Application Report
SLAA104 – November 2000

An MSP430F11x1 Sigma-Delta Type Millivolt Meter

Mark Buccini *Mixed Signal Products*

ABSTRACT

This application report describes a method of implementing a low-cost, 12-bit sigma-delta type A/D converter using the MSP430F11x1 family of 16-bit RISC-like mixed signal processors. The solution described in this report uses the comparator_A peripheral module. For demonstration purposes, the application implements a voltmeter transmitting the result in millivolts using a 9600-baud UART protocol. This report is written for the MSP430F11x1 but can be adapted to any MSP430 family member incorporating comparator_A.

Contents

Figures

References

1 Introduction

Using the MSP430F11x1 family of 16-bit mixed signal processors, a low-cost high-resolution A/D converter can easily be implemented using the on-chip comparator_A. The A/D converter described is an integrating type with characteristics similar to sigma-delta techniques. The demonstration circuit in Figure 1 achieved 1mv resolution with a 3.3-V power supply. This type of high-resolution A/D converter is suitable for measuring many types of precision sensors with slowly changing values such as temperature, pressure, light, and voltage. The precision characteristics of the A/D converter, combined with MSP430s ultra-low power, make this solution ideal for battery-powered measurement applications.

The key features of the MSP430F11x1 integrating A/D converter:

- 1 mV / 12-bits of resolution demonstrated
- A true integrating solution with excellent noise suppression
- No comparator calibration required
- Only two external components required for implementing the A/D converter.

Important: Please review the current MSP430x11x1 family data sheet and MSP430 Family Users Guide for exact MSP430x11x1 device specifications and module descriptions.

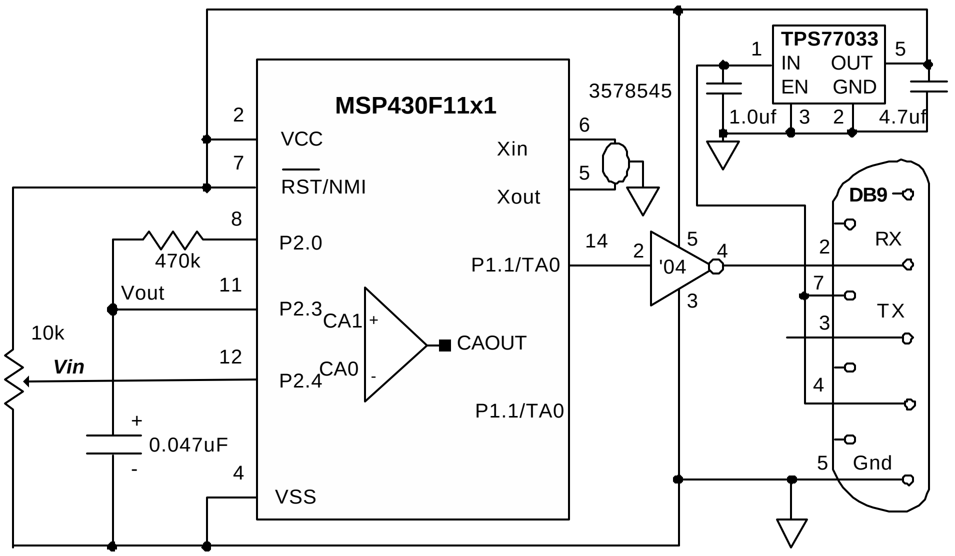

Figure 1. MSP430F11x1 Millivolt Meter

2 Theory of Operation

The concept of an integrating A/D converter is to match an unknown voltage of interest Vin, with a known voltage, Vout. Using a single digital output, the MSP430F11x1 implements a 1-bit digital to analog converter (DAC). The DAC outputs a controlled number of short symmetrical pulses to generate an output density to drive Vout equal to Vin. Software modulates and counts the DAC output pulses of a fixed length loop. The DAC is low-pass filtered, creating a constant analog voltage Vout. See Figure 2. The software feedback loop compares the analog voltage Vout to the interesting voltage Vin using comparator_A, modulating the DAC output pulse such that two analog voltages are maintained equal. Over a fixed number of symmetric DAC pulses, the feedback loop integrates the number of high DAC outputs (n) required to maintain Vout equal to Vin. The feedback loop can be modified in length to deliver the required resolution. For example, a loop of 256 counts would correspond to an 8-bit conversion, 4096 counts to a 12-bit conversion. In the demonstration circuit, the count of 3300 is selected to match the VCC of the MSP430, which is regulated to by the TPS77033 to 3.300 V (3300 millivolts). Each count is then conveniently equal to 1 mV. In all cases, measured A/D value is ratio-metric to the MSP430 Vcc, which drives the 1-bit DAC. For example, in the demonstration circuit regulated to 3.300 V, if the A/D conversion code n were 1800 (out of 3300), this would indicate Vin has a value of exactly 50% of VCC or 1800 mV. In the final application, Vin must be limited to the common mode range of comparator_A specified in the device data sheet.

Assuming that Vout is precharged to equal Vin prior to the A/D conversion cycle, and the I/O on the MSP430 is low RDSon (<300 ?) - which drives the DAC high pulse is very near VCC which is 3.300 volts, and the low pulse to VSS, Vin is:

$$\text{if} \quad Vin(t) = Vin\,, \quad \text{then} \quad \text{solving} \ \ldots$$

$$Vin(t) = Vin + n(VCC - Vin)\left(1 - e^{-\frac{t}{RC}}\right) - (3300 - n)(Vin)\left(1 - e^{-\frac{t}{RC}}\right)$$

$$Vin = VCC\,\frac{n}{3300}$$

$$Vin\,(mv) = n$$

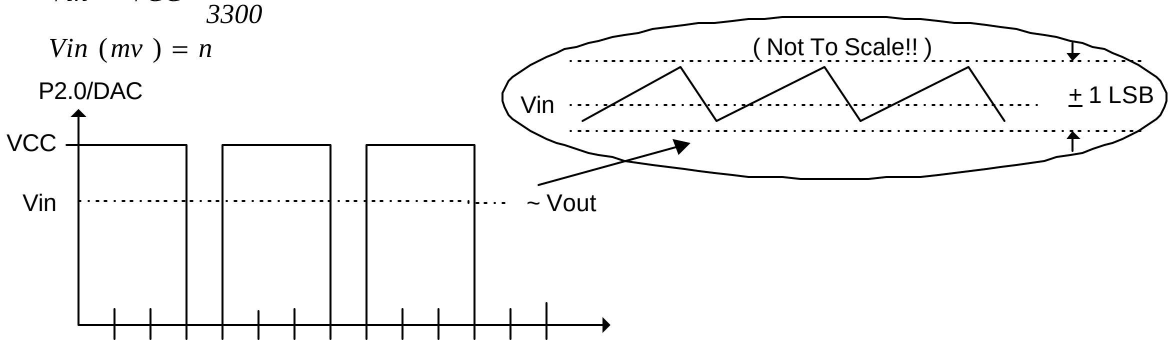

Figure 2. Integrated DAC Output

2.1 Calculating the 1-Bit DAC Low-Pass Filter Values

The resistor and capacitor selection for the 1-bit DAC low-pass filter is important and chosen such that the ripple voltage seen at Vout is not more than 1 LSB during any single DAC pulse. This small ripple ensures conversion accuracy + 1 LSB. The software feedback loop that determines the DAC pulse period, as coded, is 14 CPU clock cycles long. In this report the MSP430F11x1 CPU clock or master clock (MCLK) is set to equal that of an external 3.578545 MHz color-burst crystal. The DAC pulse period is 14/3578545s or 4.9us. The DAC ripple voltage is calculated to less than 1 LSB (1/3300). Solving for this example:

$$V(t) = V\left(1 - e^{-\frac{t}{RC}}\right)$$

$$\frac{V(t)}{V} = 1 - e^{-\frac{t}{RC}}$$

$$e^{-\frac{t}{RC}} = 1 - \frac{V(t)}{V}$$

$$-\frac{t}{RC} = \ln\left(1 - \frac{V(t)}{V}\right)$$

$$-RC = \frac{t}{\ln\left(1 - \frac{V(t)}{V}\right)} = \frac{4.9\,us}{\ln\left(\frac{3299}{3300}\right)} = -0.01616$$

$$if \quad R = 470\ k \quad then \quad C = 0.035\ uf \quad use \approx 0.047\ uf$$

2.2 MSP430F11x1 Resources Required

Few MSP430F11x1 resources are required to implement the integrating A/D converter.

- The comparator_A module and associated comparator inputs P2.3 and P2.4.
- A single digital output for the 1-bit DAC, P2.0 in the example.

Any available MSP430F1101 I/O can be used for the 1-bit DAC. P2.0 is used in this report. P2.0 is configured as a digital output using the port 2 output register (P2OUT). P2.3 and P2.4 can be general-purpose I/O or selected as inputs to comparator_A using the port 2 option select register (P2SEL), which is how they are used in this report. P2.3 and P2.4 must also be configured as inputs (the default condition) which are done using the port 2 direction register (P2DIR). Comparator_A is used to compare the analog voltages Vout and Vin and is configured using comparator_A Control Registers CACTL1 and CACTL2. The result is transmitted using a Timer_A configured as a form of UART, at 9600-baud on P1.1. This UART also uses timer_A resources. Please see the reference section at the end of this report for the UART application report reference number.

2.3 External Hardware Required

Only two external components are required to implement the A/D converter

- One resistor
- One low-leakage film -type polyester capacitor

The test circuit in Figure 1 communicates with a PC (optionally) and powers up directly from the RS232 interface. To accomplish RS232 line level translation, integrated circuits such as TI's MAX232 provide a compliant solution. For the demonstration circuit in Figure 3, a single-gate TI SN74AH1GC04 inverter is used for the PC serial port interface. A TPS77033 low-power low-dropout voltage regulator and 3.578545 MHz color-burst crystals are also used.

2.4 Reducing System Error

The overall accuracy of the A/D converter is determined by the accuracy of the 1-bit DAC. The DAC switches between the MSP430's VCC and GND. The low RDS (on) of the MSP430 port pins, which drives the DAC, insure DAC output levels extremely close to VCC and GND. The accuracy of system V_{CC} that supplies the MSP430 determines the accuracy of the high level DAC output. In a demonstration circuit, V_{CC} was measured externally to be exactly 3300 millivolts. This value 3300 is stored as VCC_Cal and called by firmware inside of the Meas_ADC loop. The A/D converter is ratio-metric to VCC. If an absolute voltage measurement is required, as in this report, the A/D reading will be as accurate as the absolute system V_{CC}, which drives the 1-bit DAC.

The comparator will have an offset that introduces error. A unique feature of comparator_A is the inclusion of input multiplexer allowing the terminal inputs of the comparator to be swapped and the output inverted. This simple but powerful multiplexer eliminates the necessity to compensate for the comparator offset. Compensation for the comparator offset is as simple as averaging two simultaneous samples with the input terminals swapped. Or, as in this report, 50% of the conversion sequence is performed terminals in one configuration, 50% of the terminals is swapped. The switching of the comparator_A terminals is controlled with the CAEX bit in the CACTL1 control register. Please see the device specific data sheet for a detailed description of this feature. If in an application external calibration is required; calibration value can be stored in MSP430F11x1 flash or even RAM if the unit is continuously powered.

Important: The demonstration circuit in this report powers directly from a PC serial port. Certain PC switching power supplies are very noisy and this noise can be carried from the serial port to the MSP430. Additionally good PCB layout techniques must always be observed to achieve precision analog performance.

The software feedback loop must be equal in paths and uninterrupted during the A/D conversion. As coded the path through the DAC high and low path is exactly 14 cycles to insure symmetry in a high or low pulse. Also it is the necessary that the complete software feedback loop be uninterrupted during the measurement, again to insure symmetry of all DAC pulses. If interrupts are possible during the measurement subroutine, consider using the DINT and EINT before and after the subroutine respectively. This will prevent unintended interrupts from disrupting the run-time critical software feedback loop.

3 Description of Software Listing fet_intADC.s43

The demonstration software listing fet_intADC.s43 is sectioned in several subroutines. On RESET an Init_Sys subroutine is called to initialize the MSP430 system in which I/O and peripherals are configured.

Inside the mainloop, the A/D converter is sampled by calling the Meas_ADC subroutine. Meas_ADC samples the A/D converter with the conversion code passed in register ADCData. The 3300-count integration loop inside of the Meas_ADC subroutine directly counts the required DAC high pulses in decimal. Thus with the MSP430 V_{CC} regulated to 3.300 V, which is the high value of the DAC output, the counted value A/D value directly represents millivolts.

```
;-----------------------------------------------------------------------------
Meas_ADC;     Subroutine: Meaure A/D Converter, BCD A/D Result --> ADCData
;             R15 used as working register and not saved
;-----------------------------------------------------------------------------
              mov.b   #CAON,&CACTL1          ; Comparator on
              clr     ADCData                ; Clear ADCData register
              call    #Sample_ADC            ; Sample --> ADCData
              call    #Sample_ADC            ; Sample --> ADCData+Previous
Meas_Over     clr.b   &CACTL1                ; Comparator off
              ret                            ; Return from Subroutine
                                             ;
Sample_ADC    mov    &VCC_Cal,R15            ; ~3300
              rra    R15                     ; VCC/2
;             PreCharge Capacitor            ;
Pre_ADC       bis.b  #DAC_Out,&P2OUT         ; Set power to capacitor
C1            bit.b  #CAOUT,&CACTL2          ; Comparator out hi/low?
              jz     C1                      ;
;             Adjust and integrate DAC Pulse ;
Test_DAC      bit.b  #CAOUT,&CACTL2          ; Comparator hi/low?
              jnc    Low1                    ; Jump --> Low
High          bic.b  #DAC_Out,&P2OUT         ; Reset power to capacitor
              jmp    Meas_                   ;
Low1          bis.b  #DAC_Out,&P2OUT         ; Set power to capacitor
              setc                           ;
              dadc    ADCData                ; BCD increment ADCData
Meas_         dec    R15                     ; Decrement Loop Count
              jnz    Test_DAC                ; Measurement Loop Over?
              bic.b  #DAC_Out,&P2OUT         ; Reset power to capacitor
              xor.b  #CAEX,&CACTL1           ; invert comparator terminals
              ret                            ; Return from Subroutine
```

Next a subroutine TX_2_PC is called to transmit the millivolt reading to a PC @ 9600 8N1. The main loop repeats. The main loop is concluded with a software delay.

3.1 Firmware for a Basic 12-bit ADC

Below is firmware that can be used to implement a basic 12-bit ADC ratio-metric to VCC. The firmware is almost identical to the demonstration software except two binary ADC samples are made (with the comparator inputs switched) and averaged. The result delivered in ADCData will be 0000h – 0FFFh.

```
;-----------------------------------------------------------------------------
Meas_ADC     Subroutine: Meaure A/D Converter, binary A/D Result --> ADCData
;            R15 used as working register and not saved
;-----------------------------------------------------------------------------
             mov.b   #CAON,&CACTL1          ; Comparator on
             clr     ADCData                ; Clear ADCData register
             call    #Sample_ADC            ; Sample --> ADCData
             call    #Sample_ADC            ; Sample --> ADCData+Previous
             rra     ADCData                ; ADCData/2
Meas_Over    clr.b   &CACTL1                ; Comparator off
             ret                            ; Return from Subroutine
                                            ;
Sample_ADC   mov    #04096,R15              ; 12-bits
;            PreCharge Capacitor            ;
Pre_ADC      bis.b  #DAC_Out,&P2OUT         ; Set power to capacitor
C1           bit.b  #CAOUT,&CACTL2          ; Comparator out hi/low?
             jz     C1                      ;
;            Adjust and integrate DAC Pulse ;
Test_DAC     bit.b  #CAOUT,&CACTL2          ; Comparator hi/low?
             jnc    Low1                    ; Jump --> Low
High         bic.b  #DAC_Out,&P2OUT         ; Reset power to capacitor
             jmp    Meas_                   ;
Low1         bis.b  #DAC_Out,&P2OUT         ; Set power to capacitor
             nop                            ;
             inc    ADCData                 ; Increment ADCData
Meas_        dec    R15                     ; Decrement Loop Count
             jnz    Test_DAC                ; Measurement Loop Over?
             bic.b  #DAC_Out,&P2OUT         ; Reset power to capacitor
             xor.b  #CAEX,&CACTL1           ; invert comparator terminals
             ret                            ; Return from Subroutine
```

3.2 Use of TI Standard Peripheral Definitions

The code for this report was developed using the MSP-FET430X11X and the included embedded workbench software development environment. The code is written in assembler (.s43). All examples in this report use TI MSP430 standard peripheral definitions. The following directive copies MSP430x11x1 peripheral register definitions into the source:

```
#include  "msp430x11x1.h"
```

TI encourages programmers to use TI standard peripheral definitions to promote commonality with the MSP430 users guides, and ease code debugging.

4 Summary

The modern 16-bit RISC-like architecture of the MSP430 is particularly well suited for applications requiring high-resolution data manipulation. The MSP430 16-bit architecture can easily handle, in single instructions, the high-resolution A/D converter data in this report.

The integrating A/D conversion technique described in the report is simple, inexpensive, and requires few MSP430 resources. The resolution results can be quite impressive considering the low-cost and low-complexity of the total solution. Integrating A/D conversion is not particularly fast. After Vin has been pre-charged to equal Vout, the 3300-count A/D described in this report requires 13 ms per conversion using a 3.578545 MHz. Techniques can be used to reduce the A/D cycle time such as leaving the charge capacitor charged if A/D conversions are done successively or using a faster MSP430 MCLK. The maximum MSP430F11x1 MCLK is much higher than 3.578545 MHz used in this report. The conversion speed of an integrating A/D is typically sufficient for remote sensors, data logging, and measuring slow changing data such a temperature, battery voltage, and pressure.

There are additional advantages of using an integrating-type A/D converter with short symmetrical pulse periods as described in this report. A/D conversion is done with conventional PWM, a single long PWM period and varying duty cycle, requiring a relatively large RC time constant to avoid ripple error on Vout and can take several periods to settle, slowing the total conversion process. The short symmetrical pulse period used with an integrating converter settles fast do to the short RC time constant. And, because integrating A/D conversion actually integrates or accumulates many single-bit conversions, errors caused from noise spikes tend to be suppressed in the final result.

fet_intADC.s43 code

```
#include  "msp430x11x1.h"
;****************************************************************************
NAME         ;FET_intADC MSP430F11x1 Demo
;
;   Description: This program demonstrates how to implement an MSP430F11x1
;   millivolt meter using comparator_A.  A voltage is read and TXed
;   in millivolts to a PC at 9600 baud using an 8N1 protocol.
;
;            CPU registers used
#define      ADCData  R11
#define      BitCnt   R6
#define      RXTXData R5
;
;            Conditions for 9600 Baud HW/SW UART, MCLK = 3.579545MHz
Bitime_5     equ    0186                    ; 0.5 bit length
Bitime       equ    0373                    ; 104 us
TXD          equ    002h                    ; TXD on P1.1
DAC_Out      equ    001h                    ; P2.0 DAC Output
CR           equ    0dh                     ; ASCII Carriage Return


;
;               MSP430F1121
;            _________________
;       3.3v|              XIN|-
;        |  |                 | 3.58MHz
;        ---|RST          XOUT|-
;           |                 |
;    AIN--->|P2.4             |
;           |                 |
;    +-470k-|P2.0             |
;    |      |                 | 9600 8N1
;    +----- |P2.3         P1.1|-------->
;    |      |                 |
;   ===.047 |                 |
;    |      |                 |
;    ------ |VSS              |
;
;   M.Buccini
;   Texas Instruments, Inc
;   September 2000
;***************************************************************************
;
;-----------------------------------------------------------------------------
            ORG     010FEh                ; Information Memory
;-----------------------------------------------------------------------------
VCC_Cal     DW      03300                  ; Actual system VCC
                                           ;
;-----------------------------------------------------------------------------
            ORG     0F000h
;-----------------------------------------------------------------------------
RESET        mov    #300h,SP               ; Initialize stackpointer
             call   #Init_Sys              ; Initialize system peripherals
                                           ;
Mainloop     call   #Meas_ADC              ; Measure ADC,
             call   #TX_2_PC               ; TX ADC results to PC/user
Delay        push   #0FFFFh                ; Software delay to TOS
L1           dec    0(SP)                  ; Decrement TOS
             jnz    L1                     ; Delay over?
             incd   SP                     ; Clean TOS
             jmp    Mainloop               ; Repeat
                                           ;
;----------------------------------------------------------------------------
Init_Sys;    Initialize System Peripherals
;----------------------------------------------------------------------------
             mov    #WDTPW+WDTHOLD,&WDTCTL  ; Stop WDT
SetupC0      mov    #OUT,&CCTL0            ; TXD Idle as mark
SetupP1_2    bis.b  #TXD,&P1SEL            ; P1.1/TA0 for TXD function
             bis.b  #TXD,&P1DIR            ; P1.1 = TXD output
             bis.b  #DAC_Out,&P2DIR         ; P2.0 = DAC
SetupCA      mov.b  #P2CA0+P2CA1,&CACTL2    ; P2.3+ P2.4-
SetupBC      bis.b  #XTS,&BCSCTL1           ; ACLK = LFXT1 HF XTAL
SetupOsc     bic.b  #OFIFG,&IFG1            ; Clear OSC fault flag
             mov    #0FFh,R15
SetupOsc1    bit.b  #OFIFG,&IFG1            ; OSC fault flag set?
             jnz    SetupOsc                ;
             dec    R15                     ; additional delay to ensure startup
             jnz    SetupOsc1
             bic.b  #OFIFG,&IFG1            ; Clear OSC fault flag
             bis.b  #SELM1+SELM0,&BCSCTL2   ; MCLK = LFXT1
                                            ;
             mov    #TASSEL0+TACLR,&TACTL   ; Timer_A, ACLK, TAR cleared
             bis    #MC1,&TACTL             ; Start TA in continous mode
             eint                           ; General enable interrupts
             ret                            ; Return from Subrountine
                                            ;
;-----------------------------------------------------------------------------
Meas_ADC;    Subroutine: Meaure A/D Converter, BCD A/D Result --> ADCData
;            R15 used as working register and not saved
;-----------------------------------------------------------------------------
             mov.b   #CAON,&CACTL1          ; Comparator on
             clr     ADCData                ; Clear ADCData register
             call    #Sample_ADC            ; Sample --> ADCData
             call    #Sample_ADC            ; Sample --> ADCData+Previous
Meas_Over    clr.b   &CACTL1                ; Comparator off
             ret                            ; Return from Subroutine
                                            ;
Sample_ADC   mov    &VCC_Cal,R15            ; ~3300
             rra    R15                     ; VCC/2
;            PreCharge Capacitor            ;
Pre_ADC      bis.b  #DAC_Out,&P2OUT         ; Set power to capacitor
C1           bit.b  #CAOUT,&CACTL2          ; Comparator out hi/low?
             jz     C1                      ;
;            Adjust and integrate DAC Pulse ;
```

```
Test_DAC     bit.b  #CAOUT,&CACTL2          ; Comparator hi/low?
             jnc    Low1                    ; Jump --> Low

High         bic.b  #DAC_Out,&P2OUT         ; Reset power to capacitor

             jmp    Meas_                   ;

Low1         bis.b  #DAC_Out,&P2OUT         ; Set power to capacitor

             setc                           ;

             dadc    ADCData                ; BCD increment ADCData

Meas_        dec    R15                     ; Decrement Loop Count

             jnz    Test_DAC                ; Measurement Loop Over?

             bic.b  #DAC_Out,&P2OUT         ; Reset power to capacitor

             xor.b  #CAEX,&CACTL1           ; invert comparator terminals

             ret                            ; Return from Subroutine

                                            ;

;-----------------------------------------------------------------------------

TX_2_PC;     Send a CR and 16-bit word ADCData (R11) to PC/user

;-----------------------------------------------------------------------------

             mov    #CR,RXTXData            ; CR to UART buffer

             call   #TX_Byte                ; CR --> PC/user

;

;            FALL TO SUBROUTINE BELOW

                                            ;

;-----------------------------------------------------------------------------

TX_Word_ASCII; TX Word from ADCData as four ASCII bytes

;-----------------------------------------------------------------------------

             swpb   ADCData                 ; ADCData = 3412

             call   #TX_Byte_ASCII          ;

             swpb   ADCData                 ; ADCData = 1234

                                            ;

;-----------------------------------------------------------------------------

TX_Byte_ASCII; TX Byte from ADCData in two ASCII bytes

;-----------------------------------------------------------------------------

             push   ADCData                 ; transmit ..x. of value

             call   #NUM_ASCIR              ;

             push   ADCData                 ; transmit ..x. of value

             call   #NUM_ASCIA              ;
             ret                              ; Return from subroutine
                                              ;
NUM_ASCIR;   Convert Numbers 0..f into ASCII left aligned 2(SP)
             rrc     2(SP)                    ; 1. and 3. pass
             rrc     2(SP)                    ;
             rrc     2(SP)                    ;
             rrc     2(SP)                    ;
                                              ;
NUM_ASCIA    and     #0fh,2(SP)               ; 2. and 4. pass
             add     #030h,2(SP)              ;
             cmp     #03ah,2(SP)              ;
             jlo     NUM_End                  ;
             add     #039,2(SP)               ;
NUM_End      mov     2(SP),RXTXData           ; load transmit buffer, FALL
             mov     @SP+,0(SP)               ; Clean-up w\ return address TOS
                                              ;
;-----------------------------------------------------------------------------
TX_Byte;     Subroutine that TX Byte from RXTXData Buffer.
;-----------------------------------------------------------------------------
             mov     #TX_Count,BitCnt         ; TX_Count --> Branch Pointer
             push    &TAR                     ; Current TA Count
             add     #Bitime,0(SP)            ; Offset to next bit
             pop     &CCR0                    ; Time to next bit in CCR0
             mov     #OUTMOD2+OUTMOD0+CCIE,&CCTL0  ; TXD = Space Start Bit
TX_Wait      bit     #CCIE,&CCTL0             ; Wait for TX completion
             jnz     TX_Wait                  ;
             ret                              ; Return from subroutine
                                              ;
;-------------------------------------------------------------------------------
TA0_ISR  ;  CCR0/UART ISR:       RXTXData Buffer holds UART Data.
;-------------------------------------------------------------------------------
             add      #Bitime,&CCR0           ; Bitime till next bit
             mov      @BitCnt+,PC             ; Branch To Routine
                                              ;
TX_Bit       rra.b    RXTXData                ; LSB is shifted to carry
```

```
             jc      TX_Mark                 ; Jump if bit = 1
TX_Space     bis     #OUTMOD2,&CCTL0         ; TX Space
             reti
TX_Comp      bic     #CCIE,&CCTL0            ; All Bits RXed, Disable Interrupt
TX_Mark      bic     #OUTMOD2,&CCTL0         ; TX Mark
             reti                            ;
                                             ;
             even                            ;
TX_Count     DW     TX_Bit                   ; TX First Data Bit
             DW     TX_Bit                   ;
             DW     TX_Bit                   ;
             DW     TX_Bit                   ;
             DW     TX_Bit                   ;
             DW     TX_Bit                   ;
             DW     TX_Bit                   ;
             DW     TX_Bit                   ;
             DW     TX_Mark                  ; TX Stop Bit= Mark
TX_End       DW     TX_Comp                  ; TX Complete and Complete
                                             ;
;-----------------------------------------------------------------------------
            RSEG    INTVEC                   ; MSP430x11x1 interrupt vectors
;-----------------------------------------------------------------------------
             DW     RESET                    ; no source
             DW     RESET                    ; no source
             DW     RESET                    ; P1.x
             DW     RESET                    ; P2.x
             DW     RESET                    ; no source
             DW     RESET                    ; no source
             DW     RESET                    ; no source
             DW     RESET                    ; no source
             DW     RESET                    ; Timer_AX
             DW     TA0_ISR                  ; Timer_A0
             DW     RESET                    ; Watchdog/Timer, Timer mode
             DW     RESET                    ; Comparator_A
             DW     RESET                    ; no source
```

```
                DW      RESET                       ; no source
                DW      RESET                       ; NMI, Osc. fault
                DW      RESET                       ; POR, ext. Reset, Watchdog
                END
```

References

1. *MSP430x11x1 Data Sheet* (SLAS241C)
2. *MSP430x1xx Family Users Guide (SLAU049*)
3. *MSP430 Application Report, Lutz Bierl* (SLAA024)
4. *Implementing a UART Function with Timer_A3, Mark Buccini, (SLAA078)*