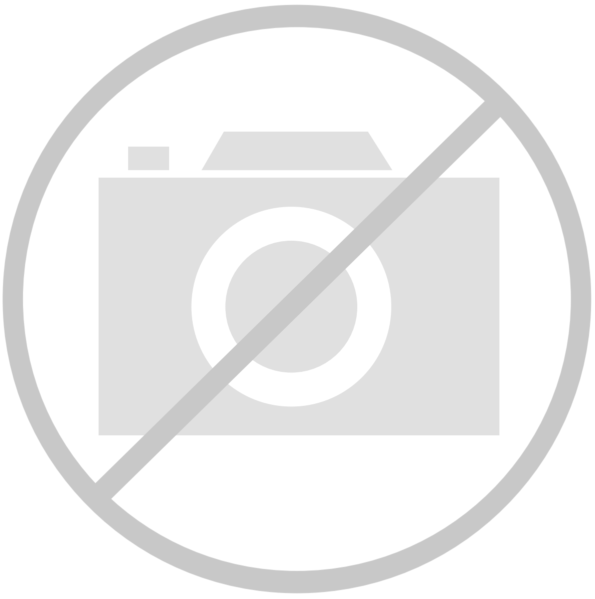
NXP USA Inc. - MC68F375BGMVR33R Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	CPU32
Core Size	32-Bit Single-Core
Speed	33MHz
Connectivity	CANbus, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	48
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	10K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.25V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	217-BBGA
Supplier Device Package	217-PBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68f375bgmvr33r

instruction pipeline. Pipeline flushes are also signaled with $\overline{\text{IFETCH}}$. Monitoring these two signals allows a bus state analyzer to synchronize itself to the instruction stream and monitor its activity.

3.10.12 On-Chip Breakpoint Hardware

An external breakpoint input and on-chip breakpoint hardware allow a breakpoint trap on any memory access. Off-chip address comparators preclude breakpoints unless show cycles are enabled. Breakpoints on instruction prefetches that are ultimately flushed from the instruction pipeline are not acknowledged; operand breakpoints are always acknowledged. Acknowledged breakpoints initiate exception processing at the address in exception vector number 12, or alternately enter background mode.

the halt monitor (HME) enable bit in SYPCR. Refer to **4.6.5.2 Double Bus Faults** for more information.

4.4.5 Spurious Interrupt Monitor

During interrupt exception processing, the CPU32 normally acknowledges an interrupt request, arbitrates among various sources of interrupt, recognizes the highest priority source, and then acquires a vector or responds to a request for autovectoring. The spurious interrupt monitor asserts the internal bus error signal ($\overline{\text{BERR}}$) if no interrupt arbitration occurs during interrupt exception processing. The assertion of $\overline{\text{BERR}}$ causes the CPU32 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled.

Refer to **4.8 Interrupts** for further information. For detailed information about interrupt exception processing, refer to **4.8.1 Interrupt Exception Processing**.

4.4.6 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in SYPCR. When enabled, the watchdog requires that a service sequence be written to the software service register (SWSR) on a periodic basis. If servicing does not take place, the watchdog times out and asserts the $\overline{\text{RESET}}$ signal.

Each time the service sequence is written, the software watchdog timer restarts. The sequence to restart the software watchdog requires the following steps:

- Write 0x55 to SWSR
- Write 0xAA to SWSR

Both writes must occur before timeout in the order listed. Any number of instructions can be executed between the two writes.

The clock rate of the watchdog timer is affected by clock mode, the software watchdog prescale (SWP) bit, and the software watchdog timing (SWT[1:0]) field in SYPCR. In slow reference mode and external clock mode, f_{ref} or $f_{ref} \div 512$ can be used to clock the watchdog timer. The options in fast reference mode are $f_{ref} \div 128$ or $(f_{ref} \div 128) \div 512$. In all cases, the divide-by-512 option is selected when SWP = 1.

The value of SWP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 4-11**. System software can change SWP value.

Table 4-11 SWP Reset States

V_{DDSYN}/MODCLK	SWP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

SWT[1:0] selects the divide ratio used to establish the software watchdog timeout period.

4.5.1.11 Autovector Signal

The autovector signal (AVEC) can be used to terminate interrupt acknowledgment cycles for external interrupts only. Assertion of AVEC causes the CPU32 to generate vector numbers to locate an interrupt handler routine. If AVEC is continuously asserted, autovectors are generated for all external interrupt requests. AVEC is ignored during all other bus cycles. Refer to **4.8 Interrupts** for more information. AVEC for external interrupt requests can also be supplied internally by chip-select logic. Refer to **4.9 Chip Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **4.6.6 External Bus Arbitration** for more information.

4.5.2 Dynamic Bus Sizing

The MCU dynamically interprets the port size of an addressed device during each bus cycle, allowing operand transfers to or from 8-bit and 16-bit ports.

During a bus transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the DSACK inputs, as shown in **Table 4-19**. Chip-select logic can generate data size acknowledge signals for an external device. Refer to **4.9 Chip Selects** for more information.

Table 4-19 Effect of DSACK Signals

DSACK1	DSACK0	Result
1	1	Insert wait states in current bus cycle
1	0	Complete cycle — Data bus port size is 8 bits
0	1	Complete cycle — Data bus port size is 16 bits
0	0	Reserved

If the CPU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the first 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the DSACK signals to indicate the port width. For instance, a 16-bit external device always returns DSACK for a 16-bit port (regardless of whether the bus cycle is a byte or word operation).

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on data bus bits [15:0], and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins.

Operand bytes are designated as shown in **Figure 4-10**. OP[0:3] represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes

4.7.2 Reset Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing and can be caused by internal or external events. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in the exception vector table. The exception vector table consists of 256 four-byte vectors and occupies 1024 bytes of address space. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **3.9 Exception Processing** for more information.

Reset is the highest-priority CPU32 exception. Unlike all other exceptions, a reset occurs at the end of a bus cycle and not at an instruction boundary. Handling resets in this way prevents write cycles in progress at the time the reset signal is asserted from being corrupted. However, any processing in progress is aborted by the reset exception and cannot be restarted. Only essential reset tasks are performed during exception processing. Other initialization tasks must be accomplished by the exception handler routine.

NOTE

External circuitry is required to disable external bus configuration logic until $\overline{\text{DS}}$ and R/$\overline{\text{W}}$ are negated to ensure that bus cycles in progress at the time $\overline{\text{RESET}}$ is asserted complete correctly.

4.7.3 Reset Source Summary

SCIM2E reset control logic determines the cause of a reset, synchronizes request signals to CLKOUT, and asserts reset control logic. All resets are gated by CLKOUT. Asynchronous resets can occur on any clock edge and are assumed to be catastrophic. Synchronous resets are timed to occur at the end of bus cycles. When a synchronous reset is detected, the SCIM2E bus monitor is automatically enabled. If the bus cycle during which a synchronous reset is detected does not terminate normally, the bus monitor will terminate the cycle and allow the reset to proceed. **Table 4-22** is a summary of reset sources.

Table 4-22 Reset Source Summary

Type	Source	Timing
External	Assertion of $\overline{\text{RESET}}$ pin	Synchronous
Power on	Rising voltage on V_{DD}	Asynchronous
Software watchdog	Timeout of software watchdog	Asynchronous
Halt	Halt monitor (e.g. double bus fault)	Asynchronous
Loss of clock	Reference failure caught by loss of clock detector	Synchronous
Test	Test submodule	Synchronous

NOTE

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

4.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the SCIM2E, the CPU32, and a device or module requesting interrupt service. This discussion provides an overview of the entire interrupt process. Chip-select logic can also be used to respond to interrupt requests. Refer to **4.9 Chip Selects** for more information.

4.8.1 Interrupt Exception Processing

The CPU32 handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in a vector table located from 0x00000 to 0x001FF. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **3.9 Exception Processing** for more information.

4.8.2 Interrupt Priority and Recognition

The CPU32 provides seven levels of interrupt priority (1–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in the condition code register (CCR).

The IP field consists of CCR bits [7:5]. Binary values 0b000 to 0b111 provide eight priority masks. Each mask prevents an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized. When the IP field contains 0b000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

There are seven interrupt request signals ($\overline{\text{IRQ[7:1]}}$) with corresponding external pins that can be asserted by microcontroller modules or external devices. Simultaneous requests of different priorities can be made. Internal assertion of an interrupt request line does not affect the state of the corresponding MCU pin.

External interrupt requests are routed to the CPU32 via the EBI and SCIM2E interrupt control logic. All requests for interrupt service are treated as if they come from internal modules. The CPU32 treats external interrupt requests as if they come from the SCIM2E.

The $\overline{\text{IRQ[6:1]}}$ pins are active-low level-sensitive inputs. The $\overline{\text{IRQ7}}$ pin is an active-low transition-sensitive input; it requires both an edge and a voltage level to be valid.

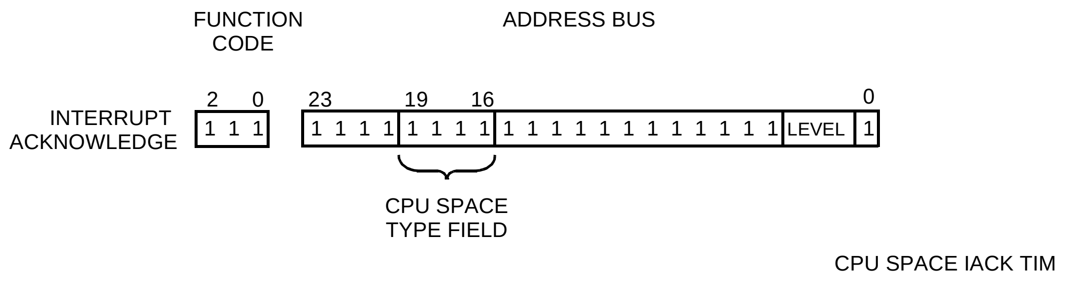


Figure 4-23 CPU Space Encoding for Interrupt Acknowledge

Chip-select address match logic functions only after the SCIM2E has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2E will never have their IACK cycles terminated by chip-select generated AVEC or DSACK.

Use the procedure that follows to configure a chip select to provide IACK cycle termination.

1. Program the base address field to all ones.
2. Program block size to no more than 64 Kbytes, so that the address comparator checks ADDR[19:16] against the corresponding bits in the base address register. (The CPU space bus cycle type is placed on ADDR[19:16]).
3. Set the R/W field to read only. An interrupt acknowledge cycle is performed as a read in CPU space.
4. Set the BYTE field to lower byte when using a 16-bit port, as the external vector for a 16-bit port is fetched from the lower byte. Set the BYTE field to upper byte when using an 8-bit port.

If an interrupting device does not provide a vector number, an autovector must be generated, either by asserting the AVEC pin or by having the chip select assert AVEC internally. The latter is accomplished by setting the chip-select option register AVEC bit. This terminates the bus cycle.

4.9.4.2 Chip-Select Reset Operation

The LSB of each of the 2-bit pin assignment fields in CSPAR0 and CSPAR1 has a reset value of one. The reset values of the MSBs of each field are determined by the states of DATA[7:1] during reset. Weak internal pull-up devices condition each of the data lines so that chip-select operation is selected by default out of reset. Excessive bus loading can overcome the internal pull-up devices, resulting in inadvertent configuration out of reset. Use external pull-up resistors or active devices to avoid this.

The base address fields in chip-select base address registers CSBAR[0,3, 5:10], CSBAR3, and CSBAR0 and chip-select option registers CSOR[10:5], CSOR3, and CSOR0 have the reset values shown in **Table 4-40**. The BYTE and R/W fields of each option register have a reset value of “disable”, so that a chip-select signal cannot be asserted until the base and option registers are initialized.

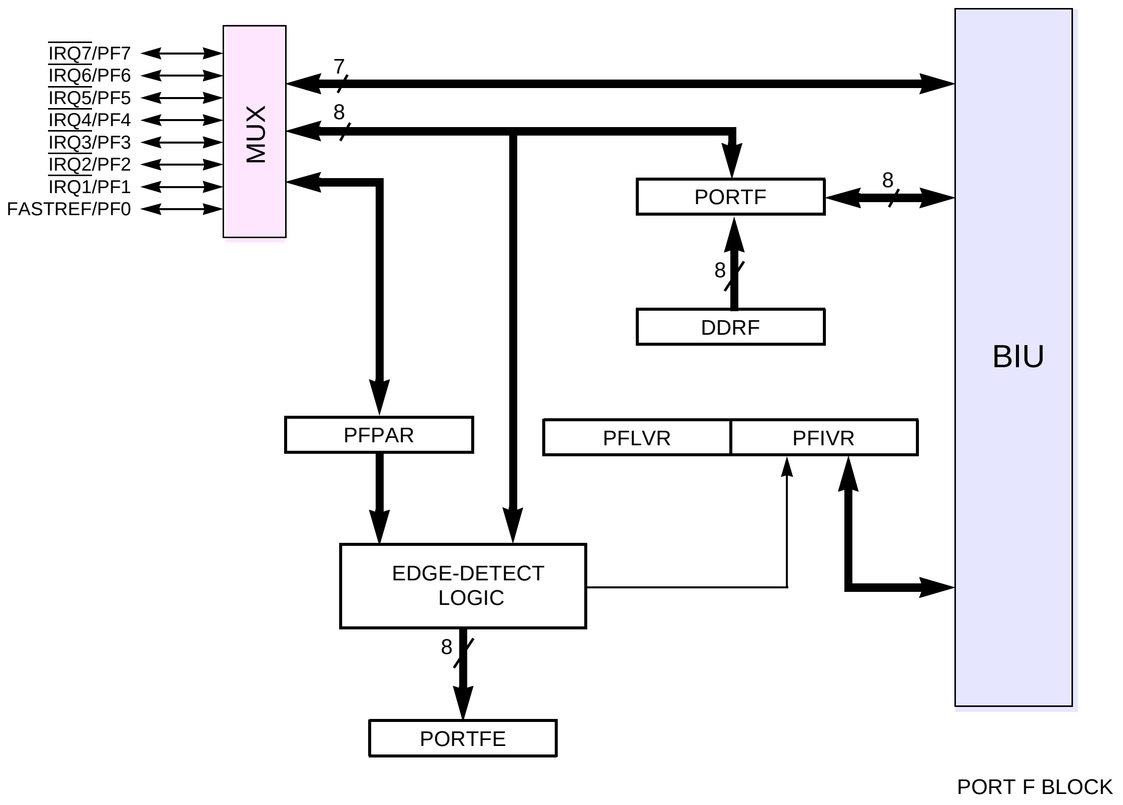


Figure 4-24 Port F Block Diagram

4.10.4.1 Port F Data Register

PORTF0 — Port F Data Register 0 **0xYF FA19**
PORTF1 — Port F Data Register 1 **0xYF FA1B**

MSB 7	6	5	4	3	2	1	LSB 0
PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:							
U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Bits [15:8] are reserved and will always read zero.

same priority. The QADC is configured to support interrupt acknowledge (IACK) cycles and vector generation.

5.11.1 Interrupt Operation

Figure 5-10 displays the QADC64 interrupt flow.

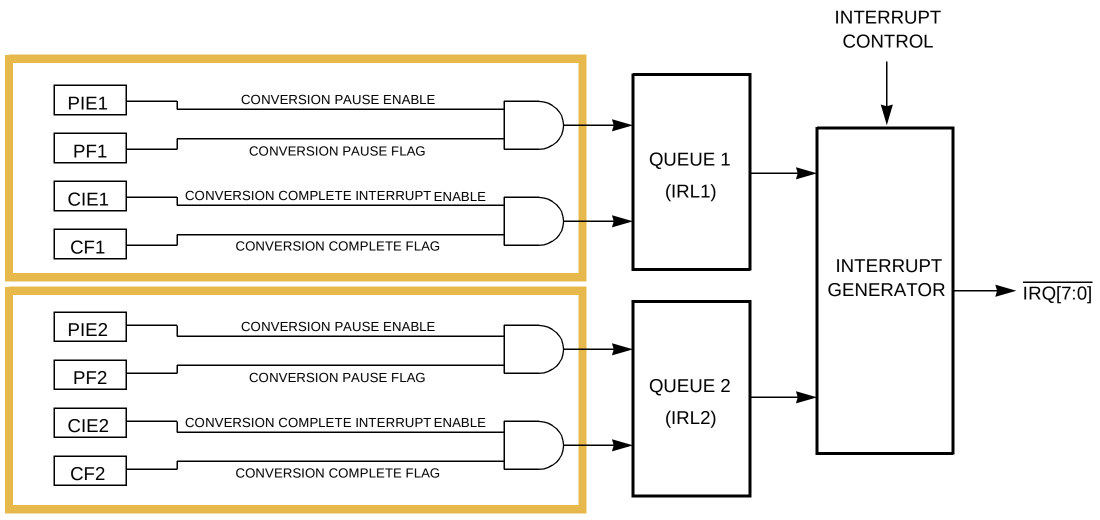


Figure 5-10 QADC64 Interrupt Flow Diagram

5.11.1.1 Polled and Interrupt-Driven Operation

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

Table 5-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity. If interrupts are enabled for an event, the QADC requests interrupt service when the event occurs. Using interrupts does not require continuously polling the status flags to see if an event has taken place. However, status flags must be cleared after an interrupt is serviced, in order to disable the interrupt request. In both polled and interrupt-driven operating modes, status flags must be re-enabled after an event occurs. Flags are re-enabled by clearing appropriate QASR bits in a particular sequence. The register must first be read, then zeros must be written to the flags that are to be cleared. If a new event occurs betwe en the time that the register is read and the time that it is written, the associate d flag is not cleared.

5.11.2 Interrupt Sources

The QADC includes four sources of interrupt service requests, each of which is separately enabled. Each time the result is written for the last conversion command word (CCW) in a queue, the completion flag for the corresponding queue is set, and when

AMCHAN[1:0]. Address lines to select between the 4 multiplexers. This selection appears redundant since the final selection between the 4 multiplexers is done by the QADC64 module. This redundancy is added to reduce the dynamic charging current required from the input pin. These pins perform a disable function more than a select function. Only 1 of the 4 multiplexers can be active at a time. These address lines disable the other 3 multiplexers so that none of their switches are closed. As the selected multiplexer cycles through its various inputs, the other multiplexers draw no dynamic charging current through their inputs. Without this disable function (as in the external multiplexer case), pins on all 4 multiplexers would draw current even though only 1 multiplexer would actually be in use.

Assuming that AMCHAN[1:0] change simultaneously with AMMA[2:0], break before make switching ensures that on each channel change the following occurs:

1. All 4 multiplexers are disabled
2. The desired input ANX0-15 is selected
3. The appropriate multiplexer is enabled

AMPMUX. When low this signal disables all multiplexer inputs. It is used in conjunction with AMCHAN[1:0] to reduce dynamic charging current into the AMUX analog pins. This signal is normally low when the direct inputs to the QADC64 are selected.

AMADB[1:0]. The 2 multiplexer outputs from the AMUX.

Table 5-22 AMUX I/O Functionality

AMPMUX	AMCHAN[1:0]	AMMA[2:0]	AMADB3	AMADB2	AMADB1	AMADB0
0	XX	XXX	Z	Z	Z	Z
1	00	000	Z	Z	Z	ANX0
1	00	001	Z	Z	Z	ANX2
1	00	010	Z	Z	Z	ANX4
1	00	011	Z	Z	Z	ANX6
1	00	100	Z	Z	Z	ANX8
1	00	101	Z	Z	Z	ANX10
1	00	110	Z	Z	Z	ANX12
1	00	111	Z	Z	Z	ANX14
1	01	000	Z	Z	ANX1	Z
1	01	001	Z	Z	ANX3	Z
1	01	010	Z	Z	ANX5	Z
1	01	011	Z	Z	ANX7	Z
1	01	100	Z	Z	ANX9	Z
1	01	101	Z	Z	ANX11	Z
1	01	110	Z	Z	ANX13	Z
1	01	111	Z	Z	ANX15	Z

6.7.1.2 QSPI Control Register 1

SPCR1 enables the QSPI and specifies transfer delays. The CPU has read/write access to SPCR1, but the QSPI has read access only to all bits except SPE. SPCR1 must be written last during initialization because it contains SPE. The QSPI automatically clears this bit after it completes all serial transfers or when a mode fault occurs. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

SPCR1 — QSPI Control Register 1 **0xYF FC1A**

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

Table 6-15 SPCR1 Bit Settings

Bit(s)	Name	Description
15	SPE	QSPI enable. Refer to **6.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O. 1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.
14:8	DSCKL	Delay before SCK. When the DSCK bit is set in a command RAM byte, this field determines the length of the delay from PCS valid to SCK transition. The following equation determines the actual delay before SCK: $\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$ where DSCKL equals is in the range of 1 to 127. Refer to **6.7.5.3 Delay Before Transfer** for more information.
7:0	DTL	Length of delay after transfer. When the DT bit is set in a command RAM byte, this field determines the length of the delay after a serial transfer. The following equation is used to calculate the delay: $\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$ where DTL is in the range of 1 to 255. A zero value for DTL causes a delay-after-transfer value of 8192 $\div f_{SYS}$ (204.8 µµs with a 40-MHz system clock). Refer to **6.7.5.4 Delay After Transfer** for more information.

6.7.1.3 QSPI Control Register 2

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU has read/write access to SPCR2, but the QSPI has read access only. Writes to this register are buffered. New SPCR2 values become effective only after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. Reads of SPCR2 return the current value of the register, not the buffer.

of the bit in PORTQS that corresponds to the chip-select pin determines the base state of the chip-select signal. If the base state is zero, chip-select assertion must be active high (PCS bit in command RAM must be set); if base state is one, assertion must be active low (PCS bit in command RAM must be cleared). PORTQS bits are cleared during reset. If no new data is written to PORTQS before pin assignment and configuration as an output, the base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

6.7.5.7 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wraparound mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven QSPI service is used, the service routine must clear the SPIF bit to end the current interrupt request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not end the current request.

Wraparound mode is exited by clearing the WREN bit or by setting the HALT bit in SPCR3. Exiting wraparound mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.

6.7.6 Slave Mode

Clearing the MSTR bit in SPCR0 selects slave mode operation. In slave mode, the QSPI is unable to initiate serial transfers. Transfers are initiated by an external SPI bus master. Slave mode is typically used on a multi-master SPI bus. Only one device can be bus master (operate in master mode) at any given time.

Before QSPI operation is initiated, QSMCM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO, MOSI, SCK, and PCS0/SS. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the serial clock input in slave mode and must be assigned to the QSPI for proper operation. Assertion of the active-low slave select signal SS initiates slave mode operation.

Before slave mode operation is initiated, DDRQS must be written to direct data flow on the QSPI pins used. Configure the MOSI, SCK and PCS0/SS pins as inputs. The MISO pin must be configured as an output.

6.9.11 QSCI1 Receive Queue Software Flow Chart

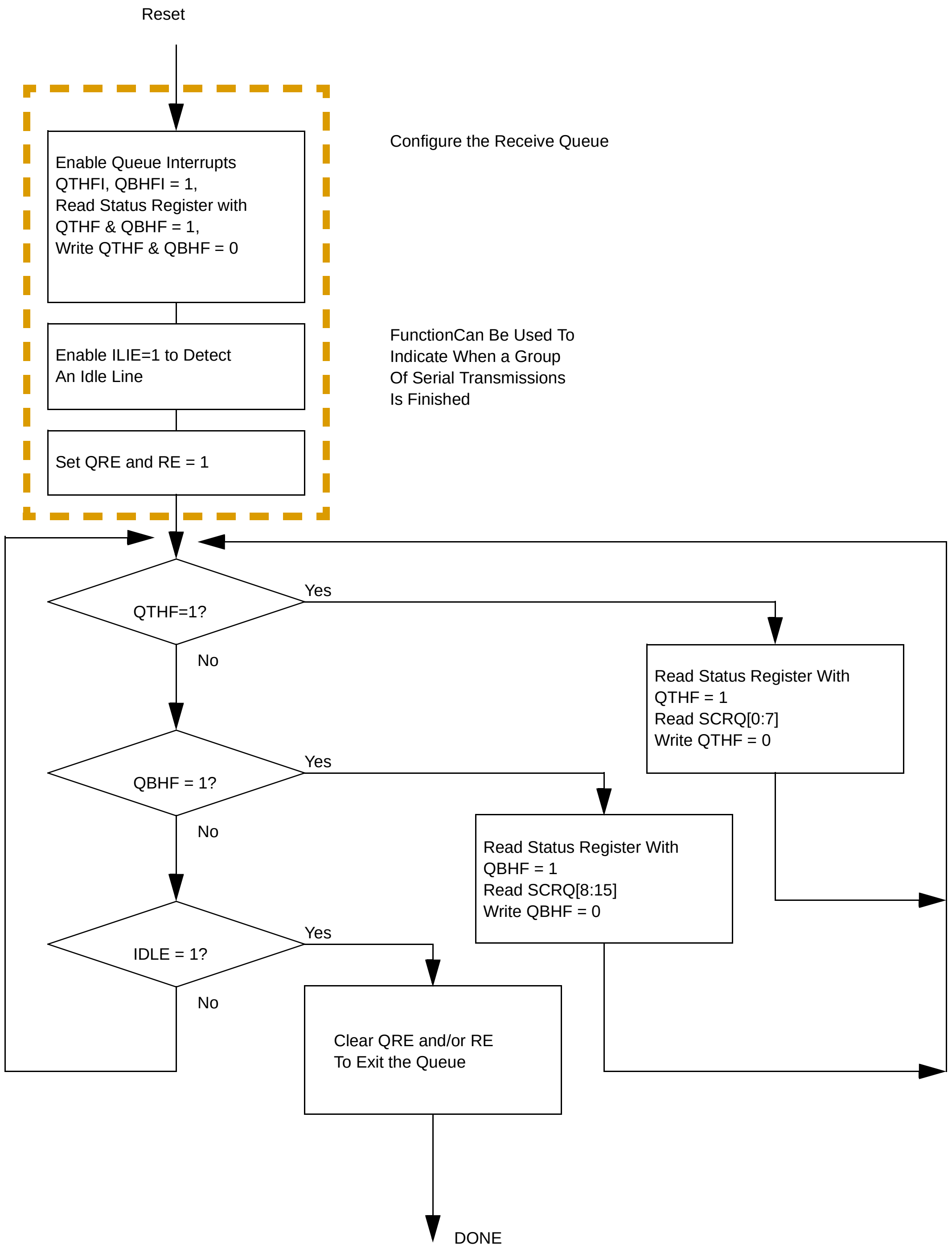


Figure 6-19 Queue Receive Software Flow

	158	7	4 3		0	
$0	TIME STAMP	CODE	LENGTH			CONTROL/STATUS
$2	ID[28-18]	SRR	IDE	ID[17-15]		ID_HIGH
$4	ID[14-0]				RTR	ID_LOW
$6	DATA BYTE 0	DATA BYTE 1				
$8	DATA BYTE 2	DATA BYTE 3				
$A	DATA BYTE 4	DATA BYTE 5				
$C	DATA BYTE 6	DATA BYTE 7				
$E	RESERVED					

Figure 7-3 Extended ID Message Buffer Structure

	158	7	4 3			0	
$0	TIME STAMP	CODE	LENGTH				CONTROL/STATUS
$2	ID[28:18]	RTR	0	0	0	0	ID_HIGH
$4	16-BIT TIME STAMP						ID_LOW
$6	DATA BYTE 0	DATA BYTE 1					
$8	DATA BYTE 2	DATA BYTE 3					
$A	DATA BYTE 4	DATA BYTE 5					
$C	DATA BYTE 6	DATA BYTE 7					
$E	RESERVED						

Figure 7-4 Standard ID Message Buffer Structure

7.4.1.1 Common Fields for Extended and Standard Format Frames

Table 7-1 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 7-17 PRESDIV Bit Settings

Bit(s)	Name	Description
15:8	PRESDIV	Prescaler divide factor. PRESDIV determines the ratio between the system clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $$\text{S-clock} = \frac{f_{sys}}{\text{PRESDIV} + 1}$$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the system clock. The valid programmed values are 0 through 255.
7:0	CANCTRL2	See **Table 7-18**.

7.8.6 Control Register 2

CANCTRL2 — Control Register 2 **0xYF F088**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PRESDIV								RJW		PSEG			PSEG2		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 7-18 CANCTRL2 Bit Settings

Bit(s)	Name	Description
15:8	PRESDIV	See **Table 7-17**.
7:6	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are 0 through 3. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta
5:3	PSEG1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment 1 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
2:0	PSEG2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment 2 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 2 is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

7.8.7 Free Running Timer

TIMER — Free Running Timer Register **0xYF F08A**

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

8.4.11 Channel Interrupt Status Register

The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions specify via microcode when an interrupt flag is set. Setting a flag causes the TPU3 to make an interrupt service request if the corresponding CIER bit is set. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU3 register that can be accessed on a byte basis.

CISR — Channel Interrupt Status Register **0xYF FE20**

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

Table 8-16 CISR Bit Settings

Bit(s)	Name	Description
15:0	CH[15:0]	Channel interrupt status 0 = Channel interrupt not asserted 1 = Channel interrupt asserted

8.4.12 Link Register

LR — Link Register **0xYF FE22**

Used for factory test only.

8.4.13 Service Grant Latch Register

SGLR — Service Grant Latch Register **0xYF FE24**

Used for factory test only.

8.4.14 Decoded Channel Number Register

DCNR — Decoded Channel Number Register **0xYF FE26**

Used for factory test only.

8.4.15 TPU3 Module Configuration Register 2

TPUMCR2 — TPU Module Configuration Register 2 **0xYF FE28**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							DIV2	SOFT RST	ETBANK		FPSCK			T2CF	DTPU
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A-1 TouCAN (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F130 — 0xYF F13F	S/U	MBUFF3	TouCAN Message Buffer 3. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F140 — 0xYF F14F	S/U	MBUFF4	TouCAN Message Buffer 4. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F150 — 0xYF F15F	S/U	MBUFF5	TouCAN Message Buffer 5. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F160 — 0xYF F16F	S/U	MBUFF6	TouCAN Message Buffer 6. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F170 — 0xYF F17F	S/U	MBUFF7	TouCAN Message Buffer 7. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F180 — 0xYF F18F	S/U	MBUFF8	TouCAN Message Buffer 8. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F190 — 0xYF F19F	S/U	MBUFF9	TouCAN Message Buffer 9. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1A0 — 0xYF F1AF	S/U	MBUFF10	TouCAN Message Buffer 10. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1B0 — 0xYF F1BF	S/U	MBUFF11	TouCAN Message Buffer 11. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1C0 — 0xYF F1CF	S/U	MBUFF12	TouCAN Message Buffer 12. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1D0 — 0xYF F1DF	S/U	MBUFF13	TouCAN Message Buffer 13. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1E0 — 0xYF F1EF	S/U	MBUFF14	TouCAN Message Buffer 14. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1F0 — 0xYF F1FF	S/U	MBUFF15	TouCAN Message Buffer 15. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U

Table A-8 SCIM2E (Single-Chip Integration Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FA60	S	CSBAR5	SCIM2E Chip-Select Base Address Register 5. See **Table 4-36** for bit descriptions.	16	X
0xYF FA62	S	CSOR5	SCIM2E Chip-Select Option Register 5. See **Table 4-37** for bit descriptions.	16	X
0xYF FA64	S	CSBAR6	SCIM2E Chip-Select Base Address Register 6. See **Table 4-36** for bit descriptions.	16	X
0xYF FA66	S	CSOR6	SCIM2E Chip-Select Option Register 6. See **Table 4-37** for bit descriptions.	16	X
0xYF FA68	S	CSBAR7	SCIM2E Chip-Select Base Address Register 7. See **Table 4-36** for bit descriptions.	16	X
0xYF FA6A	S	CSOR7	SCIM2E Chip-Select Option Register 7. See **Table 4-37** for bit descriptions.	16	X
0xYF FA6C	S	CSBAR8	SCIM2E Chip-Select Base Address Register 8. See **Table 4-36** for bit descriptions.	16	X
0xYF FA6E	S	CSOR8	SCIM2E Chip-Select Option Register 8. See **Table 4-37** for bit descriptions.	16	X
0xYF FA70	S	CSBAR9	SCIM2E Chip-Select Base Address Register 9. See **Table 4-36** for bit descriptions.	16	X
0xYF FA72	S	CSOR9	SCIM2E Chip-Select Option Register 9. See **Table 4-37** for bit descriptions.	16	X
0xYF FA74	S	CSBAR10	SCIM2E Chip-Select Base Address Register 10. See **Table 4-36** for bit descriptions.	16	X
0xYF FA76	S	CSOR10	SCIM2E Chip-Select Option Register 10. See **Table 4-37** for bit descriptions.	16	X

Table A-9 SRAM Module (Static Random Access Memory)

Address	Access	Symbol	Register	Size	Reset
0xYF FB00	S	RAMMCR	RAM Module Configuration Register. See **Table 11-1** for bit descriptions.	16	X
0xYF FB02	S	RAMTST	RAM Module Configuration Register.	16	X
0xYF FB04	S	RAMBAH	RAM Module Base Address High Register. See **Table 11-3** for bit descriptions.	16	X
0xYF FB06	S	RAMBAL	RAM Module Base Address Low Register. See **Table 11-3** for bit descriptions.	16	X

Table A-10 QSMCM (Queued Serial Multi-Channel Module)

Address	Access	Symbol	Register	Size	Reset
0xYF FC00	S	QSMCMMCR	QSMCM Module Configuration Register. See **Table 6-3** for bit descriptions.	16	X
0xYF FC02	T	QTEST	QSMCM Test Register.	16	X
0xYF FC04	S	QILR	QSMCM Interrupt Level, Interrupt Vector Register See **Table 6-4** for bit descriptions.	8	X
0xYF FC05	S	QIVR	QSMCM Interrupt Level, Interrupt Vector Register See **Table 6-5** for bit descriptions.	8	X
0xYF FC06	S	—	Reserved	—	X
0xYF FC07	S	QSPI_IL	QSMCM Interrupt Level, QSPI interrupt level. See **Table 6-6** for bit descriptions.	8	X

D.11 Fast Quadrature Decode TPU Function (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free-running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function.

Figure D-19 and **Figure D-20** show the host interface areas for the FQD function for primary and secondary channels, respectively.

Table D-4 SIOP State Timing[1]

State Number and Name	Max. CPU Clock Cycles	Number of RAM Accesses by TPU
S1 SIOP_INIT		
HSQ = X0	28	7
X1	38	7
S2 DATA_OUT		
HSQ = X0	14	4
X1	24	4
S3 DATA_IN		
HSQ = 0X	14	4
1X	28	6

NOTES:
1. Execution times do not include the time slot transition time (TST = 10 or 14 CPU clocks).

D.17.3.1 XFER_SIZE Greater than 16

XFER_SIZE is normally programmed to be in the range 1-to-16 to match the size of SIOP_DATA, and has thus been shown as a 5-bit value in the host interface diagram. However, the TPU actually uses all 16 bits of the XFER_SIZE parameter when loading BIT_COUNT. In some unusual circumstances this can be used to the user's advantage. If an input device is producing a data stream of greater than 16 bits then manipulation of XFER_SIZE will allow selective capturing of the data. In clock-only mode, the extended XFER_SIZE can be used to generate up to $FFFF clocks.

D.17.3.2 Data Positioning

As stated above, no 'justifying' of the data position in SIOP_DATA is performed by the TPU. This means that in the case of a byte transfer, the data output will be sourced from one byte and the data input will shift into the other byte. This rule holds for all data size options except 16 bits when the full SIOP_DATA register is used for both data output and input.

D.17.3.3 Data Timing

In the example given in Figure D-29, the data output transitions are shown as being completely synchronous with the relevant clock edge and it is assumed that the data input is latched exactly on the opposite clock edge. This is the simplest way to show the examples, but is not strictly true. Since the TPU is a multi-tasking system, and the data channels are manipulated directly by microcode software while servicing the clock edge, there is a finite delay between the relevant clock edge and the data-out being valid or the data-in being latched. This delay is equivalent to the latency in servicing the clock channel due to other TPU activity and is shown as 'Td' in the timing diagram. Td is the delay between the clock edge and the next output data being valid and also the delay between the opposite clock edge and the input data being read. For the vast majority of applications, the delay Td will not present a problem and can be ignored. Only for a system which heavily loads the TPU should the user calculate the worst case latency for the SIOP clock channel + actual SIOP service time (= Td) and ensure that the baud rate is chosen such that HALF_PERIOD - Td is not less that the

Table E-5 AC Timing

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} and V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation	f_{SLW}	0.13	33.6	MHz
1	Clock Period	t_{cyc}	30.0	—	ns
1A	ECLK Period	t_{Ecyc}	$8*t_{cyc}$	—	ns
1B	External Clock Input Period[2,3]	t_{Xcyc}	14.9	—	ns
2, 3	Clock Pulse Width	t_{CW}	$0.5t_{cyc}-3$	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	$t_{cyc}-7$	—	ns
2B, 3B	External Clock Input High/Low Time[2]	t_{XCHL}	7.45	—	ns
4, 5	Clock Rise and Fall Time	t_{Crf}	—	3	ns
4A, 5A	Rise and Fall Time SCIM2E pins: CSBOOT, CLKOUT, $\overline{BKPT}$, $\overline{IFETCH}$, $\overline{IPIPE}$, $\overline{BG}$, $\overline{BR}$, $\overline{BGACK}$, A[23,2:0], FREEZE, $\overline{BERR}$, R/$\overline{W}$, $\overline{HALT}$, $\overline{RESET}$ FC[2:0], A[22:3], D[15:0], SIZE[1:0], $\overline{AS}$, $\overline{DS}$, RMC, AVEC, $\overline{DSACK}$[1:0][4] Fast[5] Slow[5] IRQ[7:1], FASTREF[6]	t_{rf}	—	3 3 >200 >200	ns
4B, 5B	External Clock Rise and Fall Time[7]	t_{XCrf}	—	3	ns
6	Clock High to Address, FC, SIZE Valid	t_{CHAV}	0	0.5	t_{cyc}
7	Clock High to Address, Data, FC, SIZE High Impedance	t_{CHAZx}	0	1.0	t_{cyc}
8	Clock High to Address, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	$0.5t_{cyc}$	ns
9A	$\overline{AS}$, to $\overline{DS}$, or $\overline{CS}$ Asserted (Read)[8]	t_{STSA}	–15	8	ns
9C	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Asserted	t_{CLIA}	2	11	ns
11	Address, FC, SIZE Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted	t_{AVSA}	0.25	—	t_{cyc}
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	1	15	ns
12A	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Negated	t_{CLIN}	1	11	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	0.25	—	t_{cyc}
14	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Read) Width Asserted	t_{SWA}	50	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	23	—	ns
14B	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Write Cycle)	t_{SWDW}	20	—	ns
15	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[9]	t_{SN}	20	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	30	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ Negated	t_{SNRN}	0.25	—	t_{cyc}
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	0.5	t_{cyc}