Introduction to the Handy Board

Pre-Lab Questions

1. How can you tell that IC is running properly on the Handy Board (HB)?
2. How do you put the HB into bootstrap download mode? What indicates that it is ready to receive the P-code?
3. What IC command halts all currently running processes?
4. What is the significance of the main() function in IC? When is it run? How do you bypass it?
5. What's a list file (example: gnu.lis), and what is its use?
6. What are the 3 different methods of charging the batteries of the HB? Which mode can cause damage to the HB?

Purpose

- To become familiar with the features of the Handy Board and the Expansion Board
- To become familiar with Interactive C (IC)
- To practice programming in C

Components

Qty.	Item
1	Handy Board with expanison board and serial interface/charger
1	Serial cable with DB-9 to DB-25 adapter
1	RJ-11 cable

Introduction

In this lab you will investigate the Handy Board and get familiar with its features. You will program the Handy Board with Interactive C (IC), and practice programming.

As excerpted from the IC Manual (http://handyboard.com/software/icmanual/icmain.html), IC is a C language consisting of a compiler (with interactive command-line compilation and debugging) and a run-time machine language module. IC implements a subset of C including control structures (for, while, if, else), local and global variables, arrays, pointers, 16-bit and 32-bit integers, and 32-bit floating-point numbers. IC works by compiling into pseudo-code for a custom stack machine, rather than compiling directly into native code for a particular processor. The run-time machine language program then interprets this pseudo-code (or p-code). This unusual approach to compiler design allows IC to offer the following design tradeoffs:

- *Interpreted execution that allows run-time error checking and prevents crashing*. For example, IC does array bounds checking at run-time to protect against programming errors.
- *Ease of design*. Writing a compiler for a stack machine is significantly easier than writing one for a typical processor. Since IC 's p-code is machine-independent, porting

IC to another processor entails rewriting the p-code interpreter, rather than changing the compiler.

- *Small object code*. Stack machine code tends to be smaller than a native code representation.
- *Multi-tasking*. Because the pseudo-code is fully stack-based, a process's state is defined solely by its stack and its program counter. It is thus easy to task-switch simply by loading a new stack pointer and program counter. This task-switching is handled by the run-time module, not by the compiler.

Since IC 's ultimate performance is limited by the fact that its output p-code is interpreted, these advantages are taken at the expense of raw execution speed. Still, IC is no slouch.

Handy Board Hardware

Figures 1 and 2 depict the Handy Board (HB) and Expansion Board (EB) hardware and their major features. The HB has an integrated, rechargeable battery pack located in the compartment beneath the main board, which enables it to operate for a period of time without externally supplied power. A recharger/interface (RI) board provides a connection to the host computer so that programs can be downloaded to HB and its batteries can be charged.

Procedure

Establishing the proper connections

1. Plug the AC adapater into an outlet on the bench. Connect the other end into the recepticle on the RI board. The red “PWR” LED should be lit. (The Handy Board carries a set of rechargeable batteries, so it can operate for a period without the AC adapter, but we will use the AC adapter to keep the batteries charged.)
2. Connect the serial cable of the host computer to the DB-25 connector on the recharger/interface board. The “SER” LED should then be lit.
3. Make sure that the charge mode switch on the RI board is set to “Normal”, then connect the RJ-11 cable between the RI board and the HB. The yellow “CHARGE” LED should be lit.
4. Verify that that the P-code has been downloaded to the HB. (How do you verify that the P-code is loaded?) If not, put the HB in bootstrap download mode, start IC, and download the p-code.

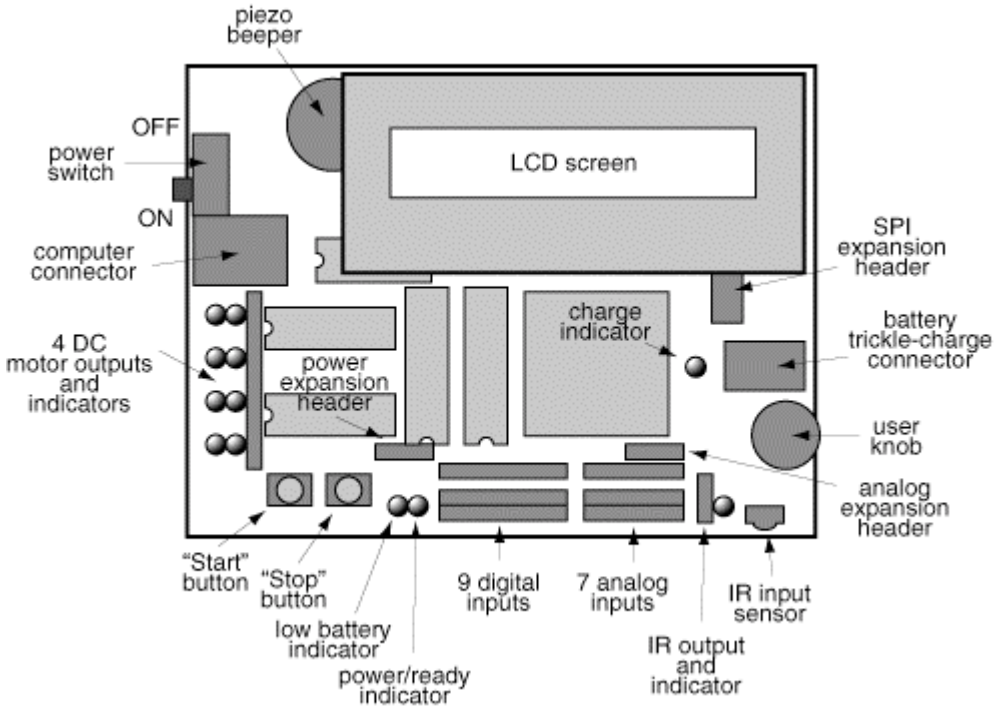

Figure 1 Handy Board Diagram. The Handy Board is based on the 52-pin Motorola MC68HC11 processor, and includes 32K of battery-backed static RAM, four outputs for DC motors, a connector system that allows active sensors to be individually plugged into the board, an LCD screen, and an integrated, rechargable battery pack. (ref.: http://handyboard.com/hardware/index.html and http://handyboard.com/hblabel.html)

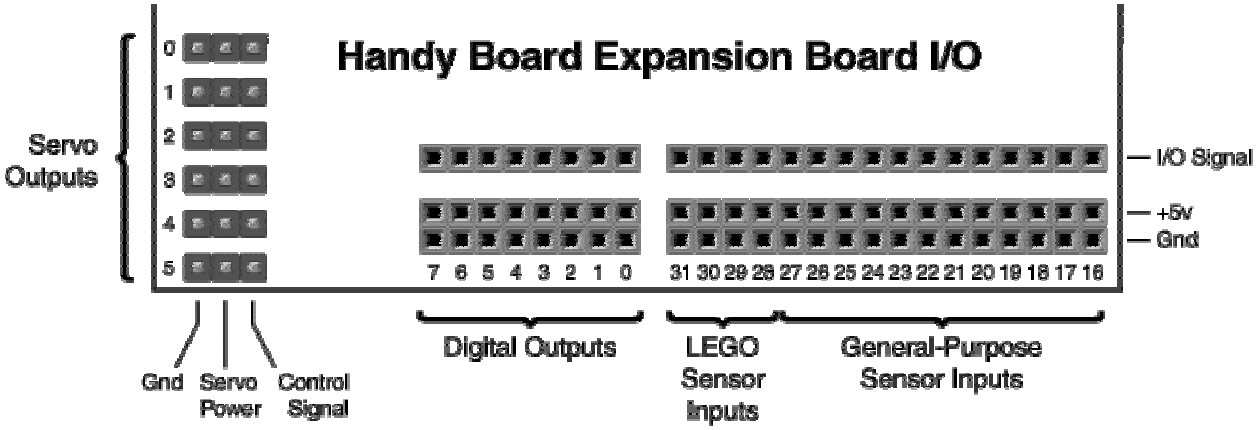

Figure 2 Expansion Board Diagram. The expansion board plugs on top of the Handy Board and provides additional I/O pins as well as R/C servo outputs. (ref.: http://handyboard.com/hbexp30/)

Start IC

5. Make sure that the RI board is connected to the serial cable of the host computer. Start IC, and you should see:

```
Synchronizing with board
Pcode version 3.10 present on board
Loading C:\IC\libs\lib_hb.lis.
Loading C:\IC\libs\lib_hb.c.
Loading C:\IC\libs\r22_ir.lis.
Loading C:\IC\libs\r22_ir.icb.
Loading C:\IC\libs\r22_ir.c.
Initializing interrupts
Downloading 1724 bytes (addresses 8000-86BB): 1724 loaded
Downloading 78 bytes (addresses 86BC-8709): 78 loaded
Downloading 16 bytes (addresses 870A-8719): 16 loaded
Code loaded.
```

Exploration of IC

6. Type beep(); to the IC prompt.
   a. You should hear a 500 Hz tone that lasts for 0.3 seconds. Beep() is a function whose definition can be found in c:\IC\libs\lib_hb.c
7. Try some arithmetic statements:
   a. Ex: 2+3;
8. Try a print statement: printf("Hello, World!\n");
   a. What happens if you do not include the \n in the printf statement above? Try it by entering the statement above without the \n twice. (Note: IC has a built-in command line editor and command history. You can retrieve a line you previously typed by using the ↑ (up-arrow) or CTRL-P keystroke. See the IC Manual for more information on the command line editor).
9. Try some some multiple statements. Multiple statements are enclosed by braces, { }.
   b. {beep(); sleep(1.0); beep();}
   c. What would happen if you entered, {beep(); sleep(1); beep();} instead? What data type does the function need for its argument?
10. Write a one-line program to print out the value of the user knob. The function knob() returns an integer between 0 and 255 depending on the angle of the potentiometer (user knob) near the IR sensor at the corner of the Handy Board. Adjust the knob in several settings and invoke your program to see the output change. Record your program, and include it in your report.
11. Modify your program above to print out the value of the user knob *continuously*. Vary the knob and verify that your program functions properly. Record your program, and include it in your report.
12. Modify your program above to also turn on the red LED associated with MOTOR 0 when the value of knob is between the values of 127 and 200. If the value of knob is not in this interval, then the LED should be off. Record your program, and include it in your report.

By now it should be obvious that it would be very tedious to write programs of any significant length by using the command line editor. IC allows C programs to written using an external editor and downloaded to run on the Handy Board.

13. In a text editor, create a function called **knob_test()**. The first line of your program should look like:

```
Void knob_test()
  {
     put your program from step 12 here.
  }
```

    Give your file a name like, knob_test.c and save it in the USERCODE directory in the IC directory. (Note: if you use anything more than just a text editor, save it as text only.)

14. Switch back to IC and download the file knob_test.c to the Handy Board by typing the following:

    load c:\IC\usercode\knob_test.c

    You should see something like the following:

```
Loading c:\IC\usercode\knob_test.c.
Initializing interrupts
Downloading 1829 bytes (addresses 8000-8724): 1829 loaded
Downloading 78 bytes (addresses 8725-8772): 78 loaded
Downloading 16 bytes (addresses 8773-8782): 16 loaded
Code loaded.
#done
```

15. Run the function knob_test(), and verify that your program functions as before. Note that once files are loaded into IC, they stay loaded until they are explicitly unloaded (or until you re-invoke IC and it reloads the hb_lib.lis and r22_lib.lis files). Verify this by aborting from the knob_test() function (just press the ENTER key while in IC), switching the Handy Board off, switching it back on again, and running knob_test(). You can also see that knob_test() is still loaded by typing, **list functions** in IC. You should spot knob_test() at the bottom of the list.

16. Unload knob_test.c by typing: **unload c:\IC\usercode\knob_test.c** Verify that knob_test() is no longer in the list of functions.

Multiple files can be downloaded together using list files. This is especially convenient in making use of programs and functions from others. Instead of copying and pasting code from other C programs, you can simply include these files in a list file, and they will be loaded when you load the list file.

You will make a list file using the function that you wrote above and a pre-written C file in the IC libs directory called “music.c”. The file music.c defines some functions by which you can play notes on the Handy Board’s piezo speaker.

17. Create a new C file in the USERCODE directory called **knob_toon.c**. Type and save the following program:

```
/* Knob and Tune Program */
/* Put your name here  Put the date here */
  char mhall_song[]="3e3d3c3d 3e3e6e 3d3d6d 3e3g6g 3e3d3c3d 3e3e3e3e 3d3d3e3d
9c";
```

```
void mhall() {
   play(mhall_song);
}

void main()
{
   int pid1;
   pid1=start_process(knob_test());
   while(1)
   {
   if(start_button())
      {mhall();}
   if(stop_button())
      {kill_process(pid1);
       alloff();
       printf("That's all folks! Bye!\n");
       break;
      }
   }
}
```

18. Create a new C file in the USERCODE directory called **knob_toon.lis**. Type and save the following lines:

```
music.c
c:\IC\usercode\knob_test.c
c:\IC\usercode\knob_toon.c
```

19. In IC execute the following command:

load c:\IC\usercode\knob_toon.lis

You should see something like:

```
IC> load c:\IC\usercode\knob_toon.lis
Loading c:\IC\usercode\knob_toon.lis.
Loading c:\IC\usercode\knob_test.c.
Loading C:\IC\libs\music.c.
Loading c:\IC\usercode\knob_toon.c.
Initializing interrupts
Downloading 3090 bytes (addresses 8000-8C11): 3090 loaded
Downloading 2176 bytes (addresses 8C12-9491): 2176 loaded
Downloading 16 bytes (addresses 9492-94A1): 16 loaded
Code loaded.
#done
```

20. What happens now if you switch the Handy Board off, then switch it on? Rotate the user knob. What is displayed on the LCD screen? Push the Start button on the Handy Board. What happens? Does the number printed to the screen update even when the song plays? What does the statement, "pid1=start_process(knob_test());" do? What difference does including main() in the program make? Switch the Handy Board off, then ***while*** holding down the Start button, switch the Handy Board on. What happens (or does not happen)?

Appendices

Schematic Diagrams

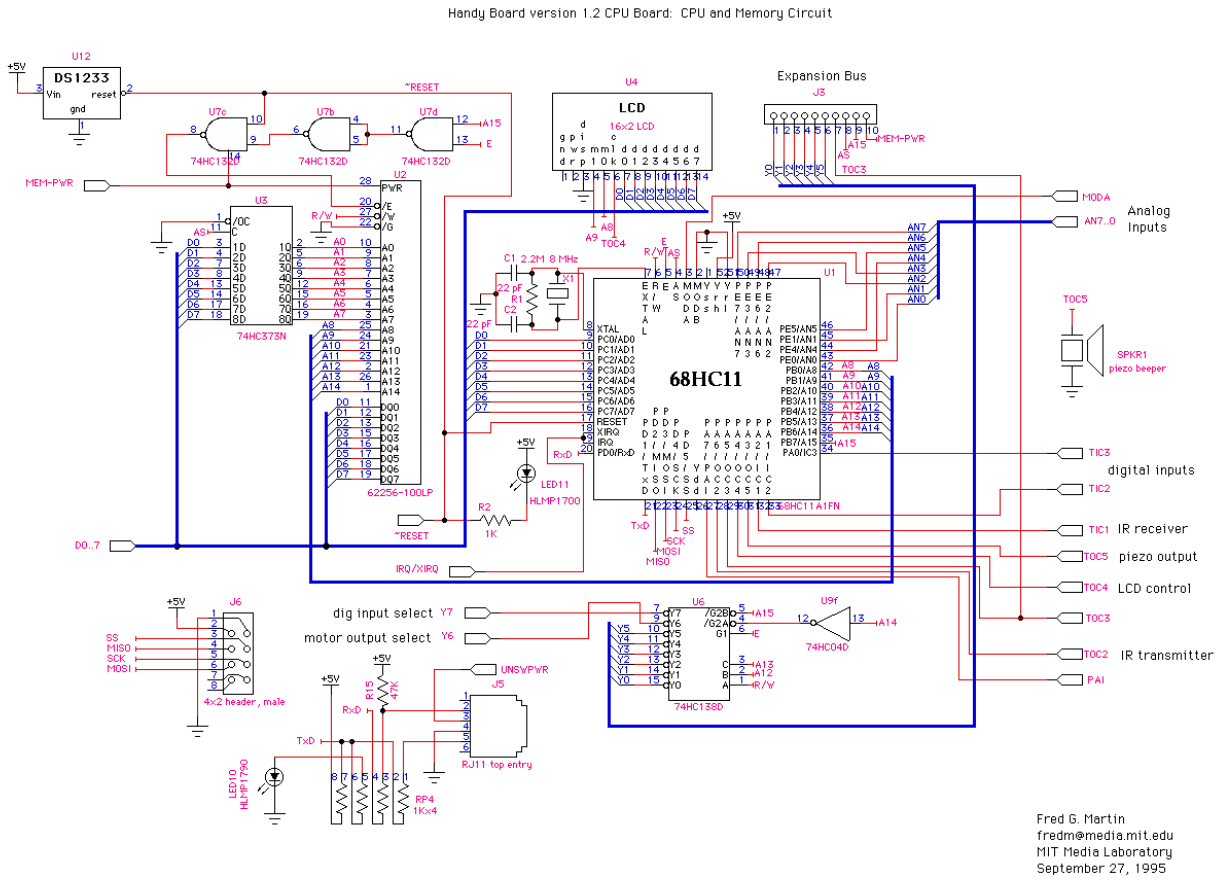

Figure A1 Handy Board CPU and Memory Circuit (ref.: http://handyboard.com/schemv12/cpu.gif)

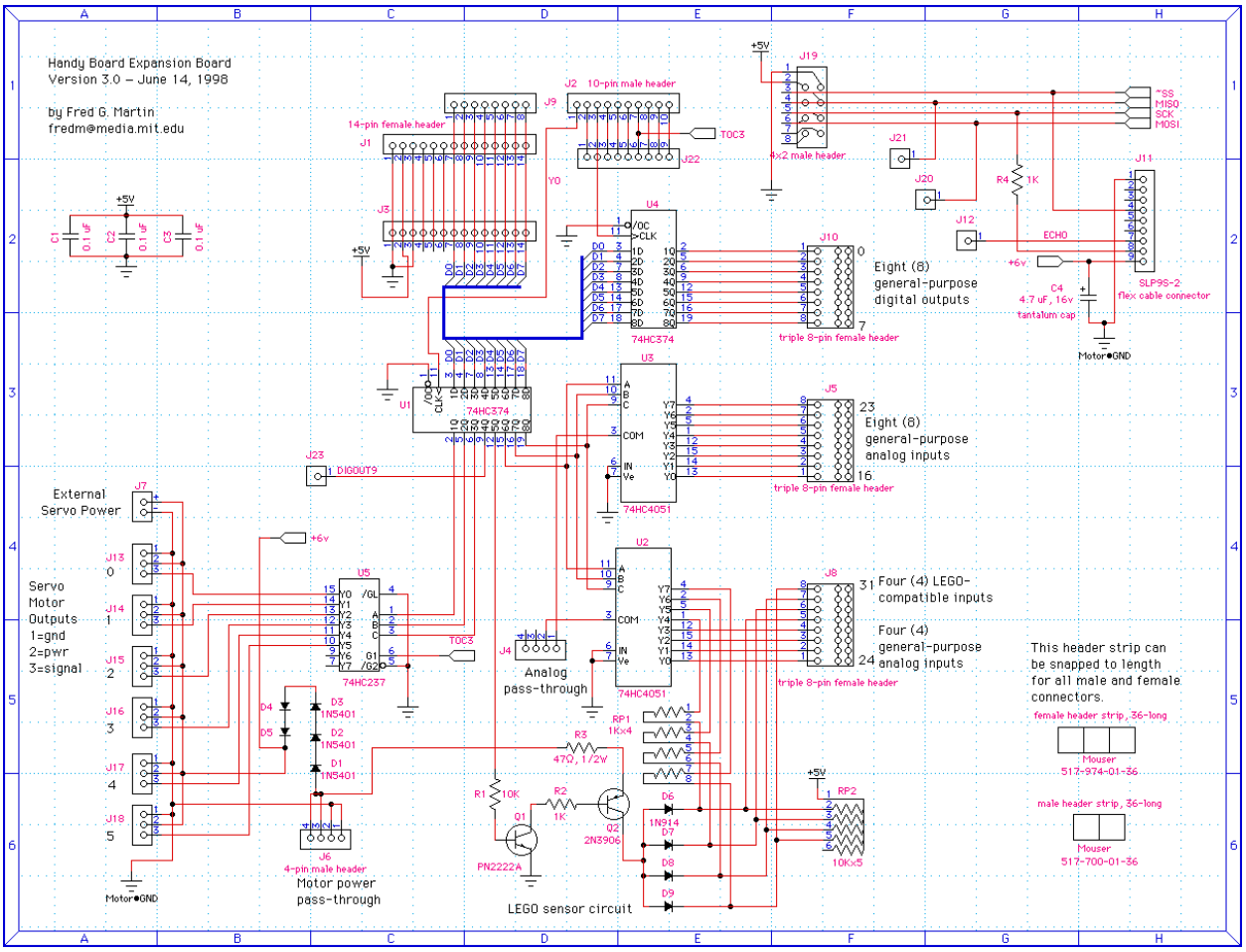

Figure A2 Expansion Board Schematic (ref.: http://handyboard.com/hbexp30/expsch30.gif)