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	16.78MHz
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)	3V ~ 3.6V
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/mc68lk332gcpv16

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

Table 1 Ordering Information (Continued)

Package Type	TPU Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
144-Pin QFP	Motion Control	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332GCFV16
				44 pc tray	MC68332GCFVV16
			20 MHz	2 pc tray	SPAKMC332GCFV20
				44 pc tray	MC68332GCFV20
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332GVFV16
				44 pc tray	MC68332GVFV16
			20 MHz	2 pc tray	SPAKMC332GVFV20
				44 pc tray	MC68332GVFV20
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332GMFV16
				44 pc tray	MC68332GMFV16
			20 MHz	2 pc tray	SPAKMC332GMFV20
				44 pc tray	MC68332GMFVV20
	Standard	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332CFV16
				44 pc tray	MC68332CFV16
			20 MHz	2 pc tray	SPAKMC332CFVV20
				44 pc tray	MC68332CFV20
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332VFV16
				44 pc tray	MC68332VFV16
			20 MHz	2 pc tray	SPAKMC332VFV20
				44 pc tray	MC68332VFV20
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332MFV16
				44 pc tray	MC68332MFV16
			20 MHz	2 pc tray	SPAKMC332MFV20
				44 pc tray	MC68332MFV20
	Std w/enhanced PPWA	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332ACFV16
				44 pc tray	MC68332ACFV16
			20 MHz	2 pc tray	SPAKMC332ACFV20
				44 pc tray	MC68332ACFV20
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332AVFV16
				44 pc tray	MC68332AVFV16
			20 MHz	2 pc tray	SPAKMC332AVFC20
				44 pc tray	MC68332AVFV20
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332AMFV16
				44 pc tray	MC68332AMFV16
			20 MHz	2 pc tray	SPAKMC332AMFV20
				44 pc tray	MC68332AMFV20

1.1 Features

- Central Processing Unit (CPU32)
  - — 32-Bit Architecture
  - — Virtual Memory Implementation
  - — Table Lookup and Interpolate Instruction
  - — Improved Exception Handling for Controller Applications
  - — High-Level Language Support
  - — Background Debugging Mode
  - — Fully Static Operation
- System Integration Module (SIM)
  - — External Bus Support
  - — Programmable Chip-Select Outputs
  - — System Protection Logic
  - — Watchdog Timer, Clock Monitor, and Bus Monitor
  - — Two 8-Bit Dual Function Input/Output Ports
  - — One 7-Bit Dual Function Output Port
  - — Phase-Locked Loop (PLL) Clock System
- Time Processor Unit (TPU)
  - — Dedicated Microengine Operating Independently of CPU32
  - — 16 Independent, Programmable Channels and Pins
  - — Any Channel can Perform any Time Function
  - — Two Timer Count Registers with Programmable Prescalers
  - — Selectable Channel Priority Levels
- Queued Serial Module (QSM)
  - — Enhanced Serial Communication Interface
  - — Queued Serial Peripheral Interface
  - — One 8-Bit Dual Function Port
- Static RAM Module with TPU Emulation Capability (TPURAM)
  - — 2-Kbytes of Static RAM
  - — May be Used as Normal RAM or TPU Microcode Emulation RAM

1.2 Block Diagram

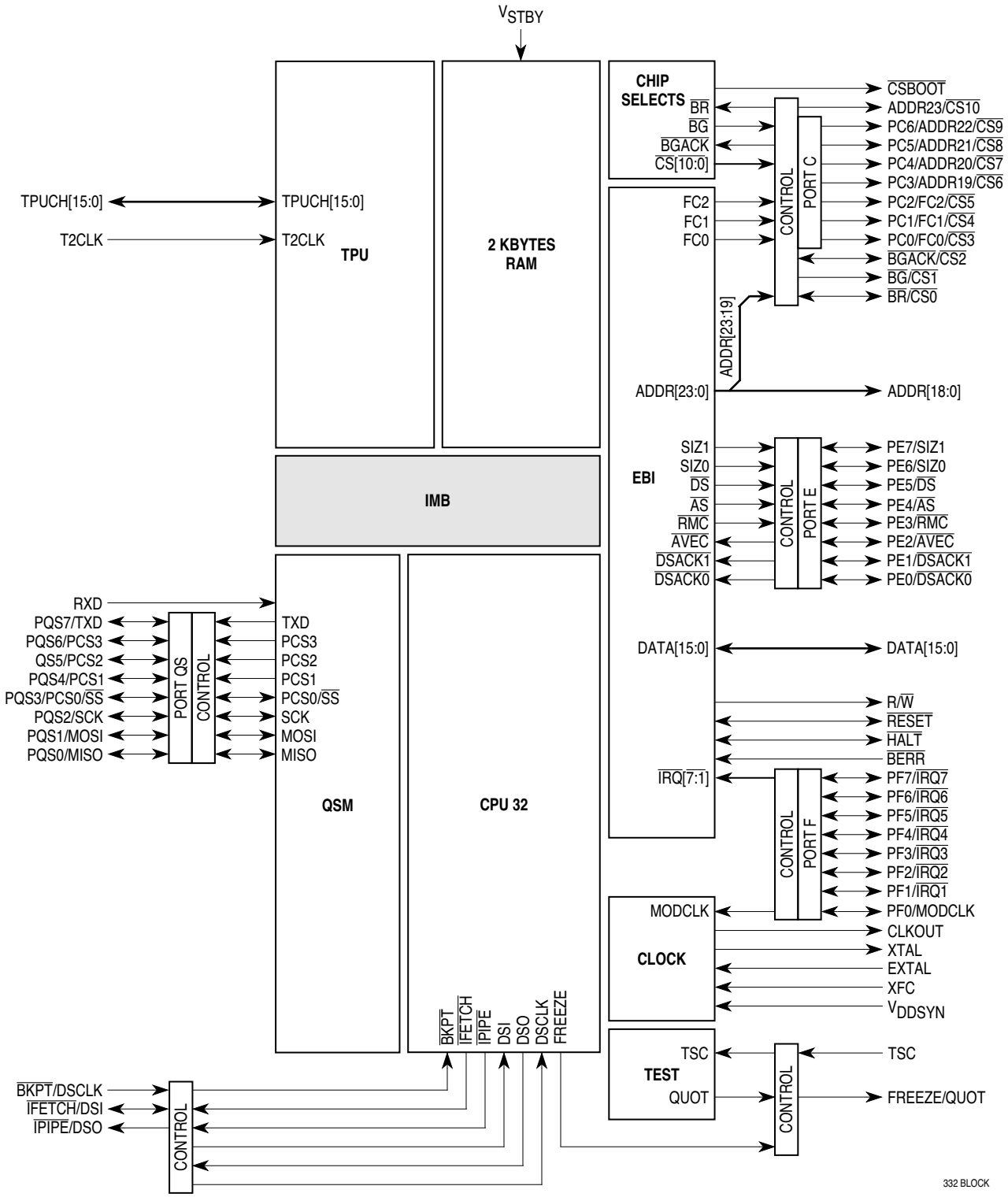


Figure 1 MCU Block Diagram

1.3 Pin Assignments

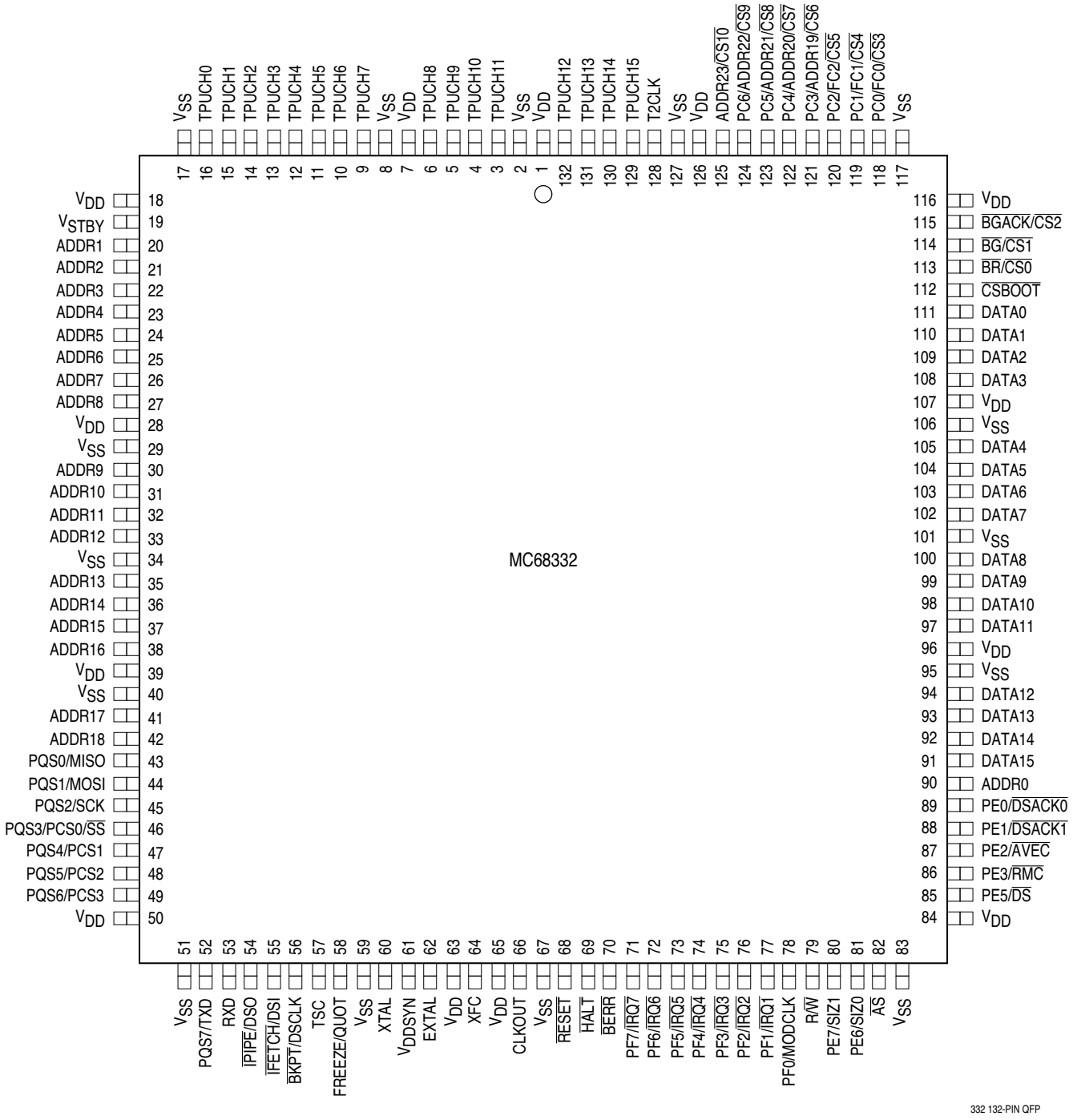


Figure 2 MC68332 132-Pin QFP Pin Assignments

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	9	8	7	6	5	4	3	2	1	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	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{MODCLK}$	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:

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

where

PIT Period = Periodic interrupt timer period

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

EXTAL Frequency = Crystal frequency

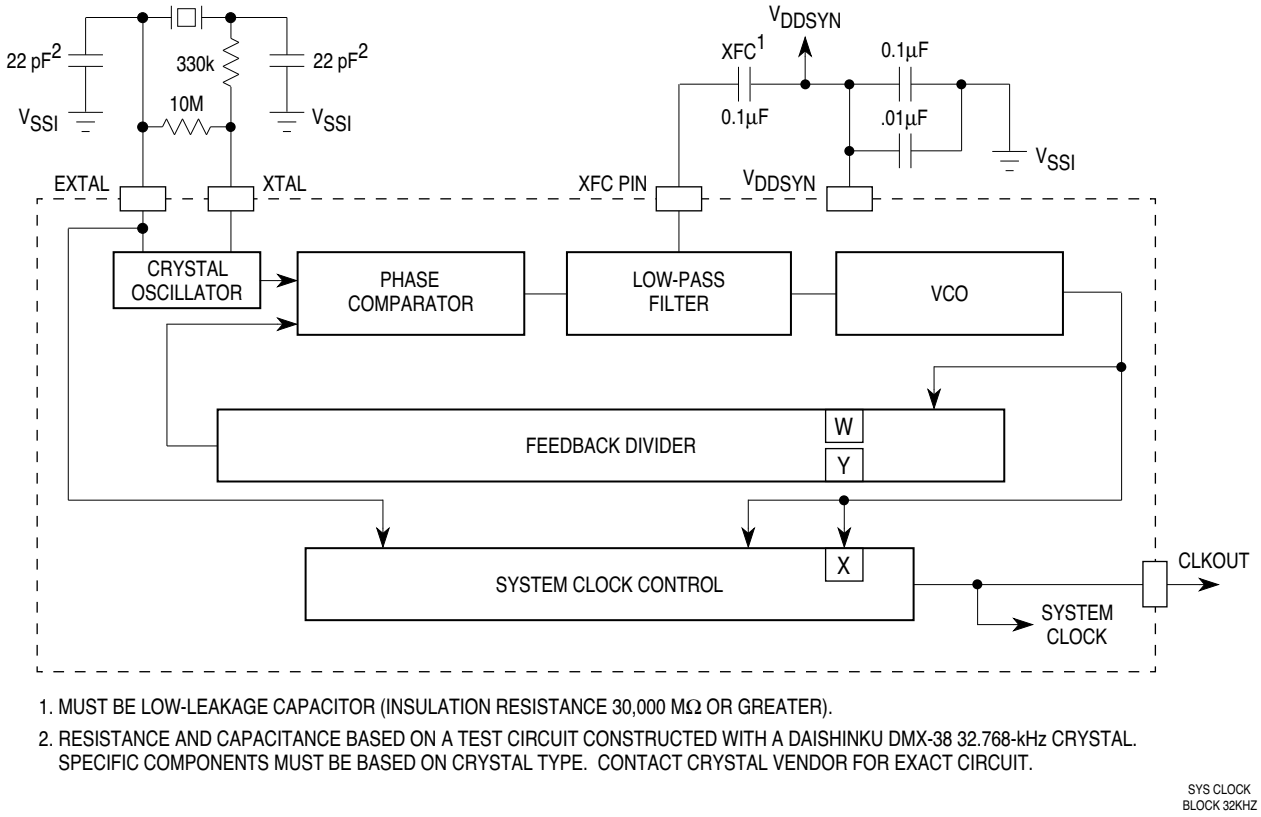
Prescale = 512 or 1 depending on the state of the PTP bit in the PITR

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.

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.

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobes, or interrupt acknowledge signals. Logic can also generate $\overline{\text{DSACK}}$ signals internally. A single $\overline{\text{DSACK}}$ generator is shared by all circuits. Multiple chip selects assigned to the same address and control must have the same number of wait states.

Chip selects can also be synchronized with the ECLK signal available on ADDR23.

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Select signals are active low. Refer to the following block diagram of a single chip-select circuit.

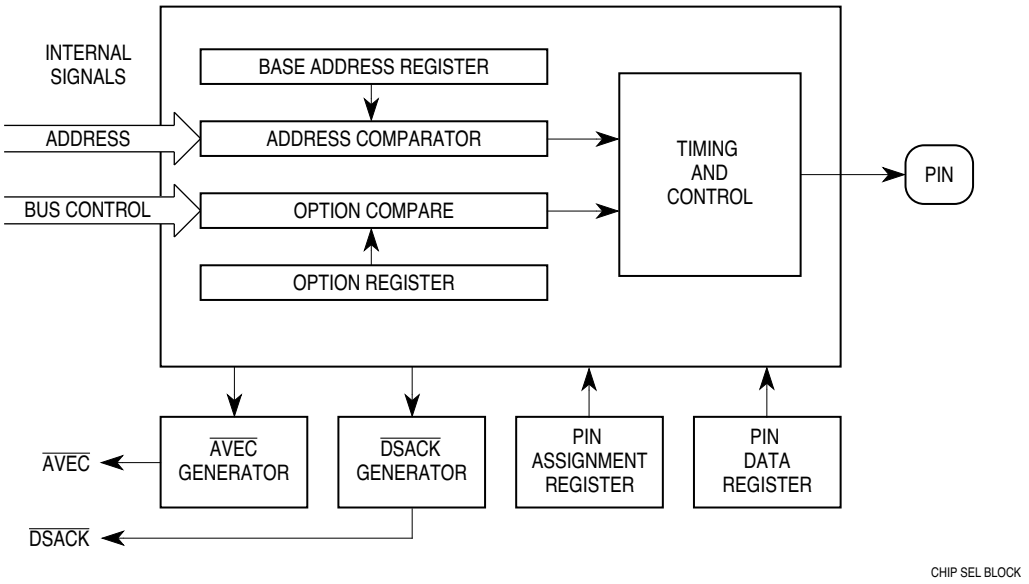


Figure 9 Chip-Select Circuit Block Diagram

The following table lists allocation of chip-selects and discrete outputs on the pins of the MCU.

Pin	Chip Select	Discrete Outputs
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
$\overline{\text{BR}}$	$\overline{\text{CS0}}$	—
$\overline{\text{BG}}$	$\overline{\text{CS1}}$	—
$\overline{\text{BGACK}}$	$\overline{\text{CS2}}$	—
FC0	$\overline{\text{CS3}}$	PC0
FC1	$\overline{\text{CS4}}$	PC1
FC2	$\overline{\text{CS5}}$	PC2
ADDR19	$\overline{\text{CS6}}$	PC3
ADDR20	$\overline{\text{CS7}}$	PC4
ADDR21	$\overline{\text{CS8}}$	PC5
ADDR22	$\overline{\text{CS9}}$	PC6
ADDR23	$\overline{\text{CS10}}$	ECLK

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 vectors. All interrupts with priorities less than seven 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 $\overline{\text{IRQ}}$[7:1] and the IP mask value. Each of the signals corresponds to an interrupt priority. $\overline{\text{IRQ1}}$ has the lowest priority, and $\overline{\text{IRQ7}}$ 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 $\overline{\text{IRQ7}}$) 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 $\overline{\text{IRQ}}$[6:1] are active-low level-sensitive inputs. External $\overline{\text{IRQ7}}$ is an active-low transition-sensitive input. $\overline{\text{IRQ7}}$ requires both an edge and a voltage level for validity.

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

3.8.2 Interrupt Processing Summary

A summary of the interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
B. Processor state is stacked. The contents of the status register and program counter are saved.
C. The interrupt acknowledge cycle begins:
    1. FC[2:0] are driven to %111 (CPU space) encoding.
    2. The address bus is driven as follows. ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the level of the interrupt request being acknowledged; and ADDR0 = %1.
    3. Request priority level is latched into the IP field in the status register from the address bus.
D. Modules or external peripherals that have requested interrupt service decode the request level in ADDR[3:1]. If the request level of at least one interrupting module or device is the same as the value in ADDR[3:1], interrupt arbitration contention takes place. When there is no contention, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.
E. After arbitration, the interrupt acknowledge cycle can be completed in one of three ways:

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 non-privileged 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.

4.3 Status Register

The status register contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program. The lower byte containing the condition codes is the only portion of the register available at the user privilege level; it is referenced as the condition code register (CCR) in user programs. At the supervisor privilege level, software can access the full status register, including the interrupt priority mask and additional control bits.

SR —Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1	T0	S	0	0	IP			0	0	0	X	N	Z	V	C
RESET:															
0	0	1	0	0	1	1	1	0	0	0	U	U	U	U	U

System Byte
- T[1:0] —Trace Enable
- S —Supervisor/User State
- Bits [12:11] —Unimplemented
- IP[2:0] —Interrupt Priority Mask

User Byte (Condition Code Register)
- Bits [7:5] — Unimplemented
- X — Extend
- N — Negative
- Z — Zero
- V — Overflow
- C — Carry

4.4 Data Types

Six basic data types are supported:

- Bits
- Packed Binary Coded Decimal Digits
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long-Word Integers (32 bits)
- Quad-Word Integers (64 bits)

4.5 Addressing Modes

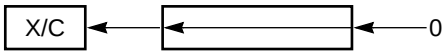
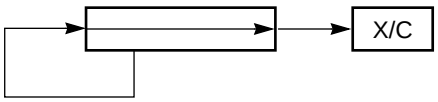
Addressing in the CPU32 is register-oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. This flexibility eliminates the need for extra instructions to store register contents in memory. The CPU32 supports seven basic addressing modes:

- Register direct
- Register indirect
- Register indirect with index
- Program counter indirect with displacement
- Program counter indirect with index
- Absolute
- Immediate

Included in the register indirect addressing modes are the capabilities to post-increment, predecrement, and offset. The program counter relative mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the status register, stack pointer, or program counter.

4.6 Instruction Set Summary

Table 20 Instruction Set Summary

Instruction	Syntax	Operand Size	Operation
ABCD	Dn, Dn – (An), – (An)	8 8	$Source_{10} + Destination_{10} + X \Rightarrow$ Destination
ADD	Dn, <ea> <ea>, Dn	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ADDA	<ea>, An	16, 32	Source + Destination ⇒ Destination
ADDI	#<data>, <ea>	8, 16, 32	Immediate data + Destination ⇒ Destination
ADDQ	# <data>, <ea>	8, 16, 32	Immediate data + Destination ⇒ Destination
ADDX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Source + Destination + X ⇒ Destination
AND	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source • Destination ⇒ Destination
ANDI	# <data>, <ea>	8, 16, 32	Data • Destination ⇒ Destination
ANDI to CCR	# <data>, CCR	8	Source • CCR ⇒ CCR
ANDI to SR[1]	# <data>, SR	16	Source • SR ⇒ SR
ASL	Dn, Dn # <data>, Dn Í	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn # <data>, Dn Í	8, 16, 32 8, 16, 32 16	
Bcc	label	8, 16, 32	If condition true, then PC + d ⇒ PC
BCHG	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{\text{bit number}\rangle\text{of destination}} \Rightarrow Z \Rightarrow$ bit of destinatic
BCLR	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination}}$ 0 ⇒ $\overline{\text{b}}$it of destination
BGND	none	none	If background mode enabled, then enter background mode, else format/vector ⇒ – (SSP); PC ⇒ – (SSP); SR ⇒ – (SSP); (vector) ⇒ PC
BKPT	# <data>	none	If breakpoint cycle acknowledged, then execute returned operation word, else trap as illegal instruction
BRA	label	8, 16, 32	PC + d ⇒ PC
BSET	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination}} \Rightarrow Z$; 1 ⇒ bit of destination
BSR	label	8, 16, 32	SP – 4 ⇒ SP; PC ⇒ (SP); PC + d ⇒ PC
BTST	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination}} \Rightarrow Z$
CHK	<ea>, Dn	16, 32	If Dn < 0 or Dn > (ea), then CHK exception
CHK2	<ea>, Rn	8, 16, 32	If Rn < lower bound or Rn > upper bound, then CHK exception
CLR	Í	8, 16, 32	0 ⇒ Destination
CMP	<ea>, Dn	8, 16, 32	(Destination – Source), CCR shows results
CMPA	<ea>, An	16, 32	(Destination – Source), CCR shows results
CMPI	# <data>, <ea>	8, 16, 32	(Destination – Data), CCR shows results
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

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.

HSQR0 — Host Sequence Register 0 **$YFFE14**

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

HSQR1 — Host Sequence Register 1 **$YFFE16**

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

CH[15:0] — Encoded Host Sequence

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

HSRR0 — Host Service Request Register 0 **$YFFE18**

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

HSRR1 — Host Service Request Register 1 **$YFFE1A**

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

CH[15:0] — Encoded Type of Host Service

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified. A host service request field cleared to %00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three nonzero states. The CPU should monitor the host service request register until the TPU clears the service request to %00 before the CPU changes any parameters or issues a new service request to the channel.

CPR0 — Channel Priority Register 0 **$YFFE1C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15		CH 14		CH13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1 **$YFFE1E**

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

CH[15:0] — Encoded One of Three Channel Priority Levels

6.2 Address Map

The “Access” column in the QSM 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 QSMCR.

Table 24 QSM Address Map

Access	Address	15 8	7 0
S	$YFFC00	QSM MODULE CONFIGURATION (QSMCR)	
S	$YFFC02	QSM TEST (QTEST)	
S	$YFFC04	QSM INTERRUPT LEVEL (QILR)	QSM INTERRUPT VECTOR (QIVR)
S/U	$YFFC06	NOT USED	
S/U	$YFFC08	SCI CONTROL 0 (SCCR0)	
S/U	$YFFC0A	SCI CONTROL 1 (SCCR1)	
S/U	$YFFC0C	SCI STATUS (SCSR)	
S/U	$YFFC0E	SCI DATA (SCDR)	
S/U	$YFFC10	NOT USED	
S/U	$YFFC12	NOT USED	
S/U	$YFFC14	NOT USED	PQS DATA (PORTQS)
S/U	$YFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
S/U	$YFFC18	SPI CONTROL 0 (SPCR0)	
S/U	$YFFC1A	SPI CONTROL 1 (SPCR1)	
S/U	$YFFC1C	SPI CONTROL 2 (SPCR2)	
S/U	$YFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
S/U	$YFFC20–$YFFCFF	NOT USED	
S/U	$YFFD00–$YFFD1F	RECEIVE RAM (RR[0:F])	
S/U	$YFFD20–$YFFD3F	TRANSMIT RAM (TR[0:F])	
S/U	$YFFD40–$YFFD4F	COMMAND RAM (CR[0:F])	

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

Command RAM is used by the QSPI when in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 16 bytes. Each byte is divided into two fields. The peripheral chip-select field enables peripherals for transfer. The command control field provides transfer options.

A maximum of 16 commands can be in the queue. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP. (Both of these fields are in SPCR2.)

CONT — Continue

0 = Control of chip selects returned to PORTQS after transfer is complete.
1 = Peripheral chip selects remain asserted after transfer is complete.

BITSE — Bits per Transfer Enable

0 = 8 bits
1 = Number of bits set in BITS field of SPCR0

DT — Delay after Transfer

The QSPI provides a variable delay at the end of serial transfer to facilitate the interface with peripherals that have a latency requirement. The delay between transfers is determined by the SPCR1 DTL field.

DSCK — PCS to SCK Delay

0 = PCS valid to SCK transition is one-half SCK.
1 = SPCR1 DSCKL field specifies delay from PCS valid to SCK.

PCS[3:0] — Peripheral Chip Select

Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select can be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided that proper fanout is observed.

$\overline{SS}$ — Slave Mode Select

Initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault will be generated.

6.5.4 Operating Modes

The QSPI operates in either master or slave mode. Master mode is used when the MCU originates data transfers. Slave mode is used when an external device initiates serial transfers to the MCU through the QSPI. Switching between the modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSM and QSPI registers must be properly initialized.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from transmit RAM and received into receive RAM.

In slave mode, operation proceeds in response to $\overline{SS}$ pin activation by an external bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multimaster operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set, nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

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 that was in progress when TE is cleared.

RE — Receiver Enable

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

RWU — Receiver Wakeup

0 = Normal receiver operation (received data recognized)
1 = Wakeup mode enabled (received data ignored until awakened)

Setting RWU enables the wakeup function, which allows the SCI to ignore received data until awakened by either an idle line or address mark (as determined by WAKE). When in wakeup mode, the receiver status flags are not set, and interrupts are inhibited. This bit is cleared automatically (returned to normal mode) when the receiver is awakened.

IDLE — Idle-Line Detected Flag

0 = SCI receiver did not detect an idle-line condition.
1 = SCI receiver detected an idle-line condition.

IDLE is disabled when RWU in SCCR1 is set. IDLE is set when the SCI receiver detects the idle-line condition specified by ILT in SCCR1. If cleared, IDLE will not set again until after RDRF is set. RDRF is set when a break is received, so that a subsequent idle line can be detected.

OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.
1 = RDRF is not cleared before new data arrives.

OR is set when a new byte is ready to be transferred from the receive serial shifter to the RDR, and RDRF is still set. Data transfer is inhibited until OR is cleared. Previous data in RDR remains valid, but data received during overrun condition (including the byte that set OR) is lost.

NF — Noise Error Flag

0 = No noise detected on the received data
1 = Noise occurred on the received data

NF is set when the SCI receiver detects noise on a valid start bit, on any data bit, or on a stop bit. It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times. If none of the three samples are the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until an entire frame is received and RDRF is set.

FE — Framing Error Flag

0 = No framing error on the received data.
1 = Framing error or break occurred on the received data.

FE is set when the SCI receiver detects a zero where a stop bit was to have occurred. FE is not set until the entire frame is received and RDRF is set. A break can also cause FE to be set. It is possible to miss a framing error if RXD happens to be at logic level one at the time the stop bit is expected.

PF — Parity Error Flag

0 = No parity error on the received data
1 = Parity error occurred on the received data

PF is set when the SCI receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set.

SCDR — SCI Data Register **$YFFC0E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

SCDR contains two data registers at the same address. Receive data register (RDR) is a read-only register that contains data received by the SCI. The data comes into the receive serial shifter and is transferred to RDR. Transmit data register (TDR) is a write-only register that contains data to be transmitted. The data is first written to TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when SCDR is read, or the first eight data bits to be transmitted when SCDR is written. R8/T8 are used when the SCI is configured for 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.

RASP — RAM Array Space Field
0 = TPURAM array is placed in unrestricted space
1 = TPURAM array is placed in supervisor space

TRAMTST — TPURAM Test Register **$YFFB02**

TRAMTST is used for factory testing of the TPURAM module.

TRAMBAR — TPURAM Base Address and Status Register **$YFFB04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	NOT USED		RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADDR[23:11] — RAM Array Base Address
These bits specify address lines ADDR[23:11] of the base address of the RAM array when enabled.

RAMDS — RAM Array Disable
0 = RAM array is enabled
1 = RAM array is disabled
The RAM array is disabled by internal logic after a master reset. Writing a valid base address to the RAM array base address field (bits [15:3]) automatically clears RAMDS, enabling the RAM array.

7.4 TPURAM Operation

There are six TPURAM operating modes, as follows:

1. The TPURAM module is in normal mode when powered by V_{DD}. The array can be accessed by byte, word, or long word. A byte or aligned word (high-order byte is at an even address) access only takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles.
2. Standby mode is intended to preserve TPURAM contents when V_{DD} is removed. TPURAM contents are maintained by V_{STBY}. Circuitry within the TPURAM module switches to the higher of V_{DD} or V_{STBY} with no loss of data. When TPURAM is powered by V_{STBY}, access to the array is not guaranteed.
3. Reset mode allows the CPU to complete the current bus cycle before resetting. When a synchronous reset occurs while a byte or word TPURAM access is in progress, the access will be completed. If reset occurs during the first word access of a long-word operation, only the first word access will be completed. If reset occurs during the second word access of a long word operation, the entire access will be completed. Data being read from or written to the RAM may be corrupted by asynchronous reset.
4. Test mode functions in conjunction with the SIM test functions. Test mode is used during factory test of the MCU.
5. Writing the STOP bit of TRAMMCR causes the TPURAM module to enter stop mode. The TPURAM array is disabled (which allows external logic to decode TPURAM addresses, if necessary), but all data is retained. If V_{DD} falls below V_{STBY} during stop mode, internal circuitry switches to V_{STBY}, as in standby mode. Stop mode is exited by clearing the STOP bit.
6. The TPURAM array may be used to emulate the microcode ROM in the TPU module. This provides a means of developing custom TPU code. The TPU selects TPU emulation mode. While in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses via the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses.