Application Report
SLAA118 – January 2001

Interfacing the ADS8320 to the TMS320C5402 DSP

Lijoy Philipose *Data Acquisition Applications*

ABSTRACT

This report presents a method for interfacing the ADS8320 16-bit SAR analog-to-digital converter to the TMS320C5402 DSP. In an effort to reduce development time, the source code written for this interface is provided in the appendixes. Project collateral discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/SLAA118.

Contents

Figures

Tables

1 Introduction

The ADS8320 is a 16-bit SAR analog-to-digital converter (ADC). It requires very little power when operating at the full 100-kHz sampling rate. This data converter operates over a supply voltage range of 2.7 V to 5.25 V. At a data rate of 10 kHz, the average power dissipation is less than 100µW. It features a synchronous serial (SPI/SSI compatible) interface and a differential input.

The TMS320C5402 Digital Signal Processor (‘C5402™ DSP) is able to interface gluelessly to the ADS8320. The two multi-channel buffered serial ports (McBSP) on-chip can be programmed for a variety of serial transfer protocols, one of which is SPI. For evaluation and system development purposes, Texas Instruments provides the TMS320C5402 DSP starter kit (DSK). This kit gives new DSP designers an affordable medium to access the industry’s most powerful DSP. This DSK was used to interface the ADS8320 to ‘C5402 DSP. The source code developed for this interface, written in ‘C54xx algebraic assembly language, is presented in the appendices. A soft copy of this software can be downloaded from TI’s web site.

2 Hardware

The TMS320C5402 DSK is an easy way to interface the TMS320C5402 DSP to the ADS8320.

2.1 TMS320C5402 DSK

The ’C5402 DSK is designed specifically for digital communications applications and comes complete with a TMS320C5402-based target board, DSK-specific code composer studio debug tools, 32K application-size-limited C compiler/assembler/linker, parallel port interface, power supply, and cables. The ’C5402 device features a 100-MHz clock, a 40-bit ALU, 16K x 16-bit dual access on-chip RAM, 4K x 16-bit on-chip ROM, advanced multibus architecture with three separate 16-bit data memory busses, and one program memory bus. The onboard parallel port controller allows the host PC to use the parallel port for emulation, or for direct access to the host port interface of the ’C5402. The ’C5402 also provides an onboard standard JTAG interface connection for optional emulation and expansion connectors for add-on accessories. Texas Instruments now provides expansion connectors or adapter boards for all data converter EVMs to interface to the ‘C5402 DSK.

The enhanced peripheral of particular interest in this report is the McBSP serial port. The McBSP is explained in Section 3. The ‘C5402 DSK supports a TMS320VC5402 DSP, which can operate at a frequency of up to 100 MHz with a core voltage of 1.8 V and an I/O voltage of 3.3 V. The DSK provides support for all the DSP interfaces and control signals. The JTAG emulation interface is used to support both embedded and external JTAG emulations. The control interface is used to reset the device and to provide external interrupts. The McBSP0 is, by default, used to interface to the telephone DAA circuit. This port is also available to the daughter board via an onboard multiplexer. The McBSP1 is, by default, used for the microphone/speaker interfaces brought to the peripheral expansion connector for daughter board use. The complex programmable-logic device (CPLD) controls the input of McBSP0 and McBSP1. Its register bit decides which devices are routed to the McBSP.

’C54xx is a trademark of Texas Instruments, Inc.

2.1.1 CPLD

In this application, McBSP1 is used to interface to the ADS8320. There are seven CPLD registers mapped starting at I/O space 0x0000. A complete list of CPLD registers and their descriptions are provided in the code composer studio help files. The CPLD control register CNTL2, located in I/O space 0x4, must be re-configured to allow the input to McBSP1 to arrive from the daughter card. This is achieved by the following code instruction:

```
port(#04) = #0002h          ;access i/o space 0x4 and write 0x0002 to the CPLD CNTL2
                            ;register.
```

2.2 ADS8320

The ADS8320 is a classic successive approximation register (SAR) analog-to-digital converter (ADC). The architecture is based on capacitive redistribution which inherently includes a sample/hold function. This 16-bit ADC is able to acquire and convert an analog signal at up to 100K conversions per second while consuming less than 4.5 mW from $+V_{CC}$.

The ADS8320 requires an external reference, an external clock, and a single power source (V_{CC}). The external reference can be any voltage between 500 mV and V_{CC}. The value of the reference voltage sets the range of the analog input. The external clock can vary from 24 kHz up to 2.4 MHz, which allows the throughput to be varied from 1 kHz up to 100 kHz. The clock's duty cycle is unimportant as long as the high and low times are at least 200 ns each (V_{CC} = 2.7 V or greater). The minimum clock frequency is set by the leakage of the capacitors internal to the ADS8320.

The analog input is provided to two input pins: +IN and –IN. When conversion is initiated, the differential inputs to these pins are sampled on the internal capacitor array. While a conversion is in progress, both inputs are disconnected from any internal function. The +IN and –IN input pins allow for differential input signals but, unlike some converters, the –IN input is not resampled later in the conversion cycle. When the converter goes into the hold mode, the voltage difference between +IN and –IN is captured on the internal capacitor array. The –IN input is limited to the range –0.1 V to +1 V (or -0.1 V to +0.5 V when using a 2.7-V supply). This allows the rejection of only small signals common to both inputs.

The digital result of the conversion is clocked out by the DCLOCK input and is provided serially, most significant bit first (MSB), on the DOUT pin. The digital data provided on the DOUT pin is from the conversion currently in progress—there is no pipeline delay.

More information on the ADS8320 can be found in *16-Bit, High-Speed, 2.7-V to 5-V microPower Sampling Analog-to-Digital Converter*, Literature Number SBAS108 [1].

2.3 Hardware Interface

The hardware interface is virtually seamless between the 'C5402 DSK and the ADS8320. The C5402 DSP provides two on-chip McBSPs, the second McBSP was used for this report. The hardware connections are shown in Figure 1. The DCLOCK pin from the ADS8320 is connected to the CLKX pin of McBSP1 (CLKX1). The chip-select (CS) pin connects to the transmit frame-sync pin, FSX1, of McBSP1. When a McBSP is programmed to operated in SPI mode, FSX1 behaves like a slave-select pin. Finally, the DOUT pin is connected to the data receive pin, DRR1, of McBSP1.

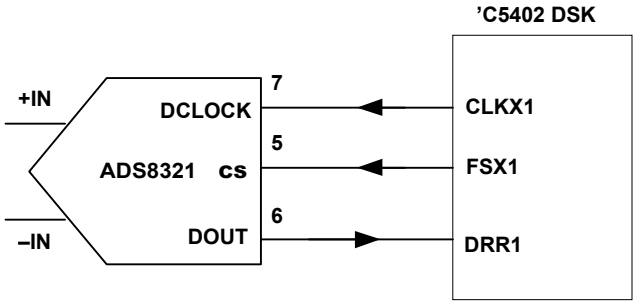


Figure 1. Hardware Interface Block Diagram

3 Serial Interface

To accomplish this interface, the McBSP and ADS8320 devices must be understood.

3.1 McBSP

The McBSP is based on the standard serial-port interface found on earlier DSPs from Texas Instruments. The McBSP is a high-speed, full-duplex, multichannel, buffered, serial port that allows direct interfacing to 'C54xx devices, codecs, data converters, and other devices in a system. In addition, the McBSP offers double-buffered registers that allow continuous data stream, and independent framing and clocking for receiver and transmitter. This flexible device gives unprecedented control over the serial interface. To use the McBSP successfully, familiarity with the sample-rate generator, the transmitter, and the receiver are required. The sample-rate generator consists of a three-stage clock divider that permits programmable data clocks (CLKG) and framing signals (FSG); these are McBSP internal signals that can be programmed to drive, receive, and/or transmit clocking (CLKR/X) and framing (FSR/X). The sample-rate generator can be driven by an internal-clock source or by an internal clock derived from an external-clock source.

The sample-rate generator registers (SRGR[1,2]) control the operation of the various functions of the sample-rate generator. These registers are used (i) to control the width of the frame-sync pulse and (ii) to decide whether the frame-sync is an external input driven by the sample rate generator or it is flag that data have been copied from DXR[1,2] to XSR[1,2]. These registers control whether the sample-rate generator clock is derived from the CPU clock or from the CLKS pin, and determine the division factor necessary to produce the desired serial clock (CLKX/R). The transmitter and receiver on the McBSP can be programmed for multiple frame-lengths and word-lengths. The transmit-control registers (XCR[1,2]) and the receive-control registers (RCR[1,2]) determine the mode of the transmitter and receiver. Frame length is the number of serial words transferred per frame. A serial-word can be 8-bits, 12-bits, 16-bits, 20-bits, 24-bits, or 32-bits long. Table 1 shows a map of all the McBSP0 registers used in this application report. The 'C5402 DSP contains two McBSP ports, namely McBSP0 and McBSP1.

Table 1. McBSP1 Register Settings

McBSP1 ADDRESS	McBSP1 SUB-ADDRESS	ACRONYM	REGISTER INITIALIZED VALUE	COMMENT
0041		DRR11		Receiver data register
0043		DXR11		Transmit data register
0048		SPSA1		McBSP1 subaddressing register
0049	0x0000	SPCR11	0x1801	Clock starts with rising edge with delay. While configuring McBSP, the LSB bit must be 0, i.e., transmitter is disabled.
	0x0001	SPCR21	0x02C1	While configuring McBSP, the LSB bit must be 0 (0x02C0) so that the receiver is disabled.
	0x0002	RCR11	0x0080	Selects one 24-bit word transfer per frame
	0x0003	RCR21	0x0000	
	0x0004	XCR11	0x0080	Selects one 24-bit word transfer per frame
	0x0005	XCR21	0x0000	
	0x0006	SRGR11	0x0084	CLKX = CPU clock / (CLKGDV +1), CLKGDV = SRGR1.[7,0].
	0x0007	SRGR21	0x2000	The sample-rate-generator clock is derived from the CPU clock.
	0x000E	PCR1	0x0A0D	FSX is determined from the sample-rate-generator frame-synchronization-mode bit SRGR2[FSGM]. CLKX output is driven by the sample rate generator. Receive data on rising edge of CLKR. FSX and FSR are active-low signals.

3.1.1 Programming the McBSP

The following sections demonstrate the process used to configure the McBSP for this application. Modifying a McBSP0 register requires writing its subaddress to the McBSP0 subaddress register (SPSA), as shown in the following code fragment:

```
MMR(#0038h) = #0000h   ;Load subaddress of SPCR1 in subaddress register
 ;in McBSP0
MMR(#0039h)=#0801h     ;Load value in SPCR1
```

The subaddress register for McBSP0 is located at address 0x0038 in program memory. The user must then write the control word for that register to address 0x0039 in program memory. With the receiver and transmitter in reset, the McBSP registers can be filled with the desired values.

Figure 2 is a general flowchart for setting up the McBSP.

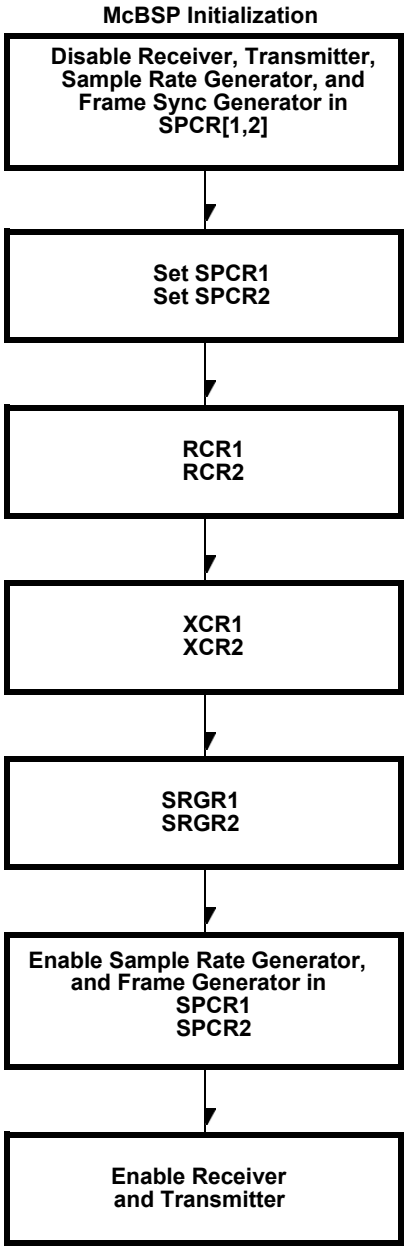
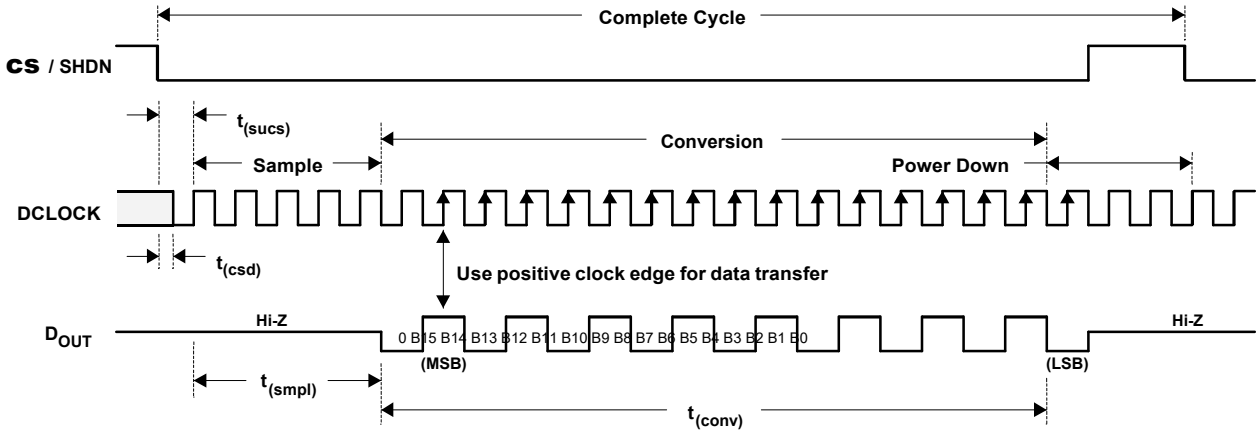


Figure 2. McBSP Programming

3.2 Digital Interface

The ADS8320 communicates with microprocessors and digital signal processors via a synchronous 3-wire serial interface, as shown in Figure 3. The DCLOCK signal synchronizes the data transfer with each bit being transmitted on the falling edge of DLOCK. The receiving device

can capture the bit-stream on the rising edge. A falling edge on CS initiates the conversion and data transfer. The first 4.5 to 5.0 clock periods of the conversion cycle are used to sample the input signal. After the fifth falling DCLOCK edge, DOUT is enabled and outputs a low value for one clock period. For the next 16 clocks periods, DOUT outputs the conversion result, most-significant-bit first. If DCLOCK persists, the device outputs data in least-significant-bit-first format. After the most significant bit (B15) has been repeated, DOUT goes into a high-impedance state. Subsequent clocks have no effect on the converter. A new conversion is initiated only when CS is take from high to low.



NOTE: Minimum 22 clock cycles required for 16-bit conversion. Shown are 24 clock cycles.
If **CS** remains LOW at the end of conversion, a new datastream with LSB-first is shifted out again.

Figure 3. ADS8320 Timing Diagram

4 Interface Software

The interface software for ADS8320 consists of setting up the DSP/McBSP and initiating new conversion cycles on the ADC. Figure 4 is the software flowchart for the program listing in Appendices A to D. The DSP portion consists of initializing the stack register, resetting the interrupt flag register (IFR), setting the interrupt flag register (IMR) and interrupt vector table pointer, and so on. The interrupt service routine (ISR) is used to read the samples read in through the McBSP1 receiver. The ISR re-formats the sample and stores that value into data memory. Once all the desired samples have been read, the IMR register is reset, disabling the ISR. The McBSP1 registers are initialized for the correct clock-stop mode, transfer word size, and the clock edge on which to read in data, see Table 1 which summarizes the McBSP register settings.

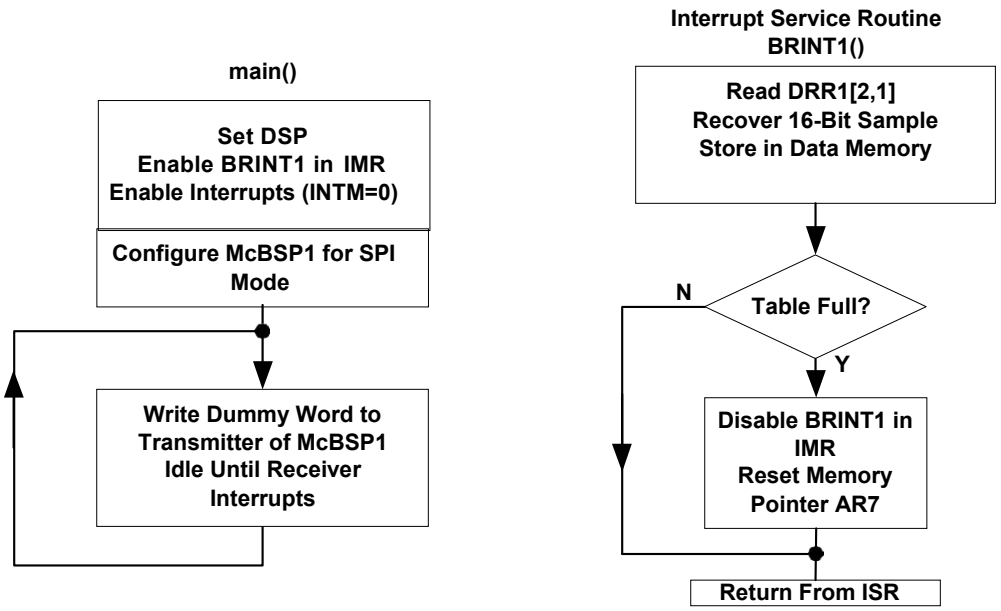


Figure 4. Software Flowchart

5 Reference

1. *16-Bit, High-Speed, 2.7-V to 5-V microPower Sampling Analog-to-Digital Converter*, Data Sheet, Texas Instruments Literature Number SBAS108

Appendix A – ADS8320.ASM

```
*************************************************************************
* TITLE       : ADS8320 Interface routine to c5402 DSK                  *
* FILE        : ADS8320.ASM                                             *
* FUNCTION    : MAIN                                                    *
* PROTOTYPE   : void MAIN ()                                            *
* CALLS       : N/A                                                     *
* PRECONDITION : N/A                                                    *
* POSTCONDITION : N/A                                                   *
* DESCRIPTION  : Program initializes DSP, McBSP and initiates conversions *
*            cycle.  Sits idle until interrupt arrives from McBSP1 Receiver.  Stores data *
*            in data space memory beginning at 0x3000.                  *
* AUTHOR        :  Data Acquisition Application Group, L. Philipose, Dallas, Tx *
*              CREATED 2000(C) BY TEXAS INSTRUMENTS INCORPORATED.       *
* REFERENCE   : TMS320C54x User's Guide, TI 1999                        *
* Note              :  N/A                                              *
*************************************************************************


      .title "ADS8320 ADC"
      .mmregs
      .width  80
      .length  55

   .sect ".vectors"
   .copy "C5402vec.asm"
   .sect ".data"
   .includeregs.h

AD_DP          .usect ".varibl", 0          ; AD data page for local variables.
ADCWORD        .usect ".varibl", 1          ; Control word for ADC
CLKGDV1        .usect ".varibl", 1          ; McBSP clock divide by factor
Temp           .usect ".varibl", 1          ; Control word for the ADC

   .sect ".text"

_main:

***Initialize DSP***

      DP= #0                  ; Set DP to Data Page Zero
      SP= #3CFFh              ; Set stack pointer register
      INTM=#1                 ; Disable global interrupts
      SWWSR= #2000h;          ; Need 2 wait-states for accessing I/O space
      PMST=#00F8h;            ; Re-map interrupt vector TABLE to 0x0080h
      IMR=#0400h;             ; Enable interrupt McBSP1 Receiver interrupt(BRINT1)
      IFR=#0FFFFh;            ; Clear IFR

      DP=#AD_DP
*Initialize CPLD*
      @Temp=#0002h            ; Write to I/O space (i.e. CPLD register CNTL2)
```

```
      port(DSP_CPLD_CNTL2) =@Temp        ; McBSP1 Select Control=Daughter board in CNTL2 of CPLD

*Initialize McBSP*
      @CLKGDV1=#0052h            ; CLKX1 = DCLOCK = CPU CLOCK / (CLKGDV + 1).
                                 ; CPU CLOCK=100MHz, CLKGDV=0x0052, CLKX=DCLOCK=1.2MHz.

      call McBSP1_init           ; Initialize McBSP1

      INTM=#0                    ; Enable global interrupts

      AR7=#3000h                 ; Store digital samples starting at 0x3000
      AR6=#3200h                 ; End of data table, collect 512 samples

**initiate conversion**
NewCycle:

      A=#0000h                   ; Dummy word used to generate Slave Select pulse
      AR2=#McBSP1_DXR1
      nop
      *AR2 = A                   ; Write to DXR11, McBSP1 Transmitter

      idle(1)                    ; DSP idle until interrupt occurs
      nop
      goto NewCycle

************************************************************
*Function : isr_brint1                                     *
*Description: Interrupt service routine- McBSP1 Receiver   *
*   Captures and stores 16-bit digital code from the 24-bit word *
*   Stores samples at location pointed to by AR7           *
*      End of data table is pointed to be AR6              *
************************************************************

isr_brint1:

      AR0=AR6
      AR1=#McBSP1_DRR2    ; Read out DRR2, McBSP1 receive data register 2
      NOP
      A=*AR1
      AR2=#McBSP1_DRR1    ; Read out DRR1, McBSP1 receive data register 1
      NOP
      B=*AR2
*Recover 16-bit sample*
      A=A<<14             ; left shift MSB bits
      B=B & #0FFFCh       ; clear bit0 and bit1
      B=B>>2              ; right shift by two
      A=B | A             ; 16-bit digital sample
      @ADCWORD=A          ; save

**Store in data table**
      TC     = (AR0 == AR7)              ; Read all samples?
      if (TC) goto Complete
      *AR7+     = data(@ADCWORD)         ; Store in data memory and increment pointer
      goto isr_Return
```

```
Complete:                          ; All samples read

*comment this next line if you want DSP to continuously read and store the data*
;    IMR =#0000h                   ; Disable interrupt service routine
     AR7=#3000h                    ; Reset data table pointer register

isr_Return:
   return_enable
************************************************************
*Function : McBSP1_init                                     *
*Description: Initializes McBSP1 for 24-BIT transfers and clock-stop     *
*       mode, sample-rate generator clock derived from CPU clock,        *
*      and sets serial clock bases on value stored in CLKGDV1 .          *
************************************************************

McBSP1_init:


       mmr(McBSP1_SPSA) = #SPCR1     ; Reset McBSP1 Receiver
       A=mmr(McBSP1_SPSD)
       A = #0FFFEh & A               ; RRST*=0
       mmr(McBSP1_SPSD) =A

       mmr(McBSP1_SPSA) = #SPCR2      ; Reset McBSP1 Receiver
       A=mmr(McBSP1_SPSD)
       A= A & #0FF2Eh                ;GRST*=0,FRST*=0,XRST*=0
       mmr(McBSP1_SPSD) = A

       mmr(McBSP1_SPSA) = #RCR1      ; Receive
       mmr(McBSP1_SPSD) = #0080h     ; One 24-bit Word per frame


       mmr(McBSP1_SPSA) = #RCR2
       mmr(McBSP1_SPSD) = #0000h


       mmr(McBSP1_SPSA) = #XCR1      ; transmit
       mmr(McBSP1_SPSD) = #0080h     ; One 24-bit Word per frame


       mmr(McBSP1_SPSA) = #XCR2
       mmr(McBSP1_SPSD) = #0000h

       mmr(McBSP1_SPSA) = #SRGR1
       A=@CLKGDV1
       mmr(McBSP1_SPSD) = A          ; CLKX=(CPU Clock /1+CLKGDV)

       mmr(McBSP1_SPSA) = #SRGR2     ; SRG clock derived from CPU clock
       mmr(McBSP1_SPSD) = #2000h     ; FSX due to transfer DXR->XSR

       mmr(McBSP1_SPSA) = #PCR       ; FSXM & CLKXM output pin driven
       mmr(McBSP1_SPSD) = #0A0Dh     ; Receive on rising edge of CLKR
                                           ; FSX & FSR are active low
```

```
                                        ; FSX/FSR and CLKX/CLKR tied internally

        mmr(McBSP1_SPSA) = #SPCR1       ; clock mode. clock starts
        A= #1800h                       ; with rising edge with delay
        mmr(McBSP1_SPSD) = A

        A= mmr(McBSP1_SPSD)
        A= A | #0001h
        mmr(McBSP1_SPSD) = A            ; Receiver enabled

        repeat(#6)
        NOP

        mmr(McBSP1_SPSA) = #SPCR2
        A=mmr(McBSP1_SPSD)
        A= #02C0h                       ; GRST*=1, FRST=1 FREE=1

        mmr(McBSP1_SPSD)  = A           ; Frame Generator and
                                        ; sample-rate generator enabled
        A= A | #0001h                   ; XRST*=1
        mmr(McBSP1_SPSD)  = A           ; Transmitter enabled

        repeat(#6)
        NOP

    RETURN


.end
```

Appendix B – C5402vec.asm

```
*************************************************************************
*           FILENAME: 54xVECS.ASM
*           This routine initializes the 54xDSKPLUS vector table.
*************************************************************************
          .title "54xx DSK Vector Table Initialization"
     .global _main,isr_brint1
     .algebraic
        .sect  ".vectors"



RESET:  dgoto  _main          ; RESET vector
          nop
          nop
NMI:   dgoto  NMI             ; Non-maskable Interrupt
          nop
          nop

*************************************************************************
*           S/W Interrupts
*************************************************************************
SINT17    return_enable
          nop
          nop
          nop
SINT18    return_enable
          nop
          nop
          nop
SINT19    return_enable
          nop
          nop
          nop
SINT20    return_enable
          nop
          nop
          nop
SINT21    return_enable
          nop
          nop
          nop
SINT22    return_enable
          nop
          nop
          nop
SINT23    return_enable
          nop
          nop
          nop
SINT24    return_enable
          nop
          nop
```

```
            nop
SINT25   return_enable
            nop
            nop
            nop
SINT26   return_enable
            nop
            nop
            nop
SINT27   return_enable
            nop
            nop
            nop
SINT28   return_enable
            nop
            nop
            nop
SINT29   return_enable
            nop
            nop
            nop
SINT30   return_enable
            nop
            nop
            nop

*************************************************************************
*           Rest of the Interrupts
*************************************************************************

INT0:     return_enable
            nop
            nop
            nop

INT1:     return_enable
            nop
            nop
            nop

INT2:     return_enable
            nop
            nop
            nop

TINT:     return_enable
            nop
            nop
            nop

BRINT0: return_enable
            nop
            nop
            nop
```

```
BXINT0: return_enable
        nop
        nop
        nop

TRINT:  return_enable
        nop
        nop
        nop

TXINT:  return_enable
        nop
        nop
        nop

INT3:   return_enable
        nop
        nop
        nop

HPINT:  return_enable
        nop
        nop
        nop

BRINT1: dgoto isr_brint1
        nop
        nop

BXINT1: return_enable
        nop
        nop
        nop

DMAC4: return_enable
        nop
        nop
        nop

DMAC5: return_enable
        nop
        nop
        nop
```

Appendix C – regs.h

```
*=============================================================================
*      FILENAME: Reg.h
*
*      TMS320VC5402 & 5402 DSK memory mapped reg definition
*
*=============================================================================

*---- include the 54xx register map defined under .mmreg directive ----

                .mmregs

*---------------------- TIMER2 Registers ------------------

TIM1            .set        0030h           ; Timer1
PRD1            .set        0031h           ; Timer1 Period Reg
TCR1            .set        0032h           ; Timer1 Ctrl Reg

*---------------------- McBSP0 Registers -------------------------

McBSP0_DRR2x            .set        0020h           ; McBSP0 Data Rx Reg2
McBSP0_DRR1             .set        0021h           ; McBSP0 Data Rx Reg1
McBSP0_DXR2             .set        0022h           ; McBSP0 Data Tx Reg2
McBSP0_DXR1             .set        0023h           ; McBSP0 Data Tx Reg1
McBSP0_SPSA             .set        0038h           ; McBSP0 Sub Bank Addr Reg
McBSP0_SPSD             .set        0039h           ; McBSP0 Sub Bank Data Reg

*---------------------- McBSP1 Registers -------------------------

McBSP1_DRR2             .set        0040h           ; McBSP1 Data Rx Reg2
McBSP1_DRR1             .set        0041h           ; McBSP1 Data Rx Reg1
McBSP1_DXR2             .set        0042h           ; McBSP1 Data Tx Reg2
McBSP1_DXR1             .set        0043h           ; McBSP1 Data Tx Reg1
McBSP1_SPSA             .set        0048h           ; McBSP1 Sub Bank Addr Reg
McBSP1_SPSD             .set        0049h           ; McBSP1 Sub Bank Data Reg

*------- McBSP0 & McBSP1 Sub-Bank Addressed Registers ----------

SPCR1           .set        0000h   ; McBSP Ser Port Ctrl Reg1
SPCR2           .set        0001h   ; McBSP Ser Port Ctrl Reg2
RCR1            .set        0002h   ; McBSP Rx Ctrl Reg1
RCR2            .set        0003h   ; McBSP Rx Ctrl Reg2
XCR1            .set        0004h   ; McBSP Tx Ctrl Reg1
XCR2            .set        0005h   ; McBSP Tx Ctrl Reg2
SRGR1           .set        0006h   ; McBSP Sample Rate Gen Reg1
SRGR2           .set        0007h   ; McBSP Sample Rate Gen Reg2
MCR1            .set        0008h   ; McBSP Multichan Reg1
MCR2            .set        0009h   ; McBSP Multichan Reg2
RCERA           .set        000Ah   ; McBSP Rx Chan Enable Reg Partition A
RCERB           .set        000Bh   ; McBSP Rx Chan Enable Reg Partition B
XCERA           .set        000Ch   ; McBSP Tx Chan Enable Reg Partition A
XCERB           .set        000Dh   ; McBSP Tx Chan Enable Reg Partition B
PCR             .set        000Eh   ; McBSP Pin Ctrl Reg
```

```
*---------------------- General Purpose I/0 Registers -------------------------

GPIOCR      .set      003Ch       ; GP I/O Pins Control Reg
GPIOSR      .set      003Dh       ; GP I/O Pins Status Reg

*------------------- on-board I/O Memory Mapped Register --------------------

DSP_CPLD_CNTL1    .set      0000h       ; Control Reg1
DSP_CPLD_STAT     .set      0001h       ; Status Reg
DSP_CPLD_DMCNTL   .set      0002h       ; Data Memory Control Reg
DSP_CPLD_DBIO     .set      0003h       ; Daughter Brd / GPIO Reg
DSP_CPLD_CNTL2    .set      0004h       ; Control Reg2
DSP_CPLD_SEM0     .set      0005h       ; Semaphore 0
DSP_CPLD_SEM1     .set      0006h       ; Semaphore 1
```

Appendix D – ADS8320.cmd

```
/*********************************************************/
/*     TMS320C54x DSK Plus Linker Command File
/*     16K words on chip DARAM is shared by Prog
/*     and Data sections
/*********************************************************/


MEMORY
      {
      PAGE 0:            /* Pgm space */
                  VECS : origin = 0080h, length = 0080h /* vector table space */
                  PROG    : origin = 0100h, length = 1EFFh /* Pgm mem space  */

      PAGE 1:           /* Data space */
                  DAT0 : origin = 0060h, length = 0020h /* Scratch Pad mem space */
                  DAT1   : origin = 2000h, length = 1C00h    /* 7K words for Data    */
                  STK : origin = 3C00h, length = 0400h /* 1K words for Stack  */
      }

SECTIONS
      {
            .vectors    : {} > VECS    PAGE 0         /* Interrupt Vector table */
            .coeffs     : {} > PROG    PAGE 0
             .data      : {} > PROG    PAGE 0         /*Data section */
            .text       : {} > PROG    PAGE 0         /* Program code goes here */
            .varibl     : {} > DAT1    PAGE 1         /* uninitialized variables */
            .bss        : {} > DAT1    PAGE 1         /* uninitialized variables */
            .stack      : {} > STK     PAGE 1         /* software stack section */
      }
```

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265