



APPLICATION NOTE 3827

Initialization and Configuration of the DS26528 Transceiver

Abstract: This application note describes how to properly configure the DS26524 and DS26828 T1, E1, J1 Transceivers. It contains C style example code and will ease the initial software development by allowing the designer to quickly achieve basic system operation.

Introduction

When developing software for a newly designed telecommunications system, the task of achieving basic device operation is often the toughest undertaking. The [DS26528](#) transceiver adds the complexity of an extensive set of functions and multiport operation. To ease the initial hurdle to getting the system up and running, Dallas Semiconductor created a C code style example which will initialize the devices for basic operation in either T1 or E1 mode. The software developer only has to modify the code for the desired operation and write code for two system-dependant functions. Once the code is compiled it should be ready to load onto the system for test and evaluation.

Code Example

The following code example in **Figure 1** will need some modification before it can be correctly compiled for use in the target system. The code for the function calls 'write(address, data)' and 'wait(millisecond)' are system-dependant, so they need to be written for the current microprocessor environment. The code assumes that the device is mapped into a 16-bit local bus at address offset 0x0000 and that the device data bus is only 8 bits. If this is not the case, either the code can be modified or the function calls can be written to account for this. The code also contains may different settings for certain registers to give the developer several options for items like clock frequency, line coding, and more. Although the code covers a wide range of basic functionality, it is in no way complete. The data sheet should be referenced for any additional desired functionality.

```
/*
Configuration Example For DS26528 running in E1 mode.

This Example assumes E1 operation so the function call for T1 configuration
has been commented out. Simply comment out the E1 configuration function
call and uncomment the T1 configuration function call for T1 operation.

This file follows C style conventions. However actual code for the function
calls listed below are implementation specific and need to be added:

Function Calls: write(address, data), wait(millisecond)

The following comments only indicate some of the possible clock sources
that can be used for either E1 or T1 operation.

Master clock configuration can use multiples of n = 1, 2, 4, or 8
MCLK = Must be a n x 2.048 MHz signal for E1 Operation
MCLK = Must be a n x 1.544 or n x 2.048 MHz signal for T1 Operation

TCLK = Must be a 2.048 MHz Signal for E1 Operation
```

TCLK = Must be a 1.544 MHz signal for T1 Operation

*/

```
void initialization_main()
{
    /* Global Initialization Begin */

    /* The Global Transceiver Clock Control Register is important for proper */
    /* device operation consult the data sheet for all possible configurations. */
    write(0x00F3, 0x00); // GTCCR, E1 Mode MCLK-2.048, BPFs-2.048, BPREF-RCLK1
    // write(0x00F3, 0x01); // GTCCR, E1 Mode MCLK-4.096, BPFs-2.048, BPREF-RCLK1
    // write(0x00F3, 0x02); // GTCCR, E1 Mode MCLK-8.192, BPFs-2.048, BPREF-RCLK1
    // write(0x00F3, 0x03); // GTCCR, E1 Mode MCLK-16.384, BPFs-2.048, BPREF-RCLK1

    // write(0x00F3, 0x08); // GTCCR, T1 Mode MCLK-2.048, BPFs-1.544, BPREF-RCLK1
    // write(0x00F3, 0x09); // GTCCR, T1 Mode MCLK-4.096, BPFs-1.544, BPREF-RCLK1
    // write(0x00F3, 0x0A); // GTCCR, T1 Mode MCLK-8.192, BPFs-1.544, BPREF-RCLK1
    // write(0x00F3, 0x0B); // GTCCR, T1 Mode MCLK-16.384, BPFs-1.544, BPREF-RCLK1

    // write(0x00F3, 0x0C); // GTCCR, T1 Mode MCLK-1.544, BPFs-1.544, BPREF-RCLK1
    // write(0x00F3, 0x0D); // GTCCR, T1 Mode MCLK-3.088, BPFs-1.544, BPREF-RCLK1
    // write(0x00F3, 0x0E); // GTCCR, T1 Mode MCLK-6.176, BPFs-1.544, BPREF-RCLK1
    // write(0x00F3, 0x0F); // GTCCR, T1 Mode MCLK-12.352, BPFs-1.544, BPREF-RCLK1

    // write(0x00F3, 0x90); // GTCCR, E1 Mode MCLK-2.048, BPFs-2.048, BPREF-MCLK
    // write(0x00F3, 0x88); // GTCCR, T1 Mode MCLK-2.048, BPFs-1.544, BPREF-MCLK

    /* Wait 1 ms for clock to settle after configuration */
    wait (1);

    /* Configure global Framer multifunction pins, IBO, and BPCLK. If IBO */
    /* mode is enabled, the RIBOC and TIBOC Framer registers must also be */
    /* configured on a per port basis instead of globally */
    write(0x00F1, 0x00); // GFCCR, Set IBO Disabled, RFL/RFS/BLK, BPCLK-2.048
    // write(0x00F1, 0x10); // GFCCR, Set IBO Disabled, RFL/RFS/BLK, BPCLK-4.096
    // write(0x00F1, 0x20); // GFCCR, Set IBO Disabled, RFL/RFS/BLK, BPCLK-8.192
    // write(0x00F1, 0x30); // GFCCR, Set IBO Disabled, RFL/RFS/BLK, BPCLK-16.384

    // write(0x00F1, 0x50); // GFCCR, Set IBO 2 Devices, RFL/RFS/BLK, BPCLK-4.096
    // write(0x00F1, 0xA0); // GFCCR, Set IBO 4 Devices, RFL/RFS/BLK, BPCLK-8.192
    // write(0x00F1, 0xF0); // GFCCR, Set IBO 8 Devices, RFL/RFS/BLK, BPCLK-16.384

    /* Enable Bulk Write to reduce Framer, LIU, and Bert initialization time. */
    write(0x00F0, 0x04); // GTCR1, Enable Global Bulk Write

    /* Perform global LIU, Framer, and BERT software reset */
    write(0x00F5, 0xFF); // GLSRR
    write(0x00F6, 0xFF); // GFSRR
    /* Wait 10 ms for reset to complete */
    wait(10);
    write(0x00F5, 0x00); // GLSRR
    write(0x00F6, 0x00); // GFSRR

    /* Perform individual Receive and Transmit Framer software reset. This */
    /* is not needed when using the Global Reset functions above. */
    // write(0x0080, 0x02); // RMMR
    // write(0x0180, 0x02); // TMMR
    /* wait 10 ms for reset to complete */
    // wait (10);
    // write(0x0080, 0x00); // RMMR
```

```

// write(0x0180, 0x00); // TMMR

/* Zero All Framer Registers */
index = 0x0000;
while(index < 0x00F0)
{
    write(index, 0x00);
    index++;
}

index = 0x0100;
while(index < 0x01F0)
{
    write(index, 0x00);
    index++;
}

/* Zero All LIU Registers */
index = 0x1000
while(index < 0x1020)
{
    write(index, 0x00);
    index++;
}

/* Zero All BERT Registers */
index = 0x1100;
while(index < 0x1110)
{
    write(index, 0x00);
    index++;
}

/* Disable Bulk Write at end of initialization */
write(0x00F0, 0x00); // GTCR1, Disable Global Bulk Write

/* Initialization Complete */

/* Configuration Begin */

e1_configure();

// t1_configure();

/* Configuration Complete */
}

void e1_configure()
{
    /* E1 Initialization Begin */

    /* This function assumes all ports are configured identically. Otherwise, */
    /* remove the Global Bulk Write portion and rewrite the function to */
    /* support individual port addressing and identification. */

    /* Enable Bulk Write to reduce Framer and LIU configuration time. */
    write(0x00F0, 0x04); // GTCR1, Enable Global Bulk Write

    write(0x0080, 0x01); // RMMR, Set Receive Framer E1 Mode
    write(0x0180, 0x01); // TMMR, Set Transmit Framer E1 Mode

```

```

write(0x0080, 0x81); // RMMR, Set Receive Framer E1 Mode and Framer Enable
write(0x0180, 0x81); // TMMR, Set Transmit Framer E1 Mode and Framer Enable

write(0x0081, 0x60); // RCR1, Set Receive E1 HDB3 (Basic Frame)
// write(0x0081, 0x68); // RCR1, Set Receive E1 HDB3, CRC4, and CCS (TS 16 Clear)
// write(0x0081, 0x48); // RCR1, Set Receive E1 HDB3, CRC4, and CAS

write(0x0084, 0x10); // RIOCR, Set RSYNC Frame Output, RSYCLK-2.048 MHz
// write(0x0084, 0x11); // RIOCR, Set RSYNC CAS M-Frame Output, RSYCLK-2.048 MHz
// write(0x0084, 0x13); // RIOCR, Set RSYNC CRC4 M-Frame Output, RSYCLK-2.048 MHz

/* Note: In CCS mode, CAS can still be enabled through the SSIEx registers */
/* Note: All TCR1 E1 configurations have Si bit pass through enabled */
write(0x0181, 0x04); // TCR1, Set Transmit E1 HDB3 (Basic Frame)
// write(0x0181, 0x05); // TCR1, Set Transmit E1 HDB3, CRC4, and CCS (TS 16 Clear)
// write(0x0181, 0x45); // TCR1, Set Transmit E1 HDB3, CRC4, and CAS

write(0x0184, 0x10); // TIOCR, Set TSYNC Frame Input, TSYCLK-2.048 MHz
// write(0x0184, 0x11); // TIOCR, Set TSYNC M-Frame Input, TSYCLK-2.048 MHz
// write(0x0184, 0x14); // TIOCR, Set TSYNC Frame Output, TSYCLK-2.048 MHz
// write(0x0184, 0x15); // TIOCR, Set TSYNC M-Frame Output, TSYCLK-2.048 MHz

/* Set the TAF and TNAF registers in E1 Mode Only */
write(0x164, 0x1B);
write(0x165, 0x40);

/* Set the Transmit Signaling registers for E1 Mode */
write(0x0140, 0x0B); // Set CAS Multiframe Sync Header
index = 0x0141;
while(index < 0x0150)
{
    write(index, 0xDD);
    index++;
}

/* Set the Transmit Signaling Insertion Enable registers for E1 Mode */
// write(0x0118, 0xFF); // SSIE1, Enable Transmit Signaling for channels 1-8
// write(0x0119, 0xFF); // SSIE2, Enable Transmit Signaling for channels 9-16
// write(0x011A, 0xFF); // SSIE3, Enable Transmit Signaling for channels 17-24
// write(0x011B, 0xFF); // SSIE4, Enable Transmit Signaling for channels 18-32

/* Set Framer or Payload Loopback if necessary */
// write(0x0083, 0x01); // RCR3, Set Framer Loopback
// write(0x0083, 0x02); // RCR3, Set Payload Loopback

/* Complete all Framer register configuration before the next step */
write(0x0080, 0xC1); // RMMR, Set Receive Framer E1 Mode, Enable, and Init Done
write(0x0180, 0xC1); // TMMR, Set Transmit Framer E1 Mode, Enable, and Init Done

/* Configure LIU */
write(0x1000, 0x00); // LTRCR, E1 Mode, ITU G.775, and Receive JA @ 128 bits
// write(0x1000, 0x00); // LTRCR, E1 Mode, ETSI 300-233, and Receive JA @ 128 bits

write(0x1001, 0x00); // LTITSR, E1 Mode 75 ohm w/Transmit Z-Term
// write(0x1001, 0x31); // LTITSR, E1 Mode 120 ohm w/Transmit Z-Term

write(0x1007, 0x03); // LRISMR, E1 Mode 75 ohm Long Haul w/Receive Z-Match
// write(0x1007, 0x33); // LRISMR, E1 Mode 120 ohm Long Haul w/Receive Z-Match
// write(0x1007, 0x00); // LRISMR, E1 Mode 75 ohm Short Haul w/Receive Z-Match
// write(0x1007, 0x30); // LRISMR, E1 Mode 120 ohm Short Haul w/Receive Z-Match

```

```

write(0x1002, 0x01); // LMCR, Enable Transmit Outputs
// write(0x1002, 0x09); // LMCR, Enable Transmit Outputs and Remote Loopback
// write(0x1002, 0x11); // LMCR, Enable Transmit Outputs and Analog Loopback
// write(0x1002, 0x21); // LMCR, Enable Transmit Outputs and Local Loopback

/* Disable Bulk Write at end of configuration */
write(0x00F0, 0x00); // GTCR1, Disable Global Bulk Write
}

void tl_configure()
{
    /* Tl Initilization Begin */

    /* This function assumes all ports are configured identically. Otherwise, */
    /* remove the Global Bulk Write portion and rewrite the function to */
    /* support individual port addressing and identification. */

    /* Enable Bulk Write to reduce Framer and LIU configuration time. */
    write(0x00F0, 0x04); // GTCR1, Enable Global Bulk Write

    write(0x0080, 0x00); // RMMR, Set Receive Framer Tl Mode
    write(0x0180, 0x00); // TMMR, Set Transmit Framer Tl Mode
    write(0x0080, 0x80); // RMMR, Set Receive Framer Tl Mode and Framer Enable
    write(0x0180, 0x80); // TMMR, Set Transmit Framer Tl Mode and Framer Enable

    write(0x0081, 0xC8); // RCR1, Set Receive Tl B8ZS, ESF, CRC6 Verify
    // write(0x0081, 0xE8); // RCR1, Set Receive Tl B8ZS, D4, Ft & Fs Verify
    // write(0x0081, 0xCC); // RCR1, Set Receive J1 B8ZS, ESF, J-CRC6 Verify
    // write(0x0081, 0x68); // RCR1, Set Receive Tl B8ZS, SLC-96 (Also T1RCR2)

    // write(0x0014, 0x10); // T1RCR2, Set Receive Tl, SLC-96

    write(0x0084, 0x10); // RIOCR, Set RSYNC Frame Output, RSYSClk-2.048 MHz
    // write(0x0084, 0x11); // RIOCR, Set RSYNC M-Frame Output, RSYSClk-2.048 MHz

    /* Note: All TCR1 Tl/J1 configurations have SSIEx signaling control enabled */
    write(0x0181, 0x14); // TCR1, Set Transmit Tl B8ZS
    // write(0x0181, 0x94); // TCR1, Set Transmit J1 B8ZS, J-CRC6

    write(0x0183, 0x00); // TCR3, Set Transmit ESF
    // write(0x0183, 0x04); // TCR3, Set Transmit D4 or SLC-96 (Also TCR2)

    // write(0x0182, 0x40); // TCR2, Set Transmit SLC-96

    write(0x0184, 0x10); // TIOCR, Set TSYNC Frame Input, TSYSClk-2.048 MHz
    // write(0x0184, 0x11); // TIOCR, Set TSYNC M-Frame Input, TSYSClk-2.048 MHz
    // write(0x0184, 0x14); // TIOCR, Set TSYNC Frame Output, TSYSClk-2.048 MHz
    // write(0x0184, 0x15); // TIOCR, Set TSYNC M-Frame Output, TSYSClk-2.048 MHz

    /* Set the T1TFDL and T1TSLCx registers in Tl Mode Only */
    write(0x162, 0x1C); // T1TFDL, Source D4 Mode Fs Bits
    write(0x164, 0x00); // T1TSLC1, Source SLC-96 Mode C8-C1 Bits
    write(0x165, 0x10); // T1TSLC2, Source SLC-96 Mode M2-M1, Sp, and C11-C9 Bits
    write(0x166, 0x80); // T1TSLC3, Source SLC-96 Mode Sp, S4-S1, A2-A1, and M3 Bits

    /* Set the Transmit Signaling registers for Tl Mode */
    index = 0x0140;
    while(index < 0x014C)
    {
        write(index, 0xDD);
    }
}

```

```

    index++;
}

/* Set the Transmit Signaling Insertion Enable registers for T1 Mode */
// write(0x0118, 0xFF); // SSIE1, Enable Transmit Signaling for channels 1-8
// write(0x0119, 0xFF); // SSIE2, Enable Transmit Signaling for channels 9-16
// write(0x011A, 0xFF); // SSIE3, Enable Transmit Signaling for channels 17-24

/* Configure Framer or Payload Loopback */
// write(0x0083, 0x01); // RCR3, Set Framer Loopback
// write(0x0083, 0x02); // RCR3, Set Payload Loopback

/* Complete all Framer register configuration before the next step */
write(0x0080, 0xC0); // RMMR, Set Receive Framer T1 Mode, Enable, and Init Done
write(0x0180, 0xC0); // TMMR, Set Transmit Framer T1 Mode, Enable, and Init Done

/* Configure LIU */
write(0x1000, 0x02); // LTRCR, T1/J1 Mode, ANSI T1.231, and Receive JA @ 128 bits

write(0x1001, 0x10); // LTITSR, T1 Mode 100 ohm 0dB CSU w/Transmit Z-Term
// write(0x1001, 0x20); // LTITSR, J1 Mode 110 ohm 0dB CSU w/Transmit Z-Term

write(0x1007, 0x13); // LRISMR, T1 Mode 100 ohm Long Haul w/Receive Z-Match
// write(0x1007, 0x23); // LRISMR, J1 Mode 110 ohm Long Haul w/Receive Z-Match
// write(0x1007, 0x10); // LRISMR, T1 Mode 100 ohm Short Haul w/Receive Z-Match
// write(0x1007, 0x20); // LRISMR, J1 Mode 110 ohm Short Haul w/Receive Z-Match

write(0x1002, 0x01); // LMCR, Enable Transmit Outputs
// write(0x1002, 0x09); // LMCR, Enable Transmit Outputs and Remote Loopback
// write(0x1002, 0x11); // LMCR, Enable Transmit Outputs and Analog Loopback
// write(0x1002, 0x21); // LMCR, Enable Transmit Outputs and Local Loopback

/* Disable Bulk Write at end of configuration */
write(0x00F0, 0x00); // GTCR1, Disable Global Bulk Write
}

```

Figure 1. Code to initialize and configure the DS26528 transceiver.

References

If you have additional questions on LIU initialization and configuration, please contact the Telecommunication Applications support team by email, telecom.support@dalsemi.com, or telephone at 01-972-371-6555.

For more information about the DS26528 Octal T1/E1/J1 Transceiver please consult the appropriate data sheet which is available on the Dallas Semiconductor/Maxim website at [T/E Carrier and Packetized Products](#).

Application Note 3827: www.maxim-ic.com/an3827

More Information

For technical support: www.maxim-ic.com/support

For samples: www.maxim-ic.com/samples

Other questions and comments: www.maxim-ic.com/contact

Automatic Updates

Would you like to be automatically notified when new application notes are published in your areas of interest?

[Sign up for EE-Mail™.](#)

Related Parts

DS26524: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS26528: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

AN3827, AN 3827, APP3827, Appnote3827, Appnote 3827

Copyright © by Maxim Integrated Products

Additional legal notices: www.maxim-ic.com/legal