

Time-Division Multiplexing MCC Channels on the MSC8101 Device

By Barbara Johnson

Unlike the other serial communication controllers on the Freescale MSC8101 device, the two multichannel controllers (MCCs) do not have their own set of pins, and communication occurs through the time-division multiplex (TDM) buses. This application note discusses how to use the MCC and the serial interface with the time-slot assigner (TSA) to multiplex data on the TDM. After a review of TDM basics, an example illustrates how to multiplex MCC channels on a TDM bus.

Specifically, this document describes how to multiplex 25 MCC2 channels on TDMC2. The TDM frame repeats every 125 μ sec and consists of 64×8 -bit time slots yielding a 4.096 Mbps data rate. The first 12 time slots are idle. No data is sent during these time slots. Channel 128 is allocated the next four time slots. Channels 129–152 are assigned two consecutive time slots each. HDLC mode is used and data is looped back internally at the TDM so that the TDM transmitter output internally connects to the TDM receiver input.

CONTENTS

1	Time-Division Multiplexing Basics	2
2	Initialize the SIx RAM	3
2.1	SI2 RAM Entries for Idle Slots	4
2.2	SI2 RAM Receive Entries for MCC2 Channels	5
2.3	SI2 RAM Transmit Entries for MCC2 Channels	6
3	Initialize the SI Mode Registers	11
4	Connect the Clock Pins to the TDMs	12
5	Configure the TDM Pins	12
6	Issue a STOP Command	14
7	Initialize the MCC	15
7.1	MCC Global Parameters	16
7.2	MCC Channel Extra Parameters	18
7.3	MCC Channel-Specific Parameters	19
7.4	INIT Command	20
7.5	MCC Interrupts	20
7.6	Buffer Descriptors	22
8	Enable the TDM	24
9	Enable the MCC Channels	24

Figure 1 shows the example TDM channel assignment.

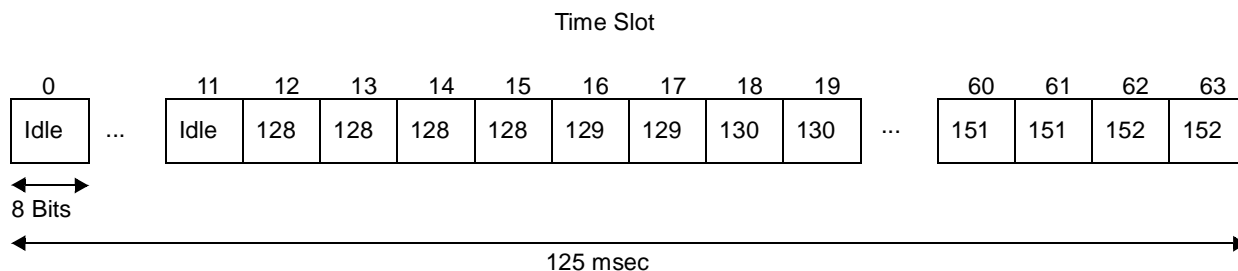


Figure 1. Example of 25 MCC Channels on TDMC2

Multiplexing 25 MCC2 channels on TDMC2 requires the following steps, and, after a brief review of TDM basics, the remainder of this application note describes each of these steps in detail:

1. Initialize the SIx RAM for desired routing.
2. Initialize the SI registers.
3. Connect the clock pins to the TDMs.
4. Configure the TDM pins.
5. Issue a STOP command.
6. Initialize the MCC.
7. Enable the TDM.
8. Enable the MCC channels.

1 Time-Division Multiplexing Basics

The TDM interface enables many devices to communicate over a single bus. A TDM periodically generates a frame consisting of a fixed number of time slots that are constant in length. Each communications channel is allocated a time slot, and it can transmit data only during its allocated time slot(s). If a channel must send more data, it must wait until its next assigned time slot is available. If it has no more data to send, the time slot is sent empty. Figure 2 illustrates how a TDM works on the T1 and E1 lines used in North America and Europe, respectively. A speech waveform is sampled 8000 times per second, and each sample is represented as 8 bits, yielding a 64 Kbps channel capacity or Digital Signal 0 (DS0). On a T1 line, 24 DS0s are time-division multiplexed into a frame. Each frame represents one sample of 8 bits from each of the 24 channels. The frame is repeated every 125 μsec. An additional bit for frame signaling and synchronization creates a T1 frame of 193 bits: (24 channels) × (8 bits per channel) + framing bit = 193 bits. Therefore, the T1 interface yields an aggregate of 193 bits per 125 μsec = 1.544 Mbps. The E1 line can multiplex 32 DS0s and consists of 256 bits: (32 channels) × (8 bits per channel) = 256 bits. Therefore, the E1 interface yields an aggregate of 256 bits per 125 μsec = 2.048 Mbps.

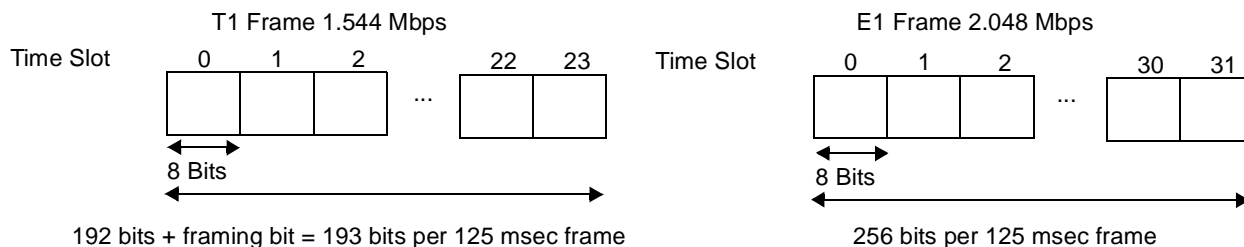


Figure 2. T1/E1 Frame

Four TDM buses are available on the MSC8101 device: TDMA1, TDMB2, TDMC2, and TDMD2. The TSA allows combinations of the following serial controller channels to be multiplexed on these buses:

- Multichannel controllers (MCC1 and MCC2)
- Fast communication controllers (FCC1, FCC2, and FCC3)
- Serial communication controllers (SCC1, SCC2, SCC3 and SCC4)
- Serial management controllers (SMC1 and SMC2)

The TSA routes the channels from the serial interface 1 (SI1) and 2 (SI2) RAM entries. SI1 RAM can route data from any serial controller, except MCC2 to TDMA1. SI2 RAM can route data from any serial controller, except MCC1 to TDMB2, TDMC2, and TDMD2. **Figure 3** shows the combination of serial controllers that are routed to the TDMs.

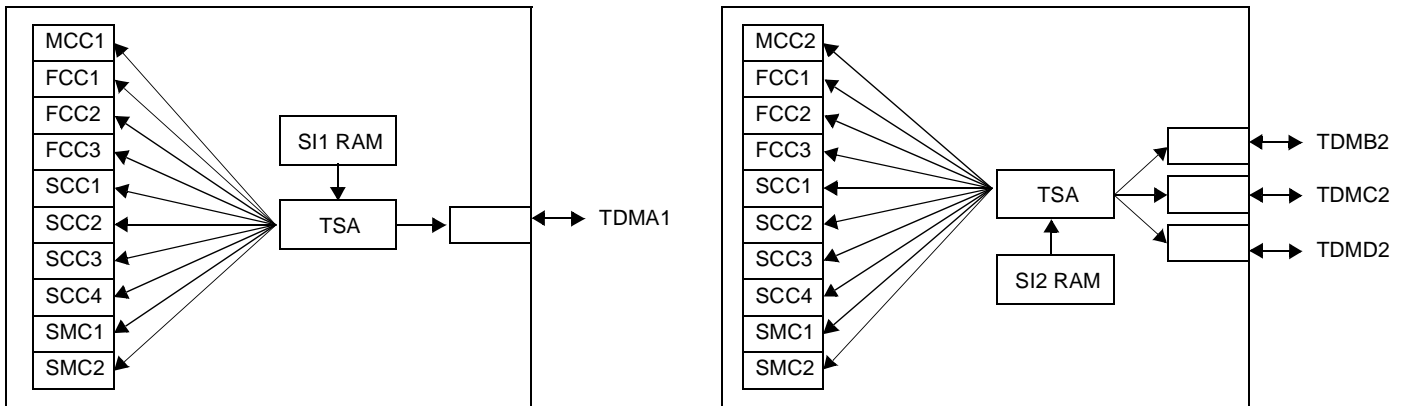


Figure 3. SI with TSA Block Diagram

2 Initialize the SIx RAM

SI1 RAM routes channels to TDMA1, and SI2 RAM routes channels to TDMB2, TDMC2, and TDMD2. **Figure 4** shows the structure of the SIx RAM. Each SIx RAM is divided into transmit and receive sections of 256 16-bit entries. Each section is divided into four blocks of 64 entries. SI1 and SI2 RAM are located in the internal memory space. The base address of the internal memory space is programmed in the Internal Memory Map Register (IMMR):

- SI1 RAM transmit is located at an offset of 0x12000 from the IMM base address, and the SI1 RAM receive is located at an offset of 0x12400 from the IMM base address.
- The SI2 RAM transmit is located at an offset of 0x12800 from the IMM base address and the SI2 RAM receive is located at an offset of 0x12C00 from the IMM base address.

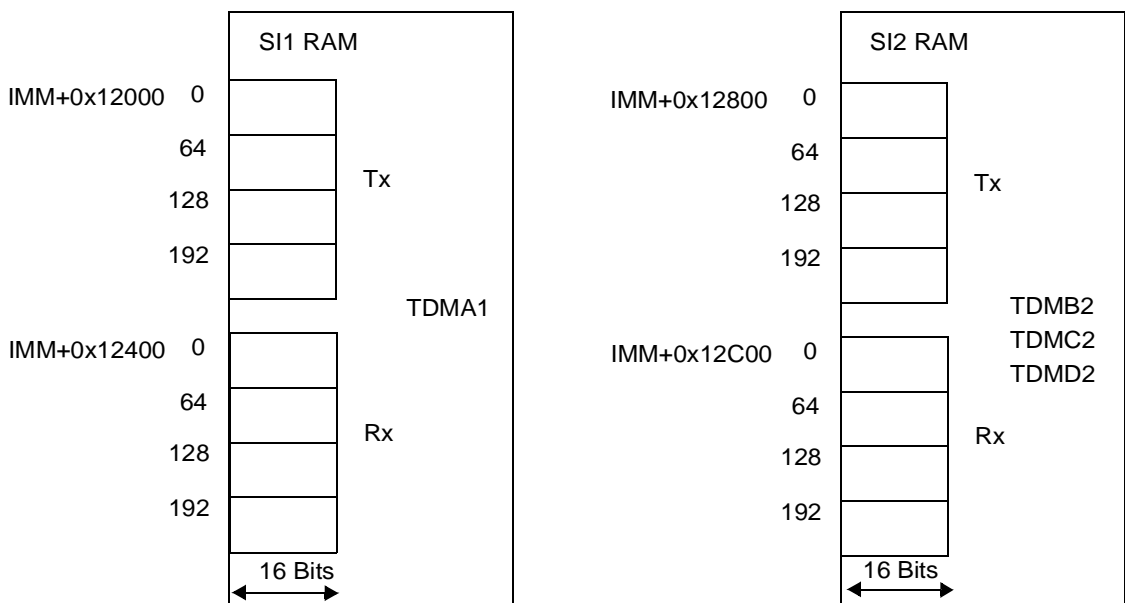


Figure 4. SIx RAM Structure

In this example, only SI2 RAM is programmed since the channels are multiplexed on TDMC2. **Figure 5** shows the two different SIx RAM entry functions. The bit definition of the SIx RAM entry¹ differs between non-MCC channels and MCC channels. When FCC, SCC, or SMC channels are routed, the MCC bit is cleared in the first SIx RAM entry definition. However, when MCC channels are routed, the second MCC SIx RAM entry definition is used, and the MCC bit is set. Our example employs the non-MCC SIx RAM entry function to define the idle slots and the MCC SIx RAM entry function to define the 25 MCC2 channels.

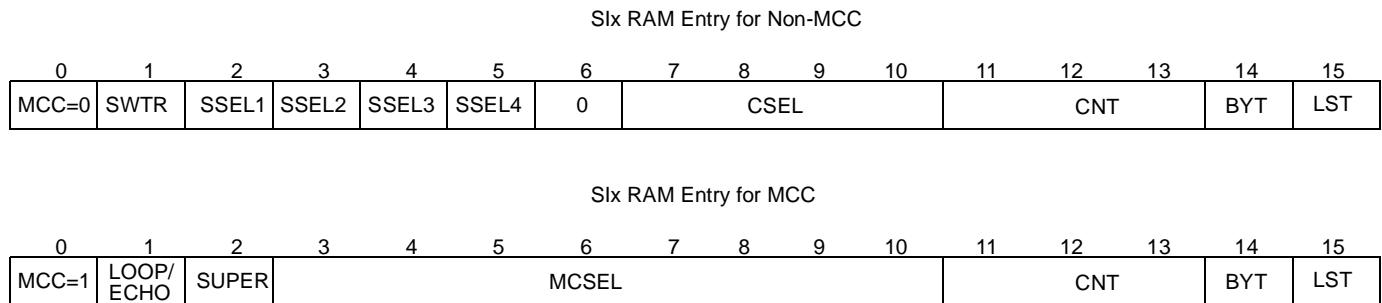


Figure 5. SIx RAM Entry

2.1 SI2 RAM Entries for Idle Slots

The only fields in the SIx RAM entry to be initialized for idle time slots are the CNT and BYT fields. The remaining fields are written with zeros. The CNT field indicates the number of bits or bytes minus one that are controlled by the entry. The BYT field specifies whether the CNT field indicates the number of bits or bytes, as shows in the following examples:

- CNT = 000 and BYT = 1 indicates 1 byte
- CNT = 111 and BYT = 0 indicates 8 bits

1. SIx RAM entry programming is discussed in the *MSC8101 Reference Manual*, in the chapter on the Serial Interface with Time-Slot Assigner.

A maximum of eight bytes is allowed per entry. The number of SIx RAM entries must be even. There are several ways to define the 12 bytes of idle time slots. For example, we can define one entry per idle time slot, which requires twelve entries. Another option is to minimize the number of entries by using one entry to define eight idle time slots and a second entry to define four other idle time slots. The example discussed here uses the second option. To use only two entries, the first entry is programmed with CNT = 111 and BYT = 1 to define the first eight bytes of idle time slots. The second entry is programmed with CNT = 011 and BYT = 1 to define the next four bytes of idle time slot. The SI2 RAM entries for the idle time slots are programmed to be 0x001E and 0x000E for the first and second entries. Since the TDM is operating in TDM loopback mode, these entries apply to both the SI2 RAM transmit and SI2 RAM receive entries.

Because the SI2 RAM entries for the MCC channels follow the two idle time slot entries, the LST bit is cleared. The LST bit allows the SI to wait for the next frame sync signal to start. Because the LST bit is active only in odd-numbered entries and the entry count starts with zero, there must be an even number of entries. **Figure 6** shows SI2 RAM transmit and receive entries for the idle time slots.

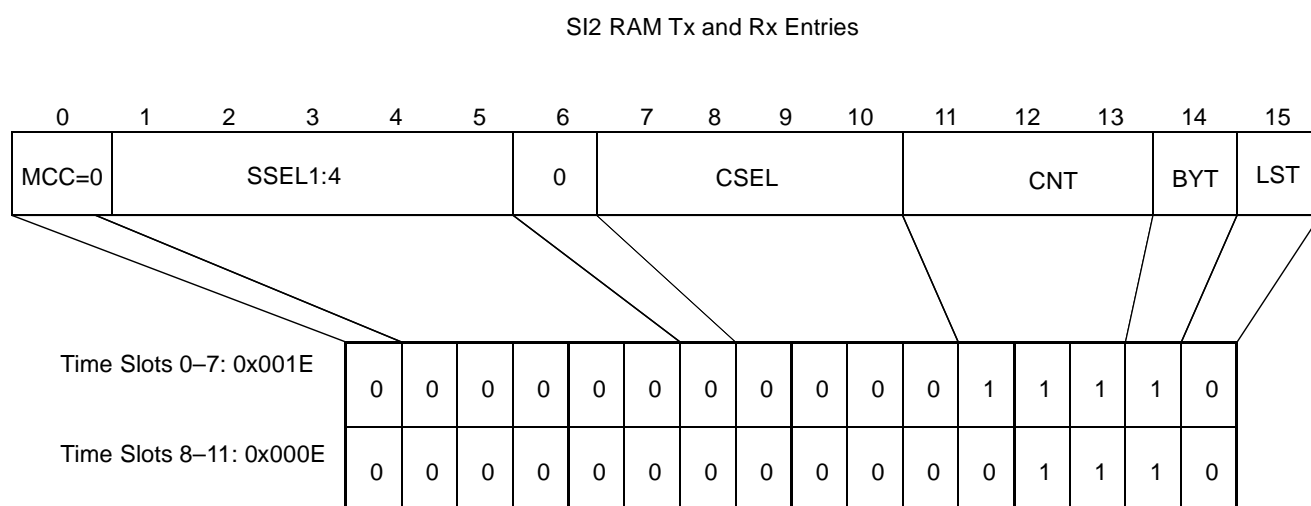


Figure 6. SI2 RAM Receive and Transmit Entries for Idle Time Slots

2.2 SI2 RAM Receive Entries for MCC2 Channels

All 25 channels on TDMC2 are MCC channels, so the MCC SIx RAM entry function defines the MCC2 channels. The SI2 RAM entries for time slots 12 through 63 must have the MCC bit set to select MCC. **Figure 5** shows the SIx RAM entry definition. Since channel loopback is not used, the LOOP/ECHO bit is cleared. Note that channel loopback is different from TDM loopback. In channel loopback, the MCC channel’s transmitted data loops back at the SI. In TDM loopback, transmitted data loops back at the physical TDM of the MSC8101.

Each of the 25 MCC channels is a super-channel and therefore is allocated more than one time slot. Normally, each time slot is allocated to a channel. However, if a subscriber requires more than one time slot, then a super-channel consisting of two or more time slots can be allocated. When the MCC operates in HDLC or Transparent mode without slot synchronization, the super-channels can be programmed in the SI2 RAM receive as normal or non-super-channels. Therefore, a super-channel table is not required. This example uses HDLC mode so the SUPER bit is cleared in the SI2 RAM receive entries. Programming the SI2 RAM transmit is handled differently for super-channels.

The MCSEL bits indicate the MCC channel associated with the SIx RAM entry. For MCC1, the possible MCSEL field channels range from 0 through 127. For MCC2, the possible MCSEL field channels range 128 through 255. This example uses channel 128 (MCSEL = 10000000) through channel 152 (MCSEL = 10011000). The CNT field indicates the entry's number of bits or bytes routed to the TDM. In this example, each entry routes one byte to the TDM so that the CNT field is cleared and the BYT field is set in all SI2 RAM receive entries. The definitions of these fields change when super-channels are transmitted. Finally, the LST bit is set in the last SI2 RAM receive entry. The LST bit of all other entries is cleared. The LST bit allows the SI to wait until the next frame for the frame sync signal to start. **Figure 7** shows the SI2 RAM receive entries for the 25 MCC channels.

SI2 RAM Rx Entries

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	MCC=1	LOOP/ECHO	SUPER				MCSEL	CSEL			CNT			BYT	LST		
Time Slot 12: 0x9002	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	Channel 128
Time Slot 13: 0x9002	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	Channel 128
Time Slot 14: 0x9002	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	Channel 128
Time Slot 15: 0x9002	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	Channel 128
Time Slot 16: 0x9022	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	Channel 129
Time Slot 17: 0x9022	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	Channel 129
Time Slot 18: 0x9042	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	Channel 130
Time Slot 19: 0x9042	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	Channel 130
Time Slot 20: 0x9062	1	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	Channel 131
Time Slot 21: 0x9062	1	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	Channel 131
~ ~ ~ ~																	
Time Slot 60: 0x92E2	1	0	0	1	0	0	1	0	1	1	1	0	0	0	1	0	Channel 151
Time Slot 61: 0x92E2	1	0	0	1	0	0	1	0	1	1	1	0	0	0	1	0	Channel 151
Time Slot 62: 0x9302	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0	Channel 152
Time Slot 63: 0x9303	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1	1	Channel 152

Figure 7. SI2 RAM Receive Entries for MCC Channels

2.3 SI2 RAM Transmit Entries for MCC2 Channels

The programming procedure for the SIx RAM transmit differs from that for the SIx RAM receive because of the use of a super-channel table (SCT). To understand the differences, consider the SI2 RAM transmit entries in **Figure 8**. All the MCC channels are super-channels, so the MCC and SUPER bits are set for each entry. The LOOP/ECHO bit in each entry is cleared to indicate no channel loopback.

Time slot 12 is assigned to channel 128, so MCSEL contains a value of 128. Because channel 128 is a super-channel, time slots 13–15 are also assigned to channel 128. However, the MCSEL values cannot be the same in two consecutive SIx RAM entries for the same super-channel. Because of this restriction, a virtual channel number is written into the MCSEL field. For time slots 13–15, channel 128 is redirected to virtual channels using an SCT. The SCT redirects an MCC slot to a different channel number. The start of the SCT is located in the base of dual-port RAM at an offset of SCTPBASE, which is defined in the MCC global parameter RAM.

Figure 10 shows the SCT entries. The real channel number is written into the SCT at an offset of $MCSEL \times 2$ from the SCTPBASE. For example, the SI2 RAM entry for time slot 12 has an $MCSEL = 128$, so the offset is $128 \times 2 = 0x100$. Therefore, the real channel number 128 is written into channel number bits 2–9 in the SCT, yielding an SCT entry of $0x2000$ at offset $0x100$. The SCT entry is calculated by multiplying the real channel number by 64.

The SI2 RAM entry for time slot 13 redirects channel 128 to virtual channel 153, so the SCT is written with $128 \times 64 = 0x2000$ at the offset $153 \times 2 = 0x132$. Virtual channel 153 is chosen since this example uses channels 128–152, and therefore channels 153–255 are available for use as virtual channels. The same procedure is used to determine the SCT entries and offsets for the other super-channels.

As noted earlier, the CNT and BYT fields have a different meaning for super-channels in the SIx RAM transmit. The CNT and BYT fields together determine whether the current entry is the first byte of the MCC super-channel. The combination $CNT = 000$ and $BYT = 1$ indicates that the current entry is the first byte of the super-channel. Subsequent bytes of a super-channel use the combination $CNT = 111$ and $BYT = 0$. For example, the SI2 RAM entry for time slot 12 is the first byte of super-channel 128, so $CNT = 000$ and $BYT = 1$. The next three SI2 RAM entries for time slots 13–15 are also for the next three bytes of super-channel 128, so $CNT = 111$ and $BYT = 0$ for these three entries.

In summary, remember the following principles when programming super-channels to transmit data:

- The value in the MCSEL field must be different for two consecutive entries.
- MCSEL is used as an index into the SCT.
- The channel number is written into the SCT.
- The first entry for a super-channel must have $CNT = 000$ and $BYT = 1$.
- Subsequent entries for a super-channel must have $CNT = 111$ and $BYT = 0$.

SI2 RAM Tx Entries

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	MCC=1	LOOP/ECHO	SUPER				MCSEL	CSEL			CNT			BYT	LST		
Time Slot 12: 0xB002	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	Channel 128
Time Slot 13: 0xB33C	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	Channel 153
Time Slot 14: 0xB35C	1	0	1	1	0	0	1	1	0	1	0	1	1	1	0	0	Channel 154
Time Slot 15: 0xB37C	1	0	1	1	0	0	1	1	0	1	1	1	1	1	0	0	Channel 155
Time Slot 16: 0xB022	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	Channel 129
Time Slot 17: 0xB39C	1	0	1	1	0	0	1	1	1	0	0	1	1	1	0	0	Channel 156
Time Slot 18: 0xB042	1	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	Channel 130
Time Slot 19: 0xB3BC	1	0	1	1	0	0	1	1	1	0	1	1	1	1	0	0	Channel 157
Time Slot 20: 0xB062	1	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	Channel 131
Time Slot 21: 0xB3DC	1	0	1	1	0	0	1	1	1	1	0	1	1	1	0	0	Channel 158
Time Slot 22: 0xB082	1	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0	Channel 132
Time Slot 23: 0xB3FC	1	0	1	1	0	0	1	1	1	1	1	1	1	1	0	0	Channel 159
Time Slot 24: 0xB0A2	1	0	1	1	0	0	0	0	1	0	1	0	0	0	1	0	Channel 133
Time Slot 25: 0xB41C	1	0	1	1	0	1	0	0	0	0	0	1	1	1	0	0	Channel 160
Time Slot 26: 0xB0C2	1	0	1	1	0	0	0	0	1	1	0	0	0	0	1	0	Channel 134
Time Slot 27: 0xB43C	1	0	1	1	0	1	0	0	0	0	1	1	1	1	0	0	Channel 161
Time Slot 28: 0xB0E2	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	0	Channel 135
Time Slot 29: 0xB45C	1	0	1	1	0	1	0	0	0	1	0	1	1	1	0	0	Channel 162
Time Slot 30: 0xB102	1	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	Channel 136
Time Slot 31: 0xB47C	1	0	1	1	0	1	0	0	0	1	1	1	1	1	0	0	Channel 163
Time Slot 32: 0xB122	1	0	1	1	0	0	0	1	0	0	1	0	0	0	1	0	Channel 137
Time Slot 33: 0xB49C	1	0	1	1	0	1	0	0	1	0	0	1	1	1	0	0	Channel 164
Time Slot 34: 0xB142	1	0	1	1	0	0	0	1	0	1	0	0	0	0	1	0	Channel 138
Time Slot 35: 0xB4BC	1	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0	Channel 165

Figure 8. SI2 RAM Transmit Entries for MCC Channels

SI2 RAM Tx Entries

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
	MCC=1	LOOP/ECHO	SUPER				MCSEL		CSEL			CNT			BYT	LST		
Time Slot 36: 0xB162	1	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	Channel 139
Time Slot 37: 0xB4DC	1	0	1	1	0	1	0	0	0	1	1	0	1	1	1	0	0	Channel 166
Time Slot 38: 0xB182	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	Channel 140
Time Slot 39: 0xB4FC	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1	0	0	Channel 167
Time Slot 40: 0xB1A2	1	0	1	1	0	0	0	0	1	1	0	1	0	0	0	1	0	Channel 141
Time Slot 41: 0xB51C	1	0	1	1	0	1	0	0	1	0	1	0	1	1	1	0	0	Channel 168
Time Slot 42: 0xB1C2	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0	1	0	Channel 142
Time Slot 43: 0xB53C	1	0	1	1	0	1	0	0	1	0	1	1	1	1	1	0	0	Channel 169
Time Slot 44: 0xB1E2	1	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	Channel 143
Time Slot 45: 0xB55C	1	0	1	1	0	1	0	0	1	0	1	0	1	1	1	0	0	Channel 170
Time Slot 46: 0xB202	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	Channel 144
Time Slot 47: 0xB57C	1	0	1	1	0	1	0	0	1	0	1	1	1	1	1	0	0	Channel 171
Time Slot 48: 0xB222	1	0	1	1	0	0	1	0	0	0	0	1	0	0	0	1	0	Channel 145
Time Slot 49: 0xB59C	1	0	1	1	0	1	0	0	1	1	0	0	1	1	1	0	0	Channel 172
Time Slot 50: 0xB242	1	0	1	1	0	0	1	0	0	0	1	0	0	0	0	1	0	Channel 146
Time Slot 51: 0xB5BC	1	0	1	1	0	1	0	0	1	1	0	1	1	1	1	0	0	Channel 173
Time Slot 52: 0xB262	1	0	1	1	0	0	1	0	0	0	1	1	0	0	0	1	0	Channel 147
Time Slot 53: 0xB5DC	1	0	1	1	0	1	0	0	1	1	1	0	1	1	1	0	0	Channel 174
Time Slot 54: 0xB282	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	Channel 148
Time Slot 55: 0xB5FC	1	0	1	1	0	1	0	0	1	1	1	1	1	1	1	0	0	Channel 175
Time Slot 56: 0xB2A2	1	0	1	1	0	0	1	0	0	1	0	1	0	0	0	1	0	Channel 149
Time Slot 57: 0xB61C	1	0	1	1	0	1	1	0	0	0	0	0	1	1	1	0	0	Channel 176
Time Slot 58: 0xB2C2	1	0	1	1	0	0	1	0	0	1	1	0	0	0	0	1	0	Channel 150
Time Slot 59: 0xB63C	1	0	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	Channel 177
Time Slot 60: 0xB2E2	1	0	1	1	0	0	1	0	0	1	1	1	0	0	0	1	0	Channel 151
Time Slot 61: 0xB65C	1	0	1	1	0	1	1	0	0	0	1	0	1	1	1	0	0	Channel 178
Time Slot 62: 0xB302	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	1	0	Channel 152
Time Slot 63: 0xB67D	1	0	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	Channel 179

Figure 9. SI2 RAM Transmit Entries for MCC Channels

3 Initialize the SI Mode Registers

There are four SI Mode Registers SI1AMR, SI2BMR, SI2CMR and SI2DMR, one for each TDM, that define SI operation and must be initialized for proper timing on the pins and SLx RAM block assignment. Since this example uses TDMC2, the SI2CMR will be initialized. **Figure 11** shows the SI Mode Register which is discussed on page 20-27 of the MSC8101 Reference Manual.

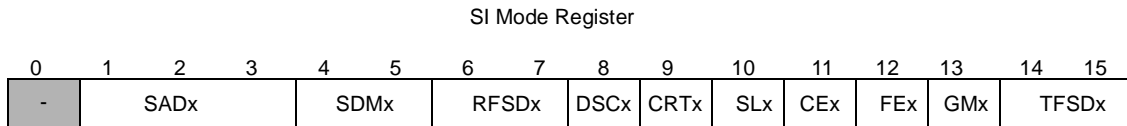


Figure 11. SI Mode Register

- *SADx*. Defines the starting bank address of the SLx RAM section. As mentioned in **Section 2** each SI RAM is divided into four banks of 64 entries for receive and four banks of 64 entries for transmit. The starting bank address for the SLx RAM for each TDM is programmable with a granularity of 32 entries. The first or second 32 entries of each of the four banks may be selected. In this example, we will select the first 32 entries of the first bank as the starting bank address such that $SADx=000$.
- *SDMx*. Defines the TDM diagnostic mode. This example requires the TDM to operate in internal loopback mode so $SDMx = 10$. The TDM transmitter output is internally connected to the TDM receiver input.
- *RFSDx*. Defines the number of clock delays between the receive sync and the first bit of the receive frame. For this example, no bit delay is used, so the first bit of the frame is transmitted or received on the same clock as the sync. This field is set to $RFSDx = 00$.
- *DSCx*. Enables the TDM clock to be doubled. For this example, the input clock is the same speed as the data rate, so $DSCx = 0$.
- *CRTx*. Enables the receive and transmit sections of a TDM to use a common clock and sync signals. Common pins are used so that L1RCLK is the clock pin and L1RSYNC as the receive and transmit sync so $CRTx=1$.
- *SLx*. Defines the sync level. The L1RSYNC is active on logic “1,” so $SLx = 0$.
- *CEx*. Defines the clock edge. Data is sent on the falling edge of the clock and received on the rising edge of the clock, so $CEx = 1$.
- *FEx*. Defines the frame sync edge. The frame sync is sampled on the falling edge of the clock ($FEx = 0$).
- *GMx*. Defines the grant mode. The grant mechanism is not used, so $GMx = 0$.
- *TFSDx*. Defines the number of clock delays between the transmit sync and the first bit of the transmit frame. There is no bit delay, so $TFSDx=00$.

The following code line sets the SI2CMR to these settings to initialize the SI Mode Register:

```
pstIMM->t_aSIRegs[SI2].avusiSIXR[MCC2_TDM] |= 0x0850;
```

4 Connect the Clock Pins to the TDMs

Table 1 shows the possible clock source options for TDMC2. Either CLK3 or CLK7 can be programmed as the TDMC2 receive clock. Either CLK4 or CLK8 can be programmed as the TDMC2 transmit clock.² In this example, a common receive and transmit clock is used as indicated in the CRTx bit in the SI2MR. CLK7 is connected to TDMC2.

Table 1. Clock Options

Clock	C	C	C	C	C	C	C	C	C	C
	L	L	L	L	L	L	L	L	L	L
	K	K	K	K	K	K	K	K	K	K
	1	2	3	4	5	6	7	8	9	10
TDMC2 Rx			√				√			
TDMC2 Tx				√				√		

The CMX SI Clock Route Registers (CMXSI1CR and CMXSI2CR) define the clock source for the TDMs. Since only TDMC2 is used, CMXSI2CR is initialized (see **Figure 12**). The RTB2CS, RTC2CS, and RTD2CS bits define the receive clock source for TDMB2, TDMC2, and TDMD2, respectively. The TTB2CS, TTC2CS, and TTD2CS bits define the transmit clock source for TDMB2, TDMC2, and TDMD2, respectively.

0	1	2	3	4	5	6	7
—	RTB2CS	RTC2CS	RTD2CS	—	TTB2CS	TTC2CS	TTD2CS

Figure 12. CMX SI2 Clock Route Register (CMXSI2CR)

In this example, RTC2CS = 1 and TTC2CS = 0 to select CLK7 as the receive and transmit TDMC2 clock source. The following code line sets the CMXSI2CR to these settings in order to connect the clock pin to the TDM:

```
pstIMM->vucCMxSI2CR = 0x20;
```

5 Configure the TDM Pins

Figure 13 shows the TDM pins. Each TDM has a transmit data L1TXD and a receive data L1RXD pin. The L1TSYNC and L1RSYNC pins indicate the start of a frame on the transmit and receive data pins. The L1RQ pin is used for ISDN applications and functions as a request to send data. The L1CLKO pin is another special function pin for General Circuit Interface (GCI) support. There are eight strobe pins on the MSC8101: TDMA1 has a set of strobe pins L1ST[1–4]. TDMB2, TDMC2, and TDM2 share another set of strobe pins. The state of these pins is controlled in the SSEL[1–4] bits in the SIx RAM entries. These pins are asserted on a bit or byte basis, and they are useful for interfacing to other devices that do not support multiplexed interface. The L1RQ, L1CLKO, and L1ST[1–4] pins are not used in this example.

2. The TDM clock source options are discussed in the *MSC8101 Reference Manual*, in the chapter on CPM Multiplexing, as is the CMXSI2CR.

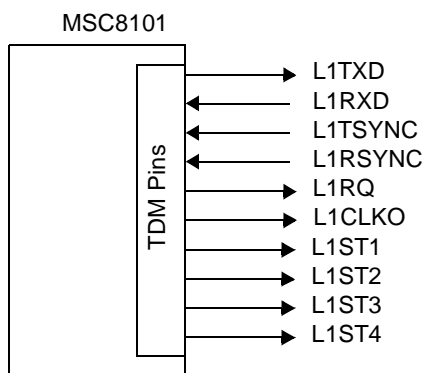


Figure 13. TDM Pins

The TDM pins share functions with other peripherals. The parallel I/O port pins must be programmed to configure the pins for TDM operation. **Table 2** shows the parallel I/O settings for each pin. The four parallel I/O registers that must be configured are the Port Special Options Registers (PSOR_x), Port Data Direction Registers (PDIR_x), Port Open-Drain Registers (PODR_x), and Port Pin Assignment Registers (PPAR_x). Note that the TDM pins are controlled by both Port B and Port C pin assignment.³

Table 2. TDMC2 Pin Configuration

Signal	Pin	PSOR _x	PDIR _x	PODR _x	PPAR _x
L1RSYNC	PB24	PSORB[24]=1	PDIRB[24]=0	PODRB[24]=0	PPARB[24]=1
L1TSYNC	PB25	PSORB[25]=1	PDIRB[25]=0	PODRB[25]=0	PPARB[24]=1
L1RXD	PB26	PSORB[26]=1	PDIRB[26]=0	PODRB[26]=0	PPARB[24]=1
L1TXD	PB27	PSORB[27]=1	PDIRB[27]=0	PODRB[27]=0	PPARB[24]=1
CLK7	PC25	PSORC[25]=0	PDIRC[25]=0	PODRC[25]=0	PPARC[25]=1

The following code configures the TDMC2 receive and transmit data lines L1RXD and L1TXD and the receive and transmit frame syncs, L1RSYNC and L1TSYNC. It enables CLK7 as the TDMC2 clock source.

3. TDM pin assignments are discussed in the chapter on Parallel I/O ports in the *MSC8101 Reference Manual*. Freescale provides the MSC8101 Pin-Mux Tool software that generates the parallel I/O port pin configuration according to pin allocation. This tool can be downloaded from the Freescale web. See the application note AN1854/D, *Initializing the MSC8101 Communications Processor Module (CPM) I/O*, and the software that accompanies it.

```

/* Program the Port Special Options Registers (PSOR) */
pstIMM->t_aIORegs[PORTB].vuliPSOR |= 0x000000F0; //enable TDMC2
pstIMM->t_aIORegs[PORTC].vuliPSOR &= ~0x00000040; //enable CLK7

/* Program the Port Data Direction Registers (PDIR) */
pstIMM->t_aIORegs[PORTB].vuliPDIR &= ~0x000000F0; //TDMC2 inouts
pstIMM->t_aIORegs[PORTC].vuliPDIR &= ~0x00000040; //CLK7 is input

/* Program the Port Open-Drain Registers (PODR) */
pstIMM->t_aIORegs[PORTB].vuliPODR &= ~0x000000F0; //no open-drain
pstIMM->t_aIORegs[PORTC].vuliPODR &= ~0x00000040; //no open-drain

/* Program the Port Pin Assignment Registers (PPAR) */
pstIMM->t_aIORegs[PORTB].vuliPPAR |= 0x000000F0; //enable TDMC2
pstIMM->t_aIORegs[PORTC].vuliPPAR |= 0x00000040; //enable CLK7

```

6 Issue a STOP Command

Before the MCC parameters are initialized, a STOP TRANSMIT and a STOP RECEIVE command must be issued to the Communications Processor Command Register (CPCR) as shown in **Figure 14**. The STOP TRANSMIT command disables transmission on the selected MCC channel, and the STOP RECEIVE command terminates reception on the selected MCC channel. These commands are issued for each channel used. The MCC channel is specified in the MCN field. **Table 3** shows the CPCR settings for MCC2 STOP TRANSMIT and STOP RECEIVE commands.⁴

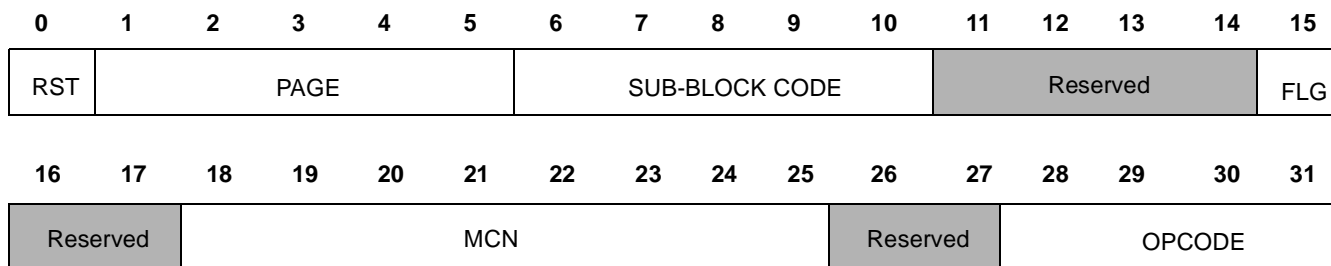


Figure 14. Communications Processor Command Register (CPCR)

Table 3. CPCR Settings for the MCC STOP Command

Command	Opcode	Page (MCC2)	Sub-Block (MCC2)
MCC STOP TX	0100	01000	11101
MCC STOP RX	1001		

4. The CPCR and the MCC commands are discussed in the Communications Processor Module chapter of the *MSC8101 Reference Manual*.

The following example shows how to issue the STOP TRANSMIT and STOP RECEIVE commands to the CPR. These commands are issued for every MCC channel designated by the ChanNum variable. In this example, ChanNum has a value of 128 through 152.

```
pstIMM->vuliCPCR = 0x23A10004 | (ChanNum << 6); // STOP TX
while (pstIMM->vuliCPCR & 0x00010000); //wait for CP to be ready
pstIMM->vuliCPCR = 0x23A10009 | (ChanNum << 6); // STOP RX
```

7 Initialize the MCC

Once the STOP TRANSMIT and STOP RECEIVE commands are issued to the CPR, the MCC parameters can be initialized. The MCC has several different types of parameters. The basic MCC data structures reside in both dual-port RAM and in external memory, as shown in **Figure 15**. Each MCC uses the following data structures:

- *Global Parameters.* The global parameters for MCC1 are common to channels 0–127 and the global parameters for MCC2 are common to channels 128–255. MCC1 and MCC2 global parameters are placed at offsets 0x8700 and 0x8800, respectively, from the dual-port RAM base address.
- *Channel Extra Parameters.* These parameters contain buffer descriptor base addresses and pointers. They are located at an offset of XTRABASE + 8 × CH_NUM from the dual-port RAM base address. XTRABASE is one of the global parameters.
- *super-channel Table.* Redirects an MCC time slot to a different channel number. This table is located at an offset of SCTPBASE from the dual-port RAM base address. SCTPBASE is one of the global parameters.
- *Channel-Specific Parameters.* These parameters are specific to each MCC channel. Two sets of channel-specific parameters are available: HDLC and Transparent. The channel-specific parameters are located at an offset of 64 × CH_NUM from the dual-port RAM base address.
- *Buffer Descriptor Tables.* In external memory, a receive channel's buffer descriptor table is located at an offset of MCCBASE + 8 × RBASE from the dual-port RAM base address. A transmit channel's buffer descriptor table is located at an offset of MCCBASE + 8 × TBASE from the dual-port RAM base address. MCCBASE is one of the global parameters and RBASE/TBASE are channel extra parameters.
- *Interrupt Queues.* The circular interrupt queues reside in external memory. There is one transmit interrupt table and four receive interrupt tables. TINTBASE points to the table for the transmitter interrupts, and RINTBASE[0–3] points to the tables for the receiver interrupts. TINTBASE and RINTBASE[0–3] are global parameters.

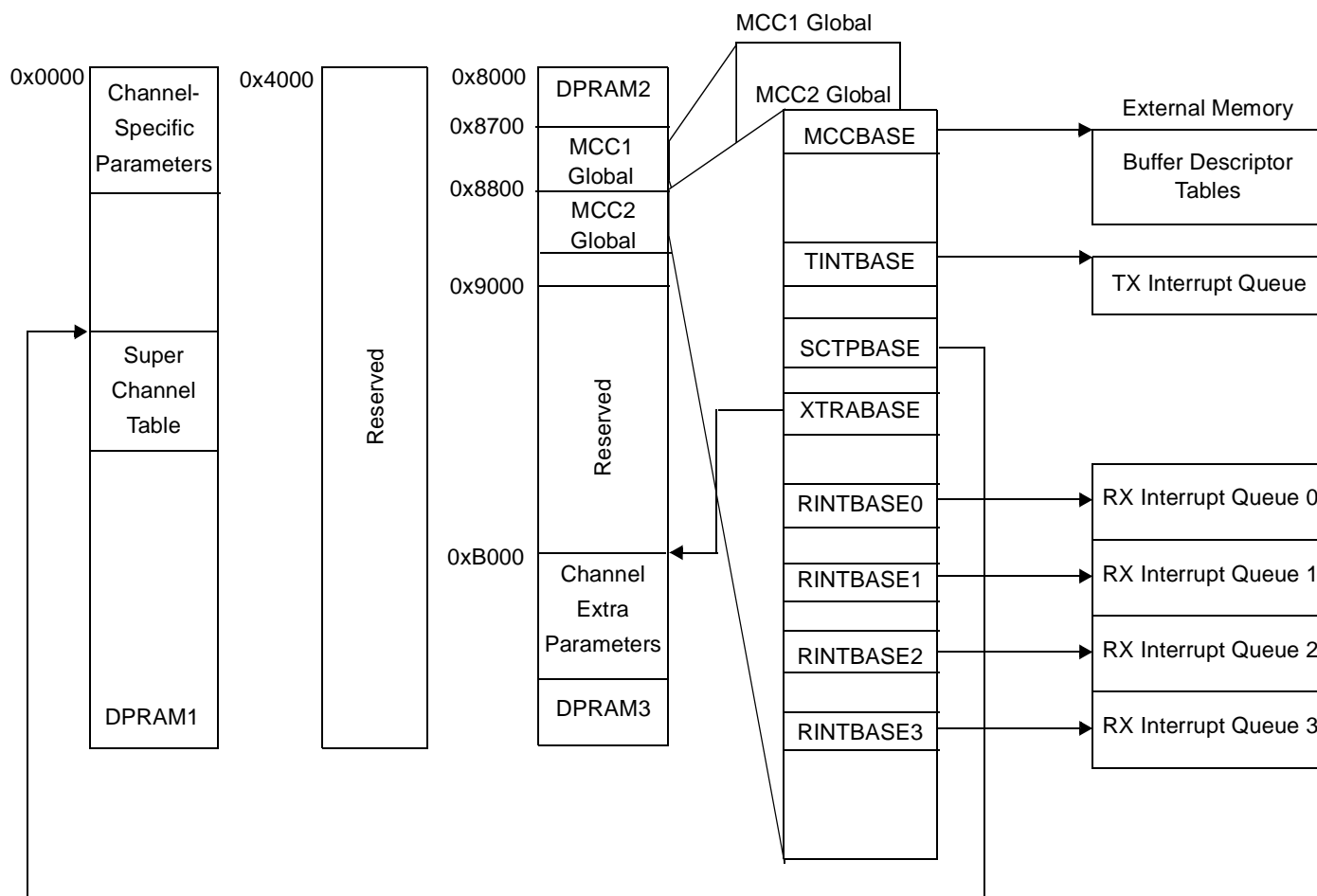


Figure 15. MCC Data Structure

7.1 MCC Global Parameters

You must initialize or clear the MCC2 global parameters in the dual-port RAM, starting at address IMM + 0x8800.⁵

- *MCCBASE*. Contains the base address of the 512 KB buffer descriptor segment in external memory.
- *MCCSTATE*. Used by the Communications Processor (CP) for global state definition. It is initialized to zero.
- *MRBLR*. Initialized to the maximum number of bytes written to a receive buffer before the next buffer is used. The value must be a multiple of eight.
- *GRFTHR*. Specifies how many receive frame (RXF) events by all channels occur before an interrupt is generated. This example generates an interrupt on all buffers received.
- *GRFCNT*. Decrements the GRFTHR value and should be initialized to the GRFTHR value.
- *RINTTMP*. This parameter is used by the CP to temporarily hold the receive interrupt table entry. It is initialized to zero.

5. The MCC global parameters are discussed in the chapter on Multi-Channel Controllers in the *MSC8101 Reference Manual*.

- *DATA0* and *DATA1*. Used by the CP to hold data temporarily. They are initialized to zeros.
- *TINTBASE*. Contains the base address of the transmit interrupt table.
- *TINTPTR*. Used by the CP to point to the transmit interrupt table. Before interrupts are enabled, the *TINTBASE* value must be copied to the *TINTPTR* parameter.
- *SCTPBASE*. Contains the base address of the super-channel table. It is an offset from the dual-port RAM base address.
- *C_MASK32*. Holds the 32-bit CRC constant. It must be initialized to 0xDEBB20E3.
- *XTRABASE*. Holds the base address of the channel extra parameters. It is an offset from the dual-port RAM base address.
- *C_MASK16*. Holds the 16-bit CRC constant. It must be initialized to 0xF0B8.
- *RINTTMP[0–3]*. Used by the CP to point to the receive interrupt tables and must be cleared before enabling interrupts.
- *RINTBASE[0–3]*. Contain the base addresses of the receive interrupt tables.
- *RINTPTR[0–3]*. Used by the CP to point to the receive interrupt tables. Before interrupts are enabled, the *RINTBASE[0–3]* values must be copied to the *RINTPTR[0–3]* parameters.

The following example shows how to initialize the MCC2 global parameters.

```

pstMccGlobal->vuliMCCBase      = MCC2_BDRINGS_BASE;
pstMccGlobal->vusiMCCState    = 0x0000;
pstMccGlobal->vusiMRBLR      = MCC2_RX_BUF_SIZE;
pstMccGlobal->vusiGRFThr      = MCC2_RX_NUM_BUF*MCC2_NUM_CH;
pstMccGlobal->vusiGRFCnt      = pstMccGlobal->vusiGRFThr;
pstMccGlobal->vuliRIntTmp     = 0x00000000;
pstMccGlobal->vuliData0       = 0x00000000;
pstMccGlobal->vuliData1       = 0x00000000;
pstMccGlobal->vuliTIntBase    = MCC2_TINT_BASE;
pstMccGlobal->vuliTIntPtr     = pstMccGlobal->vuliTIntBase;
pstMccGlobal->vuliTIntTmp     = 0x00000000;
pstMccGlobal->vusiSCTPBase    = 0x1C00;
pstMccGlobal->vulicMask32     = 0xDEBB20E3;
pstMccGlobal->vusiXtraBase    = MCC2_XTRABASE;
pstMccGlobal->vulicMask16     = 0xF0B8;
pstMccGlobal->vuliRIntTmp0    = 0x00000000;
pstMccGlobal->vuliRIntTmp1    = 0x00000000;
pstMccGlobal->vuliRIntTmp2    = 0x00000000;
pstMccGlobal->vuliRIntTmp3    = 0x00000000;
pstMccGlobal->vuliRIntBase0   = MCC2_RINT0_BASE;
pstMccGlobal->vuliRIntPtr0    = pstMccGlobal->vuliRIntBase0;
pstMccGlobal->vuliRIntBase1   = MCC2_RINT1_BASE;
pstMccGlobal->vuliRIntPtr1    = pstMccGlobal->vuliRIntBase1;
pstMccGlobal->vuliRIntBase2   = MCC2_RINT2_BASE;
pstMccGlobal->vuliRIntPtr2    = pstMccGlobal->vuliRIntBase2;
pstMccGlobal->vuliRIntBase3   = MCC2_RINT3_BASE;
pstMccGlobal->vuliRIntPtr3    = pstMccGlobal->vuliRIntBase3;
    
```

7.2 MCC Channel Extra Parameters

Each active MCC channel has channel extra parameters in the dual-port RAM at address $IMM + XTRABASE + 8 \times CH_NUM$. **Figure 16** shows the channel extra parameter table structure. For MCC2, the extra parameters for channel 128 start at address $IMM + XTRABASE + 8 \times 128$. Each channel uses 8 bytes of extra parameters. Therefore, the extra parameters for channel 129 start at address $IMM + XTRABASE + 8 \times 129$.

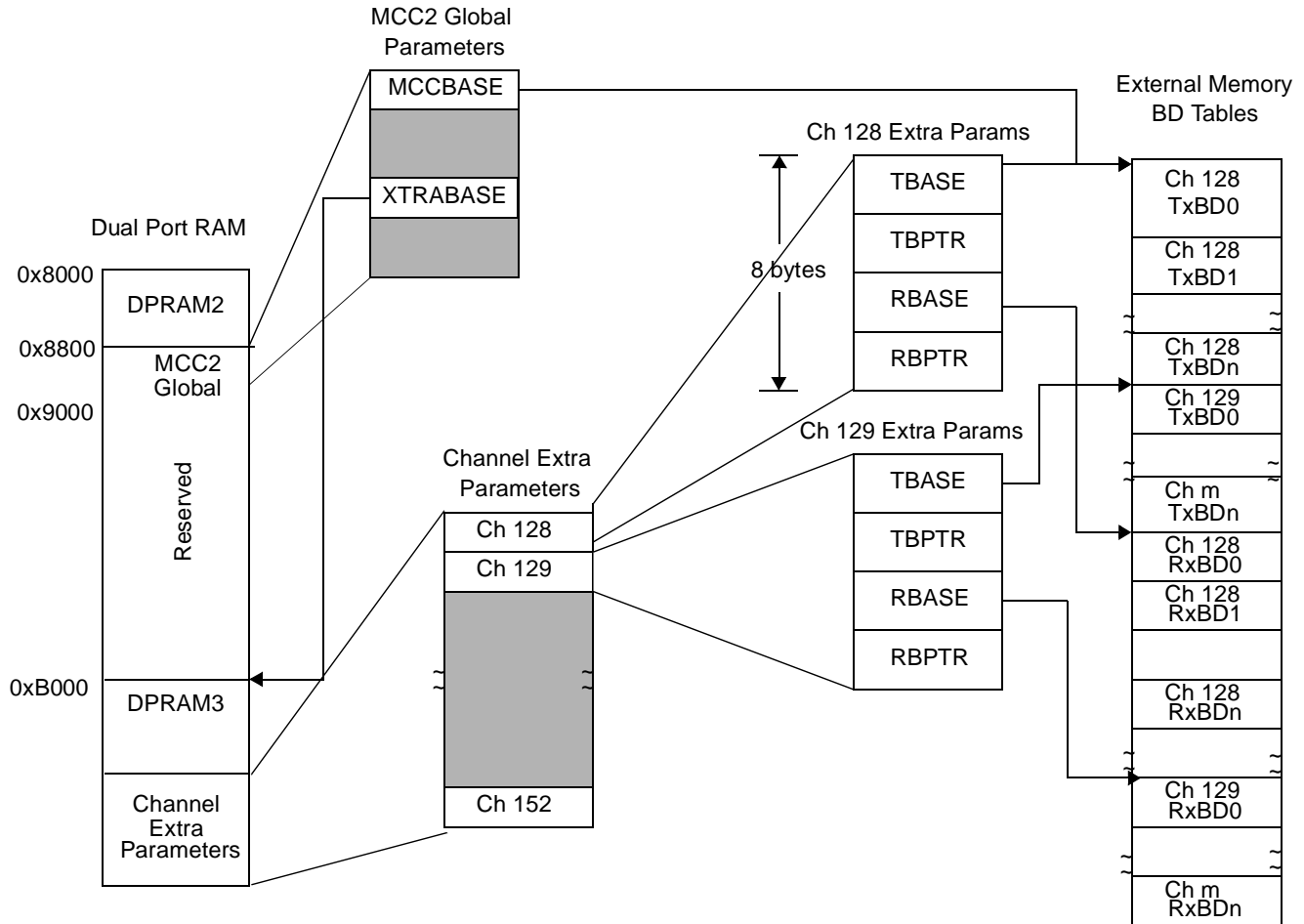


Figure 16. Channel Extra Parameter Data Structure

The channel extra parameters are initialized for each channel:

- **TBASE.** Contains the offset of the channel TxBD base address from the MCCBASE. In this example, the TBASE for channel 128 is 0x0000, so the TxBD table base address starts at MCCBASE. The TxBD table base address for channel 129 starts after the last TxBD for channel 128.
- **TBPTR.** Contains the offset of the channel's current TxBD from the MCCBASE. It must be initialized to the TBASE value before the channel is enabled.
- **RBASE.** Contains the offset of the channel RxBD table base address from the MCCBASE. In this example, the RxBD table base address for channel 128 starts after the last TxBD of the last channel.
- **RBPTR.** Contains the offset of the channel's current receive buffer descriptor from the MCCBASE. It must be initialized to the RBASE value before the channel is enabled.

The following lines of code initialize the MCC channel extra parameters to the values discussed here:

```
pstMCCmchx->vusiTBase = MCC2_TX_NUM_BUF*(ChanNum-128);
pstMCCmchx->vusiTBPtr = pstMCCmchx->vusiTBase;
pstMCCmchx->vusiRBase = MCC2_TX_NUM_BUF * MCC2_NUM_CH + MCC2_RX_NUM_BUF *(ChanNum-128);
pstMCCmchx->vusiRBPtr = pstMCCmchx->vusiRBase;
```

7.3 MCC Channel-Specific Parameters

Each MCC uses 64 bytes of channel-specific parameters that are located at an offset of $64 \times \text{CH_NUM}$ from the dual-port RAM base address. Since only MCC2 is used, channels 0–127 are inactive. The first 128×64 bytes of dual-port RAM can be used for other purposes.⁶ The channel-specific parameters are initialized for each channel, as follows:

- *TSTATE*. Written when a transmit channel is enabled. It is initialized to zero.
- *ZISTATE*. Initialized to 0x10000207 for a regular or non-inverted channel.
- *ZIDATA0* and *ZIDATA1*. The zero-insertion high and low word data buffers, which are initialized to 0xFFFFFFFF.
- *TBDFlags*. Used by the CP and is initialized to zero.
- *TBDCNT*. Used by the CP to indicate the number of remaining bytes in the transmit buffer. It is initialized to zero.
- *TBDPTR*. Used by the CP to hold the transmit internal data pointer. It is initialized to zero.
- *INTMSK*. Enables each event defined in the interrupt table entry. It is initialized to enable Transmit Buffer (TXB), Receive Frame (RXF), and Receive Buffer (RXB) interrupts.
- *CHAMR*. The Channel Mode Register is initialized for HDLC mode, polling of the TxBDs, 16-bit CRC and RINT0 is selected.
- *TCRC*. Used by the CP to hold the transmit CRC. It is initialized to zero.
- *RSTATE*. Written when a receive channel is enabled. It is initialized to zero.
- *ZDSTATE*. Must be initialized to 0x00FFFFFFE0 for a regular or non-inverted channel.
- *ZDDATA0* and *ZDDATA1*. The zero-deletion high and low word data buffers, which are initialized to 0xFFFFFFFF.
- *RBDFlags*. Used by the CP and is initialized to zero.
- *RBDCNT*. Used by the CP to indicate the number of remaining bytes in the receive buffer. It is initialized to zero.
- *RBDPTR*. Used by the CP to hold the receive internal data pointer. It is initialized to zero.
- *MFLR*. Specifies the maximum frame length for the channel. It is initialized to the transmit buffer size.
- *MAX_CNT*. Used by the CP as a maximum length counter. It is initialized to zero.
- *RCRC*. Used by the CP to hold the receive CRC. It is initialized to zero.

6. The channel-specific parameters are discussed in the chapter on Multi-Channel Controllers (MCCs) in the *MSC8101 Reference Manual*.

The following code initializes the MCC channel-specific parameters to the values discussed here.

```

pstMCCmch->vuliTState      = 0x00000000;
pstMCCmch->vuliZiState     = 0x10000207;
pstMCCmch->vuliZiData0    = 0xFFFFFFFF;
pstMCCmch->vuliZiData1    = 0xFFFFFFFF;
pstMCCmch->vusiTBDFlags   = 0x0000;
pstMCCmch->vusiTBDCnt     = 0x0000;
pstMCCmch->vuliTBDPtr     = 0x00000000;
pstMCCmch->vusiIntMask    = 0x0105;
pstMCCmch->vusiChaMR      = 0xE000;
pstMCCmch->vuliTCRC       = 0x00000000;
pstMCCmch->vuliRState     = 0x00000000;
pstMCCmch->vuliZDState    = 0x00FFFFFF0;
pstMCCmch->vuliZDData0    = 0xFFFFFFFF;
pstMCCmch->vuliZDData1    = 0xFFFFFFFF;
pstMCCmch->vusiRBDFlags   = 0x0000;
pstMCCmch->vusiRBDcnt     = 0x0000;
pstMCCmch->vuliRBDPtr     = 0x00000000;
pstMCCmch->vusiMaxRlen    = MCC2_TX_BUF_SIZE;
pstMCCmch->vusiSyncMaxCnt = 0x0000;
pstMCCmch->vuliRCRC       = 0x00000000;

```

7.4 INIT Command

The INIT TX AND RX PARAMS command must be issued to the CPCRC after the parameters are initialized and before the transmitter and receiver are enabled. This command initializes the transmit and receive parameters to their reset state. This command initializes only the 32 consecutive channels starting with the channel number specified in the MCN field in the CPCRC. To initialize more than 32 channels, the INIT TX AND RX PARAMETER command must be reissued with the appropriate channel numbers. In this example, the command is issued twice with MCN = 128 and MCN = 160 to initialize all the channels 128–191, which covers the range of channels used in this example.

Table 4. CPCRC Settings for the MCC INIT Command

Command	Opcode	Page (MCC2)	Sub-Block (MCC2)
MCC INIT TX AND RX PARAMS	0000	01000	11101

The following code lines show how to issue an MCC INIT command.

```

while (pstIMM->vuliCPCR & 0x00010000); //wait until CP is ready to rx new cmd
pstIMM->vuliCPCR = 0x23A12000; //issue init tx/rx params to ch. 128

while (pstIMM->vuliCPCR & 0x00010000); //wait until CP is ready to rx new cmd
pstIMM->vuliCPCR = 0x23A12800; //init tx and rx param cmd ch. 160

```

7.5 MCC Interrupts

The interrupt circular tables in the external memory store interrupt information when a channel event occurs. There is one transmit interrupt queue and four receive interrupt queues. The base pointers to these tables are defined in the MCC global parameters TINTBASE and RINTBASE[0–3]. The global parameters TINTPTR and

RINTPTR[0–3] are pointers to the current entry in the interrupt tables. The following channel event bits can be set in the channel’s INTMSK channel-specific parameter so that the event information and channel number are entered into the interrupt table when the event occurs:

- *UN*. No data is available to be sent to the transmitter (underrun).
- *TXB*. A buffer is completely transmitted.
- *NID*. A pattern that is not an idle is identified.
- *IDL*. The channel receiver identifies the first occurrence of HDLC idle after any non-idle pattern.
- *MRF*. More bytes are received than the value specified in the channel-specific parameter MRFLR.
- *RXF*. A complete HDLC frame was received.
- *BSY*. A frame was received but was discarded due to lack of buffers.
- *RXB*. A buffer was received on this channel that was not the last buffer in the frame.

Figure 17 shows the interrupt circular table entry. The transmit interrupt table shows only events caused by transmission, and the receive interrupt table shows only events caused by reception. When a channel event occurs and the corresponding bit is set in the INTMSK parameter, then the event bit is set and the channel number is written to the interrupt table at the location defined by the INTPTR parameter. Clearing the channel event bit in the INTMSK does not generate an entry into the interrupt table or cause an interrupt request to the interrupt controller.

When an entry is made into the interrupt table, the V bit is set to indicate that the entry contains valid interrupt information. The CP increments INTPTR, and the appropriate bit in the MCCE register is set, which can generate an interrupt to the interrupt controller. You must clear the V bit for each entry serviced by the interrupt service routine. The W bit indicates the last interrupt circular table entry. The CP reads the next event’s entry from the base of the interrupt table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	W	Reserved				UN	TXB	Reserved		NID	IDL	MRF	RXF	BSY	RXB
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved		CHANNEL NUMBER									Reserved				

Figure 17. Interrupt Circular Table Entry

As **Figure 18** shows, the MCC Event (MCCE) and MCC Mask (MCCM) registers have an identical format. The MCCM enables interrupt requests when interrupt events occur. The MCCE signals an interrupt request to the interrupt controller while the channel information is handled in the interrupt circular table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QOV0	RINT0	QOV1	RINT1	QOV2	QOV3	RINT3	Reserved				TQOV	TINT	GUN	GOV	

Figure 18. MCC Event and Mask Registers (MCCE/MCCM)

The MCCE/MCCM bits are set as follows:

- *QOV[0–3]*. An overflow in the receive interrupt table, causing the CP to generate an interrupt request.
- *RINT[0–3]*. There is a new entry in the receive interrupt table.
- *TQOV*. An overflow in the transmit interrupt table, causing the CP to generate an interrupt request.
- *TINT*. There is a new entry in the transmit interrupt table.
- *GUN*. Indicates an underrun error in the transmitter FIFO buffer. This is a fatal error and the MCC stops transmitting data in all channels.
- *GOV*. Indicates an overrun error in the receiver FIFO buffer. This is a fatal error and the MCC stops receiving data in all channels.

To initialize the MCC2 interrupts, the MCCE must be cleared by setting all the bits. Only the GUN and GOV interrupts are enabled at first.

```
pstIMM->t_aSIRegs[MCC2].vusimCCE = 0xffff;
pstIMM->t_aSIRegs[MCC2].vusimMCCM = 0x0003;
```

The transmit interrupt table contains MCC2_TINT_NUM four-byte entries. This table is initialized to zeros. The W bit in the last entry is set. The TQOV and TINT interrupts are then enabled.

```
pulimCCIrqCircTabBase = (UWord32*)MCC2_TINT_BASE;
memset(pulimCCIrqCircTabBase, 0, MCC2_TINT_NUM*4);
*(pulimCCIrqCircTabBase+MCC2_TINT_NUM-1) = 0x40000000;
pstIMM->t_aSIRegs[MCC2].vusimMCCM |= 0x000C;
```

The receive interrupt table contains MCC2_RINT0_NUM four-byte entries. This table is initialized to zeros. The W bit in the last entry is set. The QOV0 and RINT0 interrupts are then enabled.

```
pulimCCIrqCircTabBase = (UWord32*)MCC2_RINT0_BASE;
memset(pulimCCIrqCircTabBase, 0, MCC2_RINT0_NUM*4);
*(pulimCCIrqCircTabBase+MCC2_RINT0_NUM-1) = 0x40000000;
pstIMM->t_aSIRegs[MCC2].vusimMCCM |= 0xC000;
```

When an MCC2 interrupt occurs, bit 5 of the SIU Low Interrupt Pending Register (SIPNR_L) is set, clearing any previous MCC2 interrupts. Bit 5 of the SIU Low Interrupt Mask Register (SIMR_L) enables MCC2 interrupts.

```
pstIMM->vuliICSIPnRL |= 0x04000000; //clear any previous MCC2 interrupt
pstIMM->vuliICSIMRL |= 0x04000000; //enable MCC2 interrupt
```

7.6 Buffer Descriptors

Each MCC channel requires a TxBD and an RxBD table, which are located in external memory. The channel extra parameters, TBASE and RBASE, contain the offset of the channel TxBD and RxBD base addresses from the MCCBASE, as shown in **Figure 16**. Each BD is 8 bytes, as follows (see **Figure 19**):

- The first two bytes contain control and status information on the buffer defined by the BD. The TxBD and RxBD have different control and status field definitions.
- The next two bytes indicates the data length. For the TxBD, this field contains the number of bytes the MCC should transmit from the associated data buffer. This value should be greater than zero. For the RxBD, this field contains the number of bytes written by the CP into the associated data buffer. When the current RxBD is the last BD in the frame, the data length field contains the total number of bytes

received, including the CRC. The receive buffer size should equal to or greater than the MRBLR global parameter value.

- The last four bytes of the TxBD and RxBD contain the address of the associated transmit and receive buffers, respectively.

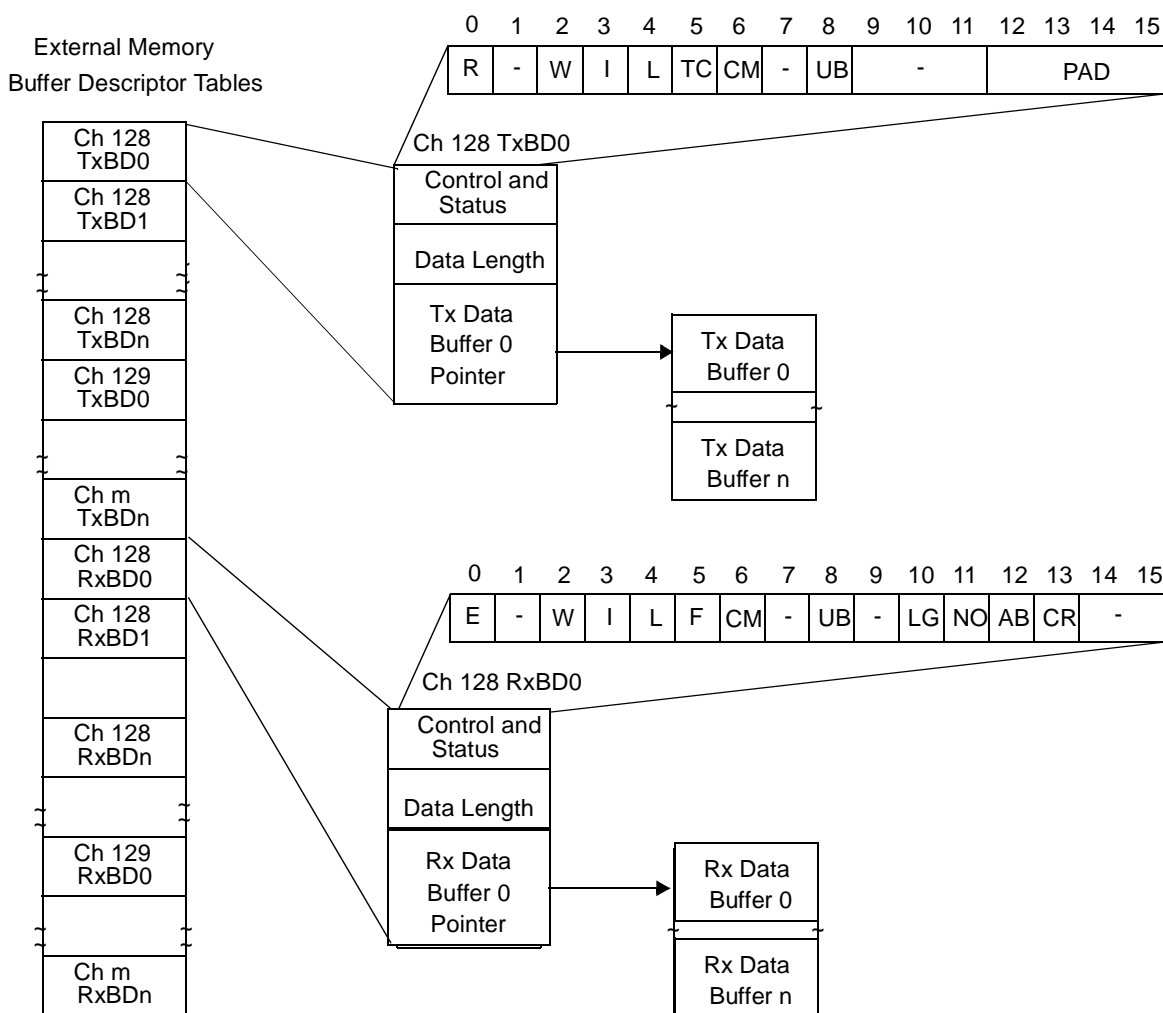


Figure 19. Buffer Descriptor Data Structure

The following loop initializes the TxBDs. The W bit in the last BD is set so that the CP transmits data from the first BD in the table after the associated buffer of the last BD is transmitted. The R bit in the BD is set to indicate that the buffer associated with the BD is ready for transmission. The number of bytes to transmit is defined as the transmit buffer size minus two bytes which are occupied by the 16-bit CRC.

```

for(i=0; i<ucTxBufNum; i++)
{
    if ((i==ucTxBufNum-1))
        pstMcc2->astTxChan[ChanNum].astBufD[i].usiBdCstatus = 0xA000;
    else
        pstMcc2->astTxChan[ChanNum].astBufD[i].usiBdCstatus = 0x8000;
    pstMcc2->astTxChan[ChanNum].astBufD[i].usiBdLength = MCC2_TX_BUF_SIZE-
2;
}
    
```

```

pstMcc2->astTxChan[ChanNum].astBufD[i].uliBdAddr =
    LOCAL_SRAM_BASE + (ChanNum - 128) * MCC2_TX_NUM_BUF;
}

```

The following loop initializes the RxBDs. The W bit in the last BD is set so that the CP receives data into the first BD in the table after the associated buffer of the last BD is used. The E bit in the BD is set to indicate that the buffer associated with the BD is empty.

```

for(i=0; i<ucRxBufNum; i++)
{
    if ((i==ucRxBufNum-1))
        pstMcc2->astRxChan[ChanNum].astBufD[i].usiBdCstatus = 0xA000;
    else
        pstMcc2->astRxChan[ChanNum].astBufD[i].usiBdCstatus = 0x8000;
    pstMcc2->astRxChan[ChanNum].astBufD[i].usiBdLength = 0;
    pstMcc2->astRxChan[ChanNum].astBufD[i].uliBdAddr =
        LOCAL_SRAM_BASE +
        MCC2_TX_NUM_BUF * MCC2_TX_BUF_SIZE * MCC2_NUM_CH +
        (ChanNum - 128) * MCC2_RX_NUM_BUF;
}

```

8 Enable the TDM

The two SI Global Mode Registers (SI1GMR and SI2GMR) define the activation state of the TDM channels. Since this example uses TDMC2, the SI2GMR is initialized.

0	1	2	3	4	5	6	7
STZD	STZC	STZB	STZA	END	ENC	ENB	ENA

Figure 20. SI Global Mode Register

The STZ_x bits are cleared since these bits apply to the GCI bus mode. The EN_x bits enable the TDM. The following example shows how to enable TDMC2 by writing to the SIGMR.

```

pstIMM->t_aSIRegs[SI2].vucSIGMR =0x04;

```

9 Enable the MCC Channels

To start an MCC channel, the TSTATE and RSTATE parameters are written with a value of 0xHH8000, where HH is the high byte of TSTATE and RSTATE. The high bytes provide transaction parameters associated with SDMA channel accesses. The TSTATE and RSTATE registers of each channel are located in the channel-specific parameter area. **Figure 21** shows the bits in these registers.

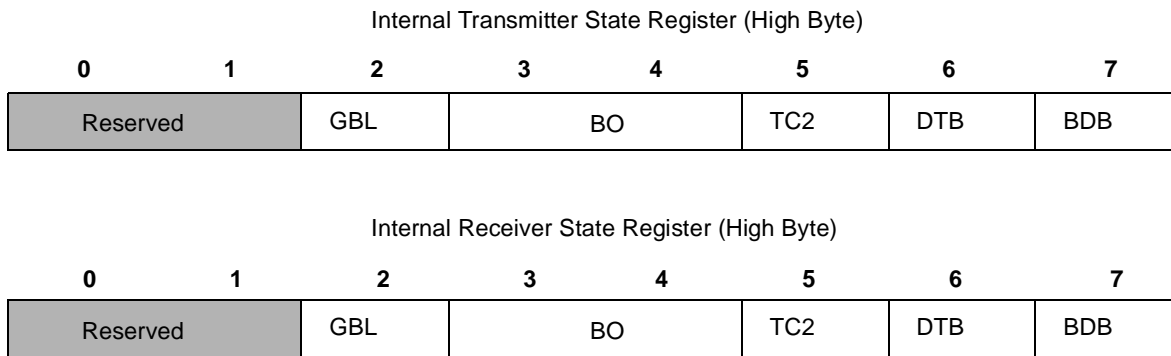


Figure 21. TSTATE/RSTATE Registers

For the example discussed here, the TSTATE/RSTATE bit values are as follows:

- *GBL*. Activates snooping. There is no bus snooping, so $GBL = 0$.
- *BO*. Defines the byte ordering for the buffer. The mode is big-endian, so $BO = 10$.
- *TC2*. Defines the transfer code value, which is 0 in this case, so $TC2 = 0$.
- *DTB*. Indicates the bus that handles transfers to and from data buffers. The local bus is selected, so $DTB=1$.
- *BDB*. Indicates the bus that handles transfers to and from buffer descriptors and interrupt circular tables so $BDB = 1$.

The following example shows how to enable the MCC channels. The TSTATE and RSTATE offset from the dual-port RAM base address of each channel is calculated as $64 \times ChanNum$, where ChanNum is the MCC channel number.

```

for (ucCh = 0; ucCh < MCC2_NUM_CH; ucCh++)
    MCC2_StartChannel(ucCh);
void MCC2_StartChannel(UByte ChanNum)
{
    pstMCCmch = (t_MChPRAM*)(IMM_BASE + 64*ChanNum);
    pstMCCmch->vuliTState = 0x1B800000; //Big endian, local bus for BD and
data
    pstMCCmch->vuliRState = 0x1B800000; //Big endian, local bus for BD and
data
}

```

NOTES:

NOTES:

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, 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 or +81 3 5437 9125
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

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. StarCore is a licensed trademark of StarCore LLC. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2002, 2005.