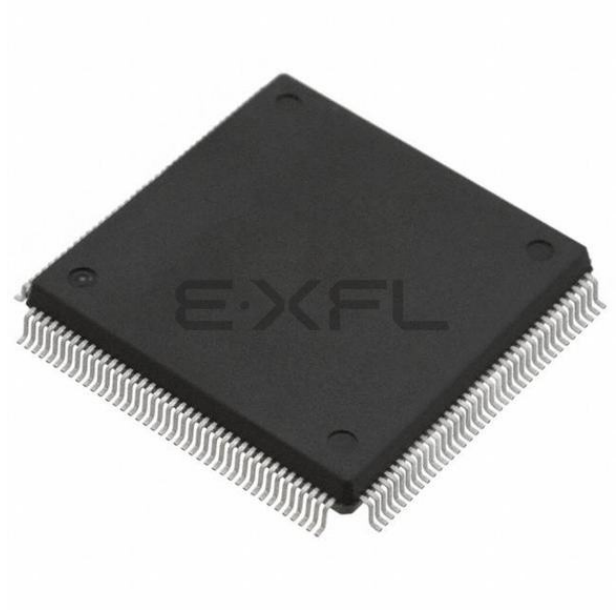


Welcome to E-XFL.COM

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	CANbus, EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	18
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	7.5K x 8
Voltage - Supply (Vcc/Vdd)	4.75V ~ 5.25V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	160-BQFP
Supplier Device Package	160-QFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68336gmab20

TSTRC — SIM Test Module Repetition Counter Register
TSTSC — SIM Test Module Shift Count Register
TTR — TouCAN Test Register
TXECTR — TouCAN Transmit Error Counter Register

Table 3-5 MC68336/376 Signal Functions

Mnemonic	Signal Name	Function
ADDR[23:0]	Address Bus	24-bit address bus used by the CPU32
AN[59:48]/[3:0]	QADC Analog Input	16 channel A/D converter analog input pins
AN[w, x, y, z]	QADC Analog Input	Four input channels utilized when operating in multiplexed mode
$\overline{AS}$	Address Strobe	Indicates that a valid address is on the address bus
$\overline{AVEC}$	Autovector	Requests an automatic vector during interrupt acknowledge
$\overline{BERR}$	Bus Error	Indicates that a bus error has occurred
$\overline{BG}$	Bus Grant	Indicates that the MCU has relinquished the bus
$\overline{BGACK}$	Bus Grant Acknowledge	Indicates that an external device has assumed bus mastership
$\overline{BKPT}$	Breakpoint	Signals a hardware breakpoint to the CPU
$\overline{BR}$	Bus Request	Indicates that an external device requires bus mastership
CLKOUT	System Clock Out	System clock output
CANRX0	TouCAN Receive Data	CAN serial data input
CANTX0	TouCAN Transmit Data	CAN serial data output
$\overline{CS[10:0]}$	Chip-Selects	Select external devices at programmed addresses
$\overline{CSBOOT}$	Boot Chip-Select	Chip-select for external bootstrap memory
CPWM[8:5]	CTM4 PWMs	Four pulse-width modulation channels
CTD[10:9]/[4:3]	CTM4 Double Action Channels	Bidirectional double action timer channels
CTM2C	CTM4 Modulus Clock	Modulus counter clock input
DATA[15:0]	Data Bus	16-bit data bus used by the CPU32
$\overline{DS}$	Data Strobe	Indicates that an external device should place valid data on the data bus during a read cycle and that valid data has been placed on the data bus by the CPU during a write cycle.
$\overline{DSACK[1:0]}$	Data and Size Acknowledge	Provides asynchronous data transfers and dynamic bus sizing
DSI, DSO, DSCLK	Developmental Serial In, Out, Clock	Serial I/O and clock for background debug mode
ECLK	E-Clock	M6800 bus clock output
ETRIG[2:1]	QADC External Trigger	External trigger pins used when a QADC scan queue is in external trigger mode
EXTAL, XTAL	Crystal Oscillator	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
FC[2:0]	Function Codes	Identify processor state and current address space
FREEZE	Freeze	Indicates that the CPU has acknowledged a breakpoint
$\overline{HALT}$	Halt	Suspend external bus activity
$\overline{IFETCH}$	Instruction Pipeline	Indicates instruction pipeline activity
$\overline{IPIPE}$	Instruction Pipeline	Indicates instruction pipeline activity
$\overline{IRQ[7:1]}$	Interrupt Request	Requests an interrupt of specified priority level from the CPU
MA[2:0]	QADC Multiplexed Address	When external multiplexing is used, these pins provide addresses to the external multiplexer
MISO	Master In, Slave Out	Serial input to QSPI in the master mode; serial output from QSPI in the slave mode
MODCLK	Clock Mode Select	Selects the source of the system clock
MOSI	Master Out, Slave In	Serial output from the QSPI in master mode; serial input to the QSPI in slave mode
PC[6:0]	Port C	SIM digital output port signals
PCS[3:0]	Peripheral Chip-Selects	QSPI peripheral chip-select
PE[7:0]	Port E	SIM digital input/output port signals
PF[7:0]	Port F	SIM digital input/output port signals

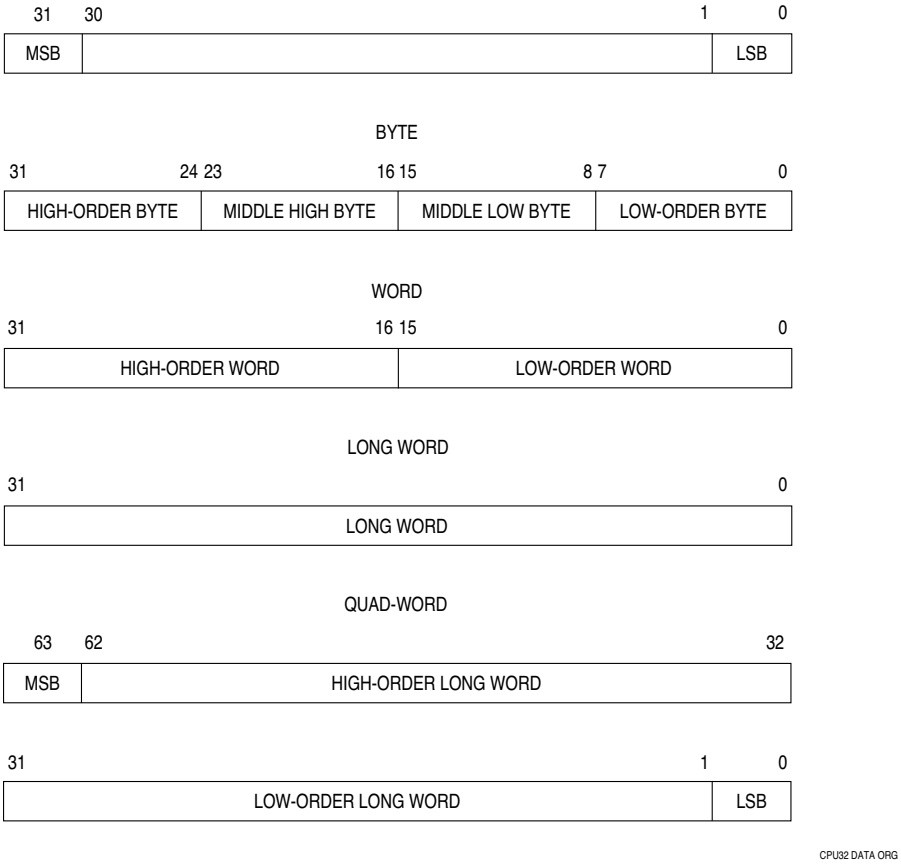


Figure 4-4 Data Organization in Data Registers

4.2.2 Address Registers

Each address register and stack pointer is 32 bits wide and holds a 32-bit address. Address registers cannot be used for byte-sized operands. Therefore, when an address register is used as a source operand, either the low-order word or the entire long-word operand is used, depending upon the operation size. When an address register is used as the destination operand, the entire register is affected, regardless of the operation size. If the source operand is a word size, it is sign-extended to 32 bits. Address registers are used primarily for addresses and to support address computation. The instruction set includes instructions that add to, subtract from, compare, and move the contents of address registers. **Figure 4-5** shows the organization of addresses in address registers.

4.2.4.2 Alternate Function Code Registers

Alternate function code registers (SFC and DFC) contain 3-bit function codes. Function codes can be considered extensions of the 24-bit linear address that optionally provide as many as eight 16-Mbyte address spaces. The processor automatically generates function codes to select address spaces for data and programs at the user and supervisor privilege levels and to select a CPU address space used for processor functions (such as breakpoint and interrupt acknowledge cycles).

Registers SFC and DFC are used by the MOVES instruction to specify explicitly the function codes of the memory address. The MOVEC instruction is used to transfer values to and from the alternate function code registers. This is a long-word transfer; the upper 29 bits are read as zeros and are ignored when written.

4.2.5 Vector Base Register (VBR)

The VBR contains the base address of the 1024-byte exception vector table, consisting of 256 exception vectors. Exception vectors contain the memory addresses of routines that begin execution at the completion of exception processing. More information on the VBR and exception processing can be found in **4.9 Exception Processing**.

4.3 Memory Organization

Memory is organized on a byte-addressable basis in which lower addresses correspond to higher order bytes. For example, the address N of a long-word data item corresponds to the address of the most significant byte of the highest order word. The address of the most significant byte of the low-order word is N + 2, and the address of the least significant byte of the long word is N + 3. The CPU32 requires long-word and word data and all instructions to be aligned on word boundaries. Refer to **Figure 4-6**. If this does not happen, an exception will occur when the CPU32 accesses the misaligned instruction or data. Data misalignment is not supported.

4.9.2 Types of Exceptions

An exception can be caused by internal or external events.

An internal exception can be generated by an instruction or by an error. The TRAP, TRAPcc, TRAPV, BKPT, CHK, CHK2, RTE, and DIV instructions can cause exceptions during normal execution. Illegal instructions, instruction fetches from odd addresses, word or long-word operand accesses from odd addresses, and privilege violations also cause internal exceptions.

Sources of external exception include interrupts, breakpoints, bus errors, and reset requests. Interrupts are peripheral device requests for processor action. Breakpoints are used to support development equipment. Bus error and reset are used for access control and processor restart.

4.9.3 Exception Processing Sequence

For all exceptions other than a reset exception, exception processing occurs in the following sequence. Refer to **5.7 Reset** for details of reset processing.

As exception processing begins, the processor makes an internal copy of the status register. After the copy is made, the processor state bits in the status register are changed — the S bit is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing. For reset and interrupt exceptions, the interrupt priority mask is also updated.

Next, the exception number is obtained. For interrupts, the number is fetched from CPU space $F (the bus cycle is an interrupt acknowledge). For all other exceptions, internal logic provides a vector number.

Next, current processor status is saved. An exception stack frame is created and placed on the supervisor stack. All stack frames contain copies of the status register and the program counter for use by RTE. The type of exception and the context in which the exception occurs determine what other information is stored in the stack frame.

Finally, the processor prepares to resume normal execution of instructions. The exception vector offset is determined by multiplying the vector number by four, and the offset is added to the contents of the VBR to determine displacement into the exception vector table. The exception vector is loaded into the program counter. If no other exception is pending, the processor will resume normal execution at the new address in the PC.

4.10 Development Support

The following features have been implemented on the CPU32 to enhance the instrumentation and development environment:

- M68000 Family Development Support
- Background Debug Mode
- Deterministic Opcode Tracking
- Hardware Breakpoints

Table 5-3 System Frequencies from 4.194 MHz Reference (Continued)

Modulus	Prescaler			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
100000	4325 kHz	8651 kHz	17302 kHz	34603 kHz
100001	4456	8913	17826	35652
100010	4588	9175	18350	36700
100011	4719	9437	18874	37749
100100	4850	9699	19399	38797
100101	4981	9961	19923	39846
100110	5112	10224	20447	40894
100111	5243	10486	20972	41943
101000	5374	10748	21496	42992
101001	5505	11010	22020	44040
101010	5636	11272	22544	45089
101011	5767	11534	23069	46137
101100	5898	11796	23593	47186
101101	6029	12059	24117	48234
101110	6160	12321	24642	49283
101111	6291	12583	25166	50332
110000	6423	12845	25690	51380
110001	6554	13107	26214	52428
110010	6685	13369	26739	53477
110011	6816	13631	27263	54526
110100	6947	13894	27787	55575
110101	7078	14156	28312	56623
110110	7209	14418	28836	57672
110111	7340	14680	29360	58720
111000	7471	14942	2988	59769
111001	7602	15204	30409	60817
111010	7733	15466	30933	61866
111011	7864	15729	31457	62915
111100	7995	15991	31982	63963
111101	8126	16253	32506	65011
111110	8258	16515	33030	66060
111111	8389	16777	33554	67109

Figure 5-7 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer.

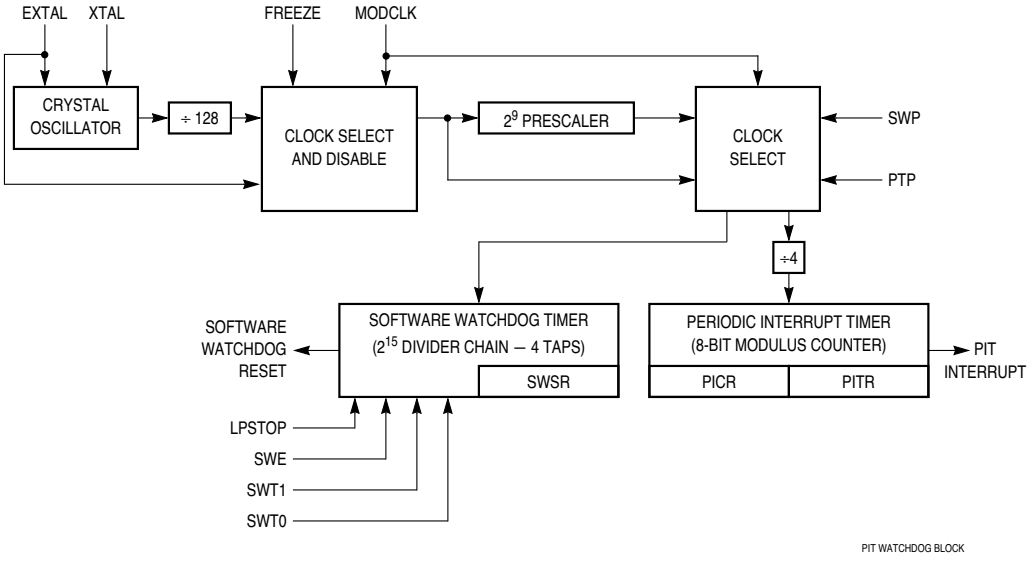


Figure 5-7 Periodic Interrupt Timer and Software Watchdog Timer

5.4.6 Periodic Interrupt Timer

The periodic interrupt timer (PIT) allows the generation of interrupts of specific priority at predetermined intervals. This capability is often used to schedule control system tasks that must be performed within time constraints. The timer consists of a prescaler, a modulus counter, and registers that determine interrupt timing, priority and vector assignment. Refer to **4.9 Exception Processing** for more information.

The periodic interrupt timer modulus counter is clocked by one of two signals. When the PLL is enabled (MODCLK = 1 during reset), $f_{ref} \div 128$ is used. When the PLL is disabled (MODCLK = 0 during reset), f_{ref} is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the periodic interrupt timer. One of two options, either no prescaling, or prescaling by a factor of 512, can be selected. The value of PTP is affected by the state of the MODCLK pin during reset, as shown in **Table 5-7**. System software can change PTP value.

Table 5-7 MODCLK Pin and PTP Bit at Reset

MODCLK	PTP
0 (PLL disabled)	1 (÷ 512)
1 (PLL enabled)	0 (÷ 1)

When the MCU completes a bus cycle while the HALT signal is asserted, the data bus goes into a high-impedance state and the AS and DS signals are driven to their inactive states. Address, function code, size, and read/write signals remain in the same state.

The halt operation has no effect on bus arbitration. However, when external bus arbitration occurs while the MCU is halted, address and control signals go into a high-impedance state. If HALT is still asserted when the MCU regains control of the bus, address, function code, size, and read/write signals revert to the previous driven states. The MCU cannot service interrupt requests while halted.

5.6.6 External Bus Arbitration

The MCU bus design provides for a single bus master at any one time. Either the MCU or an external device can be master. Bus arbitration protocols determine when an external device can become bus master. Bus arbitration requests are recognized during normal processing, HALT assertion, and when the CPU32 has halted due to a double bus fault.

The bus controller in the MCU manages bus arbitration signals so that the MCU has the lowest priority. External devices that need to obtain the bus must assert bus arbitration signals in the sequences described in the following paragraphs.

Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes bus master first. The protocol sequence is:

1. An external device asserts the bus request signal (BR);
2. The MCU asserts the bus grant signal (BG) to indicate that the bus is available;
3. An external device asserts the bus grant acknowledge (BGACK) signal to indicate that it has assumed bus mastership.

BR can be asserted during a bus cycle or between cycles. BG is asserted in response to BR. To guarantee operand coherency, BG is only asserted at the end of operand transfer. Additionally, BG is not asserted until the end of an indivisible read-modify-write operation (when RMC is negated).

If more than one external device can be bus master, required external arbitration must begin when a requesting device receives BG. An external device must assert BGACK when it assumes mastership, and must maintain BGACK assertion as long as it is bus master.

Two conditions must be met for an external device to assume bus mastership. The device must receive BG through the arbitration process, and BGACK must be inactive, indicating that no other bus master is active. This technique allows the processing of bus requests during data transfer cycles.

BG is negated a few clock cycles after BGACK transition. However, if bus requests are still pending after BG is negated, the MCU asserts BG again within a few clock cycles.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle line type (ILT) bit in SCCR1 determines which type of detection is used. When an idle line condition is detected, the IDLE flag in SCSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

In some applications, software overhead can cause a bit-time of logic level one to occur between frames. This bit-time does not affect content, but if it occurs after a frame of ones when short detection is enabled, the receiver flags an idle line.

When the ILIE bit in SCCR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCSR and SCDR in sequence. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

9.4.3.8 Receiver Wake-Up

The receiver wake-up function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wake-up mode by setting the RWU bit in SCCR1. While RWU is set, receiver status flags and interrupts are disabled. Although the CPU32 can clear RWU, it is normally cleared by hardware during wake-up.

The WAKE bit in SCCR1 determines which type of wake-up is used. When WAKE = 0, idle-line wake-up is selected. When WAKE = 1, address-mark wake-up is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wake-up allows a receiver to sleep until an idle line is detected. When an idle-line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The byte is received normally, transferred to the RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wake-up to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address-mark wake-up uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The byte is received normally, transferred to the RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address-mark wake-up allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

8.13 Interrupts

The QADC supports both polled and interrupt driven operation. Status bits in QASR reflect the operating condition of each queue and can optionally generate interrupts when enabled by the appropriate bits in QACR1 and/or QACR2.

8.13.1 Interrupt Sources

The QADC has four interrupt service sources, each of which is separately enabled. Each time the result is written for the last CCW in a queue, the completion flag for the corresponding queue is set, and when enabled, an interrupt request is generated. In the same way, each time the result is written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt request is generated.

Table 8-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity.

Table 8-5 QADC Status Flags and Interrupt Sources

Queue	Queue Activity	Status Flag	Interrupt Enable Bit
Queue 1	Result written for the last CCW in queue 1	CF1	CIE1
	Result written for a CCW with pause bit set in queue 1	PF1	PIE1
Queue 2	Result written for the last CCW in queue 2	CF2	CIE2
	Result written for a CCW with pause bit set in queue 2	PF2	PIE2

Both polled and interrupt-driven QADC operations require that status flags must be cleared after an event occurs. Flags are cleared by first reading QASR with the appropriate flag bits set to one, then writing zeros to the flags that are to be cleared. A flag can be cleared only if the flag was a logic one at the time the register was read by the CPU. If a new event occurs between the time that the register is read and the time that it is written, the associated flag is not cleared.

8.13.2 Interrupt Register

The QADC interrupt register QADCINT specifies the priority level of QADC interrupt requests and the upper six bits of the vector number provided during an interrupt acknowledge cycle.

The values contained in the IRLQ1 and IRLQ2 fields in QADCINT determine the priority of QADC interrupt service requests. A value of %000 in either field disables the interrupts associated with that field. The interrupt levels for queue 1 and queue 2 may be different.

The IVB[7:2] bits specify the upper six bits of each QADC interrupt vector number. IVB[1:0] have fixed assignments for each of the four QADC interrupt sources. Refer to **8.13.3 Interrupt Vectors** for more information.

SECTION 10 CONFIGURABLE TIMER MODULE 4

This section is an overview of CTM4 function. Refer to the *CTM Reference Manual* (CTMRM/AD) for a comprehensive discussion of CTM capabilities.

10.1 General

The configurable timer module 4 (CTM4) consists of several submodules which are located on either side of the CTM4 internal submodule bus (SMB). All data and control signals within the CTM4 are passed over this bus. The SMB is connected to the outside world via the bus interface unit submodule (BIUSM), which is connected to the intermodule bus (IMB), and subsequently the CPU32. This configuration allows the CPU32 to access the data and control registers in each CTM4 submodule on the SMB. Three time base buses (TBB1, TBB2 and TBB4), each 16-bits wide, are used to transfer timing information from counters to action submodules. **Figure 10-1** shows a block diagram of the CTM4.

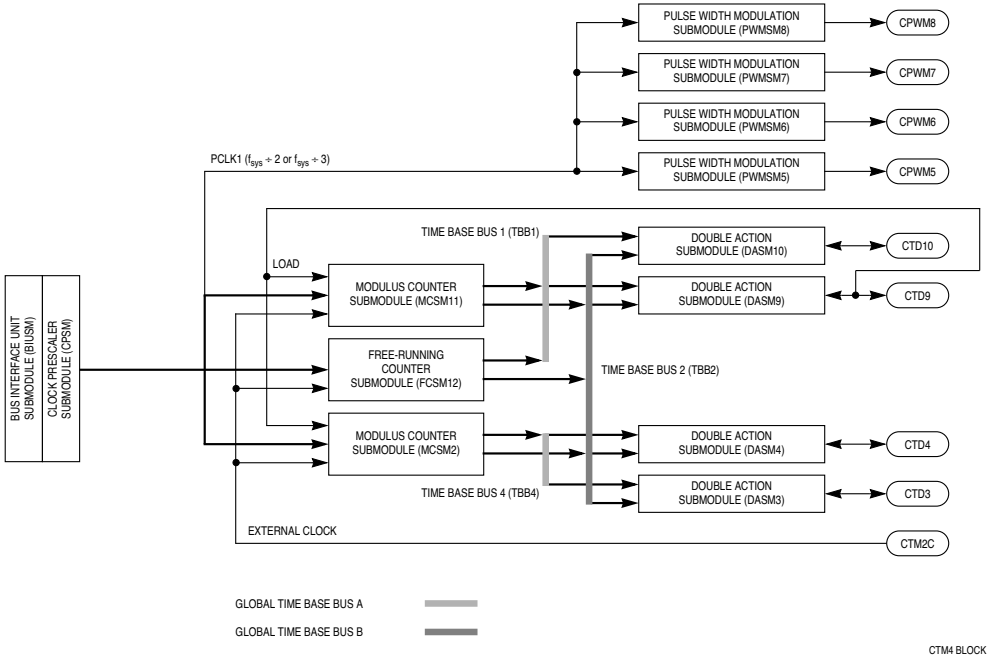


Figure 10-1 CTM4 Block Diagram

10.5.1 CPSM Registers

The CPSM contains a control register (CPCR) and a test register (CPTR). All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. Refer to **D.7.4 CPSM Control Register** and **D.7.5 CPSM Test Register** for information concerning CPSM register and bit descriptions.

10.6 Free-Running Counter Submodule (FCSM)

The free-running counter submodule (FCSM) has a 16-bit up counter with an associated clock source selector, selectable time-base bus drivers, control registers, status bits, and interrupt logic. When the 16-bit up counter overflows from $FFFF to $0000, an optional overflow interrupt request can be generated. The current state of the 16-bit counter is the primary output of the counter submodules. The user can select which, if any, time base bus is to be driven by the 16-bit counter. A software control register selects whether the clock input to the counter is one of the taps from the prescaler or an input pin. The polarity of the external input pin is also programmable.

In order to count, the FCSM requires the CPSM clock signals to be present. After reset, the FCSM does not count until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM4 submodules to be synchronized.

The CTM4 has one FCSM. **Figure 10-3** shows a block diagram of the FCSM.

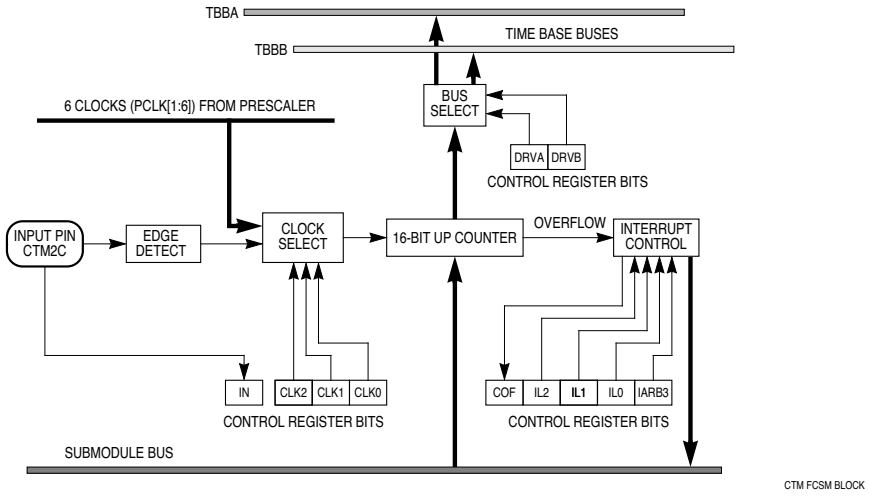


Figure 10-3 FCSM Block Diagram

10.7.7 MCSM Registers

The MCSM contains a status/interrupt/control register, a counter, and a modulus latch. All unused bits and reserved address locations return zero when read. Writes to unused bits and reserved address locations have no effect. The CTM4 contains three MCSMs, each with its own set of registers. Refer to **D.7.8 MCSM Status/Interrupt/Control Registers**, **D.7.9 MCSM Counter Registers**, and **D.7.10 MCSM Modulus Latch Registers** for information concerning MCSM register and bit descriptions.

10.8 Double-Action Submodule (DASM)

The double-action submodule (DASM) allows two 16-bit input capture or two 16-bit output compare functions to occur automatically without software intervention. The input edge detector can be programmed to trigger the capture function on user-specified edges. The output flip flop can be set by one of the output compare functions, and reset by the other one. Interrupt requests can optionally be generated by the input capture and the output compare functions. The user can select one of two incoming time bases for the input capture and output compare functions.

Six operating modes allow the DASM input capture and output compare functions to perform pulse width measurement, period measurement, single pulse generation, and continuous pulse width modulation, as well as standard input capture and output compare. The DASM can also function as a single I/O pin.

DASM operating mode is determined by the mode select field (MODE[3:0]) in the DASM status/interrupt/control register (DASMSIC). **Table 10-2** shows the different DASM operating modes.

Table 10-2 DASM Modes of Operation

MODE[3:0]	Mode	Description of Mode
0000	DIS	Disabled — Input pin is high impedance; IN gives state of input pin
0001	IPWM	Input pulse width measurement — Capture on leading edge and the trailing edge of an input pulse
0010	IPM	Input period measurement — Capture two consecutive rising/falling edges
0011	IC	Input capture — Capture when the designated edge is detected
0100	OCB	Output compare, flag set on B compare — Generate leading and trailing edges of an output pulse and set the flag
0101	OCAB	Output compare, flag set on A and B compare — Generate leading and trailing edges of an output pulse and set the flag
0110	—	Reserved
0111	—	Reserved
1xxx	OPWM	Output pulse width modulation — Generate continuous PWM output with 7, 9, 11, 12, 13, 14, 15, or 16 bits of resolution

The DASM is composed of two timing channels (A and B), an output flip-flop, an input edge detector, some control logic and an interrupt interface. All control and status bits are contained in DASMSIC.

Channel A consists of one 16-bit data register and one 16-bit comparator. To the user, channel B also appears to consist of one 16-bit data register and one 16-bit compar-

SECTION 12 STANDBY RAM WITH TPU EMULATION

The standby RAM module with TPU emulation capability (TPURAM) consists of a control register block and a 3.5-Kbyte array of fast (two system clock) static RAM, which is especially useful for system stacks and variable storage. The TPURAM responds to both program and data space accesses. The TPURAM can also be used to emulate TPU microcode ROM.

12.1 General

The TPURAM can be mapped to the lower 3.5 Kbytes of any 4-Kbyte boundary in the address map, but must not overlap the module control registers as overlap makes the registers inaccessible. Data can be read or written in bytes, words or long words. The TPURAM is powered by V_{DD} in normal operation. During power-down, TPURAM contents can be maintained by power from the V_{STBY} input. Power switching between sources is automatic.

12.2 TPURAM Register Block

There are three TPURAM control registers: the TPURAM module configuration register (TRAMMCR), the TPURAM test register (TRAMTST), and the TPURAM base address and status register (TRAMBAR). To protect these registers from accidental modification, they are always mapped to supervisor data space.

The TPURAM control register block begins at address $7FFB00 or $FFFB00, depending on the value of the module mapping (MM) bit in the SIM configuration register (SIMCR). Refer to **5.2.1 Module Mapping** for more information on how the state of MM affects the system.

The TPURAM control register block occupies eight bytes of address space. Unimplemented register addresses are read as zeros, and writes have no effect. Refer to **D.9 Standby RAM Module with TPU Emulation Capability (TPURAM)** for register block address map and register bit/field definitions.

12.3 TPURAM Array Address Mapping

The base address and status register TRAMBAR specifies the TPURAM array base address in the MCU memory map. TRAMBAR[15:4] specify the 12 high-order bits of the base address. The TPU bus interface unit compares these bits to address lines ADDR[23:12]. If the two match, then the low order address lines and the SIZ[1:0] signals are used to access the RAM location in the array.

The RAM disable (RAMDS) bit, the LSB of TRAMBAR, indicates whether the TPURAM array is active (RAMDS = 0) or disabled (RAMDS = 1). The array is disabled coming out of reset and remains disabled if the base address field is programmed with an address that overlaps the address of the module control register block. Writing a valid base address to TRAMBAR[15:4] clears RAMDS and enables the array.

2. Initialize IARB[3:0] to a non-zero value in CANMCR.
3. Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt.

E. Negate the HALT bit in the module configuration register
1. At this point, the TouCAN will attempt to synchronize with the CAN bus.

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer should be to deactivate the buffer by setting its code field to the proper value. This requirement is mandatory to assure data coherency.

13.5.3 Transmit Process

The transmit process includes preparing a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. For the user, this involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN will perform all additional steps necessary to transmit the message onto the CAN bus.

The user should prepare/change a message buffer for transmission by executing the following steps.

1. Write the control/status word to hold the transmit buffer inactive (code = %1000)
2. Write the ID_HIGH and ID_LOW words
3. Write the data bytes
4. Write the control/status word (active TX code, TX length)

NOTE

Steps 1 and 4 are mandatory to ensure data coherency while preparing a message buffer for transmission.

Once an active transmit code is written to a transmit message buffer, that buffer will begin participating in an internal arbitration process as soon as the CAN bus is sensed to be free by the receiver, or at the inter-frame space. If there are multiple messages awaiting transmission, this internal arbitration process selects the message buffer from which the next frame is transmitted.

When this process is over, and a message buffer is selected for transmission, the frame from that message buffer is transferred to the serial message buffer for transmission.

While transmitting, the TouCAN will transmit no more than eight data bytes, even if the transmit length contains a value greater than eight.

D.1.1 CPU32 Register Model

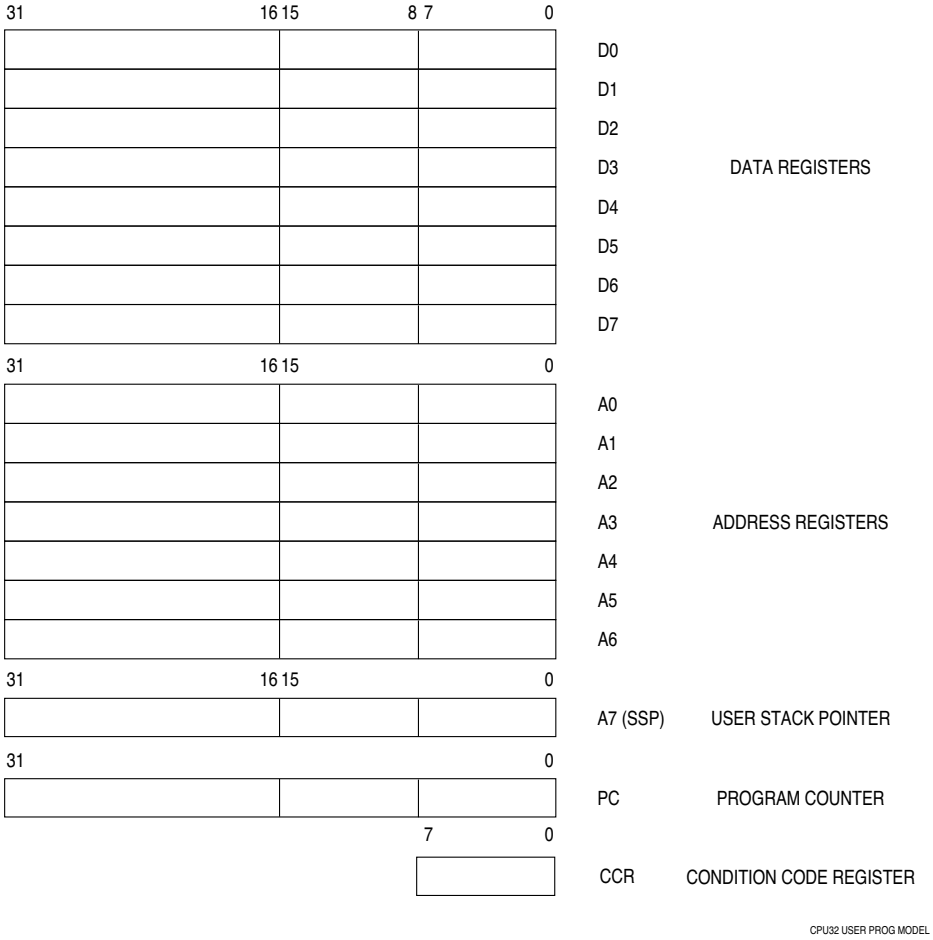


Figure D-1 User Programming Model

Table D-12 Reset Pin Function of $\overline{CS}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{CS10}$/ ADDR23	$\overline{CS9}$/ ADDR22	$\overline{CS8}$/ ADDR21	$\overline{CS7}$/ ADDR20	$\overline{CS6}$/ ADDR19
1	1	1	1	1	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	$\overline{CS6}$
1	1	1	1	0	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	ADDR19
1	1	1	0	X	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{CS10}$	$\overline{CS9}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{CS10}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

D.2.18 Chip-Select Base Address Register Boot ROM

CSBARBT — Chip-Select Base Address Register Boot ROM **$YFFA48**

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	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

D.2.19 Chip-Select Base Address Registers

CSBAR[0:10] — Chip-Select Base Address Registers **$YFFA4C–$YFFA74**

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	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each chip-select pin has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip select. CSBARBT contains the base address for selection of a bootstrap memory device. Bit and field definitions for CSBARBT and CSBAR[0:10] are the same, but reset block sizes differ.

ADDR[23:11] — Base Address

This field sets the starting address of a particular chip-select's address space. The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be an integer multiple of the block size. Base address register diagrams show how base register bits correspond to address lines.

BLKSZ[2:0] — Block Size Field

This field determines the size of the block that is enabled by the chip-select.

Table D-13 shows bit encoding for the base address registers block size field.

$\overline{\text{BOOT}}$— Boot ROM Control

Reset state of $\overline{\text{BOOT}}$ is specified at mask time. Bootstrap operation is overridden if STOP = 1 at reset. This is a read-only bit.

0 = ROM responds to bootstrap word locations during reset vector fetch.

1 = ROM does not respond to bootstrap word locations during reset vector fetch.

LOCK — Lock Registers

The reset state of LOCK is specified at mask time. If the reset state of the LOCK is zero, it can be set once after reset to allow protection of the registers after initialization. Once the LOCK bit is set, it cannot be cleared again until after a reset. LOCK protects the ASPC and WAIT fields, as well as the ROMBAL and ROMBAH registers. ASPC, ROMBAL and ROMBAH are also protected by the STOP bit.

0 = Write lock disabled. Protected registers and fields can be written.

1 = Write lock enabled. Protected registers and fields cannot be written.

EMUL — Emulation Mode Control

0 = Normal ROM operation

The MC68376 does not support emulation mode, therefore, this bit reads zero. Writes have no effect.

ASPC[1:0] — ROM Array Space

ASPC can be written only if LOCK = 0 and STOP = 1. ASPC1 places the ROM array in either supervisor or unrestricted space. ASPC0 determines if the array resides in program space only or with program and data space. The reset state of ASPC[1:0] is specified at mask time. **Table D-22** shows ASPC[1:0] encoding.

Table D-22 ROM Array Space Field

ASPC[1:0]	State Specified
00	Unrestricted program and data
01	Unrestricted program
10	Supervisor program and data
11	Supervisor program

WAIT[1:0] — Wait States

WAIT[1:0] specifies the number of wait states inserted by the MRM during ROM array accesses. The reset state of WAIT[1:0] is specified at mask time. WAIT[1:0] can be written only if LOCK = 0 and STOP = 1. **Table D-23** shows WAIT[1:0] encoding.

Table D-23 Wait States Field

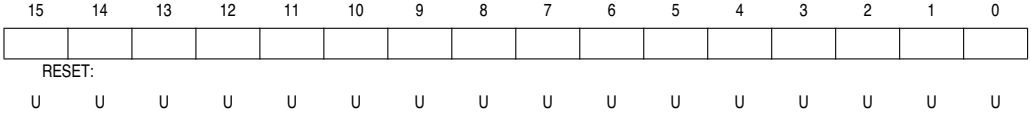
WAIT[1:0]	Cycles per Transfer
00	3
01	4
10	5
11	2

Table D-47 DASMA Operations

Mode	DASMA Operation
DIS	DASMA can be accessed to prepare a value for a subsequent mode selection
IPWM	DASMA contains the captured value corresponding to the trailing edge of the measured pulse
IPM	DASMA contains the captured value corresponding to the most recently detected user-specified rising or falling edge
IC	DASMA contains the captured value corresponding to the most recently detected user-specified rising or falling edge
OCB	DASMA is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to DASMA in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
OCAB	DASMA is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to DASMA in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
OPWM	DASMA is loaded with the value corresponding to the leading edge of the PWM pulse to be generated.

D.7.13 DASM Data Register B

DASM3B — DASM3 Data Register B **$YFF41C**
DASM4B — DASM4 Data Register B **$YFF424**
DASM9B — DASM9 Data Register B **$YFF44C**
DASM10B — DASM10 Data Register B **$YFF454**



DASMB is the data register associated with channel B. **Table D-48** shows how DASMB is used with the different modes of operation. Depending on the mode selected, software access is to register B1 or register B2.