

ex.c.c

```
1
2
3 int main(void)
4 {
5     WDTCTL = WDTPW+WDTHOLD;           // Stop watchdog timer
6     P7SEL |= 0x03;                     // Port select XT1
7
8     do
9     {
10         UCSCCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);
11                                     // Clear XT2,XT1,DCO fault flags
12         SFRIFG1 &= ~OIFIG;           // Clear fault flags
13         __delay_cycles(100000);       // Delay for Osc to stabilize
14     }while (SFRIFG1&OIFIG);           // Test oscillator fault flag
15
16     P1OUT = 0x000;                     // P1.0/1 setup for LED output
17     P1DIR |= BIT0+BIT1;                //
18     P3SEL |= BIT4+BIT5;                // P3.4,5 UART option select
19
20     UCA0CTL1 |= UCSWRST;                // **Put state machine in reset**
21     UCA0CTL1 |= UCSSEL_1;              // CLK = ACLK
22     UCA0BR0 = 0x03;                    // 32k/9600 - 3.41
23     UCA0BR1 = 0x00;                    //
24     UCA0MCTL = 0x06;                   // Modulation
25     UCA0CTL1 &= ~UCSWRST;              // **Initialize USCI state machine**
26     UCA0IE |= UCTXIE + UCRXIE;         // Enable USCI_A0 TX/RX interrupt
27
28     __bis_SR_register(LPM3_bits + GIE); // Enter LPM3 w/ interrupts enabled
29     __no_operation();                  // For debugger
30 }
31
32
33 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
34 #pragma vector=USCI_A0_VECTOR
35 __interrupt void USCI_A0_ISR(void)
36 #elif defined(__GNUC__)
37 void __attribute__((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
38 #else
39 #error Compiler not supported!
40 #endif
41 {
42     unsigned char tx_char;
43
44     switch(__even_in_range(UCA0IV,4))
45     {
46     case 0: break;                      // Vector 0 - no interrupt
47     case 2:                             // Vector 2 - RXIFG
48         P1OUT = UCA0RXBUF;             // RXBUF1 to TXBUF1
```

```
49         break;
50     case 4:                             // Vector 4 - TXIFG
51         __delay_cycles(5000);           // Add small gap between TX'ed bytes
52         tx_char = P1IN;
53         tx_char = tx_char >> 4;
54         UCA0TXBUF = tx_char;            // Transmit character
55         break;
56     default: break;
57     }
58 }
59
60
61
62
63 int main(void)
64 {
65     WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
66     P3SEL = 0x30;                       // P3.4,5 = USCI_A0 TXD/RXD
67     UCA0CTL1 |= UCSWRST;                 // **Put state machine in reset**
68     UCA0CTL1 |= UCSSEL_2;                // SMCLK
69     UCA0BR0 = 6;                         // 1MHz 9600 (see User's Guide)
70     UCA0BR1 = 0;                         // 1MHz 9600
71     UCA0MCTL = UCBRS_0 + UCBRF_13 + UCOS16; // ModIn UCBRSx=0, UCBRFx=0,
72                                     // over sampling
73     UCA0CTL1 &= ~UCSWRST;                // **Initialize USCI state machine**
74     UCA0IE |= UCRXIE;                    // Enable USCI_A0 RX interrupt
75
76     __bis_SR_register(LPM0_bits + GIE); // Enter LPM0, interrupts enabled
77     __no_operation();                    // For debugger
78 }
79
80 // Echo back RXed character, confirm TX buffer is ready first
81 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
82 #pragma vector=USCI_A0_VECTOR
83 __interrupt void USCI_A0_ISR(void)
84 #elif defined(__GNUC__)
85 void __attribute__((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
86 #else
87 #error Compiler not supported!
88 #endif
89 {
90     switch(__even_in_range(UCA0IV,4))
91     {
92     case 0: break;                       // Vector 0 - no interrupt
93     case 2:                             // Vector 2 - RXIFG
94         while (!(UCA0IFG&UCTXIFG));      // USCI_A0 TX buffer ready?
95         UCA0TXBUF = UCA0RXBUF;           // TX → RXed character
96         break;
97     case 4: break;                       // Vector 4 - TXIFG
98     default: break;
```

```

99     }
100 }
101
102
103
104 #include <msp430.h>
105
106 unsigned char MST_Data,SLV_Data;
107
108 int main(void)
109 {
110     WDTCTL = WDTPW+WDTHOLD;           // Stop watchdog timer
111
112     P1OUT |= 0x02;                     // Set P1.0 for LED
113                                         // Set P1.1 for slave reset
114     P1DIR |= 0x03;                     // Set P1.0-2 to output direction
115     P3SEL |= 0x31;                     // P3.5,4,0 option select
116
117     UCA0CTL1 |= UCSWRST;                // **Put state machine in reset**
118     UCA0CTL0 |= UCMST+UCSYNC+UCCKPL+UCMSB; // 3-pin, 8-bit SPI master
119                                         // Clock polarity high, MSB
120     UCA0CTL1 |= UCSSEL_2;               // SMCLK
121     UCA0BR0 = 0x02;                     // /2
122     UCA0BR1 = 0;                         //
123     UCA0MCTL = 0;                       // No modulation
124     UCA0CTL1 &= ~UCSWRST;               // **Initialize USCI state machine**
125     UCA0IE |= UCRXIE;                   // Enable USCI_A0 RX interrupt
126
127     P1OUT &= ~0x02;                     // Now with SPI signals initialized,
128     P1OUT |= 0x02;                       // reset slave
129
130     __delay_cycles(100);                 // Wait for slave to initialize
131
132     MST_Data = 0x01;                     // Initialize data values
133     SLV_Data = 0x00;                     //
134
135     while (!(UCA0IFG&UCTXIFG));          // USCI_A0 TX buffer ready?
136     UCA0TXBUF = MST_Data;                // Transmit first character
137
138     __bis_SR_register(LPM0_bits + GIE);  // CPU off, enable interrupts
139 }
140
141 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
142 #pragma vector=USCI_A0_VECTOR
143 __interrupt void USCI_A0_ISR(void)
144 #elif defined(__GNUC__)
145 void __attribute__((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
146 #else
147 #error Compiler not supported!
148 #endif

```

```

149 {
150     switch(__even_in_range(UCA0IV,4))
151     {
152         case 0: break;                    // Vector 0 - no interrupt
153         case 2:                            // Vector 2 - RXIFG
154             while (!(UCA0IFG&UCTXIFG));  // USCI_A0 TX buffer ready?
155
156             if (UCA0RXBUF==SLV_Data)      // Test for correct character RX'd
157                 P1OUT |= 0x01;           // If correct, light LED
158             else                            // If incorrect, clear LED
159                 P1OUT &= ~0x01;
160
161             MST_Data++;                    // Increment data
162             SLV_Data++;
163             UCA0TXBUF = MST_Data;          // Send next value
164
165             __delay_cycles(40);            // Add time between transmissions to
166                                             // make sure slave can process
167             break;
168         case 4: break;                    // Vector 4 - TXIFG
169         default: break;
170     }
171 }
172
173
174 #include <msp430.h>
175
176 unsigned char RXData;
177 unsigned char RXCompare;
178
179 int main(void)
180 {
181     WDTCTL = WDTPW + WDTHOLD;             // Stop WDT
182     P1OUT &= ~0x01;                         // P1.0 = 0
183     P1DIR |= 0x01;                         // P1.0 output
184     P3SEL |= 0x06;                         // Assign I2C pins to USCI_B0
185     UCB0CTL1 |= UCSWRST;                   // Enable SW reset
186     UCB0CTL0 = UCMST + UCMODE_3 + UCSYNC;  // I2C Master, synchronous mode
187     UCB0CTL1 = UCSSEL_2 + UCSWRST;         // Use SMCLK
188     UCB0BR0 = 12;                          // fSCL = SMCLK/12 = ~100kHz
189     UCB0BR1 = 0;
190     UCB0I2CSA = 0x48;                       // Slave Address is 048h
191     UCB0CTL1 &= ~UCSWRST;                  // Clear SW reset, resume operation
192     UCB0IE |= UCRXIE;                      // Enable RX interrupt
193     RXCompare = 0x0;                        // Used to check incoming data
194
195     while (1)
196     {
197         while (UCB0CTL1 & UCTXSTP);        // Ensure stop condition got sent

```

```

198     UCB0CTL1 |= UCTXSTT;           // I2C start condition
199     while(UCB0CTL1 & UCTXSTT);     // Start condition sent?
200     UCB0CTL1 |= UCTXSTP;           // I2C stop condition
201
202     __bis_SR_register(LPM0_bits + GIE); // Enter LPM0, enable interrupts
203     __no_operation();               // For debugger
204
205     if (RXData != RXCompare)        // Trap CPU if wrong
206     {
207         P1OUT |= 0x01;              // P1.0 = 1
208     }
209
210     RXCompare++;                    // Increment correct RX value
211 }
212 }
213
214 // USCI_B0 Data ISR
215 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
216 #pragma vector = USCI_B0_VECTOR
217 __interrupt void USCI_B0_ISR(void)
218 #elif defined(__GNUC__)
219 void __attribute__((interrupt(USCI_B0_VECTOR))) USCI_B0_ISR (void)
220 #else
221 #error Compiler not supported!
222 #endif
223 {
224     switch(__even_in_range(UCB0IV,12))
225     {
226     case 0: break;                  // Vector 0: No interrupts
227     case 2: break;                  // Vector 2: ALIFG
228     case 4: break;                  // Vector 4: NACKIFG
229     case 6: break;                  // Vector 6: STTIFG
230     case 8: break;                  // Vector 8: STPIFG
231     case 10: break;                 // Vector 10: RXIFG
232         RXData = UCB0RXBUF;         // Get RX data
233         __bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
234         break;
235     case 12: break;                 // Vector 12: TXIFG
236     default: break;
237     }
238 }
239
240
241
242 #include "msp430.h"
243 #define SMCLK_115200 0
244 #define SMCLK_9600 1
245 #define ACLK_9600 2
246
247 #define UART_MODE SMCLK_115200//ACLK_9600//

```

```

248
249 void initUART()
250 {
251     // Configure USCI_A0 for UART mode
252     UCA0CTLW0 = UCSWRST;           // Put eUSCI in reset
253     #if UART_MODE == SMCLK_115200
254
255         UCA0CTLW0 |= UCSSEL__SMCLK; // CLK = SMCLK
256         // Baud Rate calculation
257         // 16000000/(16*115200) = 8.6805
258         // Fractional portion = 0.6805
259         // Use Table 24-5 in Family User Guide
260         UCA0BR0 = 8;                // 16000000/16/9600
261         UCA0BR1 = 0x00;
262         UCA0MCTL |= UCOS16 | UCBRF_11 | UCBRS_0;
263
264     #elif UART_MODE == SMCLK_9600
265
266         UCA0CTLW0 |= UCSSEL__SMCLK; // CLK = SMCLK
267         // Baud Rate calculation
268         // 16000000/(16*9600) = 104.1667
269         // Fractional portion = 0.1667
270         // Use Table 24-5 in Family User Guide
271         UCA0BR0 = 104;               // 16000000/16/9600
272         UCA0BR1 = 0x00;
273         UCA0MCTL |= UCOS16 | UCBRF_3 | UCBRS_0;
274
275     #elif UART_MODE == ACLK_9600
276
277         UCA0CTLW0 |= UCSSEL__ACLK;   // CLK = ACLK
278         // Baud Rate calculation
279         // 32768/(9600) = 3.4133
280         // Fractional portion = 0.4133
281         // Use Table 24-5 in Family User Guide
282         UCA0BR0 = 3;                 // 32768/9600
283         UCA0BR1 = 0x00;
284         UCA0MCTL |= UCBRS_3;         // 0x0300 is UCBRSx = 0x03
285
286     #else
287         #error "Please specify baud rate to 115200 or 9600"
288     #endif
289
290     UCA0CTLW0 &= ~UCSWRST;           // Initialize eUSCI
291     UCA0IE |= UCRXIE;                // Enable USCI_A0 RX interrupt
292 }
293
294 //*****
295 // Device Initialization *****
296 //*****
297

```

```

298 void initGPIO()
299 {
300     P3SEL = BIT4 | BIT5;           // P3.4,5 = USCI_A0 TXD/RXD
301     P7SEL |= BIT0 | BIT1;         // Select XT1
302 }
303
304 void initClockTo16MHz()
305 {
306     UCSCTL3 |= SELREF_2;           // Set DCO FLL reference = REFO
307     UCSCTL4 |= SELA_0;             // Set ACLK = XT1CLK
308     __bis_SR_register(SCG0);       // Disable the FLL control loop
309     UCSCTL0 = 0x0000;             // Set lowest possible DCOx, MODx
310     UCSCTL1 = DCORSEL_5;          // Select DCO range 16MHz operation
311     UCSCTL2 = FLLD_0 + 487;       // Set DCO Multiplier for 16MHz
312                                     // (N + 1) * FLLRef = Fdco
313                                     // (487 + 1) * 32768 = 16MHz
314                                     // Set FLL Div = fDCOCLK
315     __bic_SR_register(SCG0);       // Enable the FLL control loop
316
317     // Worst-case settling time for the DCO when the DCO range bits have been
318     // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
319     // UG for optimization.
320     // 32 x 32 x 16 MHz / 32,768 Hz = 500000 = MCLK cycles for DCO to settle
321     __delay_cycles(500000);
322     // Loop until XT1,XT2 & DCO fault flag is cleared
323     do
324     {
325         UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG); // Clear XT2,XT1,DCO fault
326         flags
327         SFRIFG1 &= ~OIFG;           // Clear fault flags
328     }while (SFRIFG1&OIFG);         // Test oscillator fault flag
329
330 bool increaseVCoreToLevel2()
331 {
332     uint8_t level = 2;
333     uint8_t actlevel;
334     bool status = true;
335
336     //Set Mask for Max. level
337     level &= PMMCOREV_3;
338
339     //Get actual VCore
340     actlevel = PMMCTL0 & PMMCOREV_3;
341
342     //step by step increase or decrease
343     while((level != actlevel) && (status == true))
344     {
345         if(level > actlevel)
346         {

```

```

347             status = setVCoreUp(++actlevel);
348         }
349     }
350
351     return (status);
352 }
353
354
355 int main(void)
356 {
357     WDTCTL = WDTPW | WDTHOLD;     // Stop Watchdog
358
359     initGPIO();
360     increaseVCoreToLevel2();
361     initClockTo16MHz();
362     initUART();
363
364     #if UART_MODE == ACLK_9600
365         __bis_SR_register(LPM3_bits + GIE); // Since ACLK is source, enter LPM3,
366         interrupts enabled
367     #else
368         __bis_SR_register(LPM0_bits + GIE); // Since SMCLK is source, enter LPM0,
369         interrupts enabled
370     #endif
371     __no_operation();             // For debugger
372
373     // UART rx int
374     #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
375     #pragma vector=USCI_A0_VECTOR
376     __interrupt void USCI_A0_ISR(void)
377     #elif defined(__GNUC__)
378     void __attribute__((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
379     #else
380     #error Compiler not supported!
381     #endif
382     {
383         switch(__even_in_range(UCA0IV,4))
384         {
385             case 0:break;           // Vector 0 - no interrupt
386             case 2:                 // Vector 2 - RXIFG
387                 while (!(UCA0IFG & UCTXIFG)); // USCI_A0 TX buffer ready?
388                 UCA0TXBUF = UCA0RXBUF;       // TX → RXed character
389                 break;
390             case 4:break;           // Vector 4 - TXIFG
391             default: break;
392         }
393     }
394

```