Lab 5 IC Tester

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

Goals
- The design of an IC tester
- The parallel I/O functions of the 6812,
- The usage of linked list data structures,
- The design of a traffic light controller.

Review
- Chapters 4, 5, 10, 13 of the M68HC812A4 Technical Summary, in particular look up PUCR, RDRIV, PORTH, DDRH, PORTJ, DDRJ, PORTT, DDRT,
- Valvano Section 1.6 about open collector logic,
- Valvano Section 1.7 about initializing and accessing I/O ports,
- Valvano Section 2.4 about abstraction, linked lists and FSM's.

Starter files
- INTERP12.C, FSM12.C, MOORE12.*, MEALY12.*, MOORE2.C, RTI.C

Background

In this lab, the input-output parallel ports of your Adapt812 board will be used in designing an IC tester. You can use any of the 6812 parallel input/output ports on the H1 connector to interface the IC to the computer, but we suggest ports H, J or T because each bit has a direction register. Finally, this lab is a good example of where data structures can be used to simplify programming. You are free to choose which data structure to implement, but should consider a table or linked list. In the table, you will store character information to be used by your interactive interpreter, input/output patterns for each chip to be tested, and ASCII message strings. Some careful design is required in designing the table structure, because all entries in the table are the same size. A table solution may not run the fastest, or occupy the fewest memory bytes, but it is a structured technique that is easy to understand, easy to implement, easy to debug, and easy to upgrade.

Preparation

Your IC tester should test all the TTL combinational IC's in your kit that are not counters or flip-flops and have type numbers beginning with 74 (the 7400 and 7402 for example). You should use a data structure (e.g., table or linked list) to contain the diagnostic information concerning that chip. Explain in your source code how it could be extended to test other chips like the 74LS138 decoder. Your software should initialize the 6812 I/O ports so that some of the lines are outputs and some are inputs. The direction (input or output) will depend on the IC that is to be tested. Your tester should adopt standard connections between the IC pins and the 6812, so that the hardware need not be modified when a different IC type is to be tested. Determine a method that allows you to test all the chips (both 14 and 16 pin packages) without moving wires on your protoboard. Also your interpreter/tester program should be very user-friendly. It should ask for a chip number and return with a "GOOD" or "BAD" message.

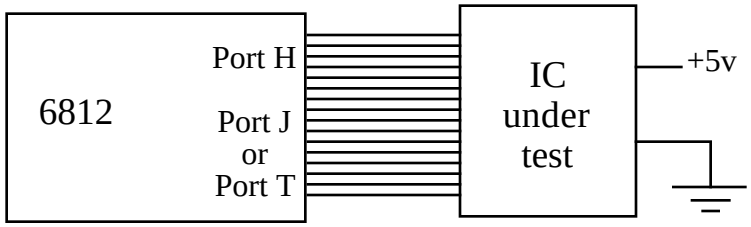

Figure 5.1. IC Tester.

The hardware for this lab includes the wiring diagram between the Adapt812 board and the generic 14/16 pin IC to be tested. A "syntax-error-free" hardcopy listing of the software is required as preparation. This will be checked off by the TA at the beginning of the lab period. You are required to do your editing before lab. The debugging will be done during lab. Document clearly the operation of the routines. Include in your lab prep a wiring diagram for an IC tester as shown in Figure 5.1.

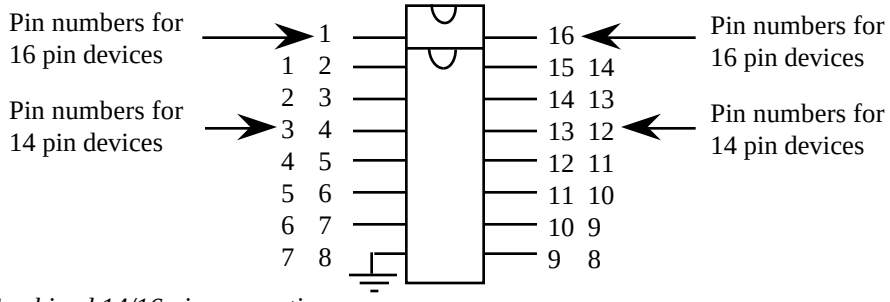

Figure 5.2. Combined 14/16 pin connection.

If you use a single 14/16 pin socket to test the chip, you could position the devices so pin 7 of the 14 pin chips and pin 8 of the 16 pin chips are always grounded.

You will need a way to selectively connect power to pin 14 of the 14 pin chips and to pin 16 of the 16 pin chips. You could use an open-emitter device like the 75491 (or even a 2N2222 transistor) to supply power to pin 14 of the 14 pin devices. In this way the computer can disable the 75491 when testing the 16 pin devices. There are other solutions (some of which are simpler to build) and you are free to implement the problem however you wish, but you are not allowed to move wires on the protoboard to accommodate different devices (you can move wires to simulate bad chips.)

Procedure

1) IC Tester

The procedure for this lab consists of making the required connections between the I/O ports on the project connector and the generic 14/16 pin socket(s). Run your program and identify the good chips and the bad chips. A bad chip can be simulated by removing a connection to the socket. Test the all features of your interpreter including incorrect commands. Draw a truth table in your notebook for a good chip and a bad device. **Obtain a hardcopy printout of the operation of your program**.

2) Understanding the C compiler

Write a C program that includes the following concepts:

- call by value parameter passing to a function;
- call/return by reference parameter passing to/from a function;
- return parameter from a function;
- local variables (allocation, binding, access, deallocation);
- global variables (allocation, access);
- signed and unsigned promotion (8 bit values increased to 16 bits when calculations are performed)
- address calculations to access the table structure,
- creating an interrupt service routine (e.g., RTI.C) and setting its interrupt vector.

Draw a stack picture at some point inside the function illustrating the local variables, and parameters.

Checkout

The IC Tester system must be demonstrated to the TA. The interpreter should check for valid/invalid operator entries. Demonstrate the operation for both good and bad chips. Show to the TA on the computer (no hardcopy required) the assembly code (the YOUR.LST file) for the short C program, and be prepared to explain the above concepts at the assembly language level.

Hints

1) Make sure you do not connect a 6812 output line to an output line of the chip. You may wish to use the internal resistor pull-up when configuring inputs (see PUCR). Don't use Port S, which implements the serial port or Port E on the H1 connector. There is no particular reason we are avoiding Ports A, B, C, and D on the H2 connector, except we wish to minimize the number of times you plug and unplug the Adapt812 board.

2) There is no single "best" way to implement your IC tester. You must use a data structure.

3) Read "Developing C Programs using ICC11/ICC12/Hiware" on the TExaS CD or on the web at
http://www.ece.utexas.edu/~valvano/embed/toc1.htm

4) Download the file "RTI.C" from the class web site, compile it, download it, and run it. This is a simple example of an interrupting program.

5. Even though the TExaS application does not simulate individual digital ICs, you can connect the input/outputs to switches and LEDs and test the software functions. In particular, the INTERP12.UC, INTERP12.IO files can be run on the TExaS simulator. The files MOORE12.UC, MOORE12.IO, and MOORE12.SCP files can also be simulated.