Freescale Semiconductor
Application Note

Document Number: AN3365
Rev. 1, 10/2007

MSC8144 TDM Unified Buffer Mode

by *Tina Redheendran and Yaniv Kagan*
Freescale Semiconductor, Inc.

The MSC8144 time-division multiplexing (TDM) interface enables communication with a variety of serial devices over a single bus. The TDM interface connects gluelessly to common telecommunication frames, such as T1 and E1, and it can interface with multiple buses, such as H.110 devices and codecs. This application note presents an example of how to use the unified buffer mode of the TDM. A basic knowledge of the MSC8144 TDM is assumed. The MSC8144 reference manual is available at the web site listed on the back cover of this document.

Contents

1 TDM Basics

The TDM interface is composed of eight identical and independent TDM modules, each supporting 256 channels running at up to 62.5 Mbps with 2-, 4-, 8-, and 16-bit word size. The TDM can be configured by all four SC3400 cores, as well as by an external host. The TDM modules are highly programmable, with register bit settings for each TDM module for many parameters including the direction of the bits in the channel (MSB first/LSB first), the direction of TSYN (input or output), the polarity of the clock (sample/drive at clock rise or fall), and the polarity of TSYN/RSYN/FSYN (positive or negative). Each TDM module also includes separate configuration bits for transmit and receive unified buffer modes.

freescale semiconductor

1.1 TDM Local Memory

The receive and transmit data of each channel is stored in a different buffer. These buffers can be located in any of the internal or external memories shared by all the SC3400 cores. In addition, received data is temporarily stored in the TDM receive local memory until it is transferred to the receive buffers, as shown in the example in Figure 1. Data transmitted from the transmit buffers is temporarily stored in the TDM transmit local memory until it is transferred externally, as shown in the example in Figure 2. Both examples use two channels, four bits per channel, and four local memory buffers. In Figure 1, receive data from each channel is written serially to the receive local memory and transferred to independent data buffers. In Figure 2, each data channel transfers data from an independent data buffer to the TDM local memory, and transmit data is read out serially from the transmit local memory.

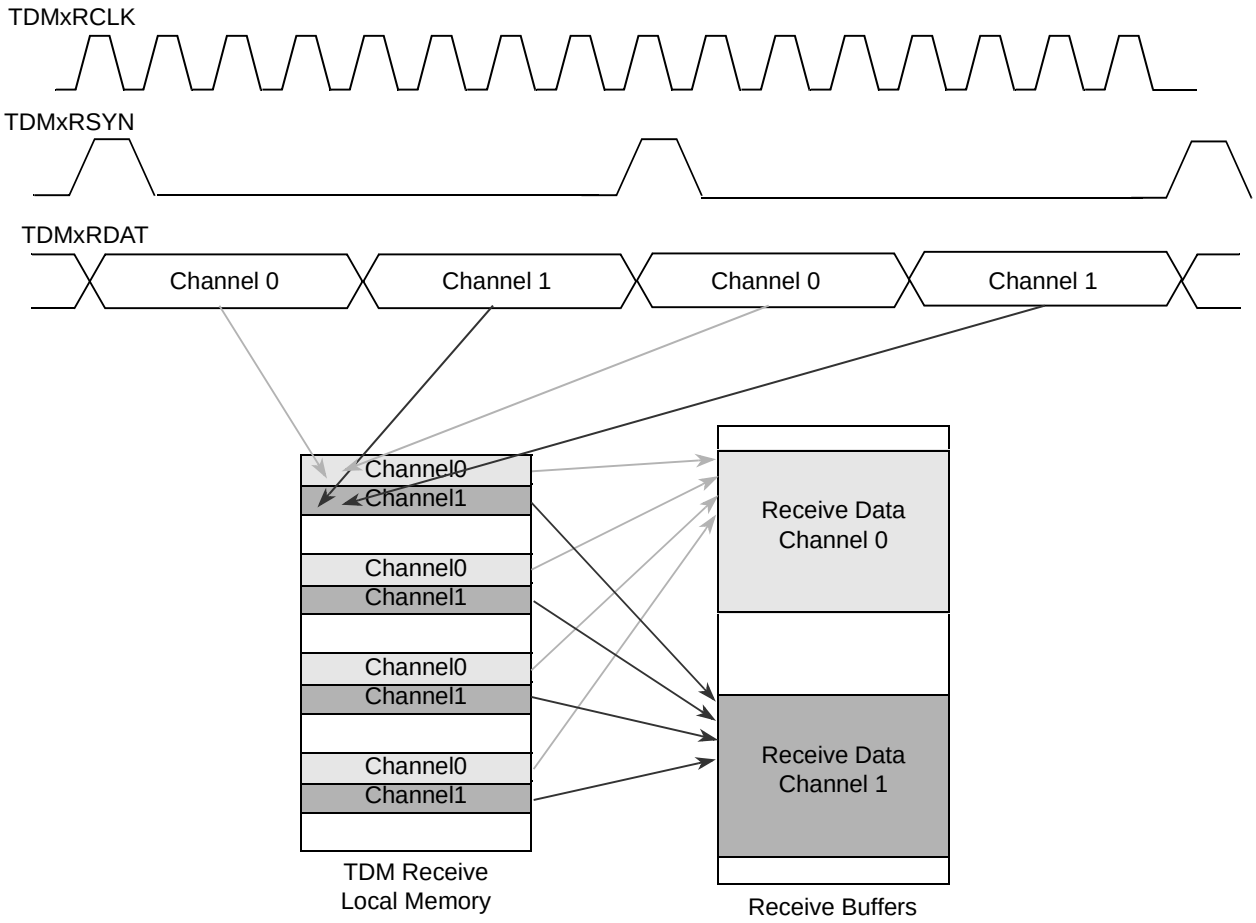

Figure 1. TDM Receive Example

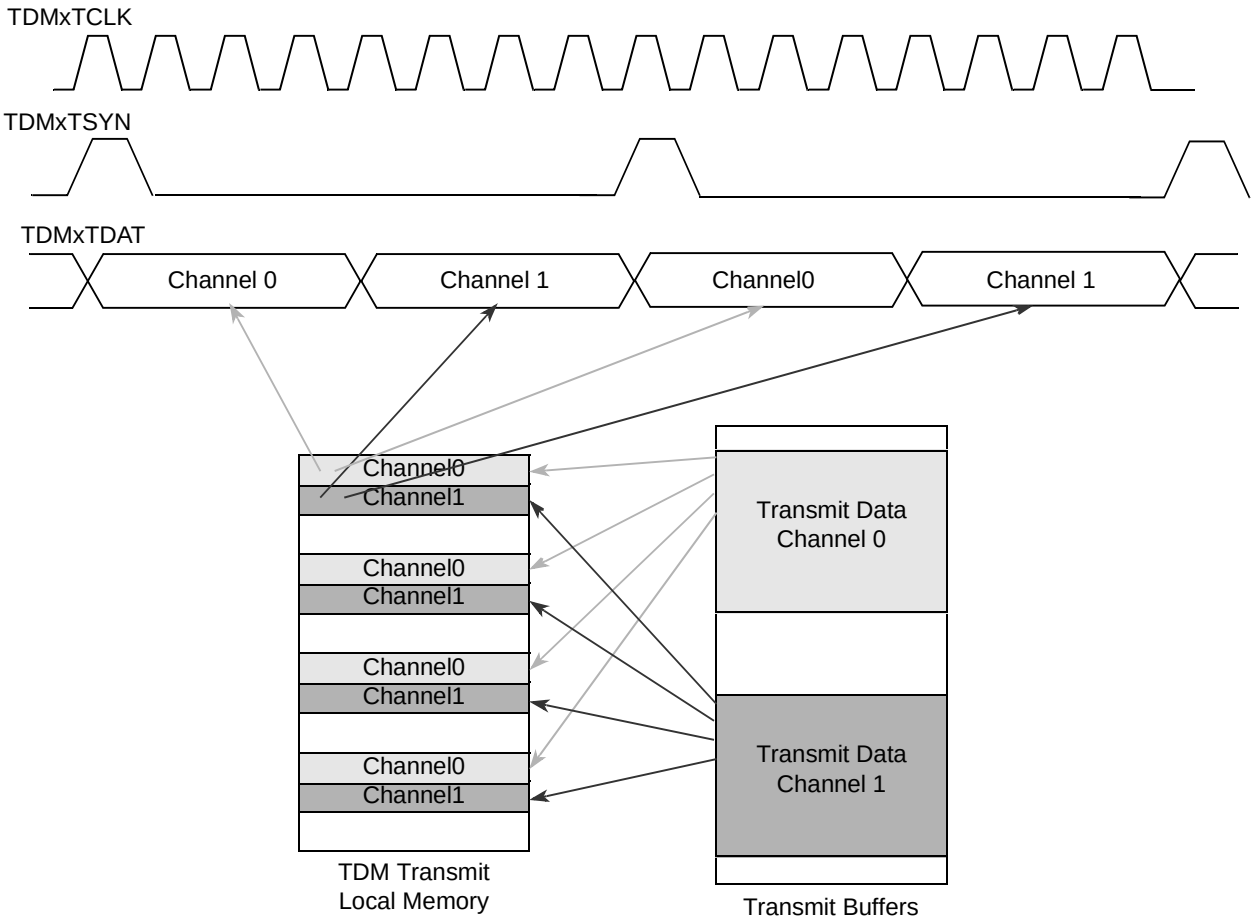

Figure 2. TDM Transmit Example

1.2 Unified Buffer Mode

Unified buffer mode is a special mode of the TDM that creates a one-channel link that is typically used in point-to-point connections. To use unified buffer mode, the number of active links must be one (TDMxGIR:RTSAL = 0b0000 or 0b0100 or 0b1100).

For receive channels, unified buffer mode is enabled when the TDMxRFP[RUBM] bit is set. When unified buffer mode is enabled for receive channels, the receive channels are directed to one buffer, as shown in Figure 3. All the channel parameters are located in the TDMxRCPR0 register.

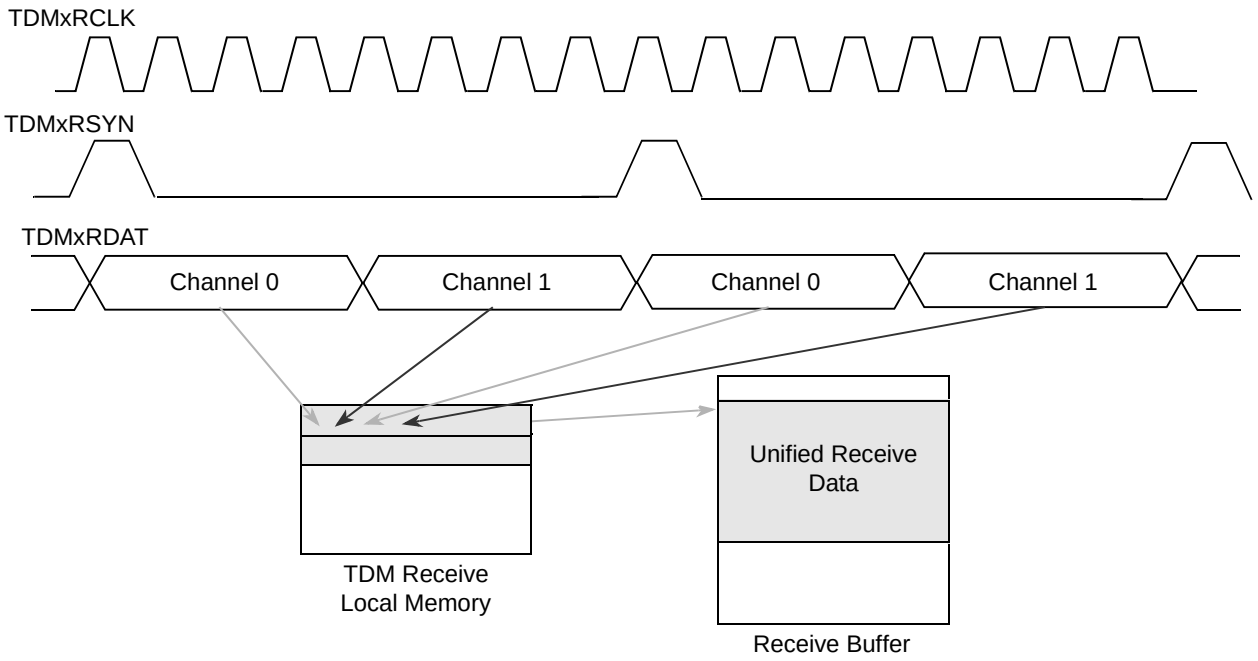

Figure 3. TDM Unified Buffer Mode Receive Example

For transmit channels, unified buffer mode is enabled when the TDMxTFP[TUBM] bit is set. When unified buffer mode is enabled for transmit channels, data is transmitted from one buffer into the transmit channels as shown in Figure 4. The channel parameters are located in the TDMxTCPR0 register.

NOTE

When the TDMxTFP[TUBM] bit is set, the first block of data that is initialized for transmission in the memory is not transmitted. The size (number of bits) of the non-transmitted block is determined by the following equation: $2 \times (\text{TDMxTFP[TNCF]} - 1) \times (\text{TDMxTFP[TCS]} + 1)$.
For example, if the number of transmit channels is 32 (that is, TDMxTFP[TNCF] = 31), and the transmit channel size is 8 (that is, TDMxTFP[TCS] = 7), then the number of non-transmitted bits is 480. The non-transmitted bits are located either in the TDM local memory and/or in the device level memory (M2, M3, DDR, and so forth).

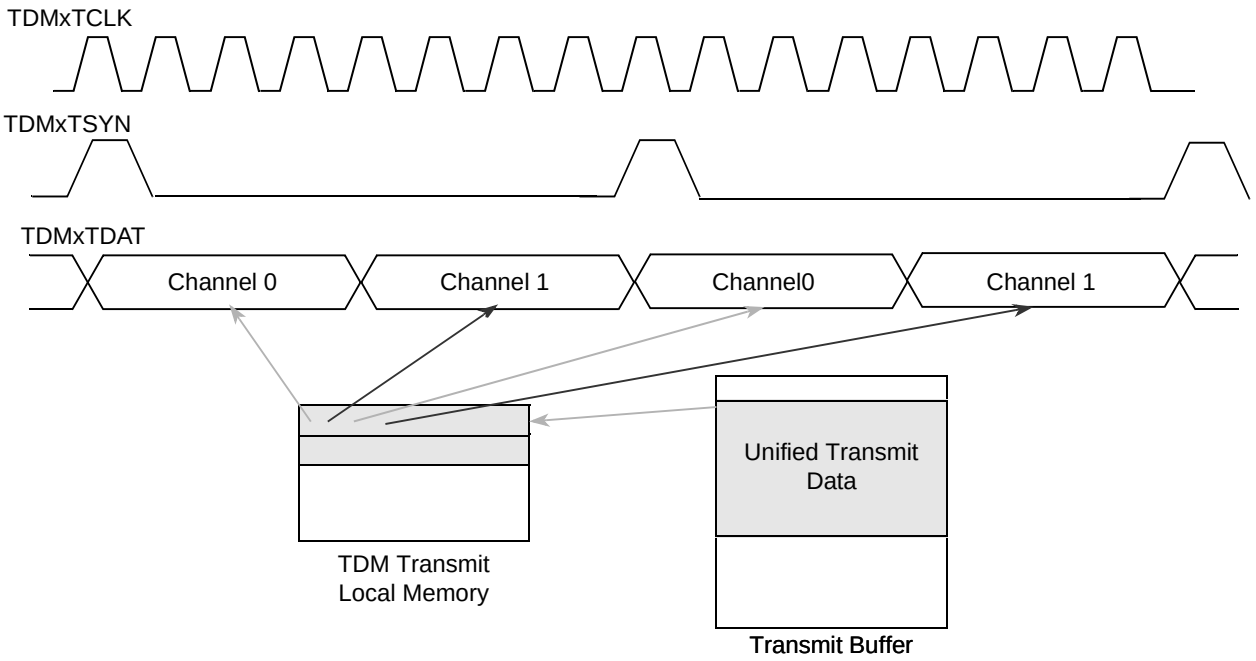

Figure 4. TDM Unified Buffer Mode Transmit Example

2 Code for Unified Buffer Mode

This section presents the example code that illustrates use of the TDM unified buffer mode. The MSC8144ADS and CodeWarrior™ for StarCore are used to download and run the code on the MSC8144 DSP. Because the MSC8144ADS does not include easy access to the TDM pins to make external connections, this example uses TDM0 in internal loopback mode. TDM0 transmits 32 channels of 8-bit data in unified buffer mode and receives the data in normal (non-unified buffer) mode. This allows us to see how the data is stored differently in the two modes. The MSC8144ADS already has a 2 MHz clock and an 8 KHz sync present at the TDM0TCLK pin.

The example code includes one CodeWarrior project for the MSC8144 target. Because TDM0 is used in internal loopback mode and the clock and sync for TDM0 are already present on the MSC8144ADS, no hardware setting are needed for this code.

2.1 Register Settings

The MSC8144 project programs the TDM0 configuration and control registers. Table 1 shows the important register settings for TDM0 transmit and receive. After these registers are programmed, the transmit and receive local memory and data buffers are initialized.

Table 1. TDM0 Register Settings

Register	Setting	Comments
TDM0GIR	0x0000000C	RTSAL = 1100 (Rx/Tx share sync and clock, 1 full duplex data link)
TDM0TIR	0x0001C016	Sets Transmit Byte Order, Transmit Frame Sync Delay, Transmit Data Edge, and Transmit Frame Sync Edge
TDM0TFP	0x0000031D | ((NUM_OF_CHANNELS-1)<<16)	TNCF = 0x1F (Tx 32 channels, NUM_OF_CHANNELS = 32) TCS = 0x7 (Tx channel size 8 bits) TUBM = 1 (Each channel reads from the same data buffer) TCDBL = 0x3 (512 bits in local memory for each channel)
TDM0TDBS	BUFFER_SIZE × NUM_OF_CHANNELS – 1	Tx data buffer size = 512 bytes (BUFFER_SIZE = 16)
TDM0RIR	0x0001C002	Sets Receive Byte Order and Receive Frame Sync Edge
TDM0RFP	0x0000031C | ((NUM_OF_CHANNELS-1)<<16)	RNCF = 0x31 (Rx 32 channels, NUM_OF_CHANNELS = 32) RCS = 0x7 (Rx channel size 8 bits) RUBM = 0 (All channels written to a different data buffer) RCDBL = 0x3 (512 bits in local memory for each channel)
TDM0RDBS	BUFFER_SIZE – 1	Rx data buffer size = 16 bytes (BUFFER_SIZE = 16)

Notice that the number of channels is set to 32 and the number of bits per channel is set to 8 for both transmit and receive. Also, notice that the TUBM bit is configured for the transmit to enable unified buffer mode and the buffer size is 32 times bigger than the receive buffer size, because all 32 receive channels are transmitted from the same buffer.

2.2 Data Results

After the TDM0 registers are programmed, the MSC8144 project enables the TDM0 transmit and receive. The TDM is allowed to run until the first 16 bytes are received in each of the 32 channels. After that, the TDM is activated again for a short period and then it is disabled. The received values are checked both times. Figure 5 shows the transmit and receive data for this example.

The transmit data consists of one unified 32-byte buffer, as shown on the upper part of Figure 5. In addition, all the transmit local memory of channel 0 is initialized with 0x11111111 and consists of 512 bits. When the TDM is activated, the transmit local memory is transmitted first. As explained in Section 1.2, because the unified buffer mode is activated on transmit, the first 480 bits of data are not transmitted. Thus only the last 32 bits of the transmit local memory are transmitted.After the data is transmitted in unified buffer mode (as shown in Figure 4) and received in normal mode, the results are as shown on the bottom part of Figure 5. The receive data consists of thirty-two 16-byte channels. The first 8 bits of transmitted data goes to channel 0, the next 8 bits go to channel 1, and so forth. The left side of the figure shows that only the first 32 transmitted bits (the first 8 bits of channels 0–3) are from the initial transmit local buffer.

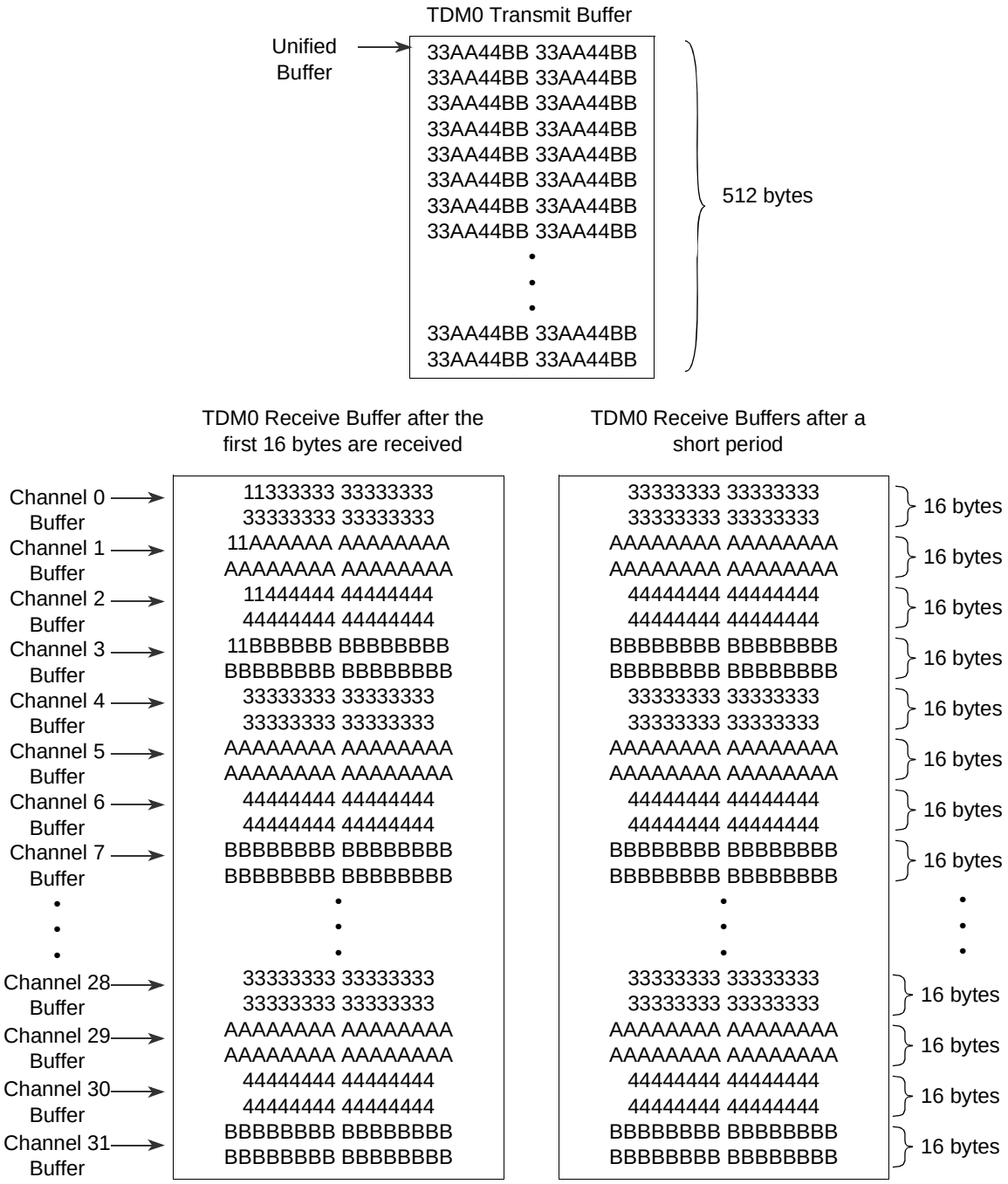

Figure 5. TDM Example Data

2.3 Example Code

This section shows the example code for the TDM setup routine. This code initializes the TDM0 registers as described in Section 2.1, “Register Settings.

Example 1. TDM Setup Code

```
//*************************************************************************
// Initialize TDM Registers
//*************************************************************************

     g_msc814x_ccsr->tdm[0].tdmgir = 0x0000000C;

     g_msc814x_ccsr->tdm[0].tdmtir = 0x0001c016;
     g_msc814x_ccsr->tdm[0].tdmrir = 0x0001c002;

     g_msc814x_ccsr->tdm[0].tdmtfp = 0x0000031d |((NUM_OF_CHANNELS-1)<<16);
     g_msc814x_ccsr->tdm[0].tdmrfp = 0x0000031c |((NUM_OF_CHANNELS-1)<<16);

     g_msc814x_ccsr->tdm[0].tdmtdbs = BUFFER_SIZE * NUM_OF_CHANNELS - 1;
     g_msc814x_ccsr->tdm[0].tdmrdbs = BUFFER_SIZE - 1;

     g_msc814x_ccsr->tdm[0].tdmtgba = 0xC000;
     g_msc814x_ccsr->tdm[tdm].tdmrgba = 0xC000;

     g_msc814x_ccsr->tdm[0].tdmtcpr[0] = (Txbuffer_addr & 0x00FFFFFF) | 0x80000000;
     for(k=0; k<NUM_OF_CHANNELS; k++)
     {
          g_msc814x_ccsr->tdm[0].tdmrcpr[k] = (Rxbuffer_addr[k] & (0x00FFFFFF) |
     0x80000000;
     }

//*************************************************************************
// Initialize the buffers for channel 0 in the TDM Tx local memory.
//*************************************************************************
     for(k=0; k<8; k++)
     {
          g_msc814x_ccsr->tdm[0].tdmtxlclmem[k] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmtxlclmem[0x100] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmtxlclmem[k+0x200] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmrxlclmem[k+0x300] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmrxlclmem[k+0x400] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmrxlclmem[k+0x500] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmrxlclmem[k+0x600] = 0x11;
          g_msc814x_ccsr->tdm[0].tdmrxlclmem[k+0x700] = 0x11;
     }
```

```
//************************************************************************
// Initialize TDM transmit and receive data buffers
//************************************************************************
     for(k=0; k<(NUM_OF_CHANNELS*BUFFER_SIZE/4); k++)
     {
          Write32(Txbuffer_addr + 4*k,0x33AA44BB);
          Write32(Rxbuffer_addr[ k/(BUFFER_SIZE/4)] + 4*(k%(BUFFER_SIZE/4)) ,0xDEADDEAD;
     }
```

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

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
support.asia@freescale.com

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

Document Number: AN3365
Rev. 1
10/2007

freescale™
semiconductor