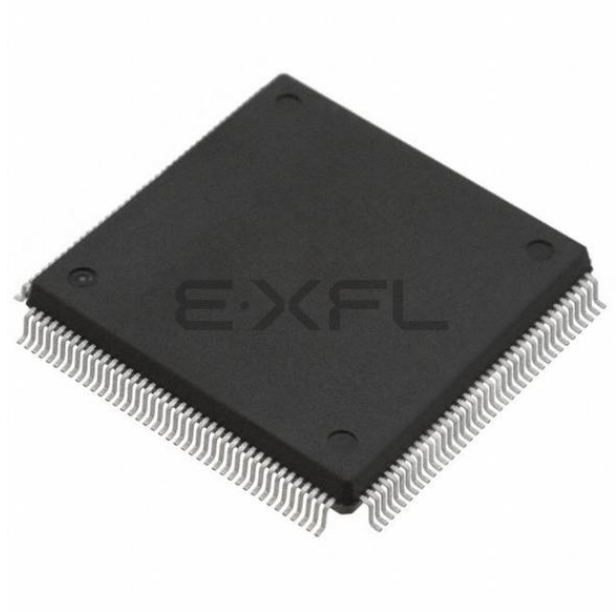

Welcome to E-XFL.COM

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 ~ 85°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/mc68376bacab20

TABLE OF CONTENTS
(Continued)

$\overline{\text{DSACK}}$[1:0] — Data and Size Acknowledge
DSCLK — Development Serial Clock
DSI — Development Serial Input
DSO — Development Serial Output
ECLK — MC6800 Devices and Peripherals Bus Clock
ETRIG[2:1] — QADC External Trigger
EXTAL — Crystal Oscillator Input
FC[2:0] — Function Codes
FREEZE — Freeze
$\overline{\text{HALT}}$ — Halt
$\overline{\text{IFETCH}}$ — Instruction Fetch
$\overline{\text{IPIPE}}$ — Instruction Pipeline
$\overline{\text{IRQ}}$[7:1] — Interrupt Request
MA[2:0] — QADC Multiplexed Address
MISO — QSM Master In Slave Out
MODCLK — Clock Mode Select
MOSI — QSM Master Out Slave In
PCS[3:0] — QSM Peripheral Chip-Selects
PQA[7:0] — QADC Port A
PQB[7:0] — QADC Port B
PC[6:0] — SIM Port C
PE[7:0] — SIM Port E
PF[7:0] — SIM Port F
QUOT — Quotient Out
R/$\overline{\text{W}}$ — Read/Write
$\overline{\text{RESET}}$ — Reset
$\overline{\text{RMC}}$ — Read-Modify-Write Cycle
RXD — SCI Receive Data
SCK — QSPI Serial Clock
SIZ[1:0] — Size
$\overline{\text{SS}}$ — Slave Select
T2CLK — TPU Clock In
TPUCH[15:0] — TPU Channel Signals
TSC — Three-State Control
$\overline{\text{TSTME}}$ — Test Mode Enable
V_{RH} — QADC High Reference Voltage
V_{RL} — QADC Low Reference Voltage
XFC — External Filter Capacitor
XTAL — Crystal Oscillator Output

When the MCU completes a bus cycle with the HALT signal asserted, DATA[15:0] is placed in a high-impedance state and bus control signals are driven inactive; the address, function code, size, and read/write signals remain in the same state. If HALT is still asserted once bus mastership is returned to the MCU, the address, function code, size, and read/write signals are again driven to their previous states. The MCU does not service interrupt requests while it is halted. Refer to **5.6.5 Bus Exception Control Cycles** for more information.

5.5.1.11 Autovector Signal

The autovector signal (AVEC) can be used to terminate external interrupt acknowledge cycles. Assertion of AVEC causes the CPU32 to generate vector numbers to locate an interrupt handler routine. If AVEC is continuously asserted, autovectors are generated for all external interrupt requests. AVEC is ignored during all other bus cycles. Refer to **5.8 Interrupts** for more information. AVEC for external interrupt requests can also be supplied internally by chip-select logic. Refer to **5.9 Chip-Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **5.6.6 External Bus Arbitration** for more information.

5.5.2 Dynamic Bus Sizing

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

During an operand transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the DSACK inputs, as shown in **Table 5-11**. Chip-select logic can generate data and size acknowledge signals for an external device. Refer to **5.9 Chip-Selects** for more information.

Table 5-11 Effect of DSACK Signals

DSACK1	DSACK0	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle — Data Bus Port Size is 8 Bits
0	1	Complete Cycle — Data Bus Port Size is 16 Bits
0	0	Reserved

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

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

1. XTRST (external reset) drives the external reset pin.
2. CLKRST (clock reset) resets the clock module.
3. MSTRST (master reset) goes to all other internal circuits.
4. SYSRST (system reset) indicates to internal circuits that the CPU32 has executed a RESET instruction.

All resets are gated by CLKOUT. Resets are classified as synchronous or asynchronous. An asynchronous reset can occur on any CLKOUT edge. Reset sources that cause an asynchronous reset usually indicate a catastrophic failure. As a result, the reset control logic responds by asserting reset to the system immediately. (A system reset, however, caused by the CPU32 RESET instruction, is asynchronous but does not indicate any type of catastrophic failure).

Synchronous resets are timed to occur at the end of bus cycles. The SIM bus monitor is automatically enabled for synchronous resets. When a bus cycle does not terminate normally, the bus monitor terminates it.

Refer to **Table 5-14** for a summary of reset sources.

Table 5-14 Reset Source Summary

Type	Source	Timing	Cause	Reset Lines Asserted by Controller		
External	External	Synch	$\overline{\text{RESET}}$ pin	MSTRST	CLKRST	EXTRST
Power up	EBI	Asynch	V_{DD}	MSTRST	CLKRST	EXTRST
Software watchdog	Monitor	Asynch	Time out	MSTRST	CLKRST	EXTRST
$\overline{\text{HALT}}$	Monitor	Asynch	Internal $\overline{\text{HALT}}$ assertion (e.g. double bus fault)	MSTRST	CLKRST	EXTRST
Loss of clock	Clock	Synch	Loss of reference	MSTRST	CLKRST	EXTRST
Test	Test	Synch	Test mode	MSTRST	—	EXTRST
System	CPU32	Asynch	RESET instruction	—	—	EXTRST

Internal single byte or aligned word writes are guaranteed valid for synchronous resets. External writes are also guaranteed to complete, provided the external configuration logic on the data bus is conditioned as shown in **Figure 5-16**.

5.7.3 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. **Table 5-15** is a summary of reset mode selection options.

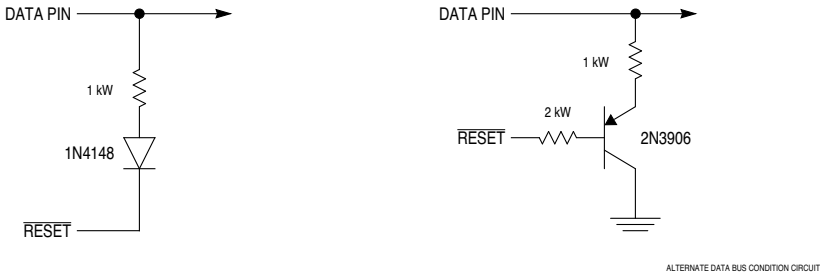

Figure 5-17 Alternate Circuit for Data Bus Mode Select Conditioning

Data bus mode select current is specified in **Table A-5**. Do not confuse pin function with pin electrical state. Refer to **5.7.5 Pin States During Reset** for more information.

Unlike other chip-select signals, the boot ROM chip-select ($\overline{\text{CSBOOT}}$) is active at the release of $\overline{\text{RESET}}$. During reset exception processing, the MCU fetches initialization vectors beginning at address $000000 in supervisor program space. An external memory device containing vectors located at these addresses can be enabled by $\overline{\text{CSBOOT}}$ after a reset.

The logic level of DATA0 during reset selects boot ROM port size for dynamic bus allocation. When DATA0 is held low, port size is eight bits; when DATA0 is held high, either by the weak internal pull-up driver or by an external pull-up, port size is 16 bits. Refer to **5.9.4 Chip-Select Reset Operation** for more information.

DATA1 and DATA2 determine the functions of $\overline{\text{CS}}$[2:0] and $\overline{\text{CS}}$[5:3], respectively. DATA[7:3] determine the functions of an associated chip-select and all lower-numbered chip-selects down through $\overline{\text{CS6}}$. For example, if DATA5 is pulled low during reset, $\overline{\text{CS}}$[8:6] are assigned alternate function as ADDR[21:19], and $\overline{\text{CS}}$[10:9] remain chip-selects. Refer to **5.9.4 Chip-Select Reset Operation** for more information.

DATA8 determines the function of the $\overline{\text{DSACK}}$[1:0], $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, and SIZE pins. If DATA8 is held low during reset, these pins are assigned to I/O port E.

DATA9 determines the function of interrupt request pins $\overline{\text{IRQ}}$[7:1] and the clock mode select pin (MODCLK). When DATA9 is held low during reset, these pins are assigned to I/O port F.

5.7.3.2 Clock Mode Selection

The state of the clock mode (MODCLK) pin during reset determines what clock source the MCU uses. When MODCLK is held high during reset, the clock signal is generated from a reference frequency using the clock synthesizer. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. Refer to **5.3 System Clock** for more information.

NOTE

The MODCLK pin can also be used as parallel I/O pin PF0. To prevent inadvertent clock mode selection by logic connected to port F, use an active device to drive MODCLK during reset.

5.7.3.3 Breakpoint Mode Selection

Background debug mode (BDM) is enabled when the breakpoint ($\overline{\text{BKPT}}$) pin is sampled at a logic level zero at the release of $\overline{\text{RESET}}$. Subsequent assertion of the $\overline{\text{BKPT}}$ pin or the internal breakpoint signal (for instance, the execution of the CPU32 BKPT instruction) will place the CPU32 in BDM.

If $\overline{\text{BKPT}}$ is sampled at a logic level one at the rising edge of $\overline{\text{RESET}}$, BDM is disabled. Assertion of the $\overline{\text{BKPT}}$ pin or execution of the execution of the BKPT instruction will result in normal breakpoint exception processing.

BDM remains enabled until the next system reset. $\overline{\text{BKPT}}$ is relatched on each rising transition of $\overline{\text{RESET}}$. $\overline{\text{BKPT}}$ is internally synchronized and must be held low for at least two clock cycles prior to $\overline{\text{RESET}}$ negation for BDM to be enabled. $\overline{\text{BKPT}}$ assertion logic must be designed with special care. If $\overline{\text{BKPT}}$ assertion extends into the first bus cycle following the release of $\overline{\text{RESET}}$, the bus cycle could inadvertently be tagged with a breakpoint.

Refer to **4.10.2 Background Debug Mode** and the *CPU32 Reference Manual* (CPU32RM/AD) for more information on background debug mode. Refer to the *SIM Reference Manual* (SIMRM/AD) and **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information concerning BKPT signal timing.

5.7.4 MCU Module Pin Function During Reset

Usually, module pins default to port functions and input/output ports are set to the 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. **Table 5-16** is a summary of module pin function out of reset.

At the release of reset, the exception vector table is located beginning at address $000000. This value can be changed by programming the vector base register (VBR) with a new value. Multiple vector tables can be used. Refer to **4.9 Exception Processing** for more information.

5.8.2 Interrupt Priority and Recognition

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

NOTE

Exceptions such as “address error” are not interrupts and have no “level” associated. Exceptions cannot ever be masked.

There are seven interrupt request signals ($\overline{\text{IRQ}}$[7:1]). These signals are used internally on the IMB, and have corresponding pins for external interrupt service requests. The CPU32 treats all interrupt requests as though they come from internal modules; external interrupt requests are treated as interrupt service requests from the SIM. Each of the interrupt request signals corresponds to an interrupt priority. $\overline{\text{IRQ1}}$ has the lowest priority and $\overline{\text{IRQ7}}$ the highest.

Interrupt recognition is determined by interrupt priority level and interrupt priority (IP) mask value. The interrupt priority mask consists of three bits in the CPU32 status register. 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 from being recognized and processed. $\overline{\text{IRQ7}}$, however, is always recognized, even if the mask value is %111.

$\overline{\text{IRQ}}$[7:1] are active-low level-sensitive inputs. The low on the pin must remain asserted until an interrupt acknowledge cycle corresponding to that level is detected.

$\overline{\text{IRQ7}}$ is transition-sensitive as well as level-sensitive: a level-7 interrupt is not detected unless a falling edge transition is detected on the $\overline{\text{IRQ7}}$ line. This prevents redundant servicing and stack overflow. A non-maskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted as well as each time the priority mask is written while $\overline{\text{IRQ7}}$ is asserted. If $\overline{\text{IRQ7}}$ is asserted and the IP mask is written to any new value (including %111), $\overline{\text{IRQ7}}$ will be recognized as a new $\overline{\text{IRQ7}}$.

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 interrupts is complete.

The SCI provides a standard non-return to zero (NRZ) mark/space format. It operates in either full- or half-duplex mode. There are separate transmitter and receiver enable bits and dual data buffers. A modulus-type baud rate generator provides rates from 110 baud to 655 kbaud with a 20.97 MHz system clock. Word length of either eight or nine bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wake-up functions allow the CPU32 to run uninterrupted until meaningful data is available.

9.2 QSM Registers and Address Map

There are four types of QSM registers: QSM global registers, QSM pin control registers, QSPI registers, and SCI registers. Refer to **9.2.1 QSM Global Registers** and **9.2.2 QSM Pin Control Registers** for a discussion of global and pin control registers. Refer to **9.3.1 QSPI Registers** and **9.4.1 SCI Registers** for further information about QSPI and SCI registers. Writes to unimplemented register bits have no effect, and reads of unimplemented bits always return zero.

The QSM address map includes the QSM registers and the QSPI RAM. The MM bit in the system integration module configuration register (SIMCR) defines the most significant bit (ADDR23) of the IMB address for each module.

Refer to **D.6 Queued Serial Module** for a QSM address map and register bit and field definitions. **5.2.1 Module Mapping** contains more information about how the state of MM affects the system.

9.2.1 QSM Global Registers

The QSM configuration register (QSMCR) contains parameters for interfacing to the CPU32 and the intermodule bus. The QSM test register (QTEST) is used during factory test of the QSM. The QSM interrupt level register (QILR) determines the priority of interrupts requested by the QSM and the vector used when an interrupt is acknowledged. The QSM interrupt vector register (QIVR) contains the interrupt vector for both QSM submodules. QILR and QIVR are 8-bit registers located at the same word address.

9.2.1.1 Low-Power Stop Operation

When the STOP bit in QSMCR is set, the system clock input to the QSM is disabled and the module enters a low-power operating state. QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable during LPSTOP. However, writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be set by the CPU32 and by reset.

System software must bring the QSPI and SCI to an orderly stop before asserting STOP to avoid data corruption. The IRQ mask level in the CPU32 status register should be set to a higher value than the IRQ level generated by the QSM module. The SCI receiver and transmitter should be disabled after transfers in progress are complete. The QSPI can be halted by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set. The IRQ mask in the CPU status register should be restored to its former level. Refer to **5.3.4 Low-Power Operation** for more information about low-power stop mode.

- The pause bit is set in CCW0A and EOQ is programmed into CCW0A.
- During queue 1 operation, the pause bit is set in CCW20, which is also BQ2.

8.12.3 Scan Modes

The QADC queuing mechanism provides several methods for automatically scanning input channels. In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed. In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue are executed. The following paragraphs describe the disabled/reserved, single-scan, and continuous-scan operations.

8.12.3.1 Disabled Mode and Reserved Mode

When the disabled mode or a reserved mode is selected, the queue is not active.

NOTE

Do not use a reserved mode. Unspecified operations may result.

Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, no wait states will be inserted by the QADC for accesses to the CCW and result word tables. When both queues are disabled, it is safe to change the QADC clock prescaler values.

8.12.3.2 Single-Scan Modes

When application software requires execution of a single pass through a sequence of conversions defined by a queue, a single-scan queue operating mode is selected.

In all single-scan queue operating modes, software must enable a queue for execution by writing the single-scan enable bit to one in the queue's control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2, respectively.

Until the single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to one during the write cycle that selects the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero but is always read as a zero.

After the single-scan enable bit is set, a trigger event causes the QADC to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue scan is complete; the QADC then clears the single-scan enable bit to zero. If the single-scan enable bit is written to one or zero before the queue scan is complete, the queue is not affected. However, if software changes the queue operating mode, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The current conversion is aborted and the new queue operating mode takes effect.

By properly programming the MQ1 field in QACR1 or the MQ2 field in QACR2, the following modes can be selected for queue 1 and/or 2:

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

11.3.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. The time needed to respond to and service an event is determined by which channels and the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU performance in a given application. Latency can be closely estimated. For more information, refer to the *TPU Reference Manual* (TPURM/AD)

11.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU channels contain identical hardware and are functionally equivalent in operation, so that any channel can be configured to perform any time function. Any function can operate on the calling channel, and, under program control, on another channel determined by the program or by a parameter. The user controls the combination of time functions.

11.3.3 Interchannel Communication

The autonomy of the TPU is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU32 intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

11.3.4 Programmable Channel Service Priority

The TPU provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

11.3.5 Coherency

For data to be coherent, all available portions of the data must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take

Arbitration is performed by means of serial assertion of IARB field bit values. The IARB of TPUMCR is initialized to $0 during reset.

When the TPU wins arbitration, it must respond to the CPU32 interrupt acknowledge cycle by placing an interrupt vector number on the data bus. The vector number is used to calculate displacement into the exception vector table. Vectors are formed by concatenating the 4-bit value of the CIBV field in TICR with the 4-bit number of the channel requesting interrupt service. Since the CIBV field has a reset value of $0, it must be assigned a value corresponding to the upper nibble of a block of 16 user-defined vector numbers before TPU interrupts are enabled. Otherwise, a TPU interrupt service request could cause the CPU32 to take one of the reserved vectors in the exception vector table.

For more information about the exception vector table, refer to **4.9 Exception Processing**. Refer to **5.8 Interrupts** for further information about interrupts.

11.4 A Mask Set Time Functions

The following paragraphs describe factory-programmed time functions implemented in the A mask set TPU microcode ROM. A complete description of the functions is beyond the scope of this manual. Refer to the *TPU Reference Manual* (TPURM/AD) for additional information.

11.4.1 Discrete Input/Output (DIO)

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on. The programmer can choose one of the three following conditions to update the parameter: 1) when a transition occurs, 2) when the CPU32 makes a request, or 3) when a rate specified in another parameter is matched. When a pin is used as a discrete output, it is set high or low only upon request by the CPU32.

Refer to TPU programming note *Discrete Input/Output (DIO) TPU Function* (TPUPN18/D) for more information.

11.4.2 Input Capture/Input Transition Counter (ITC)

Any channel of the TPU can capture the value of a specified TCR upon the occurrence of each transition or specified number of transitions and then generate an interrupt request to notify the CPU32. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, then cease channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and the number of channels within the block. The generation of links depends on the mode of operation. In addition, after each transition or specified number of transitions, one byte of the parameter RAM (at an address specified by channel parameter) can be incremented and used as a flag to notify another channel of a transition.

12.7 Low-Power Stop Operation

Setting the STOP bit in TRAMMCR places the TPURAM in low-power stop mode. In low-power stop mode, the array retains its contents, but cannot be read or written by the CPU32. STOP can be written only when the processor is operating in supervisor mode. STOP is set during resets. Low-power stop mode is exited by clearing STOP.

The TPURAM module will switch to standby mode while it is in low-power mode, provided the operating constraints discussed above are met.

12.8 Reset

Reset places the TPURAM in low-power stop mode, enables supervisor mode access only, clears the base address register, and disables the array. These actions make it possible to write a new base address into the base address register.

When a synchronous reset occurs while a byte or word TPURAM access is in progress, the access is completed. If reset occurs during the first word access of a long-word operation, only the first word access is completed. If reset occurs during the second word access of a long-word operation, the entire access is completed. Data being read from or written to the TPURAM may be corrupted by asynchronous reset. Refer to **5.7 Reset** for more information concerning resets.

12.9 TPU Microcode Emulation

The TPURAM array can emulate the microcode ROM in the TPU module. This provides a means for developing custom TPU code. The TPU selects TPU emulation mode.

The TPU is connected to the TPURAM via a dedicated bus. While the TPURAM array is in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microcode ROM to ensure accurate emulation. Normal accesses through the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses. Refer to **SECTION 11 TIME PROCESSOR UNIT** and to the *TPU Reference Manual* (TPURM/AD) for more information.

- Cascade usage of TX error counter with an additional internal counter to detect the 128 occurrences of 11 consecutive recessive bits necessary to transition from bus off into error active.

Both counters are read only (except in test/freeze/halt modes).

The TouCAN responds to any bus state as described in the CAN protocol, transmitting an error active or error passive flag, delaying its transmission start time (error passive) and avoiding any influence on the bus when in the bus off state. The following are the basic rules for TouCAN bus state transitions:

- If the value of the TX error counter or RX error counter increments to a value greater than or equal to 128, the fault confinement state (FCS[1:0]) field in the error status register is updated to reflect an error passive state.
- If the TouCAN is in an error passive state, and either the TX error counter or RX error counter decrements to a value less than or equal to 127, while the other error counter already satisfies this condition, the FCS[1:0] field in the error status register is updated to reflect an error active state.
- If the value of the TX error counter increases to a value greater than 255, the FCS[1:0] field in the error status register is updated to reflect a bus off state, and an interrupt may be issued. The value of the TX error counter is reset to zero.
- If the TouCAN is in the bus off state, the TX error counter and an additional internal counter are cascaded to count 128 occurrences of 11 consecutive recessive bits on the bus. To do this, the TX error counter is first reset to zero, then the internal counter begins counting consecutive recessive bits. Each time the internal counter counts 11 consecutive recessive bits, the TX error counter is incremented by one and the internal counter is reset to zero. When the TX error counter reaches the value of 128, the FCS[1:0] field in the error status register is updated to be error active, and both error counters are reset to zero. Any time a dominant bit is detected following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero, but does not affect the TX error counter value.
- If only one node is operating in a system, the TX error counter will increment with each message it attempts to transmit, due to the resulting acknowledgment errors. However, acknowledgment errors will never cause the TouCAN to transition from the error passive state to the bus off state.
- If the RX error counter increments to a value greater than 127, it will stop incrementing, even if more errors are detected while being a receiver. After the next successful message reception, the counter is reset to a value between 119 and 127, to enable a return to the error active state.

13.4.5 Time Stamp

The value of the free-running 16-bit timer is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp will be stored in the time stamp entry of the receive message buffer at the time the message is written into that buffer. For a message being transmitted, the time stamp entry will be written into the transmit message buffer once the transmission has completed successfully.

NOTES:

1. Applies to :
   Port E[7:4] — SIZ[1:0], $\overline{AS}$, $\overline{DS}$
   Port F[7:0] — $\overline{IRQ[7:1]}$, MODCLK
   Port QS[7:0] — TXD, PCS[3:1], PCS0/$\overline{SS}$, SCK, MOSI, MISO
   TPUCH[15:0], T2CLK, CPWM[8:5], CTD[4:3], CTD[10:9], CTM2C
   $\overline{BKPT}$/DSCLK, $\overline{IFETCH}$, $\overline{RESET}$, RXD, $\overline{TSTME}$/TSC
   EXTAL (when PLL enabled)
2. Input-Only Pins: EXTAL, $\overline{TSTME}$/TSC, $\overline{BKPT}$, PAI, T2CLK, RXD, CTM2C
   Output-Only Pins: $\overline{CSBOOT}$, $\overline{BG}$/$\overline{CS}$, CLKOUT, FREEZE/QUOT, $\overline{IPIPE}$
   Input/Output Pins:
   Group 1: DATA[15:0], $\overline{IFETCH}$, TPUCH[15:0], CPWM[8:5], CTD[4:3], CTD[10:9]
   Group 2: Port C[6:0] — ADDR[22:19]/$\overline{CS[9:6]}$, FC[2:0]/$\overline{CS[5:3]}$
   Port E[7:0] — SIZ[1:0], $\overline{AS}$, $\overline{DS}$, $\overline{AVEC}$, $\overline{RMC}$, $\overline{DSACK[1:0]}$
   Port F[7:0] — $\overline{IRQ[7:1]}$, MODCLK
   Port QS[7:3] — TXD, PCS[3:1], PCS0/$\overline{SS}$
   ADDR23/$\overline{CS10}$/ECLK, ADDR[18:0], R/$\overline{W}$, $\overline{BERR}$, $\overline{BR}$/$\overline{CS0}$, $\overline{BGACK}$/$\overline{CS2}$
   Group 3: $\overline{HALT}$, $\overline{RESET}$
   Group 4: MISO, MOSI, SCK
   Pin groups do not include QADC pins. See **Tables A-11** through **A-14** for information concerning the QADC.
3. Does not apply to $\overline{HALT}$ and $\overline{RESET}$ because they are open drain pins. Does not apply to port QS[7:0] (TXD, PCS[3:1], PCS0/$\overline{SS}$, SCK, MOSI, MISO) in wired-OR mode.
4. Use of an active pulldown device is recommended.
5. Total operating current is the sum of the appropriate I_{DD}, I_{DDSYN}, and I_{SB} values. I_{DD} values include supply currents for device modules powered by V_{DDE} and V_{DDI} pins.
6. Current measured at maximum system clock frequency, all modules active.
7. The SRAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 volts. The SRAM array cannot be accessed while the module is in standby mode.
8. When V_{DD} is transitioning during power-up or power down sequence, and V_{SB} is applied, current flows between the V_{STBY} and V_{DD} pins, which causes standby current to increase toward the maximum transient condition specification. System noise on the V_{DD} and V_{STBY} pins can contribute to this condition.
9. Power dissipation measured at system clock frequency, all modules active. Power dissipation can be calculated using the following expression:
   $$P_D = \text{Maximum } V_{DD} (\text{Run } I_{DD} + I_{DDSYN} + I_{SB}) + \text{Maximum } V_{DDA} (I_{DDA})$$
10. This parameter is periodically sampled rather than 100% tested.

Table A-6 AC Timing (Continued)

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	10	—	ns
27	Data In Valid to Clock Low (Data Setup)[5]	t_{DICL}	5	—	ns
27A	Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	15	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	60	ns
29	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[8]	t_{SNDI}	0	—	ns
29A	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[8, 9]	t_{SHDI}	—	48	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold)[8]	t_{CLDI}	10	—	ns
30A	CLKOUT Low to Data In High Impedance[8]	t_{CLDH}	—	72	ns
31	$\overline{DSACK[1:0]}$ Asserted to Data In Valid[10]	t_{DADI}	—	46	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	23	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)[11]	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
39	$\overline{BG}$ Width Negated	t_{GH}	2	—	t_{cyc}
39A	$\overline{BG}$ Width Asserted	t_{GA}	1	—	t_{cyc}
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	115	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	70	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	12	—	ns
48	$\overline{DSACK[1:0]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[12]	t_{DABA}	—	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	23	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RADC}	32	—	ns
56	$\overline{RESET}$ Pulse Width (Reset Instruction)	t_{HRPW}	512	—	t_{cyc}
57	$\overline{BERR}$ Negated to $\overline{HALT}$ Negated (Rerun)	t_{BNHN}	0	—	ns
70	Clock Low to Data Bus Driven (Show)	t_{SCLDD}	0	23	ns
71	Data Setup Time to Clock Low (Show)	t_{SCLDS}	10	—	ns
72	Data Hold from Clock Low (Show)	t_{SCLDH}	10	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	10	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}

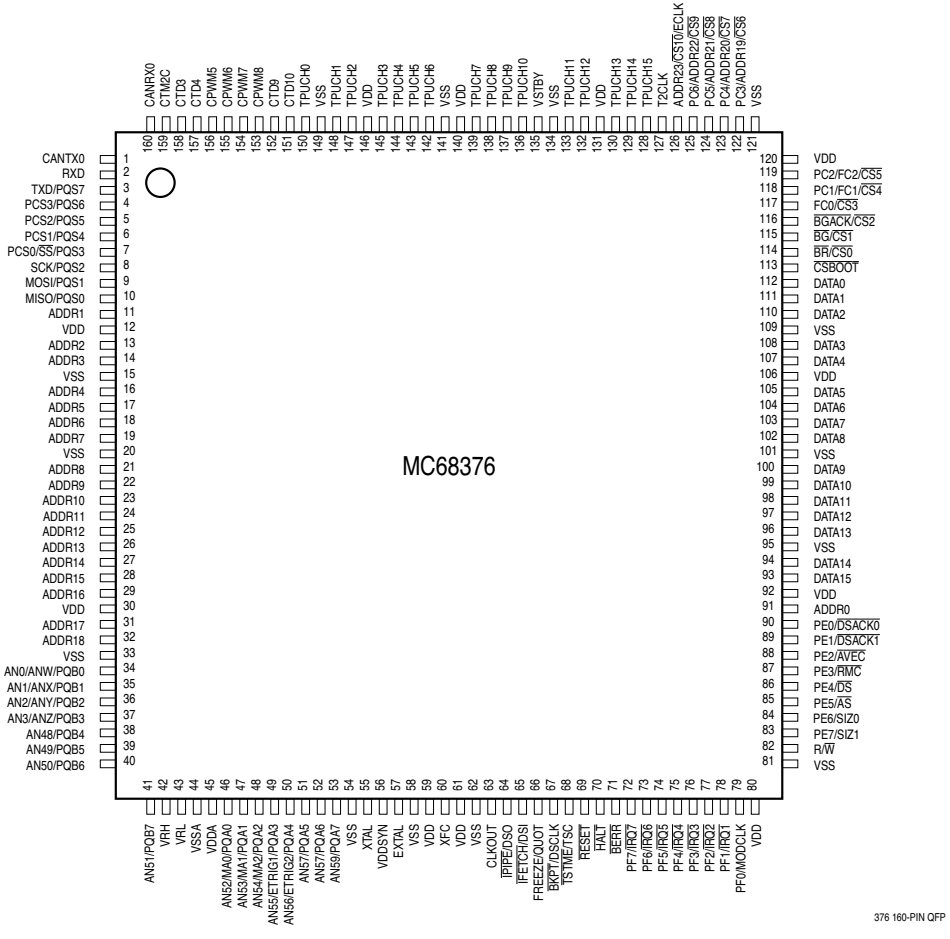

Figure B-2 MC68376 Pin Assignments for 160-Pin Package

D.3.2 RAM Test Register

RAMTST — RAM Test Register **$YFFB42**

Used for factory test only.

D.3.3 Array Base Address Register High

RAMBAH — Array Base Address Register High **$YFFB44**

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

D.3.4 Array Base Address Register Low

RAMBAL — Array Base Address Register Low **$YFFB46**

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

RAMBAH and RAMBAL specify the SRAM array base address in the system memory map. They can only be written while the SRAM is in low-power stop mode (STOP = 1, the default out of reset) and the base address lock is disabled (RLCK = 0, the default out of reset). This prevents accidental remapping of the array.

Table D-35 CTM4 Address Map (Continued)

Address[1]	15 0
$YFF45A	MCSM11 Counter (MCSM11CNT)
$YFF45C	MCSM11 Modulus Latch (MCSM11ML)
$YFF45E	Reserved
$YFF460	FCSM12 Status/Interrupt/Control Register (FCSM12SIC)
$YFF462	FCSM12 Counter (FCSM12CNT)
$YFF464 – $YFF4FE	Reserved

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

D.7.1 BIU Module Configuration Register

BIUMCR — BIU Module Configuration Register **$YFF400**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	NOT USED	VECT[7:6]		IARB[2:0]			NOT USED		TBRS1	NOT USED				TBRS0
RESET:															
0	0		1	1	0	0	0			0					0

STOP — Low-Power Stop Mode Enable

When the STOP bit is set, the clock to the CTM4 is shutdown, placing the module into low-power stop mode. The BIUSM still operates in low-power stop mode, allowing the submodule control and data registers to be accessed.

0 = Enable CTM4 clocks.
1 = Disable CTM4 clocks.

FRZ — FREEZE Assertion Response

The FRZ bit controls CTM4 response to assertion of the IMB FREEZE signal. Since the BIUSM propagates FREEZE to the CTM4 submodules via the submodule bus, the setting of FRZ affects all CTM4 submodules.

0 = CTM4 ignores the IMB FREEZE signal.
1 = CTM4 submodules freeze when the IMB FREEZE signal is asserted.

VECT[7:6] — Interrupt Vector Base Number

This bit field selects the base interrupt vector number for the CTM4. Of the eight bits necessary for a vector number, the six low-order bits are hardware defined on a sub-module basis, while the two remaining bits are provided by VECT[7:6]. This places the CTM4 vectors in one of four possible positions in the interrupt vector table. Refer to **Table D-36**.

Table D-36 Interrupt Vector Base Number Bit Field

VECT7	VECT6	Resulting Base Vector Number
0	0	$00
0	1	$40
1	0	$80
1	1	$C0

–U–

–V–

–W–

–X–