Microprocessor Systems
Lab 6: Memory Interfacing

Nick Choi Samuel Deslandes

12/05/16

1 Introduction

The overall goal of this lab was to become familiar with configuring the 8051 to utilize an LCD display and a keypad.

In the first section of the lab, a C program was written to create the user interface for the magic 8 ball through the ANSI terminal. The interface consists of four generic questions that the user can select in order to receive a randomized answer form the 8051. These questions include Yes/No questions, True/False questions, Days of the week questions, and lastly, random number questions. The questions could be selected by entering the appropriate keys on the keyboard connected to the ANSI terminal.

In the second section of the lab, the C program developed for section 1 was modified so that the user could see the answers to their questions on a LCD display. The answers to the users questions are displayed on the screen until a new answer is generated by the magic 8 ball.

In the third section of the lab, the C program used in sections 1 and 2 was further modified to incorporate a keypad as the means of user input rather than a keyboard. In order for the keypad to function properly, a digital logic circuit was constructed in order to properly trigger an external interrupt on the 8051. Additionally, the output of the keypad needed to be decoded by the 8051 to properly determine which button on the keypad was pressed.

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 this lab an electronic "Magic 8 Ball" was developed. The program generates responses to four types of questions that can be asked:

- Yes/No
- True/False
- Day of the week
- Random Number

In order to randomly select a response, Timer0 is is configured to continuously count, and is sampled when a response is required. Timer0 was configured as a 16-bit timer by setting the lower nibble of the TMOD SFR to `0x01`. Although Timer0 is a 16-bit timer, only the lower byte (stored in TL0) is sampled. The timer was also configured to use SYSCLK/12 as a base by clearing bits 0, 1, and 3 of the CKCON SFR.

When configuring ports, bits 0–3 of port 3 should be configured as inputs, and bits 4–7 as outputs. This is done by setting the 'P3MDOUT' SFR to `0xf0`, and the 'P3' SFR to `0x0F`.

In the main function, user input is retrieved using the getchar() function. A switch/case block is then used to handle the user input and generate the appropriate response. In the case of a binary response, such as for the 'Yes/No' or 'True/False' questions, TL0 is read and modded by 2. The result is then evaluated to be either 1 or 0 and a response is printed to the terminal accordingly.

To handle the 'Days of the week' question an array of strings declared as "const char*" was used to store each day of the week. The value held in TL0 is then modded by 7 and used to index the aforementioned array. In order to print the string, 'printf()' was used with the '%s' formatting keyword.

To handle the 'Random number' question case, first the user must be prompted for max and min values to designate a range of responses. This is done using 'getchar()'. To randomly select an integer within this range the following equation is used:

$$\text{min+TL0\%(max-min+1)}$$

The right side of the equation produces a number within the difference of the max and min values selected; adding the minimum value to this brings it back within the range.

In addition to the four cases corresponding to the types of questions that can be asked, there is also the default case, which is triggered when the input is invalid. In this case, the program notifies the user of the invalid input and waits for the next user input.

2.1.2 Part 2

The code for this section of the lab is an enhancement of the previous part, which prints output not only to the terminal, but also to an LCD screen. In order to print to the LCD screen, first the "LCD.h" and "LCD.c" files must be included; these files contain the functions "lcd_clear()" and "lcd_puts()" which are used to clear the LCD and write to the LCD respectively, as well as the "lcd_init()" function used to initialize the LCD. As mentioned before, the code for this section is the same as that of the previous section, with the addition of the "lcd_init()" function called together with the other init functions, and clearing the LCD and printing to it after printing to the terminal when outputting responses.

2.1.3 Part 3

The code for this section further enhances that of the previous section by using a keypad to receive user input, rather than the keyboard. The keypad is wired such that when a key is pressed, an external interrupt (/INT0) is triggered. To do this, /INT0 must be routed to port pin P0.2, which is done by setting bit 2 of the XBR1 SFR. Next, global interrupts and /INT0 must be enabled, which can be done using the bit-addressable variables 'EA' and 'EX0' respectively.

Once /INT0 has been triggered, the associated ISR must decode the signals on port 3 to determine which key has been pressed. The ISR starts by setting 'EX0' to 0, disabling interrupts on /INT0. Next, a flag variable is set, and the lower nibble of P3 is stored in a

variable called “keyvalue” for use in decoding columns. This is done using a bitwise AND operation between P3 and `0x0F`.

At this point the rows can be decoded. To check if a key in the top row was pressed, P3 is set to `0x8F`, and after a short pause the lower nibble of P3 is evaluated. If it evaluates to `0x0F`, this is the row that was selected, and the ISR moves on to decode the columns. If this is not the case, then this process is repeated setting P3 to `0x4F`, `0x2F`, or `0x1F` for each of the subsequent rows.

Decoding columns is done by simply evaluating the “keyvalue” variable declared before the row decoding to determine which bit in its lower nibble has a value of 0. This is done using the comparison operator with the values `0x07`, `0x0B`, `0x0D`, and `0x0E` corresponding to the columns of the keypad from left to right. Once the selected column has been identified, a character can be assigned to a global variable “asciichar” for use in the main function. Figure 2 in the appendices section can be used as a reference for which row and column intersection corresponds to what character.

Once ”asciichar“ has been assigned P3 is reset to `0x0F`, and after another pause /INT0 is re-enabled and the program returns from the ISR.

In the main function, rather than using ‘getchar()’ to get user input, a new function “getkeychar()” was written. This function waits for the flag variable set in the /INT0 ISR to go high, resets the flag, then returns the “asciiflag”. The rest of the main algorithm is the same as it was in the previous section.

2.2 Hardware

The hardware for this lab involved interfacing the 8051 with a LCD screen in part 2, and a keypad in part 3.

2.2.1 Part 1

No hardware was required for this section.

2.2.2 Part 2

In this section a LCD screen was wired to the 8051. A schematic for this can be viewed in Figure 1 in the appendices section below. The LCD module used for this lab had 14 pins. Pins 1 and 2 are connections to ground and power, respectively. Pin 3 controls the screen contrast, which can be controlled using a potentiometer which comes as a part of the module. Pins 4, 5, and 6 are the register select, read/write select, and enable control signals, which were connected to P7.0–P7.2 on the 8051. Since port 7 was left in open-drain mode, a 10kohm pull-up resistor was placed between P7.2 and power. Pins 7 to 14 are data pins, which were connected to all of port 6 on the 8051. Port 6 should be configured as open-drain with high-impedance on every pin. This does not have to be done explicitly in the code because this can be accomplished using the associated SFRs’ default values.

2.2.3 Part 3

In this section a keypad was wired to the 8051. A schematic can be viewed in Figure 2 in the appendices section below. The keypad requires 8 wires, 4 for the columns and 4 for the rows, each of which are connected to port 3 on the 8051. The columns are connected to the lower nibble of port 3 (P3.0–P3.3) which is configured as open-drain for input. These wires are held high using a 3.3V and 10kΩ pull-up resistors, and are also used as inputs to a four input and gate. The output of this gate is used as the /INT0 interrupt source and is connected to pin 2 of port 0 (P0.2). The rows are connected to the high nibble of port 3 (P3.4–P3.7), which is configured as push-pull for output. These wires are held low by the software.

3 Results

By completing section one of the lab, a C program was developed that would create a user interface for the magic 8 ball using the ANSI terminal. By completing section two of the lab, a C program was developed to send the magic 8 ball's answers to a LCD display. By completing section three of the lab, a C program was developed in order to interface a keypad with the magic 8 ball. The keypad replaced the keypad input and a glue logic circuit was constructed in order to properly interface with the keypad.

4 Conclusion

Our end results matched well with the general goals for this lab however there were several instances where the system did not perform as expected. In section two of the lab, the messages that were being displayed on LCD would sometimes become corrupted. It was determined that this was the case because the display was not given enough time to properly output the entire message.

In section three of the lab, the terminal was randomly receiving duplicate button presses. The cause of this issue was that the keypad was not being sufficiently debounced. Therefore, the software debouncing period was increased in order to filter unintended button presses. Another approach to this issue could have been to implement a hardware debouncing circuit using logic gates or a simple RC circuit.

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 Part 1

5.2.1 Code

```
//-----------------------------------------------------------------------------
// Lab6-1
//-----------------------------------------------------------------------------
// Nick Choi, Samuel Deslandes
// This program acts as a "Magic 8-ball". The user inputs one of the 4 types of questions
// that can be asked and the program randomly outputs an appropriate response. The question
    categories
// are: Yes/No, True/False, Day of week, or random number
//-----------------------------------------------------------------------------
// 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);
void TIMER0_INIT(void);

//-----------------------------------------------------------------------------
// MAIN Routine
```

```
//-----------------------------------------------------------------------------
void main(void)
{
    char low;
  char high;
  char choice;
  char out;
  const char* days[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "
      Saturday"};

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

    PORT_INIT();                        // Initialize the Crossbar and GPIO
    SYSCLK_INIT();                      // Initialize the oscillator
    UART0_INIT();                       // Initialize UART0
  TIMER0_INIT();

    SFRPAGE = UART0_PAGE;               // Direct output to UART0

    printf("\033[2J");                  // Erase screen & move cursor to home position
  printf("Exit command is: <ESC>  \n\n\r");

  while(1)
    {
    printf("Select an option:\n\r1. Yes/No\n\r2. True/False\n\r3. Day of the week\n\r4.
        Number within range\n\n\r");
    choice = getchar();
    // Switch/Case block to handle each question category
    // TL0 is the value of Timer0, which is used to generate
    // pseudorandom numbers
    switch(choice){
      // Yes/No
      case '1':
        if(TL0%2 == 1){
          printf("Yes.\n\r");
        } else {
          printf("No.\n\r");
        }
        break;
      // True/False
      case '2':
        if(TL0%2 == 1){
            printf("True.\n\r");
          } else {
            printf("False. \n\r");
          }
        break;
      // Day of the week
      case '3':
        printf("%s\n\r",days[TL0%7]);   // Print random element from array containing days
            of the week
        break;
      // Random number
      case '4':
        printf("Enter min value followed by max value: ");
        low = getchar();            // Get min value
        printf("%c, ",low);
        high = getchar();           // Get max value
        putchar(high);

        out = low+TL0%(high-low+1);     // Get random number within range (inclusive)

        printf("\n\r%c\n\r", out);
        break;
      // Invalid input case
      default:
        printf("Invalid input\n\r");
```

```
    }
    }

}

//-----------------------------------------------------------------------------
// 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 = 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 on P0.0 pin to push-pull
    P1MDOUT |= 0x40;                    // Set green LED output P1.6 to push-pull

    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
}

// Timer init
void TIMER0_INIT(void){
  char SFRPAGE_SAVE;
  SFRPAGE_SAVE = SFRPAGE;

  SFRPAGE = TIMER01_PAGE;

  TMOD &= 0xF0;          // Timer0, Mode 1: 16-bit counter/timer.
  TMOD |= 0x01;
  TH0 = 0x00;          // Set high byte to 0
  CKCON &= ~0x0B;        // Timer0 uses SYSCLK/12 as base
  TL0 = 0x00;          // Set low byte to 0
  TR0 = 1;           // Start timer0

  SFRPAGE = SFRPAGE_SAVE;
}
```

5.3 Part 2

5.3.1 LCD Schematic

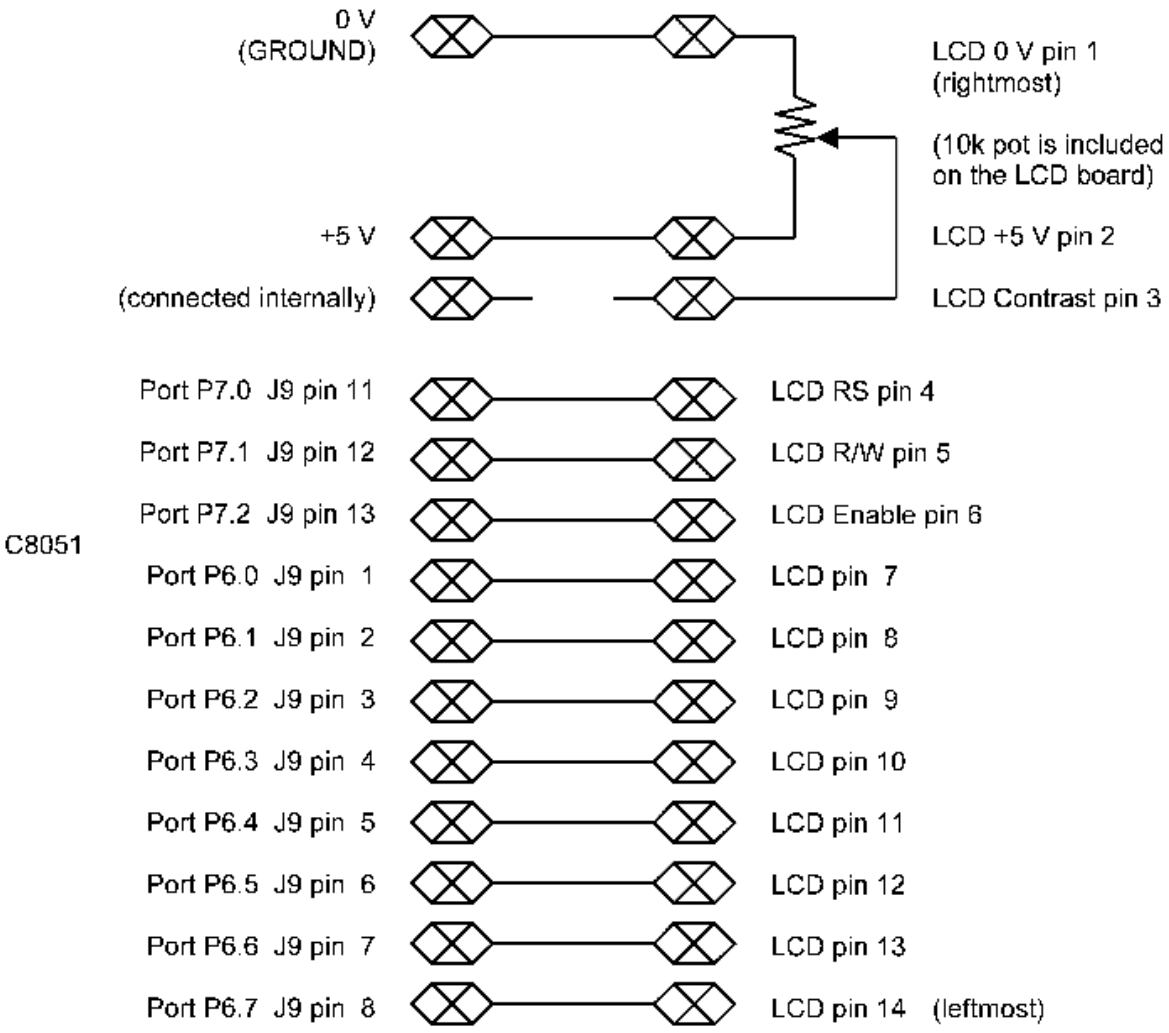

Figure 1: Circuit schematic for LCD[2]

5.3.2 Code

```
//-----------------------------------------------------------------------------
// Lab6-2
//-----------------------------------------------------------------------------
// Nick Choi, Samuel Deslandes
// This program is an extension of that of Lab6-1 in which program output is printed
// to an LCD screen.
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <c8051f120.h>
#include <stdio.h>
```

```
#include "putget.h"
#include "LCD.h"
#include "LCD.c"
//-----------------------------------------------------------------------------
// 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);
void TIMER0_INIT(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main(void)
{
    char low;
  char high;
  char choice;
  const char* days[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "
      Saturday"};
  char out[2];

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

    PORT_INIT();                        // Initialize the Crossbar and GPIO
    SYSCLK_INIT();                      // Initialize the oscillator
    UART0_INIT();                       // Initialize UART0
  TIMER0_INIT();              // Initialize TIMER0
  lcd_init();               // Initialize LCD

    SFRPAGE = UART0_PAGE;               // Direct output to UART0

    printf("\033[2J");                  // Erase screen & move cursor to home position
  printf("Exit command is: <ESC>  \n\n\r");

  while(1)
    {
    printf("Select an option:\n\r1. Yes/No\n\r2. True/False\n\r3. Day of the week\n\r4.
        Number within range\n\n\r");
    choice = getchar();
    // Switch/Case block to handle each question category
    // TL0 is the value of Timer0, which is used to generate
    // pseudorandom numbers
    switch(choice){
      // Yes/No
      case '1':
        lcd_clear();
        if(TL0%2 == 1){
          printf("Yes.\n\r");
          lcd_puts((char *) &"Yes");
        } else {
          printf("No.\n\r");
          lcd_puts((char *) &"No");
        }
        break;
      // True/False
      case '2':
        lcd_clear();
```

```
        if(TL0%2 == 1){
            printf("True.\n\r");
            lcd_puts((char *) &"True");
          } else {
            printf("False. \n\r");
            lcd_puts((char *) &"False");
          }
        break;
      // Day of the week
      case '3':
        lcd_clear();
        choice = TL0%7;           // get random index for array containing days of the week
        printf("%s\n\r",days[choice]);
        lcd_puts(days[choice]);
        break;
      // Random number
      case '4':
        printf("Enter min value followed by max value: ");
        low = getchar();
        printf("%c, ",low);
        high = getchar();
        putchar(high);

        //Array to store output. Last element is null char which signifies end of string.
        out[0] = low+TL0%(high-low+1);     // Get random number within range (inclusive)
        out[1] = '\0';

        printf("\n\r%c\n\r", out[0]);
        lcd_clear();
        lcd_puts(out);
        break;
      // Invalid input case
      default:
        lcd_clear();
        printf("Invalid input\n\r");
        lcd_puts((char*)&"Invalid input");
    }
    }

}

//-----------------------------------------------------------------------------
// 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 = 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 on P0.0 pin to push-pull
    P1MDOUT |= 0x40;                    // Set green LED output P1.6 to push-pull

    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
}

// Timer init
void TIMER0_INIT(void){
  char SFRPAGE_SAVE;
  SFRPAGE_SAVE = SFRPAGE;

  SFRPAGE = TIMER01_PAGE;

  TMOD &= 0xF0;          // Timer0, Mode 1: 16-bit counter/timer.
```

```
  TMOD |= 0x01;
  TH0 = 0x00;           // Set high byte to 0
  CKCON &= ~0x0B;         // Timer0 uses SYSCLK/12 as base
  TL0 = 0x00;           // Set low byte to 0
  TR0 = 1;            // Start timer0

  SFRPAGE = SFRPAGE_SAVE;
}
```

5.4 Part 3

5.4.1 Keypad Schematic

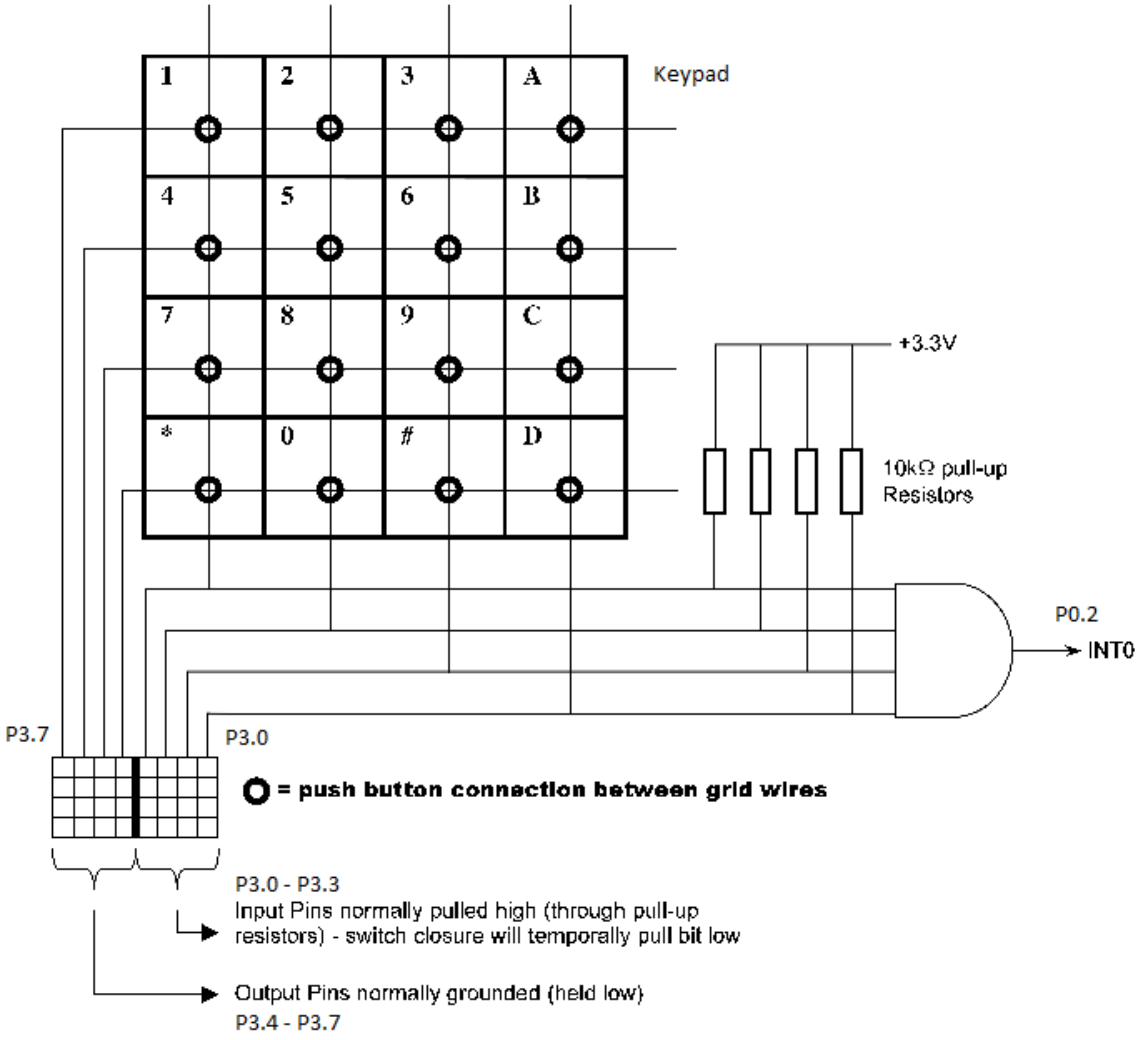

Figure 2: Circuit schematic for keypad[1]

5.4.2 Code

```
//-------------------------------------------------------------------------
// Lab6-3
```

```
//-----------------------------------------------------------------------------
// Nick Choi, Samuel Deslandes
// This program expands on that of 6-2 to get user input from a keypad, rather than the
    keyboard.
// Output is still printed to the LCD screen.
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <c8051f120.h>
#include <stdio.h>
#include "putget.h"
#include "LCD.h"
#include "LCD.c"
//-----------------------------------------------------------------------------
// Global Constants
//-----------------------------------------------------------------------------
#define EXTCLK      11059200            // External oscillator frequency in Hz
#define SYSCLK      11059200            // Output of PLL derived from (EXTCLK * 9/4)
#define BAUDRATE    115200              // UART baud rate in bps
char asciichar;
char portvalue;
char keyvalue;
int i;
char keyflag = 0;
//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------
void main(void);
void SYSCLK_INIT(void);
void PORT_INIT(void);
void UART0_INIT(void);
void TIMER0_INIT(void);
void KeypadVector(void) __interrupt 0;
char getkeychar(void);
//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main(void)
{
    char low;
  char high;
  char choice;
  const char* days[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "
      Saturday"};
  char out[2];

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

    PORT_INIT();                        // Initialize the Crossbar and GPIO
    SYSCLK_INIT();                      // Initialize the oscillator
    UART0_INIT();                       // Initialize UART0
  TIMER0_INIT();              // Initialize TIMER0
  lcd_init();               // Initialize LCD

    SFRPAGE = UART0_PAGE;               // Direct output to UART0

    printf("\033[2J");                  // Erase screen & move cursor to home position
  printf("Exit command is: <ESC>  \n\n\r");

  EX0 = 1;
  while(1)
    {
    printf("Select an option:\n\rA: Yes/No\n\rB: True/False\n\rC: Day of the week\n\rD:
        Number within range\n\n\r");

    choice = getkeychar();
    // Switch/Case block to handle each question category
```

```
    // TL0 is the value of Timer0, which is used to generate
    // pseudorandom numbers
    switch(choice){
      // Yes/No
      case 'A':
        lcd_clear();
        if(TL0%2 == 1){
          printf("Yes.\n\r");
          lcd_puts((char *) &"Yes");
        } else {
          printf("No.\n\r");
          lcd_puts((char *) &"No");
        }
        break;
      // True/False
      case 'B':
        lcd_clear();
        if(TL0%2 == 1){
          printf("True.\n\r");
          lcd_puts((char *) &"True");
        } else {
          printf("False. \n\r");
          lcd_puts((char *) &"False");
        }
        break;
      // Day of the week
      case 'C':
        lcd_clear();
        choice = TL0%7;           // get random index for array containing days of the week
        printf("%s\n\r",days[choice]);
        lcd_puts(days[choice]);
        break;
      // Random number
      case 'D':
        printf("Enter min value followed by max value: ");

        low = getkeychar();
        printf("%c, ",low);
        high = getkeychar();
        putchar(high);

        //Array to store output. Last element is null char which signifies end of string.
        out[0] = low+TL0%(high-low+1);     // Get random number within range (inclusive)
        out[1] = '\0';

        printf("\n\r%c\n\r", out[0]);
        lcd_clear();
        lcd_puts(out);
        break;
      // Invalid input case
      default:
        lcd_clear();
        printf("Invalid input\n\r");
        lcd_puts((char*)&"Invalid input");
    }
    }

}

// External Interrupt 0 ISR
void KeypadVector(void) __interrupt 0{
  EX0 = 0;             // Disable /INT0
  keyflag = 1;
  keyvalue = P3 & 0x0F;
  // Try first row
  P3=0x8F;             // check if row one (top) was active
  for(i = 0; i<400; i++);      // wait for the output and input pins to stabilize
  portvalue = P3 & 0x0F;       // read the value of the lower 4 bits
```

```
if (portvalue == 0x0F)      // if this row was selected then the value will be 0x0F
{
  if (keyvalue == 0x07){    // look at the value of the low 4 bits
    asciichar = '1';    // return the value of the matching key
  }
  else if (keyvalue == 0x0B){
    asciichar = '2';
  }
  else if (keyvalue == 0x0D){
    asciichar = '3';
  }
  else{
    asciichar = 'A';
  }

  P3 = 0x0F;          // put output lines back to 0
  while (P3 != 0x0F);     // wait while the key is still pressed
  for(i = 0; i<20000; i++); // wait for output and input pins to stabilize after key is
      released
  EX0 = 1;
  return;
}
// Try second row
P3=0x4F;            // check if row one (top) was active
for(i = 0; i<400; i++);     // wait for the output and input pins to stabilize
portvalue = P3 & 0x0F;      // read the value of the lower 4 bits
if (portvalue == 0x0F)      // if this row was selected then the value will be 0x0F
{
  if (keyvalue == 0x07){    // look at the value of the low 4 bits
    asciichar = '4';    // return the value of the matching key
  }
  else if (keyvalue == 0x0B){
    asciichar = '5';
  }
  else if (keyvalue == 0x0D){
    asciichar = '6';
  }
  else{
    asciichar = 'B';
  }

  P3 = 0x0F;          // put output lines back to 0
  while (P3 != 0x0F);     // wait while the key is still pressed
  for(i = 0; i<20000; i++); // wait for output and input pins to stabilize after key is
      released
  EX0 = 1;
  return;
}
// Try third row
P3=0x2F;            // check if row one (top) was active
for(i = 0; i<400; i++);     // wait for the output and input pins to stabilize
portvalue = P3 & 0x0F;      // read the value of the lower 4 bits
if (portvalue == 0x0F)      // if this row was selected then the value will be 0x0F
{
  if (keyvalue == 0x07){    // look at the value of the low 4 bits
    asciichar = '7';    // return the value of the matching key
  }
  else if (keyvalue == 0x0B){
    asciichar = '8';
  }
  else if (keyvalue == 0x0D){
    asciichar = '9';
  }
  else{
    asciichar = 'C';
  }

  P3 = 0x0F;          // put output lines back to 0
```

```
    while (P3 != 0x0F);      // wait while the key is still pressed
    for(i = 0; i<20000; i++); // wait for output and input pins to stabilize after key is
        released
    EX0 = 1;
    return;
  }
  // Try last row
  P3=0x1F;            // check if row one (top) was active
  for(i = 0; i<400; i++);      // wait for the output and input pins to stabilize
  portvalue = P3 & 0x0F;       // read the value of the lower 4 bits
  if (portvalue == 0x0F)       // if this row was selected then the value will be 0x0F
  {
    if (keyvalue == 0x07){     // look at the value of the low 4 bits
      asciichar = '*';     // return the value of the matching key
    }
    else if (keyvalue == 0x0B){
      asciichar = '0';
    }
    else if (keyvalue == 0x0D){
      asciichar = '#';
    }
    else{
      asciichar = 'D';
    }

    P3 = 0x0F;           // put output lines back to 0
    while (P3 != 0x0F);      // wait while the key is still pressed
    for(i = 0; i<20000; i++); // wait for output and input pins to stabilize after key is
        released
    EX0 = 1;
    return;
  }
}

// Function to wait for and return keypad input
char getkeychar(){
  while(!keyflag);
  keyflag = 0;
  return asciichar;
}

//-----------------------------------------------------------------------------
// 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  = 0x77;                     // Start ext osc with 11.0592MHz crystal /edit 0x67
    for(i=0; i < 256; i++);             // Wait for the oscillator to start up
    while(!(OSCXCN & 0x80));
    CLKSEL  = 0x01;
    OSCICN  = 0x00;

    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     = 0x04;                    // Enable UART0
    XBR1     = 0x04;          // INT0 routed to port pin P0.2
    XBR2     = 0x40;                    // Enable Crossbar and weak pull-up

  EA     = 1;           // Enable global interrupts

    P0MDOUT |= 0x01;                    // Set TX0 on P0.0 pin to push-pull
    P3MDOUT  = 0xF0;                    // Set P3 high nibble as output, low nibble as input
  P3     = 0x0F;          // P3 high nibble set to 0v

  SFRPAGE = TIMER01_PAGE;
  TCON  &= 0xFC;          // Clear INT0 flag and set for level triggered

    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     = 0xFA;          // 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
}

// Timer init
void TIMER0_INIT(void){
  char SFRPAGE_SAVE;
  SFRPAGE_SAVE = SFRPAGE;

  SFRPAGE = TIMER01_PAGE;

  TMOD &= 0xF0;          // Timer0, Mode 1: 16-bit counter/timer.
  TMOD |= 0x01;
  TH0 = 0x00;          // Set high byte to 0
  CKCON &= ~0x0B;        // Timer0 uses SYSCLK/12 as base
  TL0 = 0x00;          // Set low byte to 0
  TR0 = 1;           // Start timer0

  SFRPAGE = SFRPAGE_SAVE;
```

}

6 References

[1]"MPS Lab 6," in RPI ECSE Department, 2016. [Online]. Available: `http://www.rpi.edu/dept/ecse/mps/MPS_Lab_Ex6-Magic8Ball.pdf`. Accessed: Nov. 27, 2016.

[2]"Interfacing a Hitachi HD44780 to a Silicon Laboratories C8051F120," in RPI ECSE Department, 2016. [Online]. Available: `http://www.rpi.edu/dept/ecse/mps/LCD_Screen-8051.pdf`. Accessed: Nov. 27, 2016.

[3]"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. 27, 2016.