin)
100	Reserved mode, conversions do not occur
101	Continuous-scan software triggered mode
110	External trigger rising edge continuous-scan mode (on ETRIG1 pin)
111	External trigger falling edge continuous-scan mode (on ETRIG1 pin)

7.6.3 Control Register 2

Control register 2 is the mode control register for the operation of queue 2. Software specifies the queue operating mode of queue 2, and may enable a completion and/or a pause interrupt. All control register fields are read/write data, except the SSE2 bit, which is readable only when the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC, and not changed afterwards.

QACR2 — Control Register 2 **$####0E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIE2	PIE2	SSE2	MQ2					RES	NOT USED	BQ2					
RESET:															
0	0	0	0	0	0	0	0	0		1	0	0	1	1	1

CIE2 — Queue 2 Completion Software Interrupt Enable

CIE2 enables an interrupt upon completion of queue 2. The interrupt request is initiated when the conversion is complete for the CCW in queue 2.

0 = Disable the queue completion interrupt associated with queue 2.
1 = Enable an interrupt after the conversion of the sample requested by the last CCW in queue 2.

PIE2 — Queue 2 Pause Software Interrupt Enable

PIE2 enables an interrupt when queue 2 enters the pause state. The interrupt request is initiated when conversion is complete for a CCW that has the pause bit set.

0 = Disable the pause interrupt associated with queue 2.
1 = Enable an interrupt after the conversion of the sample requested by a CCW in queue 2 which has the pause bit set.

SSE2 — Queue 2 Single-Scan Enable Bit

SSE2 enables a single-scan of queue 2 to start after a trigger event occurs. The SSE2 bit may be set to a one during the same write cycle when the MQ2 bits are set for one of the single-scan queue operating modes. The single-scan enable bit can be written as a one or a zero, but is always read as a zero, unless a test mode is selected. The SSE2 bit enables a trigger event to initiate queue execution for any single-scan operation on queue 2. The QADC clears the SSE2 bit when the single-scan is complete.

0 = Trigger events are not accepted for single-scan modes.
1 = Accept a trigger event to start queue 2 in a single-scan mode.

MQ2[12:8] — Queue 2 Operating Mode

The MQ2 field selects the queue operating mode for queue 2.

Table 7-5 shows the bits in the MQ2 field which enable different queue 2 operating modes.

Table 7-5 Queue 2 Operating Modes

MQ2[12:8]	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	External trigger rising edge single-scan mode (on ETRIG2 pin)
00011	External trigger falling edge single-scan mode (on ETRIG2 pin)
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Continuous-scan software triggered mode
10010	External trigger rising edge continuous-scan mode (on ETRIG2 pin)
10011	External trigger falling edge continuous-scan mode (on ETRIG2 pin)
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8

Table 7-5 Queue 2 Operating Modes (Continued)

MQ2[12:8]	Operating Modes
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^{9}
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	Reserved mode

RES — Queue 2 Resume

RES selects the queue 2 resumption point after suspension due to queue 1. If RES is changed during the execution of queue 2, the change is not recognized until an end-of-queue condition is reached, or the queue operating mode of queue 2 is changed.

The primary reason for selecting re-execution of the entire queue or subqueue is to guarantee that all samples are taken consecutively in one scan (coherency).

When subqueues are not used, queue 2 execution restarts after suspension with the first CCW in queue 2. When a pause has previously occurred in queue 2 execution, queue execution restarts after suspension with the first CCW in the current subqueue.

A subqueue is considered to be a stand-alone sequence of conversions. Once a pause flag has been set to report subqueue completion, that subqueue is not repeated until all CCWs in queue 2 are executed.

An example of using the resume bit is when the frequency of queue 1 trigger events prohibit queue 2 completion. If the rate of queue 1 execution is too high, it is best for queue 2 execution to continue with the CCW that was being converted when queue 2 was suspended. This allows queue 2 to eventually complete execution.

0 = After suspension, begin execution with the first CCW in queue 2 or the current subqueue.
1 = After suspension, begin execution with the aborted CCW in queue 2.

BQ2[5:0] — Beginning of Queue 2

The BQ2 field indicates the CCW location where queue 2 begins. To allow the length of queue 1 and queue 2 to vary, a programmable pointer identifies the CCW table location where queue 2 begins. The BQ2 field also serves as an end-of-queue condition for queue 1.

Software defines the beginning of queue 2 by programming the BQ2 field in QACR2. BQ2 is usually programmed before or at the same time as the queue operating mode for queue 2 is selected. If BQ2 is 40 or greater, queue 2 does not have any entries, and the entire CCW table is dedicated to queue 1. If BQ2 is 0, the entire CCW table is dedicated to queue 2. As a special case, when a queue operating mode for queue 1 is selected and a trigger event occurs for queue 1 with BQ2 set to 0, queue 1 execution is terminated after CCW0 is read. Conversions do not occur.

The BQ2 pointer may be changed dynamically, to alternate between queue 2 scan sequences. A change in BQ2 while queue 2 is active does not take effect until an end-of-queue condition is recognized, or until the queue operating mode of queue 2 is changed. For example, two scan sequences could be defined as follows: the first sequence starts at CCW10, with a pause after CCW11 and an EOQ programmed in CCW15; the second sequence starts at CCW16, with a pause after CCW17 and an EOQ programmed in CCW39.

With BQ2 set to CCW10 and the continuous-scan mode selected, queue execution begins. When the pause is encountered in CCW11, a software interrupt routine can redefine BQ2 to be CCW16. Therefore, after the end-of-queue is recognized in CCW15, an internal retrigger event is generated and execution restarts at CCW16. When the pause software interrupt occurs again, software can change BQ2 back to CCW10. After the end-of-queue is recognized in CCW39, an internal retrigger event is created and execution now restarts at CCW10.

If BQ2 is changed while queue 1 is active, the effect of BQ2 as an end-of-queue indication for queue 1 is immediate.

NOTE

> If BQ2 was assigned to the CCW that queue 1 is currently working on, then that conversion is completed before BQ2 takes effect.

Each time a CCW is read for queue 1, the CCW location is compared with the current value of the BQ2 pointer to detect a possible end-of-queue condition. For example, if BQ2 is changed to CCW3 while queue 1 is converting CCW2, queue 1 is terminated after the conversion is completed. However, if BQ2 is changed to CCW1 while queue 1 is converting CCW2, the QADC would not recognize a BQ2 end-of-queue condition until queue 1 execution reached CCW1 again, presumably on the next pass through the queue.

7.6.4 Status Register

The status register contains information about the state of each queue and the current A/D conversion. Except for the four flag bits (CF1, PF1, CF2, and PF2) and the two trigger overrun bits (TOR1 and TOR2), all of the status register fields contain read-only data. The four flag bits and the two trigger overrun bits are cleared by writing a zero to the bit after the bit was previously read as a one.

QASR — Status Register **$####10**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CF1 — Queue 1 Completion Flag

CF1 indicates that a queue 1 scan has been completed. The scan completion flag is set by the QADC when the input channel sample requested by the last CCW in queue 1 is converted, and the result is stored in the result table.

The end of queue 1 is identified when execution is complete on the CCW in the location prior to that pointed to by BQ2, when the current CCW contains an end-of-queue code instead of a valid channel number, or when the currently completed CCW is in the last location of the CCW RAM.

When CF1 is set and interrupts are enabled for that queue completion flag, the QADC asserts an interrupt request at the level specified by IRL1 in the interrupt register (QADCINT). The software reads the completion flag during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to the completion flag bit, when the bit was previously read as a one. Once set, only software or reset can clear CF1.

CF1 is maintained by the QADC regardless of whether the corresponding interrupt is enabled. The software polls for CF1 bit to see if it is set. This allows the software to recognize that the QADC is finished with a queue 1 scan. The software acknowledges that it has detected the completion flag being set by writing a zero to the completion flag after the bit was read as a one.

Refer to **SECTION 8 INTERRUPTS** for more information.

0 = Queue 1 scan is not complete.
1 = Queue 1 scan is complete.

PF1 — Queue 1 Pause Flag

PF1 indicates that a queue 1 scan has reached a pause. PF1 is set by the QADC when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table.

Once PF1 is set, the queue enters the paused state and waits for a trigger event to allow queue execution to continue. However, if the CCW with the pause bit set is the last CCW in a queue, the queue execution is complete. The queue status becomes idle, not paused, and both the pause and completion flags are set.

When PF1 is set and interrupts are enabled for the corresponding queue, the QADC asserts an interrupt request at the level specified by IRL1 in the interrupt register. The software may read PF1 during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to PF1, when the bit was previously read as a one. Once set, only software or reset can clear PF1.

PF1 is maintained by the QADC regardless of whether the corresponding interrupts are enabled. The software may poll PF1 to find out when the QADC has reached a pause in scanning a queue. The software acknowledges that it has detected a pause flag being set by writing a zero to PF1 after the bit was last read as a one.

0 = Queue 1 has not reached a pause.
1 = Queue 1 has reached a pause.

CF2 — Queue 2 Completion Flag

CF2 indicates that a queue 2 scan has been completed. CF2 is set by the QADC when the input channel sample requested by the last CCW in queue 2 is converted, and the result is stored in the result table.

The end of queue 2 is identified when the current CCW contains an end-of-queue code instead of a valid channel number, or when the currently completed CCW is in the last location of the CCW RAM.

When CF2 is set and interrupts are enabled for that queue completion flag, the QADC asserts an interrupt request at the level specified by IRL2 in the interrupt register (QADCINT). The software reads CF2 during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to the CF2 bit, when the bit was previously read as a one. Once set, only software or reset can clear CF2.

CF2 is maintained by the QADC regardless of whether the corresponding interrupts are enabled. The software polls for CF2 to see if it is set. This allows the software to recognize that the QADC is finished with a queue 2 scan. The software acknowledges that it has detected the completion flag being set by writing a zero to the completion flag after the bit was read as a one.

Refer to **SECTION 8 INTERRUPTS** for more information.

0 = Queue 2 scan is not complete.
1 = Queue 2 scan is complete.

PF2 — Queue 2 Pause Flag

PF2 indicates that a queue 2 scan has reached a pause. PF2 is set by the QADC when the current queue 2 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table.

Once PF2 is set, the queue enters the paused state and waits for a trigger event to allow queue execution to continue. However, if the CCW with the pause bit set is the last CCW in a queue, the queue execution is complete. The queue status becomes idle, not paused, and both the pause and completion flags are set.

When PF2 is set and interrupts are enabled for the corresponding queue, the QADC asserts an interrupt request at the level specified by IRL2 in the interrupt register. The software reads PF2 during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to PF2, when the bit was previously read as a one. Once set, only software or reset can clear PF2.

PF2 is maintained by the QADC regardless of whether the corresponding interrupts are enabled. The software may poll PF2 to find out when the QADC has reached a pause in scanning a queue. The software acknowledges that it has detected a pause flag being set by writing a zero to PF2 after the bit was last read as a one.

0 = Queue 2 has not reached a pause.
1 = Queue 2 has reached a pause.

TOR1 — Queue 1 Trigger Overrun

TOR1 indicates that an unexpected trigger event has occurred for queue 1. TOR1 can be set only while queue 1 is in the active state.

A trigger event generated by a transition on the external trigger pin or by the periodic/interval timer may be recorded as a trigger overrun. TOR1 can only occur when using an external trigger mode. TOR1 cannot occur when the software initiated single-scan mode and the software initiated continuous-scan mode are selected.

TOR1 occurs when a trigger event is received while a queue is executing and before the scan has completed or paused. TOR1 can also occur when a trigger event has occurred on a queue, but execution has not yet begun. TOR1 has no effect on the queue execution.

After a trigger event has occurred for queue 1, and before the scan has completed or paused, additional queue 1 trigger events are not retained. Such trigger events are considered unexpected, and the QADC sets the TOR1 error status bit. An unexpected trigger event may be a system overrun situation, indicating a system loading mismatch. The software acknowledges that it has detected a trigger overrun being set by writing a zero to the trigger overrun, after the bit was read as a one. Once set, only software or reset can clear TOR1.

0 = No unexpected queue 1 trigger events have occurred.
1 = At least one unexpected queue 1 trigger event has occurred.

TOR2 — Queue 2 Trigger Overrun

TOR2 indicates that an unexpected trigger event has occurred for queue 2. TOR2 can be set when queue 2 is in the active, suspended, and trigger pending states.

The TOR2 trigger overrun can only occur when using an external trigger mode or a periodic/interval timer mode. Trigger overruns cannot occur when the software initiated single-scan mode and the software initiated continuous-scan mode are selected.

TOR2 occurs when a trigger event is received while a queue is executing and before the scan has completed or paused. TOR2 also occurs when a trigger event has occurred on a queue, but execution has not yet begun. TOR2 has no effect on the queue execution. A trigger event that causes a trigger overrun is not retained since it is considered unexpected.

After a trigger event has occurred for queue 2, and before the scan has completed or paused, additional queue 2 trigger events are not retained, and are considered unexpected. Also, when a queue 2 trigger event is pending, but queue 1 is active, additional queue 2 trigger events are considered unexpected. In both cases, the QADC sets the TOR2 error status bit.

An unexpected trigger event may be a system overrun situation, indicating a system loading mismatch. The software acknowledges that it has detected a trigger overrun being set by writing a zero to the trigger overrun, after the bit was read as a one. Once set, only software or reset can clear TOR2.

0 = No unexpected queue 2 trigger events have occurred.
1 = At least one unexpected queue 2 trigger event has occurred.

QS[9:6] — Queue Status

The 4-bit read-only QS field indicates the current condition of queue 1 and queue 2. The following are the five queue status conditions:

- Idle
- Active
- Paused
- Suspended
- Trigger pending

The two most significant bits are associated primarily with queue 1, and the remaining two bits are associated with queue 2. Since the priority scheme between the two queues causes the status to be interlinked, the status bits are considered as one 4-bit field.

Table 7-6 shows the bits in the QS field and how they affect the status of queue 1 and queue 2. Refer to **SECTION 9 QUEUE PRIORITY EXAMPLES**, which shows the 4-bit queue status field transitions in typical situations.

Table 7-6 Queue Status

QS[9:6]	Queue 1/Queue 2 States
0000	Queue 1 idle, Queue 2 idle
0001	Queue 1 idle, Queue 2 paused
0010	Queue 1 idle, Queue 2 active
0011	Queue 1 idle, Queue 2 trigger pending
0100	Queue 1 paused, Queue 2 idle
0101	Queue 1 paused, Queue 2 paused
0110	Queue 1 paused, Queue 2 active
0111	Queue 1 paused, Queue 2 trigger pending
1000	Queue 1 active, Queue 2 idle
1001	Queue 1 active, Queue 2 paused
1010	Queue 1 active, Queue 2 suspended
1011	Queue 1 active, Queue 2 trigger pending
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

One or both queues may be in the idle state. When a queue is idle, CCWs are not being executed for that queue, the queue is not in the pause state, and there is not a trigger pending.

The idle state occurs when a queue is disabled, when a queue is in a reserved mode, or when a queue is in a valid queue operating mode awaiting a trigger event to initiate queue execution.

A queue is in the active state when a valid queue operating mode is selected, when the selected trigger event has occurred, or when the QADC is performing a conversion specified by a CCW from that queue.

Only one queue can be active at a time. Either or both queues can be in the paused state. A queue is paused when the previous CCW executed from that queue had the pause bit set. The QADC does not execute any CCWs from the paused queue until a trigger event occurs. Consequently, the QADC can service queue 2 while queue 1 is paused.

Only queue 2 can be in the suspended state. When a trigger event occurs on queue 1 while queue 2 is executing, the current queue 2 conversion is aborted. The queue 2 status is reported as suspended. Queue 2 transitions back to the active state when queue 1 becomes idle or paused.

A trigger pending state is required since both queues cannot be active at the same time. The status of queue 2 is changed to trigger pending when a trigger event occurs for queue 2 while queue 1 is active. In the opposite case, when a trigger event occurs for queue 1 while queue 2 is active, queue 2 is aborted and the status is reported as queue 1 active, queue 2 suspended. So due to the priority scheme, only queue 2 can be in the trigger pending state.

There are two transition cases which cause the queue 2 status to be trigger pending before queue 2 is shown to be in the active state. When queue 1 is active and there is a trigger pending on queue 2, after queue 1 completes or pauses, queue 2 continues to be in the trigger pending state for a few clock cycles. The following are fleeting status conditions:

- Queue 1 idle with queue 2 trigger pending.
- Queue 1 paused with queue 2 trigger pending.

Figure 7-4 displays the status conditions of the queue status field as the QADC goes through the transition from queue 1 active to queue 2 active.

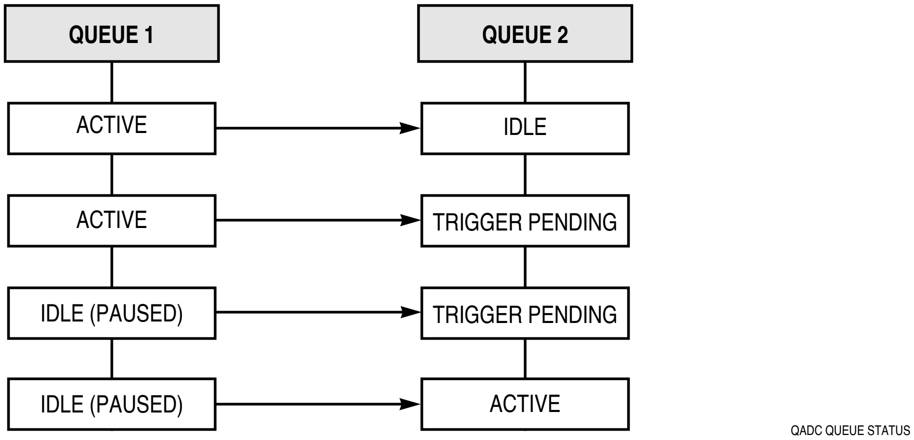


Figure 7-4 Queue Status Transition

The queue status field is affected by the stop mode. Since all of the analog logic and control registers are reset, the queue status field is reset to queue 1 idle, queue 2 idle. During the freeze mode, the queue status field is not modified. The queue status field retains the status it held prior to freezing. As a result, the queue status can show queue 1 active, queue 2 idle, even though neither queue is being executed during freeze.

CWP — Command Word Pointer

The CWP allows the software to know which CCW is executing at present, or was last completed. The command word pointer is a software read-only field, and write operations have no effect. The CWP allows software to monitor the progress of the QADC scan sequence. The CWP field is a CCW word pointer with a valid range of 0 to 39.

When a queue enters the paused state, the CWP points to the CCW with the pause bit set. While in pause, the CWP value is maintained until a trigger event occurs on the same queue or the other queue. Usually, the CWP is updated a few clock cycles before the queue status field shows that the queue has become active. For example, software may read a CWP pointing to a CCW in queue 2, and the status field shows queue 1 paused, queue 2 trigger pending.

When the QADC finishes the scan of the queue, the CWP points to the CCW where the end-of-queue condition was detected. Therefore, when the end-of-queue condition is a CCW with the EOQ code (channel 63), the CWP points to the CCW containing the EOQ.

When the last CCW in a queue is in the last CCW table location (CCW39), and it does not contain the EOQ code, the end-of-queue is detected when the following CCW is read, so the CWP points to word 40.

Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, the CWP points to the same CCW as BQ2.

Since the CWP is six bits, the value could range from 0 to 63. However, since there are 40 locations in the CCW table, the normal range of CWP is 0 to 39. The CWP shows 40 when queue execution terminates with the physical end of the CCW table. The CWP would also show a value between 40 and 63 if BQ2 were set in that range and queue 2 execution were initiated.

During the stop mode, the CWP is reset to zero, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWP is unchanged; it points to the last executed CCW.

7.7 Conversion Command Word Table

The CCW table is a RAM, 40 words long and 10 bits wide, which can be programmed by the software to request conversions of one or more analog input channels. The entries in the CCW table are 10-bit conversion command words. The CCW table is written by software and is not modified by the QADC. Each CCW requests the conversion of an analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW. The ten implemented bits of the CCW word are read/write data, but they may be written once when the software initializes the QADC, and not changed afterwards. Unimplemented bits are read as zeros, and write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry. The QADC provides 40 CCW table entries. If more results are needed, software uses one or more result table locations for multiple channels.

The beginning of queue 1 is the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer (BQ2) in QACR2. To dedicate the entire CCW table to queue 1, queue 2 is programmed to be in the disabled mode, and the beginning of BQ2 is programmed to any value greater than 39. To dedicate the entire CCW table to queue 2, queue 1 is programmed to be in the disabled mode, and BQ2 is specified as the first location in the CCW table.

Figure 7-5 illustrates the operation of the queue structure.

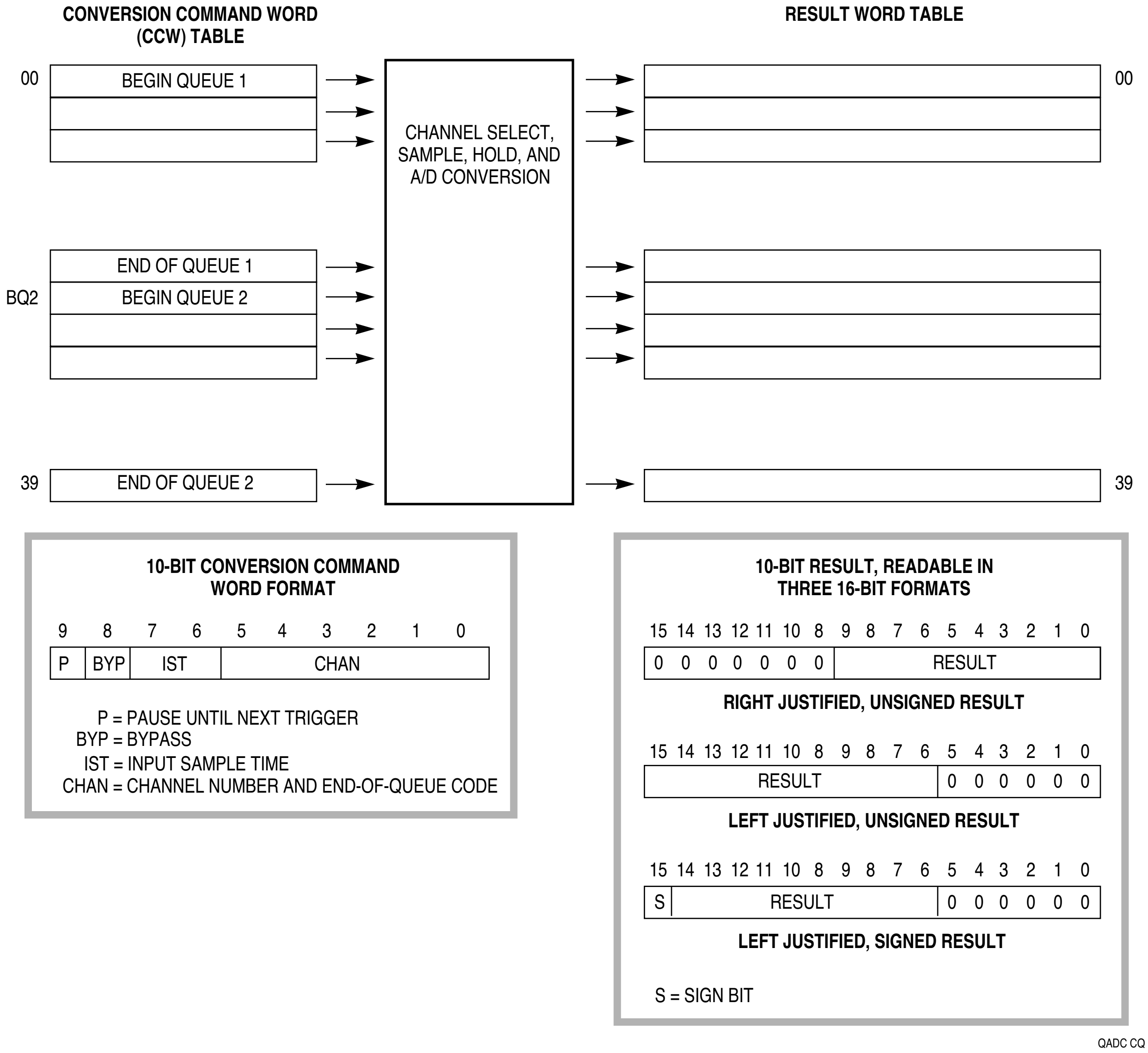


Figure 7-5 QADC Conversion Queue Operation

To prepare the QADC for a scan sequence, the software writes to the CCW table to specify the desired channel conversions. The software also establishes the criteria for initiating the queue execution by programming the queue operating mode. The queue operating mode determines what type of trigger event causes queue execution to begin. “Trigger event” is used to refer to any of the ways to cause the QADC to begin executing the CCWs in a queue or subqueue. An “external trigger” is only one of the possible “trigger events”.

A scan sequence may be initiated by the following:

- A software command
- Expiration of the periodic/interval timer
- External trigger signal

The software also specifies whether the QADC is to perform a single pass through the queue or is to scan continuously. When a single-scan mode is selected, the software selects the queue operating mode and sets the single-scan enable bit. When a continuous-scan mode is selected, the queue remains active in the selected queue operating mode after the QADC completes each queue scan sequence.

During queue execution, the QADC reads each CCW from the active queue and executes conversions in four stages:

- Initial sample
- Transfer
- Final sample
- Resolution

During initial sample, the selected input channel is connected to the sample capacitor at the input of the sample buffer amplifier.

During the transfer period, the sample capacitor is disconnected from the multiplexer, and the stored voltage is buffered and transferred to the RC DAC array.

During the final sample period, the sample capacitor and amplifier are bypassed, and the multiplexer input charges the RC DAC array directly. Each CCW specifies a final input sample time of 2, 4, 8, or 16 QCLK cycles. When an analog-to-digital conversion is complete, the result is written to the corresponding location in the result word table. The QADC continues to sequentially execute each CCW in the queue until the end of the queue is detected or a pause bit is found in a CCW.

When the pause bit is set in the current CCW, the QADC stops execution of the queue until a new trigger event occurs. The pause status flag bit is set, which may cause an interrupt to notify the software that the queue has reached the pause state. After the trigger event occurs, the paused state ends and the QADC continues to execute each CCW in the queue until another pause is encountered or the end of the queue is detected.

The following indicate the end-of-queue condition:

- The CCW channel field is programmed with 63 ($3F) to specify the end of the queue.

- The end of queue 1 is implied by the beginning of queue 2, which is specified in the BQ2 field in QACR2.
- The physical end of the queue RAM space defines the end of either queue.

When any of the end-of-queue conditions is recognized, a queue completion flag is set, and if enabled, an interrupt is issued to the software. The following situations prematurely terminate queue execution:

- Since queue 1 is higher in priority than queue 2, when a trigger event occurs on queue 1 during queue 2 execution, the execution of queue 2 is suspended by aborting the execution of the CCW in progress, and the queue 1 execution begins. When queue 1 execution is completed, queue 2 conversions restart with the first CCW entry in queue 2 or the first CCW of the queue 2 subqueue being executed when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The resume (RES) field in QACR2 allows the software to select where queue 2 begins after suspension. By choosing to re-execute all of the suspended queue 2 queue and subqueue CCWs, all of the samples are guaranteed to have been taken during the same scan pass. However, a high trigger event rate for queue 1 can prohibit the completion of queue 2. If this occurs, the software may choose to begin execution of queue 2 with the aborted CCW entry.
- Software can change the queue operating mode to disabled mode. Any current conversion taking place for that queue is aborted. Putting a queue into the disabled mode does not power down the converter.
- Software can change the queue operating mode to another valid mode. Any current conversion taking place for that queue is aborted. The queue restarts at the beginning of the queue, once an appropriate trigger event occurs.
- For low power operation, software can set the stop mode bit to prepare the module for a loss of clocks. The QADC aborts any conversion in progress when the stop mode is entered.
- When the freeze enable bit is set by software and the IMB internal FREEZE line is asserted, the QADC freezes at the end of the current conversion. When internal FREEZE is negated, the QADC resumes queue execution beginning with the next CCW entry. Refer to paragraph 3.3.2 in **SECTION 3 CONFIGURATION AND CONTROL** for more information.

CCW — Conversion Command Word Table **$####30–$####7E**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOT USED						P	BYP	IST		CHAN					
RESET:							U	U	U	U	U	U	U	U	U	U

P — Pause

The pause bit allows software to create sub-queues within queue 1 and queue 2. The QADC performs the conversion specified by the CCW with the pause bit set, and then the queue enters the pause state. Another trigger event causes execution to continue from the pause to the next CCW.

0 = Do not enter the pause state after execution of the current CCW.
1 = Enter the pause state after execution of the current CCW.

BYP — Sample Amplifier Bypass

Setting the BYP field in the CCW enables the amplifier bypass mode for a conversion, and subsequently changes the timing. The initial sample time and the transfer time are eliminated, reducing the potential conversion time by six QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKs must be allowed. This results in a shortened conversion time of four QCLKS. When using this mode, the external circuit should be of low source impedance. Loading effects of the external circuitry need to be considered, since the benefits of the sample amplifier are not present.

0 = Amplifier bypass mode disabled.
1 = Amplifier bypass mode enabled.

IST[7:6] — Input Sample Time

The IST field allows software to specify the length of the sample window. Provision is made to vary the input sample time, through software control, to offer flexibility in the source impedance of the circuitry providing the QADC analog channel inputs. Longer sample times permit more accurate A/D conversions of signals with higher source impedances.

Table 7-7 shows the four selectable input sample times.

Table 7-7 Input Sample Times

IST[7:6]	Input Sample Times
00	Input sample time = QCLK period x 2
01	Input sample time = QCLK period x 4
10	Input sample time = QCLK period x 8
11	Input sample time = QCLK period x 16

The programmable sample time can also be used to increase the time interval between conversions to adjust the queue execution time or the sampling rate.

In addition to the four sample times listed in **Table 7-7**, the sample amplifier may be bypassed entirely. This results in a shorter conversion time (by four QCLKs) may also reduce the effects of the sample amplifier on the system performance. The sample amplifier can be bypassed on each conversion by setting the BYP bit field in the CCW. This mode is mainly intended for factory testing, but may also be used in applications with some considerations.

The bypass option shortens the conversion time by six QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKS must be used. This results in a conversion time savings of four QCLKS. The slightly shorter sample time may be of use in some applications, depending on system constraints, but may be offset by the loss of the sample amplifier benefits. This means that external circuit loading from the QADC may increase due to the direct connection of the internal RC DAC array during the sample phase. Typically, the bypass mode may only be used if external source impedance is kept low. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for information on the maximum allowable input source impedance.

CHAN[5:0] — Channel Number

The CHAN field selects the input channel number. The software programs the channel field of the CCW with the channel number corresponding to the analog input pin to be sampled and converted. The analog input pin channel number assignments and the pin definitions vary depending on whether the multiplexed or non-multiplexed mode is used by the application. As far as the queue scanning operations are concerned, there is no distinction between an internally or externally multiplexed analog input. Refer to **SECTION 4 EXTERNAL MULTIPLEXING** for more information on external multiplexing.

Table 7-8 shows the channel number assignments for the nonmultiplexed mode. **Table 7-9** shows the channel number assignments for the multiplexed mode.

Table 7-8 Nonmultiplexed Channel Assignments and Pin Designations

Nonmultiplexed Input Pins				**Channel Number in CCW CHAN Field**	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type**	**Binary**	**Decimal**
PQB0	AN0	—	Input	000000	0
PQB1	AN1	—	Input	000001	1
PQB2	AN2	—	Input	000010	2
PQB3	AN3	—	Input	000011	3
—	—	Invalid	—	0XXXXX	4 to 31
—	—	Reserved	—	10XXXX	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	AN52	—	Input/Output	110100	52
PQA1	AN53	—	Input/Output	110101	53
PQA2	AN54	—	Input/Output	110110	54
PQA3	AN55	ETRIG1	Input/Output	110111	55
PQA4	AN56	ETRIG2	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
V_{RL}	Low Ref	—	Input	111100	60
V_{RH}	High Ref	—	Input	111101	61
—	—	$V_{DDA}/2$	—	111110	62
—	—	End of Queue Code	—	111111	63

Table 7-9 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				**Channel Number in CCW CHAN Field**	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type**	**Binary**	**Decimal**
PQB0	ANw	—	Input	00xxx0	0 to 14 even
PQB1	ANx	—	Input	00xxx1	1 to 15 odd
PQB2	ANy	—	Input	01xxx0	16 to 30 even
PQB3	ANz	—	Input	01xxx1	17 to 31 odd
—	—	Reserved	—	10xxxx	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	—	MA0	Input/Output	110100	52
PQA1	—	MA1	Input/Output	110101	53
PQA2	—	MA2	Input/Output	110110	54
PQA3	AN55	ETRIG1	Input/Output	110111	55
PQA4	AN56	ETRIG2	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
V_{RL}	Low Ref	—	Input	111100	60
V_{RH}	High Ref	—	Input	111101	61
—	—	$V_{DDA}/2$	—	111110	62
—	—	End of Queue Code	—	111111	63

If the CCW channel field specifies a reserved channel number (channels 32 to 47) or an invalid channel number (channels 4 to 31 in the non-multiplexed mode), the low reference level (V_{RL}) is converted. The channel field is programmed for channel 63 to indicate the end of the queue. Channels 60 to 63 are special internal channels. When one of the special channels is selected, the sampling amplifier is not used. The value of V_{RL}, V_{RH}, or $V_{DDA}/2$ is placed directly onto the converter. Also for the internal special channels, programming any input sample time other than two has no benefit except to lengthen the overall conversion time.

7.8 Result Word Table

The result word table is a RAM, 40 words long and 10 bits wide. An entry is written by the QADC after completing an analog conversion specified by the corresponding CCW table entry. Software can read or write the result word table, but in normal operation, the software reads the result word table to obtain analog conversions from the QADC. Unimplemented bits are read as zeros, and write operations do not have any effect.

While there is only one result word table, the data can be accessed in three different data formats:

- Right justified in the 16-bit word, with zeros in the higher order unused bits.
- Left justified, with the most significant bit inverted to form a sign bit, and zeros in the unused lower order bits.
- Left justified, with zeros in the lower order unused bits.

The left justified, signed format corresponds to a half-scale, offset binary, two's complement data format. The data is routed onto the IMB according to the selected format. The address used to access the table determines the data alignment format. All write operations to the result word table are right justified.

RJURR — Right Justified, Unsigned Result Format **$####B0–$####FE**

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">NOT USED</td><td colspan="10">RESULT</td></tr>
</table>

LJSRR — Left Justified, Signed Result Format **$####30–$####7E**

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>S[1]</td><td colspan="9">RESULT</td><td colspan="6">NOT USED</td></tr>
</table>

NOTES:

[1] S = Sign bit.

LJURR — Left Justified, Unsigned Result Register **$####B0–$####FE**

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="10">RESULT</td><td colspan="6">NOT USED</td></tr>
</table>

The three result data formats are produced by routing the RAM bits onto the data bus. The software chooses among the three formats by reading the result at the memory address which produces the desired data alignment.

The result word table is read/write accessible by software. During normal operation, applications software only needs to read the result table. Write operations to the table may occur during test or debug breakpoint operation. When locations in the CCW table are not used by an application, software could use the corresponding locations in the result word table as scratch pad RAM, remembering that only 10 bits are implemented. The result alignment is only implemented for software read operations. Since write operations are not the normal use for the result registers, only one write data format is supported, which is right justified data.

NOTE

Some write operations, like bit manipulation, may not operate as expected because the hardware cannot access a true 16-bit value.

SECTION 8 INTERRUPTS

Interrupt recognition and servicing involve interaction between the integration module, the CPU, and the module requesting interrupt service. This section provides an overview of the QADC interrupt process. Polled operation, an alternative to using interrupts, is discussed along with the different aspects of interrupt operation.

An interrupt is a special form of exception processing. Interrupt requests can be generated on-chip, or can come from external sources. However, the CPU services all interrupt requests as though originated by an on-chip module — to the CPU, an external interrupt request appears to come from the integration module. There are schemes to prioritize all interrupt requests and to arbitrate between simultaneous requests of the same priority.

8.1 Interrupt Operation

Figure 8-1 displays the QADC interrupt flow.

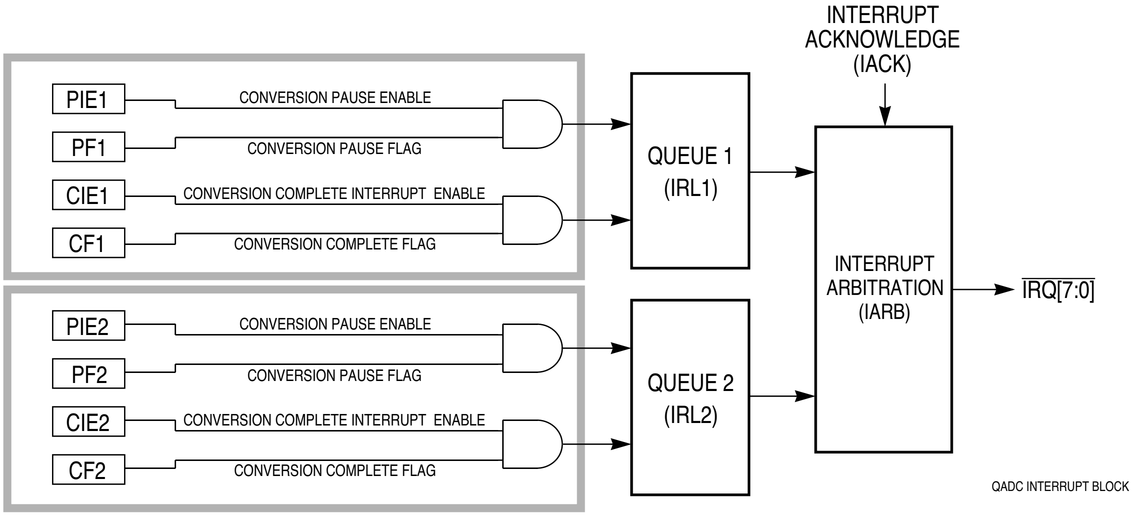


Figure 8-1 QADC Interrupt Flow Diagram

8.2 Polled and Interrupt-Driven Operation

QADC inputs can be monitored by polling or by using interrupts. When interrupts are not needed, software can disable the pause and completion interrupts, and monitor the completion flag and the pause flag for each queue in the status register (QASR). In other words, flag bits can be polled to determine when new results are available.

Table 8-1 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity.

Table 8-1 QADC Status Flags and Interrupt Sources

Queue	Queue Activity	Status Flag	Interrupt Enable Bit
Queue 1	Result written to last CCW in Queue 1	CF1	CF1
	Result written for a CCW with pause bit set in Queue 1	PF1	PF1
Queue 2	Result written to last CCW in Queue 2	CF2	CF2
	Result written for a CCW with pause bit set in Queue 2	PF2	PF2

If interrupts are enabled for an event, the QADC requests interrupt service when the event occurs. Using interrupts does not require continuously polling the status flags to see if an event has taken place. However, status flags must be cleared after an interrupt is serviced, in order to disable the interrupt request.

In both polled and interrupt-driven operating modes, status flags must be re-enabled after an event occurs. Flags are re-enabled by clearing appropriate QASR bits in a particular sequence. The register must first be read, then zeros must be written to the flags that are to be cleared. If a new event occurs between the time that the register is read and the time that it is written, the associated flag is not cleared.

8.3 Interrupt Sources

The QADC includes four sources of interrupt service requests, each of which is separately enabled. Each time the result is written for the last conversion command word (CCW) in a queue, the completion flag for the corresponding queue is set, and when enabled, an interrupt request is generated. In the same way, each time the result is written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt request is generated. Refer to **Table 8-1**.

The pause and complete interrupts for queue 1 and queue 2 have separate interrupt vector numbers, so that each source can be separately serviced.

8.4 QADC Interrupt Register

QADCINT specifies the priority level of QADC interrupt requests and the vector provided during an interrupt acknowledge cycle. The interrupt level for queue 1 and queue 2 may be different. The interrupt register is read/write accessible in supervisor data space only. The implemented interrupt register fields can be read and written, reserved locations read zero and writes have no effect. They are typically written once when the software initializes the QADC, and not changed afterwards.

QADCINT — QADC Interrupt Register **$####04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
–[1]	IRL1			–	IRL2			IVB						IVB[1:0][2]	
RESET:															
	0	0	0		0	0	0	0	0	0	0	1	1		

NOTES:
1. Reserved
2. Bits 1 and 0 supplied by the QADC.

IRL1[14:12] — Queue 1 Interrupt Level

The IRL1 field establishes the queue 1 interrupt request level. The 000 state disables queue 1 interrupts by blocking interrupt requests. When queue 1 interrupts are to be used, the IRL1 field must be initialized to a non-zero value. Level 001 is the lowest priority software interrupt request and level 111 is the highest priority request.

IRL2[10:8] — Queue 2 Interrupt Level

The IRL2 field establishes the queue 2 interrupt request level. The 000 state disables queue 2 interrupts by blocking interrupt requests. When queue 2 interrupts are to be used, the IRL2 field must be initialized to a non-zero value. Level 001 is the lowest priority interrupt request and level 111 is the highest priority request.

IVB[7:0] — Interrupt Vector Base

Initialization software inputs the upper six IVB bits in the interrupt register. During interrupt arbitration, the vector provided to the CPU by the QADC is made up of the upper six IVB bits, plus two low-order bits provided to the QADC to identify one of four QADC interrupt requests. The interrupt vector number is independent of the interrupt level and the interrupt arbitration number. A $0F vector number corresponds to the uninitialized interrupt vector. After reset, the lower byte of the interrupt register reads as $0F. Once the IVB field is written, the two least significant bits always read as zeros.

8.5 Interrupt Priority

The CPU16 and the CPU32 use an interrupt priority scheme based on that of the 68000 family. This scheme utilizes a three-bit interrupt priority mask that is located in the CPU16 condition code register and in the CPU32 status register. The interrupt priority mask can have eight possible values, from %000 to %111.

There are seven levels of interrupt priority, one to seven, each corresponding to a particular interrupt request signal. The CPU compares the priority of each interrupt service request to the mask value. Interrupt request levels greater than the mask value are accepted; interrupt request levels less than or equal to the mask value are ignored, except for the nonmaskable level seven interrupt request, which is serviced even if the CPU interrupt mask value is seven.

The values contained in the IRL1 and IRL2 fields in the interrupt register (QADCINT) determine the priority of QADC interrupt service requests. A value of %000 in either field disables the interrupts associated with that field.

IRL1 determines the priority of both queue 1 interrupt sources. IRL2 determines the priority of both queue 2 interrupt sources. As a result, queue 1 and queue 2 can have different priorities in the overall interrupt hierarchy of the MCU. The QADC also has an internal interrupt request prioritization. Queue 1 interrupt requests are higher in priority than queue 2 requests, and completion flag requests are higher in priority than pause requests.

8.6 Interrupt Arbitration

After queue 1 or queue 2 issues an interrupt service request, the CPU performs an interrupt acknowledge cycle. During the interrupt acknowledge cycle, the CPU identifies the interrupt request level being acknowledged by placing it on the address bus. The QADC compares the acknowledged interrupt level with IRL1 and IRL2 values, and responds if the values match.

The same interrupt priority level can be assigned to more than one module. For example, the QADC and the queued serial module (QSM) can both be assigned priority five. If the QADC and the QSM request interrupt service simultaneously, then the interrupt arbitration (IARB) fields in the respective module configuration registers are used to determine which module is serviced first.

The IARB field is essentially a second-level priority in case of a tie. Each module that can request interrupt service has an IARB field. Arbitration is performed by means of serial contention of IARB field bit values. Arbitration always takes place, even when a single source requests service.

IARB fields contain four bits. An IARB value of $1111 has the highest arbitration priority, and %0001 has the lowest. If a module with an IARB field value of %0000 requests interrupt service, the CPU processes a spurious interrupt exception because the module requesting the interrupt service cannot confirm that it made the request. Initialization software must assign each IARB field a unique non-zero value in order to implement the arbitration scheme. If two or more modules are assigned the same non-zero IARB field value, operation is undefined when interrupts of the same priority level are recognized.

8.7 Interrupt Vectors

When the QADC is the only module with an interrupt request pending at the level being acknowledged, or when the QADC IARB value is higher than that of other modules with requests pending at the acknowledged level, the QADC responds to the interrupt acknowledge cycle with an 8-bit interrupt vector number. The CPU uses the vector number to calculate a displacement into the exception vector table, then uses the vector at that location to jump to an interrupt service routine.

The interrupt vector base (IVB) field establishes the six high-order bits of the 8-bit interrupt vector number, and the QADC provides two low-order bits which correspond to one of the four internal QADC interrupt sources.

Figure 8-2 shows the format of the interrupt vector, and lists the binary coding of the two low-order bits for the four QADC interrupt sources.

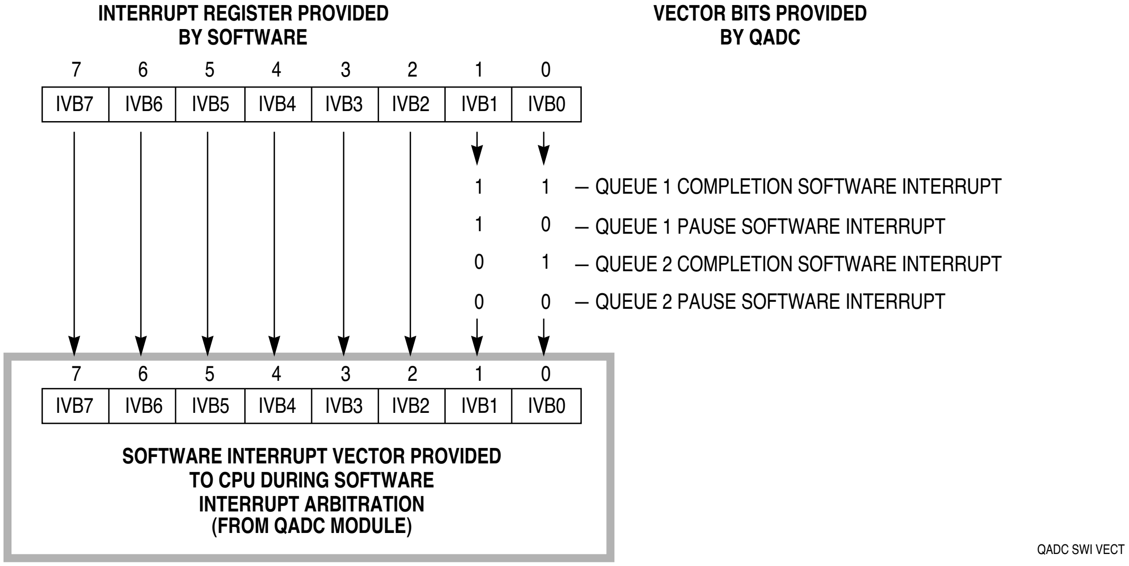

INTERRUPT REGISTER PROVIDED BY SOFTWARE
VECTOR BITS PROVIDED BY QADC
7 6 5 4 3 2 1 0
IVB7 IVB6 IVB5 IVB4 IVB3 IVB2 IVB1 IVB0
1 1 — QUEUE 1 COMPLETION SOFTWARE INTERRUPT
1 0 — QUEUE 1 PAUSE SOFTWARE INTERRUPT
0 1 — QUEUE 2 COMPLETION SOFTWARE INTERRUPT
0 0 — QUEUE 2 PAUSE SOFTWARE INTERRUPT
7 6 5 4 3 2 1 0
IVB7 IVB6 IVB5 IVB4 IVB3 IVB2 IVB1 IVB0
SOFTWARE INTERRUPT VECTOR PROVIDED TO CPU DURING SOFTWARE INTERRUPT ARBITRATION (FROM QADC MODULE)
QADC SWI VECT

Figure 8-2 QADC Interrupt Vector Format

After reset, and before the IVB field is initialized by the software, the IVB field has a default value of $0F, which corresponds to the uninitialized interrupt exception vector.

8.8 Initializing the QADC for Interrupt Driven Operation

The following steps are required to ensure proper operation of QADC interrupt processes.

1. Set the three-bit interrupt priority mask. This mask is located in the CPU32 status register or the CPU16 condition code register.
2. Assign the QADC a unique non-zero IARB value. The IARB field is located in the QADCMCR. The lowest priority IARB value is %0001, and the highest priority IARB value is %1111.
3. Set the interrupt request levels for queue 1 and queue 2 in the IRL1 and IRL2 bit fields, located in the QADCINT. They should be higher than the interrupt mask value in order for the interrupt requests to be recognized. Level 001 is the lowest priority interrupt request, and level 111 is the highest priority request.
4. Establish the six high-order bits of the eight-bit IVB field in the QADCINT. The QADC provides the two low-order vector bits to identify one of four QADC interrupt requests.
5. Load each interrupt service routine. The IVB number for each QADC interrupt request corresponds to a specific vector in the exception vector table. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) or the *CPU32 Reference Manual* (CPU32RM/AD) for information on locating an entry in the exception vector table given a vector number. Each vector in the exception vector table points to the beginning address of an exception handler routine.

8.9 Interrupt Processing Summary

A summary of the entire interrupt processing sequence follows.

1. A result is written for the last CCW in a queue and the completion flag for the corresponding queue is set, and when enabled, an interrupt request is generated. Similarly, each time a result is written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt request is generated.
2. QADC asserts the internal interrupt request line corresponding to the level programmed in the IRL1 and IRL2 field in the QADCINT.
3. An interrupt acknowledge cycle is performed on the IMB, during which time the CPU identifies the highest interrupt level requesting service. The CPU compares the level of each interrupt request it receives with the mask value. Interrupt request levels greater than the mask are accepted; interrupt request levels less than or equal to the mask are ignored.
4. If multiple modules simultaneously assert the same interrupt request level, the values in the IARB field in the respective module configuration registers determine which module is serviced first.
5. When the QADC is the only module with an interrupt pending at the level being acknowledged, or when the QADC IARB value is higher than the modules with requests pending at the given interrupt level, the QADC responds to the interrupt acknowledge cycle by providing an 8-bit vector number which the CPU uses to call an interrupt service routine.
6. The CPU calculates displacement into the exception vector table using the vector number supplied by the QADC, then jumps to the service routine pointed to by the vector.
7. The subroutine services the interrupt. As part of interrupt servicing, it must clear the pause or completion flag by writing a zero to the flag bit after reading the flag.

SECTION 9 QUEUE PRIORITY EXAMPLES

The following section describes the QADC priority scheme when trigger events on two queues overlap or conflict.

9.1 Queue Priority Schemes

Since there are two conversion command queues and only one A/D converter, there is a priority scheme to determine which conversion is to occur. Each queue has a variety of trigger events that are intended to initiate conversions, and they can occur asynchronously in relation to each other and other conversions in progress. For example, a queue can be idle awaiting a trigger event, a trigger event can have occurred, but the first conversion has not started, a conversion can be in progress, a pause condition can exist awaiting another trigger event to continue the queue, and so on.

The following paragraphs and figures outline the prioritizing criteria used to determine which conversion occurs in each overlap situation.

NOTE

> Each situation in **Figure 9-1** through **Figure 9-19** are labeled S1 through S19. In each diagram, time is shown increasing from left to right. The execution of queue 1 and queue 2 (Q1 and Q2) is shown as a string of rectangles representing the execution time of each CCW in the queue. In most of the situations, there are four CCWs (labeled C1 to C4) in both queue 1 and queue 2. In some of the situations, CCW C2 is presumed to have the pause bit set, to show the similarities of pause and end-of-queue as terminations of queue execution.

Trigger events are described in **Table 9-1**.

Table 9-1 Trigger Events

Trigger	Events
T1	Events that trigger queue 1 execution (external trigger, software initiated single-scan enable bit, or completion of the previous continuous loop)
T2	Events that trigger queue 2 execution (external trigger, software initiated single-scan enable bit, timer period/interval expired, or completion of the previous continuous loop)

When a trigger event causes a CCW execution in progress to be aborted, the aborted conversion is shown as a ragged end of a shortened CCW rectangle.

The situation diagrams also show when key status bits are set. **Table 9-2** describes the status bits.

Table 9-2 Status Bits

Bit	Function
CF Flag	Set when the end of the queue is reached
PF Flag	Set when a queue completes execution up through a pause bit
Trigger Overrun Error (TOR)	Set when a new trigger event occurs before the queue is finished serving the previous trigger event

Below the queue execution flows are three sets of blocks that show the status information that is made available to the software. The first two rows of status blocks show the condition of each queue as:

- Idle
- Active
- Pause
- Suspended
- Trigger Pending

The third row of status blocks shows the 4-bit QS status register field that encodes the condition of the two queues. Two transition status cases, QS = 0011 and QS = 0111, are not shown because they exist only very briefly between stable status conditions.

The first three examples in **Figure 9-1**, **Figure 9-2**, and **Figure 9-3** (S1, S2, and S3) show what happens when a new trigger event is recognized before the queue has completed servicing the previous trigger event on the same queue.

In situation S1 (**Figure 9-1**), one trigger event is being recognized on each queue while that queue is still working on the previously recognized trigger event. The trigger overrun error status bit is set, and otherwise, the premature trigger event is ignored. A trigger event which occurs before the servicing of the previous trigger event is through, does not disturb the queue execution in progress.

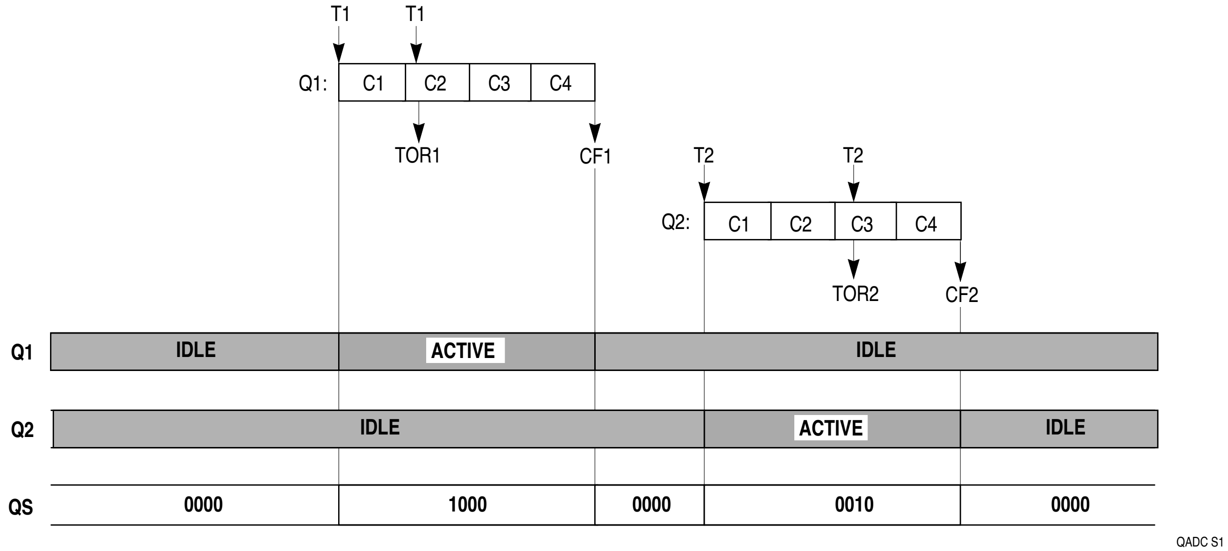


Figure 9-1 CCW Priority Situation 1

In situation S2 (**Figure 9-2**), more than one trigger event is recognized before servicing of a previous trigger event is complete, the trigger overrun bit is again set, but otherwise, the additional trigger events are ignored. After the queue is complete, the first newly detected trigger event causes queue execution to begin again. When the trigger event rate is high, a new trigger event can be seen very soon after completion of the previous queue, leaving software little time to retrieve the previous results. Also, when trigger events are occurring at a high rate for queue 1, the lower priority queue 2 channels may not get serviced at all.

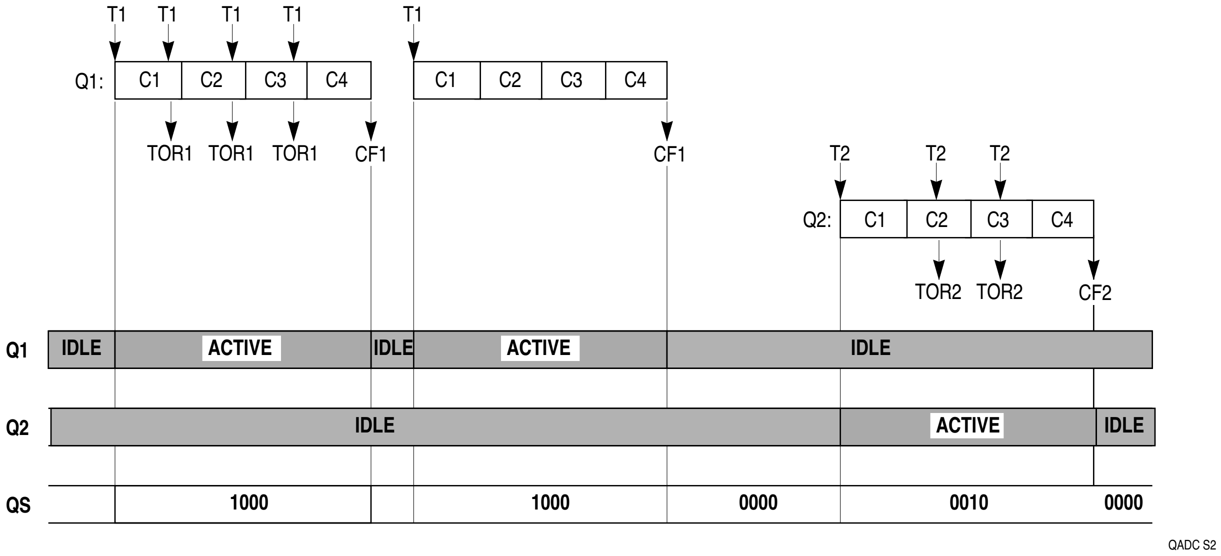


Figure 9-2 CCW Priority Situation 2

Figure 9-3 shows that when the pause feature is in use, the trigger overrun error status bit is set the same way, and that queue execution continues unchanged.

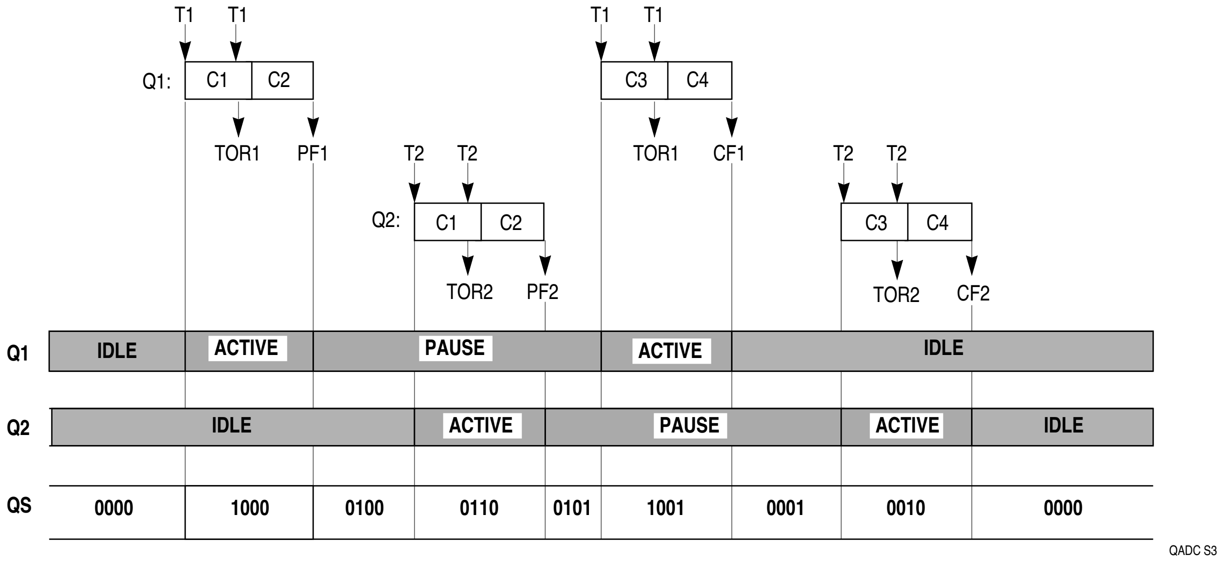


Figure 9-3 CCW Priority Situation 3

The next two situations consider trigger events that occur for the lower priority queue 2, while queue 1 is actively being serviced.

Figure 9-4 shows that a queue 2 trigger event that is recognized while queue 1 is active, is saved, and as soon as queue 1 is finished, queue 2 servicing begins.

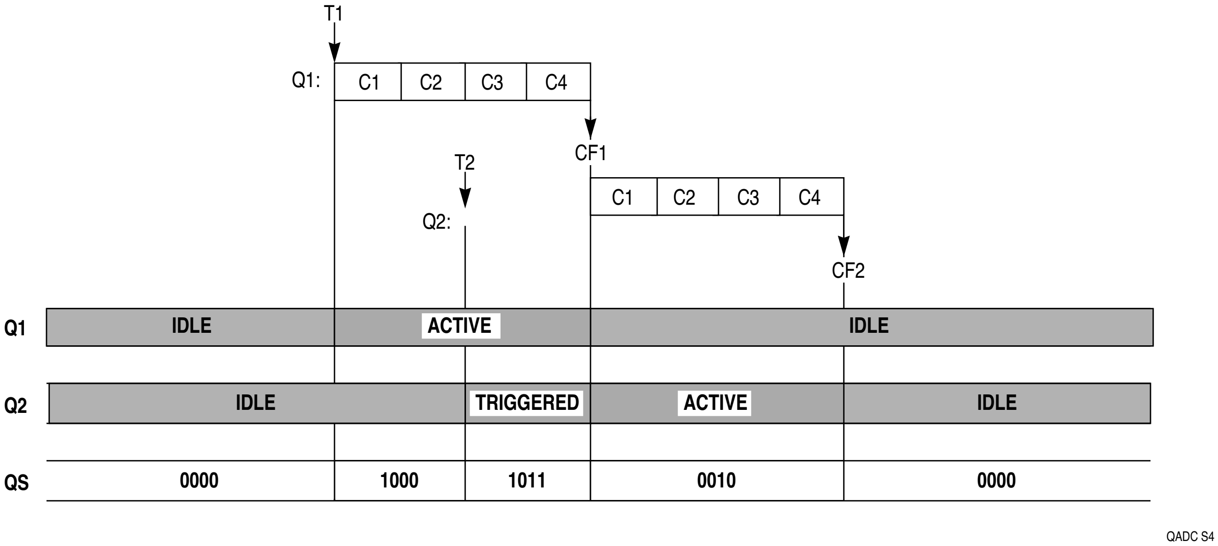


Figure 9-4 CCW Priority Situation 4

Situation S5 (**Figure 9-5**) shows that when multiple queue 2 trigger events are detected while queue 1 is busy, the trigger overrun error bit is set, but queue 1 execution is not disturbed. Situation S5 also shows that the effect of queue 2 trigger events during queue 1 execution is the same when the pause feature is in use in either queue.

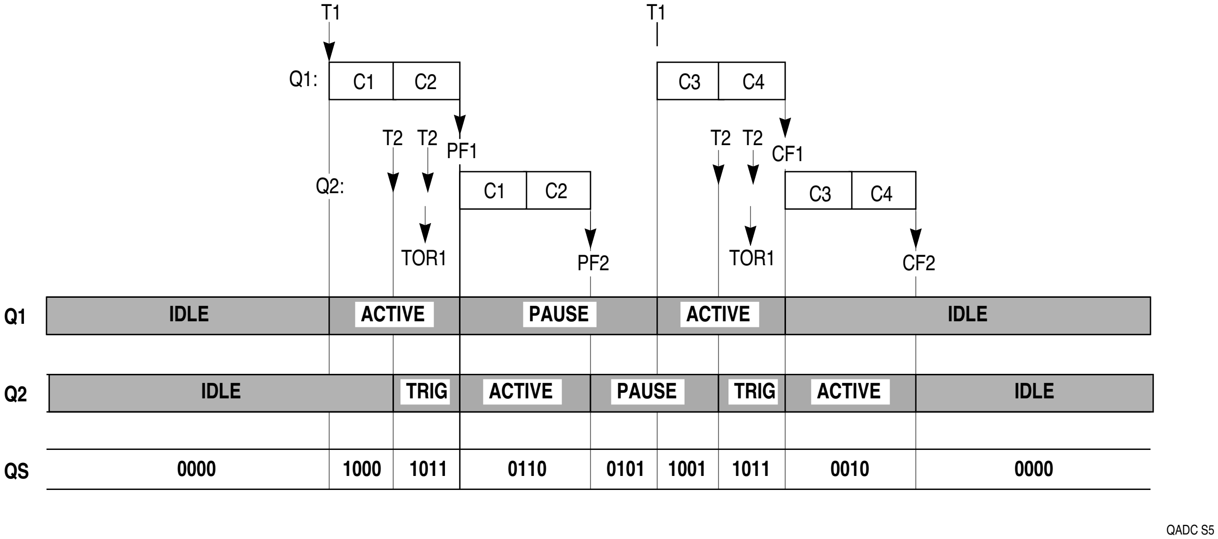


Figure 9-5 CCW Priority Situation 5

The remaining situations, S6 through S11, show the impact of a queue 1 trigger event occurring during queue 2 execution. Queue 1 is higher in priority, the conversion taking place in queue 2 is aborted, so that there is not a variable latency time in responding to queue 1 trigger events.

In situation 6 (**Figure 9-6**), the conversion initiated by the second CCW in queue 2 is aborted just before the conversion is complete, so that queue 1 execution can begin. Queue 2 is considered suspended. After queue 1 is finished, queue 2 starts over with the first CCW, when the RES (resume) control bit is set to 0. Situation S7 (**Figure 9-7**) shows that when pause operation is not in use with queue 2, queue 2 suspension works the same way.

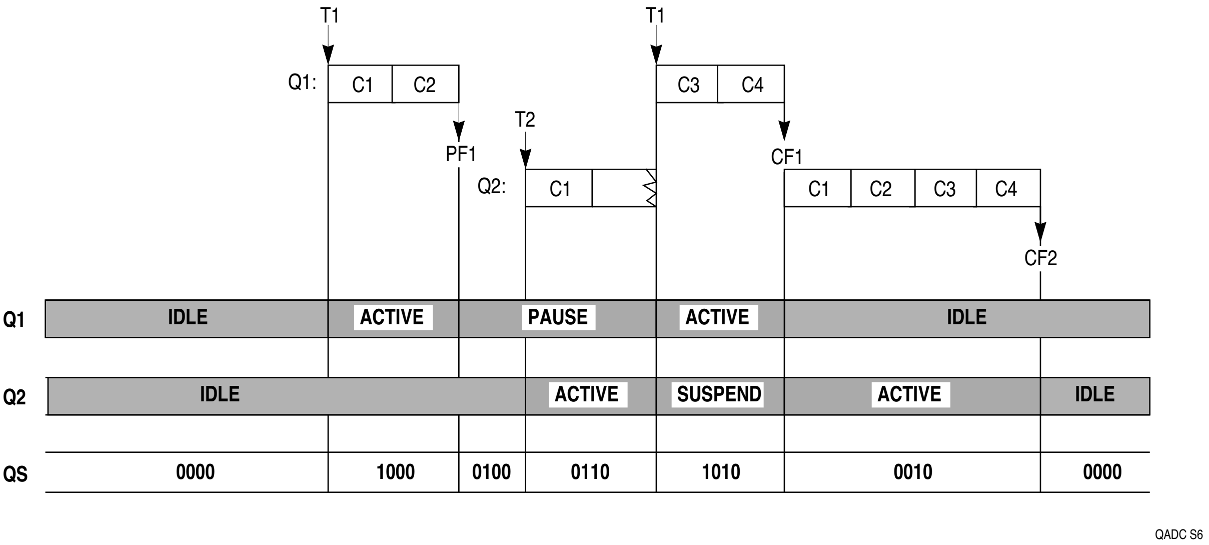


Figure 9-6 CCW Priority Situation 6

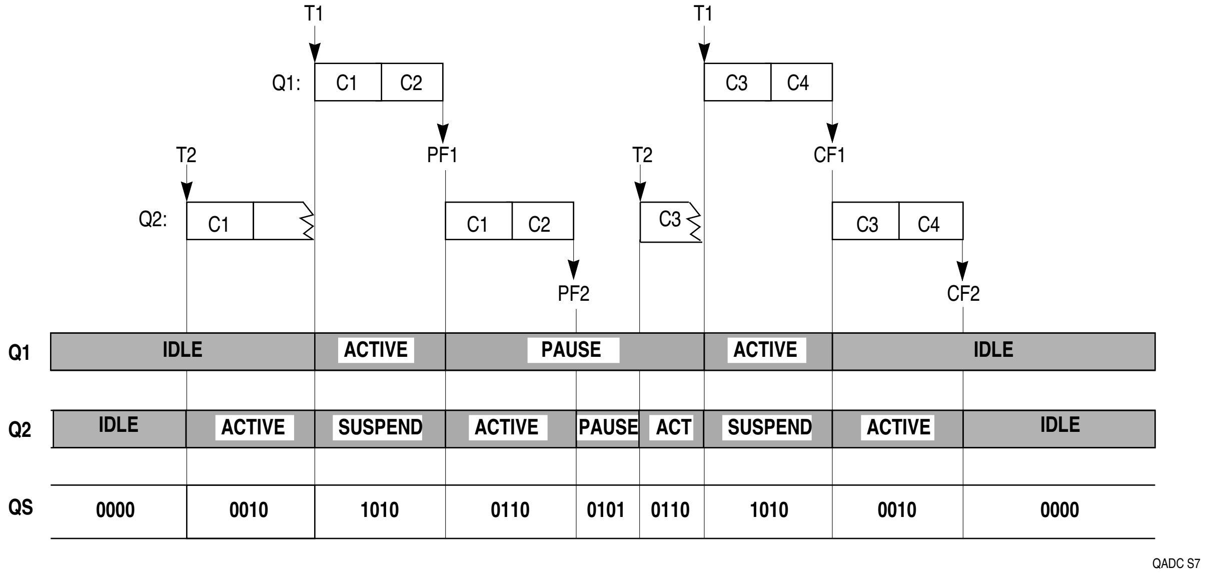


Figure 9-7 CCW Priority Situation 7

Situations S8 and S9 (**Figure 9-8** and **Figure 9-9**) repeat the same two situations with the resume bit set to a one. When the RES bit is set, following suspension, queue 2 resumes execution with the aborted CCW, not the first CCW in the queue.

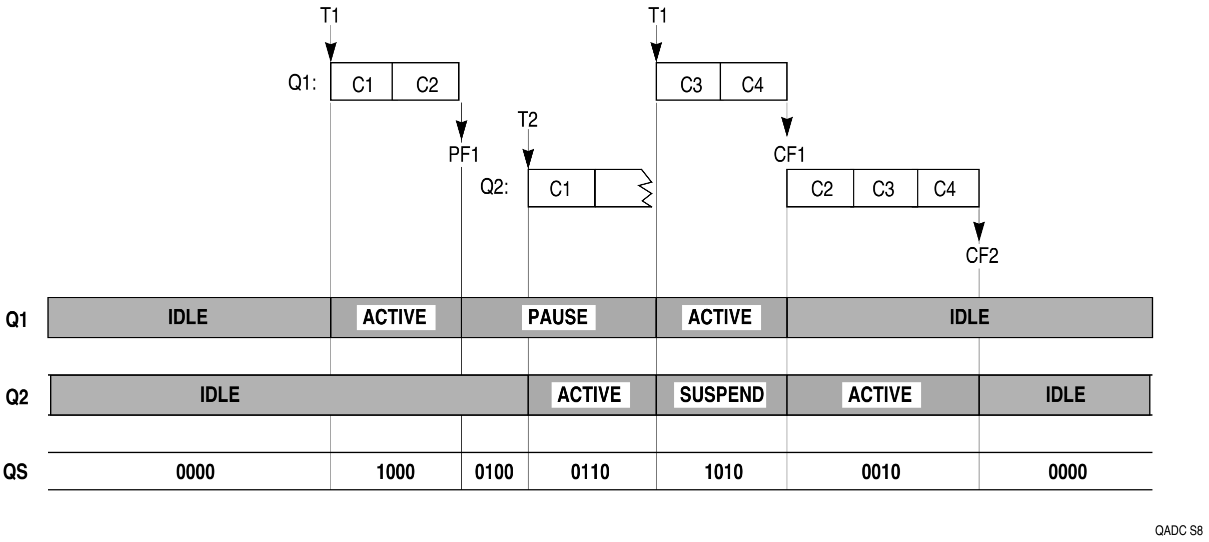


Figure 9-8 CCW Priority Situation 8

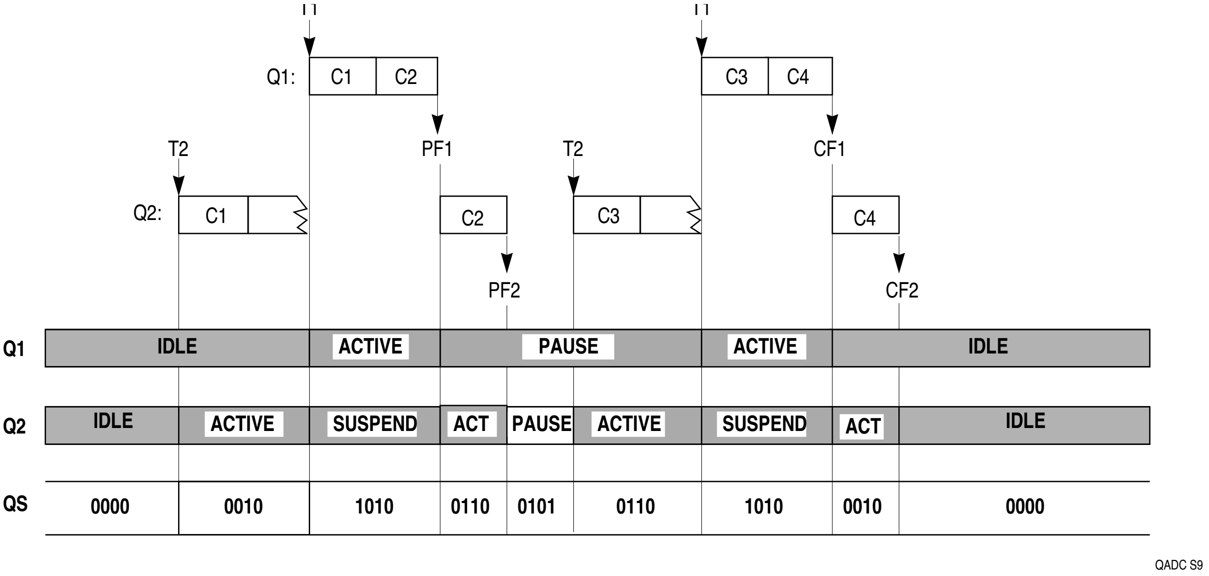


Figure 9-9 CCW Priority Situation 9

Situations S10 and S11 (**Figure 9-10** and **Figure 9-11**) show that when an additional trigger event is detected for queue 2 while the queue is suspended, the trigger overrun error bit is set, the same as if queue 2 were being executed when a new trigger event occurs. Trigger overrun on queue 2 thus permits the software to know that queue 1 is taking up so much QADC time that queue 2 trigger events are being lost.

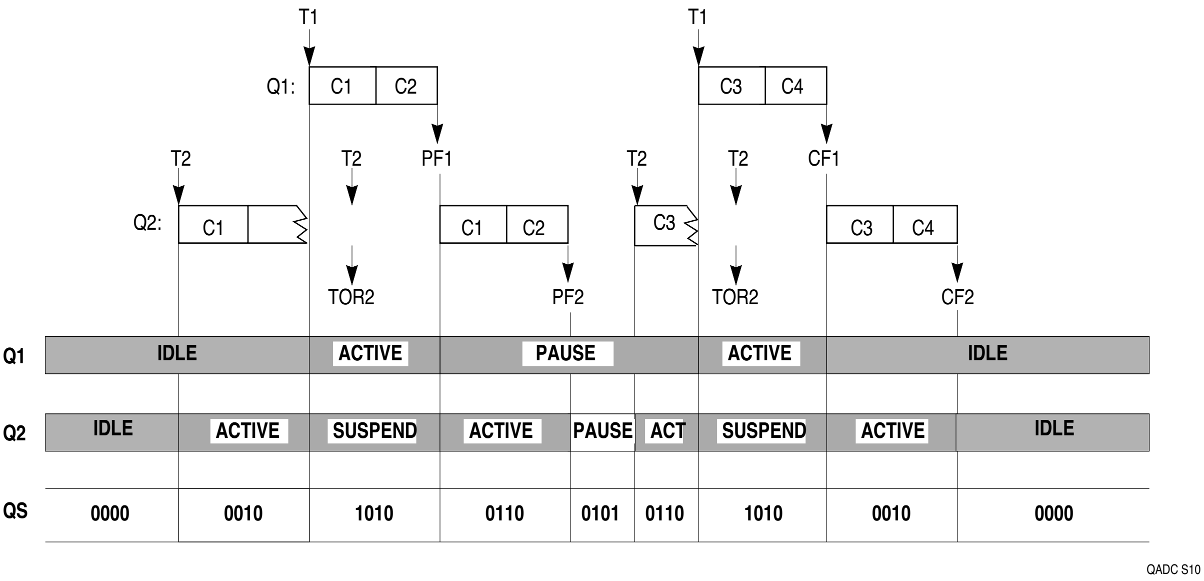


Figure 9-10 CCW Priority Situation 10

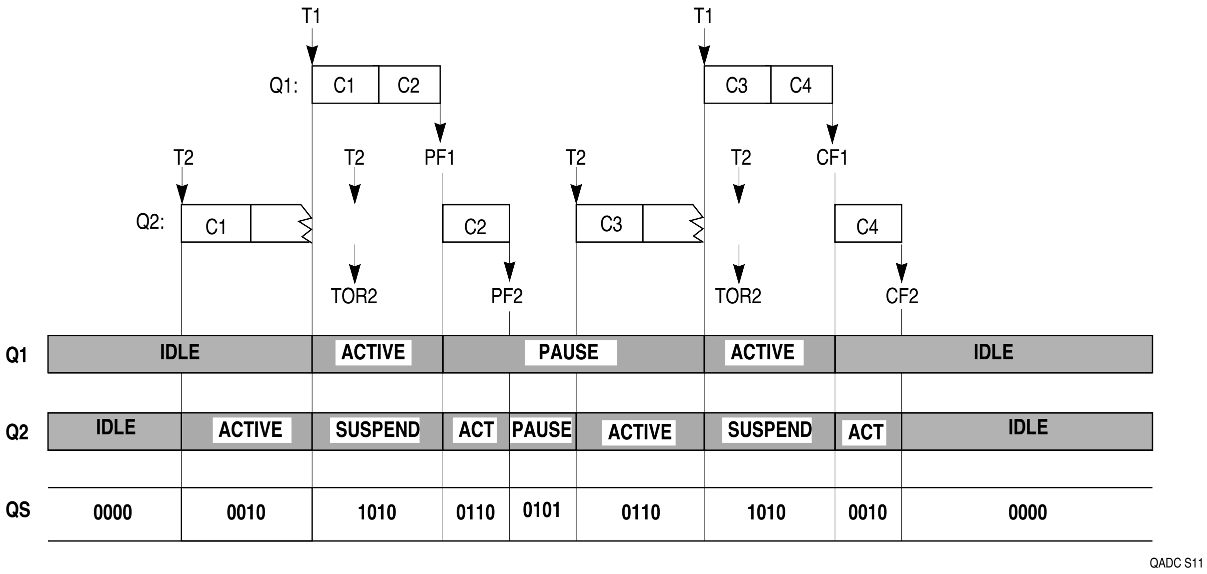


Figure 9-11 CCW Priority Situation 11

The above situations cover normal overlap conditions that arise with asynchronous trigger events on the two queues. An additional conflict to consider is that the freeze condition can arise while the QADC is actively executing CCWs. The conventional use for the freeze mode is for software/hardware debugging. When the CPU background debug mode is enabled and a breakpoint occurs, the freeze signal is issued, which can cause peripheral modules to stop operation. When freeze is detected, the QADC completes the conversion in progress, unlike queue 1 suspending queue 2. After the freeze condition is removed, the QADC continues queue execution with the next CCW in sequence.

When freeze occurs between the conversion of two simultaneous samples, the second value is not a valid result. When the software is correlating successive samples from one scan in any other way, the freeze condition can disrupt measurements. For example, when the software is averaging multiple samples of one channel to produce a more accurate conversion result, if freeze were to occur before the consecutive conversions are complete, the average value would span the breakpoint interval.

Trigger events that occur during freeze are not recorded. When a trigger event is pending for queue 2 before freeze begins, that trigger event is remembered when the freeze is passed. Similarly, when freeze occurs while queue 2 is suspended, after freeze, queue 2 resumes execution as soon as queue 1 is finished.

Situations 12 through 19 (**Figure 9-12** to **Figure 9-19**) show examples of all of the freeze situations.

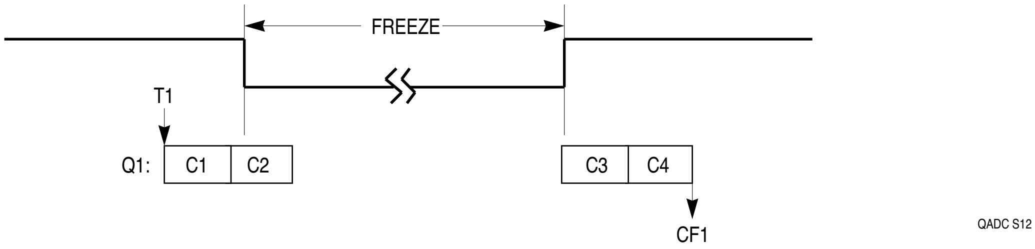


Figure 9-12 CCW Freeze Situation 12

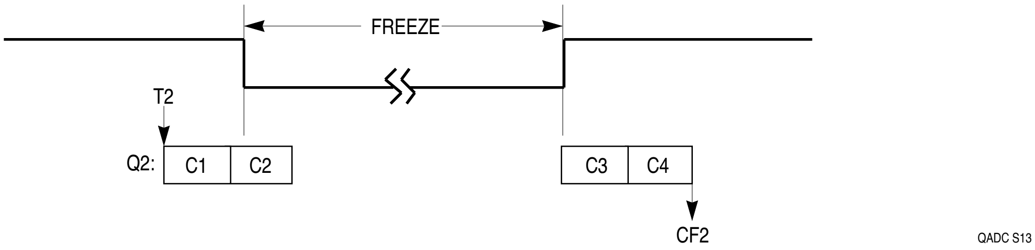


Figure 9-13 CCW Freeze Situation 13

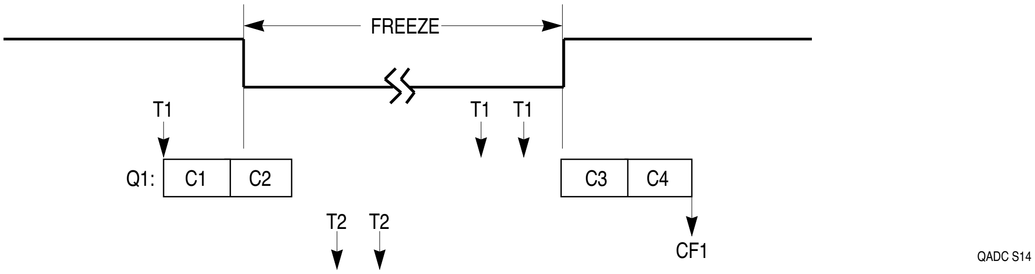


Figure 9-14 CCW Freeze Situation 14

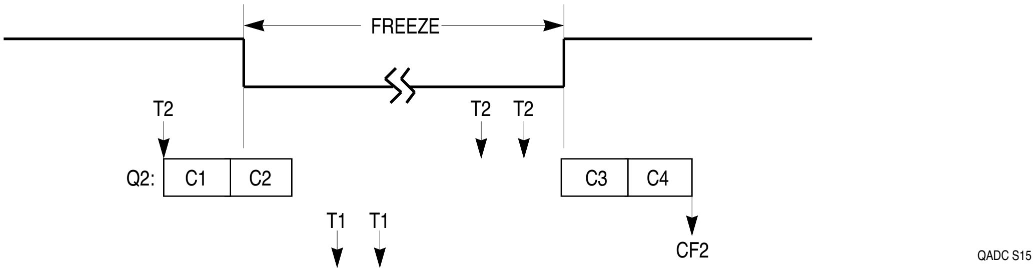


Figure 9-15 CCW Freeze Situation 15

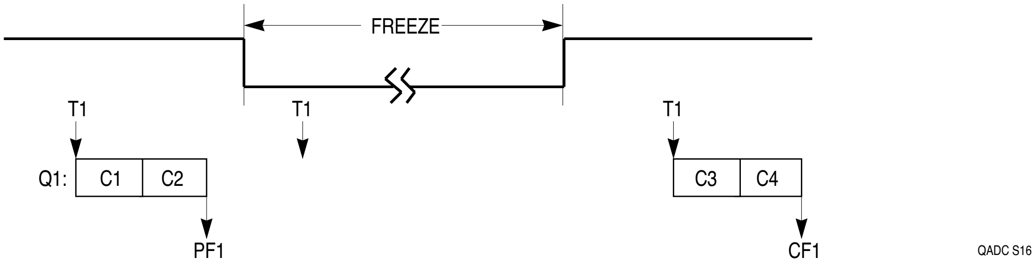


Figure 9-16 CCW Freeze Situation 16

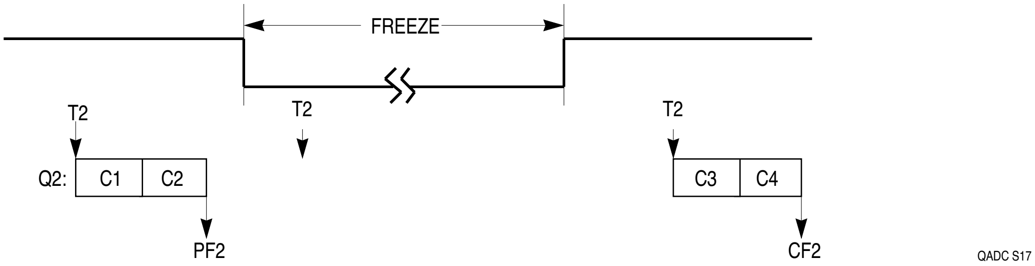


Figure 9-17 CCW Freeze Situation 17

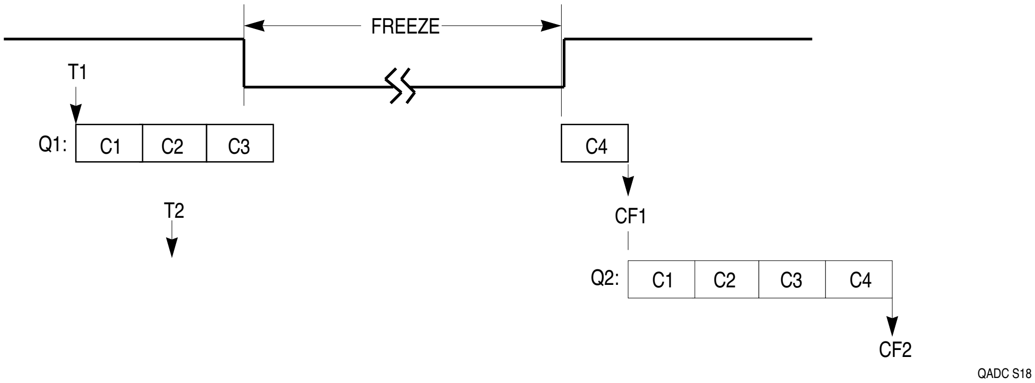


Figure 9-18 CCW Freeze Situation 18

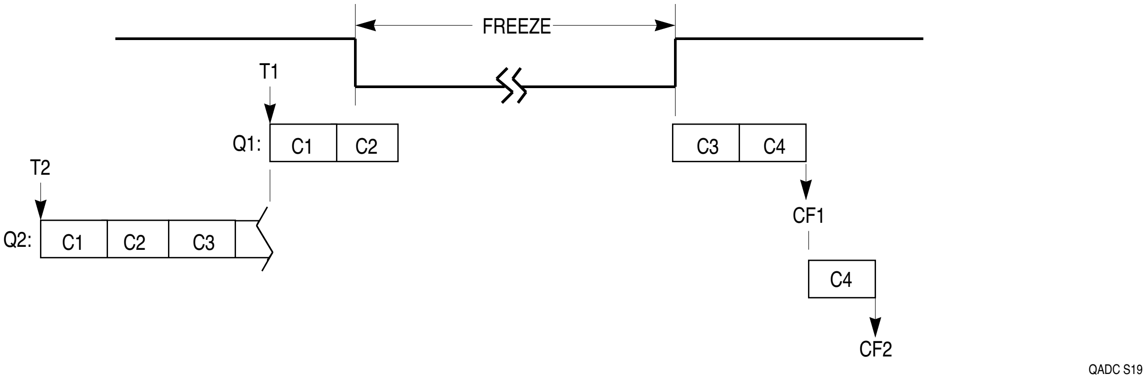


Figure 9-19 CCW Freeze Situation 19

APPENDIX A ELECTRICAL CHARACTERISTICS

Table A-1 QADC Maximum Ratings

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply, with reference to V_{SSA}	V_{DDA}	– 0.3	6.5	V
2	Internal Digital Supply, with reference to V_{SSI}	V_{DDI}	– 0.3	6.5	V
3	Reference Supply, with reference to V_{RL}	V_{RH}	– 0.3	6.5	V
4	V_{SS} Differential Voltage	$V_{SSI} – V_{SSA}$	– 0.1	0.1	V
5	V_{DD} Differential Voltage	$V_{DDI} – V_{DDA}$	– 6.5	6.5	V
6	V_{REF} Differential Voltage	$V_{RH} – V_{RL}$	– 6.5	6.5	V
7	V_{RH} to V_{DDA} Differential Voltage	$V_{RH} – V_{DDA}$	– 6.5	6.5	V
8	V_{RL} to V_{SSA} Differential Voltage	$V_{RL} – V_{SSA}$	– 6.5	6.5	V
9	Disruptive Input Current[1, 2, 3, 4, 5, 6, 7] $V_{NEGCLAMP}$ = – 0.3 V $V_{POSCLAMP}$ = 8 V	I_{NA}	– 500	500	µA
10	Positive Overvoltage Current Coupling Ratio[1, 5, 6, 8] PQA PQB	K_P	 2000 2000	—	—
11	Negative Overvoltage Current Coupling Ratio[1, 5, 6, 8] PQA PQB	K_N	 125 500	—	—
12	Maximum Input Current[3, 4, 6] $V_{NEGCLAMP}$ = – 0.3 V $V_{POSCLAMP}$ = 8 V	I_{MA}	– 25	25	mA

NOTES:

1. Below disruptive current conditions, the channel being stressed has conversion values of $3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \le V_{DDA}$ and $V_{RL} \ge V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
2. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals also affect the conversion accuracy of other channels.
3. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
4. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using positive and negative clamp values, then use the larger of the calculated values.
5. This parameter is periodically sampled rather 100% tested.
6. Condition applies to one pin at a time.
7. Determination of actual maximum disruptive input current, which can affect operation, is related to external system component values.
8. Current coupling is the ratio of the current induced from overvoltage (positive or negative, through an external series coupling resistor), divided by the current induced on adjacent pins. A voltage drop may occur across the external source impedances of the adjacent pins, impacting conversions on these adjacent pins.

Table A-2 QADC DC Electrical Characteristics (Operating)

(V_{SSI} and V_{SSA} = 0Vdc, F_{QCLK} = 2.1 MHz, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply[1]	V_{DDA}	4.5	5.5	V
2	Internal Digital Supply[1]	V_{DDI}	4.5	5.5	V
3	V_{SS} Differential Voltage	$V_{SSI} – V_{SSA}$	– 1.0	1.0	mV
4	V_{DD} Differential Voltage	$V_{DDI} – V_{DDA}$	– 1.0	1.0	V
5	Reference Voltage Low[2]	V_{RL}	V_{SSA}	—	V
6	Reference Voltage High[2]	V_{RH}	—	V_{DDA}	V
7	V_{REF} Differential Voltage[3]	$V_{RH} – V_{RL}$	4.5	5.5	V
8	Mid-Analog Supply Voltage	$V_{DDA}/2$	2.25	2.75	V
9	Input Voltage	V_{INDC}	V_{SSA}	V_{DDA}	V
10	Input High Voltage, PQA and PQB	V_{IH}	0.7 (V_{DDA})	V_{DDA} + 0.3	V
11	Input Low Voltage, PQA and PQB	V_{IL}	V_{SSA} – 0.3	0.2 (V_{DDA})	V
12	Input Hysteresis[4]	V_{HYS}	0.5	—	V
13	Output Low Voltage, PQA[5] I_{OL} = 5.3 mA I_{OL} = 10.0 µA	V_{OL}	 — —	 0.4 0.2	V
14	Analog Supply Current Normal Operation[6] Low-Power Stop	I_{DDA}	 — —	 15.0 10.0	 mA µA
15	Reference Supply Current	I_{REF}	—	150	µA
16	Load Capacitance, PQA	C_L	—	90	pF
17	Input Current, Channel Off[7] PQA PQB	I_{OFF}	 — —	 250 150	nA
18	Total Input Capacitance[8] PQA Not Sampling PQA Sampling PQB Not Sampling PQB Sampling	C_{IN}	 — — — —	 15 20 10 15	pF

NOTES:

1. Refers to operation over full temperature and frequency range.
2. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
3. Accuracy tested and guaranteed at $V_{RH} – V_{RL}$ = 5.0V ± 10%.
4. Parameter applies to the following pins:
   Port A: PQA[7:0]/AN[59:58]/ETRIG[2:1]
   Port B: PQB[7:0]/AN[3:0]/AN[51:48]/AN[Z:W]
5. Open drain only.
6. Current measured at maximum system clock frequency with QADC active.
7. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10° C decrease from maximum temperature.
8. This parameter is periodically sampled rather than 100% tested.

Table A-3 QADC AC Electrical Characteristics (Operating)

(V_{DDI} and V_{DDA} = 5.0 Vdc ± 5%, V_{SSI} and V_{SSA} = 0Vdc, T_A within operating temperature range.)

Num	Parameter	Symbol	Min	Max	Unit
1	QADC Clock (QCLK) Frequency[1]	F_{QCLK}	0.5	2.1	MHz
2	QADC Clock Duty Cycle[2, 3] High Phase Time ($T_{PSL} \le T_{PSH}$)	T_{PSH}	500	—	ns
3	Conversion Cycles[4]	CC	18	32	QCLK cycles
4	Conversion Time[2,4,5] F_{QCLK} = 0.999 MHz[6] Min = CCW/IST = %00 Max = CCW/IST = %11 F_{QCLK} = 2.097 MHz[1, 7] Min = CCW/IST = %00 Max = CCW/IST = %11	T_{CONV}	18.0 8.58	32 15.24	µs
5	Stop Mode Recovery Time	T_{SR}	—	10	µs

NOTES:

1. Conversion characteristics vary with F_{QCLK} rate. Reduced conversion accuracy occurs at max F_{QCLK} rate.
2. Duty cycle must be as close as possible to 75% to achieve optimum performance.
3. Minimum applies to 1.0 MHz operation.
4. Assumes that short input sample time has been selected (IST = 0).
5. Assumes that f_{SYS} = 20.97 MHz.
6. Assumes F_{QCLK} = 0.999 MHz, with clock prescaler values of:
   QACR0: PSH = %01111, PSA = %1, PSL = 100)
   CCW: BYP = %0
7. Assumes F_{QCLK} = 2.097 MHz, with clock prescaler values of:
   QACR0: PSH = %00110, PSA = %1, PSL = 010)
   CCW: BYP = %0

Table A-4 QADC Conversion Characteristics (Operating)

(V_{DDI} and V_{DDA} = 5.0 Vdc ± 5%, V_{SSI} and V_{SSA} = 0 Vdc, T_A within operating temperature range,
0.5 MHz ≤ F_{QCLK} ≤ 2.1 MHz, 2 clock input sample time)

Num	Parameter	Symbol	Min	Typ	Max	Unit
1	Resolution[1]	1 Count	—	5	—	mV
2	Differential nonlinearity[2]	DNL	—	—	± 0.5	Counts
3	Integral nonlinearity	INL	—	—	± 2.0	Counts
4	Absolute error[2, 3, 4]	AE				Counts
	F_{QCLK} = 0.999 MHz[5]					
	PQA		—	—	± 2.5	
	PQB		—	—	± 2.5	
	F_{QCLK} = 2.097 MHz[6]					
	PQA		—	—	± 4.0	
	PQB		—	—	± 4.0	
5	Source impedance at input[7]	R_S	—	20	—	kΩ

NOTES:

1. At $V_{RH} – V_{RL}$ = 5.12 V, one count = 5 mV.
2. This parameter is periodically sampled rather than 100% tested.
3. Absolute error includes 1/2 count (2.5 mV) of inherent quantization error and circuit (differential, integral, and offset) error. Specification assumes that adequate low-pass filtering is present on analog input pins — capacitive filter with 0.01 µF to 0.1 µF capacitor between analog input and analog ground, typical source isolation impedance of 20 KΩ.
4. Assumes f_{sys} = 20.97 MHz.
5. Assumes clock prescaler values of:
   QACR0: PSH = %01111, PSA = %1, PSL = 100)
   CCW: BYP = %0
6. Assumes clock prescaler values of:
   QACR0: PSH = %00110, PSA = %1, PSL = 010)
   CCW: BYP = %0
7. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.

   Error from junction leakage is a function of external source impedance and input leakage current. In the following expression, expected error in result value due to junction leakage is expressed in voltage (V_{errj}):

   $V_{errj} = R_S \text{ X } I_{OFF}$

   where I_{OFF} is a function of operating temperature. (See **Table A-2**, Note 7).

   Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the decoupling capacitor used. Error levels are best determined empirically. In general, continuous conversion of the same channel may not be compatible with high source impedance.

APPENDIX B REGISTER SUMMARY

B.1 Address Map

Table B-1 QADC Address Map

Access	Address	Offset	15 8	7 0
S[1]	$####00	$000	MODULE CONFIGURATION REGISTER (QADCMCR)	
S	$####02	$002	TEST REGISTER (QADCTEST)	
S	$####04	$004	INTERRUPT REGISTER (QADCINT)	
S/U[2]	$####06	$006	PORT A DATA (PORTQA)	PORT B DATA (PORTQB)
S/U	$####08	$008	PORT DATA DIRECTION REGISTER (DDRQA)	
S/U	$####0A	$00A	CONTROL REGISTER 0 (QACR0)	
S/U	$####0C	$00C	CONTROL REGISTER 1 (QACR1)	
S/U	$####0E	$00E	CONTROL REGISTER 2 (QACR2)	
S/U	$####10	$010	STATUS REGISTER (QASR)	
—	$####12– $####2E	$012– $02E	RESERVED	
S/U	$####30– $####7E	$030– $07E	CONVERSION COMMAND WORD (CCW) TABLE	
—	$####80– $####AE	$080– $0AE	RESERVED	
S/U	$####B0– $####FE	$0B0– $0FE	RESULT WORD TABLE RIGHT JUSTIFIED, UNSIGNED RESULT REGISTER (RJURR)	
—	$####00– $####2E	$100– $12E	RESERVED	
S/U	$####30– $####7E	$130– $17E	RESULT WORD TABLE LEFT JUSTIFIED, SIGNED RESULT REGISTER (LJSRR)	
—	$####80– $####AE	$180– $1AE	RESERVED	
S/U	$####B0– $####FE	$1B0– $1FE	RESULT WORD TABLE LEFT JUSTIFIED, UNSIGNED RESULT REGISTER (LJURR)	

NOTES:
1. S = Supervisor only
2. U = Unrestricted or supervisor depending on the state of the SUPV bit in the QADCMCR.

B.2 QADC Registers

The following section provides a summary of QADC registers and their content.

B.2.1 QADC Module Configuration Register

QADCMCR — Module Configuration Register **$####00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	NOT USED						SUPV	NOT USED			IARB			
RESET:															
0	0							1				0	0	0	0

STOP — Stop Enable
- 0 = Disable stop mode
- 1 = Enable stop mode

FRZ — Freeze Enable

0 = Ignores the IMB IFREEZE signal
1 = Finish any current conversion, then freeze

SUPV — Supervisor/Unrestricted Data Space

0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted.
1 = All QADC registers and tables are designated as supervisor-only data space.

IARB[3:0] — Software Interrupt Arbitration Number

IARB determines QADC interrupt arbitration priority. The reset value is $F (highest priority), to prevent QADC interrupts from being discarded during initialization.

B.2.2 QADC Test Register

QADCTEST — QADC Test Register **$####02**

QADCTEST is used only during factory testing of the MCU.

B.2.3 QADC Interrupt Register

QADCINT — QADC Interrupt Register **$####04**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—[1]	IRL1			—[1]	IRL2			IVB						IVB[1:0][2]	
RESET:																
		0	0	0		0	0	0	0	0	0	0	1	1		

NOTES:
1. Reserved
2. Bits 1 and 0 supplied by the QADC.

IRL1[14:12] — Queue 1 Interrupt Level

The IRL1 field establishes the queue 1 interrupt request level. The 000 state disables queue 1 interrupts by blocking interrupt requests. When queue 1 interrupts are to be used, the IRL1 field must be initialized to a non-zero value. Level 001 is the lowest priority software interrupt request and level 111 is the highest priority request.

IRL2[10:8] — Queue 2 Interrupt Level

The IRL2 field establishes the queue 2 interrupt request level. The 000 state disables queue 2 interrupts by blocking interrupt requests. When queue 2 interrupts are to be used, the IRL2 field must be initialized to a non-zero value. Level 001 is the lowest priority interrupt request and level 111 is the highest priority request.

IVB[7:0] — Interrupt Vector Base

Initialization software inputs the upper six IVB bits in the interrupt register. During interrupt arbitration, the vector provided to the CPU by the QADC is made up of the upper six IVB bits, plus two low-order bits provided to the QADC to identify one of four QADC interrupt requests. The interrupt vector number is independent of the interrupt level and the interrupt arbitration number. A $0F vector number corresponds to the uninitialized interrupt vector. After reset, the lower byte of the interrupt register reads as $0F. Once the IVB field is written, the two least significant bits always read as zeros.

B.2.4 Port A/B Data Register

PORTQA — Port A Data Register **$####06**
PORTQB — Port B Data Register **$####07**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PQA7	PQA6	PQA5	PQA4	PQA3	PQA2	PQA1	PQA0	PQB7	PQB6	PQB5	PQB4	PQB3	PQB2	PQB1	PQB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN3	AN2	AN1	AN0
EXTERNAL TRIGGER INPUTS:															
			ETRIG2	ETRIG1											
MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

QADC ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB).

Port A pins are referred to as PQA[7:0] when used as a bidirectional 8-bit input/output port that may be used for general-purpose digital input signals or digital open drain output signals. Port A can also be used for analog inputs (AN[59:52]), external trigger inputs (ETRIG[2:1]), and external multiplexer address outputs (MA[2:0]).

Port B pins are referred to as PQB[7:0] when used as an input only 8-bit digital port that may be used for general-purpose digital input signals. Port B can also be used for nonmultiplexed (AN[51:48])/AN[3:0]) and multiplexed (ANz, ANy, ANx, ANw) analog inputs. Read operations on the reserved bits return zeros, and writes have no effect.

B.2.5 Port Data Direction Register

DDRQA — Port Data Direction Register **$####08**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDQA7	DDQA6	DDQA5	DDQA4	DDQA3	DDQA2	DDQA1	DDQA0	RESERVED							
RESET:															
0	0	0	0	0	0	0	0								

The port data direction register (DDRQA) is associated with the port A digital I/O pins. The bidirectional pins have somewhat higher leakage and capacitance specifications. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. The software is responsible for ensuring that DDR bits are not set to one on pins used for analog inputs. When the DDR bit is set to one and the pin is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

NOTE

Caution should be exercised when mixing digital and analog inputs. This should be isolated as much as possible. Rise and fall times should be as large as possible.

Since port B is input-only, a data direction register is not needed. Therefore, the lower byte of the port data direction register is not implemented. Read operations on the reserved bits return zeros, and writes have no effect.

B.2.6 QADC Control Registers

QACR0 — Control Register 0 **$####0A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX	RESERVED						PSH					PSA	PSL		
RESET:															
0							0	0	0	1	1	0	0	1	1

MUX — Externally Multiplexed Mode

The MUX bit allows the software to select the externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA0, MA1 and MA2 pins to be outputs.

0 = Internally multiplexed, 16 possible channels.
1 = Externally multiplexed, 44 possible channels.

PSH[8:4] — Prescaler Clock High Time

The PSH field selects the QADC clock (QCLK) high time in the prescaler. **Table B-2** displays the bits in PSH field which enable a range of QCLK high times.

Table B-2 Prescaler Clock High Times

PSH[8:4]	QCLK High Time	PSH[8:4]	QCLK High Time
00000	1 System Clock Cycle	10000	17 System Clock Cycles
00001	2 System Clock Cycles	10001	18 System Clock Cycles
00010	3 System Clock Cycles	10010	19 System Clock Cycles
00011	4 System Clock Cycles	10011	20 System Clock Cycles
00100	5 System Clock Cycles	10100	21 System Clock Cycles
00101	6 System Clock Cycles	10101	22 System Clock Cycles
00110	7 System Clock Cycles	10110	23 System Clock Cycles
00111	8 System Clock Cycles	10111	24 System Clock Cycles
01000	9 System Clock Cycles	11000	25 System Clock Cycles
01001	10 System Clock Cycles	11001	26 System Clock Cycles
01010	11 System Clock Cycles	11010	27 System Clock Cycles
01011	12 System Clock Cycles	11011	28 System Clock Cycles
01100	13 System Clock Cycles	11100	29 System Clock Cycles
01101	14 System Clock Cycles	11101	30 System Clock Cycles
01110	15 System Clock Cycles	11110	31 System Clock Cycles
01111	16 System Clock Cycles	11111	32 System Clock Cycles

PSA — Prescaler Add a Clock Tic

0 = QCLK high and low times are not modified.
1 = Add one system clock half cycle to the high time of the QCLK and subtract one system clock half cycle from the low time.

PSL[2:0] — Prescaler Clock Low Time

The PSL field selects the QADC clock (QCLK) low time in the prescaler. **Table B-2** displays the bits in PSL field which enable a range of QCLK low times.

Table B-3 Prescaler Clock Low Times

PSL[2:0]	QCLK Low Time
000	1 System Clock Cycle
001	2 System Clock Cycles
010	3 System Clock Cycles
011	4 System Clock Cycles
100	5 System Clock Cycles
101	6 System Clock Cycles
110	7 System Clock Cycles
111	8 System Clock Cycles

QACR1 — Control Register 1 **$####0C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIE1	PIE1	SSE1	NOT USED		MQ1			NOT USED							
RESET:															
0	0	0			0	0	0								

CIE1 — Queue 1 Completion Interrupt Enable

0 = Disable the queue completion interrupt associated with queue 1.

1 = Enable an interrupt after the conversion of the sample requested by the last CCW in queue 1.

PIE1 — Queue 1 Pause Interrupt Enable

0 = Disable the pause interrupt associated with queue 1.

1 = Enable an interrupt after the conversion of the sample requested by a CCW in queue 1 which has the pause bit set.

SSE1 — Queue 1 Single-Scan Enable Bit

0 = Trigger events are not accepted for single-scan modes.

1 = Accept a trigger event to start queue 1 in a single-scan mode.

MQ1[10:8] — Queue 1 Operating Mode

The MQ1 field selects the queue operating mode for queue 1. **Table B-4** shows the bits in the MQ1 field which enable different queue 1 operating modes.

Table B-4 Queue 1 Operating Modes

MQ1[10:8]	Operating Modes
000	Disabled mode, conversions do not occur
001	Software triggered single-scan mode (started with SSE1)
010	External trigger rising edge single-scan mode (on ETRIG1 pin)
011	External trigger falling edge single-scan mode (on ETRIG1 pin)
100	Reserved mode, conversions do not occur
101	Continuous-scan software triggered mode
110	External trigger rising edge continuous-scan mode (on ETRIG1 pin)
111	External trigger falling edge continuous-scan mode (on ETRIG1 pin)

QACR2 — Control Register 2 **$####0E**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CIE2	PIE2	SSE2	MQ2					RES	NOT USED	BQ2					
RESET:	0	0	0	0	0	0	0	0	0		1	0	0	1	1	1

CIE2 — Queue 2 Completion Software Interrupt Enable

0 = Disable the queue completion interrupt associated with queue 2.

1 = Enable an interrupt after the conversion of the sample requested by the last CCW in queue 2.

PIE2 — Queue 2 Pause Software Interrupt Enable

0 = Disable the pause interrupt associated with queue 2.

1 = Enable an interrupt after the conversion of the sample requested by a CCW in queue 2 which has the pause bit set.

SSE2 — Queue 2 Single-Scan Enable Bit

0 = Trigger events are not accepted for single-scan modes.

1 = Accept a trigger event to start queue 2 in a single-scan mode.

MQ2[12:8] — Queue 2 Operating Mode

The MQ2 field selects the queue operating mode for queue 2. **Table B-5** shows the bits in the MQ2 field which enable different queue 2 operating modes.

Table B-5 Queue 2 Operating Modes

MQ2[12:8]	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	External trigger rising edge single-scan mode (on ETRIG2 pin)
00011	External trigger falling edge single-scan mode (on ETRIG2 pin)
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Continuous-scan software triggered mode
10010	External trigger rising edge continuous-scan mode (on ETRIG2 pin)
10011	External trigger falling edge continuous-scan mode (on ETRIG2 pin)
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^9

Table B-5 Queue 2 Operating Modes (Continued)

MQ2[12:8]	Operating Modes
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	Reserved mode

RES — Queue 2 Resume

0 = After suspension, begin execution with the first CCW in queue 2 or the current subqueue.
1 = After suspension, begin execution with the aborted CCW in queue 2.

BQ2[5:0] — Beginning of Queue 2

The BQ2 field indicates the CCW location where queue 2 begins. To allow the length of queue 1 and queue 2 to vary, a programmable pointer identifies the CCW table location where queue 2 begins. The BQ2 field also serves as an end-of-queue condition for queue 1.

B.2.7 QADC Status Register

QASR — Status Register **$####10**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP					
RESET:																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CF1 — Queue 1 Completion Flag

0 = Queue 1 scan is not complete.
1 = Queue 1 scan is complete.

PF1 — Queue 1 Pause Flag

0 = Queue 1 has not reached a pause.
1 = Queue 1 has reached a pause.

CF2 — Queue 2 Completion Flag

0 = Queue 2 scan is not complete.
1 = Queue 2 scan is complete.

PF2 — Queue 2 Pause Flag

0 = Queue 2 has not reached a pause.
1 = Queue 2 has reached a pause.

TOR1 — Queue 1 Trigger Overrun

0 = No unexpected queue 1 trigger events have occurred.
1 = At least one unexpected queue 1 trigger event has occurred.

TOR2 — Queue 2 Trigger Overrun

0 = No unexpected queue 2 trigger events have occurred.
1 = At least one unexpected queue 2 trigger event has occurred.

QS[9:6] — Queue Status

Table B-6 shows the bits in the QS field and how they affect the status of queue 1 and queue 2.

Table B-6 Queue Status

QS[9:6]	Queue 1/Queue 2 States
0000	Queue 1 idle, Queue 2 idle
0001	Queue 1 idle, Queue 2 paused
0010	Queue 1 idle, Queue 2 active
0011	Queue 1 idle, Queue 2 trigger pending
0100	Queue 1 paused, Queue 2 idle
0101	Queue 1 paused, Queue 2 paused
0110	Queue 1 paused, Queue 2 active
0111	Queue 1 paused, Queue 2 trigger pending
1000	Queue 1 active, Queue 2 idle
1001	Queue 1 active, Queue 2 paused
1010	Queue 1 active, Queue 2 suspended
1011	Queue 1 active, Queue 2 trigger pending
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

CWP — Command Word Pointer

The CWP allows the software to know which CCW is executing at present, or was last completed. The command word pointer is a software read-only field, and write operations have no effect.

B.2.8 Conversion Command Word Table

CCW — Conversion Command Word Table **$####30–$####7E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED						P	BYP	IST		CHAN					
RESET:															
						U	U	U	U	U	U	U	U	U	U

P — Pause

0 = Do not enter the pause state after execution of the current CCW.
1 = Enter the pause state after execution of the current CCW.

BYP — Sample Amplifier Bypass

0 = Amplifier bypass mode disabled.
1 = Amplifier bypass mode enabled.

IST[7:6] — Input Sample Time

The IST field allows software to specify the length of the sample window. **Table B-7** shows the bits in the IST field which enable different input sample times.

Table B-7 Input Sample Times

IST[7:6]	Input Sample Times
00	Input sample time = QCLK period x 2
01	Input sample time = QCLK period x 4
10	Input sample time = QCLK period x 8
11	Input sample time = QCLK period x 16

CHAN[5:0] — Channel Number

The CHAN field selects the input channel number. **Table B-8** shows the channel number assignments for the nonmultiplexed mode. **Table B-9** shows the channel number assignments for the multiplexed mode.

Table B-8 Nonmultiplexed Channel Assignments and Pin Designations

Nonmultiplexed Input Pins				**Channel Number in CCW CHAN Field**	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type**	**Binary**	**Decimal**
PQB0	AN0	—	Input	000000	0
PQB1	AN1	—	Input	000001	1
PQB2	AN2	—	Input	000010	2
PQB3	AN3	—	Input	000011	3
—	—	Invalid	—	0XXXXX	4 to 31
—	—	Reserved	—	10XXXX	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	AN52	—	Input/Output	110100	52
PQA1	AN53	—	Input/Output	110101	53
PQA2	AN54	—	Input/Output	110110	54
PQA3	AN55	ETRIG1	Input/Output	110111	55
PQA4	AN56	ETRIG2	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
V_{RL}	Low Ref	—	Input	111100	60
V_{RH}	High Ref	—	Input	111101	61
—	—	V_{DDA}/2	—	111110	62
—	—	End of Queue Code	—	111111	63

Table B-9 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				**Channel Number in CCW CHAN Field**	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type**	**Binary**	**Decimal**
PQB0	ANw	—	Input	00xxx0	0 to 14 even
PQB1	ANx	—	Input	00xxx1	1 to 15 odd
PQB2	ANy	—	Input	01xxx0	16 to 30 even
PQB3	ANz	—	Input	01xxx1	17 to 31 odd
—	—	Reserved	—	10xxxx	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	—	MA0	Input/Output	110100	52
PQA1	—	MA1	Input/Output	110101	53
PQA2	—	MA2	Input/Output	110110	54
PQA3	AN55	ETRIG1	Input/Output	110111	55
PQA4	AN56	ETRIG2	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
V_{RL}	Low Ref	—	Input	111100	60
V_{RH}	High Ref	—	Input	111101	61
—	—	V_{DDA}/2	—	111110	62
—	—	End of Queue Code	—	111111	63

B.2.9 Result Registers

RJURR — Right Justified, Unsigned Result Register **$####B0–$####FE**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED						RESULT									

The conversion result is unsigned, right justified data. Bits [9:0] are used for the 10-bit resolution; bits [15:10] return zero when read.

LJSRR — Left Justified, Signed Result Register **$####30–$####7E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S[1]	RESULT									NOT USED					

NOTES:

1. S = Sign bit.

The conversion result is signed, left justified data. Bits [15:6] are used for the 10-bit resolution, with the most significant bit inverted to form a sign bit; bits [5:0] return zero when read.

LJURR — Left Justified, Unsigned Result Register **$####B0–$####FE**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT										NOT USED					

The conversion result is unsigned, left justified data. Bits [15:6] are used for the 10-bit resolution; bits [5:0] return zero when read.

APPENDIX C CONVERSION ACCURACY DEFINITIONS

This section explains the terminology used to specify the analog characteristics of the QADC.

C.1 Transfer Curve

Figure C-1 is a plot of the conversion result versus the absolute input voltage. The ideal curve can be thought of as a staircase of uniform step size with perfect positioning of the endpoints (without gain or offset errors).

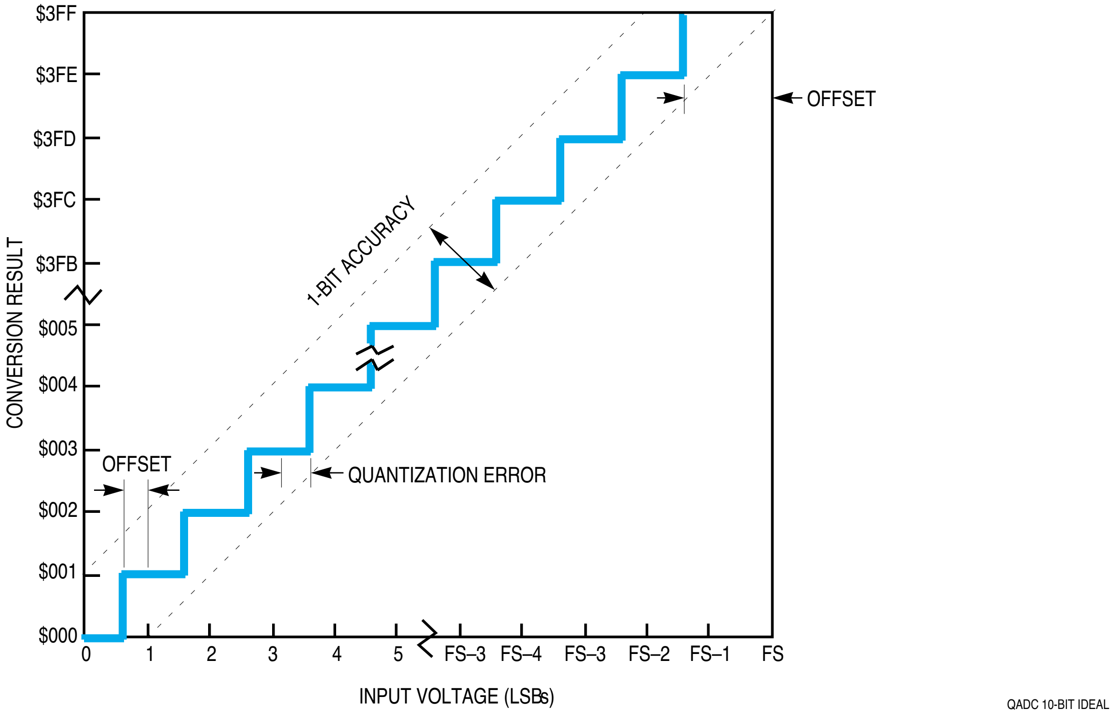


Figure C-1 Transfer Curve of an Ideal 10-Bit A/D Converter

The ideal transfer curve has the midpoint code of $000 at the origin, so that the transition of the conversion result from $000 to $001 occurs for an input of one half LSB. Since the input range spans the 1024 codes from $000 to $3FF, the full scale result of $3FF is centered at one least significant (LSB) below the full scale input for an ideal converter.

For example, a 10-bit converter with a reference voltage of 5.12 V (a binary multiple) has an LSB of exactly 5 mV; there are 1024 values from $000 to $3FF. For an ideal converter, the first transition (from $000 to $001) occurs at 2.5 mV, and the last transition (from $3FE to $3FF) occurs at 5.1125 V, or one and one half LSBs before full scale.

C.2 Offset Error

The offset error is the DC shift of the entire transfer curve of an ideal converter.

C.3 Quantizing Error

The error in the actual code transition due to finite resolution is called the quantizing error. The quantizing error is one half LSB.

C.4 Monotonicity

In a monotonic converter, the conversion result only increases as input voltage is increased. Missing or out-of-sequence codes are not allowed, within a single transfer curve. However, measuring the transfer curve repeatedly might have some variation due to noise. **Figure C-2** shows the two ways that an A/D converter can be non-monotonic.

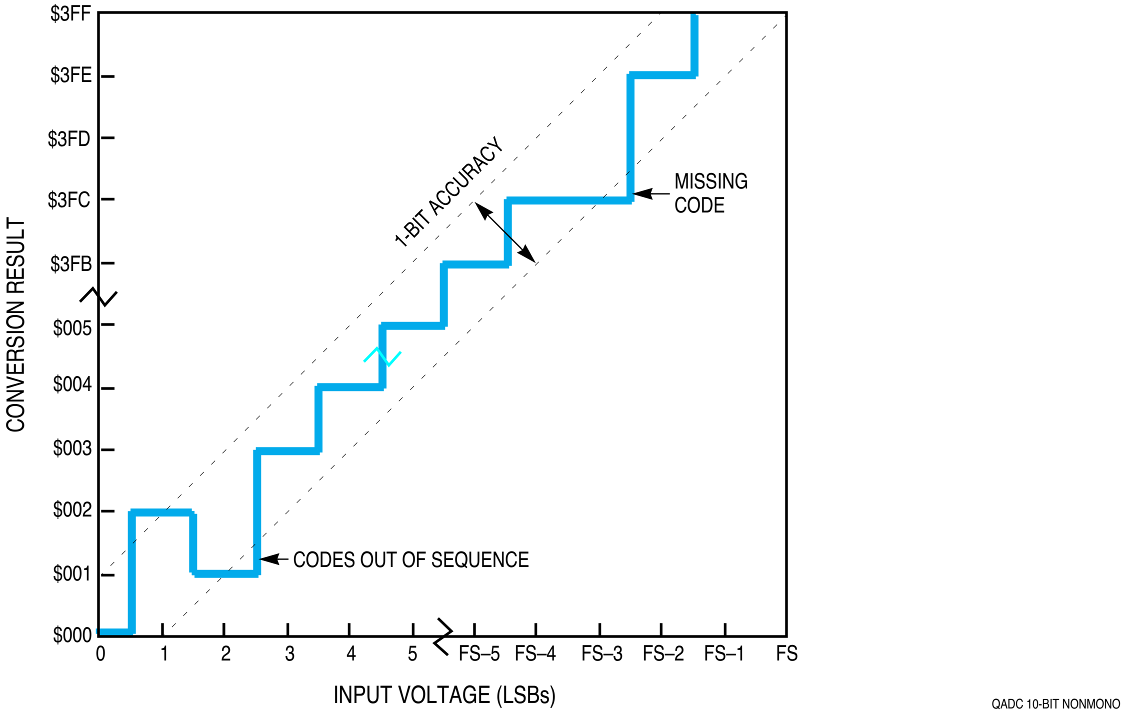
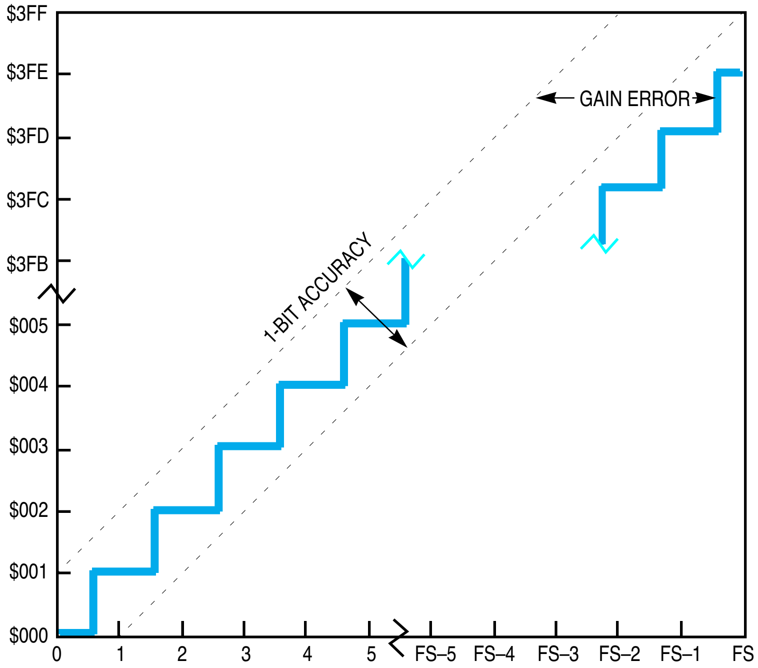


Figure C-2 Transfer Curve of a Non-Monotonic 10-Bit A/D Converter

C.5 Gain Error (Slope Error)

Figure C-3 shows that the gain error is an error in the input to output transfer ratio. Gain error causes an error in the slope of the transfer curve; the average input step size that causes a code transition is not equal to the desired LSB. In ratiometric converters, like the QADC, the gain is a function of the external reference used.



QADC 10-BIT GAIN ERR

Figure C-3 Transfer Curve of a 10-Bit A/D Converter with Gain Error

C.6 Integral Non-Linearity

Figure C-4 displays integral nonlinearity, which is defined as the worst case error from any code transition point on the offset- and gain-corrected actual transfer curve to an ideal curve of infinite resolution. Integral non-linearity is the sum of the quantizing and differential non-linearity errors. An A/D converter can be monotonic, yet not have integral linearity equal to its resolution.

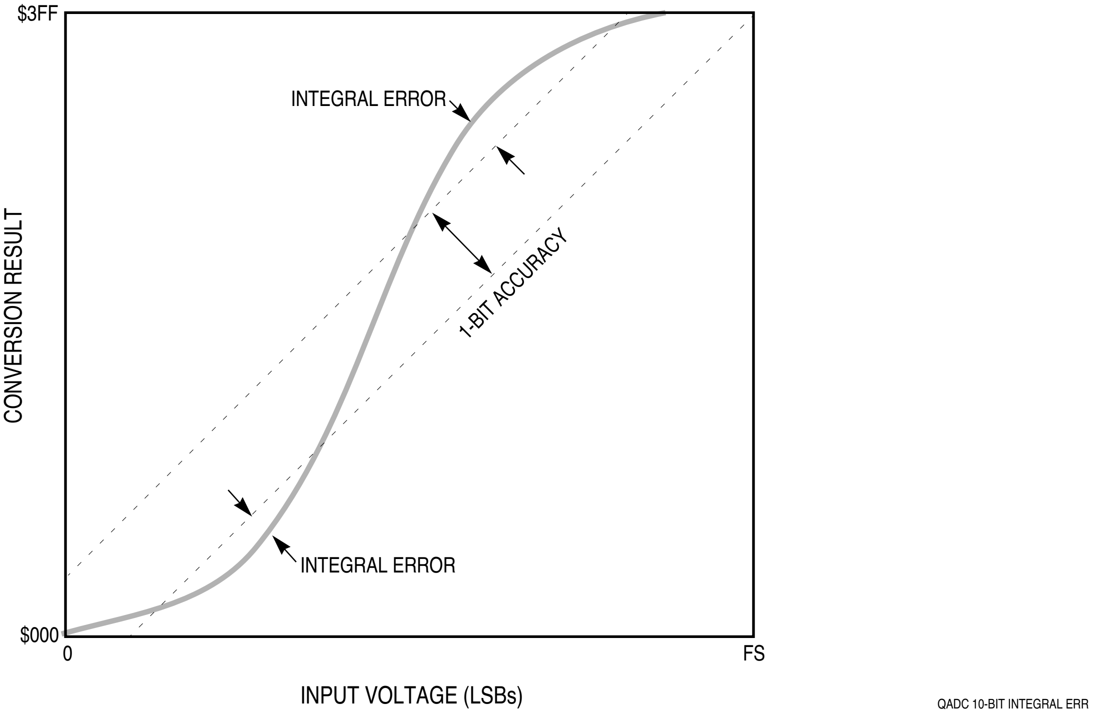


Figure C-4 Transfer Curve of a 10-Bit A/D Converter With Integral > One LSB

C.7 Differential Non-Linearity (Related to Monotonicity)

Figure C-5 displays differential nonlinearity, which is defined as the input step size between any two adjacent code transitions. To guarantee monotonicity, the input voltage step size must range between zero and two LSBs, not including the endpoints (0<Step<2). For monotonicity, the magnitude of the differential linearity error must be less than one LSB. A monotonic A/D converter could have offset or step errors causing all or part of the curve to fall outside the limits of integral accuracy, as long as all input step sizes fall between zero and two LSBs.

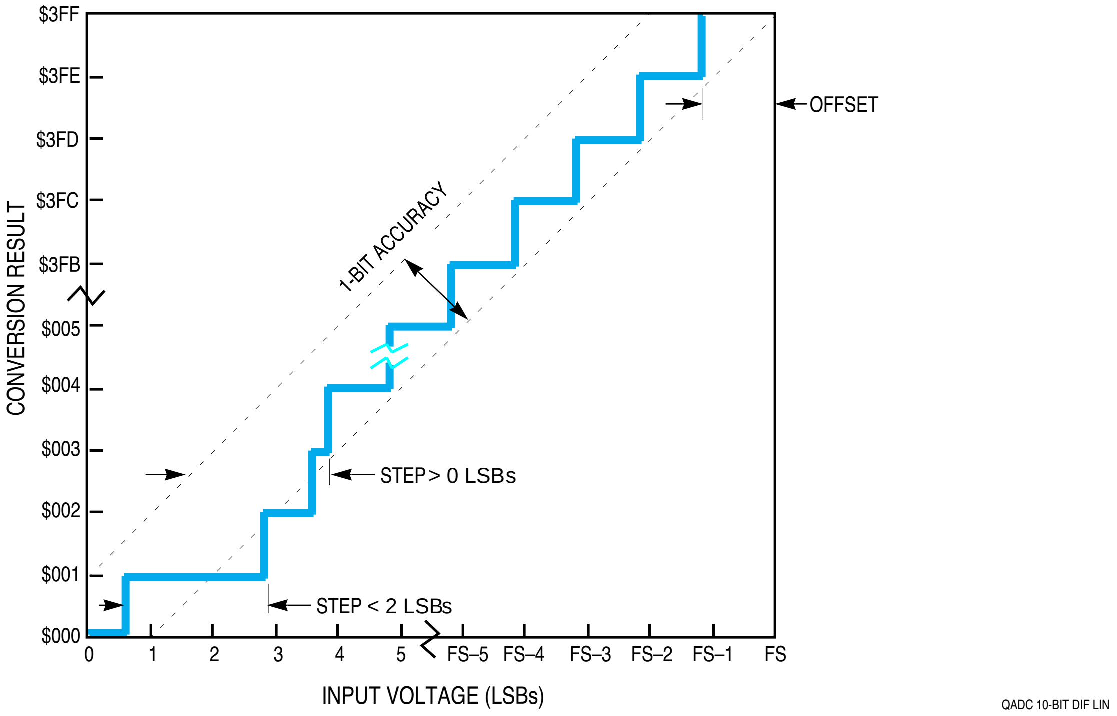


Figure C-5 Transfer Curve of a 10-Bit A/D Converter With Differential Linearity

INDEX