Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

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

Applications of Embedded - Microprocessors

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

Details	
Product Status	Obsolete
Core Processor	CPU32+
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	Communications; CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10Mbps (1)
SATA	-
USB	-
Voltage - I/O	3.3V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	240-BFQFP
Supplier Device Package	240-FQFP (32x32)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68mh360ai25vl

TABLES

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.

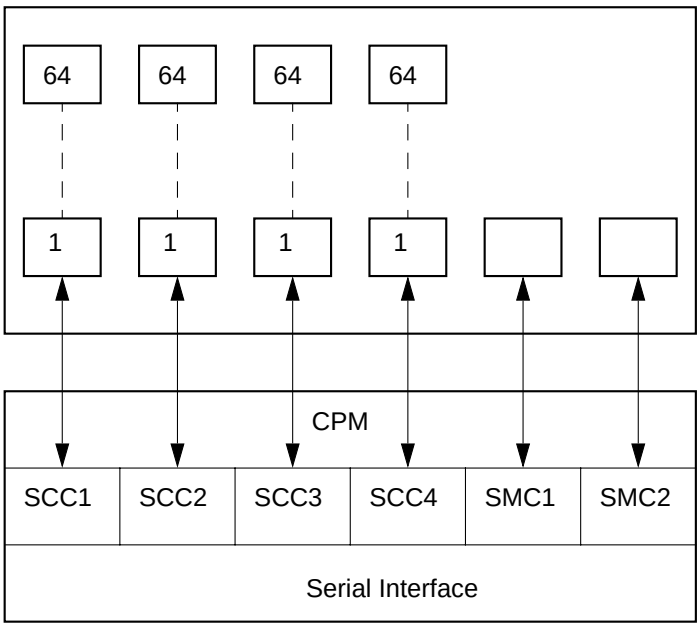


Figure 1-1. QMC Channel Addressing Capability

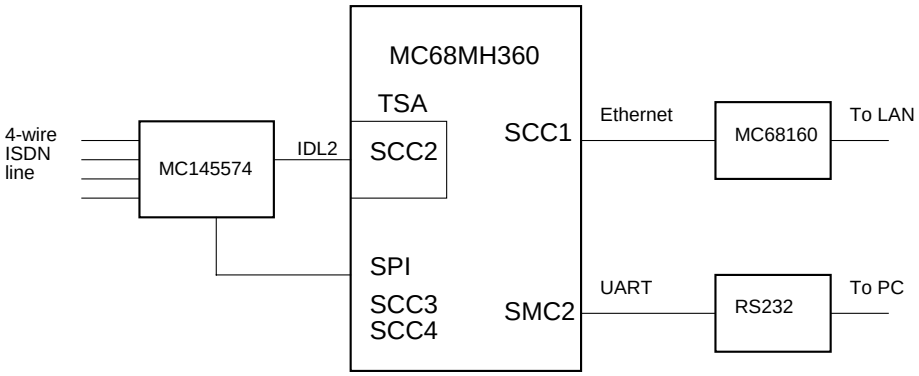


Figure 1-4. Ethernet-to-BRI Bridge Using MC68MH360

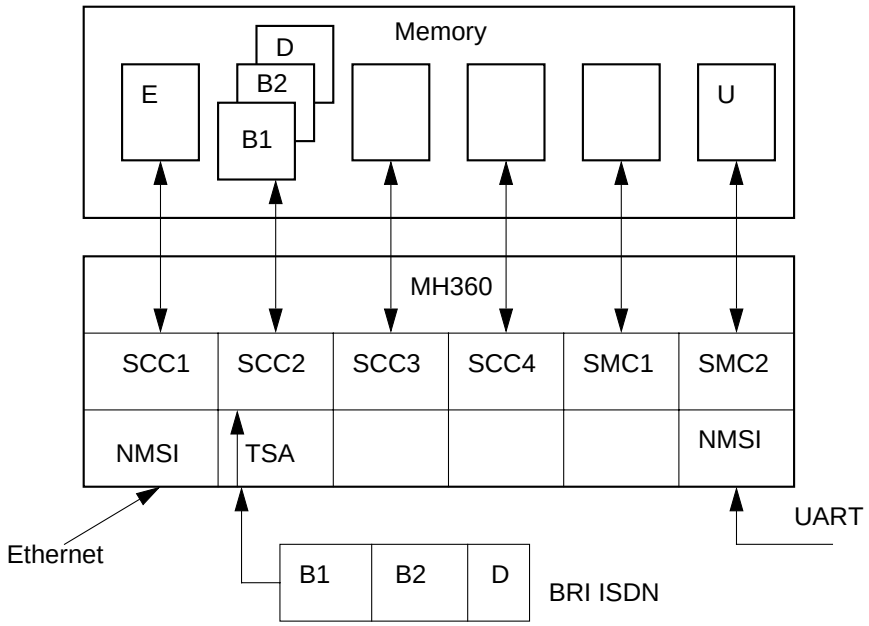


Figure 1-5. Internal Routing for Ethernet-to-BRI Bridge Using MC68MH360

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.

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.

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.

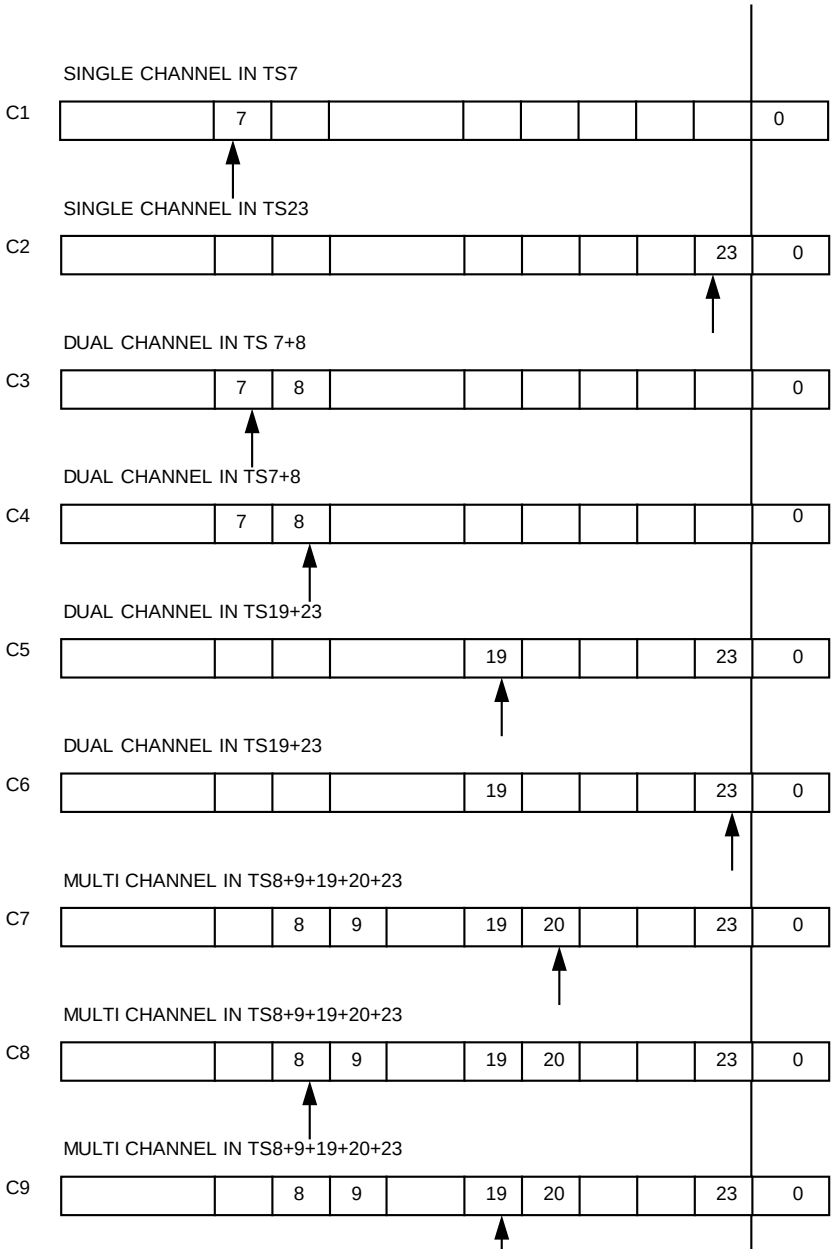


Figure 2-14. Examples of Different T1 Time Slot Allocation

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.

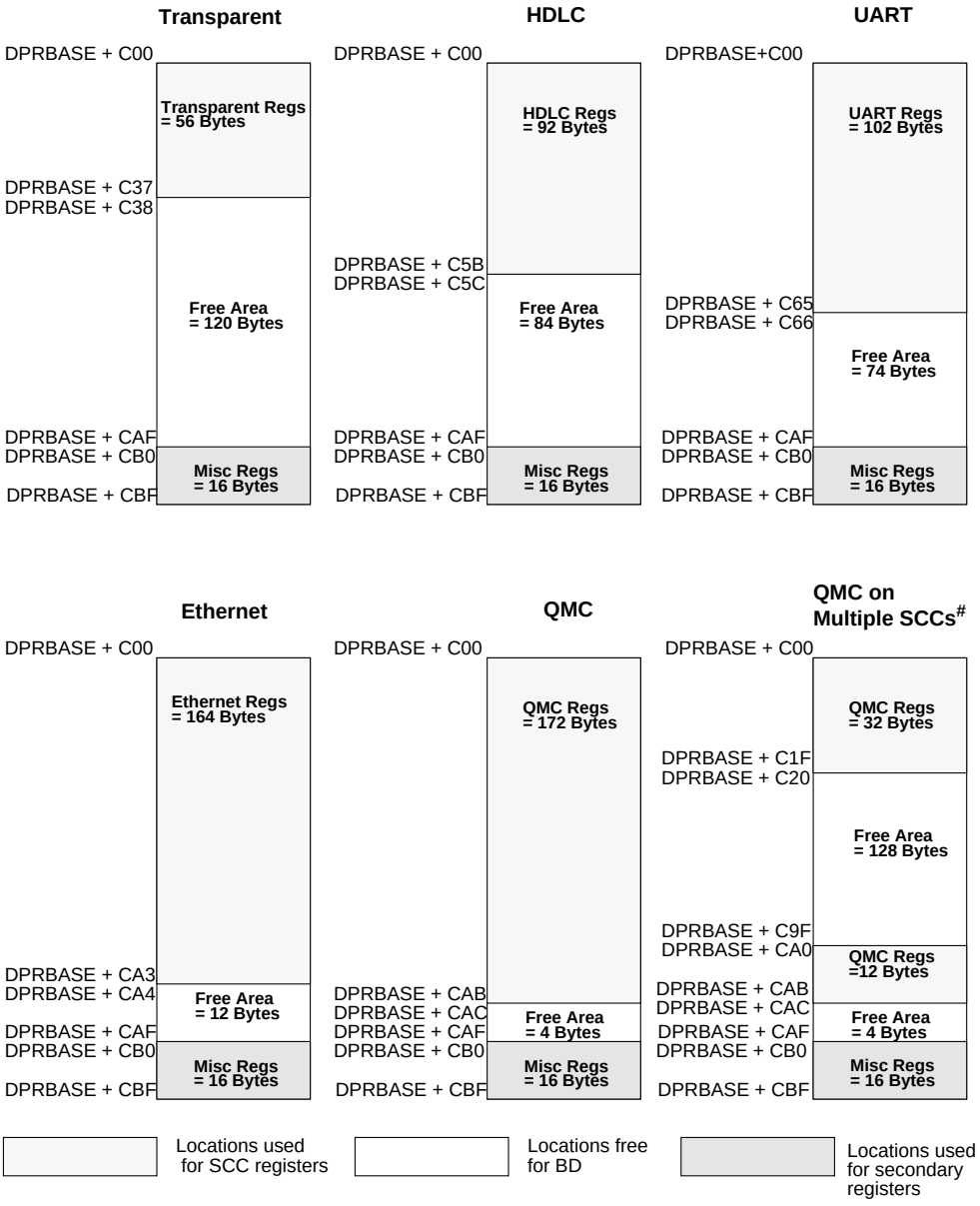
Table 2-15. RSTATE Field Descriptions for 860MH (Transparent Mode)

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.

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.



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

Figure 5-7. MC68MH360 SCC1 Parameter RAM Usage

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

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

```
      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 */
```

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 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.

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

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.

INDEX