16.070 Introduction to Computers and Programming

March 8 **Recitation 5** **Spring 2001**

Topics:

Handy Board

- Parts List
- Software
- Ports & Connectors
- Connecting to the PC
- Initializing the Handy Board
- Explanation of LEDs

Interactive C

- Summary
- Using IC
- Differences Between C & IC
- LCD Screen Printing
- Library Functions

Handy Board

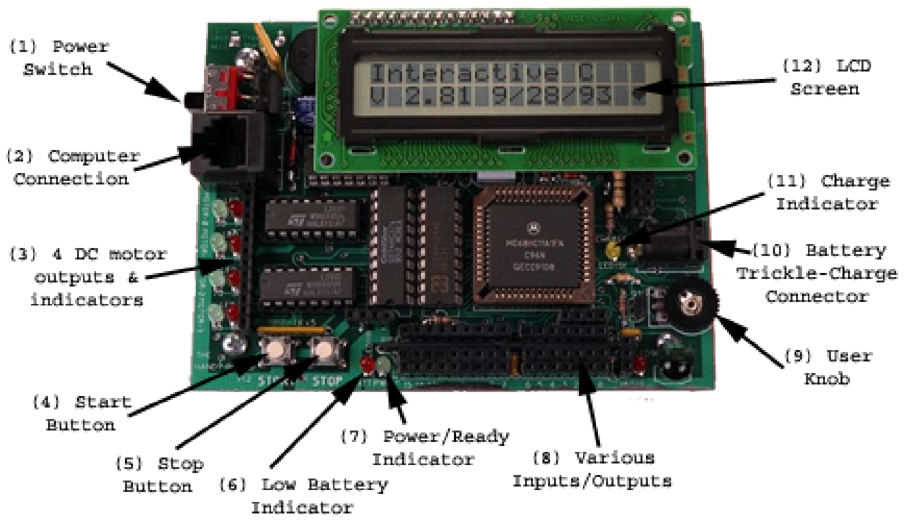
Parts List

Open the Handy Board (HB) box and make sure that all of the following parts are included:

- 1 HB w/ LCD screen
- 1 serial interface board
- 1 PC/HB serial cable
- 1 telephone cable
- 1 power adapter
- 1 HB technical reference
- 1 HB software CD

** If there are any parts missing let us know immediately.

Ports & Connectors



(1) Power Switch
(2) Computer Connection
(3) 4 DC motor outputs & indicators
(4) Start Button
(5) Stop Button
(6) Low Battery Indicator
(7) Power/Ready Indicator
(8) Various Inputs/Outputs
(9) User Knob
(10) Battery Trickle-Charge Connector
(11) Charge Indicator
(12) LCD Screen
Interactive C
V 2.81 9/28/93

Connecting to a PC

1) Plug one end of the PC/HB serial cable into the serial port on your PC
2) Plug the other end of the PC/HB serial cable into the serial interface board
3) Plug in the power adapter
4) Connect the power adapter to the power connector on the serial interface board
5) Plug one end of the phone cable into the RJ-11 jack on the serial interface board
6) Plug the other end of the phone cable into the RJ-11 jack on the HB

Software

The primary software that we will be using to control the HB is Interactive C. MIT has a site license for the commercial version of Interactive C from Newton Labs. We will be sending out an email that contains the license information that you need to run the commercial version. There are three pieces of software that you will need to use IC with the HB. All of the software can be found on the 16.070 web site on the Handy Board – Software page. All you should need the file icw32.exe. This is the IC application for Windows.

Download the IC operating program to the HB. This process should only need to be done once. However, if the HB loses the IC program stored in its memory for some reason, just follow these steps again.

Install the IC application on your computer.

1) Run the icw32.exe program that will install the IC application
2) Enter MIT's site license info. and finish installing the program. Make sure that the above installations were successful.
3) Run IC (Interactive C) from Start-->Programs-->Interactive C while the HB is switched off. IC will indicate that the board is not responding and prompt for board configuration. Click Yes.
4) Click "Download P-code" and select the file "Handy_Board_1.2.icd". Click on "Open". Switch on the HB while simultaneously holding down its stop button. You should see an entire top row of squares displayed on the HB's Liquid Crystal Display (LCD). This activates the low level programming mode of the HB. Now click "OK" and wait a few seconds until IC prompts you again.
5) At the prompt: First reset the HB (power off/on) and then press "OK". You should now see how IC is loading the HB user libraries onto the HB.
6) type *printf("Hello World")* in the bottom part of the IC window and press enter
7) You should now see *Hello World* on the LCD of the HB

Explanation of LED's

	Color	**ON**
PC/HB interface board	red	power is on
	yellow	HB battery is being charged
	green	connected to computer and serial port driver is open
HB	red	HB battery is low
	yellow	HB battery is charging, when power cord is plugged into HB
	green	HB is on, blinks during serial transmission

Interactive C

Summary

Interactive C 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, structures, 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. This pseudo-code (or p-code) is then interpreted by the run-time machine language program. This unusual approach to compiler design allows IC to offer the following design tradeoffs:

- **Interpreted execution** that allows run-time error checking. For example, IC does array bounds checking at run-time to protect against some 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.

Using IC

When the IC program is started it tries to connect to the HB, which should be turned on. After synchronizing with the HB, IC compiles and downloads the default set of library files. At this point either an IC command or a C-language expression may be entered in the bottom of the window.

All C expressions must be ended with a semicolon. To evaluate a series of expressions, create a C block by beginning and ending the series of expressions with {}.

IC Commands

IC responds to the following commands:

load *filename*
: compiles and loads the named file. The board must be attached for this to work. IC looks first in the local directory and then in the IC library path for files. Several files may be loaded into IC at once, allowing programs to be defined in multiple files.

unload *filename*
: unloads the named file, and re-downloads remaining files.

list *files*
: displays the names of all files presently loaded into IC.

list *globals*
: displays the names of all currently defined global variables.

list *functions*
: displays the names of presently defined C functions.

list *defines*
: displays the names and values of all currently defined preprocessor macros.

kill_all
: kills all currently running processes.

ps
: prints the status of currently running processes.

help
: displays a help screen of IC commands.

quit
: quit exits IC. ^c can also be used.

In IC you can edit and re-use previously typed commands by using the up and down directional keys to select the appropriate command.

After functions have been downloaded to the board, they can be invoked from the IC prompt. If one of the functions is named *main()*, it will automatically be run when the board is reset. To reset the board without running the *main()* function (for instance, when hooking the board back to the computer), hold down the board's start button while turning the board on.

Differences Between C & IC

- IC does not support the *double* data type
- IC does not support the *case* and *switch* statements
- IC is *strongly typed*
- IC does not support the **const** statement, only **#define**
- IC has basic I/O built in and does not use **#include**

Typecasting – in brief

Typecasting will convert variables of one type to another.

```
long VarLong = 45L;
int VarInt = 5;
```

If you wanted to set VarInt equal to five times the value of VarLong, you would have to typecast VarLong to be an integer. Typecasting is accomplished by placing the desired type in parenthesis immediately to the left of the expression being typed. For instance:

```
VarInt = 5 * (int) VarLong;
```

You'll find typecasting will work between floats, longs, ints, chars, and many other types. However, converting a char to an int (or vice versa) is something we'll address in more detail next week. For your homework this week, you may need to typecast numeric types.

LCD Screen Printing

IC has a version of the C function *printf* for printing to the LCD screen.

```
printf(format-string, [arg-1], …, [arg-N]);
```

The format-strings are located in the following table:

Format Command	Data Type	Description
%d	int	decimal number
%x	int	hexadecimal number
%b	int	binary number
%c	int	ASCII character
%f	float	floating point number
%s	char array	string

notes on printing to the LCD:

- The final character position of the LCD is used as a system "heartbeat." This character continuously blinks back and forth when the board is operating properly. If the character stops blinking, the HB has crashed.
- Characters that would be printed beyond the final character position are truncated
- The *printf()* command treats the two-line LCD screen as a single longer line

Library Functions

User Buttons & Knob (p. 22)
The HB has two buttons and a knob whose values can be read and used by programs.

int stop_button(void);
    Returns value of stop button: 1 if pressed and 0 if released.

int start_button(void);
    Returns value of start button: 1 if pressed and 0 if released.

void stop_press(void);
    Waits for stop button to be pressed, then released. Then issues a short beep and returns.

void start_press(void);
    Like stop_press but for the start button.

int knob(void);
    Returns the position of the knob as a value from 0 to 255.

Time Commands (p. 23)
System code keeps track of time passage in milliseconds. The time variables are implemented using the long integer data type. Standard functions allow the use of floating point variables when using the timing functions.

void reset_system_time(void);
    Resets the count of system time to zero milliseconds.

long mseconds();
    Returns the count of system time in milliseconds. mseconds() is implemented as a C primitive not a library function.

float seconds(void);
    Returns the count of system time in seconds, as a floating point number. Resolution is one millisecond.

void sleep(float sec);
    Waits for an amount of time equal to or slightly greater than sec seconds.

void msleep(long msec);
    Waits for an amount of time equal to or greater than msec milliseconds.

Tone Functions (p. 24)
Several commands are provided for producing tones on the standard beeper.

void beep(void);
    Produces a tone of 500 Hz for a period of 0.3 seconds.

void tone(float frequency, float length);
    Produces a tone at pitch frequency Hz for length seconds. Both frequency and length are floats.

void set_beeper_pitch(float frequency);
    Sets the beeper tone to be frequency Hz. The subsequent function is then used to turn the beeper on.

Handy Board IC Index

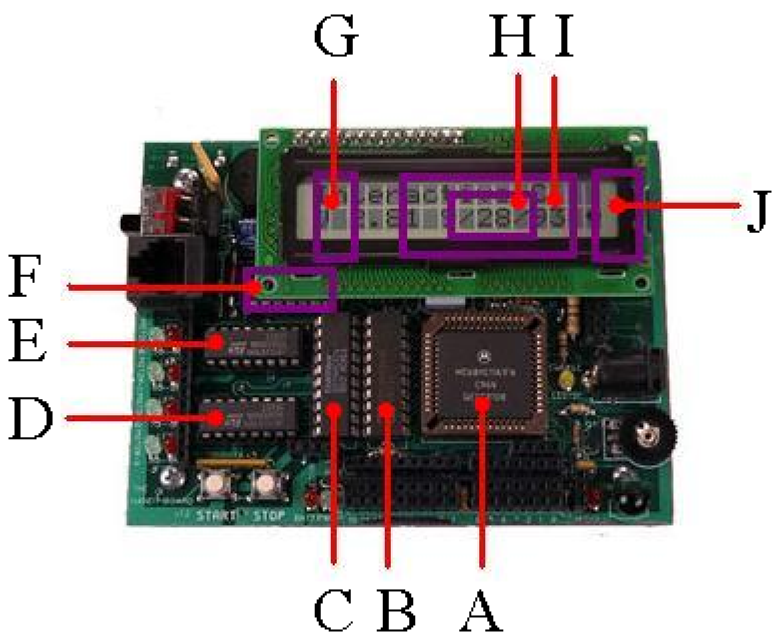


Figure 1 Handy Board

68HC11A Micro-Processor: Motorola micro-processor with address bus, data bus, 8-bit A/D converter. One A/D channel used for thumb wheel position conversion, IR Tx circuit connected directly to chip.
74HC244 Tri-state buffer: External digital inputs are buffered by this IC, before provided to processor. Tri-state allows for disconnection from processor data bus when no digital input received.
74HC374 Octal Latch: 8 D-type latches that store the motor controller settings provided by the processor, so that the processor data bus may be utilized for other tasks. (i.e. just keeps motor controllers doing what they are doing until the processor issues a different setting)
L293D Motor controller (or equiv.): PWM motor controllers capable of output voltage swing for forward/backward rotation motor control. Two LEDs connected to outputs from chip to each motor are connected to the PWM backw./forw. outputs, indicating direction of rotation. Two motor controllers per L293D IC.
L293D Motor controller (or equiv.): (See D.)
74HC04 Hex Inverter: Inverts six digital input bits. Four bits used between motor controllers and D-latch (See C.), to satisfy controller requirements. One bit used to address expansion bus J3. One bit used by IR Tx circuit.
74HC132 Quad NAND gates: Four NAND gates. Three used for reset/start-up sequence generation for memory (See I.). One gate used by IR Tx circuit.
74HC373 Octal Latch (transp.): Acts as address bus expander for processor. Expands 8-Bit address bus to 14-Bit by latching partial addresses provided by processor's data bus.
62256-100/120LP 32K Static CMOS RAM: 32K static CMOS system RAM.
74HC138 3 to 8 Decoder: Provides device address decoding for a maximum eight devices on the expansion bus J3.