QMCSUPPLEMENT/AD
8/97

QMC Supplement to
MC68360 and MPC860 User's Manuals

Home Page:
www.freescale.com
email:
support@freescale.com
USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescale.com
Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com
Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 2666 8080
support.japan@freescale.com
Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com
For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

CONTENTS

CONTENTS

CONTENTS

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS

TABLES

TABLES

About This Book

This document is a supplement to the *MC68360 Quad Integrated Communications Controller User's Manual* (MC68360UM/AD) and the *MPC860 PowerQUICC User's Manual* (MPC860UM/AD). It replaces the MC68MH360 Reference Manual (MC68MH360RM/AD).

To locate any published errata or updates for this document, refer to the website at http://www.mot.com/netcomm.

Audience

This manual is intended for system software and hardware developers. It is assumed that the reader understands basic concepts of time-division-multiplexed processors and how the MPC860 CPM operates.

Organization

Following is a summary and a brief description of the major sections of this manual:

- Chapter 1, "Overview," gives an introduction to the QMC (QUICC multichannel controller) protocol including some example applications.
- Chapter 2, "QMC Memory Organization," describes the operation specific to the QMC protocol.
- Chapter 3, "QMC Commands," discusses the transmit and receive commands.
- Chapter 4, "QMC Exceptions," describes QMC interrupt handling.
- Chapter 5, "Buffer Descriptors," describes the contents of the receive and transmit buffer descriptors for the QMC protocol and discusses the placement of QMC and non-QMC buffer descriptors in internal and external memory.
- Chapter 6, "QMC Initialization," discusses the essential steps to initialize QMC after a hard reset.
- Chapter 7, "Features Deleted in MC68MH360," lists the features deleted from the MH360.
- Chapter 8, "Performance," provides a performance table for common configurations supported by the 860MH and/or MH360; covers general guidelines and examples for determining the serial bit rate and CPM loading on a given system; and discusses system bus utilization and arbitration.

- Chapter 9, “Multi-Subchannel (MSC) Microcode,” provides the MSC microcode features and operation, and discusses how to program the MSC protocol.
- Appendix A, “68360 Bit Numbering,”shows the bit numbering used for the 68360.
- Appendix B, “Frequently-Asked Questions,” provides a list of common questions and solutions for the MH360 and 860MH.
- Appendix C, “Connecting S/T or U Interfaces to QUICC32,” shows how multiple MC145574 (S/T interface) or MC145572 (U interface) can be connected to a QUICC32. It describes the level-1 connections and explains the data flow through the devices.
- This manual also includes an index.

Additional Reading

This section provides a brief list of additional reading that supplements the information in this manual.

The following materials are available from the Motorola Literature Distribution Centers listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering:

- *MPC860 PowerQUICC User’s Manual* (MPC860UM/AD)
- *MC68360 Quad Integrated Communications Controller User’s Manual*, Rev. 1 *(M68360UM/AD)*
- *M68000 Family Programmer’s Reference Manual*, Rev. 1 (M68000PM/AD)

Conventions

This document uses the following notational conventions:

ACTIVE_HIGH	Names for signals that are active high are shown in uppercase text without an overbar. Active-high signals are referred to as asserted when they are high and negated when they are low.
$\overline{\text{ACTIVE\_LOW}}$	A bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted (active) when they are low and negated when they are high.
0x0F	Hexadecimal numbers
0b0011	Binary numbers
REG[FIELD]	Abbreviations or acronyms for registers are shown in uppercase text. Specific bit fields or ranges are shown in brackets.

Bold font (field name)Entries in boldface must be initialized by the user.

Acronyms and Abbreviations

Table i contains acronyms and abbreviations that are used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
BD	Buffer descriptor
bps	Bits per second
BRI	Basic rate interface
BRG	Baud rate generator
CPM	Communications processor module
CR	Command register
DCL	Data clock signal
FSC	Frame sync signal
GSM	Global system for mobile communications
GOV	Global receiver overrun (global error)
GUN	Global transmitting underrun (global error)
HDLC	High-level data link control
I^2C	Interprocessor-integrated controller channel
MSC	Multi-subchannel microcode
NMSI	Nonmultiplexed serial interface
QMC	QUICC multichannel controller
QUICC	QUad integrated communication controller
RCCR	RISC controller configuration register
RxBD	Receive buffer descriptor
SCC	Serial communication controller
SCCE	SCC event register
SI	Serial interface routing
SS-7	Signaling system 7
TDM	Time-division multiplexing
TSA	Time slot assigner
TSO	Time slot zero
TxBD	Transmit buffer descriptor

Chapter 1
Overview

This chapter gives an overview of the QMC protocol including some example applications.

1.1 The QMC (QUICC Multichannel Controller)

The QMC protocol emulates up to 64 logical channels within one SCC (serial communication controller) using the same time-division-multiplexed (TDM) physical interface. This multichannel protocol is implemented using the CPM ROM space and additional hardware; it is not a downloadable microcode.

The standard QUICC family members (MC68360[1], MPC860[2], etc.) work in TDM applications but can only support one logical channel per SCC. The parts currently supporting the QMC protocol are a superset to the following devices:

- MC68MH360 is a superset of the MC68EN360[3]
- MPC860MH is a superset of the MPC860EN
- MPC860DH is a superset of the MPC860DE

The QMC parts are pin-compatible with their respective family members. With minor adjustments, they can be used in identical applications such as primary rate ISDN support.

1.2 Introduction

Ideal for E1/T1 applications, the QMC protocol can multiplex any 64-channel combination of subgroups to one TDM interface.

Each of the channels can be separately programmed either to perform HDLC formatting/deformatting or to act as a transparent channel.

Both of the SI serial interfaces (for example, TDM_a or TDM_b) can be dedicated to the QMC protocol. The SI transfers the whole frame to an SCC[4]. Using the CPM RISC, the SCC

[1]MC68360 is trademarked as the QUICC.
[2]MPC860 is trademarked as the PowerQUICC.
[3]On the MC68MH360, protocol support for Centronics and BISYNC have been removed to create space for the QMC microcode.
[4]This is the normal operating mode; however, it is possible to split the TDM stream over several SCCs.

works transparently, not participating in any QMC protocol functions. The SCC only performs the parallel-to-serial conversion and adds elasticity through its FIFO memory. The CPM, with its special enhanced microcode and additional dedicated hardware for framing and masking support, does all of the protocol processing for each of the 64 channels. Note that it is executed without intervention from the on-board CPU. Figure 1-1 illustrates the QMC's multichannel capability. Note that each SCC can support up to 64 channels from the TDM; however, there are limitations depending on the device used. This is summarized in Section 1.3, "QMC Features."

Each SCC can work in QMC mode, either alone or together in any combination. The larger FIFO of SCC1 yields the best performance and is therefore recommended for QMC operation. One TDM connection can be routed to one or more SCCs operating in QMC mode, with each SCC operating on different time slots. It is possible to use both TDMs for QMC with combined routing to one SCC or to separate SCCs. When using two TDMs connected to one SCC, restrictions such as using common clocks and sync inputs apply; it is also important to avoid collisions by separating the serial interface (SI) routing.

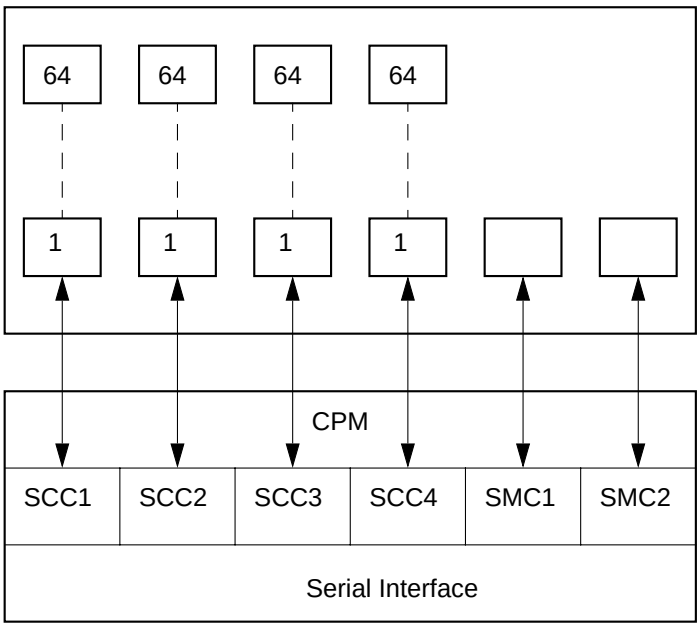


Figure 1-1. QMC Channel Addressing Capability

1.3 QMC Features

- MC68MH360-specific features
  - Up to 32 independent communication channels
  - Arbitrary mapping of any of 0–31 channels to any of 0–31 TDM time slot
  - Can support arbitrary mapping of any of 0–31 channels to any of 0–63 TDM time slots in case of common Rx and Tx mapping
  - Up to three additional HDLC 64-Kbps channels at 25-MHz system clock
  - Simultaneous Ethernet support at 33-MHz system clock
  - Up to 64 DMA channels with linear buffer array
- MPC860MH/DH-specific features
  - Up to 64 independent communication channels
  - Arbitrary mapping of any of 0–63 channels to any of 0–63 TDM time slots
  - Supports arbitrary mapping of any of 0–63 channels to any of 0–127 TDM time slots in case of common Rx and Tx mapping
  - Two simultaneous 32-channel E1 links at 50-MHz system clock
  - Up to 128 DMA channels with linear buffer array
- Common features
  - Independent mapping for receive/transmit
  - Supports either transparent or HDLC protocols for each channel
  - Interrupt circular buffer with programmable size and overflow identification
  - Global loop mode
  - Individual channel loop mode through the SI
  - Programmable frame length (via SI)
- Serial interface
  - Serial-multiplexed (full duplex) input/output 2048-, 1544-, or 1536-Kbps PCM highways
  - Compatible with T1/DS1 24-channel and CEPT E1 32-channel PCM highway, ISDN basic rate, ISDN primary rate and user-defined
  - Subchanneling on each time slot
  - Allows independent transmit and receive routing, frame syncs, and clocking
  - Concatenation of any, not necessarily consecutive, time slots to channels independently for receive/transmit
  - Supports H0, H11, and H12 ISDN channels
  - Allows dynamic allocation of channels

- System interface
  - On-chip bus arbitration for serial DMAs with no performance penalty
  - Efficient bus usage (no bus usage for nonactive channels and active channels that have nothing to transmit)
  - Efficient control of the interrupts to the CPU
  - Supports external buffer descriptors table
  - Uses on-chip enlarged dual-ported RAM for parameter storage

1.4 The Time Slot Assigner and the QMC

The time slot assigner (TSA) in the MH devices is no different from the other versions. This section discusses the new possibilities when using the TSA in combination with the QMC.

The QMC protocol can be executed in nonmultiplexed serial interface (NMSI) mode, but the usual operating mode takes advantage of the programmable time slot assigner,.

A frame synchronization pulse alerts the time slot assigner to start counting clock pulses. The user programs what bits are routed to the different internal serial channels. The TSA is an intelligent multiplexer that restarts its sequence on every frame synchronization pulse.

External strobe signals allow other devices that do not have built-in time slot assigner functions to participate in the TDM interface. This is very useful when interfacing to the MC68302 or other telecommunication devices like codecs.

The time slot assigner is not limited to standard TDM lines. It is a flexible, programmable device that allows the user to route any combination of bits and bytes to any channel. For example, the user can transmit 3 bits from SCC2, skip 12 bytes, and then transmit another 17 bits from SCC1. This routing must be programmed into the TSA memory. The complexity of the routing is limited only by the number of program entries in the TSA.

Ideal for TDM bridging applications, the MC68MH360 and MPC860MH have two independent time slot assigners and physical interfaces. A complete set of independent receive and transmit clock signals, as well as independent synchronization signals, are available for each TDM.

1.5 The Serial Interface (SI)

Functions such as frame synchronization, loopback, echo, and inverted signals are performed in the serial interface and cannot be achieved in NMSI mode. It is recommended to use the serial interface even if only one SCC is used for the TDM bus.

1.5.1 Synchronization

Independent receive and transmit clocks and frame synchronization signals control the data transfer. In NMSI operation, synchronization occurs only once to initiate a transfer using the CD (receive) and CTS (transmit) signals in pulse mode. If any noise corrupts either signal, the QMC will be out of synchronization until the whole protocol is restarted.

In contrast, the more robust SI performs a synchronization on each frame, limiting the damage from noise error on the clock or synchronization lines. Noisy channels can be restarted individually without interrupting other channels. For more details about possible errors in the TDM interface, see Section 1.8, “SI RAM Errors.”

1.5.2 Loopback Mode

The loopback from a transmitter to a receiver is implemented on a per channel basis for every logical channel. A common transmit and receive clock as well as a common frame synchronization pulse must be provided for loopback mode to work. The loopback is done on a fixed time slot, meaning that if one logical channel transmits on time slot 17, the loopback occurs through time slot 17 also, whether it is same logical channel or not that receives the incoming data. The reason for this restriction is that no buffering is performed after a channel is processed by the transmitter, or before it reaches the receiver.

Previously reserved, bit 15 of each entry in the SI-RAM is now the loopback bit controlling the loopback for the corresponding time slot. It is important to have each individual time slot as an entry in the SI-RAM for proper loopback on each individual channel.

1.5.3 Echo Mode

The SI can be programmed to echo incoming data. In this mode, the complete TDM link is retransmitted from the incoming L1RXDx to the L1TXDx pin on a bit-by-bit basis. The receiver section of the selected SCC can operate normally and also receive the incoming bit stream. This is also known as global echo mode on the whole link. Individual time slot echo is not possible with QMC without software intervention.

1.5.4 Inverted Signals

For each SCC, the DPLL can be used to invert the bitstream of the transmitter before the signal reaches the pin. This is not a bit-order inversion, but a logical level inversion. The DPLL can also invert the incoming data before it is forwarded to the receiver section. A logical inversion on a per channel basis is not possible in the QMC without external hardware. To invert a specific channel, the SI can be programmed to send a strobe signal at the channel’s corresponding time slot, assuming the SCC is operating in QMC mode. This strobe can then be connected to an external XOR gate to perform the inversion.

1.6 QMC Serial Routing and Example Applications

The QMC protocol provides multiple logical channels from a single SCC. The SCC channel dedicated to operate the QMC protocol should have all the relevant bits or time slots routed to it. Individual logical channels are selected by a combination of signals routed through the TDM and tables within the QMC protocol. Contrasting a non-QMC example application with QMC implementations highlights benefits of the multichannel protocol.

Figure 1-2 shows an Ethernet-to-BRI bridge using an MC68EN360, a non-QMC device. The configuration shows the Ethernet routed via an NMSI interface to SCC1. The ISDN BRI is routed via the TSA over an IDL2[1] interface to SCC2–SCC4 for the 2 B + D (B1, B2, and D) channels. The first byte of the frame (B1) is routed to SCC2, the second byte (B2) to SCC3, and then the next two bits (the D channel) to SCC4. In this example, SMC2 is used to connect to a PC over RS232. The internal routing is illustrated in Figure 1-3. Note that three SCCs are required to implement the ISDN BRI. This uses all the MC68EN360's serial channels without efficient use of the CPM bandwidth.

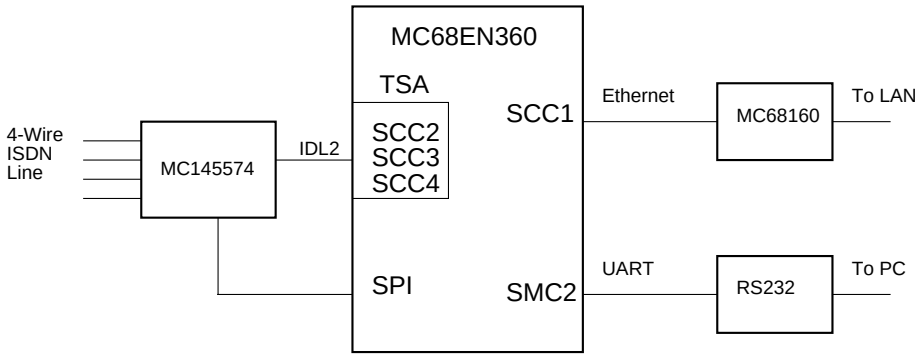


Figure 1-2. Ethernet-to-BRI Bridge Using MC68EN360

[1]The IDL2 interface is a full duplex ISDN interface used to interface to a physical layer device, such as the Motorola ISDN S/T transceiver MC145474.

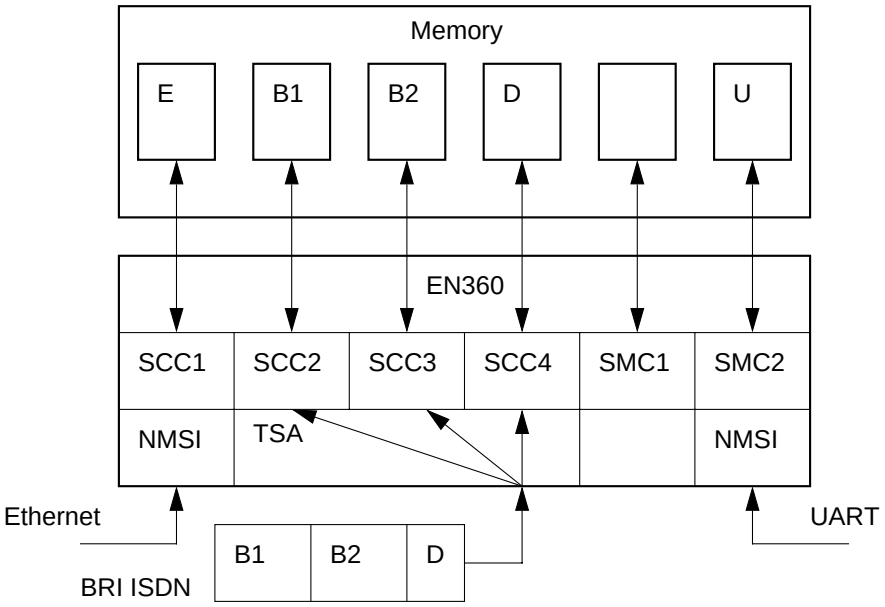


Figure 1-3. Internal Routing for Ethernet-to-BRI Bridge Using MC68EN360

The following example shows how an MC68MH360 can implement the BRI using only one SCC, leaving SCC3 and SCC4 available to run other protocols such as frame relay over HDLC and another Ethernet link, on SCC1, to the LAN. The QMC protocol allows all three channels B1, B2, and D to be routed to SCC2 using the TSA. The first byte (B1) is routed to logical channel 1, the second byte (B2) to logical channel 2, and the third byte to logical channel 3, of which only the first 2 bits represent the D channel as illustrated in Figure 1-4 and Figure 1-5. This routing is defined in the QMC time slot assignment tables. The first advantage of the QMC protocol is that it releases SCCs to run other protocols. The second advantage is highlighted in the next example.

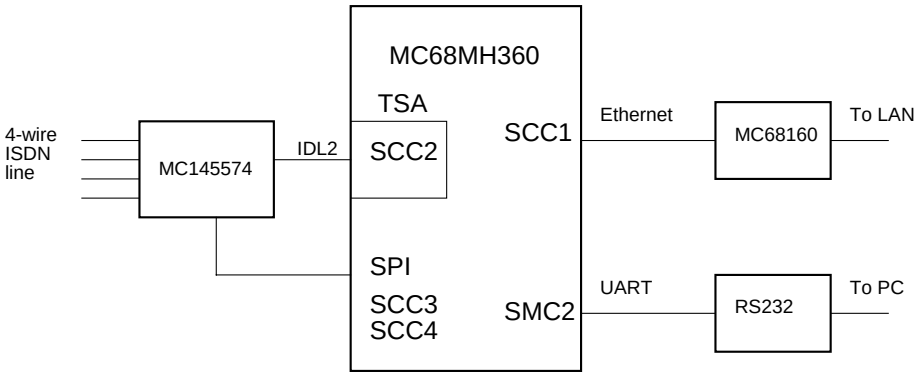


Figure 1-4. Ethernet-to-BRI Bridge Using MC68MH360

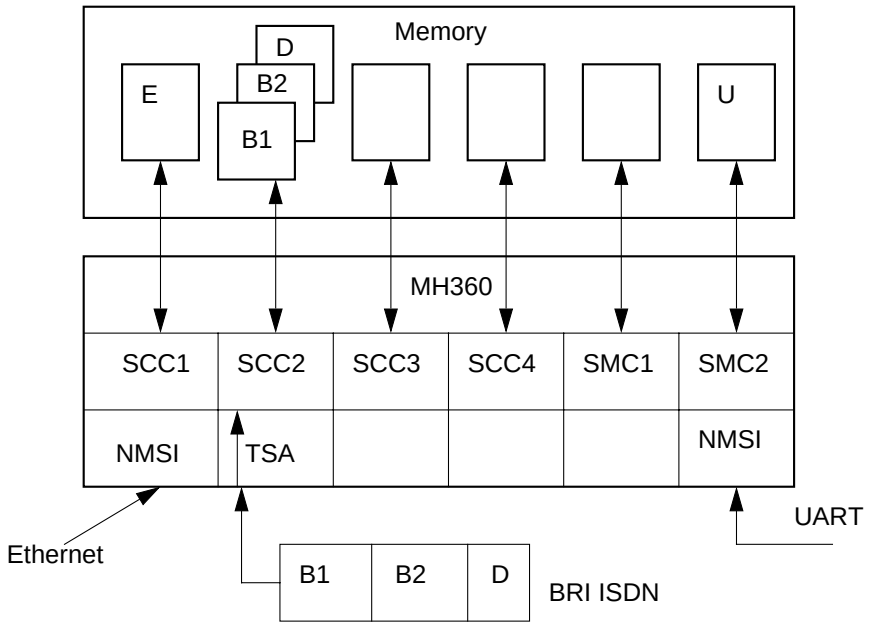


Figure 1-5. Internal Routing for Ethernet-to-BRI Bridge Using MC68MH360

Figure 1-6 and Figure 1-7 show how to build a PRI ISDN-to-Ethernet bridge using an MPC860MH. SCC1 is used for the Ethernet channel. SCC2 is configured for QMC mode in which each of the 30 B channels and the D channel are routed to separate logical channels. The true advantage of the QMC protocol is the ability to route multiple channels to a single SCC.

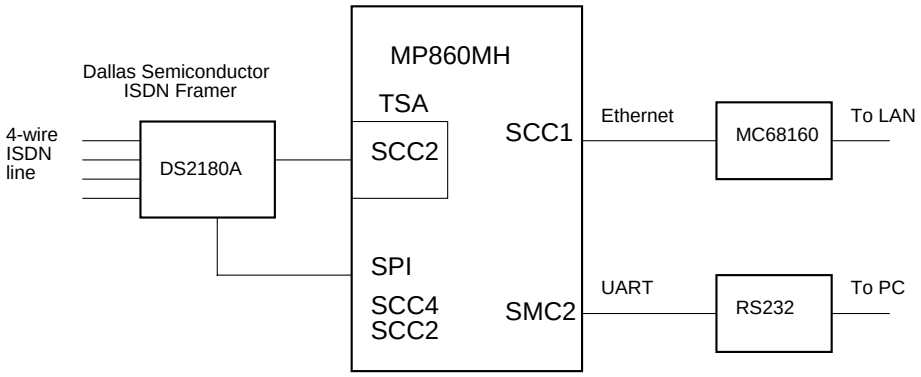


Figure 1-6. Ethernet-to-PRI Bridge Using MPC860MH

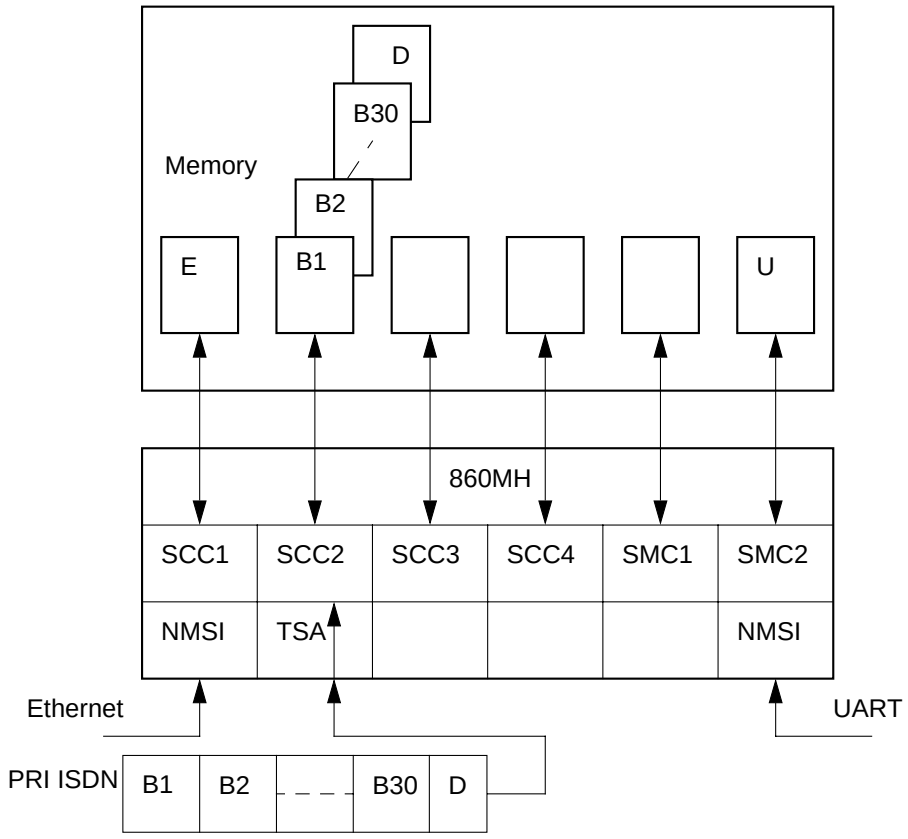


Figure 1-7. Internal Routing for Ethernet-to-PRI Bridge Using MPC860

1.7 SCC Changes on the Fly

Changes can be made on the fly in the QMC routing tables, but changes made in the SI RAM require the link to be disconnected. If the connection is maintained during changes, synchronization and routing errors are likely to happen in the current frame. A workaround uses a shadow RAM routing table. The shadow table can hold alternative routing information to be switched in at the appropriate time slot boundary. The drawback to this method is that the number of entries in the SI RAM is reduced by half. But since the routing tables in the QMC protocol are being changed anyway, the recommended solution is to have all relevant time slots routed to the SCC.

The SI RAM also gives the user the capability to multiplex other channels to and from a TDM if not all time slots are used by the QMC. A third option is to have several external devices multiplexed. Use the open collector mode if several QUICCs or PowerQUICCs are connected together for subchanneling applications.

1.8 SI RAM Errors

The following three types of errors are identified:

- Data bit error
- Clock pulse error
- Synchronization pulse error

Errors in frame-based protocols are easy to detect by the protocol controller. An error in an HDLC channel is detected at the end of a frame when a buffer is closed and all status bits are reported in the buffer descriptor (BD). The error type for bit errors is normally CRC errors. For errors occurring in the SI (noise on clock or synchronization pulses), the error may also be of type frame-length-violation or non-octet-aligned. See Chapter 5, “Buffer Descriptors,” for more information. This section covers the type of errors reported through the buffer descriptors. For transparent channels, the error detection mechanism is left to the user in higher-level software. Most transparent channels, such as voice carriers, are tolerant of errors. Frame-based channels, on the other hand, require error detection since they often rely on critical control messages.

The number of clocks that occur between sync pulses is given in the SI RAM programming. The clock-counting state machine expects a new sync pulse after the end of each frame. The following paragraphs discuss the different error cases and describe the counter state and the frame delay before synchronization is resumed.

A clock pulse error occurs if other than exactly one clock pulse is detected by the SI RAM in a given frame. In this error case, since the SI RAM bases its routing on counting clock pulses, the now corrupted signal routing affects all channels. The SI RAM expects another sync pulse when it reaches the last entry of the frame.

If a clock pulse is missing in a given frame N, the counter will fail to reach its end state before the next sync pulse (N+1) arrives, causing that sync pulse to be ignored. When the counter finally reaches its end state, it waits for the next sync pulse (N+2) before resetting. Correct routing is thus resumed in frame (N+2). In the case of an extra clock pulse, the counter reaches its end state too early and resumes synchronized routing upon detecting the next sync pulse (N+1).

Synchronization pulse errors are similar to clock pulse errors. If the frame pulse comes too late, this is similar to having missed a clock pulse in the last time slot. If the frame pulse is too early, it is similar to having one additional clock pulse.

1.9 E1/T1 Frame Description

The primary rate ISDN connections offer a cost-effective, high-speed interface. The physical connections in North America offer 24 connections over a T1 interface; in Europe an E1 (or CEPT) connection gives 32 connections of 64 Kbps each with a time-division-multiplexed architecture.

Time-division-multiplexing (TDM) allows several communication channels to share the same physical media. The data stream of each channel is divided into a number of subpackages. Each channel is then assigned a subdivision of the TDM line called a time slot. This time slot is repeated over time in a regular pattern. A concatenation of the channels' subpackages comprises a frame. The frequency of frame repetition depends on the particular communication interface. Two examples—the T1 line used in North America and the E1 interface used in Europe illustrate TDM.

For both E1 and T1, the frames must be repeated at a frequency of 8 KHz, or every 125 µs. In many applications the required channel speed is 64 Kbps. For example, almost all voice channels use 8-KHz sampling with 8-bit resolution. Each channel in a T1 or E1 interface occupies 8 bits per time slot. The T1 interface multiplexes 24 channels, requiring 24 time slots per frame. In addition to the channels' bits, one more bit, for frame signaling and synchronization, is added to create a frame totaling 193 bits. The resulting T1 physical interface is thus 1.544 Mbps (8 KHz * 193 bits). The E1 frame consists of multiplexing 32 channels resulting in a speed of 2.048 Mbps (8 KHz * 256 bits). These two frames are illustrated in Figure 1-8.

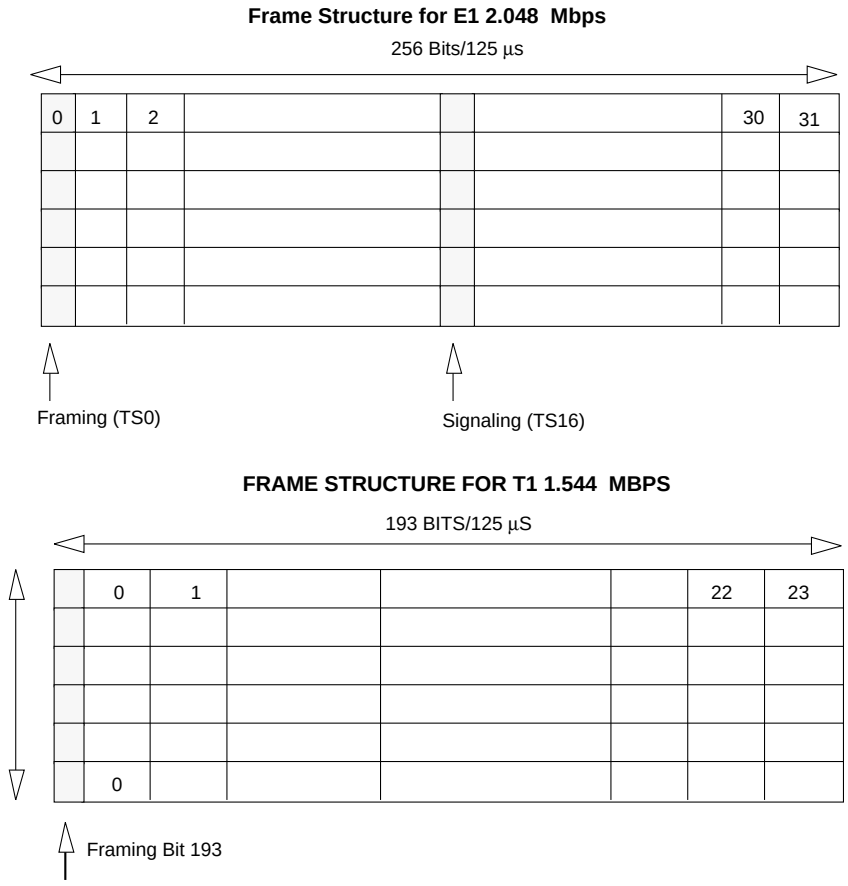


Figure 1-8. Frame Structures for E1/CEPT and T1 TDM Interfaces

For any station to receive and transmit on a TDM line, it is necessary for it to determine the correct time slot boundary. The service provider or the PTT provides a 4-wire interface with a continuous bit stream coming down the line. The T1 and E1 have information embedded in the data stream that delineates frames. The bit pattern in position 193 in T1 over a period of several frames establishes a synchronization pattern. A station may have the capability to search for this pattern and thus find the correct time for frame synchronization. In a similar way, time slots 0 and 16 are reserved not only for synchronization but also for signaling in the E1 interface.

Depending on its capability, a node can either extract this synchronization information or it can be supported by framer and time slot assigner devices.

A framer device will retrieve the 8-KHz frame synchronization pulses and clock signals for both transmit and receive sections. A time slot assigner will use these signals as inputs to generate pulses or envelope signals for individual bit patterns, i.e., strobe signals for devices without time slot assignment capability such as the MC68302, a first-generation communication processor.

For a backplane type of design without a synchronization to a network, the QMC devices are capable of generating all necessary bus signals from their timers and baud-rate generators. This design type is illustrated for a QUICC32 in Figure 1-9.

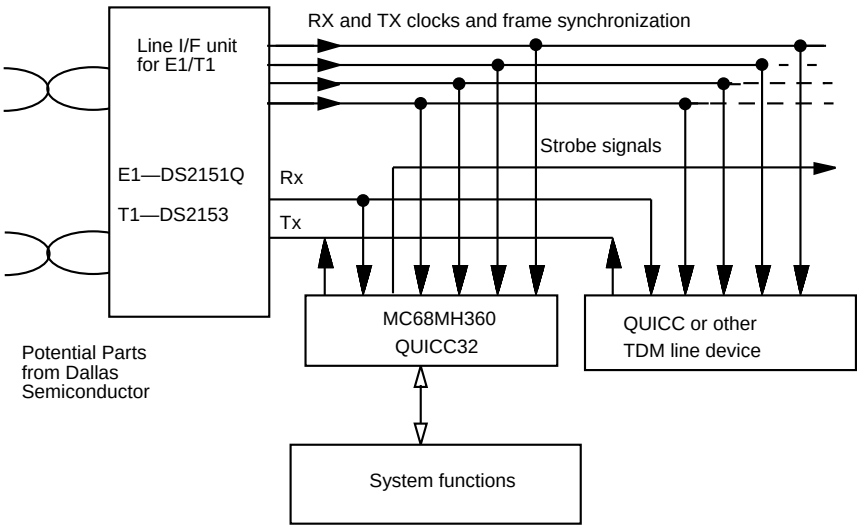


Figure 1-9. MC68MH360 Connection to a TDM Bus

Chapter 2
QMC Memory Organization

This section describes the operation specific to the QMC protocol. When not running the QMC protocol, SCCs operate as described in the MC68360 and MPC860 user's manuals.

Figure 2-1 shows the dual-ported RAM structure for the MC68MH360 and the MPC860MH. The MC68MH360 and the MPC860MH have similar functionality but are organized in a different manner.

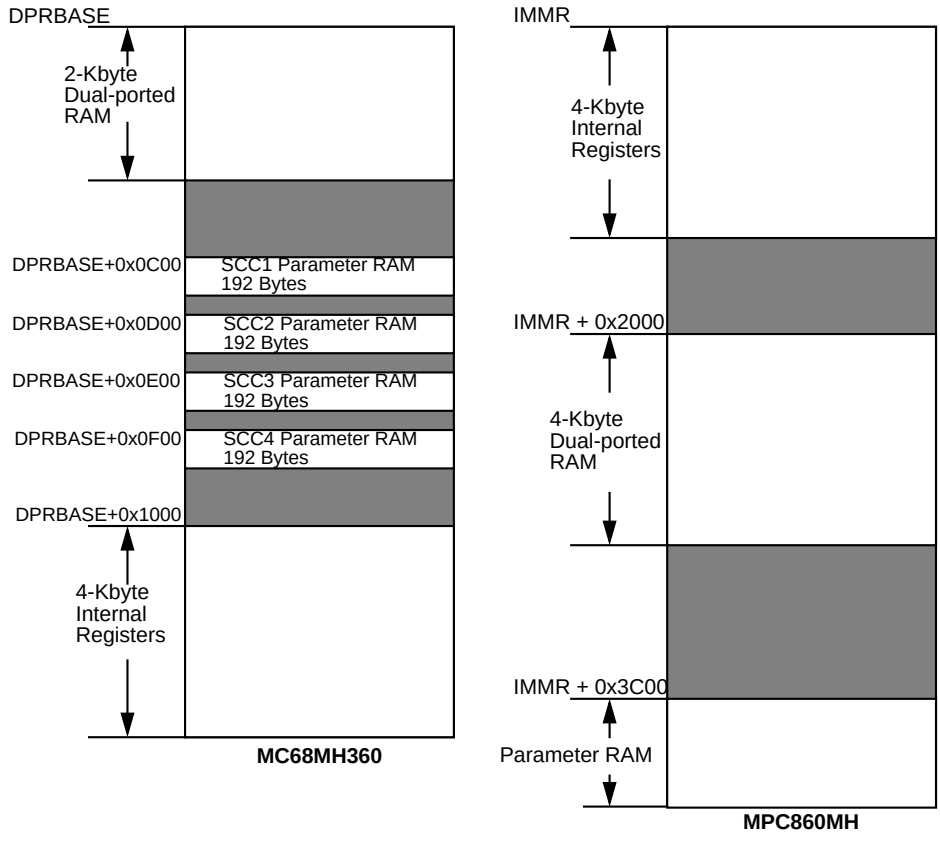


Figure 2-1. MC68MH360 and MPC860MH Internal Memory Structures

2.1 QMC Memory Structure

Figure 2-2 shows how data is addressed by the QMC protocol. It discusses addressing the dual-ported RAM to access data within the buffers.

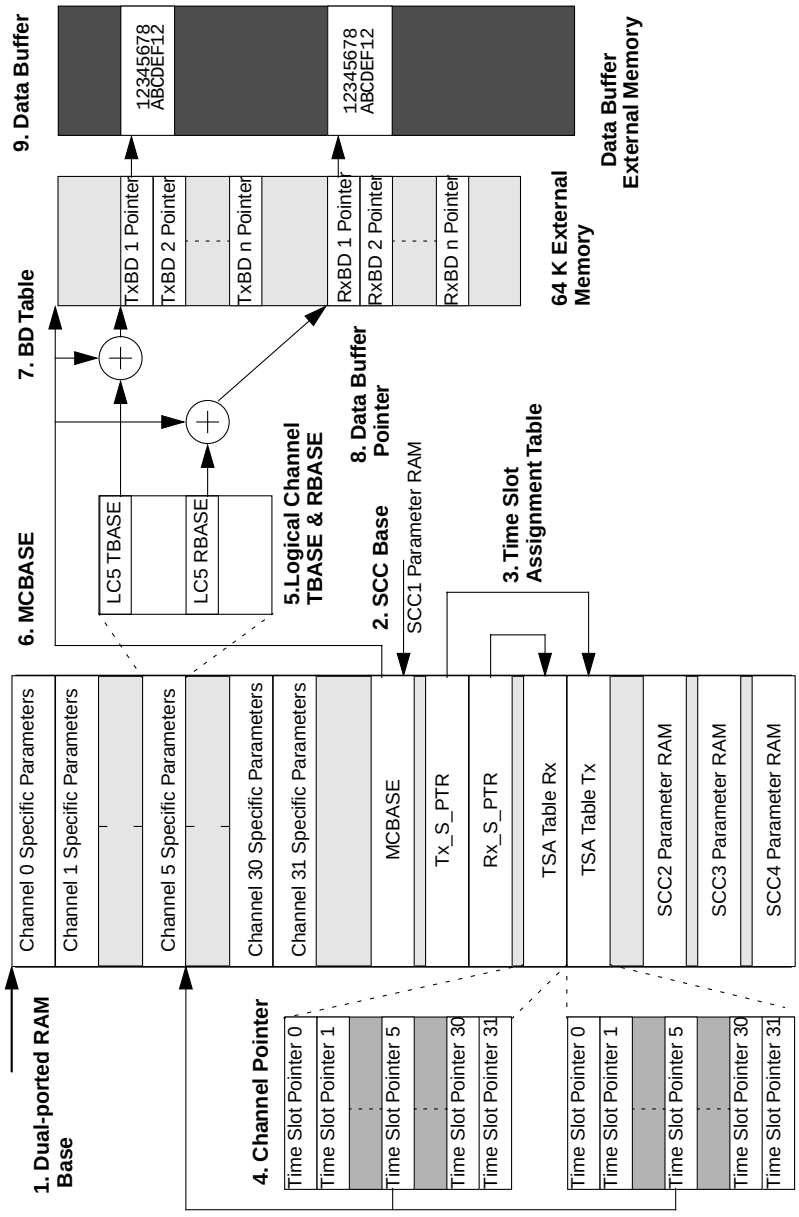


Figure 2-2. QMC Memory Structure

2.1.1 Dual-Ported RAM Base

The MC68MH360's internal memory is mapped into an 8-Kbyte block of memory, and the starting address is dictated by the DPRBASE programmed in the MBAR register. For more detail on the QUICC internal memory structure, see Section 3 of *MC68360 Quad Integrated Communications Controller User's Manual*. The MPC860MH has its internal memory mapped into a 16-Kbyte block of memory. The ISB programmed in the IMMR register determines the starting address of this memory block. For more information on the PowerQUICC internal memory structure, see Section 3 of *MPC860 PowerQUICC User's Manual*. All internal registers are addressed as offsets within the dual-ported RAM; therefore, all pointers are relative to this base address.

2.1.2 SCC Base and Global Multichannel Parameters

The SCC base points to the start of the parameter RAM for each of the SCCs at 256-byte intervals. On the MC68MH360, each SCC has 192 bytes of parameter RAM; each SCC on the MPC860MH has 256 bytes. When the QMC protocol is enabled on an SCC, its parameter RAM is used to store the global multichannel parameters for all the logical channels. This area contains parameters and pointers that are common to all channels.

NOTE

As the QMC requires 0xAF bytes of parameter RAM for its global multichannel parameters, this may cause conflict with other CPM functionality. For example, when using the MPC860MH with SCC1 in QMC mode, I^2C is unavailable.

2.1.3 TSATRx/TSATTx Pointers and Time Slot Assignment Table

The time slot assignment table pointers are within the global multichannel parameters. There are two pointers—Tx_S_PTR for transmit and Rx_S_PTR for receive. The Rx_S_PTR is normally set to SCC Base + 20; this is the normal location of the receive time slot assignment table. The Tx_S_PTR is normally set to SCC Base + 60; this is the normal location of the transmit time slot assignment table. However, if the receiver and the transmitter have the same mapping for the logical channels, Tx_S_PTR can point to SCC base + 20 so that Rx and Tx have a common time slot assignment table. Note that if a single TDM channel is routed to more than one SCC, they may also use just one time slot assignment table for all SCCs. See Section 2.3, "Multiple SCC Assignment Tables," for more information. The time slot assignment table holds one 32-bit entry for each time slot. It has options for subchanneling, a valid bit, and a logical channel pointer. For 64-channel support there is only space for one table; therefore, common Rx and Tx parameters will need to be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another SCC. Associated with the Rx/Tx_S_PTR are the Rx/TxPTR pointers that are maintained by the CPM and point to the current time slot.

2.1.4 TSATRx/TSATTx Channel Pointers

The channel pointers are 12-bit pointers to the channel-specific parameters in the internal dual-ported RAM. These should not be confused with TSATRx/TSATTx pointers as described in Section 2.1.3, “TSATRx/TSATTx Pointers and Time Slot Assignment Table.” The 6 most-significant bits of the address are taken from the time slot assignment table. For the MH360, the most-significant bit must be zero as the addressing range is only 2 Kbytes. The 6 least-significant bits are zero, mapping out a 64-byte area for each of the channel-specific parameters. The channel-specific parameters are common for Rx and Tx. For 32-channel support, 2 Kbytes of dual-ported RAM is required (32 * 64), and for 64-channel support, 4 Kbytes of dual-ported RAM is required (64 * 64). In most cases, time slot 0 channel pointer will address the base of dual-ported RAM for logical channel 0, and time slot 1 channel pointer would address the base of dual-ported RAM + 4 for logical channel 1. In Figure 2-2, time slot 5 channel pointer addresses logical channel 5, requiring the channel pointer being set to 0b000101.

NOTE

It is possible to concatenate multiple time slots to one logical channel. This is achieved by setting the channel pointers of the grouped time slots to the same logical channel.

2.1.5 Logical Channel TBASE and RBASE

TBASE and RBASE are within the channel-specific parameters. TBASE is the Tx buffer descriptor base address, and RBASE is the Rx buffer descriptor base address. These 16-bit offsets from MCBASE point to individual logical channel’s buffer descriptors located within the buffer descriptor table. Note that there are individual TBASE and RBASE values for each logical channel.

2.1.6 MCBASE

MCBASE is located in the global multichannel parameters. Each SCC has a unique MCBASE value pointing to the base of the SCC’s buffer descriptor table in external memory. For example, the address of logical channel five’s Tx buffer descriptor table is MCBASE + logical channel five TBASE.

2.1.7 Buffer Descriptor Table

A buffer descriptor table for each SCC is located in a 64-Kbyte area of external memory. This block size is determined by the TBASE and RBASE addressing range. The memory segment must be long-word-aligned but can start anywhere in memory. Each SCC has a maximum of 16,384 (64 Kbytes memory ÷ 4-byte pointers) buffers. For a 32-channel implementation, each logical channel has a maximum of 256 (16,384 / (32 * 2)) buffers for receive and 256 buffers for transmit. For each logical channel, there is a circular queue with programmable start address and length.

2.1.8 Data Buffer Pointer

As with the standard CPM protocols, the data buffer is addressed by a 32-bit pointer within the buffer descriptor. This addresses the data received or transmitted from external memory.

2.1.9 Data Buffer

The data buffers in external memory can hold up to 64 Kbytes of data as determined by the data length in the buffer descriptor.

2.2 Global Multichannel Parameters

The global multichannel parameters reside in the SCC's parameter RAM page and are common to all logical channels.

The largest portion of the global area is the time slot assigner tables for the receiver and transmitter section of the SCC. For 32-channel support, there is one table for Tx and one for Rx within the parameter RAM. If the connection is split over multiple SCCs, this table only needs to be present once for multiple SCCs operating in QMC mode. See Section 2.3, "Multiple SCC Assignment Tables," for more information. For 64-channel support there is only space for one table; therefore common Rx and Tx parameters will need to be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another SCC.

The dual-ported RAM is used for the channel-specific area for all SCCs. It is important that individual time slots are mapped to only one SCC, and that individual logical channels are separated to avoid contention.

Table 2-1 lists the global parameters. Note that the boldfaced parameters must be initialized by the user. See Chapter 6, "QMC Initialization," for more information.

Table 2-1. Global Multichannel Parameters

Offset to SCC Base	Name	Width (Bits)	Description
00	**MCBASE**	32	Multichannel base pointer—This host-initialized parameter points to the starting address of the 64-Kbyte buffer descriptor table in external memory. The MCBASE is used with the TBASE and RBASE registers in the channel-specific parameters.
04	**QMCSTATE**	16	Multichannel controller state (initialize to 0x8000)—Internal QMC state machine value used by RISC processor for global state definition.

Table 2-1. Global Multichannel Parameters (Continued)

Offset to SCC Base	Name	Width (Bits)	Description
06	**MRBLR**	16	Maximum receive buffer length—This host-initialized entry defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. This parameter is only valid in HDLC mode. The buffer area allocated in memory for each buffer is MRBLR + 4. The QMC adds another long word if non-octet-aligned frames are received in HDLC operation. The non-octet information is written only to the last buffer of a frame, but it can happen in any buffer. See Section 5.1, “Receive Buffer Descriptor,” for more information. As the QMC works on long-word alignment, MRBLR value should be a multiple of 4 bytes.
08	**Tx_S_PTR**	16	Tx time slot assignment table pointer (SCC base + 60 in normal mode; SCC base + 20 for common Rx & Tx time slot assignment tables)—This global QMC parameter defines the start value of the TSATTx table. The TSATTx table in the global multichannel parameter listing starts by default at SCC base + 60. Tx_S_PTR lets the user move the starting address of this table. If the same routing and masking are used for the transmitter and receiver, the tables can be overlaid, so Tx_S_PTR can point to SCC base + 20. This parameter is an offset from DPRBASE. This table must be present only once per SCC global area. Other SCCs can access this location.
0A	**RxPTR**	16	Rx pointer (initialize to SCC base + 20)—This global QMC parameter is a RISC variable that points to the current receiver time slot. The host must initialize this pointer to the starting location of TSATRx. The RISC processor increments this pointer whenever it completes the processing of a received time slot.
0C	**GRFTHR**	16	Global receive frame threshold—Used to reduce interrupt overhead when many short HDLC frames arrive, each causing an RXF interrupt. GRFTHR can be set to limit the frequency of interrupts. Note that the RXF event is written to the interrupt table on each received frame, but GINT is set only when the number of RXF events (by all channels) reaches the GRFTHR value. GRFTHR can be changed on the fly. For information about exception handling, see Chapter 4, “QMC Exceptions.”
0E	**GRFCNT**	16	Global receive frame count (initialized GRFCNT = GRFTHR)—A down-counter used to implement the GRFTHR feature. GRFCNT decrements for each frame received. No other receiver interrupts affect this counter. The counter value is set to the threshold during initialization. GRFCNT is automatically reset to the GRFTHR value by the CPM after a global interrupt.
10	**INTBASE**	32	Multichannel interrupt base address (host-initialized)—This pointer contains the starting address of the interrupt circular queue in external memory. Each entry contains information about an interrupt request that has been generated by the QMC to the host. Each SCC operating in QMC mode has its own interrupt table in external memory. See Chapter 4, “QMC Exceptions.”
14	**INTPTR**	32	Multichannel interrupt pointer (host-initialized)—This global parameter holds the address of the next QMC interrupt entry in the circular interrupt table. The RISC processor writes the next interrupt information to this entry when an exception occurs. The host must copy the value of INTBASE to INTPTR before enabling interrupts.

Table 2-1. Global Multichannel Parameters (Continued)

Offset to SCC Base	Name	Width (Bits)	Description
18	**Rx_S_PTR**	16	Rx time-slot assignment table pointer (default = SCC base + 20 in normal mode)—This global QMC parameter defines the start value of the TSATRx table, which must be present only once per SCC global area. Other SCCs may access this location.
1A	**TxPTR**	16	TxPTR (initialize to SCC Base + 60)—This global parameter is a RISC variable that points to the current transmitter time slot. The host must initialize it to the starting location of TSATTx. The RISC processor increments this pointer whenever it completes the processing of a transmitter time slot.
1C	**C_MASK32**	32	CRC constant (0xDEBB20E3)—Required to calculate 32-bit CRC-CCITT. C_MASK32 is written by the host during QMC initialization. It is used for 32-bit CRC-CCITT calculation if HDLC mode of operation is chosen for a selected channel. (This is a programmable option. For each HDLC channel, one of two CRCs can be chosen, as programmed in CHAMR.) For more information, see Section 2.4.1, “Channel-Specific HDLC Parameters,” and Table 2-5. This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
20	**TSATRx**	32 Entries x 16	Time slot assignment table Rx—Host-initialized, 16-bit-wide table with 32 entries that define mapping of logical channels to time slots for the QMC receiver. The QMC protocol looks at chunks of 8 bits regardless of whether they come from one physical time slot of the TDM or whatever other combination of bits the TSA transfers to the SCC. These 8 bits are referred to as a time slot in the assignment table. It is recommended but not required to route all bits from the TDM to the SCC and to do all enabling and masking in the time-slot assignment table. See Figure 2-3.
60	**TSATTx**	32 Entries x 16	Time slot assignment table Tx—Maps a specific logical channel to each physical time slot. Time slot assignment table Tx is a host-initialized, 16-bit table with 32 entries that define the mapping of channels to time slots for the QMC transmitter. The QMC protocol looks at chunks of 8 bits regardless if they go to one physical time slot of the TDM or whatever other combination of bits are transferred from the SCC to the TDM through the TSA. These 8 bits are referred to as a time slot in the assignment table. It is recommended but not required to route all bits from the TDM to the SCC and to do all enabling and masking in the time slot assignment table. See Figure 2-3.
A0	**C_MASK16**	16	CRC constant (0xF0B8)—Required to calculate 16-bit CRC-CCITT. This constant is written by the host during QMC initialization. It is used for 16-bit CRC-CCITT calculation if HDLC mode of operation is chosen for a selected channel. (This is a programmable option. For each HDLC channel, one of two CRCs can be chosen, as programmed in CHAMR.) For more information, see Section 2.4.1, “Channel-Specific HDLC Parameters,” and Table 2-5. This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
A4	TEMP_RBA	32	Temporary receive buffer address
A8	TEMP_CRC	32	Temporary cyclic redundancy check

NOTE

The area between SCC base + 20 and SCC base + 9F is normally used for TSA tables. The mapping above is ideal for 32-channel support. The exact mapping of the TSA tables is determined by the programming of Rx_S_PTR and Tx_S_PTR, and is not fixed. For 64-channel support it is suggested to use common Rx and Tx parameters. The TSA table will be common and have 64 entries starting at SCC base + 20; see Figure 2-4. Alternatively, another SCC's parameter RAM may be used, as determined by Rx_S_PTR and Tx_S_PTR; see Figure 2-6 for more information. However implemented, the TSA tables may reside anywhere in internal memory.

Figure 2-3 shows a general time slot assignment table for 32 16-bit time slots. The fields will be used to either transmit or receive channels.

Time Slot 0	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 1	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
					32 x 16
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 30	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 31	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-3. Time Slot Assignment Table

Table 2-2 describes the fields in the time slot assignment table for receive.

Table 2-2. Time Slot Assignment Table Entry Fields for Receive Section

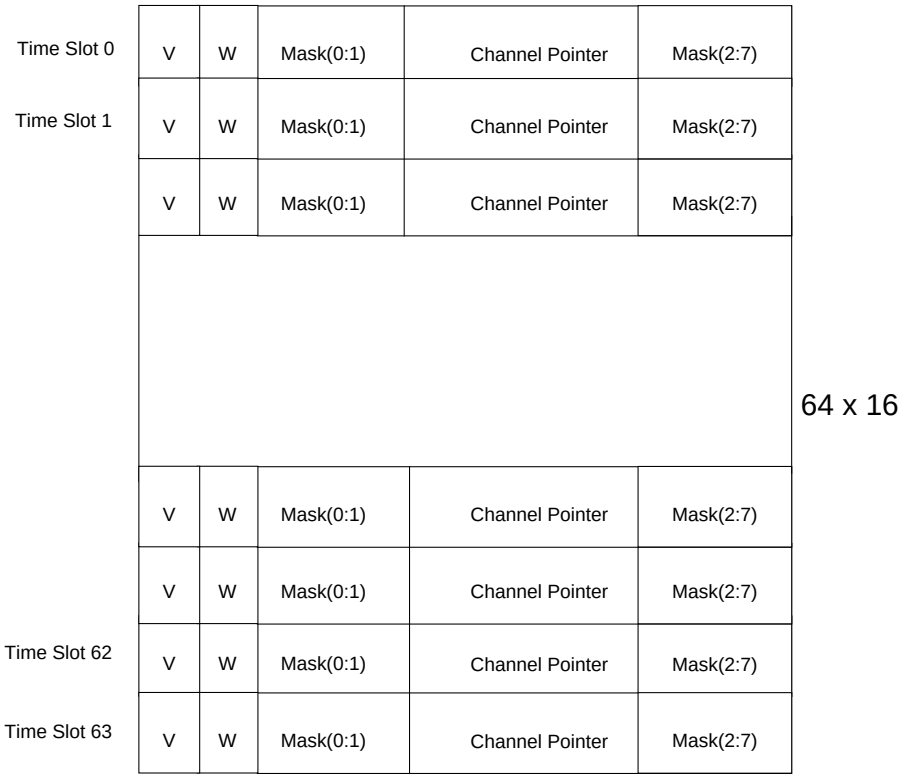
Field	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. 0 The data in this 8-bit time slot is totally ignored and not written to any buffer. 1 The data in this 8-bit time slot is valid and written to the current buffer, pointed to by the channel pointer entry, after protocol processing (e.g. zero deletion in HDLC). Individual bits can be masked out as described later.
W	Wrap bit—Identifies the last entry in TSATRx. 0 This is not the last time slot in the frame. 1 The RISC processor wraps around and handles time slot 0 or the first 8 bits transferred from the TSA in the next request. The next request is identified by a frame synchronization pulse.
Rx channel pointer	This 6-bit field of the TSATRx entry identifies the data channel routed to this time slot. The actual channel pointer is 12 bits long, and contains the starting address of the channel-specific parameter area (address of RBASE). The 6 most-significant bits are taken from the TSATRx channel pointer field, and the 6 least-significant bits are always internally set to zero. For the MH360, the most-significant bit must be set to zero, as the addressing range is 2 Kbytes.
Mask(0–7)	Mask bits—These 8 bits identify the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. Any unmasked bit (1) is processed in the receiver for a valid time slot. Any masked bit (0) is ignored by the receiver for a valid channel and no bit counter is affected.

Table 2-3 describes the fields in the time slot assignment table for transmit.

Table 2-3. Time Slot Assignment Table Entry Fields for Transmit Section

Name	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. 0 Logic 1 is transmitted. If the Tx signal of the TDM interface is programmed to be an open drain output (port B programming), other devices can transmit on nonvalid time slots. 1 Data is transmitted from its associated buffer in combination with the mask bit settings.
W	Wrap bit—The wrap bit identifies the last entry in TSATTx. 0 This is not the last time slot in the frame. 1 The RISC processor wraps around and handles time slot 0 or the first 8 bits of data in the SCC in the next request. The next request is identified by a frame synchronization pulse.
Tx channel pointer	This 6-bit field of the TSATTx entry identifies the data channel routed to this time slot. The actual channel pointer is 12 bits long, and contains the starting address of the channel-specific parameter area (address of TBASE). The 6 most-significant bits are taken from the TSATTx channel pointer field, and the 6 least-significant bits are always internally set to zero. For the MH360 the most-significant bit must be set to zero, as the addressing range is 2 Kbytes.
Mask(0–7)	Mask bits—Identifies the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. For a valid channel with an unmasked bit (1), the bit position is filled according to the protocol. A valid channel with a masked bit (0) transmits a logic high (1).

If the transmitter and receiver have the same mapping then it is possible to use a common time slot assignment table. This is initialized by setting both Tx_S_PTR and Rx_S_PTR to SCC Base + 20. For 64-channel support it is suggested to use common Rx and Tx parameters. The time slot assignment table will then also be common and have 64 entries starting at SCC Base + 20; see Figure 2-4.



Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-4. Time Slot Assignment Table for 64-Channel Common Rx and Tx Mapping

2.3 Multiple SCC Assignment Tables

Assume a scenario as depicted in Figure 2-5. A 2.048-Mbps TDM link is fed directly into the TSA. Within the SI RAM, the even channels (byte-wide) are muxed to SCC3 and the odd channels are muxed to SCC2. This arrangement is used to spread the load over two SCCs. This effectively doubles the FIFO depth on the QMC protocol. Time slots are switched to alternate SCCs to avoid data bursts that may stress the FIFOs. Each SCC sees a continuous bitstream without any gaps as described earlier.

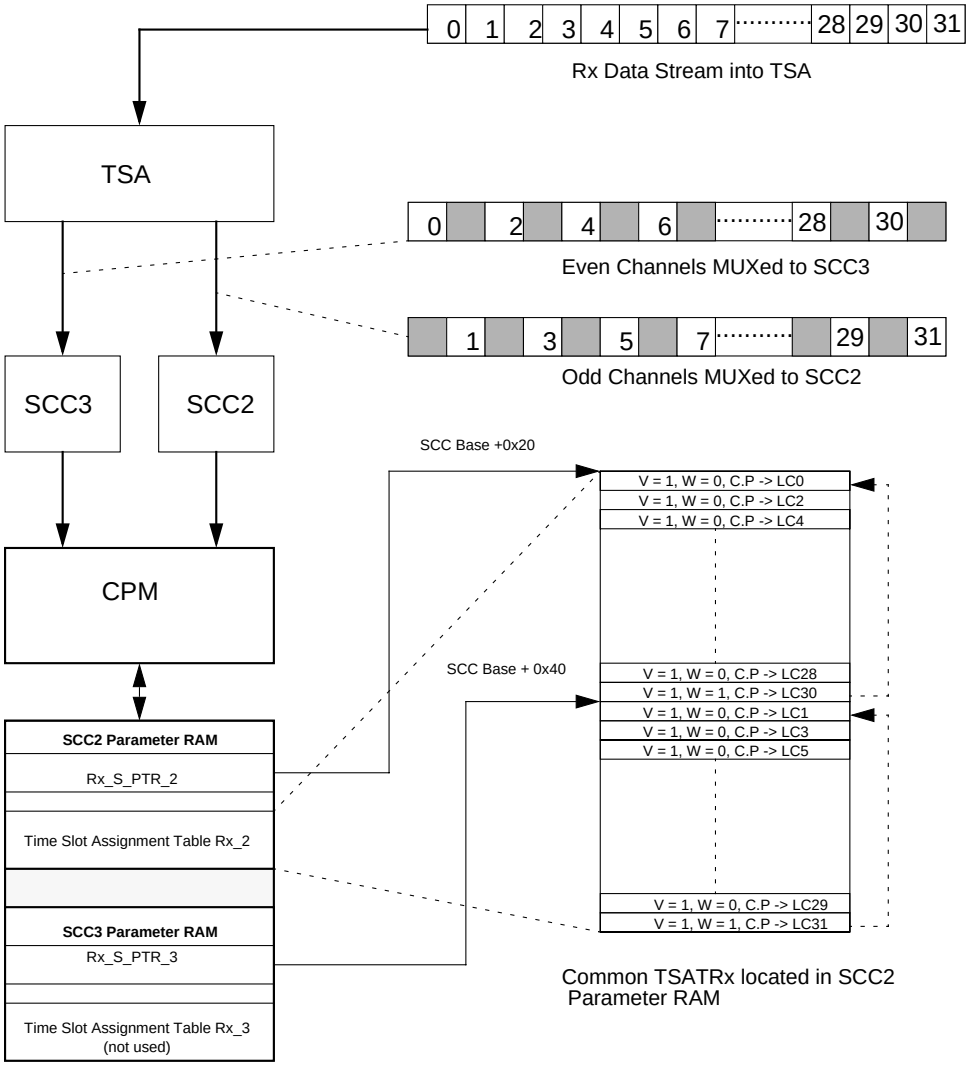


Figure 2-5. Rx Time Slot Assignment Table for 32 Channels over Two SCCs

Note

It is important that multiples of bytes are routed to each SCC to delineate between time slots. Unused bits shall be routed to the SCC and be masked in the time slot assignment table.

In Figure 2-5, each SCC has its own pointer, Rx_S_PTR_2 and Rx_S_PTR_3, addressing SCC2's time slot assignment table. This table only needs to be present once in one of the SCC2's global parameter area. Rx_S_PTR_2 points to the start of the table, address SCC base + 20. The 16 logical channels from SCC2 are located in the first 16 entries of the table. The entry for logical channel 30 has the wrap bit (W) set, causing the CPM to wrap back to logical channel 0 on reception of the next byte routed to SCC2. Rx_S_PTR_3 addresses SCC base + 40, the start of the 16 entries for SCC3. The entry for logical channel 31 has the wrap bit (W) set, causing the CPM to wrap back to logical channel 1 on reception of the next byte routed to SCC3. Each entry within the table has a channel pointer to a logical channel. It is important that different SCCs do not point to the same logical channel.

The TSATTx is also located in SCC2's parameter RAM. This means that the area reserved for the TSA tables in SCC3's parameter RAM is free for alternative use.

A second scenario is depicted in Figure 2-6. A 4.096-Mbps TDM link is fed directly into the TSA. Again, within the SI RAM, the even channels (byte-wide) are muxed to SCC3 and the odd channels are muxed to SCC2. This arrangement is used to spread the load over two SCCs. Another reason this method may be used is to facilitate separate routing for the Rx and Tx logical channels. This requires two 64-entry tables that require 256 bytes, but only 128 bytes are allocated in the parameter RAM of an SCC for time slot assignment tables. In this case, the Rx table is located in SCC2's parameter RAM, and the TX table is located in SCC3's parameter RAM, making most efficient use of memory.

Changes on the fly are easily accomplished by setting or clearing the valid bit for each time slot. Changes to the mask bits can also be made on the fly. This does not cause any problems to the QMC microcode itself, but may cause protocol errors on the channel in question depending on when this change is done.

It is possible to have a time slot assignment table for every SCC in its corresponding RAM page and have all of the TDM routed to the different SCCs. This gives the user a very flexible system that can be changed on the fly without disconnecting the TDM interface. In this case the user must ensure that no collisions occur on the transmit lines from several SCCs.

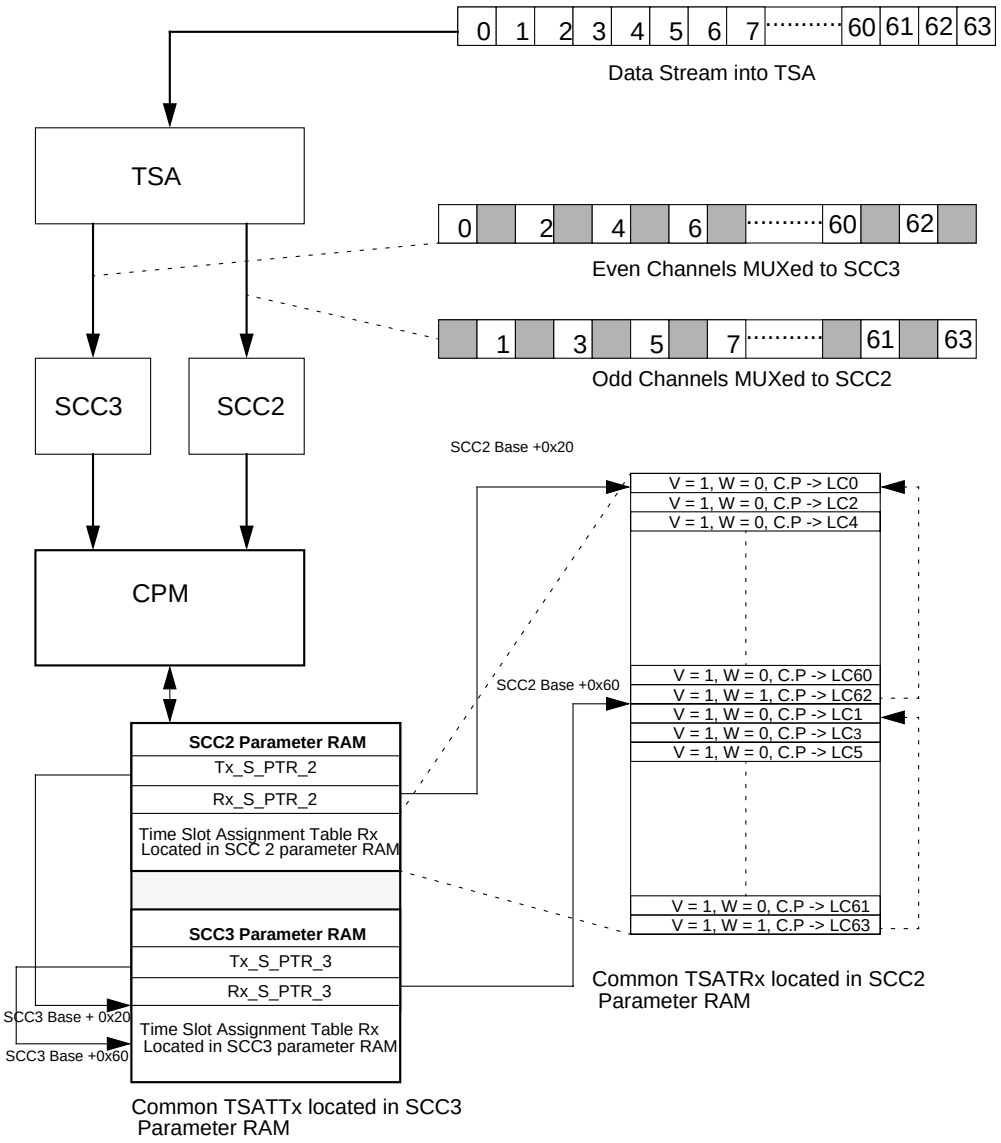


Figure 2-6. Time Slot Assignment Tables for 64 Channels over 2 SCCs

2.4 Channel-Specific Parameters

The channel-specific parameters are located in the lower part of the dual-ported RAM. Each channel occupies 64 bytes of parameters. Physical time slots can be matched to logical channels in several combinations. Unused logical channels leave a hole in the channel-specific parameters that can be used for buffer descriptors for the other SCCs.

The channel-specific area determines the operating mode—HDLC or transparent. Several entries take on different meanings depending on the protocol chosen.

2.4.1 Channel-Specific HDLC Parameters

Table 2-4 describes the channel-specific HDLC parameters. Boldfaced parameters must be initialized by the user.

Table 2-4. Channel-Specific HDLC Parameters

Offset	Name	Width (Bits)	Description
00	**TBASE**	16	Tx buffer descriptor base address—Offset of the channel's transmit buffer descriptor table relative to MCBASE, host-initialized. See Figure 2-2.
02	**CHAMR**	16	Channel mode register. See Section 2.4.1.1, "CHAMR—Channel Mode Register (HDLC)."
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Host-initialized to 0x3800_0000—FC = 8, Motorola mode for MH360. Host-initialized to 0x3000_0000— AT = 0, Motorola mode for 860MH. Initialize before enabling the channel. See Section 2.4.1.2, "TSTATE—Tx Internal State (HDLC)."
08	—	32	Tx internal data pointer—Points to current absolute address of channel.
0C	**TBPTR**	16	Tx buffer descriptor pointer (host-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel again)—Offset of current BD relative to MCBASE. See Table 2-1. MCBASE + TBPTR gives the address for the BD in use.
0E	—	16	Tx internal byte count—Number of remaining bytes
10	TUPACK	32	(Tx Temp) Unpack 4 bytes from 1 long word
14	**ZISTATE**	32	Zero-insertion state (host-initialized to 0x0000_0100 for HDLC or transparent operation)—Contains the previous state of the zero-insertion state machine.
18	TCRC	32	Temp transmit CRC—Temp value of CRC calculation result
1C	**INTMSK**	16	Channel's interrupt mask flags—See Section 2.4.1.3, "INTMSK—Interrupt Mask (HDLC)."
1E	BDFlags	16	Temp
20	**RBASE**	16	Rx buffer descriptor offset (host-initialized)— Defines the offset of the channel's receive BD table relative to MCBASE (64-Kbyte table). See Figure 2-2.

Table 2-4. Channel-Specific HDLC Parameters (Continued)

Offset	Name	Width (Bits)	Description
22	**MFLR**	16	Maximum frame length register (host-initialized)—Defines the longest expectable frame for this channel. Its maximum value is 64 Kbytes. The remainder of a frame which is larger than MFLR is discarded and a flag in the last frame's BD is set (LG). An interrupt request (RXF and RXB) might be generated depending on the interrupt mask. The frame length is considered to be everything between flags, including CRC. MFLR is checked every long word, but the content may be on any number of bytes. If MFLR is set to 5 for example, checking is done when 8 bytes have been received. At this point, the SDMA transfers the long word to memory, and all 8 bytes will be in the receive buffer. Also at this point the MFLR violation (>5) is detected and the interrupt may be generated. No more data will be written into this buffer when the MFLR violation is detected.
24	**RSTATE**	32	Rx internal state —Initialize to 0x3900_0000 FC=9, Motorola mode for MH360 or initialize to 0x3100_0000 AT=1, Motorola mode for 860MH. See Section 2.4.1.4, "RSTATE—Rx Internal State (HDLC)," for more information.
28	—	32	Rx internal data pointer—Points to current address of specific channel.
2C	**RBPTR**	16	Rx buffer descriptor pointer (host-initialized to RBASE prior to operation or due to a fatal error)—Contains the offset from MCBASE to the current receive buffer. See Table 2-1. MCBASE + RBPTR gives the address for the BD in use.
2E	—	16	Rx internal byte count—Per Channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx Temp) Packs 4 bytes to 1 long word before writing to buffer.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x0000_0080 in HDLC mode, 0x1800_0080 in transparent mode, prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of zero deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, hold the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel. More information is given in the application note section.
38	RCRC	32	Temp receive CRC—Temp value of CRC calculation result
3C	MAX_cnt	16	Max_length counter—Count length remaining
3E	TMP_MB	16	Temp—Holds MIN(MAX_cnt, Rx internal byte count)

2.4.1.1 CHAMR—Channel Mode Register (HDLC)

The channel mode register is a word-length, host-initialized register. Figure 2-7 shows the channel mode register for HDLC operation.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MODE	0	IDLM	ENT	RESERVED			POL	CRC	0	RESERVED		NOF			
Reset: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Notes: 1. All bits that are defined as reserved should be cleared (0).
2. For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-7. CHAMR—Channel Mode Register (HDLC)

Table 2-5 describes the channel mode register's fields for HDLC operation. Boldfaced parameters must be initialized by the user.

Table 2-5. CHAMR Field Descriptions (HDLC)

Field	Name	Description
0	**MODE**	Mode—Each channel has a programmable option whether to use transparent mode or HDLC mode. 0 Transparent mode 1 HDLC mode
1	—	0
2	**IDLM**	Idle mode. 0 Idle mode is disabled. No idle patterns are transmitted between frames. After transmitting the NOF + 1 flags, the transmitter starts the data of the frame. If between frames and the frame buffer is not ready, the transmitter sends flags until it can start transmitting the data. The NOF shall be greater or equal to the PAD setting; see Section 5.2, "Transmit Buffer Descriptor." If NOF = 0, this is identical to flag sharing in HDLC mode. For a high CPM load or with long bus latencies, the QMC protocol may insert additional flags. 1 Idle mode enabled. At least one idle pattern is transmitted between adjacent frames. If between frames and the frame buffer is not ready, the transmitter sends idle characters. When data is ready, the NOF + 1 flags are sent followed by the data frame. If in IDLE mode and NOF = 1, the following sequence is transmitted: .....init value, FF, FF, flag, flag, data,..... The init value before the idle will be 1's, in this case it is assumed the transmitter was uninitialized. An uninitialized SCC transmits 1s in every position.
3	**ENT**	Enable transmit. 0 Disable transmitter. If this bit is cleared and the channel's transmitter is routed to a certain time slot (within TSATTx, see Figure 2-3) the transmitter sends 1's on this time slot. 1 The transmit portion of the channel is enabled and data is sent according to protocol and to other control settings. Note that there is no ENR bit in the QMC protocol. To enable the receiver, the ZDSTATE and RSTATE parameters shall be set to their initial values.
4–6	—	Reserved
7	**POL**	Enable polling. This bit enables the transmitter to poll the transmit buffer descriptors. 0 The CPM does not check the ready bit (R) in the transmit buffer descriptor. 1 The CPM checks the ready bit (R) in the transmit buffer descriptor. The user can use this bit to prevent unnecessary external bus cycles when checking the ready bit (R) in the buffer descriptor. This bit should always be set by the software at the beginning of a transmit sequence of one or more frames. This bit is cleared (0) by the RISC processor when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared (0), i. e., at the end of a frame or at the end of a multiframe transmission. In order to prevent deadlock the software should always prepare the new BD, or multiple BDs, and set (1) the ready bit in the BD, before setting (1) the POL bit. Note that as this bit is automatically cleared by the CPM; the user should not attempt to clear this bit in software.
8	**CRC**	This bit selects the type of CRC when using the HDLC channel mode. 0 16-bit CCITT-CRC is selected for this channel. 1 32-bit CCITT-CRC is selected.
9	—	0

Table 2-5. CHAMR Field Descriptions (HDLC) (Continued)

Field	Name	Description
10–11	—	Reserved
12–15	**NOF**	Number of flags—Defines the minimum number of flags before frames. However, even if NOF = 0, at least one flag is transmitted before the first frame. See the description of the IDLM bit for more information.

2.4.1.2 TSTATE—Tx Internal State (HDLC)

TSTATE defines the internal transmitter state. The high byte of TSTATE defines the function code/address type and the Motorola/Intel bit. Bit 3 (or bit 28 for the 68360) should always be set to 1. Figure 2-8 shows the TSTATE register for HDLC operation.

0	1	2	3	4	5	6	7
0	0	1	MOT	FC[3–0]/ AT[1–3]			

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-8. TSTATE—Tx Internal State (HDLC)

For the MH360, TSTATE should be host-initialized to 0x3800_0000 before enabling the channel—function code 8. Table 2-6 describes the TSTATE fields for the MH360 with boldfaced parameters to be initialized by the user.

Table 2-6. TSTATE Field Descriptions for MH360 (HDLC)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4–7	**FC[3–0]**	Function code—This field contains the function code for the transmitter DMA channel for data buffers in external memory (transmit buffers). Function codes are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

For the 860MH, TSTATE should be host-initialized to 0x3000_0000 before enabling the channel—AT = 0. Note that for the 860MH, bit 4 should always be zero as only bits 5–7 map to AT[1–3]. Table 2-7 describes the TSTATE fields for the 860MH with boldfaced parameters to be initialized by the user.

Table 2-7. TSTATE Field Descriptions for 860MH (HDLC)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4	—	0
5–7	**AT[1–3]**	Address type—This field contains the address type for the transmitter DMA channel for data buffers in external memory (transmit buffers). Address types are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

2.4.1.3 INTMSK—Interrupt Mask (HDLC)

Each event defined in the interrupt circular queue entry maps directly to a bit in INTMSK as shown in Figure 2-9. There is one mask bit for each event—NID (bit 2), IDL (bit 3), MRF (bit 10), UN (bit 11), RXF (bit 12), BSY (bit 13), TXB (bit 14) and RXB (bit 15). Bits that do not map to an event are reserved. Reserved bits must be set to zero. Refer to Chapter 4, “QMC Exceptions,” for more detail.

- 0 = No interrupt request is generated and no new entry is written in the circular interrupt table.
- 1 = Interrupts are enabled.

This register is initialized by the host prior to operation.

Interrupt Table Entry:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	W	NID	IDL	—	CHANNEL NUMBER					MRF	UN	RXF	BSY	TXB	RXB
Reset: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTMSK:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED		INTERRUPT MASK		RESERVED						INTERRUPT MASK BITS					
RESET: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-9. INTMSK and Interrupt Table Entry (HDLC)

2.4.1.4 RSTATE—Rx Internal State (HDLC)

Host-initialized to 0x3900_0000 before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command.

The high byte of RSTATE defines the function code/address type and the Motorola/Intel bit. Bit 3 (or bit 28 for the 68360) should always be set to 1. Figure 2-10 shows the RSTATE register for HDLC operation.

0	1	2	3	4	5	6	7
0	0	1	MOT	FC[3–0]/ AT[1–3]			

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-10. RSTATE—Rx Internal State (HDLC)

For the MH360, RSTATE should be host-initialized to 0x3900_0000 before enabling the channel—function code 9. Table 2-8 describes the RSTATE fields for the MH360 with boldfaced parameters to be initialized by the user.

Table 2-8. RSTATE Field Descriptions for MH360 (HDLC)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4–7	**FC[3–0]**	Function code—This field contains the function code for the transmitter DMA channel for data buffers in external memory (transmit buffers). Function codes are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

For the 860MH, RSTATE should be host-initialized to 0x3100_0000 before enabling the channel—AT = 1. Note that for the 860MH, bit 4 should always be zero as only bits 5–7 map to AT[1–3]. Table 2-9 describes the RSTATE fields for the 860MH with boldfaced parameters to be initialized by the user.

Table 2-9. RSTATE Field Descriptions for 860MH (HDLC)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4	—	0
5–7	**AT[1–3]**	Address type—This field contains the address type for the transmitter DMA channel for data buffers in external memory (transmit buffers). Address types are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

2.4.2 Channel-Specific Transparent Parameters

Table 2-10 describes the channel-specific transparent parameters. Boldfaced parameters must be initialized by the user.

Table 2-10. Channel-Specific Transparent Parameters

Offset	Name	Width	Description
00	**TBASE**	16	Tx buffer descriptor base address—Defines the offset of the channel's transmit BD table relative to MCBASE, host-initialized. See Figure 2-2.
02	**CHAMR**	16	Channel mode register. See Section 2.4.2.1, "CHAMR—Channel Mode Register (Transparent Mode)."
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Host-initialized to 0x3800_0000—FC = 8, Motorola mode for MH360. Host-initialized to 0x3000_0000—AT = 0, Motorola mode for 860MH. Initialize before enabling the channel. See Section 2.4.2.2, "TSTATE—Tx Internal State (Transparent Mode)."
08		32	Tx internal data pointer—Points to current absolute address of channel.
0C	**TBPTR**	16	Tx buffer descriptor pointer (host-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel)—Contains the offset of current BD relative to MCBASE. See Table 2-1. MCBASE + TBPTR gives the address for the BD in use.
0E		16	Tx internal byte count—Number of remaining bytes
10	TUPACK	32	(Tx temp) Unpack 4 bytes from 1 long word
14	**ZISTATE**	32	Zero-insertion machine state (host-initialized to 0x0000_0100)—Contains the previous state of the zero-insertion state machine.
18	RES	32	
1C	**INTMSK**	16	Channel's interrupt mask flags. See Figure 2-9.
1E	BDFlags	16	Temp
20	**RBASE**	16	Receive buffer descriptor base offset—Defines the offset of the channel's 64-Kbyte receive BD table relative to MCBASE. Host-initialized. See also Figure 2-2.

Table 2-10. Channel-Specific Transparent Parameters (Continued)

Offset	Name	Width	Description
22	**TMRBLR**	16	Transparent maximum receive buffer length (host-initialized entry)—Defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. Note that this value must be a multiple of 4 bytes as the QMC works on long-word alignment.
24	**RSTATE**	32	Rx internal state —Initialize to 0x3900_0000 FC = 9, Motorola mode for MH360, initialize to 0x3100_0000 AT = 1, Motorola mode for 860MH. See Section 2.4.2.5, “RSTATE—Rx Internal State (Transparent Mode),” for more information.
28		32	Rx internal data pointer—Points to current address of specific channel.
2C	**RBPTR**	16	Rx buffer descriptor pointer (host-initialized to RBASE, prior to operation or due to a fatal error)—Contains the offset from MCBASE to the current receive buffer. See Figure 2-2. MCBASE + RBPTR gives the address for the BD in use.
2E		16	Rx internal byte count—Per Channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx temp)—Packs 4 bytes to 1 long word before writing to buffer.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x0000_0080 in HDLC mode, 0x1800_0080 in transparent mode, prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of the zero-deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, holds the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel.
38	RES	32	
3C	**TRNSYNC**	16	Transparent synchronization—In transparent mode, this register controls synchronization for single time slots or superchannel applications. See Section 2.4.2.4, “TRNSYNC—Transparent Synchronization.”
3E	RES	16	

2.4.2.1 CHAMR—Channel Mode Register (Transparent Mode)

The channel mode register is a word-length, host-initialized register. Figure 2-11 shows the channel mode register for transparent mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MODE	RD	1	ENT	RES'D	SYNC	RES	POL	0	0	RESERVED		0			
Reset: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Notes: 1. All bits defined as reserved are cleared (0).
2. For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-11. CHAMR—Channel Mode Register (Transparent Mode)

Table 2-11 describes the channel mode register’s fields for transparent operation. Boldfaced parameters must be initialized by the user.

Table 2-11. CHAMR Bit Settings (Transparent Mode)

Field	Name	Description
0	**MODE**	Mode—Each channel has a programmable option whether to use transparent mode or HDLC mode. 0 Transparent mode 1 HDLC mode
1	**RD**	Reverse data 0 The bit order will not be reversed, transmitting/receiving the LSB of each octet first. 1 The bit order as seen on the channels is reversed, transmitting/receiving the MSB of each octet first.
2	—	1
3	**ENT**	Enable transmit 0 Disable transmitter. If this bit is cleared and the channel's transmitter is routed to a certain time slot (within TSATTx, see Figure 2-3) the transmitter sends 1's on this time slot. 1 The transmit portion of the channel is enabled and data is sent according to protocol and to other control settings.
4	—	Reserved
5	**SYNC**	Synchronization—Controls synchronization of multichannel operation in transparent mode. 0 The first byte is put in the first available time slot or is read from the first available time slot to this logical channel. 1 The synchronization algorithm according to TRANSYNC is done.
6	RES	Reserved
7	**POL**	Enable polling—Enables the transmitter to poll the transmit BDs. 0 The CPM will not check the ready (R) bit in the transmit buffer descriptor. 1 The CPM will go check the ready (R) bit in the transmit buffer descriptor. The user can use this bit to prevent unnecessary external bus cycles when checking the ready bit (R) in the buffer descriptor. Software should always set POL at the beginning of a transmit sequence of one or more frames. The RISC processor clears POL (0) when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared (0), that is, at the end of a frame or at the end of a multiframe transmission. To prevent deadlock, software should prepare the new BD, or multiple BDs, and set (1) the ready (R) bit in the BD before setting (1) POL. Note that the CPM automatically clears this bit; the user should never try to clear this bit in software.
8–9	—	0
10–11	—	Reserved
12–15	—	0

2.4.2.2 TSTATE—Tx Internal State (Transparent Mode)

TSTATE defines the internal transmitter state. The high byte of TSTATE defines the function code/address type and the Motorola/Intel bit (bit 3) that should always be set to 1. Figure 2-12 shows the TSTATE register for transparent mode.

0	1	2	3	4	5	6	7
0	0	1	MOT	FC[3–0]/ AT[1–3]			

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-12. TSTATE—Tx Internal State (Transparent Mode)

For the MH360, TSTATE should be host-initialized to 0x3800_0000 before enabling the channel—function code 8. Table 2-12 describes the TSTATE fields for the MH360 with boldfaced parameters to be initialized by the user.

Table 2-12. TSTATE Field Descriptions for MH360 (Transparent Mode)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4–7	**FC[3–0]**	Function code—This field contains the function code for the transmitter DMA channel for data buffers in external memory (transmit buffers). Function codes are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

For the 860MH, TSTATE should be host-initialized to 0x3000_0000 before enabling the channel—AT = 0. Note that for the 860MH bit 4 should always be zero as only bits 5–7 map to AT[1–3]. Table 2-13 describes the TSTATE fields for the 860MH with boldfaced parameters to be initialized by the user.

Table 2-13. TSTATE Field Descriptions for 860MH (Transparent Mode)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4	—	0
5–7	**AT[1–3]**	Address type—This field contains the address type for the transmitter DMA channel for data buffers in external memory (transmit buffers). Address types are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

2.4.2.3 INTMSK—Interrupt Mask (Transparent Mode)

Each event defined in the interrupt circular queue entry maps directly to a bit in INTMSK as shown in Figure 2-13. There is one mask bit for each event—UN (bit 11), BSY (bit 13), TXB (bit 14) and RXB (bit 15). Bits that do not map to an event are reserved. Reserved bits must be set to zero.

- 0 = No interrupt request is generated and no new entry is written in the circular interrupt table.
- 1 = Interrupts are enabled.

This register is initialized by the host before operation.

INTERRUPT TABLE ENTRY:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	W	RES	RES	–	CHANNEL NUMBER					RES	UN	RES	BSY	TXB	RXB
RESET: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTMSK:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED		RESERVED		RESERVED						RES	INTERRUPT MASK BITS				
Reset: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-13. INTMSK and Interrupt Table Entry (Transparent Mode)

2.4.2.4 TRNSYNC—Transparent Synchronization

In transparent mode, the TRNSYNC register controls the synchronization for single time slots or superchannel applications.

Note

This register has no meaning if the SYNC bit in the channel mode register (CHAMR) is cleared (0).

When sending a transparent message over several time slots, it is necessary to know in which time slot the first byte of data appears.

The TRNSYNC word-length register is divided into two parts with the high byte controlling the first received time slot and the low byte controlling the transmitter synchronization.

Take the example of a superchannel of several time slots:

TSn, TSn + 1, TSn + 2 ..... TSn + x

The algorithm for the receiver byte in decimal is:

(TSn + 1) * 2

The algorithm for the transmit byte in decimal is:

(TSn + x + 1) * 2

The result from these calculations is a decimal value programmed into TRNSYNC.

NOTE

Note that TS*n* is not necessarily the first time slot in the frame. For example, if a superchannel is produced from TS2, TS4, and TS6, the message may be arranged with TS4 holding the first byte, then TS6, and the final byte held in TS2 of the following frame.

The following nine cases in Figure 2-14, named C1 to C9, show different scenarios ranging from a single time slot per logical channel to a superchannel using several time slots. In this application, 24 time slots are routed to this SCC from the SI RAM. After time slot 23, the frame starts with 0 again. The arrow in all the figures illustrates the starting position.

C1 is for a single byte in TS7, so TSn = 7

Rx Byte: (7+1) * 2 = 16
As x = 0, TSn + x = TSn = 7, so
TX Byte: (7 + 1) * 2 = 16

C2 is a single byte in TS23, so TSn = 23. Note that time slot after 23 is 0, so in the calculations below 23 + 1 = 0.

Rx Byte: (23 + 1) * 2 = 0
As x = 0, TSn + x = TSn = 23, so
TX Byte: (23 + 1) * 2 = 0

C3 is a 2-byte pattern TS7, TS8, so TSn = 7

Rx Byte: (7 + 1) * 2 = 16
As x = 1, TSn + x = 8, so
TX Byte: (8 + 1) * 2 = 18

C4 is a 2-byte pattern TS8, TS7, so TSn = 8

Rx Byte: (8+1) * 2 = 16
As x = 1, TSn + x = 7, so
TX Byte: (7 + 1) * 2 = 16

C5 is a 2-byte pattern TS19, TS23, so TSn = 19

Rx Byte: (19 + 1) * 2 = 40
As x = 1, TSn + x = 23, so
TX Byte: (23 + 1) * 2 = 0

C6 is a 2-byte pattern TS23, TS19, so TSn = 23

Rx Byte: (23 + 1) * 2 = 0
As x = 1, TSn + x = 19, so;
TX Byte: (19 + 1) * 2 = 40

C7 is a 4-byte pattern TS20, TS23, TS8, TS9 & TS19, so TSn = 20

Rx Byte: (20 + 1) * 2 = 42
As x = 5, TSn + x = 19, so
TX Byte: (19 + 1) * 2 = 40

C8 is a 4-byte pattern TS8, TS9, TS19, TS20 & TS23, so TSn = 8

Rx Byte: (8 + 1) * 2 = 18
As x = 5, TSn + x = 23, so
TX Byte: (23 + 1) * 2 = 0

C9 is a 4-byte pattern TS19, TS20, TS23, TS8 & TS9, so TSn = 19

Rx Byte: (19 + 1) * 2 = 40
As x = 5, TSn + x = 9, so
TX Byte: (9 + 1) * 2 = 20

NOTE

Case C1 and C2 can be used as described above. A more elegant solution for single time slot applications is to have the SYNC bit cleared (0) in the channel mode register. Both scenarios produce the same result.

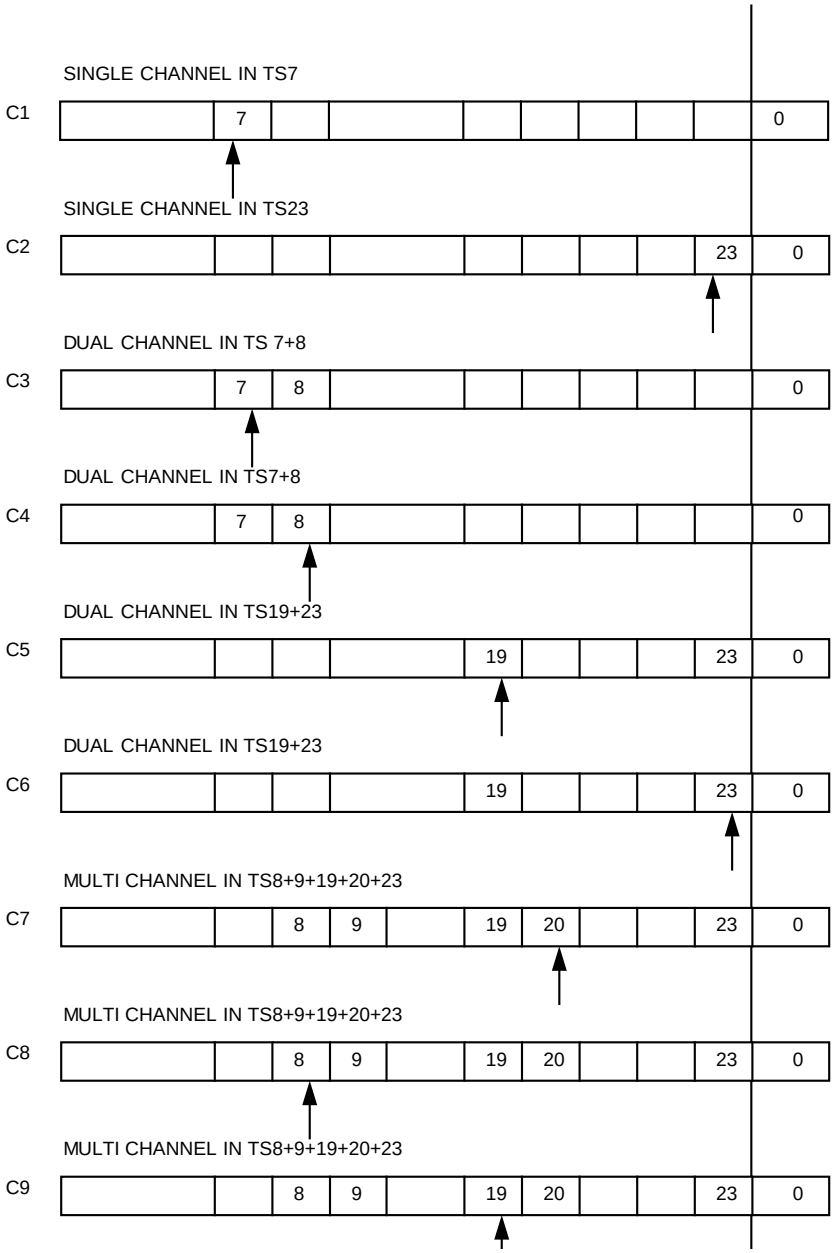


Figure 2-14. Examples of Different T1 Time Slot Allocation

2.4.2.5 RSTATE—Rx Internal State (Transparent Mode)

The RSTATE is host-initialized before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command.

The high byte of RSTATE defines the function code/address type and the Motorola/Intel bit. Bit 3 (or bit 29 for the 68360) should always be set to 1. Figure 2-15 shows the RSTATE register for transparent mode.

0	1	2	3	4	5	6	7
0	0	1	MOT	FC[3–0]/ AT[1–3]			

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 2-15. RSTATE—Rx Internal State (Transparent Mode)

For the MH360, RSTATE should be initialized to 0x3900_0000 before enabling the channel—function code 9. Table 2-14 describes the RSTATE fields for the MH360 with boldfaced parameters to be initialized by the user.

Table 2-14. RSTATE Field Descriptions for MH360 (Transparent Mode)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4–7	**FC[3–0]**	Function code—This field contains the function code for the transmitter DMA channel for data buffers in external memory (transmit buffers). Function codes are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

For the 860MH, RSTATE should be initialized to 0x3100_0000 before enabling the channel —AT = 1. Note that for the 860MH, bit 4 should always be zero as only bits 5–7 map to AT[1–3]. Table 2-15 describes the RSTATE fields for the 860MH with boldfaced parameters to be initialized by the user.

Table 2-15. RSTATE Field Descriptions for 860MH (Transparent Mode)

Field	Name	Description
0–1	—	0
2	—	1
3	**MOT**	Motorola/Intel bit 0 = The bus format is Intel format (little-endian). 1 = The system bus is considered to be organized in Motorola format (big-endian).
4	—	0
5–7	**AT[1–3]**	Address type—This field contains the address type for the transmitter DMA channel for data buffers in external memory (transmit buffers). Address types are needed by the memory controller to decode a correct memory cycle and activate the correct handshaking.

Chapter 3
QMC Commands

The host issues commands to the QMC by writing to the command register (CR). The QMC commands are similar to those of standard QUICC HDLC protocol. The CR format for QMC is shown in Figure 3-1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RST	QMC OPCODE			1	1	1	0	0	CHANNEL NUMBER					–	FLG

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 3-1. Command Register (CR)

3.1 Transmit Commands

STOP TRANSMIT <channel> (QMC opcode = 001)

The stop transmit command disables the transmission of data on the selected channel and clears the POL bit in the CHAMR register. Upon asserting this command in the middle of a frame, the RISC processor sends an ABORT indication (7F) followed by IDLEs or FLAGs, depending on the mode, on the selected channel. If this command is issued between frames, the RISC processor continues sending IDLEs or FLAGs (depending on the IDLM mode bit in the CHAMR register) in this channel.

The Tx buffer descriptor pointer (TBPTR) is not advanced to the next buffer; see Table 2-4 and Section 2.2, “Global Multichannel Parameters.”

Set (1) the POL bit to start transmission or to continue after a stop command.

Only after transmission start for a deactivated channel, which is identified by a cleared (0) V bit in the time slot assignment table or a cleared (0) ENT bit, is it necessary to initialize ZISTATE and TSTATE before setting (1) the POL bit.

To deactivate a channel, clear the V bit in the TSA table and the ENT bit in the channel mode register (CHAMR).

3.2 Receive Commands

STOP RECEIVE<channel> (QMC opcode = 000)

The stop receive command forces the receiver of the selected channel to stop receiving. After issuing this command, the microcode does not change any of the receive parameters in the dual-ported RAM.

Initialize ZDSTATE and RSTATE to their initial values to start reception or to continue receiving after a stop command.

NOTE

No commands exist to start transmission and reception. This function is realized via the GSMR register.

Chapter 4
QMC Exceptions

QMC interrupt handling involves two principle data structures—the SCC event register (SCCE) and the circular interrupt table. Figure 4-1 illustrates the circular interrupt table.

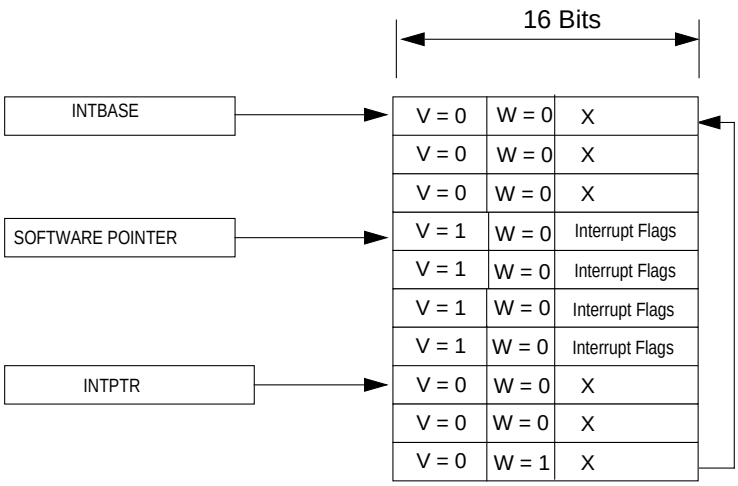


Figure 4-1. Circular Interrupt Table in External Memory

INTBASE (interrupt base) points to the starting location of the queue in external memory, and INTPTR (interrupt pointer) marks the current empty position available to the RISC processor. Both pointers are host-initialized global QMC parameters; see Table 2-1. The entry whose W (wrap) bit is set to 1 marks the end of the queue. When one of the QMC channels generates an interrupt request, the RISC processor writes a new entry to the queue. In addition to the channel's number, this entry contains a description of the exception. The V (valid) bit is then set and INTPTR is incremented. When INTPTR reaches the entry with W = 1, INTPTR is reset to INTBASE.

An interrupt is written to the interrupt table only if it survives a mask with the INTMASK (interrupt mask) register. Following a write to the queue, the QMC protocol updates the SCC event register (SCCE) according to the type of exception.

Following a request that is not masked out by the INTMASK or the SCCM (SCC mask) register, an interrupt is generated to the host. The host reads the SCCE to determine the cause of interrupt. A dedicated SCCE bit (GINT) indicates that at least one new entry was added to the queue. After clearing GINT, the host starts processing the queue. The host then clears this entry's valid bit (V). The host follows this procedure until it reaches an entry with V = 0, indicating an invalid entry.

NOTE

It is important that the user clear all interrupt flags in a queue entry even though its valid bit may be cleared since old flags are not necessarily overwritten with each new event.

4.1 Global Error Events

A global error affects the operation of the SCC. A global error can occur for two reasons—serial data rates being too high for the CPM to handle, and CPM bus latency being too long for correct FIFO operation.

There are two global errors— global transmitting underrun (GUN) and global receiver overrun (GOV). GUN indicates that transmission has failed due to lack of data; and GOV indicates that the receiver has failed because the RISC processor did not write previous data to the receive buffer. In both cases, it is unknown which channel(s) are affected.

Nonglobal, individual channel errors are handled differently. See Section 4.3, "Interrupt Table Entry," for underrun and overrun in a specific channel.

The incoming data to the CPM is governed by transfers between the SCC and the SI. Every transfer in either direction causes a request to the CPM state machine. If requests are received too quickly, the CPM crashes due to an overload of serial data. This causes a global error depending on whether it happened in the transmit side or the receive side. This error affects all QMC channels.

The other error condition is bus latency. A receiving channel submits data to the FIFO for transfer to external memory as long as the channel operates normally. If the bus latency for the SDMA channels is too long and the receive FIFO is filled and overwritten, a receiver overflow occurs. The overwriting channels cannot be traced, affecting entire QMC operation.

A similar situation can occur during transmission when the SDMA cannot fill the FIFO from external memory because of bus latency. Again, it cannot be determined which channel is underrun, and the whole QMC operation is affected.

Global errors are unlikely to occur in normal system operation, if correct serial speed is used. The only area of concern is data movement between the FIFO and external memory. To avoid problems, the user must understand the bus arbitration mechanism of the QUICC and meet the latency requirements; see Chapter 8, "Performance," for more information.

4.1.1 Global Underrun (GUN)

The QMC performs the following actions when it detects a GUN event:

- Transmits an abort sequence of minimum sixteen 1's in each time slot.
- Generates an interrupt request to the host (if enabled) and sets the GUN bit in the SCCE register.
- Stops reading data from buffer.
- Sends IDLEs or FLAGs in all time slots depending on channel mode settings until the host does the following:

  Host initializes all transmitting channels and time slots by preparing all buffer descriptors for transmission (R bits are set) and setting the POL bit. No other re-initialization is needed.

4.1.2 Global Overrun (GOV) in the FIFO

A global overrun affects all channels operating from an SCC. Following GOV, the QMC performs the following:

- Updates the RSTATE register to prevent further reception on this channel. Bit 20 in the RSTATE register indicates that the receiver is stopped.
- Generates an interrupt request to the host (if enabled) and sets the GOV bit in the SCCE.
- Stops writing data to all channel's buffers.
- Waits for host to initialize all the receiving channels by setting first the ZDSTATE followed by the RSTATE to their initial values.

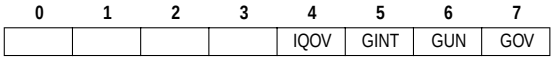
4.1.3 Restart from a Global Error

The last two bullets in the above two sections describe the only steps necessary for re-initialization. The transmit and receive sections must be restarted individually for each separate logical channel.

For details about initialization, see Chapter 6, "QMC Initialization."

4.2 SCC Event Register (SCCE)

The QMC's SCCE is a word-length register used to report events and generate interrupt requests. See Figure 4-2 and Table 4-1 for SCCE field descriptions. For each of its flags, a corresponding programmable mask/enable bit in the SCCM determines whether an interrupt request is generated. If a bit in the SCCM register is zero, the corresponding interrupt flag does not survive, and the CPM does not proceed with its usual interrupt handling. If a bit in the SCCM is set, the corresponding interrupt flag in the SCCE survives, and the SCC event bit is set in the CPM interrupt-pending register. See Figure 4-3 for SCCM assignments.



Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 4-2. SCC Event Register

Table 4-1. SCC Event Register Field Descriptions

Field	Name	Description
0–3	—	Reserved
4	IQOV	Interrupt table (interrupt queue) overflow 0 No interrupt table overflow has occurred. 1 An overflow condition in the circular interrupt table occurs (and an interrupt request is generated). This condition occurs if the RISC processor attempts to write a new interrupt entry into an entry that was not handled by the host. Such an entry is identified by V = 1. This entry is cleared by the software immediately after entering the interrupt routine. When this occurs, the last interrupt is lost and not overwritten on the first entry.
5	GINT	Global interrupt 0 No global interrupt has occurred. 1 This flag indicates that at least one new entry in the circular interrupt table has been generated by the QMC. The host clears GINT by writing a 1 to its location in SCCE. After clearing it, the host reads the next entry from the circular interrupt table, and starts processing a specific channel's exception. The user must make sure that no more valid interrupts are pending in the interrupt table after clearing the GINT bit, before performing the RTE to avoid deadlock. This procedure ensures that no pending interrupts exist in the queue.
6	GUN	Global transmitter underrun 0 No global transmitter underrun has occurred. 1 This flag indicates that an underrun occurred in the SCC's transmitter FIFO. This error is fatal since it is unknown which channel(s) are affected. Following the assertion of the GUN bit in the SCCE, the QMC stops transmitting data on all channels. The TDM Tx line goes into idle mode. This error affects only the transmitter; the receiver continues to work. After initializing all the individual channels, the host may resume transmitting. If enabled in the SCCM, an interrupt request is generated when GUN is set. The host may clear GUN by writing 1 to its location in the SCCE.
7	GOV	Global receiver overrun 0 No global receiver overrun has occurred. 1 This flag indicates that an overrun occurred in the SCC's transmitter FIFO. This error is fatal since it is unknown which channel(s) are affected. Following the assertion of the GOV bit in the SCCE, the QMC stops receiving data on all channels. Data is no longer written to memory. This error affects only the receiver; the transmitter continues to work. After initializing all the individual channels, the host may resume receiving. If enabled in SCCM, an interrupt request is generated when GOV is set. The host may clear GOV by writing 1 to its location in the SCCE.

0	1	2	3	4	5	6	7
				IQOV	GINT	GUN	GOV

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 4-3. SCCM Register

4.3 Interrupt Table Entry

The interrupt table contains information about channel-specific events. Its flags are shown in Figure 4-4. Note that some bits have no meaning when operating in transparent mode. For more detailed description on which bits are used in HDLC and transparent operation, refer to Section 2.4, “Channel-Specific Parameters.” Table 4-2 describes the fields of an interrupt table entry.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	W	NID	IDL	—	CHANNEL NUMBER					MRF	UN	RXF	BSY	TXB	RXB
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 4-4. Interrupt Table Entry

Table 4-2. Interrupt Table Entry Field Descriptions

Field	Name	Description
0	V	Valid bit 0 = Entry is not valid. 1 = Valid entry containing interrupt information. Upon generating a new entry, the RISC processor sets this bit. The V bit is cleared by the host immediately after it reads the interrupt flags in this entry (before processing the interrupt). The V bits in the queue are host-initialized. During the initialization procedure, the host must clear those bits in all queue entries.
1	W	Wrap bit 0 = This is not the last entry in the circular interrupt table. 1 = This is the last circular interrupt table entry. The next event’s entry is written/read (by RISC/host) from the address contained in INTBASE. During initialization, the host must clear all W bits in the queue except the last one which must be set. The length of the queue is left to the user and can be a maximum of 64 Kbytes.
2	NID	Not idle 0 = No NID event has occurred. 1 = A pattern which is not an idle pattern was identified. NID interrupts are not generated in transparent mode.

Table 4-2. Interrupt Table Entry Field Descriptions (Continued)

Field	Name	Description
3	IDL	Idle 0 = No IDL event has occurred. 1 = The channel's receiver has identified the first occurrence of HDLC idle (FFFE) after any non-idle pattern. IDL interrupts are not generated in transparent mode.
4		Reserved
5–9	Channel Number	This 5-bit field identifies the requesting logical channel index (0–31).
10	MRF	Maximum receive frame length violation—This interrupt occurs in HDLC mode when more than MFLR number bytes are received. As soon as MFLR is exceeded, this interrupt is generated and the remainder of the frame is discarded. At this point the receive buffer is not closed and the reception process continues. The receive buffer is closed upon detecting a flag. The length field written to this buffer descriptor is the entire number of bytes received between the two flags. MRF interrupts are not generated in transparent mode. **Note**: The MRF interrupt is generated directly when the MFLR value is a multiple of 4 bytes. The checking of this is done on a long-word boundary whenever the SDMA transfers 32 bits to memory. If MFLR is not aligned to 4x bytes, this interrupt may be 1- to 3-byte timings late for this channel. In any case, the violation can be checked to any number of bytes. The last entry in the data buffer is always a full long word.
11	UN	Tx no data 0 = No UN event has occurred. 1 = There is no valid data to send to the transmitter. The transmitter sends an abort indication consisting of 16 consecutive 1's and then sends idles or flags according to the protocol and the channel mode register setting. This error occurs when a transmit channel has no data buffer ready for a multibuffer transmission already in progress. Transmission of a frame is a continuous bitstream without gaps or interruption. When a buffer is not ready in the middle of this sequence, it is an error situation. This can be viewed as channel underrun. The transmitter for this channel is stopped. See Section 6.3, "Restarting the Transmitter," for recovery information.
12	RXF	Rx frame 0 = No RXF event has occurred. 1 = A complete HDLC frame is received. Data is stored in external memory and the buffer descriptor is updated. If during frame reception an abort sequence of at least seven 1's is detected, the buffer is closed and both RXB and RXF are reported along with the AB in the buffer descriptor. As a result of end-of-frame, the global frame counter GRFCNT is decremented for interrupt generation. This counter is decremented on RXF only, regardless if the RXF was caused by correct closing or due to an error. RXF interrupts are not generated in transparent mode.
13	BSY	Busy 0 = No BSY event has occurred. 1 = A frame was received but was discarded due to lack of buffers. This can be viewed as channel overrun. After a busy condition, the receiver for this channel is disabled and no more data is transferred to memory. Receiver restart is described in Section 6.4, "Restarting the Receiver."

Table 4-2. Interrupt Table Entry Field Descriptions (Continued)

Field	Name	Description
14	TXB	Tx buffer 0 = No TXB event has occurred. 1 = A buffer has been completely transmitted. This bit is set (and an interrupt request is generated) as soon as the programmed number of PAD characters (or the closing flag, for PAD = 0) is written to the SCC's transmit FIFO. The number of PAD characters determines the timing of the TXB interrupt in relation to the closing flag sent out at TXD. See Chapter 5, “Buffer Descriptors,” for an explanation of PAD characters and their use.
15	RXB	Rx buffer 0 = No RXB event has occurred. 1 = A buffer has been received on this channel.

4.4 Channel Interrupt Processing Flow

Figure 4-5 illustrates the flow of a channel interrupt. Note that this does not describe the processing of the global interrupts GUN and GOV.

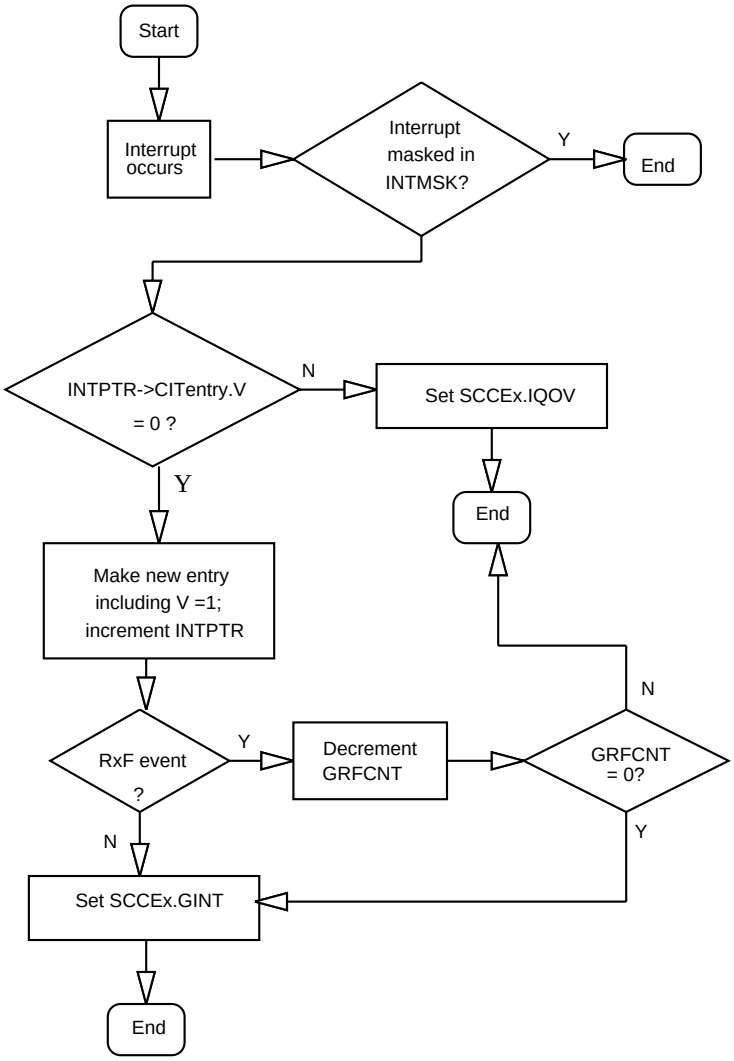


Figure 4-5. Channel Interrupt Flow

Chapter 5
Buffer Descriptors

QMC buffer descriptors are located within 64 Kbytes in external memory; see Figure 2-2. Each buffer descriptor contains key information about the buffer it defines.

The first two sections describe the contents of the receive and transmit buffer descriptors for the QMC protocol.

The third section discusses the placement of QMC and non-QMC buffer descriptors in internal and external memory.

5.1 Receive Buffer Descriptor

Figure 5-1 shows a receive buffer descriptor.

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

Notes: For the 68360, the bit numbering is reversed. See Appendix A for more information.
Entries in boldface must be initialized by the user.

Figure 5-1. Receive Buffer Descriptor (RxBD)

Table 5-1 describes the individual fields of a receive buffer descriptor. Boldfaced entries must be initialized by the user.

Table 5-1. Receive Buffer Descriptor (RxBD) Field Descriptions

Field	Name	Description
0	**E**	Empty 0 The data buffer associated with this buffer descriptor has been filled with received data, or data reception has been aborted due to an error. The core is free to examine or write to any field of this RxBD. The CPM does not use this buffer descriptor again while the empty bit remains zero. 1 The data buffer associated with this buffer descriptor is empty, or reception is currently in progress. This RxBD and its associated receive buffer are owned by the CPM. If E = 1, the CPU core should not write to any field of this RxBD.

Table 5-1. Receive Buffer Descriptor (RxBD) Field Descriptions (Continued)

Field	Name	Description
1	—	—
2	**W**	Wrap (final buffer descriptor in table) 0 This is not the last buffer descriptor in the RxBD table. 1 This is the last buffer descriptor in the RxBD table. After this buffer is used, the CPM receives incoming data into the first buffer descriptor in the table (the buffer descriptor pointed to by RBASE). The number of RxBDs in this table is programmable and is determined only by the wrap bit and by the space constraints of the dual-ported RAM.
3	**I**	Interrupt 0 The RXB bit will not be set after this buffer has been used, but RXF operation remains unaffected. 1 The RXB bit (and/or the RXF bit in HDLC mode) of the interrupt table entry will be set when this buffer has been used by the HDLC controller. These two bits may cause interrupts (if enabled).
4	L	Last-in-frame (HDLC mode only)—The HDLC controller sets L when this buffer is the last in a frame. This implies the receipt of a closing flag or reception of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 This buffer is not the last in a frame. 1 This buffer is the last in a frame.
5	F	First-in-frame—The controller sets this bit when this buffer is the first in a frame. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
6	**CM**	Continuous Mode 0 Normal operation. 1 The empty bit is not cleared by the CPM after this buffer descriptor is closed, allowing the associated data buffer to be overwritten automatically when the CPM next accesses this buffer descriptor. The empty bit is not cleared if an error occurs during reception. The user must terminate continuous mode by clearing this bit.
7	—	—
8	UB	User bit—The CPM never touches, sets, or clears this user-defined bit. The user determines how this bit is used. For example, it can be used to signal between higher level protocols whether a buffer has been processed by the CPU.
9	—	—
10	LG	Rx frame length violation (HDLC mode only)—A frame length greater than the maximum value was received in this channel. Only the maximum-allowed number of bytes, MFLR rounded to the nearest higher longword alignment, are written to the data buffer. This event is recognized as soon as the MFLR value is exceeded when data is long-word-aligned. When data is not long-word-aligned, this interrupt occurs when the SDMA writes 32 bits to memory. The worst-case latency from MFLR violation until detected is 3-byte timings for this channel. When MFLR violation is detected, the receiver is still receiving even though the data is discarded. The buffer is closed when a flag is detected, and this is considered to be the closing flag for this buffer. At this point, LG = 1 and an interrupt may be generated. The length field for this buffer includes everything between the opening flag and this last identified flag.

Table 5-1. Receive Buffer Descriptor (RxBD) Field Descriptions (Continued)

Field	Name	Description
11	NO	Rx non-octet-aligned frame (HDLC mode only)—A frame that contained a number of bits not exactly divisible by eight was received. NO = 1 for any type of nonalignment regardless of frame length. The shortest frame that can be detected is of type Flag-Bit-Flag. This causes the buffer to be closed with the NO error indicated. Figure 5-2 shows how the non-octet alignment is reported and where data can be found.
12	AB	Rx abort sequence—A minimum of seven consecutive ones was received during frame reception. Abort is not detected between frames. The sequence ...closing-flag, data, CRC, flag, AB, flag, data, opening-flag... does not cause an abort error. If the abort is long enough to be an idle, an idle line interrupt may be generated. An abort within the frame is not reported by a unique interrupt but rather with an RXF interrupt; the user has to examine the buffer descriptor.
13	CR	Rx CRC error—This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.
14–15	—	—
16–31	DL	Data length—the number of octets written by the CPM into this buffer descriptor's data buffer. It is written by the CPM once when the buffer descriptor is closed. When this buffer descriptor is the last buffer descriptor of a frame (L = 1), the data length equals the total number of octets in the frame (including the two- or four-byte CRC). Note: The amount of memory allocated for this buffer should be greater than or equal to the contents of the maximum receive buffer length register (MRBLR + 4).
32–63	**RxBP**	Rx buffer pointer—The receive buffer pointer, which always points to the first location of the associated data buffer, may reside in either internal or external memory. The Rx buffer pointer must be divisible by 4.

Figure 5-2 shows how non-octet alignment is reported and how data is stored. The two diagrams on the left show the reception of a single-buffer, 12-byte frame including the CRC. In the top case, the reception is correctly octet-aligned and the frame length indicates 12 bytes.

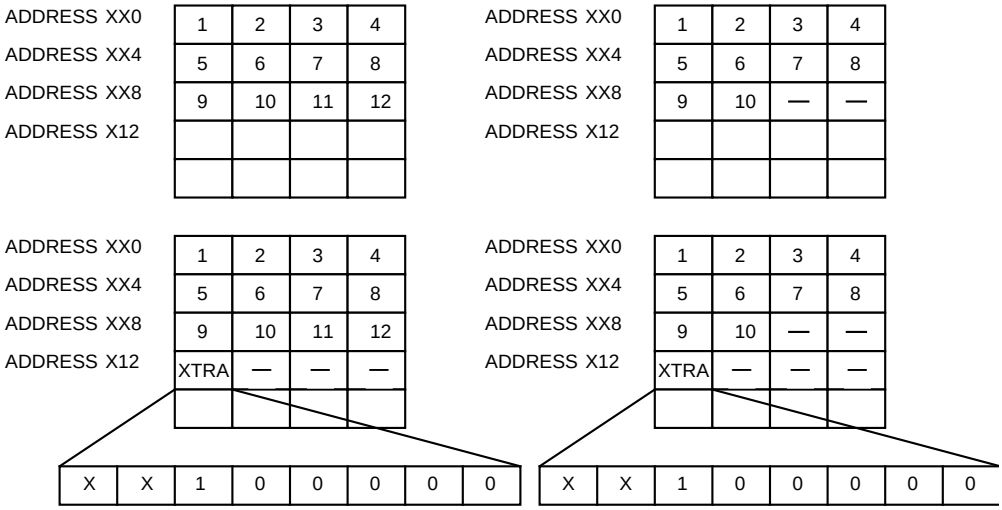


Figure 5-2. Nonoctet Alignment Data

In the bottom case, two more bits are received. The frame length is now 13 bytes, and the address positions X13 through X15 point to invalid data. Address position X12 contains information about the non-octet alignment. Valid information is written starting at the MSB position, shown as ‘x’ in the diagram. Starting from the LSB position, zeros are filled in followed by a ‘1’ immediately preceding the valid data.

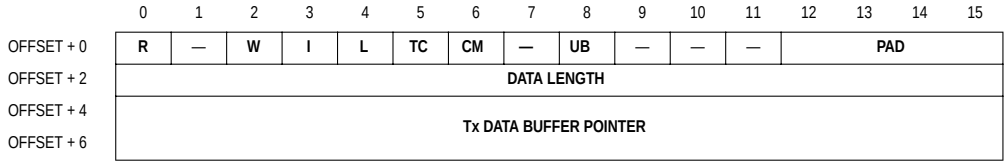
The two diagrams on the right show how the data and the extra information is stored for a frame length that is not a multiple of 4 bytes. The additional information is always on a long-word boundary. In the top case the frame length is 10 bytes and in the bottom case the length is 11 bytes.

For a minimum frame consisting of ‘flag, 1 bit, flag’ the frame length is 1. The only valid entry is at address XX0 with content of x1000000.

To accommodate the extra long word that may be written at the end of a frame, it is recommended to reserve MRBLR + 4 bytes for each buffer area.

5.2 Transmit Buffer Descriptor

Figure 5-3 shows the transmit buffer descriptor.



Notes: Entries in boldface must be initialized by the user.
For the 68360, the bit numbering is reversed. See Appendix A for more information.

Figure 5-3. Transmit Buffer Descriptor (TxBD)

Table 5-2 describes the individual fields of a transmit buffer descriptor. Boldfaced entries must be initialized by the user.

Table 5-2. Transmit Buffer Descriptor (TxBD) Field Descriptions

Field	Name	Description
0	**R**	Ready 0 The data buffer associated with this buffer descriptor is not ready for transmission. The user can manipulate this buffer descriptor or its associated data buffer. The CPM clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is being transmitted. If R = 1, the user cannot write to fields of this buffer descriptor.
1	—	—
2	**W**	Wrap (final buffer descriptor in table) 0 This is not the last buffer descriptor in the TxBD table. 1 This is the last descriptor in the Tx buffer descriptor table. After this buffer is used, the CPM transmits data from the first buffer descriptor in the table (the buffer descriptor pointed to by TBASE). The number of TxBDs in this table is programmable and is determined only by the wrap bit and the overall space constraints of the dual-ported RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 TXB in the circular interrupt table entry is set when the controller services this buffer. This bit can cause an interrupt (if enabled).
4	**L**	Last 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
5	**TC**	Tx CRC (HDLC mode only). This bit is valid only when L = 1; otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used for testing purposes to send an erroneous CRC after the data. 1 Transmit the CRC sequence after the last data byte.

Table 5-2. Transmit Buffer Descriptor (TxBD) Field Descriptions (Continued)

Field	Name	Description
6	**CM**	Continuous mode 0 Normal operation. 1 The R bit is not cleared by the CPM after this buffer descriptor is closed, allowing the associated data buffer to be retransmitted automatically when the CPM next accesses this buffer descriptor.
7	—	—
8	**UB**	User bit—The CPM never touches, sets, or clears this user-defined bit. The user determines how this bit is used. For example, it can be used to signal between higher-level protocols whether a buffer has been processed by the CPU.
9–11	—	—
12–15	**PAD**	Padding bits—These four bits indicate the number of PAD characters (0x7E or 0xFF depending on IDLM mode in the CHAMR register) that the transmitter sends after the closing flag. The transmitter issues a TXB interrupt only after sending the programmed value of pads to the Tx FIFO. The user can use the PAD value to guarantee that a TXB interrupt occurs after the closing flag has been sent on the TXD line. PAD = 0 means the TXB interrupt is issued immediately after sending the closing flag to the Tx FIFO. The number of PAD characters depends on the FIFO size and the number of time slots in use. An example explains the calculation: In SCC1 the FIFO is 32 bytes. If 16 time slots are used in the link, the resulting number of PAD characters is 32/16 = 2, to append to this buffer to ensure that the TXB interrupt is not given before the closing flag has been transmitted through the TXD line. The number of PAD characters must not exceed the NOF characters, ensuring that the closing of one buffer (the interrupt generation) occurs before the start of the next frame (clearing of R-bit). After the sequence of a closing flag followed by (PAD + 1) flag characters, a TXB interrupt will be generated; see Figure 5-4.
16–31	**DL**	Data length—The data length is the number of bytes the CPM should transmit from this buffer descriptor's data buffer. It is never modified by the CPM. This field should be greater than zero.
32–63	**TxBP**	Tx buffer pointer—The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory. This value is never modified by the CPM.

Figure 5-4 shows a TXB interrupt generated after (PAD + 1) flag characters following the closing flag. Four flags (NOF = 3) precede the next data. To set up this sequence correctly, the PAD value must not exceed NOF.

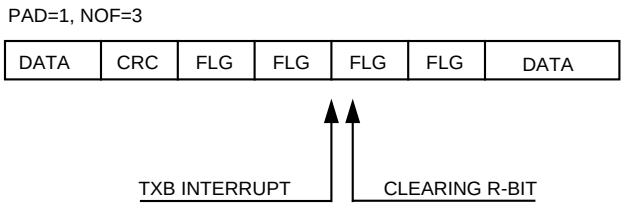


Figure 5-4. Relation between PAD and NOF

5.3 Placement of Buffer Descriptors

The internal dual-ported RAM is used to store the buffer descriptors for all non-QMC operation. This solution causes minimum loading of the external bus. When starting any operation or switching between buffers during operations, several accesses must be made by the CPM to find the actual data buffers and to read and write control and status information. This process is unseen by the user for internal accesses, and any external bus master or memory refresh control can occur uninterrupted.

5.3.1 MC68MH360 Internal Memory Structure

To support 32 channels on the MC68MH360, the entire 2-Kbyte dual-ported RAM is needed for channel-specific parameters. Each logical channel occupies 64 bytes; thus 32 channels require 2 Kbytes. No conflicts arise for QMC operation since it uses external memory for the buffer descriptors; however, buffer descriptors for other protocols must be in internal memory.

If the QMC uses all 32 channels, no space is left in the lower 2-Kbyte area; the only free areas are in the RAM pages, each 192-bytes large. Depending on the functions, channels and protocols used, some areas remain free for buffer descriptors. If a function is not enabled, its parameter RAM area may be used.

If fewer than 32 logical channels are used or if physical channels are concatenated to super channels, space is freed in the dual-ported RAM. Each logical channel creates a 64-byte hole in the dual-ported RAM that an SCC can use for buffer descriptors. QMC channels can use this area rather than external memory for buffer descriptors, reducing the load on the external bus.

If the external 64-Kbyte area overlaps the internal dual-ported RAM, external and internal buffer descriptors can be combined for the QMC. This is controlled by the show cycles setting. If split buses are used with SHEN1/SHEN0 = 00, a memory access is always made to the dual-ported RAM if the source is an internal master. For more information, refer to the description of the module configuration register in the *MC68360 Quad Integrated Communications Controller User's Manual*.

Figure 5-5 shows the internal memory map. Figure 5-6 shows the SCC parameter RAM overlap example. Figure 5-7 through Figure 5-10 give a detailed memory map for each SCC, showing parameter RAM usage for different functions.

- RAM page one is dedicated to SCC1 and for miscellaneous storage.
- RAM page two is dedicated for SCC2, RISC timers, and the SPI channel.
- RAM page three is dedicated for SCC3, SMC1, and IDMA1 operations.
- RAM page four is dedicated for SCC4, SMC2, and IDMA2 operations.

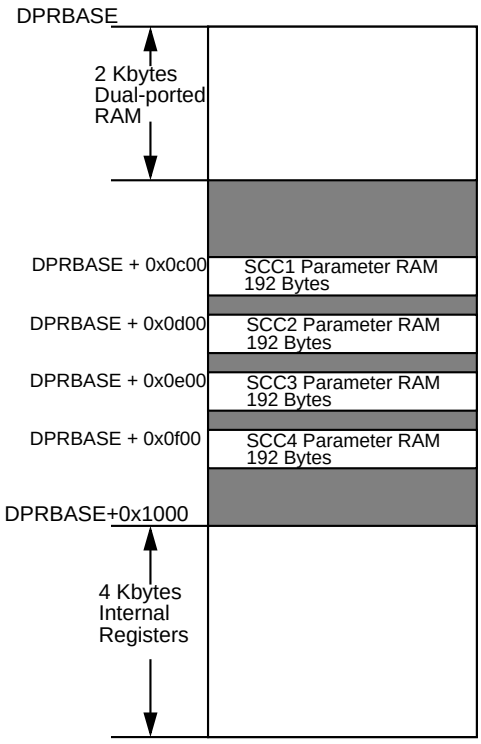


Figure 5-5. MC68MH360 Internal Memory

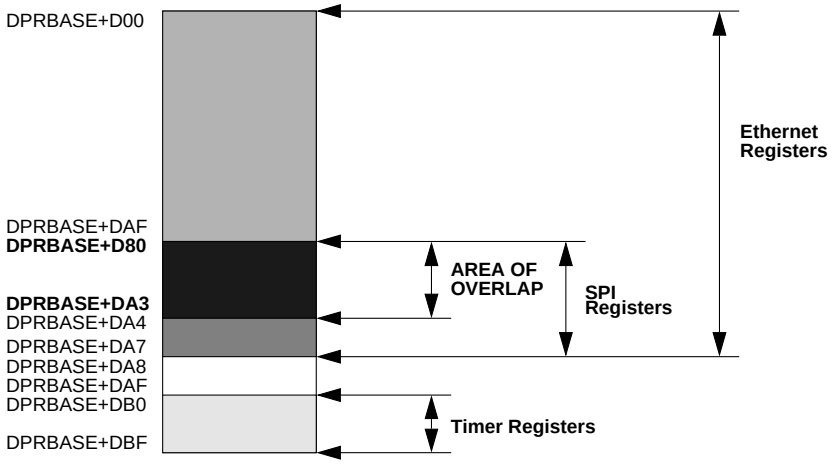


Figure 5-6. SCC2 Parameter RAM Overlap Example

NOTE

The gaps shown between RAM pages are not implemented on the MC68360MH and cannot be addressed.

Table 5-3 shows functions available for various protocols on each SCC for the MC68360.

Table 5-3. MC68360 Functions Available

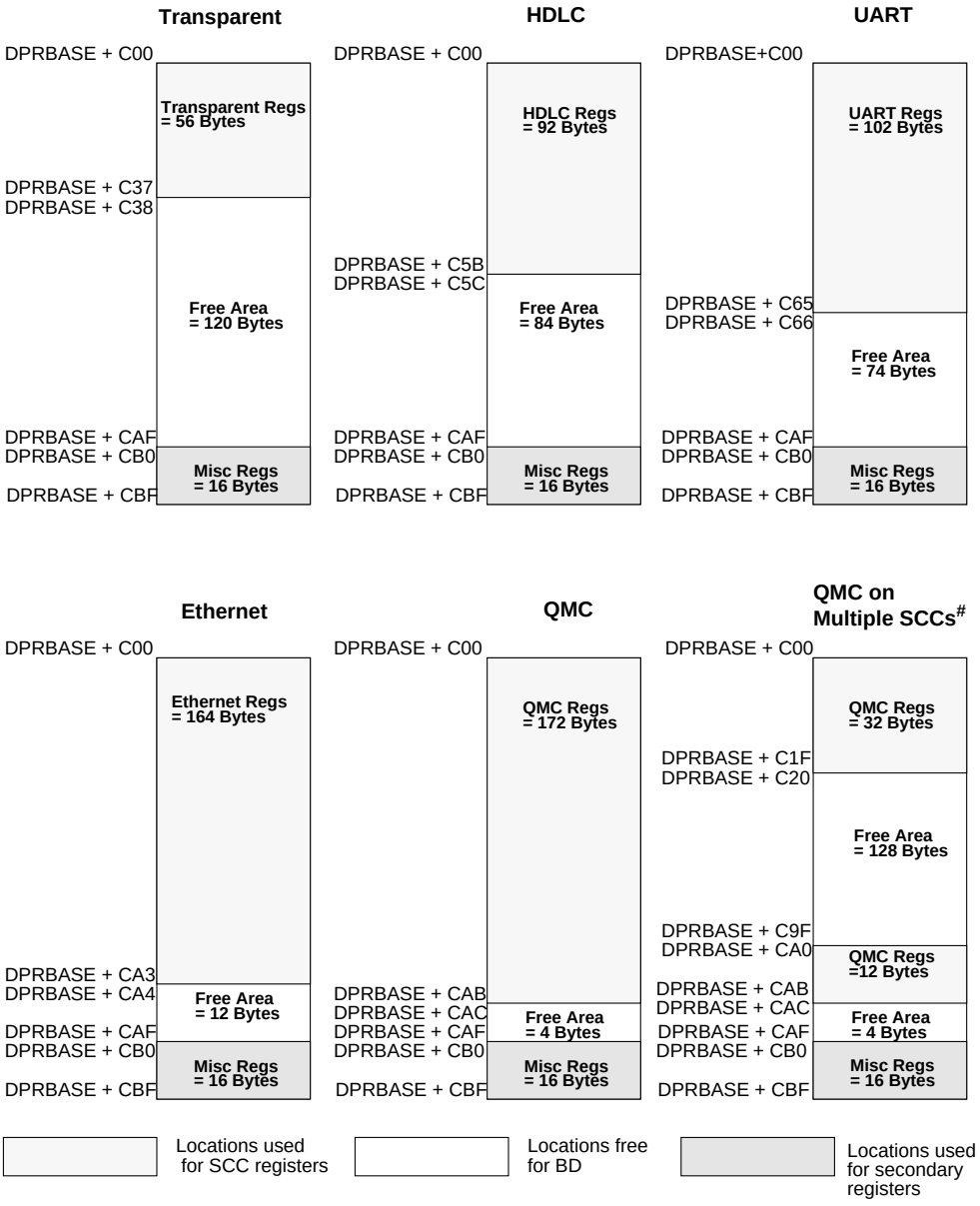
	Function Available?	Transparent	HDLC	UART	Ethernet	QMC	Shared QMC
SCC1	Yes	Misc	Misc	Misc	Misc	Misc	Misc
	No						
SCC2	Yes	SPI, Timer	SPI, Timer	SPI, Timer	Timer	Timer	Timer
	No				SPI	SPI	SPI
SCC3	Yes	IDMA1, SMC1	IDMA1, SMC1	IDMA1, SMC1			IDMA1
	No				IDMA1, SMC1	IDMA1, SMC1	SMC1
SCC4	Yes	IDMA2, SMC2	IDMA2, SMC2	IDMA2, SMC2			IDMA2
	No				IDMA2, SMC2	IDMA2, SMC2	SMC2

Figure 5-7 to Figure 5-10 show that not all the functions available to each SCC can be used simultaneously due to overlaps of the register locations stored in the parameter RAM. For example, in Figure 5-8, the SPI area overlaps the Ethernet area, which means the user can use either SPI or Ethernet on SCC2, but not both at once.

5.3.2 Parameter RAM Usage for QMC over Several SCCs

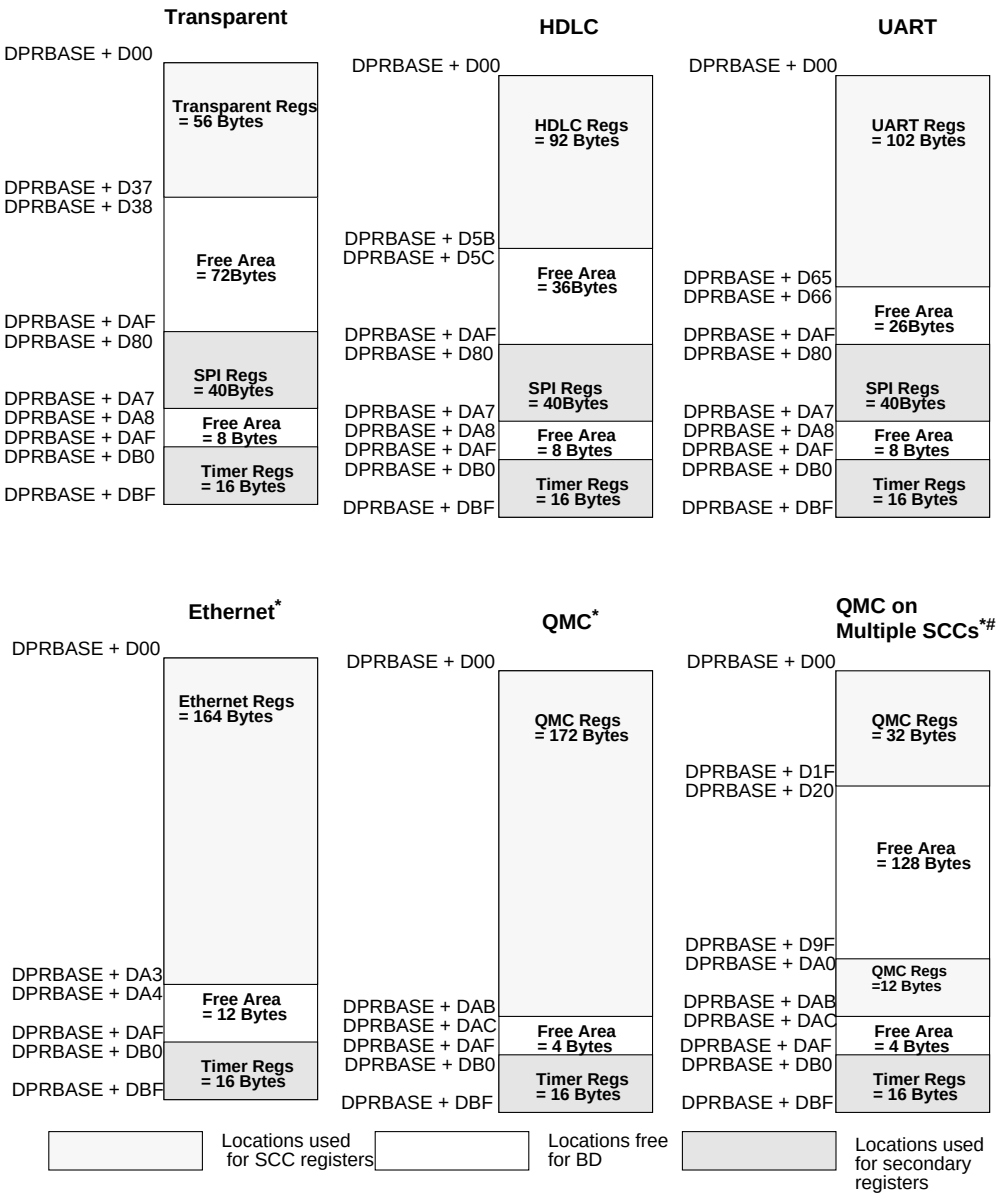
There are two possible memory configurations for operating the QMC protocol on several SCCs. For each SCC in QMC protocol, a global parameter area is used, consuming at most 170 bytes if every global area contains the routing tables. The time slot assignment tables (TSATRx and TSATTx) together occupy 128 bytes. The tables for each SCC in the parameter RAM can be duplicated, requiring 170 bytes per SCC.

Alternatively, multiple SCCs can use one set of common time slot assignment tables (TSATRx and TSATTx), as described in Section 2.1.3, “TSATRx/TSATTx Pointers and Time Slot Assignment Table.” One SCC RAM page uses 172 bytes, 44 bytes for registers and 128 bytes for the common time slot assignment tables, allowing the other SCCs to use only 44 bytes of parameter RAM for QMC protocol, freeing 128 bytes. This alternative is referred to as ‘QMC on Multiple SCCs’ in Figure 5-7 to Figure 5-15.



 # —TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM.

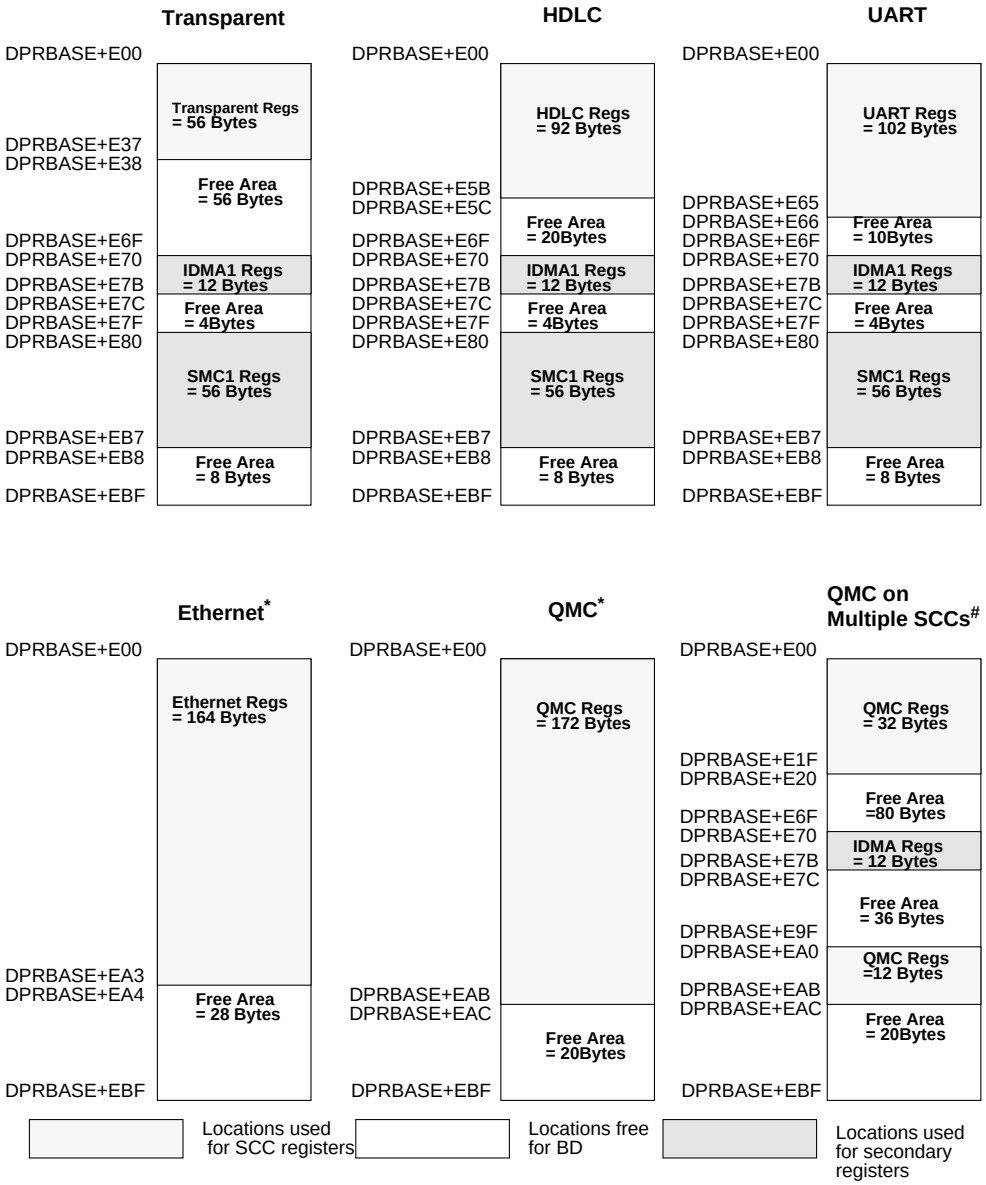
Figure 5-7. MC68MH360 SCC1 Parameter RAM Usage



#—TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM.

***—SPI not available in these configurations due to memory conflict.**

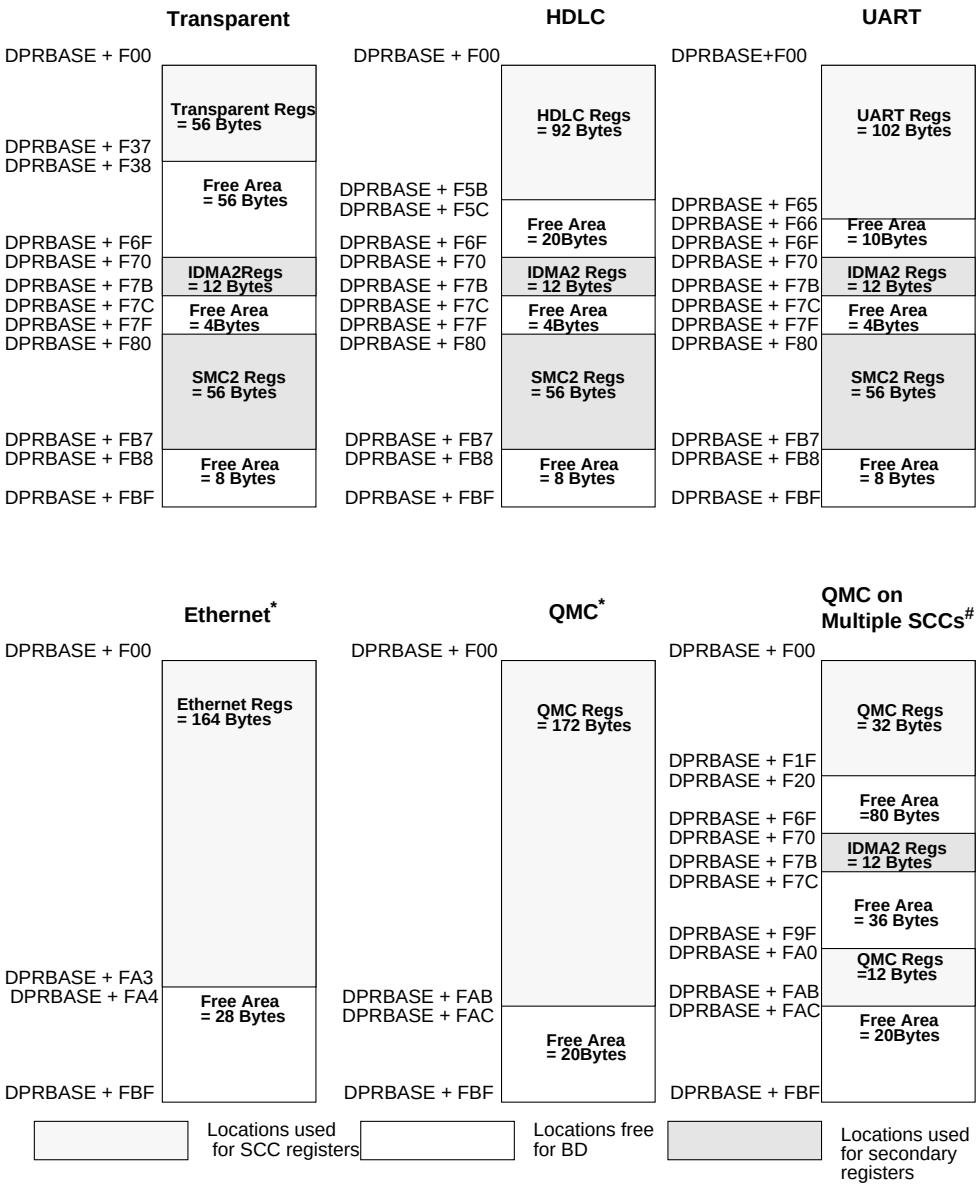
Figure 5-8. MC68MH360 SCC2 Parameter RAM Usage



 # - TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM, SMC1 is not available in this configuration due to memory conflict.

*** - SMC1 and IDMA1 not available in these configurations due to memory conflict.**

Figure 5-9. MC68MH360 SCC3 Parameter RAM Usage



#—TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM, SMC2 is not available in this configuration due to memory conflict.

***—SMC2 and IDMA2 are not available in these configurations due to memory conflict.**

Figure 5-10. MC68MH360 SCC4 Parameter RAM Usage

5.3.3 MPC860MH Internal Memory Structure

Figure 5-11 shows the internal memory structure of the MPC860MH. To support 32 channels on the MP860, only 2-Kbyte dual-ported RAM is needed for channel-specific parameters. Each logical channel occupies 64 bytes; thus 32 channels require 2 Kbytes, leaving 2 Kbytes free in the dual-ported RAM for buffer descriptors for other protocols.

To support 64 channels on the MPC860, 4-Kbyte dual-ported RAM is required for channel-specific parameters. Each logical channel occupies 64 bytes, requiring 4 Kbytes for 64 channels.

Non-QMC protocol implementations may be constrained by these memory requirements since their buffer descriptors need to use internal memory space.

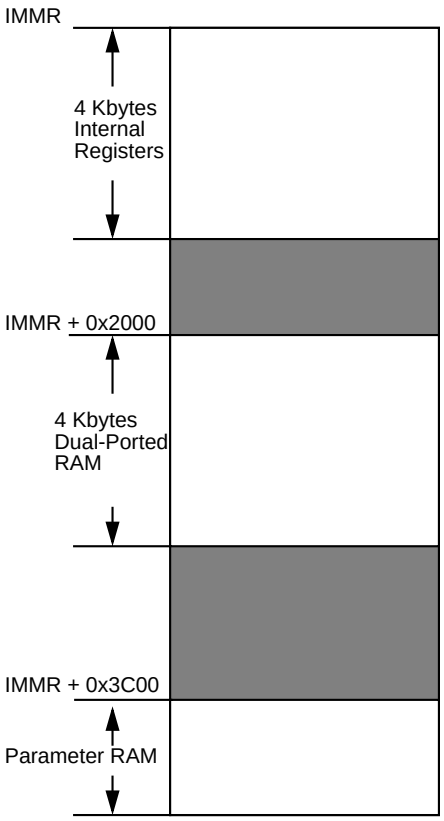


Figure 5-11. MPC860MH Internal Memory

If the QMC operates with a full 64 channels, no space is left in the lower 4-Kbyte area. In this case, the only free areas are in the RAM pages, each 256-bytes large. Depending on the functions, channels and protocols used, some areas remain free for buffer descriptors. Also, if a particular function is not enabled, its parameter RAM area may also be used.

If fewer than 64 logical channels are used or if physical channels are concatenated to super channels, space is freed in the dual-ported RAM. Each unused logical channel creates a 64-byte hole in the dual-ported RAM. This area is free for buffer descriptors for any SCC. QMC channels can also use this space instead of external memory for buffer descriptors, reducing the load on the external bus.

Figure 5-11 shows the internal memory map for the MPC860MH. Figure 5-12 to Figure 5-15 show a more detailed memory map for each SCC, showing the parameter RAM usage for different functions.

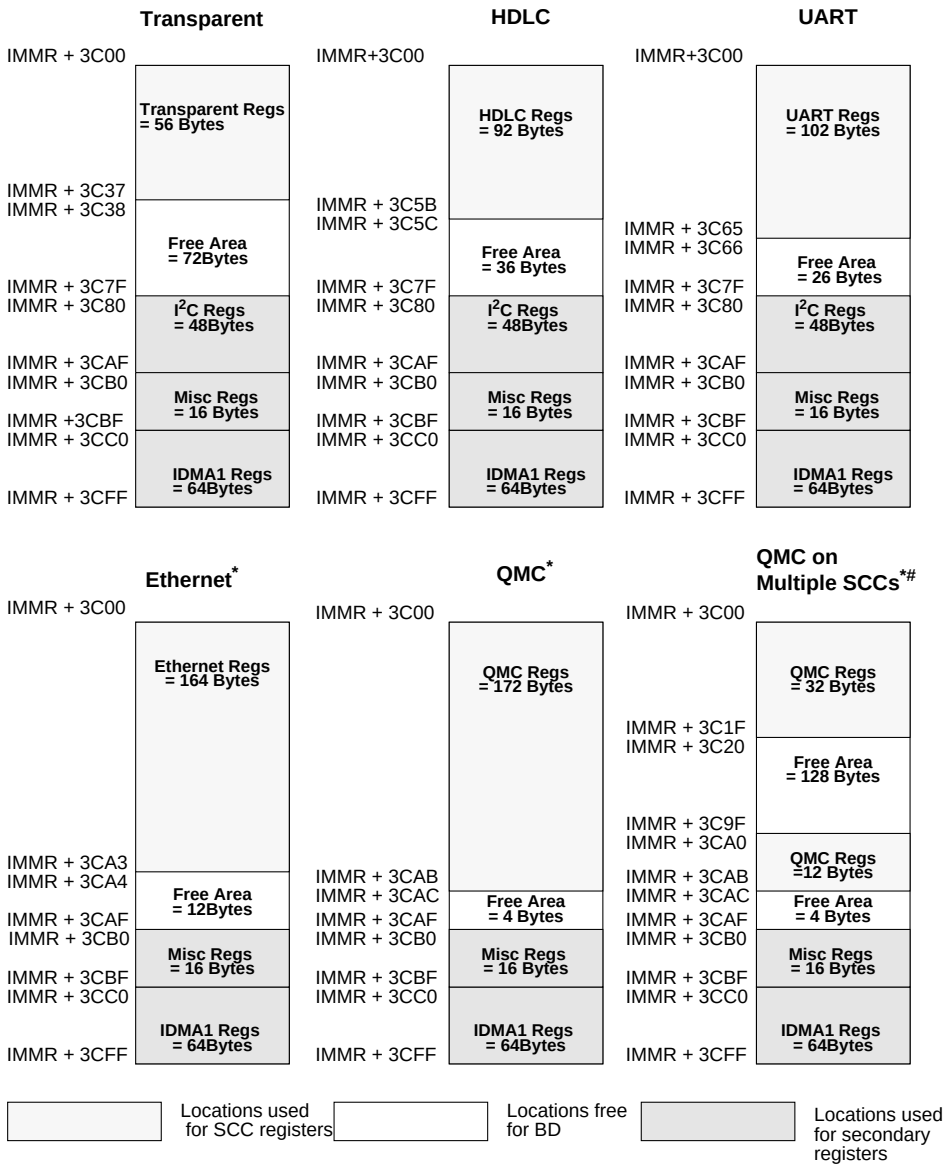
- RAM page one is dedicated to SCC1, I^2C, IDMA1 and for miscellaneous storage.
- RAM page two is dedicated for SCC2, SPI, IDMA2 and RISC timers
- RAM page three is dedicated for SCC3, SMC1 and DSP1 operations.
- RAM page four is dedicated for SCC4, SMC2 and DSP2 operations.

Table 5-4 shows the functions available for various protocols on each SCC for the MPC860MH.

Table 5-4. MPC860MH Functions Available

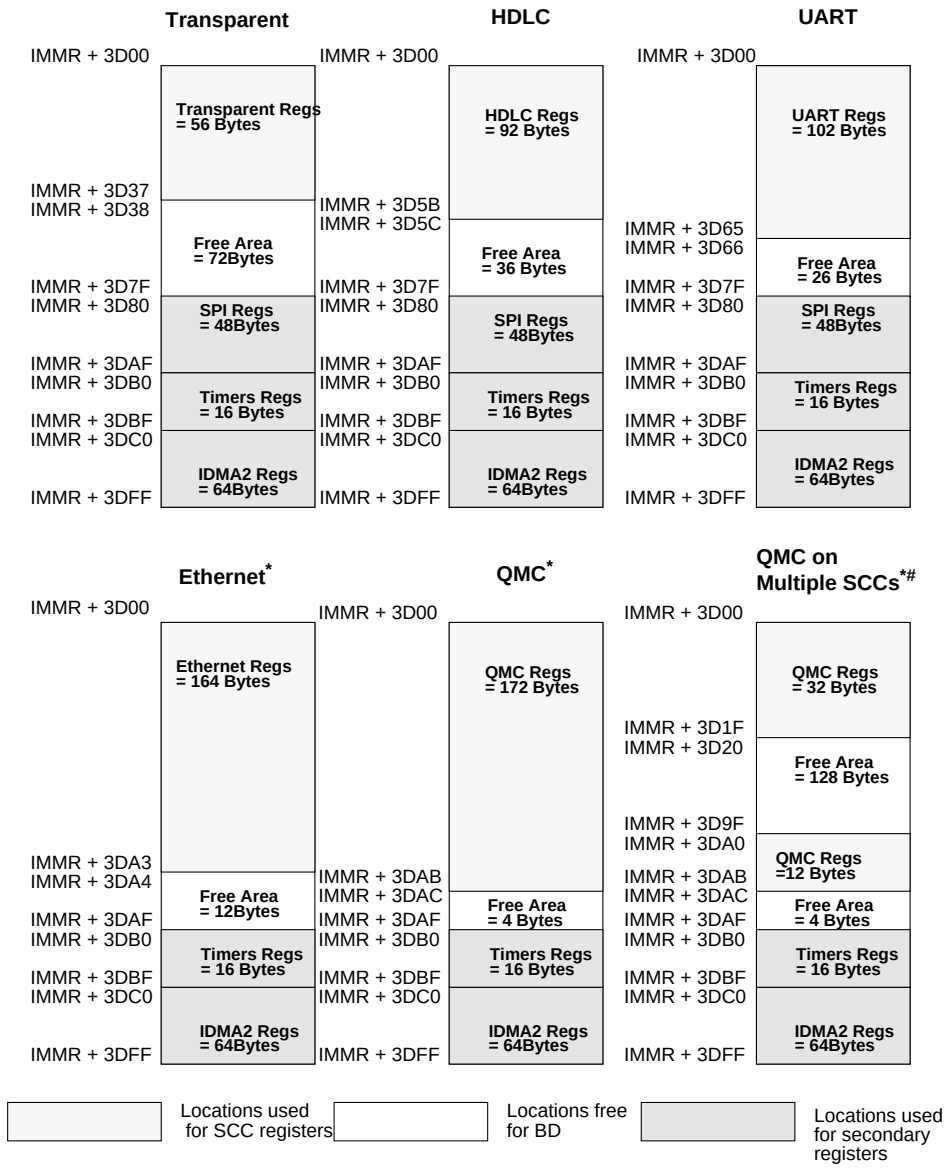
	Function Available?	Transparent	HDLC	UART	Ethernet	QMC	Shared QMC
SCC1	Yes	Misc, I^2C, IDMA1	Misc, I^2C, IDMA1	Misc, I^2C, IDMA1	Misc, IDMA1	Misc, IDMA1	Misc, IDMA1
	No				I^2C	I^2C	I^2C
SCC2	Yes	SPI, Timer, IDMA2	SPI, Timer, IDMA2	SPI, Timer, IDMA2	Timer, IDMA2	Timer, IDMA2	Timer, IDMA2
	No				SPI	SPI	SPI
SCC3	Yes	SMC1, DSP1	SMC1, DSP1	SMC1, DSP1	DSP1	DSP1	DSP1
	No				SMC1	SMC1	SMC1
SCC4	Yes	SMC2, DSP2	SMC2, DSP2	SMC2, DSP2	DSP2	DSP2	DSP2
	No				SMC2	SMC2	SMC2

Figure 5-12 to Figure 5-15 show that not all the functions available on each SCC can be used simultaneously due to overlaps of the register locations stored in the parameter RAM.



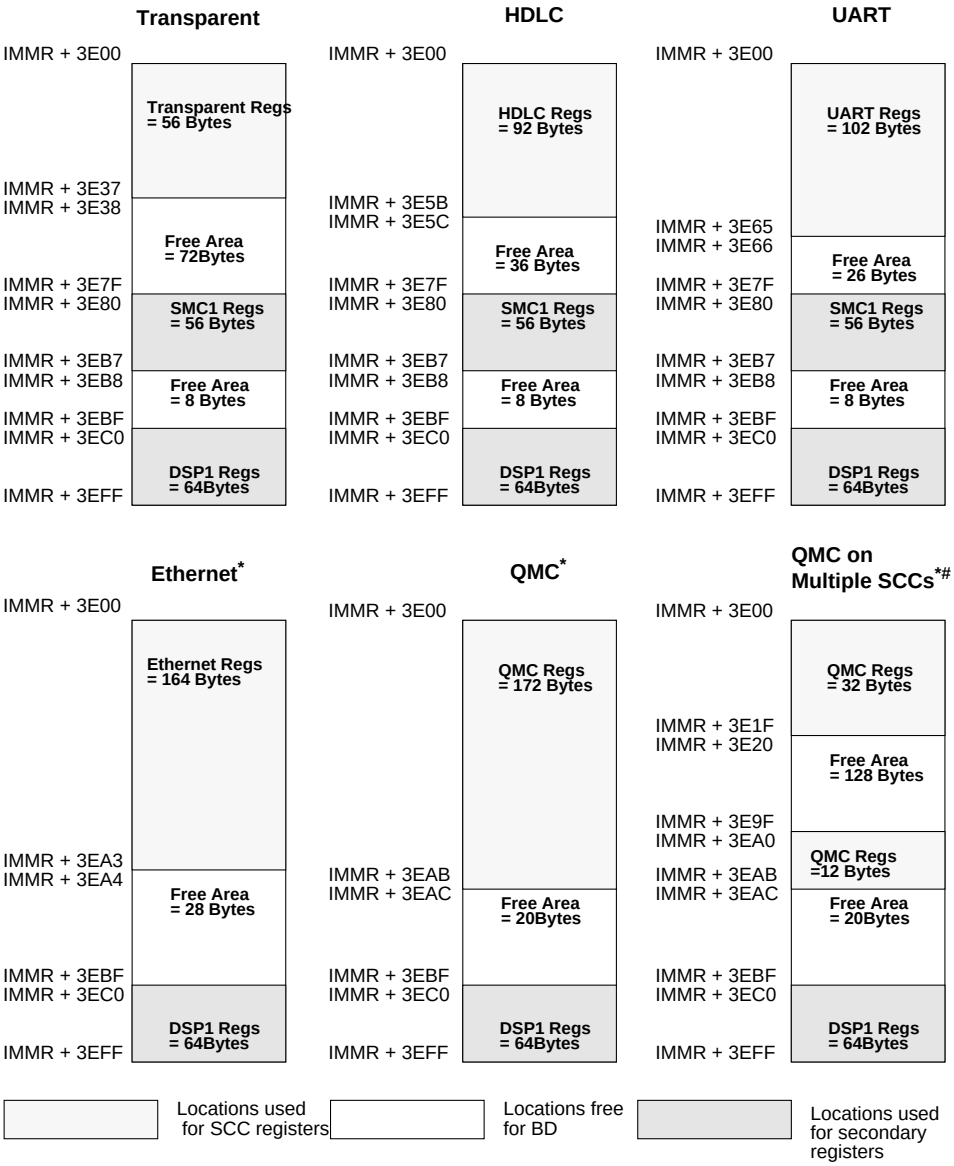
#—TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM.
***—I²C is not available in these configurations due to memory conflict.**

Figure 5-12. MPC860MH SCC1 Parameter RAM Usage



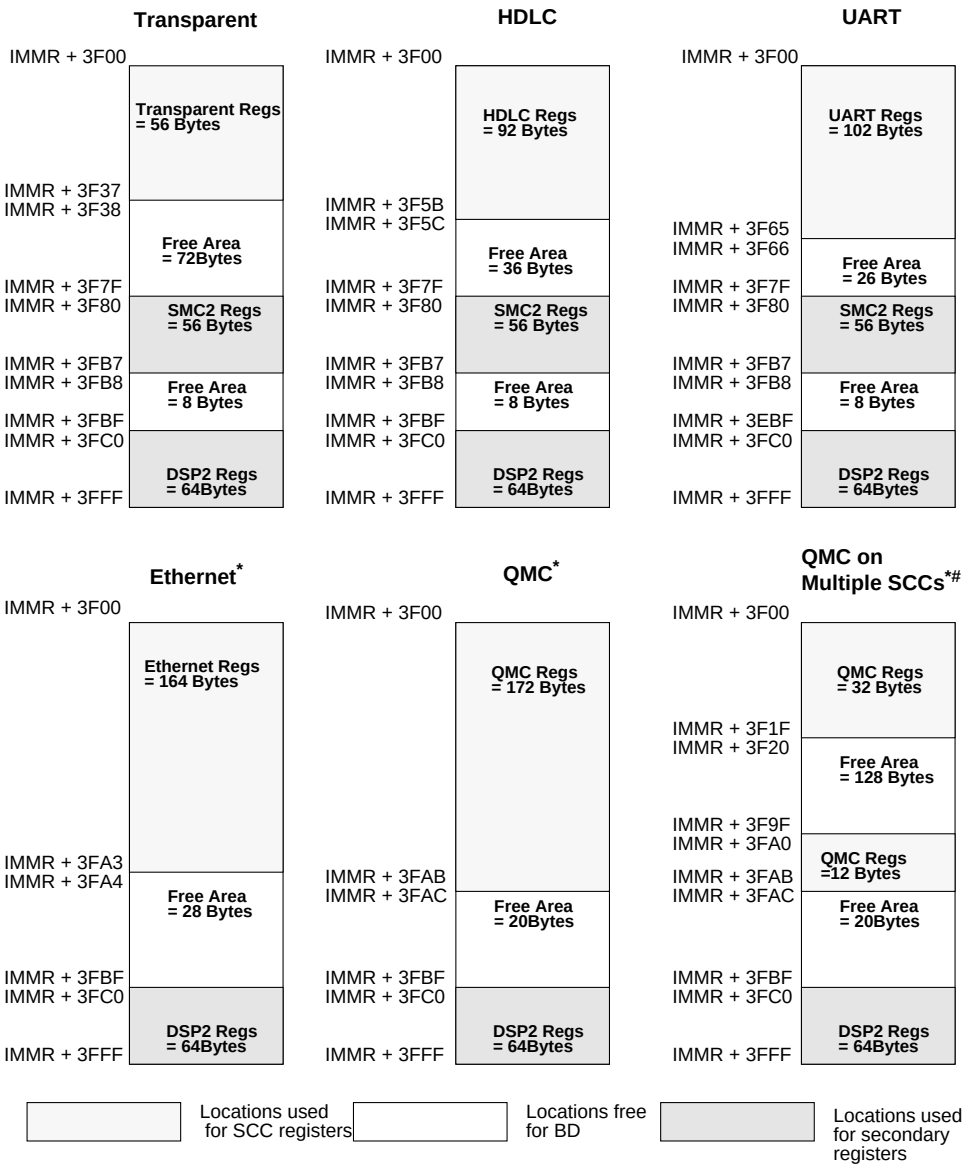
#—TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM.
***—SPI is not available in these configurations due to memory conflict.**

Figure 5-13. MPC860MH SCC2 Parameter RAM Usage



#—TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM.
***—SMC1 is not available in these configurations due to memory conflict.**

Figure 5-14. MPC860MH SCC3 Parameter RAM Usage



#—TSATRx & TSATTx are mapped to an alternative SCC's parameter RAM.
*** —SMC2 is not available in these configurations due to memory conflict.**

Figure 5-15. MPC860MH SCC4 Parameter RAM Usage

5.3.4 MC68MH360 Configured for QMC and Ethernet

Certain difficulties may arise when QMC and Ethernet are used together.

In a 25-MHz system, Ethernet can work together with 16 QMC channels. In this case, a careful use of logical channels can free a 1-Kbyte space in the parameter area for up to 128 buffer descriptors. For more information, see Section 5.3.2, “Parameter RAM Usage for QMC over Several SCCs.”

For 32-channel QMC operation coupled with Ethernet, a 33-MHz part is required. In addition, Ethernet must operate on SCC1, and the QMC protocol must be divided over SCC3 and SCC4 to use the combined FIFO size. This leaves SCC4’s parameter RAM page unused as the largest single open space for the SCC1 Ethernet, resulting in 24 buffer descriptors for transmission and reception in a single block. See Chapter 2, “QMC Memory Organization,” for more information.

Chapter 6
QMC Initialization

This section describes the essential steps to initialize QMC after a hard reset. Section 6.1, “Initialization Steps,” discusses the steps required to initialize the QMC protocol, and Section 6.2, “68MH360 T1 Example,” provides example code.

6.1 Initialization Steps

This section describes the steps required to initialize the QMC protocol.

Step 1: Initialize the SIMODE (serial interface mode) register. The SIMODE register is defined on page 7-78 of the MC68360 User’s Manual, and page 16-114 of the MPC860 user’s manual. Table 6-1 shows the transmit buffer descriptor field descriptions.

Table 6-1. Transmit Buffer Descriptor Field Descriptions

Name	No. of Bits	Description	Setting
SMCx	1	Connect to TDM or NMSI	X
SMCxCS	3	Specify clock source	X
SDMx	2	Normal, echo, or loopback mode	00
RFSDx	2	Receive frame sync delay	System-specific
DSCx	1	Double-speed clock (GCI)	System-specific
CRTx	1	Common transmit and receive sync & clk	System-specific
STZx	1	Set L1TXDx to until serial clks	0
CEx	1	Clock edge for xmit	System-specific
FEx	1	Frame sync edge	System-specific
GMx	1	Grant mode support	0
TFSDx	2	Transmit frame sync delay	System-specific

Since the SIMODE register defaults to 0x0000_0000, a typical application may set these bits as follows:

```
SIMODE.CRTa = 1;            /* common syncs & clocks */
SIMODE.RFSDa = 1;           /* receive frame sync = 1 clock delay */
SIMODE.TFSDa = 1;           /* transmit frame sync = 1 clock delay */
```

Step 2: Initialize the SICR (SI clock route) register. The SICR register is defined on page 7-86 of the MC68360 User's Manual and page 16-121 of the MPC860 User's Manual. Table 6-2 shows the SICR bit settings.

Table 6-2. SICR Bit Settings

Name	No. of Bits	Description	Setting
GRx	1	Support SCCx grant mode	0
SCx	1	Connect SCCx to TDM or NMSI	1
RxCSx	3	Connect SCCx receive to a clock	X - as SCx = 1
TxCSx	3	Connect SCCx transmit to a clock	X - as SCx = 1

As the SICR register defaults to 0x0000_0000, a typical application may set these bits as follows:

```
SICR.SC1 = 1;               /* SCC1 is TDM */
```

Step 3: Configure port A for TDMa and/or TDMb signals, L1TXDx, L1RXDx, L1TCLKx, and L1RCLKx. For more information on port A, see page 7-358 of the MC68360 User's Manual and page 16-455 of the MPC860 User's Manual.

The following setting enables TDMa and TDMb, and selects both L1TCLKx and L1RCLKx pins. Note that only L1RCLKx is required if common clocking is selected by the CRTx bit in the SIMODE register.

```
PAPAR = 0xA5F0;             /* init port A pin assignment register */
PADIR = 0x00F0;             /* init port A data direction register */
```

Step 4: Configure port B for TDMa and/or TDMb signals, L1CLKOx and L1ST1, 2, 3, and/or 4. For more information on port B, see page 7-363 of the MC68360 User's Manual and page 16-460 of the MPC860 User's Manual.

The following setting enables both L1CLKOx and all L1STx strobes. Note that the L1STx functions are repeated on port C and should only be configured on one port.

```
PBPAR = 0xFC00;             /* init port B pin assignment register */
PBDIR = 0x0C00;             /* init port B data direction register */
```

Step 5: Configure port C for TDMa and/or TDMb signals L1ST1, 2, 3, and/or 4; L1TSYNCx and/or L1RSYNCx. For more information on port C, see page 7-365 of the MC68360 User's Manual and page 16-465 of the MPC860 User's Manual.

The following setting enables L1RSYNCx, L1TSYNCx, and all L1STx strobes. Note that if common clocking is used, selected by the CRTx bit in the SI MODE register, only L1RSYNCx is required. Note that the L1STx functions are repeated on Port B and should only be configured on one port.

```
PCPAR = 0x0F0F;              /* init port C pin assignment register */
PCDIR = 0x000F;              /* init port C data direction register */
```

Step 6: Write the values to the SI RAM locations that will route the time slots required. For more information on SI RAM programming, see page 7-72 of the MC68360 User's Manual and page 16-106 of the MPC860 User's Manual.

The following example configures every byte to be transferred to SCC1; note that the final entry in both the Tx and Rx tables has the LST bit set. Also, this example for TDMa assumes that SIGMR[RDM] is set to 0b10 (32 entries for Rx and Tx). If n is the last entry, the following applies:

```
SIRAM[0] = 0x0042;           /* init 1st receive element */
SIRAM[1] = 0x0042;           /* init 2nd receive element */
SIRAM[n-1] = 0x0043;         /* init nth receive element */
SIRAM[32] = 0x0042;          /* init 1st xmit element */
SIRAM[33] = 0x0042;          /* init 2nd xmit element */
SIRAM[34] = 0x0042;          /* init 3rd xmit element */
SIRAM[32+n] = 0x0043;        /* init nth xmit element */
```

NOTE

On the MH360, the SI RAM is mapped in a continuous 256-byte block from REGB + 700->7FF. On the 860MH, it is mapped to a 512-byte block from C00->DFF. Also on the 860MH, the SI RAM has 16-bit entries aligned to 32-bit boundaries; therefore, only half of the actual address space is valid (there are 256 valid bytes).

Step 7: Enable TDM. The TDMs are enabled via the SI global mode register, SIGMR. For more information on SIGMR programming, see page 7-77 of the MC68360 User's Manual and page 16-113 of the MPC860 User's Manual. See Table 6-3 for SIGMR bit settings.

Table 6-3. SIGMR Bit Settings

Name	Number of Bits	Description	Setting
ENB	1	Enable TDMb	System-specific
ENA	1	Enable TDMa	System-specific
RDM	2	RAM division mode	System-specific

The following example enables both TDM channels for 32 entries.

```
SIGMR = 0x0E;          /* enable TDMa, TDMb, each 32 entries, no shadow */
```

Note that SIGMR[RDM] must be 0b1x if TDMb is used even if TDMa is not enabled.

Step 8. If shadow RAM is used, the SI command register (SICMR) is used to alternate between normal and shadow RAM routings. For more information on SICMR programming, see page 7-87 of the MC68360 user's manual and page 16-122 of the MPC860 user's manual.

To enable both the Rx and Tx normal RAM area, use the following command:

```
SICMR = 0x00;           /* enable Rx and Tx normal RAM */
```

To enable both the Rx and TX shadow RAM area, use the following command:

```
SICMR = 0xF0;           /* enable Rx and Tx shadow RAM on both TDMs */
```

Change this entry dynamically to allow switching between the shadow and normal RAM.

Step 9. Initialize general SCCx mode reg high, GSMR_H (see Table 6-4). For more information on GSMR programming, see page 7-111 of the MC68360 User's Manual and page 16-148 of the MPC860 User's Manual.

Table 6-4. GSMR_H Bit Settings

Name	No. of Bits	Description	Setting
IPR	1	Infrared RX polarity, only on 860MH	X
GDE	1	Glitch detect enable	X
TCRC	2	Transparent CRC	System-specific
REVD	1	Reverse data	0
TRX	1	Transparent receiver	0
TTX	1	Transparent transmitter	0
CDP	1	CD pulse	1
CTSP	1	$\overline{\text{CTS}}$ pulse	1

Table 6-4. GSMR_H Bit Settings (Continued)

Name	No. of Bits	Description	Setting
CDS	1	CD sampling	1
CTSS	1	$\overline{\text{CTS}}$ sampling	1
TFL	1	Transmit FIFO length	0
RFW	1	Receive FIFO width	0
TXSY	1	Transmitter synchronized to the receiver	0
SYNL	2	Sync length	00b
RTSM	1	RTS mode	0
RSYN	1	Receive synchronization timing	0

A typical setting would be:

```
GSMR_H = 0x0000_0780;          /* enable pulse mode and sampling */
```

Step 10. Initialize general SCCx mode reg low, GSMR_L (see Table 6-5). For more information on GSMR programming, see page 7-111 of the MC68360 User's Manual and page 16-153 of the MPC860 User's Manual.

Table 6-5. GSMR_L Bit Settings

Name	No. of Bits	Description	Setting
SIR	1	Infrared encoding, only on 860MH	X
EDGE	2	Clock edge	00
TCI	1	Transmit clock invert	0
TSNC	2	Transmit sense	00b
RINV	1	DPLL receive input invert data	0
TINV	1	DPLL transmit input invert data	0
TPL	3	Tx preamble length	0b000
TPP	2	Tx preamble pattern	0b00
Tend	1	Transmitter frame ending	0
TDCR	2	Transmit divide clock rate	00
RDCR	2	Receive divide clock rate	00
RENC	2	Receive decoding	00
TENC	2	Transmitter decoding	00
DIAG	2	Diagnostic mode	system-specific
ENR	1	Enable receive	0
ENT	1	Enable transmit	0
MODE	4	Channel protocol mode	0b1010

Clear the ENR and ENT bits at the end of the initialization. The MODE setting for QMC mode is 0b1010.

A typical setting would be:

```
GSMR_L = 0x0000_000A;          /* enable QMC */
```

Step 11. Initialize basic global multichannel parameters as follows. See Chapter 2, “QMC Memory Organization,” for more information.

- MCBASE: (multichannel base pointer) is a pointer to a 64-Kbyte buffer descriptor table in external memory. For example:

  ```
  SCC1.MCBASE = 0x1_0000;        /* BD base located 0x1_0000 */
  ```

- INTBASE: (interrupt table base pointer) - points to the interrupt table in external memory. For example:

  ```
  SCC1.INTBASE = 0xF000;         /* interrupt table base 0xF000 */
  ```

- MRBLR: (maximum receive buffer length) - should be large (> 30) for better performance and should be a multiple of 4 bytes. This is valid for HDLC only. For example:

  ```
  SCC1.MRBLR = 60;               /* set receive buffer length to 60 */
  ```

- GRFTHR: (global receive frame threshold) - normally set to 1. For example:

  ```
  SCC1.GRFTHR = 1;               /* 1 receive frame to interrupt */
  ```

- GRFCNT: (global receive frame count) - should be initialized to the same value as GRFTHR. For example:

  ```
  SCC1.GRFCNT = 1;               /* 1 receive frame to interrupt */
  C_MASK32:CRC constant, 32-bit =0xDEBB20E3
  SCC1.C_MASK32 = 0xDEBB20E3;    /* init 32-bit CRC const */
  C_MASK16:CRC constant,16-bit=0xFOB8
  SCC1.C_MASK16 = 0xF0B8;        /* init 16-bit CRC const */
  ```

Step 12. Copy INTBASE to INTPTR (multichannel interrupt pointer). See Chapter 4, “QMC Exceptions,” for more information.

```
SCC1.INTPTR = SCC1.INTBASE;          /* init intptr */
```

Step 13. Initialize the time slot assignment tables, TSATTx and TSATRx. Each valid entry should have the V bit set. Clear the W bit in all entries except the last entry in the table. The 'mask' bits determine which bits of the time slot are processed by the CPM–normally set to 0xFF to process all 8 bits. The 6-bit CP field holds the most-significant bits of the starting address of the channel-specific parameter area. For the MH360, the most-significant bit must be zero. The 6 least-significant bits are always cleared. See Section 2.1.3, "TSATRx/ TSATTx Pointers and Time Slot Assignment Table," for more information. The following is example pseudocode for TSA table programming:

```
for (x = 0; x < time slots; x++)
{
      SCC1.TSATR[x].W = 0;          /* not last time slot */
      SCC1.TSATR[x].CP = x;         /* mark channel number */
      SCC1.TSATR[x].mask0_1 = 3;    /* no subchanneling */
      SCC1.TSATR[x].mask2_7 = 0x3F; /* no subchanneling */
      SCC1.TSATR[x].V = 1;          /* mark time slot valid */
}
      SCC1.TSATR[last].W = 1;       /* last time slot wrap */
```

Step 14. Initialize TSAT pointers (Tx_S_PTR and Rx_S_PTR), and the current time slot entry pointers, (RxPTR and TxPTR). Initialize both Tx_S_PTR and TxPTR to the first entry of the TSATx. Also initialize both Rx_S_PTR and RxPTR to the first entry of the TSARx. For common Rx and Tx time slot assignment tables, they all should point to SCC base + 20; however, they may be located anywhere within the dual-ported RAM. See Section 2.1.3, "TSATRx/TSATTx Pointers and Time Slot Assignment Table," for more information. The following is an example configuration:

```
SCC1.Tx_S_PTR = SCC1.MCBASE+0x20;/* init pointer to TSATTx table */
SCC1.TxPTR = SCC1.Tx_S_PTR;
SCC1.Rx_S_PTR = SCC1.MCBASE+0x20;/* init pointer to TSATRx table */
SCC1.RxPTR = SCC1.Rx_S_PTR;
```

Step 15. Initialize multichannel controller state QMC-STATE to 0x8000.

```
pdpr->SCC1.QMC_STATE = 0x8000;
```

Step 16. Initialize channel-specific parameters for HDLC or transparent mode as follows. For more information on HDLC, see Section 2.4.1, “Channel-Specific HDLC Parameters,” and for transparent mode, see Section 2.4.2, “Channel-Specific Transparent Parameters.” Repeat for each of the enabled channels.

- TBASE: TxBD descriptors base address. Initialize to location of first TxBD = MCBASE+TBASE.
  RBASE: RxBD descriptors base address. Initialize to location of first RxBD = MCBASE+RBASE.

```
ch[0].RBASE = 0;              /* locate CH0 RxBDS at 0 */
ch[0].TBASE = 8;              /* locate CH0 TxBDS at 8 */
ch[1].RBASE = 0x10;           /* locate CH1 RxBDS 0x10 */
ch[1].TBASE = 0x18;           /* locate CH1 TxBDS 0x18 */
ch[2].RBASE = 0x20;           /* locate CH2 RxBDS 0x20 */
ch[2].TBASE = 0x28;           /* locate CH2 TxBDS 0x28 */
ch[3].RBASE = 0x30;           /* locate CH3 RxBDS 0x30 */
ch[3].TBASE = 0x38;           /* locate CH3 TXBDS 0x38 */
```

  Copy RBASE to RBPTR (Rx buffer descriptor pointer) and TBASE to TBPTR (Tx buffer descriptor pointer).

```
ch[x].TBPTR = ch[x].TBASE;
ch[x].RBPTR = ch[x].RBASE;
```

- TSTATE: Tx internal state. For the MH360, this is typically 0x3800_0000. For the 860MH, this is typically 0x3000_0000.

```
ch[x].TSTATE = 0x3800_0000;   /* setting for MH360 */
ch[x].TSTATE = 0x3000_0000;   /* setting for 860MH */
```

- RSTATE: Rx internal state. For the MH360, this is typically 0x3900_0000. For the 860MH, this is typically 0x3100_0000.

```
ch[x].RSTATE = 0x3900_0000;   /* setting for MH360 */
ch[x].RSTATE = 0x3100_0000;   /* setting for 860MH */
```

- ZISTATE: zero insertion should be initialized to 0x0000_0100.

```
ch[x].ZISTATE = 0x100;
```

- ZDSTATE: zero deletion machine state should be initialized to 0x1800_0080 for transparent mode or 0x0000_0080 for HDLC.

```
ch[x].ZDSTATE = 0x80;         /* set ZDZTATE for HDLC */
```

- INTMSK: channel’s interrupt mask flags. Bits should be set to enable the corresponding interrupts.

```
ch[x].INTMSK = 0xA;
```

- MFLR/MRBLR: MFLR (HDLC mode)—application-dependent.
  MRBLR (transparent mode)—must be divisible by 4 and large (>30) for better performance.

  ```
  ch[x].MFLR = 60;
  ```

- TRNSYNC: transparent synchronization, system-specific.

Step 17. Initialize RxBDs. Prepare an adequate number of receive buffers at the location addressed by RBASE. In the status word, set the E bit, set the I bit if interrupts are required and set the W bit for the last buffer descriptor. The data length is normally cleared, and the buffer pointer is set to a location in external memory. See Section 5.1, “Receive Buffer Descriptor,” for more information. Repeat for each enabled channel.

Step 18. Initialize TxBDs. Prepare an adequate number of transmit buffers at the location addressed by TBASE. In the status word, set the R bit, set the I bit if interrupts are required, and set the W bit for the last buffer descriptor. Other options are available and may be set or cleared depending on the application. The data length is written with the number of bytes to transmit, and the buffer pointer is set to a location in external memory. See Section 5.2, “Transmit Buffer Descriptor,” for more information. Repeat for each enabled channel.

Step 19. Initialize the circular interrupt table. If interrupts are required, initialize the interrupt table as explained in Chapter 4, “QMC Exceptions.” Clear the V and W bits, but make sure to set the last entry’s W bit.

Step 20. Initialize the channel mode register CHAMR (see Table 6-6). For more information see Section 2.4.1.1, “CHAMR—Channel Mode Register (HDLC),” for HDLC mode and Section 2.4.2.1, “CHAMR—Channel Mode Register (Transparent Mode),” for transparent mode.

Table 6-6. CHAMR Bit Settings

Name	Number of Bits	Description	Setting
MODE	1	0—transparent; 1—HDLC	X
RD/0	1	Transparent only: reverse data	X
1/IDLM	1	HDLC only: idle mode	X
ENT	1	Enable transmit	0
SYNC	1	Transparent only: synchronization	X
POL	1	Enable polling	0
CRC	1	HDLC only: CRC	X
NOF	4	Minimum number of flags	X

Note the ENT bit is initially cleared, but then must be set when the channel is ready to start transmitting. Similarly, the POL bit is initially cleared, but then must be set each time a buffer descriptor is enabled to transmit. Example settings are as follows:

```
ch[x].CHAMR.MODE = 1;          /* select HDLC */
ch[x].CHAMR.IDLM = 0;          /* no idles between frames */
ch[x].CHAMR.ENT = 1;           /* enable channel xmit */
ch[x].CHAMR.CRC = 1;           /* select 32-bit CRC */
ch[x].CHAMR.NOF = 7;           /* 7 flags between frames */
ch[x].CHAMR.POL = 1;           /* enable polling by RISC */
```

Step 21. Initialize the SCCE register. From reset, SCCEx will be zero requiring no initialization. However, if required, it can be cleared by writing a 1 in each of the status bits. See Section 4.1, “Global Error Events,” for more information.

```
SCCE1 = 0xF;                   /* clear all interrupts */
```

Step 22. Initialize the mask register, SCCMx. Any interrupts which are not used should be masked in the SCCM register. SCC interrupts should be enabled using the CIMR register, if required. The CIMR register is defined on page 7-381 of the MC68360 User’s Manual and page 16-483 of the MPC860 User’s Manual.

```
SCCM1 = 0xF;                   /* enable all interrupts */
CIMR.SCC1 = 1;                 /* SCC1 interrupts enabled */
```

Step 23. Enable the transmitter (ENT bit) and the receiver (ENR bit) in the general SCC mode register (GSMR).

```
GSMR_L1.ENR = 1;               /* enable receiver */
GSMR_L1.ENT = 1;               /* enable transmit */
```

6.2 68MH360 T1 Example

```
/* This is an example of transmitting and receiving on four  */
/* HDLC channels in loopback mode. */
/* Equipment : SBC360 Evaluation Board with QUICC32 */
/* (T1MH.C) */

void *const stdout = 0;            /* standard output device */
#include <string.h>               /* string functions */
#include <stdio.h>                /* I/O functions */
#define qmc1                      /* SCC1 is multichannel comm */
#include "68360.h"                /* dual-ported RAM equates */
struct dprbase *pdpr;             /* pointer to dual-ported RAM */
```

```
struct  descs {
      rxbdq recvbd0;               /* receive buffer 0 */
      txbdq xmitbd0;               /* transmit buffer 0 */
      } chbd[4];                   /* 4 sets of chan descriptors */
static char *poem[6];              /* poem area */
short linecntr;                    /* line counter */
struct intrpten {
      unsigned V:1;                /* entry valid bit */
      unsigned W:1;                /* entry wrap bit */
      unsigned NID:1;              /* not-an-idle has occurred */
      unsigned IDL:1;              /* an idle has occurred */
      unsigned :1;
      unsigned CHNMBR:5;           /* channel number */
      unsigned MRF:1;              /* maximum frame length violation */
      unsigned UN:1;               /* Tx underrun */
      unsigned RXF:1;              /* receive frame */
      unsigned BSY:1;              /* frame discarded, no buffers */
      unsigned TXB:1;              /* buffer transmitted */
      unsigned RXB:1;              /* receive buffer closed */
      }  intrpt[10];               /* interrupt table */
short recvcnt,xmitcnt,othrcnt = 0;  /* interrupt counters */

main()
{
      void SCC1esr();              /* exception service rtn */
      int *pvec;                   /* exception vector pntr */
      char vecblk = 3;             /* vector number block */
      char intlvl = 4;             /* interrupt level */
      pdpr = (struct dprbase *) (getmbar() & 0xFFFFE000);/*  init  dual-ported
      RAM ptr */
      pdpr->CICR.VBA2_VBA0 = (unsigned) (vecblk);/* vecs at vec num 0x60-7F */
      pdpr->CICR.IRL2_IRL0 = (unsigned) (intlvl);/* CPM interrupts level 4 */
      pdpr->CICR.HP4_HP0 = 0x1F;    /* no int priority change */
/* SCdP is zero from reset */
      pdpr->CICR.SCcP = 1;         /* SCC2 to SCCC position */
      pdpr->CICR.SCbP = 2;         /* SCC3 to SCCB position */
      pdpr->CICR.SCaP = 3;         /* SCC4 to SCCA position */
```

```
      pvec = (int *) (getvbr() + (((vecblk << 5) + 0x1E) * 4));/*    calculate
      vector address */
      *pvec = (int) SCC1esr;        /* init int vector */
      pdpr->TRR4 = 8;               /* init timer ref */
      pdpr->TMR4.PS = 0;            /* init prescalar */
      pdpr->TMR4.OM = 1;            /* select toggle */
      pdpr->TMR4.FRR = 1;           /* restart after ref reac */
      pdpr->TMR4.ICLK = 1;          /* master clock */
      pdpr->TGCR.RST4 = 1;          /* enable timer */
      pdpr->PAPAR |= 0x8000         /* enable TOUT4 pin */
      pdpr->PADIR |= 0x8000;        /* pin 15 is an output */
      pdpr->TRR3 = 174;             /* init timer ref */
      pdpr->TMR3.PS = 0;            /* init prescalar */
      pdpr->TMR3.OM = 0;            /* select pulse */
      pdpr->TMR3.FRR = 1;           /* restart after ref reac */
      pdpr->TMR3.ICLK = 3;          /* external clock */
      pdpr->TGCR.RST3 = 1;          /* enable timer */
      pdpr->PAPAR |= 0x3000;/* enable TOUT3 & TIN3 */
      pdpr->PADIR |= 0x2000;        /* pin 13 is an  output */
/* SDCR has been initialized by the debugger */
      pdpr->SIMODE.CRTa = 1;        /* common syncs & clocks */
      pdpr->SIMODE.RFSDa = 1;       /* receive frame sync 1 clk dly */
      pdpr->SIMODE.TFSDa = 1;       /* xmit frame sync 1 clk dly */
      pdpr->SICR.SC1 = 1;           /* SCC1 is TDM */
      pdpr->PAPAR |= 0x100;         /* enable L1RCLKa */
/* PAPAR, PADIR & PAODR are cleared at reset */
      pdpr->PCPAR = 0x800;          /* enable L1RSYNCa */
/* Port C registers are cleared at reset */
      pdpr->SIGMR = 4;              /* enable TDMa */
      pdpr->SIRAM[0] = 0x8042;      /* init 1st receive element */
      pdpr->SIRAM[1] = 0x8042;      /* init 2nd receive element */
      pdpr->SIRAM[2] = 0x8042;      /* init 3rd receive element */
      pdpr->SIRAM[3] = 0x8043;      /* init 4th receive element */
      pdpr->SIRAM[64] = 0x8042;     /* init 1st xmit element */
      pdpr->SIRAM[65] = 0x8042;     /* init 2nd xmit element */
      pdpr->SIRAM[66] = 0x8042;     /* init 3rd xmit element */
      pdpr->SIRAM[67] = 0x8043;     /* init 4th xmit element */
```

```
      pdpr->GSMR_H1.CDP = 1;        /* enable CD* pulse */
      pdpr->GSMR_H1.CTSP = 1;       /* enable CTS* pulse */
      pdpr->GSMR_H1.CDS = 1;        /* enable CDS* sampling */
      pdpr->GSMR_H1.CTSS = 1;       /* enable CTS* sampling */
/* GSMR_H1 is zero from reset */
     pdpr->GSMR_L1.MODE = 0xA;      /* select QMC mode */
/* GSMR_L1 is zero from reset */
      pdpr->SCC1.MCBASE = 0x1_0000; /* BD base located 0x1_0000 */
      pdpr->SCC1.INTBASE = 0xF000;  /* interrupt table base 0xF000 */
      pdpr->SCC1.MRBLR = 60;        /* set receive buffer length to 60 */
      pdpr->SCC1.GRFTHR = 1;        /* 1 receive frame to intrpt */
      pdpr->SCC1.GRFCNT = 1;        /* 1 receive frame to intrpt */
      pdpr->SCC1.C_MASK32 = 0xDEBB20E3;/* init 32-bit CRC const */
      pdpr->SCC1.C_MASK16 = 0xF0B8; /* init 16-bit CRC const */
      pdpr->SCC1.INTPTR = pdpr->SCC1.INTBASE;/* init intptr */
inittsatr(4);
      pdpr->SCC1.TSATR[3].W = 1;    /* last time slot */
      pdpr->SCC1.Tx_S_PTR = (short *)(pdpr->SCC1.MCBASE+0x20);/* init pntr to
      TSATTx table */
      pdpr->SCC1.TxPTR = pdpr->SCC1.Tx_S_PTR;/* init curr TSATTx pntr */
      pdpr->SCC1.Rx_S_PTR = (short *)(pdpr->SCC1.MCBASE+0x20);/* init pntr to
      TSATRx table */
      pdpr->SCC1.RxPTR = pdpr->SCC1.Rx_S_PTR;/* init curr TSATRx pntr */
      pdpr->SCC1.QMC_STATE = 0x8000;/* init QMC-STATE */
chbd = (struct descs *)((int)(pdpr->SCC1.MBASE));
      pdpr->ch[0].RBASE = 0;        /* locate CH0 RxBDS at 0 */
      pdpr->ch[0].TBASE = 8;        /* locate CH0 TxBDS at 8 */
      pdpr->ch[1].RBASE = 0x10;     /* locate CH1 RxBDS 0x10 */
      pdpr->ch[1].TBASE = 0x18;     /* locate CH1 TxBDS 0x18 */
      pdpr->ch[2].RBASE = 0x20;     /* locate CH2 RxBDS 0x20 */
      pdpr->ch[2].TBASE = 0x28;     /* locate CH2 TxBDS 0x28 */
      pdpr->ch[3].RBASE = 0x30;     /* locate CH3 RxBDS 0x30 */
      pdpr->ch[3].TBASE = 0x38;     /* locate CH3 TXBDS 0x38 */
      for (v1 = 0; v1 < 4; v1++)
      {
            pdpr->ch[v1].TSTATE = 0x3800_0000;
            pdpr->ch[v1].RSTATE = 0x3900_0000;
            pdpr->ch[v1].ZISTATE = 0x100;
```

```
            pdpr->ch[v1].ZDSTATE = 0x80;
            pdpr->ch[v1].INTMSK = 0xA;
            pdpr->ch[v1].MFLR = 60;
            pdpr->ch[v1].TBPTR = pdpr->ch[v1].TBASE;
            pdpr->ch[v1].RBPTR = pdpr->ch[v1].RBASE;
      };
      chbd[0].recvbd0.rxbdptr = (char *)((int)chbd + 0x100);/*   receive   BD0
      pointer=0x100 */
      chbd[1].recvbd0.rxbdptr = (char *)((int)chbd + 0x200);/*   receive   BD1
      pointer=0x200 */
      chbd[2].recvbd0.rxbdptr = (char *)((int)chbd + 0x300);/*   receive   BD2
      pointer=0x300 */
      chbd[3].recvbd0.rxbdptr = (char *)((int)chbd + 0x400);/*   receive   BD3
      pointer=0x400 */
      for (v1 = 0; v1 < 4; v1++)
      {
            chbd[v1].recvbd0.rxbdsac.E = 1;     /* mark receive bufs empty */
            chbd[v1].recvbd0.rxbdsac.W = 1; /*endreceive buffer descriptors */
            chbd[v1].recvbd0.rxbdsac.I = 1;     /* enable intrpts */
/* Continuous mode bits are initialized to 0 from reset */
/* Transmit data length field is initialized to 0 from */
/* reset; the Ready bit is also 0 from reset. */
            chbd[v1].xmitbd0.txbdsac.W = 1;/* end xmit buffer descriptors */
            chbd[v1].xmitbd0.txbdsac.I = 1;     /* enable intrpts */
            chbd[v1].xmitbd0.txbdsac.L = 1;     /* last buffer in frame */
            chbd[v1].xmitbd0.txbdsac.TC= 1;     /* xmit CRC sequence */
            chbd[v1].xmitbd0.txbdcnt = 0;       /* clear data length field */
      };
/* Clear interrupt table */
      intrpt = pdpr->INTBASE;
      for (v1 = 0; v1 < 10; v1++)
      {
            intrpt[v1].V = 0;                   /* clear valid bits */
            intrpt[v1].W = 0;                   /* clear wrap bits */
      };
      intrpt[9].W = 1;
      poem[0] = "Humpty Dumpty sat on a wall\n\r";
      poem[1] = "Humpty Dumpty had a great fall\n\r";
      poem[2] = "All the king's horses and all the king's men\n\r";
```

```
      poem[3] = "Couldn't put Humpty together again\n\r";
      poem[4] = "";
      poem[5] = "";
      linecntr = 0;                 /* init line counter */
      for (linecntr = 0; linecntr < 4; linecntr++)
      {
            chbd[linecntr].xmitbd0.txbdptr = poem[linecntr];/*    init    xmit
      pointer */
            chbd[linecntr].xmitbd0.txbdcnt = strlen(poem[linecntr]) + 1;/*
      init xmit cnt */
            chbd[linecntr].xmitbd0.txbdsac.R = 1;/* set xmit in BD */
      }
      for (v1 = 0; v1 < 3; v1++)
      {
            pdpr->ch[v1].CHAMR.MODE = 1;        /* select HDLC */
            pdpr->ch[v1].CHAMR.IDLM = 0;        /* no idles between frames */
            pdpr->ch[v1].CHAMR.ENT = 1;         /* enable channel xmit */
            pdpr->ch[v1].CHAMR.CRC = 1;         /* select 32-bit CRC */
            pdpr->ch[v1].CHAMR.NOF = 7;         /* 7 flags between frames */
            pdpr->ch[v1].CHAMR.POL = 1;         /* enable polling by RISC */
      }
 /* SCCE1 is cleared from reset */
      pdpr->SCCM1 = 0xF;            /* enable all intrpts */
      pdpr->CIMR.SCC1 = 1;          /* SCC1 interrupts enabled */
      pdpr->GSMR_L1.ENR = 1;        /* enable receiver */
      pdpr->GSMR_L1.ENT = 1;        /* enable transmit */
      while (pdpr->GSMR_L1.ENR == 1 | pdpr->GSMR_L1.ENT == 1)
      asm (" stop #$2000");         /* stop for next interrupt */
}

#pragma interrupt()                 /* make function an exception sr */
void SCC1esr()                      /* SCC1 exception service rtn */
{
      short er;                     /* event register scratchpad loc */
      asm(" move.w #$2300,sr");     /* decrement interrupt mask level */
      er = pdpr->SCCE1;             /* init event register scratchpad */
      pdpr->SCCE1 = er;             /* clear event register */
      if ((er & 8) == 8)            /* if interrupt table overflow */
```

```
        {};
        if ((er & 4) == 4)            /* if global interrupt occurred */
        {};
        if ((er & 2) == 2)            /* if global underrun */
        {};
        if ((er & 1) == 1)            /* if global overrun */
        {};
        pdpr->CISR = 0x4000_0000;     /* clear in-service bit */
}
getmbar()
{
        asm(" move.w #7,d0");         /* CPU space func code to d0 */
        asm(" movec.l d0,sfc");       /* load SFC for CPU space */
        asm(" lea.l $3ff00,a0");      /* A0 points to MBAR */
        asm(" moves.l (a0),d0");      /* get MBAR */
}
getvbr()
{
        asm(" movec.l vbr,d0");       /* get vector base reg value */
}
inittsatr(maxts)
short maxts;
{
        short curts;
        for (curts = 0; curts < maxts; curts++)
        {
                pdpr->SCC1.TSATR[curts].W = 0;      /* not last time slot */
                pdpr->SCC1.TSATR[curts].CP = curts;/* mark chan nmbr */
                pdpr->SCC1.TSATR[curts].mask7_6 = 3;/* no subchaneling */
                pdpr->SCC1.TSATR[curts].mask5_0 = 0x3F;/* no subchnlng */
                pdpr->SCC1.TSATR[curts].V = 1;      /* mark time slot valid */
        }
}
```

6.3 Restarting the Transmitter

A global underrun may require the SCC transmitter to be restarted. However, for channel-specific errors, only the affected channel need be restarted. The following steps are required to restart each channel:

- Prepare buffer descriptors.
- Set the POL bit in the channel mode register.

A stopped, but not deactivated channel is started as described above. A deactivated channel must first have the ZISTATE and TSTATE reinitialized to their correct values, followed by setting TSATTx[V] and CHAMR[ENT]. Lastly, set CHAMR[POL] if the buffers are ready.

6.4 Restarting the Receiver

A global receiver overrun may require the SCC receiver to be restarted. However, for channel-specific errors, only the affected channel need be restarted. The following steps are required to restart each channel:

- Prepare buffer descriptors.
- Initialize the ZDSTATE to either 0x080 (HDLC) or 0x1800_0080 (transparent).
- Initialize the RSTATE to 0x3900_0000 for MH360 and 860MH.

6.5 Disabling Receiver and Transmitter

A transmit channel can be stopped from sending any more data to the line with the STOP command described in Section 3.1, “Transmit Commands.” The transmitter will continue to send IDLEs or FLAGs according to the channel mode register setting. To deactivate a channel, the V bit has to be cleared in the time slot assignment table and the ENT bit has to be cleared in the channel mode register.

To stop a channel while receiving, use the STOP command as described in Section 3.2, “Receive Commands,” then perform a restart as described above.

6.6 Debugging Hints

Note that the following guidelines are subject to change; code should not rely on this information. The hints are for debugging purposes only.

6.6.1 Pointer Registers

Table 6-7 discusses the debugging hints for pointer registers. See Section 2.4.1, “Channel-Specific HDLC Parameters,” and See Section 2.4.2, “Channel-Specific Transparent Parameters,” for more information.

Table 6-7. Pointer Registers

	Offset	Name	Comments
Channel-specific parameters—HDLC and transparent	08	Tx internal data pointer	Holds pointer to current data in buffer.
	0E	Tx internal byte count	Holds down-counter of data left to transmit in this buffer descriptor—loaded when the buffer descriptor is opened with the BD length field.
	28	Rx internal data pointer	Holds pointer to address inside the current buffer where data will be received.
	2E	Rx internal byte count	Holds down-counter of space left in this Rx buffer—loaded when the BD is opened with the MRBLR field.

6.6.2 State Registers

The QMC has two sets of state parameters—global and channel-specific. These registers change if the QMC is running and the SCC is receiving clock.

Table 6-8. State Registers

	Offset	Name	Comments
Global state parameter	04	QMCSTATE	bit 0 • 1 if the QMC is stopped or has not started • 0 if the QMC is running
Channel-specific parameters—HDLC[1, 2]	04	TSTATE	bit 8— currently transmitting a frame bit 9— buffer descriptor is currently open (data is being transmitted from a BD) bit 12— channel has been initialized and is running
	24	RSTATE	bit 8—currently receiving a frame bit 9— buffer descriptor is currently open (data is being received into a BD) bit 11— reception is halted low word—current BD status word
Channel-specific parameters—transparent	04	TSTATE	bit 9— buffer descriptor is currently open (data is being transmitted from a BD) bit 12—channel has been initialized and is running.
	24	RSTATE	bit 9— buffer descriptor is currently open (data is being received into a BD) bit 11— reception is halted low word—current BD status word

Notes: [1] For more information on TSTATE and RSTATE, see Section 2.4.1, “Channel-Specific HDLC Parameters.”

[2] ZISTATE and ZDSTATE contain no meaningful parameters.

Chapter 7
Features Deleted in MC68MH360

An MC68MH360 operating in normal mode without the QMC protocol can perform the same functions as the MC68360 and MC68EN360 with two exceptions in protocol support. In order to create space in the CPM ROM for the QMC protocol, the support for Centronics and BiSync have been removed from the MH360. In all other respects, the MC68MH360 is compatible with its predecessors.

Chapter 8
Performance

Calculating performance is key to choosing the clock frequency required for a given system. For the 860MH and MH360, the large number of possible channel combinations complicates performance estimation.

This chapter addresses the problem by first providing a performance table for common configurations supported by the 860MH and/or the MH360 in Section 8.1, “Common Channel Combinations.” For configurations not covered in the first section, Section 8.2, “CPM Loading,” covers general guidelines and examples for determining the serial bit rate and CPM loading on a given system. The final section, Section 8.3, “Bus Latency and Peak Load,” addresses system bus utilization and arbitration.

For more definitive answers to performance questions, benchmark the desired configuration on a development board.

8.1 Common Channel Combinations

Table 8-1 provides some common channel combinations configured on the MH devices along with suggested operating frequencies. Note that the MH360 is available only at 25 and 33 MHz,; the 860MH is currently available at 25, 40, and 50 MHz.

Table 8-1. Common QMC Configurations

Protocols Selected	Frequency Supported			
	25 MHz	33 MHz	40 MHz	50 MHz
SCC1: 24-channel QMC. Serial bit rate 1.544 Mbps (T1)	Yes	Yes	Yes	Yes
SCC1: 32-channel QMC. Serial bit rate 2.048 Mbps (E1/CEPT)	Yes	Yes	Yes	Yes
SCC1: 10-Mbps Ethernet; SCC2: 12 x 64-Kbps QMC; SCC3: 12 x 64-Kbps QMC. TDM bit rate = 1.544 Mbps	Yes	Yes	Yes	Yes
SCC1: 10-Mbps Ethernet; SCC2: 16 x 64-Kbps QMC; TDM bit rate = 1.544 Mbps	Yes	Yes	Yes	Yes
SCC1: 32 x 64-Kbps QMC; SCC2: 64 Kbps; SCC3:64 Kbps; SCC4: 64 Kbps; all HDLC.	Yes	Yes	Yes	Yes

Table 8-1. Common QMC Configurations (Continued)

Protocols Selected	Frequency Supported			
	25 MHz	33 MHz	40 MHz	50 MHz
SCC1: 10-Mbps Ethernet; SCC2: 16 x 64-Kbps QMC; SCC3: 16 x 64-Kbps QMC; SCC4: 64-Kbps HDLC. TDM bit rate = 2.048 Mbps	No	Yes	Yes	Yes
SCC1: 10-Mbps Ethernet; SCC2: 12 x 64-Kbps QMC; SCC3: 12 x 64-Kbps QMC; SCC4: 64-Kbps HDLC. TDM bit rate = 1.544 Mbps	No	Yes	Yes	Yes
SCC1: 24-channel QMC; SCC2: 24-channel QMC. Serial bit rate 2 x 1.544 Mbps — 2 x T1	No	No	Yes	Yes
SCC1: 10-Mbps Ethernet SCC2: 24-channel QMC; SCC3: 24 Channel QMC. Serial bit rate 2 x 1.544 Mbps — 2 x T1	No	No	No	Yes
SCC1: 32-channel QMC; SCC2: 32-channel QMC. Serial bit rate 2 x 2.048 Mbps (E1/CEPT)	No	No	No	Yes

8.2 CPM Loading

This section primarily deals with the CPM loading of the MH360 and 860MH. As the CPM architecture is identical on both devices, the performance for a given clock frequency is identical. Compared to standard protocols, the QMC protocol places more demands on the CPM RISC because it requires the CPM to handle all of the bit manipulation normally implemented with hardware support built into the SCCs.

The SCC operates transparently in QMC mode. The SCC's main function is serial-to-parallel conversion of the data stream out of the time slot assigner, and parallel-to-serial conversion of the data stream gated into the time slot assigner. All bit manipulating is done in the CPM RISC software or hardware. Thus, the CPM has a much higher load when operating in QMC mode, even if all time slots are concatenated to one logical channel. This loading is reflected in the measured performance.

Table 8-2 gives loading guidelines. The table assumes a single SCC running at 100% of the CPM bandwidth. For each protocol supported, the table gives the ratio of the SCC bit rate versus clock frequency, and the maximum serial throughput at standard frequencies.

Table 8-2. CPM Performance Table

Protocol	SCC Rate: Clock Frequency Mbps: MHz	Maximum Serial Throughput			
		25 MHz Mbps	33 MHz Mbps	40 MHz Mbps	50 MHz Mbps
Transparent	1 : 3.125 FD	8	10.56	12.8	16
HDLC	1 : 3.125 FD	8	10.56	12.8	16
UART	1 : 10.4 FD	2.4	3.168	3.84	4.8

Table 8-2. CPM Performance Table (Continued)

Protocol	SCC Rate: Clock Frequency Mbps: MHz	Maximum Serial Throughput			
		25 MHz Mbps	33 MHz Mbps	40 MHz Mbps	50 MHz Mbps
Ethernet	1 : 1.136 HD	22	29	35	44
SMC transparent	1 : 16.67 FD	1.5	1.98	2.4	3
SMC UART	1 : 113.636 FD	0.220	0.290	0.352	0.440
QMC	1 : 11.90 FD	2.1	2.8	3.36	4.2
Bisync	1: 16.67 FD	1.5	1.98	2.4	3

NOTE

FD = Full duplex, HD = Half duplex

Further examples are given in Appendix A of the MPC860 user's manual.

Using Table 8-2, estimations of bandwidth utilization may be made. To calculate the total system load, add the CPM utilization from every channel together. Assuming approximately linear performance versus frequency[1], the general problem reduces to taking simple ratios:

$$\text{CPM Utiilization} = \left(\frac{\text{serial rate}_1}{\text{max serial rate}_1}\right) + \left(\frac{\text{serial rate}_2}{\text{max serial rate}_2}\right) \cdots$$

For example, since a 25-MHz Ethernet running at 22 Mbps consumes approximately 100% of the bandwidth, what bandwidth does a 10-Mbps channel require?

$$\text{CPM Utilization} = \frac{\text{serial rate}}{\text{max serial rate}} = \frac{10}{22} = 0.45$$

The above equation shows the 10-Mbps channel requiring 45% of the CPM bandwidth. More examples follow.

[1]Most protocols' performance is scalable linearly to frequency with the exception of Ethernet which has nonlinear behavior. However, for these calculations we assume linear scaling with frequency.

Example #1

A device is operating at 25 MHz. SCC1 runs 1x10-Mbps Ethernet in half duplex, SCC2 runs 1 x 2-Mbps HDLC, SCC3 runs 1 x 64-Kbps HDLC, SCC4 runs 1 x 9.6-Kbps UART and SMC1 runs 1 x 38-Kbps SMC UART. The following equation applies:

$$\left(\frac{10}{22}\right)+\left(\frac{2}{8}\right)+\left(\frac{0.064}{2.4}\right)+\left(\frac{0.0096}{2.4}\right)+\left(\frac{0.038}{0.22}\right) = 0.96 \qquad \textbf{(<1)}$$

This yields a percentage CPM utilization of 96% meaning the device can handle these protocols at this frequency. Note the 9.6-Kbps UART link only requires 0.4% of the CPM bandwidth, implying that in any configuration where there is free bandwidth that it will be possible to run a low-rate UART link.

Example #2

A device operating at 25 MHz is required to run 24 channels at 64 Kbps in QMC mode with an additional 2 HDLC channels operating at 128 Kbps each. The following equation applies:

$$\left(\frac{2\times 0.128}{8}\right)+\left(\frac{24\times 0.064}{2.1}\right) = 0.76 \qquad \textbf{(<1)}$$

Example #3

The last example shows an application with 32 QMC channels and one additional 2-Mbps HDLC channel. The following equation applies:

$$\left(\frac{2}{8}\right)+\left(\frac{32\times 0.064}{2.1}\right) = 1.22 \qquad \textbf{(will not work)}$$

Since the result above is greater than 1, this configuration may not work at 25 MHz. If a 33-MHz operation is used, CPM utilization will drop below 1, allowing the combination to be supported. The following equation applies:

$$1.22\times\left(\frac{25}{33}\right) = 0.92 \qquad \textbf{(<1)}$$

In general, a channel combination will work if the combined load is less than 1. The equations are scalable to frequency with the exception of the nonlinear Ethernet protocol. Taking Ethernet into account is difficult. Designers will need to benchmark performance for results near 1.

8.3 Bus Latency and Peak Load

Each time slot is 8 bits, but the QMC protocol transfers 32 bits of data whenever possible. Thus, for each active channel operating within large frames, two 32-bit SDMA data transfers (one for Tx and one for Rx) occur approximately every fourth TDM frame (every 500 µs in CEPT/T1 interfaces). During buffer closing or opening, load will increase approximately 3 to 4 times. Table 8-3 illustrates the bus activities involved when one QMC channel switches from one Tx HDLC frame to the next.

Table 8-3. QMC Actions in Tx Buffer Switch

Frame Number	Actions	Number of Bus Cycles
1	Read long word from buffer (last in frame)	1
2	Send byte	0
3	Send byte	0
4	Send byte	0
5	Send last byte in frame	0
6	Send CRC	0
7	Send CRC	0
8	Send flag Read interrupt table Write interrupt table Write BD	3
9	Send flag Read next BD status/length Read BD data pointer Read data	3
10	Send first byte of next frame	0

Note: The table assumes the channel uses one time slot per TDM frame and that no PAD characters, preceding flags or flag sharing is used.

In Table 8-3, the frame number refers to the 125-µs frame; the numbering is arbitrary but sequential. The actions refer to the visible functions executed on the CPM. The number of external bus cycles executed by the CPM represents the load on the bus.

The sequence in Table 8-3 starts when the last 32 bits are read from a buffer. One byte is transferred over the TDM link per frame over the next four 125-µs frames. Then the CRC is sent. In this case, it is a 16-bit CRC requiring two time slots over the next two frames. The heavy load on the bus starts when the CPM must close the buffer in frame 8. At this point the CPM needs three accesses to the bus to read and write to the interrupt table and update the buffer descriptor's status. In the following frame, the next buffer is opened requiring three accesses to read the status and length, read the data pointer and read the data.

In the worst-case scenario, all channels open and close a buffer during the same TDM frame resulting in the peak load all performance calculations are based on. This peak load is far from the norm and can be controlled by the transmitter spreading the starting point of transmit buffers over several TDM frames.

In multimaster systems, bus latency may affect the performance of the device. The maximum external bus latency figures shown in Table 8-4 are measured from the assertion of the BR (bus request) to the assertion of the BGACK (bus grant acknowledge); that is, from start of bus request output being active until the cycle is completed. For multimaster systems, bus arbitration overhead is included. Latencies of up to 40 clocks were simulated; for values over 40, the acceptable latency may be larger.

Table 8-4 shows average maximum acceptable bus latencies, meaning the device can tolerate longer bus delays if they are infrequent. For lengthy delays, a larger FIFO can pick up the slack, continuing emptying or filling depending on the data flow direction. Therefore, the larger the FIFO the more tolerant the system is to infrequent peaks in bus delays. However, the average acceptable bus latency still depends on the overall data rate and frame length and not on the FIFO size.

Table 8-4. Simulated Latencies

Maximum Acceptable Latency (Bus Cycles)		Channel Combinations
25 MHz	33 MHZ	
Not supported	12	SCC1: Ethernet; SCC2: 16 x 64 Kbps; SCC3: 16 x 64 Kbps
Not supported	11	SCC1: Ethernet; SCC2: 16 x 64 Kbps; SCC3: 16 x 64 Kbps; SCC4: 64 Kbps HDLC
9 clocks	>40	SCC1: 32 x 64 Kbps. Serial bit rate 2.048 Mbps (E1/CEPT)
8 clocks	35	SCC1: 32 x 64 Kbps; SCC2: 64 Kbps; SCC3: 64 Kbps; SCC4: 64 Kbps; all HDLC
40 clocks	>40	QMC with 24 channels. Serial bit rate 1.544 Mbps (T1)
33 clocks	>40	SCC1: 24 x 64 Kbps; SCC2: 64 Kbps; SCC3: 64 Kbps; SCC4: 64 Kbps; all HDLC
8 clocks	24	SCC1: Ethernet; SCC2: 12 x 64 Kbps; SCC3: 12 x 64 Kbps. TDM bit rate = 1.544 Mbps
Not supported	23	SCC1: Ethernet; SCC2: 12 x 64 Kbps; SCC3: 12 x 64 Kbps; SCC4: 64-Kbps HDLC. TDM bit rate = 1.544 Mbps
40 clocks	>40	SCC1: 16 x 128 Kbps. TDM bit rate = 2.048 Mbps

Chapter 9
Multi-Subchannel (MSC) Microcode

The RISC processor in the PowerQUICC has an option to execute microcode from the internal dual-ported RAM. Motorola uses this feature to enhance existing protocols or implement additional protocols. Customers can purchase RAM microcodes in an object-code format and download it to the PowerQUICC dual-ported RAM during system initialization.

The RAM microcode is provided by Motorola as a set of S records that can be downloaded directly to an application development system or stored in a system EPROM; for more information on S records, see Appendix C of the *M68000 Family Programmer's Reference Manual*. After system reset, the binary of the microcode should be copied to the dual-ported RAM. The QUICC registers, including the RISC controller configuration register (RCCR), should be initialized as specified in the microcode RAM documentation. Before the RISC is used in the system, the user should issue a reset command to the communications processor command register (CR). The microcode RAM functions are available in addition to all protocols available in the standard QUICC microcode ROM.

9.1 MSC Microcode Features

The multi-subchannel (MSC) microcode is a downloadable microcode for the MPC860MH and has the following key features:

- General
  - Multiple subchannels within a single 8-bit time slot
  - Bit resolution for subchannels
  - Up to 32 independent communications channels for both Rx and Tx
  - Supports either transparent or HDLC protocols per subchannel
- Performance
  - 32 channels + 10-Mbps ethernet support at 40-MHz system clock

9.2 MSC Microcode Operation

In normal operation (without the MSC microcode), the QMC protocol allows specific bits in an 8-bit time slot to be masked to create a single subchannel per SCC. A problem arises when multiple subchannels are multiplexed within a single time slot as in GSM (global system for mobile communications) where four 16-Kbps subchannels are multiplexed into a single 64-Kbps channel over a 2.048-Mbps A bis link. A brute-force solution routes the separate subchannels to different SCCs, consuming all four SCCs for the single TDM link as shown in Figure 9-1. Each SCC filters out one of the four 2-bit subchannels in time slot 2 (TS2) using a unique mask located in its time slot assignment tables (TSATRx/TSATTx). With the MSC microcode, subchannels can be regenerated using only one SCC.

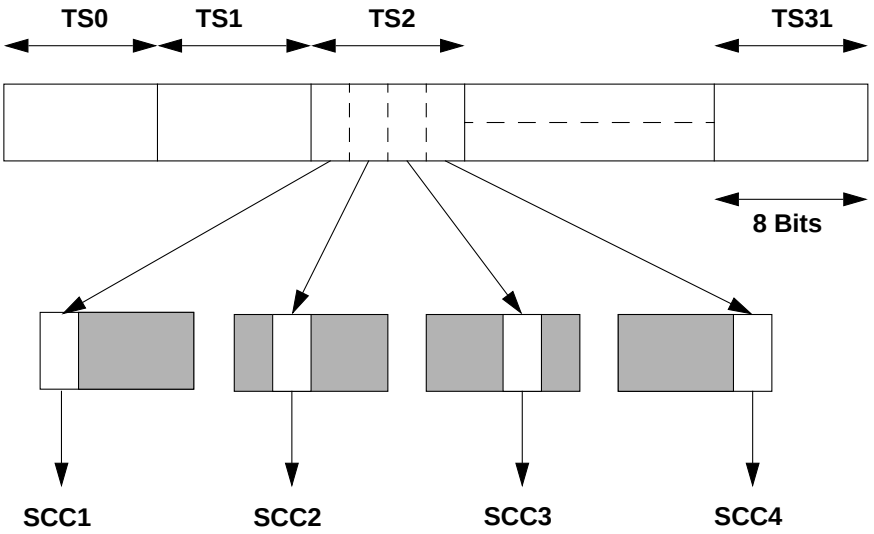
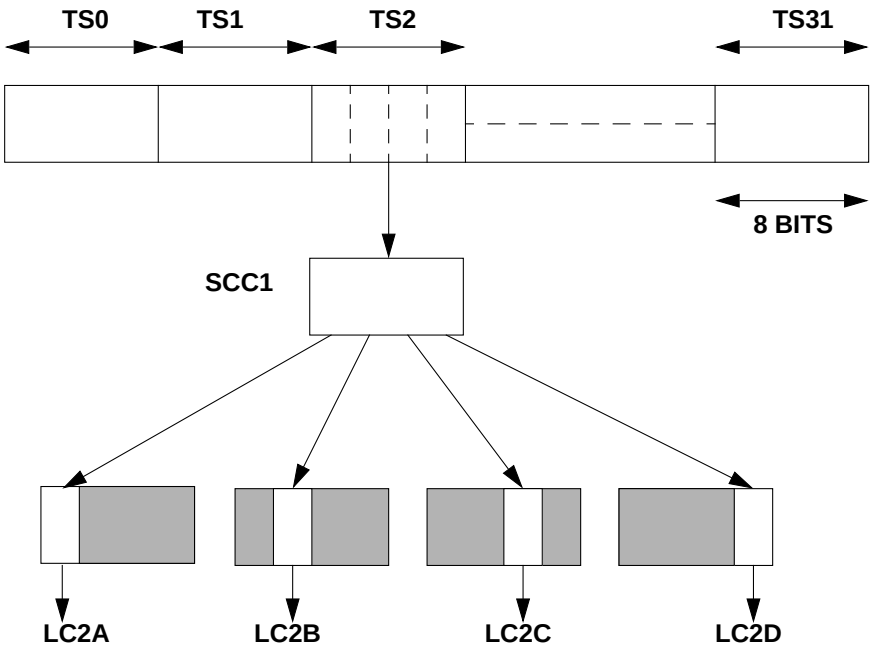


Figure 9-1. Two-Bit Subchannel Implementation without MSC Microcode

The MSC microcode enables an 8-bit time slot to be split into multiple, bit-resolution subchannels. The microcode applies user-defined masks in a time slot assignment table entry to subdivide a given channel. Bit 11 of a table entry is now called the L bit to mark the last subchannel of a given time slot. Figure 9-2 shows the MSC microcode solution to the above GSM problem. Again in this example, time slot 2 contains four 2-bit channels, but now the full time slot can be routed to a single SCC and split into subchannels within the time slot assignment tables.

NOTE

As the L bit (bit 11 of a time slot assignment table entry) is taken from the original channel pointer field, the addressing capability of the QMC is reduced from 6 bits (64 channels) to 5 bits (32 channels) for receive and transmit.



Masking performed by a single SCC to create four 2-bit channels

Figure 9-2. Two-Bit Subchannel Implementation with MSC Microcode

9.3 Programming the MSC Protocol

The MSC protocol configuration is identical to the QMC configuration with the exception of the time slot assignment tables. Figure 9-3 shows the addition of the L bit needed for the MSC configuration. Note that it is possible to have up to 32 channels coming from any permutation of the time slots.

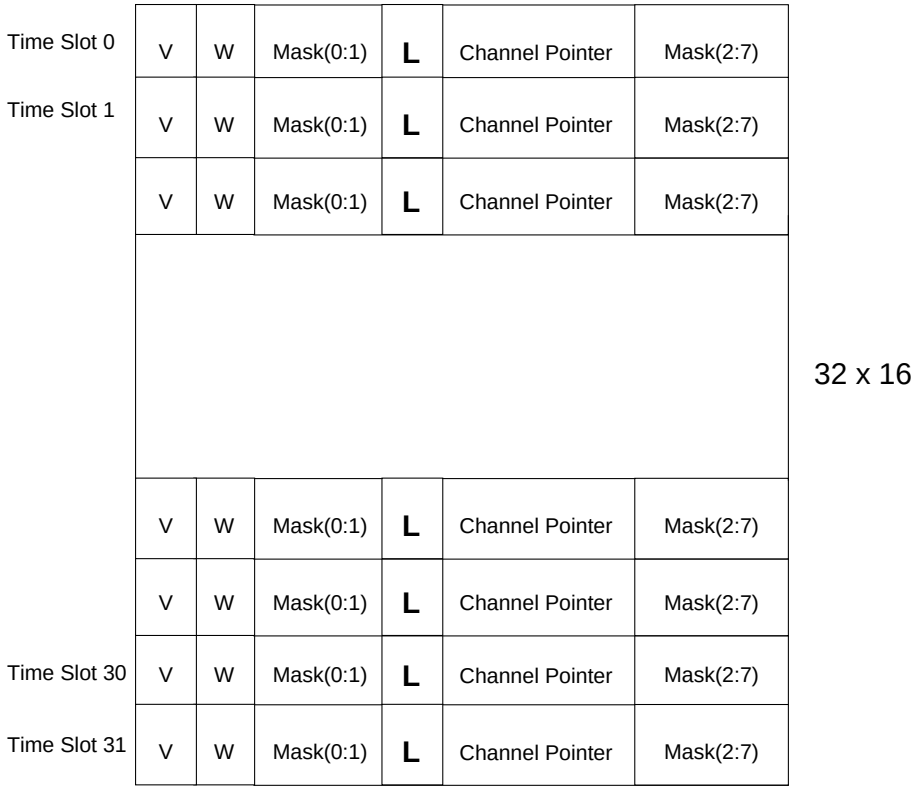


Figure 9-3. Time Slot Assignment Table Showing MSC Configuration

Table 9-1 describes the fields in the time slot assignment table for receive (TSATRx) when the MSC microcode is enabled.

Table 9-1. Time Slot Assignment Table Entry Fields for Receive (MSC)

Field	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. 0 The data in this 8-bit time slot is totally ignored and not written to any buffer. 1 The data in this 8-bit time slot is valid and written to the current buffer, pointed to by the channel pointer entry, after protocol processing (for example, zero deletion in HDLC).
W	Wrap bit—Identifies the last entry in TSATTx. 0 This is not the last time slot in the frame. 1 The RISC processor wraps around and handles time slot 0 or the first 8 bits transferred from the TSA in the next request. The next request is identified by a frame synchronization pulse.

Table 9-1. Time Slot Assignment Table Entry Fields for Receive (MSC) (Continued)

Field	Description
L	Last bit—Identifies the last subchannel in a time slot. 0 This is not the last subchannel in the time slot. 1 This is the last sub channel within an 8-bit time slot.The RISC processor handles the next time slot transferred from the TSA in the next request.
Rx channel pointer	This field of the TSATRx entry identifies the data channel that is routed to this time slot. The actual channel pointer is 11 bits long, and contains the starting address of the channel-specific parameter area (address of TBASE). The 5 most-significant bits are taken from TSATRx, and the 6 least-significant bits are always internally set to zero. For the MSC operation, the addressing range is 2 Kbytes.
Mask(0–7)	Mask bits—These 8 bits identify the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. Any unmasked bit (1) is processed in the receiver for a valid time slot. Any masked bit (0) is ignored by the receiver for a valid channel and no bit counter is affected.

Table 9-2 describes the fields in the time slot assignment table for transmit (TSATTx) for MSC operation.

Table 9-2. Time Slot Assignment Table Entry Fields for Transmit (MSC)

Name	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. 0 Logic 1 is transmitted. If the Tx signal of the TDM interface is programmed to be an open drain output (port B programming), other devices can transmit on nonvalid time slots. 1 Data is transmitted from its associated buffer in combination with the mask bit settings.
W	Wrap bit—The wrap bit identifies the last entry in TSATTx. 0 This is not the last time slot in the frame. 1 The RISC processor wraps around and handles time slot 0 or the first 8 bits of data in the SCC in the next request. The next request is identified by a frame synchronization pulse.
L	Last bit—Identifies the last subchannel in a time slot. 0 This is not the last subchannel in the time slot. 1 This is the last sub channel within an 8-bit time slot.The RISC processor handles the next time slot transferred to the TSA in the next request.
Tx channel pointer	This field of the TSATTx entry identifies a data channel which is routed to this time slot. The actual channel pointer is 11 bits long, and contains the starting address of the channel-specific parameter area (address of TBASE). The 5 most significant bits are taken from TSATTx channel pointer field, and the 6 least significant bits are always internally set to zero. For MSC protocol, the addressing range is 2 Kbytes.
Mask(0–7)	Mask bits—Identifies the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. For a valid channel with an unmasked bit (1), the bit position is filled according to the protocol. A valid channel with a masked bit (0) transmits a logic high (1).

9.4 MSC Subchanneling Example

Figure 9-4 shows an example for eight 20-bit subchannels.

Time Slot 0	V	W	00	**0**	Channel #0A	000011
	V	W	00	**0**	Channel #0B	001100
	V	W	00	**0**	Channel #0C	110000
	V	W	11	**1**	Channel #0D	000000
Time Slot 31	V	W	00	**0**	Channel #31A	000011
	V	W	00	**0**	Channel #31B	001100
	V	W	00	**0**	Channel #31C	110000
	V	W	11	**1**	Channel #31D	000000

Figure 9-4. Example for Eight 2-Bit Subchannels

The example in Figure 9-4 uses two time slots to handle eight 2-bit subchannels. Time slot 0 is subdivided into four 2-bit subchannels. Note that time slot 0 is processed four times for the channels labeled 0A, 0B, 0C and 0D, each with different masks. It is only in the fourth entry that the L-bit (last bit) is set, instructing the CPM to process the next time slot. The same is true for time slot 31. It again is subdivided into four 2-bit subchannels. Note that the maximum number of channels is 32 due to the 5-bit channel pointer.

9.5 QMC Memory Organization

Figure 9-5 shows the internal memory map of the MPC860MH when the MSC microcode is resident. Note that the microcode resides in the first 1 Kbyte of dual-ported RAM and uses the last 256 bytes of dual-ported RAM for data. This means that the channel-specific parameters are offset from IMMR + 0x2400, and may occupy up to IMMR + 0x2BFF. This space can hold parameters for up to 32 channels. Any unused memory, and the 768 bytes from IMMR = 0x2C00 may be used by the user for buffer descriptors or as standard RAM.

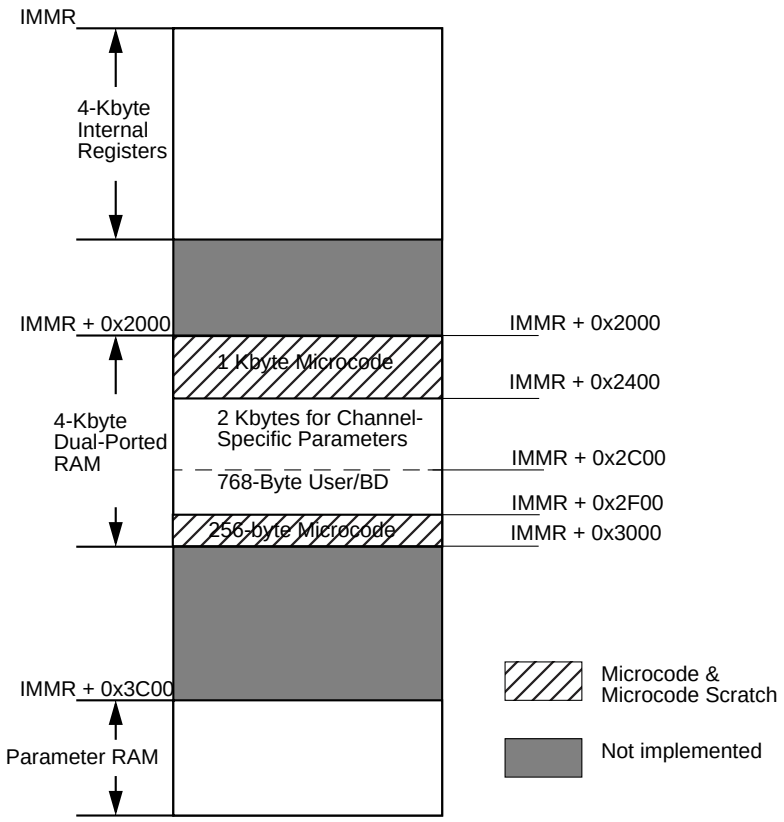


Figure 9-5. MPC860MH Internal Memory Map with MSC Microcode Enabled

9.6 Multi-Subchannel Initialization

The initialization of the MSC microcode is the same as the standard QMC initialization with the following additions:

- Prepare TSATTx and TSATRx to include the L (last) bit
- The RISC controller trap registers for the MPC860 rev A must be set as follows:
  - Address SCC base + 9CC, RCTR1 = 0x8074
  - Address SCC base + 9CE, RCTR2 = 0x8054
  - Address SCC base + 9D0, RCTR3 = 0x92F2
  - Address SCC base + 9D2, RCTR4 = 0x9097
- Load microcode S records into dual-ported RAM
- Set the RCCR (address SCC base + 9C4) = 0x0002
- Set the GSMR_H = 0x07A0 (RFW = 1)

Appendix A
68360 Bit Numbering

This appendix shows the bit numbering for the 68360.

A.1 Time Slot Assignment Table

Figure A-1 shows the 68360 bit numbering for a general time slot assignment table for thirty-two 16-bit time slots. The fields will be used to either transmit or receive channels.

Time Slot 0	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)
Time Slot 1	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)
	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)
					32 x 16
	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)
	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)
Time Slot 30	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)
Time Slot 31	V	W	Mask(7:6)	Channel Pointer	Mask(5:0)

Figure A-1. Time Slot Assignment Table

Figure A-2 shows the 68360 bit numbering for a time slot assignment table for 64-channel common Rx and Tx mapping.

Figure A-2. Time Slot Assignment Table for 64-Channel Common Rx and Tx Mapping

A.2 Registers in HDLC Mode

Figure A-3 to Figure A-6 show the 68360 bit numbering for registers in HDLC mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	0	IDLM	ENT	RESERVED			POL	CRC	0	RESERVED		NOF			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure A-3. CHAMR—Channel Mode Register (HDLC)

Interrupt Table Entry:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V	W	NID	IDL	—	CHANNEL NUMBER					MRF	UN	RXF	BSY	TXB	RXB
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTMSK:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		INTERRUPT MASK		RESERVED						INTERRUPT MASK BITS					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure A-4. Interrupt Table Entry and INTMSK (HDLC)

31	30	29	28	27	26	25	24
0	0	1	MOT	FC3–FC0			

Figure A-5. TSTATE (HDLC)

31	30	29	28	27	26	25	24
0	0	1	MOT	FC[3–0]			

Figure A-6. RSTATE (HDLC)

A.3 Registers in Transparent Mode

Figure A-7 to Figure A-10 show the 68360 bit numbering for registers in transparent mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	RD	1	ENT	RESVD	SYNC	–	POL	0	0	RESERVED		0			
RESET:						MODCK	MODCK								
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Figure A-7. CHAMR—Channel Mode Register (Transparent Mode)

INTMSK:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		RESERVED		RESERVED						RES	INTERRUPT MASK BITS				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTMSK:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		RESERVED		RESERVED						RES	INTERRUPT MASK BITS				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure A-8. INTMSK and Interrupt Table Entry (Transparent Mode)

.

31	30	29	28	27	26	25	24
0	0	1	MOT	FC[3–0]			

Figure A-9. TSTATE (Transparent Mode)

.

31	30	29	28	27	26	25	24
0	0	1	MOT	FC3–FC0			

Figure A-10. RSTATE (Transparent Mode)

A.4 Command Register

Figure A-11 shows the 68360 bit numbering for the command register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST	QMC OPCODE			1	1	1	0	0	CHANNEL NUMBER					–	FLG

Figure A-11. Command Register

A.5 SCC Event Register

Figure A-12 shows the 68360 bit numbering for the SCC event register.

7	6	5	4	3	2	1	0
				IQOV	GINT	GUN	GOV

Figure A-12. SCC Event (SCCE) Register

A.6 SCCM Register

Figure A-13 shows the 68360 bit numbering for the SCCM register.

7	6	5	4	3	2	1	0
				IQOV	GINT	GUN	GOV

Figure A-13. SCCM Register

A.7 Receive and Transmit Buffer Descriptors

Figure A-14 and Figure A-15 show the 68360 bit numbering for the receive and transmit buffer descriptors.

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

Figure A-14. Receive Buffer Descriptor (RxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**R**	—	**W**	**I**	**L**	**TC**	**CM**	**—**	**UB**	—	—	—	**PAD**			
OFFSET + 2	DATA LENGTH															
OFFSET + 4	TX DATA BUFFER POINTER															
OFFSET + 6																

Figure A-15. Transmit Buffer Descriptor (TxBD)

Appendix B
Frequently-Asked Questions

This appendix provides a list of frequently-asked questions and solutions for the MH360 and 860MH.

B.1 Questions Common to MH360 and 860MH

Q: What are the performance differences between the 68MH360 and 860MH?

A: Since the 860 and 360 have the same CPM, performance scales linearly with frequency as shown in Table B-1.

Table B-1. CPU Performance

Device	Frequency	Performance
MH360	25 MHz	6.3 Dhrystone MIPS
	33 MHz	8.4 Dhrystone MIPS
PowerQUICC	25 MHz	33 Dhrystone MIPS
	40 MHz	52 Dhrystone MIPS
	50 MHz	66 Dhrystone MIPS

Q: Comparing the MH360 and 860MH, what is the best way to support two T1 TDM channels?

A: There are two bottlenecks in running the QMC protocol on the 360 and 860. One is CPM performance, and the other is parameter RAM space.

Running the QMC protocol consumes nearly all of the CPM bandwidth at 25 MHz. As the CPM is the same for the 360 and 860, the same limitation applies to both when running at 25 MHz.

In addition, the QMC protocol consumes nearly all of the dual-ported RAM of a 360 preventing more than one QMC from running at the same time (even if the 360's speed could be increased).

However, since the 860 has twice as much dual-ported RAM as the 360, it does have enough space to run two QMCs. Also, the CPM bandwidth is doubled at 50 MHz.

Therefore, a 50-MHz MPC860MH will be needed to run 64 channels of HDLC on one device.

B.2 860MH-Related Questions

Q: Is Ethernet only available on SCC1 for both 860EN and 860MH?

A: Ethernet is available on any channel. We recommend it on SCC1 due to its larger FIFO.

Q: How is 64-channel QMC implemented on the 50-MHz 860MH? What is the serial speed of the TDM channels?

A: Use two SCCs running 32-channel QMC protocol. Each channel is assumed to be 64-Kbps, like a normal time slot on a T1/E1 line, giving an aggregate rate of 4 Mbps (that is, twice the E1 rate).

Q: Does running transparent-mode processing on the QMC channels decrease the load on the CPM?

A: CPM loading in transparent mode is not significantly different from the loading in HDLC mode; therefore, performance will be the same.

Q: How many channelized T1/E1 ports does the 860MH support? (where E1 is thirty-two 64-Kbps channels and T1 is 24 channels)

A: With respect to running multiple channels of HDLC, the major limitation of the current 860MH is clock frequency. A 25-MHz part can run only 32 HDLC channels, while a 50-MHz part can run 64 channels. At this point, however, the size of the dual-ported RAM limits the number of HDLC channels to 64.

The MPC860 also has just two time slot assigners. Therefore, it can directly terminate at most two T1s or E1s.

Q: How is the 860MH configured to support more than 32 channels.

A: The QMC protocol for the 860MH can be used to support more than 32 HDLC channels in three ways:

- In one method, use shared transmit/receive channel routing on one SCC to run the QMC protocol linking the maximum of 64 time slots of a single multiplexed line to 64 separate logical channels.
- In another method, run the QMC protocol on two separate SCCs, each with its own set of parameters. With this method, two separate E1s can be routed to the two separate SCCs. It is not possible, however, to share channels from both E1s at random between the SCCs. (One E1 will map to the 32 logical channels of one SCC.)

- A hybrid approach runs a single line of up to 64 multiplexed time slots to two separate SCCs, each with its own set of parameters. Normally this would route 32 time slots to each SCC. This would have the benefit of doubling your effective FIFO depth, allowing greater system design flexibility.

Q: My understanding is that the MH360 could not support the SS-7 microcode. Is this also true of the 860MH?

A: The 860MH will not support SS-7 over the multiplexed (QMC) channels. If SS-7 is to be run, it must be run over its own dedicated SCC. However, by using the time slot assigner, the traffic from this SCC could be routed over the same E1 or T1 as the other multiplexed HDLC channels.

For example, one channel of an E1 could be routed to an SCC running SS-7, and the other 31 channels to an SCC running QMC. Thus, the number of SS-7 channels allowed is limited to the number of SCCs (that is, at most 4).

Q: What is the CPM's maximum CPU bus utilization when running two 2-Mbps QMC channels?

A: For 64 channels (two E1 lines), two 2-Mbps full duplex means 8 Mbps of aggregate traffic. Factoring in a large margin for buffer descriptor accesses bumps this 8 Mbps up to 10 Mbps. The 10 Mbps of traffic translates to 0.3 megatransfers of 32 bits each requiring only 1.5 MHz out of a 50-MHz bus (assuming a 5 wait-state memory). A similar calculation for Ethernet would account for higher data traffic and fewer descriptor accesses.

Q: Is the 860MH pin compatible with the 860DH?

A: Yes, it is pin compatible.

Q: Are BISYNC and Centronics still removed from the 860MH as they are with the MH360?

A: No, Centronics and Bisync are both supported on the 860MH.

Q: How is time slot 0 identified on an SCC? Is an external sync required?

A: The TSA identifies time slot 0. A sync pulse must be provided to the TSA at the beginning of a frame.

Q: What does the larger dual-ported RAM on the 860MH provide?

A: The larger dual-ported RAM means that up to 64 QMC channels may be supported. It also provides more buffer descriptor area needed for the higher serial performance at higher speeds.

B.3 MH360-Related Questions

Q: Does the MH360 still have a full Ethernet controller?

A: Yes.

Q: What frequency MH360 is required to support 10-Mbps Ethernet and 64-channel QMC on a 2.048-Mbps TDM?

A: A 33-MHz MH360 is required.

Q: How should an MH360 be configured to run both 2.048-Mbps QMC and 10-Mbps Ethernet?

A: Put Ethernet on SCC1 and split the TDM time slot—16 channels on SCC2 and 16 on SCC3.

Q: When the part has the core disabled (68040 companion mode), will it support both the Ethernet and a 32-channel QMC when clocked at 25 MHz? If not, how many QMC channels will it handle?

A: A 33-MHz part is needed to run 32-channel QMC and Ethernet. A 25-MHz MH360 can support 24-channel QMC and Ethernet. The bottleneck is in the CPM performance not the CPU.

Q: Is the pinout for the MH360 the same as the standard 360?

A: Yes.

Q: Can two GCI ports be connected to an MH360?

A: Yes, with some caveats. Run each GCI channel into a TSA. Then use SCC1 for the two B channels and SCC2 for the other two B channels. Finally, use SCC3 and SCC4 for the respective D channels. However, this method provides no support for M-, C/I-, A-, or E-channels.

Q: What are the differences in loopback modes for the time slot assigner, particularly with respect to the MH360?

A: There are three ways to do loopback:

- Use the GSMR to make all 32 virtual channels loop back as they pass through the SCC. Each virtual channel will transmit through the SCC in the order defined in the transmit portion of the SI RAM. The transmit side of the virtual channel will loop back directly into the receive side of the virtual channel with no indirection by the SI.
- Use the SIMODE to cause global loopback of the multiplexed data stream in the SI. By this method, the first transmit time slot will loop back into the first receive time slot, etc. However, different time slots can be assigned to different virtual channels in the SI RAM. For example, virtual channel 1 could be assigned to

transmit time slot 1, but virtual channel 5 could be assigned to receive time slot 1. Therefore, global loopback would cause the data transmitted on channel 1 to be received on channel 5.

- For loopback on an individual time slot, set bit 15 of the corresponding entry in the SIRAM.

For the last two methods, use a common transmit/receive clock and transmit/receive sync pulses. Also in these last two methods, if the loopbacked data is to be received on the same virtual channel, make sure the transmit and receive portions of your SI RAM entries match.

Q: Does the MH360 support signalling system 7 (SS-7) on all 32 channels?

A: No. The MH360 does not support SS-7. Run the SS-7 microcode on a normal QUICC to get 4 channels of SS-7 support.

Q: Is any extra support logic required to attach an MH360 to an ISDN primary rate line?

A: Yes, a T1 transceiver and framer unit will be needed. Manufacturers include Dallas Semiconductor, Mitel, and Level One.

Q: Can the MH360 be used as an ISDN basic rate terminator?

A: Yes, send multiple Motorola U Interface outputs into a single SCC using the QMC controller and the Motorola IDL2.

Q: 360 documentation says that the time slot assigner can be used with all of the SCCs and the SMCs. However, the MH360 reference manual states that the time slot assigner works with the SCCs. Can the time slot assigner work with the SMCs?

A: Yes.

Q: What is the maximum bus latency at 25 MHz when using 32 64-Kbps channels?

A: Nine clocks. It is very important. At 25 MHz, the CPM is just able to empty the FIFO quickly enough to prevent underrun. If the part is run at 33 MHz or if only 24 channels are needed, the margin increases allowing for a better worst case.

Q: Can TDM_b be used for the QMC on the MH360?

A: Yes.

Q: Can the transparent channels of the MH360 (QMC mode) synchronize on an inline data pattern?

A: No.

Appendix C
Connecting ISDN Multiple S/T or U Interfaces to QUICC32

Using IDL or GCI protocols, the MC145574 (S/T interface) and the MC145572 (U interface) can be gluelessly interfaced to members of the MC68302 family for low-cost, active-ISDN basic rate terminal applications.

For applications needing to support more than one basic rate interface (BRI), such as LAN/WAN bridges, PBX, line cards or multiple-line terminal adaptors, a system solution using multiple MC145574s or MC145572s can be built around a QUICC32 (MC68MH360).

The QUICC32 and the QMC (QUICC's multichannel controller) protocol are useful for such ISDN applications requiring several logical channels on one physical medium.

This appendix shows how multiple MC145574s or MC145572s can be connected to a QUICC32, describing the level-1 connections and explaining the data flow through the devices.

No software issues are addressed in this appendix.

C.1 The QMC Protocol

Based upon the IDL bus, the QMC protocol implemented on the QUICC32 generates a TDM (time-division multiplexing) bus with programmable time slots for each ISDN interface. With 32 time slots, each carrying 8 consecutive bits forming 64-Kbps channels, a 2-Mbps TDM line (roughly equivalent to a CEPT/E1 link) can be created.

Time slot zero (TS0) is dedicated to the first B1 channel, with TS1 assigned to the first B2 channel and TS2 to the first D channel. Even though only 2 bits are used for signaling, the D channel has 8 bits reserved on the TDM link since the QMC microcode must process data on 8-bit boundaries for correct delineation of channels. The unused 6 bits are masked in the QMC time slot assignment table.

Since the TDM line allows a maximum of 32 channels, the above process of routing channels to time slots (that is, the second B1 channel routed to TS3 and so on) can be repeated for up to 10 BRIs.

Using on-chip time slot assigners, the S/T and U interfaces in IDL2 mode can match the QMC bus structure—both interfaces can be connected to a 2.048-MHz IDL2 bus and route the B1 channel, B2 channel and D channel to any time slot.

Figure C-1 shows the IDL2 bus configured to match the QMC protocol.

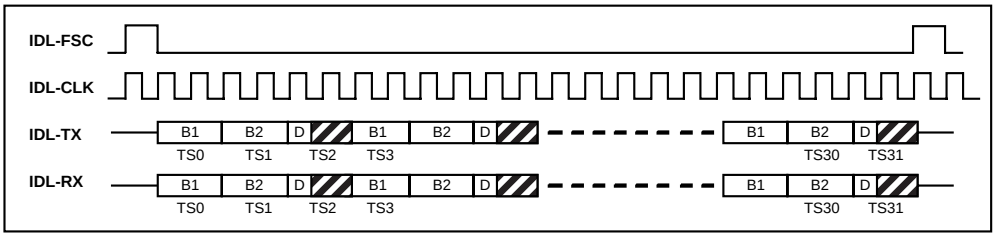


Figure C-1. IDL2 Bus Structure for a Connection to the QMC Bus

C.2 Control and Status Information

Using the SPI port, the QUICC32 and the ISDN interfaces exchange control and status information via out-of-band signaling. Optionally, the MC145572s could use an 8-bit parallel port for control and status transfer, allowing the U interfaces to be connected to the processor bus.

Figure C-2 shows the connection between the QUICC32 and an S/T interface.

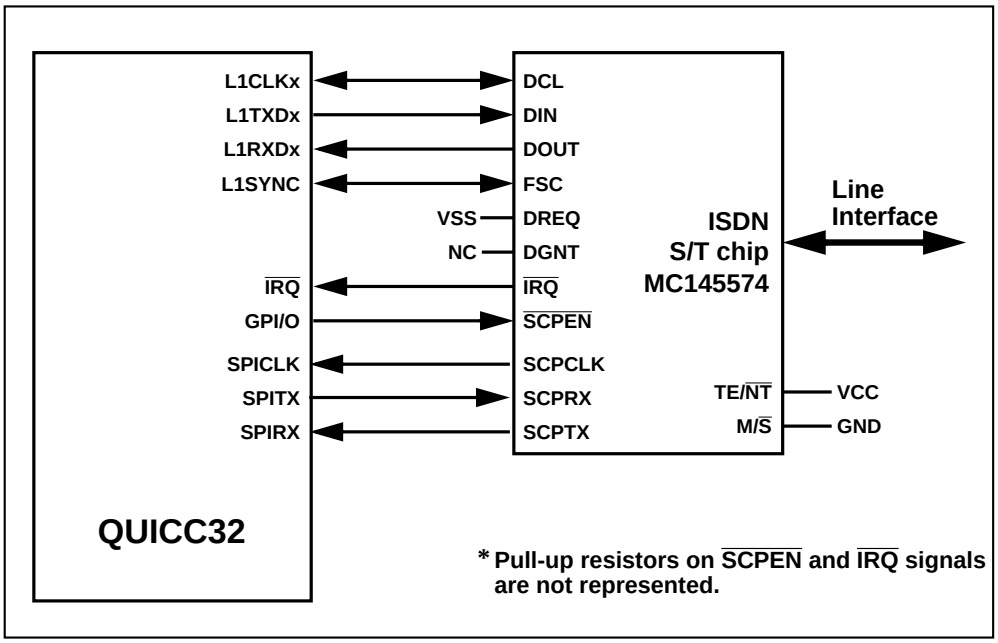


Figure C-2. IDL and SCP Connections between the QUICC32 and the S/T Interface

The QUICC32 is always a slave on the IDL bus, with the data clock (DCL) and frame sync (FSC) signals acting as inputs. As explained in Section C.3, “Data Clock (DCL) and Frame Sync (FSC) Generation,” this specific application requires the S/T interface be configured in slave mode on the IDL bus, with external circuitry providing the DCL and FSC signals to both the QUICC32 and the S/T interface.

The D channel request/grant function is not required for this application since each S/T is assumed to be directly connected to the NT1 (that is, point-to-point configuration with only one TE connected to one NT).

The DGNT signal is left unconnected, and DREQ can be directly connected to VSS. As there is no contention on the D channel, the D channel contention procedure of the MC145574 should be disabled by setting BR7[6].

Figure C-3 shows the connection between the QUICC32 and a U interface.

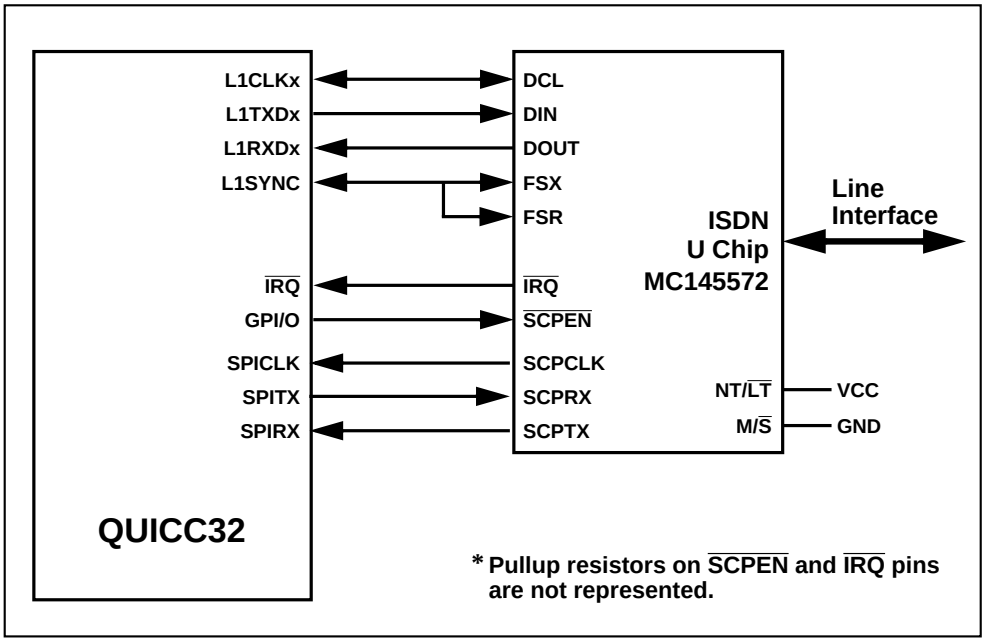


Figure C-3. IDL and SCP Connections between the QUICC32 and the U Interface

As with the S/T interface example, the U interface must be configured in slave mode with the DCL and FSC signals provided to both the QUICC32 and the U interface.

In slave mode, the FSX and FSR signals of the MC145572 must be connected together.

There are no DREQ/DGRANT signals on the U interface.

C.3 Data Clock (DCL) and Frame Sync (FSC) Generation

The main consideration when connecting multiple interfaces to the same physical medium is the clock and frame sync generation.

As stated earlier, the QUICC32 does not provide DCL and FSC signals. The U or S/T interface could be configured in master mode on the IDL bus and generate both DCL and FSC synchronized to the line-interface signal.

To have all the B- and D channels of a multi-interface system synchronized on the same TDM bus, only one S/T or U interface should be configured as master to provide common DCL and FSC signals to the QUICC32 and all other interfaces.

However, this configuration works only if the single master S/T or U interface is guaranteed to be active, constantly generating synchronized DCL and FSC signals for the network. Since the designated master transceiver cannot be guaranteed active, all transceivers must be configured in slave mode, and the problem of synchronization remains.

DCL and FSC must be generated from within the network and cannot be derived from an independent source (e.g., crystal or oscillator). However, the S/T and U interfaces, even in slave mode, provide a way to generate these signals as explained in the following sections.

C.3.1 MC145574 S/T Interface

To facilitate the generation of timing signals required by the slave IDL interface, TCLK is provided in the S/T interface. The TCLK signal will output a clock synchronized to the received data transmitted by the NT. That clock is output only when the S/T interface is active and when the DCL and FSC signals are present (both conditions are required simultaneously). This TCLK signal can be used to provide network timing. Its frequency is selectable via the SCP.

The TCLK signal of the MC145574 is enabled by setting OR7[5]. BR13[5] and BR7[2] determine the TCLK frequency as shown in Table C-1.

Table C-1. TCLK Frequencies Selected by BR13[5] and BR7[2]

BR13[5]	BR7[2]	TCLK
0	0	2.56 MHz
0	1	2.048 MHz
1	0	1.536 MHz
1	1	512 KHz

A TCLK signal configured as a 2.048-MHz clock can be used as the DCL for the IDL bus.

Dividing the TCLK signal by 256 provides the 8-KHz frame sync FSC. Since the FSC must have a pulse between one DCL and eight DCLs in width, additional logic may be needed after the divider to generate a signal with a correct duty cycle.

Figure C-4 and Figure C-5 show a schematic and a timing diagram, respectively, for a logic design used to generate a 8-KHz FSC with a 1-DCL width pulse from a 2.048-MHz clock (TCLK).

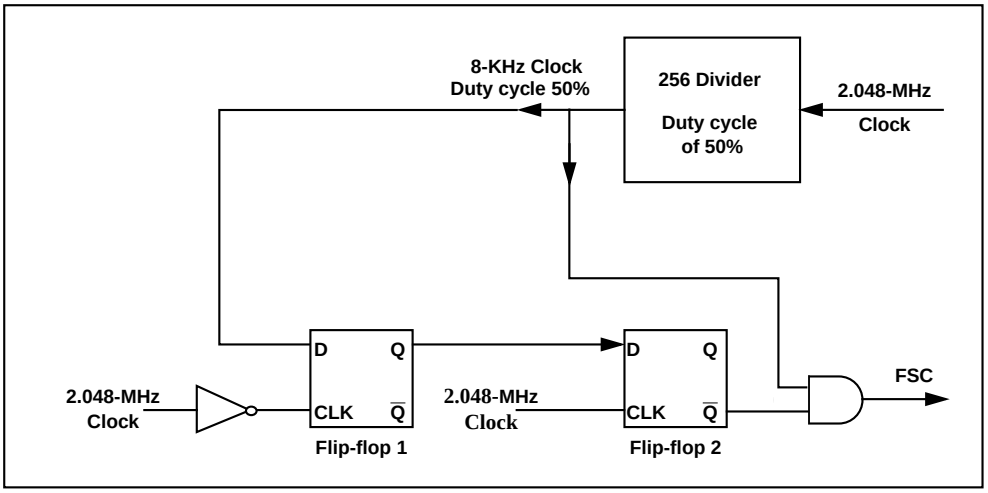


Figure C-4. FSC Generation from a 2.048-MHz Clock—Block Diagram

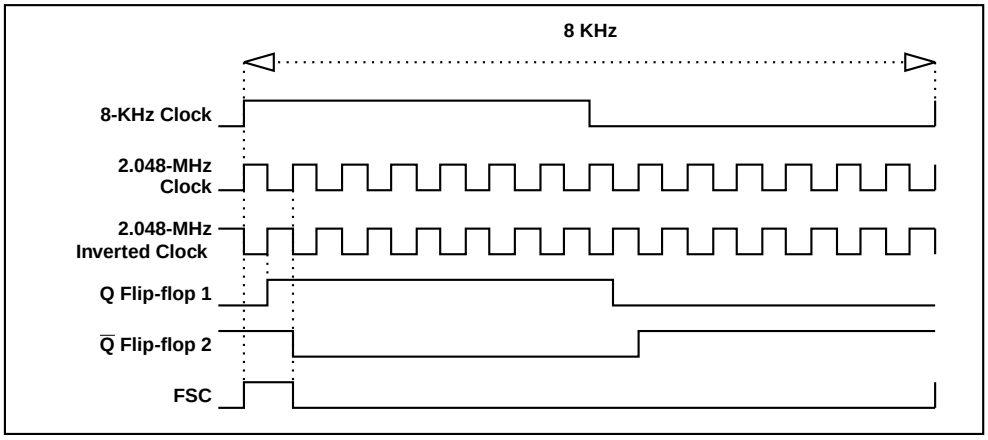


Figure C-5. FSC Generation from a 2.048-MHz Clock—Timing

The S/T interface includes elastic buffers allowing continued operation under any phase relationship between the IDL frame sync and the network. These buffers allow the frame sync to wander with respect to the network up to 60 µs peak-to-peak, exceeding the Q.502 requirements of 18 µs peak-to-peak over a 24-hour period.

Figure C-6 shows a block diagram of the connection between four S/T interfaces and the QUICC32. The diagram would be the same for up to 10 S/T interfaces.

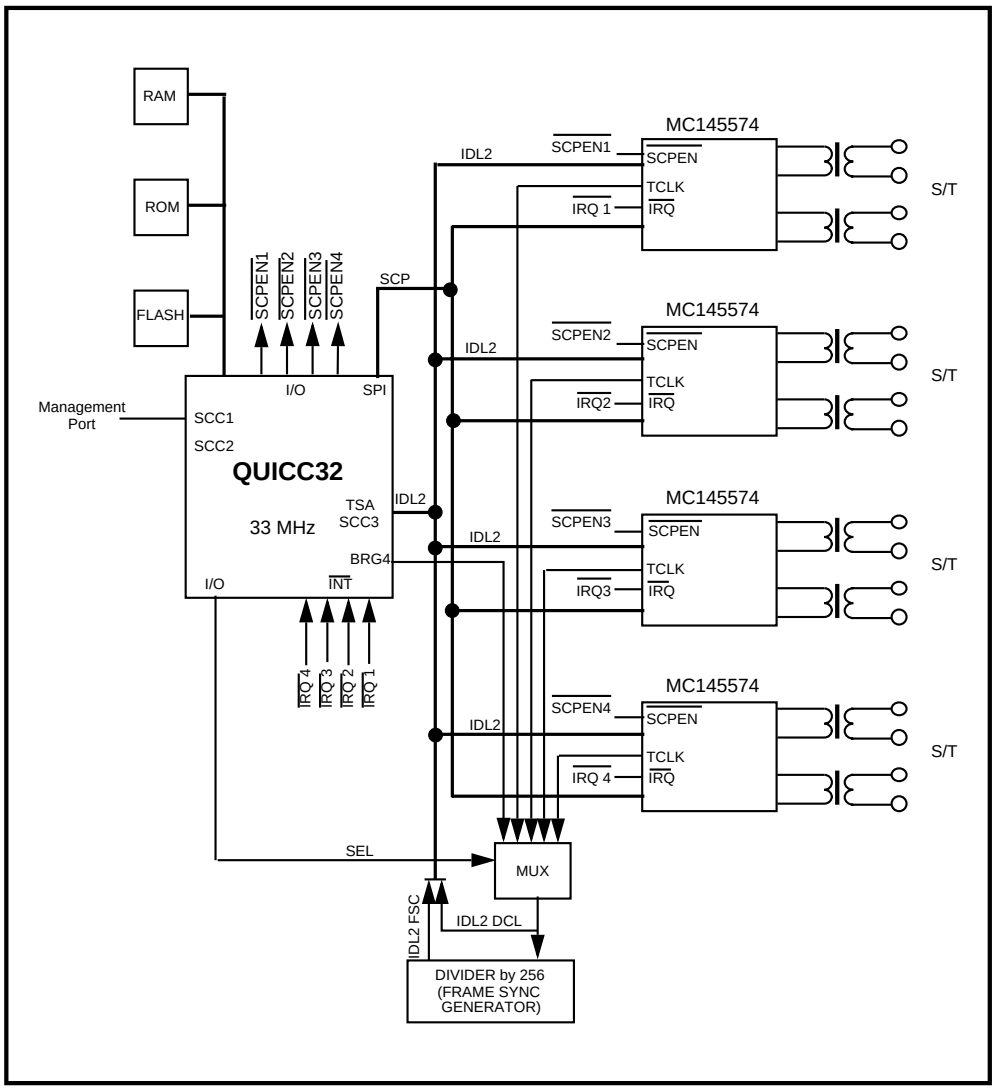


Figure C-6. Connection between Four S/T Interfaces and the QUICC32

C.3.1.1 Activation Procedure

If no S/T transceiver is active, no TCLK clock is generated. Once the first transceiver is activated, it will generate a TCLK signal only if DCL and FSC signals are present as well.

Pseudo DCL and FSC signals generated from one of the baud rate generators (BRG) of the QUICC32 can be used to generate the TCLK signal. The BRG can generate a clock based on the QUICC32's system clock. A divider factor should be chosen so that the BRG frequency is close to 2.048 MHz. This clock then feeds into the 256-divider circuitry of Figure C-4 to generate a pseudo DCL and a pseudo FSC.

A multiplexer commanded by the QUICC32 is required to select either the BRG signal or the TCLK signals of the transceivers to be the clock master generating the DCL and FSC signals.

When no transceiver is activated, the QUICC32 selects the BRG to be the clock master, and the S/T interface receives the pseudo DCL and FSC signals. (These two signals are not synchronized to the network but are not used to sample data.)

As the first MC145574 is activated, it will be able to generate the TCLK signal; see Figure C-7. This transceiver will then send an interrupt to the QUICC32 (IRQ3—register NR3[3]—meaning Info 2 has been received) indicating that the activation process has begun. The QUICC32 then uses the multiplexer to select the TCLK signal of that MC145574 to be the clock master.

As shown in Figure C-7 and Figure C-8, the TCLK signal is present before the interruption, with at least 750 µs between the IRQ and the received Info 4. The QUICC32 therefore has 750 µs to react to the IRQ and to select the new clock master.

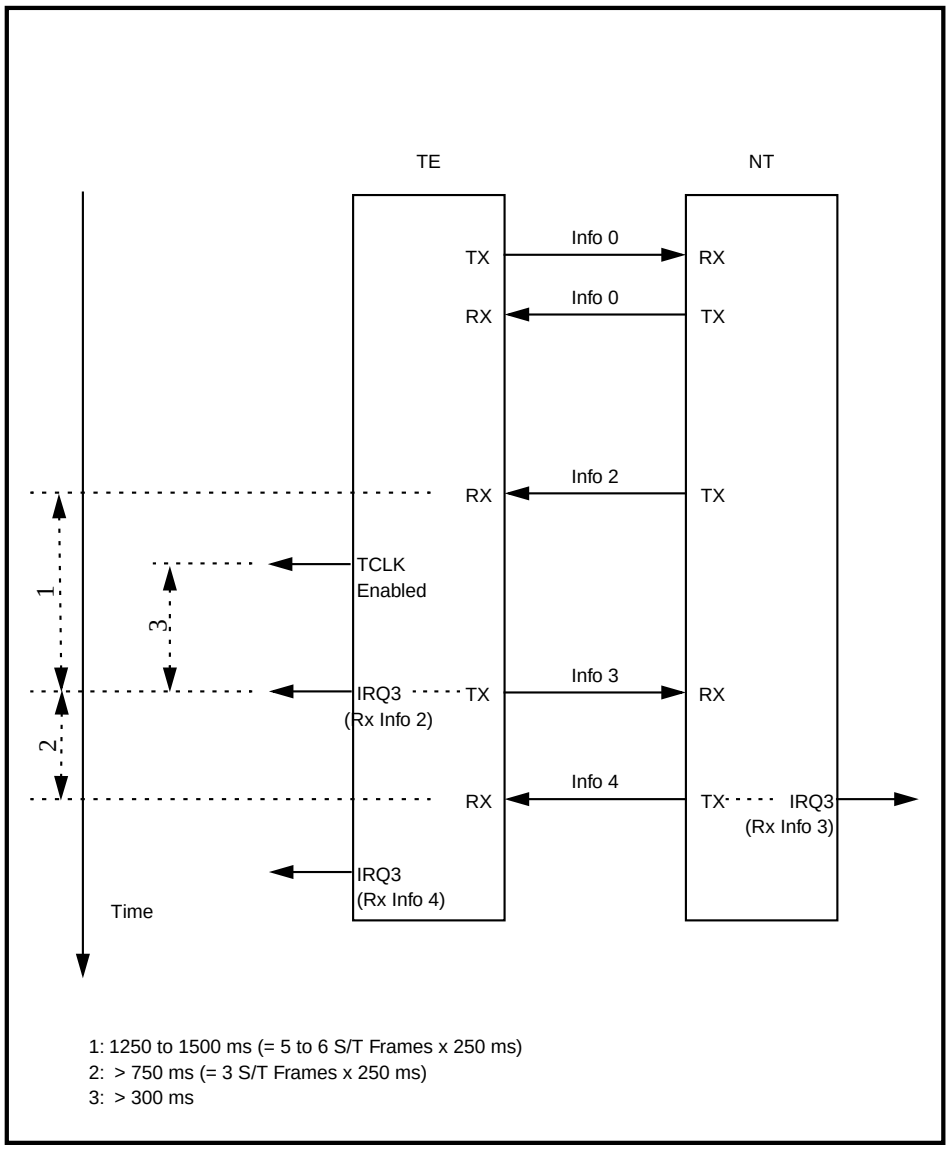


Figure C-7. Timing Diagram for an Activation Initiated by the NT

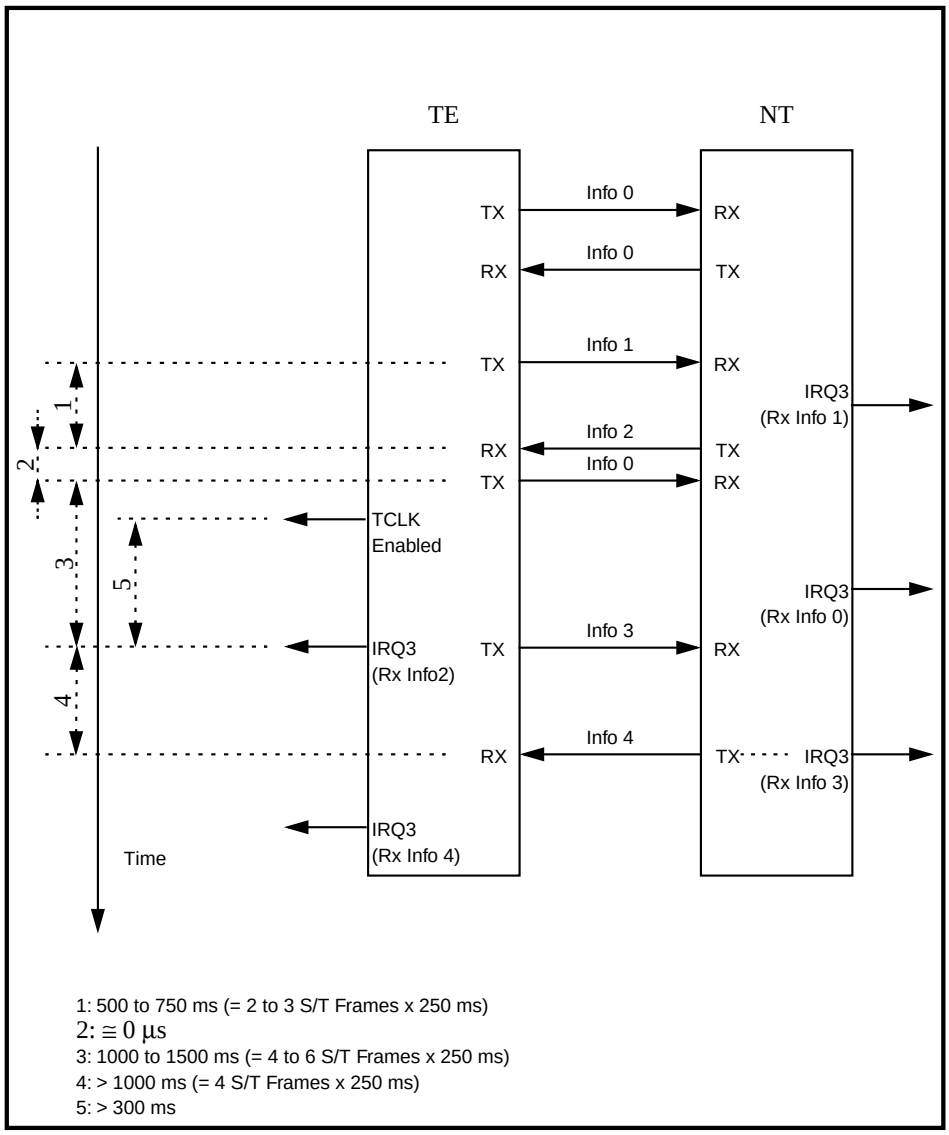


Figure C-8. Timing Diagram for an Activation Initiated by the TE

C.3.1.2 Deactivation Procedure

When the clock-master S/T interface is deactivated, the QUICC32 receives an interrupt indicating the deactivation status (IRQ3 —register NR3 bit 3— meaning Info 0 of Figure C-9 has been received). Then, if another S/T interface is active, its TCLK signal must be selected to become the clock master; otherwise, the QUICC32 can select the BRG to be the clock master.

As shown in Figure C-9, the TCLK signal is disabled about 72.8 µs after the interruption. Therefore, the QUICC32 has 72.8 µs to react to the IRQ and to select another clock master.

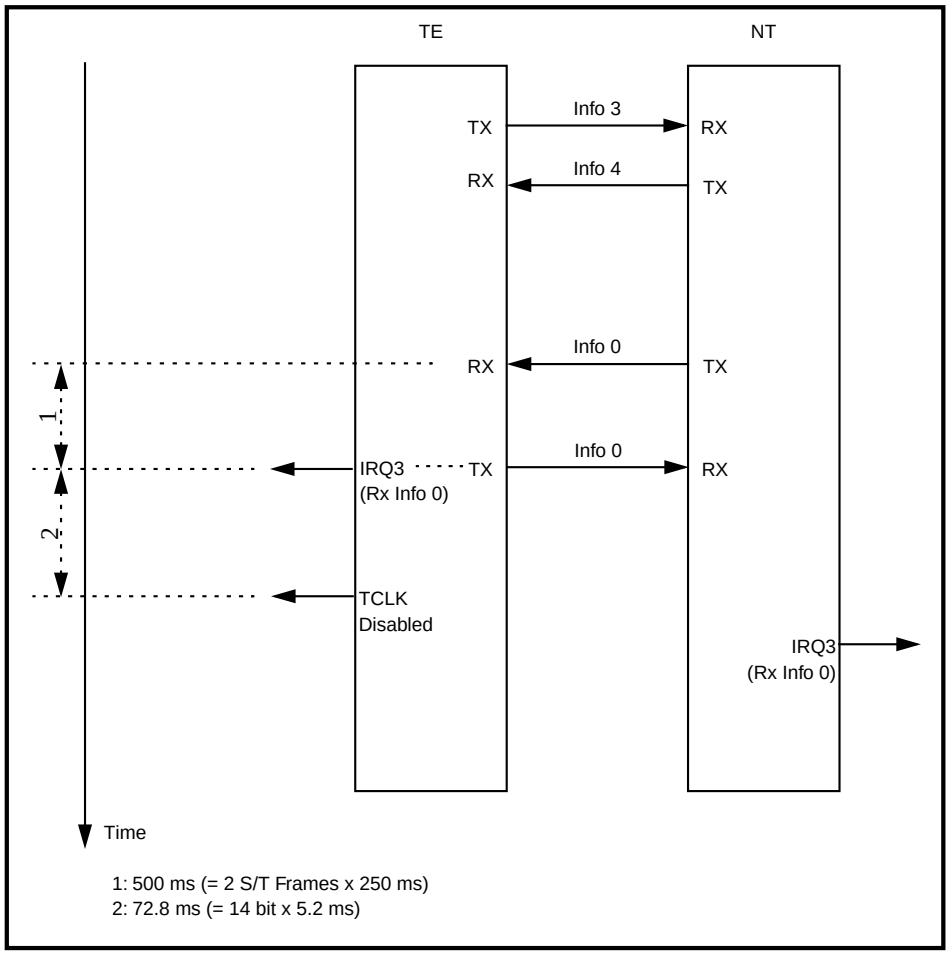


Figure C-9. Timing Diagram for a Deactivation (Always Initiated by the NT)

C.3.2 MC145572 U Interface

The FREQREF signal of the MC145572 provides a clock synchronized to the network timing for the U interface. This frequency reference is a fixed 2.048-MHz clock enabled by setting OR8[4] in the MC145572 register set.

The U interface's FREQREF differs from the TCLK of the S/T interface. When enabled, the FREQREF signal generates the 2.048-MHz clock regardless of the activation status of the U interface (but that clock is synchronized to the network only when the U interface is activated). Also, FREQREF does not require the DCL and FSC signals to enable clock generation.

Like the S/T interface, on the other hand, the FREQREF signal can be used as the DCL for the IDL bus. When divided by 256, FREQREF can also be used to generate the 8-KHz frame sync FSC. The same logic design used for the S/T interface must be added to insure a correct FSC duty cycle; see Figure C-4.

Also like the S/T interface, elastic buffers are included to allow the U interface to operate with any phase relationship between the IDL frame sync and the network.

Figure C-10 shows a block diagram of the connection between four U interfaces and the QUICC32. The diagram would be the same for up to 10 U interfaces.

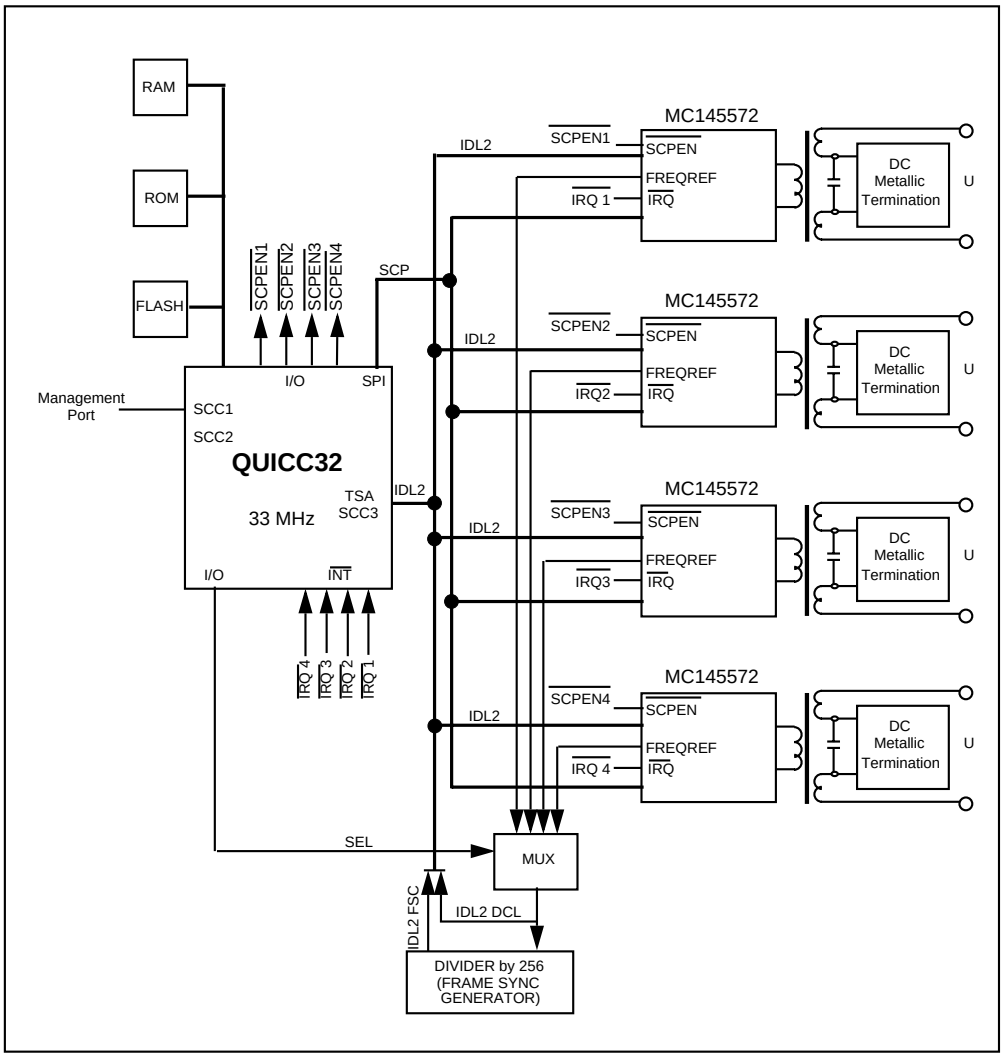


Figure C-10. Connection between Four U Interfaces and the QUICC32

C.3.2.1 Activation Procedure

During the initialization, the FREQREF signal of each U interface is enabled.

A multiplexer commanded by the QUICC32 is used to select the U interface clock master.

When the first MC145572 is activated, its FREQREF signal synchronizes to the network. The MC145572 then sends an interrupt to the QUICC32 (IRQ1— register NR3[1]— meaning uao = 1 has been received) indicating that the activation process has begun. Before responding to LT with act = 1 (which will enable the data transfer), the QUICC32 can select, through the multiplexer, this particular FREQREF signal to be the clock master.

Since the QUICC32 has the initiative to enable the data transfer, there is no timing constraint to react to the interrupt.

C.3.2.2 Deactivation Procedure

According to the ANSI specification T1.601-1988, prior to deactivating, the LT should notify the NT of the pending deactivation by clearing the M4 channel dea bit towards the NT for at least three superframes. Then, the NT can be deactivated by sending a deactivation request.

The MC145572 not only has the ability to generate an interrupt after the reception of the third dea bit = 0, but also after the reception of the second dea bit = 0.

When the clock-master U interface is deactivated, the QUICC32 receives an interrupt indicating that the second dea bit = 0 has been received. The QUICC32 has then the ability to select another activated U interface (if there is one), to be the clock master. The QUICC32 has 12 ms (1 superframe) until receiving the next dea bit = 0, indicating the pending deactivation, and therefore 12 ms to react to the interrupt.

If none of the U interfaces are activated, no change in the multiplex selection is required.

C.3.3 System Configuration

The following sections provide a checklist of the main features that need to be configured for each device.

C.3.3.1 S/T-Interface Configuration

Do the following for an S/T-interface configuration:

- IDL2 with time slot assigner (TSA enabled in reg. OR6[5–7]; TSA selection in OR0 to OR5)
- Slave mode (DCL & FSC are input) - (pin M/$\overline{S}$ to GND)
- TCLK enabled at 2.048 MHz (OR7[5] = 1; BR13[5] = 0; BR7[2] = 1)
- D channel contention procedure disabled (BR7[6] = 1)

C.3.3.2 U-Interface Configuration

Do the following for U-interface configuration:

- IDL2 with time slot assigner (TSA enabled in reg. OR6[5–7]; TSA selection in reg. OR0 to OR5)
- Slave mode (DCL & FSC are input) - (pin M/$\overline{S}$ to GND)
- FREQREF enabled at 2.048 MHz (reg. OR8[4] = 1)

C.3.3.3 QUICC32 Configuration

Do the following for QUICC32 configuration:

- SCC3 using the QMC protocol for handling the different channels of the multiplexed IDL2 bus. (D channels are HDLC encoded/decoded and B-channels can be configured for transparent or HDLC framing)
- SCC1 can be configured for Ethernet, HDLC, transparent, or UART.
- SCC2 and SCC4 can be configured for HDLC, transparent, or UART.
- The SPI is connected to the SCP port of each S/T or U interface for handling configuration and control information.
- For the U interface, the SPI/SCP connection can be replaced by a connection of the 8-bit parallel port of the U transceiver to the processor bus of the QUICC.
- One I/O signal can be dedicated for handling the $\overline{SCPEN}$ signal of each S/T or U interface.
- One interrupt signal can be dedicated for handling the $\overline{IRQ}$ signal of each S/T or U interface.

INDEX

INDEX

INDEX

INDEX