Enabling UART0 for sending data out for debugging purpose

```
#include "stdio.h"
#include "string.h"

// Define size of allocated UART RX/TX buffer (just an example).
#define SIZE_OF_UART_RX_BUFFER 30
#define SIZE_OF_UART_TX_BUFFER 30
// Baudrate = 9600 (U0BAUD.BAUD_M = 59, U0GCR.BAUD_E = 8), given 32 MHz system clock.
#define UART_BAUD_M 59
#define UART_BAUD_E 8
// Buffer for UART RX/TX.
static uint8 __xdata uartTxBuffer[SIZE_OF_UART_TX_BUFFER];

// Prototype for local functions.
void uart0Send(uint8* uartTxBuf, uint16 uartTxBufLength);
void Uartinit(void);

void uart0Send(uint8* uartTxBuf, uint16 uartTxBufLength)
{
uint16 uartTxIndex;
// Clear any pending TX interrupt request (set U0CSR.TX_BYTE = 0).
U0CSR &= ~ BV(1);
// Loop: send each UART0 sample on the UART0 TX line.
for (uartTxIndex = 0; uartTxIndex < uartTxBufLength; uartTxIndex++)
{
U0DBUF = uartTxBuf[uartTxIndex];
while(! (U0CSR & BV(1)) );
U0CSR &= ~ BV(1);
}
}
```

```
void Uartinit( void)
{
  /*****************************************************************************
* Clock setup
* See basic software example "clk_xosc_cc254x"
*/
// Set system clock source to HS XOSC, with no pre-scaling.
CLKCONCMD &=~ BV(6);
CLKCONCMD &=~ BV(2)|BV(1)|BV(0);
// Wait until clock source has changed.
while (CLKCONSTA & BV(6));
/* Note the 32 kHz RCOSC starts calibrating, if not disabled. */
/*****************************************************************************
* Setup I/O ports
*
* Port and pins used by USART0 operating in UART-mode, at the Alternative 1
* location are:
* RX : P0_2
* TX : P0_3
* CT/CTS : P0_4
* RT/RTS : P0_5
*
* These pins must be set to function as peripheral I/O to be used by UART0.
* The TX pin on the transmitter must be connected to the RX pin on the receiver.
* If enabling hardware flow control (U0UCR.FLOW = 1) the CT/CTS (Clear-To-Send)
* on the transmitter must be connected to the RS/RTS (Ready-To-Send) pin on the
* receiver.
*/
PERCFG &=~ BV(1);
// Give priority to USART 0 over Timer 1 for port 0 pins.
```

```
P2DIR &=~ BV(7);P2DIR &=~ BV(6);
// Set pins 2, 3 and 5 as peripheral I/O and pin 4 as GPIO output.
P0SEL |= BV(2);P0SEL |= BV(3);P0SEL |= BV(4);P0SEL |= BV(5);
// Initialize P0_1 for SRF05EB S1 button.
P0SEL &= ~BV(1); // Function as General Purpose I/O.
P0DIR &= ~BV(1); // Input.
/***************************************************************************
* Configure UART
*
*/
// Initialise bitrate = 57.6 kbps.
U0BAUD = UART_BAUD_M;
U0GCR = 0xFF & UART_BAUD_E;
// Initialise UART protocol (start/stop bit, data bits, parity, etc.):
// USART mode = UART (U0CSR.MODE = 1)
U0CSR |= BV(7);
// Start bit level = low => Idle level = high (U0UCR.START = 0).
U0UCR &= ~BV(0);
// Stop bit level = high (U0UCR.STOP = 1).
U0UCR |= BV(1);
// Number of stop bits = 1 (U0UCR.SPB = 0).
U0UCR &= ~BV(2);
// Parity = disabled (U0UCR.PARITY = 0).
U0UCR &= ~BV(3);
// 9-bit data enable = 8 bits transfer (U0UCR.BIT9 = 0).
U0UCR &= ~BV(4);
// Level of bit 9 = 0 (U0UCR.D9 = 0), used when U0UCR.BIT9 = 1.
// Level of bit 9 = 1 (U0UCR.D9 = 1), used when U0UCR.BIT9 = 1.
// Parity = Even (U0UCR.D9 = 0), used when U0UCR.PARITY = 1.
// Parity = Odd (U0UCR.D9 = 1), used when U0UCR.PARITY = 1.
U0UCR &= ~BV(5);
```

```
// Flow control = disabled (U0UCR.FLOW = 0).
U0UCR &= ~BV(6);
// Bit order = LSB first (U0GCR.ORDER = 0).
U0GCR &= ~BV(5);
}

void main()
{
  Uartinit();  // initiliaze uart0
  strcpy(uartTxBuffer,"EP 0 LIGHT ON \n");  //convert string into array
  uart0Send(uartTxBuffer, SIZE_OF_UART_TX_BUFFER);  // transmit data
}
```