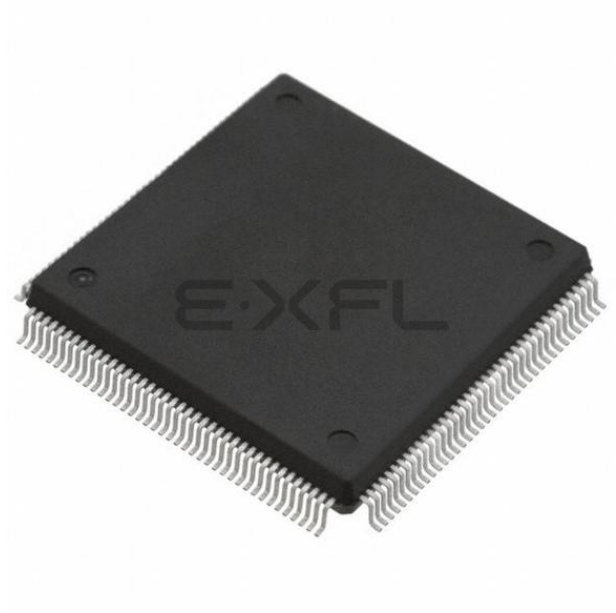

Welcome to E-XFL.COM

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	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 ~ 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/sc68376bgvab25r

$\overline{\text{DSACK}}$[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

4.4 Virtual Memory

The full addressing range of the CPU32 on the MC68336/376 is 16 Mbytes in each of eight address spaces. Even though most systems implement a smaller physical memory, the system can be made to appear to have a full 16 Mbytes of memory available to each user program by using virtual memory techniques.

A system that supports virtual memory has a limited amount of high-speed physical memory that can be accessed directly by the processor and maintains an image of a much larger virtual memory on a secondary storage device. When the processor attempts to access a location in the virtual memory map that is not resident in physical memory, a page fault occurs. The access to that location is temporarily suspended while the necessary data is fetched from secondary storage and placed in physical memory. The suspended access is then restarted or continued.

The CPU32 uses instruction restart, which requires that only a small portion of the internal machine state be saved. After correcting the fault, the machine state is restored, and the instruction is fetched and started again. This process is completely transparent to the application program.

4.5 Addressing Modes

Addressing in the CPU32 is register-oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. There is no need for extra instructions to store register contents in memory.

There are seven basic addressing modes:

- Register Direct
- Register Indirect
- Register Indirect with Index
- Program Counter Indirect with Displacement
- Program Counter Indirect with Index
- Absolute
- Immediate

The register indirect addressing modes include postincrement, predecrement, and offset capability. The program counter indirect mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the status register, stack pointer, and/or program counter.

4.6 Processing States

The processor is always in one of four processing states: normal, exception, halted, or background. The normal processing state is associated with instruction execution; the bus is used to fetch instructions and operands and to store results.

4.10.1 M68000 Family Development Support

All M68000 Family members include features to facilitate applications development. These features include the following:

Trace on Instruction Execution — M68000 Family processors include an instruction-by-instruction tracing facility as an aid to program development. The MC68020, MC68030, MC68040, and CPU32 also allow tracing only of those instructions causing a change in program flow. In the trace mode, a trace exception is generated after an instruction is executed, allowing a debugger program to monitor the execution of a program under test.

Breakpoint Instruction — An emulator may insert software breakpoints into the target code to indicate when a breakpoint has occurred. On the MC68010, MC68020, MC68030, and CPU32, this function is provided via illegal instructions, $4848–$484F, to serve as breakpoint instructions.

Unimplemented Instruction Emulation — During instruction execution, when an attempt is made to execute an illegal instruction, an illegal instruction exception occurs. Unimplemented instructions (F-line, A-line, . . .) utilize separate exception vectors to permit efficient emulation of unimplemented instructions in software.

4.10.2 Background Debug Mode

Microcomputer systems generally provide a debugger, implemented in software, for system analysis at the lowest level. The background debug mode (BDM) on the CPU32 is unique in that the debugger has been implemented in CPU microcode.

BDM incorporates a full set of debugging options: registers can be viewed or altered, memory can be read or written to, and test features can be invoked.

A resident debugger simplifies implementation of an in-circuit emulator. In a common setup (refer to **Figure 4-8**), emulator hardware replaces the target system processor. A complex, expensive pod-and-cable interface provides a communication path between the target system and the emulator.

By contrast, an integrated debugger supports use of a bus state analyzer (BSA) for incircuit emulation. The processor remains in the target system (refer to **Figure 4-9**) and the interface is simplified. The BSA monitors target processor operation and the on-chip debugger controls the operating environment. Emulation is much “closer” to target hardware, and many interfacing problems (for example, limitations on high-frequency operation, AC and DC parametric mismatches, and restrictions on cable length) are minimized.

Table 4-6 Background Mode Command Summary

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results via the serial interface.
Write D/A Register	WDREG/WAREG	The data operand is written to the specified address or data register.
Read System Register	RSREG	The specified system control register is read. All registers that can be read in supervisor mode can be read in background mode.
Write System Register	WSREG	The operand data is written into the specified system control register.
Read Memory Location	READ	Read the sized data at the memory location specified by the long-word address. The source function code register (SFC) determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The destination function code (DFC) register determines the address space accessed.
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and retrieve the first result. Subsequent operands are retrieved with the DUMP command.
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. An initial WRITE is executed to set up the starting address of the block and supply the first operand. Subsequent operands are written with the FILL command.
Resume Execution	GO	The pipe is flushed and re-filled before resuming instruction execution at the current PC.
Patch User Code	CALL	Current program counter is stacked at the location of the current stack pointer. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts $\overline{\text{RESET}}$ for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU RESET instruction.
No Operation	NOP	NOP performs no operation and may be used as a null command.

4.10.7 Background Mode Registers

BDM processing uses three special purpose registers to keep track of program context during development. A description of each follows.

4.10.7.1 Fault Address Register (FAR)

The FAR contains the address of the faulting bus cycle immediately following a bus or address error. This address remains available until overwritten by a subsequent bus cycle. Following a double bus fault, the FAR contains the address of the last bus cycle. The address of the first fault (if there was one) is not visible to the user.

4.10.7.2 Return Program Counter (RPC)

The RPC points to the location where fetching will commence after transition from background mode to normal mode. This register should be accessed to change the flow of a program under development. Changing the RPC to an odd value will cause an address error when normal mode prefetching begins.

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.

SECTION 9 QUEUED SERIAL MODULE

This section is an overview of the queued serial module (QSM). Refer to the *QSM Reference Manual* (QSMRM/AD) for complete information about the QSM.

9.1 General

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). **Figure 9-1** is a block diagram of the QSM.

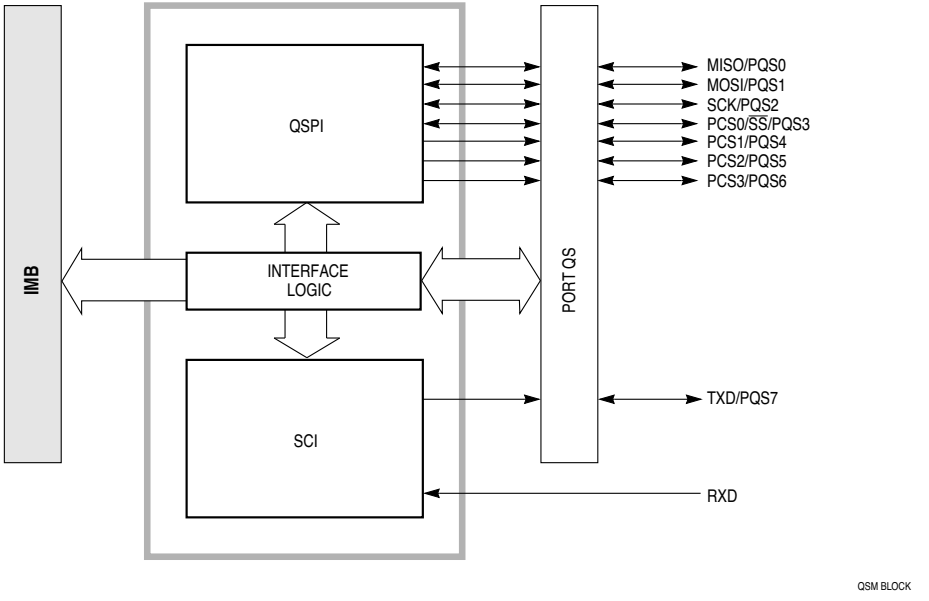

Figure 9-1 QSM Block Diagram

The QSPI provides peripheral expansion or interprocessor communication through a full-duplex, synchronous, three-line bus. Four programmable peripheral chip-selects can select up to sixteen peripheral devices by using an external one of sixteen line selector. A self-contained RAM queue allows up to sixteen serial transfers of eight to sixteen bits each or continuous transmission of up to a 256-bit data stream without CPU32 intervention. A special wrap-around mode supports continuous transmission/reception modes.

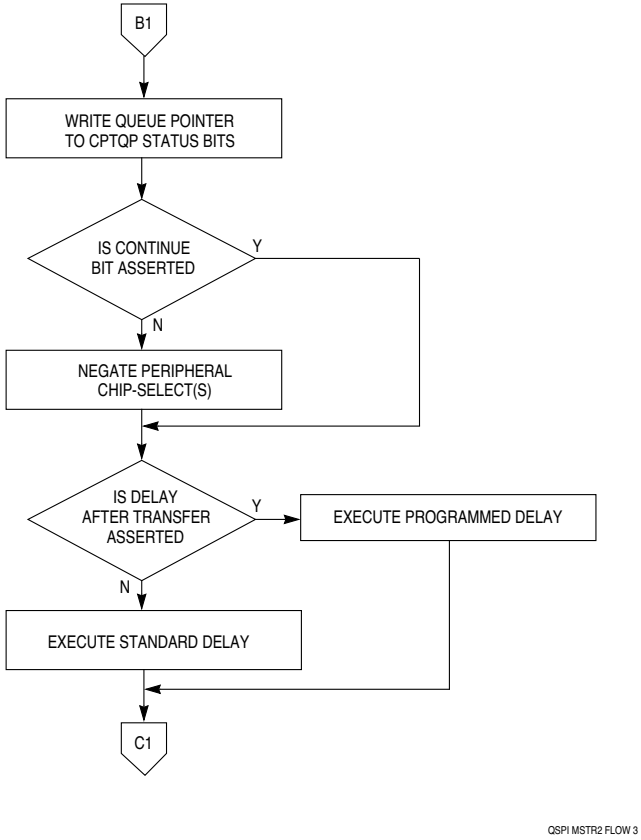

Figure 9-6 Flowchart of QSPI Master Operation (Part 2)

Table 8-4 QADC Clock Programmability

Control Register 0 Information				f_{sys} = 20.97 Input Sample Time (IST) = %00	
Example Number	PSH[4:0]	PSA	PSL[2:0]	QCLK (MHz)	Conversion Time (µs)
1	7	0	7	1.0	18.0
2	7	1	7	1.0	18.0

The MCU system clock frequency is the basis of QADC timing. The QADC requires that the system clock frequency be at least twice the QCLK frequency. Refer to **Table A-13** for information on the minimum and maximum allowable QCLK frequencies.

Example 1 in **Figure 8-9** shows that when PSH = 3, the QCLK remains high for four system clock cycles. It also shows that when PSL = 3, the QCLK remains low for four system clock cycles.

In order to tune QCLK for the fastest possible conversion time for any given system clock frequency, the QADC permits one more programmable control of the QCLK high and low time. The PSA bit in QACR0 allows the QCLK high phase to be stretched for a half cycle of the system clock, and correspondingly, the QCLK low phase is shortened by a half cycle of the system clock.

Example 2 in **Figure 8-9** is the same as Example 1, except that the PSA bit is set. The QCLK high phase has 4.5 system clock cycles; the QCLK low phase has 3.5 system clock cycles.

8.12.5 Periodic/Interval Timer

The QADC periodic/interval timer can be used to generate trigger events at programmable intervals to initiate scans of queue 2. The periodic/interval timer is held in reset under the following conditions:

- Queue 2 is programmed to any queue operating mode which does not use the periodic/interval timer
- Interval timer single-scan mode is selected, but the single-scan enable bit is cleared to zero
- IMB system reset or the master reset is asserted
- The QADC is placed in low-power stop mode with the STOP bit
- The IMB FREEZE line is asserted and the QADC FRZ bit is set to one

Two other conditions which cause a pulsed reset of the timer are:

- Rollover of the timer counter
- A queue operating mode change from one periodic/interval timer mode to another periodic/interval timer mode

During the low-power stop mode, the periodic/interval timer is held in reset. Since low-power stop mode initializes QACR2 to zero, a valid periodic or interval timer mode must be written to QACR2 when exiting low-power stop mode to release the timer from reset.

8.13.3 Interrupt Vectors

When the QADC is the only module with an interrupt request pending at the level being acknowledged, or when the QADC IARB value is higher than that of other modules with requests pending at the acknowledged IRQ level, the QADC responds to the interrupt acknowledge cycle with an 8-bit interrupt vector number. The CPU32 uses the vector number to calculate a displacement into the exception vector table, then uses the vector at that location to jump to an interrupt service routine.

The interrupt vector base field IVB[7:2] specifies the six high-order bits of the 8-bit interrupt vector number, and the QADC provides two low-order bits which correspond to one of the four QADC interrupt sources.

Figure 8-11 shows the format of the interrupt vector, and lists the binary coding of the two low-order bits for the four QADC interrupt sources.

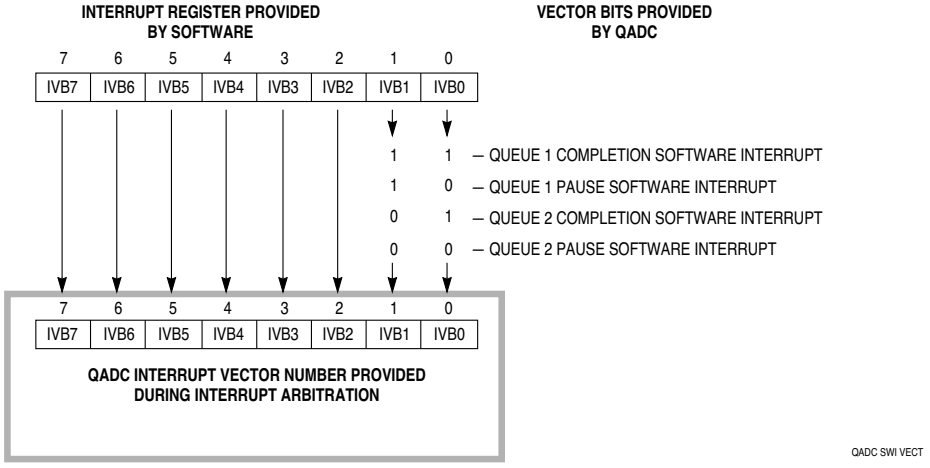

Figure 8-11 QADC Interrupt Vector Format

The IVB field has a reset value of $0F, which corresponds to the uninitialized interrupt exception vector.

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge aligned mode uses n + 1 TPU channels for n PWMs; center aligned mode uses 2n + 1 channels. Center aligned mode allows a user-defined “dead time” to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications.

Refer to TPU programming note *Multichannel Pulse-Width Modulation (MCPWM) TPU Function* (TPUPN05/D) for more information.

11.5.6 Fast Quadrature Decode (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU32 with a 16-bit free running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the NITC function.

Refer to TPU programming note *Fast Quadrature Decode (FQD) TPU Function* (TPUPN02/D) for more information.

11.5.7 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous serial communication. Data word length is programmable from one to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bidirectional UART channels running in excess of 9600 baud can be implemented.

Refer to TPU programming note *Universal Asynchronous Receiver/Transmitter (UART) TPU Function* (TPUPN07/D) for more information.

11.5.8 Brushless Motor Commutation (COMM)

This function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless DC motors. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. An offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU32. This feature is useful for torque maintenance at high speeds.

Refer to TPU programming note *Brushless Motor Commutation (COMM) TPU Function* (TPUPN09/D) for more information.

Table 13-7 Mask Examples for Normal/Extended Messages

Message Buffer (MB) /Mask	Base ID ID[28:18]	IDE	Extended ID ID[17:0]	Match
MB2	11111111000	0	—	—
MB3	11111111000	1	01010101010101010	—
MB4	00000011111	0	—	—
MB5	00000011101	1	01010101010101010	—
MB14	11111111000	1	01010101010101010	—
RX Global Mask	11111111110		11111110000000001	—
RX Message In	11111111001	1	01010101010101010	3[1]
	11111111001	0	—	2[2]
	11111111001	1	01010101010101010	—[3]
	01111111000	0	—	—[4]
	01111111000	1	01010101010101010	—[5]
RX 14 Mask	01111111111		11111110000000000	—
RX Message In	10111111000	1	01010101010101010	—[6]
	01111111000	1	01010101010101010	14[7]

NOTES:
1. Match for extended format (MB3).
2. Match for standard format (MB2).
3. No match for MB3 because of ID0.
4. No match for MB2 because of ID28.
5. No match for MB3 because of ID28, match for MB14.
6. No match for MB14 because of ID27.
7. Match for MB14.

13.4.3 Bit Timing

The TouCAN module uses three 8-bit registers to set-up the bit timing parameters required by the CAN protocol. Control registers 1 and 2 (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow the user to configure the bit timing parameters. The prescaler divide register (PRESDIV) allows the user to select the ratio used to derive the S-clock from the system clock. The time quanta clock operates at the S-clock frequency. **Table 13-8** provides examples of system clock, CAN bit rate, and S-clock bit timing parameters. Refer to **APPENDIX D REGISTER SUMMARY** for more information on the bit timing registers.

Table A-5 DC Characteristics

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

Num	Characteristic	Symbol	Min	Max	Unit
1	Input High Voltage	V_{IH}	0.7 (V_{DD})	V_{DD} + 0.3	V
2	Input Low Voltage	V_{IL}	V_{SS} – 0.3	0.2 (V_{DD})	V
3	Input Hysteresis[1]	V_{HYS}	0.5	—	V
4	Input Leakage Current[2] $V_{in} = V_{DD}$ or V_{SS} Input-only pins	I_{in}	–2.5	2.5	µA
5	High Impedance (Off-State) Leakage Current[2] $V_{in} = V_{DD}$ or V_{SS} All input/output and output pins	I_{OZ}	–2.5	2.5	µA
6	CMOS Output High Voltage[2, 3] I_{OH} = –10.0 µA Group 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} – 0.2	—	V
7	CMOS Output Low Voltage[2] I_{OL} = 10.0 µA Group 1, 2, 4 input/output and all output pins	V_{OL}	—	0.2	V
8	Output High Voltage[2, 3] I_{OH} = –0.8 mA Group 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} – 0.8	—	V
9	Output Low Voltage[2] I_{OL} = 1.6 mA Group 1 I/O Pins, CLKOUT, FREEZE/QUOT, $\overline{\text{IPIPE}}$ I_{OL} = 5.3 mA Group 2 and Group 4 I/O Pins, $\overline{\text{CSBOOT}}$, BG/$\overline{\text{CS}}$ I_{OL} = 12 mA Group 3	V_{OL}	— — —	0.4 0.4 0.4	V
10	Three State Control Input High Voltage	V_{IHTSC}	1.6 (V_{DD})	9.1	V
11	Data Bus Mode Select Pull-up Current[4] $V_{in} = V_{IL}$ DATA[15:0] $V_{in} = V_{IH}$ DATA[15:0]	I_{MSP}	— –15	–120 —	µA
12A	MC68336 V_{DD} Supply Current[5] RUN [6] RUN, TPU emulation mode LPSTOP, 4.194 MHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	 I_{DD} I_{DD} S_{IDD} S_{IDD}	 — — — —	 140 150 3 7	 mA mA mA mA
12B	MC68376 V_{DD} Supply Current[5] RUN[6] RUN, TPU emulation mode LPSTOP, 4.194 MHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	 I_{DD} I_{DD} S_{IDD} S_{IDD}	 — — — —	 150 160 3 7	 mA mA mA mA
13	Clock Synthesizer Operating Voltage	V_{DDSYN}	4.75	5.25	V
14	V_{DDSYN} Supply Current[5] 4.194 MHz crystal, VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 4.194 MHz crystal, VCO off (STSIM = 0) 4.194 MHz crystal, V_{DD} powered down	 I_{DDSYN} I_{DDSYN} S_{IDDSYN} I_{DDSYN}	 — — — —	 3 5 3 3	 mA mA mA mA

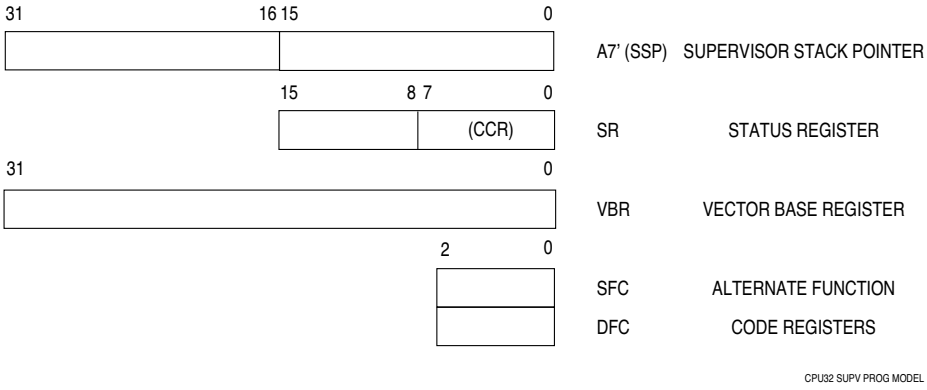

Figure D-2 Supervisor Programming Model Supplement

D.1.2 Status Register

SR — Status Register

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

The status register (SR) contains condition codes, an interrupt priority mask, and three control bits. The condition codes are contained in the condition code register (CCR), the lower byte of the SR. (The lower and upper bytes of the status register are also referred to as the user and system bytes, respectively.) In user mode, only the CCR is available. In supervisor mode, software can access the full status register.

T[1:0] — Trace Enable

This field places the processor in one of two tracing modes or disables tracing. Refer to **Table D-2**.

Table D-2 T[1:0] Encoding

T[1:0]	Response
00	No tracing
01	Trace on change of flow
10	Trace on instruction execution
11	Undefined; reserved

D.2 System Integration Module

Table D-3 shows the SIM address map. The column labeled “Access” indicates the privilege level at which the CPU32 must be operating to access the register. A designation of “S” indicates that supervisor mode is required. A designation of “S/U” indicates that the register can be programmed for either supervisor mode access or unrestricted access.

Table D-3 SIM Address Map

Access	Address[1]	15 8	7 0
S	$YFFA00	SIM Module Configuration Register (SIMCR)	
S	$YFFA02	SIM Test Register (SIMTR)	
S	$YFFA04	Clock Synthesizer Control Register (SYNCR)	
S	$YFFA06	Not Used	Reset Status Register (RSR)
S	$YFFA08	SIM Test Register E (SIMTRE)	
S	$YFFA0A	Not Used	
S	$YFFA0C	Not Used	
S	$YFFA0E	Not Used	
S/U	$YFFA10	Not Used	Port E Data (PORTE0)
S/U	$YFFA12	Not Used	Port E Data (PORTE1)
S/U	$YFFA14	Not Used	Port E Data Direction (DDRE)
S	$YFFA16	Not Used	Port E Pin Assignment (PEPAR)
S/U	$YFFA18	Not Used	Port F Data (PORTF0)
S/U	$YFFA1A	Not Used	Port F Data (PORTF1)
S/U	$YFFA1C	Not Used	Port F Data Direction (DDRF)
S	$YFFA1E	Not Used	Port F Pin Assignment (PFPAR)
S	$YFFA20	Not Used	System Protection Control (SYPCR)
S	$YFFA22	Periodic Interrupt Control Register (PICR)	
S	$YFFA24	Periodic Interrupt Timing Register (PITR)	
S	$YFFA26	Not Used	Software Service (SWSR)
S	$YFFA28	Not Used	
S	$YFFA2A	Not Used	
S	$YFFA2C	Not Used	
S	$YFFA2E	Not Used	
S	$YFFA30	Test Module Master Shift A (TSTMSRA)	
S	$YFFA32	Test Module Master Shift B (TSTMSRB)	
S	$YFFA34	Test Module Shift Count (TSTSC)	
S	$YFFA36	Test Module Repetition Counter (TSTRC)	
S	$YFFA38	Test Module Control (CREG)	
S/U	$YFFA3A	Test Module Distributed (DREG)	
	$YFFA3C	Not Used	
	$YFFA3E	Not Used	
S/U	$YFFA40	Not Used	Port C Data (PORTC)
	$YFFA42	Not Used	
S	$YFFA44	Chip-Select Pin Assignment (CSPAR0)	
S	$YFFA46	Chip-Select Pin Assignment (CSPAR1)	
S	$YFFA48	Chip-Select Base Boot (CSBARBT)	
S	$YFFA4A	Chip-Select Option Boot (CSORBT)	
S	$YFFA4C	Chip-Select Base 0 (CSBAR0)	
S	$YFFA4E	Chip-Select Option 0 (CSOR0)	

This field only affects the response of chip-selects and does not affect interrupt recognition by the CPU32.

$\overline{\text{AVEC}}$ — Autovector Enable

This field selects one of two methods of acquiring an interrupt vector during an interrupt acknowledge cycle. It is not usually used with a chip-select pin.

0 = External interrupt vector enabled
1 = Autovector enabled

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE[1:0] = %00) and the $\overline{\text{AVEC}}$ field is set to one, the chip-select automatically generates $\overline{\text{AVEC}}$ in response to the interrupt acknowledge cycle. Otherwise, the vector must be supplied by the requesting device.

D.2.22 Master Shift Registers

TSTMSRA — Master Shift Register A **$YFFA30**

Used for factory test only.

TSTMSRB — Master Shift Register B **$YFFA32**

Used for factory test only.

D.2.23 Test Module Shift Count Register

TSTSC — Test Module Shift Count **$YFFA34**

Used for factory test only.

D.2.24 Test Module Repetition Count Register

TSTRC — Test Module Repetition Count **$YFFA36**

Used for factory test only.

D.2.25 Test Submodule Control Register

CREG — Test Submodule Control Register **$YFFA38**

Used for factory test only.

D.2.26 Distributed Register

DREG — Distributed Register **$YFFA3A**

Used for factory test only.

D.6.10 Port QS Pin Assignment Register/Data Direction Register

PQSPAR — PORT QS Pin Assignment Register **$YFFC16**
DDRQS — PORT QS Data Direction Register **$YFFC17**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PQSPA6	PQSPA5	PQSPA4	PQSPA3	0	PQSPA1	PQSPA0	DDQS7	DDQS6	DDQS5	DDQS4	DDQS3	DDQS2	DDQS1	DDQS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clearing a bit in PQSPAR assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. PQSPAR does not affect operation of the SCI. **Table D-32** displays PQSPAR pin assignments.

Table D-32 PQSPAR Pin Assignments

PQSPAR Field	PQSPAR Bit	Pin Function
PQSPA0	0 1	PQS0 MISO
PQSPA1	0 1	PQS1 MOSI
—	— —	PQS2[1] SCK
PQSPA3	0 1	PQS3 PCS0/$\overline{SS}$
PQSPA4	0 1	PQS4 PCS1
PQSPA5	0 1	PQS5 PCS2
PQSPA6	0 1	PQS6 PCS3
—	— —	PQS7[2] TXD

NOTES:
1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes the QSPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes the SCI serial output TXD.

DDRQS determines whether pins configured for general purpose I/O are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function. **Table D-33** shows the effect of DDRQS on QSM pin function.

D.7 Configurable Timer Module 4

Table D-35 shows the CTM4 address map. All CTM4 control registers reside in supervisor space only.

Table D-35 CTM4 Address Map

Address[1]	15 0
$YFF400	BIUSM Module Configuration Register (BIUMCR)
$YFF402	BIUSM Test Register (BIUTEST)
$YFF404	BIUSM Time Base Register (BIUTBR)
$YFF406	Reserved
$YFF408	CPSM Control Register (CPCR)
$YFF40A	CPSM Test Register (CPTR)
$YFF40C – $YFF40E	Reserved
$YFF410	MCSM2 Status/Interrupt/Control Register (MCSM2SIC)
$YFF412	MCSM2 Counter (MCSM2CNT)
$YFF414	MCSM2 Modulus Latch (MCSM2ML)
$YFF416	Reserved
$YFF418	DASM3 Status/Interrupt/Control Register (DASM3SIC)
$YFF41A	DASM3 Register A (DASM3A)
$YFF41C	DASM3 Register B (DASM3B)
$YFF41E	Reserved
$YFF420	DASM4 Status/Interrupt/Control Register (DASM4SIC)
$YFF422	DASM4 Register A (DASM4A)
$YFF424	DASM4 Register B (DASM4B)
$YFF426	Reserved
$YFF428	PWMSM5 Status/Interrupt/Control Register (PWM5SIC)
$YFF42A	PWMSM5 Period (PWM5A)
$YFF42C	PWMSM5 Pulse Width (PWM5B)
$YFF42E	PWMSM5 Counter (PWM5C)
$YFF430	PWMSM6 Status/Interrupt/Control Register (PWM6SIC)
$YFF432	PWMSM6 Period (PWM6A)
$YFF434	PWMSM6 Pulse Width (PWM6B)
$YFF436	PWMSM6 Counter (PWM6C)
$YFF438	PWMSM7 Status/Interrupt/Control Register (PWM7SIC)
$YFF43A	PWMSM7 Period (PWM7A)
$YFF43C	PWMSM7 Pulse Width (PWM7B)
$YFF43E	PWMSM7 Counter (PWM7C)
$YFF440	PWMSM8 Status/Interrupt/Control Register (PWM8SIC)
$YFF442	PWMSM8 Period (PWM8A)
$YFF444	PWMSM8 Pulse Width (PWM8B)
$YFF446	PWMSM8 Counter (PWM8C)
$YFF448	DASM9 Status/Interrupt/Control Register (DASM9SIC)
$YFF44A	DASM9 Register A (DASM9A)
$YFF44C	DASM9 Register B (DASM9B)
$YFF44E	Reserved
$YFF450	DASM10 Status/Interrupt/Control Register (DASM10SIC)
$YFF452	DASM10 Register A (DASM10A)
$YFF454	DASM10 Register B (DASM10B)
$YFF456	Reserved
$YFF458	MCSM11 Status/Interrupt/Control Register (MCSM11SIC)

–D–

–E–

–F–