

Interfacing the ADS7822 to TMS320C5402 DSP

Lijoy Philipose
Data Acquisition Applications

ABSTRACT

This report presents a method for interfacing the ADS7822 12-bit SAR analog-to-digital converter to the TMS320C5402 DSP.

Contents

1 Introduction.....	2
2 Hardware Interface	2
2.1 Serial Interface	3
3 Interface Software.....	4
Appendix A – ads7822.ASM	6
Appendix B – C5402vec.asm.....	11
Appendix C – regs.h	14
Appendix D - ads7822.cmd.....	16

Figures

1 Hardware Interface Block Diagram	2
2 ADS7822 Timing Diagrams	3
3 Software Flowchart	4
4 DSP Interface Timing Diagram	5

Tables

1 McBSP1 Register Settings.....	5
---------------------------------	---

1 Introduction

The ADS7822 is a 12-bit SAR analog-to-digital converter (ADC). It features 75-kHz sampling rate, differential inputs, and simple three-wire serial communication interface. This ADC requires an external reference, an external clock, and a single power source. It operates over an external reference voltage range between 50 mV and V_{CC} . The external I/O clock can vary between 10 kHz and 1.2 MHz. The ADS7822 operates over a wide voltage range from 2.0 V to 5 V. At 7.5 kHz, power dissipation is less than 60 μ W. The TMS320C5402 DSP can be programmed to interface to this data converter seamlessly. The McBSP peripherals on the C54x™ family of DSPs can be programmed for various serial-transfer protocols, one of which conforms to this interface. This report presents a simple method for interfacing to the TMS320C5402 DSP. The C54x algebraic-assembly-language source code developed for this interface is attached as appendices.

2 Hardware Interface

There are three digital communication lines on the ADS7822: $\overline{CS}/SHDN$, D_{OUT} , and DCLOCK. $\overline{CS}/SHDN$ provides the chip select function when the line is low and shutdown mode when the line is high. The D_{OUT} line is used to provide conversion output during every cycle. The DCLOCK signal synchronizes data transfer, with each bit being transmitted on the falling edge.

This device provides one pair of differential inputs: +In and -In. The -In input is not re-sampled later in the conversion cycles. Once the converter goes into hold mode, the voltage difference between +In and -In is saved on the internal capacitor array. This differential input can be used to reject small signals that are common to both inputs. The range of the -In input is limited to the range -0.2V to +1V.

The ADS7822 is designed to operate with a reference in the range from 50 mV to V_{CC} . For more discussion on reference design and specifications, refer to the ADS7822 datasheet (Texas Instruments Literature Number SBAS062).

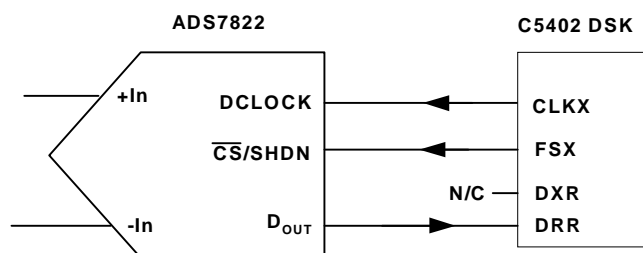


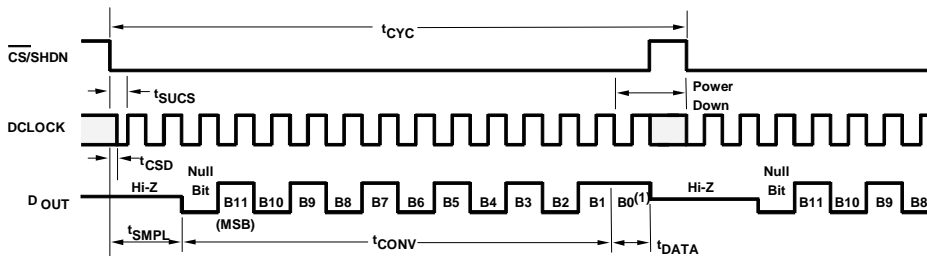
Figure 1. Hardware Interface Block Diagram

Trademark of Texas Instruments, Inc.

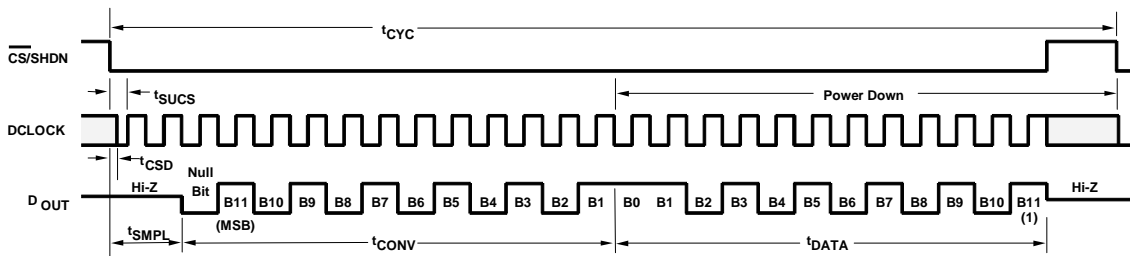
Figure 1 is a block diagram of the interface. The C5402 DSK provides signals at 3-V logic levels. If the ADS7822 is powered at 5 V, it is accepting 5-V logic signals. A digital buffer (SN74AHC244) with the same supply as the data converter would convert the DSP 3-V logic levels to the equivalent ADS7822 logic levels. If the ADC is powered at 3 V, the buffer can be removed and the interface becomes glueless.

2.1 Serial Interface

The falling edge of chip-select (CS) initiates conversion and data transfer. The first 1.5 to 2.0 clock periods are used to sample the input signal. On the second falling DCLOCK edge, the D_{OUT} line is enabled and it will output a low value, or null bit, for one clock period. The next 12 periods will produce the conversion result. The data can be read out on the rising edge of DCLOCK. If clock signals are present after the LSB(B0), the device will output the conversion result again, but this time LSB first (see Figure 2). If clock signals are still present, the D_{OUT} line goes into a high-impedance state, but does not begin a new conversion cycle. A new conversion is initiated only when CS is taken high to low.



Note: (1) After completing the data transfer, if further clocks are applied with CS LOW, the A/D will output LSB-First data then followed with zeroes indefinitely.



Note: (1) After completing the data transfer, if further clocks are applied with CS LOW, the A/D will output zeroes indefinitely.

t_{DATA} : During this time, the bias current and the comparator power down and the reference becomes a high impedance node, leaving the CLK running to clock out LSB-First data or zeroes.

Figure 2. ADS7822 Timing Diagrams

3 Interface Software

The interface software for ADS7822 comprises setting up the DSP/McBSP and initiating new conversion cycles on the ADC. Figure 3 is the software flowchart for the program listing in Appendices A to D. The DSP portion consists of initializing the stack register, clearing the interrupt flag register (IFR), setting the interrupt flag register (IMR) and interrupt vector table pointer, and so on. Since this application note uses the TMS320C5402 DSK, the CPLD register CNTL2 must be set to allow input to be McBSP1 to arrive from the expansion bus. The McBSP registers are initialized for the correct clock-stop mode, transfer word size, and the clock edge on which to read in data. Table 1 summarizes the McBSP register settings and Figure 4 is the interface timing captured on a logic analyzer.

The interrupt service routine (ISR) is used to re-format and store the samples read in on McBSP1. Once the desired number of samples have been read, the IMR register is reset, thus disabling all ISRs. The DSP will sit idle.

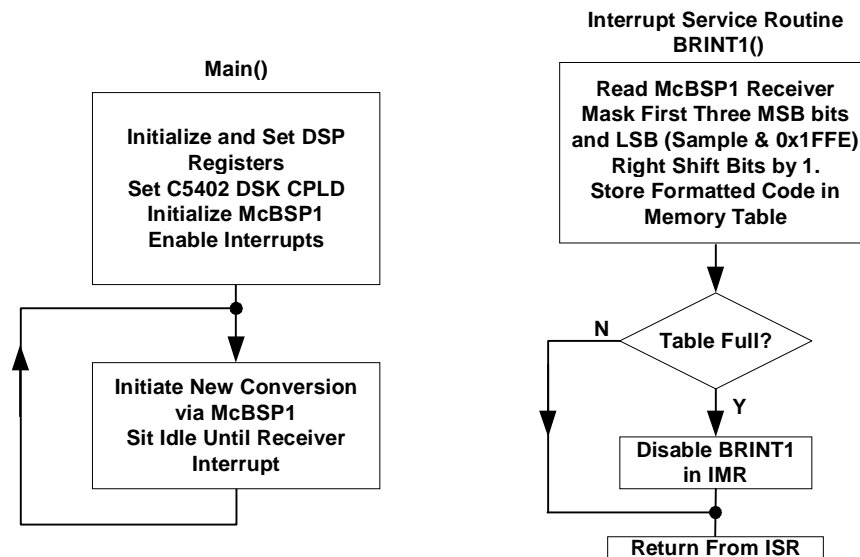


Figure 3. Software Flowchart

Table 1. McBSP1 Register Settings

McBSP1 Address	McBSP1 Sub-Address	Acronym	Register Initialized	Comment
0041		DRR11		Receiver data register
0043		DXR11		Transmit data register
0048		SPSA1		McBSP1 sub-addressing 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	0x0040	Selects one 16-bit word transfer per frame
	0x0003	RCR21	0x0000	
	0x0004	XCR11	0x0040	Selects one 16-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 is output is driven by the sample rate generator. Receive data on rising edge of CLKR. FSX and FSR are active- low signals.



Figure 4. DSP Interface Timing Diagram

Appendix A – ads7822.ASM

```

*****
* TITLE           : ADS7822 Interface routine           *
* FILE            : ads7822.ASM                       *
* FUNCTION        :main                               *
* PROTOTYPE       : void MAIN ()                     *
* CALLS           : N/A                               *
* PRECONDITION    : N/A                               *
* POSTCONDITION   : N/A                               *
* DESCRIPTION     : Program initializes DSP, McBSP1 and initiates conversions
                  cycles. Sits idle until interrupt arrives from McBSP1 Receiver.
* AUTHOR          :Data Acquisition Products, L. Philipose, Dallas, TX
*                 :
*                 :   CREATED 2000(C) BY TEXAS INSTRUMENTS INCORPORATED.
* REFERENCE       : TMS320C54x User's Guide, TI 1997
*                 : ADS7822 datasheet
*****

        .title "ADS7822 ADC"
        .mmregs
        .width 80
        .length 55

        .sect ".vectors"
        .copy "C5402vec.asm"
        .sect ".data"
        .include regs.h

AD_DP      .usect ".varibl", 0      ; AD data page for local variables.
ADCWORD    .usect ".varibl", 1      ; control word for ADC0
CLKDV1     .usect ".varibl", 1      ; McBSP clock divide by factor
Temp       .usect ".varibl", 1      ; control word for the ADC

        .sect ".text"

main:

***Initialize and set DSP registers***

DP=#0
SP=#3CF0h      ; set stack pointer register
INTM=#1        ; disable global interrupts
SWWSR = #2000h ; Need Wait-State of at least 2.
PMST = #00F8h  ; remapp interrupt vector TABLE to 0x0080h
IMR = #0400h   ;enable interrupt McBSP1 Receiver interrupt(BRINT1)
IFR = #0FFFFh  ; CLEAR IFR

***Set C5402 DSK CPLD registers***

DP=#AD_DP
@Temp=#0002h
port(DSP_CPLD_CNTL2) =@Temp      ; McBSP1 Select Control=Daughter board in CNTL2 of CPLD

```

```

***Initialize McBSP1 ***
@CLKDV1=#0052h      ; 0x0052 = 1.2MHz DCLOCK, assuming CPU clock=100MHz
call McBSP1_init    ; set serial i/o clock

INTM=#0             ; enable global interrupts

AR7=#3000h         ; Start storing receiver samples at 0x3000
AR6=#3200h         ; end of Conversion Result Table

**initiate new conversion**
NewCycle:

A=#0000h           ; dummy word = 0x0000
AR2=#McBSP1_DXR1   ; Write 0x0000 out via McBSP1 which
nop                ; will trigger new conversion cycle.
*AR2 = A           ; write out DXR11

idle(1)            ; DSP idle until conversion complete.
goto NewCycle

*****
*Interrupt Service Routine: isr_brint1 *
*Description:Interrupt due to McBSP1 receiver ready with new data *
* Reads DRR11, formats and stores 16-bit sample in data memory *
*****

isr_brint1:
AR0=AR6
AR2=#McBSP1_DRR1   ; Read out McBSP1 Receiver
NOP
B=*AR2
B=B & #1FFEh       ; Mask first two bits, null bit and LSB of the 16 bits
B=B>>>1           ; read and then right shift by one.
@ADCWORD=B         ; store re-formatted sample

TC  = (AR0 == AR7) ; read all samples?
  if (TC) goto Complete
*AR7+ = data(@ADCWORD) ; save to memory and increment to next
goto isr_Return    ; location in Data memory

Complete:
IMR =#0000h        ; disable interrupt service routine

isr_Return:
  return_enable

*****
*McBSP1_initialize: McBSP1_init *
*Description: Sets all McBSP1 registers *
*****
McBSP1_init:

mmr(McBSP1_SPSA) = #SPCR1 ; Reset McBSP1 Receiver
A=mmr(McBSP0_SPD)

```

```

A = #OFFFEh & A          ; RRST*=0
mmr(McBSP1_SPSD) = A

mmr(McBSP1_SPSA) = #SPCR2 ; Reset McBSP1 Receiver
A=mmr(McBSP1_SPSD)
A= A & #OFF2Eh          ; GRST*=0,FRST*=0,XRST*=0
mmr(McBSP1_SPSD) = A    ;

mmr(McBSP1_SPSA) = #RCR1 ;
mmr(McBSP1_SPSD) = #0040h ; One 16-bit Word per frame

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

mmr(McBSP1_SPSA) = #XCR1
mmr(McBSP1_SPSD) = #0040h ; One 16-bit Word per frame

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

mmr(McBSP1_SPSA) = #SRGR1
A=@CLKDV1
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 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      ; Reciever 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 Generatr and
                          ; sample-rate generator enabled
A= A | #0001h            ; XRST*=1
  mmr(McBSP1_SPSD) = A    ; Transmitter enabled

```

```

repeat(#6)
NOP

RETURN

*****
* McBSP1_Write:                                     *
*Description: Write to McBSP1 if transmitter is ready for *
* new data, otherwise wait.                         *
*****

McBSP1_Write:                                     ; Word to written out in A
PUSH (AR2)
PUSH (AR1)
PUSH (AR0)
Wait_On_Transmitter_1:
DP=#0
mmr(McBSP1_SPSA) = #SPCR2
B=mmr(McBSP1_SPSD)                               ; Check if McBSP1 Transmitter
                                                ; is ready for new data

B=B & #0002h;
AR1=B
nop
nop
AR0=#2
nop
nop
TC = (AR0 == AR1)                               ; transmitter ready?
if (NTC) goto Wait_On_Transmitter_1
AR2=#McBSP1_DXR1
nop
*AR2 = A                                         ; transmit word
NOP
NOP
AR0 =POP()
AR1 =POP()
AR2 =POP()

RETURN

*****
* McBSP1_Read :                                     *
*Description: Read McBSP1 if Receiver is ready with *
* new data, otherwise wait.                         *
*****

McBSP1_Read:                                     ; Receiver Word stored in Accum. B
PUSH (AR2)
PUSH (AR1)
PUSH (AR0)
Wait_On_Receiver_1:

```

```
DP=#0
mmr(McBSP1_SPSA) = #SPCR1
B=mmr(McBSP1_SPSD)      ; check if receiver is ready with data
AR0=#2                  ; to be read from DRR1
B=B & #0002h;
AR1=B
nop
nop

TC = (AR0 == AR1)
if (NTC) goto Wait_On_Receiver_1
DP=#AD_DP
AR2=#McBSP1_DRR1
NOP
B=*AR2                  ; Read out receiver
AR0 =POP()
AR1 =POP()
AR2 =POP()
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          ; NMI
        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   ; Timer 1
PRD1     .set   0031h   ; Timer 1 Period Reg
TCR1     .set   0032h   ; Timer 1 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/O 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 - ads7822.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 /* */
  .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

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

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

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265