Assignment 8

Due November 19, 1996

Introduction

You will program, assemble and test a microcomputer-based controller for a pedestrian crossing traffic light. The controller must operate according to the specifications given below. You will be supplied with parts that you will use to build the system. 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 controller chip to test your design.

Specifications

Inputs and Outputs

The controller has one input, a switch that the pedestrian uses to request that the lights change. The controller has five outputs that drive five LEDs that simulate the traffic lights: red/yellow/green lights for controlling the traffic and red/green lights for the pedestrian crossing.

Start-Up Behaviour

When your controller is turned on it should go through a diagnostic sequence similar to that given in the example code. This will help you verify that all the inputs and outputs are properly connected and also allows you to 'sign' your chip in case it gets mixed up with other groups' chips.

This diagnostic routine cycles through a set of patterns on the LEDs at the rate of one pattern per second until the switch is pressed. The patterns are: all LEDs on, all LEDs off, followed by a sequence of 2 patterns which depend on your assignment group number. These last two patterns should be the binary representation of the two digits of your group number (this is "binary coded decimal", or BCD). The binary value of each digit is displayed on the 4 LEDs connected to P1.7 (MS bit) through P1.4 (LS bit). See the example code for more details. You *must* change these two patterns to the ones for your group number.

The first time the button is pressed the traffic light control behaviour should start. The initial state of your controller should have the traffic light green and the pedestrian light red.

Traffic Light Behaviour

The controller should behave like the traffic lights used at pedestrian crossings in Vancouver. However, the delays will be shorter to allow the design to be tested more quickly.

If the "crossing request" button has been pushed at any time since the previous time the pedestrian light turned green then the lights go through the following sequence:

- The traffic light turns yellow for 1 second.
- The traffic light turns red and the pedestrian light turns green for 1 second.
- The pedestrian light turns red for 2 seconds.
- The traffic light turns green.
- The traffic light must stay green for at least 5 seconds regardless of whether the button has been pushed or not.

Any button push of 10 ms or more must be detected (you may also capture shorter button pushes).

Components

You will be supplied with all the parts required except for a solderless breadboard and two 1.5 V batteries which you must supply.

Microcontroller

You will use an ATMEL 89C1051[1] microcontroller. The data sheets are available in PDF format from `http://www.atmel.com/atmel/acrobat/doc504.pdf`. The first three pages are probably all you'll need to refer to.

The controller has two 8-bit parallel I/O ports, P1 and P3. Both ports are bidirectional and have internal pull-ups. In the descriptions below the notation Pn.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 the following components:

- an envelope marked with the group number and the names of the group members. It contains the parts described above. Keep the envelope and use it to hand in the components when finished with the assignment.
- 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.
- two 27 pF capacitors that must be connected from each of pins 4 and 5 to ground. These are required for the oscillator to start reliably.
- a 0.1 μF capacitor that must be connected from Vcc to pin 1 (ReSeT). This drives the reset input high for a short time when power is first applied and resets the processor.
- five LEDs (two red, one yellow, two green). These are used to simulate the traffic lights.
- five 150 (or 120) ohm current-limiting resistors to be connected in series with the LEDs.
- wire to hook up two 1.5 volt batteries in series between pin 20 (Vcc) and pin 10 (ground). You must supply the batteries and devise some method to connect them in series (rolling them in a paper tube and using tape and an elastic band to hold it all together seems to work).

  You should also use two short pieces of these wires to make a "switch" that temporarily connects pin 6 (P3.2) to ground. Connect short (3 cm) wires to P3.2 and to ground and leave stripped ends sticking out so they can be shorted together temporarily.

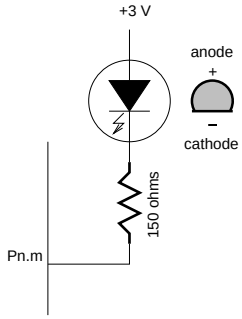
[1] Some groups who were late registering may have to use an 89C2051. The two devices differ only in the amount of EPROM available and the on-chip peripherals available. Either device can be used for this project.

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.

LED Drivers

Read the warning below. Use the following circuit to drive each LEDs:



In order for your circuit to be compatible with the TA's test hardware you must drive the LEDs with the following pins:

Pin	Port	Colour	Purpose
19	P1.7	Red	Traffic Stop
18	P1.6	Yellow	Traffic Warning
17	P1.5	Green	Traffic Go
16	P1.4	Red	Pedestrian Stop
15	P1.3	Green	Pedestrian Go

WARNINGS

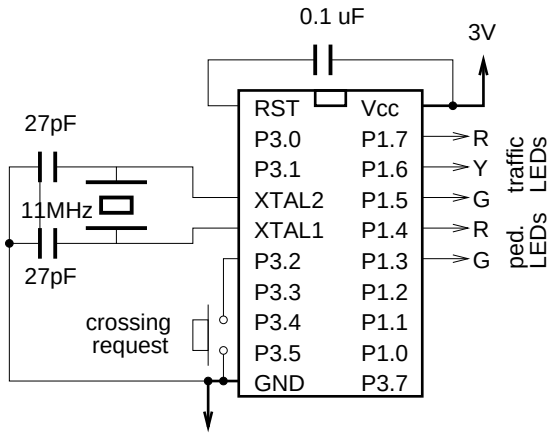
The LEDs must be *forward-biased*: the anode must be more positive than the cathode. Reverse-biasing the LEDs will destroy them immediately. The lead on the flat side of the LED is the cathode (i.e. negative).

The LEDs must be connected *in series with a resistor* to limit the current through the LED.

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:



You'll find it convenient to use a bus strip (the vertical strips) on the right side of the breadboard for Vcc and the one on the left side for ground. Note that the bus strips on many breadboards are disconnected half-way down the board.

- Plug the chip into the breadboard.
- Attach LEDs between the Vcc terminal strip and free terminal rows. Make sure the LED polarity is correct.
- Use the resistors to connect LEDs to the correct output pins on the microcontroller.
- Attach the 0.1 μF capacitor from Pin 20 to Pin1.
- Attach the crystals and 27 pF capacitors to Pins 4 and 5.
- Make a switch using wire.
- Make sure the chip is properly connected to both Vcc 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 the PC at one end of Room 322. Please use this computer only for programming the devices and do not store any files on this computer's hard disk. The lab is open during the day but is often in use for other courses. Please use this lab only for programming your chip and avoid disturbing the other users.

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).

The PC has been configured to start up the device programmer software when it boots up (the program it runs is `C:\PRGRMR\ACCESS.EXE`).

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. `A:ASG8.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.

Copy the file `~elec464/51demo.exe` (about 800 kBytes) to a floppy disk (the file can also be downloaded 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 `asg8`.

- 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, `asg8.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. `asg8.hex`) to a floppy and use it to program the microcontroller as described above.

Submitting Your Assignment

Return the parts in the envelope provided along with a listing of your C code. You must return all the parts to get a mark for the assignment. Your controller will be tested by plugging the programmed chip you submit into a test circuit and checking that the device displays the correct power-on signature and operates according to the specifications.

Bonus Feature

To get a possible 10% extra on this assignment you can adding a blinking feature to your design so that:

1. when the traffic light is green it blinks on and off at about 2 blinks per second (2 Hz), and
2. when the pedestrian light is red and the traffic light is red the pedestrian light blinks on and off at about 2 Hz

Hint: keep a backup copy of your most recent known-good code in case you run out of time.