



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	20MHz
Connectivity	EBI/EMI, SCI, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	15
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	132-BQFP Bumpered
Supplier Device Package	132-PQFP (24.13x24.13)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68332acfc20b1">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68332acfc20b1</a>

**Table 1 Ordering Information**

Package Type	TPU Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
132-Pin PQFP	Motion Control	-40 to +85 °C	16 MHz	2 pc tray	SPAKMC332GCFC16
				36 pc tray	MC68332GCFC16
			20 MHz	2 pc tray	SPAKMC332GCFC20
				36 pc tray	MC68332GCFC20
		-40 to +105 °C	16 MHz	2 pc tray	SPAKMC332GVFC16
				36 pc tray	MC68332GVFC16
			20 MHz	2 pc tray	SPAKMC332GVFC20
				36 pc tray	MC68332GVFC20
		-40 to +125 °C	16 MHz	2 pc tray	SPAKMC332GMFC16
				36 pc tray	MC68332GMFC16
			20 MHz	2 pc tray	SPAKMC332GMFC20
				36 pc tray	MC68332GMFC20
	Standard	-40 to +85 °C	16 MHz	2 pc tray	SPAKMC332CFC16
				36 pc tray	MC68332CFC16
			20 MHz	2 pc tray	SPAKMC332CFC20
				36 pc tray	MC68332CFC20
		-40 to +105 °C	16 MHz	2 pc tray	SPAKMC332VFC16
				36 pc tray	MC68332VFC16
			20 MHz	2 pc tray	SPAKMC332VFC20
				36 pc tray	MC68332VFC20
		-40 to +125 °C	16 MHz	2 pc tray	SPAKMC332MFC16
				36 pc tray	MC68332MFC16
			20 MHz	2 pc tray	SPAKMC332MFC20
				36 pc tray	MC68332MFC20
	Std w/enhanced PPWA	-40 to +85 °C	16 MHz	2 pc tray	SPAKMC332ACFC16
				36 pc tray	MC68332ACFC16
			20 MHz	2 pc tray	SPAKMC332ACFC20
				36 pc tray	MC68332ACFC20
		-40 to +105 °C	16 MHz	2 pc tray	SPAKMC332AVFC16
				36 pc tray	MC68332AVFC16
			20 MHz	2 pc tray	SPAKMC332AVFC20
				36 pc tray	MC68332AVFC20
		-40 to +125 °C	16 MHz	2 pc tray	SPAKMC332AMFC16
				36 pc tray	MC68332AMFC16
			20 MHz	2 pc tray	SPAKMC332AMFC20
				36 pc tray	MC68332AMFC20

## 1.3 Pin Assignments

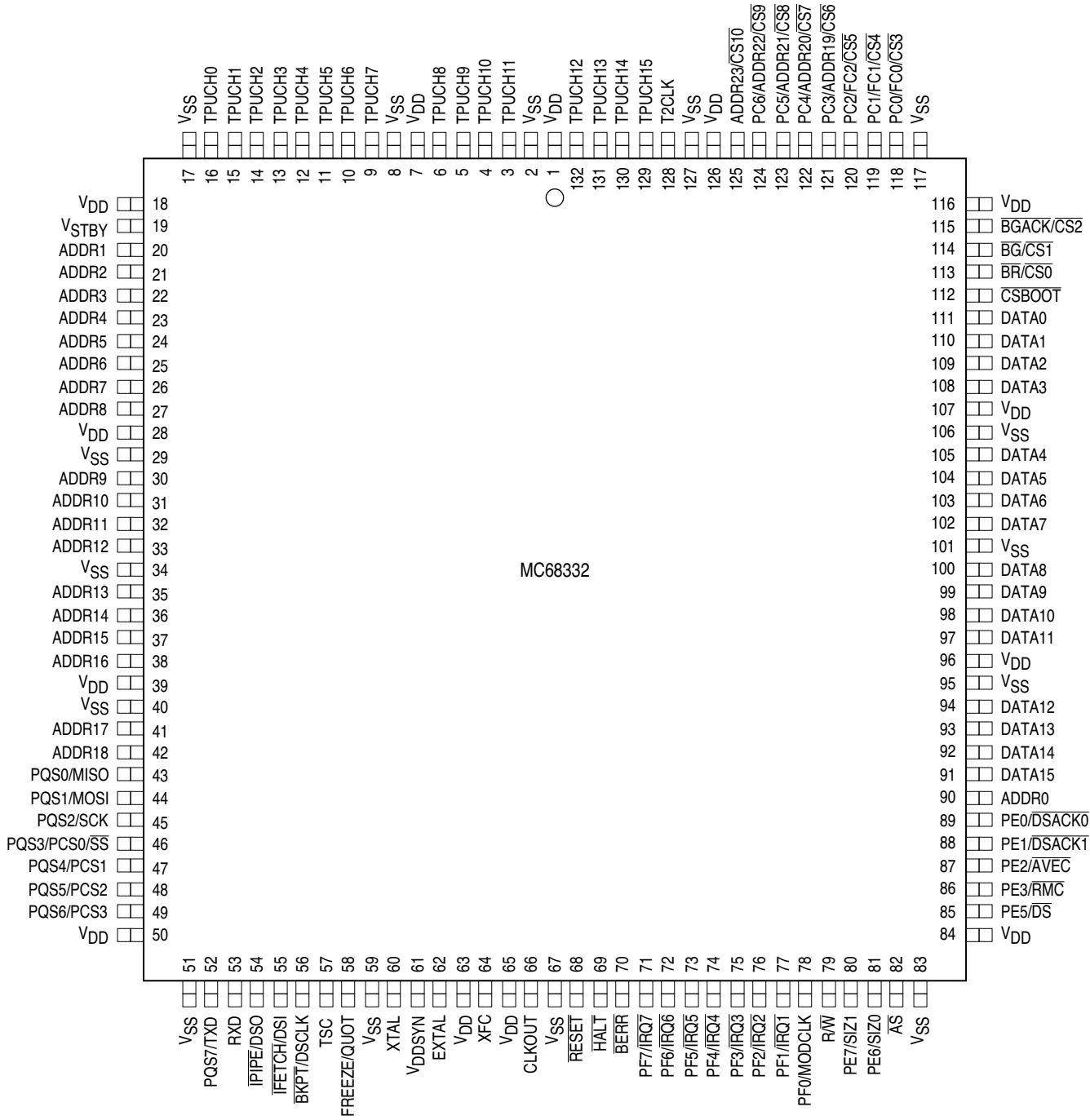
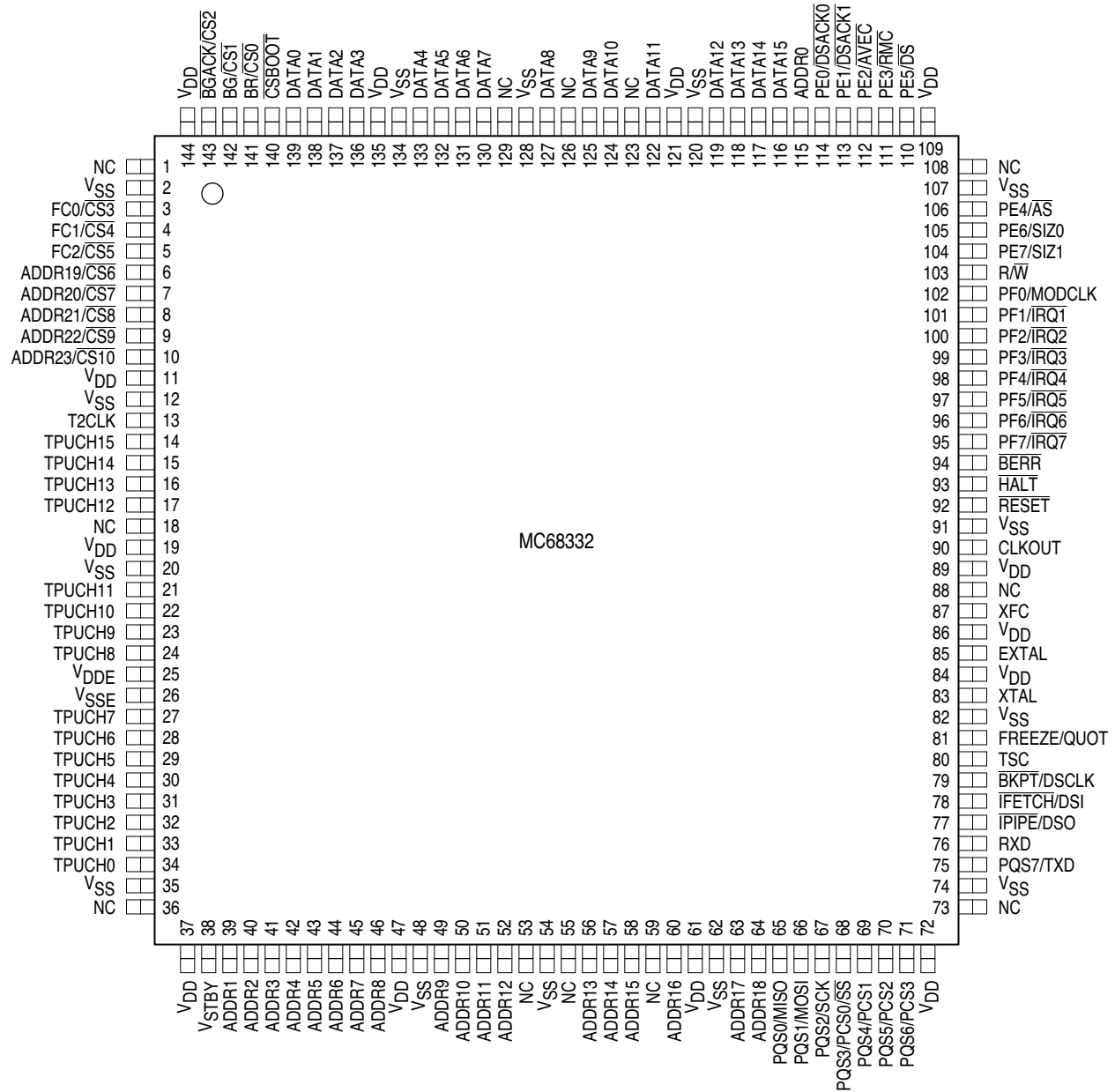


Figure 2 MC68332 132-Pin QFP Pin Assignments

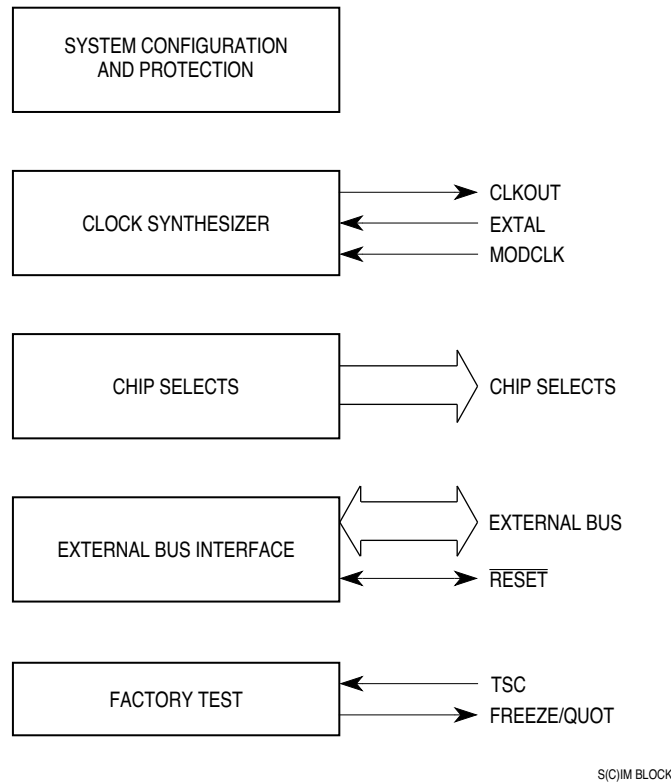


332 144-PIN QFP

Figure 3 MC68332 144-Pin QFP Pin Assignments

## 3 System Integration Module

The MCU system integration module (SIM) consists of five functional blocks that control system start-up, initialization, configuration, and external bus.



**Figure 5 SIM Block Diagram**

### 3.1 Overview

The system configuration and protection block controls MCU configuration and operating mode. The block also provides bus and software watchdog monitors.

The system clock generates clock signals used by the SIM, other IMB modules, and external devices. In addition, a periodic interrupt generator supports execution of time-critical control routines.

The external bus interface handles the transfer of information between IMB modules and external address space.

The chip-select block provides eleven general-purpose chip-select signals and a boot ROM chip select signal. Both general-purpose and boot ROM chip-select signals have associated base address registers and option registers.

The system test block incorporates hardware necessary for testing the MCU. It is used to perform factory tests, and its use in normal applications is not supported.

The SIM control register address map occupies 128 bytes. Unused registers within the 128-byte address space return zeros when read. The “Access” column in the SIM address map below indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the SIMCR.

Table 7 SIM Address Map (Continued)

Access	Address	15	8 7	0
S	\$YFFA56	CHIP-SELECT OPTION 2 (CSOR2)		
S	\$YFFA58	CHIP-SELECT BASE 3 (CSBAR3)		
S	\$YFFA5A	CHIP-SELECT OPTION 3 (CSOR3)		
S	\$YFFA5C	CHIP-SELECT BASE 4 (CSBAR4)		
S	\$YFFA5E	CHIP-SELECT OPTION 4 (CSOR4)		
S	\$YFFA60	CHIP-SELECT BASE 5 (CSBAR5)		
S	\$YFFA62	CHIP-SELECT OPTION 5 (CSOR5)		
S	\$YFFA64	CHIP-SELECT BASE 6 (CSBAR6)		
S	\$YFFA66	CHIP-SELECT OPTION 6 (CSOR6)		
S	\$YFFA68	CHIP-SELECT BASE 7 (CSBAR7)		
S	\$YFFA6A	CHIP-SELECT OPTION 7 (CSOR7)		
S	\$YFFA6C	CHIP-SELECT BASE 8 (CSBAR8)		
S	\$YFFA6E	CHIP-SELECT OPTION 8 (CSOR8)		
S	\$YFFA70	CHIP-SELECT BASE 9 (CSBAR9)		
S	\$YFFA72	CHIP-SELECT OPTION 9 (CSOR9)		
S	\$YFFA74	CHIP-SELECT BASE 10 (CSBAR10)		
S	\$YFFA76	CHIP-SELECT OPTION 10 (CSOR10)		
	\$YFFA78	NOT USED		NOT USED
	\$YFFA7A	NOT USED		NOT USED
	\$YFFA7C	NOT USED		NOT USED
	\$YFFA7E	NOT USED		NOT USED

Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMCR.

### 3.2 System Configuration and Protection

This functional block provides configuration control for the entire MCU. It also performs interrupt arbitration, bus monitoring, and system test functions. MCU system protection includes a bus monitor, a HALT monitor, a spurious interrupt monitor, and a software watchdog timer. These functions have been made integral to the microcontroller to reduce the number of external components in a complete control system.

## 3.2.7 Periodic Interrupt Timer

The periodic interrupt timer (PIT) generates interrupts of specified priorities at specified intervals. Timing for the PIT is provided by a programmable prescaler driven by the system clock.

### PICR — Periodic Interrupt Control Register

\$YFFA22

15	14	13	12	11	10	8	7	0
0	0	0	0	0	PIRQL		PIV	

RESET:

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

This register contains information concerning periodic interrupt priority and vectoring. Bits [10:0] can be read or written at any time. Bits [15:11] are unimplemented and always return zero.

### PIRQL[2:0] — Periodic Interrupt Request Level

The following table shows what interrupt request level is asserted when a periodic interrupt is generated. If a PIT interrupt and an external  $\overline{IRQ}$  signal of the same priority occur simultaneously, the PIT interrupt is serviced first. The periodic timer continues to run when the interrupt is disabled.

PIRQL	Interrupt Request Level
000	Periodic Interrupt Disabled
001	Interrupt Request Level 1
010	Interrupt Request Level 2
011	Interrupt Request Level 3
100	Interrupt Request Level 4
101	Interrupt Request Level 5
110	Interrupt Request Level 6
111	Interrupt Request Level 7

### PIV[7:0] — Periodic Interrupt Vector

The bits of this field contain the vector generated in response to an interrupt from the periodic timer. When the SIM responds, the periodic interrupt vector is placed on the bus.

### PITR — Periodic Interrupt Timer Register

\$YFFA24

15	14	13	12	11	10	9	8	7	0
0	0	0	0	0	0	0	PTP	PITM	

RESET:

0 0 0 0 0 0 0 MODCLK 0 0 0 0 0 0 0 0

The PITR contains the count value for the periodic timer. A zero value turns off the periodic timer. This register can be read or written at any time.

### PTP — Periodic Timer Prescaler Control

0 = Periodic timer clock not prescaled

1 = Periodic timer clock prescaled by a value of 512

The reset state of PTP is the complement of the state of the MODCLK signal during reset.

### PITM[7:0] — Periodic Interrupt Timing Modulus Field

This is an 8-bit timing modulus. The period of the timer can be calculated as follows:

$$\text{PIT Period} = [(PITM)(\text{Prescaler})(4)]/\text{EXTAL}$$

where

PIT Period = Periodic interrupt timer period

PITM = Periodic interrupt timer register modulus (PITR[7:0])

EXTAL Frequency = Crystal frequency

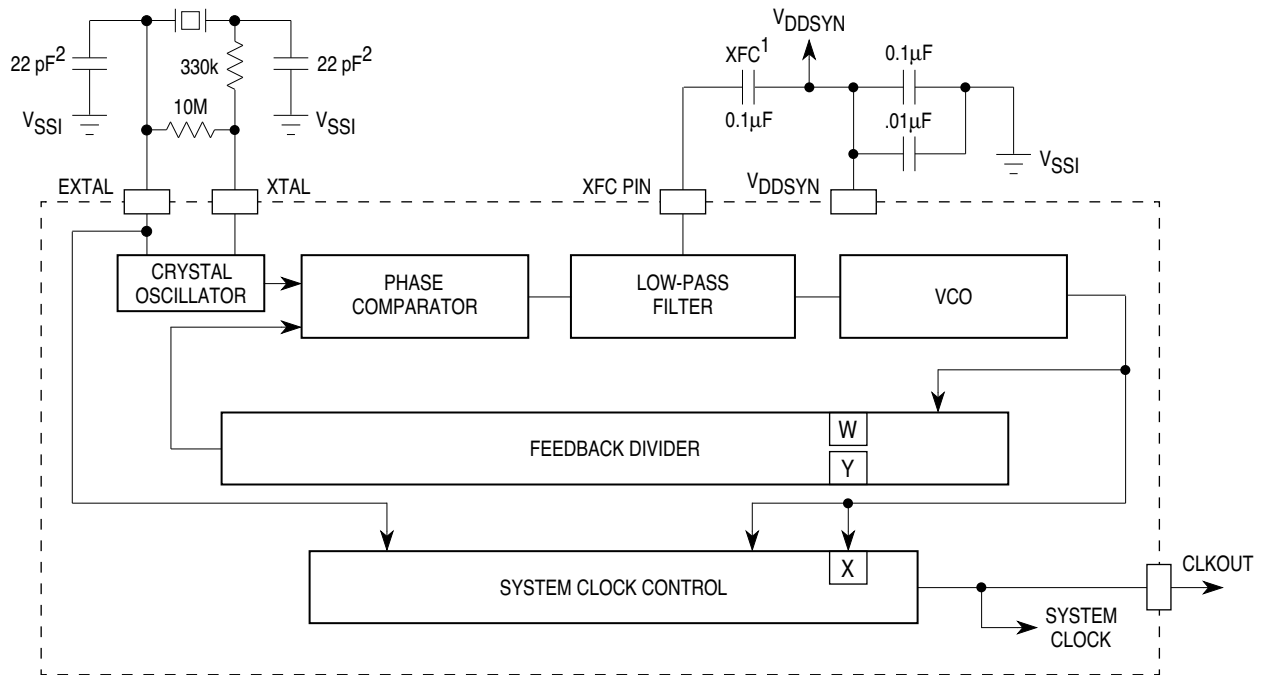
Prescale = 512 or 1 depending on the state of the PTP bit in the PITR

## 3.3 System Clock

The system clock in the SIM provides timing signals for the IMB modules and for an external peripheral bus. Because MCU operation is fully static, register and memory contents are not affected when the clock rate changes. System hardware and software support changes in the clock rate during operation.

The system clock signal can be generated in three ways. An internal phase-locked loop can synthesize the clock from an internal or external frequency source, or the clock signal can be input from an external source.

Following is a block diagram of the clock submodule.



1. MUST BE LOW-LEAKAGE CAPACITOR (INSULATION RESISTANCE 30,000 MΩ OR GREATER).
2. RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768-kHz CRYSTAL. SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

SYS CLOCK  
BLOCK 32KHZ

Figure 7 System Clock Block Diagram

### 3.3.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines the clock source. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from either a crystal oscillator or an external reference input. Clock synthesizer control register SYNCR determines operating frequency and various modes of operation. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. When the synthesizer is disabled, SYNCR control bits have no effect.

A reference crystal must be connected between the EXTAL and XTAL pins to use the internal oscillator. Use of a 32.768-kHz crystal is recommended. These crystals are inexpensive and readily available. If an external reference signal or an external system clock signal is applied through the EXTAL pin, the XTAL pin must be left floating. External reference signal frequency must be less than or equal to maximum specified reference frequency. External system clock signal frequency must be less than or equal to maximum specified system clock frequency.



**Table 9 CPU32 Address Space Encoding**

FC2	FC1	FC0	Address Space
0	0	0	Reserved
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

### 3.4.3 Address Bus

Address bus signals ADDR[23:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while  $\overline{AS}$  is asserted.

### 3.4.4 Address Strobe

$\overline{AS}$  is a timing signal that indicates the validity of an address on the address bus and the validity of many control signals. It is asserted one-half clock after the beginning of a bus cycle.

### 3.4.5 Data Bus

Data bus signals DATA[15:0] make up a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer 8 or 16 bits of data in one bus cycle. During a read cycle, the data is latched by the MCU on the last falling edge of the clock for that bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size. The MCU places the data on the data bus one-half clock cycle after  $\overline{AS}$  is asserted in a write cycle.

### 3.4.6 Data Strobe

Data strobe ( $\overline{DS}$ ) is a timing signal. For a read cycle, the MCU asserts  $\overline{DS}$  to signal an external device to place data on the bus.  $\overline{DS}$  is asserted at the same time as  $\overline{AS}$  during a read cycle. For a write cycle,  $\overline{DS}$  signals an external device that data on the bus is valid. The MCU asserts  $\overline{DS}$  one full clock cycle after the assertion of  $\overline{AS}$  during a write cycle.

### 3.4.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals ( $\overline{DSACK1}$  and  $\overline{DSACK0}$ ). During a read cycle, the signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can end. These signals also indicate to the MCU the size of the port for the bus cycle just completed. (Refer to 3.4.9 Dynamic Bus Sizing.)

The bus error ( $\overline{BERR}$ ) signal is also a bus cycle termination indicator and can be used in the absence of  $\overline{DSACK1}$  and  $\overline{DSACK0}$  to indicate a bus error condition. It can also be asserted in conjunction with these signals, provided it meets the appropriate timing requirements. The internal bus monitor can be used to generate the  $\overline{BERR}$  signal for internal and internal-to-external transfers. When  $\overline{BERR}$  and  $\overline{HALT}$  are asserted simultaneously, the CPU takes a bus error exception.

Autovector signal ( $\overline{AVEC}$ ) can terminate external  $\overline{IRQ}$  pin interrupt acknowledge cycles.  $\overline{AVEC}$  indicates that the MCU will internally generate a vector number to locate an interrupt handler routine. If it is continuously asserted, autovectors will be generated for all external interrupt requests.  $\overline{AVEC}$  is ignored during all other bus cycles.

## 3.4.8 Data Transfer Mechanism

The MCU architecture supports byte, word, and long-word operands, allowing access to 8- and 16-bit data ports through the use of asynchronous cycles controlled by the data transfer and size acknowledge inputs ( $\overline{\text{DSACK1}}$  and  $\overline{\text{DSACK0}}$ ).

## 3.4.9 Dynamic Bus Sizing

The MCU dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8- and 16-bit ports. During an operand transfer cycle, the slave device signals its port size and indicates completion of the bus cycle to the MCU through the use of the  $\overline{\text{DSACK0}}$  and  $\overline{\text{DSACK1}}$  inputs, as shown in the following table.

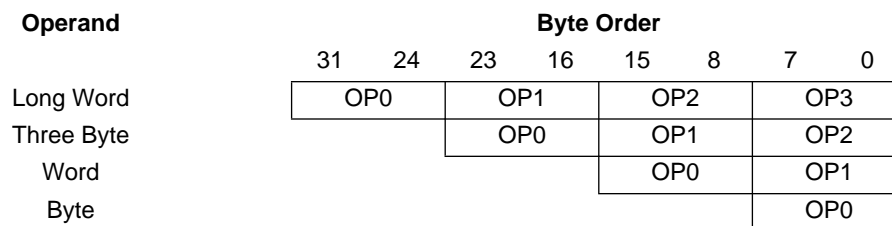
**Table 10 Effect of  $\overline{\text{DSACK}}$  Signals**

$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	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

For example, if the MCU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the 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  $\overline{\text{DSACK0}}$  and  $\overline{\text{DSACK1}}$  signals to indicate the port width. For instance, a 16-bit device always returns  $\overline{\text{DSACK0}} = 1$  and  $\overline{\text{DSACK1}} = 0$  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 the following figure. OP0 is the most significant byte of a long-word operand, and OP3 is the least significant byte. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.



**Figure 8 Operand Byte Order**

## 3.4.10 Operand Alignment

The data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the size and address outputs. SIZ1 and SIZ0 indicate the remaining number of bytes to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

### 3.7.3 Reset Timing

The  $\overline{\text{RESET}}$  input must be asserted for a specified minimum period in order for reset to occur. External  $\overline{\text{RESET}}$  assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor time-out period) in order to protect write cycles from being aborted by reset. While  $\overline{\text{RESET}}$  is asserted, SIM pins are either in a disabled high-impedance state or are driven to their inactive states.

When an external device asserts  $\overline{\text{RESET}}$  for the proper period, reset control logic clocks the signal into an internal latch. The control logic drives the  $\overline{\text{RESET}}$  pin low for an additional 512 CLKOUT cycles after it detects that the  $\overline{\text{RESET}}$  signal is no longer being externally driven, to guarantee this length of reset to the entire system.

If an internal source asserts a reset signal, the reset control logic asserts  $\overline{\text{RESET}}$  for a minimum of 512 cycles. If the reset signal is still asserted at the end of 512 cycles, the control logic continues to assert  $\overline{\text{RESET}}$  until the internal reset signal is negated.

After 512 cycles have elapsed, the reset input pin goes to an inactive, high-impedance state for ten cycles. At the end of this 10-cycle period, the reset input is tested. When the input is at logic level one, reset exception processing begins. If, however, the reset input is at logic level zero, the reset control logic drives the pin low for another 512 cycles. At the end of this period, the pin again goes to high-impedance state for ten cycles, then it is tested again. The process repeats until  $\overline{\text{RESET}}$  is released.

### 3.7.4 Power-On Reset

When the SIM clock synthesizer is used to generate the system clock, power-on reset involves special circumstances related to application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin  $V_{\text{DDSYN}}$  in order for the MCU to operate. The following discussion assumes that  $V_{\text{DDSYN}}$  is applied before and during reset. This minimizes crystal start-up time. When  $V_{\text{DDSYN}}$  is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design.  $V_{\text{DD}}$  ramp-up time also affects pin state during reset.

During power-on reset, an internal circuit in the SIM drives the internal (IMB) and external reset lines. The circuit releases the internal reset line as  $V_{\text{DD}}$  ramps up to the minimum specified value, and SIM pins are initialized. When  $V_{\text{DD}}$  reaches the specified minimum value, the clock synthesizer VCO begins operation. Clock frequency ramps up to the specified limp mode frequency. The external  $\overline{\text{RESET}}$  line remains asserted until the clock synthesizer PLL locks and 512 CLKOUT cycles elapse.

The SIM clock synthesizer provides clock signals to the other MCU modules. After the clock is running and the internal reset signal is asserted for four clock cycles, these modules reset.  $V_{\text{DD}}$  ramp time and VCO frequency ramp time determine how long these four cycles take. Worst case is approximately 15 milliseconds. During this period, module port pins may be in an indeterminate state. While input-only pins can be put in a known state by means of external pull-up resistors, external logic on input/output or output-only pins must condition the lines during this time. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

### 3.7.5 Use of Three State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in an inactive, high-impedance state. The signal must remain asserted for ten clock cycles in order for drivers to change state. There are certain constraints on use of TSC during power-on reset:

When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the ten cycles take. Worst case is approximately 20 milliseconds from TSC assertion.

When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as ten clock pulses have been applied to the EXTAL pin.

### 5.1.2 Input Capture/Input Transition Counter (ITC)

Any channel of the TPU can capture the value of a specified TCR upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the CPU. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, then cease channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and the number of channels within the block. The generation of links depends on the mode of operation. In addition, after each transition or specified number of transitions, one byte of the parameter RAM (at an address specified by channel parameter) can be incremented and used as a flag to notify another channel of a transition.

### 5.1.3 Output Compare (OC)

The output compare function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time.
2. At a programmable delay time from a user-specified time.
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} * \text{Ratio}$$

where Ratio is a parameter supplied by the user.

This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins.

### 5.1.4 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin.

### 5.1.5 Synchronized Pulse-Width Modulation (SPWM)

The TPU generates a PWM waveform in which the CPU can change the period and/or high time at any time. When synchronized to a time function on a second channel, the synchronized PWM low-to-high transitions have a time relationship to transitions on the second channel.

### 5.1.6 Period Measurement with Additional Transition Detect (PMA)

This function and the following function are used primarily in toothed-wheel speed-sensing applications, such as monitoring rotational speed of an engine. The period measurement with additional transition detect function allows for a special-purpose 23-bit period measurement. It can detect the occurrence of an additional transition (caused by an extra tooth on the sensed wheel) indicated by a period measurement that is less than a programmable ratio of the previous period measurement.

Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to \$FFFF. Alternatively, a byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to \$FFFF once the next additional transition is detected.

lation parameter. From 1 to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (1 to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter. By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels.

## 5.1.11 Quadrature Decode (QDEC)

The quadrature decode function uses two channels to decode a pair of out-of-phase signals in order to present the CPU with directional information and a position value. It is particularly suitable for use with slotted encoders employed in motor control. The function derives full resolution from the encoder signals and provides a 16-bit position counter with rollover/under indication via an interrupt.

The counter in parameter RAM is updated when a valid transition is detected on either one of the two inputs. The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of servicing the transition. The user can read or write the counter at any time. The counter is free running, overflowing to \$0000 or underflowing to \$FFFF depending on direction. The QDEC function also provides a time stamp referenced to TCR1 for every valid signal edge and the ability for the host CPU to obtain the latest TCR1 value. This feature allows position interpolation by the host CPU between counts at very slow count rates.

## 5.2 MC68332G Time Functions

The following paragraphs describe factory-programmed time functions implemented in the motion-control microcode ROM. A complete description of the functions is beyond the scope of this summary. Refer to *Using the TPU Function Library and TPU Emulation Mode* (TPUPN00/D) for more information about specific functions.

### 5.2.1 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in PRAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

### 5.2.2 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

## 5.4 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the TPU module address map. Channel parameters are organized as 128 16-bit words. However, only 100 words are actually implemented. The parameter RAM address map shows how parameter words are organized in memory.

**Table 23 TPU Parameter RAM Address Map**

Channel Number	Base Address	Parameter Address							
		0	1	2	3	4	5	6	7
0	\$YFFFF##	00	02	04	06	08	0A	—	—
1	\$YFFFF##	10	12	14	16	18	1A	—	—
2	\$YFFFF##	20	22	24	26	28	2A	—	—
3	\$YFFFF##	30	32	34	36	38	3A	—	—
4	\$YFFFF##	40	42	44	46	48	4A	—	—
5	\$YFFFF##	50	52	54	56	58	5A	—	—
6	\$YFFFF##	60	62	64	66	68	6A	—	—
7	\$YFFFF##	70	72	74	76	78	7A	—	—
8	\$YFFFF##	80	82	84	86	88	8A	—	—
9	\$YFFFF##	90	92	94	96	98	9A	—	—
10	\$YFFFF##	A0	A2	A4	A6	A8	AA	—	—
11	\$YFFFF##	B0	B2	B4	B6	B8	BA	—	—
12	\$YFFFF##	C0	C2	C4	C6	C8	CA	—	—
13	\$YFFFF##	D0	D2	D4	D6	D8	DA	—	—
14	\$YFFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	\$YFFFF##	F0	F2	F4	F6	F8	FA	FC	FE

—= Not Implemented

Y = M111, where M represents the logic state of the MM bit in the SIMCR.

## 5.5 TPU Registers

The TPU memory map contains three groups of registers:

- System Configuration Registers
- Channel Control and Status Registers
- Development Support and Test Verification Registers

### 5.5.1 System Configuration Registers

**TPUMCR** — TPU Module Configuration Register

**\$YFFE00**

15	14	13	12	11	10	9	8	7	6	5	4	3	0
STOP	TCR1P	TCR2P	EMU	T2CG	STF	SUPV	PSCK	0	0			IARB	

RESET:

0 0 0 0 0 0 0 0 1 0 0 0 0 0

**STOP** — Stop Bit

- 0 = TPU operating normally
- 1 = Internal clocks shut down



## 6.3 Pin Function

The following table is a summary of the functions of the QSM pins when they are not configured for general-purpose I/O. The QSM data direction register (DDRQS) designates each pin except RXD as an input or output.

	Pin	Mode	Pin Function
QSPI Pins	MISO	Master	Serial Data Input to QSPI
		Slave	Serial Data Output from QSPI
	MOSI	Master	Serial Data Output from QSPI
		Slave	Serial Data Input to QSPI
	SCK	Master	Clock Output from QSPI
		Slave	Clock Input to QSPI
SCI Pins	PCS0/SS	Master	Input: Assertion Causes Mode Fault Output: Selects Peripherals
		Slave	Input: Selects the QSPI
	PCS[3:1]	Master	Output: Selects Peripherals
		Slave	None
	TXD	Transmit	Serial Data Output from SCI
	RXD	Receive	Serial Data Input to SCI

## 6.4 QSM Registers

QSM registers are divided into four categories: QSM global registers, QSM pin control registers, QSPI submodule registers, and SCI submodule registers. The QSPI and SCI registers are defined in separate sections below. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The module mapping bit of the SIM configuration register (SIMCR) defines the most significant bit (ADDR23) of the address, shown in each register figure as Y (Y = \$7 or \$F). This bit, concatenated with the rest of the address given, forms the absolute address of each register. Refer to the SIM section of this technical summary for more information about how the state of MM affects the system.

### 6.4.1 Global Registers

The QSM global registers contain system parameters used by both the QSPI and the SCI submodules. These registers contain the bits and fields used to configure the QSM.

#### QSMCR — QSM Configuration Register

\$YFFC00

15	14	13	12	11	10	9	8	7	6	5	4	3	0
STOP	FRZ1	FRZ0	0	0	0	0	0	SUPV	0	0	0	IARB	

RESET:

0 0 0 0 0 0 0 0 1 0 0 0 0 0

The QSMCR contains parameters for the QSM/CPU/intermodule bus (IMB) interface.

#### STOP — Stop Enable

0 = Normal QSM clock operation

1 = QSM clock operation stopped

STOP places the QSM in a low-power state by disabling the system clock in most parts of the module. The QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable. However, writes to RAM or any register are guaranteed to be valid while STOP is asserted. STOP can be negated by the CPU and by reset.

The system software must stop each submodule before asserting STOP to avoid complications at re-start and to avoid data corruption. The SCI submodule receiver and transmitter should be disabled, and the operation should be verified for completion before asserting STOP. The QSPI submodule should be stopped by asserting the HALT bit in SPCR3 and by asserting STOP after the HALTA flag is set.

## FRZ1 — Freeze 1

- 0 = Ignore the FREEZE signal on the IMB
- 1 = Halt the QSPI (on a transfer boundary)

FRZ1 determines what action is taken by the QSPI when the FREEZE signal of the IMB is asserted. FREEZE is asserted whenever the CPU enters the background mode.

## FRZ0 — Freeze 0

Reserved

## Bits [12:8] — Not Implemented

## SUPV — Supervisor/Unrestricted

- 0 = User access
- 1 = Supervisor access

SUPV defines the assignable QSM registers as either supervisor-only data space or unrestricted data space.

## IARB — Interrupt Arbitration Identification Number

The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value. Refer to **3.8 Interrupts** for more information.

## QTEST — QSM Test Register

**\$YFFC02**

QTEST is used during factory testing of the QSM. Accesses to QTEST must be made while the MCU is in test mode.

## QILR — QSM Interrupt Levels Register

**\$YFFC04**

15	14	13	11	10	8	7	0
0	0	ILQSPI			ILSCI		QIVR

RESET:

0 0 0 0 0 0 0 0

QILR determines the priority level of interrupts requested by the QSM and the vector used when an interrupt is acknowledged.

## ILQSPI — Interrupt Level for QSPI

ILQSPI determines the priority of QSPI interrupts. This field must be given a value between \$0 (interrupts disabled) to \$7 (highest priority).

## ILSCI — Interrupt Level of SCI

ILSCI determines the priority of SCI interrupts. This field must be given a value between \$0 (interrupts disabled) to \$7 (highest priority).

If ILQSPI and ILSCI are the same nonzero value, and both submodules simultaneously request interrupt service, QSPI has priority.



## QIVR — QSM Interrupt Vector Register

\$YFFC05

15	8	7									0				
QILR								INTV							
RESET:															
								0	0	0	0	1	1	1	1

At reset, QIVR is initialized to \$0F, which corresponds to the uninitialized interrupt vector in the exception table. This vector is selected until QIVR is written. A user-defined vector (\$40–\$FF) should be written to QIVR during QSM initialization.

After initialization, QIVR determines which two vectors in the exception vector table are to be used for QSM interrupts. The QSPI and SCI submodules have separate interrupt vectors adjacent to each other. Both submodules use the same interrupt vector with the least significant bit (LSB) determined by the submodule causing the interrupt.

The value of INTV0 used during an interrupt-acknowledge cycle is supplied by the QSM. During an interrupt-acknowledge cycle, INTV[7:1] are driven on DATA[7:1] IMB lines. DATA0 is negated for an SCI interrupt and asserted for a QSPI interrupt. Writes to INTV0 have no meaning or effect. Reads of INTV0 return a value of one.

### 6.4.2 Pin Control Registers

The QSM uses nine pins, eight of which form a parallel port (PORTQS) on the MCU. Although these pins are used by the serial subsystems, any pin can alternately be assigned as general-purpose I/O on a pin-by-pin basis.

Pins used for general-purpose I/O must not be assigned to the QSPI by register PQSPAR. To avoid driving incorrect data, the first byte to be output must be written before DDRQS is configured. DDRQS must then be written to determine the direction of data flow and to output the value contained in register PORTQS. Subsequent data for output is written to PORTQS.

## PORTQS — Port QS Data Register

\$YFFC14

15	8	7	6	5	4	3	2	1	0
NOT USED		PQS7	PQS6	PQS5	PQS4	PQS3	PQS2	PQS1	PQS0
RESET:									
		0	0	0	0	0	0	0	0

PORTQS latches I/O data. Writes drive pins defined as outputs. Reads return data present on the pins. To avoid driving undefined data, first write a byte to PORTQS, then configure DDRQS.

## PQSPAR — PORT QS Pin Assignment Register

\$YFFC16

## DDRQS — PORT QS Data Direction Register

\$YFFC17

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PQSPA6	PQSPA5	PQSPA4	PQSPA3	0	PQSPA1	PQSPA0	DDQS7	DDQS6	DDQS5	DDQS4	DDQS3	DDQS2	DDQS1	DDQS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clearing a bit in the PQSPAR assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. The PQSPAR does not affect operation of the SCI.

SCK baud rate:

$$\text{SCK Baud Rate} = \text{System Clock}/(2\text{SPBR})$$

or

$$\text{SPBR} = \text{System Clock}/(2\text{SCK})(\text{Baud Rate Desired})$$

where SPBR equals {2, 3, 4,..., 255}

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, baud rate is initialized to one eighth of the system clock frequency.

## SPCR1 — QSPI Control Register 1

\$YFFC1A

15	14	8	7	0
SPE	DSCKL		DTL	

RESET:

0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0

SPCR1 contains parameters for configuring the QSPI before it is enabled. The CPU can read and write this register, but the QSM has read access only, except for SPE, which is automatically cleared by the QSPI after completing all serial transfers, or when a mode fault occurs.

### SPE — QSPI Enable

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.

### DSCKL — Delay before SCK

When the DSCK bit in command RAM is set, this field determines the length of delay from PCS valid to SCK transition. PCS can be any of the four peripheral chip-select pins. The following equation determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = [\text{DSCKL}/\text{System Clock}]$$

where DSCKL equals {1, 2, 3,..., 127}.

When the DSCK value of a queue entry equals zero, then DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half SCK period.

### DTL — Length of Delay after Transfer

When the DT bit in command RAM is set, this field determines the length of delay after serial transfer. The following equation is used to calculate the delay:

$$\text{Delay after Transfer} = [(32\text{DTL})/\text{System Clock}]$$

where DTL equals {1, 2, 3,..., 255}.

A zero value for DTL causes a delay-after-transfer value of 8192/System Clock.

If DT equals zero, a standard delay is inserted.

$$\text{Standard Delay after Transfer} = [17/\text{System Clock}]$$

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

## SPCR2 — QSPI Control Register 2

\$YFFC1C

15	14	13	12	11	8	7	6	5	4	3	0
SPIFIE	WREN	WRT0	0	ENDQP	0	0	0	0	0	NEWQP	

RESET:

0 0 0 0 0 0 0 0 0 0 0 0

SPCR2 contains QSPI configuration parameters. The CPU can read and write this register; the QSM has read access only. Writes to SPCR2 are buffered. A write to SPCR2 that changes a bit value while the QSPI is operating is ineffective on the current serial transfer, but becomes effective on the next serial transfer. Reads of SPCR2 return the current value of the register, not of the buffer.

### SPIFIE — SPI Finished Interrupt Enable

0 = QSPI interrupts disabled

1 = QSPI interrupts enabled

SPIFIE enables the QSPI to generate a CPU interrupt upon assertion of the status flag SPIF.

### WREN — Wrap Enable

0 = Wraparound mode disabled

1 = Wraparound mode enabled

WREN enables or disables wraparound mode.

### WRT0 — Wrap To

When wraparound mode is enabled, after the end of queue has been reached, WRT0 determines which address the QSPI executes.

### Bit 12 — Not Implemented

### ENDQP — Ending Queue Pointer

This field contains the last QSPI queue address.

### Bits [7:4] — Not Implemented

### NEWQP — New Queue Pointer Value

This field contains the first QSPI queue address.

## SPCR3 — QSPI Control Register 3

\$YFFC1E

15	14	13	12	11	10	9	8	7	0
0	0	0	0	0	LOOPQ	HMIE	HALT		SPSR

RESET:

0 0 0 0 0 0 0 0

SPCR3 contains QSPI configuration parameters. The CPU can read and write SPCR3, but the QSM has read-only access.

### Bits [15:11] — Not Implemented

### LOOPQ — QSPI Loop Mode

0 = Feedback path disabled

1 = Feedback path enabled

LOOPQ controls feedback on the data serializer for testing.

### HMIE — HALTA and MODF Interrupt Enable

0 = HALTA and MODF interrupts disabled

1 = HALTA and MODF interrupts enabled

HMIE controls CPU interrupts caused by the HALTA status flag or the MODF status flag in SPSR.

HALT — Halt

0 = Halt not enabled

1 = Halt enabled

When HALT is asserted, the QSPI stops on a queue boundary. It is in a defined state from which it can later be restarted.

**SPSR** — QSPI Status Register

**\$YFFC1F**

15	8	7	6	5	4	3	0
SPCR3							
SPIF							
MODF							
HALTA							
0							
CPTQP							
RESET:							
0 0 0 0 0 0 0 0							

SPSR contains QSPI status information. Only the QSPI can assert the bits in this register. The CPU reads this register to obtain status information and writes it to clear status flags.

SPIF — QSPI Finished Flag

0 = QSPI not finished

1 = QSPI finished

SPIF is set after execution of the command at the address in ENDQP.

MODF — Mode Fault Flag

0 = Normal operation

1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ( $\overline{SS}$  input taken low).

The QSPI asserts MODF when the QSPI is the serial master (MSTR = 1) and the  $\overline{SS}$  input pin is negated by an external driver.

HALTA — Halt Acknowledge Flag

0 = QSPI not halted

1 = QSPI halted

HALTA is asserted when the QSPI halts in response to CPU assertion of HALT.

Bit 4 — Not Implemented

CPTQP — Completed Queue Pointer

CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value (\$0) or a pointer to the last command completed in the previous queue.

## 6.5.3 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that is used by both the QSPI and the CPU. The RAM is divided into three segments: receive data, transmit data, and command control data. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external peripheral. Command control data is used to perform the transfer.

Refer to the following illustration of the organization of the RAM.

## 6.6 SCI Submodule

The SCI submodule is used to communicate with external devices through an asynchronous serial bus. The SCI is fully compatible with the SCI systems found on other Motorola MCUs, such as the M68HC11 and M68HC05 Families.

### 6.6.1 SCI Pins

There are two unidirectional pins associated with the SCI. The SCI controls the transmit data (TXD) pin when enabled, whereas the receive data (RXD) pin remains a dedicated input pin to the SCI. TXD is available as a general-purpose I/O pin when the SCI transmitter is disabled. When used for I/O, TXD can be configured either as input or output, as determined by QSM register DDRQS.

The following table shows SCI pins and their functions.

Pin Names	Mnemonics	Mode	Function
Receive Data	RXD	Receiver Disabled Receiver Enabled	Not Used Serial Data Input to SCI
Transmit Data	TXD	Transmitter Disabled Transmitter Enabled	General-Purpose I/O Serial Data Output from SCI

### 6.6.2 SCI Registers

The SCI programming model includes QSM global and pin control registers, and four SCI registers. There are two SCI control registers, one status register, and one data register. All registers can be read or written at any time by the CPU.

Changing the value of SCI control bits during a transfer operation may disrupt operation. Before changing register values, allow the transmitter to complete the current transfer, then disable the receiver and transmitter. Status flags in the SCSR may be cleared at any time.

#### SCCR0 — SCI Control Register 0

**\$YFFC08**

15	14	13	12	0
0	0	0	SCBR	

RESET:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

SCCR0 contains a baud rate selection parameter. Baud rate must be set before the SCI is enabled. The CPU can read and write this register at any time.

Bits [15:13] — Not Implemented

#### SCBR — Baud Rate

SCI baud rate is programmed by writing a 13-bit value to BR. The baud rate is derived from the MCU system clock by a modulus counter.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receiver sampling clock with a frequency 16 times that of the expected baud rate of the incoming data. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period. Receiver sampling rate is always 16 times the frequency of the SCI baud rate, which is calculated as follows:

$$\text{SCI Baud Rate} = \text{System Clock} / (32\text{SCBR})$$

or

$$\text{SCBR} = \text{System Clock} / (32\text{SCK})(\text{Baud Rate desired})$$

where SCBR is in the range {1, 2, 3, ..., 8191}