Lab 3 - Microcontroller

due October 31, 1997

Introduction

You will program, assemble and test a microcontroller-based circuit that displays your student number in binary. The device must operate according to the specifications given below. You will be supplied with parts that you will use to build the system on a prototyping board in the lab. You will write the control program in C and program the microcontroller using a device programmer. You will hand in a programmed device and a program listing. The TA will use your programmed microcontroller chip to test your design.

Specifications

Inputs and Outputs

The device has four outputs. Each output drives an LED. The LEDs display a digit of your student number in binary. The device has one input: a pushbutton switch that is used to command the device to display the next digit of your student number.

The microcontroller's RESET input should be connected to a switch or pushbutton so that microcontroller can be reset when the power is first applied.

Microcontroller Behaviour

When your microcontroller is reset your program will begin to execute. Your program should turn on all the LEDs to help you verify that all the outputs are properly connected.

The first time the button is pressed the microcontroller should display the first digit of your student number. The binary value of each digit is displayed on the 4 LEDs connected to Port 1 bit 7 on pin 19 (MS bit) through Port 1 bit 4 on pin 16 (LS bit) (see below). Each time the button is pressed the microcontroller should display the next digit. If the last digit has been displayed and the button is pressed again the microcontroller should turn on all the LEDs. Further button pushes should not have any effect.

If you have been allowed to work in groups of two, the microcontroller should display both student numbers. The two student numbers must be separated by a display which has all the LEDs on (as with the initial display).

Components

You will be supplied with a microcontroller and crystal. Both components must be handed in to the TA in the lab when you finish your lab (do not put them in the assignment box).

Microcontroller

You will use an ATMEL 89C1051[1] You may want to look at the data sheets for the microcontroller which are available in PDF format from `http://www.atmel.com/atmel/acrobat/0366.pdf`. The first three pages are probably all you'll want to look at.

The microcontroller has two 8-bit parallel I/O ports, called P1 and P3. In the descriptions below the notation P$n.m$ refers to bit m of the 8-bit parallel port n. n can be 1 or 3 (P1 or P3) and m is 0 to 7.

Other Parts

You will also be supplied with an 11.059 MHz crystal that must be connected between pins 4 and 5 (XTAL2 and XTAL1 respectively). The crystal determines the CPU's clock frequency.

[1]Some students may have to use an 89C2051. The two devices differ only in the amount of EPROM available and the on-chip peripherals available. Either device can be used for this project.

You will be supplied with wire and wire cutters/strippers in the lab. Use these to connect the microcontroller to the power and ground outputs, the LEDs and the pushbutton switches. Use reasonably short wires and keep your wiring neat.

Circuit Description

Important: You must connect the switch and LEDs to the correct pins or your microcontroller will not work properly in the TA's test circuit and you will receive a low mark.

You will wire up your interface circuit on a "DigiDesigner" prototyping unit. The unit includes a power supply, a solderless breadboard, 4 LED displays, 4 slide switches and two pushbutton switches mounted in a metal box.

The breadboard (the rectangular white plastic block) is used to connect the various components to your chip. The chip should be plugged in horizontally over the wide horizontal channel running down the middle of the board. Each pin of the chip is connected to the other five sockets in each column.

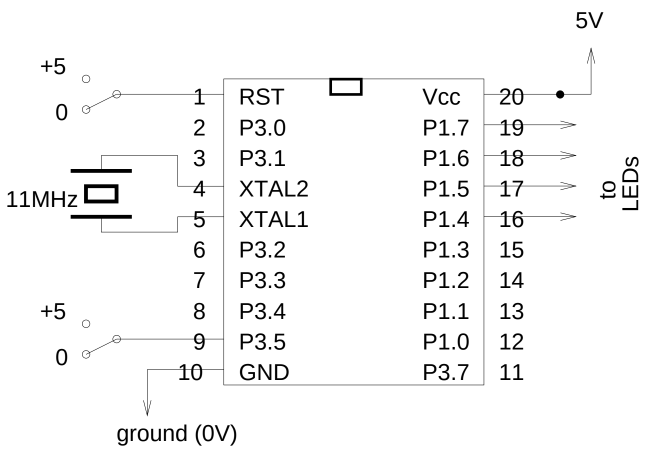
WARNINGS

Turn off the power when setting up or making changes to your circuit. Double check your connections before turning on the power (ask the TA if you are unsure of anything).

Semiconductor devices can be damaged by static electricity. The damage is invisible and can be cumulative. Ideally you would work on a bench with grounding straps and anti-static surfaces. Since this equipment is not available you should take common-sense precautions such as touching a grounded piece of equipment before handling the chip and avoiding touching the pins.

Assembling the Circuit

A schematic of the circuit is given below:



- Plug the chip into the breadboard.
- Connect pin 10 to the ground (0V) power supply output.
- Connect pin 20 to the +5 volt power supply output.
- Connect pins 19 through 16 to the LEDs.
- Connect a pushbutton to pin 9.
- Connect a switch to pin 1 (reset).
- Connect the crystal to Pins 4 and 5.
- Make sure the chip is properly connected to both +5 V and ground rails. Reverse biasing the chip or it's inputs will destroy the chip. Double check your connections before applying power.

Programming the Microcontroller

You should first program the microcontroller with the example code available on the course Web site. This will allow you to check that your hardware operates properly.

You will use the device programmer attached to one of the PCs in the lab. Please use this computer only for programming the devices.

The programmer has ZIF (zero insertion force) socket. Flip the lever to open the contacts, insert the chip into the programmer and flip the lever back to hold the device in the socket. Be careful to follow the chip alignment diagram drawn beside the socket (the bottom of the chip should be aligned with the bottom of the socket).

Type `ACCESS` to run the device programmer software.

The device programmer software is menu-driven. Select the following options:

Device	MPU/MCU
MFR	Atmel
Type	AT89C1051 (or AT89C2051)

This will start a second program that is specific to the 8051 microcontrollers. First select option 2 "Load Hex". Enter the name of your file (e.g. `lab3.hex`). Select the options for Intel hex format and to set unused bytes to FF. Then select option "A" to automatically erase, blank check, program and verify. Select the option to not program any lock bits.

It will take a few seconds for the device to be programmed. You may remove the device when the "Done" LED on the programmer goes on.

Compiling your Code

You will use a free (demo version) of a C cross-compiler for the 8051 that runs under MS-DOS.

The compiler has been installed in the PCs in the lab.

If you want to run the on another PC you can download the file `51demo.exe` from the course Web page. Copy it to any convenient directory on a DOS machine and execute it to unpack the demo compiler. You will need about 2 Megabytes of free disk space.

Type the command `bin\hpd51` to start the interactive editor/compiler environment. There is no command-line version.

Define A Project

You should start by defining a project so that you do not have to re-enter the compiler options each time you start the compiler.

From the main menu select the menu item "New Project" from the "Make" project. You will be presented with a series of dialog boxes:

- Project Name

  Enter a project name, for example `lab3`.

- Processor and Memory Model

  select:

  – Generic 8051
  – Small memory model

- Output File Format

  select:

  – Intel HEX

- ROM and RAM Addresses

  Set all RAM and ROM addresses and sizes to zero (0).

- Compiler Optimizations

  select:

  – Full Optimization (press 'F')
  – Global Optimization level: 1

- Source Files

  Add the name of your C file to the list, for example, `lab3.c`.

- select DONE (press Esc)

Edit/Compile/Link

Enter your source code in the edit window. You may want to start with the sample code. Select the "Make" item from the "Make" menu or press F5. Correct any errors, and re-run the make command as necessary. Make sure the code does not take up more than 1k Bytes and the internal RAM (IRAM) does not take more than 64 bytes.

The object code will be written to a file in "Intel Hex" format. This is one of several formats that are commonly used to transfer data to device programmers.

Copy the resulting hex file (e.g. `lab3.hex`) to a floppy and use it to program the microcontroller as described above.

Submitting The Lab

Hand in the programmed microcontroller and crystal along with a listing of your C code to the TA in the lab when you are done. Do *not* put the components in the assignment box. You must return all the parts to get a mark for the assignment. Your microcontroller will be tested by plugging the programmed chip you submit into a test circuit and checking that the device displays your student number(s) according to the specifications.

You do not need to demonstrate your chip to the TA during the lab, he will test it later and assign a mark to the student number(s) programmed into the chip.