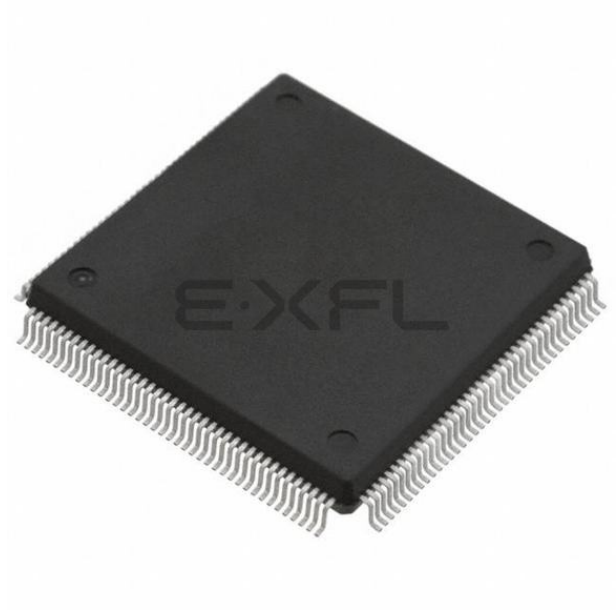

Welcome to E-XFL.COM

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 ~ 105°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/mc68376bavab20

SECTION 1 INTRODUCTION

The MC68336 and the MC68376 are highly-integrated 32-bit microcontrollers, combining high-performance data manipulation capabilities with powerful peripheral subsystems.

MC68300 microcontrollers are built up from standard modules that interface through a common intermodule bus (IMB). Standardization facilitates rapid development of devices tailored for specific applications.

The MC68336 incorporates a 32-bit CPU (CPU32), a system integration module (SIM), a time processor unit (TPU), a configurable timer module (CTM4), a queued serial module (QSM), a 10-bit queued analog-to-digital converter module (QADC), a 3.5-Kbyte TPU emulation RAM module (TPURAM), and a 4-Kbyte standby RAM module (SRAM).

The MC68376 includes all of the aforementioned modules, plus a CAN 2.0B protocol controller module (TouCAN™) and an 8-Kbyte masked ROM (MRM).

The MC68336/376 can either synthesize the system clock signal from a fast reference or use an external clock input directly. Operation with a 4.194 MHz reference frequency is standard. The maximum system clock speed is 20.97 MHz. System hardware and software allow changes in clock rate during operation. Because MCU operation is fully static, register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of the MCU low. Power consumption can be minimized by stopping the system clock. The CPU32 instruction set includes a low-power stop (LPSTOP) instruction that efficiently implements this capability.

Documentation for the Modular Microcontroller Family follows the modular construction of the devices in the product line. Each microcontroller has a comprehensive user's manual that provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Refer to Motorola publication *Advanced Microcontroller Unit (AMCU) Literature* (BR1116/D) for a complete listing of documentation.

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.

The exception processing state is associated with interrupts, trap instructions, tracing, and other exception conditions. The exception may be internally generated explicitly by an instruction or by an unusual condition arising during the execution of an instruction. Exception processing can be forced externally by an interrupt, a bus error, or a reset.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts.

The background processing state is initiated by breakpoints, execution of special instructions, or a double bus fault. Background processing is enabled by pulling $\overline{\text{BKPT}}$ low during $\overline{\text{RESET}}$. Background processing allows interactive debugging of the system via a simple serial interface.

4.7 Privilege Levels

The processor operates at one of two levels of privilege: user or supervisor. Not all instructions are permitted to execute at the user level, but all instructions are available at the supervisor level. Effective use of privilege level can protect system resources from uncontrolled access. The state of the S bit in the status register determines the privilege level and whether the user stack pointer (USP) or supervisor stack pointer (SSP) is used for stack operations.

4.8 Instructions

The CPU32 instruction set is summarized in **Table 4-2**. The instruction set of the CPU32 is very similar to that of the MC68020. Two new instructions have been added to facilitate controller applications: low-power stop (LPSTOP) and table lookup and interpolate (TBLS, TBLSN, TBLU, TBLUN).

Table 4-1 shows the MC68020 instructions that are not implemented on the CPU32.

Table 4-1 Unimplemented MC68020 Instructions

BFxx	—	Bit Field Instructions (BFCHG, BFCLR, BFEXTS, BFEXTU, BFFFO, BFINS, BFSET, BFTST)
CALLM, RTM	—	Call Module, Return Module
CAS, CAS2	—	Compare and Swap (Read-Modify-Write Instructions)
cpxxx	—	Coprocessor Instructions (cpBcc, cpDBcc, cpGEN)
PACK, UNPK	—	Pack, Unpack BCD Instructions
Memory	—	Memory Indirect Addressing Modes

The CPU32 traps on unimplemented instructions or illegal effective addressing modes, allowing user-supplied code to emulate unimplemented capabilities or to define special purpose functions. However, Motorola reserves the right to use all currently unimplemented instruction operation codes for future M68000 core enhancements.

4.8.2.3 Loop Mode Instruction Execution

The CPU32 has several features that provide efficient execution of program loops. One of these features is the DBcc looping primitive instruction. To increase the performance of the CPU32, a loop mode has been added to the processor. The loop mode is used by any single word instruction that does not change the program flow. Loop mode is implemented in conjunction with the DBcc instruction. **Figure 4-7** shows the required form of an instruction loop for the processor to enter loop mode.

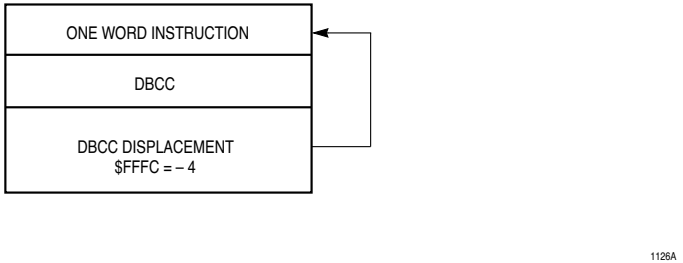

Figure 4-7 Loop Mode Instruction Sequence

The loop mode is entered when the DBcc instruction is executed, and the loop displacement is –4. Once in loop mode, the processor performs only the data cycles associated with the instruction and suppresses all instruction fetches. The termination condition and count are checked after each execution of the data operations of the looped instruction. The CPU32 automatically exits the loop mode on interrupts or other exceptions. All single word instructions that do not cause a change of flow can be looped.

4.9 Exception Processing

An exception is a special condition that preempts normal processing. Exception processing is the transition from normal mode program execution to execution of a routine that deals with an exception.

4.9.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. The vector base register (VBR) contains the base address of a 1024-byte exception vector table, which consists of 256 exception vectors. Sixty-four vectors are defined by the processor, and 192 vectors are reserved for user definition as interrupt vectors. Except for the reset vector, each vector in the table is one long word in length. The reset vector is two long words in length. Refer to **Table 4-3** for information on vector assignment.

CAUTION

Because there is no protection on the 64 processor-defined vectors, external devices can access vectors reserved for internal purposes. This practice is strongly discouraged.

5.3.3 External Bus Clock

The state of the E-clock division bit (EDIV) in SYNCR determines clock rate for the E-clock signal (ECLK) available on pin ADDR23. ECLK is a bus clock for M6800 devices and peripherals. ECLK frequency can be set to system clock frequency divided by eight or system clock frequency divided by sixteen. The clock is enabled by the $\overline{CS10}$ field in chip-select pin assignment register 1 (CSPAR1). ECLK operation during low-power stop is described in the following paragraph. Refer to **5.9 Chip-Selects** for more information about the external bus clock.

5.3.4 Low-Power Operation

Low-power operation is initiated by the CPU32. To reduce power consumption selectively, the CPU can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU can execute the LPSTOP instruction, which causes the SIM to turn off the system clock.

When individual module STOP bits are set, clock signals inside each module are turned off, but module registers are still accessible.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SIM brings the MCU out of low-power stop mode when one of the following exceptions occur:

- $\overline{RESET}$
- Trace
- SIM interrupt of higher priority than the stored interrupt mask

Refer to **5.6.4.2 LPSTOP Broadcast Cycle** and **4.8.2.1 Low-Power Stop (LPSTOP)** for more information.

During low-power stop mode, unless the system clock signal is supplied by an external source and that source is removed, the SIM clock control logic and the SIM clock signal (SIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{RESET}$ and $\overline{IRQ}$ pins are clocked by SIMCLK, and can be used to bring the processor out of LPSTOP. Optionally, the SIM can also continue to generate the CLKOUT signal while in low-power stop mode.

STSIM and STEXT bits in SYNCR determine clock operation during low-power stop mode.

The flowchart shown in **Figure 5-5** summarizes the effects of the STSIM and STEXT bits when the MCU enters normal low power stop mode. Any clock in the off state is held low. If the synthesizer VCO is turned off during low-power stop mode, there is a PLL relock delay after the VCO is turned back on.

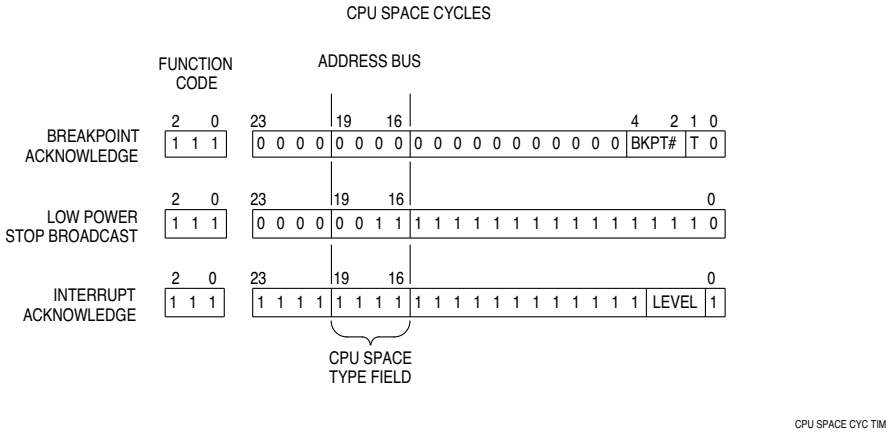

Figure 5-12 CPU Space Address Encoding

5.6.4.1 Breakpoint Acknowledge Cycle

Breakpoints stop program execution at a predefined point during system development. Breakpoints can be used alone or in conjunction with background debug mode. In M68300 microcontrollers, both hardware and software can initiate breakpoints.

The CPU32 $\overline{\text{BKPT}}$ instruction allows the user to insert breakpoints through software. The CPU responds to this instruction by initiating a breakpoint acknowledge read cycle in CPU space. It places the breakpoint acknowledge (%0000) code on ADDR[19:16], the breakpoint number (bits [2:0] of the BKPT opcode) on ADDR[4:2], and %0 (indicating a software breakpoint) on ADDR1.

External breakpoint circuitry decodes the function code and address lines and responds by either asserting $\overline{\text{BERR}}$ or placing an instruction word on the data bus and asserting $\overline{\text{DSACK}}$. If the bus cycle is terminated by $\overline{\text{DSACK}}$, the CPU32 reads the instruction on the data bus and inserts the instruction into the pipeline. (For 8-bit ports, this instruction fetch may require two read cycles.)

If the bus cycle is terminated by $\overline{\text{BERR}}$, the CPU32 then performs illegal instruction exception processing: it acquires the number of the illegal instruction exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address.

Assertion of the $\overline{\text{BKPT}}$ input initiates a hardware breakpoint. The CPU32 responds by initiating a breakpoint acknowledge read cycle in CPU space. It places the breakpoint acknowledge code of %0000 on ADDR[19:16], the breakpoint number value of %111 on ADDR[4:2], and ADDR1 is set to %1, indicating a hardware breakpoint.

5.7.9 Reset Processing Summary

To prevent write cycles in progress from being corrupted, a reset is recognized at the end of a bus cycle instead of at an instruction boundary. Any processing in progress at the time a reset occurs is aborted. After SIM reset control logic has synchronized an internal or external reset request, the MSTRST signal is asserted.

The following events take place when MSTRST is asserted:

A. Instruction execution is aborted.
B. The status register is initialized.
    1. The T0 and T1 bits are cleared to disable tracing.
    2. The S bit is set to establish supervisor privilege level.
    3. The interrupt priority mask is set to $7, disabling all interrupts below priority 7.
C. The vector base register is initialized to $000000.

The following events take place when MSTRST is negated after assertion.

A. The CPU32 samples the $\overline{\text{BKPT}}$ input.
B. The CPU32 fetches the reset vector:
    1. The first long word of the vector is loaded into the interrupt stack pointer.
    2. The second long word of the vector is loaded into the program counter.
    3. Vectors can be fetched from external ROM enabled by the $\overline{\text{CSBOOT}}$ signal.
C. The CPU32 fetches and begins decoding the first instruction to be executed.

5.7.10 Reset Status Register

The reset status register (RSR) contains a bit for each reset source in the MCU. When a reset occurs, a bit corresponding to the reset type is set. When multiple causes of reset occur at the same time, only one bit in RSR may be set. The reset status register is updated by the reset control logic when the $\overline{\text{RESET}}$ signal is released. Refer to **D.2.4 Reset Status Register** for more information.

5.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the SIM, the CPU32, and a device or module requesting interrupt service.

The following paragraphs provide an overview of the entire interrupt process. Chip-select logic can also be used to terminate the IACK cycle with either $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$. Refer to **5.9 Chip-Selects** for more information.

5.8.1 Interrupt Exception Processing

The CPU32 processes interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Each exception has an assigned vector in an exception vector table that points to an associated handler routine. The CPU32 uses vector numbers to calculate displacement into the table. During exception processing, the CPU fetches the appropriate vector and executes the exception handler routine to which the vector points.

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.

9.3.2.3 Command RAM

Command RAM is used by the QSPI in master mode. The CPU32 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).

9.3.3 QSPI Pins

The QSPI uses seven pins. These pins can be configured for general-purpose I/O when not needed for QSPI application.

Table 9-2 shows QSPI input and output pins and their functions.

Table 9-2 QSPI Pins

Pin Names	Mnemonics	Mode	Function
Master In Slave Out	MISO	Master Slave	Serial data input to QSPI Serial data output from QSPI
Master Out Slave In	MOSI	Master Slave	Serial data output from QSPI Serial data input to QSPI
Serial Clock	SCK	Master Slave	Clock output from QSPI Clock input to QSPI
Peripheral Chip Selects	PCS[3:1]	Master	Select peripherals
Slave Select	PCS0/$\overline{SS}$	Master Master Slave	Selects peripherals Causes mode fault Initiates serial transfer

9.3.4 QSPI Operation

The QSPI uses a dedicated 80-byte block of static RAM accessible by both the QSPI and the CPU32 to perform queued operations. The RAM is divided into three segments. There are 16 command bytes, 16 transmit data words, and 16 receive data words. QSPI RAM is organized so that one byte of command data, one word of transmit data, and one word of receive data correspond to one queue entry, $0–$F.

The CPU32 initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU32 or waits for intervention.

There are four queue pointers. The CPU32 can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), contained in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

data space accesses. The SUPV bit in QADCMCR designates the assignable space as supervisor or unrestricted.

Attempts to read supervisor-only data space when the CPU32 is not in supervisor mode causes a value of $0000 to be returned. Attempts to read assignable data space when the CPU32 is not in supervisor mode and when the space is programmed as supervisor space, causes a value of $FFFF to be returned. Attempts to write supervisor-only or supervisor-assigned data space when the CPU32 is in user mode has no effect.

The supervisor-only data space segment contains the QADC global registers, which include QADCMCR, QADCTEST, and QADCINT. The supervisor/unrestricted space designation for the CCW table, the result word table, and the remaining QADC registers is programmable. Refer to **D.5.1 QADC Module Configuration Register** for more information.

8.6.4 Interrupt Arbitration Priority

Each module that can request interrupts, including the QADC, has an interrupt arbitration number (IARB) field in its module configuration register. Each IARB field must have a different non-zero value. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level.

The reset value of IARB in the QADCMCR is $0. Initialization software must set the IARB field to a non-zero value in order for QADC interrupts to be arbitrated. Refer to **D.5.1 QADC Module Configuration Register** for more information.

8.7 Test Register

The QADC test register (QADCTEST) is used only during factory testing of the MCU.

8.8 General-Purpose I/O Port Operation

QADC port pins, when used as general-purpose input, are conditioned by a synchronizer with an enable feature. The synchronizer is not enabled until the QADC decodes an IMB bus cycle which addresses the port data register to minimize the high-current effect of mid-level signals on the inputs used for analog signals. Digital input signals must meet the input low voltage (V_{IL}) or input high voltage (V_{IH}) specifications in **APPENDIX A ELECTRICAL CHARACTERISTICS**. If an analog input pin does not meet the digital input pin specifications when a digital port read operation occurs, an indeterminate state is read.

During a port data register read, the actual value of the pin is reported when its corresponding bit in the data direction register defines the pin to be an input (port A only). When the data direction bit specifies the pin to be an output, the content of the port data register is read. By reading the latch which drives the output pin, software instructions that read data, modify it, and write the result, like bit manipulation instructions, work correctly.

6. Calculate the QCLK frequency (f_{QCLK}).

$$\frac{(f_{QCLK}\ (\text{in MHz}) = 1000)}{\text{QCLK high time (in ns)} + \text{QCLK low time (in ns)}}$$

7. Choose the number of input sample cycles (2, 4, 8, or 16) for a typical input channel.
8. If the calculated conversion times are not sufficient, return to step 3. Conversion time is determined by the following equation:

$$\text{Conversion time (in } \mu s) = \frac{16 + \text{Number of input sample cycles}}{f_{QCLK}\text{(in MHz)}}$$

9. Code the selected PSH, PSL, and PSA values into the prescaler fields of QACR0.

Figure 8-9 and **Table 8-4** show examples of QCLK programmability. The examples include conversion times based on the following assumptions:

- f_{sys} = 20.97 MHz.
- Input sample time is as fast as possible (IST[1:0] = %00, 2 QCLK cycles).

Figure 8-9 and **Table 8-4** also show the conversion time calculated for a single conversion in a queue. For other MCU system clock frequencies and other input sample times, the same calculations can be made.

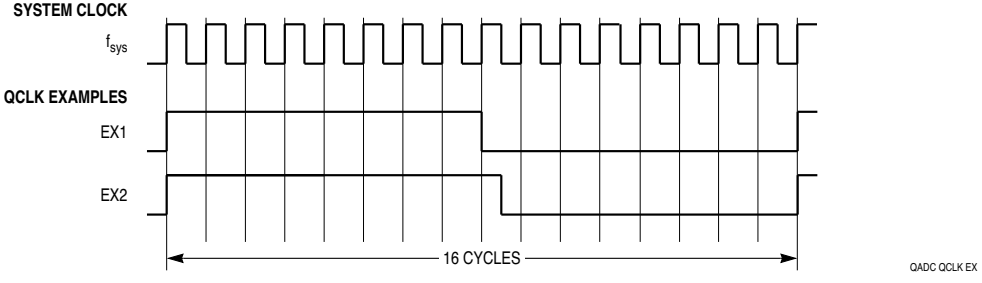

Figure 8-9 QADC Clock Programmability Examples

ator, though internally, channel B has two data registers (B1 and B2). DASM operating mode determines which register is software accessible. Refer to **Table 10-3**.

Table 10-3 Channel B Data Register Access

Mode	Data Register
Input Capture (IPWM, IPM, IC)	Registers A and B2 are used to hold the captured values. In these modes, the B1 register is used as a temporary latch for channel B.
Output Compare (OCA, OCAB)	Registers A and B2 are used to define the output pulse. Register B1 is not used in these modes.
Output Pulse Width Modulation Mode (OPWM)	Registers A and B1 are used as primary registers and hidden register B2 is used as a double buffer for channel B.

Register contents are always transferred automatically at the correct time so that the minimum pulse (measured or generated) is just one time base bus count. The A and B data registers are always read/write registers, accessible via the CTM4 submodule bus.

The CTM4 has four DASMs. **Figure 10-5** shows a block diagram of the DASM.

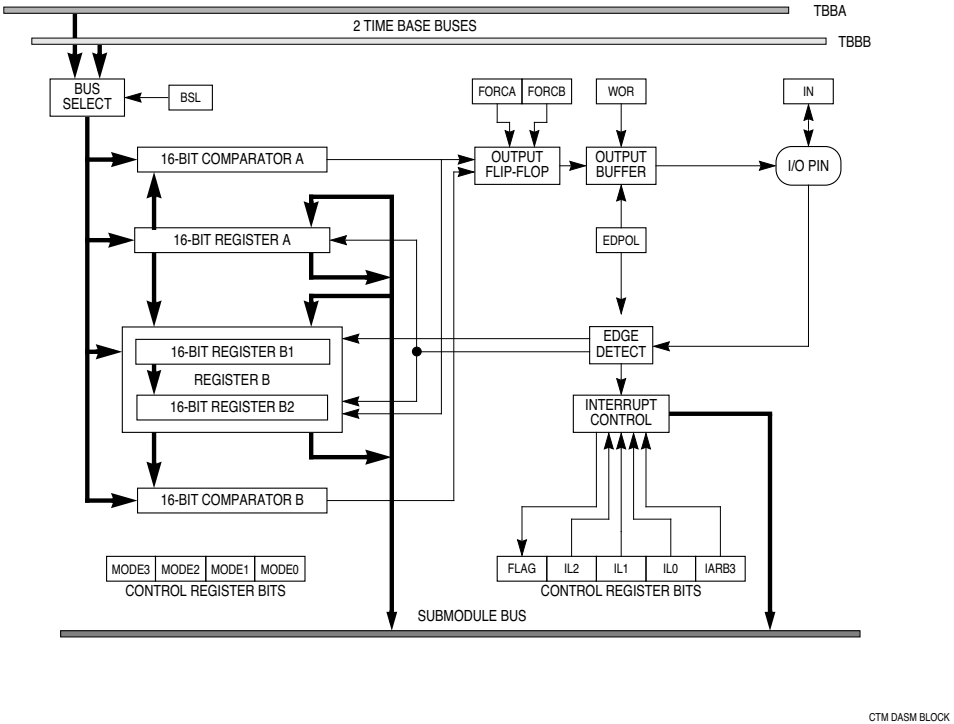

Figure 10-5 DASM Block Diagram

TCR1 have the capability to resolve down to the TPU system clock divided by 4. Refer to **Figure 11-2** and **Table 11-1**.

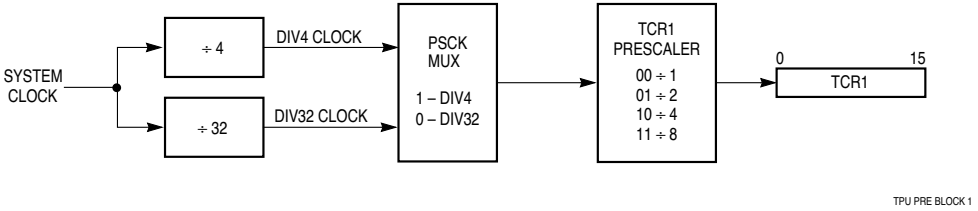

Figure 11-2 TCR1 Prescaler Control

Table 11-1 TCR1 Prescaler Control

		PSCK = 0		PSCK = 1	
TCR1 Prescaler	**Divide By**	**Number of Clocks**	**Rate at 20.97 MHz**	**Number of Clocks**	**Rate at 20.97 MHz**
00	1	32	1.6 µs	4	200 ns
01	2	64	3.2 µs	8	400 ns
10	4	128	6.4 µs	16	0.8 µs
11	8	256	12.8 µs	32	1.6 µs

11.6.1.2 Prescaler Control for TCR2

Timer count register two (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG bit in TPUMCR determines whether the T2CLK pin functions as an external clock source for TCR2 or as the gate in the use of TCR2 as a gated pulse accumulator. The function of the T2CG bit is shown in **Figure 11-3**.

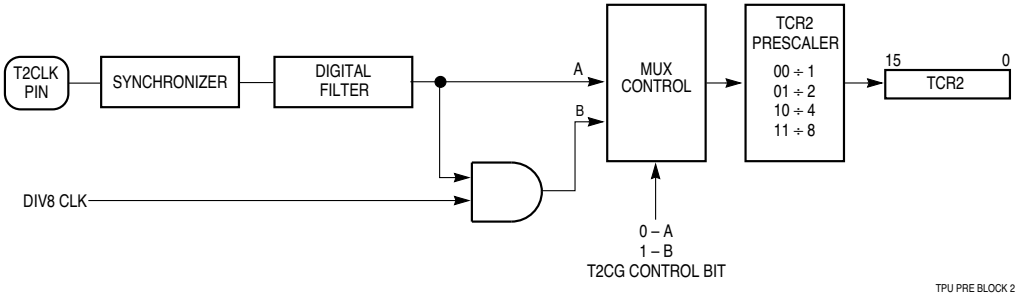

Figure 11-3 TCR2 Prescaler Control

bit in the matching transmit message buffer is set, the TouCAN will transmit a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission.

13.5.6 Overload Frames

Overload frame transmissions are not initiated by the TouCAN unless certain conditions are detected on the CAN bus. These conditions include:

- Detection of a dominant bit in the first or second bit of intermission.
- Detection of a dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames.
- Detection of a dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter.

13.6 Special Operating Modes

The TouCAN module has three special operating modes:

- Debug mode
- Low-power stop mode
- Auto power save mode

13.6.1 Debug Mode

Debug mode is entered by setting the HALT bit in the CANMCR, or by assertion of the IMB FREEZE line. In both cases, the FRZ1 bit in CANMCR must also be set to allow HALT or FREEZE to place the TouCAN in debug mode.

Once entry into debug mode is requested, the TouCAN waits until an intermission or idle condition exists on the CAN bus, or until the TouCAN enters the error passive or bus off state. Once one of these conditions exists, the TouCAN waits for the completion of all internal activity. When this happens, the following events occur:

- The TouCAN stops transmitting/receiving frames.
- The prescaler is disabled, thus halting all CAN bus communication.
- The TouCAN ignores its RX pins and drives its TX pins as recessive.
- The TouCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR are set.
- The CPU32 is allowed to read and write the error counter registers.

After engaging one of the mechanisms to place the TouCAN in debug mode, the user must wait for the FRZACK bit to be set before accessing any other registers in the TouCAN, otherwise unpredictable operation may occur.

To exit debug mode, the IMB FREEZE line must be negated or the HALT bit in CANMCR must be cleared.

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
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{\text{RESET}}$ Assertion Time[13]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{\text{RESET}}$ Rise Time[14, 15]	t_{RSTR}	—	10	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. The base configuration of the MC68336/376 requires a 20.97 MHz crystal reference.
3. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period is reduced when the duty cycle of the external clock signal varies. The relationship between external clock input duty cycle and minimum t_{Xcyc} is expressed:
   Minimum t_{Xcyc} period = minimum t_{XCHL} / (50% –external clock input duty cycle tolerance).
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal. If transitions occur within the correct clock period, rise/fall times and duty cycle are not critical.
5. Address access time = (2.5 + WS) $t_{cyc} - t_{CHAV} - t_{DICL}$
   Chip select access time = (2 + WS) $t_{cyc} - t_{LSA} - t_{DICL}$
   Where: WS = number of wait states. When fast termination is used (2 clock bus) WS = –1.
6. Specification 9A is the worst-case skew between $\overline{\text{AS}}$ and $\overline{\text{DS}}$ or $\overline{\text{CS}}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{\text{AS}}$ and $\overline{\text{DS}}$ to fall outside the limits shown in specification 9.
7. If multiple chip selects are used, $\overline{\text{CS}}$ width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{\text{CS}}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
8. Hold times are specified with respect to $\overline{\text{DS}}$ or $\overline{\text{CS}}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
9. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
10. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{\text{DSACK}}$[1:0] low to data setup time (specification 31) and $\overline{\text{DSACK}}$[1:0] low to $\overline{\text{BERR}}$ low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle. $\overline{\text{BERR}}$ must satisfy only the late $\overline{\text{BERR}}$ low to clock low setup time (specification 27A) for the following clock cycle.
11. To ensure coherency during every operand transfer, $\overline{\text{BG}}$ will not be asserted in response to $\overline{\text{BR}}$ until after all cycles of the current operand transfer are complete and $\overline{\text{RMC}}$ is negated.
12. In the absence of $\overline{\text{DSACK}}$[1:0], $\overline{\text{BERR}}$ is an asynchronous input using the asynchronous setup time (specification 47A).
13. After external $\overline{\text{RESET}}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SIM drives $\overline{\text{RESET}}$ low for 512 t_{cyc}.
14. External assertion of the $\overline{\text{RESET}}$ input can overlap internally-generated resets. To insure that an external reset is recognized in all cases, $\overline{\text{RESET}}$ must be asserted for at least 590 CLKOUT cycles.
15. External logic must pull $\overline{\text{RESET}}$ high during this period in order for normal MCU operation to begin.

M — Mode Select
0 = 10-bit SCI frame
1 = 11-bit SCI frame

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 disabled.
1 = SCI TDRE interrupts enabled.

TCIE — Transmit Complete Interrupt Enable
0 = SCI TC interrupts disabled.
1 = SCI TC interrupts enabled.

RIE — Receiver Interrupt Enable
0 = SCI RDRF and OR interrupts disabled.
1 = SCI RDRF and OR interrupts enabled.

ILIE — Idle-Line Interrupt Enable
0 = SCI IDLE interrupts disabled.
1 = SCI IDLE interrupts enabled.

TE — Transmitter Enable
0 = SCI transmitter disabled (TXD pin can be used as I/O).
1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter).

RE — Receiver Enable
0 = SCI receiver disabled.
1 = SCI receiver enabled.

RWU — Receiver Wakeup
0 = Normal receiver operation (received data recognized).
1 = Wakeup mode enabled (received data ignored until receiver is awakened).

SBK — Send Break
0 = Normal operation
1 = Break frame(s) transmitted after completion of current frame.

FE — Framing Error

0 = No framing error detected in the received data.

1 = Framing error or break detected in the received data.

PF — Parity Error

0 = No parity error detected in the received data.

1 = Parity error detected in the received data.

D.6.8 SCI Data Register

SCDR — SCI Data Register **$YFFC0E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED							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 consists of two data registers located at the same address. The receive data register (RDR) is a read-only register that contains data received by the SCI serial interface. Data comes into the receive serial shifter and is transferred to RDR. The transmit data register (TDR) is a write-only register that contains data to be transmitted. 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 nine-bit operation. When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect.

D.6.9 Port QS Data Register

PORTQS — Port QS Data Register **$YFFC15**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PQS7	PQS6	PQS5	PQS4	PQS3	PQS2	PQS1	PQS0
RESET															
								0	0	0	0	0	0	0	0

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

ENDQP[3:0] — Ending Queue Pointer
This field contains the last QSPI queue address.

Bits [7:4] — Not Implemented

NEWQP[3:0] — New Queue Pointer Value
This field contains the first QSPI queue address.

D.6.14 QSPI Control Register 3

SPCR3 — QSPI Control Register **$YFFC1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	LOOPQ	HMIE	HALT	SPSR							
RESET:															
0	0	0	0	0	0	0	0								

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enable, and the halt control bit. The CPU32 has read/write access to SPCR3, but the QSM has read access only. SPCR3 must be initialized before QSPI operation begins. Writing a new value to SPCR3 while the QSPI is enabled disrupts operation.

Bits [15:11] — Not Implemented

LOOPQ — QSPI Loop Mode
0 = Feedback path disabled.
1 = Feedback path enabled.
LOOPQ controls feedback on the data serializer for testing.

HMIE — HALTA and MODF Interrupt Enable
0 = HALTA and MODF interrupts disabled.
1 = HALTA and MODF interrupts enabled.
HMIE enables interrupt requests generated by the HALTA status flag or the MODF status flag in SPSR.

HALT — Halt QSPI
0 = QSPI operates normally.
1 = QSPI is halted for subsequent restart.
When HALT is set, the QSPI stops on a queue boundary. It remains in a defined state from which it can later be restarted.

CHBK — Channel Register Breakpoint Flag

CHBK is asserted if a breakpoint occurs because of a CHAN register match with the CHAN register breakpoint register. CHBK is negated when the BKPT flag is cleared.

SRBK — Service Request Breakpoint Flag

SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is cleared.

TPUF — TPU FREEZE Flag

TPUF is set whenever the TPU is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU exits the halted state because of FREEZE being negated.

D.8.5 TPU Interrupt Configuration Register

TICR — TPU Interrupt Configuration Register **$YFFE08**

15	10	9	8	7	6	5	4	3	0
NOT USED	CIRL[2:0]			CIBV[3:0]				NOT USED	
RESET:									
	0	0	0	0	0	0	0		

CIRL[2:0] — Channel Interrupt Request Level

This three-bit field specifies the interrupt request level for all channels. Level seven for this field indicates a non-maskable interrupt; level zero indicates that all channel interrupts are disabled.

CIBV[3:0] — Channel Interrupt Base Vector

The TPU is assigned 16 unique interrupt vector numbers, one vector number for each channel. The CIBV field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers. The lower nibble of the TPU interrupt vector number is determined by the channel number on which the interrupt occurs.

D.8.6 Channel Interrupt Enable Register

CIER — Channel Interrupt Enable Register **$YFFE0A**

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	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] — Channel Interrupt Enable/Disable

0 = Channel interrupts disabled

1 = Channel interrupts enabled

ADDR[23:11] — TPURAM Array Base Address

These bits specify ADDR[23:12] of the base address of the TPURAM array when enabled. The 3.5-Kbyte array resides at the lower end of the 4-Kbyte page into which it is mapped.

RAMDS — RAM Array Disable

0 = RAM array is enabled.

1 = RAM array is disabled.

RAMDS indicates whether the TPURAM is active or disabled. The array is disabled at reset. Writing a valid base address into TRAMBAR clears the RAMDS bit and enables the array.