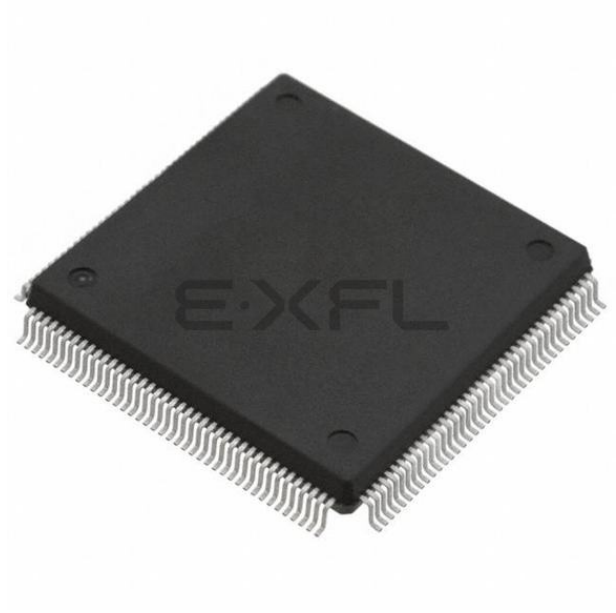

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Not For New Designs
Core Processor	CPU32
Core Size	32-Bit Single-Core
Speed	25MHz
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/sc68376bacab25

TABLE OF CONTENTS
(Continued)

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.

Table 3-5 MC68336/376 Signal Functions (Continued)

Mnemonic	Signal Name	Function
PQA[7:0]	QADC Port A	QADC port A digital input/output port signals
PQB[7:0]	QADC Port B	QADC port B digital input port signals
PQS[7:0]	Port QS	QSM digital input/output port signals
QUOT	Quotient Out	Provides the quotient bit of the polynomial divider (test mode only)
R/$\overline{W}$	Read/Write	Indicates the direction of data transfer on the bus
$\overline{RESET}$	Reset	System reset
$\overline{RMC}$	Read-Modify-Write Cycle	Indicates an indivisible read-modify-write instruction
RXD	SCI Receive Data	Serial input to the SCI
SCK	QSPI Serial Clock	Clock output from QSPI in master mode; clock input to QSPI in slave mode
SIZ[1:0]	Size	Indicates the number of bytes remaining to be transferred during a bus cycle
$\overline{SS}$	Slave Select	Starts serial transmission when QSPI is in slave mode; chip-select in master mode
T2CLK	TPU Clock	TPU clock input
TPUCH[15:0]	TPU I/O Channels	Bidirectional TPU channels
TSC	Three-State Control	Places all output drivers in a high impedance state
$\overline{TSTME}$	Test Mode Enable	Hardware enable for SIM test mode
TXD	SCI Transmit Data	Serial output from the SCI
XFC	External Filter Capacitor	Connection for external phase-locked loop filter capacitor

SECTION 4 CENTRAL PROCESSOR UNIT

The CPU32, the instruction processing module of the M68300 family, is based on the industry-standard MC68000 processor. It has many features of the MC68010 and MC68020, as well as unique features suited for high-performance controller applications. This section is an overview of the CPU32. For detailed information concerning CPU operation, refer to the *CPU32 Reference Manual* (CPU32RM/AD).

4.1 General

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction format reflects a philosophy emphasizing register-memory interaction. There are eight multifunction data registers and seven general-purpose addressing registers.

All data resources are available to all operations requiring those resources. The data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long-word) operand lengths for all operations. Word and long-word operations support address manipulation. Although the program counter (PC) and stack pointers (SP) are special-purpose registers, they are also available for most data addressing activities. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

A block diagram of the CPU32 is shown in **Figure 4-1**. The major blocks operate in a highly independent fashion that maximizes concurrency of operation while managing the essential synchronization of instruction execution and bus operation. The bus controller loads instructions from the data bus into the decode unit. The sequencer and control unit provide overall chip control, managing the internal buses, registers, and functions of the execution unit.

5.4.8 Low-Power STOP Mode Operation

When the CPU32 executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, internal clocks are disabled according to the state of the STSIM bit in the SYNCR, and the MCU enters low-power stop mode. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low-power stop mode.

During low-power stop mode, the clock input to the software watchdog timer is disabled and the timer stops. The software watchdog begins to run again on the first rising clock edge after low-power stop mode ends. The watchdog is not reset by low-power stop mode. A service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run during LPSTOP. To stop the periodic interrupt timer, PITR must be loaded with a zero value before the LPSTOP instruction is executed. A PIT interrupt, or an external interrupt request, can bring the MCU out of low-power stop mode if it has a higher priority than the interrupt mask value stored in the clock control logic when low-power stop mode is initiated. LPSTOP can be terminated by a reset.

5.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. **Figure 5-8** shows a basic system with external memory and peripherals.

SIZ[1:0] signals reflect bus allocation during show cycles. Only the appropriate portion of the data bus is valid during the cycle. During a byte write to an internal address, the portion of the bus that represents the byte that is not written reflects internal bus conditions, and is indeterminate. During a byte write to an external address, the data multiplexer in the SIM causes the value of the byte that is written to be driven out on both bytes of the data bus.

5.7 Reset

Reset occurs when an active low logic level on the $\overline{\text{RESET}}$ pin is clocked into the SIM. The $\overline{\text{RESET}}$ input is synchronized to the system clock. If there is no clock when $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ is asserted.

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SIM determines whether a reset is valid, asserts control signals, performs basic system configuration and boot ROM selection based on hardware mode-select inputs, then passes control to the CPU32.

5.7.1 Reset Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing, and can be caused by internal or external events. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in the exception vector table. The exception vector table consists of 256 four-byte vectors and occupies 1024 bytes of address space. The exception vector table can be relocated in memory by changing its base address in the vector base register (VBR). The CPU32 uses vector numbers to calculate displacement into the table. Refer to **4.9 Exception Processing** for more information.

Reset is the highest-priority CPU32 exception. Unlike all other exceptions, a reset occurs at the end of a bus cycle, and not at an instruction boundary. Handling resets in this way prevents write cycles in progress at the time the reset signal is asserted from being corrupted. However, any processing in progress is aborted by the reset exception and cannot be restarted. Only essential reset tasks are performed during exception processing. Other initialization tasks must be accomplished by the exception handler routine. Refer to **5.7.9 Reset Processing Summary** for details on exception processing.

5.7.2 Reset Control Logic

SIM reset control logic determines the cause of a reset, synchronizes reset assertion if necessary to the completion of the current bus cycle, and asserts the appropriate reset lines. Reset control logic can drive four different internal signals:

E. After arbitration, the interrupt acknowledge cycle is completed in one of the following ways:
   1. When there is no contention (IARB = %0000), the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and the CPU32 generates the spurious interrupt vector number.
   2. The dominant interrupt source (external or internal) supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU32 acquires the vector number.
   3. The $\overline{\text{AVEC}}$ signal is asserted (the signal can be asserted by the dominant external interrupt source or the pin can be tied low), and the CPU32 generates an autovector number corresponding to interrupt priority.
   4. The bus monitor asserts $\overline{\text{BERR}}$ and the CPU32 generates the spurious interrupt vector number.

F. The vector number is converted to a vector address.
G. The content of the vector address is loaded into the PC and the processor transfers control to the exception handler routine.

5.8.5 Interrupt Acknowledge Bus Cycles

Interrupt acknowledge bus cycles are CPU32 space cycles that are generated during exception processing. For further information about the types of interrupt acknowledge bus cycles determined by $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$, refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** and the *SIM Reference Manual* (SIMRM/AD).

5.9 Chip-Selects

Typical microcontrollers require additional hardware to provide external chip-select and address decode signals. The MCU includes 12 programmable chip-select circuits that can provide 2 to 16 clock-cycle access to external memory and peripherals. Address block sizes of two Kbytes to one Mbyte can be selected. **Figure 5-19** is a diagram of a basic system that uses chip-selects.

tion to restart at the designated location. Reads of SPCR2 return the current value of the register, not of the buffer. Writing the same value into any control register except SPCR2 while the QSPI is enabled has no effect on QSPI operation.

9.3.1.2 Status Register

SPSR contains information concerning the current serial transmission. Only the QSPI can set the bits in this register. The CPU32 reads SPSR to obtain QSPI status information and writes SPSR to clear status flags.

9.3.2 QSPI RAM

The QSPI contains an 80-byte block of dual-port access static RAM that can be accessed by both the QSPI and the CPU32. The RAM is divided into three segments: receive data RAM, transmit data RAM, and command data RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored for transmission to an external device. Command control data defines transfer parameters. Refer to **Figure 9-3**, which shows RAM organization.

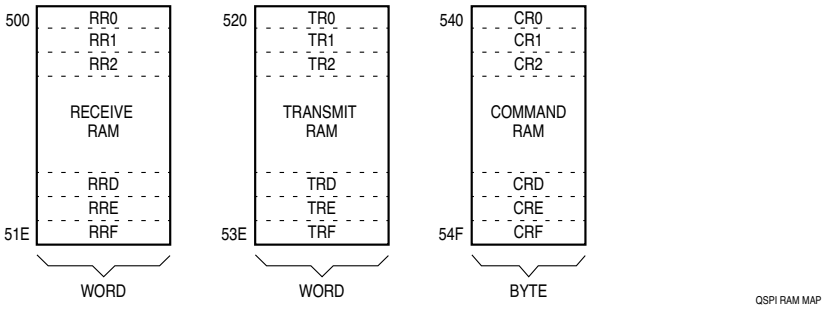

Figure 9-3 QSPI RAM

9.3.2.1 Receive RAM

Data received by the QSPI is stored in this segment. The CPU32 reads this segment to retrieve data from the QSPI. Data stored in the receive RAM is right-justified. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU32 can access the data using byte, word, or long-word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU32 uses this information to determine which locations in receive RAM contain valid data before reading them.

9.3.2.2 Transmit RAM

Data that is to be transmitted by the QSPI is stored in this segment and must be written to transmit RAM in a right-justified format. The QSPI cannot modify information in the transmit RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

Refer to TPU programming note *Table Stepper Motor (TSM) TPU Function* (TPUPN04/D) for more information.

11.5.2 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the CPU32. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

Refer to TPU programming note *New Input Capture/Transition Counter (NITC) TPU Function* (TPUPN08/D) for more information.

11.5.3 Queued Output Match (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin.

Refer to TPU programming note *Queued Output Match (QOM) TPU Function* (TPUPN01/D) for more information.

11.5.4 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The accumulation can start on a rising or falling edge. After the specified number of periods or pulses, PTA generates an interrupt request and optionally generates links to other channels.

From one to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU32, providing instantaneous or average frequency measurement capability, and the latest complete accumulation (over the programmed number of periods).

Refer to TPU programming note *Programmable Time Accumulator (PTA) TPU Function* (TPUPN06/D) for more information.

11.5.5 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

11.5.9 Frequency Measurement (FQM)

FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement; measurement of slow pulses with noise rejection can be made with PTA.

Refer to TPU programming note *Frequency Measurement (FQM) TPU Function* (TPUPN03/D) for more information.

11.5.10 Hall Effect Decode (HALLD)

This function decodes the sensor signals from a brushless motor, along with a direction input from the CPU32, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding option switches.

Refer to TPU programming note *Hall Effect Decode (HALLD) TPU Function* (TPUPN10/D) for more information.

11.6 Host Interface Registers

The TPU memory map contains three groups of registers:

- System configuration registers
- Channel control and status registers
- Development support and test verification registers

All registers except the channel interrupt status register (CISR) must be read or written by means of word accesses. The address space of the TPU memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

11.6.1 System Configuration Registers

The TPU configuration control registers, TPUMCR and TICR, determine the value of the prescaler, perform emulation control, specify whether the external TCR2 pin functions as a clock source or as gate of the DIV8 clock for TCR2, and determine interrupt request level and interrupt vector number assignment. Refer to **D.8.1 TPU Module Configuration Register** and **D.8.5 TPU Interrupt Configuration Register** for more information about TPUMCR and TICR.

11.6.1.1 Prescaler Control for TCR1

Timer count register one (TCR1) is clocked from the output of a prescaler. Two fields in TPUMCR control TCR1. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK bit. The prescaler divides this input by 1, 2, 4, or 8, depending on the value of TCR1P. Channels using

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}

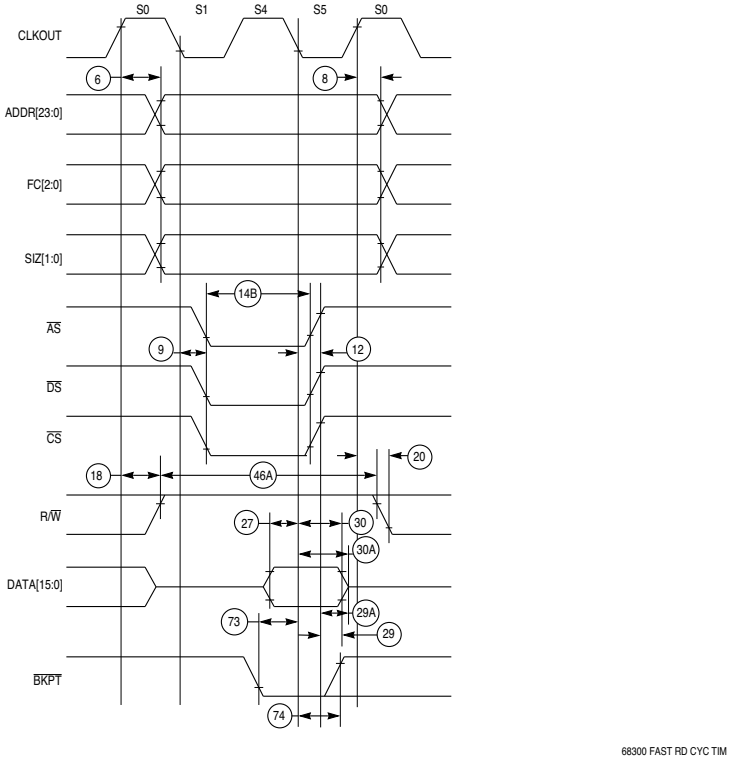

Figure A-6 Fast Termination Read Cycle Timing Diagram

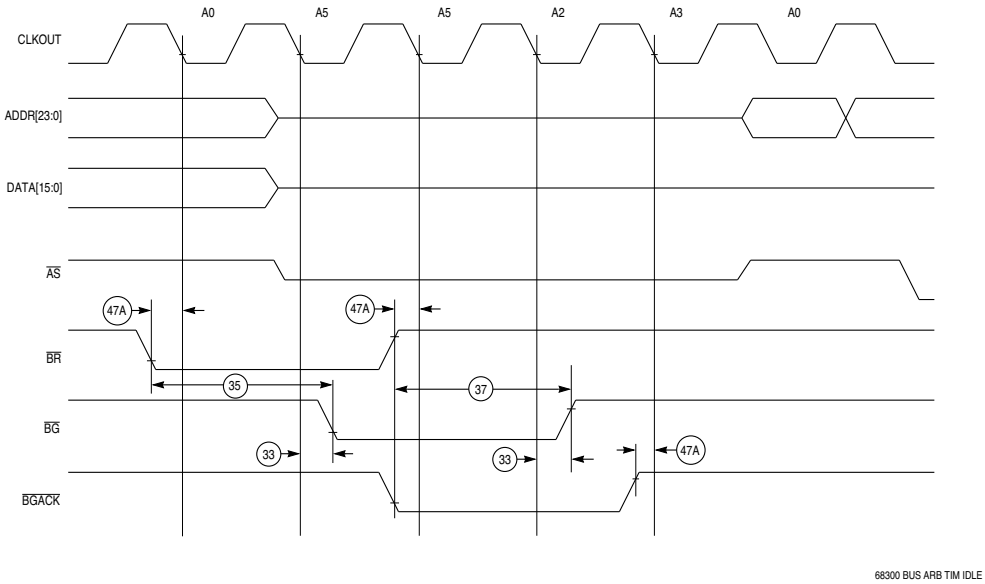

Figure A-9 Bus Arbitration Timing Diagram — Idle Bus Case

Table A-18 DASM Timing Characteristics

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

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin low time	t_{PINL}	$2.0/f_{sys}$	—	µs
2	Input pin high time	t_{PINH}	$2.0/f_{sys}$	—	µs
3	Input capture resolution[1]	t_{RESCA}	—	$2.0/f_{sys}$	µs
4	Pin to input capture delay	t_{PCAPT}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
5	Pin to FLAG set	t_{PFLAG}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
6	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
7	OCT output pulse	t_{OCT}	$2.0/f_{sys}$	—	µs
8	Compare resolution	t_{RESCM}	—	$2.0/f_{sys}$	µs
9	TBB change to FLAG set	t_{CFLAG}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
10	TBB change to pin change[2]	t_{CPIN}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
11	FLAG to IMB interrupt request	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs

NOTES:

1. Minimum resolution depends on counter and prescaler divide ratio selection.
2. Time given from when new value is stable on time base bus.

$\overline{\text{BOOT}}$— 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-26 Queue 2 Operating Modes

MQ2[4:0]	Queue 2 Operating Mode
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	External trigger rising edge single-scan mode (on ETRIG2 pin)
00011	External trigger falling edge single-scan mode (on ETRIG2 pin)
00100	Interval timer single-scan mode: interval = QCLK period x 2^7
00101	Interval timer single-scan mode: interval = QCLK period x 2^8
00110	Interval timer single-scan mode: interval = QCLK period x 2^9
00111	Interval timer single-scan mode: interval = QCLK period x 2^{10}
01000	Interval timer single-scan mode: interval = QCLK period x 2^{11}
01001	Interval timer single-scan mode: interval = QCLK period x 2^{12}
01010	Interval timer single-scan mode: interval = QCLK period x 2^{13}
01011	Interval timer single-scan mode: interval = QCLK period x 2^{14}
01100	Interval timer single-scan mode: interval = QCLK period x 2^{15}
01101	Interval timer single-scan mode: interval = QCLK period x 2^{16}
01110	Interval timer single-scan mode: interval = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Software triggered continuous-scan mode (started with SSE2)
10010	External trigger rising edge continuous-scan mode (on ETRIG2 pin)
10011	External trigger falling edge continuous-scan mode (on ETRIG2 pin)
10100	Periodic timer continuous-scan mode: period = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: period = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: period = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: period = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: period = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: period = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: period = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: period = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: period = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: period = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: period = QCLK period x 2^{17}
11111	Reserved mode

RES — Queue 2 Resume

RES selects the resumption point after queue 2 is suspended by queue 1. If RES is changed during execution of queue 2, the change is not recognized until an end-of-queue condition is reached, or the queue operating mode of queue 2 is changed.

0 = After suspension, begin execution with the first CCW in queue 2 or the current subqueue.

1 = After suspension, begin execution with the aborted CCW in queue 2.

Table D-48 DASMB Operations

Mode	DASMB Operation
DIS	DASMB can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value as register B1 is written.
IPWM	DASMB contains the captured value corresponding to the trailing edge of the measured pulse. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
IPM	DASMB contains the captured value corresponding to the most recently detected user-specified rising or falling edge. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
IC	DASMB contains the captured value corresponding to the most recently detected user-specified rising or falling edge. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OCB	DASMB is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to DASMB in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OCAB	DASMB is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to DASMB in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OPWM	DASMB is loaded with the value corresponding to the trailing edge of the PWM pulse to be generated. In this mode, register B1 is accessed. Buffer register B2 is hidden and cannot be accessed.

D.7.14 PWM Status/Interrupt/Control Register

PWM5SIC — PWM5 Status/Interrupt/Control Register **$YFF428**
PWM6SIC — PWM6 Status/Interrupt/Control Register **$YFF430**
PWM7SIC — PWM7 Status/Interrupt/Control Register **$YFF438**
PWM8SIC — PWM8 Status/Interrupt/Control Register **$YFF440**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	IL[2:0]			IARB3	NOT USED			PIN	NOT USED	LOAD	POL	EN	CLK[2:0]		
RESET:															
0	0	0	0	0				0		0	0	0	0	0	0

FLAG — Period Completion Status

This status bit indicates when the PWM output period has been completed.

0 = PWM period is not complete.
1 = PWM period is complete.

The FLAG bit is set each time a PWM period is completed. Whenever the PWM is enabled, the FLAG bit is set immediately to indicate that the contents of the buffer registers PWMA2 and PWMB2 have been updated, and that the period using these new values has started. It also indicates that the user accessible period and pulse width registers PWMA1 and PWMB1 can be loaded with values for the next PWM period. Once set, the FLAG bit remains set and is not affected by any subsequent period completions, until it is cleared.

Only software can clear the FLAG bit. To clear FLAG, first read the bit as one then write a zero to the bit. Writing a one to FLAG has no effect. When the PWM is disabled, FLAG remains cleared.

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

D.10.9 Receive Global Mask Registers

RXGMSKHI — Receive Global Mask Register High **$YFF090**
RXGMSKLO — Receive Global Mask Register Low **$YFF092**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
RESET:															
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

The receive global mask registers use four bytes. The mask bits are applied to all receive-identifiers, excluding receive-buffers 14-15, which have their own specific mask registers.

Base ID mask bits MID[28:18] are used to mask standard or extended format frames. Extended ID bits MID[17:0] are used to mask only extended format frames.

The RTR/SRR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. However, remote request frames (RTR = 1) once received, are never stored into the message buffers. RTR mask bit locations in the mask registers (bits 20 and 0) are always zero, regardless of any write to these bits.

The IDE bit of a received frame is always compared to determine if the message contains a standard or extended identifier. Its location in the mask registers (bit 19) is always one, regardless of any write to this bit.

D.10.10 Receive Buffer 14 Mask Registers

RX14MSKHI — Receive Buffer 14 Mask Register High **$YFF094**
RX14MSKLO — Receive Buffer 14 Mask Register Low **$YFF096**

The receive buffer 14 mask registers have the same structure as the receive global mask registers and are used to mask buffer 14.

D.10.11 Receive Buffer 15 Mask Registers

RX15MSKHI — Receive Buffer 15 Mask Register High **$YFF098**
RX15MSKLO — Receive Buffer 15 Mask Register Low **$YFF09A**

The receive buffer 15 mask registers have the same structure as the receive global mask registers and are used to mask buffer 15.