Application Report
SLLA105 – October 2001

DSP to DSP Link Using LVDS

Marcus Gößler, Frank Dehmelt *High Performance Linear*

ABSTRACT

In digital signal processor (DSP) applications, it is important to transfer data to and from DSPs, as well as between them, at high speed and with high reliability.

On single-board designs, the on-chip integrated inputs/outputs (I/O) of the DSP (e.g., the multichannel buffered serial port (McBSP), can be used for short-distance runs. For board-to-board-links or even between racks/cabinets, another transmission standard has to be applied. A suitable solution is a low voltage differential signaling (LVDS) link between DSPs.

The following application report describes such a connection. The McBSPs of two DSP starter kits (DSK) are connected to the SN65LVDS31/32A evaluation modules (EVM). The LVDS data is transmitted over 30 meters of twisted pair cable at signaling rates up to 75 megabits per second (Mbps).

Three different modes of synchronizing the transmitting and receiving DSP are shown: polling, event-triggered DMA and interrupt controlled. The advantages and disadvantages of each, resulting in speed limitations, are presented herein.

Contents

Figures

Tables

1 Introduction

1.1 Design Problem

Digital signal processors are well suited for a wide range of applications such as video or audio processing, digital motor control, telecommunication, and image/voice processing.

The primary strength of a DSP is its ability to process external data. Once data has been loaded into internal memory, the full computing power of the DSP can be used. Whether this data comes from an analog-to-digital converter (ADC), first-in/first-out buffer (FIFO), or another DSP in a distributed architecture (Figures 1 and 2), the DSP needs to keep step with the data stream.

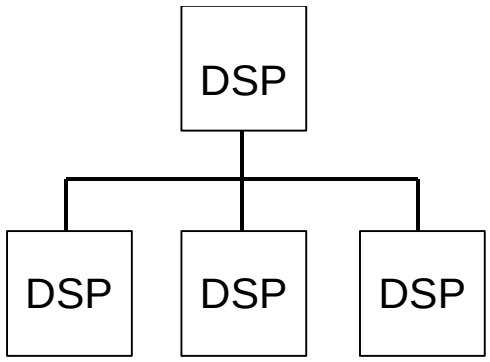

Figure 1. Distributed Architecture

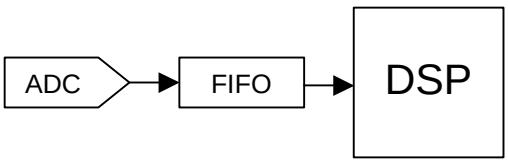

Figure 2. Connecting ADC/FIFO to a DSP

Another consideration is the origin of data. Sometimes it is necessary to transmit data over longer distances, and thus, only an appropriate transmission standard can ensure the fast and reliable data transfer (e.g., from a remote ADC to the DSP).

1.2 Configuration

One or more of the multichannel buffered serial ports (McBSPs) of the TMS320C6000™ DSP family can be used to establish a reliable high-speed connection.

In addition, a suitable solution to bridging longer distances is to use a low voltage differential signaling (LVDS) link as shown in Figure 3.

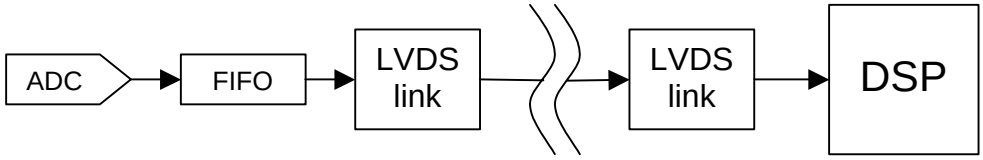

Figure 3. LVDS Link Used for Longer Distances

A serial connection between McBSPs can be implemented using LVDS as the physical layer between DSPs. This provides a cost-effective solution as well as a high-speed, low-power signal path between devices.

Some applications require multiple sensors (e.g, ADCs sourcing data to a single DSP). Also, there are situations that require bidirectional communications between several DSPs. Figures 4 and 5 illustrate these configurations.

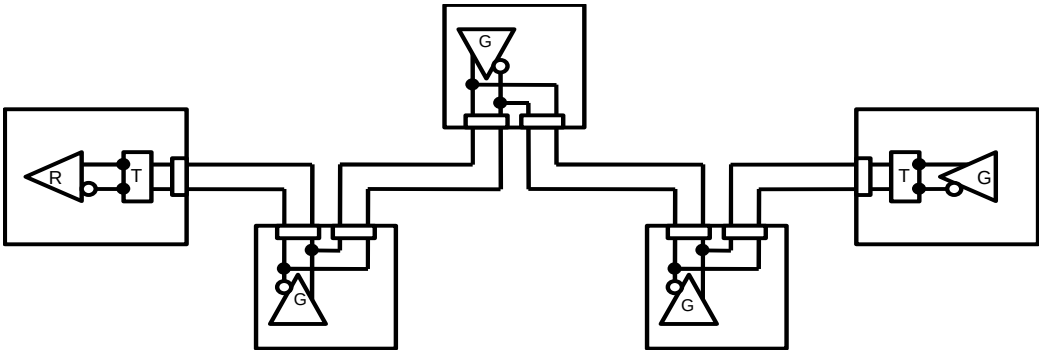

Figure 4. Multipoint Configuration With Many Drivers and One Receiver

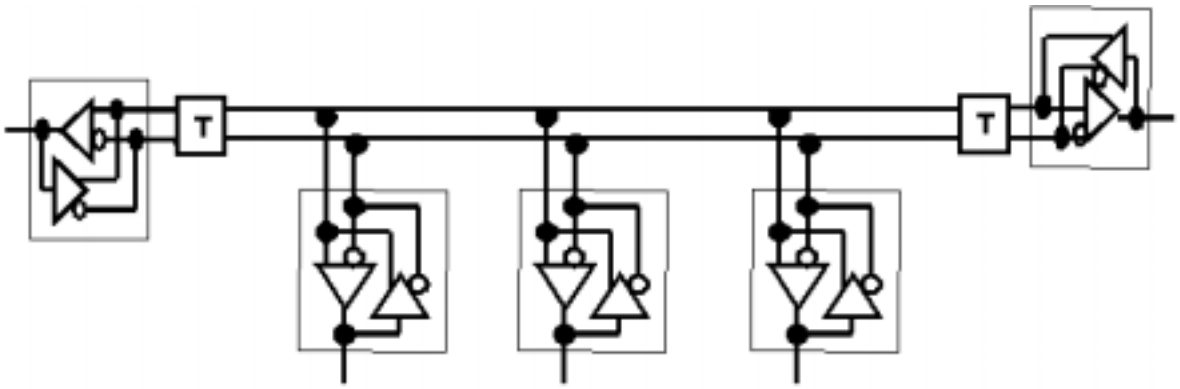

Figure 5. Multipoint Configuration With Several Transceivers

A new standard (TIA/EIA-899), specifically targeting multipoint configurations (as in Figure 4 and Figure 5) is currently in development. TI has already developed line drivers providing increased drive strength to support these types of applications. These drivers provide twice the output current of standard LVDS drivers. This maintains the standard LVDS voltage levels on the signal lines when the transmission line is terminated at both ends, which is required in the case when two or more drivers are located on the transmission line. TI uses the term *LVDM* to distinguish between standard LVDS and multipoint LVDS devices. The term *LVDM* also appears in the part number (i.e., SN65LVD*M*1676) to identify devices that provide this higher output current. Application information on this configuration is available online at the TI web site. Refer to application report number SLLA088 - *Transmission at 200Mbps in VME Card Cage using LVDS*.

1.3 Design Example

Figure 6 shows two TMS320C6211™ DSP devices, each located on a separate DSP starter kit (DSK), connected via McBSP1. Signals from one processor, which acts as a master, are linked to one SN65LVDS31/32A EVM, where they are translated to LVDS through the LVDS driver located on the first 31/32A EVM. These signals are transferred through several meters of twisted pair cable to a second 31/32A EVM, where the LVDS receivers are used to convert the signals back to single-ended LVTTL signals. They are then routed to the McBSP1 port on the second processor, which acts as a slave.

[Note: The LVDS EVM contains both drivers and receivers; therefore, a single LVDS EVM could have been used, and the cable simply *looped* from the driver output to the receiver input on a single EVM. We chose to use two separate EVMs.]

The same configuration can be used along with the pin-to-pin and code-compatible TMS320C6711™ DSP.

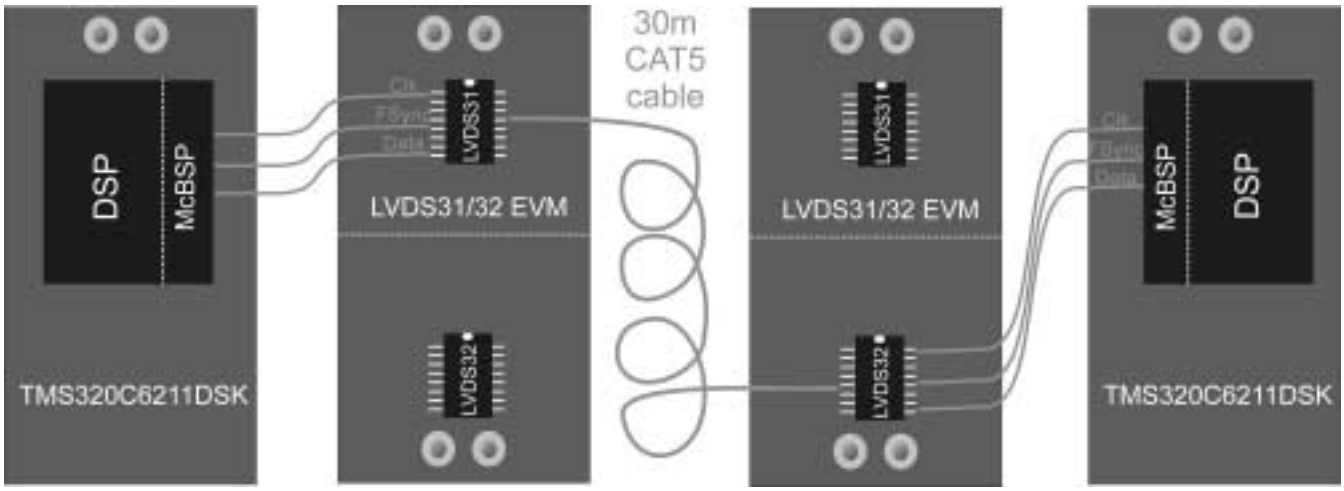

Figure 6. DSP to DSP Link Using LVDS

2 Basics of Differential Signaling and Serial Ports

2.1 LVDS (Low Voltage Differential Signaling)

The LVDS approach achieves higher data rates using low power on commonly used copper media. Limitation of previous commonly used differential interface standards, such as TIA/EIA-422 or 485 are related to; 1) the maximum achievable slew rate, 2) EMI restrictions, and 3) power consumption. LVDS uses low voltage differential signaling, where the receiver thresholds are reached much faster, even while the slew rate remains the same. There is also a dramatic reduction in power and radiated emissions due to the lower voltages. (See application report SLLA030 – *Reducing EMI Using LVDS*). This makes LVDS an extremely attractive alternative to conventional electrical layers used in the past, when the transmission distance is moderate (<100 m).

The TIA/EIA-644-A standard specifies a theoretical maximum signaling rate of 1.932 Gbps (gigabits per second). The maximum distance of the interconnect depends on the environment (noise, ground shifts, etc.) and the transmission media (quality and type of cable). High-quality cables in an environment with low ground shifts (i.e., remaining within the common mode range of the devices) can support cable lengths up to 100 meters at 70 Mbps. See application report SLLA053 – *Performance of LVDS with Different Cables* for more details on cables and related performance. For the measurements in this report, a cable length of 30 meters was chosen.

As its name indicates, LVDS uses differential transmission mode and low signal amplitudes. The swing is in the range of only 350 mV, generated across a 100-Ω termination resistor. TI LVDS drivers are actually current mode drivers, forcing an output current between 2.47 mA and 4.54 mA into either one of the two outputs. Since the driver output is differential, the driver and receiver are always drawing the same supply current (I_{CC}) regardless of the data (logic 1 or 0) being transmitted, and I_{CC} vs signaling rate is almost constant over the entire range of signaling rates. This eliminates most of the ΔI_{CC} feedback to the supply.

2.2 McBSP (Multichannel Buffered Serial Port)

2.2.1 Functional Description

The McBSP consists of two major functional blocks, separated into smaller functional units as shown in Figure 7.

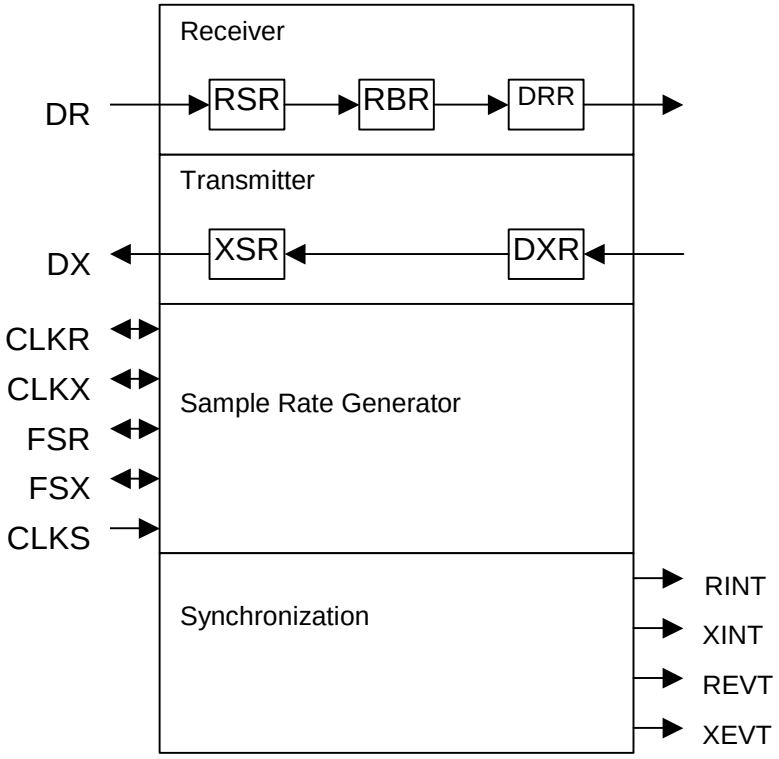

Figure 7. McBSP Functional Block Diagram

- **Data Path**

  Either the CPU or the EDMA controller can be used to control the data flow to and from the McBSP. Data flow can be either a *write to* the data transmit register (DXR) or a *read from* the data receive register (DRR).

  - **Reception**

    The receive frame synchronization indicates a new incoming data frame. Data on the DR pin is then shifted into the receive shift register (RSR) and once a whole element is transferred, its content is copied to the receive buffer register (RBR). Finally, the RBR is copied to DRR.

  - **Transmission**

    If the transmitter is ready to transfer data, DXR is copied to the transmit shift register (XSR). After the frame synchronization occurs, the bit values in XSR are shifted out on the DX pin.

- **Control Path**

  Several registers configure the mechanisms of the McBSP, namely the internal clock generation, multichannel selection, frame synchronization, and notification of interrupts to the CPU and events to the EDMA controller.

  - **Sample Rate Generator**

    Two internal signals are generated; they can be programmed to drive the framing (FSR/X) and clocking (CLKR/X).

    The internal clock source can be either an external clock or one half of the CPU clock. This source, divided down to the desired transmit/receive frequency, is used to generate the frame synchronization signal of proper frame period and width.

  - **Synchronization Events and Interrupts**

    Once data is copied from RBR to DRR the receiver ready bit (RRDY) is set and can be used to generate a receive interrupt (RINT) or a receive event (REVT). The EDMA can use this event as a read synchronization event to start an EDMA transfer.

    Once data is copied from DXR to XSR, the transmitter ready bit (XRDY) is set and can be used to generate a transmit interrupt (XINT) or a transmit event (XEVT). The EDMA can use this event as a write synchronization event to start an EDMA transfer.

2.2.2 Synchronization

The bits RRDY and XRDY are indicators for the ready status of the receive and transmit portion of the McBSP. The reaction of all three possible synchronization methods is based on the value of these bits:

- **Polling**

  RRDY is set to 1 when RBR is copied to DRR. Therefore, the status of this bit can be polled to identify the need to read received data out of DRR. Reading DRR clears RRDY.

  XRDY is set to 1 when DXR is copied to XSR. Again, the status of this bit can be polled to recognize when there is the need to write data to be transmitted into DXR. Writing to DXR clears XRDY.

- **Interrupts**

  RINT can be set up to be driven by RRDY and XINT by XRDY. The DSP interrupt-service-routine then can read DRR and write to DXR, respectively.

- **EDMA**

  For each McBSP, two EDMA channels exist with the associated events REVT and XEVT, respectively. EDMA transfers can be configured to be triggered by these events, thus transferring one element per event.

  See Reference 1 for a detailed description of EDMA functionality.

2.2.3 Initialization

In order to assure proper operation of the McBSP, especially along with EDMA or interrupts, it is necessary to initialize a set of control registers (Figure 8 through 12). This can be accomplished easily by making use of the chip support library (CSL) which is shipped with code composer studio 1.2 (CCS) and newer releases. It is also important to follow a certain sequence in setting up those registers so that no data is lost. One requirement is that the slave (receiving data) has to be brought up before the master (transmitting data).

Please see Reference 2.

31 – 24	23	22	21	20	19	18	17	16
reserved	/FRST	/GRST	XINTM		XSYNCERR	/XEMPTY	XRDY	/XRST

15	14	13	12	11	10	8	7	6	5	4	3	2	1	0
DLB	RJUST	CLKSTP		Rsvd			DXENA	Rsvd	RINTM		RSYNCERR	RFULL	RRDY	/RRST

Figure 8. Serial Port Control Register (SPCR)

31	30	24	23	21	20	19	18	17	16
RPHASE	RFRLEN2		RWDLEN2		RCOMPAND		RFIG	RDATDLY	

15	14	8	7	5	4	3	0
RPHASE 2	RFRLEN1		RWDLEN1		RWDREVRS	Reserved	

Figure 9. Receive Control Register (RCR)

31	30	24	23	21	20	19	18	17	16
XPHASE	XFRLEN2		XWDLEN2		XCOMPAND		XFIG	XDATDLY	

15	14	8	7	5	4	3	0
XPHASE 2	XFRLEN1		XWDLEN1		XWDREVRS	Reserved	

Figure 10. Transmit Control Register (XCR)

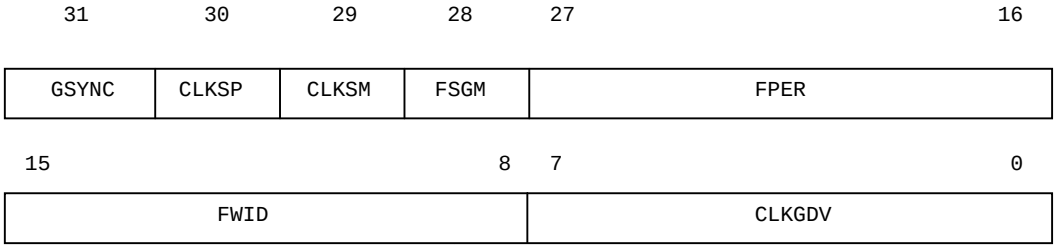

Figure 11. Sample Rate Generator Register (SRGR)

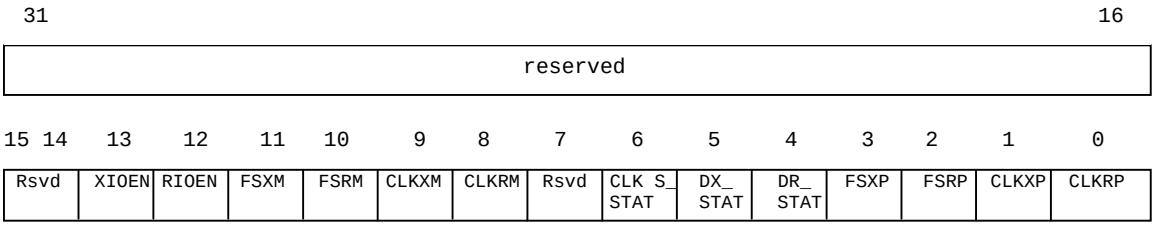

Figure 12. Pin Control Register (PCR)

The following procedure shows in general how the McBSP has to be set up to work properly together with the EDMA and/or interrupts.

- **Polling**

1. Put the receiver, transmitter, sample rate generator and frame synch generator into reset state.
2. Program SRGR, SPCR, PCR and RCR to the desired values while keeping units described in step 1 in reset.
3. Activate the sample rate generator by taking it out of reset.
4. Activate the frame slave.
5. Activate the frame master.
6. Start frame synch generation.

- **Interrupts**

1. Put the receiver, transmitter, sample rate generator and frame synch generator into reset state.
2. Program SRGR, SPCR, PCR and RCR to the desired values while keeping units described in step 1 in reset.

3. Activate the sample rate generator by taking it out of reset.
4. Map the interrupt sources XINT and/or RINT to particular interrupts.
5. Enable the mapped interrupts.
6. Activate the frame slave.
7. Activate the frame master.
8. Start frame synch generation.

- **EDMA (Enhanced Direct Memory Access)**

1. Put the receiver, transmitter, sample rate generator and frame synch generator into reset state.
2. Program SRGR, SPCR, PCR and RCR to the desired values while keeping units described in step 1 in reset.
3. Activate the sample rate generator by taking it out of reset.
4. Disable the EDMA channels by disabling the associated events.
5. Initialize the associated parameter RAM and reload parameters.
6. Enable EDMA channels by enabling the associated events.
7. Activate the frame slave.
8. Activate the frame master.
9. Start frame synch generation.

3 Implementation

3.1 Hardware

The TI SN65LVDS31/32A EVM evaluation module (EVM) is designed for the analysis of low-voltage differential signaling with a SN65LVDS31 quad driver and a SN65LVDS32A quad receiver. It can be ordered with any of several different quad drivers and receivers installed on the EVM.

For this particular application report, an EVM equipped with the SN65LVDS32A has been used. The interconnect media was a 30-meter category 5 (CAT 5) cable. See application report SLLA053 – *Performance of LVDS with Different Cables*. Three out of four pairs have been used to transmit the clock, the frame synchronization, and the data (Figure 6). For the receiver, either the receiver on the same EVM or a second board can be used.

Adapters were set up to interface the TTL driver inputs and receiver outputs to the McBSPs, respectively, to the daughterboard connectors of the DSKs.

A detailed description of the EVM can be found in the EVM manual (TI Literature No. SLLU016).

This configuration of two DSKs linked via two LVDS EVMs can be used in a wide range of applications where data has to be transferred from one DSP to another over a distance of up to 30 meters (e.g., video-phone, surveillance).

3.2 Software

One C6211™ DSP DSK is set up as master and provides data, clock, and frame signals for the second DSK, which is set up as slave. 32-bit words are transmitted in a single-phase frame. Signals are generated by using the internal clock source.

As already mentioned in Section 2.2.3, the TMS320C6000 DSP chip support library (CSL), which is part of CCS 1.2, can be used to configure and control the on-chip peripherals. This library is a set of application programming interfaces (API) intended to shorten the development cycle time. Changes in the configuration can be handled easily because of hardware abstraction.

The library contains several discrete modules, each of which covers a specific peripheral. Of interest for this application are the EDMA, IRQ, and MCBSP modules.

Some methods and techniques are common to most of the modules and show the general usage of the CSL.

- OPEN and CLOSE Functions

  A so-called *handle* concept exists for each peripheral that consists of several channels or ports. For example, in order to use a certain McBSP port it is necessary to open this resource by calling the appropriate *open* function. In the same way, this resource has to be freed up again by calling the proper *close* function.

- MK Macros

  These macros where MK stands for *make* produce register values out of field arguments or predefined value constants. Think of a register called REG containing four bit fields f0 to f3 for a peripheral called PER. The appropriate macro would be:

```
PER_MK_REG(f0, f1, f2, f3)
```

  Bit fields f0 to f3 could contain the desired bit pattern or some predefined constants looking like these:

```
PER_REG_F0_VAL0
PER_REG_F1_VAL0
PER_REG_F1_VAL1
…
PER_REG_F3_VAL4
```

- Register Configuration

  In order to configure the peripheral with MK macros there are two functions. If the peripheral is called PER, the corresponding function names are *PER_ConfigA()* and *PER_ConfigB().* Together with a structure called *PER_CONFIG*, it is possible to choose from two methods to initialize registers of a peripheral.

  Method A initializes the configuration structure with the desired register values. The address of the structure is passed to *PER_ConfigA()* in order to initialize the registers.

```
PER_CONFIG YourConfig = {
        Reg0_val,
        Reg1_val,
        …
};
PER_ConfigA(&YourConfig);
```

Method B does not use the configuration structure. Instead the register values are passed directly to *PER_ConfigB().*

```
PER_ConfigB(Reg0_val, Reg1_val, …);
```

Reference 3 provides a detailed description of CSL functionality.

Please refer to Appendix A.1 for an example on how to initialize the McBSP using CSL.

3.2.1 Register Settings

When using two DSPs in a master-slave configuration (as previously described) it is important to set the appropriate register values accordingly. Therefore, some bit fields are of special interest, while others are *don't cares* or are disabled. The important settings for different synchronization modes are listed in the next three subsections.

3.2.1.1 Polling

- SPCR

  Receiver, transmitter, sample rate generator, and frame synch generator have to be put in reset. Digital loop back is off, and no clock stop mode is used.

- RCR

  Receive 32-bit reversal feature is disabled. Receive element length in phase 1 is set to 32 bits. Receive frame length in phase 1 is set to 1 word (32 bit). Unexpected receive frame synch signals are ignored. The transfer starts with MSB first. Only single-phase transfers are performed.

- XCR

  Transmit 32-bit reversal feature is disabled. Transmit element length in phase 1 is set to 32 bit. Transmit frame length in phase 1 is set to 1 word. Unexpected transmit frame synch signals are ignored. The transfer starts with MSB first. Only single-phase transfers are performed.

- SRGR

  Set clock divide to a value to obtain the desired transfer rate according to $rate = (clock/2)/(clkdiv+1)$. Width of the frame synch signal is set to 1, and frame period is set to 33. FSX is driven by sample rate generator's frame synch generator (FSG). Sample rate generator clock, derived from internal clock source, is free running.

- PCR

  Received data is sampled on falling edge of CLKR. Transmitted data is driven on rising edge of CLKX. FSX/R are active high. CLKR is an input pin. CLKX is an output pin. Receive

frame synchronization is driven externally. Transmit frame synchronization is driven internally. Both the receiving and transmitting parts of the McBSP are not general-purpose I/O pins.

3.2.1.2 Interrupt

Please refer to appendix A.2 for an example on how to synchronize the McBSP with interrupts.

In addition to the described settings in Section 3.2.1, the following register configuration applies when using interrupts to service the McBSP:

- SPCR

  Receiver interrupt is driven by RRDY. Transmitter interrupt is driven by XRDY.

3.2.1.3 EDMA

Please refer to appendix A.3 for an example on how to initialize the EDMA to be used for synchronization.

In addition to the described settings in Section 3.2.1, the following important parameter configurations have to be applied along with EDMA:

- Options Parameter

  Frame synch is not needed to start a frame transfer that corresponds to read/write synchronization. No source address modification for receiving. No destination address modification for transmitting. Element size is 32 bits.

- Source

  Address of DRR is source for receiving.

- Destination

  Address of DXR is destination for transmitting.

4 Technical Details

4.1 Timings

When using the McBSP to transmit and receive data in a master-slave scheme, several delays and constraints have to be considered. These are listed in Table 1.

The maximum data rate that can be achieved in this way is 33 Mbps with two C6211/C6711 DSPs corresponding to a clock period of 30 ns.

FSR must have a hold time of 3 ns (t_h(CKRL – FRH)) while FSX has a delay time of –11ns minimum (t_d(CKXH - FXV)). Taking into account 1 ns variation of the high/low time of CLKX, the maximum shift frequency results in:

$$f_{\max}[MHz] = 1000/[2*(3ns + 11ns + 1ns)]$$

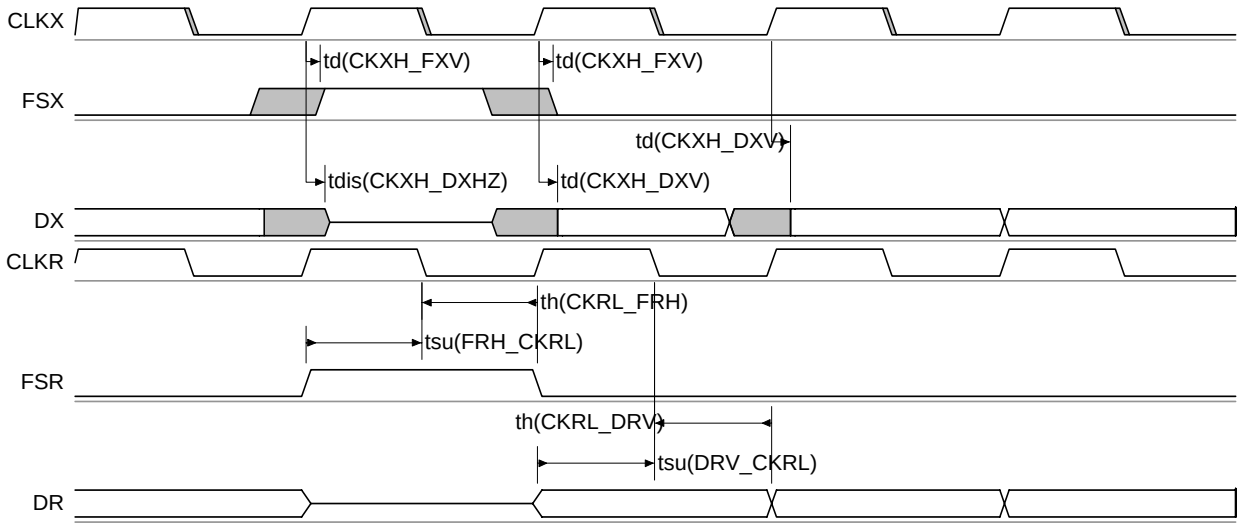

Figure 13. Timing Diagram for Maximum Data Rate

Table 1. McBSP Timing Parameters

Transmit			
Parameter	**MIN**	**MAX**	**Description**
t_w(CKRW)	C – 1	C + 1	Pulse duration, CLKX high or CLKX low
t_d(CKXH - FXV)	-11	3	Delay, CLKX high to internal FSX valid
t_{dis}(CKXH - DXHZ)	-9	4	Disable, DX high impedance following last data bit from CLKX high
t_d(CKXH - DXV)	-9	4	Delay, CLKX high to DX valid
Receive			
t_{su}(FRH - CKRL)	1		Setup, external FSR high before CLKR low
t_h(CKRL – FRH)	3		Hold, external FSR high after CLKR low
t_{su}(DRV – CKRL)	3		Setup, DR valid before CLKR low
t_h(CKRL - DRV)	4		Hold, DR valid after CLKR low

For a detailed description of timing parameters along with McBSPs, refer to the data sheet (Reference 4). Note that the 33 MHz maximum shift frequency only applies to the master-slave connection of two C6211/C6711™ DSPs and that different system configurations and/or other C6000 DSP devices achieve higher transmission rates according to their specification.

4.2 Speed

Depending on the type of solution (Polling, IRQ, EDMA) for transferring data from one C6211 DSP to another DSP, different maximum transfer rates can be achieved and are listed in Table 2.

Table 2. Maximum Transfer Rates Versus Synchronization Methods

Polling			
CLKDIV	**Transfer Rate [Mbps]**	**Speed Optimization**	**2-way Cache**
0x1	37.5 (out of spec)	✓	✓
Interrupt			
0x6	10.7	✓	✓
Enhanced DMA			
0x0	75 (out of spec)	✗	✓

Every solution has its pros and cons, mainly related to speed, programming effort, and code space. Depending on the application, the proper transmission method can be chosen to satisfy the system needs.

Polling, which is probably the easiest way to set up a transfer burdens the DSP by checking the ready bits, which is in fact a busy-wait. Taking advantage of the compiler's optimization option, a transfer rate of 37.5 Mbps could be achieved, which exceeds the maximum specified rate of 33 Mbps, but shows that this method can be used at high rates.

The interrupt driven solution provides the opportunity to perform other calculations while the McBSP is running. The occurrence of an interrupt indicates that the McBSP is ready to be served. Within the interrupt service routine (ISR), the appropriate actions to read from or write to the McBSP, respectively, can be taken. Unlike the previous method of polling, overhead is added to the program flow by entering or leaving the ISR and performing the necessary context switch. While this leads to slower transfer rates, it allows the DSP to work more effectively.

Finally, the highest transfer rates can be achieved by direct memory access (DMA) transfers. In this way, the DSP can work completely independent from the McBSP and EDMA controller (once set up correctly), thus achieving transfer rates up to 75 Mbps.

Conclusion

The McBSP can be used to establish a reliable high-speed data connection between two or more DSPs. To bridge distances of several meters, an LVDS interface provides a high speed, low power, low EMI interface.

The configuration of the serial port allows the synchronization of data transfers via polling, interrupt, or EDMA. LVDS and its differential transmission standard connect the devices via twisted pair cables over several meters.

A data transmission rate of up to 75 Mbps (faster than that specified in the DSP data sheet) over 30 meters was demonstrated using Texas Instruments LVDS devices. This ensures a reliable data transfer at 33 Mbps (the maximum speed recommended in the DSP data sheet) with plenty of headroom for future growth

References

1. *TMS320C6000 Enhanced DMA: Example Applications* (SPRA636)
2. *TMS320C6000 McBSP Initialization* (SPRA488)
3. *TMS320C6000 Chip Support Library API Reference Guide* (SPRU401)
4. *TMS320C6211 data sheet* (SPRS073)
5. *Transmission at 200Mbps in VME Card Cage using LVDS* (SLLA088)
6. *Performance of LVDS With Different Cables* (SLLA053)
7. *Low Voltage Differential Signaling LVDS EVM User's Guide* (SLLU016).
8. *SN65LVDS31 data sheet* (SLLS261)
9. *SN65LVDS32B data sheet* (SLLS440)
10. Analog Design Seminar

Appendix A Example Code

A.1 Initialization of McBSP

```
/*****************************************************************************
* FUNCTION : mcbsp1_init
* This function initializes McBSP1 for polling, interrupt and EDMA usage.
*****************************************************************************/
void mcbsp1_init()
{
  MCBSP_CONFIG mcbsp1_config = {
    MCBSP_MK_SPCR(
      MCBSP_SPCR_RRST_YES,
      MCBSP_SPCR_RINTM_RRDY,
      MCBSP_SPCR_DXENA_OFF,
      MCBSP_SPCR_CLKSTP_DISABLE,
      MCBSP_SPCR_RJUST_RZF,
      MCBSP_SPCR_DLB_OFF,
      MCBSP_SPCR_XRST_YES,
      MCBSP_SPCR_XINTM_XRDY,
      MCBSP_SPCR_GRST_YES,
      MCBSP_SPCR_FRST_YES
    ),
    MCBSP_MK_RCR(
      MCBSP_RCR_RWDREVRS_DISABLE,
      MCBSP_RCR_RWDLEN1_32BIT,
      MCBSP_RCR_RFRLEN1_OF(0),
      MCBSP_RCR_RPHASE2_NORMAL,
      MCBSP_RCR_RDATDLY_1BIT,
      MCBSP_RCR_RFIG_YES,
      MCBSP_RCR_RCOMPAND_MSB,
      MCBSP_RCR_RWDLEN2_32BIT,
      MCBSP_RCR_RFRLEN2_OF(0),
      MCBSP_RCR_RPHASE_SINGLE
    ),
    MCBSP_MK_XCR(
      MCBSP_XCR_XWDREVRS_DISABLE,
      MCBSP_XCR_XWDLEN1_32BIT,
      MCBSP_XCR_XFRLEN1_OF(0),
      MCBSP_XCR_XPHASE2_NORMAL,
      MCBSP_XCR_XDATDLY_1BIT,
      MCBSP_XCR_XFIG_YES,
      MCBSP_XCR_XCOMPAND_MSB,
      MCBSP_XCR_XWDLEN2_32BIT,
      MCBSP_XCR_XFRLEN2_OF(0),
      MCBSP_XCR_XPHASE_SINGLE
    ),
    MCBSP_MK_SRGR(
      MCBSP_SRGR_CLKGDV_OF(0x5),
      MCBSP_SRGR_FWID_OF(0),
      MCBSP_SRGR_FPER_OF(32),
      MCBSP_SRGR_FSGM_FSG,
      MCBSP_SRGR_CLKSM_INTERNAL,
      MCBSP_SRGR_CLKSP_RISING,
      MCBSP_SRGR_GSYNC_FREE
    ),
    MCBSP_MK_MCR(0, 0, 0, 0, 0, 0),
    MCBSP_MK_RCER(0, 0),
    MCBSP_MK_XCER(0, 0),
```

```
    MCBSP_MK_PCR(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0)
  };

  hMcbsp = MCBSP_Open(MCBSP_DEV1, MCBSP_OPEN_RESET);
  MCBSP_ConfigA(hMcbsp, &mcbsp1_config);
  MCBSP_EnableSrgr(hMcbsp);
  MCBSP_EnableRcv(hMcbsp);
  MCBSP_EnableXmt(hMcbsp);
  MCBSP_EnableFsync(hMcbsp);
}
```

A.2 Interrupt Synchronization

```
/*****************************************************************************
* Function: irq_setup_tx
* Map physical interrupt 6 to transmit event of McBSP 1, clear pending
* interrupts, enable transmit event of McBSP 1, set initial event to start
* operation.
*****************************************************************************/
void irq_setup_tx()
{
  IRQ_Map(IRQ_EVT_XINT1, 6);
  IRQ_Clear(IRQ_EVT_XINT1);
  IRQ_Enable(IRQ_EVT_XINT1);
  IRQ_Set(IRQ_EVT_XINT1);
}

/*****************************************************************************
* Function: irq_setup_rx
* Map physical interrupt 7 to receive event of McBSP 1, clear pending
* interrupts, enable receive event of McBSP 1.
*****************************************************************************/
void irq_setup_rx()
{
  IRQ_Map(IRQ_EVT_RINT1, 7);
  IRQ_Clear(IRQ_EVT_RINT1);
  IRQ_Enable(IRQ_EVT_RINT1);
}

/*****************************************************************************
* Interrupt Service Routine: my_xint
* Send data from a table to the serial port.
*****************************************************************************/
void my_xint()
{
  if (i > TAB_SIZE - 1)
    i = 0;
  MCBSP_Write(hMcbsp, table[i++]);
}

/*****************************************************************************
* Interrupt Service Routine: my_rint
* Store data from the serial port in a table.
*****************************************************************************/
void my_rint()
{
  if (i > TAB_SIZE - 1)
    i = 0;
  table[i++] = MCBSP_Read(hMcbsp);
}
```

A.3 EDMA Initialization

```
/*****************************************************************************
* FUNCTION : edma_xevt1_init
*****************************************************************************/
void edma_xevt1_init()
{
  EDMA_HANDLE hEdmaTable, hEdma;

  hEdmaTable = EDMA_AllocTable(0);
  hEdma = EDMA_Open(EDMA_CHA_XEVT1, EDMA_OPEN_ENABLE);
  EDMA_ConfigB(hEdmaTable,
    0x21000002,
    TAB,
    TAB_SIZE,
    McBSP1_DXR,
    0x0,
    0x00000180
  );
  EDMA_ConfigB(hEdma,
    0x21000002,
    TAB,
    TAB_SIZE,
    McBSP1_DXR,
    0x0,
    0x00000180
  );
  EDMA_SetChannel(hEdma);
}

/*****************************************************************************
* FUNCTION : edma_revt1_init
*****************************************************************************/
void edma_revt1_init()
{
  EDMA_HANDLE hEdmaTable, hEdma;

  hEdmaTable = EDMA_AllocTable(0);
  hEdma = EDMA_Open(EDMA_CHA_REVT1, EDMA_OPEN_ENABLE);
  EDMA_ConfigB(hEdmaTable,
    0x202f0002,
    McBSP1_DRR,
    TAB_SIZE,
    TAB,
    0x0,
    0x00000180
  );
  EDMA_ConfigB(hEdma,
    0x202f0002,
    McBSP1_DRR,
    TAB_SIZE,
    TAB,
    0x0,
    0x00000180
  );
}
```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265