Application Report
SPRA957 − September 2003

TMS320C54xx McBSP to TLV320AIC24 Interface

Hong-Swee Lim *Semiconductor Group*

ABSTRACT

This application note describes a method for interfacing the TMS320C5400 multichannel buffer serial ports (McBSP) to the TLV320AIC24.

The TLV320AIC24 innovation is the smart-time division, multiplexed serial port (SMARTDM™) that optimizes the DSP performance with an advanced synchronous 4-wire serial port in TDM format for glue-free interface to popular DSPs (i.e., C5000, C6000) and microcontrollers. The SMARTDM supports both continuous data transfer mode and on-the-fly reconfiguration programming mode (both ADC/DAC and control data). The SMARTDM maximizes the bandwidth of data transfer (ADC/DAC data only) between the TLV320AIC24 DSP codec and the DSP. In normal operation, it automatically detects the number of codecs in the serial interface and adjusts the number of time slots to match the number of codecs so that no time slot in the TDM frame is wasted. In the turbo operation, it maintains the same number of time slots but maximizes the bit transferred rate to 25 MHz, to allow other serial DSP peripheral devices to share the same serial bus within the same sampling period. The TLV320AIC24 can be gluelessly cascaded to any SMARTDM-based device to form a multichannel codec. Up to eight TLV320AIC24 codecs can be cascaded to a single serial port.

A discussion on the hardware interface and the software will be presented in sections 2 and 3. The sample code described in this document can be downloaded from http://www.ti.com/lit/zip/SPRA957.

Contents

List of Figures

List of Tables

1 Introduction

The TLV320AIC24 codec from Texas Instruments (TI) provide a glueless serial interface to the McBSPs on the TMS320C54xx DSP family. The TLV320AIC24 features two 16-bit analog-to-digital (A/D) channels and two 16-bit digital-to-analog (D/A) channels, which can be connected to a handset, headset, microphone, or a subscriber line via a programmable analog crosspoint. The TLV320AIC24 provides a flexible host port. The host port interface is a two-wire serial interface that can be programmed to be either an industrial standard I^{2}C or a simple S^{2}C (start-stop communication protocol). The TLV320AIC24 innovation is the smart-time division multiplexed serial port (SMARTDM) that optimizes the DSP performance with an advanced synchronous 4-wire serial port in TDM format for glue-free interface to popular DSPs (i.e., C5x, C6x) and microcontrollers. The SMARTDM supports both continuous data transfer mode and on-the-fly reconfiguration programming mode (both ADC/DAC and control data).

This application note presents the hardware connections and software necessary to interface the TLV320AIC24 with the McBSP on the TMS320C54xx DSP.

2 Hardware Interface

To successfully use the McBSP to interface to the TLV320AIC24, four SMARTDM signals: frame-sync (FS), serial data out (DOUT), serial data in (DIN) and clock source (SCLK) have to connect to FSX/FSR, DR, DX and CLKX/CLKR on the McBSP respectively. Figure 1 shows the necessary connections between Cascade TLV320AIC24 and the DSP.

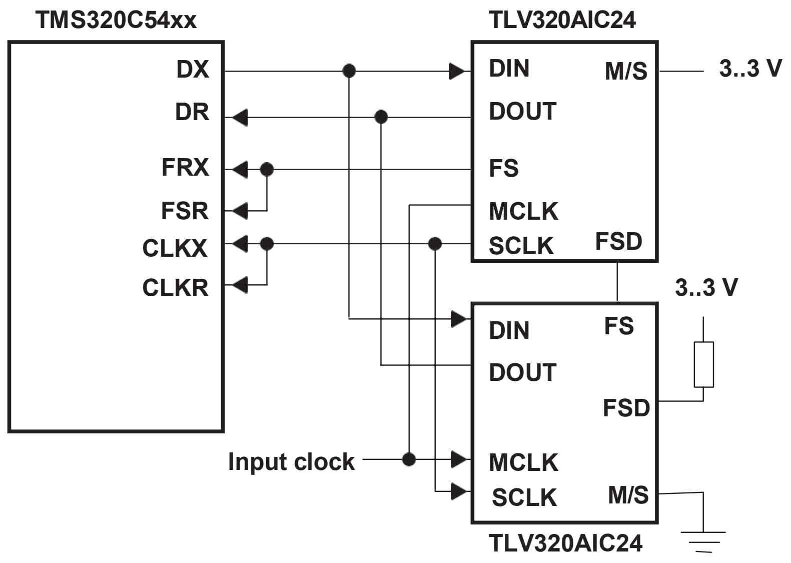

Figure 1. McBSP Connection to Cascade TLV320AIC24 Codec

In the application shown, the TLV320AIC24 generates the clock and frame sync, which are connected to the McBSP CLKX/CLKR and FSX/FSR pins, respectively. A 20.48 MHz crystal is connected to the TLV320AIC24 voice codec. The AIC24 internally divides this fixed-rate clock signal by a factor of 2560 (which is software programmable) to produce an 8 KHz clock signal output on the FS pin. This FS signal is connected to the FSX/FSR pin of the McBSP. The AIC24 also generate the SCLK from the frame sync signal.

In this application note, we will be using a master-slave cascade configuration to connect two TLV320AIC24 codecs in standard operation using programming mode to initialize the AIC24 and remain in programming mode, even in data transfer mode.

3 Configuring the McBSP Registers

The McBSP is configured with single phase communication and 16-bit word length. Table 1 through Table 7 show the settings for the McBSP registers.

The DSP initializes the TLV320AIC24 via the SMARTDM link by using the DMA. The code provided with this report demonstrates an example of programming the codec registers.

Table 1. Serial Port Control Register 1 (SPCR1x)

Bit	Name	Value	Function
15	DLB	0	Digital loop back mode disabled
14−13	RJUST	00	Right-justify and zero-fill MSBs in DRR[1,2]
12−11	CLKSTP	00	Normal clocking for non-SPI mode
10−8	Reserved	xxx	Reserved
7	DXENA	0	DX enabler is off
6	ABIS	0	A-bis mode is disabled
5−4	RINTM	00	Receiver interrupt driven by RRDY
3	RSYNCERR	0	No synchronization error
2	RFULL	x	Read only
1	RRDY	x	Read only
0	$\overline{\text{RRST}}$	0	Serial port receiver is disabled

Table 2. Serial Port Control Register 2 (SPCR2x)

Bit	Name	Value	Function
15−10	Reserved	xxxxxx	Reserved
9	FREE	1	Free running
8	SOFT	x	Don't care when FREE is 1
7	$\overline{\text{FRST}}$	0	Frame-sync generator is disabled
6	GRST	0	Sample-rate generator is disabled
5−4	XINTM	00	Transmit interrupt driven by XRDY
3	XSYNCERR	0	No synchronization error detection
2	$\overline{\text{XEMPTY}}$	0	Read only
1	XRDY	0	Read only
0	$\overline{\text{XRST}}$	0	Serial port transmitter is disabled

Table 3. Pin Control Register (PCR)

Bit	Name	Value	Function
15−14	Reserved	xx	Reserved
13	XIOEN	0	DX, FSX and CLKX are not GPIO pins
12	RIOEN	0	DR, FSR and CLKR are not GPIO pins
11	FSXM	0	External transmit frame sync signal
10	FSRM	0	External receive frame sync signal
9	CLKXM	0	External transmit clock signal
8	CLKRM	0	External receive clock signal
7	Reserved	x	Reserved
6	CLKS_STAT	0	Read only
5	DX_STAT	0	Read only
4	DR_STAT	0	Read only
3	FSXP	1	Transmit frame sync is active low
2	FSRP	1	Receive frame sync is active low
1	CLKXP	0	Transmit data sampled on rising CLKX edge
0	CLKRP	0	Receive data sampled on falling CLKR edge

Table 4. Receive Control Register 1 (RCR1x)

Bit	Name	Value	Function
15	Reserved	x	Reserved
14−8	RFRLEN1	0000000	1 word per frame
7−5	RWDLEN1	010	16 bits per word
0−4	Reserved	xxxxx	Reserved

Table 5. Receive Control Register 2 (RCR2x)

Bit	Name	Value	Function
15	RPHASE	0	Single phase frame
14−8	RFRLEN2	xxxxxxx	Don’t care when single phase frame
7−5	RWDLEN2	xxx	Don’t care when single phase frame
4−3	RCOMPAND	00	No companding transfer starts with MSB
2	RFIG	1	Ignore receive frame syncs after the first
1−0	RDATDLY	00	0-bit data delay

Table 6. Transmit Control Register 1 (XCR1x)

BIT	NAME	VALUE	FUNCTION
15	Reserved	x	Reserved
14−8	RFRLEN1	0000000	1 word per frame
7−5	RWDLEN1	010	16 bits per word
0−4	Reserved	x	Reserved

Table 7. Transmit Control Register 2 (XCR2x)

BIT	NAME	VALUE	FUNCTION
15	XPHASE	0	Single phase frame
14−8	XFRLEN2	xxx	Don’t care when single phase frame
7−5	XWDLEN2	xxx	Don’t care when single phase frame
4−3	XCOMPAND	00	No companding transfer starts with MSB
2	XFIG	1	Ignore transmit frame syncs after the first
1−0	XDATDLY	00	0-bit data delay

4 Conclusion

An example to configure the TLV320ACI24 using SMARTDM link via McBSP on the TMS320C54xx is presented. Modular C-code is also attached in this application report that performs McBSP, DMA and codec initialization.

5 References

1. *TLV320AIC24 Data Manual* (literature number SLAS366A)

Appendix A Sample Code

A.1 Main.c

```
/******************************************************************************
*
* FILE: MAIN.C
*
* DESCRIPTION:    Main program for interfacing TLV320AIC24 CODEC to TMS320C5410 using SMARTDM
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
******************************************************************************/
#include <stdlib.h>
#include ”evm5410.h”
#include ”dmac.h”
#include ”mcbsp5410.h”
#include ”codec_controls.h”
extern short dmac_ch0A[];
extern short dmac_ch1A[];
extern short dmac_ch0B[];
extern short dmac_ch1B[];
/*****************************************************************************
* PUBLIC FUNCTION DEFINITIONS
*****************************************************************************/
int dmacount0 = 0;
int dmacount1 = 0;
int receive =0;
int transmit =0;
int tx_empty=1;
int rx_full=0;
unsigned short mode = 0;
unsigned short frame = 0;
main()
{
    unsigned short i = 0;

    init_core();

    // Serial Port initialisation
    SerialPort0Init();
    // DMA initialisation
    DMAC_init();
    DMAC_ch_enable(DMAC_CH_0);
    DMAC_ch_enable(DMAC_CH_1);
    *IMR = DMAC0 |DMAC1;
    *IFR = 0xffff;  /*clear all pending interrupts*/
    SerialPort0_TXRX_Enable();
    GLOBAL_INT_ENABLE;

    while ( 1 )
    {
          if((mode==1) &&(rx_full==1)&&(tx_empty==0))
          {
if((receive==1)&&(transmit==1))
    {
         for (i=0; i<8; i++)
             dmac_ch1A[i]=dmac_ch0A[i];
        }
    if((receive==0)&&(transmit==0))
    {
         for (i=0; i<8; i++)
             dmac_ch1B[i]=dmac_ch0B[i];
        }
         rx_full=0;
         tx_empty=1;
          }
    }
}
```

A.2 AIC24.CMD

```
/*********************************************************************************
*
* FILE: AIC24.CMD
*
* DESCRIPTION:    Linker command file for interfacing TLV320AIC24 CODEC to TMS320C5410 using
SMARTDM
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
******************************************************************************/
MEMORY
```

```
{
    PAGE 0:  PRAM0:   origin = 0x4000 length = 0x2000
    PAGE 0:  PRAM1:   origin = 0x6000 length = 0x2000
    PAGE 0:  MYVECT:  origin = 0xFF80 length = 0x80
    PAGE 1:  DRAM0:   origin = 0x0060 length = 0x0020
    PAGE 1:  DRAM1:   origin = 0x0080 length = 0x1f80
    PAGE 1:  DRAM2:   origin = 0x6000 length = 0x2000
}
SECTIONS
{
    myvectors  > MYVECT    PAGE 0
    .text      > PRAM0     PAGE 0
    .cinit     > PRAM1     PAGE 0
    .bss       > DRAM1     PAGE 1
    .stack     > DRAM2     PAGE 1
    dmaMem     > DRAM1     PAGE 1
}
```

A.3 CODEC_CONTROLS.C

```
/********************************************************************************
*
* FILE: Codec_controls.c
*
* DESCRIPTION:    C functions for Codec Control
*
* AUTHOR:         HONG-SWEE LIM  –  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
*******************************************************************************/
#include ”codec_controls.h”
#include ”mcbsp5410.h”
/*******************************************************************************
* EXTERNAL REFERENCE
*******************************************************************************/
extern volatile unsigned int *bdxr0Ptr;
extern volatile unsigned int  *bdrr0Ptr;
extern volatile unsigned int  *bdxr1Ptr; /*from main program*/
extern volatile unsigned int  *bdrr1Ptr;
#define CTRL_BUF_SIZE    40
#pragma DATA_SECTION(dmac_ctrl, ”dmaMem”);
short dmac_ctrl[CTRL_BUF_SIZE]=
{   0,0,0,0, AIC_RESETs, 0,0,0,
    0,0,0,0, AIC_Ms, AIC_NFs, AIC_DACs, AIC_MUTE_SITONs,
    0,0,0,0, AIC_CH1_IN, AIC_CH2_IN, AIC_CH1_IN, AIC_CH2_IN,
    0,0,0,0, AIC_CH1_OUT, AIC_CH2_OUT, AIC_CH1_OUT, AIC_CH2_OUT,
    0,0,0,0, AIC_ADCs, AIC_16BITs ,0,0
};
extern unsigned short write_command;
#define REG_DATA_MASK     0x00FF;
unsigned short regerror = 0;
unsigned short regcount = 0;
#define GLOBAL_INT_ENABLE    asm( ” rsbx intm ”) /* enables interrupts*/
#define GLOBAL_INT_DISABLE   asm( ” ssbx intm ”) /* disable interrupts*/

void write_aic24_ctl( unsigned short write_cmd)
{

    codec_control.write_cmd = write_cmd;
    *MCBSP0_DXR1= write_cmd;
}
```

A.4 CODEC_CONTROLS.H

```
/*******************************************************************************
*
* FILE: Codec_controls.h
*
* DESCRIPTION: C header file for Codec Control
*
* AUTHOR: HONG-SWEE LIM - Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
******************************************************************************/
#ifndef codec_controls_h
#define codec_controls_h
#define CODEC_NOOP            0x0700
#define CODEC_REG1            0x2700
#define CODEC_REG2            0x4700
#define CODEC_REG3            0x6700
#define CODEC_REG4            0x8700
#define CODEC_REG5            0xA700
#define CODEC_REG6            0xC700
#define   CODEC_READ_AIC24    0x1000
#define CODEC_BROADCAST       0x0800
```

```
//  Control Register 1
#define REG1_CONTINUOUS        0x0040
#define REG1_IIR_EN            0x0020
#define   REG1_MIC_BIAS_235    0x0008
#define  REG1_ANALOG_LOOP_BACK 0x0004
#define REG1_DIGITAL_LOOP_BACK 0x0002
#define REG1_DAC16             0x0001

// Control Register 2
#define REG2_TURBO_EN          0x0080
#define REG2_FIR_BYPASS        0x0040
#define REG2_GPIO              0x0002
#define REG2_GPIO_1            0x0006
// Control Register 3A
#define REG3_PWDN_ALL          0x0030
#define REG3_PWDN_ADC          0x0010
#define REG3_PWDN_DAC          0x0020
#define   REG3_SW_RESET        0x0008
#define  REG3_SAMPLING_FACTOR1 0x0001
#define REG3_SAMPLING_FACTOR2  0x0002
// Control Register 3B
#define REG3_8KBP_EN           0x0060
#define REG3_MUTE_OUTP1        0x0042
#define REG3_MUTE_OUTP2        0x0048
#define REG3_MUTE_OUTP3        0x0044
//  Control Register 4
#define REG4_FSDIV_M           0x0085  //M=5
#define REG4_FSDIV_NP          0x0008  //N=1, P=8
#define REG4_FSDIV_NP1         0x0002  //N=16, P=2
//  Control Register 5
#define REG5A_ADC_GAIN         0x0002  //3dB
#define REG5A_ADC_MUTE         0x000f  //Mute
#define REG5B_DAC_GAIN         0x0042  //-3dB
#define REG5B_DAC_MUTE         0x004f  //Mute
#define REG5C_SIDETONE_MUTE    0x00BF

//  Control Register 6
#define REG6A_AIC24A_CH1_IN     0x0008  //INP1 to ADC
#define REG6B_AIC24A_CH1_OUT    0x0082  //OUTP2 to DAC
#define REG6A_AIC24A_CH2_IN     0x0002  //INP2 to ADC
#define REG6B_AIC24A_CH2_OUT    0x0081  //OUTP3 to DAC
```

```
//  AIC24 command
#define AIC_RESETs       CODEC_REG3|REG3_SW_RESET |CODEC_BROADCAST
#define AIC_Ms           CODEC_REG4|REG4_FSDIV_M|CODEC_BROADCAST
#define AIC_NFs          CODEC_REG4|REG4_FSDIV_NP|CODEC_BROADCAST
#define AIC_16BITs       CODEC_REG1|REG1_DAC16|CODEC_BROADCAST
#define AIC_MUTE_SITONs CODEC_REG5|REG5C_SIDETONE_MUTE|CODEC_BROADCAST
#define AIC_CH1_IN       CODEC_REG6|REG6A_AIC24A_CH1_IN
#define AIC_CH1_OUT      CODEC_REG6|REG6B_AIC24A_CH1_OUT
#define AIC_CH2_IN       CODEC_REG6|REG6A_AIC24A_CH2_IN
#define AIC_CH2_OUT      CODEC_REG6|REG6B_AIC24A_CH2_OUT
#define AIC_ADCs         CODEC_REG5|REG5A_ADC_GAIN|CODEC_BROADCAST
#define AIC_DACs         CODEC_REG5|REG5B_DAC_GAIN|CODEC_BROADCAST

void write_aic24_ctl( unsigned short write_cmd);
#endif /*  codec_controls_h END OF FILE -------------------------------------*/
```

A.5 DMAC.C

```
/********************************************************************************
*
* FILE: DMAC.C
*
* DESCRIPTION:    C functions for DMA initialisation and ISR
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
******************************************************************************/
#include "dmac.h"
#include "mcbsp5410.h"
extern int dmacount0;
extern int dmacount1;
extern int receive;
extern int transmit;
extern int tx_empty;
extern int rx_full;
extern unsigned short mode;
extern unsigned short frame;
extern short dmac_ctrl[];
#define IPC_BUF_SIZE  256
#pragma DATA_SECTION(dmac_ch0A, "dmaMem");
#pragma DATA_SECTION(dmac_ch1A, "dmaMem");
short dmac_ch0A[IPC_BUF_SIZE];
short dmac_ch1A[IPC_BUF_SIZE];
#pragma DATA_SECTION(dmac_ch0B, ”dmaMem”);
#pragma DATA_SECTION(dmac_ch1B, ”dmaMem”);
short dmac_ch0B[IPC_BUF_SIZE];
short dmac_ch1B[IPC_BUF_SIZE];
void DMAC_init()
{
     int i;
     // Initialize all channel to be disable and low priority
     *(DMPREC) = 0x8380;         //Free, INTOSEL: 10B, Channel 0,1 high priority
     // Configure channel 0
     *(DMSA) = 0;
     *(DMSDI) = (unsigned short) MCBSP0_DRR1;// DMSRC0 – ch 0 source address register
     *(DMSDI) = (unsigned short) dmac_ch0A;  // DMDST0 – ch 0 destination address register
     *(DMSDI) = 0x07;     // DMCTR0 – ch 0 element count register. Set 15 will transfer 16
     *(DMSDI) = 0x1000;   // DMSFC0 – ch 0 sync on McBSP0 receive event and frame count of 64
     *(DMSDI) = 0x4045;   // DMMCR0 – ch 0 transfer mode control register
     // Configure channel 1
     *(DMSA) = 5;
     *(DMSDI) = (unsigned short) dmac_ctrl+1;// DMSRC1 – ch 1 source address register
     *(DMSDI) = (unsigned short) MCBSP0_DXR1;// DMDST1 – ch 1 destination address register
     *(DMSDI) = 0x06;    // DMCTR1 – ch 1 element count register
     *(DMSDI) = 0x2000;  // DMSFC1 – ch 1 sync on McBSP0 transmit event and frame count of 64
     *(DMSDI) = 0x4141;  // DMMCR1 – ch 1 transfer mode control register
     // Set Element & Frame Address Index Register 0
     *(DMSA) = 0x20;
     *(DMSDN)= 0x1;
     *(DMSA) = 0x22;
     *(DMSDN)= 0x1;
     for (i = 0; i < IPC_BUF_SIZE; i++)
     {
          dmac_ch0A[i] = 0x0555;
          dmac_ch0B[i] = 0x0555;
          dmac_ch1A[i] = 0x0555;
          dmac_ch1B[i] = 0x0555;
     }
}
void DMAC_ch_enable(int ch_num)
{
     *(DMPREC) |= ch_num;
}
void DMAC_ch_disable(int ch_num)
{
     *(DMPREC) &= ~(ch_num);
}
interrupt void DMAC0_isr()
{
     if (mode==1)
     {
          if (rx_full==0)
          {
    if (receive ==0)
    {
         // Configure channel 0
         *(DMSA) = 0;
         *(DMSDI) = (unsigned short) MCBSP0_DRR1;// DMSRC0 – ch 0 source address register
         *(DMSDI) = (unsigned short) dmac_ch0B;// DMDST0 – ch 0 destination address register
         *(DMSDI) = 0x07;  // Check if reload is needed. DMCTR0 – ch 0 element count register
         *(DMSDI) = 0x1000;
         receive = 1;
    }
    else
    {
         *(DMSA) = 0;
         *(DMSDI) = (unsigned short) MCBSP0_DRR1;// DMSRC0 – ch 0 source address register
         *(DMSDI) = (unsigned short) dmac_ch0A; // DMDST0 – ch 0 destination address register
         *(DMSDI) = 0x07;  // Check if reload is needed. DMCTR0 – ch 0 element count register
         *(DMSDI) = 0x1000;
          receive = 0;
    }
    dmacount0++;
    rx_full=1;
          }
          DMAC_ch_enable(DMAC_CH_0);
     }

     else
     {
          if(frame>=4)
          {
    // Configure channel 0
    *(DMSA) = 0;
    *(DMSDI) = (unsigned short) MCBSP0_DRR1;// DMSRC0 – ch 0 source address register
    *(DMSDI) = (unsigned short) dmac_ch0A; // DMDST0 – ch 0 destination address register
    *(DMSDI) = 0x07; // Check if reload is needed. DMCTR0 – ch 0 element count register
    *(DMSDI) = 0x1000; // DMSFC0 – ch 0 sync on McBSP0 receive event and frame count of 64
    *(DMSDI) = 0x4045; // DMMCR0 – ch 0 transfer mode control register
          }
          else
          {
    // Configure channel 0
    *(DMSA) = 0;
    *(DMSDI) = (unsigned short) MCBSP0_DRR1;// DMSRC0 – ch 0 source address register
         *(DMSDI) = (unsigned short) dmac_ch0A; // DMDST0 – ch 0 destination address register
         *(DMSDI) = 0x07;    // DMCTR0 – ch 0 element count register. Set 15 will transfer 16
         *(DMSDI) = 0x1000;
    }
    DMAC_ch_enable(DMAC_CH_0);
          }
}
interrupt void DMAC1_isr()
{
     if(mode==1)
     {
          if (tx_empty==1)
          {
               if (transmit ==0)
    {
         // Configure channel 1
         *(DMSA) = 5;
         *(DMSDI) = (unsigned short) dmac_ch1B;  // DMSRC1 – ch 1 source address register
         *(DMSDI) = (unsigned short) MCBSP0_DXR1;// DMDST1– ch 1 destination address register
         *(DMSDI) = 0x07;                        // DMCTR1 – ch 1 element count register
         *(DMSDI) = 0x2000;
         transmit = 1;
    }
    else
    {
         // Configure channel 1
         *(DMSA) = 5;
         *(DMSDI) = (unsigned short) dmac_ch1A;  // DMSRC1 – ch 1 source address register
         *(DMSDI) = (unsigned short) MCBSP0_DXR1;// DMDST1 –ch 1 destination address register
         *(DMSDI) = 0x07;                        // DMCTR1 – ch 1 element count register
         *(DMSDI) = 0x2000;
         transmit = 0;
    }
    dmacount1++;
    tx_empty=0;
          }
          DMAC_ch_enable(DMAC_CH_1);
     }

     else
     {
          if(frame==0)
          {
               // Configure channel 1
   *(DMSA) = 5;
   *(DMSDI) = (unsigned short) dmac_ctrl+8;// DMSRC1 - ch 1 source address register
   *(DMSDI) = (unsigned short) MCBSP0_DXR1;// DMDST1 - ch 1 destination address register
   *(DMSDI) = 0x07;                         // DMCTR1 - ch 1 element count register
   *(DMSDI) = 0x2000;
          }
          if(frame==1)
          {
   // Configure channel 1
   *(DMSA) = 5;
   *(DMSDI) = (unsigned short) dmac_ctrl+16;// DMSRC1 - ch 1 source address register
   *(DMSDI) = (unsigned short) MCBSP0_DXR1; // DMDST1 - ch 1 destination address register
   *(DMSDI) = 0x07;                          // DMCTR1 - ch 1 element count register
   *(DMSDI) = 0x2000;
          }
          if(frame==2)
          {
   // Configure channel 1
   *(DMSA) = 5;
   *(DMSDI) = (unsigned short) dmac_ctrl+24;// DMSRC1 - ch 1 source address register
   *(DMSDI) = (unsigned short) MCBSP0_DXR1; // DMDST1 - ch 1 destination address register
   *(DMSDI) = 0x07;                          // DMCTR1 - ch 1 element count register
   *(DMSDI) = 0x2000; // DMSFC1 - ch 1 sync on McBSP0 transmit event and frame count of 64
          }
          if(frame==3)
          {
   // Configure channel 1
   *(DMSA) = 5;
   *(DMSDI) = (unsigned short) dmac_ctrl+32;// DMSRC1 - ch 1 source address register
   *(DMSDI) = (unsigned short) MCBSP0_DXR1; // DMDST1 - ch 1 destination address register
   *(DMSDI) = 0x07;                          // DMCTR1 - ch 1 element count register
   *(DMSDI) = 0x2000; // DMSFC1 - ch 1 sync on McBSP0 transmit event and frame count of 64
          }
          if(frame==4)
          {
```

```
    // Configure channel 1
    *(DMSA) = 5;
    *(DMSDI) = (unsigned short) dmac_ch1A;  // DMSRC1 - ch 1 source address register
    *(DMSDI) = (unsigned short) MCBSP0_DXR1;// DMDST1 - ch 1 destination address register
    *(DMSDI) = 0x07;                        // DMCTR1 - ch 1 element count register
    *(DMSDI) = 0x2000; // DMSFC1 - ch 1 sync on McBSP0 transmit event and frame count of 64
    *(DMSDI) = 0x4141; // DMMCR1 - ch 1 transfer mode control register
    mode=1;
          }
          frame++;
          DMAC_ch_enable(DMAC_CH_1);
     }
}
```

A.6 DMAC.H

```
/********************************************************************************
*
* FILE: Dmac.h
*
* DESCRIPTION:    Header file for DMA
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
*******************************************************************************/
#ifndef _DMAC_H_
#define _DMAC_H_
/* DMA Registers  */
#define DMPREC    (volatile unsigned short *) 0x0054 /* Channel Priority and Enable Control
Register */
#define DMSA      (volatile unsigned short *) 0x0055 /* Sub-bank Address Register */
#define DMSDI     (volatile unsigned short *) 0x0056 /* Sub-bank Data Register with
autoincrement */
#define DMSDN     (volatile unsigned short *) 0x0057 /* Sub-bank Data Register without
modification */
#define DMSRC0    0x00    /* Channel 0 Source Address Register */
#define DMDST0    0x01    /* Channel 0 Destination Address Register */
#define DMCTR0    0x02    /* Channel 0 Element Count Register */
#define DMSFC0    0x03    /* Channel 0 Sync Select and Frame Count Register */
#define DMMCR0    0x04    /* Channel 0 Transfer Mode Control Register */
#define DMSRC1    0x05    /* Channel 1 Source Address Register */
#define DMDST1    0x06    /* Channel 1 Destination Address Register */
#define DMCTR1    0x07    /* Channel 1 Element Count Register */
```

```
#define DMSFC1    0x08    /* Channel 1 Sync Select and Frame Count Register */
#define DMMCR1    0x09    /* Channel 1 Transfer Mode Control Register */
#define DMSRC2    0x0A    /* Channel 2 Source Address Register */
#define DMDST2    0x0B    /* Channel 2 Destination Address Register */
#define DMCTR2    0x0C    /* Channel 2 Element Count Register */
#define DMSFC2    0x0D    /* Channel 2 Sync Select and Frame Count Register */
#define DMMCR2    0x0E    /* Channel 2 Transfer Mode Control Register */
#define DMSRC3    0x0F    /* Channel 3 Source Address Register */
#define DMDST3    0x10    /* Channel 3 Destination Address Register */
#define DMCTR3    0x11    /* Channel 3 Element Count Register */
#define DMSFC3    0x12    /* Channel 3 Sync Select and Frame Count Register */
#define DMMCR3    0x13    /* Channel 3 Transfer Mode Control Register */
#define DMSRC4    0x14    /* Channel 4 Source Address Register */
#define DMDST4    0x15    /* Channel 4 Destination Address Register */
#define DMCTR4    0x16    /* Channel 4 Element Count Register */
#define DMSFC4    0x17    /* Channel 4 Sync Select and Frame Count Register */
#define DMMCR4    0x18    /* Channel 4 Transfer Mode Control Register */
#define DMSRC5    0x19    /* Channel 5 Source Address Register */
#define DMDST5    0x1A    /* Channel 5 Destination Address Register */
#define DMCTR5    0x1B    /* Channel 5 Element Count Register */
#define DMSFC5    0x1C    /* Channel 5 Sync Select and Frame Count Register */
#define DMMCR5    0x1D    /* Channel 5 Transfer Mode Control Register */
#define DMSRCP    0x1E    /* Source Program Page Address */
#define DMDSTP    0x1F    /* Destination Program Page Address */
#define DMIDX0    0x20    /* Element Address Index Register 0 */
#define DMIDX1    0x21    /* Element Address Index Register 1 */
#define DMFRI0    0x22    /* Frame Address Index Register 0 */
#define DMFRI1    0x23    /* Frame Address Index Register 1 */
#define DMGSA     0x24    /* Global Source Address Reload Register */
#define DMGDA     0x25    /* Global Destination Address Reload Register */
#define DMGCR     0x26    /* Global Element Count Reload Register */
#define DMGFR     0x27    /* Global Frame Count Reload Register */
// DMAC channel ID
#define   DMAC_CH_0   0x0001
#define   DMAC_CH_1   0x0002
#define   DMAC_CH_2   0x0004
#define   DMAC_CH_3   0x0008
#define   DMAC_CH_4   0x0010
#define   DMAC_CH_5   0x0020
#define DMAC_INT  0x0080
// DMAC priority ID
#define DMAC_PR_0     0x0100
#define DMAC_PR_1     0x0200
```

```
#define   DMAC_PR_2   0x0400
#define   DMAC_PR_3   0x0800
#define   DMAC_PR_4   0x1000
#define DMAC_PR_5  0x2000
// DMAC enable ID
#define DMAC0     0x0040
#define DMAC1     0x0080
#define DMAC2     0x0400
#define DMAC3     0x0800
#define DMAC4     0x1000
#define DMAC5     0x2000
void DMAC_init();
void DMAC_ch_enable(int ch_num);
void DMAC_ch_disable(int ch_num);
// prototype for ISRs
interrupt void DMAC0_isr();
interrupt void DMAC1_isr();
#endif
```

A.7 MCBSP.C

```
/*******************************************************************************
*
* FILE: McBSP5410.c
*
* DESCRIPTION:    Contains the initializations functions for the mcbsp of the TMS320C5410
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
*******************************************************************************/
#include ”mcbsp5410.h”
extern void sw_wait(unsigned int);
volatile unsigned int *SerPortSubAddPtr = (unsigned int *) 0x0038;
volatile unsigned int *SerPortSubAddDataPtr = (unsigned int *) 0x0039;
volatile unsigned int *SerPort1SubAddPtr = (unsigned int *) 0x0048;
volatile unsigned int *SerPort1SubAddDataPtr = (unsigned int *) 0x0049;
volatile unsigned int *SerPort2SubAddPtr = (unsigned int *) 0x0034;
volatile unsigned int *SerPort2SubAddDataPtr = (unsigned int *) 0x0035;
/*-- Serial Port 0 Configuration Functions ---------------------------------*/
void SerialPort0Init( void )
{
    write_mcbsp0_subreg( SPCR1_sa, I_SPCR10);
    write_mcbsp0_subreg( SPCR2_sa, I_SPCR20);
    write_mcbsp0_subreg( RCR1_sa,  I_RCR10);
    write_mcbsp0_subreg( RCR2_sa,  I_RCR20);
    write_mcbsp0_subreg( XCR1_sa,  I_XCR10);
    write_mcbsp0_subreg( XCR2_sa,  I_XCR20);
    write_mcbsp0_subreg( PCR_sa,  I_PCR0);
}
/*-- Re-set Serial Port 0 Configuration Functions---------------------------------*/
void SerialPort0_TXRX_Enable( void )
{
    unsigned int temp;

    /* Enable Receiver */
    read_mcbsp0_subreg( SPCR1_sa, &temp);
    temp |= RRST_s;
    write_mcbsp0_subreg( SPCR1_sa, temp);
    /* Enable Transmitter */
    read_mcbsp0_subreg( SPCR2_sa, &temp);
    temp |= XRST_s;
    write_mcbsp0_subreg( SPCR2_sa, temp);
    sw_wait(100);
}
void write_mcbsp0_subreg( unsigned int reg_add, unsigned int reg_val )
{
    *SerPortSubAddPtr = reg_add;
    *SerPortSubAddDataPtr = reg_val;
}
void read_mcbsp0_subreg( unsigned int reg_add, unsigned int *reg_val )
{
    *SerPortSubAddPtr = reg_add;
    *reg_val = *SerPortSubAddDataPtr;
}

void write_mcbsp1_subreg( unsigned int reg_add, unsigned int reg_val )
{
    *SerPort1SubAddPtr = reg_add;
    *SerPort1SubAddDataPtr = reg_val;
}
void read_mcbsp1_subreg( unsigned int reg_add, unsigned int *reg_val )
{
    *SerPort1SubAddPtr = reg_add;
    *reg_val = *SerPort1SubAddDataPtr;
}
```

```
void SerialPort0Reset(void)
{
    write_mcbsp0_subreg( SPCR1_sa, 0x0000);
    write_mcbsp0_subreg( SPCR2_sa, 0x0000);

}
```

A.8 MCBSP.H

```
/********************************************************************************
*
* FILE: McBSP5410.h
*
* DESCRIPTION:    Header file for mcbsp of the TMS320C5410
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
******************************************************************************/
#ifndef mcbsp5410_h
#define mcbsp5410_h
/*-- mcbsp Registers--------------------------------------------------------*/
#define SPCR1_sa      0x0000
#define SPCR2_sa      0x0001
#define RCR1_sa       0x0002
#define RCR2_sa       0x0003
#define XCR1_sa       0x0004
#define XCR2_sa       0x0005
#define SRGR1_sa      0x0006
#define SRGR2_sa      0x0007
#define MCR1_sa       0x0008
#define MCR2_sa       0x0009
#define RCERA_sa      0x000A
#define RCERB_sa      0x000B
#define XCERA_sa      0x000C
#define XCERB_sa      0x000D
#define PCR_sa        0x000E
/*SPCR1*/
#define CLKSTP_s      0x1000 /*clock starts w/o delay*/
#define RINTM_s       0x0020 /*int by new frame sync*/
#define RRST_s        0x0001 /*receuver enable*/
```

```
/*SPCR2*/
#define FREE_s           0x0200 /*serial port clock free run*/
#define FRST_s           0x0080 /*frame sync generator enable*/
#define GRST_s           0x0040 /*sample rate generator enable*/
#define XRST_s           0x0001 /*transmitter enable*/
#define XINTM_s          0x0020 /*int by new frame sync*/
/*RCR1*/
#define RWDLEN1_s8       0x0000 /*8 bit*/
#define RWDLEN1_s16      0x0040 /*16 bit*/
#define RWDLEN1_s20      0x0060 /*20 bit*/
#define RWDLEN1_s24      0x0080 /*24 bit*/
#define RWDLEN1_s32      0x00A0 /*32 bit*/
#define  RFRLEN1_s2  0x0100  /*2  words  per  frame*/
#define  RFRLEN1_s4  0x0300  /*2  words  per  frame*/
#define RFRLEN1_s8 0x0700 /*8 words per frame*/
/*RCR2*/
#define RDATDLY_s        0x0001 /*1 bit delay*/
#define RFIG_s           0x0004 /*ignore frame syncs after first*/
/*XCR1*/
#define XWDLEN1_s8       0x0000 /*8 bit*/
#define XWDLEN1_s16      0x0040 /*16 bit*/
#define XWDLEN1_s24      0x0080 /*24 bit*/
#define XWDLEN1_s32      0x00A0 /*32 bit*/
#define XFRLEN1_s2       0x0100 /*2 words per frame*/
#define XFRLEN1_s4       0x0300 /*4 words per frame*/
#define XFRLEN1_s8       0x0700 /*8 words per frame*/
/*XCR2*/
#define XDATDLY_s        0x0001 /*1 bit delay*/
#define XFIG_s           0x0004 /*ignore frame syncs after first*/
/*PCR*/
#define XIOEN_s          0x2000
#define FSXM_s           0x0800 /*0 input, 1 output*/
#define FSRM_s           0x0400 /*0 input, 1 output*/
#define CLKXM_s          0x0200 /*0 input, 1 output*/
#define CLKRM_s          0x0100 /*0 input, 1 output*/
#define DX_STAT_s        0x0020
#define FSXP_s           0x0008 /*0 active high, 1 active low*/
#define FSRP_s           0x0004 /*0 active high, 1 active low*/
#define CLKXP_s          0x0002 /*0 rising edge, 1 falling edge*/
#define CLKRP_s          0x0001 /*0 falling edge, 1 rising edge*/
/*SRGR1*/
#define CLKGDV_s         0x0018 /*Internal Transmit Clock Division factor*/
#define FWID_s           0x0100 /*Frame Width = FWID + 1*/
```

```
/*SRGR2*/
#define CLKSM_s         0x2000 /*SRG clock from CPU*/
#define FSGM_s          0x1000 /*Frame Sync driven by SRG Frame sync (FSG)*/
#define FPER_s          0x0006 /*Frame Period = FPER + 1*/
/*Serial Port 0 Configurations*/
#define I_SPCR10        0x0000
#define I_SPCR20        FREE_s
#define  I_RCR10   RWDLEN1_s16   |   RFRLEN1_s8
#define  I_RCR104  RWDLEN1_s16   |   RFRLEN1_s4
#define I_RCR20 0x0000 | RFIG_s
#define  I_XCR10   XWDLEN1_s16   |   XFRLEN1_s8
#define  I_XCR104  XWDLEN1_s16   |   XFRLEN1_s4
#define I_XCR20 0x0000 | XFIG_s
#define I_PCR0          FSXP_s | FSRP_s
#define MCBSP0_DRR2 (volatile unsigned short *) 0x20
#define MCBSP0_DRR1 (volatile unsigned short *) 0x21
#define MCBSP0_DXR2 (volatile unsigned short *) 0x22
#define MCBSP0_DXR1 (volatile unsigned short *) 0x23
/*---- global data declarations --------------------------------------------*/
/*---- global function prototypes------------------------------------------*/
void SerialPort0Init( void );
void SerialPort0_TXRX_Enable( void );
void write_mcbsp0_subreg( unsigned int reg_add, unsigned int reg_val );
void write_mcbsp1_subreg( unsigned int reg_add, unsigned int reg_val );
void read_mcbsp0_subreg( unsigned int reg_add, unsigned int *reg_val );
void read_mcbsp1_subreg( unsigned int reg_add, unsigned int *reg_val );
void SerialPort0Reset(void);
#endif /* mcbsp5410_h ------ END OF FILE -------------------------------------*/
```

A.9 EVM5410.C

```
/******************************************************************************
*
* FILE: EVM5410.C
*
* DESCRIPTION: C functions for TMS320C5410 initialisation
*
* AUTHOR: HONG–SWEE LIM – Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
*****************************************************************************/
#include ”evm5410.h”
```

```
void init_core(void)
{
    *CLKMD = 0x0000;
    *CLKMD = 0x90b2;
}
```

A.10 EVM5410.H

```
/*******************************************************************************
*
* FILE: EVM5410.h
*
* DESCRIPTION:    Header file for TMS320C5410
*
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments
*
* LAST MODIFIED:03/20/2003
*
*******************************************************************************/
#ifndef evm5410_h
#define evm5410_h
/*-- Interrupt macros-------------------------------------------------*/
#define GLOBAL_INT_ENABLE asm( ” rsbx intm ”) /* enables interrupts*/
#define GLOBAL_INT_DISABLE asm( ” ssbx intm ”) /* disable interrupts*/
#define NOP asm( ” nop ”);
#define CLKMD  (volatile unsigned int *) 0x0058
#define IMR ( (unsigned short *) 0x0000)
#define IFR ( (unsigned short *) 0x0001)
/*----- Protos for assembly functions from evm5410.c    --------------------*/
void init_core(void);
#endif /*  evm5410.h ------ END OF FILE -------------------------------------*/
```

A.11 VECT_C54X.ASM

```
*******************************************************************************
*                                                                             *
* FILE: VECT_C54X.ASM                                                         *
*                                                                             *
* DESCRIPTION:    Reset and Interrupt Vector Table                            *
*                                                                             *
* AUTHOR:         HONG-SWEE LIM  -  Texas Instruments                         *
*                                                                             *
* LAST MODIFIED:03/20/2003                                                    *
*                                                                             *
*******************************************************************************
```

```
      .length 58
      .mmregs

      ;all interrupts used in the program must be referenced
      .ref     _c_int00
      .ref     _DMAC0_isr
      .ref     _DMAC1_isr
      .global  _xboot


****************************************************************************
*   INTERRUPT VECTORS                                                      *
*   Make sure that every interrupt is defined in the .sect ”myvectors”
*   section. These are all the standard interrupts for the C54X. Also, you
*   need to referrence the ones that you use in your code in the .ref part.
*
*   NOTE: There are 4 words per instructions per interrupt.
*   The B (branch) statements takes 2, while NOP are 1 each
*   Maps a name (on the left) to a interrupt (on the right).
****************************************************************************
      .sect ”myvectors”
VECT_PMST:
_reset: B     _xboot
        NOP
        NOP
_nmi:   B     _nmi
        NOP
        NOP
_sint17:B     _sint17
        NOP
        NOP
_sint18:B     _sint18
        NOP
        NOP
_sint19:B     _sint19
        NOP
        NOP
_sint20:B     _sint20
        NOP
        NOP
_sint21:B      _sint21
        NOP
        NOP
```

```
_sint22:B      _sint22
        NOP
        NOP
_sint23:B      _sint23
        NOP
        NOP
_sint24:B      _sint24
        NOP
        NOP
_sint25:B      _sint25
        NOP
        NOP
_sint26:B      _sint26
        NOP
        NOP
_sint27:B      _sint27
        NOP
        NOP
_sint28 B      _sint28
        NOP
        NOP
_sint29 B      _sint29
        NOP
        NOP
_sint30 B      _sint30
        NOP
        NOP
_int0   B      _int0
        NOP
        NOP
_int1   B      _int1
        NOP
        NOP
_int2   B      _int2
        NOP
        NOP
;the commented section is for extended memory
;        .if  EXTENDED_MEMORY=1
;_tint  PSHM XPC
;   nop
;   FB  _c_int19   ; timer interrupt
;_rint0  PSHM XPC
;   nop
;   FB  _c_int20  ; serial port receive interrupt
;        .else
_tint    B     _tint
         NOP
         NOP   ; timer interrupt
_rint0   B     _rint0
         NOP
         NOP  ; serial port receive interrupt
;        .endif
_xint0  B     _xint0
        NOP
        NOP
_rint2  B     _DMAC0_isr
        NOP
        NOP
_xint2  B     _DMAC1_isr
        NOP
        NOP
_int3   B     _int3
        NOP
        NOP
_hint   B     _hint
        NOP
        NOP
_brint1 B     _brint1
        NOP
        NOP
_bxint1 B     _bxint1
        NOP
        NOP
              .text
; this sets the PMST up, and makes sures that interrupts are disabled before
; resetting the chip
_xboot:                            ; b       $ for debug
              ld      #0,dp
              stm     #0,sp
              rsbx    sxm          ; Disable sign extension
              ssbx    intm         ; Disable interrupts
              stm     #0,imr
              stm     #0ffffh,ifr  ; Clear ifr
              ld      #VECT_PMST,a ; Set PMST for vectors
              or      #0060h,a
              stlm    a,pmst       ;
              rpt     #4
              nop
              b       _c_int00

              .end
```

A.12 WAIT.ASM

```
*****************************************************************************
*                                                                           *
* FILE: WAIT.ASM                                                            *
*                                                                           *
* DESCRIPTION:  C-callable function. Maximum delay is 65535 cycles, which   *
*   equates to 655us on a 100MHz device                                     *
* AUTHOR:       HONG-SWEE LIM  -  Texas Instruments                         *
*                                                                           *
* LAST MODIFIED:03/20/2003                                                  *
*                                                                           *
*****************************************************************************
    .def    _sw_wait
    .text
_sw_wait:
    STM     #0008h, AR0
    RPT     *AR0
    NOP
    RET
```