MPS

Magic 8 Ball Lab Exercises

Magic Eight Ball 1, 2, & 3

Student's name & ID: _____

Partner's name & ID: _____

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

Grader's signature: _____

Date: _____

Lab 1: Introduction to the 68HC12 - The Magic 8 Ball

Overview:

The 68HC12 is a 16-bit microprocessor that is the follow on to the 6811 that was used in COCO. The 6812 is far more powerful than the 6811. It provides more support of interrupts, full 16 bit addressing and several new features not found on the 6811. The objective of this lab is to introduce you to programming the 6812 using C. This will be done by writing a small program, which will be modified in later labs and used as the basis of a larger project.

For this lab you will create a small program that is used to generate answers to questions. The user will be able to input information via the terminal keyboard and the answers will be displayed on the screen. In the later labs you will add a keypad for the input and an LCD for displaying the answers.

Preparation:

The preparation for this lab is very simple. You should have a copy of the 68HC12 User's Manual, which was provided in class. This will be used as a reference for programming the 6812 and contains sample code that may be useful to you in your projects. You should read the sections on programming the 6812 and using the DBug12 monitor before coming to lab.

Prelab:

1. Read the 68HC12 User's Manual.

2. 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.

3. A complete C program that meets all specifications stated in the project description.

Project Description:

For this project 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.

The code must be logically written and well commented. This is very important since you will be using this code as the basis of Labs 2 and 3.

After being signed off by the TA you will turn in your flowchart from the prelab 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. Remember all Debug 12 functions must be prefaced by DB12->

3. Remember your main function must be called __main.

4. Random number generation. The free running counter _H12TCNT is a good source of random numbers. This has a value from 0000 to 0xFFFF. There are several different ways this can be used. How you use it is up to you.

5. To avoid possible compiler errors, functions and function prototypes should use void for the variables to be passed and the return type if there is nothing being passed to the function or returned by the function.

Lab 2: Interfacing an LCD Screen to the 68HC12

Objective:

The objective of this lab is to learn how an LCD screen operates and to interface it to the 68HC12 by using a header file that contains the interface routines. The screen will be used to display the answers generated by the magic eight ball program.

Preparation:

Read the LCD screen manual (Interfacing a Hitachi HD44780 to a Motorola 68HC11 or Motorola 68HC12).

Prelab:

1. Become familiar with the different LCD screen functions included in LCD12.H (http://www.ecse.rpi.edu/Courses/CStudio/hc12/sampleprograms/).

2. Rewrite your code from Lab 1 so that the answers from the program are displayed on the LCD screen. The menu used to select the answer type is still displayed on the computer screen.

Project Description:

For this lab you will be adding an LCD screen to the Magic 8 Ball. 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. The different LCD functions are in the LCD12.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 LCD12.H in your program.

3. Remember to initialize the screen before trying to use it and that you have set the format type (number of lines and character size) to the desired value.

Lab 3: Interfacing a keypad to the 68HC12

Objectives:

The objective for this lab is to introduce a new feature of the 6812, the Key Wakeup interrupt and to demonstrate how to interface a keypad to a microprocessor. This lab will also show you how to write your own header file for use with a microcontroller.

Preparation:

Read the section of the 68HC12 User's Manual that covers interrupts and the Key Wakeup interrupt.

Prelab:

1. Write a header file, called keypad.h, that has the Key Wakeup Interrupt initialization routine and Interrupt Service Routine.

2. Develop a new version of your code from Lab 2 that incorporates the header file and uses the keypad for all user input.

3. Write a flow chart for the operation of the Key Wakeup Interrupt Service Routine.

Project Description:

This lab adds a keypad to the program that was developed in Lab 2. The keypad is connected to the microcontroller by the Key Wakeup Interrupt of Port J. 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 keypad used in the lab is a Greyhill series 96 4x4 keypad. There are 8 pins connected to this keypad. Four are used for the row select and 4 are used for the column select. The 4 row select pins are normally connected to 4 port outputs bits that are held low. When a button is pressed, one of the column select pins will then go low, which will trigger the Key Wakeup Interrupt. The Key Wakeup Interrupt must then determine which row was activated and return the correct character for that button.

The code will consist of two functions. One will be used to initialize the Key Wakeup Interrupt and the other will be the Key Wakeup ISR.

The initialization function must do the following:
1. Set Port J to detect falling edges.
2. Select Pull-ups for Port J.
3. Enable pull-ups.
4. Enable the Port J bits that will read in the column selects.
5. Set the row select bits for output.

6. Write lows to the row select bits.

The Key Wakeup ISR must do the following:

1. Save the state of the Key Wakeup Flags.

2. Get the value of the 4 low order bits on Port J. (Hint: use a mask. The low bit in the 4-bit value will tell you which column contains the key that was pressed but not the row of the key. Note that the interrupt flag register should contain the same information. Of the two, which one will latch and which one may have a problem with contact bounce?)

3. Parse through the different rows to determine which key was pressed.

This step is the most complicated step of the process. When a key is pressed the value on the low bits determines which column was activated. Since the keypad we are using does not have a common ground for all the keys, the ground that is seen on the low bit of Port J is passed in through the row select bits. In order to determine which row is active the program must write a high to each row individually and then look at the value on the low nibble of Port J. If the row you write a one to is the row in which the button is pressed then the value on the low bits of Port J will be (----1111). Your program needs to write a one to each row and look for the (1111). When the (1111) is detected you will then look at the original value of the low bits to determine which key was activated.

For example if row 1 is active when you see (1111) and the low bits of Port J were previously (0111), then the key that was pressed was number 1 (assuming the pins for the leftmost column, containing the ‘1’ key, has been wired to bit 3 of Port J).

4. You must have a global variable that is included in your header file. This variable will be used to pass the value of the key back to the main program. The value of the key should be stored in this variable so the main program can read it.

5. Reset Port J to all low outputs (although only the high nibble is actually used for output).

6. Wait for the key to be released.

7. Clear the Key Wakeup Flags by writing a 1 to the bit in the flag register (KWIFJ) that corresponds to the interrupt you just processed. Good programming practice would not automatically set all the bits since later revisions of the program may use other interrupts that may happen simultaneously but be processed by other ISRs. A particular ISR should only clear the bit of the interrupt it specifically handles.

Helpful Hints:

1. You must use port J.

2. Remember to assign the vector for the Key Wakeup using DB12->SetUserVector

3. Remember to clear the flag for the Key Wakeup hardware.

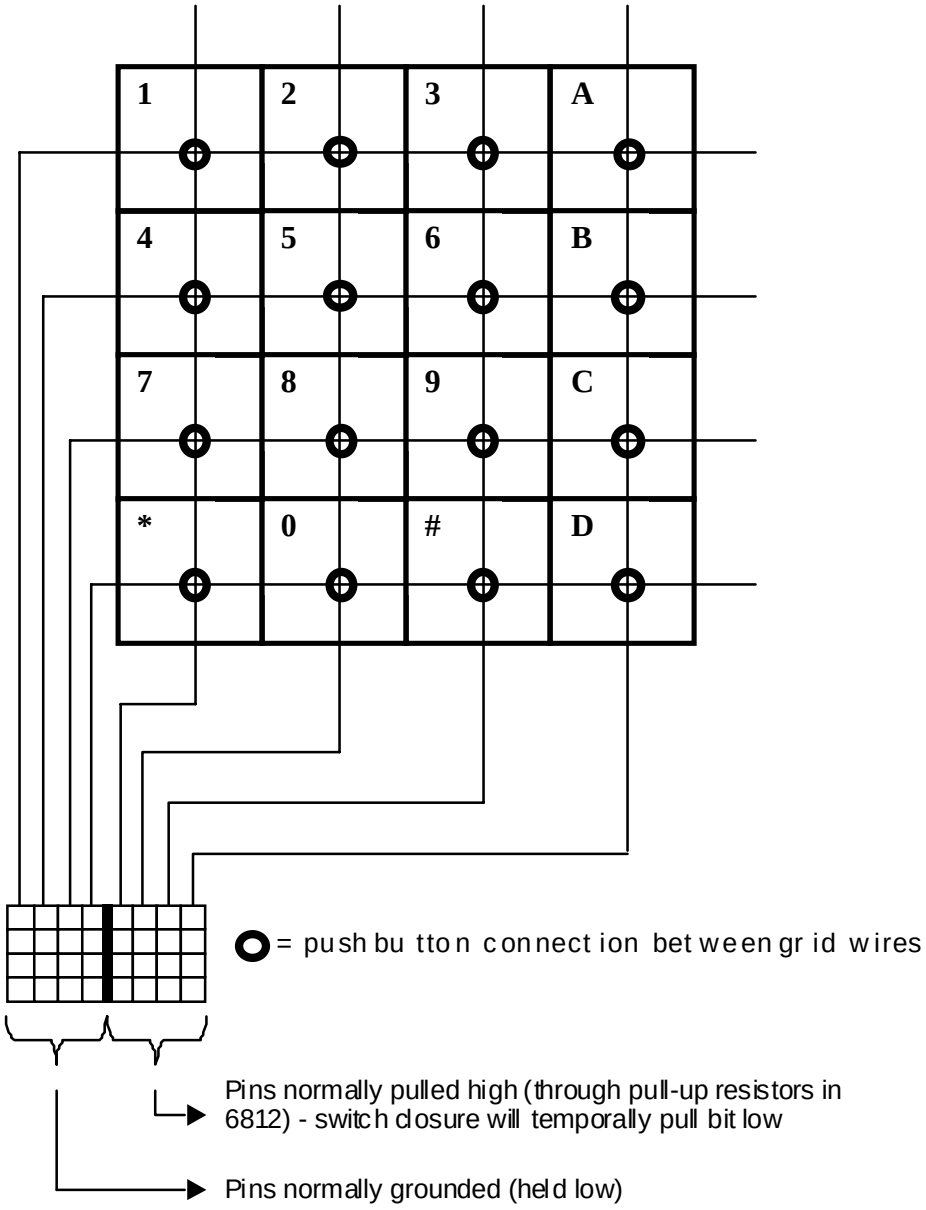
4. You must include HC812A4.H in the header file. If you do not include this then the compiler will not recognize any of the register names for the 6812 registers. You must include DBUG12.H if you want to use any of the dbug12 functions.

5. Be sure to enable the pull-ups AFTER you specify the use of pull-ups. See the sample code for this in the 68HC12 User's Manual.

6. Remember the column select values are 1,2,4,8 not 1,2,3,4.

Notes:

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.



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