Tutorial 4: UART Communication and Watchdog Timer

TNE097 Micro Computer Systems

Naveen Venkategowda

2025-12-04

LINKÖPING UNIVERSITY

MSP430 Universal Serial Communication Interface (USCI)

Setting Prescaler and Modulator in C code

```
UCA0CTL1 |= UCSWRST;          // Set UCSWRST
UCA0CTL1 |= UCSSEL_2;         // SMCLK
UCA0BR0 = x;                  // Least significant byte of divider
UCA0BR1 = y;                  // Most significant byte of divider
UCA0MCTL = UCBRS_1;           // Modulation UCBRSx = 1
UCA0CTL1 &= ~UCSWRST;         // Initialize USCI state machine
IE2 |= UCA0RXIE;              // Enable USCI_A0 RX interrupt
```

```
#define UCBRS2          (0x08)    /* USCI Second Stage Modulation Select 2 */
#define UCBRS1          (0x04)    /* USCI Second Stage Modulation Select 1 */
#define UCBRS0          (0x02)    /* USCI Second Stage Modulation Select 0 */

#define UCBRS_0         (0x00)    /* USCI Second Stage Modulation: 0 */
#define UCBRS_1         (0x02)    /* USCI Second Stage Modulation: 1 */
#define UCBRS_2         (0x04)    /* USCI Second Stage Modulation: 2 */
#define UCBRS_3         (0x06)    /* USCI Second Stage Modulation: 3 */
#define UCBRS_4         (0x08)    /* USCI Second Stage Modulation: 4 */
#define UCBRS_5         (0x0A)    /* USCI Second Stage Modulation: 5 */
#define UCBRS_6         (0x0C)    /* USCI Second Stage Modulation: 6 */
#define UCBRS_7         (0x0E)    /* USCI Second Stage Modulation: 7 */
```

Setting Prescaler and Modulator in C

```
UCA0CTL1 |= UCSWRST;       // Set UCSWRST
UCA0CTL1 |= UCSSEL_2;     // SMCLK
UCA0BR0 = x;          // Least significant byte of divider
UCA0BR1 = y;          // Most significant byte of divider
UCA0MCTL = UCBRS_1;       // Modulation UCBRSx = 1
UCA0CTL1 &= ~UCSWRST;     // Initialize USCI state machine
IE2 |= UCA0RXIE;          // Enable USCI_A0 RX interrupt
```

Configure UART step by step

- Initializing or Re-Configuring the USCI Module

1. Set UCSWRST: `UCA0CTL1 |= UCSWRST;`
2. Initialize all USCI registers with UCSWRST = 1 (including UCAxCTL1)
3. Configure IO ports for Tx/Rx pins
4. Configure UART registers
    - `UCA0BR0 = x;                // Least significant byte of divider`
    - `UCA0BR1 = y;                // Most significant byte of divider`
    - `UCA0MCTL = UCBRS_1;                    // Modulation UCBRSx = 1`
5. Clear UCSWRST via software: `UCA0CTL1 &= ~UCSWRST;`
6. Enable interrupts via UCAxRXIE and/or UCAxTXIE: `IE2 |= UCA0RXIE;`

Lab 4: Watchdog Timer and UART Communication

Interrupt Vector Table

```
/************************************************************
* Interrupt Vectors (offset from 0xFFE0)
************************************************************/
#define USCIAB0TX_VECTOR    (6 * 1u)  /* 0xFFEC USCI A0/B0 Transmit */
#define USCIAB0RX_VECTOR    (7 * 1u)  /* 0xFFEE USCI A0/B0 Receive */
#define TIMER0_A1_VECTOR    (8 * 1u)  /* 0xFFF0 Timer0_A CC1-2, TA0 */
#define TIMER0_A0_VECTOR    (9 * 1u)  /* 0xFFF2 Timer0_A CC0 */
#define WDT_VECTOR          (10 * 1u) /* 0xFFF4 Watchdog Timer */
```

Task 1: Wake the Dog!

1. Enable the watchdog timer
    - By default WDT is enabled
    - Use WDT password (WDTPW) every time WDT settings are modified
2. Configure the watchdog as
    - Watchdog mode: WDTTMSEL = 0
    - WDT clock source from SMCLK: WDTSSEL = 0
    - Clear WDT counter: WDTCNTCL =1

Task 1: Wake the Dog!

3. Find a place to "wake the dog" all the time before the watchdog timer fires and reset the device.
   - Use Timer 0 to measure time interval < 32ms
   - Use Timer 0 ISR to reset the WDT counter (WDTCNTCL =1)

```
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
      WDTCTL =   WDTPW + WDTCNTCL
}
```

Note: WDT expires after 32678 clock cycle = 32.6ms with SMCLK = 1MHz

Task 2: Watchdog Timer as Interval Timer

1. Configure the WDT as timer mode and resent WDT counter
   - `WDTCTL = WDTPW+WDTTMSEL+WDTCNTL;`
2. Select the WDT clock source from SMCLK
3. Enable the WDT interrupt
   - `IE1 |= WDTIE;`
4. Configures general purpose I/O ports
5. Enable general interrupts

Task 2: Watchdog Timer as Interval Timer

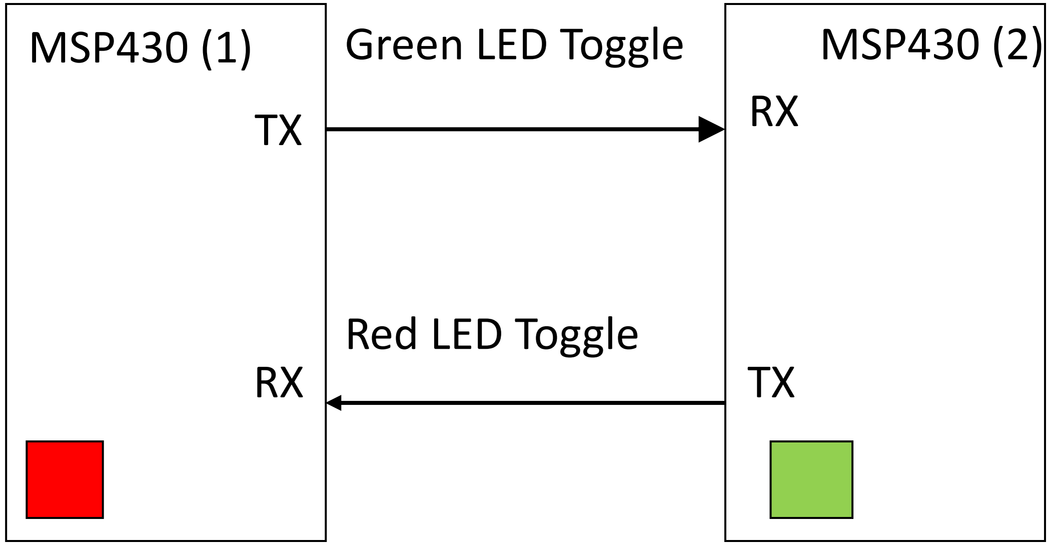
6. Handle the WDT interrupt in the interrupt service routine
   - Toggle P0_1 (Red led) in the interrupt service routine

```
// Watchdog Timer interrupt service routine
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
        // Toggle P1.0
}
```

7. Connect P0_1 with the oscilloscope, change the WDT interval to each of the four intervals and observe the output frequency from P0_1.

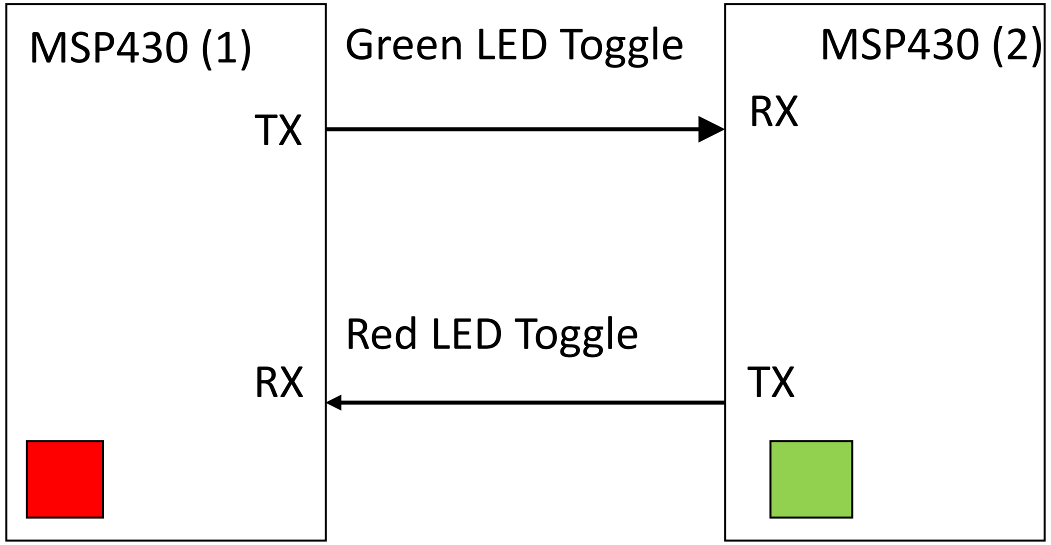
Task 3: Communication

- Develop the code that 2 MSP430 boards can send commands to each other.
- Once a command is received, the MSP430 board toggles the green or red LED.
- MSP430 (1) sends a command to MSP430 (2).
  - When the command is received, MSP430 (2) toggles the green LED.
- MSP430 (2) sends a command to MSP430 (1).
  - When MSP430 (1) receives the command, it toggles the red LED.
- A timer is utilized to control the interval between 2 commands.



Task 3: Communication

- In this task, one needs to develop two sets of code, one for MSP430 (1) and one for MSP430 (2).
- Basic logic and the function are same.
- Only difference is that MSP430 (1) sends out “Green LED Toggle” command and MSP430 (2) sends out “Red LED Toggle” command.



Task 3 Code Flow

1. Define the commands as a specific character
   - For example, "Green LED Toggle" defined as 'g' and "Red LED Toggle" defined as 'r'.
2. Stop the watchdog timer
3. Configure general purpose I/O ports to switch on/off LEDS
4. Configure the timer that defines the interval between two commands
   - Relevant registers: TACCR0, TACCTL0
   - Enable Timer 0 interrupt
   - Load the counter value that corresponds to desired time between transmission
   - Configure Timer 0 operation

Task 4 Code Flow

5. Configure the UART port: use calibrated DCO clock
   - BCSCTL1 = CALBC1_1MHZ;
   - DCOCTL = CALDCO_1MHZ;
6. Configure the TX and RX pin function
   - Configure P1SEL and P1SEL2 registers to use I/O ports as transmit and receive pins
7. Configure the UART using SMCLK
   - Modify UCA0CTL1 registers
8. Configure the baud rate as 19200 (the SMCLK is configured as 1MHz)
   - Load registers UCBRO, UCBR1, and modulation values

Task 4 Code Flow

9. Load Initialize USCI state machine
10. Enable the UART RX interrupt

```
UCA0CTL1 |= UCSWRST;          // Set UCSWRST
UCA0CTL1 |= UCSSEL_2;       // SMCLK
UCA0BR0 = x;            // Least significant byte of divider
UCA0BR1 = y;            // Most significant byte of divider
UCA0MCTL = UCBRS_1;         // Modulation UCBRSx = 1
UCA0CTL1 &= ~UCSWRST;       // Initialize USCI state machine
IE2 |= UCA0RXIE;            // Enable USCI_A0 RX interrupt
```

Task 4 Code Flow

11. Send the command when TIMER interrupt triggered
    - MSP430 TX buffer name: UCA0TXBUF
    - MSP430(1) sends “Green LED Toggle” while MSP430(2) sends “Red LED Toggle”.

```
#pragma vector=TIMER0_A0_VECTOR
__interrupt void SendCMD(void)
{
   UCA0TXBUF = //data corresponding to red or green
}
```

Task 4 Code Flow

12. Handle the received command in the UART RX interrupt service routine with the interrupt vector “USCIAB0RX_VECTOR”
    - Compare the command in the UART receive buffer UCA0RXBUF
    - Toggle the corresponding LED according to the received command

```
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
  if(UCA0RXBUF == //red message )
      {P1OUT ^= 0x01;}  //Turn on Red LED
  if(UCA0RXBUF == //green message )
      {P1OUT ^= 0x40;} // //Red message Green LED
}
```

Next Lecture: C++ Programming