Module 11 – Clocked ADC with output stream – **Whole picture**

SetUp:
    Init CLK, Timer0_A3, ADC, UART

Loop:
    Enable Interrupts, Low Power Mode (Standby)

1

Timer0_A0 ISR
    Start ADC sample on ADC CH1

2

ADC ISR
    Create Output String of ADC Value
    Start TX-IE

3

TX ISR
    Get next char
    End-Of-String?
        No – RETI
        Yes- Turn off TX-IE

Develop incrementally –

4 programs

Each one builds on the other

Start simple

Each one increases complexity

- RT_ADC1 – Print alphabet, 10 char/second
- RT_ADC2 – Output fixed string use TX Interrupt
- RT_ADC3 – Output strings with varying content – Sprintf
- RT_ADC4- Put whole thing together

  TA0 Interrupt – Start ADC

  ADC Interrupt – Create output string and start Transmit

  TX Interrupt – Output until End-of-String

RT_ADC1 – Print alphabet, 10 chars/second

SetUp:

GPIO - Enable RED LED

Clock System – MCLK & SMCLK = 1 MHz

Timer0 - CCR0 set to 10/second

UART – 9600 Baud, No Parity, 1 Stop Bit

Loop:

Interrupt Enable, Low Power Mode 0

Timer0_A0_Interrupt:

Putchar and increment character value (timerValue)

Toggle RED LED

If (End of alphabet) Print CR, LF

Putchar:

poll until TX is empty then output char

RT_ADC1

1. Stop watchdog timer
2. Configure GPIO Setup
3. Disable the GPIO power-on default
4. Clock System Setup
5. set BAUD rate
6. Timer0_A3 Setup
7. Go to sleep
8. Timer A0 ISR: print alphabet 10 /second
9. putchar: output single char

```
21  //
22  //   Working & Energia  - H Watson  20180731
23  //
24  //  sketch_RT_ADC1.ino
25  //*************************************************
26  #include <msp430.h>
27
28
29  int putchar(int TxByte);  // output char
30  void UARTSetup (void);
31
32  volatile unsigned char timerValue=0x41 ; // Upper
33
34
35  int main(void)
36  {
37      WDTCTL = WDTPW | WDTHOLD;                    // Stop watchdog timer
38
39      // Configure GPIO Setup
40      // RED LED
41      P1DIR |= BIT0;                               // Set P1.0 as output
42      P1OUT |= BIT0;                               // P1.0 high
43
44
45      // Disable the GPIO power-on default high-impedance mode to activate
46      // previously configured port settings
47      PM5CTL0 &= ~LOCKLPM5;
48
```

1

2

3

```
49
50  //   Clock System Setup  ACLK = 32786, MCLK = SMCLK = 1MHz
51       __bis_SR_register(SCG0);                          // disable FLL
52      CSCTL3 |= SELREF__REFOCLK;                         // Set REFOCLK as FLL reference source
53      CSCTL0 = 0;                                        // clear DCO and MOD registers
54      CSCTL1 &= ~(DCORSEL_7);                            // Clear DCO frequency select bits first
55      CSCTL1 |= DCORSEL_3;                               // Set DCOCLK = 8MHz
56      CSCTL2 = FLLD_1 + 121;                             // FLLD = 1, DCODIV = 4MHz
57      __delay_cycles(3);
58      __bic_SR_register(SCG0);                           // enable FLL
59      while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));         // Poll until FLL is locked
60      CSCTL4 = SELMS__DCOCLKDIV | SELA__XT1CLK;          // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
61      CSCTL5 |= DIVM1;                                  // SMCLK = MCLK = DCODIV/2 = 1MHz, by default
62
63
64      UARTSetup();        // set BAUD rate
65
66
67      // Timer0_A3 Setup  ISR 10/second:
68      TA0CCTL0 |= CCIE;                       // TACCR0 interrupt enabled
69      TA0EX0 |= TAIDEX_3;                     // SMCLK/8/4 = 31250 Hz
70       TA0CCR0 = 3125;                        // 10 per second
71      TA0CTL = TASSEL_2 | MC_1 | ID_3;              // SMCLK/8 = 125K , UP mode
72
73      // Go to sleep
74      __bis_SR_register(LPM0_bits | GIE);
75
76  }
```

```
77
78  // Timer A0 interrupt service routine
79  #pragma vector = TIMER0_A0_VECTOR
80  __interrupt void Timer_A (void)
81  {
82      P1OUT ^= BIT0;
83      // print ASCII alphabet 10 char/second
84      if(timerValue>=0x7B)
85      {
86        timerValue=0x41;
87        putchar(0x0D);  //CR
88        putchar(0x0A);  //LF
89      }
90      putchar((int)timerValue++);
91
92
93  }
94
```

ASCII Table

Decimal	Hex	Char	Decimal	Hex	Char
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z
91	5B	[	123	7B	{
92	5C	\	124	7C	|
93	5D	]	125	7D	}
94	5E	^	126	7E	~
95	5F	_	127	7F	[DEL]

```
95
96  void UARTSetup (void)
97  {
98    // Configure UART pins
99    P1SEL0 |= BIT4 | BIT5;                    // set 2-UART pin as second function
100   // Configure UART
101   UCA0CTLW0 |= UCSWRST;                     // reset UART
102   UCA0CTLW0 |= UCSSEL__SMCLK;               // use SMCLK input
103     UCA0BR0 = 104;                          // 1MHz SMCLK/9600 BAUD
104     UCA0MCTLW = 0x1100; //                  // remainder of Baud Rate
105   UCA0CTLW0 &= ~UCSWRST;
106 }
107
108 int putchar(int TxByte)
109 {
110       while(!(UCA0IFG&UCTXIFG));
111       UCA0TXBUF = TxByte;
112       return 1;
113 }
114
```

RT_ADC2 – Print fixed string 10/second with TXISR

SetUp:
- GPIO - Enable RED LED
- Clock System – MCLK & SMCLK = 1 MHz
- Timer0 - CCR0 set to 10/second
- UART Setup – 9600 Baud, No Parity, 1 Stop Bit

Loop: Sleep - Interrupt Enable, Low Power Mode 0

Timer0_A0_Interrupt:
- Toggle RED LED
- UARTPutString – Send whole string to TX

UARTPutString:
- TxPtr = String Location
- Load first TX char and TXIE

USCI_A0_ISR:
- IF(TxPtr points to EOS) Turn TXIE off
- ELSE Output TX char and Increment TxPtr

RT_ADC2

1. Stop watchdog timer
2. Configure GPIO Setup
3. Disable the GPIO power-on default
4. Clock System Setup
5. set BAUD rate
6. Timer0_A3 Setup: 10/sec
7. Go to sleep
8. Timer A0 ISR: Start string print
9. USCI_A0_ISR: If EOS, Turn off TXIE
   else, output next char
10. Load Char & Set TXIE

```
28  //  sketch_RT_ADC2.ino
29  //***************************************************************
30  #include <msp430.h>
31
32  void UARTPutString(const char* strptr); // begin output of
33  void UARTSetup (void);
34
35  const char* TxPtr ;
36
37  int main(void)
38  {
1   39      WDTCTL = WDTPW | WDTHOLD;                   // Stop watchdog timer
40
41      // Configure GPIO Setup
42      // RED LED
2   43      P1DIR |= BIT0;                              // Set P1.0 as output
44      P1OUT |= BIT0;                              // P1.0 high
45
46
47      // Disable the GPIO power-on default high-impedance mode to activate
3   48      // previously configured port settings
49      PM5CTL0 &= ~LOCKLPM5;
50
```

```
51
52  //   Clock System Setup  ACLK = 32786, MCLK = SMCLK = 1MHz
53       __bis_SR_register(SCG0);                          // disable FLL
54      CSCTL3 |= SELREF__REFOCLK;                         // Set REFOCLK as FLL reference source
55      CSCTL0 = 0;                                        // clear DCO and MOD registers
56      CSCTL1 &= ~(DCORSEL_7);                            // Clear DCO frequency select bits first
57      CSCTL1 |= DCORSEL_3;                               // Set DCOCLK = 8MHz
58      CSCTL2 = FLLD_1 + 121;                             // FLLD = 1, DCODIV = 4MHz
59      __delay_cycles(3);
60      __bic_SR_register(SCG0);                           // enable FLL
61      while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));         // Poll until FLL is locked
62      CSCTL4 = SELMS__DCOCLKDIV | SELA__XT1CLK;     // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
63      CSCTL5 |= DIVM1;                                  // SMCLK = MCLK = DCODIV/2 = 1MHz, by default
64
65
66      UARTSetup();        // set BAUD rate
67
68
69      // Timer0_A3 Setup  ISR 10/second:
70      TA0CCTL0 |= CCIE;                       // TACCR0 interrupt enabled
71      TA0EX0 |= TAIDEX_3;                     // SMCLK/8/4 = 31250 Hz
72       TA0CCR0 = 3125;                        // 10 per second
73      TA0CTL = TASSEL_2 | MC_1 | ID_3;        // SMCLK/8 = 125K , UP mode
74
75      // Go to Standby
76      __bis_SR_register(LPM0_bits | GIE);
77
78  }
```

4

5

6

7

8

```
79
80 // Timer A0 interrupt service routine
81 #pragma vector = TIMER0_A0_VECTOR
82 __interrupt void Timer_A (void)
83 {
84     P1OUT ^= BIT0;
85     // print ASCII 10 strings / second
86     UARTPutString("This is the test string\n"); // begin output of string
87 }
88
```

```
89
90  #pragma vector=USCI_A0_VECTOR
91  __interrupt void USCI_A0_ISR(void)
92  {
93    switch(UCA0IV)
94    {
95      case USCI_NONE: break;
96      case USCI_UART_UCRXIFG:
97        while(!(UCA0IFG&UCTXIFG));
98        UCA0TXBUF = UCA0RXBUF;
99        __no_operation();
100       break;
101     case USCI_UART_UCTXIFG:
102         // load char value
103        // unsigned char testVal=*TxPtr++;
104       if(!(*TxPtr))  // if zero, then stop
105       {
106          UCA0IE &= ~UCTXIE;  // turn off interrupt
107       }
108       else
109       {
110          UCA0TXBUF = *TxPtr++ ;
111       }
112       break;
113     case USCI_UART_UCSTTIFG: break;
114     case USCI_UART_UCTXCPTIFG: break;
115     default: break;
116   }
117 }
118
```

Stop – End of String

```
118
119  void UARTSetup (void)
120  {
121    // Configure UART pins
122    P1SEL0 |= BIT4 | BIT5;                    // set 2-UART pin as second function
123    // Configure UART
124    UCA0CTLW0 |= UCSWRST;                     // reset UART
125    UCA0CTLW0 |= UCSSEL__SMCLK;               // use SMCLK input
126      UCA0BR0 = 104;                          // 1MHz SMCLK/9600 BAUD
127      UCA0MCTLW = 0x1100; //                  // remainder of Baud Rate
128    UCA0CTLW0 &= ~UCSWRST;
129
130     //UCA0IE |= UCRXIE;                      // Enable USCI_A0 RX interrupt
131  }
132
133  void UARTPutString(const char* strptr) // begin output of string
134  {
135    // load TxBuf with first char then enable interrupt
136      TxPtr = strptr;
137      UCA0TXBUF = *TxPtr++;  //load first, assume at least one char in buffer
138      UCA0IE |= UCTXIE;   // interrupt when transmitted  - ISR turns off when done
139  }
140
```

Start

RT_ADC3 – Sprint create string Values 10/second

SetUp:
- GPIO - Enable RED LED
- Clock System – MCLK & SMCLK = 1 MHz
- Timer0 - CCR0 set to 10/second
- UART Setup – 9600 Baud, No Parity, 1 Stop Bit

Loop: Sleep - Interrupt Enable, Low Power Mode 0

Timer0_A0_Interrupt:
- Toggle RED LED
- Sprintf OutStr & Count value
- UARTPutString – Send whole string to TX

UARTPutString:
- TxPtr = String Location
- Load first TX char and TXIE

USCI_A0_ISR:
- IF(TxPtr points to EOS) Turn TXIE off
- ELSE Output TX char and Increment TxPtr

```
28  //   Working & Energia  - H Watson  20180731
29  //
30  //  sketch_RT_ADC3.ino
31  //*************************************************************
32  #include <msp430.h>
33
34  void UARTPutString(const char* strptr); // begin output of s
35  void UARTSetup (void);
36
37  const char* TxPtr ;
38  char OutStr[50];  // buffer to hold output string
39  unsigned char Count;
40
41  int main(void)
42  {
1   43      WDTCTL = WDTPW | WDTHOLD;                   // Stop
44
45      // Configure GPIO Setup
46      // RED LED
2   47      P1DIR |= BIT0;                              // Set P1.0 as output
48      P1OUT |= BIT0;                              // P1.0 high
49
50
51      // Disable the GPIO power-on default high-impedance mode to activate
3   52      // previously configured port settings
53      PM5CTL0 &= ~LOCKLPM5;
54
```

1 Stop watchdog timer
2 Configure GPIO Setup
3 Disable the GPIO power-on default
4 Clock System Setup
5 set BAUD rate
6 Timer0_A3 Setup: 10/sec
7 Go to sleep
8 Timer A0 ISR: Sprintf to OutStr
Start string print
9 USCI_A0_ISR: If EOS, Turn off TXIE
else, output next char
10 Load Char & Set TXIE

```
55
56  //   Clock System Setup  ACLK = 32786, MCLK = SMCLK = 1MHz
57       __bis_SR_register(SCG0);                          // disable FLL
58      CSCTL3 |= SELREF__REFOCLK;                         // Set REFOCLK as FLL reference source
59      CSCTL0 = 0;                                        // clear DCO and MOD registers
60      CSCTL1 &= ~(DCORSEL_7);                            // Clear DCO frequency select bits first
61      CSCTL1 |= DCORSEL_3;                               // Set DCOCLK = 8MHz
62      CSCTL2 = FLLD_1 + 121;                             // FLLD = 1, DCODIV = 4MHz
63      __delay_cycles(3);
64      __bic_SR_register(SCG0);                           // enable FLL
65      while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));         // Poll until FLL is locked
66      CSCTL4 = SELMS__DCOCLKDIV | SELA__XT1CLK;     // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
67      CSCTL5 |= DIVM1;                                  // SMCLK = MCLK = DCODIV/2 = 1MHz, by default
68
69
70      UARTSetup();        // set BAUD rate
71
72
73      // Timer0_A3 Setup  ISR 10/second:
74      TA0CCTL0 |= CCIE;                       // TACCR0 interrupt enabled
75      TA0EX0 |= TAIDEX_3;                     // SMCLK/8/4 = 31250 Hz
76       TA0CCR0 = 3125;                        // 10 per second
77      TA0CTL = TASSEL_2 | MC_1 | ID_3;              // SMCLK/8 = 125K , UP mode
78
79      // go to Standby
80      __bis_SR_register(LPM0_bits | GIE);
81
82  }
83
```

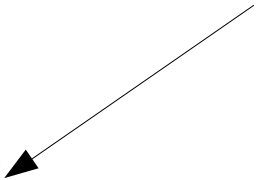
4

5

6

7

```
83
84  // Timer A0 interrupt service routine
85  #pragma vector = TIMER0_A0_VECTOR
86  __interrupt void Timer_A (void)
87  {
88      P1OUT ^= BIT0;
89      // print ASCII alphabet 10 char/second
90      if(!(UCA0IE & UCTXIE))
91      { // if flag is clear, means last string output is done
92        sprintf(OutStr,"The value of Count is %d \n",Count++);
93        UARTPutString(OutStr); // begin output of string
94      }
95  }
96
```



```
 97
 98  #pragma vector=USCI_A0_VECTOR
 99  __interrupt void USCI_A0_ISR(void)
100  {
101    switch(UCA0IV)
102    {
103      case USCI_NONE: break;
104      case USCI_UART_UCRXIFG:
105        while(!(UCA0IFG&UCTXIFG));
106        UCA0TXBUF = UCA0RXBUF;
107        __no_operation();
108        break;
109      case USCI_UART_UCTXIFG:
110          // load char value
111         // unsigned char testVal=*TxPtr++;
112          if(!(*TxPtr))  // if zero, then stop
113          {
114             UCA0IE &= ~UCTXIE;  // turn off interrupt
115          }
116          else
117          {
118             UCA0TXBUF = *TxPtr++ ;
119          }
120        break;
121      case USCI_UART_UCSTTIFG: break;
122      case USCI_UART_UCTXCPTIFG: break;
123      default: break;
124    }
125  }
```

```
126
127  void UARTSetup (void)
128  {
129    // Configure UART pins
130    P1SEL0 |= BIT4 | BIT5;                    // set 2-UART pin as second function
131    // Configure UART
132    UCA0CTLW0 |= UCSWRST;                     // reset UART
133    UCA0CTLW0 |= UCSSEL__SMCLK;               // use SMCLK input
134      UCA0BR0 = 104;                         // 1MHz SMCLK/9600 BAUD
135      UCA0MCTLW = 0x1100; //                 // remainder of Baud Rate
136    UCA0CTLW0 &= ~UCSWRST;
137
138     //UCA0IE |= UCRXIE;                     // Enable USCI_A0 RX interrupt
139  }
140
141  void UARTPutString(const char* strptr) // begin output of string
142  {
143    // load TxBuf with first char then enable interrupt
144      TxPtr = strptr;
145      UCA0TXBUF = *TxPtr++;  //load first, assume at least one char in buffer
146      UCA0IE |= UCTXIE;   // interrupt when transmitted  - ISR turns off when done
147  }
148
```

RT_ADC4 – Sprint ADC Values 10/second

SetUp:
- GPIO - Enable RED LED
- Clock System – MCLK & SMCLK = 1 MHz
- ADC Setup – 10 Bits, CH1, Interrupt when complete
- Timer0 - CCR0 set to 10/second
- UART Setup – 9600 Baud, No Parity, 1 Stop Bit

Loop: Sleep - Interrupt Enable, Low Power Mode 0

Timer0_A0_Interrupt:
- Toggle RED LED
- Start ADC Conversion

ADC_ISR:
- Sprintf OutStr of ADC_Result
- UARTPutString – TX whole string

UARTPutString:
- TxPtr = String Location
- Load first TX char and TXIE

USCI_A0_ISR:
- IF(TxPtr points to EOS) Turn TXIE off
- ELSE Output TX char and Increment TxPtr

RT_ADC4

1 Stop watchdog timer
2 Configure GPIO Setup
3 Disable the GPIO power-on default
4 Clock System Setup
5 set BAUD rate
6 ADC setup: CH1 & ADCIE
7 Timer0_A3 Setup: 10/sec
8 Go to sleep
9 Timer A0 ISR: Begin conversion
10 USCI_A0_ISR: If EOS, Turn off TXIE
else, output next char
11 ADC ISR: Sprintf ADC_RESULT UARTPutString
12 UARTPutString: Load Char & Set TXIE

```
33  //  sketch_RT_ADC4.ino
34  //*************************************************
35  #include <msp430.h>
36
37  void UARTPutString(const char* strptr); // begi
38  void UARTSetup (void);
39
40  const char* TxPtr ;
41  char OutStr[50];  // buffer to hold output stri
42  int ADC_Result;
43
44
45  int main(void)
46  {
1   47      WDTCTL = WDTPW | WDTHOLD;
48
49      // Configure GPIO Setup
50      // RED LED
2   51      P1DIR |= BIT0;                          // Set P1.0 as output
52      P1OUT |= BIT0;                          // P1.0 high
53
54
55      // Disable the GPIO power-on default high-impedance mode to activate
56      // previously configured port settings
3   57      PM5CTL0 &= ~LOCKLPM5;
```

4

```
59
60  //   Clock System Setup  ACLK = 32786, MCLK = SMCLK = 1MHz
61       __bis_SR_register(SCG0);                          // disable FLL
62      CSCTL3 |= SELREF__REFOCLK;                         // Set REFOCLK as FLL reference source
63      CSCTL0 = 0;                                        // clear DCO and MOD registers
64      CSCTL1 &= ~(DCORSEL_7);                            // Clear DCO frequency select bits first
65      CSCTL1 |= DCORSEL_3;                               // Set DCOCLK = 8MHz
66      CSCTL2 = FLLD_1 + 121;                             // FLLD = 1, DCODIV = 4MHz
67      __delay_cycles(3);
68      __bic_SR_register(SCG0);                           // enable FLL
69      while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));         // Poll until FLL is locked
70      CSCTL4 = SELMS__DCOCLKDIV | SELA__XT1CLK;    // set ACLK = XT1 = 32768Hz, DCOCLK as source
71      CSCTL5 |= DIVM1;                                  // SMCLK = MCLK = DCODIV/2 = 1MHz, by default
72
73
74      UARTSetup();        // set BAUD rate
75
```

5

```
76
77      // add ADC setup
78      // Configure ADC10
79      ADCCTL0 |= ADCSHT_2 | ADCON;                        // ADCON, S&H=16 ADC clks
80      ADCCTL1 |= ADCSHP;                                  // ADCCLK = MODOSC; sampling timer
81      ADCCTL2 |= ADCRES;                                  // 10-bit conversion results
82      ADCMCTL0 |= ADCINCH_1;                              // A1 ADC input select; Vref=AVCC
83      ADCIE |= ADCIE0;                                    // Enable ADC conv complete interrupt
84
85      // Timer0_A3 Setup  ISR 10/second:
86      TA0CCTL0 |= CCIE;                       // TACCR0 interrupt enabled
87      TA0EX0 |= TAIDEX_3;                     // SMCLK/8/4 = 31250 Hz
88       TA0CCR0 = 3125;                        // 10 per second
89      TA0CTL = TASSEL_2 | MC_1 | ID_3;              // SMCLK/8 = 125K , UP mode
90
91      // go to Standby
92      __bis_SR_register(LPM0_bits | GIE);
93
94  }
95
96  // Timer A0 interrupt service routine
97  #pragma vector = TIMER0_A0_VECTOR
98  __interrupt void Timer_A (void)
99  {
100     P1OUT ^= BIT0;
101     //  begin conversion
102      ADCCTL0 |= ADCENC | ADCSC;          // Sampling and conversion start
103
104 }
105
```

6

7

8

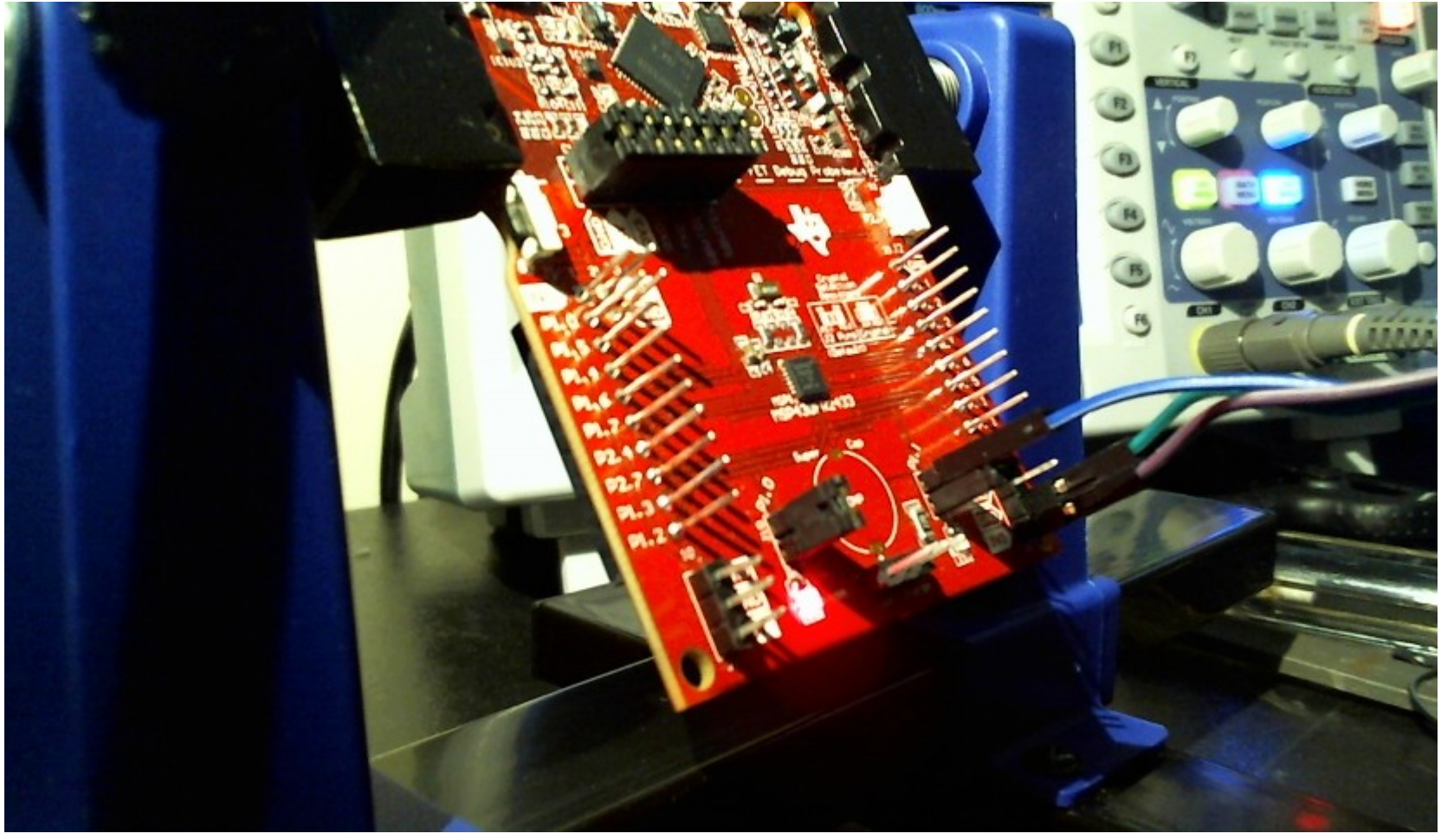
9

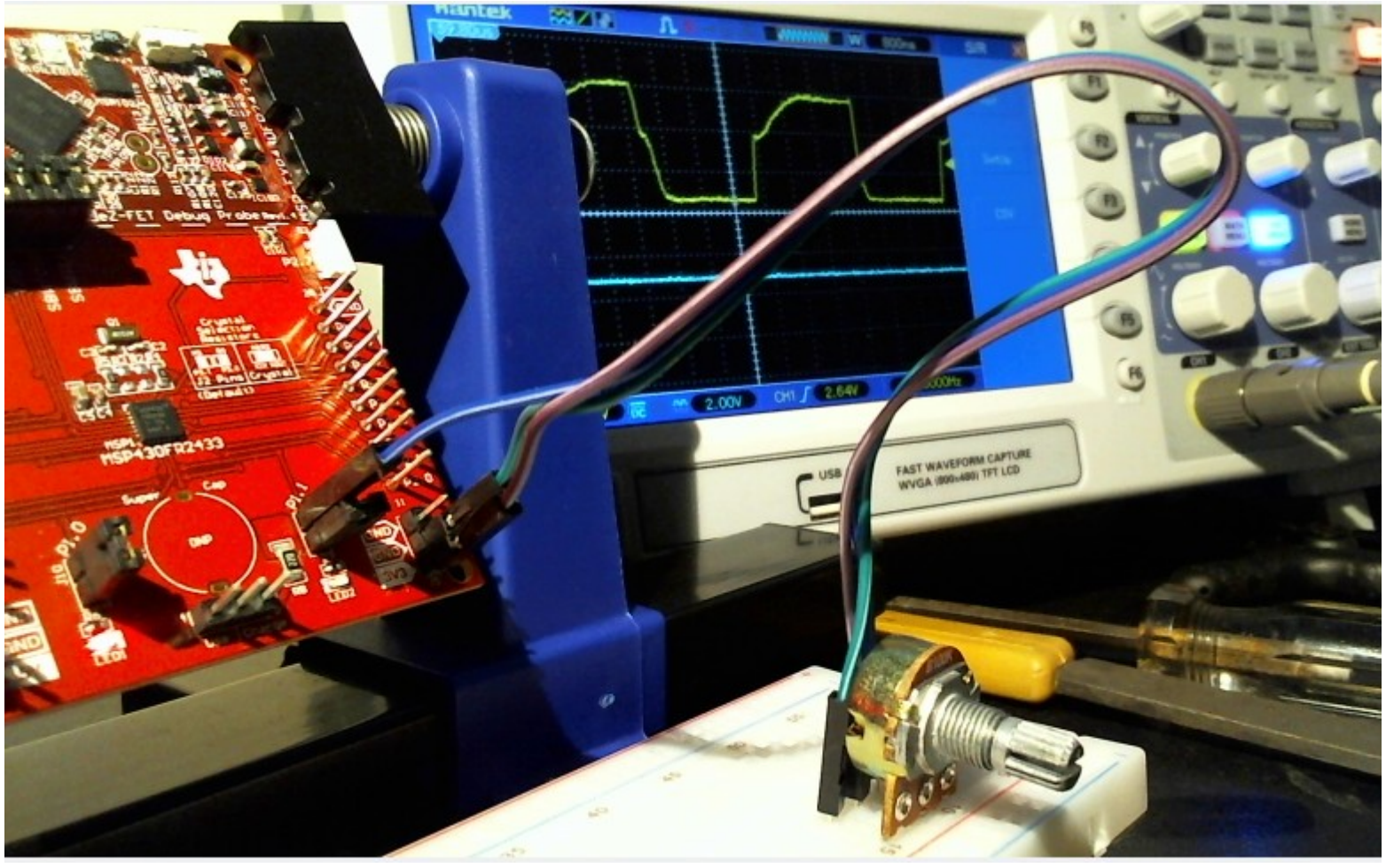
```
105
106  #pragma vector=USCI_A0_VECTOR
107  __interrupt void USCI_A0_ISR(void)
108  {
109    switch(UCA0IV)
110    {
111      case USCI_NONE: break;
112      case USCI_UART_UCRXIFG:
113        while(!(UCA0IFG&UCTXIFG));
114        UCA0TXBUF = UCA0RXBUF;
115        __no_operation();
116        break;
117      case USCI_UART_UCTXIFG:
118          // load char value
119         // unsigned char testVal=*TxPtr++;
120          if(!(*TxPtr))  // if zero, then stop
121          {
122             UCA0IE &= ~UCTXIE;  // turn off interrupt
123          }
124          else
125          {
126             UCA0TXBUF = *TxPtr++ ;
127          }
128        break;
129      case USCI_UART_UCSTTIFG: break;
130      case USCI_UART_UCTXCPTIFG: break;
131      default: break;
132    }
133  }
```

```
135  // ADC interrupt service routine
136  #pragma vector=ADC_VECTOR
137  __interrupt void ADC_ISR(void)
138  {
139      switch(ADCIV)
140      {
141          case ADCIV_NONE:
142              break;
143          case ADCIV_ADCOVIFG:
144              break;
145          case ADCIV_ADCTOVIFG:
146              break;
147          case ADCIV_ADCHIIFG:
148              break;
149          case ADCIV_ADCLOIFG:
150              break;
151          case ADCIV_ADCINIFG:
152              break;
153          case ADCIV_ADCIFG:
154              ADC_Result = ADCMEM0;
155              // last string has to be complete or trouble here - no blocking allowed
156              //  worst case, will overwrite part of string being output
157              sprintf(OutStr,"%d\n",ADC_Result);
158              UARTPutString(OutStr); // begin output of string
159
160
161              //__bic_SR_register_on_exit(LPM0_bits);           // Clear CPUOFF bit from LPM0
162              break;
163          default:
164              break;
165      }
166  }
```

End of Conversion

```
170
171  void UARTSetup (void)
172  {
173    // Configure UART pins
174    P1SEL0 |= BIT4 | BIT5;                    // set 2-UART pin as second function
175    // Configure UART
176    UCA0CTLW0 |= UCSWRST;                     // reset UART
177    UCA0CTLW0 |= UCSSEL__SMCLK;               // use SMCLK input
178      UCA0BR0 = 104;                         // 1MHz SMCLK/9600 BAUD
179      UCA0MCTLW = 0x1100; //                 // remainder of Baud Rate
180    UCA0CTLW0 &= ~UCSWRST;
181
182     //UCA0IE |= UCRXIE;                     // Enable USCI_A0 RX interrupt
183  }
184
185  void UARTPutString(const char* strptr) // begin output of string
186  {
187    // load TxBuf with first char then enable interrupt
188      TxPtr = strptr;
189      UCA0TXBUF = *TxPtr++;  //load first, assume at least one char in buffer
190      UCA0IE |= UCTXIE;   // interrupt when transmitted  - ISR turns off when done
191  }
192
```




eZ-FET Debug
MSP430FR2433
FAST WAVEFORM CAPTURE
WVGA (800x480) TFT LCD


/dev/ttyACM1
Send
669
670
670
670
669
670
669
670
669
670
669
669
670
670
670
670
669
Autoscroll
Both NL & CR
9600 baud