MPS

Magic 8 Ball Lab Exercise

Magic 8 Ball

Student's name & ID (1): _____

Partner's name & ID (2): _____

Your Section number & TA's name _____

Notes:

You must work on this assignment with your partner.

Hand in a printer copy of your software listings for the team.
Hand in a neat copy of your circuit schematics for the team.

These will be returned to you so that they may be used for reference.

---------- do not write below this line ----------

	#1 POINTS (1)	#1 POINTS (2)	TA init.	#2 POINTS (1)	#2 POINTS (2)	TA init.	#3 POINTS (1)	#3 POINTS (2)	TA init.
Grade for performance verification (50% max.)									
Grade for answers to TA's questions (20% max.)									
Grade for documentation and appearance (30% max.)									
			TOTAL			TOTAL			TOTAL

Grader's signature: _____

Date: _____

Part 1: Introduction to the Magic 8 Ball

GOAL

By doing this lab assignment, you will learn:
1. Review of basic serial I/O to the terminal.
2. Logic programming

PREPARATION

- Review the C language studio utilities.
- Review the 8051 hardware port and interrupt features
- Develop a flowchart for the program you need to write. This will be signed off by your TA and must be included when you turn in your code for the lab.
- Write a C program that is free from syntax errors (i.e., it should assemble without error messages).

INTRODUCTION TO THE MAGIC 8 BALL

Programming Tasks

For this lab you will be developing an electronic Magic 8 Ball. You will write a program that generates answers to different kinds of questions. The user selects the type of answer that is desired and the answer is displayed on the screen.

There are four different types of answers that can be generated. They are:
1. Yes/No: This will output either a yes or no answer.
2. True/False: This will output either true or false.
3. Day of the week: This will randomly select a day of the week.
4. Random Number: This will select a random integer number within any range stated by the user (e.g. a random number between 3 and 7).

The program should have a menu that allows the user to select which type of answer will be generated. After the answer is displayed the program should return to the menu. It is not necessary to clear the screen when returning to the menu. The answers that are generated are to be determined randomly. The section below provides a suggestion for developing random values. In order to earn full credit on the project, the answers must be generated randomly. You should first develop a flowchart for the program you need to write. This will be signed off by your TA and must be included when you turn in your code for the lab.

The code must be logically written and well commented. This is very important since you will be using this code as the basis of Parts 2 and 3. After being signed off by the TA you will turn in your flowchart from the preparation work and the fully commented copy of your code.

Helpful Hints

Here are several suggestions to get you started on the project:
1. Start small. Break the project up into several functions and begin by only debugging one at a time.
2. Random number generation. The rand() function can be used to generate a random number. You can also use a free running counter and read in a value, which will be fairly random as well.

Part 2: Interfacing an LCD Screen to the 8051

GOAL

By doing this lab assignment, you will learn to:
1. Interface the C8051F120 to a Hitachi HD44780 2-line LCD display.
2. How the LCD screen operates.
3. Print results on the LCD screen. This will be used to display the answers generated in Part 1.

PREPARATION

- Read Hitachi HD44780U LCD Controller manual
- Become familiar with the different LCD screen functions included in LCD.c & LCD.h and the example LCDTEST8051.c (http://www.rpi.edu/dept/ecse/mps/mps_handouts.html).
- Rewrite your code from Part 1 so that the answers from the program are displayed on the LCD screen. The menu used to select the answer type will still displayed on the UART terminal screen.

INTRODUCTION TO THE LCD SCREEN

Programming Tasks

For this lab you will be adding an LCD screen to the Magic 8 Ball system. This screen will be used to display the answers that are generated by the program. When an answer is written to the screen, the screen should first be cleared and then the answer printed. The answer should remain on the screen until a new answer is generated. A list of the LCD functions is in the LCD.h header file.

Helpful Hints:

Here are several suggestions to help you with the lab.
1. Remember that all characters to be sent to the screen must be sent as ASCII. This can be done by creating a `char *` with the character string or by passing in each character separately as an ASCII character.
2. Don't forget to include LCD.h (found on the course web page) in your program.
3. Remember to initialize the screen and make sure you have set the format type (number of lines and character size) to the desired value before trying to use the screen.
4. There are several ways of passing characters to the lcd_puts() function:

```
   char buffer[] = "Hello.";
   lcd_puts(buffer);   //But the 2008 SDCC compiler doesn't allow
                       //lcd_puts(buffer[]);
OR
   lcd_puts((char *) &"Hello.");
OR
   char buf;
   buf = 'H';
   lcd_puts((char *) &buf);
```

5. Add delays between LCD commands in program if the display doesn't respond correctly every time.

Part 3: Interfacing a Keypad to the 8051

GOAL

By doing this lab assignment, you will learn to:
1. Interface the C8051F120 to a Grayhill 16-button keypad.
2. Develop interrupt-activated software to decode key presses on the keypad.

PREPARATION

- References: *C8051F12x-13x Reference Manual*, Ch. 13, 14, 23, 24
- Write a header file, called keypad.h that has the interrupt initialization routine and Interrupt Service Routine.
- Develop a new version of your code from Part 2 that incorporates the header file and uses the keypad for all user input.
- Write a flow chart for the operation of the Interrupt Service Routine.
- Create a circuit schematic to handle the interrupt generation

INTRODUCTION TO THE KEYPAD

Programming Tasks

This part of the lab adds a keypad to the program that was developed in Part 2. The keypad is connected to the microcontroller by the External Interrupt 0. This will allow the keypad to operate on an interrupt-based method. The code for controlling the keypad will be located in a header file called keypad.h. This will allow the file to be easily transported to different programs.

The Grayhill keypad is an industry standard device that was designed to work with another standardized microcontroller port with a built-in hardware interrupt capability called a Key Wakeup. For historical perspective, each bit of the port supporting Key Wakeup can be used to generate a Key Wakeup interrupt when either a rising or falling edge is detected. It is important to note that even though each bit can generate an interrupt independently of the others, the same interrupt will be called regardless of which bit triggered it. The Key Wakeup Interrupt for each individual bit is enabled using a SFR. Writing a 1 to a bit in the register will enable the corresponding bit of the port to generate an interrupt when a falling edge is detected. The flag bits for the Key Wakeup interrupt are located in another SFR. Multiple flags can be set at the same time, although software must be written in order to determine which flags have been set. The flags are cleared by writing a one to the flag bits that have been set. It is a good idea to clear the flags before the Key Wakeup Interrupt is enabled to prevent any false triggers.

Unfortunately the C8051 does not support this standard hardware interface. To implement the same functionality, an extra logic device must be added to the hardware to permit keypad hits to generate interrupts. The ANDing of the keypad input lines (4 LSBits) creates a suitable signal for generating interrupts when connected to INT0.

You may want to consider using a keypad for your final project. One idea for this would be, instead of returning the value to the main program, to have the buttons call different functions, which in effect allows the keypad to operate as a more advanced flexible version of the IRQ.

Required Additional Hardware

- Add four 10kΩ pull-up resistors on P3.3-0 to the +3.3V supply.
- An ANDing operation of the four bits P3.3-0 will be used to generate an INT0 interrupt. A 4-input AND gate is made from 3 2-input 7408 DIP AND gates available in the lab.

Required Software Setup

- Port 3 pins P3.3-0 must be configured as inputs: Open-drain option on P3MDOUT and set data values to 1's (P3 = 0x0F).
- Interrupt INT0 is used to detect a key press on the keypad. The key decode subroutine must be assigned to INT0.
- The interrupt works better when set for level triggering rather than edge triggering.
- The keypad switches are noisy so delays must be added to wait for the signals to settle. Delays must also be used when switching the outputs on and off to allow the slow signals to settle.
- It will be necessary to use the "xdata" keyword on many variables to move them to external memory where there is more room.

Helpful Hints

- 1. You should use port P3 for the keypad.
- 2. Remember to assign the ISR vector to INT0 used for the "Key Wakeup" with:
-
- `void KeypadVector(void) interrupt 0`
-
- 3. Remember to clear the flag for INT0 triggered by the keypad hardware.
- 4. You must include C8051F120.h in the header file. If you do not include this then the compiler will not recognize any of the register names for the F120 SFRs. You must include putget.h, or LCD.c & LCD.h if you want to use any of those I/O functions.
- 5. Be sure to add the external 10kΩ pull-ups and configure the output pins as push-pull & the input pins as open-drain. The keypad may work (intermittently) without the pull-up resistors. In general, the rise & fall times of the voltage pulses on the data line are a little slow and the pull-ups improve the signals at the expense of increasing the current draw and power requirements of the system. See the sample code below.
- 6. Remember the column select bit values are 1,2,4,8 not 1,2,3,4.

KEYPAD INITIALIZATION FRAGMENT

Use SYSCLK set to 22.1184MHz from the external crystal and the following:

```
P3MDOUT=0xF0;           // hi nibble to push-pull, lo nibble to open-drain
P3=0x0F;                // write 0's to Port 3 hi nibble, lo nibble set for input

TCON=TCON & 0xFC;       // Clear INT0 flag and set for level triggered
IE=IE|0x81;             // Enable all interrupts & enable INT0
```

KEYPAD DECODING FRAGMENT

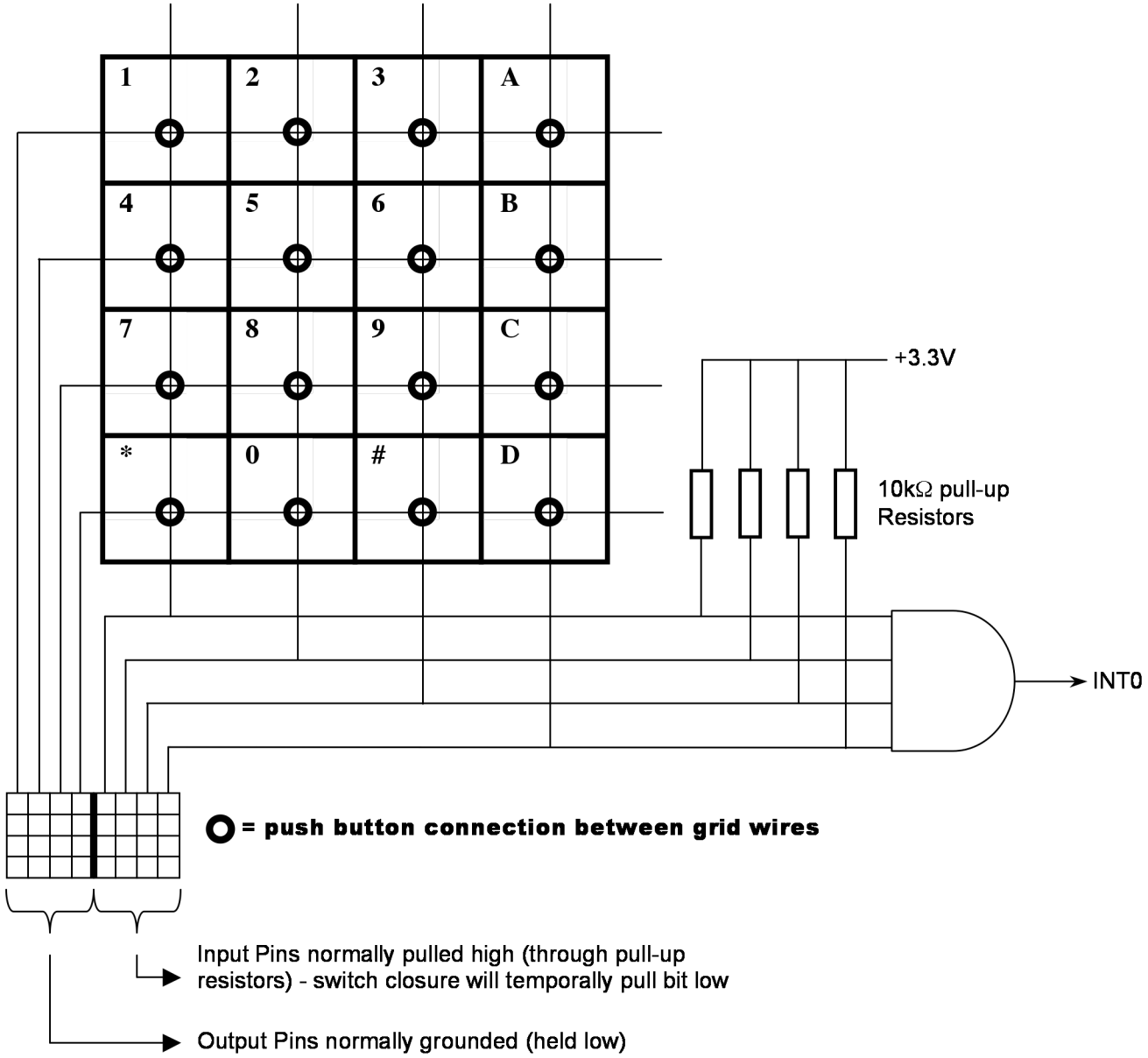
Once an interrupt has been detected, reading P3 will indicate which column the pressed key is in. Switching the 4 outputs on one at a time will determine which row contains the pressed key and the combination of the row and column will determine the precise key that was pressed. Global variables are necessary to pass values from ISRs back to the main program. Here asciichar is declared to be global.

```
P3=0x8F;                    // check if row one (top) was active
for(i = 0; i<300; 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
                            // since the 1 on bit 7 will allow the 4 inputs to be hi
{
   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';
}
else ...                    // must be another row
```

Before returning from the interrupt routine, make sure the key has been released and then delay a little before re-enabling interrupts to prevent any stray bounces from retriggering the interrupt.

```
   while (P3 != 0x0F);      // wait while the key is still pressed
   for(I = 0; i<10000; i++);// wait for output and input pins to stabilize
                            // after key is released
   IE = IE|0x81;            // enable INT0 interrupt
```

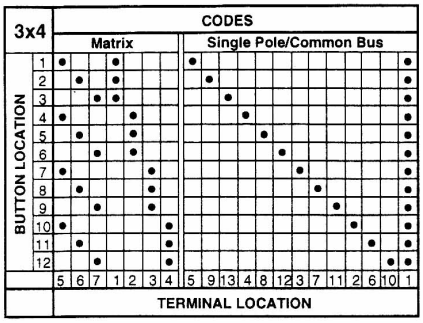


Keypad wiring diagram - The keypad is a passive device with only switches that connect wires on the crossbar matrix of 4 x 4 wires.

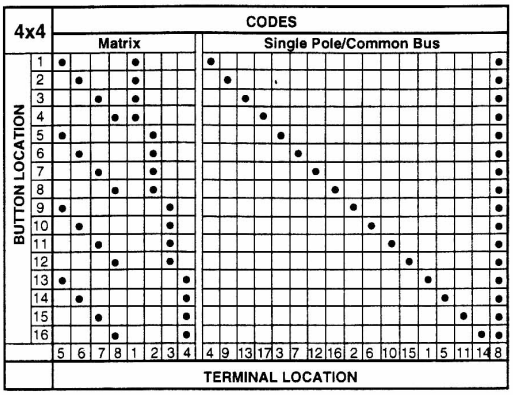
CODE AND TRUTH TABLES

Dots in the chart indicate connected terminals when switch is closed.
Terminals are identified on the keyboard.

12 Button Keypads



16 Button Keypads



SPECIFICATIONS

Rating Criteria

Rating at 12 Vdc: 5 milliamps for .5 seconds
Contact Bounce: < 12 milliseconds
Contact Resistance: < 100 ohms (@ stated operating force)
Voltage Breakdown: 250 Vac between components
Mechanical Operation Life: 3,000,000 operations per key
Insulation Resistance: > 10^{12}ohms @ 500 Vdc
Push Out Force Per Pin: 5 lbs.

Operating Features

Travel: .040 minimum
Operating Force: 175 ± 40 grams
Operating Temperature: -30°C to +80°C

Material and Finishes

Terminal Pin: Phosphor bronze, solder plated
PC Board: FR-4 glass cloth epoxy
Keypad: Silicone rubber, durometer 50 ± 5
Housing: ABS, cycolac "KJW"
Housing Color: Black

STANDARD LEGENDS

12 Position

-102 Legend: *Black legends on white button*
-152 Legend: *White legends on black button*

16 Position

-006 Legend: *Black legends on white button*
-056 Legend: *White legends on black button*

See page C-4 for Standard Legend configurations.

ORDERING INFORMATION

96AB2-102-F

Grayhill Series Number

Size Option
A=3x4
B=4x4

Circuitry
B2=Matrix (terminal pin header)
C2=Single Pole/Common Bus

Mounting Option
F=Front panel mount
R=Rear panel mount

Standard Legend Choices
12 Position legends
102=Black legends on a white button
152=White legends on a black button
16 Position legends
006=Black legends on a white button
056=White legends on a black button

Available from your local Grayhill Distributor
For prices and discounts, contact a local Sales Office, an authorized local Distributor, or Grayhill.

Grayhill An ISO 9001 Company
561 Hillgrove Avenue • LaGrange, Illinois 60525 • USA • Phone: (708) 354-1040 • Fax: (708) 354-2820 • http://www.grayhill.com