TEXAS INSTRUMENTS

Understanding the MSP430x325 14-Bit ADC

Application Report

1999

Mixed Signal Products

SLAA039

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 acknowledgement, 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.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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.

Contents

Understanding the MSP430x325 14-Bit ADC

John D. Morgan

ABSTRACT

This application report describes and explains techniques for using the MSP430x325 ADC in preset and auto range modes.

1 Introduction

Using the MSP430x325 12+2-bit analog-to-digital converter (ADC) in the auto mode is fairly straightforward; but some programmers or applications may need to use the ADC with preset ranges. This application report describes the techniques needed to use the ADC in the preset and auto range modes.

2 Working with the MSP430x325 ADC

Section 15 of the *MSP430 Architecture Guide and Module Library Data Book* [1] has an in-depth discussion on how to program the 12+2 bit ADC, but there is no intuitive functional description on how to use the ADC in the preset range mode. Refer to section 15 throughout this report. Pages 15-18 and 15-19 give an accurate description of each bit in the 16-bit ACTL register, and with a little effort, one can set up the ACTL to properly control the ADC. The location and function of each bit used in the ADC control register is described in detail on pages 15-18 and 15-19.

2.1 Program Description

Appendix A contains a program listing of an assembly file that can be used in conjunction with this report. This program is for use with the MSP-STK430A320. The MSP-EVK430A330 can also be used but will require the addition of LM285 voltage reference and a series 2.7 kΩ resistor as shown in the schematic diagram on page 10-11 of the *Starter Kit Evaluation Manual* [2] (included in the literature of the EVK/STK kits).

The MSP-STK430A320 ADC demonstration program is specifically for MSP430x320 devices as a demonstration for using the 14-bit ADC. The LM285-25, located on the MSP-STK430A320, is used as a reference (2.5 V) for the ADC, or if the MSP-EVK430A320 kit is used, an external voltage can be used as reference on pin Svcc. The LM285-25 reference is powered directly by port pin P0.5. The program runs normally in LPM1; the basic timer provides a 250-ms interrupt that places the MSP430 in an active mode.

The 12+2 bit ADC mode of operation can be selected by pressing the trigger button to step into one of the five modes. The five modes are auto range and the 4 manual modes (A, B, C, D).

The input voltage port is A.5; an external voltage source applied to this pin can test the ADC in its five modes. *Convert* is a routine that measures the input voltage at A5 using the 14-bit ADC; the result is read and displayed for about 500 ms, with the mode displayed between each reading.

An auto range conversion routine or one of four manual ranges can be selected for A/D readings of specific input values. The value for each bit within these ranges is dependent upon the source of the reference voltage used and is derived by the following equation: (Vref/2^{14}).

The manually selectable ranges are as follows:

Range A Preset (0.00 … 0.25 × V ref)
Range B Preset (0.25 … 0.50 × V ref)
Range C Preset (0.50 … 0.75 × V ref)
Range D Preset (0.75 … 1.00 × V ref)

For example, if the onboard LM285-25, 2.5-V regulator is used, then voltages can be measured using the following preset ranges. Use caution because attempts to read values outside these limits will result in erroneous values.

Range A Preset (0.0 ... 0.625 Volts)
Range B Preset (0.625 … 1.250 Volts)
Range C Preset (1.25 … 1.875 Volts)
Range D Preset (1.875 … 2.500 Volts)

Readings in either the automatic or manual set ranges will result in a 14-bit value.

2.2 PRESET RANGE MODE

In the preset range mode, a range is selected (A–D). It is assumed that the value of the reading is within the range selected above. An A/D conversion is then executed; the final value (actual voltage) can be calculated. The A/D conversion result will yield a binary number. This binary number is then converted to a BCD number in a routine labeled BINBCD. This number represents a value above the minimum range value but below the maximum value of the preselected range.

In the above case, assuming that, for example, range B is selected. An input voltage must fall within 0.25 – 0.5 × Vref., in other words, 0.625 – 1.25 V for a 2.5-V Vref. Assuming the BCD count is 2458, using the above minimum voltage value for the B range, the final value will be:

0.625 V + ((Vref/2^{14}) × (the resulting ADC count)).

- Range B minimum value in the above example is 0.625 V.
- Vref is 2.5 V: Vref/2^{14} = 152.5879 µV/count.
- The resultant BCD count is 2458.
- 0.625 V + (152.58 µV X 2458) = 1 V.

In the manual mode, a voltage input below the minimum for the selected range results in a 0 count; input voltages above the maximum for that range result in a full 12-bit count of 4095.

2.3 Auto Range Mode Operation

The auto mode conversion process is much simpler. All that is needed is to set the proper bit on the ADC control register and call for an auto range conversion. In the attached program, the result will be in the form of a BCD number that, when multiplied by the bit value, will yield the voltage under test; no range addition is necessary. The bit value, as above, is based upon the same principal as in the preset example above, where Vref is 2.50 volts:

((Vref/2^{14}) × (the resulting ADC count)).

- Vref is 2.50 V: Vref/2^{14} = 152.5879 µV/count.
- The resultant BCD count is 6554.
- (152.58 µV × 6554) = 1 V.

2.4 12-Bit Operation

A 12-bit A/D conversion can be done by disregarding the lower two bits of the 14-bit conversion. A 12-bit conversion is part of the auto and preset range modes, but this is the lower 12 bits of a 14-bit conversion. This result is in addition to the top two preset range bits; thus 12+2 bits or 14-bit total conversion. If there is a need for a 12-bit conversion, ignore the lower 2 bits of the 14-bit conversion and adjust the bit weight value up to a 12-bit equivalent.

3 Summary

In summary, the auto range function is much easier to use. In a preset range A/D conversion, time to do a conversion may be faster, requiring only 96 A/D clock periods as opposed to the 132 A/D clocks required for the auto range mode. Both modes, when applied correctly, yield accurate results.

4 References

1. *MSP430 Family Architecture Guide and Module Library* — SLAUE10B
2. *MSP430 Family Starter Kit Evaluation Manual*
3. *MSP430 Family Applications Report* — SLAEE10C
4. *MSP430 Family Software User’s Guide* — SLAUE11

Appendix A Assembly Listing

```
*******************************************************************************************************************
;                   MSP–STK430A320 ADC DEMONSTRATION PROGRAM
;
;      Programmed specifically for MSP430x320 device as a Demonstration for
;      using the 14–ADC. The LM285-25 which is located on the MSP–STK430A320
;      can be used as reference (2.5V), or if the MSP–EVK430A320 Kit is used,
;      an external voltage can be used as reference on pin Svcc. On the
;      MSPSTK430A320. The LM285-25 is powered directly by port pin P0.5. The
;      program runs normally in LPM1, basic timer is used to provide a 250 ms
;      interrupt which restarts program to active.
;
;      The 12+2 bit ADC mode of operation can be selected by stepping into
;      one of the 5 modes by pressing the ”Trigger” button. The five modes
;      are Auto Range and the 4 manual modes (A,B,C,D).

;      The input voltage port is A.5, an external voltage source can be
;      applied to this pin and used to test the ADC in it’s 5 modes.
;
;      Convert: is routine which with the input voltage at A5, using the
;      14-bit ADC, result is read and displayed for about 500 ms, with the
;      mode displayed between each reading.
;
*******************************************************************************************************************
RAM_orig     .set   00230h              ; Program start address loaded in
RAMI_vectors .set   003FFh              ; Interrupt Vectors relocated to RAM
Stack        .set   003Deh              ; Stackpointer

IE1          .equ   0h                  ; Interrupt Enable register 1
P00IE        .equ   04h                 ; IE1 bit position P0.0 interr enable
P01IE        .equ   08h                   IE1 bit position P0.1 interr enable
IE2          .equ   01h                 ; Interrupt Enable register 2
ADIE         .set   004h                ; IE2 bit position for ADC interr
                                        ; enable
BTIE         .set   080h                ; IE2 bit position for BT interr
                                        ; enable
IFG1         .equ   02h                 ; Interrupt Flag register 1
IFG2         .equ   03h                 ; Interrupt Flag register 2
CPUOFF       .set   10h                 ; CPUOFF bit in SR
P0IN         .equ   010h                ; Port 0 input register
P0OUT        .equ   011h                ; Port 0 output register
P0DIR        .equ   012h                ; Port 0 direction register
P0IFG        .equ   013h                ; Port 0 interrupt flag register
P0IES        .equ   014h                ; Port 0 interrupt edge register
P0IE         .equ   015h                ; Port 0 interrupt enable register
LCDCTL       .equ   030h                ; LCD Control register
LCDM1        .equ   031h                ; First LCD display RAM location
BTCTL        .equ   040h                ; BT control Register
ACTL         .equ   0114h               ; ADC Control register
ADAT         .equ   0118h               ; ADC Data register
CS           .set   1                   ; ACTL bit position for Conversion
                                        ; Start
ISelA3       .set   00Ch                ; ACTL bit position for input channel
A3ISelA5     .set   014h                ; ACTL bit position for input channel
A5CSelAx     .set   0100h               ; ACTL bit position for no current
                                        ; source
RSelAut      .set   0800h               ; ACTL bit position for  auto range
RangeA       -set   0000h               ; Range A Preset
RangeB       .set   0200h               ; Range B Preset
RangeC       .set   0400h               ; Range C Preset
```

```
RangeD       .set   0600h               ; Range D Preset
Vref         .set   0002h               ; ACTL bit position for source of
                                        ; Vref
Pd           .set   1000h               ; ACTL bit position to select Power
                                        ; Down
WDTCTL       .equ   0120h               ; Watchdog control register
WDTHold      .equ   80h                 ; Pattern to hold watchdog
WDT_key      .equ   05A00h              ; Key to access watchdog
WDT_stop     .equ   05A80h              ; Watchdog hold+key

Index        .equ   R15                 ; Index for branching
SEC          .equ   0220h               ; Byte for counting full seconds
MIN          .equ   0221h               ; Byte for counting of minutes
SEC25        .equ   0222h               ; Byte for counting .25 seconds

;****************************************************************************
; RESET: Main LOOP of Program
;****************************************************************************
;
sect         ”MAIN”,RAM_orig
RESET        MOV    #Stack,SP           ; Initialize stackpointer
             CALL   #Setup              ; Setup Peripherals

LOOP         BIS    #CPUOFF,SR          ; Turn off CPU save some power, stop here
             BIS    #CPUOFF,SR          ; Delay
             BIS    #CPUOFF,SR          ; Delay
             CALL   #ClearLCD           ; Clear LCD display
             BR     BRANCH(Index)       ; Branch to mode

             .even
BRANCH       .word  ADCauto
             .word  ADCrngA
             .word  ADCrngB
             .word  ADCrngC
             .word  ADCrngD

:****************************************************************************
;  Setup Peripherals: Prepare UART, LCD, Basic Timer, ADC, PO and clear
;  LCD RAM
;****************************************************************************
;
Setup        MOV    #(WDT_stop),&WDTCTL ; Stop Watchdog Timer
             BIS.b  #(BTIE+ADIE),&IE2   ; Enable Basic Timer, ADC interrupts
             MOV.b  #(P00IE+P01IE),&IE1 ; Enable P0.1/RS232 and P0.0
                                        ; Interrupts
             CLR.b  &IFG1               ; Clear any Interrupt flags
             CLR.b  &IFG2               ; Clear any Interrupt flags
SetupP0      MOV.b  #0E0h,&P0DIR        ; P0.7,6,5 outputs for LM285& TSL252
             CLR.b  &P0OUT              ; All Outputs low
             BIS.b  #01h,&P0IES         ; P0.0 interrupt is High/Low

SetupMEM     CLR    Index               ; Clear mode branching register
             CLR.b SEC                  ; Clear Byte for counting Seconds
             CLR.b MIN                  ; Clear Byte for counting Minutes
             MOV.b #04,SEC25            ; Byte for counting .25 Seconds
SetupLCD     MOV.b #0FFh,&LCDCTL        ; STK LCD, 4Mux, all segments
SetupBT      MOV.b #0B4h,&BTCTL         ; STK LCD, 250ms interrupt

             EINT                       ; Enable interrupts
```

```
ClearLCD      MOV #15,R5                ; 15 bytes of LCD RAM to clear
Clear1        MOV.b   #0,LCDM1–1(R5)    ; Move #0 to LCD display RAM
              DEC R5                    ; All 15 bytes
              JNZ Clear1                ; Not done?
              RET

;****************************************************************************
;  Displays ”Auto” on LCD
;****************************************************************************
;  ADCauto: Reads 14–bit ADC–value, convert binary to BCD, display 14–bit,
;  5 BCD digit result, starts next ADC conversion.
;****************************************************************************
;
ADCauto       MOV.b #0BBh,LCDM1+5       ; ”A” Display ”Auto”
              MOV.b #094h,LCDM1+4       ; ”u”
              MOV.b #0ACh,LCDM1+3       ; ”t”
              MOV.b #09Ch,LCDM1+2       ; ”o”
              MOV     #Pd+ISelA5+CSelAx+RSelAut,&ACTL
              JMP     Convert

;****************************************************************************
;  Displays ”A” on LCD
;****************************************************************************
;  ADCrngA: Reads 12–bit ADC–value, convert binary to BCD, This value
;  can then be;  added to Range a minimum value to durive 14 bit
;  solution
;****************************************************************************

ADCrngA       MOV.b #0BBh,LCDM1+5       ; Display ”A”
              MOV #Pd+ISelA5+CSelAx+RangeA,&ACTL
              JMP Convert

;****************************************************************************
;      Displays ”b” on LCD
;****************************************************************************
;  ADCrngB: Reads 12–bit ADC–value, convert binary to BCD, This value
;  can then be;  added to Range a minimum value to durive 14 bit
;  solution
;****************************************************************************

ADCrngB       MOV.b #0BCh,LCDM1+5       ; Display ”b”
              MOV #Pd+ISelA5+CSelAx+RangeB,&ACTL
              JMP Convert

;****************************************************************************
;  Displays ”C” on LCD
;****************************************************************************
;  ADCrngC: Reads 12–bit ADC–value, convert binary to BCD, This value
;  can then be;  added to Range a minimum value to durive 14 bit
;  solution
;****************************************************************************

ADCrngC       MOV.b #08Ch,LCDM1+5       ; Display ”C”
              MOV #Pd+ISelA5+CSelAx+RangeC,&ACTL
              JMP Convert
```

```
;****************************************************************************
;  Displays ”d” on LCD
;****************************************************************************
;  ADCrngD: Reads 12–bit ADC–value, convert binary to BCD, This value
;  can then be;  added to Range a minimum value to durive 14 bit
;  solution
;****************************************************************************

ADCrngD      MOV.b #09Eh,LCDM1+5        ; Display ”d”
             MOV #Pd+ISelA5+CSelAx+RangeD,&ACTL
             JMP Convert

;****************************************************************************
;  Convert: Reads 12/14–bit ADC–value, Range dependent, convert binary
;  to BCD,;  display 14–bit, 5 BCD digit result, starts next ADC
;  conversion.
;****************************************************************************

Convert      BIS #CPUOFF,SR             ; Turnoff CPU, Wait 250 ms
             BIS #CPUOFF,SR             ; Turnoff CPU, Wait 250 ms
             BIC #Pd,&ACTL              ; Power up ADC for next conversion
             BIS.b #0E0h,P0OUT          ; Set P0.7,6,5 to power LM285& TSL252
             BIS #CS,&ACTL              ; Start next ADC conversion
             BIS #CPUOFF,SR             ; Turn off CPU, quite ADC, stop here
             MOV &ADAT,R5               ; Get ADC–value in R5
             CALL #BINBCD               ; ADC–value in R5 to BCD R8|R7
             CALL #Display              ; Display result on LCD
             JMP LOOP                   ; Wait for next time interval

;****************************************************************************
;  Display: 4 BCD digits contained in R7 and 1 digit in R8 to LCD;  R5 is ;
;  pointer to LCD digit
;****************************************************************************

Display      MOV #LCDM1+1,R5            ; R5 point to LCD display RAM
L1           MOV R7,R6                  ; Copy BCD value to R6
             RRA R7                     ; Rotate a nibble (digit)
             RRA R7
             RRA R7
             RRA R7
             AND #0Fh,R6                ; Expose single digit
             MOV.b LCD_Tab(R6),0(R5)    ; Move digit to LCD display RAM
             INC R5                     ; Point to next LCD display RAM
             CMP #LCDM1+5,R5            ; 4 digits displayed?
             JL L1                      ; Still in 4 digit range?

             AND #0Fh,R8                ; Display 1 digit from R8
             MOV.b LCD_Tab(R8),0(R5)    ; Move digit to LCD display RAM
             RET
```

```
;****************************************************************************
;  BINBCD: Converts binary number in R5 is to packed BCD and result stored;
;  in R8 and R7: R8|R7   (Lutz Bierl)
;****************************************************************************

BINBCD       MOV #16,R6                  ; LOOP COUNTER
             CLR R8                      ; 0 –> RESULT MSD
             CLR R7                      ; 0 –> RESULT LSD
L$1          RLA R5                      ; Binary MSB to carry
             DADD R7,R7                  ; RESULT x2 LSD
             DADD R8,R8                  : MSD
             DEC R6                      ; THROUGH?
             JNZ L$1                     ; Not through
             RET

;****************************************************************************
;  ADC Interrupt Service Routine:  ;  Turn off ADC and Sensor, result remains
;  in ADAT
;****************************************************************************

ADInt        BIS #Pd,&ACTL               ; Power down ADC
             BIC.b #0E0h,P0OUT           ; Clear P0.7,6,5 turn off
                                         ; LM285& TSL252
             BIC #CPUOFF,0(SP)           ; Turn on CPU after RETI
             RETI                        ; Return from interrupt

;****************************************************************************
;  Basic Timer Interrupt Service Routine: Restart program execution and
;  up dates RTC
;****************************************************************************

BTInt        BIC #CPUOFF,0(SP)           ; Turn on CPU after RETI
RTCend       RETI

;****************************************************************************
;  P0.0 Interrupt: Increment mode Index register. P0.0 interrupt flag is
;  recleared at end  of ISR and Delay counted to reduce key bounce. CMP #06
;  because you need to look at every even address, i.e.,  2,4,6
;****************************************************************************

P00Int       INCD Index                  ; Double increment branch index
             CMP #010,Index              ; Out of range of branch table?
             JL Delay                    ; Jump if still in range less than 3
             CLR Index                   ; Reset mode index to 0
Delay        MOV #0FFFFh,R5              ; Load delay, R5
L3           DEC R5                      ; Delay to reduce key bounce
             JNZ L3 BIC.b #04h,&IFG1     ; reClear P0.0 in case key bounce!
             RETI                        ; flag is cleared on interrupt also.
```

```
;*****************************************************************************
;   STK LCD  Definitions
;*****************************************************************************

LCD_TYPE
a      .equ   01h
b      .equ   02h
c      .equ   10h
d      .equ   04h
e      .equ   80h
f      .equ   20h
g      .equ   08h
h      .equ   40h

;––– character definitions

LCD_Tab .byte a+b+c+d+e+f         ; displays “0”
       .byte   b+c                ; displays “1”
       .byte   a+b+d+e+g          ; displays “2”
       .byte   a+b+c+d+g          ; displays “3”
       .byte   b+c+f+g            ; displays “4”
       .byte   a+c+d+f+g          ; displays “5”
       .byte   a+c+d+e+f+g        ; displays “6”
       .byte   a+b+c              ; displays “7”
       .byte   a+b+c+d+e+f+g      ; displays ”8”
       .byte   a+b+c+d+f+g        ; displays “9”
       .byte   a+b+c+e+f+g        ; displays “A”
       .byte   c+d+e+f+g          ; displays “B”b
       .byte   d+e+g              ; displays “c”
       .byte   b+c+d+e+g          ; displays “D” d
       .byte   d+e+f+g            ; displays “T” t
       .byte   c+d+e              ; displays ”U” u
LCD_Tab_End

;*****************************************************************************
; Interrupt vectors
;*****************************************************************************

       .even                      ; Following section must be evenly aligned
       .sect  ”Int_Vect”,I_vectors–31
       .word  RESET               ; Port0, bit 2 to bit 7
       .word  BTInt               ; Basic Timer
       .word  RESET               ; no source
       .word  RESET               ; no source
       .word  RESET               ; Timer Port
       .word  ADInt               ; EOC from ADC
       .word  RESET               ; no source
       .word  RESET               ; no source
       .word  RESET               ; no source
       .word  RESET               ; no source
       .word  RESET               ; Watchdog/Timer, Timer mode
       .word  RESET               ; no source
       .word  RESET               ; Address of UART handler
       .word  P00Int              ; P0.0
       .word  RESET               ; NMI, Osc. fault
       .word  RESET                ; POR, ext. Reset, Watchdog
       .end
```