**Welcome to E-XFL.COM**

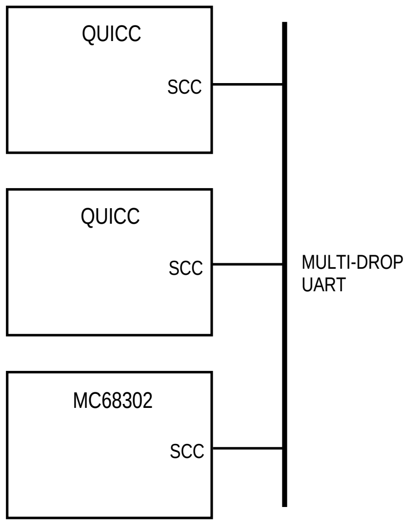
## Understanding Embedded - Microprocessors

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

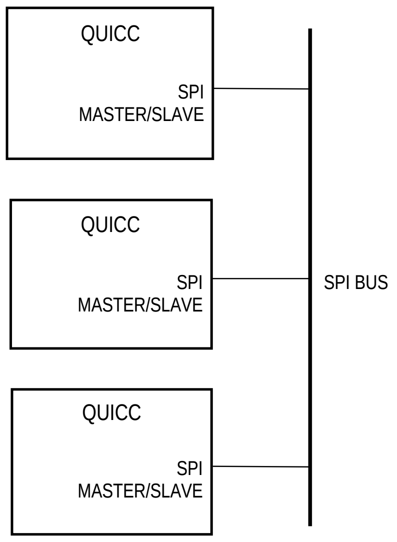
| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | CPU32+ |
| Number of Cores/Bus Width | 1 Core, 32-Bit |
| Speed | 25MHz |
| Co-Processors/DSP | Communications; CPM |
| RAM Controllers | DRAM |
| Graphics Acceleration | No |
| Display & Interface Controllers | - |
| Ethernet | 10Mbps (1) |
| SATA | - |
| USB | - |
| Voltage - I/O | 5.0V |
| Operating Temperature | 0°C ~ 70°C (TA) |
| Security Features | - |
| Package / Case | 357-BBGA |
| Supplier Device Package | 357-PBGA (25x25) |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68360zq25lr2 |



NOTES:
1. Simple LAN based on UART mode.
2. Ninth bit is an "address" bit.

**Figure 1-8. UART LAN Implementation**

Figure 1-9 shows how the SPIs on the QUICC can be used to connect devices together into a local bus. The SPI exists on many other Motorola devices, such as the MC68HC11 micro-controller, and a number of peripherals such as A/D and D/A converters, LED drivers, LCD drivers, real-time clocks, serial EEPROM, PLL frequency synthesizers, and shift registers.



NOTE: SPI bus configuration—each QUICC can be the master in turn.

**Figure 1-9. SPI Local Bus Implementation**

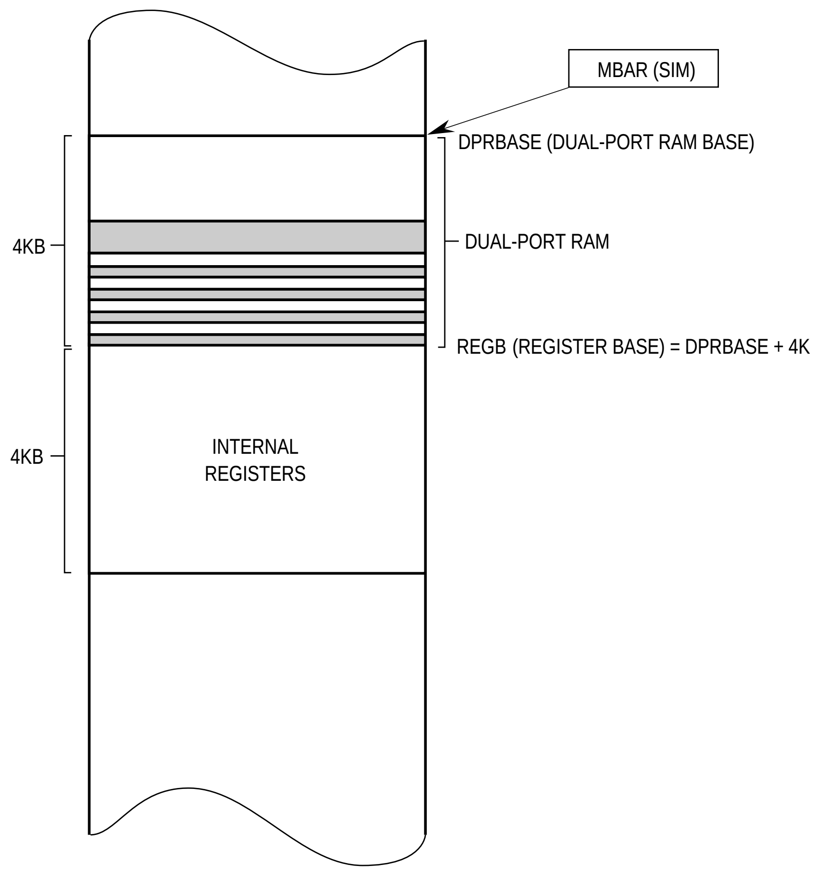

MBAR (SIM)
DPRBASE (DUAL-PORT RAM BASE)
4KB
DUAL-PORT RAM
REGB (REGISTER BASE) = DPRBASE + 4K
4KB
INTERNAL REGISTERS

**Figure 3-1. QUICC Memory Map**

## 3.1 DUAL-PORT RAM MEMORY MAP

The internal 2816-byte (2560-byte on REV A and B mask) dual-port RAM is partitioned to 1792 bytes (1536 bytes on REV A and B mask) of system RAM, 256-byte microcode scratch area, and 768 bytes of parameter RAM (see Table 3-1). Its base address, called dual-port RAM base (DPRBASE), is the address pointed to by the MBAR.

**NOTE**

Rev A mask is C63T, Rev B mask are C69T, and F35G

The system RAM may be used for microcode program area, data area, and buffer descriptors (BDs). It may be partitioned in several ways, allowing programmable partition sizes to fit the system requirements. This is described in Section 7 Communication Processor Module (CPM).

received $\overline{\text{BG}}$ through the arbitration process, and $\overline{\text{BGACK}}$ must be inactive, indicating that no other bus master has claimed ownership of the bus.

Figure 4-34 is a flowchart showing the detail involved in bus arbitration for a single device. This technique allows processing of bus requests during data transfer cycles.

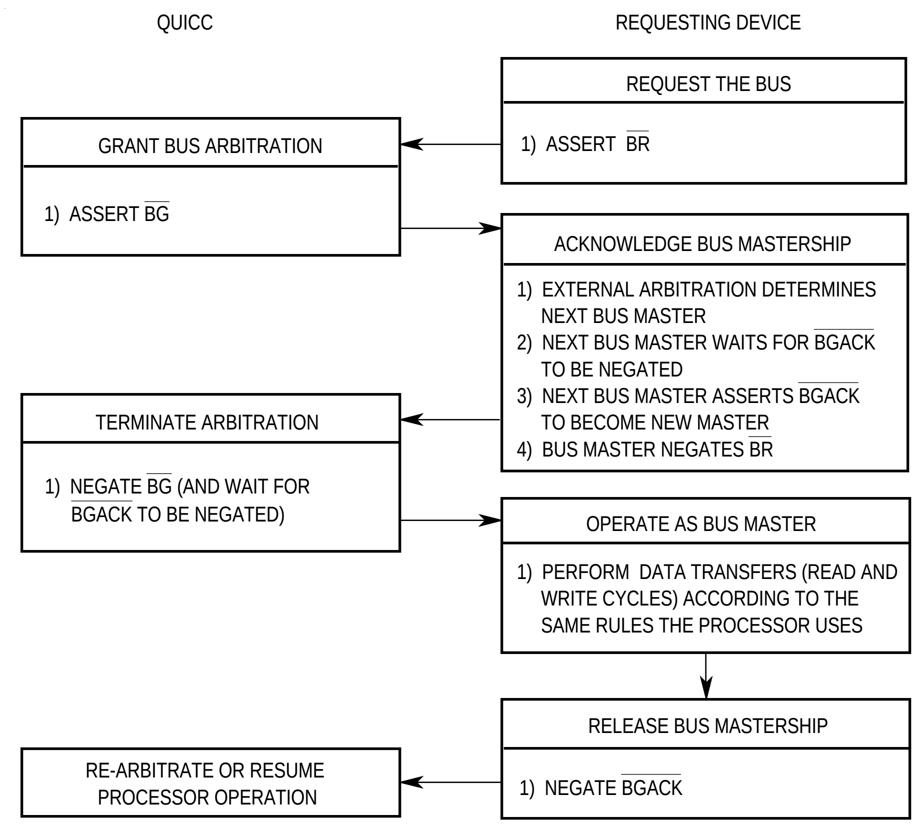

QUICC
REQUESTING DEVICE
REQUEST THE BUS
1) ASSERT $\overline{\text{BR}}$
GRANT BUS ARBITRATION
1) ASSERT $\overline{\text{BG}}$
ACKNOWLEDGE BUS MASTERSHIP
1) EXTERNAL ARBITRATION DETERMINES NEXT BUS MASTER
2) NEXT BUS MASTER WAITS FOR $\overline{\text{BGACK}}$ TO BE NEGATED
3) NEXT BUS MASTER ASSERTS $\overline{\text{BGACK}}$ TO BECOME NEW MASTER
4) BUS MASTER NEGATES $\overline{\text{BR}}$
TERMINATE ARBITRATION
1) NEGATE $\overline{\text{BG}}$ (AND WAIT FOR $\overline{\text{BGACK}}$ TO BE NEGATED)
OPERATE AS BUS MASTER
1) PERFORM DATA TRANSFERS (READ AND WRITE CYCLES) ACCORDING TO THE SAME RULES THE PROCESSOR USES
RELEASE BUS MASTERSHIP
1) NEGATE $\overline{\text{BGACK}}$
RE-ARBITRATE OR RESUME PROCESSOR OPERATION

**Figure 4-34. Bus Arbitration Flowchart for Single Request**

The QUICC has a synchronous arbitration timing mode to reduce the $\overline{\text{BR}}$ to $\overline{\text{BG}}$ delay to one clock in the idle bus case (see Figure 4-35). Figure 4-36 illustrates the active bus case.

$\overline{\text{BR}}$ is negated at the time that $\overline{\text{BGACK}}$ is asserted. This type of operation applies to a system consisting of the QUICC and one device capable of bus mastership. In a system having a number of devices capable of bus mastership, $\overline{\text{BR}}$ from each device can be wire-ORed to the QUICC. In such a system, more than one bus request could be asserted simultaneously. $\overline{\text{BG}}$ is negated a few clock cycles after the transition of $\overline{\text{BGACK}}$. However, if bus requests are still pending after the negation of $\overline{\text{BG}}$, the QUICC asserts another $\overline{\text{BG}}$ within a few clock cycles after it was negated. This additional assertion of $\overline{\text{BG}}$ allows external arbitration circuitry to select the next bus master before the current bus master has finished using the bus. The following paragraphs provide additional information about the three steps in the arbitration process. Bus arbitration requests are recognized during normal processing, $\overline{\text{HALT}}$ assertion, and when the CPU32+ has halted due to a double bus fault.

The register is incremented using unsigned arithmetic and will roll over if an overflow occurs. For example, if a register contains $FFFFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one, two, or four, depending on the SSIZE bits and the starting address in this register.

The SAPR may be initialized by the host processor or by the RISC controller via a buffer descriptor's ring structure when the RCI bit is set for special buffer handling modes.

**7.6.2.4 DESTINATION ADDRESS POINTER REGISTER (DAPR).** The DAPR contains 32 address bits of the destination operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA write cycle, the address on the master address bus is driven from this register. The DAPR may be programmed by the DAPI bits to be incremented or remain constant after each operand transfer.

The register is incremented using unsigned arithmetic and will roll over if overflow occurs. For example, if a register contains $FFFFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one, two, or four, depending on the DSIZE bit and the starting address.

The DAPR may be initialized by the host processor or by the RISC controller via a buffer descriptor's ring structure when the RCI bit is set for special buffer handling modes.

**7.6.2.5 FUNCTION CODE REGISTER (FCR).** Each IDMA channel has an 8-bit FCR that is initialized to $00 at reset.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DFC3–DFC0 | | | | SFC3–SFC0 | | | |

During an IDMA bus cycle, the SFC and DFC bits define the source and destination function code values that are output by the IDMA and the appropriate address registers. The address space on the function code lines may be used by an external memory management unit (MMU) or other memory-protection device to translate the IDMA logical addresses to proper physical addresses. The function code value programmed into the FCR is placed on pins FC3–FC0 during a bus cycle to further qualify the address bus value.

**NOTES**

This register is typically set to 1xxx1xxxb to cause the IDMA to operate in the DMA function code space, as opposed to a CPU program or data space.

To keep interrupt acknowledge cycles unique in the system, do not set this register to $77.

**7.6.2.6 BYTE COUNT REGISTER (BCR).** This 32-bit register specifies the number of bytes of data to be transferred by the IDMA. The largest value that can be specified is 4 Gbytes (BCR = $00000000). This register is decremented once for each byte transferred successfully, for a total of 1, 2, or 4 per operand transfer. BCR may be even or odd as desired. The

—CSR1 = $FF. Clear any CSR bits that are currently set.
—CMAR1 = $00. Disable interrupts for this example.
—CMR1 = $47A1. Internal maximum transfer rate; starts IDMA.

| Bus Access # | Address (Hex) | Operation | No. Bytes | No. Bytes in DHR |
|---|---|---|---|---|
| 1 | $00000001 | Read | 1 | 1 |
| 2 | $00000002 | Read | 2 | 3 |
| 3 | $20000000 | Write | 2 | 1 |
| 4 | $00000002 | Write | 1 | 0 |

Example 2. This more complicated example shows how packing is performed when the source and destination sizes are the same—long word. This example also shows the entire 7-byte DHR in use. The source address is $00000000, and the destination address is $20000003. The number of bytes to be transferred is 16.

IDMA channel 1 initialization required for this example:

- ICCR = $0720. Recommended normal configuration.
- FCR1 = $89. Source function code is 1000; destination function code is 1001.
- SAPR1 = $00000000. Source address.
- DAPR1 = $20000003. Destination address.
- BCR1 = $00000010. Byte transfer count.
- CSR1 = $FF. Clear any CSR bits that are currently set.
- CMAR1 = $00. Disable interrupts for this example.
- CMR1 = $4701. Internal maximum transfer rate; starts IDMA.

| Bus Access # | Address (Hex) | Operation | No. Bytes | No. Bytes in DHR |
|---|---|---|---|---|
| 1 | $00000000 | Read | 4 | 4 |
| 2 | $20000003 | Write | 1 | 3 |
| 3 | $00000004 | Read | 4 | 7 |
| 4 | $20000004 | Write | 4 | 3 |
| 5 | $00000008 | Read | 4 | 7 |
| 6 | $20000008 | Write | 4 | 3 |
| 7 | $0000000C | Read | 4 | 7 |
| 8 | $2000000C | Write | 4 | 3 |
| 9 | $20000010 | Write | 2 | 1 |
| 10 | $20000012 | Write | 1 | 0 |

Example 3. This example shows how packing operates when the source and destination sizes are different. The source address is $00000002, and the destination address is $20000002. The source size is long word, and the destination size is byte. The number of bytes to be transferred is 8.

CRTx bits, and program the GRx bits to transfer the D channel grant to the SCC that supports this channel. The user should mark the received bit, which is the grant bit, by programming the channel select bits of the SI RAM to 111 for an internal assertion of a strobe on this bit. This bit will be sampled by the SI and transferred to the D channel SCC as the grant. The bit is generally bit 4 of the C/I in channel 2 of GCI, but any other bit may be selected using the SI RAM.

For example, assuming SCC1 is connected to the D channel, SCC2 is connected to the B1 channel, and SCC4 is connected to the B2 channel, SMC1 is used to handle the C/I channels, and the D channel grant is on bit 4 of the C/I on SCIT channel 2, the initialization sequence is as follows:

1. Program the SI RAM. Write all entries that are not used with $0001, setting the LST bit and disabling the routing function.

| Entry | RAM Word | | | | | | |
|---|---|---|---|---|---|---|---|
| No. | SWTR | SSEL | CSEL | CNT | BYT | LST | Description |
| 1 | 0 | 0000 | 010 | 0000 | 1 | 0 | 8 Bits SCC2 |
| 2 | 0 | 0000 | 100 | 0000 | 1 | 0 | 8 Bits SCC4 |
| 3 | 0 | 0000 | 101 | 0000 | 1 | 0 | 8 Bits SMC1 |
| 4 | 0 | 0000 | 001 | 0001 | 0 | 0 | 2 Bits SCC1 |
| 5 | 0 | 0000 | 101 | 0101 | 0 | 0 | 6 Bits SMC1 |
| 6 | 0 | 0000 | 000 | 0110 | 1 | 0 | Skip 7 Bytes |
| 7 | 0 | 0000 | 000 | 0001 | 0 | 0 | Skip 2 Bits |
| 8 | 0 | 0000 | 111 | 0000 | 0 | 1 | D Grant Bit |

**NOTE**

Since GCI requires the same routing for both receive and transmit, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI RAM beginning at addresses 0 and 128, respectively.

2. SIMODE = $000080E0. Only TDMa is used; SMC1 is connected. SCIT mode is used in this example.

**NOTE**

If SCIT mode is not used, delete the last three entries of the SI RAM and set the LST bit in the new last entry.

3. SICR = $400040C0. SCC4, SCC2, and SCC1 are connected to the TSA. SCC1 supports the grant mechanism since it is on the D channel.
4. PAODR bit 6 = 1. Configures L1TXDa to an open-drain output.
5. PAPAR bits 6, 7, and 8 = 1. Configures L1TXDa, L1RXDa, and L1RCLKa.
6. PADIR bits 6 and 7 = 1. PADIR bit 8 = 0. Configures L1TXDa, L1RXDa, and L1RCLKa.

TPL—Tx Preamble Length

The TPL bits determine the length of the preamble configured by the TPP bits.

000 = No preamble (default)
001 = 8 bits (1 byte)
010 = 16 bits (2 bytes)
011 = 32 bits (4 bytes)
100 = 48 bits (6 bytes) (Select this setting for Ethernet operation.)
101 = 64 bits (8 bytes)
110 = 128 bits (16 bytes)
111 = Reserved

TPP—Tx Preamble Pattern

The TPP bits determine what, if any, bit pattern should precede the start of each transmit frame. The preamble pattern will be sent prior to the first flag/sync of the frame. TPP is ignored if the SCC is programmed to UART mode. The length of the preamble is programmed in TPL. The preamble pattern is typically transmitted to a receiving station that uses a DPLL for clock recovery. The receiving DPLL uses the regular pattern of the preamble to help it lock onto the received signal in a short, predictable time period.

00 = All zeros
01 = Repeating 10's (Select this setting for Ethernet operation.)
10 = Repeating 01's
11 = All ones (Select this setting for LocalTalk operation.)

Tend—Transmitter Frame Ending

This bit is intended particularly for the NMSI transmitter encoding of the DPLL. Tend determines whether the TXD line should idle in a high state or in an encoded ones state (which may be either high or low). It may, however, be used with other encodings besides NMSI.

0 = Default operation. The TXD line is encoded only when data is transmitted (including the preamble and opening and closing flags/syncs). When no data is available to transmit, the line is driven high.
1 = The TXD line is always encoded (even when idles are transmitted).

TDCR—Transmit Divide Clock Rate

The TDCR bits determine the divider rate of the transmitter. If the DPLL is not used, the 1× value should be chosen, except in asynchronous UART mode where 8×, 16×, or 32× must be chosen. The user should program TDCR to equal RDCR in most applications.

If the DPLL is used in the application, the selection of TDCR depends on the encoding. NRZI usualy requires 1×; whereas, FM0/FM1, Manchester, and Differential Manchester allow 8×, 16×, or 32×. The 8× option allows highest speed; whereas, the 32× option provides the greatest resolution. TDCR is usually equal to RDCR to allow the same clock frequency source to control both the transmitter and receiver.

00 = 1× clock mode (Only NRZ or NRZI encodings are allowed.)
01 = 8× clock mode
10 = 16× clock mode (normally chosen for UART and AppleTalk)
11 = 32× clock mode

generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

1. Receiving a user-defined control character
2. Detecting an error
3. Detecting a full receive buffer
4. Issuing the ENTER HUNT MODE command
5. Issuing the CLOSE Rx BD command

| 212 | 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.

1 = The data buffer associated with this Rx BD is empty, or reception is currently in progress. This Rx BD and its associated receive buffer are owned by the CP. Once the E-bit is set, the CPU32+ core should not write any fields of this Rx BD.

Bits 14, 8, 6, 5—Reserved

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.

1 = This is the last BD in the Rx BD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table (the BD pointed to by RBASE). The number of Rx BDs in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.

1 = The RX bit in the BISYNC event register will be set when this buffer has been closed by the BISYNC controller. The RX bit can cause an interrupt if it is enabled.

L—Last in Frame

This bit is set by the transparent controller when this buffer is the last in a frame. This implies the negation of CD in envelope mode or the reception of an error, in which (according to the PAD bit in the Tx BD and the PAD value in the parameter RAM). PADs will be added to make the transmit frame MINFLR bytes in length.

MAXD1. This parameter gives the user the ability to stop system bus writes from occurring after a frame has exceeded a certain size. The value of this register is valid only if an address match was detected. The Ethernet controller checks the length of an incoming Ethernet frame against the user-defined value given in this 16-bit register. Typically, this register is set to 1518 decimal. If this limit is exceeded, the remainder of the incoming frame is discarded. The Ethernet controller waits to the end of the frame (or until MFLR bytes have been received) and reports the frame status and the frame length in the last Rx BD.

MAXD2. This parameter gives the user the ability to stop system bus writes from occurring after a frame has exceeded a certain size. The value of this register is valid in promiscuous mode when no address match was detected. The Ethernet controller checks the length of an incoming Ethernet frame against the user-defined value given in this 16-bit register. Typically, this register is set to 1518 decimal. If this limit is exceeded, the remainder of the incoming frame is discarded. The Ethernet controller waits to the end of the frame (or until MFLR bytes have been received) and reports the frame status and the frame length in the last Rx BD.

In a monitor station, MAXD2 can be programmed to a value much less than MAXD1 to receive entire frames addressed to this station, but receive only the headers of all other frames.

GADDR1–4. These four registers are used in the hash table function of the group addressing mode. The user may write zeros to these values after reset and before the Ethernet channel is enabled to disable all group hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table.

PADDR1. The user writes the 48-bit individual address of this station into this location. PADDR1_L is the lowest order word, and PADDR1_H is the highest order word.

P_Per**.** This parameter allows the Ethernet controller to be less aggressive in its behavior following a collision. Normally, this parameter should be set to $0000. To decrease the aggressiveness of the Ethernet controller, P_Per can be set to a value from 1 to 9, with 9 being the least aggressive. The P_Per value is added to the retry count in the backoff algorithm to reduce the probability of transmission on the next time slot.

**NOTE**

The use of P_Per is fully allowed within Ethernet/802.3 specifications. In a heavily congested Ethernet LAN, a less aggressive backoff algorithm used by multiple stations on the LAN increases the overall LAN throughput by reducing the probability of collisions.

The SBT bit in the PSMR offers another way to reduce the aggressiveness of the Ethernet controller.

Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled or when the user is sure that no receive buffer is currently in use.

**7.11.4.5 TRANSMITTER BUFFER DESCRIPTOR POINTER (TBPTR).** The TBPTR for each SMC channel points to the next BD that the transmitter will transfer data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of the BD table is reached, the CP initializes this pointer to the value programmed in the TBASE entry. Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled or when the user is sure that no transmit buffer is currently in use (e.g., after a STOP TRANSMIT command is issued, or after a GRACEFUL STOP TRANSMIT command is issued, and the frame completes its transmission.)

**7.11.4.6 OTHER GENERAL PARAMETERS.** Additional parameters are listed in Table 7-12. These parameters do not need to be accessed by the user in normal operation, and are listed only because they may provide helpful information for experienced users and for debugging.

The Rx and Tx internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.

The Tx internal byte count is a down-count value that is initialized with the Tx BD data length and decremented with every byte read by the SDMA channels. The Rx internal byte count is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.

**NOTE**

To extract data from a partially full receive buffer, the CLOSE Rx BD command may be used.

The Rx internal state, Tx internal state, Rx temp, Tx temp, and reserved areas are for RISC use only.

## 7.11.5 Disabling the SMCs on the Fly

If an SMC is not needed for a period of time, it may be disabled and re-enabled later. In this case, a sequence of operations is followed.

This sequence ensures that any buffers in use will be properly closed and that new data will be transferred to/from a new buffer. Such a sequence is required if the parameters that must be changed are not allowed to be changed dynamically. If the register or bit description states that dynamic (on-the-fly) changes are allowed, the following sequences are not required, and the register or bit may be changed immediately. In all other cases, the sequence should be used. For instance, the baudrate generators allow on-the-fly changes.

Bits 7, 5, 3—Reserved.

BRKe—Break End

The end of break sequence was detected. This indication will be no sonner than after one idle bit is received following a break sequence.

BRK—Break Character Received

A break character was received. If a very long break sequence occurs, this interrupt will occur only once after the first all-zeros character is received.

BSY—Busy Condition

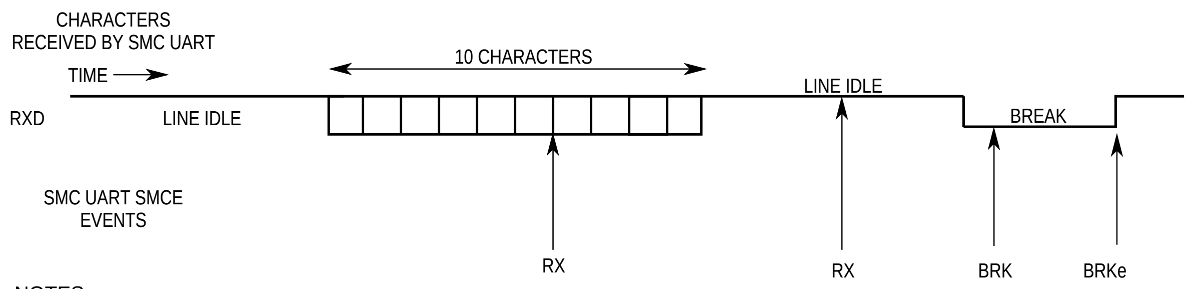
A character was received and discarded due to lack of buffers. This bit is be set no sooner than the middle of the last stop bit of the first receive character for which there is no available buffer. Reception continues when an empty buffer is provided.

TX—Tx Buffer

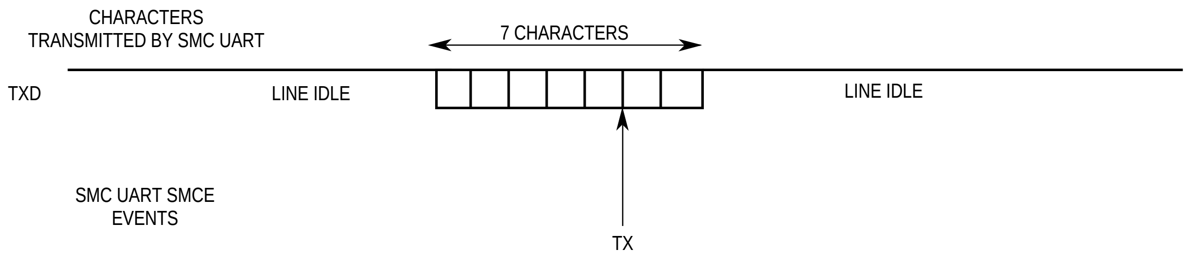
A buffer has been transmitted over the UART channel. This bit is set once the transmit data of the last character in the buffer was written to the transmit FIFO. The user must wait two character times to be sure that the data was completely sent over the transmit pin.

RX—Rx Buffer

A buffer has been received and its associated Rx BD is now closed. This bit is set no sooner than the middle of the last stop bit of the last character that was written to the receive buffer.



NOTES:
1. The first RX event assumes receive buffers are six bytes each.
2. The second RX event position is programmable based on the max_IDL value.
3. The BRK event occurs after the first break character is received.



NOTE: The TX event assumes all seven characters were put into a single buffer, and the TX event occurred when the seventh character was written to the SMC transmit FIFO.

**Figure 7-77. SMC UART Interrupts Example**

Table 7-22. Prioritization of CPM Interrupt Sources

| Number | Priority Level | Interrupt Source Description | Multiple Events |
|---|---|---|---|
| 1F | Highest | Parallel I/O–PC0 | No |
| 1E | | SCCa (Grouped and Spread) | Yes |
| 1D | | SCCb (Grouped) | Yes |
| 1C | | SCCc (Grouped) | Yes |
| 1B | | SCCd (Grouped) | Yes |
| 1A | | Parallel I/O–PC1 | No |
| 19 | | Timer 1 | Yes |
| 18 | | Parallel I/O–PC2 | No |
| 17 | | Parallel I/O–PC3 | No |
| 16 | | SDMA Channel Bus Error | Yes |
| 15 | | IDMA1 | Yes |
| 14 | | IDMA2 | Yes |
| 13 | | SCCb (Spread) | Yes |
| 12 | | Timer 2 | Yes |
| 11 | | RISC Timer Table | Yes |
| 10 | | Reserved | Yes |
| F | | Parallel I/O–PC4 | No |
| E | | Parallel I/O–PC5 | No |
| D | | SCCc (Spread) | Yes |
| C | | Timer 3 | Yes |
| B | | Parallel I/O–PC6 | No |
| A | | Parallel I/O–PC7 | No |
| 9 | | Parallel I/O–PC8 | No |
| 8 | | SCCd (Spread) | Yes |
| 7 | | Timer 4 | Yes |
| 6 | | Parallel I/O–PC9 | No |
| 5 | | SPI | Yes |
| 4 | | SMC1 | Yes |
| 3 | | SMC2/PIP | Yes |
| 2 | | Parallel I/O–PC10 | No |
| 1 | | Parallel I/O–PC11 | No |
| 0 | Lowest | Reserved | — |

**7.15.2.3 NESTED INTERRUPTS.** The CPIC supports a fully nested interrupt environment that allows a higher priority interrupt (from another CPM source) to suspend a lower priority interrupt's service routine. This nesting is achieved by the CPM interrupt in-service register (CISR).

EEPROM may be accessed in succession. The $\overline{\text{CS4}}$ pin should be programmed to respond to an 8-Kbyte area in this design.

Only one byte should be written at a time. After a write is made, software is responsible for waiting the appropriate time (e.g., 10 ms) or for performing data polling to see if the newly written data byte is correct.

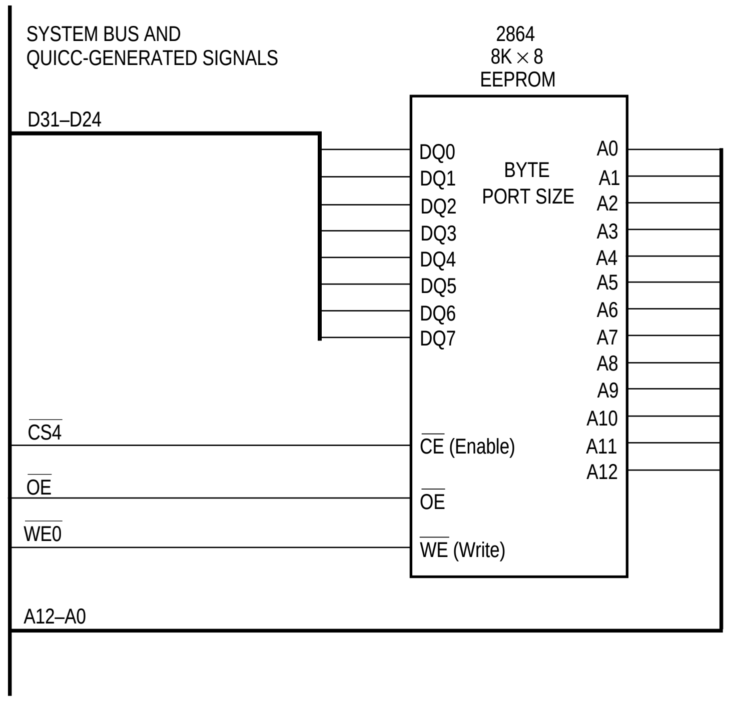


**Figure 9-5. Glueless Interface to EEPROM**

**9.1.2.6 DRAM SIMM.** Figure 9-6 shows the glueless interface to an MCM36100S DRAM single in-line memory module (SIMM). The $\overline{\text{RAS1}}$ line should be programmed to respond to a 4-Mbyte address space.

This particular SIMM also includes parity support, which is supported with the PRTY3–PRTY0 signals.

This design also uses the $\overline{\text{RAS1}}$ double-drive capability, whereby the $\overline{\text{RAS1DD}}$ signal is output by the QUICC to increase the effective drive capability of the $\overline{\text{RAS1}}$ signal.

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

**9.4.1.13 DOUBLE BUS FAULT**. In MC68040 companion mode, the QUICC double bus fault monitor is not operational.

**9.4.1.14 JTAG AND THREE-STATE**. Both the MC68EC040 and the QUICC provide JTAG test access ports, commonly known as JTAG. This interface uses five pins: TMS, TDI, TDO, TCK, and $\overline{\text{TRST}}$. TMS and TDI are left unconnected because they have internal pullups. The JTAG ports of both parts are disabled in this application; however, the capability could be easily added.

When the QUICC is in master mode, it provides a $\overline{\text{TRIS}}$ pin that allows all outputs on the device to be three-stated. In slave mode, this feature is not available since the QUICC is a peripheral of the system. Thus, the transfer start ($\overline{\text{TS}}$) pin is available instead of the $\overline{\text{TRIS}}$ pin and does not conflict with it.

**9.4.1.15 QUICC SERIAL PORTS**. The functions on QUICC parallel I/O ports A, B, and C may be used as desired in this application and have no bearing on the MC68EC040 interface. However, any unused parallel I/O pins should be configured as outputs so they are not left floating.

## 9.4.2 Memory Interfaces

In this application, a number of memory arrays have been developed for EPROM, burst EPROM, flash EPROM, EEPROM, SRAM, burst SRAM, and DRAM. Each memory interface can be attached to the system bus as desired.

One issue not discussed is the decision of whether external buffers are needed on the system bus. This issue depends on the number of memory arrays used in the design and possibly the layout (i.e., capacitance) of the system bus.

Another issue left to the user is the number of wait states used with each memory system. This depends on the memory speed, whether external buffers are used, and the loading on the system bus pins. (The QUICC provides capacitance de-rating figures to calculate the effect of more or less capacitance on the AC Timing Specifications.)

**9.4.2.1 QUICC MEMORY INTERFACE PINS.** In this design, a number of QUICC pins are available to the memory arrays (see Figure 9-8). These pins are active, regardless of whether the bus cycle was originated by the MC68EC040 or by one of the QUICC DMA cycles. The QUICC detects the MC68EC040 bus cycle by the $\overline{\text{TS}}$ pin. If the QUICC generates the bus cycle, the QUICC asserts the $\overline{\text{AS}}$ pin.

Eight $\overline{\text{CSx}}$ or $\overline{\text{RASx}}$ pins are available in the system. In this design, $\overline{\text{CS0}}$ is used for any of the EPROM arrays since it is the global (boot) chip select. $\overline{\text{RAS1}}$ is used for the DRAM arrays because of its double-drive capability. $\overline{\text{CS2}}$/$\overline{\text{RAS2}}$ is not used in the design and is available for other purposes, such as a second DRAM bank. $\overline{\text{CS3}}$ is for SRAM arrays; $\overline{\text{CS4}}$ is for EEPROM. $\overline{\text{CS5}}$, $\overline{\text{CS6}}$, and $\overline{\text{CS7}}$ are unused.

In this design, it is assumed that the full 32-bit capability of the MC68EC040 is used; thus, all memory arrays are 32 bits wide. (The only exception to this is the EEPROM, which is han-

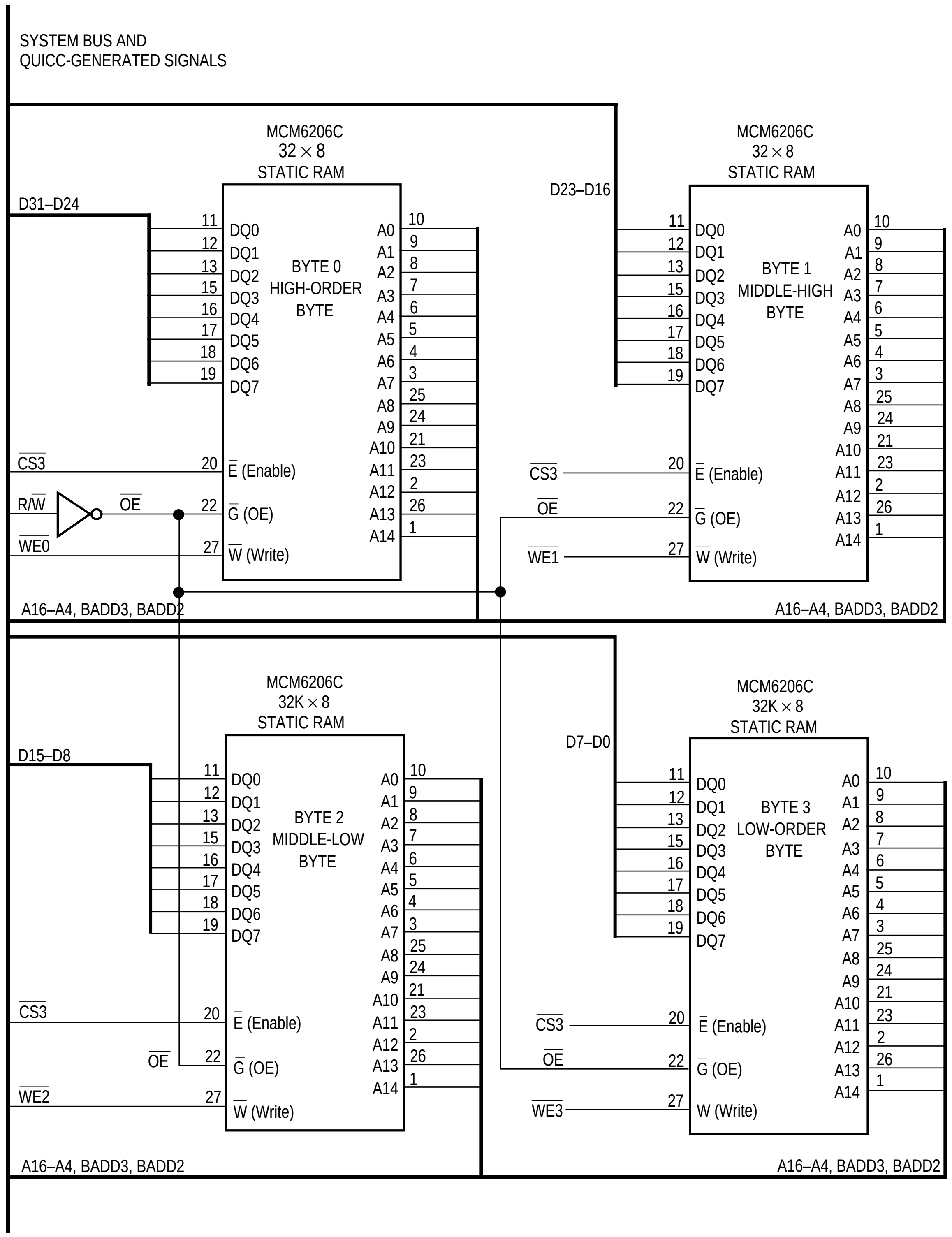


Figure 9-12. 128-Kbyte Static RAM Bank—32 Bits Wide

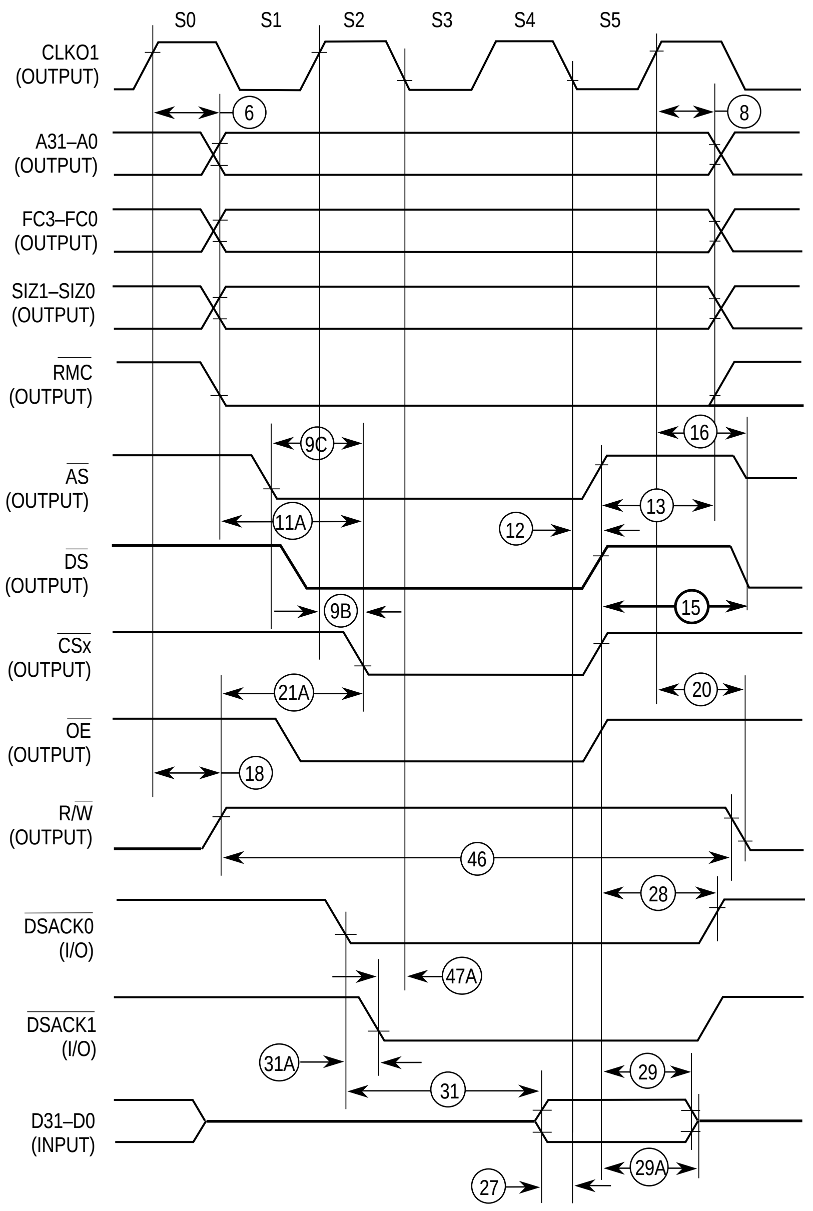


Figure 10-6. SRAM: Read Cycle (TRLX = 1)

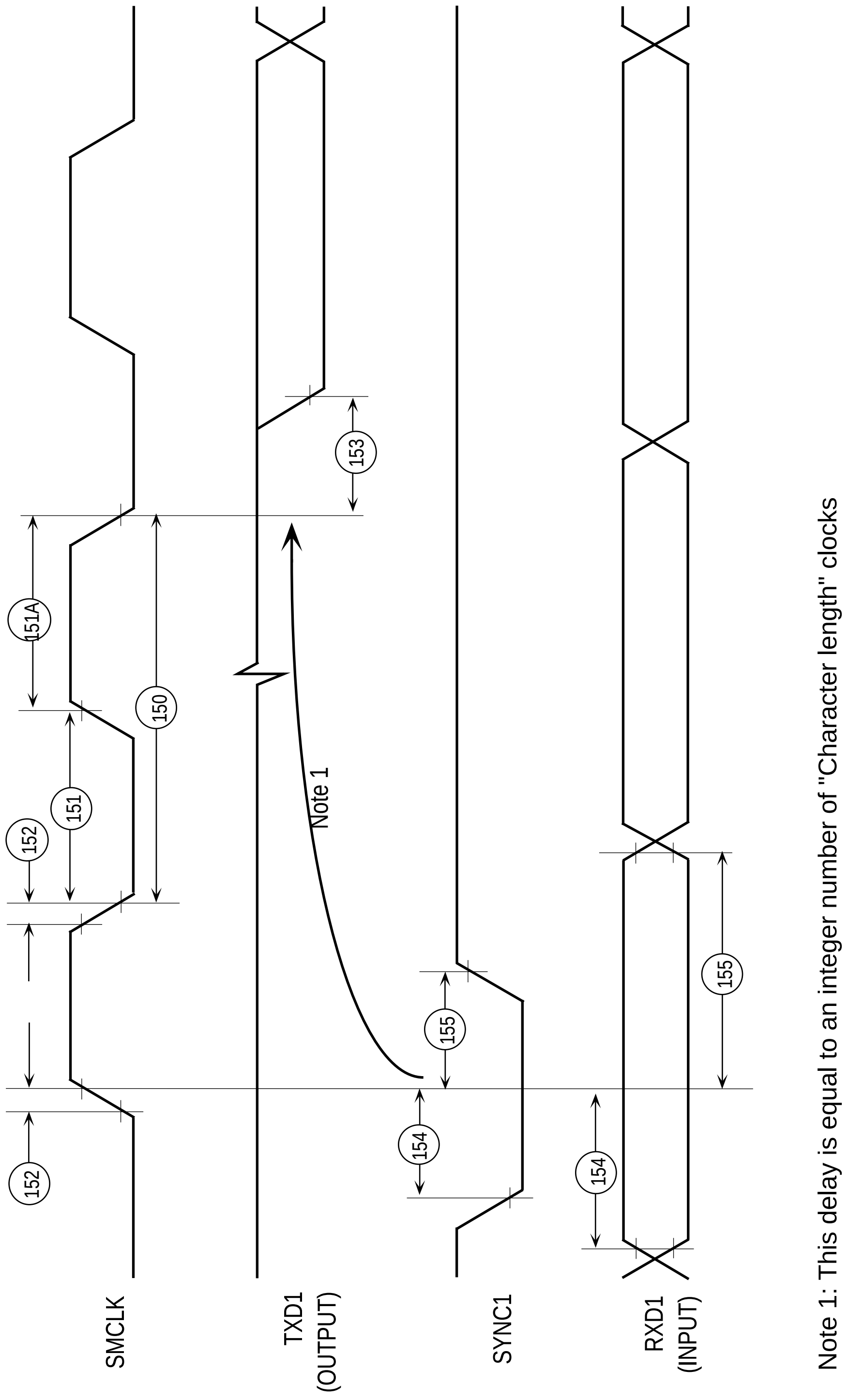

SMCLK
TXD1 (OUTPUT)
SYNC1
RXD1 (INPUT)
150
151
151A
152
152
153
154
154
155
155
Note 1
Note 1: This delay is equal to an integer number of "Character length" clocks

Figure 10-74. SMC Transparent

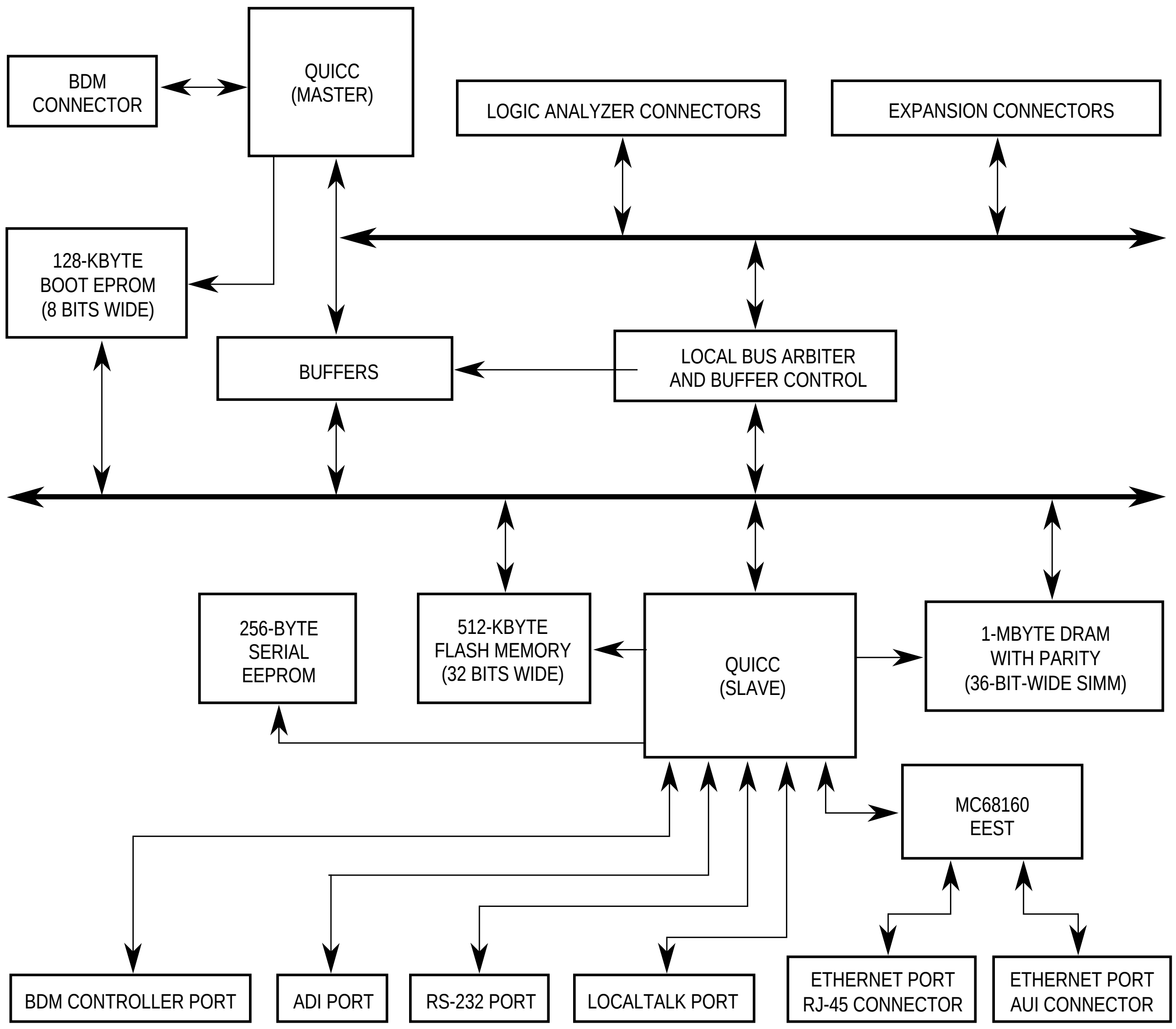

BDM CONNECTOR
QUICC (MASTER)
LOGIC ANALYZER CONNECTORS
EXPANSION CONNECTORS
128-KBYTE BOOT EPROM (8 BITS WIDE)
BUFFERS
LOCAL BUS ARBITER AND BUFFER CONTROL
256-BYTE SERIAL EEPROM
512-KBYTE FLASH MEMORY (32 BITS WIDE)
QUICC (SLAVE)
1-MBYTE DRAM WITH PARITY (36-BIT-WIDE SIMM)
MC68160 EEST
BDM CONTROLLER PORT
ADI PORT
RS-232 PORT
LOCALTALK PORT
ETHERNET PORT RJ-45 CONNECTOR
ETHERNET PORT AUI CONNECTOR

**Figure B-3. QUADS Block Diagram**

To connect the QUADS board to a host computer, a parallel interface cable and host interface (ADI) board are supplied. Hosts include the IBM-PC and SUN-4 systems. Available software executing on the host allows uploading and downloading of files and data between the host and the QUADS board and allows control of the user interface module through a window-based interface program resident on the IBM-PC. Source-level debugging support through this interface and the serial interface is provided by third-party vendors.

To serve as a convenient platform for software development, the QUADS board is supplied with a simple kernel and a CPU32BUG monitor/debugger. The real-time kernel (EDX) offers basic operating system services such as multitasking and memory management. The CPU32BUG monitor/debugger provides operations of memory dump and set (with optional assembly/disassembly of CPU32+ instructions), single instruction execution, breakpoints, and downloads over the serial and parallel interfaces.

Additionally, the QUADS board contains the software modules (protocol, driver, and user interface modules) in EPROM or in downloadable S-record format. An ADI board is required if the software (360sw) provided with the protocol or driver modules is to be used on the