



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](#)"

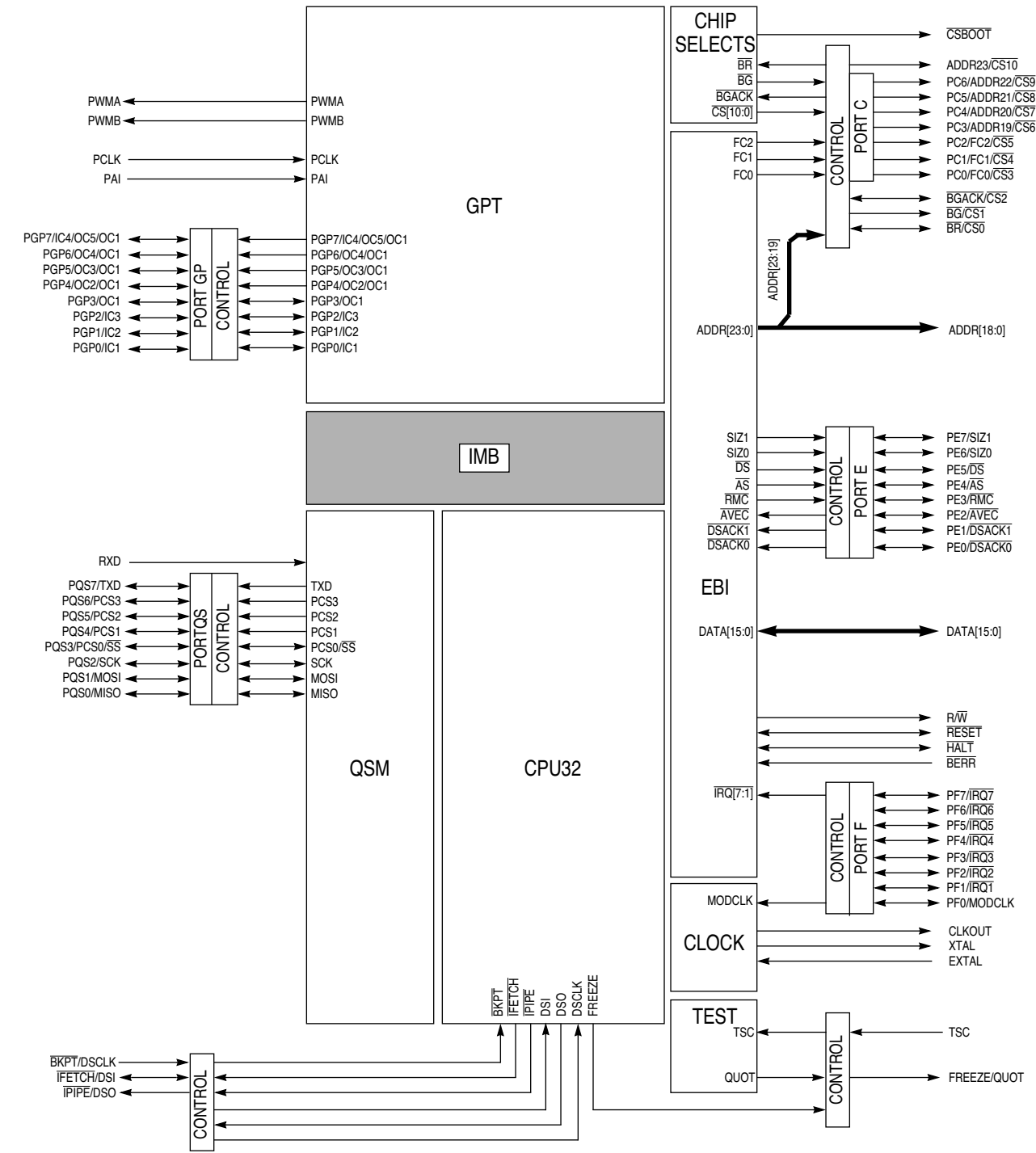
Details

Product Status	Not For New Designs
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	18
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	-
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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68331cag20

Table 1 Ordering Information

Package Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
132-Pin PQFP	-40 to +85 °C	16 MHz	2 pc tray	SPAKMC331CFC16
			36 pc tray	MC68331CFC16
		20 MHz	2 pc tray	SPAKMC331CFC20
			36 pc tray	MC68331CFC20
	-40 to +105 °C	16 MHz	2 pc tray	SPAKMC331VFC16
			36 pc tray	MC68331VFC16
		20 MHz	2 pc tray	SPAKMC331VFC20
			36 pc tray	MC68331VFC20
	-40 to +125 °C	16 MHz	2 pc tray	SPAKMC331MFC16
			36 pc tray	MC68331MFC16
		20 MHz	2 pc tray	SPAKMC331MFC20
			36 pc tray	MC68331MFC20
144-Pin QFP	-40 to +85 °C	16 MHz	2 pc tray	SPAKMC331CFV16
			44 pc tray	MC68331CFV16
		20 MHz	2 pc tray	SPAKMC331CFV20
			44 pc tray	MC68331CFV20
	-40 to +105 °C	16 MHz	2 pc tray	SPAKMC331VFV16
			44 pc tray	MC68331VFV16
		20 MHz	2 pc tray	SPAKMC331VFV20
			44 pc tray	MC68331VFV20
	-40 to +125 °C	16 MHz	2 pc tray	SPAKMC331MFV16
			44 pc tray	MC68331MFV16
		20 MHz	2 pc tray	SPAKMC331MFV20
			44 pc tray	MC68331MFV20

1.2 Block Diagram



331 BLOCK

Figure 1 MCU Block Diagram

2 Signal Descriptions

2.1 Pin Characteristics

The following table shows MCU pins and their characteristics. All inputs detect CMOS logic levels. All inputs can be put in a high-impedance state, but the method of doing this differs depending upon pin function. Refer to **Table 4**, for a description of output drivers. An entry in the discrete I/O column of **Table 2** indicates that a pin has an alternate I/O function. The port designation is given when it applies. Refer to the MCU Block Diagram for information about port organization.

Table 2 MCU Pin Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/CS10/ECLK	A	Y	N	O	—
ADDR[22:19]/CS[9:6]	A	Y	N	O	PC[6:3]
ADDR[18:0]	A	Y	N	—	—
\overline{AS}	B	Y	N	I/O	PE5
AVEC	B	Y	N	I/O	PE2
BERR	B	Y	N	—	—
BG/CS1	B	—	—	—	—
BGACK/CS2	B	Y	N	—	—
BKPT/DSCLK	—	Y	Y	—	—
BR/CS0	B	Y	N	—	—
CLKOUT	A	—	—	—	—
CSBOOT	B	—	—	—	—
DATA[15:0] ¹	Aw	Y	N	—	—
\overline{DS}	B	Y	N	I/O	PE4
DSACK1	B	Y	N	I/O	PE1
DSACK0	B	Y	N	I/O	PE0
DSI/IFETCH	A	Y	Y	—	—
DSO/IPIPE	A	—	—	—	—
EXTAL ²	—	—	Special	—	—
FC[2:0]/CS[5:3]	A	Y	N	O	PC[2:0]
FREEZE/QUOT	A	—	—	—	—
IC4/OC5	A	Y	Y	I/O	GP4
IC[3:1]	A	Y	Y	I/O	GP[7:5]
HALT	Bo	Y	N	—	—
IRQ[7:1]	B	Y	Y	I/O	PF[7:1]
MISO	Bo	Y	Y	I/O	PQS0
MODCLK ¹	B	Y	N	I/O	PF0
MOSI	Bo	Y	Y	I/O	PQS1
OC[4:1]	A	Y	Y	I/O	GP[3:0]
PAI ³	—	Y	Y	I	—
PCLK ³	—	Y	Y	I	—
PCS0/SS	Bo	Y	Y	I/O	PQS3
PCS[3:1]	Bo	Y	Y	I/O	PQS[6:4]
PWMA, PWMB	A	—	—	O	—
R/W	A	Y	N	—	—
RESET	Bo	Y	Y	—	—
RMC	B	Y	N	I/O	PE3

Table 2 MCU Pin Characteristics (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
RXD	—	N	N	—	—
SCK	Bo	Y	Y	I/O	PQS2
SIZ[1:0]	B	Y	N	I/O	PE[7:6]
TSC	—	Y	Y	—	—
TXD	Bo	Y	Y	I/O	PQS7
XFC ²	—	—	—	Special	—
XTAL ²	—	—	—	Special	—

NOTES:

1. DATA[15:0] are synchronized during reset only. MODCLK is synchronized only when used as an input port pin.
2. EXTAL, XFC, and XTAL are clock reference connections.
3. PAI and PCLK can be used for discrete input, but are not part of an I/O port.

2.2 MCU Power Connections

Table 3 MCU Power Connections

V _{DDSYN}	Clock Synthesizer Power
V _{SSE} /V _{DDE}	External Periphery Power (Source and Drain)
V _{SSI} /V _{DDI}	Internal Module Power (Source and Drain)

2.3 MCU Driver Types

Table 4 MCU Driver Types

Type	I/O	Description
A	O	Output-only signals that are always driven; no external pull-up required
Aw	O	Type A output with weak P-channel pull-up during reset
B	O	Three-state output that includes circuitry to pull up output before high impedance is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while the pin is in the high-impedance state.
Bo	O	Type B output that can be operated in an open-drain mode

2.4 Signal Characteristics

Table 5 MCU Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SIM	Bus	—
AS	SIM	Output	0
AVEC	SIM	Input	0
BERR	SIM	Input	0
BG	SIM	Output	0
BGACK	SIM	Input	0
BKPT	CPU32	Input	0
BR	SIM	Input	0
CLKOUT	SIM	Output	—
CS[10:0]	SIM	Output	0
CSBOOT	SIM	Output	0
DATA[15:0]	SIM	Bus	—
DS	SIM	Output	0

Table 7 SIM Address Map (Continued)

Access	Address	15	8 7	0
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.

3.3.2 Clock Synthesizer Operation

A voltage controlled oscillator (VCO) generates the system clock signal. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is a reference signal, either from the internal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is identical to reference frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must re-lock. Lock status is shown by the SLOCK bit in SYNCR.

The MCU does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

The low-pass filter requires an external low-leakage capacitor, typically 0.1 μ F, connected between the XFC and V_{DDSYN} pins.

V_{DDSYN} is used to power the clock circuits. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. Use a quiet power supply as the V_{DDSYN} source, since PLL stability depends on the VCO, which uses this supply. Place adequate external bypass capacitors as close as possible to the V_{DDSYN} pin to ensure stable operating frequency.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. SYNCR can be read only when the processor is operating at the supervisor privilege level.

The SYNCR X bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting X doubles clock speed without changing VCO speed. There is no VCO relock delay. The SYNCR W bit controls a 3-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When either W or Y value changes, there is a VCO relock delay.

Clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE} [4(Y + 1)(2^{2W + X})]$$

In order for the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified for the MCU. The VCO frequency is twice the system clock frequency if X = 1 or four times the system clock frequency if X = 0.

The reset state of SYNCR (\$3F00) produces a modulus-64 count.

3.3.3 Clock Control

The clock control circuits determine system clock frequency and clock operation under special circumstances, such as following loss of synthesizer reference or during low-power operation. Clock source is determined by the logic state of the MODCLK pin during reset.

SYNCR —Clock Synthesizer Control Register

\$YFFA04

15	14	13					8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	U	U	0	0	0

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 ($\overline{DSACK1}$ and $\overline{DSACK0}$).

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 $\overline{DSACK0}$ and $\overline{DSACK1}$ inputs, as shown in the following table.

Table 10 Effect of \overline{DSACK} Signals

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

3.7 Resets

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The system integration module determines whether a reset is valid, asserts control signals, performs basic system configuration based on hardware mode-select inputs, then passes control to the CPU.

Reset occurs when an active low logic level on the $\overline{\text{RESET}}$ pin is clocked into the SIM. Resets are gated by the CLKOUT signal. Asynchronous resets are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. Synchronous resets are timed to occur at the end of bus cycles. If there is no clock when $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked in order to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ is asserted.

Reset is the highest-priority CPU32 exception. Any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

3.7.1 SIM Reset Mode Selection

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. The following table is a summary of reset mode selection options.

Table 18 Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2
DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{\text{CS6}}$ $\overline{\text{CS[7:6]}}$ $\overline{\text{CS[8:6]}}$ $\overline{\text{CS[9:6]}}$ $\overline{\text{CS[10:6]}}$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, AVEC, DS, AS, SIZ[1:0]	PORTE
DATA9	$\overline{\text{IRQ[7:1]}}$ MODCLK	PORTF
DATA11	Test Mode Disabled	Test Mode Enabled
MODCLK	VCO = System Clock	EXTAL = System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled

3.7.2 Functions of Pins for Other Modules During Reset

Generally, pins associated with modules other than the SIM default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers. Refer to individual module sections in this manual for more information. The following table is a summary of module pin function out of reset.

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_{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 10 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 10 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 10 clock pulses have been applied to the EXTAL pin.

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.

3.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the central processing unit, the system integration module, and a device or module requesting interrupt service.

The CPU32 provides for eight levels of interrupt priority (0–7), seven automatic interrupt vectors and 200 assignable interrupt vector. All interrupts with priorities less than 7 can be masked by the interrupt priority (IP) field in the status register. The CPU32 handles interrupts as a type of asynchronous exception.

Interrupt recognition is based on the states of interrupt request signals $\overline{IRQ}[7:1]$ and the IP mask value. Each of the signals corresponds to an interrupt priority. $\overline{IRQ1}$ has the lowest priority, and $\overline{IRQ7}$ has the highest priority.

The IP field consists of three bits. Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value (except for $\overline{IRQ7}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

Interrupt request signals can be asserted by external devices or by microcontroller modules. Request lines are connected internally by means of a wired NOR — simultaneous requests of differing priority can be made. Internal assertion of an interrupt request signal does not affect the logic state of the corresponding MCU pin.

External interrupt requests are routed to the CPU via the external bus interface and SIM interrupt control logic. The CPU treats external interrupt requests as though they come from the SIM.

External $\overline{IRQ}[6:1]$ are active-low level-sensitive inputs. External $\overline{IRQ7}$ is an active-low transition-sensitive input. $\overline{IRQ7}$ requires both an edge and a voltage level for validity.

$\overline{IRQ}[6:1]$ are maskable. $\overline{IRQ7}$ is nonmaskable. The $\overline{IRQ7}$ input is transition-sensitive in order to prevent redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{IRQ7}$ is asserted, and each time the priority mask changes from %111 to a lower number while $\overline{IRQ7}$ is asserted.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis. To be valid, a request signal must be asserted for at least two consecutive clock periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority exceptions is complete.

The CPU32 does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request of equal or lower priority than the current IP mask value is made, the CPU does not recognize the occurrence of the request in any way.

3.8.1 Interrupt Acknowledge and Arbitration

Interrupt acknowledge bus cycles are generated during exception processing. When the CPU detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it performs a CPU space read from address \$FFFF : [IP] : 1.

The CPU space read cycle performs two functions: it places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value also serves two purposes: it is latched into the CCR IP field in order to mask lower-priority interrupts during exception processing, and it is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle pertains to them.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the interrupt acknowledge cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by means of serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest priority). A value of %0000 in an IARB field causes the CPU to process a spurious interrupt exception when an interrupt from that module is recognized.

Because the EBI manages external interrupt requests, the SIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000. Initialization software must assign different IARB values in order to implement an arbitration scheme.

Each module must have a unique IARB value. When two or more IARB fields have the same nonzero value, the CPU interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source requests service. This point is important for two reasons: the CPU interrupt acknowledge cycle is not driven on the external bus unless the SIM wins contention, and failure to contend causes an interrupt acknowledge bus cycle to be terminated by a bus error, which causes a spurious interrupt exception to be taken.

When arbitration is complete, the dominant module must place an interrupt vector number on the data bus and terminate the bus cycle. In the case of an external interrupt request, because the interrupt acknowledge cycle is transferred to the external bus, an external device must decode the mask value and respond with a vector number, then generate bus cycle termination signals. If the device does not respond in time, a spurious interrupt exception is taken.

The periodic interrupt timer (PIT) in the SIM can generate internal interrupt requests of specific priority at predetermined intervals. By hardware convention, PIT interrupts are serviced before external interrupt service requests of the same priority. Refer to **3.2.7 Periodic Interrupt Timer** for more information.

4 Central Processor Unit

Based on the powerful MC68020, the CPU32 processing module provides enhanced system performance and also uses the extensive software base for the Motorola M68000 family.

4.1 Overview

The CPU32 is fully object code compatible with the M68000 Family, which excels at processing calculation-intensive algorithms and supporting high-level languages. The CPU32 supports all of the MC68010 and most of the MC68020 enhancements, such as virtual memory support, loop mode operation, instruction pipeline, and 32-bit mathematical operations. Powerful addressing modes provide compatibility with existing software programs and increase the efficiency of high-level language compilers. Special instructions, such as table lookup and interpolate and low-power stop, support the specific requirements of controller applications. Also included is the background debugging mode, an alternate operating mode that suspends normal operation and allows the CPU to accept debugging commands from the development system.

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction set is optimized for high performance. The eight 32-bit general-purpose data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long word) operations. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

Use of high-level languages is increasing as controller applications become more complex and control programs become larger. High-level languages aid rapid development of software, with less error, and are readily portable. The CPU32 instruction set supports high-level languages.

4.2 Programming Model

The CPU32 has sixteen 32-bit general registers, a 32-bit program counter, one 32-bit supervisor stack pointer, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register.

The programming model of the CPU32 consists of a user model and supervisor model, corresponding to the user and supervisor privilege levels. Some instructions available at the supervisor level are not available at the user level, allowing the supervisor to protect system resources from uncontrolled access. Bit S in the status register determines the privilege level.

The user programming model remains unchanged from previous M68000 Family microprocessors. Application software written to run at the nonprivileged user level migrates without modification to the CPU32 from any M68000 platform. The move from SR instruction, however, is privileged in the CPU32. It is not privileged in the M68000.

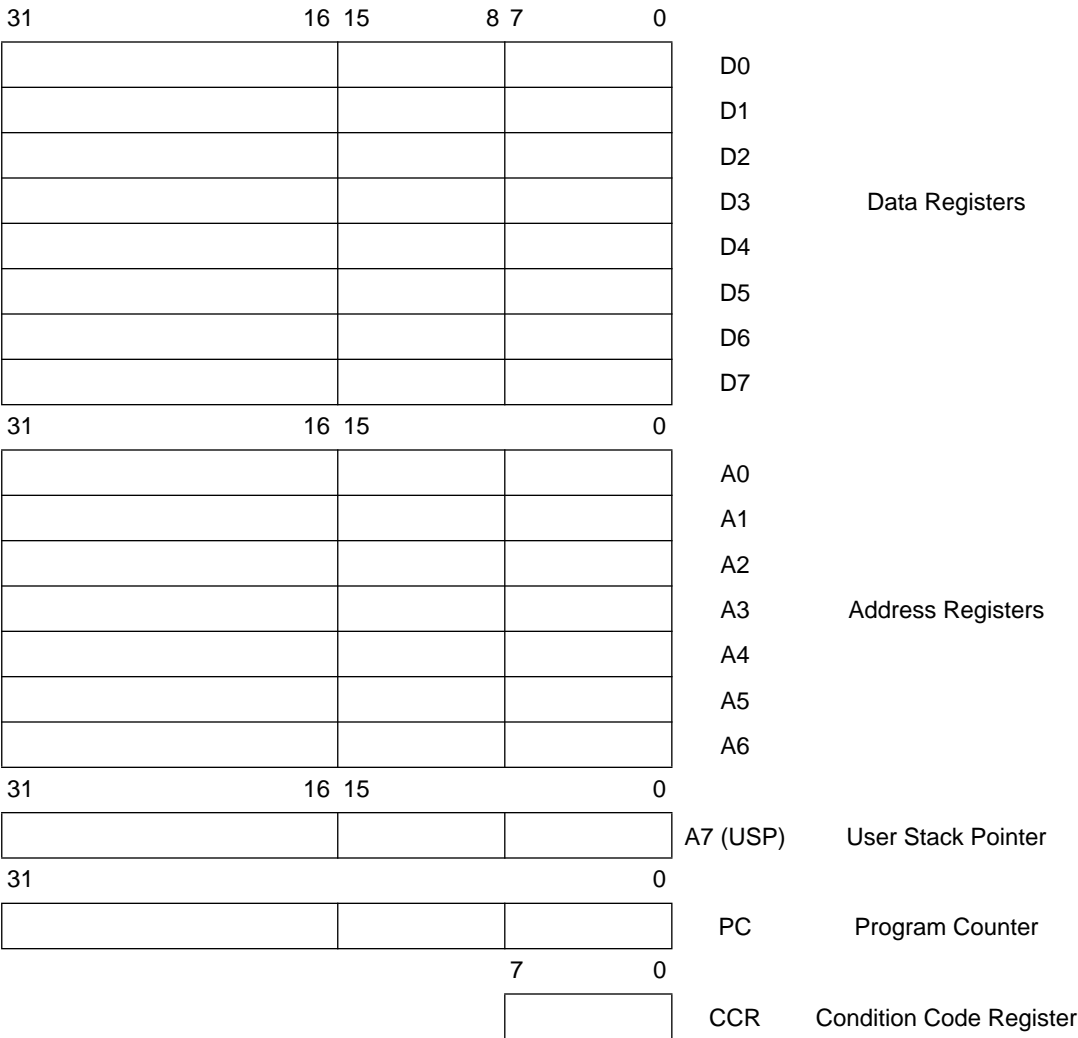


Figure 10 User Programming Model

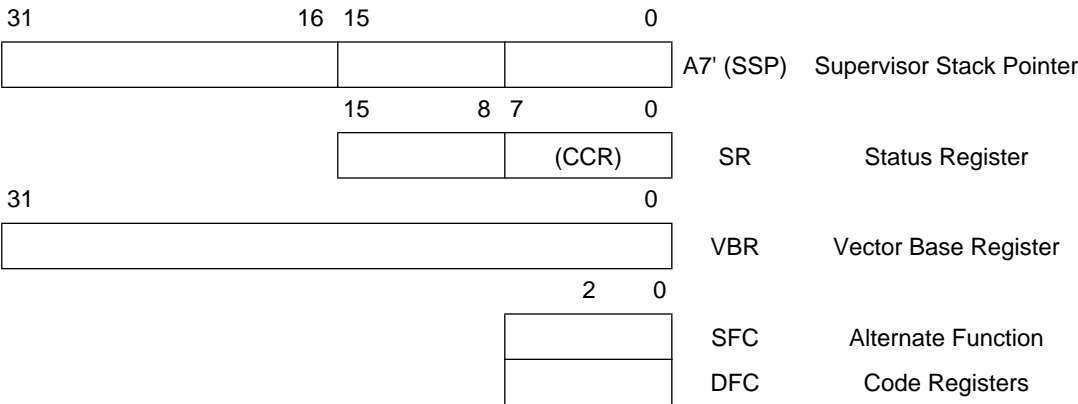
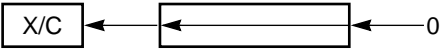
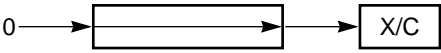


Figure 11 Supervisor Programming Model Supplement

Table 20 Instruction Set Summary (Continued)

Instruction	Syntax	Operand Size	Operation
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
CMP2	<ea>, Rn	8, 16, 32	Lower bound ≤ Rn ≤ Upper bound, CCR shows result
DBcc	Dn, label	16	If condition false, then Dn – 1 ⇒ PC; if Dn ≠ (– 1), then PC + d ⇒ PC
DIVS/DIVU	<ea>, Dn	32/16 ⇒ 16 : 16	Destination / Source ⇒ Destination (signed or unsigned)
DIVSL/DIVUL	<ea>, Dr : Dq <ea>, Dq <ea>, Dr : Dq	64/32 ⇒ 32 : 32 32/32 ⇒ 32 32/32 ⇒ 32 : 32	Destination / Source ⇒ Destination (signed or unsigned)
EOR	Dn, <ea>	8, 16, 32	Source ⊕ Destination ⇒ Destination
EORI	# <data>, <ea>	8, 16, 32	Data ⊕ Destination ⇒ Destination
EORI to CCR	# <data>, CCR	8	Source ⊕ CCR ⇒ CCR
EORI to SR ¹	# <data>, SR	16	Source ⊕ SR ⇒ SR
EXG	Rn, Rn	32	Rn ⇒ Rn
EXT	Dn Dn	8 ⇒ 16 16 ⇒ 32	Sign extended Destination ⇒ Destination
EXTB	Dn	8 ⇒ 32	Sign extended Destination ⇒ Destination
ILLEGAL	none	none	SSP – 2 ⇒ SSP; vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); Illegal instruction vector address ⇒ PC
JMP	<ea>	none	Destination ⇒ PC
JSR	<ea>	none	SP – 4 ⇒ SP; PC ⇒ (SP); destination ⇒ PC
LEA	<ea>, An	32	<ea> ⇒ An
LINK	An, # d	16, 32	SP – 4 ⇒ SP, An ⇒ (SP); SP ⇒ An, SP + d ⇒ SP
LPSTOP ¹	# <data>	16	Data ⇒ SR; interrupt mask ⇒ EBI; STOP
LSL	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
MOVE	<ea>, <ea>	8, 16, 32	Source ⇒ Destination
MOVEA	<ea>, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEA ¹	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVE from CCR	CCR, <ea>	16	CCR ⇒ Destination
MOVE to CCR	<ea>, CCR	16	Source ⇒ CCR
MOVE from SR ¹	SR, <ea>	16	SR ⇒ Destination
MOVE to SR ¹	<ea>, SR	16	Source ⇒ SR
MOVE USP ¹	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVEC ¹	Rc, Rn Rn, Rc	32 32	Rc ⇒ Rn Rn ⇒ Rc
MOVEM	list, <ea> <ea>, list	16, 32 16, 32 ⇒ 32	Listed registers ⇒ Destination Source ⇒ Listed registers

5.3 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.

5.3.1 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 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 restart 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

QIVR determines which two vector numbers in the exception vector table are to be used for QSM interrupts. The seven MSB of a user-defined vector number (\$40–\$FF) must be written into the INTV field during initialization. The value of INTV0 is supplied by the QSM when an interrupt service request is acknowledged.

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.

At reset, QIVR is initialized to \$0F, which corresponds to the uninitialized interrupt vector in the exception table.

5.3.2 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.

DDRQS determines whether pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function.

Table 24 Effect of DDRQS on QSM Pin Function

QSM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQ0	0	Serial Data Input to QSPI
			1	Disables Data Input
	Slave		0	Disables Data Output
			1	Serial Data Output from QSPI
MOSI	Master	DDQ1	0	Disables Data Output
			1	Serial Data Output from QSPI
	Slave		0	Serial Data Input to QSPI
			1	Disables Data Input
SCK ¹	Master	DDQ2	0	Disables Clock Output
			1	Clock Output from QSPI
	Slave		0	Clock Input to QSPI
			1	Disables Clock Input
PCS0/ $\overline{\text{SS}}$	Master	DDQ3	0	Assertion Causes Mode Fault
			1	Chip-Select Output
	Slave		0	QSPI Slave Select Input
			1	Disables Select Input
PCS[3:1]	Master	DDQ[4:6]]	0	Disables Chip-Select Output
			1	Chip-Select Output
	Slave		0	Inactive
			1	Inactive
TXD ²	Transmit	DDQ7	X	Serial Data Output from SCI
RXD	Receive	None	NA	Serial Data Input to SCI

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output.

5.4 QSPI Submodule

The QSPI submodule communicates with external devices through a synchronous serial bus. The QSPI is fully compatible with the serial peripheral interface (SPI) systems found on other Motorola products. A block diagram of the QSPI is shown below.

SPBR — Serial Clock Baud Rate

The QSPI uses a modulus counter to derive SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into the SPBR field. The following equation determines the 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.

PT — Parity Type

- 0 = Even parity
- 1 = Odd parity

When parity is enabled, PT determines whether parity is even or odd for both the receiver and the transmitter.

PE — Parity Enable

- 0 = SCI parity disabled
- 1 = SCI parity enabled

PE determines whether parity is enabled or disabled for both the receiver and the transmitter. If the received parity bit is not correct, the SCI sets the PF error flag in SCSR.

When PE is set, the most significant bit (MSB) of the data field is used for the parity function, which results in either seven or eight bits of user data, depending on the condition of M bit. The following table lists the available choices.

M	PE	Result
0	0	8 Data Bits
0	1	7 Data Bits, 1 Parity Bit
1	0	9 Data Bits
1	1	8 Data Bits, 1 Parity Bit

M — Mode Select

- 0 = SCI frame: 1 start bit, 8 data bits, 1 stop bit (10 bits total)
- 1 = SCI frame: 1 start bit, 9 data bits, 1 stop bit (11 bits total)

WAKE — Wakeup by Address Mark

- 0 = SCI receiver awakened by idle-line detection
- 1 = SCI receiver awakened by address mark (last bit set)

TIE — Transmit Interrupt Enable

- 0 = SCI TDRE interrupts inhibited
- 1 = SCI TDRE interrupts enabled

TCIE — Transmit Complete Interrupt Enable

- 0 = SCI TC interrupts inhibited
- 1 = SCI TC interrupts enabled

RIE — Receiver Interrupt Enable

- 0 = SCI RDRF interrupt inhibited
- 1 = SCI RDRF interrupt enabled

ILIE — Idle-Line Interrupt Enable

- 0 = SCI IDLE interrupts inhibited
- 1 = SCI IDLE interrupts enabled

TE — Transmitter Enable

- 0 = SCI transmitter disabled (TXD pin may be used as I/O)
- 1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter)

The transmitter retains control of the TXD pin until completion of any character transfer in progress when TE is cleared.

RE — Receiver Enable

- 0 = SCI receiver disabled (status bits inhibited)
- 1 = SCI receiver enabled

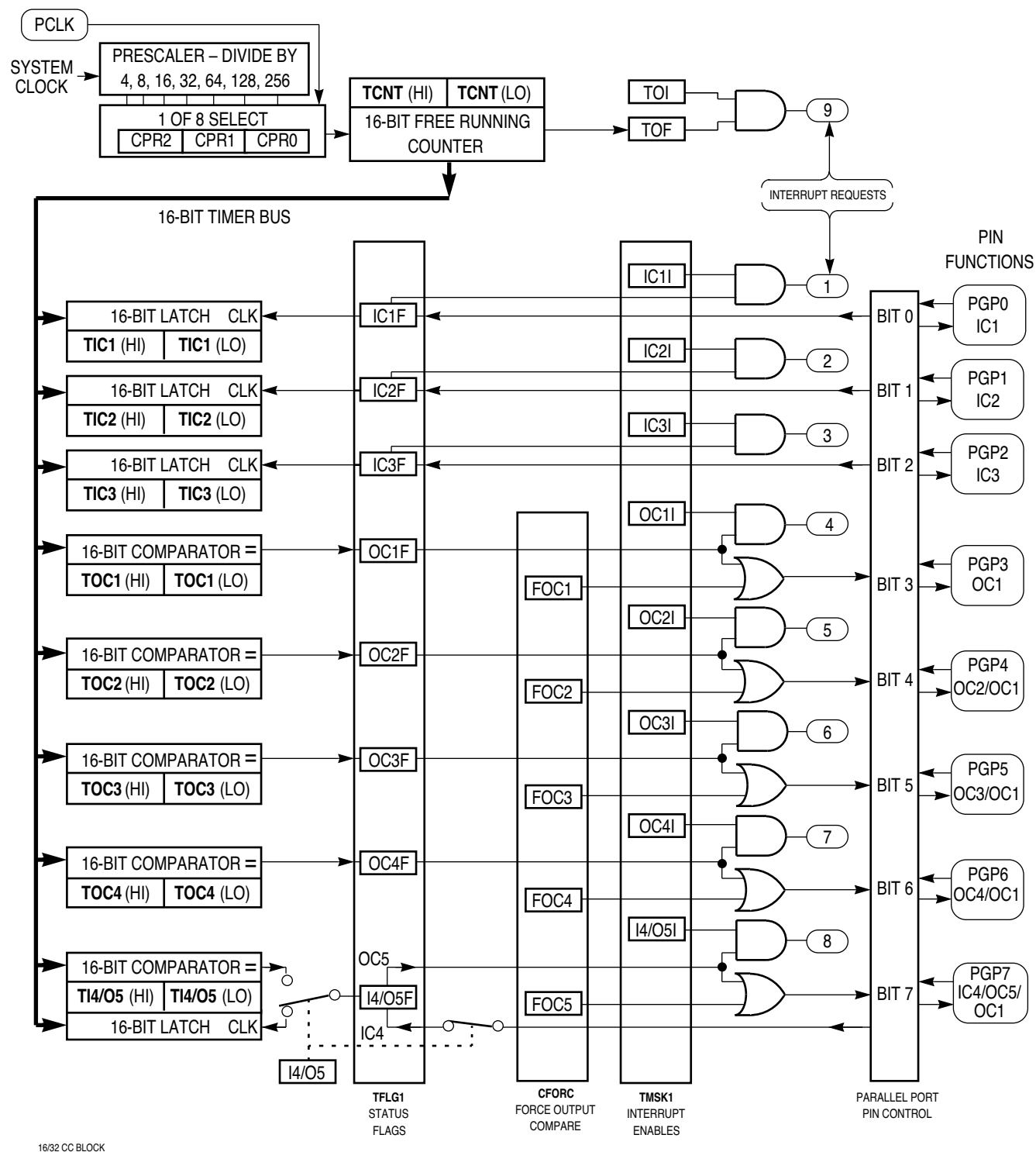


Figure 16 GPT Timer Block Diagram

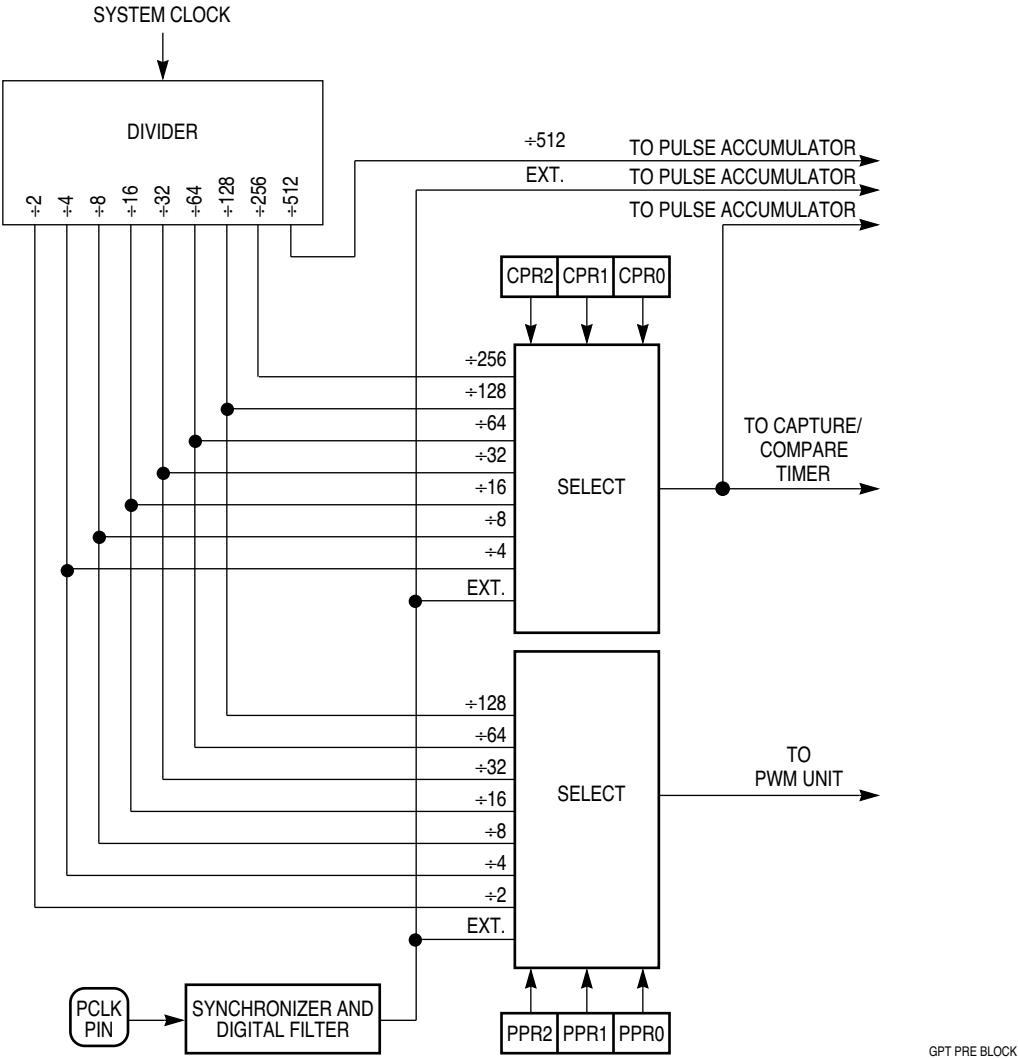


Figure 17 Prescaler Block Diagram