Microprocessor Systems
Lab 5: Memory Interfacing

Nick Choi Samuel Deslandes

11/14/16

1 Introduction

2 Methods

2.1 Software

The code for parts 1, 2 and 3 can be found in Appendix A, B and C respectively. All code was uploaded and run on the 8051 through the programming/debugging USB port.

2.1.1 Part 1

In the first section of the lab, a C program was written to write a value to address `0x1FF0`, an on-board address space on the 8051. This address is then read and its value written to all of the addresses on the Am9128: `0x2000`–`0x27FF`. Each time a value was written it was immediately read back and printed to the terminal to ensure the chip was performing properly and that the value read was the same as what was written. Address `0x2800`, beyond the range of the 2k chip, was also written to, with the expectation that no value would be read back.

In order to interface with the memory chips, the EMI0CF SFR must be configured such that ports 4–7 are used as data buses, address buses, or control signals for the chip, and that EMIF operates in split mode with bank select. This can be done by setting EMI0CF to `0x3B`. The EMI0TC SFR is used for timing control for the chips. Since these are older chips, the slowest setting are used, which is done by setting EMI0TC to `0xFF`. In addition to this, ports 4–7 were configured for push-pull operation using the PnMDOUT SFRs.

In order to access certain memory addresses pointers declared with the "__xdata" keyword were used. The program starts by initializing a pointer to address `0x1FF0` and writing the character 'a' to it. This address is then read and printed to the terminal in order to verify the character 'a' gets read back. The pointer is then changed to point to address `0x2000`, and using a loop the value previously read is written the pointer's location, with the pointer being incremented at the end of the loop. As described above, the loop terminates at address `0x2800`, at which point the dereferenced pointer cannot be read.

2.1.2 Part 2

In principle the program for this part is the same as that of part 1, the main difference being the address range used is now `0x2800`–`0x2FFF`. This program writes '`0xAA`' to all of the addresses in this range, immediately reading them back and printing the result to the terminal. This process is then repeated using the value '`0x55`'. To ensure that all addresses are being updated accordingly, any addresses that do not read back the expected '`0x55`' are added to an array. At the end of the program the contents of this array are printed to the terminal.

2.1.3 Part 3

The program for this part is meant for interfacing with the Am91L14 1024x4 chip, starting at address `0x4000`. First, an array is initialized with the values 0–15. Values will be written

from this array to the memory on the Am91L14. This is accomplished using pointers and the same methods described in part 1.

2.1.4 Enchancement

An enhancement was made to the program described in part 1. This program queries the user for a starting address (within the range `0x1FF0`–`0x1FFF`), and a 4-bit value used for writing to address spaces. Once the input has been parsed, the program writes the value to all of the additional external spaces, reading back the result to the terminal. This represented the range `0x2000`–`0x4400`, with the values in the range of `0x3000`–`0x3FFF` reading back garbage data as none of the chips were enabled in this range.

In order to parse the user input from character inputs to hex values, the program first had to check whether the input was a digit. This was done using the built-in C function 'isdigit()'. If the input was a digit, subtracting 48 (ASCII for '0') would result in the appropriate value (0–9). If the input was not a digit, the program would then check that the input is in the range of letters used by the hexadecimal number system ('A'–'F') and used a similar subtraction method to get the appropriate value. In the case of a capital letter input, 55 was subtracted (ASCII for '7'), and if the input was lowercase 87 was subtracted (ASCII for 'W').

Once the an input had been converted to its hex equivalent, a variable was used in order to form 4 digit hex number. This variable was initialized to 0, and after each input to hex conversion, the variable was multiplied by 16 then incremented by the converted value, resulting in the correct 4 digit hex starting memory address.

2.2 Hardware

The hardware for this lab involved wiring two kinds of memory chips (Am9128 and Am91L14) to the 8051. A full schematic can be seen in the appendix below.

2.2.1 Part 1

The Am91L14 has 11 address lines which connected to all of port 6 for the lower byte of the address and pins 0–2 of port 5 for the higher byte of the address. The pins on port 5 not connected to the chip were used in the glue logic for enabling the chip. The 8 data lines on the Am91L14 were connected to all of port 7, and pins P4.6 and P4.7 were used for the output enable and write enable pins on the chip, respectively.

In order for the chip to be enabled only withing the desired range (`0x2000`–`0x27FF`) a glue logic circuit representing the logic equation $\overline{(\overline{a_{15}} \wedge \overline{a_{14}} \wedge a_{13} \wedge \overline{a_{12}} \wedge \overline{a_{11}})}$ was constructed, where a_n represents the address bit n. This, and all other glue logic circuits were implemented using NAND gates and a hex inverter chip (7404).

2.2.2 Part 2

The hardware for this part involved wiring a second Am91L14, which would operate on the range of `0x2000`–`0x27FF`. The only wiring that changed from the previous part is the glue logic, which now represented the equation: $\overline{(\overline{a_{15}} \wedge \overline{a_{14}} \wedge a_{13} \wedge \overline{a_{12}} \wedge a_{11})}$.

2.2.3 Part 3

For this part a 1024x4 memory chip (the Am91L14). As implied by its description this chip has only 10 address pins and 4 data pins. Although the unconnected address lines were used for glue logic, the superfluous data lines remained unused. This chip was meant to be enabled over the range `0x4000`–`0x4400`, which can be represented by the equation $\overline{(\overline{a_{15}} \wedge a_{14} \wedge \overline{a_{13}} \wedge a_{12} \wedge \overline{a_{11}} \wedge \overline{a_{10}})}$.

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;
}
```

5.2 Lab Schematic

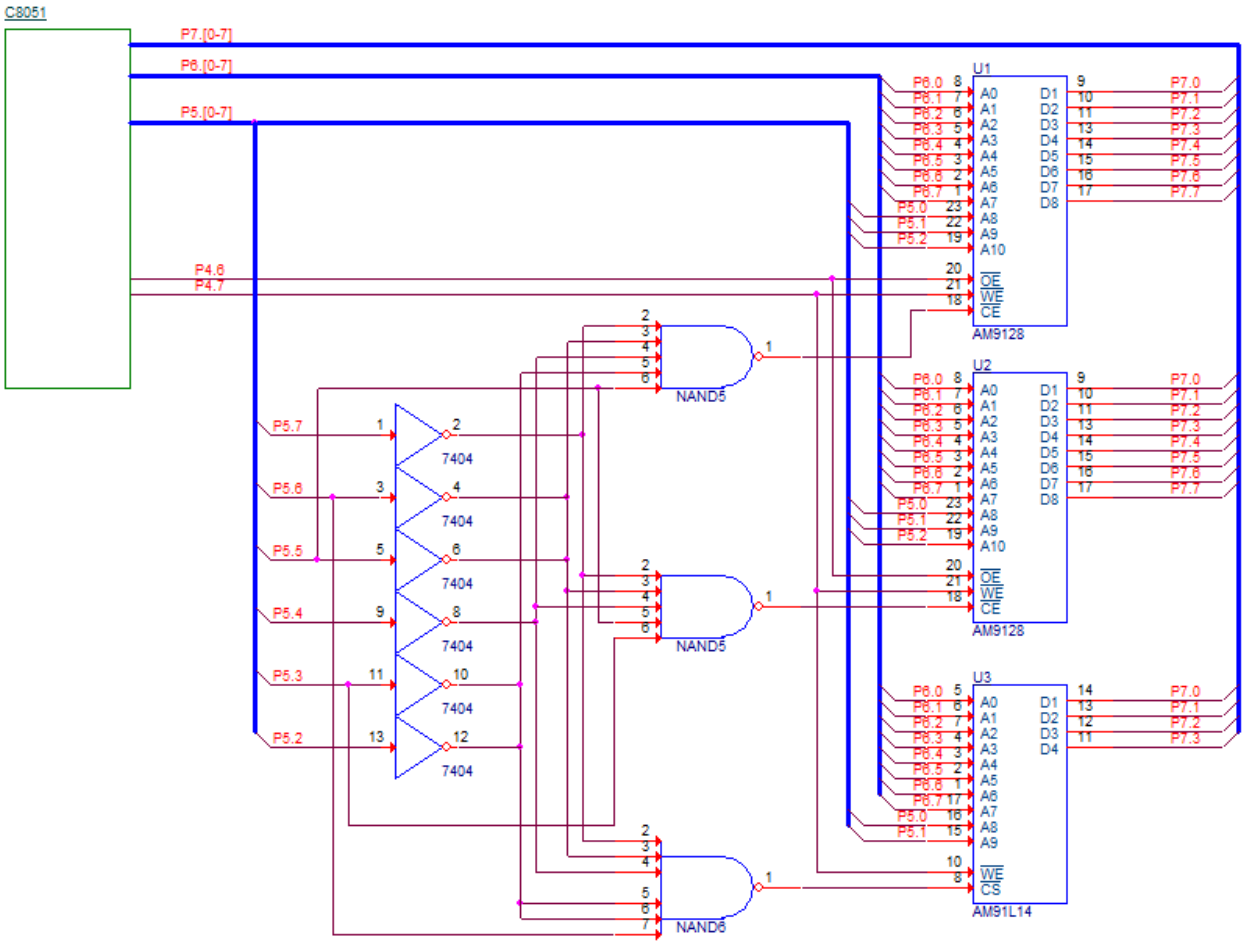

Figure 1: Circuit schematic for lab

5.3 Part 1

5.3.1 Code

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

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------
void main(void);
```

```
void SYSCLK_INIT(void);
void PORT_INIT(void);
void UART0_INIT(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main(void)
{
  __xdata unsigned char *int_ram;
  __xdata unsigned char *ext_ram;
  unsigned int memAddr;
  unsigned char c;
  unsigned char 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

  SFRPAGE = UART0_PAGE;               // Direct output to UART0

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

  // Write 'a' to on-board ram at address 0x1FF0
  int_ram = (__xdata unsigned char *)(0x1FF0);
  *int_ram = 'a';
  c = *int_ram;
  // Read back from on-board ram and write to external ram
  for(memAddr = 0x2000; memAddr < 0x2800; memAddr++)
    {
    ext_ram = (__xdata unsigned char *)(memAddr);
    *ext_ram = c;

    for(i=0;i<100;i++);
    printf("Value on external xram: 0x%x\t0x%4x\n\n\r",*ext_ram,memAddr);
  }
  while(1);
}


//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// 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 Current SFR page

  SFRPAGE = CONFIG_PAGE;
  XBR0 = 0x04;              // Enable UART0
  XBR1 = 0x00;
  XBR2 = 0x40;            // Enable Crossbar and weak pull-up
  P0MDOUT |= 0x01;          // Set TX0 pin to push-pull
  P4MDOUT = 0xFF;           // Output configuration for P4 all pushpull
  P5MDOUT = 0xFF;           // Output configuration for P5 pushpull EM addr
  P6MDOUT = 0xFF;           // Output configuration for P6 pushpull EM addr
  P7MDOUT = 0xFF;           // Output configuration for P7 pushpull EM data

  // EMI_Init, split mode with banking
  SFRPAGE = EMI0_PAGE;
  EMI0CF = 0x3b;
  EMI0TC = 0xFF;

  SFRPAGE = SFRPAGE_SAVE;      // Restore SFR page
}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1
//
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
}
```

5.4 Part 2

5.4.1 Code

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

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------
void main(void);
void SYSCLK_INIT(void);
void PORT_INIT(void);
void UART0_INIT(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main(void)
{
  __xdata unsigned int *error_ptr;
  __xdata unsigned char *ext_ram;
  unsigned static int __xdata count[512];
  unsigned int size = 0;
  unsigned int memAddr;
  unsigned char c;
  unsigned char 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

  SFRPAGE = UART0_PAGE;                 // Direct output to UART0

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

  c = 0xAA;
  for(memAddr = 0x2800; memAddr < 0x3000; memAddr++)
    {
    ext_ram = (__xdata unsigned char *)(memAddr);
    *ext_ram = c;

    for(i=0;i<100;i++);
    printf("Value on external xram: 0x%x\t0x%4x\n\n\r",*ext_ram,memAddr);    // read back
        value
  }
  error_ptr = count;
  c = 0x55;
  for(memAddr = 0x2800; memAddr < 0x3000; memAddr++)
    {
    ext_ram = (__xdata unsigned char *)(memAddr);
    *ext_ram = c;
```

```
    for(i=0;i<100;i++);
    printf("Value on external xram: 0x%x\t0x%4x\n\n\r",*ext_ram,memAddr);    // read back
        value
    if(*ext_ram != 0x55){           // if expected value is not read back add faulty address
         to array
      *error_ptr = memAddr;
      error_ptr += 1;
      size += 1;
    }
    if(size == 512){              // if error array is full empty contents to terminal
      for(size;size!=0;size--){
        pritnf("0x%4x\n\r",count[size-1]);
        error_ptr = count;
      }
    }
  }

  for(size;size!=0;size--){             // print error array
    printf(pritnf("0x%4x\n\r",count[size-1]););
  }
  while(1);
}


//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// 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 Current SFR page

  SFRPAGE = CONFIG_PAGE;
  XBR0 = 0x04;              // Enable UART0
  XBR1 = 0x00;
  XBR2 = 0x40;            // Enable Crossbar and weak pull-up
  P0MDOUT |= 0x01;          // Set TX0 pin to push-pull
  P4MDOUT = 0xFF;           // Output configuration for P4 all pushpull
  P5MDOUT = 0xFF;           // Output configuration for P5 pushpull EM addr
  P6MDOUT = 0xFF;           // Output configuration for P6 pushpull EM addr
  P7MDOUT = 0xFF;           // Output configuration for P7 pushpull EM data

  // EMI_Init, split mode with banking
  SFRPAGE = EMI0_PAGE;
  EMI0CF = 0x3b;
  EMI0TC = 0xFF;

  SFRPAGE = SFRPAGE_SAVE;      // Restore SFR page
}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1
//
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
}
```

5.5 Part 3

5.5.1 Code

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

```
//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------
void main(void);
void SYSCLK_INIT(void);
void PORT_INIT(void);
void UART0_INIT(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main(void)
{
  __xdata unsigned char *ext_ram;
  unsigned int memAddr;
  unsigned char c;
  unsigned char a[16];
  unsigned char 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

  SFRPAGE = UART0_PAGE;                 // Direct output to UART0

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

  for(i=0;i<16;i++){           // Initialize array with 0-15
    a[i] = i;
  }


  for(i=0, memAddr = 0x4000; i<16; i++,memAddr ++)    // Write contents of array to 16
      addresses starting at 0x4000
    {
    ext_ram = (__xdata unsigned char *)(memAddr);
    c = a[i];
    *ext_ram = c;

    for(c=0;c<100;c++);
    printf("Value on external xram: 0x%x\t0x%4x\n\r",*ext_ram,memAddr);     // read back
        value
  }
  while(1);
}


//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// 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 Current SFR page

  SFRPAGE = CONFIG_PAGE;
  XBR0 = 0x04;              // Enable UART0
  XBR1 = 0x00;
  XBR2 = 0x40;            // Enable Crossbar and weak pull-up
  P0MDOUT |= 0x01;          // Set TX0 pin to push-pull
  P4MDOUT = 0xFF;           // Output configuration for P4 all pushpull
  P5MDOUT = 0xFF;           // Output configuration for P5 pushpull EM addr
  P6MDOUT = 0xFF;           // Output configuration for P6 pushpull EM addr
  P7MDOUT = 0xFF;           // Output configuration for P7 pushpull EM data

  // EMI_Init, split mode with banking
  SFRPAGE = EMI0_PAGE;
  EMI0CF = 0x3b;
  EMI0TC = 0xFF;

  SFRPAGE = SFRPAGE_SAVE;      // Restore SFR page
}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1
//
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
}
```

5.6 Enhancement

5.6.1 Code

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

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------
void main(void);
void SYSCLK_INIT(void);
void PORT_INIT(void);
void UART0_INIT(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main(void)
{
  __xdata unsigned char *int_ram;
  __xdata unsigned char *ext_ram;
  unsigned int memAddr;
  unsigned char c,r;
  unsigned char i;
  unsigned long wait;

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

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

  SFRPAGE = UART0_PAGE;               // Direct output to UART0

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


  while(1){
    // Get user input and convert to 4 digit hex value
    printf("Enter a memory address (1FF0 - 1FFF) and a hex value (0-f) to input\n\r0x");
    memAddr = 0;
    for(i=0;i<4;i++){
      c = getchar();
```

```
      putchar(c);
      if(isdigit(c)){
        memAddr = memAddr*16 + (c-'0');
      }
      else if(c < 'g' && c > '`'){
        memAddr = memAddr*16 + (c-'W');
      }
      else if(c < 'G' && c > '@'){
        memAddr = memAddr*16 + (c-'7');
      }
      else{
        printf("\n\rBad input. Restart program.");
        while(1);
      }
    }
    printf(", 0x");
    // Get 4bit value from user
    c = getchar();
    putchar(c);
    if(isdigit(c)){
      c= (c-'0');
    }
    else if(c < 'g' && c > '`'){
      c= (c-'W');
    }
    else if(c < 'G' && c > '@'){
      c= (c-'7');
    }
    else{
      printf("\n\rBad input. Restart program.");
      while(1);
    }

    int_ram = (__xdata unsigned char *)(memAddr);
    *int_ram = c;
    r = *int_ram;

    printf("\n\rValue '0x%x' written to memory address 0x%4x.\n\r",*int_ram,memAddr);
    printf("Press a key to write '0x%x' to all external memory...\n\r", *int_ram);
    getchar();

    // Write value to all memory addresses (0x2000-0x4400)
    for(memAddr = 0x2000; memAddr < 0x4400; memAddr++)
      {

      ext_ram = (__xdata unsigned char *)(memAddr);
      *ext_ram = r;

      for(i=0;i<100;i++);
      printf("Value at address 0x%4x: 0x%x\n\r",memAddr,*ext_ram);
    }

    while(1);
  }
}


//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// 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 Current SFR page

  SFRPAGE = CONFIG_PAGE;
  XBR0 = 0x04;              // Enable UART0
  XBR1 = 0x00;
  XBR2 = 0x40;            // Enable Crossbar and weak pull-up
  P0MDOUT |= 0x01;          // Set TX0 pin to push-pull
  P4MDOUT = 0xFF;           // Output configuration for P4 all pushpull
  P5MDOUT = 0xFF;           // Output configuration for P5 pushpull EM addr
  P6MDOUT = 0xFF;           // Output configuration for P6 pushpull EM addr
  P7MDOUT = 0xFF;           // Output configuration for P7 pushpull EM data

  // EMI_Init, split mode with banking
  SFRPAGE = EMI0_PAGE;
  EMI0CF = 0x3b;
  EMI0TC = 0xFF;

  SFRPAGE = SFRPAGE_SAVE;      // Restore SFR page
}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1
//
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 5," in RPI ECSE Department, 2016. [Online]. Available: `http://www.rpi.edu/dept/ecse/mps/MPS_Lab_Ex5-Memory.pdf`. Accessed: Nov. 13, 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: Nov. 13, 2016.