COMPUTER ENGINEERING DESIGN II
Tutorial

MSP430F149 RS232

Prepared by Zexin Pan

February 2005

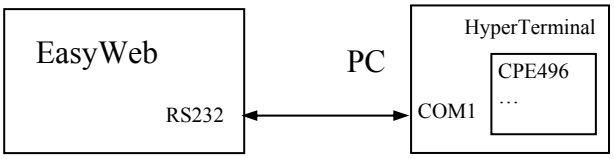
This tutorial describes how to utilize MSP430F149 UART to send/receive data to/from a PC. The template has been tested on MSP430-easyWeb2 prototyping board from OLIMEX Ltd. (www.olimex.com). The features of this board include:

- MSP430F149 running Andreas' open source TCPIP stack
- CS8900 LAN controller + LAN transformer and RJ45 LAN connector
- three status LEDs for the LAN
- one general purpose LED for the easyWeb2
- two 10A/240VAC relay outputs
- four optocoupled inputs
- four push button switches
- RS232 output
- Buzzer
- LCD 16x2 display
- 24LC515 (64K eeprom memory for web data storage)
- Dallas iButton interface for temperature RTC and etc iButton measurements
- frequency input
- extension port for the ADC port
- extension port with the power supply
- JTAG connector
- 32 768 Hz oscillator crystal
- 8Mhz oscillator crystall
- Dimensions: 138x83 mm (5.45x3.25")

Note: The schematic diagram of this board can be found from here:

http://www.ece.uah.edu/~jovanov/msp430/msp430-easyweb2-sch.gif

1. Demonstration Setup



The configuration of HyperTerminal is as follows.

- Connecting using: COM1
- Bit per second: 38400
- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: use Xon/Xoff

How to send character from HyperTerminal to program:

- Create a text file using Notepad, e.g. test.txt
- Write a character into the file, e.g. 'A'
- Save this file to your project directory
- Under HyperTerminal, choose "transfer" -> "Send Text File". Then load text.txt. HyperTerminal will send each character in the text file sequentially through com1 port

2. Examples

The universal synchronous/asynchronous receive/transmit (USART) peripheral interface supports two serial modes with one hardware module: (a) UART mode; (b) SPI mode. This tutorial discusses the operation of the asynchronous UART mode. In addition to USART0, the MSP430x14x devices implement a second identical USART module, USART1.

Note:

- More information about the UART can be found from "*MSP430X1XX Family User's Guide*" by Texas Instruments, Chapter 13: *USART Peripheral Interface, UART Mode*
- MSP430 UART Calculator (i.e. calculate the values for UART *U0MCTL, U0BR0 and U0BR1 r*egisters when using different clock rate and baud rate) can be found from here: http://www.ece.uah.edu/~jovanov/msp430/MSP430_UART_Calculator.xls
- More examples about RS232 can be found from here: http://www.ece.uah.edu/~jovanov/msp430/

E.g. serial_example.c from serial_example.zip

The following program shows how to send character to HyperTerminal, how to receive character from HyperTerminal (by polling or by UART RX interrupt service routine).

```
/*****************************************************************
;
;       Description: MSP430F149 serial link example for CPE496
;       Date : 02/14/2005
;       Auth : Zexin Pan
;
*****************************************************************/

#include "msp430x14x.h"

/* uncomment this to make UART RX interrupt driven */
//#define UART0_INT_DRIVEN_RX

unsigned char thr_char;  /* hold char from UART RX*/
unsigned char rx_flag;   /* receiver rx status flag */

// universal delay function
void Delay(void);

// global clock configuration
void SpeedUpClock(void);


// UART0 rx int.
interrupt [UART0RX_VECTOR] void UART0_RX_interrupt(void);
// UART Initializaion
void UART_Initialize(void);
//send char function
void UART0_putchar(char c);
//receive char function
char UART0_getchar(void);

void main(void)
{


  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer


  SpeedUpClock();

  UART_Initialize();

  #ifdef UART0_INT_DRIVEN_RX
    rx_flag=0x00;        /* rx default state "empty" */
  #endif

  _EINT();       // enable global interrupts

  #ifdef UART0_INT_DRIVEN_RX
    while(!(rx_flag&0x01)); /* wait until receive the character from
HyperTerminal*/
  #else
    thr_char=UART0_getchar();
  #endif
```

```
  // send a message
    UART0_putchar('C');
    UART0_putchar('P');
    UART0_putchar('E');
    UART0_putchar('4');
    UART0_putchar('9');
    UART0_putchar('6');

  while(1)
    {
    Delay();
    UART0_putchar(thr_char);        // send the char we just received!
    }
}

void SpeedUpClock(void)
{
  IFG2 = 0;
  IFG1 = 0;


  //using a 8MHz clock from LFXT1, and ACLK=LFXT1
  BCSCTL1 |= XTS;

  /*wait in loop until crystal stabalizes*/
  do
  {
    IFG1 &= ~OFIFG;
  }
  while (OFIFG & IFG1);

  Delay();
  //Reset osc. fault flag again
  IFG1 &= ~OFIFG;
}



void Delay(void)
{
  volatile unsigned int delay_count;
  for(delay_count=0xffff;delay_count!=0;delay_count--);
}


void UART_Initialize(void)
{
//////////////////////////////////////////////////////////////
//   UART0 Initialization
//////////////////////////////////////////////////////////////


// UART0 - 38400 bps
//IMPORTANT NOTICE: the following configuration works only for 8MHZ
clock rate
// AND 38400 bps baud rate
//you have to recalculate these values if the cpu clock rate or baud
rate has changed
  U0MCTL = 146;
  U0BR0  = 208;
  U0BR1  = 0;

  P3SEL |= 0x30; // bits 4-5 are for special function UART0
```

```
  ME1 |= URXE0 + UTXE0; // UART module enable
  U0CTL = CHAR;     /* 8-bit characters */
  U0TCTL = SSEL0; /* clock source ACLK, 8MHZ */
  U0RCTL = 0;

  #ifdef UART0_INT_DRIVEN_RX
    IE1 |= URXIE0;  // enable RX interrupts
  #endif
}  //end UART_Initialize()


interrupt [UART0RX_VECTOR] void UART0_RX_interrupt(void)
{

  thr_char = U0RXBUF;
  rx_flag=0x01;          // signal main function receiving a char
 }


void UART0_putchar(char c)
{

  // wait for other character to transmit
  while (!(IFG1 & UTXIFG0));
  U0TXBUF = c;
}

char UART0_getchar(void)
{

  for (;;)
  {
    if (U0RCTL & RXERR)
    {
      // if you want to answer receive errors, do it here.
      U0RCTL &= ~(FE+PE+OE+BRK+RXERR);
    }
    else
    {
      if (IFG1 & URXIFG0)
      {  // we've received a character
        return U0RXBUF;
      }
    }
  }
}
```