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	-40°C ~ 85°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/mc68360czq25lr2

1.2.1 CPU32+ Core

The CPU32+ core is a CPU32 that has been modified to connect directly to the 32-bit IMB and apply the larger bus width. Although the original CPU32 core had a 32-bit internal data path and 32-bit arithmetic hardware, its interface to the IMB was 16 bits. The CPU32+ core can operate on 32-bit external operands with one bus cycle. This allows the CPU32+ core to fetch a long-word instruction in one bus cycle and to fetch two word-length instructions in one bus cycle, filling the internal instruction queue more quickly. The CPU32+ core can also read and write 32-bits of data in one bus cycle.

Although the CPU32+ instruction timings are improved, its instruction set is identical to that of the CPU32. It will also execute the entire M68000 instruction set. It contains the same background debug mode (BDM) features as the CPU32. No new compilers, assemblers, or other software support tools need be implemented for the CPU32+; standard CPU32 tools can be used.

The CPU32+ delivers approximately 4.5 MIPS at 25 MHz, based on the standard (accepted) assumption that a 10-MHz M68000 delivers 1 VAX MIPS. If an application requires more performance, the CPU32+ can be disabled, allowing the rest of the QUICC to operate as an intelligent peripheral to a faster processor. The QUICC provides a special mode called MC68040 companion mode to allow it to conveniently interface to members of the M68040 family. This two-chip solution provides a 22-MIPS performance at 25 MHz.

The CPU32+ also offers automatic byte alignment features that are not offered on the CPU32. These features allow 16 or 32-bit data to be read or written at an odd address. The CPU32+ automatically performs the number of bus cycles required.

1.2.2 System Integration Module (SIM60)

The SIM60 integrates general-purpose features that would be useful in almost any 32-bit processor system. The term “SIM60” is derived from the QUICC part number, MC68360. The SIM60 is an enhanced version of the SIM40 that exists on the MC68340 and MC68330 devices.

First, new features, such as a DRAM controller and breakpoint logic, have been added. Second, the SIM40 was modified to support a 32-bit IMB as well as a 32-bit external system bus. Third, new configurations, such as slave mode and internal accesses by an external master, are supported.

Although the QUICC is always a 32-bit device internally, it may be configured to operate with a 16-bit data bus. Regardless of the choice of the system bus size, dynamic bus sizing is supported. Bus sizing allows 8-, 16-, and 32-bit peripherals and memory to exist in the 32-bit system bus mode and 8- and 16-bit peripherals and memory to exist in the 16-bit system bus mode.

Spurious Interrupt Monitor

If no interrupt arbitration occurs during an interrupt acknowledge cycle, the bus error signal is asserted internally.

Software Watchdog Timer (SWT)

The SWT asserts a reset or level 7 interrupt (as selected by the system protection control register (SYPCR)) if the software fails to service the SWT for a designated period of time (i.e., because the software is trapped in a loop or lost). There are eight selectable timeout periods. After a system reset, this function is enabled, selects a timeout of approximately 1 second, and asserts a system reset if the timeout is reached. The SWT may be disabled, or its timeout period may be changed in the SYPCR; however, once SYPCR is written, it cannot be written again until a system reset. This mechanism is used to ensure the proper operation of the SWT.

Periodic Interrupt Timer (PIT)

The SIM60 provides a timer to generate periodic interrupts for use with a real-time operating system or the application software. The PIT period can vary from 122 ms to 15.94 s (assuming a 32.768-kHz crystal is used to generate the general system clock). This function can be disabled.

Freeze Support

The SIM60 allows control of whether the SWT and PIT should continue to run during freeze mode.

Low-Power Stop Support

When executing the LPSTOP instruction, the QUICC can provide reduced power consumption with only the SIM60 remaining active.

Low-Power Standby Support

In addition to the low-power stop support, the QUICC can provide low power consumption while other modules or sub-modules are functioning. In this mode, the baud rate generators and serial ports run with a fixed frequency while the rest of the chip (including the SIM60) runs with a divided clock.

Figure 6-2 shows a block diagram of the system configuration and protection logic.

6.3.1 System Configuration

Many aspects of the system configuration are controlled by the MCR.

For debug purposes, accesses to internal peripherals can be shown on the external bus. This function is called show cycles. The SHEN1, SHEN0 bits in the MCR control the show cycles. External bus arbitration can be either enabled or disabled during show cycles.

The SIM60 provides eight bus arbitration levels for determining the priority of bus access (0–7). The SIM60 is fixed at the highest level (level 7). The CPU32+ is fixed at the lowest level (level 0). Only the SIM60, the CPU32+, the two-channel independent direct memory access (IDMA), and the serial direct memory access (SDMA) can be bus masters and arbitrate for

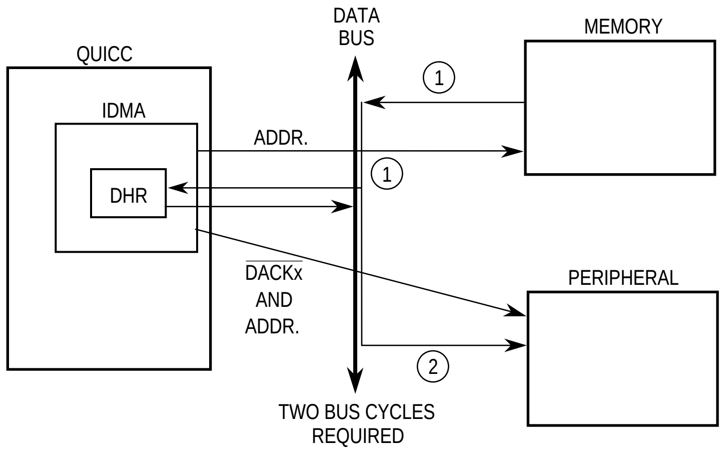


Figure 7-13. Dual Address Transfer Example

Dual Address Destination Write*.* During this type of IDMA cycle, the data in the DHR is written to the device or memory selected by the address in the DAPR, the destination function codes in the FCR, and the size in the CMR. The same options exist for operand size and alignment as in the dual address source read. When the complete operand is written, the DAPR is incremented by 1, 2, or 4, according to the DAPI and DSIZE bits of the CMR, and the BTC is decremented by the number of bytes transferred. If the BTC is equal to zero, the DONEx signal for the IDMA handshake is asserted, and if the transfer is completed with no errors, the DONE bit in the CSR is set. See 7.5.2.4 Timer Reference Registers (TRR1, TRR2, TRR3, TRR4) and 7.6.2.6 Byte Count Register (BCR) for more information.

Dual Address Packing*.* When dual address mode is selected, the IDMA can perform packing. Regardless of the source size, destination size, source starting address, or destination starting address, the IDMA will use the most efficient packing algorithm possible to perform the transfer in the fewest possible number of bus cycles.

NOTE

The packing algorithms are subject to the restriction that the IDMA never performs 3-byte transfers.

Three examples of the packing technique follow.

Example 1. This simple example shows how packing is performed when the source and destination sizes are the same—word. The source address is $00000001, and the destination address is $20000000. The number of bytes to be transferred is 4.

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 = $00000001. Source address.
—DAPR1 = $20000000. Destination address.
—BCR1 = $00000003. Byte transfer count.

In normal UART mode with 16× oversampling, the FSB bits (14–11) in the DSR are decoded as follows:

1111 =	Last Transmitted Stop Bit 16/16 (the default value after reset)
1110 =	Last Transmitted Stop Bit 15/16
1101 =	Last Transmitted Stop Bit 14/16
1100 =	Last Transmitted Stop Bit 13/16
1011 =	Last Transmitted Stop Bit 12/16
1010 =	Last Transmitted Stop Bit 11/16
1001 =	Last Transmitted Stop Bit 10/16
1000 =	Last Transmitted Stop Bit 9/16
0xxx =	Invalid. Do not use.

When the UART is configured for 32× oversampling, the FSB bits (14–11) in the DSR are decoded as follows:

1111 =	Last Transmitted Stop Bit 32/32 (the default value after reset)
1110 =	Last Transmitted Stop Bit 31/32
1101 =	Last Transmitted Stop Bit 30/32
1100 =	Last Transmitted Stop Bit 29/32
1011 =	Last Transmitted Stop Bit 28/32
1010 =	Last Transmitted Stop Bit 27/32
1001 =	Last Transmitted Stop Bit 26/32
1000 =	Last Transmitted Stop Bit 25/32
0111 =	Last Transmitted Stop Bit 24/32
0110 =	Last Transmitted Stop Bit 23/32
0101 =	Last Transmitted Stop Bit 22/32
0100 =	Last Transmitted Stop Bit 21/32
0011 =	Last Transmitted Stop Bit 20/32
0010 =	Last Transmitted Stop Bit 19/32
0001 =	Last Transmitted Stop Bit 18/32
0000 =	Last Transmitted Stop Bit 17/32

When the UART is configured for 8× oversampling, the FSB bits (14–11) in the DSR are decoded as follows:

1111 =	Last Transmitted Stop Bit 8/8 (the default value after reset)
1110 =	Last Transmitted Stop Bit 7/8
1101 =	Last Transmitted Stop Bit 6/8
1100 =	Last Transmitted Stop Bit 5/8
10xx =	Invalid. Do not use.
01xx =	Invalid. Do not use.
00xx =	Invalid. Do not use.

The UART receiver can always receive fractional stop bits. The next character's start bit may begin at any time after the three middle samples of the stop bit have been taken.

7.10.16.14 UART ERROR-HANDLING PROCEDURE. The UART controller reports character reception and transmission error conditions via the channel BDs, the error counters, CM—Continuous Mode

0 = Normal operation.
1 = The E-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

ID—Buffer Closed on Reception of Idles

The buffer was closed due to the reception of the programmable number of consecutive idle sequences (defined in MAX_IDL).

AM—Address Match

This bit has meaning only if the address bit is set and the automatic multidrop mode was selected in the UM bits. Following an address match, this bit defines which address character matched the user-defined address character, enabling the UART to receive data.

0 = The address matched the value in UADDR2.
1 = The address matched the value in UADDR1.

BR—Break Received

A break sequence was received while receiving data into this buffer.

FR—Framing Error

A character with a framing error was received and is located in the last byte of this buffer. A framing error is a character without a stop bit. A new receive buffer will be used for further data reception.

PR—Parity Error

A character with a parity error was received and is located in the last byte of this buffer. A new receive buffer will be used for further data reception.

OV—Overrun

A receiver overrun occurred during message reception.

CD—Carrier Detect lost

The carrier detect signal was negated during message reception.

Data Length

Data length is the number of octets written by the CP into this BD's data buffer. It is written by the CP once as the BD is closed.

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.

(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.

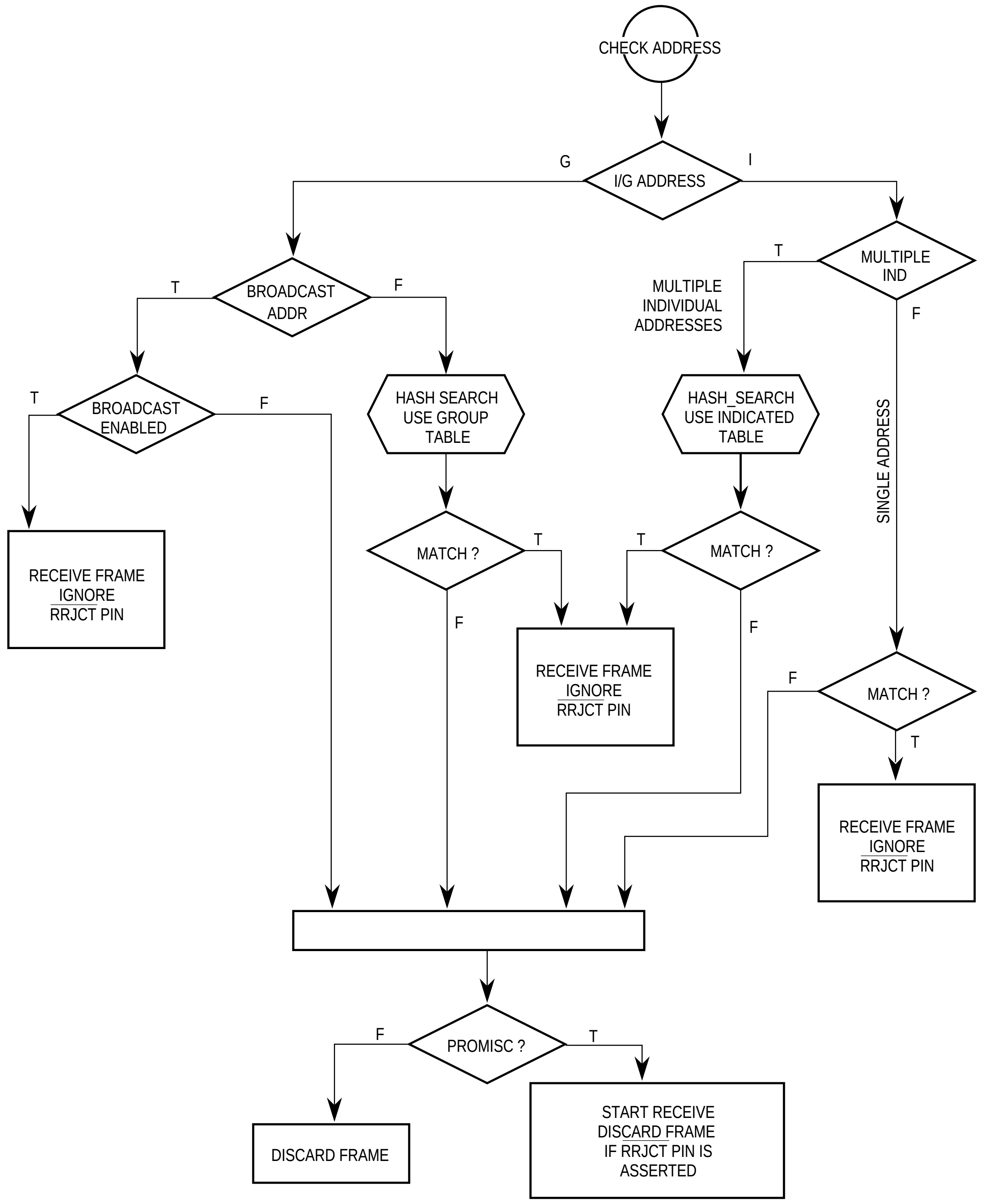

CHECK ADDRESS
I/G ADDRESS
G
I
BROADCAST ADDR
T
F
MULTIPLE IND
T
F
MULTIPLE INDIVIDUAL ADDRESSES
BROADCAST ENABLED
HASH SEARCH USE GROUP TABLE
HASH_SEARCH USE INDICATED TABLE
SINGLE ADDRESS
RECEIVE FRAME IGNORE RRJCT PIN
MATCH ?
RECEIVE FRAME IGNORE RRJCT PIN
MATCH ?
RECEIVE FRAME IGNORE RRJCT PIN
PROMISC ?
DISCARD FRAME
START RECEIVE DISCARD FRAME IF RRJCT PIN IS ASSERTED

Figure 7-70. Ethernet Address Recognition Flowchart

7.10.23.12 HASH TABLE ALGORITHM. The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit address

Data Length

The data length is the number of octets that the CP should transmit from this BD's data buffer. It is never modified by the CP. This value should normally be greater than zero. The data length may be equal to zero with the P-bit set, and only a preamble will be sent.

If the number of data bits in the UART character is greater than 8, then the data length should be even. Example: to transmit three UART characters of 8-bit data, 1 start, and 1 stop, the data length field should be initialized to 3. However, to transmit three UART characters of 9-bit data, 1 start, and 1 stop, the data length field should be initialized to 6, since the three 9-bit data fields occupy three words in memory (the 9 LSBs of each word).

Tx Data Buffer Pointer

The transmit buffer pointer, which always points to the first location of the associated data buffer, may be even or odd (unless the number of actual data bits in the UART character is greater than 8 bits, in which case the transmit buffer pointer must be even.) For instance, the pointer to 8-bit data, 1 start, and 1 stop characters may be even or odd, but the pointer to 9-bit data, 1 start, and 1 stop characters must be even. The buffer may reside in either internal or external memory.

7.11.7.14 SMC UART EVENT REGISTER (SMCE). When the UART protocol is selected, the SMCE register is called the SMC UART event register. It is an 8-bit register used to report events recognized by the SMC UART channel and to generate interrupts. On recognition of an event, the UART will set the corresponding bit in the SMC UART event register.

The SMC UART event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

An example of the timing of various events in the SMC UART event register is shown in Figure 7-77.

7	6	5	4	3	2	1	0
—	BRKe[1]	—	BRK	—	BSY	TX	RX

NOTES:
1: Only available on REV C mask or later. NOT Available on REV A or B.
Rev A mask is C63T
Rev B mask are C69T, and F35G
Current Rev C mask are E63C, E68C and F15W

TIMEOUT Command. This transmitter command may be issued when the QUICC implements the monitor channel protocol. It is issued because the device is not responding or because GCI A-bit errors are detected. When issued, the QUICC sends an abort request on the E-bit.

7.11.14.4 SMC GCI MODE REGISTER (SMCMR). The operating mode of an SMC is defined by the SMCMR. The SMCMR is a 16-bit, memory-mapped, read-write register. The register is cleared at reset. The functions of bits 7–0 are common to each SMC protocol. The functions of bits 15–8 vary according to the protocol selected by the SM bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	CLEN				ME	—	C#	—		SM		DM		TEN	REN

Bit 15, 9, 7, 6—Reserved

These bits should be cleared by the user.

CLEN—Character Length

This value is used to define the total number of bits in the C/I and monitor channels of the SCIT channel 0 or channel 1. CLEN ranges from 0 to 15 and specifies values from 1 to 16 bits. CLEN should be written with 13 for the SCIT channel 0 or GCI (8 data bits, plus A and E bits, plus 4 C/I bits = 14 bits). CLEN should be written with 15 for the SCIT channel 1 (8 data, bits, plus A and E bits, plus 6 C/I bits = 16 bits).

ME—Monitor Enable

0 = The SMC does not support the monitor channel.
1 = The SMC supports the monitor channel with either the transparent or monitor channel protocol as defined in the MP bit.

C#—SCIT Channel Number

0 = SCIT channel 0
1 = SCIT channel 1 (required for Siemens ARCOFI and SGS S/T chips)

SM—SMC Mode

00 = GCI or SCIT support (required for SMC GCI or SCIT operation)
01 = Reserved
10 = UART
11 = Totally transparent operation

DM—Diagnostic Mode

00 =Normal operation
01 =Local loopback mode
10 = Echo mode
11 = Reserved

TEN—SMC Transmit Enable

0 = SMC transmitter disabled
1 = SMC transmitter enabled

RAM values will not normally need to be accessed by user software. They should only be modified when no SPI activity is in progress.

7.12.5.3.1 BD Table Pointer (RBASE, TBASE). The RBASE and TBASE entries define the starting location in the dual-port RAM for the set of BDs for receive and transmit functions of the SPI. This provides a great deal of flexibility in how BDs for an SPI are partitioned. By setting the W-bit in the last BD in each BD list, the user may select how many BDs to allocate for the transmit and receive side of the SPI. The user must initialize these entries before enabling the SPI. Furthermore, the user should not configure BD tables of the SPI to overlap any other serial channel's BDs, or erratic operation will occur.

NOTE

RBASE and TBASE should contain a value that is divisible by 8.

7.12.5.3.2 SPI Function Code Registers (RFCR, TFCR). The FC entry contains the value that the user would like to appear on the function code pins (FC3–FC0), when the associated SDMA channel accesses memory. It also controls the byte-ordering convention to be used in the transfers.

Receive Function Code Register

7	6	5	4	3	2	1	0
—			MOT	FC3–FC0			

Bits 7–5—Reserved

These bits should be set to zero by the user.

MOT—Motorola

This bit should be set by the user to achieve normal operation. MOT *must be set* if the data buffer is located in external memory and has a 16-bit wide memory port size.

0 = DEC and Intel convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering—normal operation. It is also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. The user should write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000. To keep interrupt acknowledge cycles unique in the system, do not write the value 0111 binary to these bits.

Transmit Function Code Register

7	6	5	4	3	2	1	0
—			MOT	FC 3–FC0			

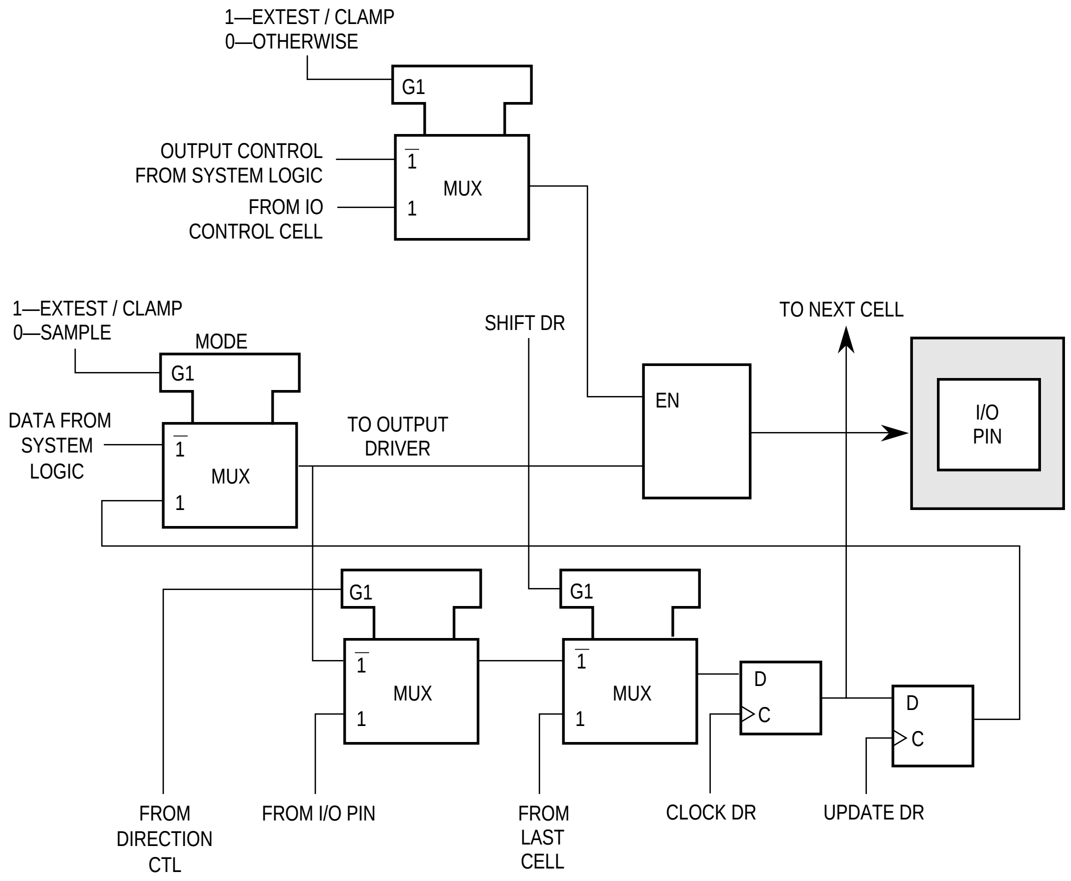
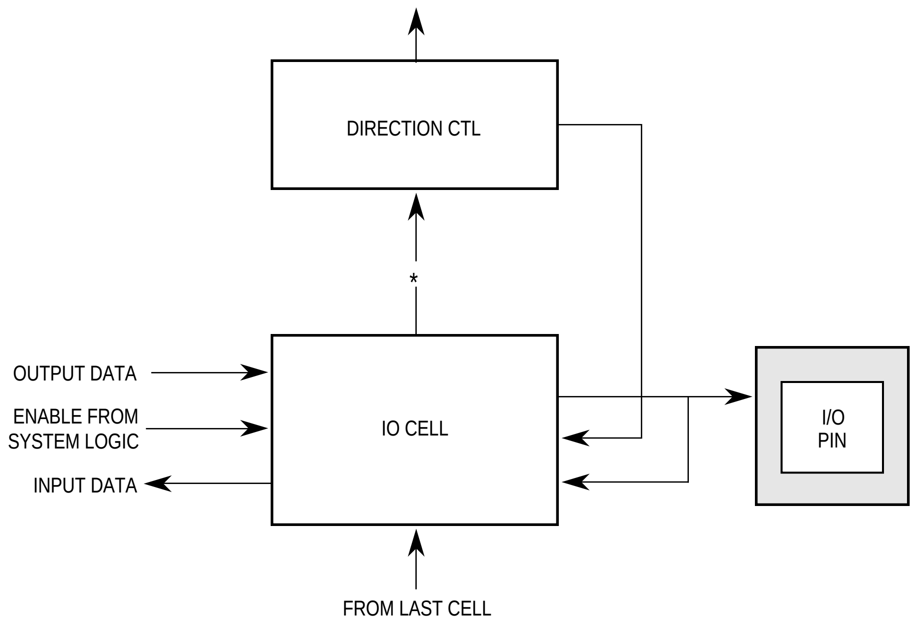


Figure 8-6. Bidirectional Data Cell (IO.Cell)



NOTE: More than one IO.Cell could be serially connected and controlled by a single IO.Ctl.

Figure 8-7. General Arrangement for Bidirectional Pins

BA23–BA13 that was set in A23–A13, and clear BA31–BA24 and BA12–BA11.

The three function code (FC2–FC0) bits become the four function code (FC3–FC0) bits of the QUICC ORx.

The MC68302 ORx register most closely corresponds to the QUICC ORx.

The CFC bit is implemented as the FCM3–FCM0 bits on the QUICC ORx. This gives more flexibility in determining the function codes that cause the chip select to activate. If the MC68302 CFC bit was cleared, then clear FCM3–FCM0 on the QUICC. Also, to match MC68302 behavior, clear the NCS bit in the QUICC GMR.

The MRW bit in the ORx and the RW bit in the BRx of the MC68302 simply become the WP bit on the QUICC BRx. On the QUICC, the choice exists for asserting the chip select for reads and writes (WP = 0) or just reads (WP = 1).

The MC68302 base address mask bits A23–A13 become the AM27–AM11 bits in the QUICC ORx. Note that this allows both larger and smaller block sizes than what the MC68302 provides. To transfer a block range, take bits A23–A13 of the MC68302 and write them to AM23–AM13. Then set AM32–AM24 and clear AM12–AM11.

The three MC68302 DTACK bits become the four TCYC3–TCYC0 bits of the QUICC ORx. Note that the maximum number of wait states is increased from 6 to 15 on the QUICC.

TIMERS

The MC68302 contains two general-purpose timers. The QUICC contains four general-purpose timers, which are the same as the MC68302 timers, but slightly enhanced. The QUICC also contains a timer global configuration register (TGCR), which allows all four timers to be enabled simultaneously and allows the timers to be internally cascaded into 32-bit timers.

The MC68302 TMRx register most closely corresponds to the QUICC TMRx.

The RST bit is now located in the QUICC TGCR.

The ICLK bits are still in the same location of the QUICC TMR. The bit encodings are the same except for 00 combination, which is now implemented by the EN bit in the QUICC TGCR.

The FRR bit is still in the same location of the QUICC TMR.

The ORI bit is still in the same location of the QUICC TMR.

The OM bit is still in the same location of the QUICC TMR, but the meaning of OM = 0 mode can be different. The active-low pulse can be longer than on the MC68302, depending on the frequency of the input clock source.

The CE bits are still in the same location of the QUICC TMR.

The PS bits are still in the same location of the QUICC TMR.

The MC68302 TRRx register is the same as the QUICC TRRx.

The MC68302 TCRx register is the same as the QUICC TCRx.

9.8.1.1 MC68EC030 READS AND WRITES TO QUICC. The basic connection is made through the data and address bus. All 32 data lines are routed between devices, which is required for the connection. In slave mode, the QUICC is not allowed to use its 16-bit data bus mode.(Assertion of $\overline{\text{16BM}}$ pin during reset)

Twenty-eight address lines are routed between devices, giving a 256-Mbyte shared address capability. It is possible to share all 32 address lines between devices, but the QUICC would then lose its write enable lines ($\overline{\text{WE3}}$–$\overline{\text{WE0}}$). Since these lines are very useful in memory interfaces, they are used in this application.

When running in normal slave mode with an MC68EC030 master, the QUICC provides a few signal changes to support the MC68EC030. These signal changes allow the QUICC to monitor and control the system buses in a glueless manner. The changed bus signals are bus request ($\overline{\text{BR}}$), bus grant ($\overline{\text{BG}}$), and bus grant acknowledge ($\overline{\text{BGACK}}$). When operating in normal slave mode, the direction of these signals is reversed. Therefore, $\overline{\text{BR}}$ is an output; $\overline{\text{BG}}$ is an input. In addition, $\overline{\text{BGACK}}$ becomes an I/O signal, rather than just an input.

9.8.1.2 CLOCKING STRATEGY. In this application, a single 25-MHz external oscillator is used to drive the QUICC and the MC68EC030, which allows the synchronous mode of the QUICC memory controller to be used. When considering buffering of outputs and board lay-out, designers need to consider the synchronous timing requirements of the QUICC. Designers considering the possibility of running asynchronously or with faster MC68EC030 clock speeds should reference paragraph 9.8.5 Using a Higher Speed MC68EC030 Master with the QUICC.

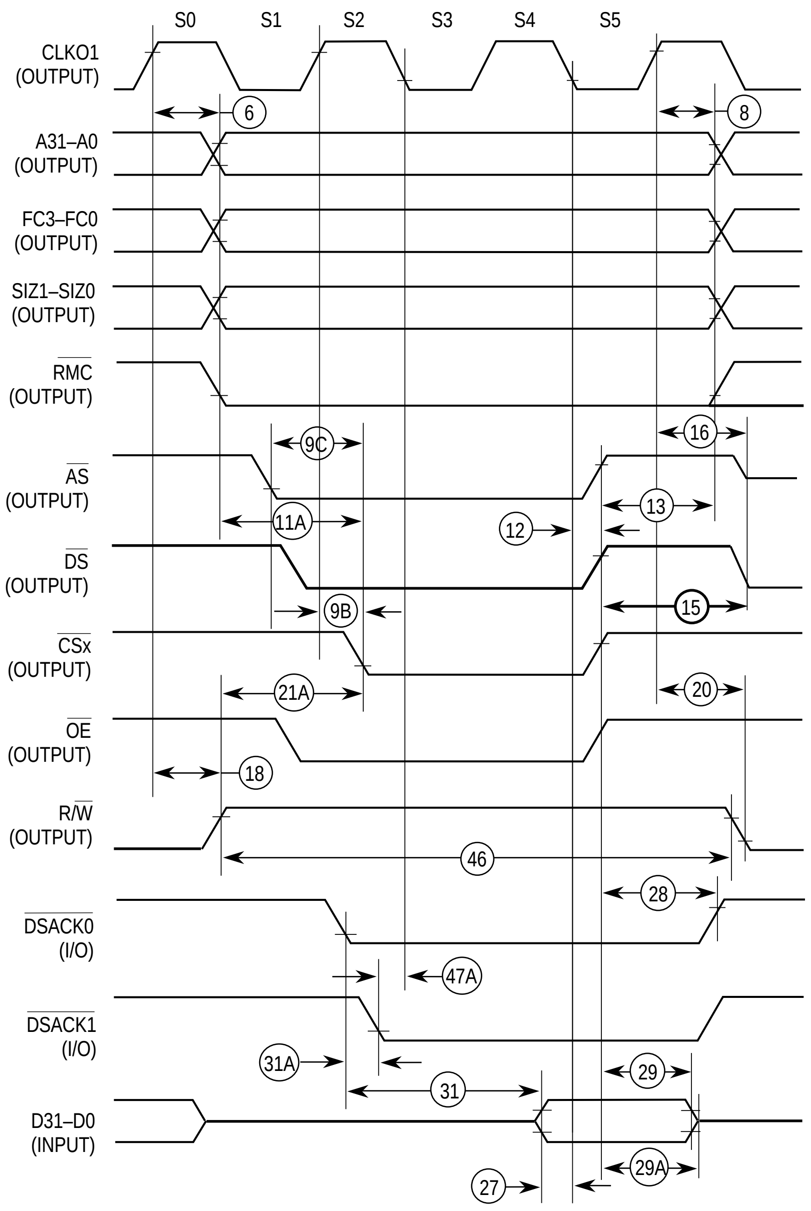

S0
S1
S2
S3
S4
S5
CLKO1 (OUTPUT)
A31–A0 (OUTPUT)
FC3–FC0 (OUTPUT)
SIZ1–SIZ0 (OUTPUT)
$\overline{\text{RMC}}$ (OUTPUT)
$\overline{\text{AS}}$ (OUTPUT)
$\overline{\text{DS}}$ (OUTPUT)
$\overline{\text{CSx}}$ (OUTPUT)
$\overline{\text{OE}}$ (OUTPUT)
R/$\overline{\text{W}}$ (OUTPUT)
$\overline{\text{DSACK0}}$ (I/O)
$\overline{\text{DSACK1}}$ (I/O)
D31–D0 (INPUT)
6
8
9C
16
11A
12
13
9B
15
21A
20
18
46
28
47A
31A
29
31
27
29A

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

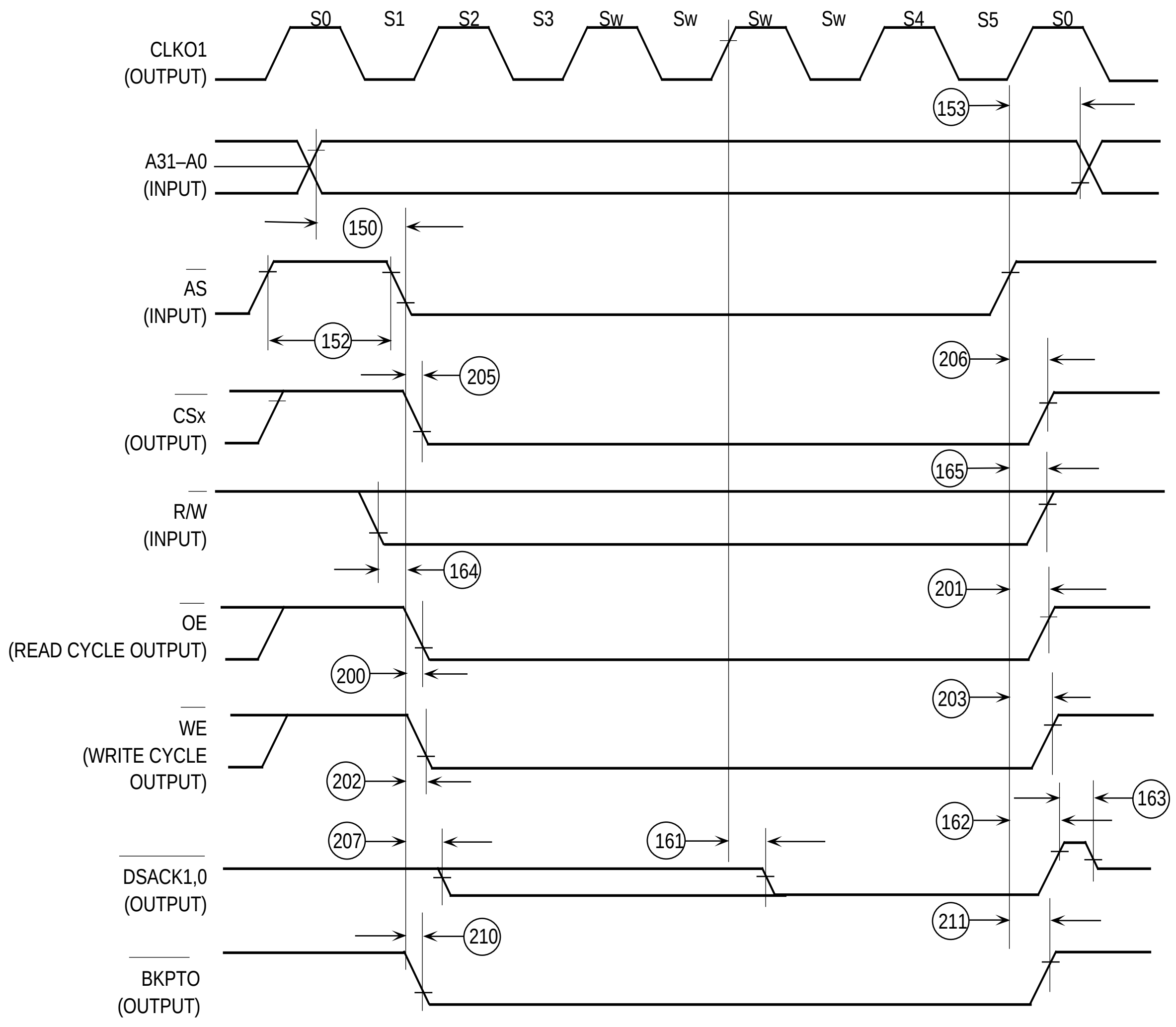


Figure 10-33. External MC68030/MC68360 SRAM Asynchronous Cycle Timing Diagram (BSTN = 0/1; SYNC = 0)

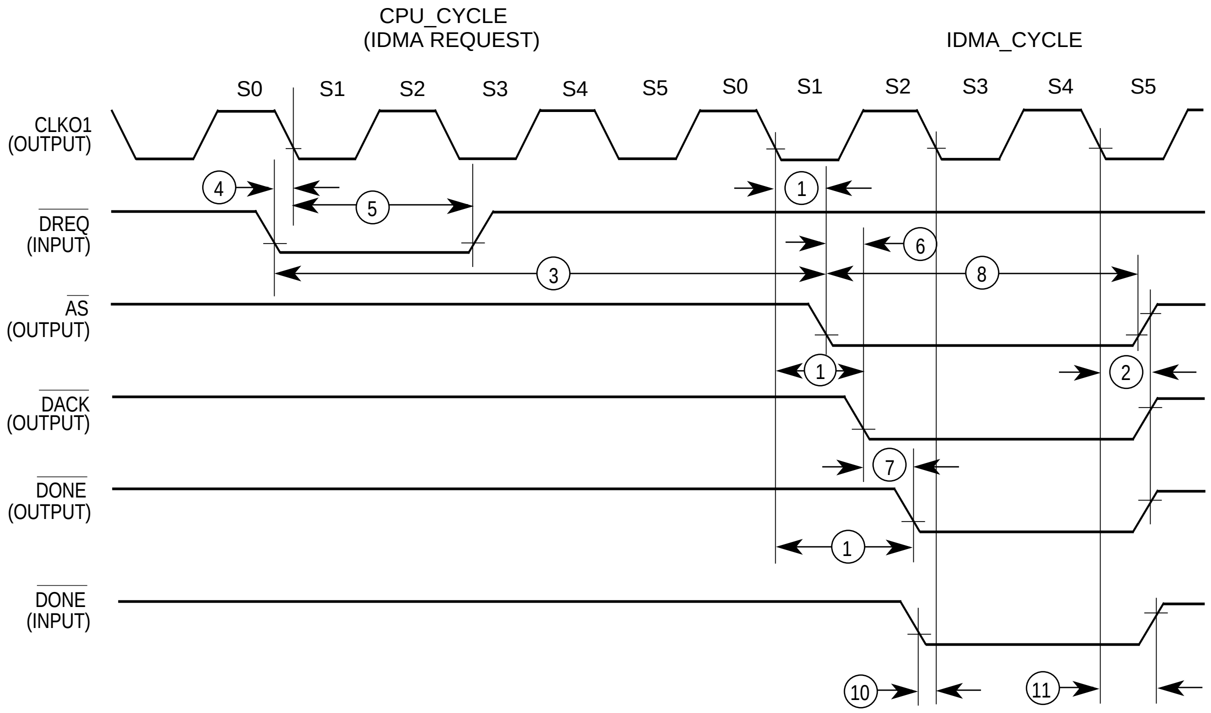


Figure 10-49. IDMA Signal Asynchronous Timing diagram

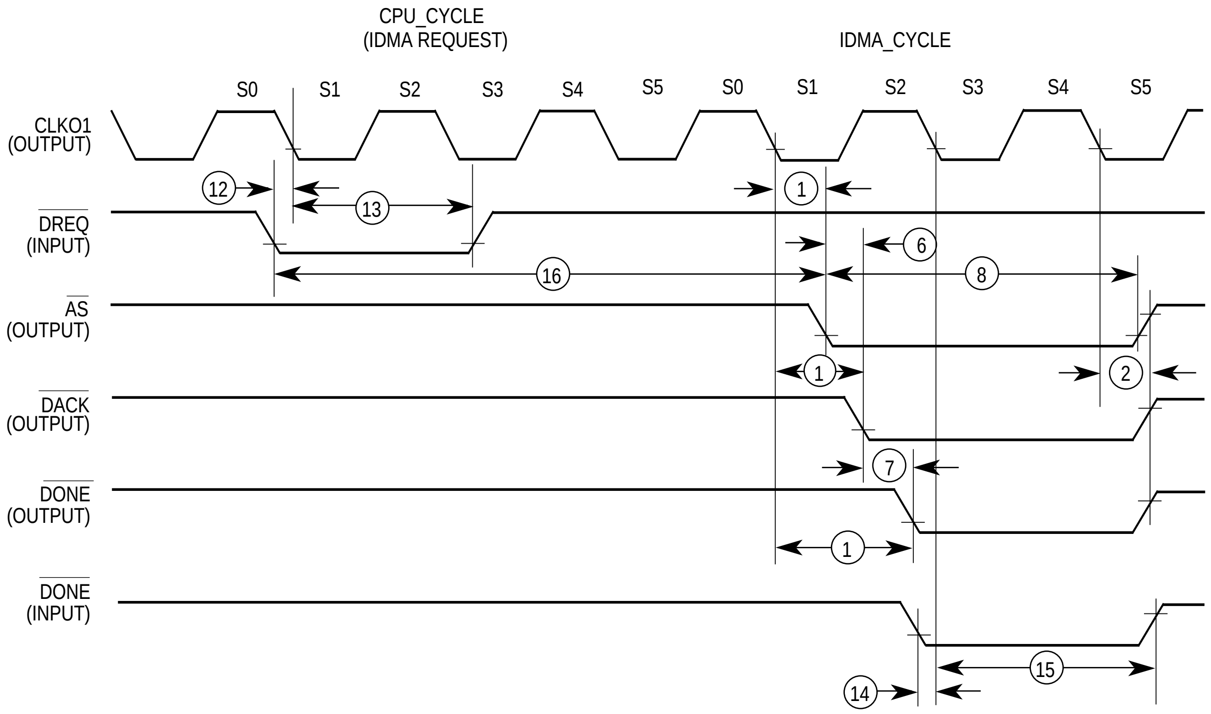


Figure 10-50. IDMA Signal Synchronous Timing diagram

APPENDIX B
DEVELOPMENT TOOLS AND SUPPORT

Several software development packages are offered as a set of independent modules that provide the following features:

- QUICC Chip Evaluation
- By running the modules on the development board described in B.4 M68360QUADS Development System, it is possible to examine and evaluate the QUICC. Symbolic, user-friendly menus allow control of all parts of the QUICC.
- QUICC Simple Drivers
- Written in C, the source code of the QUICC drivers is available on the Motorola Freeware Bulletin Board (512-891-3733, 8 data, no parity, 1 stop) or through Anonymous FTP to freeware.aus.sps.mot.com under /pub/mcu360. These drivers were developed to support the needs of the general user. Although they are based on the regular QUICC drivers, they provide call-oriented interfaces and self-contained QUICC initialization routines and interrupt handling for a given protocol.
- QUICC Chip Drivers
- Written in C, the source code of the QUICC drivers is available on electronic media. These drivers were developed to support the needs of the protocol implementations.
- Protocol Implementations
- Modules implementing common ISO/OSI layer 2 and 3 protocols are available under license. These are in the form of source code written in C.
- Portability
- All of the higher layer software modules may be ported from source to different QUICC implementations and combined with user-developed code. Software interface documentation is available for all provided modules.

B.1 MOTOROLA SOFTWARE MODULES

Chip driver routines written in C illustrate initialization of the QUICC, interrupt handling, and the management of data transmission and reception on all channels.

In addition to the chip drivers, protocol modules are provided. Layer 2 modules include LAPB and LAPD. The layer 3 module is the X.25 packet layer protocol.

Since the modules require some minimal operating system services, the EDX operating system kernel is provided. EDX is the kernel implemented on the QUADS board (see B.4 M68360QUADS Development System). Use of EDX with the protocol modules is not required as long as some other operating system support is provided by the user.

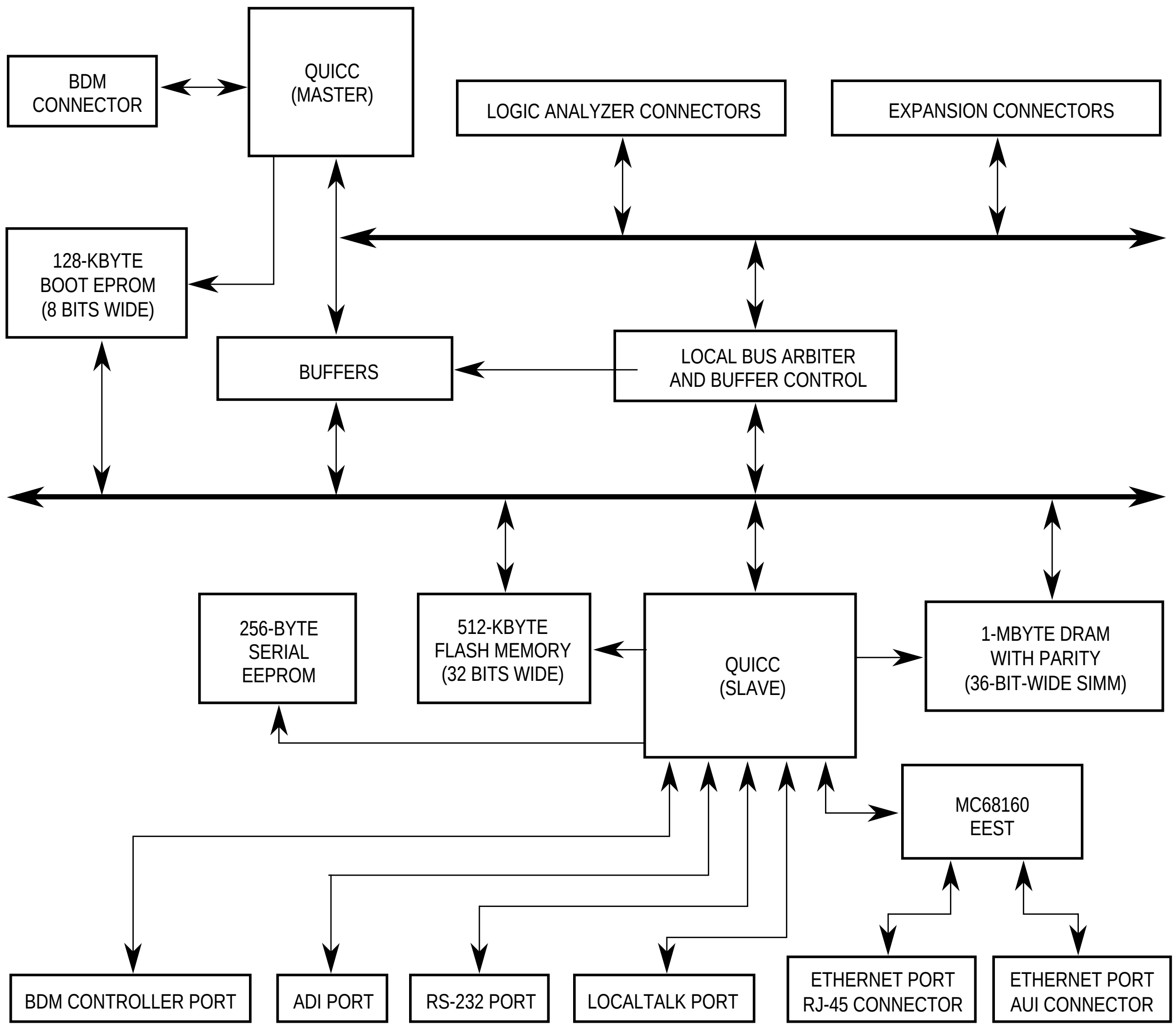

BDM CONNECTOR
QUICC (MASTER)
LOGIC ANALYZER CONNECTORS
EXPANSION CONNECTORS
128-KBYTE BOOT EPROM (8 BITS WIDE)
BUFFERS
LOCAL BUS ARBITER AND BUFFER CONTROL
256-BYTE SERIAL EEPROM
512-KBYTE FLASH MEMORY (32 BITS WIDE)
QUICC (SLAVE)
1-MBYTE DRAM WITH PARITY (36-BIT-WIDE SIMM)
MC68160 EEST
BDM CONTROLLER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