Microprocessor Systems
Lab 3: Asynchronous & Synchronous Serial Communications

Nick Choi   Samuel Deslandes

10/17/16

1 Introduction

2 Methods

2.1 Software

The code for parts 1, 2 and 3 can be found in the appendix below. All code was uploaded and run on the 8051 through the programming/debugging USB port.

2.1.1 Part 1

The purpose of C program for the first section of the lab was to write a procedure that would monitor the UART0 and UART1 serial ports continuously, and echo a message received on one port to the other. For this section of the lab, this was accomplished by polling the receive interrupt flags RI0 and RI1 for UART0 and UART1 respectively.

Since this lab uses both UART0 and UART1, both ports must be enabled on the crossbar. The enable bit for UART0 can be found on the XBR0 SFR, XBR2 for that of UART1. As always, the crossbar enable bit in XBR2 must be set. UART0 was configured to be in mode 1 (8-bit UART with variable baud rate), and use Timer2 to generate a baud rate of 9600bps. UART0 configuration can set in the SSTA0 and SCON0 SFRs. Timer2 was configured to be in auto-reload mode and use SYSCLK as a base. These settings can be found in the TMR2CN and TMR2CF SRFs. To count to 9600, the reload value must be set to `0xFEBC`. UART1 was similarly configured, but used Timer1 to generate a baud rate of 115200bps. UART1 can be configured in the SCON1 SFR. Timer1 was set to be an 8-bit counter with auto-reload, and use SYSCLK as a base; these changes can be made in the TMOD SFR. The Timer1 reload value should be set as `0x28`. When configuring both UARTs be sure that the receive enable bits are set in the SCON SFRs and that the timers are have been started.

The main routine of this program consists of two sub-routines 'checkSBUF()' and 'echo()' which are run in an infinite loop. The 'checkSBUF()' function continuously polls RI0 and RI1, and returns the value stored in the UART data register when one of the flags is set. The UART data register (SBUFn) is used for both a receive and a transmit buffer. When data is read from SBUF, as is the case for the 'checkSBUF()' function, it comes from the receive buffer. When data is written to SBUF it goes to the transmit buffer and is eventually transmitted when a full byte has been written. The returned value from 'checkSBUF()' is then passed into the 'echo()' function, where it is loaded into both SBUF0 and SBUF1 for transmission to both terminals. Recall that SBUF0 and SBUF1 are on different pages, and that SFRPAGE should be changed accordingly.

2.1.2 Part 2

This section of the lab was divided into two parts. The task for the first part was similar to that of section 1, however rather than polling, interrupts were used. The second part involved connecting UART1 to that of another 8051, so that when data is sent to UART0 of one of the 8051s, it echoed to the UART0 of the other microcontroller. The initial configurations were all the same as they were in section 1, with the addition of having to set

the global interrupt enable bit (EA), as well as the individual UART0 and UART1 interrupt enable bits which can be found on the IE and EIE2 SFRs respectively. With these bits set, whenever the transmit interrupt flags (TIn) or receive interrupt flags (RIn) are set, the CPU vectors to the appropriate interrupt service routine (ISR); These flags must be cleared manually by the software. The UART0 interrupt has a priority level of 4, while the UART1 interrupt has a priority of 20. One thing to keep in mind in regard to these interrupts is that while UART0 interrupts are enabled, unless the UART0 and UART1 interrupt priorities are swapped, UART1 will not receive any interrupts. Since both receiving and transmitting will trigger an interrupt, the ISR must poll each flag to verify the source of the interrupt.

For part 1 of this section of the lab, the main would only check if the command to end the program had been sent; everything else was handled in the ISRs. The UART0 ISR checks if the source of the interrupt was a receive by checking if RI0 has been set. If so, it is cleared and the 'echo()' subroutine is called. After this, UART0 interrupts are temporarily disabled to allow UART1 interrupts to go though. The UART1 ISR has the same routine, however rather than disabling UART0 interrupts, it re-enables them.

The program for part 2 was handled slightly differently. The main routine now continuously polled two flags 'UART0_flag' and 'UART1_flag', which would be set in the ISRs of UART0 and UART1 respectively. The 'echo()' function is also used for this part, of the lab, but with the slight modification of disabling UART0 interrupts before loading data into the SBUF registers. The UART0 ISR checks whether the source of the interrupt is a receive, and if it is it clears RI0, reads SBUF0 into a global variable, and sets the UART0_flag. The UART1 ISR does the same, but enables UART0 interrupts at the end of the ISR.

2.1.3 Part 3

2.2 Hardware

Sections 1 and 3 of this lab did were identical in terms of hardware. Section 2 required no hardware other than a serial-to-USB adapter in order to interface with the terminal.

In sections 1 and 3, the pushbutton included on the breadboard was used. It was wired such that the button would be grounded when pressed, with the output connected to pin 18 on the EVB, corresponding to P0.2 of the 8051. A pull-up resistor was used to prevent false external interrupts.

3   Results

4   Conclusion

5   Appendices

5.1   Modified putget.h

```
//————————————————————————————————————————————————————————————————————
// putget.h
//————————————————————————————————————————————————————————————————————
// Title:                  Microcontroller Development: putchar() & getchar() functions.
// Author:                 Dan Burke
// Date Created:           03.25.2006
// Date Last Modified:     03.25.2006
//
// Description:            http://chaokhun.kmitl.ac.th/~kswichit/easy1/easy1_3.html
//
//
// Target:                 C8051F120
// Tool Chain:             KEIL C51

//————————————————————————————————————————————————————————————————————
// putchar()
//————————————————————————————————————————————————————————————————————
void putchar(char c)
{
    while(!TI0);
    TI0=0;
    SBUF0 = c;
}

//————————————————————————————————————————————————————————————————————
// getchar()
//————————————————————————————————————————————————————————————————————
char getchar(void)
{
    char c;
    while(!RI0);
    RI0 =0;
    c = SBUF0;
// Echoing the get character back to the terminal is not normally part of getchar()
//    putchar(c);    // echo to terminal
    return SBUF0;
}
```
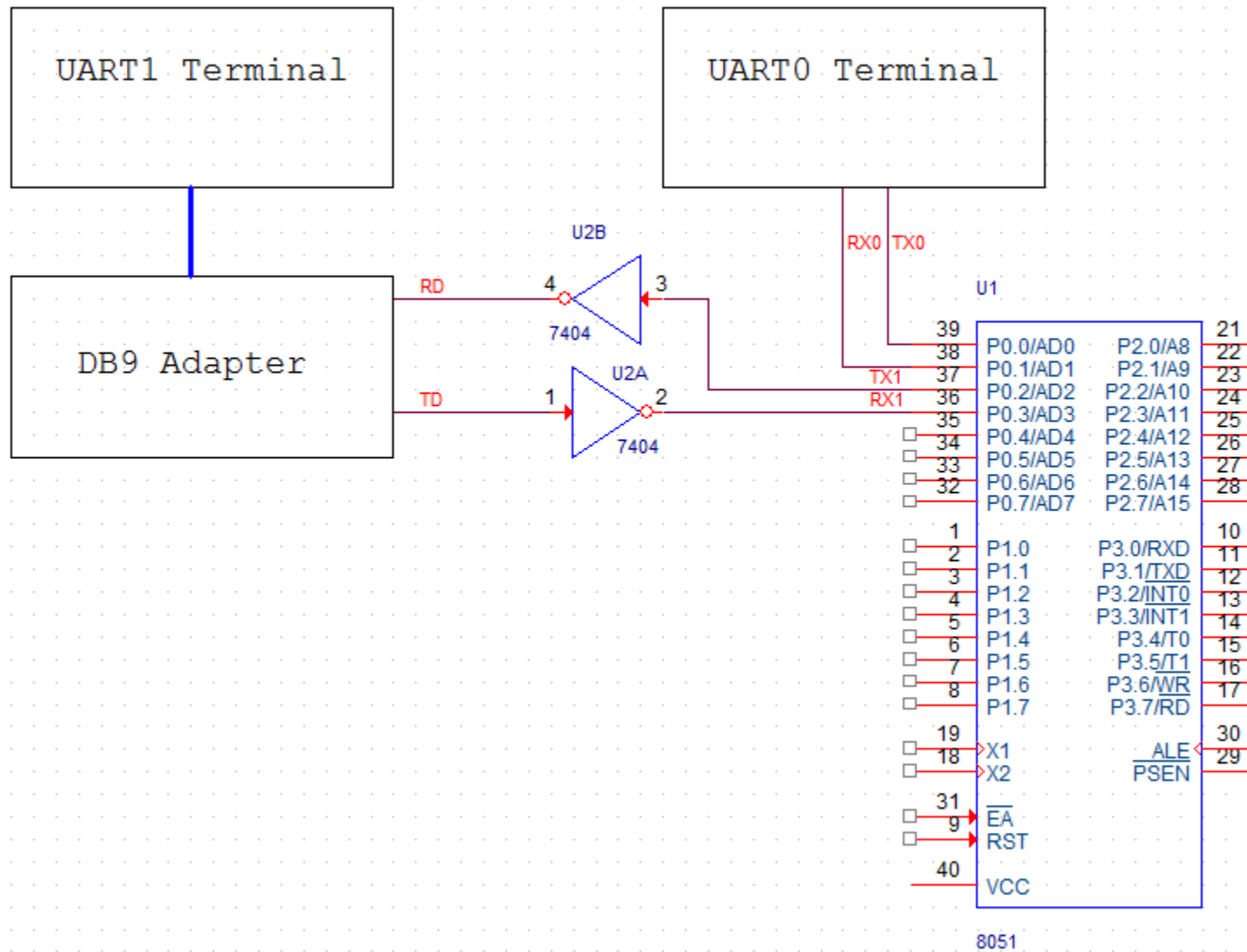
4

5.2 Circuit Schematic for sections 1 and 3

Figure 1: Circuit schematic for parts 1 and 3

5.3 Part 1

5.3.1 Code

```
//-----------------------------------------------------------
// Includes
//-----------------------------------------------------------
#include <c8051f120.h>
#include <stdio.h>
#include <stdlib.h>
#include "putget.h"


//-----------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------
```

```
#define EXTCLK        22118400      // External oscillator frequency in Hz
#define SYSCLK        49766400      // Output of PLL derived from (EXTCLK * 9/4)
#define BAUDRATE      115200        // UART baud rate in bps
//#define BAUDRATE    19200         // UART baud rate in bps

//————————————————————————————————————————————————————
// Function PROTOTYPES
//————————————————————————————————————————————————————
void main(void);
void PORT_INIT(void);
void SYSCLK_INIT(void);
void UART0_INIT(void);
void UART1_INIT(void);
void TIMER0_INIT(void);
char checkSBUF(char CUR_PAGE);
void echo(char character);

//————————————————————————————————————————————————————
// Main Function
//————————————————————————————————————————————————————

void main(void){
  // Declare local variables
  char message;
  unsigned long i;

    WDTCN = 0xDE;                          // Disable the watchdog timer
    WDTCN = 0xAD;

    PORT_INIT();                           // Initialize the Crossbar and GPIO
    SYSCLK_INIT();                         // Initialize the oscillator
    UART0_INIT();                          // Initialize UART0
  UART1_INIT();                       // Initialize UART1

    SFRPAGE = UART1_PAGE;                  // Direct output to UART1

    printf("\033[2J");                     // Erase screen & move cursor to home position
    printf("Test of the printf() function.\n\n\r");

  SFRPAGE = UART0_PAGE;           // Direct output to UART1

  printf("\033[2J");                       // Erase screen & move cursor to home position
    printf("Test of the printf() function.\n\n\r");

    while(1)
    {
    message = checkSBUF(SFRPAGE);
    if(message == 27){
      printf("\n\n\rStopping now...");
      SFRPAGE = UART1_PAGE;
      printf("\n\n\rStopping now...");
      for(i = 0;i<2000000;i++);
      return;
    }
    echo(message);
  }
}

char checkSBUF(char CUR_PAGE){
  char serial_dat0;
  char serial_dat1;

  while(1){
    SFRPAGE = UART0_PAGE;
    if(RI0){
      serial_dat0 = SBUF0;
      RI0 = 0;
```

```
        SFRPAGE = CUR_PAGE;
         return serial_dat0;
      }
     SFRPAGE = UART1_PAGE;
      if(RI1){
         serial_dat1 = SBUF1;
         RI1 = 0;
        SFRPAGE = CUR_PAGE;
         return serial_dat1;
      }
   }
}

void echo(char character){
  char SFRPAGE_SAVE;
  SFRPAGE_SAVE = SFRPAGE;

  SFRPAGE = UART0_PAGE;
  SBUF0 = character;

  SFRPAGE = UART1_PAGE;
  SBUF1 = character;

  SFRPAGE = SFRPAGE_SAVE;
}

//————————————————————————————————
// Interrupts
//————————————————————————————————



//————————————————————————————————————————————————
// PORT_Init
//————————————————————————————————————————————————
// Configure the Crossbar and GPIO ports

void PORT_INIT(void){
  char SFRPAGE_SAVE;

  SFRPAGE_SAVE = SFRPAGE;          // Save Current SFR page.

  SFRPAGE = CONFIG_PAGE;
  EA       = 1;                    // Enable interrupts as selected.
  XBR0     = 0x04;                 // Enable UART0.
  XBR1     = 0x00;                 //
  XBR2     = 0x44;                 // Enable Crossbar and weak pull-ups and UART1.
  P0MDOUT = 0x05;                  // P0.0 (TX0) and P0.2 (TX1) are configured as Push-Pull for
      output, P0.1 (RX0) and P0.3 (RX1) are Open-drain
  P0       = ~0x05;                // Additionally, set P0.0=0, P0.1=1, P0.2=0, P0.3=1

  SFRPAGE = SFRPAGE_SAVE;          // Restore SFR page.

}

//————————————————————————————————————————————
// SYSCLK_Init
//————————————————————————————————————————————

// Initialize the system clock 22.1184Mhz

void SYSCLK_INIT(void){
  int i;

  char SFRPAGE_SAVE;

  SFRPAGE_SAVE = SFRPAGE;          // Save Current SFR page.
```

```
  SFRPAGE = CONFIG_PAGE;
  OSCXCN  = 0x67;                 // Start external oscillator
  for(i=0; i < 256; i++);        // Wait for the oscillator to start up.
  while(!(OSCXCN & 0x80));        // Check to see if the Crystal Oscillator Valid Flag is set.
  CLKSEL  = 0x01;                 // SYSCLK derived from the External Oscillator circuit.
  OSCICN  = 0x00;                 // Disable the internal oscillator.

  SFRPAGE = CONFIG_PAGE;
  PLL0CN  = 0x04;
  SFRPAGE = LEGACY_PAGE;
  FLSCL   = 0x10;
  SFRPAGE = CONFIG_PAGE;
  PLL0CN |= 0x01;
  PLL0DIV = 0x04;
  PLL0FLT = 0x01;
  PLL0MUL = 0x09;
  for(i=0; i < 256; i++);
  PLL0CN |= 0x02;
  while(!(PLL0CN & 0x10));
  CLKSEL  = 0x02;                 // SYSCLK derived from the PLL.

  SFRPAGE = SFRPAGE_SAVE;         // Restore SFR page.
}

//------------------------------------------------------------
// UART0_Init
//------------------------------------------------------------

// Configure the UART0 using Timer1, for 9600 and 8-N-1.
void UART0_INIT(void)
{
  char SFRPAGE_SAVE;

  SFRPAGE_SAVE = SFRPAGE;         // Save Current SFR page.

  SFRPAGE = TMR2_PAGE;
    TMR2CN  = 0x00;               // Auto-reload mode, use clock defined in TMR2CF,
  TMR2CF  = 0x08;        // Timer 2 uses SYSCLK as time base

  RCAP2H  = 0xFE;                 // Set timer 2 auto-reload value for 9600bps
  RCAP2L  = 0xBC;

  TR2 = 1;              // Start timer 2

  SFRPAGE = UART0_PAGE;
  SCON0   = 0x50;                 // Set Mode 1: 8-Bit UART
  SSTA0   = 0x15;                 // UART0 baud rate divide-by-two disabled (SMOD0 = 1) and use
      TMR2.
  TI0     = 1;                    // Indicate TX0 ready.

  SFRPAGE = SFRPAGE_SAVE;         // Restore SFR page
}


// Configure the UART1 using Timer1, for 115200 and 8-N-1.
void UART1_INIT(void)
{
    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;               // Save Current SFR page

    SFRPAGE = TIMER01_PAGE;
    TMOD    &= ~0xF0;
    TMOD    |=  0x20;                      // Timer1, Mode 2, 8-bit reload
    TH1     = 40;           // Set Timer1 reload baudrate value T1 Hi Byte
  CKCON |= 0x10;                          // Timer1 uses SYSCLK as time base
    TL1     = TH1;
    TR1     = 1;                          // Start Timer1
```

8

```
   SFRPAGE = UART1_PAGE;
      SCON1   = 0x30;                         // Mode 1, 8−bit UART, enable RX
      TI1      = 1;                           // Indicate TX1 ready

   SFRPAGE = SFRPAGE_SAVE;          // Restore SFR page
}
```

5.4   Part 2

5.4.1   Code

```
//————————————————————————————————————————————
// Includes
//————————————————————————————————————————————
#include <c8051f120.h>
#include <stdio.h>
#include <stdlib.h>
#include "putget.h"


//————————————————————————————————————————————————
// Global CONSTANTS
//————————————————————————————————————————————————

#define EXTCLK        22118400      // External oscillator frequency in Hz
#define SYSCLK        49766400      // Output of PLL derived from (EXTCLK ∗ 9/4)
#define BAUDRATE      115200        // UART baud rate in bps
//#define BAUDRATE    19200         // UART baud rate in bps
char CURR_PAGE;
char exit_flag = 0;
char UART0_flag = 0;
char UART1_flag = 0;
char choice = 0;
//————————————————————————————————————————————
// Function PROTOTYPES
//————————————————————————————————————————————
void main(void);
void PORT_INIT(void);
void SYSCLK_INIT(void);
void UART0_INIT(void);
void UART1_INIT(void);
void TIMER0_INIT(void);
char checkSBUF(char CUR_PAGE);
void echo(char character);
void INTERRUPT_INIT(void);
void UART0_int(void) __interrupt 4;
void UART1_int(void) __interrupt 20;


//————————————————————————————————————————————————
// Main Function
//————————————————————————————————————————————————

void main(void){
  // Declare local variables
  char message;
  unsigned long i;
  char j;

   WDTCN = 0xDE;                          // Disable the watchdog timer
   WDTCN = 0xAD;

   PORT_INIT();                           // Initialize the Crossbar and GPIO
   SYSCLK_INIT();                         // Initialize the oscillator
   UART0_INIT();                          // Initialize UART0
  UART1_INIT();                    // Initialize UART1
  INTERRUPT_INIT();
```

```c
    SFRPAGE = UART0_PAGE;              // Direct output to UART1

  printf("\033[2J");                           // Erase screen & move cursor to home position
    printf("Test of the printf() function.\n\n\r");

    while(1)
    {
    SFRPAGE = UART0_PAGE;
    if(exit_flag){
      printf("\n\n\rStopping now...");
      SFRPAGE = UART1_PAGE;
      printf("\n\n\rStopping now...");
      for(i = 0;i<20000;i++);
      break;
    }

    for(j=0;j<255;j++){
      ES0 = 0;
    }
    ES0 = 1;

    if(UART0_flag){
      echo(choice);
      UART0_flag = 0;
    }
    if(UART1_flag){
      ES0 = 0;
      SBUF0 = choice;
      ES0 = 1;
      UART1_flag = 0;
    }

  }
  return;
}

void echo(char character){
  if(character == 27){
    exit_flag = 1;
    return;
  }
  ES0 = 0;
  SFRPAGE = UART0_PAGE;
  SBUF0 = character;

  SFRPAGE = UART1_PAGE;
  SBUF1 = character;
}

//————————————————————————————————————
// Interrupts
//————————————————————————————————————
void UART0_int(void) __interrupt 4{
  CURR_PAGE = SFRPAGE;
  SFRPAGE = UART0_PAGE;
  if(RI0){
    RI0 = 0;
    choice = SBUF0;
    UART0_flag = 1;
  }
  SFRPAGE = CURR_PAGE;
}

void UART1_int(void) __interrupt 20{
  CURR_PAGE = SFRPAGE;
  SFRPAGE = UART1_PAGE;
  if(RI1){
    RI1 = 0;
```

```c
        choice = SBUF1;
        UART1_flag = 1;
    }
    ES0 = 1;
    SFRPAGE = CURR_PAGE;
}


//————————————————————————————————————————————
// PORT_Init
//————————————————————————————————————————————
// Configure the Crossbar and GPIO ports

void PORT_INIT(void){
    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;        // Save Current SFR page.

    SFRPAGE = CONFIG_PAGE;
    XBR0    = 0x04;                // Enable UART0.
    XBR1    = 0x00;                //
    XBR2    = 0x44;                // Enable Crossbar and weak pull-ups and UART1.
    P0MDOUT = 0x05;               // P0.0 (TX0) and P0.2 (TX1) are configured as Push-Pull for
            output, P0.1 (RX0) and P0.3 (RX1) are Open-drain
    P0      = 0x0A;               // Additionally, set P0.0=0, P0.1=1, P0.2=0, P0.3=1

    SFRPAGE = SFRPAGE_SAVE;        // Restore SFR page.

}

//————————————————————————————————————————————
// SYSCLK_Init
//————————————————————————————————————————————

// Initialize the system clock 22.1184Mhz

void SYSCLK_INIT(void){
    int i;

    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;        // Save Current SFR page.

    SFRPAGE = CONFIG_PAGE;
    OSCXCN  = 0x67;                // Start external oscillator
    for(i=0; i < 256; i++);       // Wait for the oscillator to start up.
    while(!(OSCXCN & 0x80));      // Check to see if the Crystal Oscillator Valid Flag is set.
    CLKSEL  = 0x01;               // SYSCLK derived from the External Oscillator circuit.
    OSCICN  = 0x00;               // Disable the internal oscillator.

    SFRPAGE = CONFIG_PAGE;
    PLL0CN  = 0x04;
    SFRPAGE = LEGACY_PAGE;
    FLSCL   = 0x10;
    SFRPAGE = CONFIG_PAGE;
    PLL0CN |= 0x01;
    PLL0DIV = 0x04;
    PLL0FLT = 0x01;
    PLL0MUL = 0x09;
    for(i=0; i < 256; i++);
    PLL0CN |= 0x02;
    while(!(PLL0CN & 0x10));
    CLKSEL  = 0x02;               // SYSCLK derived from the PLL.

    SFRPAGE = SFRPAGE_SAVE;        // Restore SFR page.
}

void INTERRUPT_INIT(void){
```

```
   EA  = 1;                    // Enable interrupts as selected.
   ES0 = 1;
   EIE2 |= 0x40;
}

// Configure the UART0 using Timer1, for 9600 and 8-N-1.
void UART0_INIT(void)
{
  char SFRPAGE_SAVE;

  SFRPAGE_SAVE = SFRPAGE;      // Save Current SFR page.

  SFRPAGE = TMR2_PAGE;
    TMR2CN  = 0x00;                // Auto-reload mode, use clock defined in TMR2CF,
  TMR2CF  = 0x08;            // Timer 2 uses SYSCLK as time base

  RCAP2H  = 0xFD;                 // Set timer 2 auto-reload value for 4800bps
  RCAP2L  = 0x78;

  TR2 = 1;             // Start timer 2

  SFRPAGE = UART0_PAGE;
  SCON0   = 0x50;               // Set Mode 1: 8-Bit UART
  SSTA0   = 0x15;               // UART0 baud rate divide-by-two disabled (SMOD0 = 1) and use
       TMR2.
  TI0     = 1;                  // Indicate TX0 ready.

  SFRPAGE = SFRPAGE_SAVE;       // Restore SFR page
}


// Configure the UART1 using Timer1, for 115200 and 8-N-1.
void UART1_INIT(void)
{
    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;                 // Save Current SFR page

    SFRPAGE = TIMER01_PAGE;
    TMOD    &= ~0xF0;
    TMOD    |=  0x20;                       // Timer1, Mode 2, 8-bit reload
    TH1     = 40;           // Set Timer1 reload baudrate value T1 Hi Byte
  CKCON   |= 0x10;                       // Timer1 uses SYSCLK as time base
    TL1     = TH1;
    TR1     = 1;                           // Start Timer1

  SFRPAGE = UART1_PAGE;
    SCON1   = 0x30;                       // Mode 1, 8-bit UART, enable RX
    TI1     = 1;                          // Indicate TX1 ready

  SFRPAGE = SFRPAGE_SAVE;                // Restore SFR page
}
```

5.5 Part 3

5.5.1 Code

```
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <c8051f120.h>
#include <stdio.h>
#include "putget.h"

//-----------------------------------------------------------------------------
// Global Constants
```

```c
//─────────────────────────────────────────────────────────────────────────────
#define EXTCLK          22118400                // External oscillator frequency in Hz
#define SYSCLK          49766400                // Output of PLL derived from (EXTCLK * 9/4)
#define BAUDRATE        115200                  // UART baud rate in bps
char choice;
char scroll_dwn = 0;
char dummy;
//─────────────────────────────────────────────────────────────────────────────
// Function Prototypes
//─────────────────────────────────────────────────────────────────────────────
void main(void);
void SYSCLK_INIT(void);
void PORT_INIT(void);
void UART0_INIT(void);
void SPI0_READ(void);
void SPI0_WRITE(void);
void write_dummy(void);


//─────────────────────────────────────────────────────────────────────────────
// MAIN Routine
//─────────────────────────────────────────────────────────────────────────────
void main(void)
{
  WDTCN = 0xDE;                         // Disable the watchdog timer
    WDTCN = 0xAD;

    SYSCLK_INIT();                      // Initialize the oscillator
  PORT_INIT();
  UART0_INIT();



  SFRPAGE = UART0_PAGE;

  printf("\033[2J");                    // Erase screen & move cursor to home position
  printf("Local char typed");
  printf("\033[13;0H");
  printf("Received char\n\r");               // jump down type
  printf("\033[s");                     // save position


  while(1)
    {
    SFRPAGE = UART0_PAGE;
    if(RI0){
      RI0 = 0;
      SPI0_WRITE();
      SPI0_READ();
    }
    }
}

void SPI0_WRITE(void){
  char i;
  char j;

  choice = SBUF0;          // get char from terminal
  SFRPAGE = SPI0_PAGE;
  SPIF = 0;
  NSSMD0 = 0;             // select slave
  while(SPI0CFG & 0x80);     // Make sure SPI is not busy
  SPIF = 0;             // clear SPIF
  SPI0DAT = choice;             // Load char into SPI0DAT
  for(i=0;i<100;i++);       // Small delay
  while(!SPIF);          // Wait until transmission complete
  SPIF = 0;

  //Read Dummy
```

```c
      NSSMD0 = 1;                 // Release slave
      dummy = SPI0DAT;          // Get received char from SPI0DAT (dummy)
      //for(i=0;i<100;i++);      // Small delay

   if(choice == 0x7F){
      while(SPI0DAT != 0xFF){
         for(i=0;i<100;i++);
         write_dummy();
         SPI0_READ();
         //if(choice == 0xFF) break;
      }
   }
   else{
      //ANSI formatting
      printf("\033[2;12r");
      printf("\033[2;0H");
      for(j=0;j<scroll_dwn;j++){
         printf("\033[B");
      }

      //Output local char
      printf("Choice is: %c\r",choice);

      write_dummy();
   }
}

void write_dummy(void){
   //Write Dummy
   NSSMD0 = 0;                 // Select slave
   while(SPI0CFG & 0x80);      // Make sure SPI is not busy
   SPIF = 0;                   // Clear SPIF
   SPI0DAT = 0xFE;             // Load dummy into SPI0DAT
   //for(i=0;i<100;i++);       // Small delay
   while(!SPIF);               // Wait until transmission complete
   SPIF = 0;
}

void SPI0_READ(void){
   char j;

   NSSMD0 = 1;                 // Release slave
   //Output dummy
   printf("\033[2;30H");
   printf("DUMMY: 0x%x", dummy);
   SPIF = 0;

   //ANSI format
   printf("\033[14;24r");
   printf("\033[14;0H");
   for(j=0;j<scroll_dwn;j++){
      printf("\033[B");
   }

   //Output choice
   //choice = SPI0DAT;
   printf("Data read from SPI0DAT is: %c\r",choice);

   scroll_dwn += 1;
   if (scroll_dwn > 10) scroll_dwn = 10;

   SPIF = 0;
}


//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
```

```c
//
// Initialize the system clock to use a 22.1184MHz crystal as its clock source
//
void SYSCLK_INIT(void)
{
    int i;
    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;                 // Save Current SFR page

    SFRPAGE = CONFIG_PAGE;
    OSCXCN  = 0x67;                         // Start ext osc with 22.1184MHz crystal
    for(i=0; i < 256; i++);                 // Wait for the oscillator to start up
    while(!(OSCXCN & 0x80));
    CLKSEL  = 0x01;
    OSCICN  = 0x00;

    SFRPAGE = CONFIG_PAGE;
    PLL0CN  = 0x04;
    SFRPAGE = LEGACY_PAGE;
    FLSCL   = 0x10;
    SFRPAGE = CONFIG_PAGE;
    PLL0CN |= 0x01;
    PLL0DIV = 0x04;
    PLL0FLT = 0x01;
    PLL0MUL = 0x09;
    for(i=0; i < 256; i++);
    PLL0CN |= 0x02;
    while(!(PLL0CN & 0x10));
    CLKSEL  = 0x02;

    SFRPAGE = SFRPAGE_SAVE;                 // Restore SFR page
}

//-------------------------------------------------------------------------------------
// PORT_Init
//-------------------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports
//
void PORT_INIT(void)
{
    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;                 // Save Current SFR page


  SFRPAGE = CONFIG_PAGE;

  XBR0     = 0x06;                          // Enable UART0, SPI
    XBR1     = 0x00;
    XBR2     = 0x40;                        // Enable Crossbar and weak pull-up

  P0MDOUT &= ~0x02;  //~0x0A
  P0MDOUT |= 0x35;                          // Set pins 0,2,4,5 to push-pull
    //P0     |= 0x02;                       // RX0 pin and to high impedance

  SFRPAGE  = SPI0_PAGE;
    SPI0CFG = 0x40;            // Master mode
  SPI0CN  = 0x0D;             // Enable SPI
  SPI0CKR  =  0x18;             // SPI clock rate for 230399
  //SPI0CKR  = 19;                          // 1.244 MHz

  //SPIF = 1;

  //EIE1 = 0x01;
  EA = 1;
  //ES0 = 1;
```

```
    SFRPAGE  = SFRPAGE_SAVE;               // Restore SFR page
}

void UART0_INIT(void)
{
    char SFRPAGE_SAVE;

    SFRPAGE_SAVE = SFRPAGE;                // Save Current SFR page

    SFRPAGE = TIMER01_PAGE;
    TMOD    &= ~0xF0;
    TMOD    |=  0x20;                      // Timer1, Mode 2, 8-bit reload
    TH1      = -(SYSCLK/BAUDRATE/16);      // Set Timer1 reload baudrate value T1 Hi Byte
    CKCON   |= 0x10;                       // Timer1 uses SYSCLK as time base
    TL1      = TH1;
    TR1      = 1;                          // Start Timer1

    SFRPAGE = UART0_PAGE;
    SCON0   = 0x50;                        // Mode 1, 8-bit UART, enable RX
    SSTA0   = 0x10;            // SMOD0 = 1
    TI0      = 1;                          // Indicate TX0 ready

    SFRPAGE = SFRPAGE_SAVE;                // Restore SFR page
}
```

6   References

"MPS Lab 2," in RPI ECSE Department, 2016. [Online]. Available: `http://www.rpi.edu/dept/ecse/mps/MPS_Lab_Ex2-Intrpt.pdf`. Accessed: Sep. 22, 2016.

"C8051 Manual," in RPI ECSE Department, 1.4 ed., 2005. [Online]. Available: `https://www.ecse.rpi.edu/courses/CStudio/Silabs/C8051F12x-13x.pdf`. Accessed: Sep. 22, 2016.