AN2147
Rev. 2, 11/2004

Initializing MSC8101/MSC8103 CPM Parameter RAM and Buffer Descriptors

By Barbara Johnson

When using the MSC8101/MSC8103 communications processor module (CPM) serial controllers, you must initialize the parameter RAM, buffer descriptor tables, and data buffers. It is important that you understand the structure of the parameter RAM and buffer descriptor tables as you define the parameters for the operation of the serial controller and allocate the data buffers. All serial controllers have a common structure that includes the following protocols:

- Serial communications controllers (SCCs): UART, HDLC, BISYNC, Transparent, Ethernet, and AppleTalk modes
- Fast communications controllers (FCCs): HDLC, Fast Ethernet, and Transparent modes
- Serial management controllers (SMCs): UART, Transparent, and GCI modes
- Multi-channel controllers (MCCs): HDLC and Transparent modes
- Serial peripheral interface (SPI)
- Inter-integrated circuit (I^2C)

This application note describes the structure of the parameter RAM and buffer descriptors and provides examples in C for defining them.

CONTENTS

1 Parameter RAM

The parameter RAM is a section of memory located in Banks 9 and 10 of the dual-port RAM, as shown in **Table 1**. Each bank is 2 KB. Bank 9 is located at an offset of 0x8000 from the Internal Space Base (ISB) which is defined in the Internal Memory Map Register (IMMR). Bank 10 is located at an offset of 0x8800 from ISB. For example, if the IMMR = 0x14705000, then the ISB is at address 0x14700000. The absolute start addresses of Bank 9 and Bank 10 are 0x14708000 and 0x1478800, respectively.

Table 1. Dual-Port RAM Memory Map

Offset from ISB	Bank	Storage	Size
0x0000	1	BD/Data/Code	2 KB
0x0800	2	BD/Data/Code	2 KB
0x1000	3	BD/Data/Code	2 KB
0x1800	4	BD/Data/Code	2 KB
0x2000	5	BD/Data/Code	2 KB
0x2800	6	BD/Data/Code	2 KB
0x3000	7	BD/Data/Code	2 KB
0x3800	8	BD/Data/Code	2 KB
0x4000	Reserved		16 KB
0x8000	9	Parameter RAM	2 KB
0x8800	10	Parameter RAM	2 KB
0x9000	Reserved		8 KB
0xB000	11	FCC Data	2 KB
0xB800	12	FCC Data	2 KB

1.1 Parameter RAM Memory Map

The MSC8101/MSC8103 parameter RAM structure memory map is shown in **Table 2**. Because the exact definition of the parameter RAM differs for each protocol, the number of parameters for each protocol varies. For example, SCC1 has 256 bytes allocated for parameters but the SCC1 UART protocol uses only 102 bytes and the SCC1 HDLC protocol uses only 92 bytes. Unused parameter RAM can be used for storage area for data buffers.

Some protocols, such as SMC*x*, SPI, and I^2C have only two bytes allocated for storage in the parameter RAM. Parameters are not stored in these two bytes. Instead, these two bytes contain a user-programmable pointer to the location where the parameters are stored. For example, ISB+0x8AFC contains a pointer to the I^2C parameters that can be placed in the dual-port RAM.

Table 2. Parameter RAM Memory Map

Bank	Offset from ISB	Peripheral	Size (Bytes)	Bank	Offset from ISB	Peripheral	Size (Bytes)
Bank 9	0x8000	SCC1	256	Bank 10	0x8800	MCC2	128
	0x8100	SCC2	256		0x88FC	SMC2	2
	0x8200	SCC3	256		0x88FE	IDMA2	2
	0x8300	SCC4	256		0x89FC	SPI	2
	0x8400	FCC1	256		0x89FE	IDMA3	2
	0x8500	FCC2	256		0x8AE0	RISC Timers	16
	0x8600	FCC3	256		0x8AF0	REV_NUM	2
	0x8700	MCC1	128		0x8AF8	RAND	4
	0x87FC	SMC1	2		0x8AFC	I^2C	2
	0x87FE	IDMA1	2		0x8AFE	IDMA4	2

1.2 General Parameters

Although each protocol has a different set of parameters, some parameters are shared by many protocols. This section describes parameters that are common to many protocols:

- *RBASE and TBASE*. Define the base addresses for the receive buffer descriptor (BD) and the transmit BD. These parameters define the starting location in the memory map for the BDs. RBASE and TBASE are each 32 bits long when the FCC controller is used and 16 bits long when any controller other than the FCC is used. These parameters define the offset from the beginning of dual-port RAM. The BD tables can be placed in Banks 1 – 8 of the dual-port RAM or in any unused parameter RAM area. Because each BD is 8 bytes, RBASE and TBASE values should be multiples of 8.
- *RBPTR and TBPTR*. During frame processing, RBPTR points to the current receive BD. In idle state, RBPTR points to the next receive BD. Similarly, TBPTR points to the current transmit BD during frame transmission and to the next transmit BD in idle state.
- *RFCR and TFCR*. Specify byte ordering, transfer code, and bus location of data and BDs. The same information is specified in RSTATE and TSTATE when the FCC controller is used.
- *MRBLR*. Defines the maximum number of bytes the receiver writes to a receive buffer before moving to the next buffer. The receiver can write fewer bytes than MRBLR if an error condition or an end-of-frame occurs. It never writes more bytes than the MRBLR value, so the user-supplied buffers should be at least as large as the MRBLR. The MRBLR should be greater than zero. The size of the transmit buffers is not affected by the MRBLR value. The MRBLR should not be changed dynamically while the receiver is operating.

1.3 SCC Parameter RAM Example

Table 3 shows the parameter RAM for all SCC protocols. The boldfaced entries are the parameters that you must initialize in order to enable the SCC. Refer to the product reference manual for the protocol-specific parameters.

Table 3. SCC Parameter RAM

Offset from ISB+0x8000	Name	Width	Description
0x00	RBASE	16 bits	**RxBD/TxBD table base address**. Offset from the beginning of dual-port RAM. The BD tables can be placed in any unused portion of Banks 1 – 8. The CP starts BD processing at the top of the table. These values need to be initialized before the corresponding channels are enabled. RBASE and TBASE values must be multiples of 8.
0x02	TBASE	16 bits	
0x04	RFCR	8 bits	**Rx/Tx function code**. Contains the transaction specification associated with SDMA channel accesses to external memory.
0x05	TFCR	8 bits	
0x06	MRBLR	16 bits	**Maximum receive buffer length**. Defines the maximum number of bytes the MSC8101/MSC8103 writes to a receive buffer before it goes to the next buffer. The MSC8101/MSC8103 can write fewer bytes than MRBLR if an error or an end-of-frame occurs. It never writes more bytes than the MRBLR value. MRBLR should be changed only while the receiver is disabled.
0x08	RSTATE	32 bits	**Rx internal state**. For CP use only.
0x0C	—	32 bits	**Rx internal buffer pointer**. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	16 bits	**Current RxBD pointer**. Points to the BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the BD table is reached, the CP initializes RBPTR to the value in RBASE.
0x12	—	16 bits	**Rx internal byte count**. Down-count value initialized with MRBLR and decremented with each byte written by the supporting SDMA channel.
0x14	—	32 bits	**Rx temp**. For CP use only.
0x18	TSTATE	32 bits	**Tx internal state**. For CP use only.
0x1C	—	32 bits	**Tx internal buffer pointer**. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	16 bits	**Current TxBD pointer**.
0x22	—	16 bits	**Tx internal byte count**. Down-count value initialized with TxBD.length and decremented with each byte read by the supporting SDMA channel.
0x24	—	32 bits	**Tx temp**. For CP use only.
0x28	RCRC	32 bits	**Temp receive/transmit CRC**. Does not need to be accessed for normal operation but may be helpful for debugging.
0x2C	TCRC	32 bits	
0x30	—		**Protocol-specific area**.

2 Buffer Descriptors

A BD contains the essential information about each buffer in memory. Each buffer is referenced by a BD that can reside anywhere in dual-port RAM. Each 64-bit BD has the structure shown in **Figure 1**. This structure is common to all the serial controllers. A receive BD (RxBD) table and a transmit BD (TxBD) table are associated with each serial controller. A BD table can have one or more BDs. The base address of the BDs is defined in the RBASE and TBASE parameters and should be multiples of 8 bytes.

Figure 1. Buffer Descriptor Structure

Offset from RBASE or TBASE (multiple of 8)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	Status and Control															
0x2	Data Length															
0x4	High-Order Buffer Pointer															
0x6	Low-Order Buffer Pointer															

- *Status and Control.* The 16-bit value at offset+0x0 contains status and control bits that control and report status information on the data transfer. The CPM updates the status bits after the buffer is sent or received. Only this field differs for each protocol. Refer to the product reference manual for each protocol's status and control field bit descriptions.
- *Data Length.* The 16-bit value at offset+0x2 contains the number of bytes sent or received.
- *RxBD Data Length*. The number of bytes the communications processor (CP) writes into the RxBD buffer once the BD closes. The CP updates this field after the received data is placed into the buffer and the buffer is closed. You do not need to initialize this field.
- In frame-based protocols, except for the SCC transparent mode, the RxBD data length field contains the total frame length including CRC bytes. If a received frame's length, including CRC, is an exact multiple of the parameter RAM maximum receive buffer length MRBLR, the last BD's buffer holds no actual data but the BD contains the total frame length.
- TxBD Data Length. The number of data bytes the controller needs to transmit from its buffer. The CP never modifies this field. You must initialize this field.

2.1 Buffer Pointer

The 32-bit data at offset+0x4 points to the beginning of the buffer in internal or external memory. For an RxBD the buffer pointer value must be a multiple of four to be word-aligned. For a TxBD the buffer pointer value can be even or odd.

2.2 BD Naming Conventions

In this discussion, the BD type and field values use the convention **`BD.field`**. **Table 1** shows the possible BD and field values. Individual bits in the BD status and control field are referred to as **`BD.cstat.bit`**. Consult the product reference manual for the specific protocol's status and control field bit definition.

Table 4. Buffer Descriptor Name Convention

BD Name	Field Name	
`RxBD` or `TxBD`	cstat	Status and Control
	length	Data Length
	addr	Address Pointer

Examples of BD naming conventions are as follows:

- **TxBD.cstat.R** refers to the ready bit in the TxBD's status and control field.
- **RxBD.length** refers to RxBD's data length field.
- **RxBD.addr** refers to RxBD's buffer pointer field.

3 BD and Buffer Memory Structure

The BDs of all protocols can point to data buffers located in the internal dual-port RAM. Banks 1–8, which are located at ISB+0x0 through ISB+0x4FFF, are available for storing BDs and their buffers. However, if the data buffers are large, they can be located in external memory.

3.1 SCC2 Example

Figure 2 shows that the SCC2 parameter RAM is located at ISB+0x8100. The RBASE parameter contains a pointer to the base address of the RxBD table that is located in the dual-port RAM at ISB+0. Since there are three RxBDs, the first RxBD is located at ISB+0x0, the second is located at ISB+0x08, and the third is located at ISB+0x10. The TBASE parameter contains a pointer to the base address of the TxBD table. Since the TxBD table immediately follows the RxBD table, the first TxBD table is located at ISB+0x18, the second is at ISB+0x20, and the third is located at ISB+0x28. **`RxBD.addr`** contains a pointer to the receive buffer in external memory, and **`TxBD.addr`** contains a pointer to the transmit buffer in external memory.

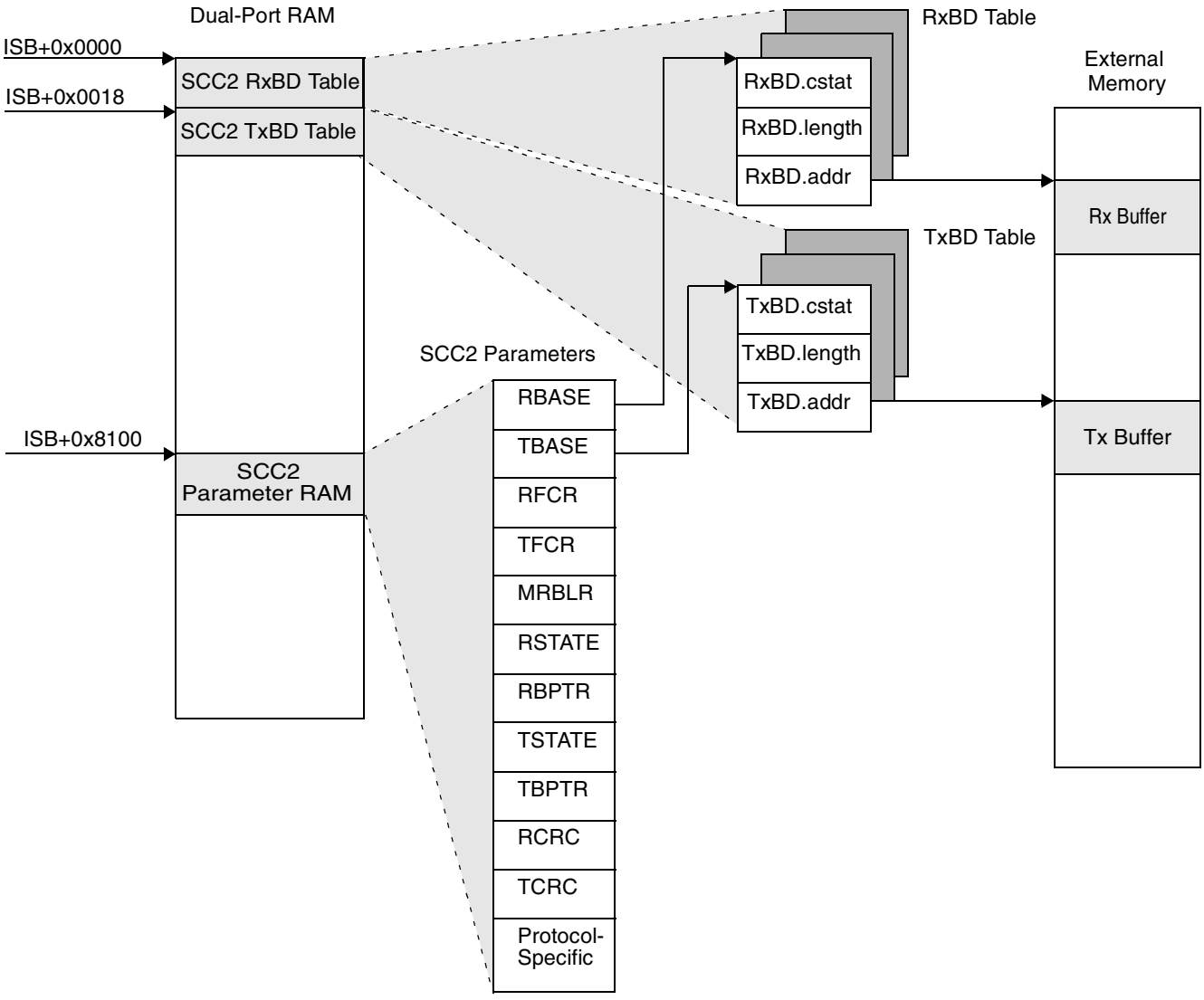

Figure 2. SCC2 BD and Buffer Memory Structure

3.2 SPI Example

Figure 3 shows that the 2-byte SPI_BASE parameter RAM is located at ISB+0x89FC, which contains a pointer to the SPI parameter table. The SPI parameter table can be placed at any 64-byte aligned address in the dual-port RAM's general-purpose area or in Banks 1–8. In the example presented here, the parameter table is placed at ISB+0x0100. The RBASE parameter contains a pointer to the base address of the RxBD table, which is located in the dual-port RAM at ISB+0. Since there are three RxBDs, the first RxBD is located at ISB+0x0, the second is located at ISB+0x08, and the third is located at ISB+0x10. The TBASE parameter contains a pointer to the base address of the TxBD table. Since the TxBD table immediately follows the RxBD table, the first TxBD table is located at ISB+0x18, the second is at ISB+0x20, and the third is at ISB+0x28.

`RxBD.addr` contains a pointer to the receive buffer in external memory, and **`TxBD.addr`** contains a pointer to the transmit buffer in external memory.

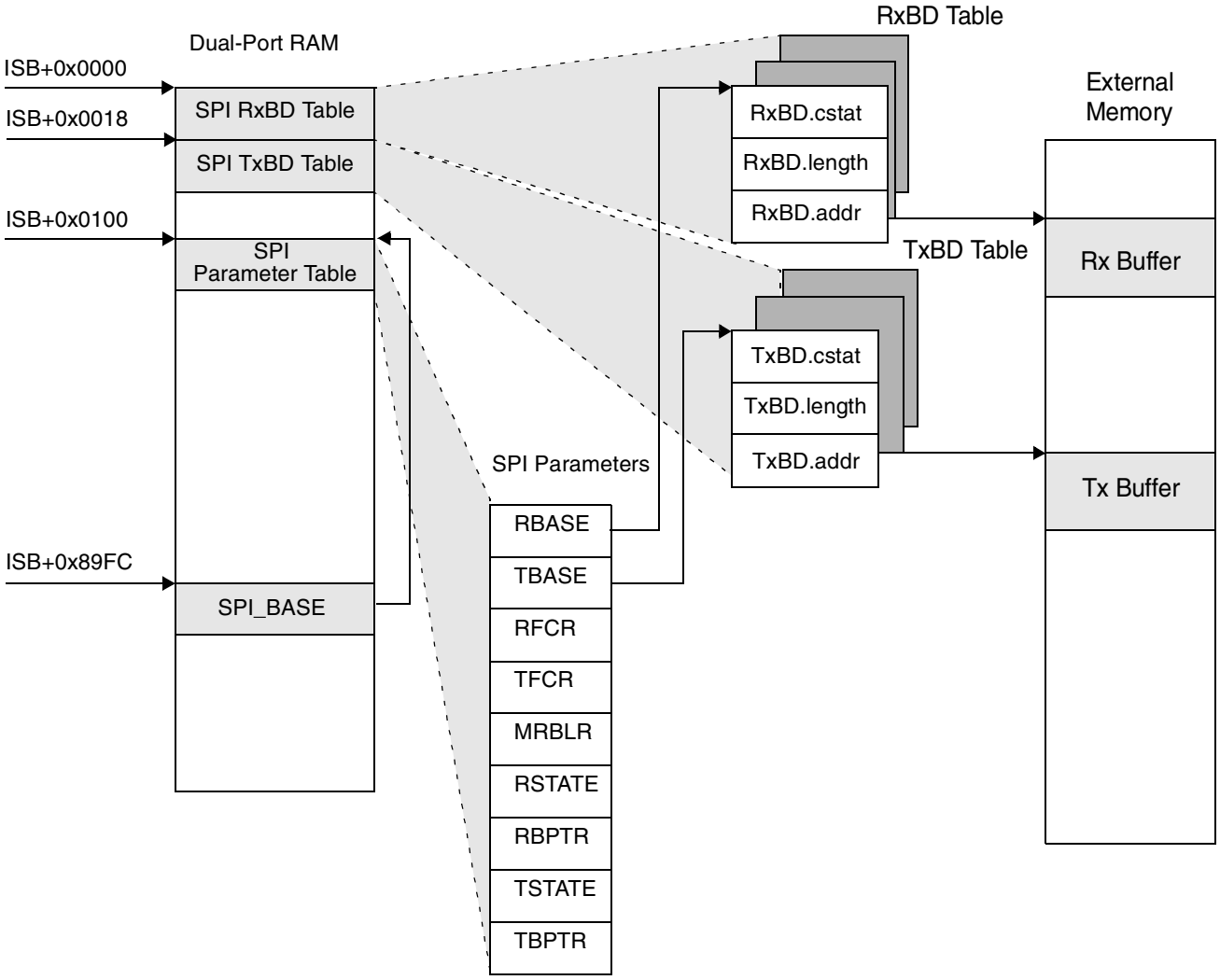

Figure 3. Example SPI BD and Buffer Memory Structure

4 Initialization Examples

This section gives examples of how to initialize I^{2}C parameters, BD tables, and data buffers using the C programming language.

4.1 Parameter RAM

Example 1 shows that the I^{2}C parameter RAM table is implemented as a structure called `t_I2c_Pram`, with each parameter being a structure member. The size of each parameter is specified by the following definitions:

- `VUBYTE` is 8 bits
- `VUHWORD` is 16 bits
- `VUWORD` is 32 bits

Example 1. Setting up the I^{2}C Parameter RAM Table

```
typedef struct
{
      VUHWORD rbase;          /* RX BD base address */
      VUHWORD tbase;          /* TX BD base address */
      VUBYTE  rfcr;           /* Rx function code */
      VUBYTE  tfcr;           /* Tx function code */
      VUHWORD mrblr;          /* Rx buffer length */
      VUWORD  rstate;         /* Rx internal state */
      VUWORD  rptr;           /* Rx internal data pointer */
      VUHWORD rbptr;          /* Rx BD Pointer */
      VUHWORD rcount;         /* Rx internal byte count */
      VUWORD  rtemp;          /* Rx temp */
      VUWORD  tstate;         /* Tx internal state */
      VUWORD  tptr;           /* Tx internal data pointer */
      VUHWORD tbptr;          /* Tx BD pointer */
      VUHWORD tcount;         /* Tx byte count */
      VUWORD  ttemp;          /* Tx temp */
}t_I2c_Pram;
```

In this example, assume that the internal space base address is initialized to 0x14700000 and that the I^{2}C parameter table needs to be located at 0x14703800. **Example 2** shows that `I2CPRAM`, which is a pointer to the I^{2}C parameter table structure `t_I2c_Pram`, is set to address 0x14703800. The I^{2}C base address is configured as an array of two bytes as shown in the last two lines in **Example 2**. The first and second arrays consist of the upper and lower bytes of the I^{2}C base address, respectively. These lines of code set the I^{2}C base address to an offset of 0x3800 from the dual-port RAM.

Example 2. Setting the I^{2}C Parameter Table Location

```
t_I2c_Pram *I2CPRAM;              /* I2C Parameter RAM pointer */
I2CPRAM = (t_I2c_Pram *)(0x14703800);/* I2C Parameters base address */
IMM->pram.standard.i2c[0] = 0x38;
IMM->pram.standard.i2c[1] = 0x00;
```

4.2 Buffer Descriptors and Data Buffers

Example 3 shows that the BD table is implemented as a structure called BD, with each BD field being a structure member. The BDRINGS structure allows for multiple RxBDs and TxBDs by implementing the BD table as an array of BDs. For example, the first RxBD in the table is RxBD[0], the second is RxBD[1], and the last RxBD in the table is RxBD[NUM_RXBDS-1].

Example 3. Setting up the Buffer Descriptor Tables

```
typedef struct BufferDescriptor
{
      unsigned short cstat;      /* control and status*/
      unsigned short length;     /* data length                    */
      char* addr;                /* buffer address                 */
} BD;

typedef struct BufferDescRings
{
      BD RxBD[NUM_RXBDS];     /* Rx BD ring */
      BD TxBD[NUM_TXBDS];     /* Tx BD ring */

} BDRINGS;
```

In this example, assume that the RxBD and TxBD tables need to be located at the start of the dual-port RAM at 0x14700000. **Example 4** shows that RxTxBD, which is a pointer to the BDRINGS structure, is set to point to address 0x14700000. The last two lines of code set the parameters RBASE and TBASE to the start of the first RxBD and the first TxBD in the tables, respectively.

Example 4. Setting the Buffer Descriptor Table Location

```
RxTxBD = (BDRINGS *)(0x1470000);        /* Pointer to BD area of DPRAM */
I2CPRAM->rbase = (UHWORD) & RxTxBD->RxBD[0]; /* point RBASE to first RX BD */
I2CPRAM->tbase = (UHWORD) & RxTxBD->TxBD[0]; /* point TBASE to first TX BD */
```

Example 5 shows how to set the first RxBD field values. Assume that rxbuf[0] is the buffer referenced by RxBD[0]. This buffer can be placed in the dual-port RAM or in external memory.

Example 5. Setting the Buffer Descriptor Field Values

```
RxTxBD->RxBD[0].cstat = 0xB000;
RxTxBD->RxBD[0].length = 0;
RxTxBD->RxBD[0].addr = (char *) & rxbuf[0];
```

5 RxBD Processing Example

Figure 4 shows how the RxBD is processed in the SCC UART mode. This example assumes that the maximum receive buffer length MRBLR is 8 bytes. The MRBLR is the number of bytes the MSC8101/MSC8103 writes to a receive buffer before it moves to the next buffer. However, the MSC8101/MSC8103 can write fewer bytes than the MRBLR value if an error or end-of-frame (for frame-based protocols) occurs. It never writes more bytes than the MRBLR value, so the receive buffers cannot be smaller than the MRBLR.

When data arrives, the CP moves the data to the buffer to which the first RxBD in the table points. The CP continues to move data until the buffer is full or an error occurs. When the buffer is full or an error occurs, the buffer is closed. Subsequent data uses the next BD. If RxBD.cstat.E is cleared, the current buffer is not empty

and it reports a busy error. The CP does not move from the current BD until the SC140 core sets RxBD.cstat.E to indicate that the buffer is empty. After using a descriptor, the CP clears RxBD.cstat.E and does not reuse a BD until the SC140 core has processed it. However, in continuous mode when RxBD.cstat.CM is set, RxBD.cstat.E remains set to allow the buffer to be overwritten when the CP accesses this BD again. When the CP discovers a descriptor's RxBD.cstat.W (wrap) is set, indicating that it is the last BD in the circular BD table, it returns to the beginning of the table when it is time to move to the next BD.

When the UART receives idle characters (all ones), the channel begins counting consecutive idle characters received. If the maximum idle characters MAX_IDL is reached, RxBD.cstat.ID is set, the buffer is closed and an interrupt is generated if not masked.

When the UART receives no stop bit it reports framing errors. The channel writes the received character to the buffer, closes it, sets RxBD.cstat.FR, generates an interrupt if not masked, and increments the received characters with framing error counter FRMEC. A new receive buffer is used to receive subsequent data.

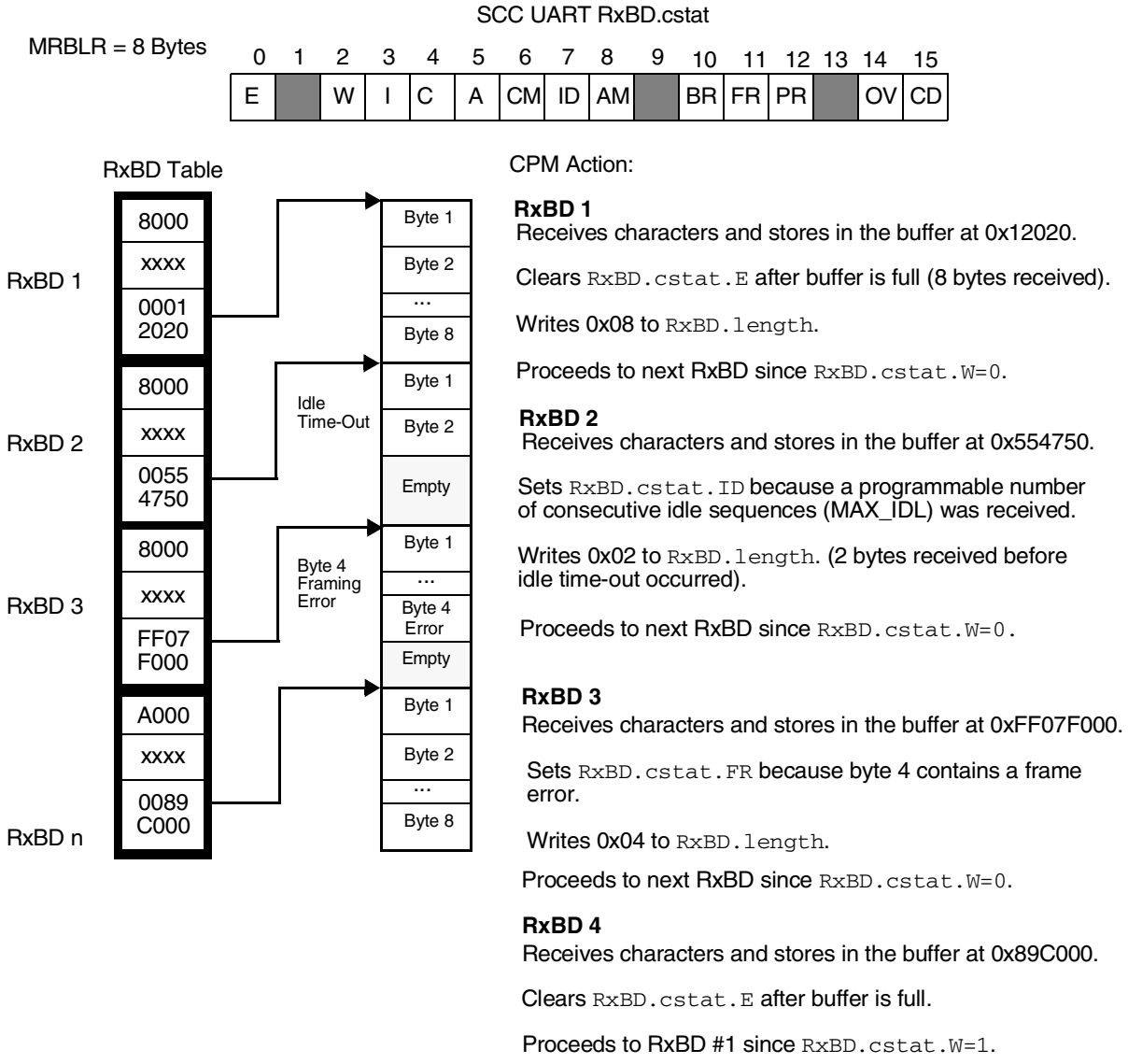

Figure 4. Example SCC UART RxBD Processing

6 TxBD Processing Example

Figure 5 shows how the TxBD is processed in the SCC UART mode. When the CP detects that the `TxBD.cstat.R` (ready) is set, it starts transmitting the buffer. After the buffer is transmitted, the CP waits for the next descriptor's `TxBD.cstat.R` to be set before proceeding. When the CP detects that a descriptor's `TxBD.cstat.W` (wrap) is set, indicating this BD is the last in the BD table, it returns to the start of the BD table after this last BD is processed. The CP clears `TxBD.cstat.R` (not ready) after using a TxBD, which keeps it from being retransmitted before it is confirmed by the SC140 core. However, some protocols support a continuous mode for which `TxBD.cstat.R` remains set after the buffer is closed to allow the buffer to be resent next time the CP accesses this BD. Continuous mode is enabled by setting `TxBD.cstat.CM`.

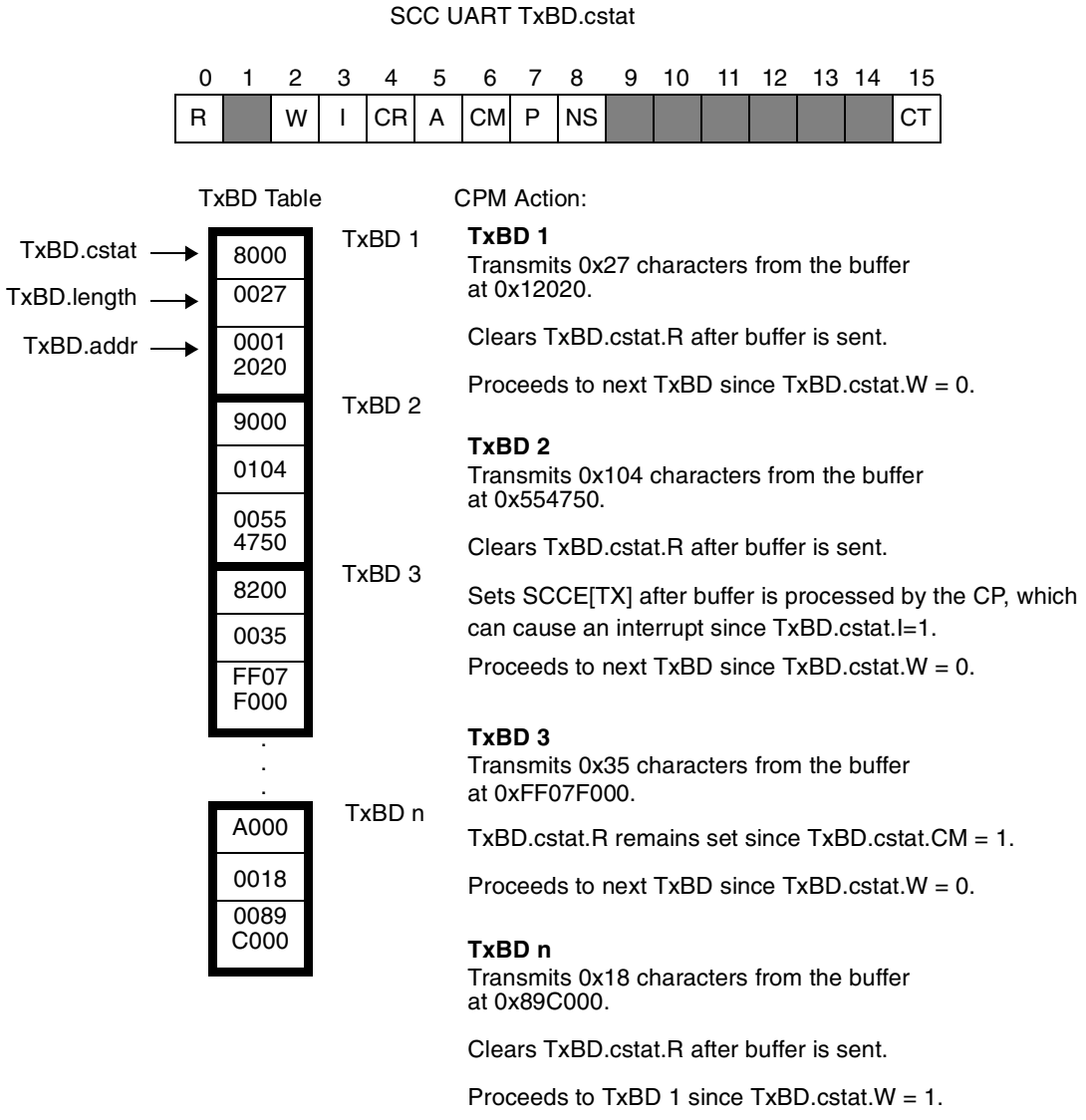

Figure 5. Example SCC UART TxBD Processing

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

AN2147
Rev. 2
11/2004

freescale™
semiconductor