SECTION 13
QUEUED ANALOG-TO-DIGITAL CONVERTER (QADC64E)

The two queued analog-to-digital converter (QADC) modules on MPC565 / MPC566 are 10-bit, unipolar, successive approximation converters with analog multiplexers. These modules are configured so each module can access all 40 of the analog inputs to the part.

For this revision of the **QADC,** the name **QADC64E** implies an enhanced version of the **QADC64**. For simplicity, the names QADC and QADC64E may be used interchangeably throughout this document.

13.1 Features, Overview and Quick Reference Diagrams

This section gives an overview of the implementation of the two QADC64E modules on MPC565 / MPC566. It can also be used for a quick reference while programming the modules.

13.1.1 Features of the QADC64E (Each Module)

- Internal sample and hold
- Up to 40 analog input channels using QADC64E multiplexing and AMUX multiplexing
- Directly supports up to four external multiplexers (for example, the MC14051)
- Up to 65 total input channels using QADC64E, AMUX, and external multiplexing
- Supports AMUX without loss of Port A and Port B functionality
- Programmable input sample time for various source impedances
- Typical conversion time of less than 5 µs (> 200 kSamples/s)
- Modulus prescaler can divide the system clock for the converter by two to 128
- Two conversion command queues with a total of 64 entries
- Sub-queues possible using pause mechanism
- Queue complete and pause software interrupts available on both queues
- Queue pointers indicate current location for each queue
- Automated queue modes initiated by:
  - — External edge trigger and gated trigger
  - — Periodic/interval timer, within QADC64E module [queue 1 and 2]
  - — Software command
- Single-scan or continuous-scan of queues

- 64 result registers in each QADC64E
- Output data readable in three formats:
  - — Right-justified unsigned
  - — Left-justified signed
  - — Left-justified unsigned
- Unused analog channels can be used as digital ports
- Multi-module operation allows shared channels (in all queues) and shared clock for synchronization
- Alternate reference input, with control in CCW

13.1.2 QADC64E Block Diagrams

Figure 13-1 displays the major components of the QADC64E modules on the MPC565 / MPC566 including the pads, AMUX and both QADCs.

Figure 13-2 shows the sub-module arrangement of the QADC64E.

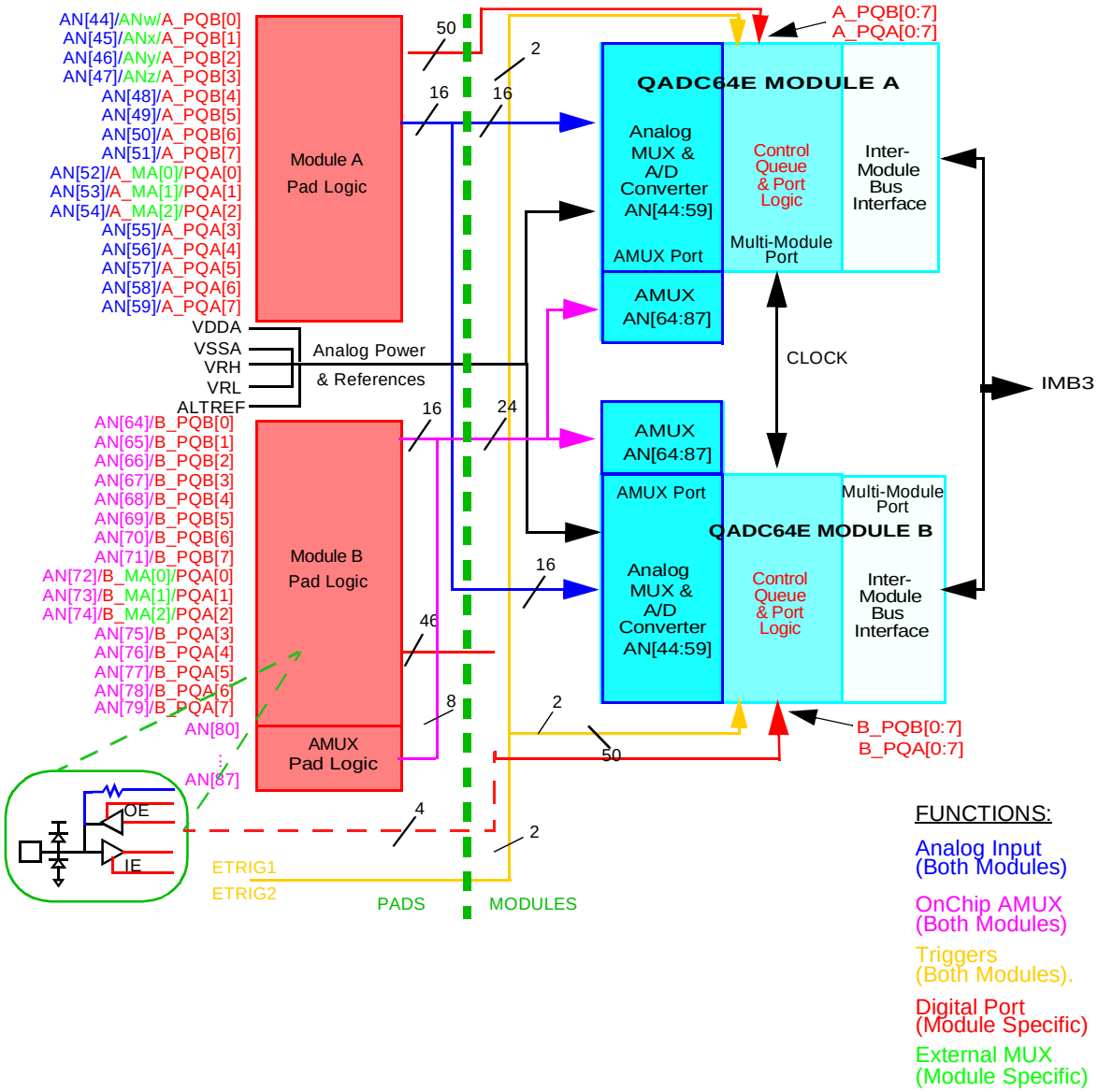


Figure 13-1 Dual QADC64Es on MPC565 / MPC566

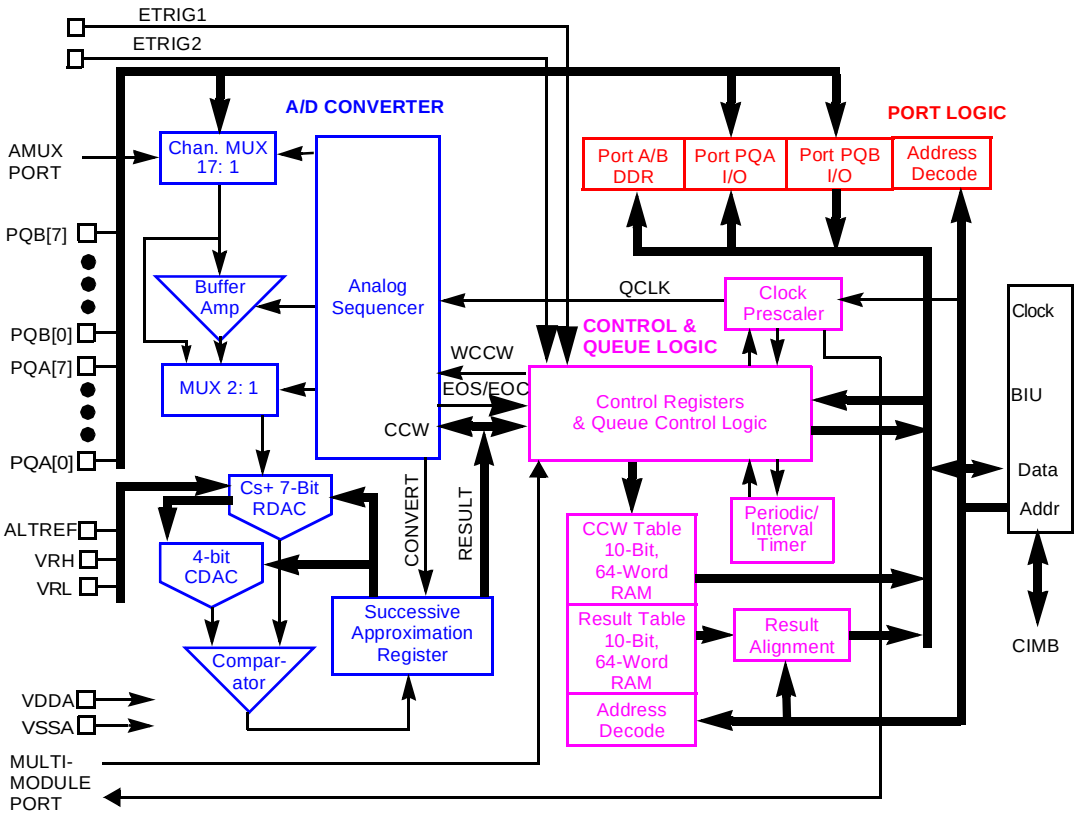


Figure 13-2 Block Diagram of QADC64E

The analog section includes input pins, an analog multiplexer, and the sample and hold 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 queue control logic to sequence the conversion process and interrupt generation logic. Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table RAM, and the result table RAM.

The bus interface unit (BIU) allows the QADC64E to operate with the applications software through the IMB environment.

13.1.3 Memory Map

The QADC64E occupies one Kbyte, or 512 16-bit entries, of address space. Ten 16-bit registers are control, port, and status registers, 64 16-bit entries are the CCW table, and 64 16-bit entries are the result table, and occupy 192 16-bit address locations because the result data is readable in three data alignment formats.

Each QADC on MPC565 / MPC566 has its own memory location. **Table 13-1** shows the memory map for QADC64E module A. Module B has the same offset scheme starting at 0x30 4C00 (refer to **Table 13-2**).

QADC64E_A occupies 0x30 4800 to 0x30 4BFF

QADC64E_B occupies 0x30 4C00 to 0x30 4FFF

Table 13-1 QADC64E_A Address Map

Address	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15	Register
0x30 4800	STOP	FRZ	--						SUPV	MSTR	EXT CLK	--					Module Config.*
0x30 4802	TEST MODE		---														Test
0x30 4804	IRL1					IRL2					--						Interrupt
0x30 4806	PORTQA								PORTQB								Port Data
0x30 4808	DDRQA								DDRQB								Port Direction
0x30 480A	EMUX	---		TRIG	--				QCLK PRESCALER								Control 0
0x30 480C	CIE1	PIE1	SSE1	MQ1					---								Control 1
0x30 480E	CIE2	PIE2	SSE2	MQ2					RESU.	BQ2							Control 2
0x30 4810	CF1	PF1	CF2	PF2	TOR1	TOR2	QS		CWP								Status 0
0x30 4812	--		CWPQ1						---	CWPQ2							Status 1
0x30 4814-0x30 49FF	--14 Words Reserved --																Reserved
0x30 4A00-0x30 4A7F	--						PAUS	REF	IST	CHAN							CCWs
0x30 4A80-0x30 4AFF	0000 00						UNSIGNED RIGHT JUSTIFIED										Results
0x30 4B00-0x30 4B7F	SIGN	SIGNED LEFT JUSTIFIED									00 0000						Results
0x30 4B80 0x30 4BFF	UNSIGNED LEFT JUSTIFIED										00 0000						Results

***Registers in RED are accessible only as supervisor data space**

Table 13-2 QADC64E_B Address Map

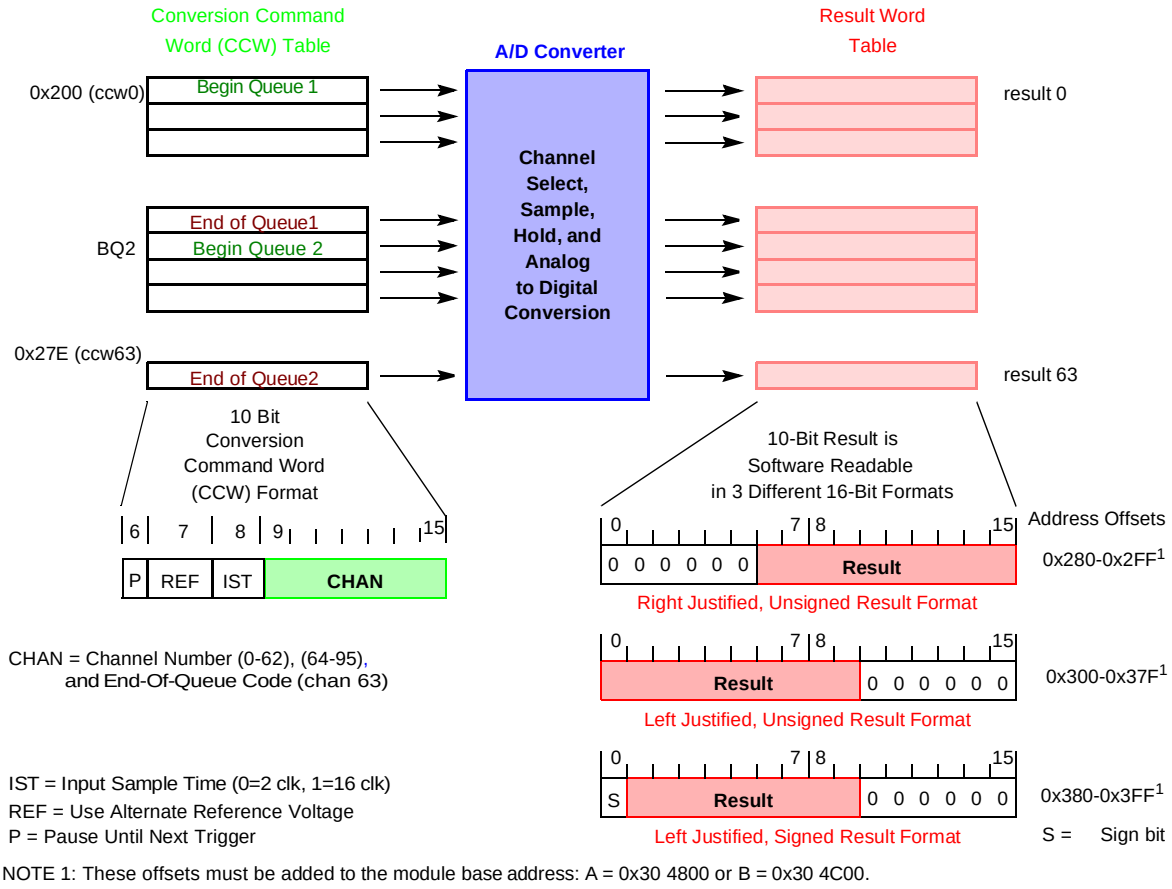
Address	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15	Register
0x30 4C00	STOP	FRZ	--						SUPV	MSTR	EXT CLK	--					Module Config.*
0x30 4C02	TEST MODE								---								Test
0x30 4C04	IRL1								IRL2								Interrupt
0x30 4C06	PORTQA								PORTQB								Port Data
0x30 4C08	DDRQA								DDRQB								Port Direction
0x30 4C0A	EMUX	---		TRIG		--			QCLK PRESCALER								Control 0
0x30 4C0C	CIE1	PIE1	SSE1	MQ1					---								Control 1
0x30 4C0E	CIE2	PIE2	SSE2	MQ2					RESU.	BQ2							Control 2
0x30 4C10	CF1	PF1	CF2	PF2	TOR1	TOR2	QS		CWP								Status 0
0x30 4C12	--		CWPQ1						---	CWPQ2							Status 1
0x30 4C14-0x30 4DFF	--14 Words Reserved --																Reserved
0x30 4E00-0x30 4E7F	--						PAUS	REF	IST	CHAN							CCWs
0x30 4E80-0x30 4EFF	0000 00						UNSIGNED RIGHT JUSTIFIED										Results
0x30 4F00-0x30 4F7F	SIGN	SIGNED LEFT JUSTIFIED									00 0000						Results
0x30 4F80 0x30 4FFF	UNSIGNED LEFT JUSTIFIED										00 0000						Results

*Registers in RED are accessible only as supervisor data space

Access to supervisor-only data space is permitted only when the bus master 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. See **13.2.1.3 Supervisor/Unrestricted Address Space**.

13.1.4 Using the Queue and Result Word Table

The heart of the QADC is its conversion command word (CCW) queues. This is where the module is programmed to convert a particular channel according to a particular requirement. The queues are created by writing CCWs into the CCW table in the register memory. The queues are controlled by the three control registers, and their status can be read from status registers. As conversions are completed the digital value is written into the result word table. **Figure 13-3** shows the CCW queue and the result word table.



NOTE 1: These offsets must be added to the module base address: A = 0x30 4800 or B = 0x30 4C00.

Figure 13-3 CCW Queue and Result Table Block Diagram

13.1.5 External Multiplexing

The QADC can use from one to four 8-input external multiplexer chips to expand the number of analog signals that may be converted. The externally multiplexed channels are automatically selected from the channel field of the conversion command word (CCW) table. The software selects the external multiplexed mode by setting the MUX bit in control register 0 (QACR0).

Figure 13-4 shows the maximum configuration of four external multiplexer chips connected to the QADC. The QADC provides three multiplexed address signals — MA[0], MA[1], and MA[2], 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. The QADC converts the proper input channel (AN_W, AN_X, AN_Y, and AN_Z) by interpreting the CCW channel number.

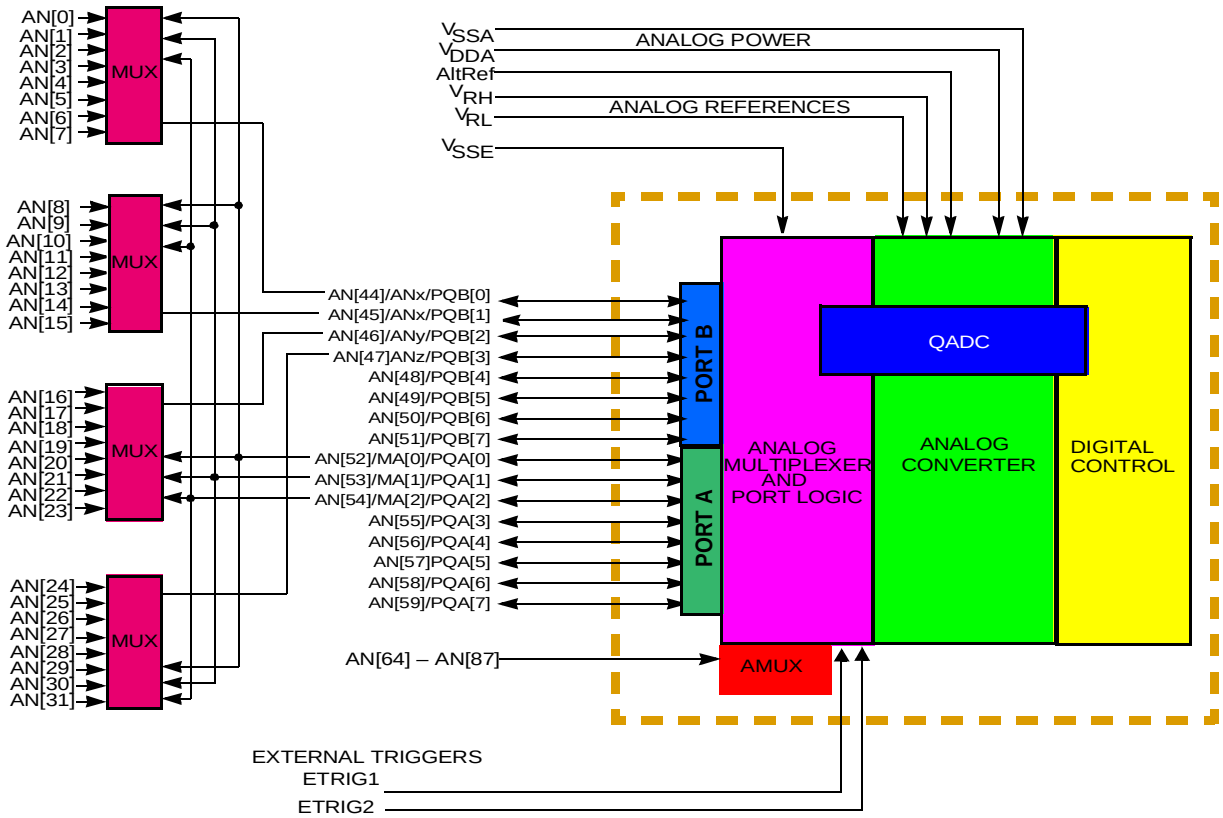


Figure 13-4 Example of External Multiplexing

In the external multiplexed mode, four of the port A pins are redefined to each represent eight input channels. Refer to **Table 13-3**.

Table 13-3 Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw (AN[44])	Channels from 0 to 7
ANx (AN[45])	Channels from 8 to 15
ANy (AN[46])	Channels from 16 to 23
ANz (AN[47])	Channels from 24 to 31

Table 13-4 shows the total number of analog input channels supported with zero to four external multiplexer chips using one QADC module.

Table 13-4 Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels				
No External MUX Chips	**One External MUX Chip**	**Two External MUX Chips**	**Three External MUX Chips**	**Four External MUX Chips**
40	36 + 8 = 44	35 + 16 = 51	34 + 24 = 58	33 + 32 = 65

NOTE: QADC64E External MUX Users

If either QADC64E_A or QADC64E_B is in external MUX (EMUX) mode, AN[44:47] (analog inputs) and AN[52:54] and/or AN[72:74] should not be programmed into queues as they will be EMUX input and address pins.

If both QADC64E_A or QADC64E_B are in external MUX mode, they independently control MUX addresses, but SHARE INPUT PINS so a total of four external multiplexers can be used, but each should be assigned to only A or B QADCs. Each QADC module queue can only control external addresses that are controlled by its own MA[2:0] multiplexer address pins.

13.2 Programming the QADC64E Registers

The QADC64E has three global registers for configuring module operation: the module configuration register (**13.2.1 QADC64E Module Configuration Register**), the interrupt register (**13.2.2 QADC64E Interrupt Register**), and a test register (QADCTEST). The global registers are always defined to be in supervisor-only data space. Refer to **Table 13-1** for the QADC64E_A address map and **Table 13-2** for the QADC64E_B address map. See **13.2.1.3 Supervisor/Unrestricted Address Space** for access modes of these registers.

The remaining five registers in the control register block control the operation of the queuing mechanism, and provide a means of monitoring the operation of the QADC64E.

- Control register 0 (QACR0) contains hardware configuration information (**13.2.5 Control Register 0**)
- Control register 1 (QACR1) is associated with queue 1 (**13.2.6 Control Register 1**)
- Control register 2 (QACR2) is associated with queue 2 (**13.2.7 Control Register 2**)
- Status registers (QASR0 and QASR1) provide visibility on the status of each queue and the particular conversion that is in progress (**13.2.8 Status Registers**)

The CCW table follows the register block in the address map. There are 64 entries to hold the desired analog conversion sequences. Each CCW is a 16-bit entry, with ten implemented bits in four fields.

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.

13.2.1 QADC64E Module Configuration Register

The QADCMCR contains fields and bits that control freeze and stop modes and determine the privilege level required to access most registers

QADCMCR — Module Configuration Register **0x30 4800**
0x30 4C00

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ	RESERVED						SUPV	MSTR	EXT CLK	RESERVED				
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 13-5 QADCMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop Enable — Refer to **13.2.1.1 Low Power Stop Mode** for more information. 0 = Disable stop mode 1 = Enable stop mode
1	FRZ	Freeze Enable — Refer to **13.2.1.2 Freeze Mode** for more information. 0 = Ignores the IMB internal FREEZE signal 1 = Finish any conversion in progress, then freeze
2:7	—	Reserved
8	SUPV	Supervisor/Unrestricted Data Space — Refer to **13.2.1.3 Supervisor/Unrestricted Address Space** and **Table 13-6** for more information. 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 QADC64E registers and CCW/result tables are designated as supervisor-only data space.
9	MSTR	Master/Slave Operation — Refer to **13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks** for more information. 0 = Module is a slave, and QADCMCR EXTCLK should be used to select the conversion clock source. 1 = Module is the master, and QADCMCR EXTCLK should be left clear.
10	EXTCLK	External Clock Select — 0 = Internal conversion clock derived from IMB3 clock will be used for the converter and QADC Periodic/Interval Timer. 1 = Internal conversion clock derived from a master QADC module (see MSTR, bit 9). Refer to **13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks**.
11:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in QADC64E implementations that use hardware interrupt arbitration. These bits are not used on the MPC565 / MPC566.

13.2.1.1 Low Power Stop Mode

When the STOP bit in the QADCMCR is set, the QADC64E clock (QCLK) which clocks the A/D converter, is disabled and the analog circuitry is powered down . This results in a static, low power consumption, idle condition. The stop mode aborts any conver-

sion sequence in progress. Because the bias currents to the analog circuits are turned off in stop mode, the QADC64E requires some recovery time (T_{SR} in **APPENDIX E ELECTRICAL CHARACTERISTICS**) to stabilize the analog circuits after the stop enable bit is cleared.

In stop mode:

- BIU state machine and logic do not shut down
- The RAM is not reset and is not accessible
- The module configuration register (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
- Control register 1 (QACR1), control register 2 (QACR2), and the status registers (QASR0 and QASR1) are reset and are read-only accessible
- In addition, the periodic/interval timer is held in reset during stop mode

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

13.2.1.2 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 QADC64E responds to an IMB internal FREEZE signal assertion. Freeze is very useful when debugging an application.

When the FRZ bit is set, the QADC64E finishes any conversion in progress then freezes. Depending on when the FRZ bit is asserted, there are three possible queue “freeze” scenarios.

- When a queue is not executing, the QADC64E freezes immediately
- When a queue is executing, the QADC64E completes the conversion in progress 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 QADC64E freezes immediately

When the QADC64E enters the freeze mode while a queue is active, the current CCW location of the queue pointer is saved.

During freeze, the analog clock, QCLK, is held in reset and the periodic/interval timer is held in reset. External trigger events that occur during the freeze mode are not captured. The BIU remains active to allow IMB access to all QADC64E registers and RAM. Although the QADC64E saves a pointer to the next CCW in the current queue, the software can force the QADC64E to execute a different CCW by writing new queue operating modes for normal operation. The QADC64E looks at the queue operating

modes, the current queue pointer, and any pending trigger events to decide which CCW to execute when exiting freeze.

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

13.2.1.3 Supervisor/Unrestricted Address Space

The QADC64E 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.

The following information applies to accesses to address space located within the module's 16-bit boundaries and where the module response causes the bus error. See **Table 13-6** for more information.

Table 13-6 QADC64E Bus Error Response

S/U Mode	SUPV Bit	Supervisor-Only Register	Supervisor/ Unrestricted Register	Reserved/ Unimplemented Register
U	0	QADC64E bus error	Valid access	QADC64E bus error
U	1	Master bus error	Master bus error	Master bus error
S	0	Valid access	Valid access	QADC64E bus error
S	1	Valid access	Valid access	QADC64E bus error

NOTES:
S/U = Supervisor/Unrestricted
QADC64E bus error = Caused by QADC64E
Master bus error = Caused by bus master
No bus error = Writes have no effect, reads will return zeroes.
Access to QADCTEST register will act as a reserved/unimplemented register unless in factory test mode

- Attempts to read a supervisor-only data space when not in the supervisor access mode and SUPV = 1, causes the bus master to assert a bus error condition. No data is returned. If SUPV = 0, the QADC64E asserts a bus error condition and no data is returned,
- Attempts to write supervisor-only when not in the supervisor access mode and SUPV = 1, causes the bus master to assert a bus error condition. No data is written. If SUPV = 0, the QADC64E asserts a bus error condition and the register is not written.
- Attempts to read unimplemented data space in the unrestricted access mode and SUPV = 1, causes the bus master to assert a bus error condition and no data is returned. In all other attempts to read unimplemented data space, the QADC64E causes a bus error condition and no data is returned.

- Attempts to write unimplemented data space in the unrestricted access mode and SUPV= 1, causes the bus master to assert a bus error condition and no data is written. In all other attempts to write unimplemented data space, the QADC64E causes a bus error condition and no data is written.
- Attempts to read assignable data space in the unrestricted access mode when the space is programmed as supervisor space causes the bus master to assert a bus error condition and no data is returned.
- Attempts to write assignable data space in the unrestricted access mode when the space is programmed as supervisor space causes the bus master to assert a bus error condition and the register is not written.

The bus master indicates the supervisor and user space access with the function code bits (FC[2:0]) on the IMB. For privilege violations, refer to the MPC565 / MPC566 reference manual to determine the consequence of a bus error cycle termination.

The supervisor-only data space segment contains the QADC64E 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 QADC64E registers are programmable.

13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks

Master/slave mode operation is controlled by two bits in the module configuration register. The first is QADCMCR MSTR field, which, when set makes the module the master and causes that module's conversion clock to be output on the slave clock input/output. If the QADCMCR MSTR bit is cleared, then that module is set up as a slave. When set as a slave, the module's QADCMCR EXTCLK may be set, causing the module to use the slave clock signal input as the conversion clock.

If a module is configured as a master, then the QADCMCR EXTCLK bit should be set low.

If a module is configured as a slave, then the QADCMCR EXTCCLK bit may be left clear, in which case the module will operate off of the internal conversion clock, or it may be set, in which case the module will operate off of the slave clock input/output.

13.2.2 QADC64E Interrupt Register

QADCINT specifies the priority level of QADC64E interrupt requests and the vector provided. 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 bits read zero and writes have no effect. They are typically written once when the software initializes the QADC64E, and not changed afterwards.

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IRL1					IRL2					RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-7 QADCMCR Bit Descriptions

Bit(s)	Name	Description
0:4	IRL1	Queue 1 Interrupt Request Level — The IRL1 field establishes the queue 1 interrupt request level. The 00000 state provides a level 0 interrupt while 11111 provides a level 31 interrupt. All interrupts are presented on the IMB. Interrupt level priority software determines which level has the highest priority request.
5:9	IRL2	Queue 2 Interrupt Request Level — The IRL2 field establishes the queue 2 interrupt request level. The 00000 state provides a level 0 interrupt while 11111 provides a level 31 interrupt. All interrupts are presented on the IMB. Interrupt level priority software determines which level has the highest priority request.
10:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in QADC64E implementations that use hardware interrupt arbitration. These bits are not used on the MPC565 / MPC566.

The QADC64E conditionally generates interrupts to the bus master via the IMB IRQ signals. When the QADC64E sets a status bit assigned to generate an interrupt, the QADC64E drives the IRQ bus. The value driven onto IRQ[7:0] represents the interrupt level assigned to the interrupt source. Under the control of ILBS, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 13-5** displays the interrupt levels on IRQ with ILBS.

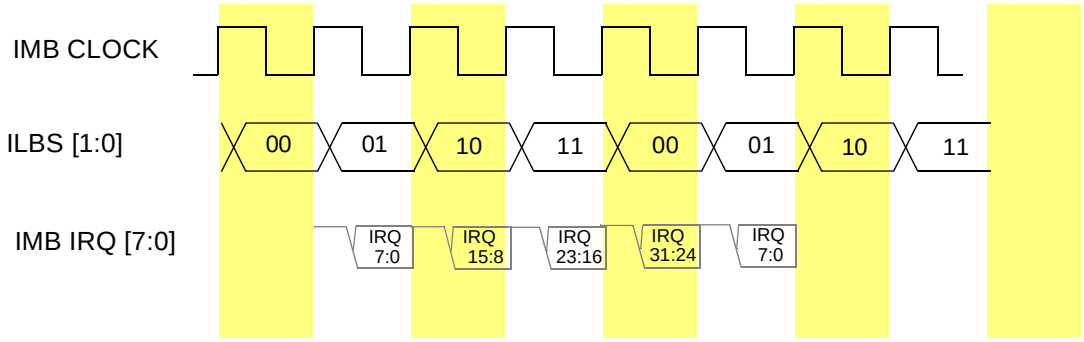


Figure 13-5 Interrupt Levels on IRQ with ILBS

13.2.3 Port Data Register

QADC64E ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB) in each QADC64E.

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 output signals. Port A of module A can also be used for analog inputs (AN[59:52]) and external multiplexer address outputs (MA[2:0]). Module B Port A pins are shared with AN[79:72] and MA[2:0].

Port B pins are referred to as PQB[7:0] when used as a bidirectional 8-bit input/output port that may be used for general-purpose digital input signals or digital output signals. Data for PQB[7:0] is accessed in the lower half of the Module A. Port B can also be used for non-multiplexed (AN[51:44]) and multiplexed PORTQB ANw, ANx, ANy, ANz) analog inputs. Module B Port B is shared with analog inputs AN[71:64].

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. When the data direction bit specifies the pin to be an output, the content of the port data register is read.

PORTQA and PORTQB are not initialized by reset.

PORTQA — Port A Data Register **0x30 4806**
0x30 4C06

PORTQB — Port B Data Register **0x30 4807**
0x30 4C07

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
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
QADC64E_A ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN47	AN46	AN45	AN44
QADC64E_B ANALOG CHANNEL:															
AN79	AN78	AN77	AN76	AN75	AN74	AN73	AN72	AN71	AN70	AN69	AN68	AN67	AN66	AN65	AN64
QADC64E_A and QADC64E_B MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
QADC64E_A (Only) MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

13.2.4 Port Data Direction Register

The port data direction register (DDRQA and DDRQB) is associated with the port A and B digital I/O pins. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information. Any bit set to one in this register configures the corresponding pin as an output. Any bit cleared to zero in this register 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.

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 PORTQA[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.

DDRQA — Port A Data Direction Register **0x30 4808**
0x30 4C08

DDRQB — Port B Data Direction Register **0x30 4809**
0x30 4C09

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
DDQA 7	DDQA 6	DDQA 5	DDQA 4	DDQA 3	DDQA 2	DDQA 1	DDQA 0	DDQB 7	DDQB 6	DDQB 5	DDQB 4	DDQB 3	DDQB 2	DDQB 1	DDQB 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.2.5 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, and reserved fields read zero and writes have no effect. They are typically written once when the software initializes the QADC64E, and not changed afterwards.

QACR0 — Control Register 0 **0x30 480A**
0x30 4C0A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MUX	RESERVED		TRG	RESERVED					PRESCALER						
RESET:															
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Table 13-8 QACR0 Bit Descriptions

Bit(s)	Name	Description
0	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 MA[0], MA[1] and MA[2] pins to be outputs. 0 = Internally multiplexed, 40 possible channels 1 = Externally multiplexed, up to 65 possible channels See **Table 13-4**
1:2	—	Reserved
3	TRG	Trigger Assignment — The TRG bit allows the software to assign the ETRIG[2:1] pins to queue 1 and queue 2. 0 = ETRIG[1] triggers queue 1, ETRIG[2] triggers queue 2 1 = ETRIG[1] triggers queue 2, ETRIG[2] triggers queue 1
4:8	—	Reserved
9:15	PRESCAL-ER	Prescaler Value — The PRESCALER determines the QCLK frequency. Refer **APPENDIX E ELECTRICAL CHARACTERISTICS** to for more information on QADC64E operating clock frequency (F_{QCLK}) values. To keep the QCLK within the specified range, the PRES-CALAR field selects the divisor -1 of the system clock to create the frequency of the QCLK, which can range from zero to 128 system clock cycles.

Table 13-9 displays the bits in PRESCALER field which enable a range of QCLK frequencies

Table 13-9 Prescaler F_{SYSCLK} Divide-by Values

Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	Fsysclk Div	Prescaler [6:0]	F_{SYSCLK} Div
0000000	2	0100000	33	1000000	65	1100000	97
0000001	2	0100001	34	1000001	66	1100001	98
0000010	3	0100010	35	1000010	67	1100010	99
0000011	4	0100011	36	1000011	68	1100011	100
0000100	5	0100100	37	1000100	69	1100100	101
0000101	6	0100101	38	1000101	70	1100101	102
0000110	7	0100110	39	1000110	71	1100110	103
0000111	8	0100111	40	1000111	72	1100111	104
0001000	9	0101000	41	1001000	73	1101000	105
0001001	10	0101001	42	1001001	74	1101001	106
0001010	11	0101010	43	1001010	75	1101010	107
0001011	12	0101011	44	1001011	76	1101011	108
0001100	13	0101100	45	1001100	77	1101100	109
0001101	14	0101101	46	1001101	78	1101101	110
0001110	15	0101110	47	1001110	79	1101110	111
0001111	16	0101111	48	1001111	80	1101111	112
0010000	17	0110000	49	1010000	81	1110000	113
0010001	18	0110001	50	1010001	82	1110001	114
0010010	19	0110010	51	1010010	83	1110010	115
0010011	20	0110011	52	1010011	84	1110011	116

Table 13-9 Prescaler F_{SYSCLK} Divide-by Values (Continued)

Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	Fsysclk Div	Prescaler [6:0]	F_{SYSCLK} Div
0010100	21	0110100	53	1010100	85	1110100	117
0010101	22	0110101	54	1010101	86	1110101	118
0010110	23	0110110	55	1010110	87	1110110	119
0010111	24	0110111	56	1010111	88	1110111	120
0011000	25	0111000	57	1011000	89	1111000	121
0011001	26	0111001	58	1011001	90	1111001	122
0011010	27	0111010	59	1011010	91	1111010	123
0011011	28	0111011	60	1011011	92	1111011	124
0011100	29	0111100	61	1011100	93	1111100	125
0011101	30	0111101	62	1011101	94	1111101	126
0011110	31	0111110	63	1011110	95	1111110	127
0011111	32	0111111	64	1011111	96	1111111	128

13.2.6 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 QADC64E, and not changed afterwards.

QACR1 — Control Register 1 **0x30 480C**
0x30 4C0C

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CIE1	PIE1	SSE1	MQ1					RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-10 QACR1 Bit Descriptions

Bit(s)	Name	Description
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. Refer to **Table 13-11** for more information. 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
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. Refer to **Table 13-11** for more information. 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

Table 13-10 QACR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
2	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 QADC64E clears the SSE1 bit when the single-scan is complete. Refer to **Table 13-11** for more information. 0 = Trigger events are not accepted for single-scan modes 1 = Accept a trigger event to start queue 1 in a single-scan mode
3:7	MQ1	Queue 1 Operating Mode — The MQ1 field selects the queue operating mode for queue 1. **Table 13-11** shows the bits in the MQ1 field which enable different queue 1 operating mode
8:15	—	Reserved

Table 13-11 Queue 1 Operating Modes

MQ1[3:7]	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE1)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
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	External gated single-scan mode (started with SSE1)
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
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 timer continuous-scan mode: time = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}

Table 13-11 Queue 1 Operating Modes (Continued)

MQ1[3:7]	Operating Modes
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^{1}
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 l timer continuous-scan mode: time = QCLK period x 2^{17}
11111	External gated continuous-scan mode

13.2.7 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 QADC64E, and not changed afterwards.

QACR2 — Control Register 2 **0x30 480E**
0x30 4C0E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CIE2	PIE2	SSE2	MQ2					RE-SUME	BQ2						
RESET:															
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 13-12 QACR2 Bit Descriptions

Bit(s)	Name	Description
0	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. Refer to **Table 13-13** for more information. 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
1	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. Refer to **Table 13-13** for more information. 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

Table 13-12 QACR2 Bit Descriptions (Continued)

Bit(s)	Name	Description
2	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 QADC64E clears the SSE2 bit when the single-scan is complete. Refer to **Table 13-13** for more information. 0 = Trigger events are not accepted for single-scan modes 1 = Accept a trigger event to start queue 2 in a single-scan mode
3:7	MQ2	Queue 2 Operating Mode — The MQ2 field selects the queue operating mode for queue 2. Refer to **Table 13-13** for more information.
8	RESUME	0 = After suspension, begin executing with the first CCW in queue 2 or the current sub-queue 1 = After suspension, begin executing with the aborted CCW in queue 2
9:15	BQ2	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. Setting BQ2 beyond physical CCW table memory space allows queue 1 all 64 entries. 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 64 or greater, queue 2 has no entries, and the entire CCW table is dedicated to queue 1 and CCW63 is the end-of-queue 1. If BQ2 is zero, 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 zero, 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 after queue 2 has begun or if queue 2 has a trigger pending does not affect queue 2 until queue 2 is started again.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. However, beware of the risk of losing the end-of-queue 1 through moving BQ2. Recommend use of EOQ (chan63) to end queue 1. Refer to **Table 13-13** for more information.

Table 13-13 shows the bits in the MQ2 field which enable different queue 2 operating modes.

Table 13-13 Queue 2 Operating Modes

MQ2[3:7]	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
00011	External trigger falling edge single-scan mode
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	Software triggered continuous-scan mode
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
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
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

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 QADC64E would not recognize a BQ2 end-of-queue condition until queue 1 execution reached CCW1 again, presumably on the next pass through the queue.

13.2.8 Status Registers

The status registers 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.

QASR0 — Status Register 0 **0x30 4810**
0x30 4C10

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
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

Table 13-14 QASR0 Bit Descriptions

Bit(s)	Name	Description
0	CF1	Queue 1 Completion Flag — CF1 indicates that a queue 1 scan has been completed. The scan completion flag is set by the QADC64E 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 QADC64E 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 QADC64E 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 QADC64E 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.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
1	PF1	Queue 1 Pause Flag — PF1 indicates that a queue 1 scan has reached a pause. PF1 is set by the QADC64E 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. Another exception occurs in software controlled mode, where the PF1 can be set but queue 1 never enters the pause state since queue 1 continues without pausing. When PF1 is set and interrupts are enabled for the corresponding queue, the QADC64E 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. In external gated single-scan and continuous-scan mode the definition of PF1 has been redefined. When the gate closes before the end-of-queue 1 is reached, PF1 becomes set to indicate that an incomplete scan has occurred.In single-scan mode, setting PF1 can be used to cause an interrupt and software can then determine if queue 1 should be enabled again. In either external gated mode, setting PF1 indicates that the results for queue 1 have not been collected during one scan (coherently). NOTE: If a pause in a CCW is encountered in external gated mode for either single-scan and continuous-scan mode, the pause flag **will not set**, and execution continues without pausing. This has allowed for the added definition of PF1 in the external gated modes. PF1 is maintained by the QADC64E regardless of whether the corresponding interrupts are enabled. The software may poll PF1 to find out when the QADC64E 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 (or gate has not closed before end-of-queue in gated mode) 1 = queue 1 has reached a pause (or gate closed before end-of-queue in gated mode) Refer to **Table 13-15** for a summary of pause response in all scan modes.
2	CF2	Queue 2 Completion Flag — CF2 indicates that a queue 2 scan has been completed. CF2 is set by the QADC64E 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 QADC64E 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 QADC64E 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 QADC64E 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 **Table 13-15** for a summary of pause response in all scan modes.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
3	PF2	Queue 2 Pause Flag — PF2 indicates that a queue 2 scan has reached a pause. PF2 is set by the QADC64E 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. Another exception occurs in software controlled mode, where the PF2 can be set but queue 2 never enters the pause state. When PF2 is set and interrupts are enabled for the corresponding queue, the QADC64E 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 QADC64E regardless of whether the corresponding interrupts are enabled. The software may poll PF2 to find out when the QADC64E 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 Refer to **Table 13-15** for a summary of pause response in all scan modes.
4	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 captured as a trigger overrun. TOR1 cannot occur when the software initiated single-scan mode or 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 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 QADC64E sets the TOR1 error status bit. An unexpected trigger event may be a system overrun situation, indicating a system loading mismatch. In external gated continuous-scan mode the definition of TOR1 has been redefined. In the case when queue 1 reaches an end-of-queue condition for the second time during an open gate, TOR1 becomes set. This is considered an overrun condition. In this case CF1 has been set for the first end-of-queue 1 condition and then TOR1 becomes set for the second end-of-queue 1 condition. For TOR1 to be set, software must not clear CF1 before the second end-of-queue 1. 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 (or queue 1 reaches an end-of-queue condition for the second time in gated mode) Refer to **Table 13-15** for a summary of pause response in all scan modes.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
5	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 queue 2 is executing, suspended, or a trigger is pending. TOR2 has no effect on the queue execution. A trigger event that causes a trigger overrun is not retained since it is considered unexpected. 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 Refer to **Table 13-15** for a summary of pause response in all scan modes.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
6:9	QS	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 13-16** shows the bits in the QS field and how they affect the status of queue 1 and queue 2. Refer to **13.5 Trigger and Queue Interaction Examples**, which shows the 4-bit queue status field transitions in typical situations.
10:15	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 QADC64E scan sequence. The CWP field is a CCW word pointer with a valid range of 0 to 63. 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 QADC64E 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 or 127), the CWP points to the CCW containing the EOQ. When the last CCW in a queue is in the last CCW table location (CCW63), 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 CCW0. 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. 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.

Table 13-15 Pause Response

Scan Mode	Q Operation	PF Asserts?
External Trigger Single-scan	Pauses	Yes
External Trigger Continuous-scan	Pauses	Yes
Periodic/Interval Timer Trigger Single-scan	Pauses	Yes
Periodic/Interval Timer Continuous-scan	Pauses	Yes
Software Initiated Single-scan	Continues	Yes
Software Initiated Continuous-scan	Continues	Yes
External Gated Single-scan	Continues	No
External Gated Continuous-scan	Continues	No

Table 13-16 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 QADC64E 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 QADC64E does not execute any CCWs from the paused queue until a trigger event occurs. Consequently, the QADC64E 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 13-6 displays the status conditions of the queue status field as the QADC64E goes through the transition from queue 1 active to queue 2 active.

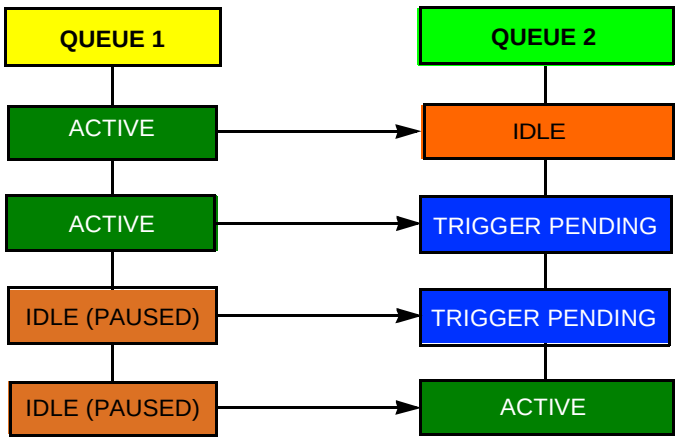


Figure 13-6 Queue Status Transition

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.

QASR1 — Status Register 1 **0x30 4812**
0x30 4C12

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED		CWPQ1						RESERVED		CWPQ2					
RESET:															
0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Table 13-17 QASR1 Bit Descriptions

Bit(s)	Name	Description
0:1	—	Reserved
2:7	CWPQ1	Command Word Pointer for Q1 — CWPQ1 allows the software to know what CCW was last completed for queue 1. This field is a software read-only field, and write operations have no effect. CWPQ1 allows software to read the last executed CCW in queue 1, regardless of which queue is active. The CWPQ1 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ1 is updated when the conversion result is written. When the QADC64E finishes a conversion in queue 1, both the result register is written and the CWPQ1 are updated. Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, CWP points to BQ2 while CWPQ1 points to the last CCW queue 1. During the stop mode, the CWPQ1 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWPQ1 is unchanged; it points to the last executed CCW in queue 1.
8:9	—	Reserved
10:15	CWPQ2	Command Word Pointer for Q2 — CWPQ2 allows the software to know what CCW was last completed for queue 2. This field is a software read-only field, and write operations have no effect. CWPQ2 allows software to read the last executed CCW in queue 2, regardless which queue is active. The CWPQ2 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ2 is updated when the conversion result is written. When the QADC64E finishes a conversion in queue 2, both the result register is written and the CWPQ2 are updated. During the stop mode, the CWPQ2 is reset to 63, 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 in queue 2.

13.2.9 Conversion Command Word Table

The conversion command word (CCW) table is a RAM, 64 words long on 16-bit address boundaries where 10-bits of each entry are implemented. A CCW can be programmed by the software to request a conversion of one analog input channel. The CCW table is written by software and is not modified by the QADC64E. 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, where they may be written when the software initializes the QADC64E. The remaining 6-bits are unimplemented so these 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 QADC64E provides 64 CCW table entries.

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 BQ2 is programmed to 64 or greater. 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 13-7 illustrates the operation of the queue structure.

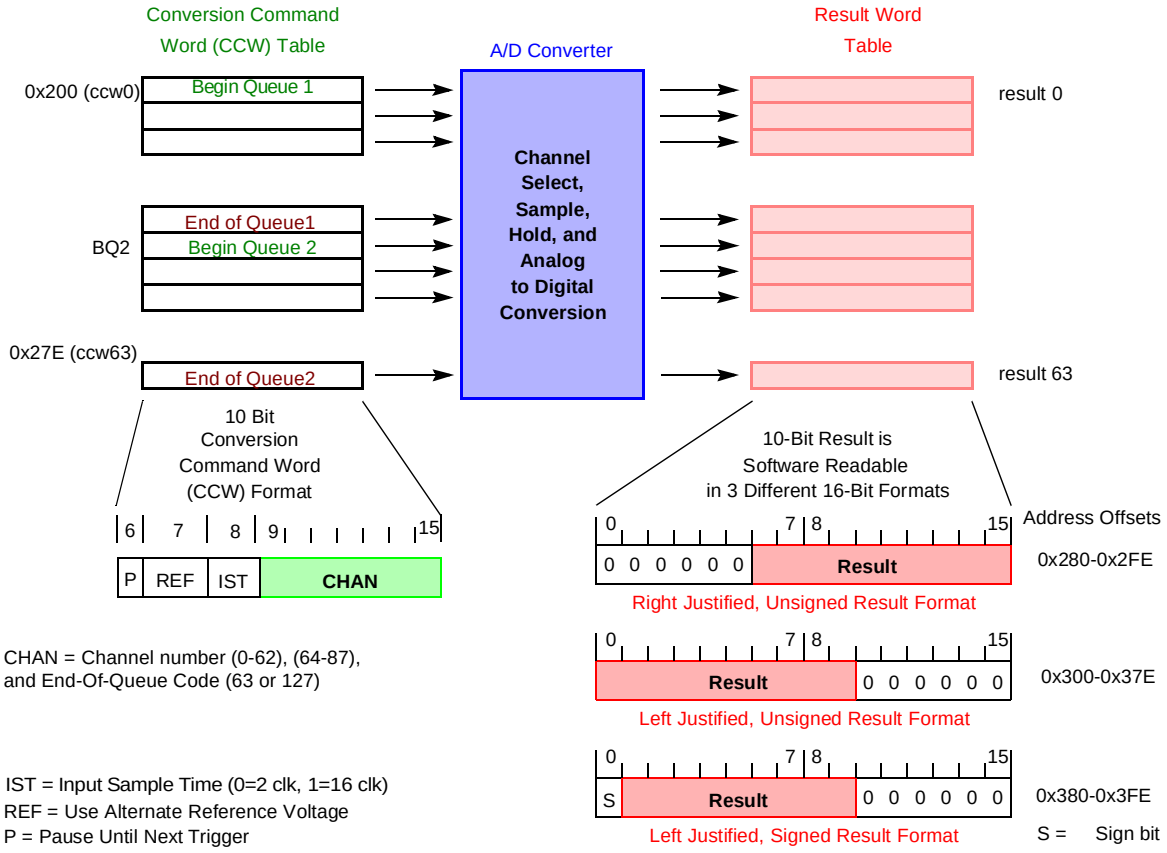


Figure 13-7 QADC64E Conversion Queue Operation

To prepare the QADC64E 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. A “trigger event” is used to refer to any of the ways to cause the QADC64E to begin executing the CCWs in a queue or sub-queue. 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
- External gated signal (queue 1 only)

The software also specifies whether the QADC64E 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 QADC64E completes each queue scan sequence.

During queue execution, the QADC64E reads each CCW from the active queue and executes conversions in three stages:

- Initial sample
- Final sample
- Resolution

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

During the final sample period, the sample buffer amplifier is bypassed, and the multiplexer input charges the sample capacitor directly. Each CCW specifies a final input sample time of two 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 QADC64E 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 QADC64E 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 QADC64E 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 (0x3F) or 127 (0x7F) 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 sub-queue being executed when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The RESUME bit 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 sub-queue 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 conversion in progress 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 conversion in progress 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 QADC64E 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 QADC64E freezes at the end of the conversion in progress. When internal FREEZE is negated, the QADC64E resumes queue execution beginning with the next CCW entry. Refer to **13.4.7 Configuration And Control using the IMB Interface** for more information.

CCW — Conversion Command Word Table **0x30 4A00 – 0x30 4A7F**
0x30 4E00 – 0x30 4E7F

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED						P	REF	IST	CHAN[6:0]						
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 13-18 CCW Bit Descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	P	Pause — The pause bit allows software to create sub-queues within queue 1 and queue 2. The QADC64E 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 **NOTE:** The pause bit will not cause the queue to pause in the software controlled modes or external gated modes.
7	REF	Alternate Reference Enabled — Setting REF high in the CCW enables the use of an alternate reference. 0 = VRH is used as high reference 1 = AltRef pin is used as the high reference

Table 13-18 CCW Bit Descriptions (Continued)

Bit(s)	Name	Description
8	IST	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 QADC64E analog channel inputs. Longer sample times permit more accurate A/D conversions of signals with higher source impedances. 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. **Table 13-19** shows the two selectable input sample times.
9:15	CHAN[6:0]	CHAN[6:0] — 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 **13.1.5 External Multiplexing** for more information on external multiplexing. **Table 13-20** shows the channel number assignments

Table 13-19 Input Sample Times

IST	Input Sample Times
0	Input sample time = QCLK period x 2
1	Input sample time = QCLK period x 16

Table 13-20 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				Channel Number in CCW CHAN Field	
Port Pin Name	**Analog Pin Name**	**Other Functions / Descriptions**	**Pin Type**	**Binary**	**Decimal**
ANw/ A_PQB[0]	AN[00] to AN[07]	—	Input	0000000 to 0000111	0 to 7
ANx/A_PQB[1]	AN[08] to AN[15]	—	Input	0001000 to 0001111	8 to 15
ANy/A_PQB[2]	AN[16] to AN[23]	—	Input	0010000 to 0010111	16 to 23
ANz/A_PQB[3]	AN[24] to AN[31]	—	Input	0011000 to 0011111	24 to 31
—	RESERVED	—	—	0100000 to 0101001	32 to 41
—	RESERVED	—	—	0101010	42
—	RESERVED	—	—	0101011	43
A_PQB[0] A_PQB[1] A_PQB[2] A_PQB[3]	AN[44] AN[45] AN[46] AN[47]	ANw ANx ANy ANz	Input/Output Input/Output Input/Output Input/Output	0101100 0101101 0101110 0101111	44 45 46 47
A_PQB[4] A_PQB[5] A_PQB[6] A_PQB[7]	AN[48] AN[49] AN[50] AN[51]	— — — —	Input/Output Input/Output Input/Output Input/Output	0110000 0110001 0110010 0110011	48 49 50 51
A_PQA[0] A_PQA[1] A_PQA[2] A_PQA[3]	AN[52] AN[53] AN[54] AN[55]	MA[0] MA[1] MA[2] —	Input/Output Input/Output Input/Output Input/Output	0110100 0110101 0110110 0110111	52 53 54 55
A_PQA[4] A_PQA[5] A_PQA[6] A_PQA[7]	AN[56] AN[57] AN[58] AN[59]	— — — —	Input/Output Input/Output Input/Output Input/Output	0111000 0111001 0111010 0111011	56 57 58 59
VRL VRH/ALTREF[1] —	Low Ref High Ref —	— — (VRH – VRL)/2	Input Input —	0111100 0111101 0111110	60 61 62
—	—	End of Queue Code	—	0111111	63

Table 13-20 Multiplexed Channel Assignments and Pin Designations (Continued)

Multiplexed Input Pins				Channel Number in CCW CHAN Field	
Port Pin Name	Analog Pin Name	Other Functions / Descriptions	Pin Type	Binary	Decimal
B_PQB[0]	AN[64]	—	AMUX Input	1000000	64
B_PQB[1]	AN[65]	—	AMUX Input	1000001	65
B_PQB[2]	AN[66]	—	AMUX Input	1000010	66
B_PQB[3]	AN[67]	—	AMUX Input	1000011	67
B_PQB[4]	AN[68]	—	AMUX Input	1000100	68
B_PQB[5]	AN[69]	—	AMUX Input	1000101	69
B_PQB[6]	AN[70]	—	AMUX Input	1000110	70
B_PQB[7]	AN[71]	—	AMUX Input	1000111	71
B_PQA[0]	AN[72]	MA[0]	AMUX Input	1001000	72
B_PQA[1]	AN[73]	MA[1]	AMUX Input	1001001	73
B_PQA[2]	AN[74]	MA[2]	AMUX Input	1001010	74
B_PQA[3]	AN[75]	—	AMUX Input	1001011	75
B_PQA[4]	AN[76]	—	AMUX Input	1001100	76
B_PQA[5]	AN[77]	—	AMUX Input	1001101	77
B_PQA[6]	AN[78]	—	AMUX Input	1001110	78
B_PQA[7]	AN[79]	—	AMUX Input	1001111	79
—	AN[80]	—	AMUX Input	1010000	80
—	AN[81]	—	AMUX Input	1010001	81
—	AN[82]	—	AMUX Input	1010010	82
—	AN[83]	—	AMUX Input	1010011	83
—	AN[84]	—	AMUX Input	1010100	84
—	AN[85]	—	AMUX Input	1010101	85
—	AN[86]	—	AMUX Input	1010110	86
—	AN[87]	—	AMUX Input	1010111	87
—	RESERVED	—	—	1011000 to 1111110	88 to 126
—	—	End of Queue Code	—	1111111	127

NOTES:
1. Whichever is selected in the CCW.

RESERVED CHANNELS

The channel field is programmed for channel 63 or 127 to indicate the end of the queue. Channels 60 to 62 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_{RH} - V_{RL})/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.

13.2.10 Result Word Table

The result word table is a RAM, 64 words long and 10 bits wide. An entry is written by the QADC64E 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

QADC64E. Unimplemented bits are read as zeros, and write operations do not have any effect. See **Figure 13-7** for a diagram of the result word table

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 **0x30 4A80 – 0x30 4AFF**
0x30 4E80 – 0x30 4EFF

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED						RESULT									
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

LJSRR — Left Justified, Signed Result Format **0x30 4B00 – 0x03 4B7F**
0x30 4F00 – 0x03 4F7F

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
S[1]	RESULT									RESERVED					
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

NOTES:
1. S = Sign bit.

LJURR — Left Justified, Unsigned Result Register **0x30 4B80 – 0x03 4BFF**
0x30 4F80 – 0x03 4FFF

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESULT										RESERVED					
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

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.

13.3 Analog Subsystem

This section describes the QADC64E analog subsystem, which includes the front-end analog multiplexer and analog-to-digital converter.

13.3.1 Analog-to-Digital Converter Operation

The analog subsystem consists of the path from the input pins to the A/D converter block. Signals from the queue control logic are fed to the multiplexer and state machine. The end of convert (EOC) signal and the successive-approximation register (SAR) are the result of the conversion. **Figure 13-8** shows a block diagram of the QADC64E analog subsystem.

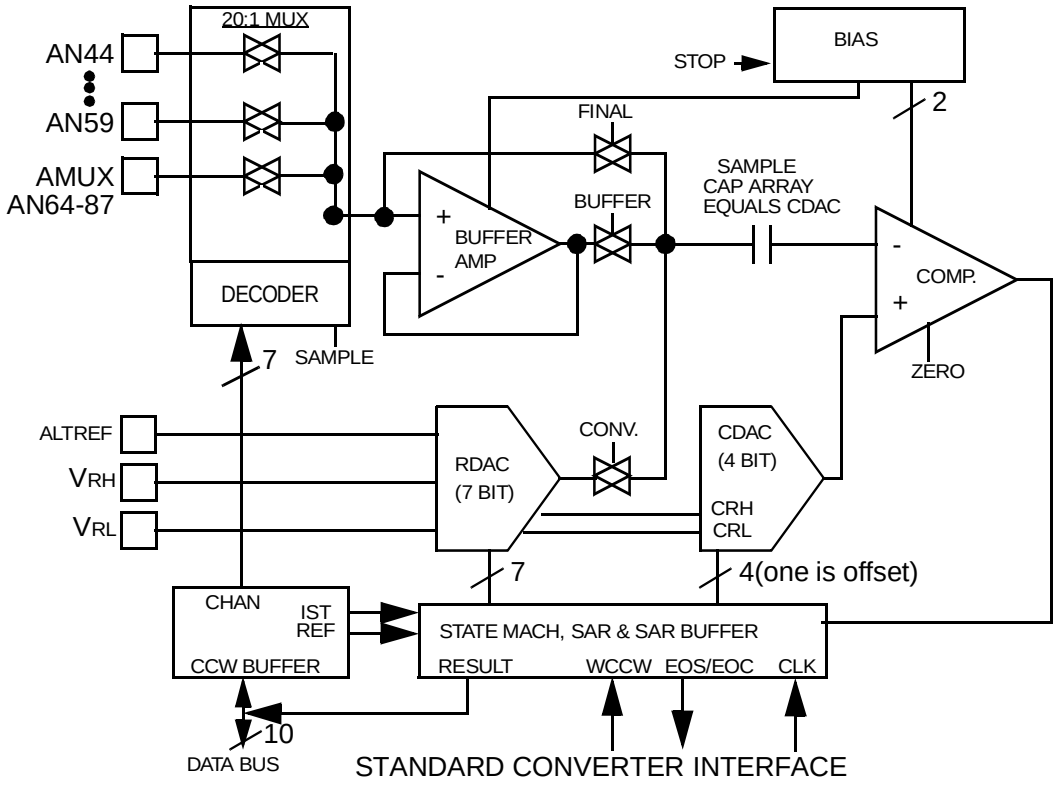


Figure 13-8 QADC64E Analog Subsystem Block Diagram

13.3.1.1 Conversion Cycle Times

Total conversion time is made up of initial sample time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is coupled through the buffer amplifier to the sample capacitor. This buffer is used to quickly reproduce its input signal on the sample capacitor and minimize charge sharing errors. During the final sampling period the amplifier is bypassed, and the multiplexer input charges the sample capacitor array directly for improved accuracy. During the resolution period, the voltage in the sample capacitor is converted to a digital value and stored in the SAR.

Initial sample time is fixed at two QCLK cycles. Final sample time can be two or sixteen QCLK cycles, depending on the value of the IST field in the CCW. Resolution time is ten QCLK cycles.

Therefore, conversion time requires a minimum of 14 QCLK clocks (seven µs with a 2.0-MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 28 QCLKs or 14 µs (with a 2.0-MHz QCLK)

Figure 13-9 illustrates the timing for conversions.

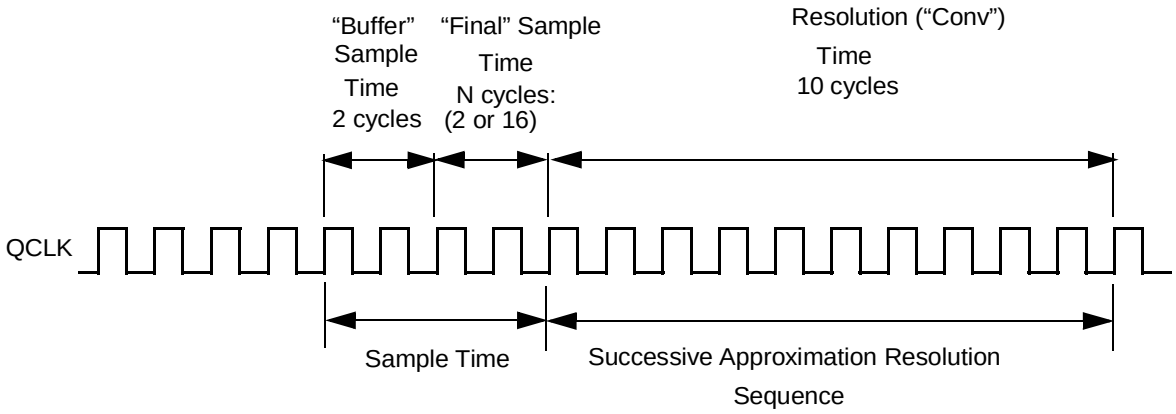


Figure 13-9 Conversion Timing

13.3.2 Channel Decode and Multiplexer

The internal multiplexer selects one of the 40 analog input pins for conversion. The selected input is connected to the sample buffer amplifier. The multiplexer also includes positive and negative stress protection circuitry, which prevents deselected channels from affecting the selected channel when current is injected into the deselected channels. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for specific current levels.

13.3.3 Sample Buffer Amplifier

The sample buffer is used to raise the effective input impedance of the A/D converter, so that external components (higher bandwidth or higher impedance) are less critical to accuracy. The input voltage is buffered onto the sample capacitor to reduce crosstalk between channels.

13.3.4 Digital to Analog Converter (DAC) Array

The digital to analog converter (DAC) array consists of binary-weighted capacitors and a resistor-divider chain. The reference voltages, V_{RH} and V_{RL}, are used by the DAC to perform ratiometric conversions. The DAC also converts the following three internal channels:

- V_{RH} — Reference voltage high
- V_{RL} — Reference voltage low
- $(V_{RH} - V_{RL})/2$ — Reference voltage

The DAC array serves to provide a 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 comparator and successive-approximation register (SAR) logic.

- Sample Capacitor — The sample capacitor is employed to sample and hold the voltage to be converted.

13.3.5 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, beginning with the MSB.

13.3.6 Bias

The bias circuit is controlled by the STOP signal to power-up and power-down all the analog circuits.

13.3.7 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.

13.3.8 State Machine

The state machine receives the QCLK, RST, STOP, IST, CHAN[6:0], and START CONV. signals, from which it generates all timing to perform an A/D conversion. The start convert (start conv.) signal indicates to the A/D converter that the desired channel has been sent to the MUX. IST indicates the desired sample time. BYP indicates whether to bypass the sample amplifier. The end of conversion (EOC) signal, notifies the queue control logic that a result is available for storage in the result RAM.

13.4 Digital Subsystem

The digital control subsystem includes the control logic to sequence the conversion activity, 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 QADC64E conversions is the 64-entry 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, sub queues can be created in the two queues. Each queue can be operated using one of 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 QADC64E to begin executing the CCWs

in a queue or sub-queue), the QADC64E performs a sequence of conversions and places the results in the result word table.

13.4.1 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 captured 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 captured 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 **13.2.7 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.

13.4.2 Sub-Queues That are Paused

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

Figure 13-10 shows the CCW format and an example of using pause to create sub-queues. Queue 1 is shown with four CCWs in each sub-queue and queue 2 has two CCWs in each sub-queue.

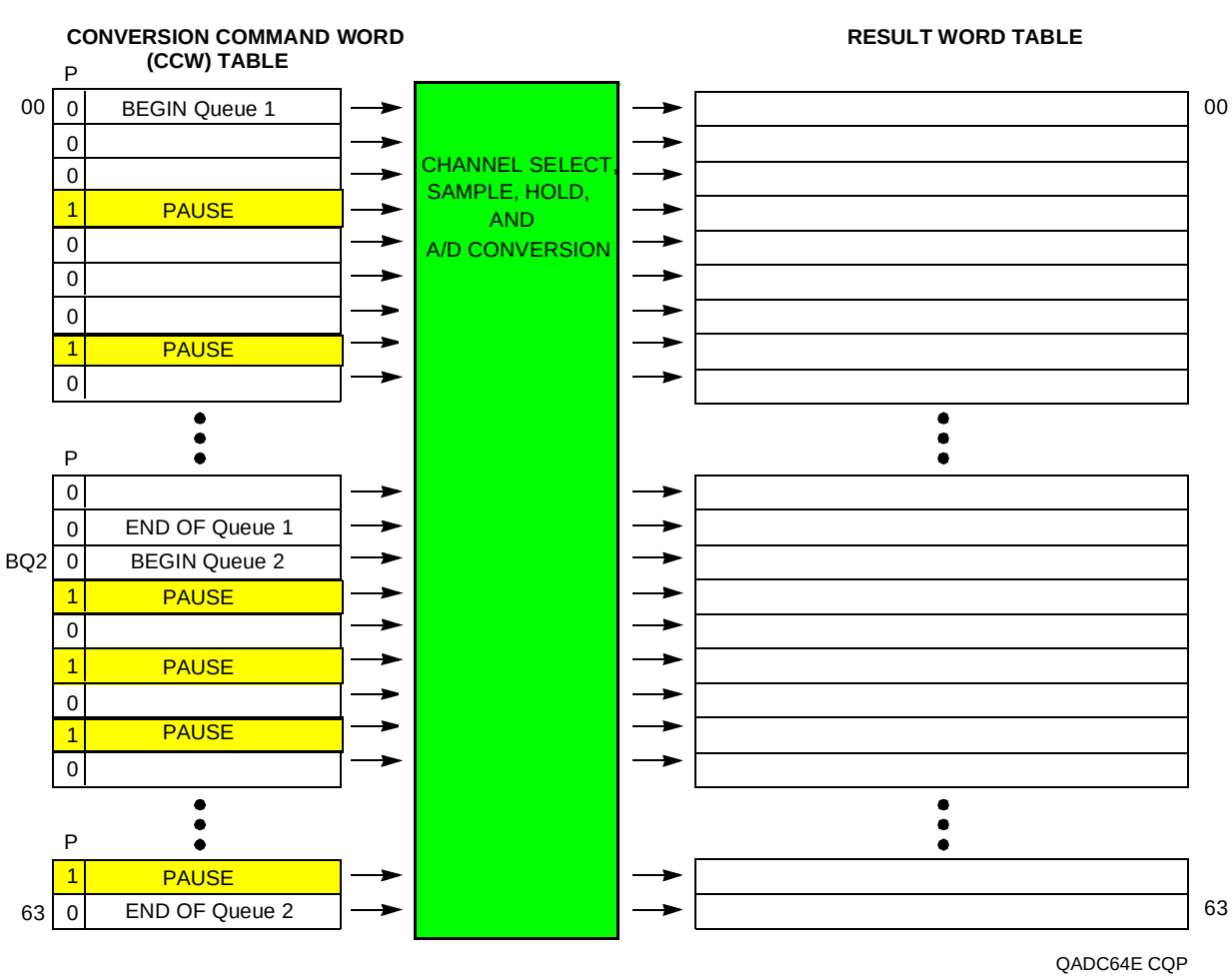


Figure 13-10 QADC64E Queue Operation With Pause

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

EXAMPLE

When the external trigger rising edge continuous-scan mode is selected for queue 1, and there are six sub-queues within queue 1, a separate rising edge is required on the external trigger pin after every pause to begin the execution of each sub-queue (refer to **Figure 13-10**). Refer to **13.4.4 Scan Modes** for information on different scan modes.

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each sub-queue. Once a sub-queue is initiated, each CCW is executed sequentially until the last CCW in the sub-queue is executed and the pause state is entered. Execution can only continue with the next CCW, which is the beginning of the

next sub-queue. A sub-queue cannot be executed a second time before the overall queue execution has been completed. Refer to **13.2.7 Control Register 2** for more information.

Trigger events which occur during the execution of a sub-queue 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 sub-queue (after the queue completion flag is set), causes the execution to continue with the first sub-queue, starting with the first CCW in the queue.

When the QADC64E 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 sub-queue. The QADC64E then waits for another trigger event to again begin execution of the next sub-queue.

13.4.3 Boundary Conditions

The following are queue operation boundary conditions:

- The first CCW in a queue contains channel 63 or 127, 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 at the end of the CCW table (63 or 127) and a trigger event occurs on queue 2. Refer to **13.2.7 Control Register 2** for more information on BQ2. The end-of-queue condition is recognized, a conversion is performed, the completion flag is set, and the queue becomes idle.
- 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.
- BQ2 is set beyond the end of the CCW table (64 – 127) and a trigger event occurs on queue 2. The end-of-queue condition is recognized immediately, 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 QADC64E behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC64E 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 com-

pletion 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 or 127 (EOQ) code is in CCW6
- The pause is in CCW63
- 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 QADC64E 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

13.4.4 Scan Modes

The QADC64E 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 are executed. The possible modes are:

- Disabled and reserved mode
- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode
- Interval timer single-scan mode
- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode
- Periodic timer continuous-scan mode

The following paragraphs describe single-scan and continuous-scan operations.

13.4.4.1 Disabled Mode

When the disabled 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 QCLK prescaler values.

13.4.4.2 Reserved Mode

Reserved mode allows for future mode definitions. When the reserved mode is selected, the queue is not active. It functions the same as disabled mode.

CAUTION

Do not use a reserved mode. Unspecified operations may result.

13.4.4.3 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 MQ field in QACR1 or QACR2, the following modes can be selected:

- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode
- Interval timer single-scan mode

NOTE

Queue 2 cannot be programmed for external gated single-scan mode.

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 is set through software, but will always read as a zero. Once set, writing the single-scan enable bit to zero has no effect. Only the QADC64E can clear the single-scan enable bit. 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 QADC64E 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 QADC64E 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 conversion in progress 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 QADC64E 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. The single-scan enable bit allows the entire queue to be scanned once. A trigger overrun is captured if a trigger event occurs during queue execution in an edge-sensitive external trigger mode or a periodic/interval timer mode.

In the periodic/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.

13.4.4.4 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 QADC64E immediately begins execution of the first CCW in the queue. If a pause occurs, another trigger event is generated internally, and then execution continues without pausing.

The QADC64E 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.

13.4.4.5 External Trigger Single-Scan Mode

The external trigger single-scan mode is available on 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 QADC64E 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.

When a pause bit is encountered during external trigger 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.

13.4.4.6 External Gated Single-Scan Mode

The QADC64E provides external gating for queue 1 only. When external gated single-scan mode is selected, the input level on the associated external trigger pin enables and disables queue execution. The polarity of the external gated signal is fixed so only a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. Software must enable the scan to occur by setting the single-scan enable bit for queue 1. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

While the gate is open, queue 1 executes one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When queue 1 completes, the QADC64E sets the completion flag (CF1) and clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of queue 1 to be initiated during the next open gate.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops, the single-scan enable bit is cleared, and the PF1 bit is set. Software can read the CWPQ1 to determine the last valid conversion in the queue. Software must set the single-scan enable bit again and should clear the PF1 bit before another scan of queue 1 is initiated during the next open gate. The start of queue 1 is always the first CCW in the CCW table.

Since the condition of the gate is only sampled after each conversion during queue execution, closing the gate for a period less than a conversion time interval does not guarantee the closure will be captured.

13.4.4.7 Periodic/Interval Timer Single-Scan Mode

Both queues can use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 128- to 128-Kbyte QCLK cycles in binary multiples. When the periodic/ interval timer single-scan mode is selected and the software sets the single-scan enable bit in QACR1 or QACR2 , the timer begins counting. When the time interval elapses, an internal trigger event is created to start the queue and the QADC64E begins execution with the first CCW.

The QADC64E 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 elapses again, and then queue execution continues. When the queue execution reaches an end-of-queue situation, 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 periodic/interval timer.

The periodic/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.

Normally only one queue will be enabled for periodic/interval timer single-scan mode and the timer will reset at the end-of-queue. However, if both queues are enabled for either single-scan or continuous periodic/interval timer mode, the end-of-queue condition will not reset the timer while the other queue is active. In this case, the timer will reset when both queues have reached end-of-queue. See **13.4.6 Periodic/Interval Timer** for a definition of periodic/interval timer reset conditions.

The periodic/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/interval 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

13.4.4.8 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 or the MQ2 field in QACR2, the following software initiated modes can be selected:

- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode
- Periodic/interval timer continuous-scan mode

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 continuous-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger continuous-scan mode and the periodic/interval timer continuous-scan mode.

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

Coherent samples are guaranteed. The time between consecutive conversions has been designed to be consistent. However, there is one exception. For queues that end with a CCW containing EOQ code (channel 63 or 127), the last queue conversion to the first queue conversion requires 1 additional CCW fetch cycle. Therefore continuous samples are not coherent at this boundary.

In addition, the time from trigger to first conversion cannot be guaranteed since it is a function of clock synchronization, programmable trigger events, queue priorities, and so on.

13.4.4.9 Software Initiated Continuous-Scan Mode

When the software initiated continuous-scan mode is programmed, the trigger event is generated automatically by the QADC64E. Queue execution begins immediately. If a pause is encountered, another trigger event is generated internally, and then execution continues without pausing. 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 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 QADC64E without software involvement. Software can read a result value at any time.

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 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.

13.4.4.10 External Trigger Continuous-Scan Mode

The QADC64E 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 select a mode which begins queue execu-

tion 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.

When a pause bit is encountered in external trigger continuous-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.

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.

13.4.4.11 External Gated Continuous-Scan Mode

The QADC64E provides external gating for queue 1 only. When external gated continuous-scan mode is selected, the input level on the associated external trigger pin enables and disables queue execution. The polarity of the external gated signal is fixed so a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. When the gate opens again, the queue execution automatically begins again from the beginning of the queue. Software initialization is not needed between trigger events. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

The purpose of external gated continuous-scan mode is to continuously collect digitized samples while the gate is open and to have the most recent samples available. It is up to the programmer to ensure that the queue is large enough so that a maximum gate open time will not reach an end-of-queue. However it is useful to take advantage of a smaller queue in the manner described in the next paragraph.

In the event that the queue completes before the gate closes, a completion flag will be set and the queue will roll over to the beginning and continue conversions until the gate closes. If the gate remains open and the completion flag is not cleared, when the queue completes a second time the trigger overrun flag will be set and the queue will roll-over again. The queue will continue to execute until the gate closes or the mode is disabled.

If the gate closes before queue 1 completes execution, the current CCW completes execution of queue 1 stops and QADC64E sets the PF1 bit to indicate an incomplete queue. Software can read the CWPQ1 to determine the last valid conversion in the queue. In this mode, if the gate opens again, execution of queue 1 begins again. The start of queue 1 is always the first CCW in the CCW table.

Since the condition of the gate is only sampled after each conversion during queue execution, closing the gate for a period less than a conversion time interval does not guarantee the closure will be captured.

13.4.4.12 Periodic/Interval Timer Continuous-Scan Mode

The QADC64E includes a dedicated periodic/interval timer for initiating a scan sequence on queue 1 and/or queue 2. Software selects a programmable timer interval ranging from 128 to 128 Kbytes times the QCLK period in binary multiples. The QCLK period is prescaled down from the IMB MCU clock.

When a periodic/interval timer continuous-scan mode is selected for queue 1 and/or queue 2, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the appropriate queue. Meanwhile, the QADC64E automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC64E waits for the periodic interval to expire again, then continues with the queue. Once end-of-queue has been detected, the next trigger event causes queue execution to begin again with the first CCW in the queue.

The periodic/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 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. Since both queues may be triggered by the periodic/interval timer, see **13.4.6 Periodic/Interval Timer** for a summary of periodic/interval timer reset conditions.

Software enables the completion interrupt when using the periodic/interval 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.

13.4.5 QADC64E Clock (QCLK) Generation

Figure 13-11 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 E ELECTRICAL CHARACTERISTICS**.

Before using the QADC64E, 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.

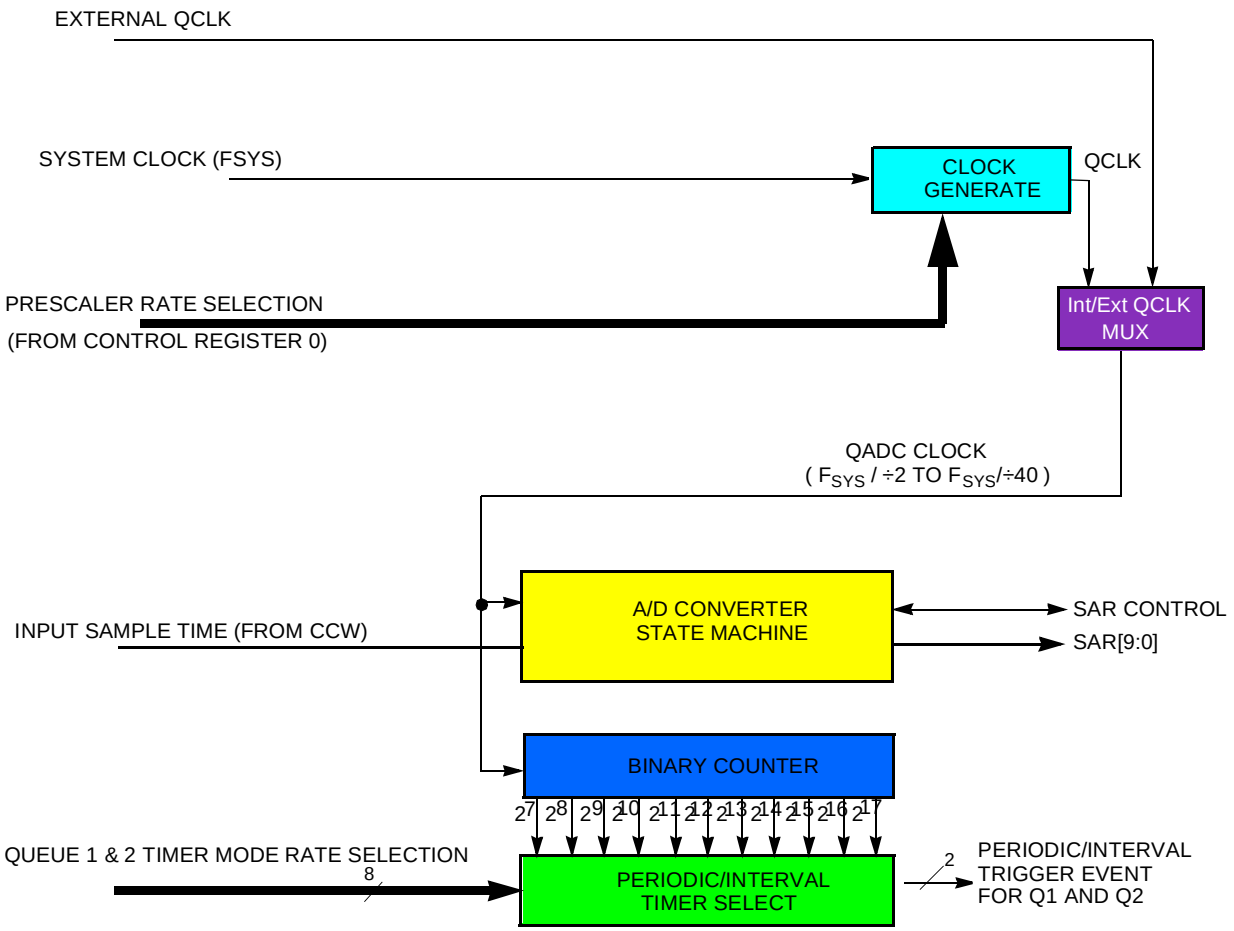


Figure 13-11 QADC64E Clock Subsystem Functions

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 allow the A/D conversion time to be maximized across the spectrum of system clock frequencies, the QADC64E prescaler permits the frequency of QCLK to be software selectable. The software establishes the frequency of the QCLK waveform by setting the PRESCALAR field in QACR0. The frequency resulting from a value in the PRESCALAR field that is > 0, is calculated by the following formula:

$$F_{QCLK} = F_{SYSCLK} / (PRESCALAR + 1)$$

if the value in the PRESCALAR field is set to 0, the resulting QCLK frequency may be calculated to be:

$$F_{QCLK} = F_{SYSCLK} / 2$$

Table 13-21 QADC64E Clock Programmability

Control Register 0 Information			Input Sample Time (IST) =%00	
Example Number	Frequency	PRESCALER	QCLK (MHz)	Conversion Time (µs)
1	20 MHz	0x09	2.0	7.0
2	40 MHz	0x13	2.0	7.0
3	56 MHz	0x1B	2.0	7.0

The MCU system clock frequency is the basis of the QADC64E timing. The QADC64E requires that the system clock frequency be at least twice the QCLK frequency. The QCLK frequency is established by the prescaler parameter in QACR0.

13.4.6 Periodic/Interval Timer

The on-chip periodic/interval timer can be used to generate trigger events at a programmable interval, initiating execution of queue 1 and/or queue 2. The periodic/interval timer stays reset under the following conditions:

- Both queue 1 and queue 2 are programmed to any mode which does not use the periodic/interval timer
- IMB system reset or the master reset is asserted
- Stop mode is selected
- Freeze mode is selected

NOTE

Interval timer single-scan mode does not use the periodic/interval timer until the single-scan enable bit is set.

The following two conditions will cause a pulsed reset of the periodic/interval timer during use:

- A queue 1 operating mode change to a mode which uses the periodic/interval timer, even if queue 2 is already using the timer
- A queue 2 operating mode change to a mode which uses the periodic/interval timer, provided queue 1 is not in a mode which uses the periodic/interval timer
- Roll over of the timer

During the low power stop mode, the periodic timer is held in reset. Since low power stop mode causes QACR1 and QACR2 to be reset to zero, a valid periodic or interval timer mode must be written 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 conversion in progress 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 **13.4.7 Configuration And Control using the IMB Interface** for more information.

13.4.7 Configuration And Control using the IMB Interface

The QADC64E module communicates with other microcontroller modules via the IMB. The QADC64E bus interface unit (BIU) coordinates IMB activity with internal QADC64E bus activity. This section describes the operation of the BIU, IMB read/write accesses to QADC64E memory locations, module configuration, and general-purpose I/O operation.

13.4.7.1 QADC64E 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
- The BIU state machine
- Clock prescaler logic
- Data bus routing logic
- Interface to the internal module data bus

NOTE

Normal accesses from the IMB to the QADC64E require two clocks. However, if the CPU tries to access table locations while the QADC64E is accessing them, the QADC64E produces IMB wait states. From one to four IMB wait states may be inserted by the QADC64E in the process of reading and writing.

13.4.7.2 QADC64E Bus Accessing

The QADC64E supports 8-bit, 16-bit, and 32-bit data transfers, at even and odd addresses. Coherency of results read (ensuring that all results read were taken consecutively in one scan) is not guaranteed. For example, if a read of two consecutive 16-bit locations in a result area is made, the QADC64E could change one 16-bit location in the result area between the bus cycles. There is no holding register for the second 16-bit location. All read and write accesses that require more than one 16-bit access to complete occur as two or more independent bus cycles. Depending on bus master protocol, these accesses could include misaligned and 32-bit accesses.

Figure 13-12 shows the three bus cycles which are implemented by the QADC64E. The following paragraphs describe how the three types of accesses are used, including misaligned 16-bit and 32-bit accesses.

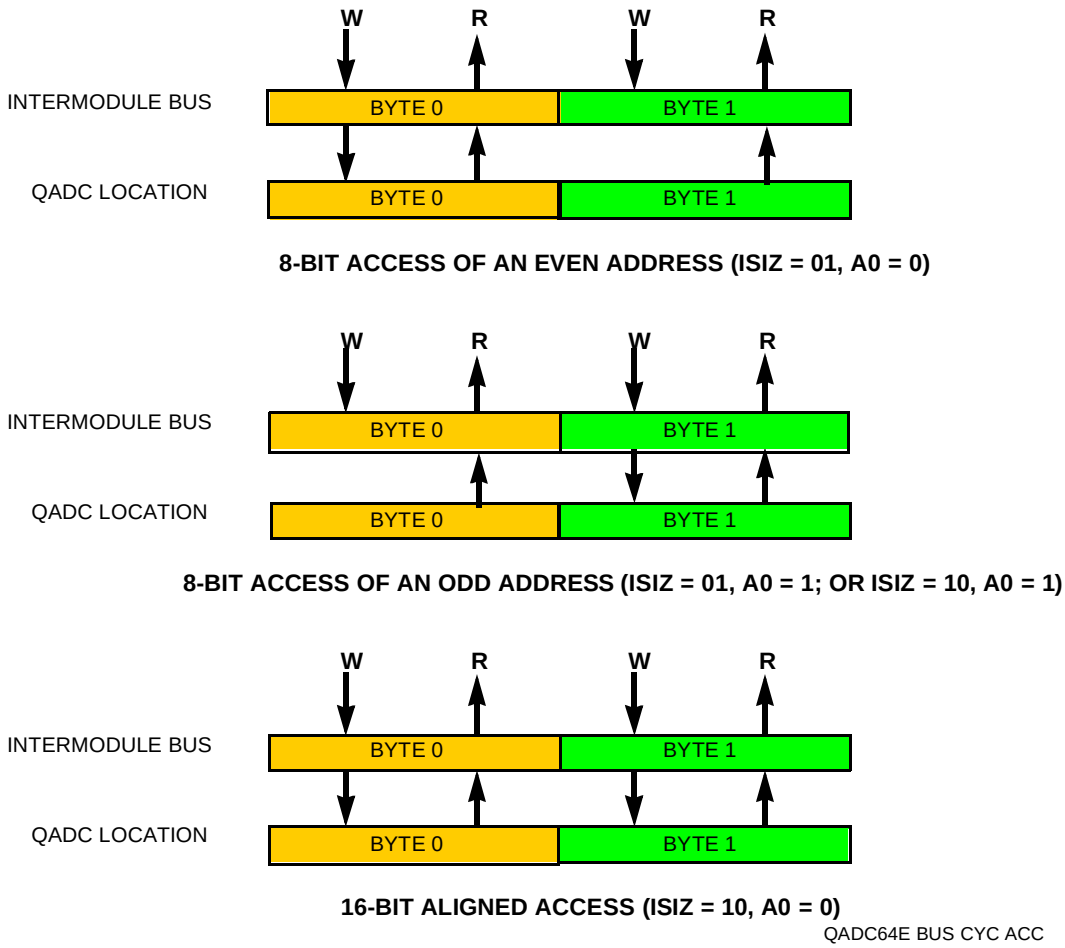


Figure 13-12 Bus Cycle Accesses

Byte access to an even address of a QADC64E location is shown in the top illustration of **Figure 13-12**. In the case of write cycles, byte1 of the register is not disturbed. In the case of a read cycle, the QADC64E provides both byte 0 and byte1.

Byte access to an odd address of a QADC64E location is shown in the center illustration of **Figure 13-12**. In the case of write cycles, byte 0 of the register is not disturbed. In the case of read cycles, the QADC64E provides both byte 0 and byte1.

16-bit accesses to an even address read or write byte 0 and byte 1 as shown in the lowest illustration of **Figure 13-12**. The full 16 bits of data is written to and read from the QADC64E location with each access.

16-bit accesses to an odd address require two bus cycles; one byte of two different 16-bit QADC64E locations is accessed. The first bus cycle is treated by the QADC64E 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 QADC64E 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 QADC64E location, and the upper half of the following QADC64E location.

32-bit accesses to an even address require two bus cycles to complete the access, and two full 16-bit QADC64E locations are accessed. The first bus cycle reads or writes the addressed 16-bit QADC64E location and the second cycle reads or writes the following 16-bit location.

32-bit accesses to an odd address require three bus cycles. Portions of three different QADC64E locations are accessed. The first bus cycle is treated by the QADC64E 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 QADC64E 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 QADC64E location, the full 16-bit content of the following QADC64E location, and the upper half of the third QADC64E location.

13.5 Trigger and Queue Interaction Examples

This section contains examples describing queue priority and conversion timing schemes.

13.5.1 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

The situations in **Figure 13-13** through **Figure 13-31** 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 13-22**.

Table 13-22 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 13-23** describes the status bits.

Table 13-23 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 (queue 2 only)
- 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 13-13** through **Figure 13-15** (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 13-13**), 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.

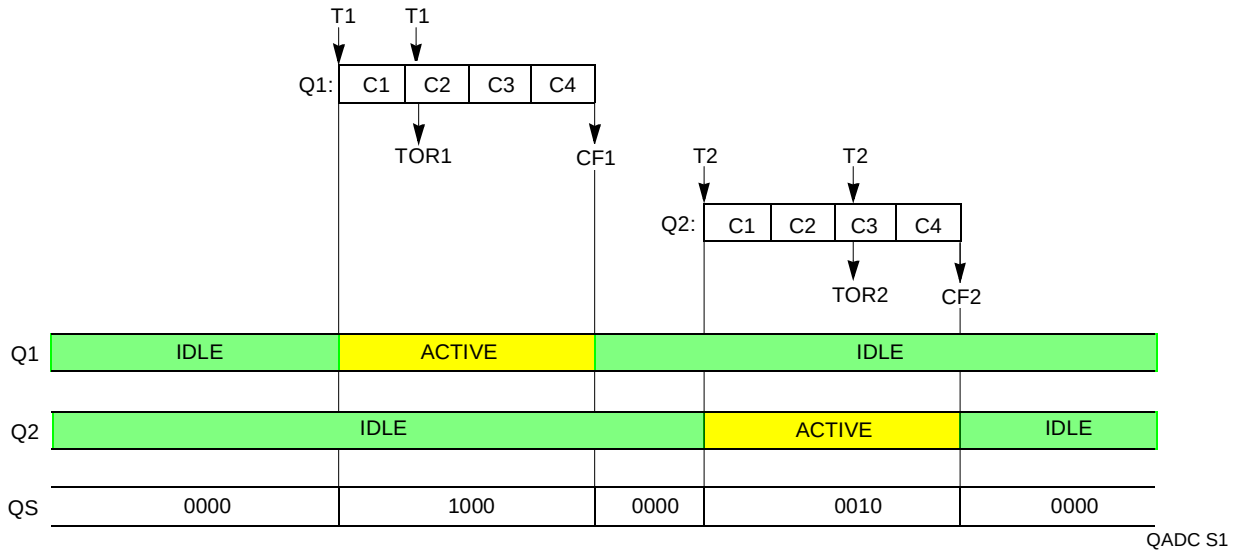


Figure 13-13 CCW Priority Situation 1

In situation S2 (**Figure 13-13**), 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.

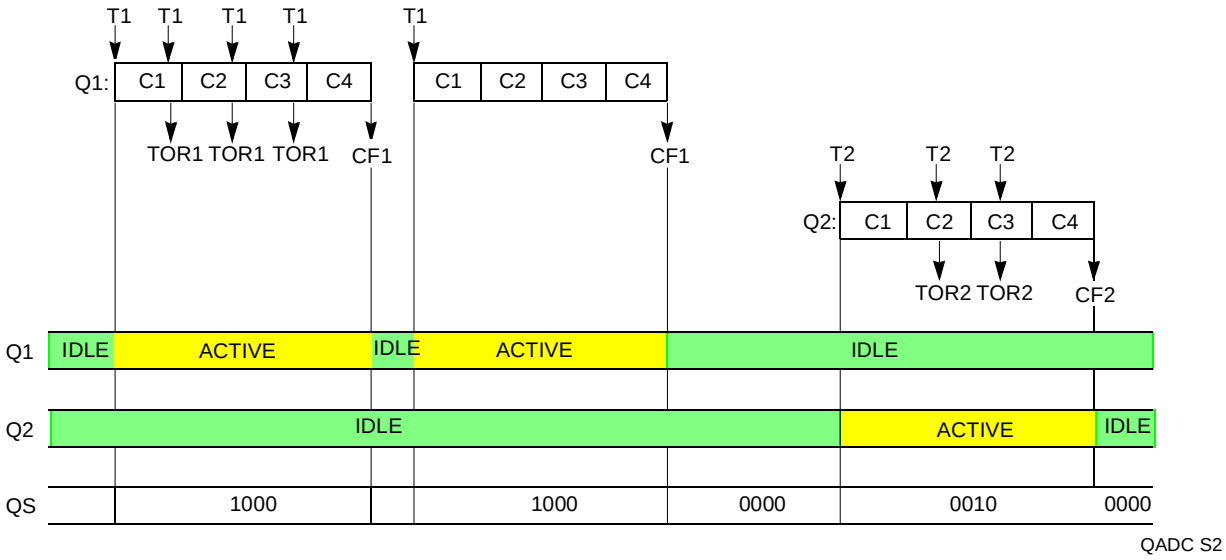


Figure 13-14 CCW Priority Situation 2

Situation S3 (**Figure 13-14**) 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.

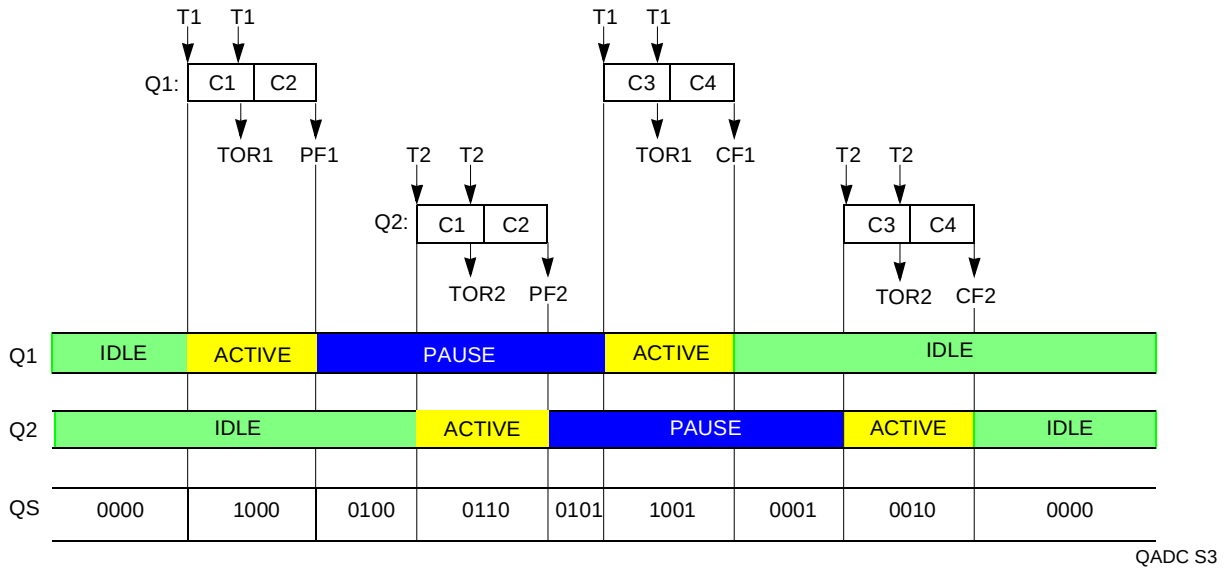


Figure 13-15 CCW Priority Situation 3

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

Situation S4 (**Figure 13-16**) 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.

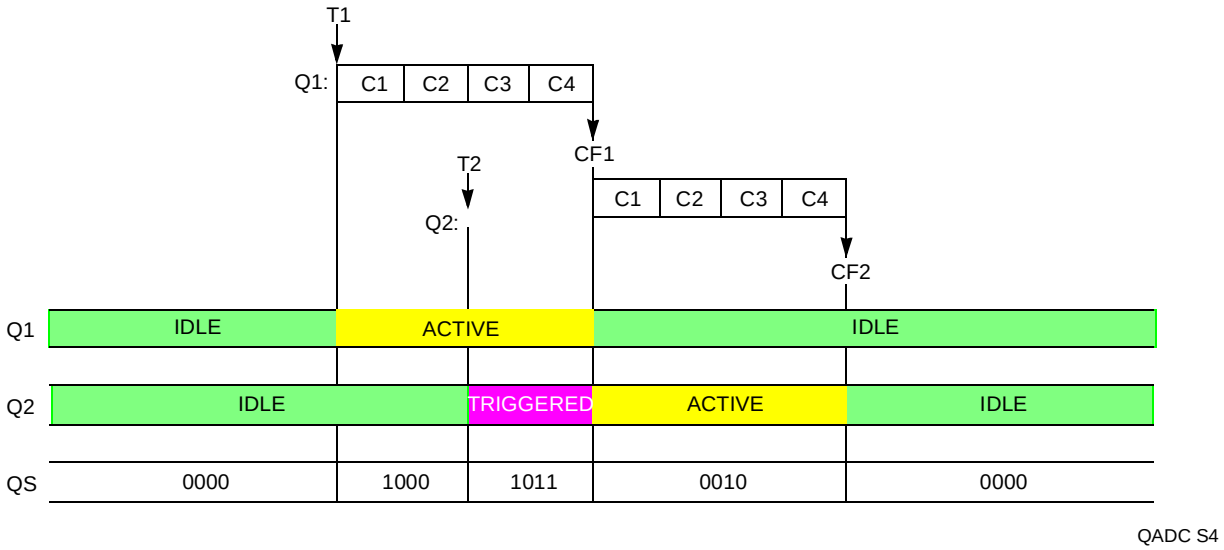


Figure 13-16 CCW Priority Situation 4

Situation S5 (**Figure 13-17**) 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.

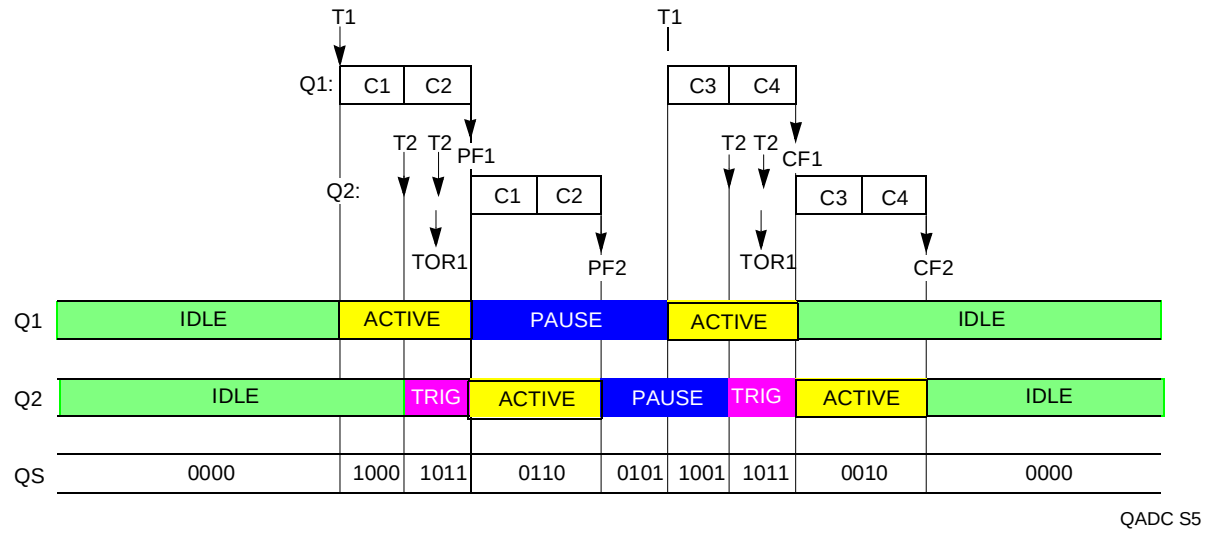


Figure 13-17 CCW Priority Situation 5

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 S6 (**Figure 13-18**), 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 13-19**) shows that when pause operation is not in use with queue 2, queue 2 suspension works the same way.

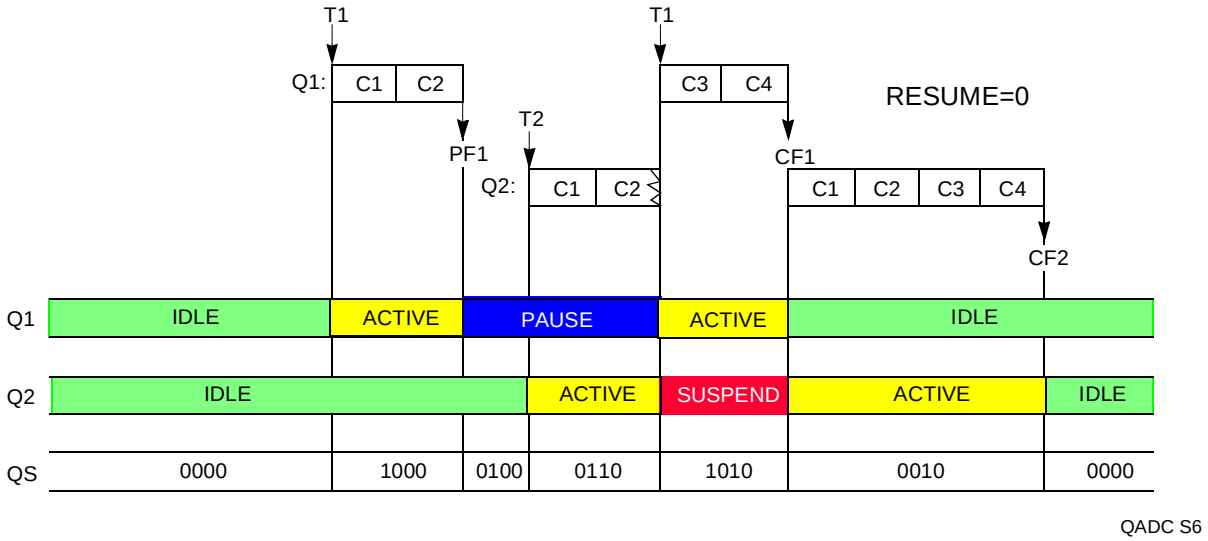


Figure 13-18 CCW Priority Situation 6

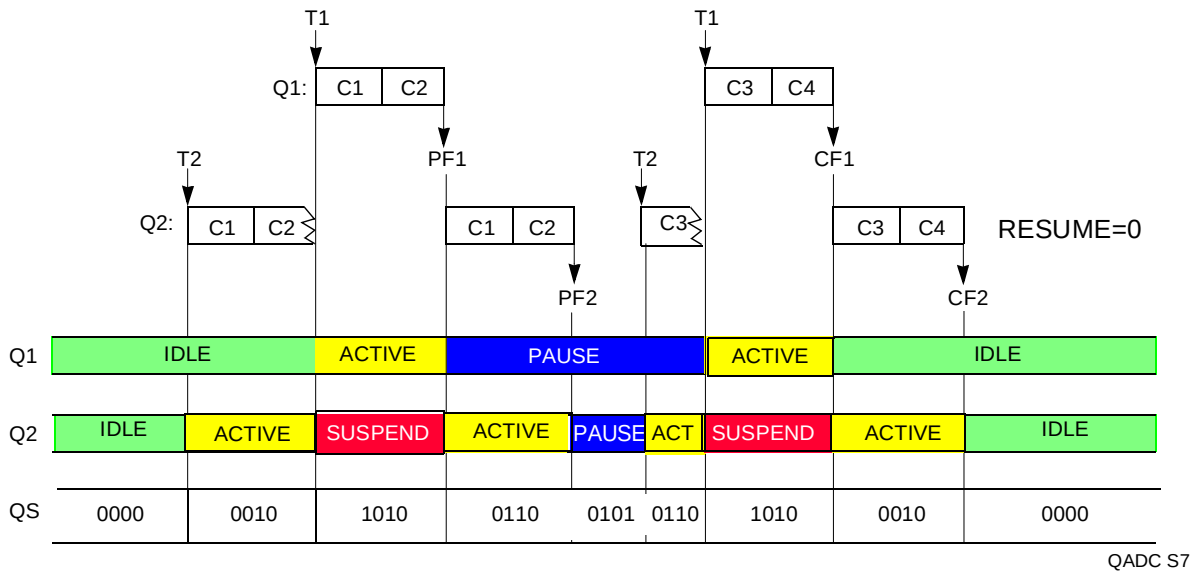


Figure 13-19 CCW Priority Situation 7

Situations S8 and S9 (**Figure 13-20** and **Figure 13-21**) 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.

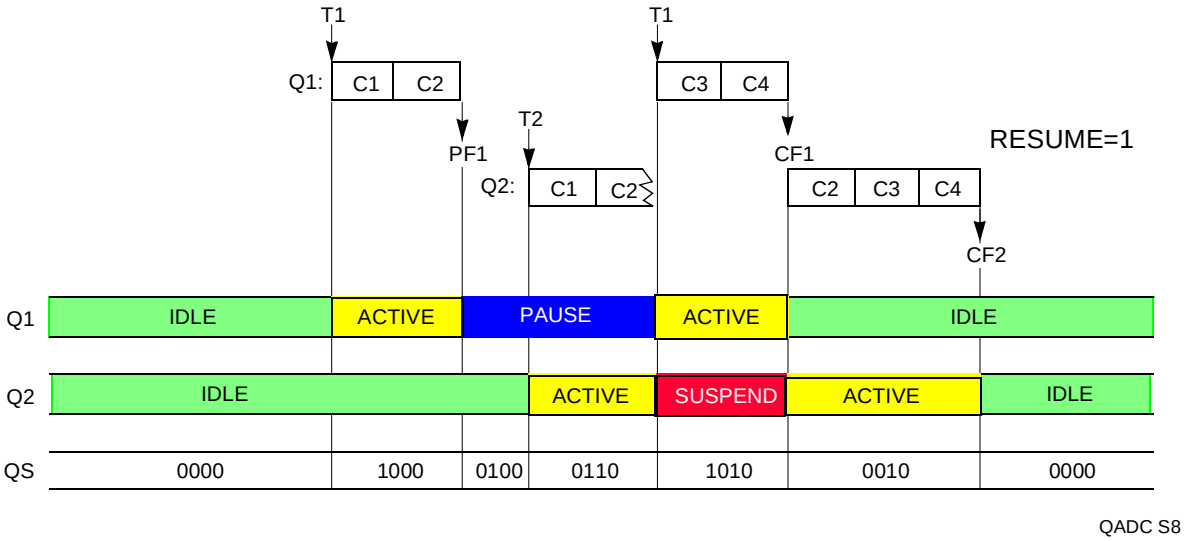


Figure 13-20 CCW Priority Situation 8

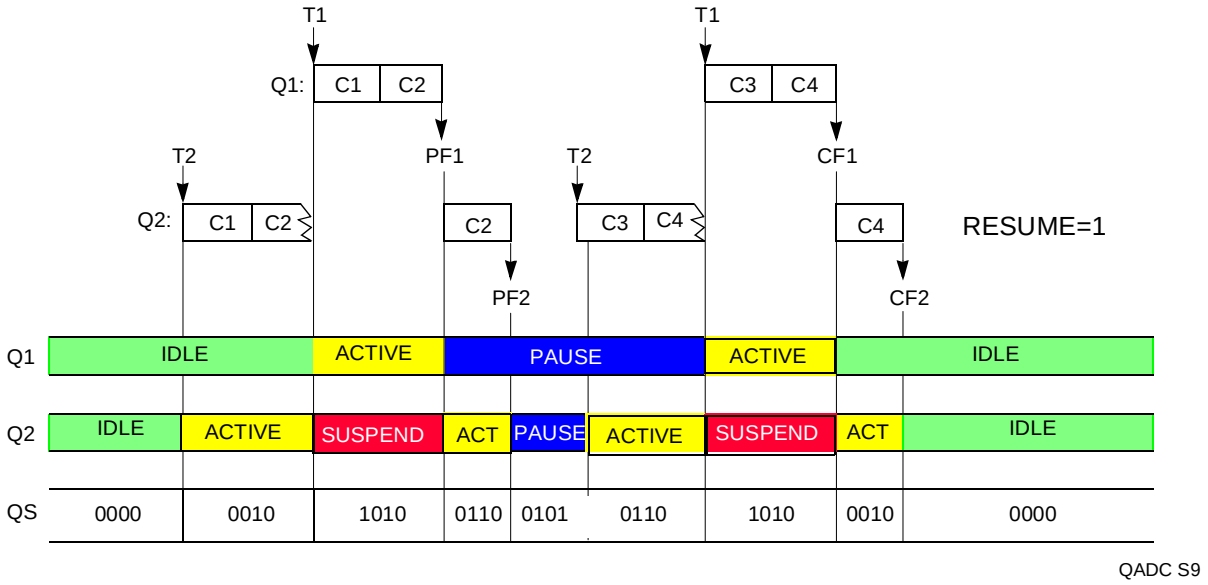


Figure 13-21 CCW Priority Situation 9

Situations S10 and S11 (Figures **Figure 13-22** and **Figure 13-23**) 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 QADC64E time that queue 2 trigger events are being lost.

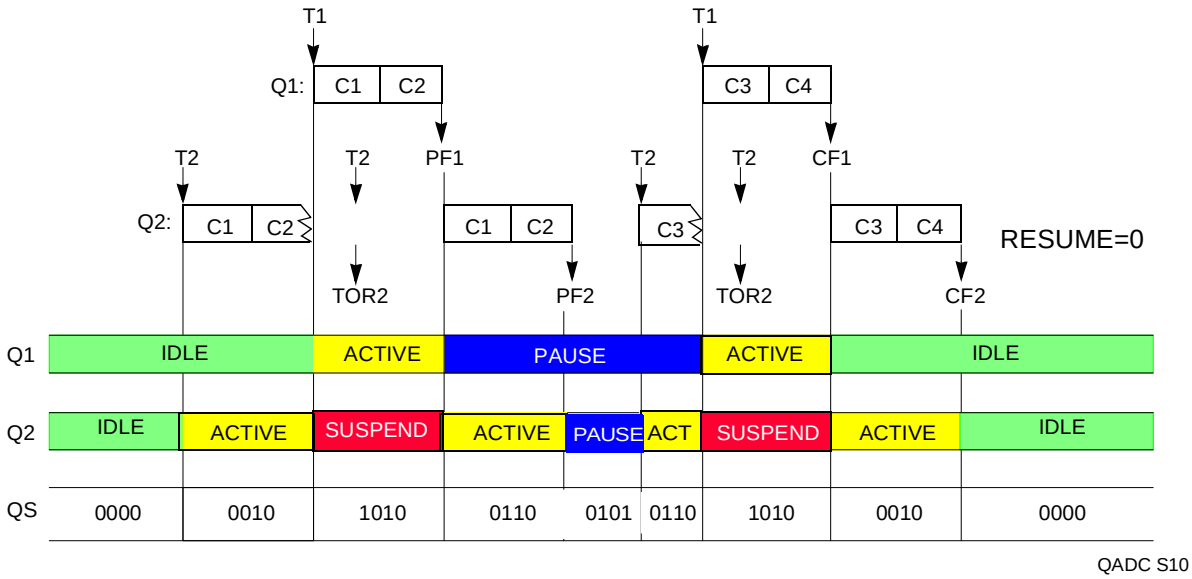


Figure 13-22 CCW Priority Situation 10

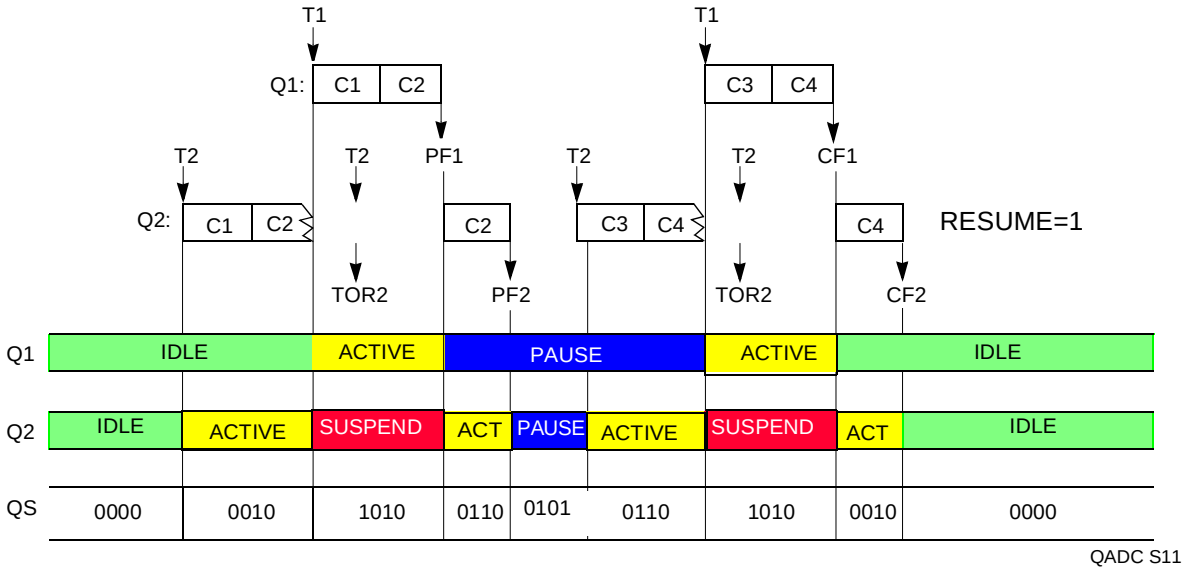


Figure 13-23 CCW Priority Situation 11

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 QADC64E 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 QADC64E completes the conversion in progress, unlike queue 1 suspending queue 2. After the freeze condition is removed, the QADC64E continues queue execution with the next CCW in sequence.

Trigger events that occur during freeze are not captured. 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 13-24** to **Figure 13-31**) show examples of all of the freeze situations.

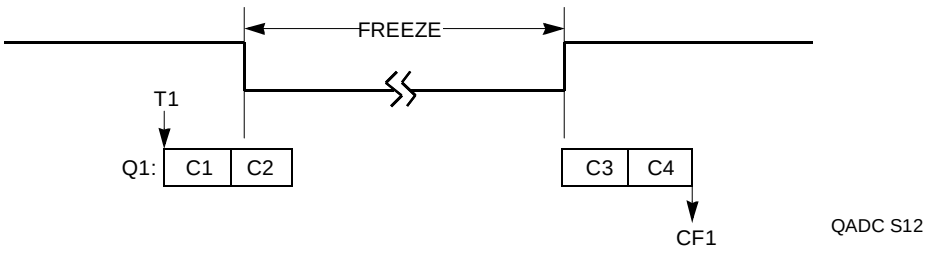


Figure 13-24 CCW Freeze Situation 12

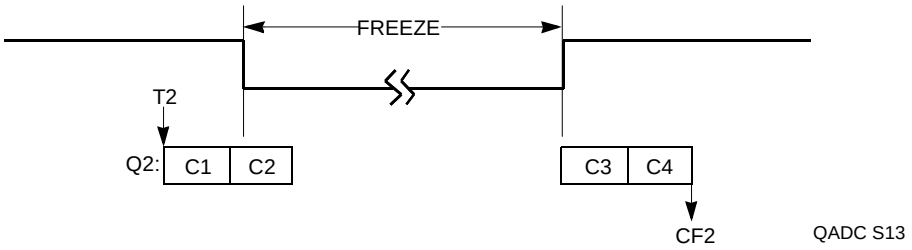


Figure 13-25 CCW Freeze Situation 13

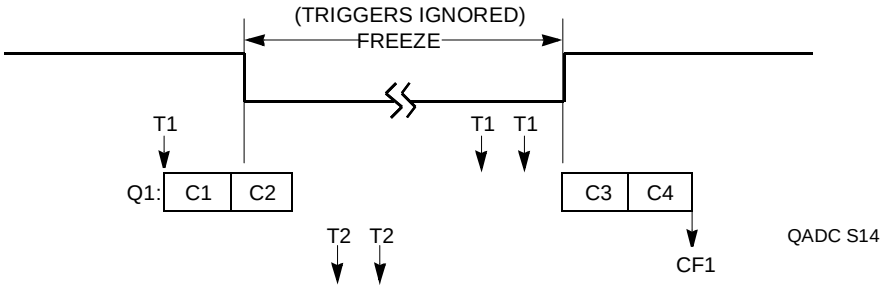


Figure 13-26 CCW Freeze Situation 14

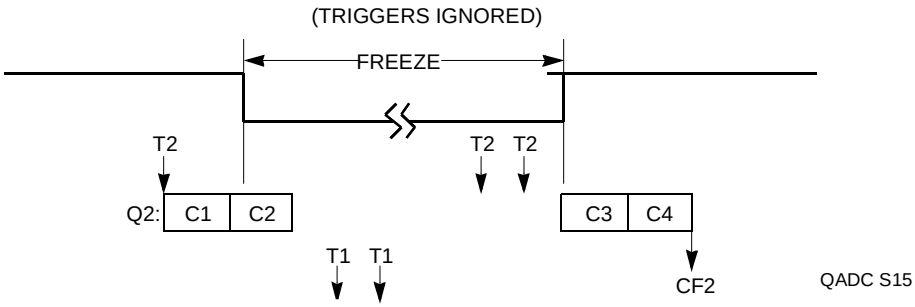


Figure 13-27 CCW Freeze Situation 15

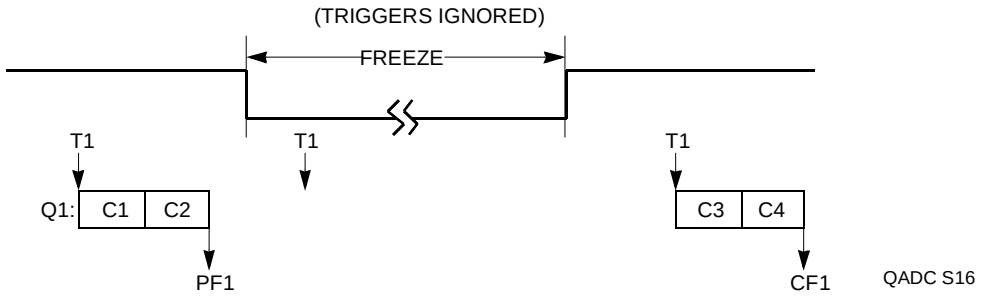


Figure 13-28 CCW Freeze Situation 16

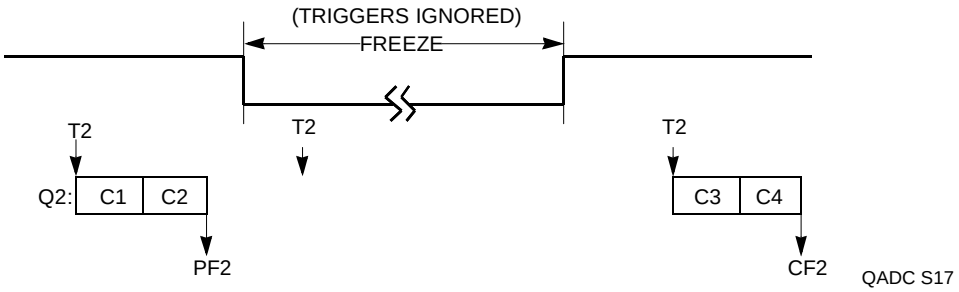


Figure 13-29 CCW Freeze Situation 17

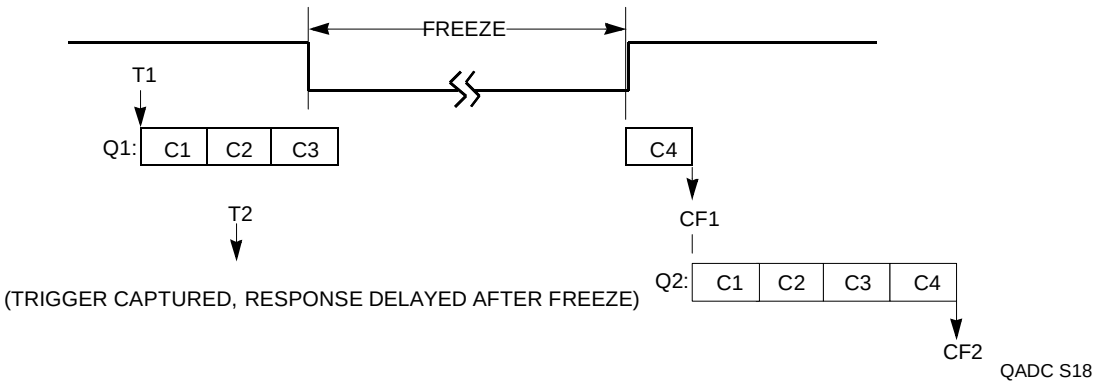


Figure 13-30 CCW Freeze Situation 18

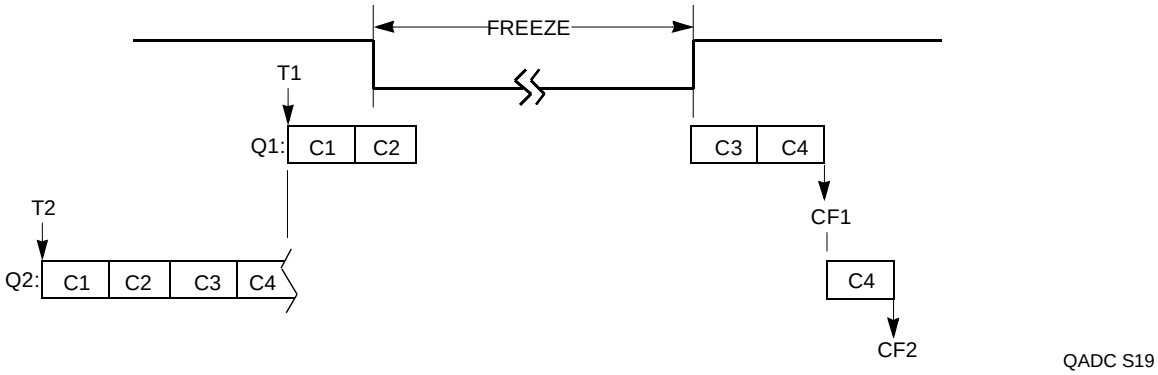


Figure 13-31 CCW Freeze Situation 19

13.5.2 Conversion Timing Schemes

This section contains some conversion timing examples. Example 1 below shows the timing for basic conversions where the following is assumed:

- Q1 begins with CCW0 and ends with CCW3
- CCW0 has pause bit set
- CCW1 does not have pause bit set
- External trigger rise-edge for Q1
- CCW4 = BQ2 and Q2 is disabled
- Q1 RES shows relative result register updates

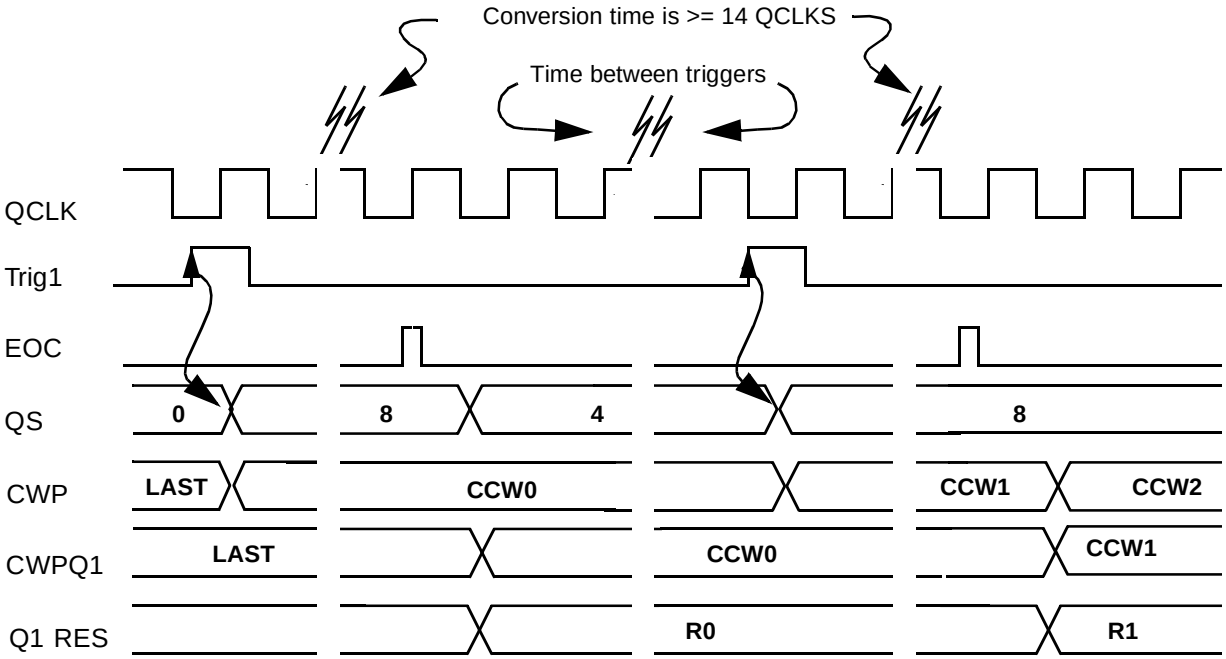


Figure 13-32 External Trigger Mode (Positive Edge) Timing With Pause

Recall QS = 0 => Queues disabled; QS = 8 => Q1 active, Q2 disabled; QS= 4 => Q1 paused, Q2 disabled.

A time separator was provided between the triggers and end of conversion (EOC). The relationship to QCLK displayed is not guaranteed.

CWPQ1 or CWPQ2 typically lag CWP and only match CWP when the associated queue is inactive. Another way to view CWPQ1 or CWPQ2 is that these registers update when EOC triggers the result register to be written.

When the pause bit is set (CCW0), please note that CWP does not increment until triggered. When the pause is not set (CCW1), the CWP increments with EOC.

The conversion results Q1 RES(x) show the result associated with CCW(x). So that R0 represents the result associated with CCW0.

Example 2 below shows the timing for conversions in gated mode single-scan with the same assumptions as example 1 except:

- No pause bits set in any CCW
- External trigger gated single-scan mode for Q1
- Single-scan bit is set

When the gate closes and opens again the conversions start with the first CCW in Q1.

When the gate closes the active conversion completes before the queue goes idle.

When Q1 completes both the CF1 bit sets and the SSE bit clears.

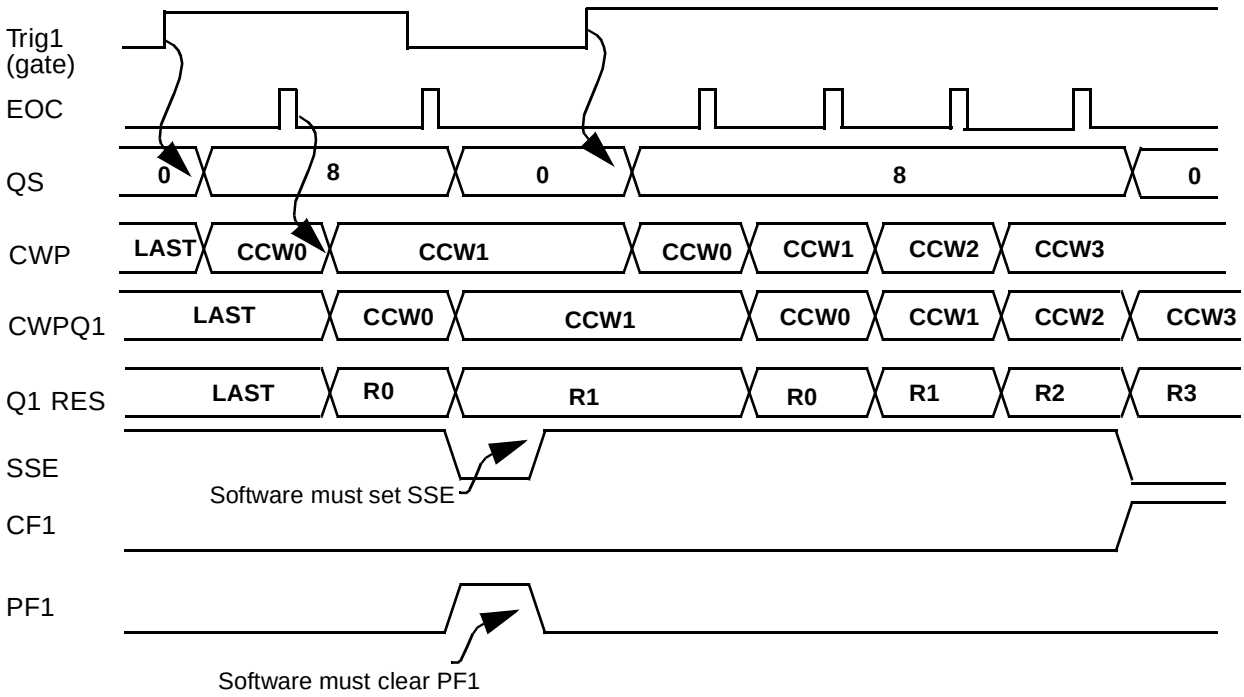


Figure 13-33 Gated Mode, Single-Scan Timing

Example 3 below shows the timing for conversions in gated continuous-scan mode with the same assumptions in the amended definition for the PF bit in this mode to reflect the condition that a gate closing occurred before the queue completed is a proposal under consideration at this time as example 2.

NOTE

At the end of Q1,the completion flag CF1 sets and the queue restarts. Also, note that if the queue starts a second time and completes, the trigger overrun flag TOR1 sets.

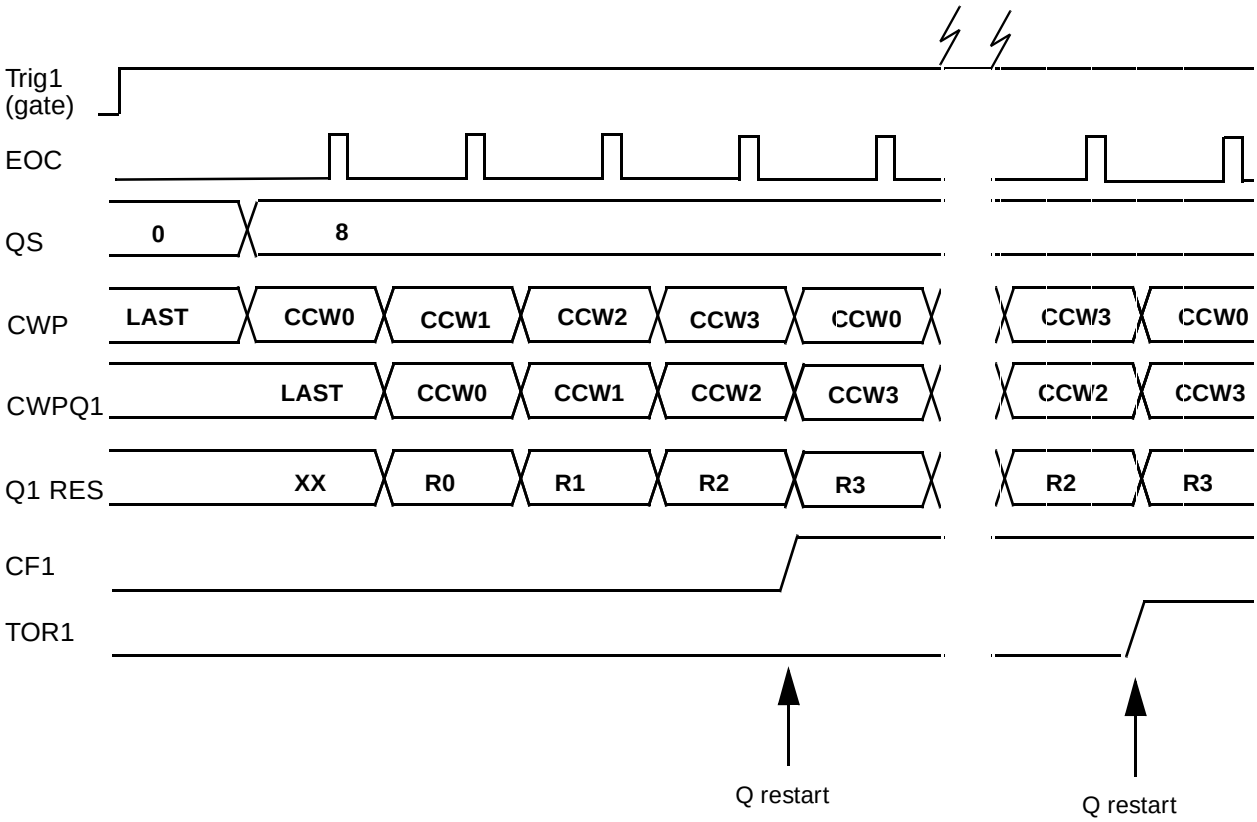


Figure 13-34 Gated Mode, Continuous Scan Timing

13.6 QADC64E Integration Requirements

The QADC64E requires accurate, noise-free input signals for proper operation. This section discusses the design of external circuitry to maximize QADC64E performance.

The QADC64E uses the external pins shown in **Figure 13-1.** There are 32 channel/ port pins and a total of 40 analog input channels. With external multiplexing MPC565 / MPC566 can support 65 analog inputs. 32 of the channel pins can also be used as general-purpose digital port pins. In addition, there are also two analog reference pins, and two analog submodule power shared by both QADC modules.

13.6.1 Port Digital Input/Output Pins

The sixteen port pins can be used as analog inputs, or as a bidirectional 16-bit digital input/output port.

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 push-pull digital output signals. Port B pins are referred to as PQB[7:0] and operate the same as Port A

Port A and B 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 E ELECTRICAL CHARACTERISTICS** for more information on voltage requirements.

Each port A or B pin is configured as an input or output by programming the port data direction register (DDRQA or DDRQB). 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 or B pins when the corresponding bit in the port data direction register specifies output. Refer to **APPENDIX A INTERNAL MEMORY MAP** for more information. Since the outputs are configured as push-pull drivers, external pull-up provisions are not necessary when the output is used to drive another integrated circuit.

13.6.2 External Trigger Input Pins

The QADC64E uses two external trigger pins (ETRIG[2:1]). Each of the two input external trigger pins is associated with one of the scan queues, queue 1 or queue 2 The assignment of ETRIG[2:1] to a queue is made in the QACR0 register by the TRG bit. When TRG=0, ETRIG[1] triggers queue 1 and ETRIG[2] triggers queue 2. When TRG=1, ETRIG[1] triggers queue 2 and ETRIG[2] triggers queue 1.

13.6.3 Analog Power Pins

V_{DDA} and V_{SSA} pins supply power to the analog subsystems of the QADC64E 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 E ELECTRICAL CHARACTERISTICS** for more information.

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 13-35** is a diagram of the analog input circuitry.

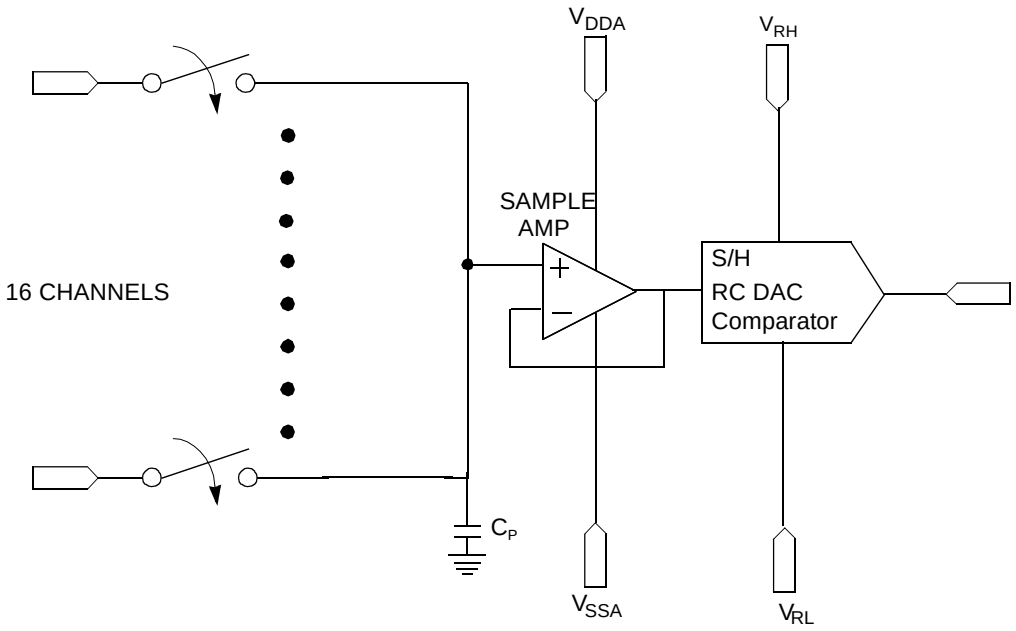


Figure 13-35 Equivalent Analog Input Circuitry

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 13-36 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 0x3FE. 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.

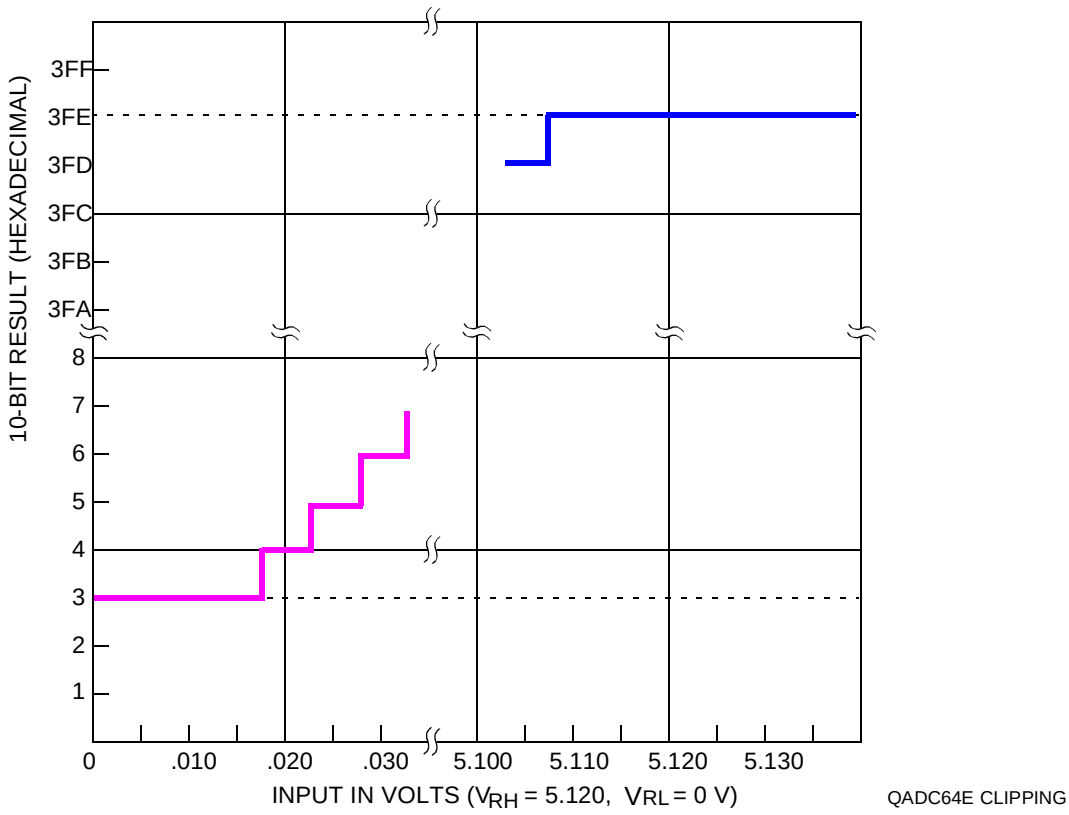


Figure 13-36 Errors Resulting from Clipping

13.6.3.1 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 VDD/VSS pin pair. This applies to analog sub-modules also. The distribution of power and ground is equally important.

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.

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 stand-alone 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 13-37**.

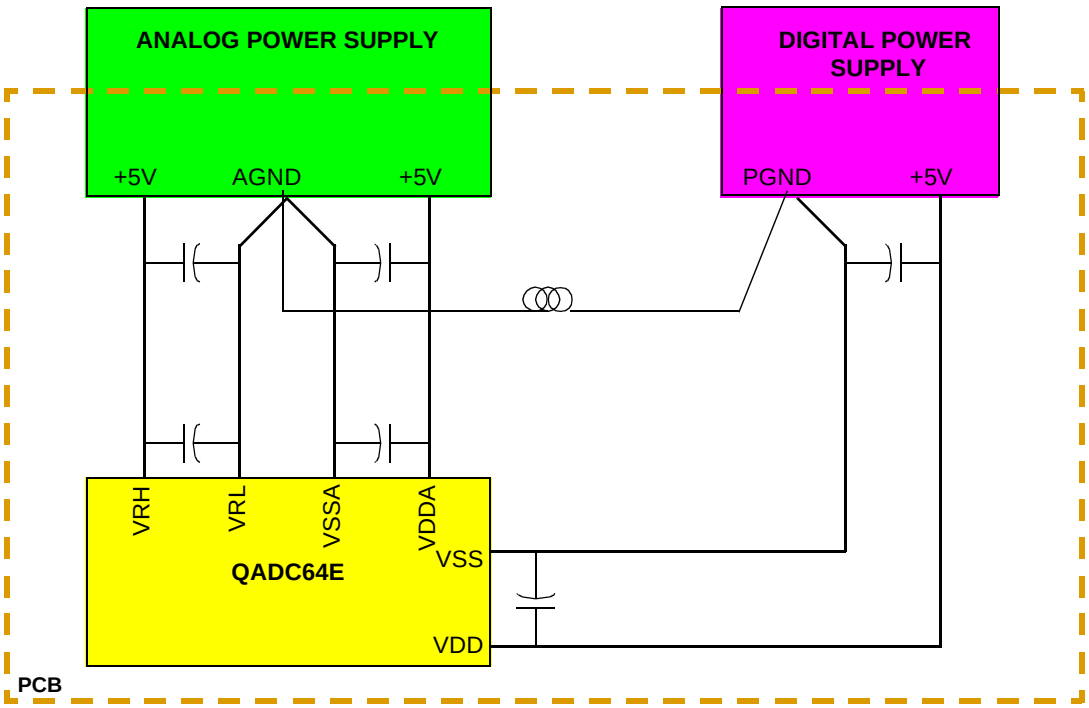


Figure 13-37 Star-Ground at the Point of Power Supply Origin

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 QADC64E 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.
- Distance for trace runs should be minimized where possible.

13.6.4 Analog 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.

The AltRef pin may be selected through the CCW as the high reference for a conversion. This allows for the ability to “zoom” in on a portion of the convertible range with the full 10 bits. Refer to **Table 13-18**.

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 QADC64E, supplied by pins V_{RH}, AltRef, 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 sufficed. 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.

13.6.5 Analog Input Pins

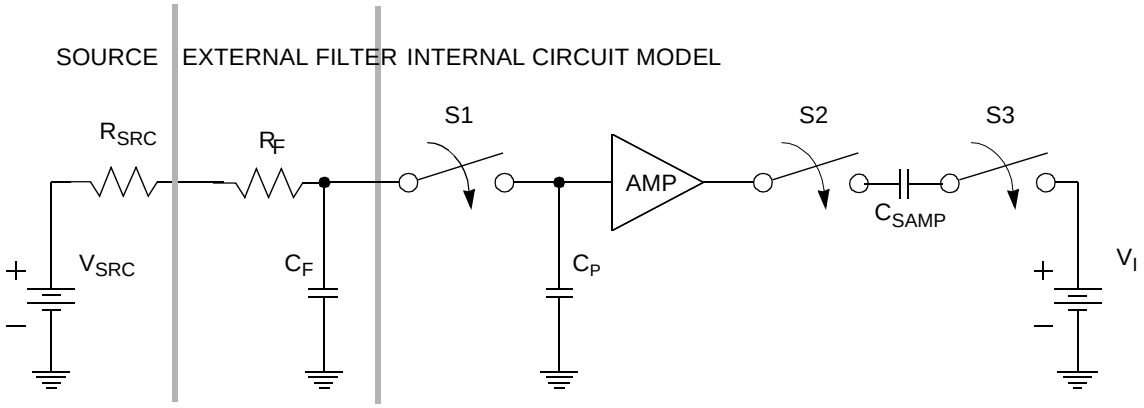
Analog inputs should have low AC impedance at the pins. Low AC impedance can be realized by placing a capacitor with good high frequency characteristics at the input pin of the part. Ideally, that capacitor should be as large as possible (within the practical range of capacitors that still have good 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 **13.6.5.3 Error Resulting from Leakage** for more information. In some cases, the size of the capacitor at the pin may be very small.

Figure 13-38 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 QADC64E.



V_{SRC}= SOURCE VOLTAGE
R_{SRC} = SOURCE IMPEDANCE
R_F = FILTER IMPEDANCE
C_F = FILTER CAPACITOR
C_P = INTERNAL PARASITIC CAPACITANCE
C_{SAMP}= SAMPLE CAPACITOR
V_I = INTERNAL VOLTAGE SOURCE DURING SAMPLE AND HOLD

QADC64E SAMPLE AMP MODEL

Figure 13-38 Electrical Model of an A/D Input Pin

In **Figure 13-38**, R_F, R_{SRC} and C_F comprise the external filter circuit. C_P is the internal parasitic capacitor. C_{SAMP} is the capacitor array used to sample and hold the input voltage. V_I is an internal voltage source used to provide charge to C_{SAMP} during sample phase.

The following paragraphs provide a simplified description of the interaction between the QADC64E and the external circuitry. This circuitry is assumed to be a simple RC low-pass filter passing a signal from a source to the QADC64E input pin. The following simplifying assumptions are made:

- 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 0 Ω and the “off” resistance is infinite

13.6.5.1 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 13-39** shows the connection of eight typical analog signal sources to one QADC64E analog input pin through a separate multiplexer chip. Also, an example of an analog signal source connected directly to a QADC64E analog input channel is displayed.

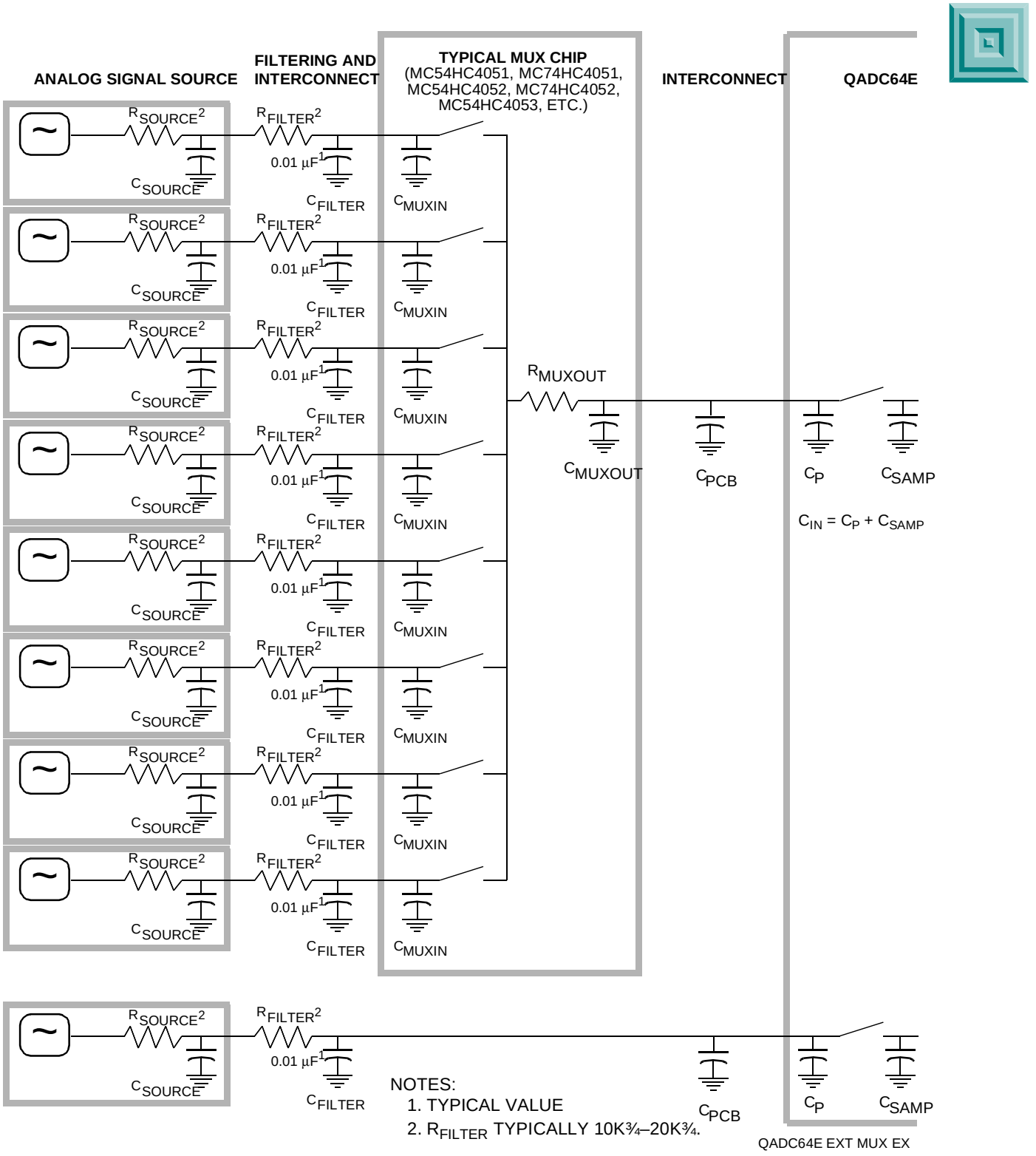


Figure 13-39 External Multiplexing of Analog Signal Sources

13.6.5.2 Settling Time for the External Circuit

The values for R_{SRC}, R_F and C_F in the external circuitry determine the length of time required to charge C_F to the source voltage level (V_{SRC}). At time t = 0, V_{SRC} changes in **Figure 13-38** while S1 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:

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

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 13-24** 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 13-24** is based on the RC network in **Figure 13-38**.

NOTE

The following times are completely independent of the A/D converter architecture (assuming the QADC64E is not affecting the charging).

Table 13-24 External Circuit Settling Time to 1/2 LSB (10-Bit Conversions)

Filter Capacitor (CF)	Source Resistance ($R_F + R_{SRC}$)			
	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 13-24** 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.

13.6.5.3 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 E ELECTRICAL CHARACTERISTICS**. Input leakage is greater at higher operating temperatures. In the temperature range from 125° C to 50° C, the leakage current is halved for every 8 – 12° C reduction in temperature.

Assuming $V_{RH} - V_{RL}$ = 5.12 V, one count (assuming 10-bit resolution) corresponds to 5 mV of input voltage. A typical input leakage of 200 nA acting through 10 kΩ of external series resistance results in an error of 0.4 count (2.0 mV). If the source impedance is 100 kΩ and a typical leakage of 100 nA is present, an error of two counts (10 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 13-25** 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 below 200 nA is obtainable only within a limited temperature range.

Table 13-25 Error Resulting From Input Leakage (IOFF)

Source Impedance	Leakage Value (10-bit Conversions)			
	100 nA	200 nA	500 nA	1000 nA
1 kΩ	—	—	0.1 counts	0.2 counts
10 kΩ	0.2 counts	0.4 counts	1 counts	2 counts
100 kΩ	2 counts	4 count	10 counts	20 counts

13.6.5.4 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. QADC64E 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 **APPENDIX E ELECTRICAL CHARACTERISTICS** to for more information on exact magnitudes.

Either stress condition 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 injected on an adjacent pin can cause errors on the selected channel by developing a voltage drop across the selected channel's impedances.

Figure 13-40 shows an active parasitic bipolar NPN transistor when an input pin is subjected to negative stress conditions. **Figure 13-41** shows positive stress conditions can activate a similar PNP transistor.

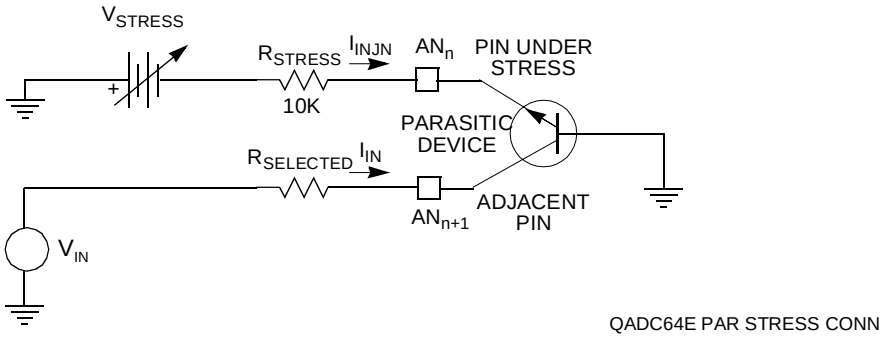


Figure 13-40 Input PIn Subjected to Negative Stress

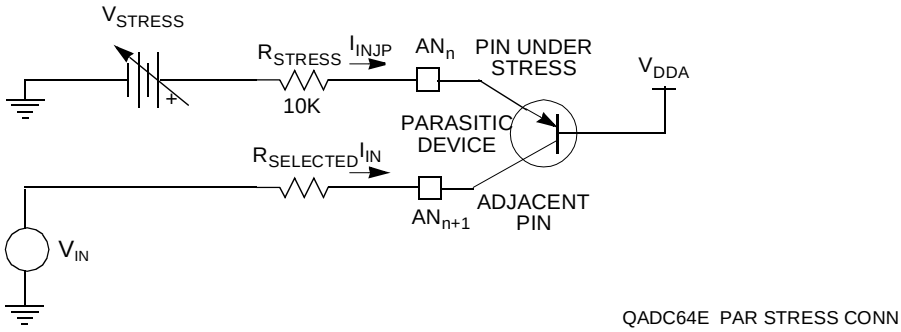


Figure 13-41 Input Pin Subjected to Positive Stress

The current into the pin (I_{INJN} or I_{INJP}) under negative or positive stress is determined by the following equations:

$$I_{INJN} = \frac{-(V_{STRESS} - V_{BE})}{R_{STRESS}}$$

$$I_{INJP} = \frac{V_{STRESS} - V_{EB} - V_{DDA}}{R_{STRESS}}$$

where:

V_{STRESS} = Adjustable voltage source

V_{EB} = Parasitic PNP emitter/base voltage (refer to $V_{NEGCLAMP}$ in **APPENDIX E ELECTRICAL CHARACTERISTICS**)

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

R_{STRESS} = Source impedance
(10-kΩ resistor in **Figure 13-40** and **Figure 13-41** on stressed channel)

$R_{SELECTED}$ = Source impedance on channel selected for conversion

The current into (I_{IN}) the neighboring pin is determined by the K_N (current coupling ratio) of the parasitic bipolar transistor ($K_N << 1$). The I_{IN} can be expressed by the following equation:

$$I_{IN} = - K_N * I_{INJ}$$

where I_{INJ} is either I_{INJN} or I_{INJP}.

A method for minimizing the impact of stress conditions on the QADC64E is to strategically allocate QADC64E inputs so that the lower accuracy inputs are adjacent to the inputs most likely to see stress conditions.

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