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	16MHz
Connectivity	EBI/EMI, SCI, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	15
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68332acag16

Table 2 MCU Pin Characteristic (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
T2CLK	—	Y	Y	—	—
TPUCH[15:0]	A	Y	Y	—	—
TSC	—	Y	Y	—	—
TXD	Bo	Y	Y	I/O	PQS7
XFC[2]	—	—	—	Special	—
XTAL[2]	—	—	—	Special	—

NOTES:

1. DATA[15:0] are synchronized during reset only. MODCLK is synchronized only when used as an input port pin.
2. EXTAL, XFC, and XTAL are clock reference connections.

2.2 MCU Power Connections

Table 3 MCU Power Connections

V_{STBY}	Standby RAM Power/Clock Synthesizer Power
V_{DDSYN}	Clock Synthesizer Power
V_{SSE}/V_{DDE}	External Periphery Power (Source and Drain)
V_{SSI}/V_{DDI}	Internal Module Power (Source and Drain)

2.3 MCU Driver Types

Table 4 MCU Driver Types

Type	I/O	Description
A	O	Output-only signals that are always driven; no external pull-up required
Aw	O	Type A output with weak P-channel pull-up during reset
B	O	Three-state output that includes circuitry to pull up output before high impedance is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while the pin is in the high-impedance state.
Bo	O	Type B output that can be operated in an open-drain mode

3.2.3 Bus Monitor

The internal bus monitor checks for excessively long $\overline{\text{DSACK}}$ response times during normal bus cycles and for excessively long $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ response times during interrupt acknowledge cycles. The monitor asserts $\overline{\text{BERR}}$ if response time is excessive.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. The maximum allowable response time can be selected by setting the BMT field.

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU initiates the bus cycle. The BME bit in the SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal to external bus monitor option must be disabled.

3.2.4 Halt Monitor

The halt monitor responds to an assertion of $\overline{\text{HALT}}$ on the internal bus. A flag in the reset status register (RSR) indicates that the last reset was caused by the halt monitor. The halt monitor reset can be inhibited by the HME bit in the SYPCR.

3.2.5 Spurious Interrupt Monitor

The spurious interrupt monitor issues $\overline{\text{BERR}}$ if no interrupt arbitration occurs during an interrupt-acknowledge cycle.

3.2.6 Software Watchdog

The software watchdog is controlled by SWE in the SYPCR. Once enabled, the watchdog requires that a service sequence be written to SWSR on a periodic basis. If servicing does not take place, the watchdog times out and issues a reset. This register can be written at any time, but returns zeros when read.

SWSR —Software Service Register **$YFFA27**

15	8	7	6	5	4	3	2	1	0
NOT USED		0	0	0	0	0	0	0	0
RESET:									
		0	0	0	0	0	0	0	0

Register shown with read value

Perform a software watchdog service sequence as follows:

a. Write $55 to SWSR.
b. Write $AA to SWSR.

Both writes must occur before time-out in the order listed, but any number of instructions can be executed between the two writes.

The watchdog clock rate is affected by SWP and SWT in SYPCR. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period takes effect.

The reset value of SWP is affected by the state of the MODCLK pin on the rising edge of reset, as shown in the following table.

MODCLK	SWP
0	1
1	0

When an external system clock signal is applied (i.e., the PLL is not used), duty cycle of the input is critical, especially at near maximum operating frequencies. The relationship between clock signal duty cycle and clock signal period is expressed:

$$\text{Minimum external clock period} =$$

$$\frac{\text{minimum external clock high/low time}}{\text{50\% — percentage variation of external clock input duty cycle}}$$

3.3.2 Clock Synthesizer Operation

A voltage controlled oscillator (VCO) generates the system clock signal. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is a reference signal, either from the internal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is identical to reference frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must re-lock. Lock status is shown by the SLOCK bit in SYNCR.

The MCU does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

The low-pass filter requires an external low-leakage capacitor, typically 0.1 µF, connected between the XFC and V_{DDSYN} pins.

V_{DDSYN} is used to power the clock circuits. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. Use a quiet power supply as the V_{DDSYN} source, since PLL stability depends on the VCO, which uses this supply. Place adequate external bypass capacitors as close as possible to the V_{DDSYN} pin to ensure stable operating frequency.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. SYNCR can be read only when the processor is operating at the supervisor privilege level.

The SYNCR X bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting X doubles clock speed without changing VCO speed. There is no VCO relock delay. The SYNCR W bit controls a 3-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When either W or Y value changes, there is a VCO relock delay.

Clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE}\,[4(Y + 1)(2^{2W + X})]$$

In order for the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified for the MCU.

The VCO frequency is twice the system clock frequency if X = 1 or four times the system clock frequency if X = 0.

The reset state of SYNCR ($3F00) produces a modulus-64 count.

3.4.8 Data Transfer Mechanism

The MCU architecture supports byte, word, and long-word operands, allowing access to 8- and 16-bit data ports through the use of asynchronous cycles controlled by the data transfer and size acknowledge inputs ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$).

3.4.9 Dynamic Bus Sizing

The MCU dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8- and 16-bit ports. During an operand transfer cycle, the slave device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ inputs, as shown in the following table.

Table 10 Effect of $\overline{\text{DSACK}}$ Signals

$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle —Data Bus Port Size is 8 Bits
0	1	Complete Cycle —Data Bus Port Size is 16 Bits
0	0	Reserved

For example, if the MCU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ signals to indicate the port width. For instance, a 16-bit device always returns $\overline{\text{DSACK0}}$ = 1 and $\overline{\text{DSACK1}}$ = 0 for a 16-bit port, regardless of whether the bus cycle is a byte or word operation.

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

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins. Operand bytes are designated as shown in the following figure. OP0 is the most significant byte of a long-word operand, and OP3 is the least significant byte. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

Operand	**Byte Order**			
	31 24	23 16	15 8	7 0
Long Word	OP0	OP1	OP2	OP3
Three Byte		OP0	OP1	OP2
Word			OP0	OP1
Byte				OP0

Figure 8 Operand Byte Order

3.4.10 Operand Alignment

The data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the size and address outputs. SIZ1 and SIZ0 indicate the remaining number of bytes to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During an operand transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base.

3.4.11 Misaligned Operands

CPU32 processor architecture uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address. The CPU32 does not support misaligned operand transfers.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand via a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word on a following bus cycle.

3.4.12 Operand Transfer Cases

The following table summarizes how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle.

Table 11 Operand Alignment

Transfer Case	SIZ1	SIZ0	ADDR0	DSACK1	DSACK0	DATA [15:8]	DATA [7:0]
Byte to 8-Bit Port (Even/Odd)	0	1	X	1	0	OP0	(OP0)
Byte to 16-Bit Port (Even)	0	1	0	0	X	OP0	(OP0)
Byte to 16-Bit Port (Odd)	0	1	1	0	X	(OP0)	OP0
Word to 8-Bit Port (Aligned)	1	0	0	1	0	OP0	(OP1)
Word to 8-Bit Port (Misaligned)[3]	1	0	1	1	0	OP0	(OP0)
Word to 16-Bit Port (Aligned)	1	0	0	0	X	OP0	OP1
Word to 16-Bit Port (Misaligned)[3]	1	0	1	0	X	(OP0)	OP0
3 Byte to 8-Bit Port (Aligned)[2]	1	1	0	1	0	OP0	(OP1)
3 Byte to 8-Bit Port (Misaligned)[2, 3]	1	1	1	1	0	OP0	(OP0)
3 Byte to 16-Bit Port (Aligned)[2]	1	1	0	0	X	OP0	OP1
3 Byte to 16-Bit Port (Misaligned)[2, 3]	1	1	1	0	X	(OP0)	OP0
Long Word to 8-Bit Port (Aligned)	0	0	0	1	0	OP0	(OP1)
Long Word to 8-Bit Port (Misaligned)[3]	1	0	1	1	0	OP0	(OP0)
Long Word to 16-Bit Port (Aligned)	0	0	0	0	X	OP0	OP1
Long Word to 16-Bit Port (Misaligned)[3]	1	0	1	0	X	(OP0)	OP0

NOTES:

1. Operands in parentheses are ignored by the CPU32 during read cycles.
2. Three-byte transfer cases occur only as a result of a long word to byte transfer.
3. The CPU32 does not support misaligned word or long-word transfers.

3.5 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. Twelve independently programmable chip selects provide fast two-cycle access to external memory or peripherals. Address block sizes of 2 Kbytes to 1 Mbyte can be selected.

3.5.1 Chip-Select Registers

Pin assignment registers CSPAR0 and CSPAR1 determine functions of chip-select pins. These registers also determine port size (8- or 16-bit) for dynamic bus allocation.

A pin data register (PORTC) latches discrete output data.

Blocks of addresses are assigned to each chip-select function. Block sizes of 2 Kbytes to 1 Mbyte can be selected by writing values to the appropriate base address register (CSBAR). Address blocks for separate chip-select functions can overlap.

Chip-select option registers (CSORBT and CSOR[10:0]) determine timing of and conditions for assertion of chip-select signals. Eight parameters, including operating mode, access size, synchronization, and wait state insertion can be specified.

Initialization code often resides in a peripheral memory device controlled by the chip-select circuits. A set of special chip-select functions and registers (CSORBT, CSBARBT) is provided to support bootstrap operation.

3.5.2 Pin Assignment Registers

The pin assignment registers (CSPAR0 and CSPAR1) contain pairs of bits that determine the function of chip-select pins. The pin assignment encodings used in these registers are shown below.

Table 12 Pin Assignment Encodings

Bit Field	Description
00	Discrete Output
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

CSPAR0 —Chip Select Pin Assignment Register 0 **$YFFA44**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CSPA0[6]		CSPA0[5]		CSPA0[4]		CSPA0[3]		CSPA0[2]		CSPA0[1]		CSBOOT	
RESET:															
0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 contains seven 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect.

Table 13 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	$\overline{\text{CS4}}$	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
CSPA0[1]	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—	—

CSPAR1 —Chip Select Pin Assignment Register 1 **$YFFA46**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	CSPA1[4]		CSPA1[3]		CSPA1[2]		CSPA1[1]		CSPA1[0]	
RESET:															
0	0	0	0	0	0	DATA7	1	DATA [7:6]	1	DATA [7:5]	1	DATA [7:4]	1	DATA [7:3]	1

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect.

Table 14 CSPAR1 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA1[4]	$\overline{\text{CS10}}$	ADDR23	ECLK
CSPA1[3]	$\overline{\text{CS9}}$	ADDR22	PC6
CSPA1[2]	$\overline{\text{CS8}}$	ADDR21	PC5
CSPA1[1]	$\overline{\text{CS7}}$	ADDR20	PC4
CSPA1[0]	$\overline{\text{CS6}}$	ADDR19	PC3

At reset, either the alternate function (01) or chip-select function (11) can be encoded. DATA pins are driven to logic level one by a weak interval pull-up during reset. Encoding is for chip-select function unless a data line is held low during reset. Note that bus loading can overcome the weak pull-up and hold pins low during reset. The following table shows the hierarchical selection method that determines the reset functions of pins controlled by CSPAR1.

Table 15 Reset Pin Function of $\overline{\text{CS[10:6]}}$

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

A pin programmed as a discrete output drives an external signal to the value specified in the port C pin data register (PORTC), with the following exceptions:

1. No discrete output function is available on pins $\overline{\text{BR}}$, $\overline{\text{BG}}$, or $\overline{\text{BGACK}}$.
2. ADDR23 provides E-clock output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ internally on an address match.

Port size is determined when a pin is assigned as a chip select. When a pin is assigned to an 8-bit port, the chip select is asserted at all addresses within the block range. If a pin is assigned to a 16-bit port, the upper/lower byte field of the option register selects the byte with which the chip select is associated.

PFPAR — Port F Pin Assignment Register **$YFFA1F**

15	8	7	6	5	4	3	2	1	0
NOT USED		PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:									
		DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

The bits in this register control the function of each port F pin. Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be an interrupt request signal or MODCLK. The MODCLK signal has no function after reset.

Table 17 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{\text{IRQ7}}$
PFPA6	PF6	$\overline{\text{IRQ6}}$
PFPA5	PF5	$\overline{\text{IRQ5}}$
PFPA4	PF4	$\overline{\text{IRQ4}}$
PFPA3	PF3	$\overline{\text{IRQ3}}$
PFPA2	PF2	$\overline{\text{IRQ2}}$
PFPA1	PF1	$\overline{\text{IRQ1}}$
PFPA0	PF0	MODCLK

Data bus pin 9 controls the state of this register following reset. If DATA9 is set to one during reset, the register is set to $FF, which defines all port F pins as interrupt request inputs. If DATA9 is cleared to zero during reset, this register is set to $00, defining all port F pins as I/O pins.

3.7 Resets

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

Reset occurs when an active low logic level on the $\overline{\text{RESET}}$ pin is clocked into the SIM. Resets are gated by the CLKOUT signal. Asynchronous resets are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. Synchronous resets are timed to occur at the end of bus cycles. If there is no clock when $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked in order to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ is asserted.

Reset is the highest-priority CPU32 exception. Any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

3.7.1 SIM Reset Mode Selection

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. The following table is a summary of reset mode selection options.

Table 18 Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)

4 Central Processor Unit

Based on the powerful MC68020, the CPU32 processing module provides enhanced system performance and also uses the extensive software base for the Motorola M68000 family.

4.1 Overview

The CPU32 is fully object code compatible with the M68000 Family, which excels at processing calculation-intensive algorithms and supporting high-level languages. The CPU32 supports all of the MC68010 and most of the MC68020 enhancements, such as virtual memory support, loop mode operation, instruction pipeline, and 32-bit mathematical operations. Powerful addressing modes provide compatibility with existing software programs and increase the efficiency of high-level language compilers. Special instructions, such as table lookup and interpolate and low-power stop, support the specific requirements of controller applications. Also included is the background debugging mode, an alternate operating mode that suspends normal operation and allows the CPU to accept debugging commands from the development system.

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction set is optimized for high performance. The eight 32-bit general-purpose data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long word) operations. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

Use of high-level languages is increasing as controller applications become more complex and control programs become larger. High-level languages aid rapid development of software, with less error, and are readily portable. The CPU32 instruction set supports high-level languages.

4.2 Programming Model

The CPU32 has sixteen 32-bit general registers, a 32-bit program counter, one 32-bit supervisor stack pointer, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register.

The programming model of the CPU32 consists of a user model and supervisor model, corresponding to the user and supervisor privilege levels. Some instructions available at the supervisor level are not available at the user level, allowing the supervisor to protect system resources from uncontrolled access. Bit S in the status register determines the privilege level.

The user programming model remains unchanged from previous M68000 Family microprocessors. Application software written to run at the non-privileged user level migrates without modification to the CPU32 from any M68000 platform. The move from SR instruction, however, is privileged in the CPU32. It is not privileged in the M68000.

4.7 Background Debugging Mode

The background debugger on the CPU32 is implemented in CPU microcode. The background debugging commands are summarized below.

Table 21 Background Debuggung Mode

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results through 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. Initially, a WRITE is executed to set up the starting address of the block and supply the first operand. The FILL command writes subsequent operands.
Resume Execution	GO	The pipe is flushed and refilled 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 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 can be used as a null command.

5.1.2 Input Capture/Input Transition Counter (ITC)

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

5.1.3 Output Compare (OC)

The output compare function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time.
2. At a programmable delay time from a user-specified time.
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} * \text{Ratio}$$

where Ratio is a parameter supplied by the user.

This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins.

5.1.4 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin.

5.1.5 Synchronized Pulse-Width Modulation (SPWM)

The TPU generates a PWM waveform in which the CPU can change the period and/or high time at any time. When synchronized to a time function on a second channel, the synchronized PWM low-to-high transitions have a time relationship to transitions on the second channel.

5.1.6 Period Measurement with Additional Transition Detect (PMA)

This function and the following function are used primarily in toothed-wheel speed-sensing applications, such as monitoring rotational speed of an engine. The period measurement with additional transition detect function allows for a special-purpose 23-bit period measurement. It can detect the occurrence of an additional transition (caused by an extra tooth on the sensed wheel) indicated by a period measurement that is less than a programmable ratio of the previous period measurement.

Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. Alternatively, a byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next additional transition is detected.

lation parameter. From 1 to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (1 to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter. By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels.

5.1.11 Quadrature Decode (QDEC)

The quadrature decode function uses two channels to decode a pair of out-of-phase signals in order to present the CPU with directional information and a position value. It is particularly suitable for use with slotted encoders employed in motor control. The function derives full resolution from the encoder signals and provides a 16-bit position counter with rollover/under indication via an interrupt.

The counter in parameter RAM is updated when a valid transition is detected on either one of the two inputs. The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of servicing the transition. The user can read or write the counter at any time. The counter is free running, overflowing to $0000 or underflowing to $FFFF depending on direction. The QDEC function also provides a time stamp referenced to TCR1 for every valid signal edge and the ability for the host CPU to obtain the latest TCR1 value. This feature allows position interpolation by the host CPU between counts at very slow count rates.

5.2 MC68332G Time Functions

The following paragraphs describe factory-programmed time functions implemented in the motion-control microcode ROM. A complete description of the functions is beyond the scope of this summary. Refer to *Using the TPU Function Library and TPU Emulation Mode* (TPUPN00/D) for more information about specific functions.

5.2.1 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in PRAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

5.2.2 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 bus master. 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.

5.4 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the TPU module address map. Channel parameters are organized as 128 16-bit words. However, only 100 words are actually implemented. The parameter RAM address map shows how parameter words are organized in memory.

Table 23 TPU Parameter RAM Address Map

Channel	Base	Parameter Address							
Number	Address	0	1	2	3	4	5	6	7
0	$YFFFF##	00	02	04	06	08	0A	—	—
1	$YFFFF##	10	12	14	16	18	1A	—	—
2	$YFFFF##	20	22	24	26	28	2A	—	—
3	$YFFFF##	30	32	34	36	38	3A	—	—
4	$YFFFF##	40	42	44	46	48	4A	—	—
5	$YFFFF##	50	52	54	56	58	5A	—	—
6	$YFFFF##	60	62	64	66	68	6A	—	—
7	$YFFFF##	70	72	74	76	78	7A	—	—
8	$YFFFF##	80	82	84	86	88	8A	—	—
9	$YFFFF##	90	92	94	96	98	9A	—	—
10	$YFFFF##	A0	A2	A4	A6	A8	AA	—	—
11	$YFFFF##	B0	B2	B4	B6	B8	BA	—	—
12	$YFFFF##	C0	C2	C4	C6	C8	CA	—	—
13	$YFFFF##	D0	D2	D4	D6	D8	DA	—	—
14	$YFFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	$YFFFF##	F0	F2	F4	F6	F8	FA	FC	FE

—= Not Implemented

Y = M111, where M represents the logic state of the MM bit in the SIMCR.

5.5 TPU Registers

The TPU memory map contains three groups of registers:

System Configuration Registers
Channel Control and Status Registers
Development Support and Test Verification Registers

5.5.1 System Configuration Registers

TPUMCR — TPU Module Configuration Register **$YFFE00**

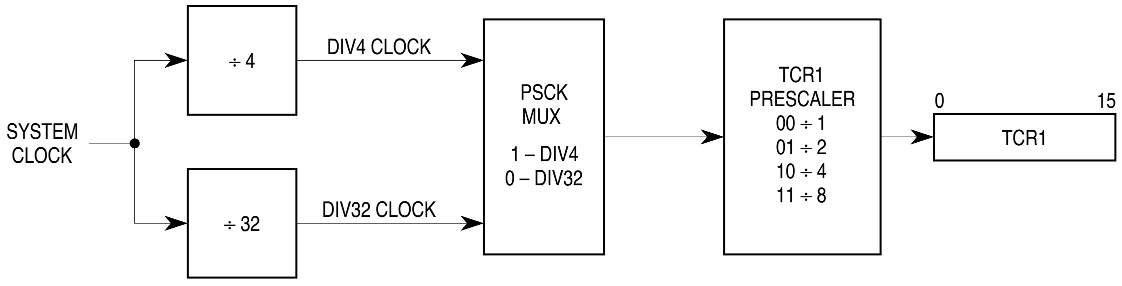
15	14	13	12	11	10	9	8	7	6	5	4	3			0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

STOP — Stop Bit

0 = TPU operating normally
1 = Internal clocks shut down

TCR1P — Timer Count Register 1 Prescaler Control

TCR1 is clocked from the output of a prescaler. 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. Channels using TCR1 have the capability to resolve down to the TPU system clock divided by 4.

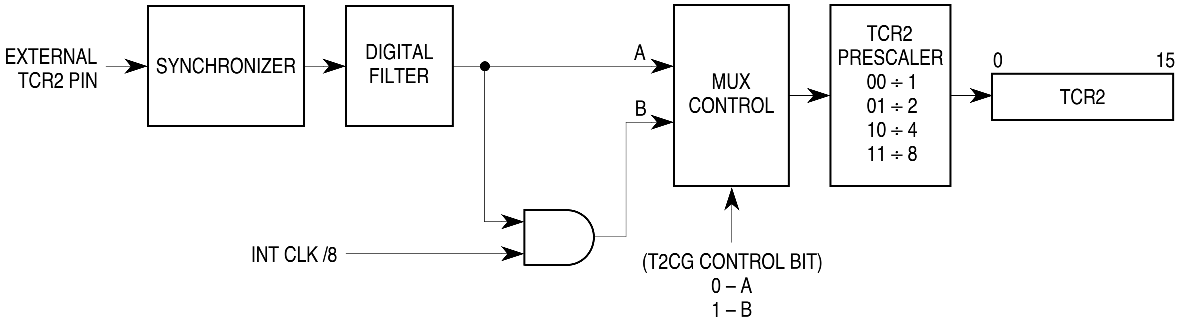


PRESCALER CTL BLOCK 1

TCR1 Prescaler	Divide By	PSCK = 0		PSCK = 1	
		Number of Clocks	Rate at 16 MHz	Number of Clocks	Rate at 16 MHz
00	1	32	2 ms	4	250 ns
01	2	64	4 ms	8	500 ns
10	4	128	8 ms	16	1 ms
11	8	256	16 ms	32	2 ms

TCR2P — Timer Count Register 2 Prescaler Control

TCR2 is clocked from the output of a prescaler. If T2CG = 0, the input to the TCR2 prescaler is the external TCR2 clock source. If T2CG = 1, the input is the TPU system clock divided by eight. The TCR2P field specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by 8. The following table is a summary of prescaler output.



PRESCALER CTL BLOCK 2

TCR2 Prescaler	Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

6.2 Address Map

The “Access” column in the QSM address map below indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the QSMCR.

Table 24 QSM Address Map

Access	Address	15 8	7 0
S	$YFFC00	QSM MODULE CONFIGURATION (QSMCR)	
S	$YFFC02	QSM TEST (QTEST)	
S	$YFFC04	QSM INTERRUPT LEVEL (QILR)	QSM INTERRUPT VECTOR (QIVR)
S/U	$YFFC06	NOT USED	
S/U	$YFFC08	SCI CONTROL 0 (SCCR0)	
S/U	$YFFC0A	SCI CONTROL 1 (SCCR1)	
S/U	$YFFC0C	SCI STATUS (SCSR)	
S/U	$YFFC0E	SCI DATA (SCDR)	
S/U	$YFFC10	NOT USED	
S/U	$YFFC12	NOT USED	
S/U	$YFFC14	NOT USED	PQS DATA (PORTQS)
S/U	$YFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
S/U	$YFFC18	SPI CONTROL 0 (SPCR0)	
S/U	$YFFC1A	SPI CONTROL 1 (SPCR1)	
S/U	$YFFC1C	SPI CONTROL 2 (SPCR2)	
S/U	$YFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
S/U	$YFFC20– $YFFCFF	NOT USED	
S/U	$YFFD00– $YFFD1F	RECEIVE RAM (RR[0:F])	
S/U	$YFFD20– $YFFD3F	TRANSMIT RAM (TR[0:F])	
S/U	$YFFD40– $YFFD4F	COMMAND RAM (CR[0:F])	

Y = M111, where M is the logic state of the MM bit in the SIMCR.

6.3 Pin Function

The following table is a summary of the functions of the QSM pins when they are not configured for general-purpose I/O. The QSM data direction register (DDRQS) designates each pin except RXD as an input or output.

	Pin	Mode	Pin Function
QSPI Pins	MISO	Master	Serial Data Input to QSPI
		Slave	Serial Data Output from QSPI
	MOSI	Master	Serial Data Output from QSPI
		Slave	Serial Data Input to QSPI
	SCK	Master	Clock Output from QSPI
		Slave	Clock Input to QSPI
	PCS0/$\overline{SS}$	Master	Input: Assertion Causes Mode Fault Output: Selects Peripherals
		Slave	Input: Selects the QSPI
	PCS[3:1]	Master	Output: Selects Peripherals
		Slave	None
SCI Pins	TXD	Transmit	Serial Data Output from SCI
	RXD	Receive	Serial Data Input to SCI

6.4 QSM Registers

QSM registers are divided into four categories: QSM global registers, QSM pin control registers, QSPI submodule registers, and SCI submodule registers. The QSPI and SCI registers are defined in separate sections below. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The module mapping bit of the SIM configuration register (SIMCR) defines the most significant bit (ADDR23) of the address, shown in each register figure as Y (Y = $7 or $F). This bit, concatenated with the rest of the address given, forms the absolute address of each register. Refer to the SIM section of this technical summary for more information about how the state of MM affects the system.

6.4.1 Global Registers

The QSM global registers contain system parameters used by both the QSPI and the SCI submodules. These registers contain the bits and fields used to configure the QSM.

QSMCR — QSM Configuration Register **$YFFC00**

15	14	13	12	11	10	9	8	7	6	5	4	3			0
STOP	FRZ1	FRZ0	0	0	0	0	0	SUPV	0	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The QSMCR contains parameters for the QSM/CPU/intermodule bus (IMB) interface.

STOP — Stop Enable

0 = Normal QSM clock operation

1 = QSM clock operation stopped

STOP places the QSM in a low-power state by disabling the system clock in most parts of the module. The QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable. However, writes to RAM or any register are guaranteed to be valid while STOP is asserted. STOP can be negated by the CPU and by reset.

Table 26 Effect of DDRQS on QSM Pin Function

QSM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQ0	0	Serial Data Input to QSPI
			1	Disables Data Input
	Slave		0	Disables Data Output
			1	Serial Data Output from QSPI
MOSI	Master	DDQ1	0	Disables Data Output
			1	Serial Data Output from QSPI
	Slave		0	Serial Data Input to QSPI
			1	Disables Data Input
SCK[1]	Master	DDQ2	0	Disables Clock Output
			1	Clock Output from QSPI
	Slave		0	Clock Input to QSPI
			1	Disables Clock Input
PCS0/$\overline{\text{SS}}$	Master	DDQ3	0	Assertion Causes Mode Fault
			1	Chip-Select Output
	Slave		0	QSPI Slave Select Input
			1	Disables Select Input
PCS[3:1]	Master	DDQ[4:6]	0	Disables Chip-Select Output
			1	Chip-Select Output
	Slave		0	Inactive
			1	Inactive
TXD[2]	Transmit	DDQ7	X	Serial Data Output from SCI
RXD	Receive	None	NA	Serial Data Input to SCI

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI 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 SCI serial output TXD.

DDRQS determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output.

Writing a value of zero to SCBR disables the baud rate generator.

The following table lists the SCBR settings for standard and maximum baud rates using 16.78-MHz and 20.97-MHz system clocks.

Table 27 SCI Baud Rates

Nominal Baud Rate	Actual Rate with 16.78-MHz Clock	SCBR Value	Actual Rate with 20.97-MHz Clock	SCBR Value
64*	64.0	$1FFF	—	—
110	110.0	$129E	110.0	$1745
300	299.9	$06D4	300.1	$0888
600	599.9	$036A	600.1	$0444
1200	1199.7	$0165	1200.3	$0222
2400	2405.0	$00DA	2400.6	$0111
4800	4810.0	$006D	4783.6	$0089
9600	9532.5	$0037	9637.6	$0044
19200	19418.1	$0016	19275.3	$0022
38400	37449.1	$000E	38550.6	$0011
76800	74898.3	$0007	72817.8	$0009
Maximum Rate	524288.0	$0001	655360.0	$0001

SCCR1 — SCI Control Register 1 **$YFFC0A**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCCR1 contains SCI configuration parameters. The CPU can read and write this register at any time. The SCI can modify RWU in some circumstances. In general, interrupts enabled by these control bits are cleared by reading SCSR, then reading (receiver status bits) or writing (transmitter status bits) SCDR.

Bit 15 — Not Implemented

LOOPS — Loop Mode

0 = Normal SCI operation, no looping, feedback path disabled
1 = Test SCI operation, looping, feedback path enabled

LOOPS controls a feedback path on the data serial shifter. When loop mode is enabled, SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

WOMS — Wired-OR Mode for SCI Pins

0 = If configured as an output, TXD is a normal CMOS output.
1 = If configured as an output, TXD is an open-drain output.

WOMS determines whether the TXD pin is an open-drain output or a normal CMOS output. This bit is used only when TXD is an output. If TXD is used as a general-purpose input pin, WOMS has no effect.

ILT — Idle-Line Detect Type

0 = Short idle-line detect (start count on first one)
1 = Long idle-line detect (start count on first one after stop bit(s))

PT — Parity Type

0 = Even parity
1 = Odd parity

When parity is enabled, PT determines whether parity is even or odd for both the receiver and the transmitter.

RASP — RAM Array Space Field

0 = TPURAM array is placed in unrestricted space
1 = TPURAM array is placed in supervisor space

TRAMTST — TPURAM Test Register **$YFFB02**

TRAMTST is used for factory testing of the TPURAM module.

TRAMBAR — TPURAM Base Address and Status Register **$YFFB04**

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

ADDR[23:11] — RAM Array Base Address

These bits specify address lines ADDR[23:11] of the base address of the RAM array when enabled.

RAMDS — RAM Array Disable

0 = RAM array is enabled
1 = RAM array is disabled

The RAM array is disabled by internal logic after a master reset. Writing a valid base address to the RAM array base address field (bits [15:3]) automatically clears RAMDS, enabling the RAM array.

7.4 TPURAM Operation

There are six TPURAM operating modes, as follows:

1. The TPURAM module is in normal mode when powered by V_{DD}. The array can be accessed by byte, word, or long word. A byte or aligned word (high-order byte is at an even address) access only takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles.
2. Standby mode is intended to preserve TPURAM contents when V_{DD} is removed. TPURAM contents are maintained by V_{STBY}. Circuitry within the TPURAM module switches to the higher of V_{DD} or V_{STBY} with no loss of data. When TPURAM is powered by V_{STBY}, access to the array is not guaranteed.
3. Reset mode allows the CPU to complete the current bus cycle before resetting. When a synchronous reset occurs while a byte or word TPURAM access is in progress, the access will be completed. If reset occurs during the first word access of a long-word operation, only the first word access will be completed. If reset occurs during the second word access of a long word operation, the entire access will be completed. Data being read from or written to the RAM may be corrupted by asynchronous reset.
4. Test mode functions in conjunction with the SIM test functions. Test mode is used during factory test of the MCU.
5. Writing the STOP bit of TRAMMCR causes the TPURAM module to enter stop mode. The TPURAM array is disabled (which allows external logic to decode TPURAM addresses, if necessary), but all data is retained. If V_{DD} falls below V_{STBY} during stop mode, internal circuitry switches to V_{STBY}, as in standby mode. Stop mode is exited by clearing the STOP bit.
6. The TPURAM array may be used to emulate the microcode ROM in the TPU module. This provides a means of developing custom TPU code. The TPU selects TPU emulation mode. While in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses via the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses.