ANALOG DEVICES Engineer To Engineer Note EE-*156*

Technical Notes on using Analog Devices' DSP components and development tools
Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

Support for the H.100 protocol on the ADSP-2191

DSP Applications Group, March 2002 - Rev 0.1

The serial ports of the ADSP-2191 provide support for the H.100 standard protocol. It is also the International Telecommunication Union's recommendation for visual telephone systems. This application note describes the configurations necessary to be compatible with the H.100 standard protocol.

The hardware used to test the support for this protocol was the ADSP-2191 EZ-KIT Lite. The software used was VisualDSP++ 2.0 with SP1. The example code is included at the end of this application note.

Introduction

The ADSP-2191 has three independent, synchronous serial ports (SPORT0, SPORT1, and SPORT2). Each one of the serial ports supports H.100. In this example, SPORT1 is used. On each SPORT, data can be simultaneously transferred in both transmit and receive directions.

For each SPORT, data is transmitted from the IO bus to the Transmit Data register. After optional companding, data is transferred to the Transmit Shift register. Here the bits are shifted out serially on the SPORT's DT pin.

The reverse happens for the receive direction. Data is accepted on the SPORT's DR pin, and serially transmitted to the Receive Shift register. After a word is received and optional de-companding, the data is transferred to the Receive Data register.

By writing to different control registers, the serial clock frequency, data format and length, multi-channel mode select, and other parameters can be programmed on the SPORTs.

Direct Memory Access (DMA) is also supported on each SPORT. The transmit channel enables DMA transfers from memory to SPORT, while the receive channel enables DMA transfers from SPORT to memory.

Hardware Connections

On the ADSP-2191 EZ-KIT Lite, the SPORT0 and SPORT1 pins are brought out via the P6 and P7 SPORT connectors, respectively.

Using SPORT1 in this application, the DT1 and DR1 pins are connected together to create a loopback of the data transmitted and received.

Because H.100 applications always runs in multi-channel mode, the TCLK1 and RCLK1 pins must be wired together. In multi-channel mode, the TCLK pin is always an input. In this application, RCLK1 is generated internally.

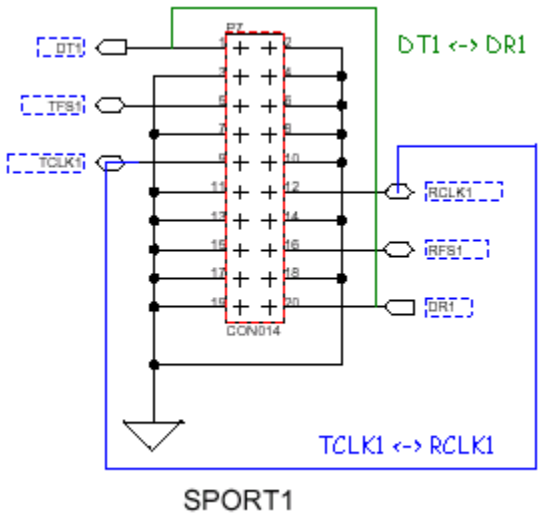

Figure 1: SPORT pins on the EZ-KIT Lite

Software Configurations

The assembly program consists of 4 main sections. These sections perform the tasks of initializing SPORT1, configuring SPORT1 for multi-channel mode operation, setting up autobuffer-based DMA, and enabling SPORT1.

To be compatible with the H.100 standard protocol, certain parameters must be set accordingly to meet the H.100 specification. The following is a list of these parameters:

- multi-channel mode enabled
- 8-bit words
- 1024 clock cycles per frame, 122 ns wide, 125 us period frame sync
- transmit/receive frame sync required
- active low transmit/receive frame sync
- 8.192 MHz (+/- 2% bit clock)
- no frame delay between frame sync and first data bit
- half clock cycle early frame sync

Initializing SPORT1

The transmit configuration register is initialized to the Hex value of 0x1CF0. This value configures SPORT1 to transmit 8-bit words, generate a transmit frame sync for every data word, and selects an active low transmit frame sync. Note that the SPORT1 Transmit Enable bit is not set here. This bit will be set last, after all other configurations have been programmed.

SP1_TCR (0x03:0x000)

0	0	0	1	1	1	0	0	1	1	1	1	0	0	0	0

Figure 2: Transmit Configuration Register

The receive configuration register is initialized to the Hex value of 0x1EF2. This value configures SPORT1 to receive 8-bit words, generate a receive frame sync for every data word, and selects an active low receive frame sync. Note that the SPORT1 Receive Enable bit is not set here. This bit will be set last, after all other configurations have been programmed.

SP1_RCR (0x03:0x001)

0	0	0	1	1	1	1	0	1	1	1	1	0	0	1	0

Figure 3: Receive Configuration Register

The SP1_TFSDIV and SP1_RFSDIV registers hold the number of transmit and receive clock cycles to count before generating a transmit or receive frame sync. Both of these registers are set to a Hex value of 0x03FF. This gives a value of 1024 clock cycles per frame.

The serial clock frequency is calculated according to the following equation:

$$SP1\_CLK = \frac{HCLK}{2 * (SP1\_SCKDIV + 1)}$$

With an input clock of 13 MHz and a multiplier of 10, the core clock frequency becomes 130 MHz. Having the peripheral clock HCLK be half of core clock and setting both the SP1_TSCKDIV and SP1_RSCKDIV registers to be 0x3, the SP1_TCLK and SP1_RCLK become 8.125 MHz. This frequency is within the +/- 2% of 8.192 MHz range.

Multi-channel mode operation

There are 8 multi-channel transmit selection registers and 8 multi-channel receive selection registers. Each register consists of 16 bits. Each bit corresponds to 1 channel. By setting the 8 multi-channel transmit selection registers to a Hex value of 0xFFFF, all 128 transmit channels (8 registers * 16 channels) are enabled.

By setting the 8 multi-channel receive selection registers to a Hex value of 0xFFFF, all 128 receive channels (8 registers * 16 channels) are enabled.

There are two multi-channel configuration registers for SPORT1. SP1_MCMC1 is initialized to a Hex value of 0x01E1. This setting enables multi-channel mode operations and a window size corresponding to 128 channels.

SP1_MCMC1 (0x03:0x019)

0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1

Figure 4: Mulit-Channel Configuration Register 1

SP1_MCMC2 is initialized to a Hex value of 0x02AC. In this setting, a half clock cycle early frame sync is enabled.

SP1_MCMC2 (0x03:0x01A)

0	0	0	0	0	0	1	0	1	0	1	0	1	1	0	0

Figure 5: Mulit-Channel Configuration Register 2

Autobuffer-based DMA

In this section of code, four registers must be configured to setup SPORT1 autobuffer-based DMA on both the transmit and receive.

Both the transmit and receive DMA configuration registers are initially set to a Hex value of 0x0010. This setting enables autobuffer mode.
Then, the receive DMA start page and address registers are written with the page and address at which the receive data buffer is located. The receive DMA count register is written with the length of the receive data buffer.

The transmit DMA start page and address registers are written with the page and address at which the transmit data buffer is located. The transmit DMA count register is written with the length of the transmit data buffer.

Next, the receive DMA configuration register is written again with a Hex value of 0x0013. This setting configures a receive DMA transfer and enables DMA.

SP1DR_CFG (0x03:0x101)

0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Figure 6: SPORT1 Receive DMA Configuration Register

The transmit DMA configuration register is written again with a Hex value of 0x0015. This

setting enables DMA and an interrupt upon completion of the DMA.

SP1DT_CFG (0x03:0x181)

0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1

Figure 7: SPORT1 Transmit DMA Configuration Register

Enable SPORT1

After all of the above configurations are set up, SPORT1 for both transmit and receive are enabled by setting bit 0 of the SP1_TCR and SP1_RCR registers, respectively.

Results

The following diagrams are some screenshots taken from the results of running this applications.

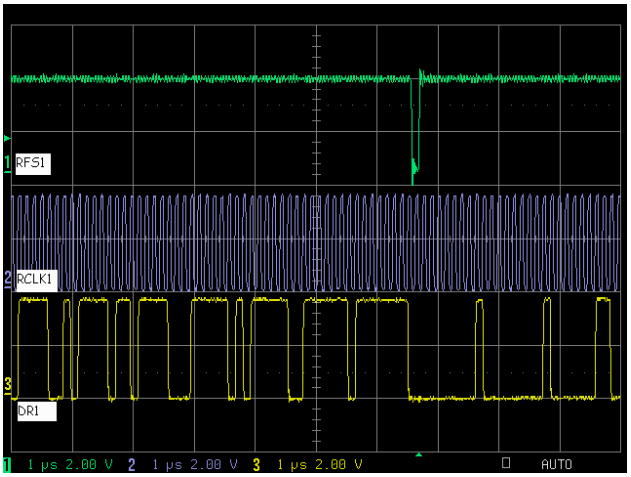

Figure 8: Timing Diagram 1

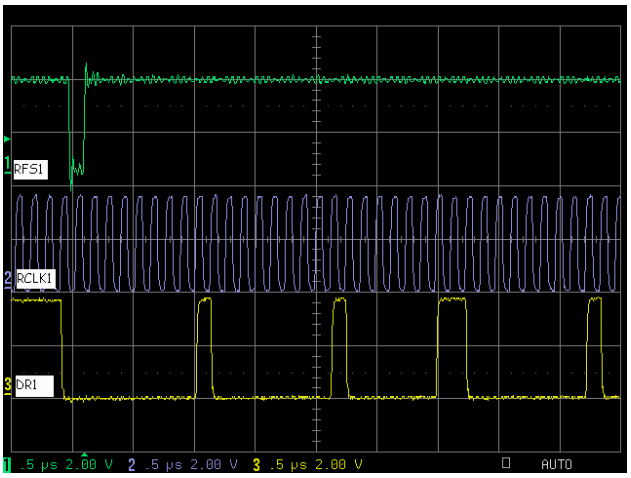

Figure 9: Timing Diagram 2

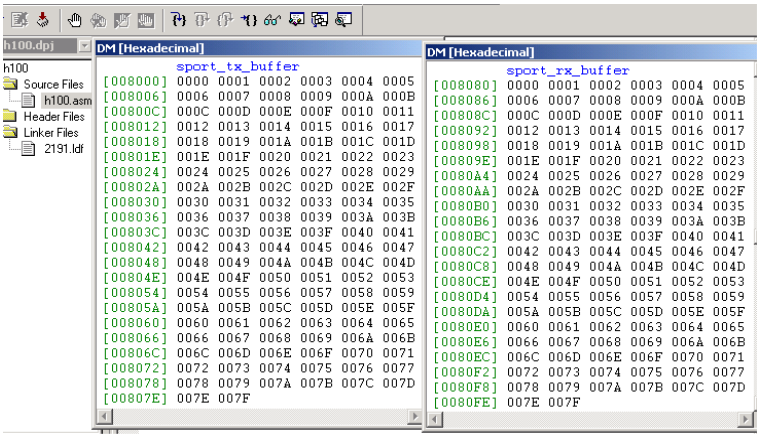

Figure 10: Transmit and Receive Data Buffer Results

References

1. Analog Devices, Inc. DSP Home: http://www.analog.com/technology/dsp

2. International Telecommunication Union: www.itu.int
3. H.100 References: http://www.scsa.org/ http://www.ectf.org/

Example Code

```
#include <def2191.h>

.section/dm seg_dmdata;
.var sport_tx_buffer[128] = "h100.dat";
.var sport_rx_buffer[128];

/************************/
/***** Reset Vector *****/
/************************/

.section/pm seg_ivt;
   jump start;

.section/pm seg_ivtint4;
   ay0 = IOPG;

   /*Set SPORT1 Registers*/
   IOPG = SPORT1_Controller_Page;
   ar = 3;
   io(SP1DT_IRQ) = ar;
   IOPG = ay0;

   nop; nop;

   rti;

.section/pm seg_pmcode;

start:

/*********************************/
/*******SPORT Initialization*******/
/*********************************/

/*Disable interrupts*/
   DIS int;
   IRPTL = 0x0;
   ICNTL = 0;
   IMASK = 0;
/*Set interrupt priorities*/
    IOPG = Interrupt_Controller_Page;
   ar = 0xbbb0;
   io(IPR1) = ar;
   ar = 0xbbbb;
   io(IPR0) = ar;
   io(IPR2) = ar;
   io(IPR3) = ar;

   /*Set SPORT1 Registers*/
   IOPG = SPORT1_Controller_Page;
   ar = 0x0000;
   io(SP1_TCR) = ar;
   io(SP1_RCR) = ar;
   io(SP1DR_CFG) = ar;
   io(SP1DT_CFG) = ar;
   io(SP1_MCMC1) = ar;
   ar = 3;
   io(SP1DT_IRQ) = ar;
   io(SP1DR_IRQ) = ar;

/*SPORT Tx and Rx Clock Divide*/
ax0=0x3;
io(SP1_RSCKDIV)=ax0;
io(SP1_TSCKDIV)=ax0;

/*SPORT Tx and Rx Frame Sync*/
/*1024 clock cycles per frame,
8bits*128channels*/
ax0=0x3ff;
io(SP1_RFSDIV)=ax0;
io(SP1_TFSDIV)=ax0;

/*SPORT Rx Configuration Register*/
/*SLEN, TFSR/RFSR, LTFS, DITFS(only
TX)*/
ax0=0x1Ef2;
io(SP1_RCR)=ax0;

/*SPORT Tx Configuration Register*/
/*SLEN, TFSR/RFSR, LTFS, DITFS(only
TX)*/
ax0=0x1cf0;
io(SP1_TCR)=ax0;

/*************************************/
/** Multichannel Mode Configuration **/
/*************************************/

/*MCM Tx and Rx Channel Select
Register*/
/*Enable all 128 channels for Tx*/
ax0=0xffff;
io(SP1_MTCS0)=ax0;
io(SP1_MTCS1)=ax0;
io(SP1_MTCS2)=ax0;
io(SP1_MTCS3)=ax0;
io(SP1_MTCS4)=ax0;
io(SP1_MTCS5)=ax0;
io(SP1_MTCS6)=ax0;
io(SP1_MTCS7)=ax0;
/*Enable all 128 channels for Rx*/
ax0=0xffff;
io(SP1_MRCS0)=ax0;
```

```
io(SP1_MRCS1)=ax0;
io(SP1_MRCS2)=ax0;
io(SP1_MRCS3)=ax0;
io(SP1_MRCS4)=ax0;
io(SP1_MRCS5)=ax0;
io(SP1_MRCS6)=ax0;
io(SP1_MRCS7)=ax0;
/*SPORT MCM Configuration Reg 1*/
ax0=0x01e1;
io(SP1_MCMC1)=ax0;
ax0=0x02ac;
io(SP1_MCMC2)=ax0;

/* SPORT1 Interrupts Unmasked */
   AX0=IMASK;
   AR=SETBIT 4 of AX0;
   IMASK=AR;

/************************************/
/******Autobuffer-based DMA Setup*****/
/************************************/

ax0 = 0x0010;
io(SP1DR_CFG)=ax0;
ax1=page(sport_rx_buffer);
io(SP1DR_SRP)=ax1;
ax1=address(sport_rx_buffer);
io(SP1DR_SRA)=ax1;
ax1=length(sport_rx_buffer);
io(SP1DR_CNT)=ax1;

ax0=0x0010;
io(SP1DT_CFG)=ax0;
ax1=page(sport_tx_buffer);
io(SP1DT_SRP)=ax1;
ax1=address(sport_tx_buffer);
io(SP1DT_SRA)=ax1;
ax1=length(sport_tx_buffer);
io(SP1DT_CNT)=ax1;

ax0 = 0x0013;
io(SP1DR_CFG)=ax0;

ax0 = 0x0015;
io(SP1DT_CFG)=ax0;

/*****************************/
/********Enable SPORT1*******/
/*****************************/

ax0=io(SP1_RCR);
ar=setbit 0 of ax0;
io(SP1_RCR)=ar;
ax0=io(SP1_TCR);
ar=setbit 0 of ax0;
io(SP1_TCR)=ar;

ena int;

wait_here:  idle;
       jump wait_here;
```