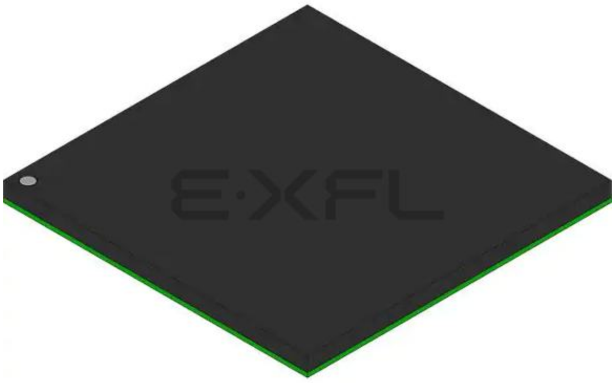
Freescale Semiconductor - MC68MH360VR25VL Datasheet



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	Active
Core Processor	CPU32+
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	Communications; CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10Mbps (1)
SATA	-
USB	-
Voltage - I/O	3.3V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	357-BBGA
Supplier Device Package	357-PBGA (25x25)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68mh360vr25vl

TABLES

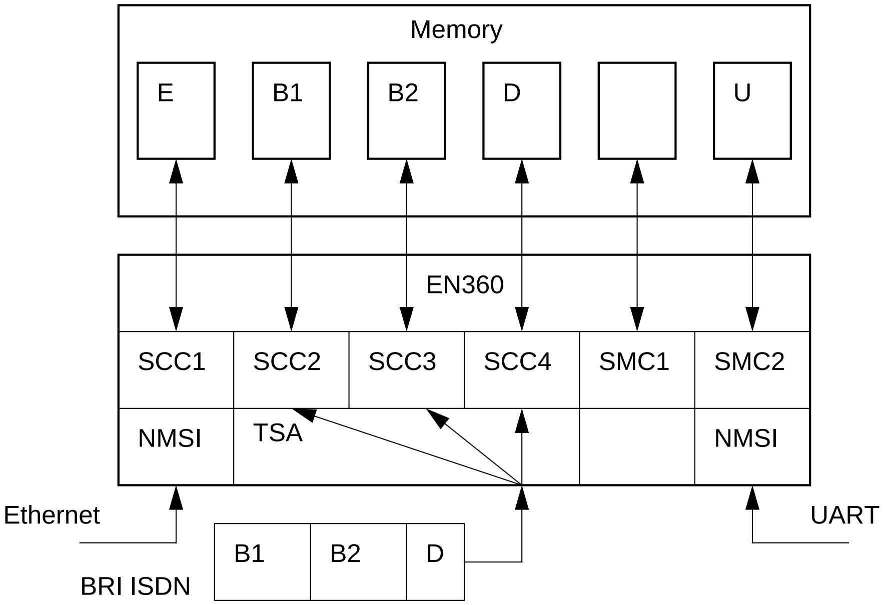


Figure 1-3. Internal Routing for Ethernet-to-BRI Bridge Using MC68EN360

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.

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

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.

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 2-10. 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.

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.

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 5-2. 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.

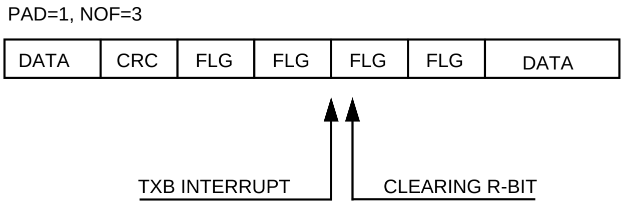


Figure 5-4. Relation between PAD and NOF

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.

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 */
      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;
```

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.

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.

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

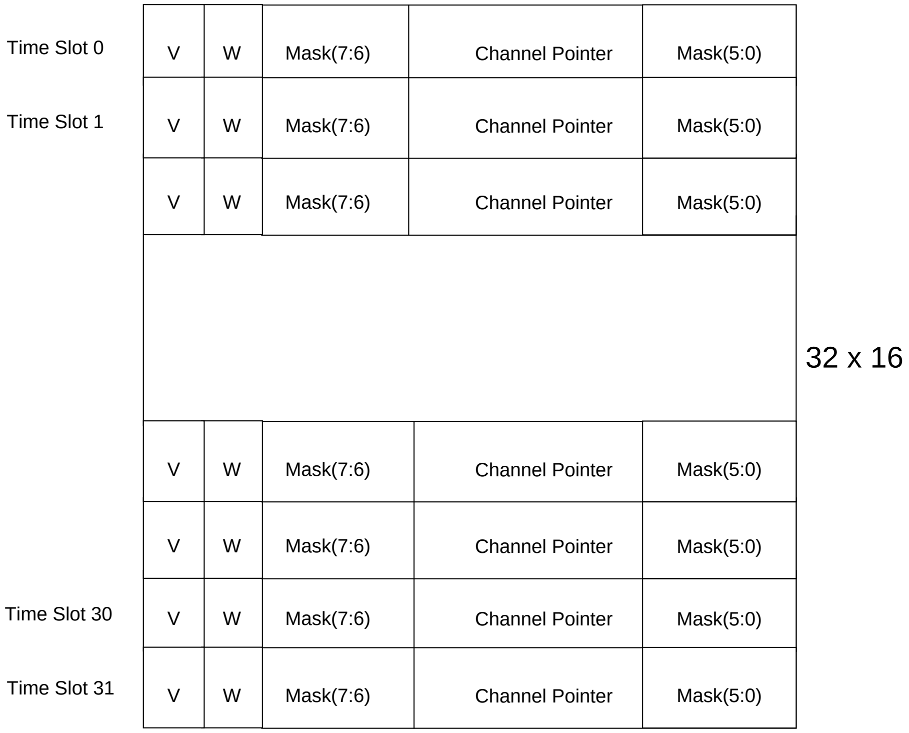
The example in Figure 9-4 uses two time slots to handle eight 2-bit subchannels. Time slot 0 is subdivided into four 2-bit subchannels. Note that time slot 0 is processed four times for the channels labeled 0A, 0B, 0C and 0D, each with different masks. It is only in the fourth entry that the L-bit (last 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.

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