AN1062

Using the QSPI for Analog Data Acquisition

By: Craig Shaw

INTRODUCTION

To effectively use digital microcontroller units (MCUs) in an analog world, analog information must be converted into digital form. In all applications, fast, accurate, and inexpensive conversion is desirable. Minimizing printed circuit board space and interconnections is also desirable.

NOTE

> This application note can be applied to any MCU (i.e., MC68332, MC68HC16Z1, etc.) containing queued serial peripheral interface (QSPI) circuitry.

The MC68332 lacks any direct analog-to-digital (A/D) conversion capabilities. This deficiency is easily and inexpensively remedied by connecting the QSPI to an external serial A/D converter.

This application note presents hardware and software examples detailing use of the QSPI with multichannel 8- and 10-bit A/D converters, specifically the MC145040 and the MC145050. It describes design methodology for obtaining maximum A/D throughput, using one or more A/D converters. It also discusses how to simultaneously use other peripherals with the QSPI and how to determine overall system performance.

OPERATION OF THE MC145040, MC145050 FAMILY A/D CONVERTERS

The following paragraphs give a brief overview of the Motorola serial A/D converters. For a more thorough treatment of the subject, refer to References 3 and 4.

The MC145040, MC145041, MC145050, and MC145051 are low-cost, ratiometric, 11-channel A/D converters. They are designed for connection to a microcomputer system with channel selection and conversion results being conveyed through a serial interface port. They require only 14 mW from a single 5-V power supply and yield ±1 LSB accuracy over the – 40 to +125°C range. The reference voltage can be anywhere from +2.5 V to V_{DD}, and the analog input voltage may range from V_{SS} to V_{DD}.

The MC145050 and MC145051 are 10-bit converters; whereas, the MC145040 and MC145041 are 8-bit converters. The MC145040 and MC145050 use external clock sources to perform the conversion; the MC145041 and MC145051 use internal RC oscillators. The parts using external oscillators guarantee faster conversion rates because internal oscillator frequency must be limited to guarantee reasonable yield despite manufacturing tolerances. The remaining A/D converter description refers specifically to the MC145050 since it is the converter used in the examples presented.

Figure 1 shows the pinout of the MC145050. It has 13 analog pins, consisting of 11 analog inputs, labeled AN0–AN11, and two voltage reference inputs, labeled V_{AG} (analog ground) and V_{REF} (positive reference voltage). Power is supplied through the V_{SS} and V_{DD} pins and is a nominal 5 V. The MC145050 requires an external clock to be supplied on the A/D CLK pin to regulate the data conversion.

Channel selection and conversion results are transferred through the digital serial communication pins. A serial transfer synchronizing clock must be fed into the SCLK input pin when the chip-select ($\overline{CS}$) pin is driven low. The address to be converted is serially transmitted into the DIN pin, and the conversion results are serially shifted out the DOUT pin.

The MC145050 is designed to be used in conjunction with multiple serial devices on a common bus; consequently, the DOUT pin is driven only when $\overline{CS}$ is asserted. The serial protocol employed is Motorola SPI, which is compatible with the National Semiconductor Microwire™ system and the Texas Instrument TMS370 series SPI units. The Motorola queued serial module (QSM) also contains a QSPI that efficiently implements this protocol.

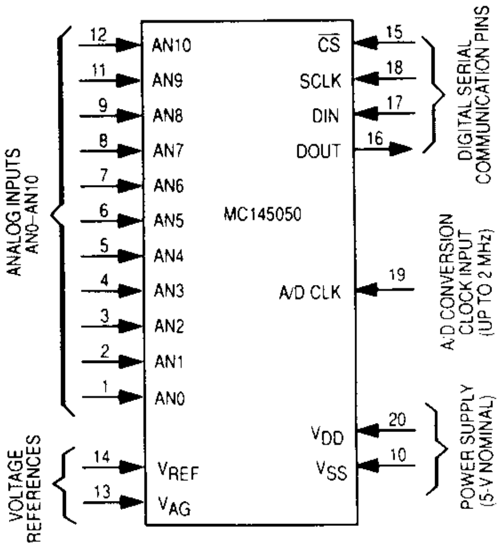


Figure 1. MC145050 Pinout

FUNDAMENTALS OF QSPI OPERATION

The following paragraphs give a brief overview of the QSPI as it applies to the examples that are presented. A more detailed description of the QSPI is contained in section 5 of the *MC68332 User's Manual* (see Reference 2).

The QSPI is an intelligent, synchronous serial interface with a 16-entry, full-duplex queue. It can continuously scan up to 16 independent peripherals and maintain a queue of the most recently acquired information with no central processor unit (CPU) intervention. It features variable word lengths, programmable chip selects, and selectable data/clock phase relationship. The baud rate and the delay between transfers are also programmable. The QSPI has a maximum transfer speed of one-fourth the MC68332 system clock speed.

Since the QSPI is capable of operation as a master or as a slave, all pins are bidirectional. Figure 2 shows a typical master mode configuration. The slave peripherals are selected via the peripheral chip-select pins, PCS0–PCS3, and the serial clock is provided by the SCK pin. QSPI output data is presented on the master out slave in (MOSI) pin, and input is taken from the master in slave out (MISO) pin.

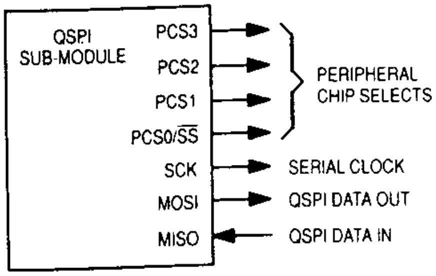


Figure 2. Master Mode Representation of the QSPI

One of the most powerful elements of the QSPI is its queue. Figure 3 depicts the structure of the QSPI queue RAM. The queue may contain up to 16 entries, each consisting of a transmit word, a receive word, and a command control byte. The transmit and receive words are from 8 to 16 bits long and are LSB justified. For any given queue entry, the transmit and receive words are the same length.

An important subset of the queue RAM is the command control RAM. Figure 4 shows a breakdown of a single command control byte, and Figure 5 depicts a basic QSPI master mode timing diagram. The control byte allows the programmer to customize each serial transfer to the specific needs of the targeted peripheral. Chip-select patterns are stored in the PCS0–PCS3 bit fields of each applicable control byte and are driven onto the chip-select pins when the specified transfer begins. If set, the continue (CONT) bit allows the QSPI to continue driving the programmed chip-select value until the beginning of the next transfer. This procedure has the effect of concatenating multiple serial transfers to a single peripheral and allowing more than 16 bits per exchange. If the CONT bit is clear, a user-defined default value is driven onto the chip-select pins between serial transfers.

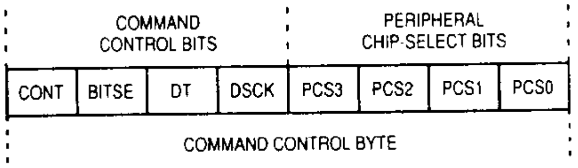


Figure 4. Command Control Byte

The PCS to SCK delay (DSCK) and delay after transfer (DT) bits enable user-defined delays before and after the specified transfer. If DSCK is set, the first clock following the chip-select assertion is delayed by a user-specified amount of time. Otherwise, the first clock pulse is delayed one-half of an SCK period. This delay is necessary because some peripherals require a relatively long period of time to respond.

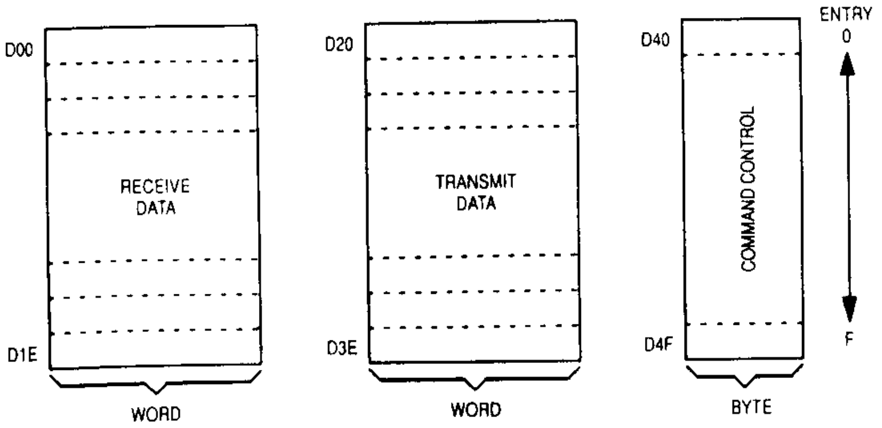
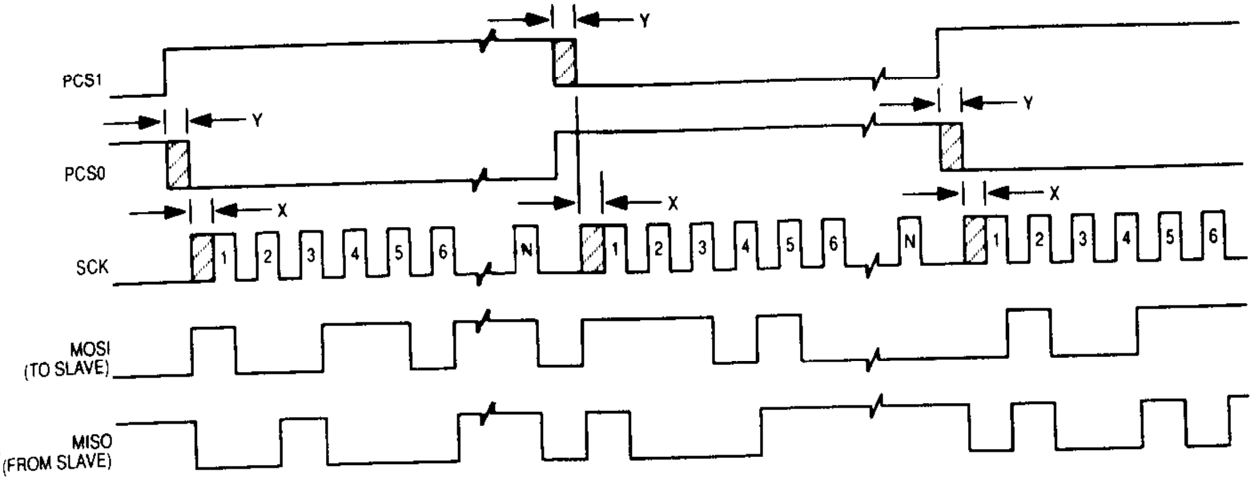


Figure 3. Organization of the QSPI Queue RAM



PROGRAMMABLE FEATURES:
N = NUMBER OF BITS
X = DELAY BEFORE FIRST CLOCK
Y = DELAY BETWEEN TRANSFERS
CLOCK RATE, POLARITY
DATA PHASE SHIFT
CHIP-SELECT PATTERN

Figure 5. Basic QSPI Master Mode Timing Diagram

If DT is set, a user-specified delay elapses before the next serial transfer is begun. Otherwise, the QSPI executes the next transfer as soon as possible (approximately 1 μs when the MC68332 operates at 16.778 MHz). This delay is useful if a peripheral needs time to perform a function that affects subsequent serial transfers. One example might be to wait for an A/D converter to perform a conversion.

The remaining element in the control byte is the bits per transfer enable (BITSE) bit. If BITSE is set, the transfer length is a user-specified value, ranging from 8 to 16 bits. If BITSE is cleared, the transfer length will default to 8 bits.

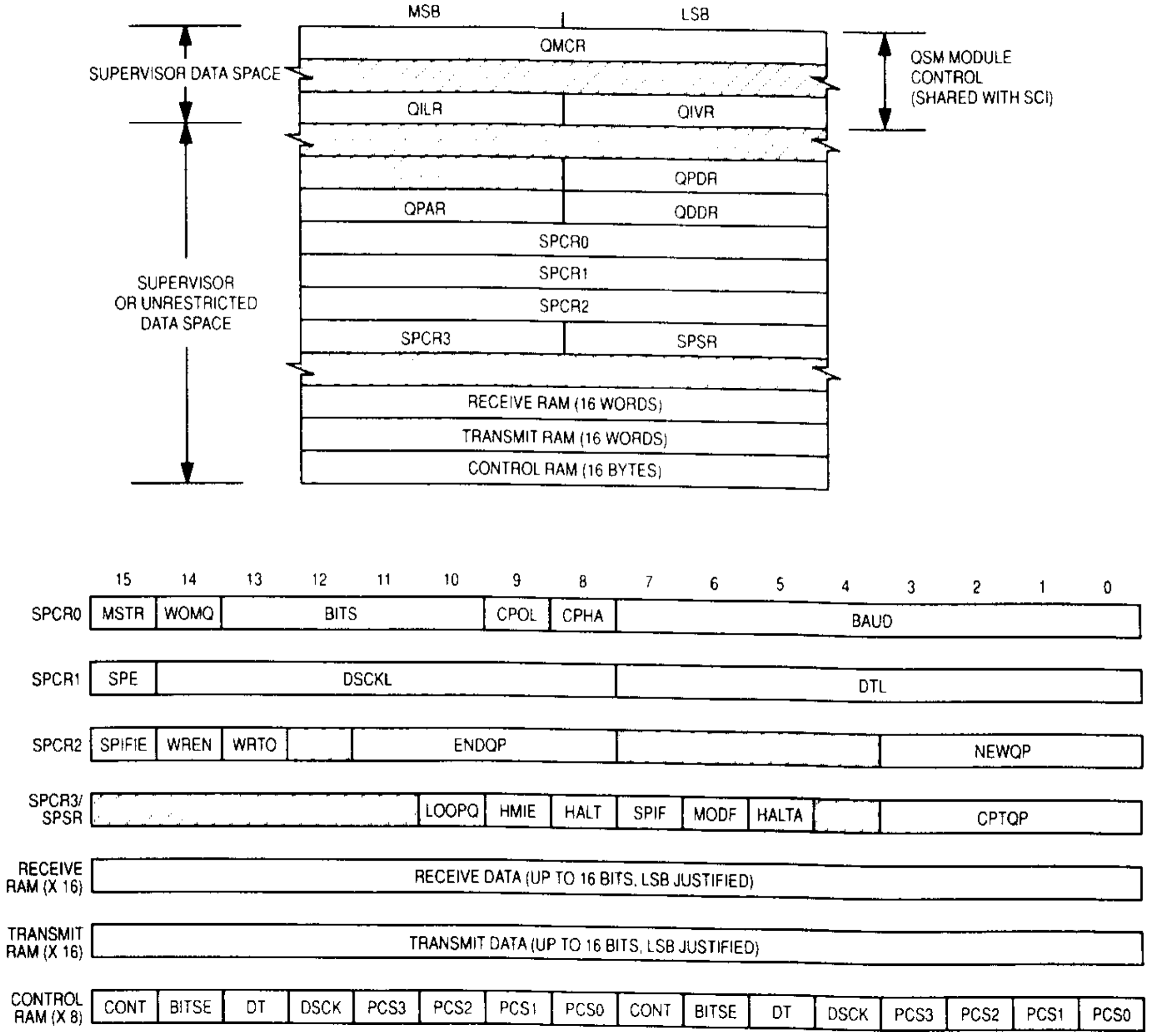
Figure 6 represents a programmer's model of the QSPI. The QSM data direction register (QDDR) determines whether a given QSPI pin is an input or an output. When read, the QSM port data register (QPDR) provides the logic level present on a QSM input pin or the data latched in an output pin. When written, the write data is latched into the output register. The QSM pin assignment register (QPAR) controls whether a pin is to be controlled by the QSPI or is to function as a general-purpose I/O pin.

Serial peripheral control register 0 (SPCR0) specifies six different functions. The master/slave mode select (MSTR) bit, if set, causes the QSPI to operate as the controller of the SPI transfer. The wired-OR mode for QSPI pins (WOMQ) bit, if set, causes all QSPI outputs to function in an open-drain mode, requiring external pullup resistors. The bits per transfer (BITS) field allows the programmer to specify the number of bits in a nondefault transfer (used if BITSE is set). The clock polarity (CPOL) bit determines the polarity of the SCK output, and the clock phase (CPHA) bit dictates the data's phase relationship to the SCK. The serial clock baud rate (BAUD) field determines the QSPI SCK frequency, from 33 kHz to 4.2 MHz (with the MC68332 system clock frequency at 16.778 MHz).

Serial peripheral control register 1 (SPCR1) specifies three different functions. Setting the QSPI enable (SPE) bit causes the QSPI to begin operation; clearing SPE causes operation to stop immediately. SPE is automatically cleared by the QSPI when it completes all specified transfers. The DSCKL field allows the programmer to set the nondefault delay before SCK (used if DSCK is set). The DTL field controls the nondefault delay after the transfer is completed (used if DT is set).

Serial peripheral control register 2 (SPCR2) specifies five queue control functions. The new queue pointer value (NEWQP) field determines which queue entry is to be transferred first. More queue entries are sequentially transferred until the entry specified by the ending queue pointer (ENDQP) field is completed. If the wrap enable (WREN) bit is set, transfers continue either at queue entry 0 or at the entry specified by the NEWQP field. The point the queue wraps to (entry 0 or NEWQP) is determined by the wrap to (WRTO) bit. The SPI finished interrupt enable (SPIFIE) bit is an interrupt enable. If set, an interrupt will be generated upon completion of the queue entry specified by the ENDQP field.

Serial peripheral control register 3 (SPCR3) controls self-test and program debug functions, which will not be discussed in this application note. The serial peripheral status register (SPSR) contains two status fields of importance for this application. The completed queue pointer (CPTQP) field contains the queue entry number that was most recently completed. The QSPI finished flag (SPIF) bit is set when the CPTQP matches the ENDQP, which indicates that the specified queue has been completed and the QSPI has either shut down or wrapped to the designated point.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPCR0	MSTR	WOMQ	BITS				CPOL	CPHA	BAUD							
SPCR1	SPE	DSCKL							DTL							
SPCR2	SPIFIE	WREN	WRTO		ENDQP								NEWQP			
SPCR3/ SPSR						LOOPQ	HMIE	HALT	SPIF	MODF	HALTA		CPTQP			
RECEIVE RAM (X 16)	RECEIVE DATA (UP TO 16 BITS, LSB JUSTIFIED)															
TRANSMIT RAM (X 16)	TRANSMIT DATA (UP TO 16 BITS, LSB JUSTIFIED)															
CONTROL RAM (X 8)	CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0	CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0

NOTE: Shading denotes not used area.

Figure 6. QSPI Programmer's Model

BASIC SYSTEM IMPLEMENTATION

The schematic diagram shown in Figure 7 depicts the basic minimal serial A/D data acquisition system. The only extraneous logic required for this system is the 2-MHz oscillator. The oscillator can be used to supply a number of other peripheral devices as well as additional A/D converters. Also, the oscillator can be eliminated entirely, and an MC145051 can be used in place of the MC145050; however, the speed of the conversions would be reduced.

The timing diagram (see Figure 8) shows significant events on the pins of the MC145050. This timing sequence corresponds to the timing sequence illustrated in Figure 9 of Reference 4. Although not the fastest method for sampling the A/D converter, this timing sequence allows efficient use of the MC145050 on a bus in conjunction with other peripherals. During A/D conversion, the QSPI can select and exchange data with another device, maximizing overall serial bandwidth. The timing for 10-clock transfer not using $\overline{CS}$ may be slightly faster, but if it is used with other peripherals, the QSPI must wait for the conversion to be completed.

For successful operation, power supply decoupling and wiring should be carefully considered. The 0.1-µF decoupling capacitor should be placed as close as possible to the V_{DD}

and V_{SS} pins. A nearby decoupling capacitor is also needed between the V_{REF} and V_{AG} pins. Separate lines should be run to the V_{REF} and V_{AG} inputs since any current drain will cause IR voltage drop in the traces. If an active IC is being powered by the same trace, the switching current transients can cause enormous errors.

As the timing diagram shows, the MC145050 requires valid data on the DIN pin during the rising edge of SCK. The data is allowed to change on the falling edge of SCK. This determines the clock polarity and phase values that need to be programmed into the QSPI (CPOL = 0, CPHA = 0).

TIMING CONSIDERATIONS

One factor determining overall system speed is the source impedance of the signal being measured. The impedance limits the maximum SCK clock frequency because the SCK frequency is what determines the actual sample interval. For more information on source impedance effect on clock frequency, refer to Reference 4. A source impedance of less than 1000 ohms is assumed so that sample interval is not a constraint.

Calculate the maximum SCK frequency according to the following procedures. According to Reference 4, the minimum SCLK pulse high and low widths (t_{wh}, t_{wl}) are both 190 ns, the maximum propagation delay from SCK to DOUT (t_{PHL}, t_{PLH}) is 240 ns, and the minimum setup time from DIN to SCK (t_{su}.A/D) is 100 ns.

Assuming a QSPI minimum data setup time (t_{su}.Q, MISO to SCK) of 10 ns, to meet QSPI input data timing requirements, the minimum clock pulse width is the greater of (t_{PLH} + t_{su}.Q) or (t_{PHL} + t_{su}.Q). This yields 250 ns.

Assuming a QSPI maximum data delay time (t_{dd}.Q, SCK to MOSI) of 10 ns, to meet MC145050 input data timing requirements, the minimum clock pulse width is the greater of t_{wh}, t_{wl}, or (t_{dd}.Q + t_{su}.A/D). This figure is 190 ns.

Data hold times on both the QSPI and the MC145050 are too minimal to present a problem, since data is not allowed to change until one-half SCK period after the latch is triggered. The minimum SCK period must be twice the largest minimum clock pulse width since the QSPI generates a symmetrical SCK waveform. This number is 500 ns, indicating a maximum SCK frequency of 2 MHz. The MC68332 will be clocked at a system clock frequency of 16 MHz, allowing an SCK frequency of exactly 2 MHz. The BAUD field value can be found from the following equation:

BAUD = system clock frequency/
(2 · desired SCK frequency)

Therefore, the BAUD field should be programmed to

BAUD = [16 MHz/(2 · 2 MHz)] = 4

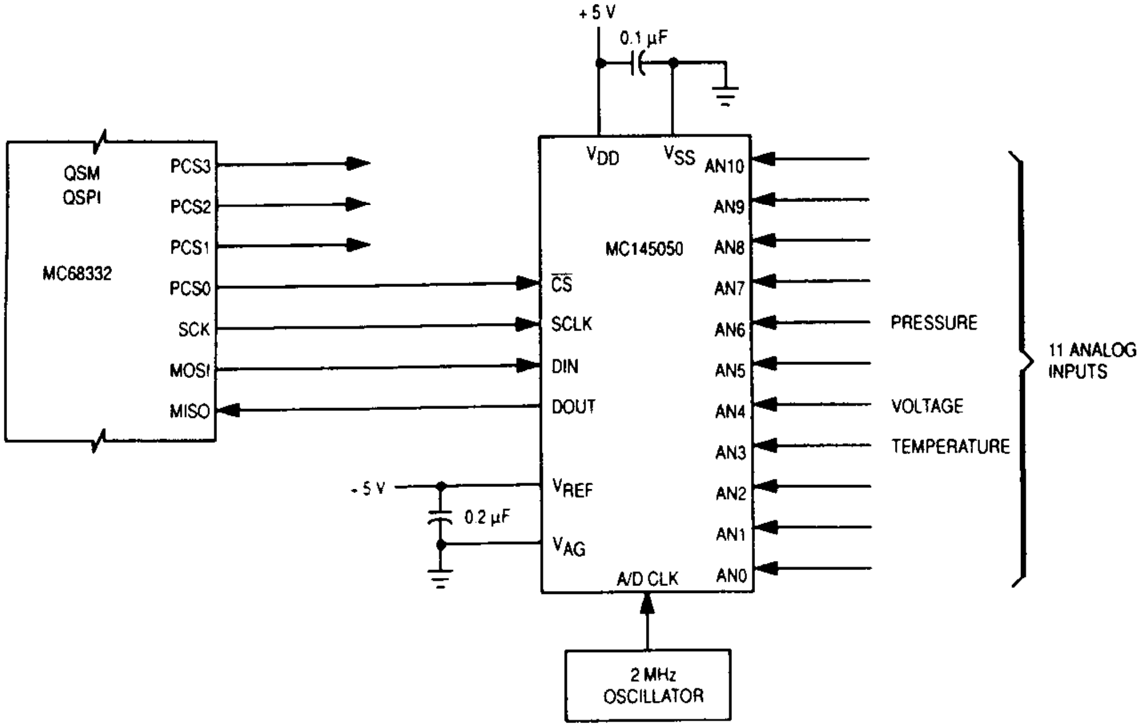


Figure 7. Basic Serial A/D Data Acquisition System

Another parameter that must be determined is the minimum time that must elapse between asserting the MC145050 $\overline{CS}$ pin and providing the first SCK pulse. According to Reference 4, the maximum propagation delay from $\overline{CS}$ to DOUT driven (t_{PZL}, t_{PZH}) is 2 A/D CLKs + 300 ns. Assuming a QSPI input data setup time of 10 ns and an A/D CLK frequency of 2 MHz, the total delay must be at least 10 + 300 + (2 · 500) = 1.31 µs. A minimum setup time from $\overline{CS}$ to SCK (t_{SU}) is 2 A/D CLKs + 425 ns. Since this value is 1.425 µs and is the larger value, the DSCKL field in QSPI SPCR1 must be programmed to provide at least this amount of delay. The *MC68332 User's Manual* (see Reference 2) states the formula for DSCKL as follows:

$$\text{delay time} = \text{DSCKL/system clock frequency}$$

Solving for DSCKL gives

$$\text{DSCKL} = (1425\ \text{ns}/62.5\ \text{ns}) = 22.8$$

Rounding up to the nearest whole delay, there are 23 DSCKL units for a total delay of 1.4375 µs. Also, the DSCK bit must be set in each command control byte that governs a transfer to the MC145050; otherwise, the standard delay of one-half SCK period will be used (in this case, 250 ns).

For a successful conversion to occur, a delay of 44 A/D CLKs must elapse from the last falling edge of SCK to the next assertion of $\overline{CS}$ (see **AC ELECTRICAL CHARACTERISTICS** of Reference 4). The QSPI always provides a one-half SCK delay after the last SCK edge before the $\overline{CS}$ pins change state. The delay time before the next $\overline{CS}$ assertion must then be

$$(44 \cdot 500\ \text{ns}) - 250\ \text{ns} = 21.75\ \mu\text{s}$$

The equation for delay between transfers is

$$\text{delay time} = (32 \cdot \text{DTL})/\text{system clock frequency}$$

thus, it follows that

$$\text{DTL} = (\text{system clock frequency} \cdot \text{delay time})/32$$

therefore,

$$\text{DTL} = ((16 \cdot 10^6)\ \text{hertz} \cdot (21.75 \cdot 10^{-6})\ \text{seconds})/32$$
$$\text{DTL} = 10.88 \text{ which rounds up to } 11$$

Plugging DTL = 11 into the original equation gives an actual delay of 22 µs.

QSPI INITIALIZATION AND OPERATION

Since the fastest throughput is possible when using 10-bit transfers, the BITS field in SPCR0 must be set to 10. Additionally, the BITSE bit must be set in each command control byte associated with a transfer to the MC145050.

To simplify the example, assume conversions are only wanted from A/D channels 3, 4, and 6. Those channels will be sampled repeatedly, and each channel will have a separate fixed memory address where the most recently acquired result will always be available to the CPU. The WREN bit in SPCR2 and the first three queue entries will be used. The transmit RAM must contain the A/D multiplexer address to be converted, and the receive RAM will hold the conversion results.

Figure 9 is an assembly language listing showing how the QSPI is configured to perform the stated functions.The first portion of the program is definitions, followed by initialization. The QSPI is then activated. The program waits until all conversions have been performed once before utilizing the results.

Figure 10 shows the setup and operation of the queue RAM in this example. It is important to note that the conversion data requested by one queue entry is not shifted out until the next transfer; thus, the data is stored in the receive RAM corresponding to the latter transfer. Also, the very first transfer of output data from the A/D converter is invalid and should be ignored. This issue can be handled by simply waiting a known amount of time (until the first result has been updated).

Using a different approach, start the queue from entry F and then transfer and loop on entries 0, 1, and 2. Queue entry F executes once; whereas, entries 0–2 will repeat indefinitely, causing the invalid data word from the A/D converter to be stored in unused RAM (associated with queue entry F). After SPIF in the SPSR is set, all A/D result locations will contain valid data. From then on, the CPU merely reads the latest A/D results from their fixed locations, effectively making the serial A/D converter appear to the CPU as a parallel, memory-mapped peripheral. Having fixed locations for each channel's result allows the programmer to equate them with sensor names, making software easier to write and maintain (especially when compared to serial systems funnelling all results through a single receive register).

The example in Figure 9 shows an interrupt service routine which will generate a warning if fuel pressure drops below a specific level. To cancel the warning, the pressure must increase above a second threshold. Similarly, a heating element is controlled to maintain an operator-specified temperature within a given range. Finally, an unknown voltage is measured, scaled into millivolts, then displayed on an LED readout. Again, note that the CPU just reads the latest conversion results.

The total time to complete the entire queue is calculated as follows:

$$\begin{aligned}
\text{time per entry} &= (\text{no. of bits} \cdot \text{SCK period}) + \text{DSCKL period} + \text{DTL period} \\
&= (10 \cdot 500\ \text{ns}) + 1.4375\ \mu\text{s} + 22\ \mu\text{s} \\
&= 28.4375\ \mu\text{s} \\
\text{time per wrap} &= (\text{no. of entries}) \cdot \text{time per entry} \\
&= 3 \cdot 28.4\ \mu\text{s} = 85.3\ \mu\text{s}
\end{aligned}$$

The age of the oldest result is calculated as follows:

$$\begin{aligned}
\text{maximum age} &= [\text{time per entry} \cdot (\text{no. of entries} + 1)] + \text{sample time} \\
\text{sample time} &= 6 \cdot \text{SCK period} = 6 \cdot 500\ \text{ns} = 3\ \mu\text{s} \\
\text{maximum age} &= [28.4\ \mu\text{s} \cdot (3 + 1)] + 3\ \mu\text{s} = 116.75\ \mu\text{s}
\end{aligned}$$

The maximum-age equation accounts for the fact that the analog level may change while sampling, conversion, and transfer occurs. If the sample time is not considered, the oldest data is simply the sum of the time per wrap and the time per entry because the A/D result data always emerges on the transfer following the transfer requesting the conversion.

OTHER USEFUL CONCEPTS

If the QSPI is to be used to control another peripheral in addition to an A/D converter, it may be advisable to interleave the transfers to the two peripherals. Interleaving can improve the overall serial transfer rate (queue entries per second) by constructively utilizing the time ordinarily wasted waiting for a conversion.

If faster data acquisition is necessary, this concept can also apply to a second A/D converter. The conversion workload must be split between the two A/D converters so that one is

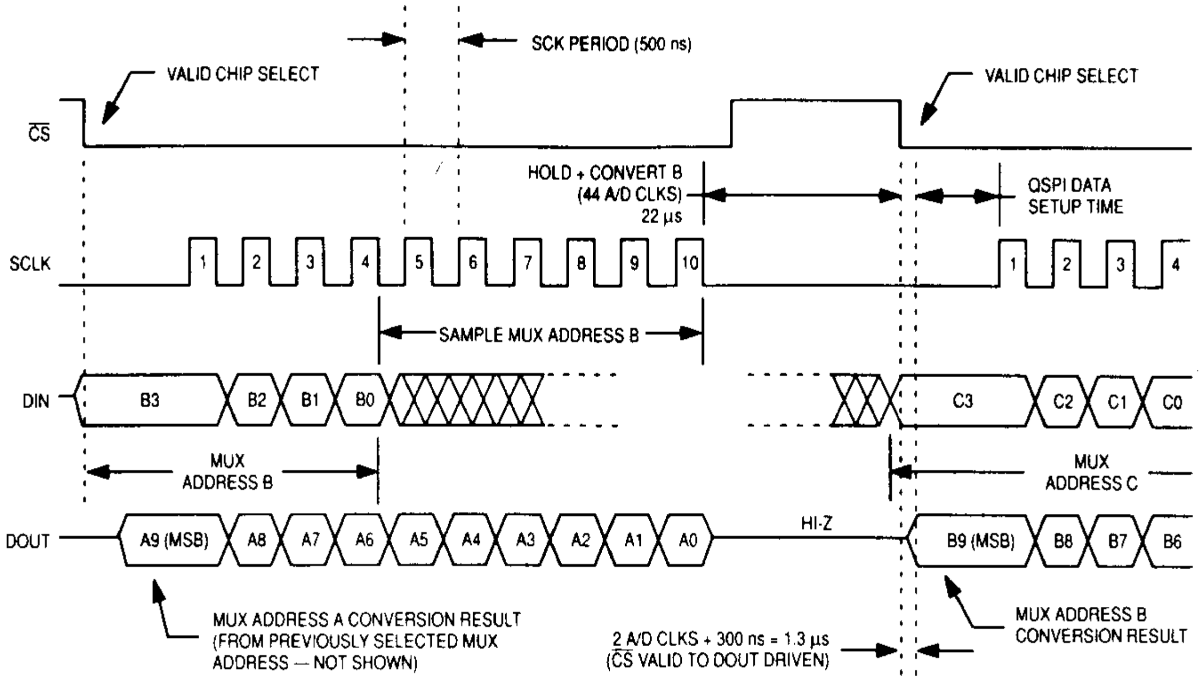


Figure 8. MC14050 Conversion and Transfer Timing

sampling while the other is converting, reducing the average time between conversions from 28.4 µs to 14.2 µs. If three A/D converters are employed, the time drops to 9.5 µs. If a fourth A/D converter is used, the total acquisition time is reduced to the theoretical minimum value, 7.5 µs. The theoretical minimum is the sum of the transfer time (5 µs), the minimum DSCK time (1.4375 µs), and the minimum delay after transfer (1.0625 µs).

Another useful feature of the QSPI is the ability to support subqueues. Subqueues are formed when the normal queue execution sequence is altered to perform a special task. Often, the special task needs attention as soon as possible. Afterward, it is usually desirable to resume execution of the previously defined queue.

An example would be the continuous scanning of three A/D converter channels (as previously described), but upon detection of an interrupt, quickly setting an output port to a given value. After the output data is transferred, the QSPI should continue scanning the three A/D channels. This operation is easy due to the branching capability of the QSPI. While the QSPI is operating, writing to the NEWQP field (lower byte of SPCR2) will cause the QSPI to complete the transfer already in progress, then execute the transfer specified by NEWQP. Normal operation (transferring queue entries in sequence) continues from the point indicated by NEWQP. If a new ENDQP value is also written, its value is used to determine the end of the queue. There is no implicit return mechanism, but if the queue is properly structured, the original operation will resume automatically.

Figure 11 shows the queue structure and operation flow that demonstrates this capability. Assuming the QSPI is already in operation (scanning A/D channels 3, 4, and 6) when the interrupt arrives, the software merely sets up the QSPI RAM associated with the special event, then writes $0E to the lower byte of SPCR2. This procedure causes the QSPI to complete the present transfer, then transfer queue entries E and F. Since ENDQP is still 2, the QSPI will then transfer entries 0, 1, and 2, then wrap back to entry 0. The software never has to modify any control registers or respond to QSPI interrupts because the original queue is resumed automatically. For minimum latency, the program should initialize the control RAM (and the transmit RAM, if possible) for the special operation before the operation is to occur to initiate the subqueue transfer.

REFERENCES

1. Harman, Thomas L. *The Motorola MC68020 and MC68030 Microprocessors: Assembly Language, Interfacing, and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
2. *MC68332 User's Manual* (MC68332 UM/AD). Motorola, Inc., 1990.
3. *8-Bit A/D Converters with Serial Interface* (MC145040/D). Motorola, Inc., 1990.
4. *10-Bit A/D Converters with Serial Interface* (MC145050/D). Motorola, Inc., 1990.

```
                    ********************************************************************************
                    ********************************************************************************
                    *
                    *            Example showing use of QSPI to control 3 A/D conversions
                    *
                    *           All timing numbers assume system clock frequency of 16.000 MHz
                    *
                    ********************************************************************************
                    ****************************** EQUATES ******************************************
                    ********************************************************************************
                    *
                    ******  QSPI bit definitions      (just what's needed for this example)
                    *
00000080            CONT    EQU   $80                 control RAM structure
00000040            BITSE   EQU   $40
00000020            DT      EQU   $20
00000010            DSCK    EQU   $10
00000008            PCS3    EQU   $08
00000004            PCS2    EQU   $04
00000002            PCS1    EQU   $02
00000001            PCS0    EQU   $01
                    *
00000008            REGCS0  EQU   $08                 QPDR, QPAR, QDDR
00000004            SCK     EQU   $04
00000002            MOSI    EQU   $02
00000001            MISO    EQU   $01
                    *
00008000            MSTR    EQU   $8000               SPCR0
00000400            BITS    EQU   $400
                    *
00008000            SPE     EQU   $8000               SPCR1
00000100            DSCKL   EQU   $100
                    *
00004000            WREN    EQU   $4000               SPCR2
00000100            ENDQ    EQU   $100
                    *
00000080            SPIF    EQU   $80                 SPSR
                    *
                    ******  QSPI register addresses
                    *
fffffc14            QPDRW   EQU   $FFFFFC14              QPDR as aligned WORD
fffffc18            SPCR0   EQU   $FFFFFC18              control register 0
fffffc1c            SPCR2   EQU   $FFFFFC1C              control register 2
fffffc1f            SPSR    EQU   $FFFFFC1F              QSPI status register
                    *
                    ******  Control register initialization values
                    *
                    * QPDR, QPAR, QDDR
                    *
00000008            INQPDR  EQU   REGCS0                           PCS0 default value = 1
0000000f            INQPAR  EQU   REGCS0+SCK+MOSI+MISO             pins assigned to QSPI
0000000e            INQDDR  EQU   REGCS0+SCK+MOSI                  QSPI output pins
00080f0e            INQPORT EQU   INQPDR*$100+INQPAR*$100+INQDDR   form into a LONG WORD
                    *
                    * SPCR0, SPCR1
                    *
0000a804            INQS0   EQU   10*BITS+MSTR+4        master, 10 bits, CPOL,CPHA=0,0, baud=2 MHz
0000970b            INQS1   EQU   23*DSCKL+SPE+11       start QSPI, DSCK=1.4375 uS, DTL=22 uS
a804970b            INQS01  EQU   INQS0*$10000+INQS1    form into long word
                    *
                    * SPCR2, SPCR3
                    *
0000420f            INQS2   EQU   2*ENDQ+WREN+$F        wrap, endq = $2, newq = $F
00000000            INQS3   EQU   $0000                 nothing special, same as RESET state
420f0000            INQS23  EQU   INQS2*$10000+INQS3    form into long word
                    *
                    ******  QSPI RAM addresses and initialization values
                    *
fffffd20            TXRAM0  EQU   $FFFFFD20     transmit RAM, entry 0
fffffd24            TXRAM2  EQU   $FFFFFD24     transmit RAM, entry 2
fffffd3e            TXRAMF  EQU   $FFFFFD3E     transmit RAM, entry F
                    *
fffffd40            CRAM0   EQU   $FFFFFD40     control RAM, entry 0
fffffd4f            CRAMF   EQU   $FFFFFD4F     control RAM, entry F
                    *
```

Figure 9. Use of QSPI to Control A/D Conversions — 2 MHz A/D (Sheet 1 of 3)

```
                          *********************
                          * QSPI RECEIVE RAM *
                          *********************
                          *                                   entry #
                          *                                   -------
ffffd00                   FUELPSI EQU     $FFFFFD00             0    QSPI location of A/D pressure result
ffffd02                   TEMP    EQU     $FFFFFD02             1    QSPI location of A/D temperature result
ffffd04                   VOLTAGE EQU     $FFFFFD04             2    QSPI location of A/D voltage result
                          *
                          *
                          ***********************************************
                          * QSPI TRANSMIT RAM INITIALIZATION CONSTANTS *
                          ***********************************************
                          *
                          *
                                                                            TXQ entry     sensor
                                                                            ---------     ------
000000c0                  TXR0     EQU   3*64            A/D channel 3 address    0       temperature
00000100                  TXR1     EQU   4*64            A/D channel 4 address    1       voltage
00000180                  TXR2     EQU   6*64            A/D channel 6 address    2       pressure
                          *
00000180                  TXRF     EQU   6*64            A/D channel 6 address    F       pressure
                          *
00c00100                  TXR01    EQU   TXR0*$10000+TXR1     form into a LONG WORD
                          *
                          * multiply A/D address by 64 to put the LSB into bit 6 of the 10-bit transfer
                          *  (MSB of the 4-bit A/D address will be MSB of 10-bit transfer)
                          *
                          * NOTE: transmit queue entry 0 requests a conversion on A/D channel 3,
                          *       the temperature sensor. This result will be returned into receive
                          *       RAM in queue entry 1. The A/D result always gets transmitted
                          *       on the A/D transfer following its request.
                          *
                          **********************************************
                          * QSPI CONTROL RAM INITIALIZATION CONSTANTS *
                          **********************************************
                          *
00000070                  CRXB     EQU   BITSE+DSCK+DT        10-bits, both delays - same for all transfers
                          *
00007070                  CRXW     EQU   CRXB*$100+CRXB       form into a WORD
70707070                  CRXL     EQU   CRXW*$10000+CRXW     form into a LONG WORD
                          *
                          *
                          *****  Misc.
                          *
00001388                  VREF     EQU   5000                 VREF is 5000 millivolts
00004000                  SETPT    EQU   $4000                address of temperature setpoint variable
                          *
                          ************************************************************************
                          *****************  QSPI initialization and startup   ******************
                          ************************************************************************
                          *
00005000                           ORG   $5000
                          *
                          * Initialize QSPI TRANSMIT RAM
                          *
00005000 21fc 00c0 0100 START      MOVE.L     #TXR01,TXRAM0       entries 0, 1
         fd20
00005008 31fc 0180 fd24            MOVE.W     #TXR2,TXRAM2        entry 2
0000500e 31fc 0180 fd3e            MOVE.W     #TXRF,TXRAMF        entry F
                            *
                            * Initialize QSPI CONTROL RAM
                            *
00005014 21fc 7070 7070            MOVE.L     #CRXL,CRAM0         entries 0, 1, 2, 3      (3 is superfluous)
         fd40
0000501c 11fc 0070 fd4f            MOVE.B     #CRXB,CRAMF         entry F
                            *
                            * Initialize QSPI control registers, START transfers
                            *
00005022 21fc 0008 0f0e            MOVE.L     #INQPORT,QPDRW      setup QPDR, QPAR, QDDR
         fc14
0000502a 21fc 420f 0000            MOVE.L     #INQS23,SPCR2       setup SPCR2, SPCR3
         fc1c
00005032 21fc a804 970b            MOVE.L     #INQS01,SPCR0       setup SPCR0, SPCR1, start QSPI.
         fc18
                            *
                            *
```

Figure 9. Use of QSPI to Control A/D Conversions — 2 MHz A/D (Sheet 2 of 3)

```
0000503a 0838 0007 fc1f WAIT     BTST.B   #7,SPSR          wait until a valid conversion result
                                                                        is available for all channe!s
00005040 67f8                    BEQ.B    WAIT
                           *
                           *     All data available, continue on to main program.
                           *
                           ***********************************************************************
                           **********************   CPU data acquisition   ***********************
                           ***********************************************************************
                           *
                           *    The following code could be periodically executed in response
                           *  to a real-time interrupt. The interrupt could even be generated
                           *  by the QSPI, upon completion of each queue.
                           *
                           *
00005042 303c 0117          INTSRV MOVE.W  #279,D0          load constant for minimum fuel pressure
00005046 b078 fd00                 CMP.W   FUELPSI,D0       test if A/D pressure result is below minimum
0000504a 6504                      BCS.B   CHKRCV
                           *
0000504c 6146                      BSR.B   LOPRESS          generate fuel pressure warning
0000504e 600c                      BRA.B   CHKTEMP          speeds up interrupt service routine
                           *
00005050 303c 0145          CHKRCV MOVE.W  #325,D0          constant for recovered fuel pressure
00005054 b078 fd00                 CMP.W   FUELPSI,D0       test if A/D pressure result is above minimum
00005058 6202                      BHI.B   CHKTEMP
                           *
0000505a 6138                      BSR.B   PRESSOK          cancel fuel pressure warning
                           *
                           *
                           *    The following code segment will control
                           * a temperature using a +/- 5 count deadband.
                           *
                           *
0000505c 3038 4000         CHKTEMP MOVE.W  SETPT,D0         get temperature setpoint
00005060 5b40                      SUBQ.W  #5,D0            compute lower threshold
00005062 b078 fd02                 CMP.W   TEMP,D0          compare with A/D result
00005066 6508                      BCS.B   OK1              branch if actual temp. is above threshold
                           *
00005068 6100 002a                 BSR     HEATON           activate heater
0000506c 6000 0012                 BRA     DOVOLTS          speeds up interrupt service routine
                           *
00005070 3038 4000         OK1     MOVE.W  SETPT,D0         get temperature setpoint
00005074 5a40                      ADDQ.W  #5,D0            compute upper threshold
00005076 b078 fd02                 CMP.W   EMP,D0           compare with A/D result
0000507a 6204                      BHI.B   DOVOLTS          branch if actual temp. is below threshold
                           *
0000507c 6100 0016                 BSR     HEATON           activate heater
                           *
                           *
                           *    The following code segment will measure voltage on
                           * A/D channel 4 and scale the result into millivolts.
                           *
                           *
00005080 303c 1388         DOVOLTS MOVE.W  #VREF,D0         load scale numerator (VREF = 5000 mV)
00005084 c0f8 fd04                 MULU.W  VOLTAGE,D0       multiply by A/D channel 4 conversion result
00005088 e088                      LSR.L   #8,D0            divide by 256
0000508a e488                      LSR.L   #2,D0            divide by 4 (total of divide by 1024)
0000508c 4241                      CLR.W   D1
0000508e d141                      ADDX.W  D1,D0            round for maximum accuracy, result in D0
00005090 6102                      BSR.B   DISPV            display voltage on a digital readout
                           *
                           *
00005092 4e73                      RTE           return from interrupt service routine
                           *
                           *
00005094                   LOPRESS EQU     *     dummy subroutines
00005094                   PRESSOK EQU     *
00005094                   HEATON  EQU     *
00005094                   HEATOFF EQU     *
00005094                   DISPV   EQU     *
00005094 4e75                      RTS
                           *
                           *


=====                      0 Error(s)
=====                      0 Warning(s)
```

Figure 9. Use of QSPI to Control A/D Conversions — 2 MHz A/D (Sheet 3 of 3)

QUEUE ENTRY NUMBER	TRANSMIT RAM (ADDR) CONTENTS	CONTROL RAM (ADDR) CONTENTS	RECEIVE RAM (ADDR) CONTENTS
0	(FFFD20,1) A/D MUX. ADDR. 3	(FFFD40) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD00,1) A/D CHANNEL 6 RESULT
1	(FFFD22,3) A/D MUX. ADDR. 4	(FFFD41) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD02,3) A/D CHANNEL 3 RESULT
ENDQP → 2	(FFFD24,5) A/D MUX. ADDR. 6	(FFFD42) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD04,5) A/D CHANNEL 4 RESULT
3	(X) X	(X) X	(X) X
4	(X) X	(X) X	(X) X
.	.	.	.
.	.	.	.
.	.	.	.
E	(X) X	(X) X	(X) X
NEWQP → F	(FFFD3E,F) A/D MUX. ADDR. 6	(FFFD4F) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD1E,F) A/D INVALID DATA

X = DON'T CARE, UNUSED

	ENTRY NUMBER	QSPI OPERATION FLOW
START NEWQP →	F	REQUEST A/D CHANNEL 6, GET UNDEFINED DATA
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT
ENDQP →	2	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT
	←	SET SPIF AFTER COMPLETION OF ENTRY #2
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT
	2	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT
	2	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT
	.	
	.	
	.	

NOTE:
WRTO = 0
WREN = 1
NEWQP = F
ENDQP = Z

Figure 10. Example Queue Structure and Operation Flow

QUEUE ENTRY NUMBER	TRANSMIT RAM (ADDR) CONTENTS	CONTROL RAM (ADDR) CONTENTS	RECEIVE RAM (ADDR) CONTENTS
0	(FFFD20,1) A/D MUX. ADDR. 3	(FFFD40) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD00,1) A/D CHANNEL 6 RESULT
1	(FFFD22,3) A/D MUX. ADDR. 4	(FFFD41) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD02,3) A/D CHANNEL 3 RESULT
ENDQP → 2	(FFFD24,5) A/D MUX. ADDR. 6	(FFFD42) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD04,5) A/D CHANNEL 4 RESULT
3	(X) X	(X) X	(X) X
.	.	.	.
.	.	.	.
.	.	.	.
D	(X) X	(X) X	(X) X
E	(FFFD3C,D) OUTPUT PORT DATA	(FFFD4E) 8 BIT, NO DELAYS, PCS1 = 0	(FFFD1C,D) PORT INPUT DATA
F	(FFFD3E,F) A/D MUX. ADDR. 6	(FFFD4F) 10 BIT, DSCK, DT ENABLES, PCS0 = 0	(FFFD1E,F) LAST A/D CHANNEL RESULT

X = DON'T CARE, UNUSED

	ENTRY NUMBER	QSPI OPERATION FLOW	
	.		
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT	PRIMARY QUEUE
ENDQP →	2	← REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT	
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT	
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT	
ENDQP →	2	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT	
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT	
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT	
WRITE NEWQP = E →	E	TRANSFER TO PORT	SUBQUEUE
	F	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT	
NORMAL QUEUE RESUMES →	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT	PRIMARY QUEUE
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT	
ENDQP →	2	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT	
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT	
	1	REQUEST A/D CHANNEL 4, GET CHANNEL 3 RESULT	
ENDQP →	2	REQUEST A/D CHANNEL 6, GET CHANNEL 4 RESULT	
	0	REQUEST A/D CHANNEL 3, GET CHANNEL 6 RESULT	

NOTE:
WRTO = 0
WREN = 1
INITIAL NEWQP = F
ENDQP = Z

Figure 11. Example Subqueue Structure and Operation Flow