Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	CPU32+
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	Communications; CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10Mbps (1)
SATA	-
USB	-
Voltage - I/O	3.3V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	357-BGA
Supplier Device Package	357-PBGA (25x25)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68360zp25vlr2

1.2.3 Communications Processor Module (CPM)

The CPM contains features that allow the QUICC to excel in communications and control applications. These features may be divided into three sub-groups:

- Communications Processor (CP)
- Two IDMA Controllers
- Four General-Purpose Timers

The CP provides the communication features of the QUICC. Included are a RISC processor, four SCCs, two SMCs, one SPI, 2.5 Kbytes of dual-port RAM, an interrupt controller, a time slot assigner, three parallel ports, a parallel interface port, four independent baud rate generators, and fourteen serial DMA channels to support the SCCs, SMCs, and SPI.

The IDMAs provide two channels of general-purpose DMA capability. They offer high-speed transfers, 32-bit data movement, buffer chaining, and independent request and acknowledge logic. The RISC controller may access the IDMA registers directly in the buffer chaining modes. The QUICC IDMAs are similar to, yet enhancements of, the two DMA channels found on the MC68340 and the one IDMA channel found on the MC68302.

The four general-purpose timers on the QUICC are functionally similar to the two general-purpose timers found on the MC68302. However, they offer some minor enhancements, such as the internal cascading of two timers to form a 32-bit timer. The QUICC also contains a periodic interval timer in the SIM60, bringing the total to five on-chip timers.

1.3 UPGRADING DESIGNS FROM THE MC68302

Since the QUICC is a next-generation MC68302, many designers currently using the MC68302 may wish to use the QUICC in a follow-on design. The following paragraphs briefly discuss this endeavor in terms of architectural approach, hardware issues, and software issues. See Section 9 Applications for further information.

1.3.1 Architectural Approach

The QUICC is the logical extension of the MC68302, but the overall architecture and philosophy of the MC68302 design remains intact in the QUICC. The QUICC keeps the best features of the MC68302, while making the changes required to provide for the increased flexibility, integration, and performance requested by customers. Because the CPM is probably the most difficult module to learn, anyone who has used the MC68302 can easily become familiar with the QUICC since the CPM architectural approach remains intact.

The most significant architectural change made on the QUICC was the translation of the design into the standard M68300 family IMB architecture, resulting in a faster CPU and different system integration features.

Although the features of the SIM60 do not exactly correspond to those of the MC68302 SIM, they are very similar. The QUICC SIM60 combines the best MC68302 SIM features with the best MC68340 SIM features for improved performance.

appropriate timing described in this section and in Section 10 Electrical Characteristics. Additionally, BERR and HALT can be asserted together to indicate a retry termination. Refer to 4.5 Bus Exception Control Cycles for additional information on the use of these signals.

See the memory controller description in Section 6 System Integration Module (SIM60) for precautions about asserting BERR externally too early during DRAM and SRAM cycles controlled by the memory controller.

The internal bus monitor can be used to generate the BERR signal for internal and external transfers in all the following descriptions.

4.1.9.3 AUTOVECTOR (AVEC). This signal can be used to terminate interrupt acknowledge cycles, indicating that the QUICC should internally generate a vector (autovector) number to locate an interrupt handler routine. AVEC can be generated either externally or internally by the SIM60 (refer to Section 6 System Integration Module (SIM60) for additional information). AVEC is ignored during all other bus cycles.

4.2 DATA TRANSFER MECHANISM

The QUICC supports byte, word, and long-word operands, allowing access to 8-,16-, and 32-bit data ports through the use of asynchronous cycles controlled by DSACK1 and DSACK0. The QUICC also supports byte, word, and long-word operands, allowing access to 8-, 16, and 32-bit data ports through the use of synchronous cycles controlled by the fast-termination capability of the SIM60.

4.2.1 Dynamic Bus Sizing

The QUICC dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8-, 16-, and 32-bit ports. During an operand transfer cycle, the slave device signals its port size (byte, word, or long word) and indicates completion of the bus cycle to the QUICC through the use of the DSACKx inputs. Refer to Table 4-2 for DSACKx encoding.

Table 4-2. DSACKx Encoding

DSACK1	DSACK0	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle—Data Bus Port Size is 8 Bits
0	1	Complete Cycle—Data Bus Port Size is 16 Bits
0	0	Complete Cycle—Data Bus Port Size is 32 Bits

For example, if the QUICC is executing an instruction that reads a long-word operand from a long-word aligned address, it attempts to read 32 bits during the first bus cycle. (Refer to 4.2.2 Misaligned Operands for the case of a word or byte address.) If the port responds that it is 32 bits wide, the QUICC latches all 32 bits of data and continues with the next operation. If the port responds that it is 16 bits wide, the QUICC latches the 16 bits of valid data and 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 DSACKx signals to indicate

asserted in any asynchronous system. If this maximum delay time is violated, the QUICC may exhibit erratic behavior.

4.2.5 Synchronous Operation with DSACKx

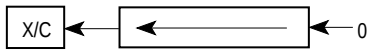
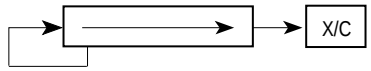
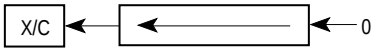
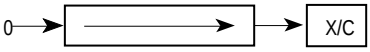
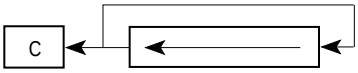
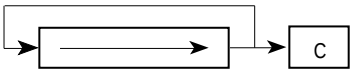
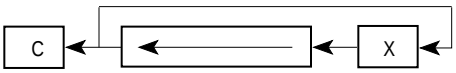
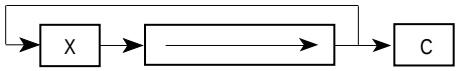
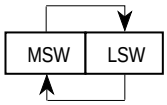
Although cycles terminated with DSACKx are classified as asynchronous, cycles terminated with DSACKx can also operate synchronously in that signals are interpreted relative to clock edges. The devices that use these cycles must synchronize the response to the QUICC clock (CLKO1) to be synchronous. Since the devices terminate bus cycles with DSACKx, the dynamic bus sizing capabilities of the QUICC are available. The minimum cycle time for these cycles is also three clocks. To support systems that use the system clock to generate DSACKx and other asynchronous inputs, the asynchronous input setup time and the asynchronous input hold time are given. If the setup and hold times are met for the assertion or negation of a signal, such as DSACKx, the QUICC is guaranteed to recognize that signal level on that specific falling edge of the system clock. If the assertion of DSACKx is recognized on a particular falling edge of the clock, valid data is latched into the QUICC (for a read cycle) on the next falling clock edge if the data meets the data setup time. In this case, the parameter for asynchronous operation can be ignored. The timing parameters are described in Section 10 Electrical Characteristics.

If a system asserts DSACKx for the required window around the falling edge of S2 and obeys the proper bus protocol by maintaining DSACKx (and/or BERR/HALT) until and throughout the clock edge that negates AS (with the appropriate asynchronous input hold time), no wait states are inserted. The bus cycle runs at its maximum speed for bus cycles terminated with DSACKx (three clocks per cycle). When BERR (or BERR and HALT) is asserted after DSACKx, BERR (and HALT) must meet the appropriate setup time prior to the falling clock edge one clock cycle after DSACKx is recognized. This setup time is critical, and the QUICC may exhibit erratic behavior if it is violated. When operating synchronously, the data-in setup and hold times for synchronous cycles may be used instead of the timing requirements for data relative to DS.

4.2.6 Fast Termination Cycles

With an external device that has a fast access time, the memory controller circuits can provide a two-clock external bus transfer. Since the memory controller circuits are driven from the system clock, the bus cycle termination is inherently synchronized with the system clock. Refer to Section 6 System Integration Module (SIM60) for more information on chip selects and the DRAM controller. To use the fast termination (cycle length is two clocks) option, an external device should be fast enough to have data ready, within the specified setup time, by the falling edge of S4. Figure 4-14 shows the DSACKx timing for a read with two wait states, followed by a fast termination read and write.

Table 5-7. Shift and Rotate Operations

Instruction	Operand Syntax	Operand Size	Operation
ASL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
SWAP	Dn	16	

5.3.3.6 BIT MANIPULATION INSTRUCTIONS. Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). All bit manipulation operations can be performed on either registers or memory. The bit number is specified as immediate data or in a data register. Register operands are 32 bits, and memory operands are 8 bits. Table 5-8 is a summary of bit manipulation instructions.

If the frame was generated by an interrupt, breakpoint, trap, or instruction exception, the SR and PC are restored to the values saved on the supervisor stack, and execution resumes at the restored PC address, with access level determined by the S-bit of the restored SR.

If the frame was generated by a bus error or an address error exception, the entire processor state is restored from the stack.

5.5 EXCEPTION PROCESSING

An exception is a special condition that pre-empts normal processing. Exception processing is the transition from normal mode program execution to execution of a routine that deals with an exception. The following paragraphs discuss system resources related to exception handling, exception processing sequence, and specific features of individual exception processing routines.

5.5.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. The VBR contains the base address of a 1024-byte exception vector table, which consists of 256 exception vectors. Sixty-four vectors are defined by the processor, and 192 vectors are reserved for user definition as interrupt vectors. Except for the reset vector, which is two long words, each vector in the table is one long word. Refer to Table 5-16 for information on vector assignment.

All exception vectors, except the reset vector, are located in supervisor data space. The reset vector is located in supervisor program space. Only the initial reset vector is fixed in the processor memory map. When initialization is complete, there are no fixed assignments. Since the VBR stores the vector table base address, the table can be located anywhere in memory. It can also be dynamically relocated for each task executed by an operating system.

Each vector is assigned an 8-bit number. Vector numbers for some exceptions are obtained from an external device; others are supplied by the processor. The processor multiplies the vector number by 4 to calculate vector offset, then adds the offset to the contents of the VBR. The sum is the memory address of the vector.

5.5.1.1 TYPES OF EXCEPTIONS. An exception can be caused by internal or external events.

An internal exception can be generated by an instruction or by an error. The TRAP, TRAPcc, TRAPV, BKPT, CHK, CHK2, RTE, and DIV instructions can cause exceptions during normal execution. Illegal instructions, instruction fetches from odd addresses, word or long-word operand accesses from odd addresses, and privilege violations also cause internal exceptions.

Sources of external exception include interrupts, breakpoints, bus errors, and reset requests. Interrupts are peripheral device requests for processor action. Breakpoints are used to support development equipment. Bus error and reset are used for access control and processor restart.

TP—BERR Frame Type

The TP field defines the class of the faulted bus operation. Two bus error exception frame types are defined. One is for faults on prefetch and operand accesses, and the other is for faults during exception frame stacking.

0 = Operand or prefetch bus fault
1 = Exception processing bus fault

MV—MOVEM in Progress

MV is set when the operand transfer portion of the MOVEM instruction is in progress at the time of a bus fault. If a prefetch bus fault occurs while prefetching the MOVEM opcode and extension word, both the MV and IN bits will be set.

0 = MOVEM was not in progress when fault occurred
1 = MOVEM was in progress when fault occurred

SZC1,SCZ0—Original Operand Size

The SZC1,SZC0 field specifies the size of the original bus cycle (i.e., the size bits of the first cycle, when a transaction is divided into two or three cycles due to bus size or operand address).

00 = Original operand size was long word
01 = Original operand size was byte
10 = Original operand size was word
11 = Unused, reserved

TR—Trace Pending

TR indicates that a trace exception was pending when a bus error exception was processed. The instruction that generated the trace will not be restarted upon return from the exception handler. This includes MOVEM and released write bus errors indicated by the assertion of either MV or RR in the SSW.

0 = Trace not pending
1 = Trace pending

B1—Breakpoint Channel 1 Pending

B1 indicates that a breakpoint exception was pending on channel 1 (external breakpoint source) when a bus error exception was processed. Pending breakpoint status is stacked, regardless of the type of bus error exception.

0 = Breakpoint not pending
1 = Breakpoint pending

B0—Breakpoint Channel 0 Pending

B0 indicates that a breakpoint exception was pending on channel 0 (internal breakpoint source) when the bus error exception was processed. Pending breakpoint status is stacked, regardless of the type of bus error exception.

0 = Breakpoint not pending
1 = Breakpoint pending

5.5.3.1 TYPES OF FAULTS. An efficient implementation of instruction restart dictates that faults on some bus cycles be treated differently than faults on other bus cycles. The CPU32+ defines four fault types: released write faults, faults during exception processing, faults during MOVEM operand transfer, and faults on any other bus cycle.

5.5.3.1.1 Type I—Released Write Faults. CPU32+ instruction pipelining can cause a final instruction write to overlap the execution of a following instruction. A write that is overlapped is called a released write. A released write fault occurs when a bus error or some other fault occurs on the released write.

Released write faults are taken at the next instruction boundary. The stacked PC is that of the next unexecuted instruction. If a subsequent instruction attempts an operand access while a released write fault is pending, the instruction is aborted and the write fault is acknowledged. This action prevents the instruction from using stale data.

The SSW for a released write fault contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
0	0	SZC1	TR	B1	B0	1	0	0	0	SZC0	SIZ		FUNC	

TR , B1, and B0 are set if the corresponding exception is pending when the bus error exception is taken. Status regarding the faulted bus cycle is reflected in the SZCx, SIZ, and FUNC fields.

The remainder of the stack contains the PC of the next unexecuted instruction, the current SR, the address of the faulted memory location, and the contents of the data buffer that was to be written to memory. This data is written on the stack in the format depicted in Figure 5-15. When a released write fault exception handler executes, the machine will complete the faulted write and then continue executing instructions wherever the PC indicates.

5.5.3.1.2 Type II—Prefetch, Operand, RMW, and MOVEP Faults. The majority of bus error exceptions are included in this category—all instruction prefetches, all operand reads, all RMW cycles, and all operand accesses resulting from execution of MOVEP (except the last write of a MOVEP Rn,⟨ea⟩ or the last write of MOVEM, which are type I faults). The TAS, MOVEP, and MOVEM instructions account for all operand writes not considered released write faults.

All type II faults cause an immediate exception that aborts the current instruction. Any registers that were altered as the result of an EA calculation (i.e., postincrement or predecrement) are restored prior to processing the bus cycle fault.

The SSW for faults in this category contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
0	0	SZC1	0	B1	B0	0	RM	IN	RW	SZC0	SIZ		FUNC	

The trace pending bit is always cleared since the instruction will be restarted upon return from the handler. Saving a pending exception on the stack causes a trace exception to be

exception frame on top of the stack, and resume execution at the exception handler address.

5.5.4 CPU32+ Stack Frames

The CPU32+ generates three different stack frames: four-word frames, six-word frames, and twelve-word bus error frames.

5.5.4.1 FOUR-WORD STACK FRAME. This stack frame is created by interrupt, format error, TRAP #n, illegal instruction, A-line and F-line emulator trap, and privilege violation exceptions. Depending on the exception type, the PC value is either the address of the next instruction to be executed or the address of the instruction that caused the exception (see Figure 5-12).

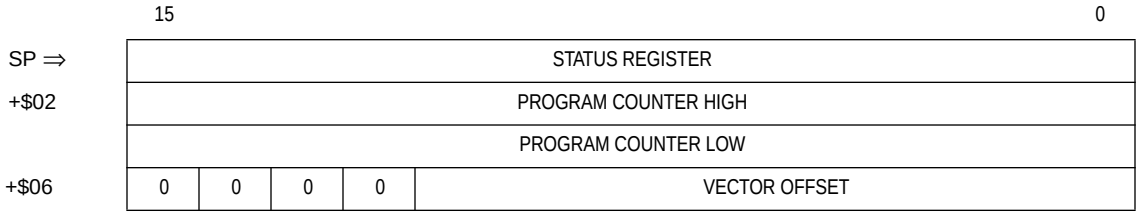


Figure 5-12. Format $0—Four-Word Stack Frame

5.5.4.2 SIX-WORD STACK FRAME. This stack frame (see Figure 5-13) is created by instruction-related traps, which include CHK, CHK2, TRAPcc, TRAPV, and divide-by-zero, and by trace exceptions. The faulted instruction PC value is the address of the instruction that caused the exception. The next PC value (the address to which RTE returns) is the address of the next instruction to be executed.

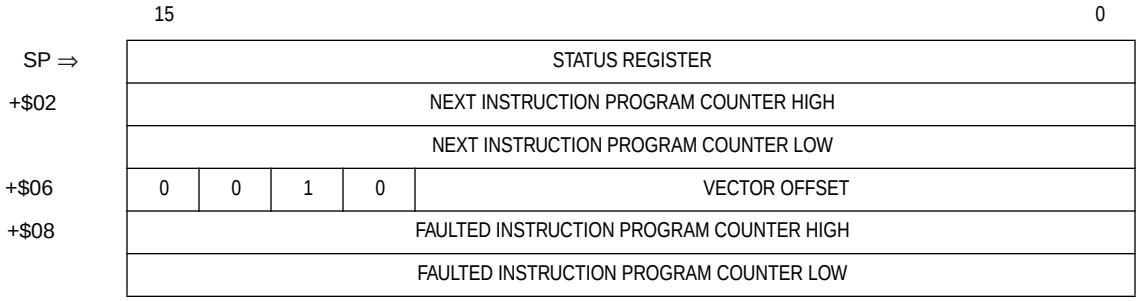


Figure 5-13. Format $2—Six-Word Stack Frame

Hardware breakpoints also utilize this format. The faulted instruction PC value is the address of the instruction executing when the breakpoint was sensed. Usually this is the address of the instruction that caused the breakpoint, but, because released writes can overlap following instructions, the faulted instruction PC may point to an instruction following the instruction that caused the breakpoint. The address to which RTE returns is the address of the next instruction to be executed.

5.5.4.3 BUS ERROR STACK FRAME. This stack frame is created when a bus cycle fault is detected. The CPU32+ bus error stack frame differs significantly from the equivalent stack

frames of other M68000 family members. The only internal machine state required in the CPU32+ stack frame is the bus controller state at the time of the error and a single register.

Bus operation in progress at the time of a fault is conveyed by the SSW.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP	MV	SZC1	TR	B1	B0	RR	RM	IN	RW	SZC0	SIZ		FUNC		

The bus error stack frame is 12 words in length. There are three variations of the frame, each distinguished by different values in the SSW TP and MV fields.

An internal transfer count register appears at location SP + $14 in all bus error stack frames. The register contains an 8-bit microcode revision number and, for type III faults, an 8-bit transfer count. Register format is shown in Figure 5-14.

15	8	7	0
MICROCODE REVISION NUMBER		TRANSFER COUNT	

Figure 5-14. Internal Transfer Count Register

The microcode revision number is checked before a bus error stack frame is restored via RTE. In a multiprocessor system, this check ensures that a processor using stacked information is at the same revision level as the processor that created it.

The transfer count is ignored unless the MV bit in the stacked SSW is set. If the MV bit is set, the least significant byte of the internal register is reloaded into the MOVEM transfer counter during RTE execution.

For faults occurring during normal instruction execution (both prefetches and non-MOVEM operand accesses), SSW TP,MV = 00. Stack frame format is shown in Figure 5-15.

Faults that occur during the operand portion of the MOVEM instruction are identified by SSW TP,MV = 01. Stack frame format is shown in Figure 5-16.

When a bus error occurs during exception processing, SSW TP,MV = 10. The frame shown in Figure 5-17 is written below the faulting frame. Stacking begins at the address pointed to by SP – 6 (SP value is the value before initial stacking on the faulted frame).

The frame can have either four or six words, depending on the type of error. Four-word stack frames do not include the faulted instruction PC. (The internal transfer count register is located at SP + $10 and the SSW is located at SP + $12.)

The fault address of a dynamically sized bus cycle is the address of the upper byte, regardless of the byte that caused the error.

5.6.2.2.1 External $\overline{\text{BKPT}}$ Signal. Once enabled, BDM is initiated whenever assertion of $\overline{\text{BKPT}}$ is acknowledged. If BDM is disabled, a breakpoint exception (vector $0C) is acknowledged. The $\overline{\text{BKPT}}$ input has the same timing relationship to the data strobe trailing edge as read cycle data. There is no breakpoint acknowledge bus cycle when BDM is entered.

5.6.2.2.2 BGND Instruction. An illegal instruction, $4AFA, is reserved for use by development tools. The CPU32+ defines $4AFA (BGND) to be a BDM entry point when BDM is enabled. If BDM is disabled, an illegal instruction trap is acknowledged. Illegal instruction traps are discussed in 5.5.2.8 Illegal or Unimplemented Instructions.

5.6.2.2.3 Double Bus Fault. The CPU32+ normally treats a double bus fault (two bus faults in succession) as a catastrophic system error and halts. When this condition occurs during initial system debug (a fault in the reset logic), further debugging is impossible until the problem is corrected. In BDM, the fault can be temporarily bypassed so that its origin can be isolated and eliminated.

5.6.2.3 ENTERING BDM. When the processor detects a $\overline{\text{BKPT}}$ or a double bus fault or decodes a BGND instruction, it suspends instruction execution and asserts the FREEZE output. FREEZE assertion is the first indication that the processor has entered BDM. Once FREEZE has been asserted, the CPU enables the serial communication hardware and awaits a command.

The CPU writes a unique value indicating the source of BDM transition into temporary register A (ATEMP) as part of the process of entering BDM. A user can poll ATEMP and determine the source (see Table 5-20) by issuing a read system register command (RSREG). ATEMP is used in most debugger commands for temporary storage—it is imperative that the RSREG command be the first command issued after transition into BDM.

Table 5-20. Polling the BDM Entry Source

Source	ATEMP 31–16	ATEMP 15–0
Double Bus Fault	SSW*	$FFFF
BGND Instruction	$0000	$0001
Hardware Breakpoint	$0000	$0000

*SSW is described in detail in 5.5.3 Fault Recovery.

A double bus fault during initial SP/PC fetch sequence is distinguished by a value of $FFFFFFFF in the current instruction PC. At no other time will the processor write an odd value into this register.

5.6.2.4 COMMAND EXECUTION. Figure 5-21 summarizes BDM command execution. Commands consist of one 16-bit operation word and can include one or more 16-bit extension words. Each incoming word is read as it is assembled by the serial interface. The microcode routine corresponding to a command is executed as soon as the command is complete. Result operands are loaded into the output shift register to be shifted out as the next command is read. This process is repeated for each command until the CPU returns to normal operating mode.

X = There is one bus cycle for byte and word operands and two bus cycles for long operands. For long bus cycles, add two clocks to the tail and to the number of cycles. Timing is calculated with the CPU32+ in 16-bit mode.
< = Maximum time (certain data or mode combinations may execute faster).
su = The execution time is identical for signed or unsigned operands.

* = These instructions have an additional save operation that other instructions do not have. To calculate total instruction time, calculate save, ⟨ea⟩, and operation execution times, then combine in the order listed, using equations in 5.7.1.6 Instruction Execution Time Calculation. A save operation is not run for long-word divide and multiply instructions when ⟨FEA⟩ = Dn.

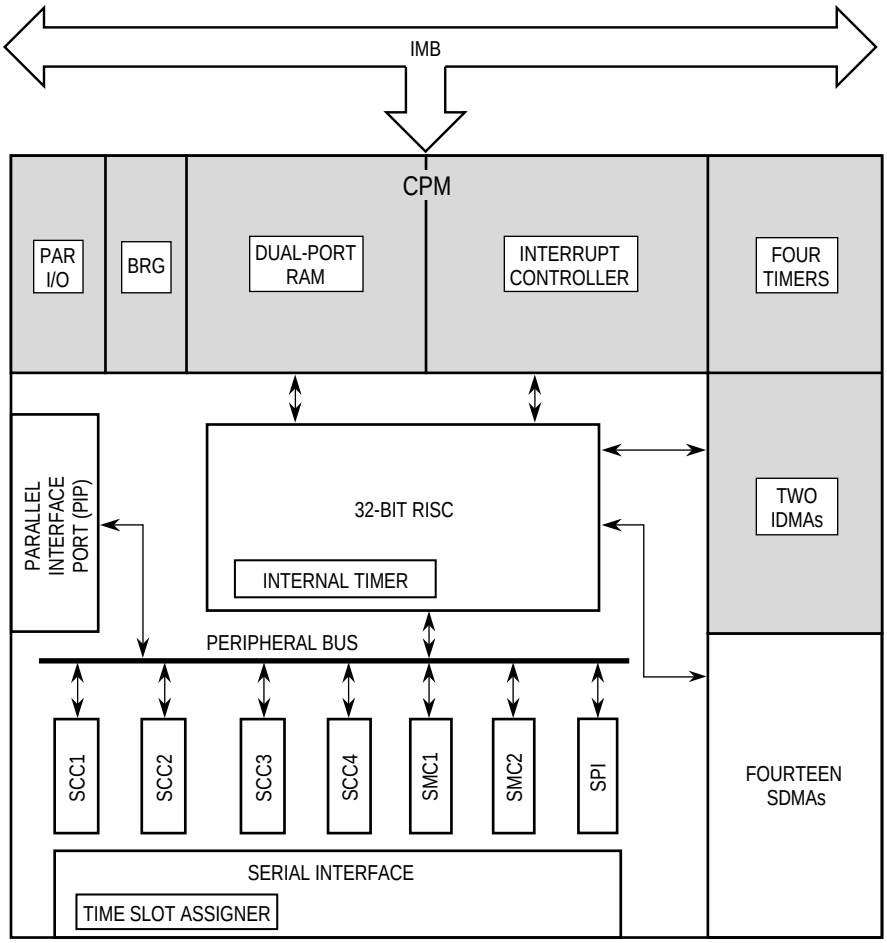
5.7.2.6 IMMEDIATE ARITHMETIC/LOGIC INSTRUCTIONS. The immediate arithmetic/logic instruction table indicates the number of clock periods needed for the processor to fetch the source immediate data value and to perform the specified arithmetic/logic instruction using the specified addressing mode. Footnotes indicate when to account for the appropriate fetch effective or fetch immediate EA times. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
MOVEQ#, Dn	0	0	2(0/1/0)
ADDQ #, Rn	0	0	2(0/1/0)
ADDQ #, ⟨FEA⟩	0	3	5(0/1/x)
SUBQ #, Rn	0	0	2(0/1/0)
SUBQ #, ⟨FEA⟩	0	3	5(0/1/x)
ADDI #, Rn	0	0	2(0/1/0)∗
ADDI #, ⟨FEA⟩	0	3	5(0/1/x)∗
ANDI #, Rn	0	0	2(0/1/0)∗
ANDI #, ⟨FEA⟩	0	3	5(0/1/x)∗
EORI #, Rn	0	0	2(0/1/0)∗
EORI #, ⟨FEA⟩	0	3	5(0/1/x)∗
ORI #, Rn	0	0	2(0/1/0)∗
ORI #, ⟨FEA⟩	0	3	5(0/1/x)∗
SUBI #, Rn	0	0	2(0/1/0)∗
SUBI #, ⟨FEA⟩	0	3	5(0/1/x)∗
CMPI #, Rn	0	0	2(0/1/0)∗
CMPI #, ⟨FEA⟩	0	3	5(0/1/x)∗

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles. Timing is calculated with the CPU32+ in 16-bit mode.
∗ = An # fetch EA time must be added for this instruction: ⟨FEA⟩ + ⟨FEA⟩ + ⟨OPER⟩

5.7.2.7 BINARY-CODED DECIMAL AND EXTENDED INSTRUCTIONS. The BCD and extended instruction table indicates the number of clock periods needed for the processor to perform the specified operation using the specified addressing mode. No additional tables are needed to calculate total effective execution time for these instructions. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

- Serial Peripheral Interface (SPI) for Synchronous Interchip Communication
- Fourteen Serial Direct Memory Access (SDMA) Channels Support the SCC, SMCs, and SPI
- Two Independent Direct Memory Access (IDMA) Channels Support External Memory and Peripherals
- A Command Set Register Supports the RISC, IDMA, SCCs, SMCs, and SPI
- Four General-Purpose 16-Bit Timers or Two 32-Bit Timers
- Internal Timers to Implement Up to 16 Additional Timers
- General-Purpose Parallel Port for Parallel Protocols such as Centronics (Can Also Be Used as Standard Parallel I/O)
- CPM Interrupt Controller
- 2.5-kbyte Dual-Port RAM
- Twelve Parallel I/O Lines with Interrupt Capability



NOTE: The term "CP" refers to the nonshaded portion of the CPM.

Figure 7-1. CPM Block Diagram

NOTE

An interrupt will only be generated if the SDMA bit is set in the CP interrupt mask register.

INTB—Interrupt Breakpoint

This bit is the enable bit for the SBKP status bit in the SDSR.

0 = A zero masks the interrupt generated by the corresponding bit in the SDSR. When a breakpoint is recognized while the SDMA is bus master, the channel does not generate an interrupt to the QUICC interrupt controller. The SBKP bit is still set in the SDSR.

1 = When a breakpoint is recognized while the SDMA is bus master, the channel generates an interrupt to the QUICC interrupt controller and sets the SBKP bit in the SDSR.

NOTE

An interrupt will only be generated if the SDMA bit is set in the CP interrupt mask register. The interrupt can suspend SDMA activity immediately if it is programmed to be at a higher level than the SDMA channels. Alternatively, the interrupt can be processed after the SDMA transfer is complete.

7.7.2.2 SDMA STATUS REGISTER (SDSR). Shared by all 14 SDMA channels, the SDSR is an 8-bit register used to report events recognized by the SDMA controller. On recognition of an event, the SDMA sets its corresponding bit in the SDSR (regardless of the INTE, INTB, and INTR bits in the SDCR). The SDSR is a memory-mapped register that may be read at any time. A bit is reset by writing a one and is left unchanged by writing a zero. More than one bit may be reset at a time, and the register is cleared by reset.

7	6	5	4	3	2	1	0
—					RINT	SBER	SBKP

Bits 7–3—Reserved

RINT—Reserved Interrupt

This status bit is reserved for factory testing. RINT is cleared by writing a one; writing a zero has no effect.

SBER—SDMA Channel Bus Error

This bit indicates that the SDMA channel terminated with an error during a read or write cycle. The SDMA bus error address can be read from the SDAR. SBER is cleared by writing a one; writing a zero has no effect.

SBKP—SDMA Breakpoint

This bit indicates that the breakpoint signal was asserted during an SDMA transfer. SBKP is cleared by writing a one; writing a zero has no effect.

7.7.2.3 SDMA ADDRESS REGISTER (SDAR). The 32-bit read-only SDAR shows the system address that was accessed during an SDMA bus error. It is undefined at reset.

7.10.21.7.2 Reception Errors. The following paragraphs describe various types of reception errors.

Overrun Error**.** The SCC maintains an internal FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when 8 or 32 bits (according to the RFW bit in the GSMR) are received in the FIFO. If a FIFO overrun occurs, the SCC writes the received data byte to the internal FIFO over the previously received byte. The previous character and its status bits are lost. Following this, the channel closes the buffer, sets the OV bit in the BD, and generates the RX interrupt (if enabled). The receiver then enters hunt mode immediately.

CD Lost During Message Reception**.** When this error occurs, the channel terminates message reception, closes the buffer, sets the CD bit in the BD, and generates the RX interrupt (if enabled). This error has the highest priority; the rest of the message is lost, and no other errors are checked in the message. The receiver then enters hunt mode immediately.

7.10.21.8 TRANSPARENT MODE REGISTER (PSMR). The PSMR is called the transparent mode register when an SCC is programmed for transparent mode. However, since all transparent mode selections are in the GSMR, this register is not used by the transparent controller. If transparent mode is only selected for the transmitter/receiver, then the transmitter/receiver may be programmed to support another protocol. In such a case, the PSMR may be used for that other protocol.

7.10.21.9 TRANSPARENT RECEIVE BUFFER DESCRIPTOR (RX BD). •The CP reports information about the received data for each buffer using an Rx BD. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

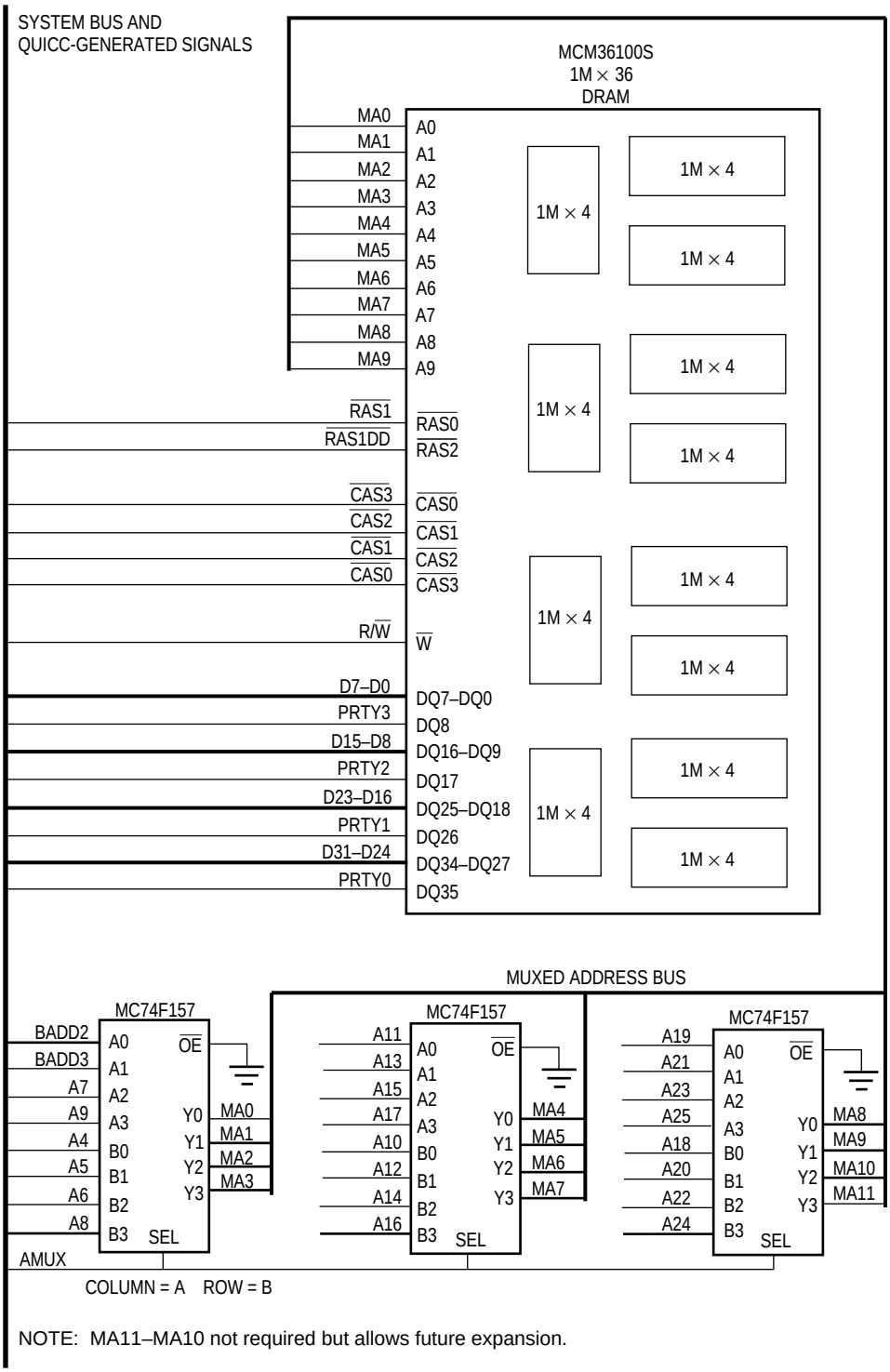
1. Detecting an error
2. Detecting a full receive buffer
3. Issuing the ENTER HUNT MODE command
4. Issuing the CLOSE Rx BD command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	**I**	L	**F**	**CM**	—	DE	—	—	NO	—	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX DATA BUFFER POINTER*															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

E—Empty

0 = The data buffer associated with this Rx BD has been filled with received data, or data reception has been aborted due to an error condition. The CPU32+ core is free to examine or write to any fields of this Rx BD. The CP will not use this BD again while the E-bit remains zero.



NOTE: MA11–MA10 not required but allows future expansion.

Figure 9-15. 4-Mbyte DRAM Bank—36 Bits Wide

9.6.7.2 CONFIGURING THE MEMORY CONTROLLER. The following paragraphs describe configuring the memory controller.

For information on configuring the global memory register (GMR), refer to 9.1 Minimum System Configuration.

The memory controller status register (MSTAT) is used for reporting parity errors and does not require initialization.

Eight base registers (BRs) exist, one for each memory bank. BR6 and BR7 for $\overline{CS6}$ and $\overline{CS7}$ will be specified with the 53C90 address being $04001000. Please refer to 9.1 Minimum System Configuration for DRAM and other memory configurations.

BR7 = $0400104D. Address decoded is $04001xxx, function codes are xxxx (don't care, will be masked in OR7), TRLXQ and CSNTQ are set, parity is enabled, read and write accesses are allowed, and this base register is valid.

BR6 = $04001805. Same as BR7, except TRLXQ and CSNTQ are not set and the next consecutive 2K memory block is selected.

Eight option registers (ORs) exist, one for each memory bank. The following information is valid for registers OR6 and OR7:

DSSEL should be 0.

SPS1–SPS0 should be 10 (indicating port size is 8 bits).

PGME should be 0 since this is not DRAM.

BCYC1–BCYC0 are not used and should be cleared.

FCM3–FCM0 may be cleared to zeros to allow the chip select or $\overline{RAS}$ line to assert on all function codes, except CPU space (interrupt acknowledge). It is advisable to program FCM3–FCM0 to zeros, at least during the initial stages of debugging.

The AM27–AM11 bits will mask the address if they are cleared. In this application, they are all set to allow decoding.

The TCYC bits should be set to determine the number of wait states required—one wait state on $\overline{CS7}$ (0010) and no wait states on $\overline{CS6}$ (0001).

Therefore, OR7 = $2FFFF804 and OR6 = $1FFFF804.

9.7 USING THE QUICC AS A TAP CONTROLLER FOR BOARD SELF-TEST

An assembled board is often tested with complex test equipment using a unique test port or a bed-of-nails fixture. This procedure becomes more difficult as device packages and features become smaller. The objective of the JTAG standard is to define a boundary scan architecture that can be adopted as a part of an integrated circuit to perform both an in-circuit test and a verification of the interconnection between different devices.

The JTAG standard defines test logic that can be integrated into a device to perform:

1. Testing of the interconnection between devices once they have been mounted on a printed circuit board or any other substrate.

This design also uses the RAS1 double-drive capability, whereby the RAS1DD signal is output by the QUICC on the BCLRI pin to increase the effective drive capability of the RAS1 signal. The RAS1 line should be programmed to respond to a 4-Mbyte address space.

After power-on reset, the software must wait the required time before accessing the DRAM. The required eight read cycles must then be performed either in software or by waiting for the refresh controller to perform these accesses.

9.8.2.6 DRAM DEVICES. Figure 9-33 shows the interface to a standalone DRAM device. In this case the MCM54260 256K × 16 DRAM device is chosen. This allows a full 32-bit wide DRAM solution using only two DRAM devices, with byte writes still supported using the upper and lower CAS pins. Both the MC68EC030 and the QUICC can access the DRAM array. The RAS1 line should be programmed to respond to a 1-Mbyte address space.

The address multiplexing scheme shown is the same as that for the DRAM SIMM. No parity support is provided in this case. The RAS1DD signal is not used in this case, since only two devices are supported.

After power-on reset, the software must wait the required time before accessing the DRAM, and then perform the required eight read cycles, either in software or by waiting for the refresh controller to perform these accesses.

9.8.3 Software Configuration

The following paragraphs discuss a number of key points for a software engineer desiring to initialize the system. The only items discussed are those that are required to allow the previously discussed hardware configuration.

9.8.3.1 BASIC INITIALIZATION. The following register initializations are basic to all types of applications.

The module base address register (MBAR) should be set as desired. However, the QUICC 8-Kbyte block should not overlap any memory array.

The module base address register enable (MBARE) should not be accessed.

In the module configuration register (MCR), ASTM and BSTM should be set to indicate synchronous operation. SHEN1–SHEN0 should be cleared.

In the system protection control register (SYPCR), DBFE should be cleared. BME should be set. If the software watchdog is used, the SWRI bit should be set.

The periodic interrupt control register (PICR) may be set as desired.

The port E pin assignment register (PEPAR) should be set to $51C0. This configures three IOUTx lines to go out on the unused parity pins, the RAS1DD pin, WE lines instead of the A31–A28 lines, the AMUX pin (assuming DRAM is used in the system; otherwise, the OE function should be programmed), four CASx lines, CS7, and AVECO.