6808 Training Kit
Lab Book

Date: November 22, 2001

Document Revision: 1.02

BiPOM Electronics

11246 South Post Oak #205, Houston, Texas 77035
Telephone: (713) 661- 4214 Fax: (713) 661- 4201
E-mail: info@bipom.com
Web: www.bipom.com

WARRANTY:

BiPOM Electronics warrants 6808 Training Kit for a period of 1 year. If the Kit becomes defective during this period, BiPOM Electronics will at its option, replace or repair the Kit. This warranty is voided if the product is subjected to physical abuse or operated outside stated electrical limits. BiPOM Electronics will not be responsible for damage to any external devices connected to the Kit. BiPOM Electronics disclaims all warranties express or implied warranties of merchantability and fitness for a particular purpose. In no event shall BiPOM Electronics be liable for any indirect, special, incidental or consequential damages in connection with or arising from the use of this product. BiPOM's liability is limited to the purchase price of this product.

TABLE OF CONTENTS

Introduction

The purpose of the 6808 Training Lab is to familiarize the student with developing practical applications using the Motorola 68HC08 series of micro-controllers. The 68HC08 micro-controller is one of the most popular micro-controllers in use today with applications ranging from industrial, medical, home automation to automotive.

The 6808 Training Kit consists of the following components:

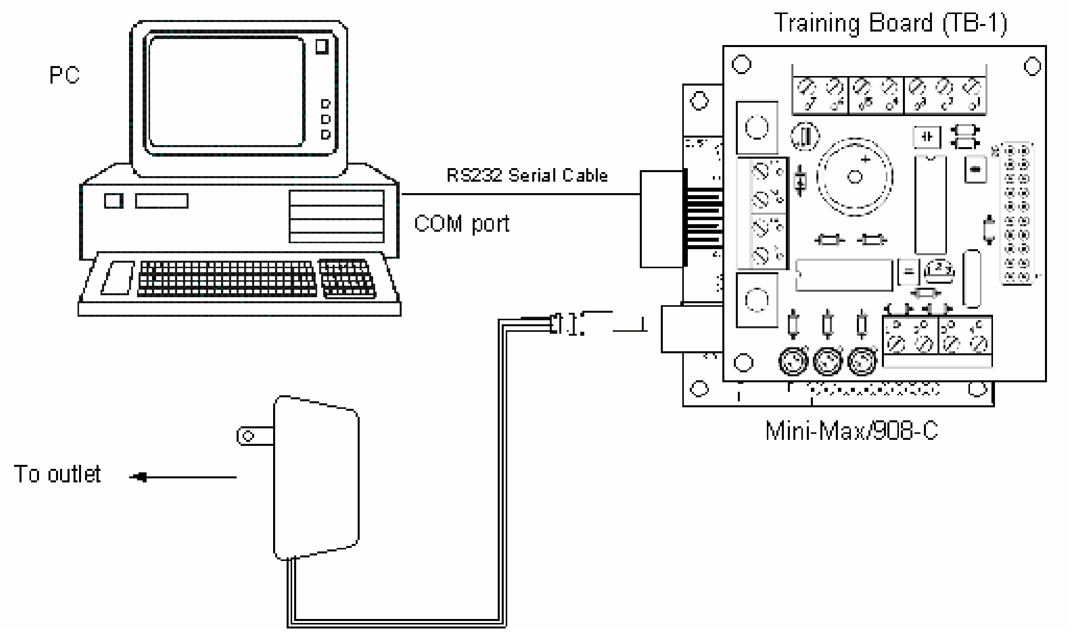
- MINI-MAX/908-C Micro-controller Board
- Training Board (TB-1)
- Micro C Compiler/Linker/Assembler
- Micro-IDE Integrated Development Environment
- Serial cable
- Power Supply
- User's Guide and this Lab book

The following external items are required for each training kit station:

- IBM Compatible Personal Computer (PC) running Windows 95/98 or NT 4.0.
  (6808 Training Kit will not work with DOS, Windows 3.1 or lower).
  Minimum 16MB memory and 10 MB of available hard disk space.

- One available RS232 Serial port (COM1 through COM4).

Figure 1 shows all the components connected together.



Layout of the 6808 Training Board (TB-1) is shown in Figure 2.

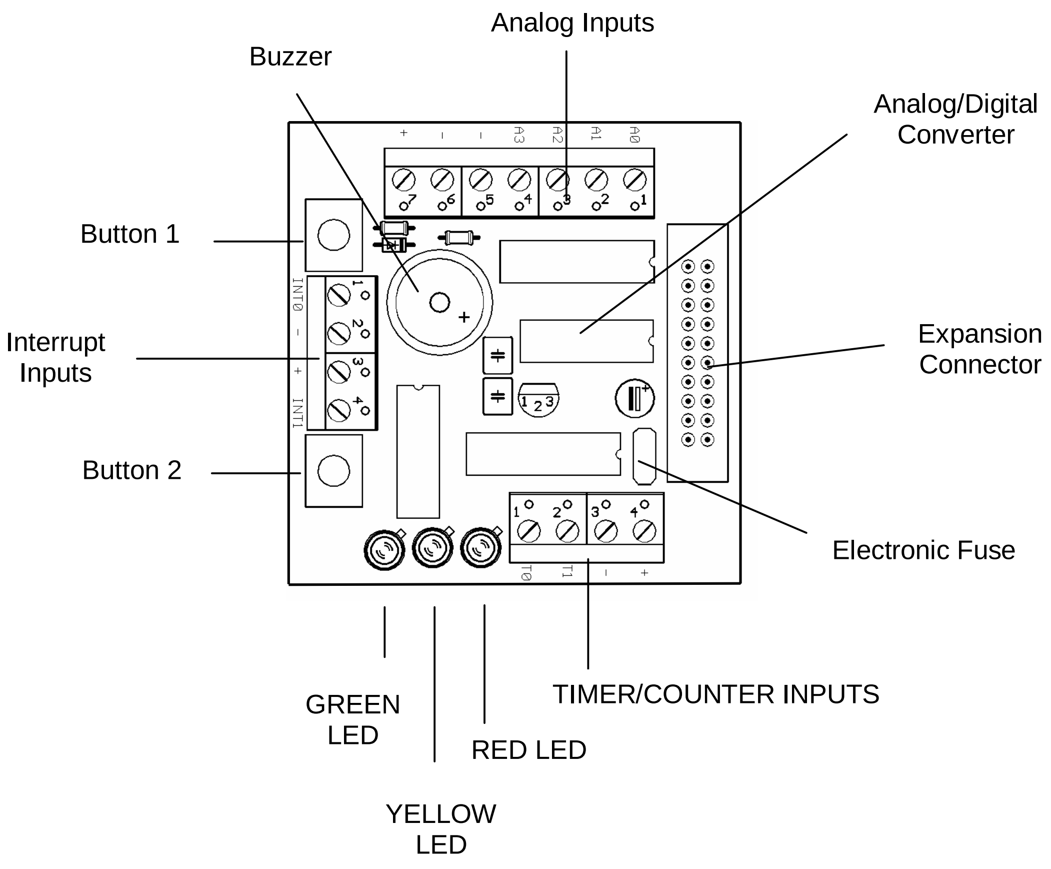


Figure 2

Getting Started

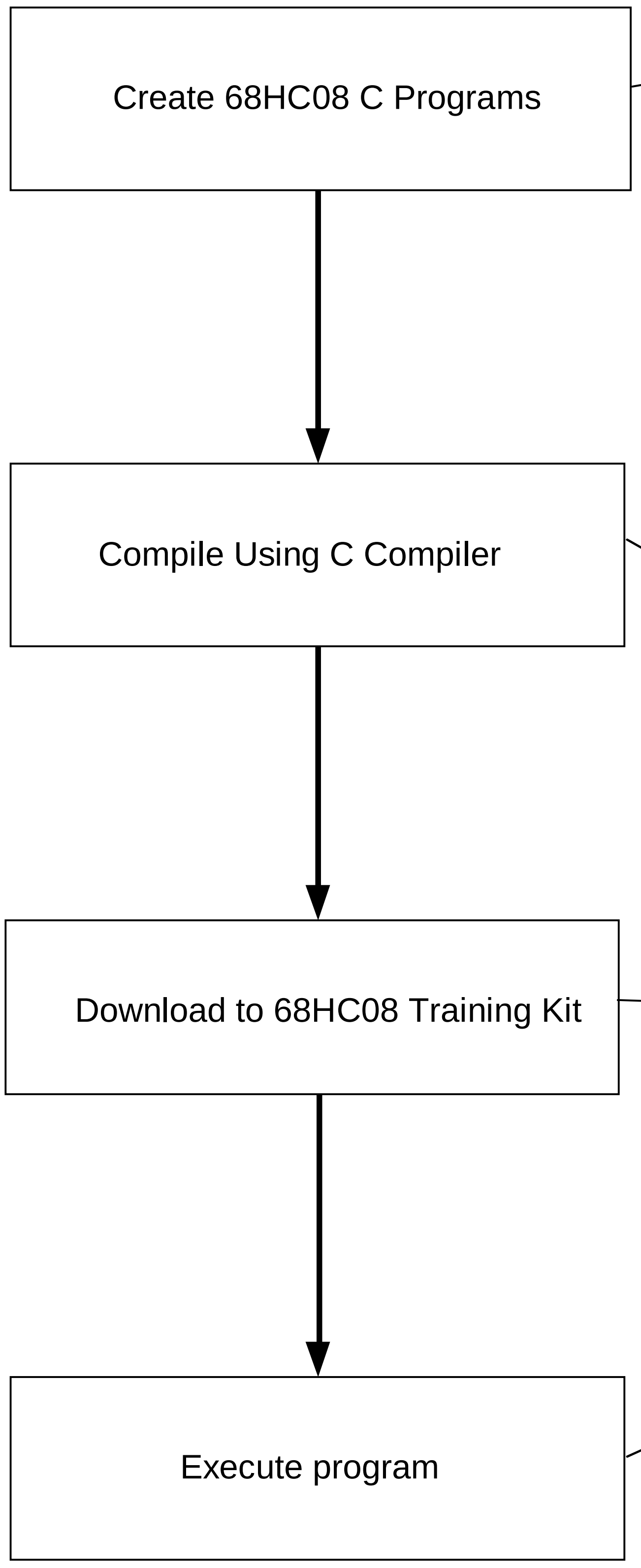
Log on to Windows before using the 68HC08 Training Kit. Enter the user name and password that your instructor has given to you before the Lab. Make sure to log out when you are done with the PC at the end of the Lab. Do not share your username or password with anybody.

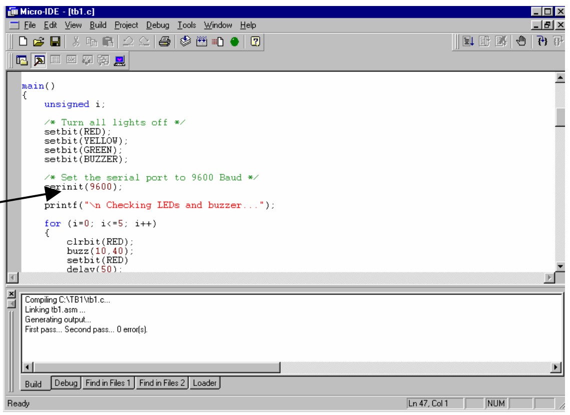
All programs are written as C Language Program files, which have the extension **.C** but they are plain text files. These C Program files are created using Micro-IDE Integrated Development Environment for Windows.

You can edit and save programs and download to the Training Board using Micro-IDE.
Creating programs and running them on the Training Board consists of the following steps:

1. Edit an existing or create a new program using Micro-IDE Program Editor.
2. Compile the program using Micro-IDE Build Toolkit (*C Compiler*)
3. Download the program to the Training Board using Micro-IDE Mini-Max/908-C Loader.
4. Run and debug the program on the Training Board using Micro-IDE Terminal Window.

Figure 3 illustrates these steps.

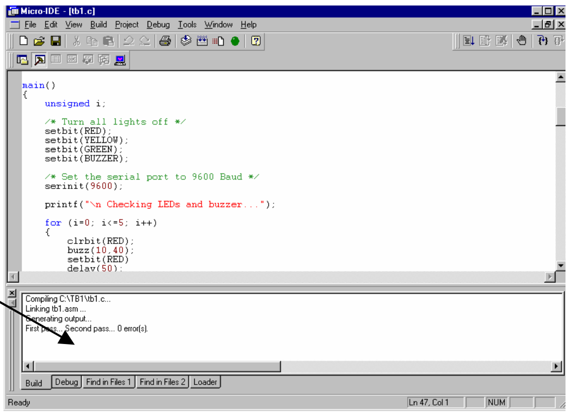




Micro-IDE - [tb1.c]
File Edit View Build Project Debug Tools Window Help
Build Debug Find in Files 1 Find in Files 2 Loader
Ready
Ln 47, Col 1
NUM

```
main()
{
    unsigned i;

    /* Turn all lights off */
    setbit(RED);
    setbit(YELLOW);
    setbit(GREEN);
    setbit(BUZZER);

    /* Set the serial port to 9600 Baud */
    serinit(9600);

    printf("\n Checking LEDs and buzzer...");

    for (i=0; i<=5; i++)
    {
        clrbit(RED);
        buzz(10,40);
        setbit(RED)
        delay(50);
```

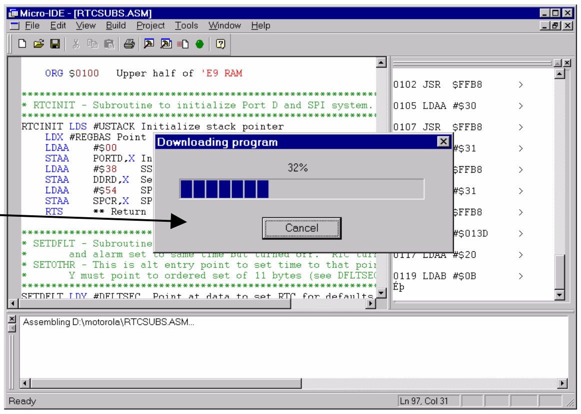
```
Compiling C:\TB1\tb1.c...
Linking tb1.asm ...
Generating output...
First pass... Second pass... 0 error(s).
```



Micro-IDE - [tb1.c]
File Edit View Build Project Debug Tools Window Help
Build Debug Find in Files 1 Find in Files 2 Loader
Ready
Ln 47, Col 1
NUM

```
main()
{
    unsigned i;

    /* Turn all lights off */
    setbit(RED);
    setbit(YELLOW);
    setbit(GREEN);
    setbit(BUZZER);

    /* Set the serial port to 9600 Baud */
    serinit(9600);

    printf("\n Checking LEDs and buzzer...");

    for (i=0; i<=5; i++)
    {
        clrbit(RED);
        buzz(10,40);
        setbit(RED)
        delay(50);
```

```
Compiling C:\TB1\tb1.c...
Linking tb1.asm ...
Generating output...
First pass... Second pass... 0 error(s).
```

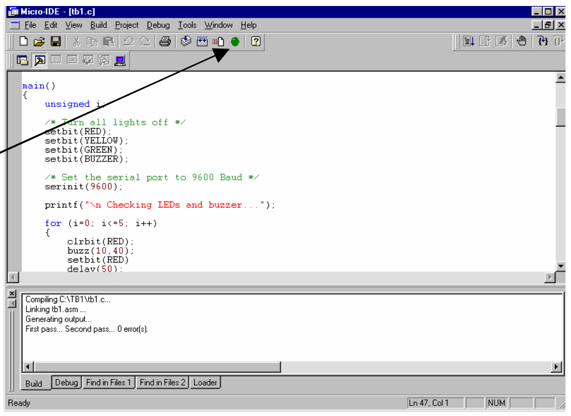


Micro-IDE - [RTCSUBS.ASM]
File Edit View Build Project Tools Window Help
Downloading program
32%
Cancel
Ready
Ln 97, Col 31

```
   ORG $0100   Upper half of 'E9 RAM

*****************************************************************
* RTCINIT - Subroutine to initialize Port D and SPI system.
*****************************************************************
RTCINIT LDS #USTACK Initialize stack pointer
   LDX #REGBAS Point
   LDAA    #$00
   STAA    PORTD,X In
   LDAA    #$38    SS
   STAA    DDRD,X  Se
   LDAA    #$54    SP
   STAA    SPCR,X  SP
   RTS     ** Return

*********************
* SETDFLT - Subroutine
*       and alarm set to same time but turned off.  RTC turn
* SETOTHR - This is alt entry point to set time to that poin
*       Y must point to ordered set of 11 bytes (see DFLTSEC
*****************************************************************
SETDFLT LDY #DFLTSEC  Point at data to set RTC for defaults
```

```
0102 JSR  $FFB8      >
0105 LDAA #$30       >
0107 JSR  $FFB8      >
          #$31       >
          $FFB8      >
          #$31       >
          $FFB8      >
          #$013D     >
0117 LDAA #$20       >
0119 LDAB #$0B       >
Ëþ
```

```
Assembling D:\motorola\RTCSUBS.ASM...
```



Micro-IDE - [tb1.c]
File Edit View Build Project Debug Tools Window Help
Build Debug Find in Files 1 Find in Files 2 Loader
Ready
Ln 47, Col 1
NUM

```
main()
{
    unsigned i;

    /* Turn all lights off */
    setbit(RED);
    setbit(YELLOW);
    setbit(GREEN);
    setbit(BUZZER);

    /* Set the serial port to 9600 Baud */
    serinit(9600);

    printf("\n Checking LEDs and buzzer...");

    for (i=0; i<=5; i++)
    {
        clrbit(RED);
        buzz(10,40);
        setbit(RED)
        delay(50);
```

```
Compiling C:\TB1\tb1.c...
Linking tb1.asm ...
Generating output...
First pass... Second pass... 0 error(s).
```

Figure 3.

LAB1 – Introduction to the 6808 Training Kit

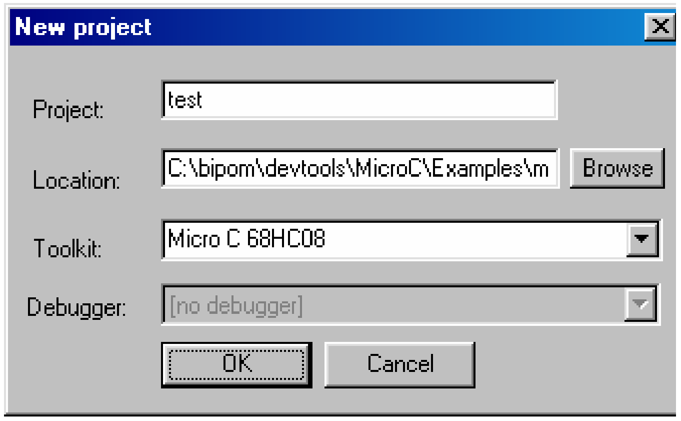
Overview

The purpose of this lab is to become familiar with the 6808 Training Kit and the program development environment. In this lab, you will create a simple program in C language, compile the program to form a hex file, download the program to the board and execute the program.

The knowledge developed in this lab will be very useful in subsequent labs when working with the Motorola 68HC08 micro-controller to develop programs.

Instructions

To create your own project, select Project menu and then New Project. This will display the New Project dialog. Enter the name of the new project and its location (this example uses **test** as the project name and **c:\test** as the project location). Toolkit is already pre-selected as Micro C 68HC08. Click OK; the new project with the name of test under c:\test will be created.


New project

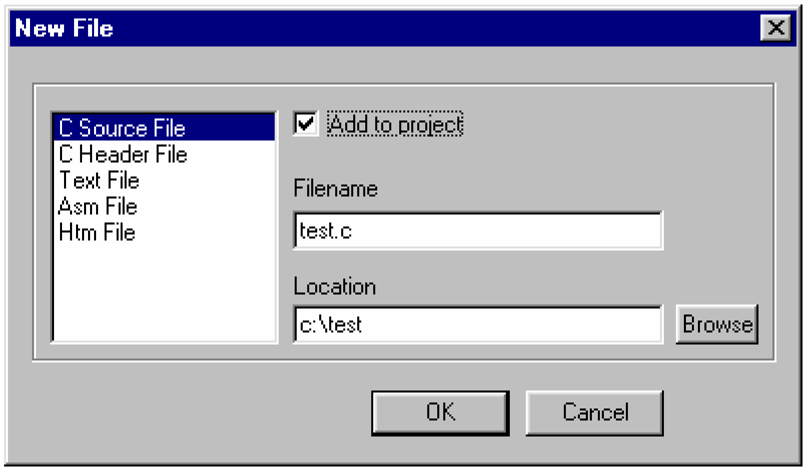
Project: test

Location: C:\bipom\devtools\MicroC\Examples\m

Toolkit: Micro C 68HC08

Debugger: [no debugger]

To add a C source file to the project, first create the C source file. Select File menu and New. New File dialog will appear. Enter the name of the C source file (for example, test.c) and its location (c:\test). The file type is C Source File. Check the Add to project box so test.c will be automatically added to test project. Click OK.


New File

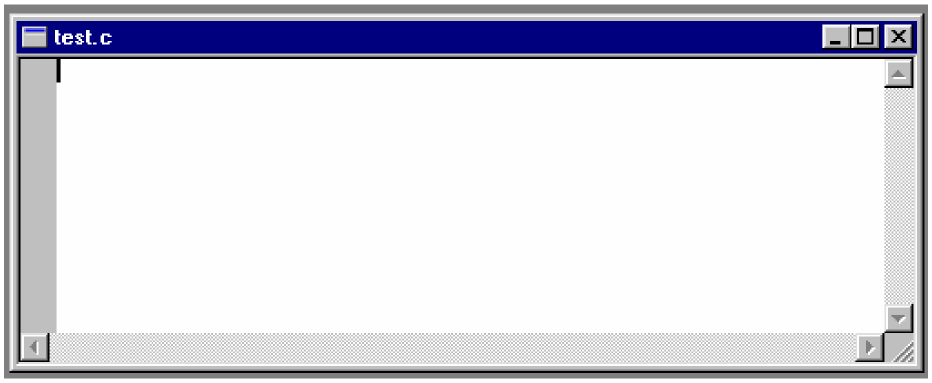
C Source File
C Header File
Text File
Asm File
Htm File

☑ Add to project

Filename: test.c

Location: c:\test

A blank C source file (test.c) will be created:



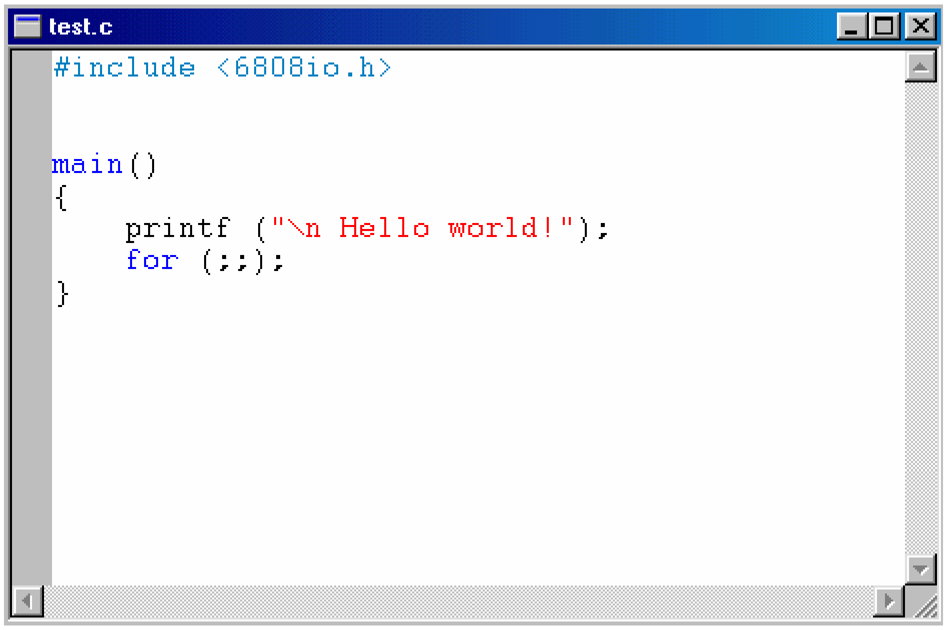
Type the following lines which are the header files required by the Micro C Compiler:

```
#include <6808io.h>
```

Then type the following small C program:

```
main()
{
      printf( “\n Hello World!” );
      for(;;);
}
```

The C source file should now look like this:

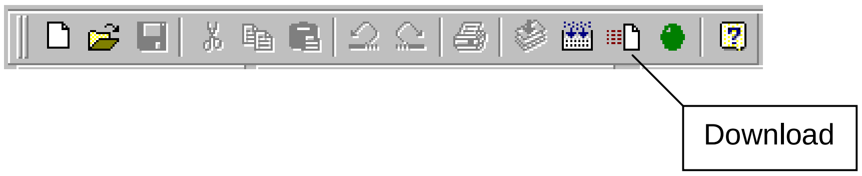


```
#include <6808io.h>


main()
{
    printf ("\n Hello world!");
    for (;;);
}
```

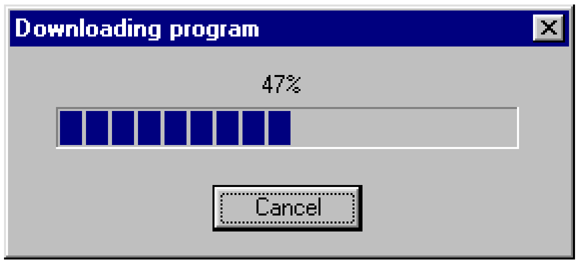
This program prints " Hello World!" on the terminal window. The program then enters an infinite loop. Build the program by clicking the Build button. If the program builds successfully, you will see the following messages on the Output Window:

```
Compiling c:\test\test.c…
Linking test.asm …
Generating 'test.hex'…
First pass… Second pass… 0 error(s)
```

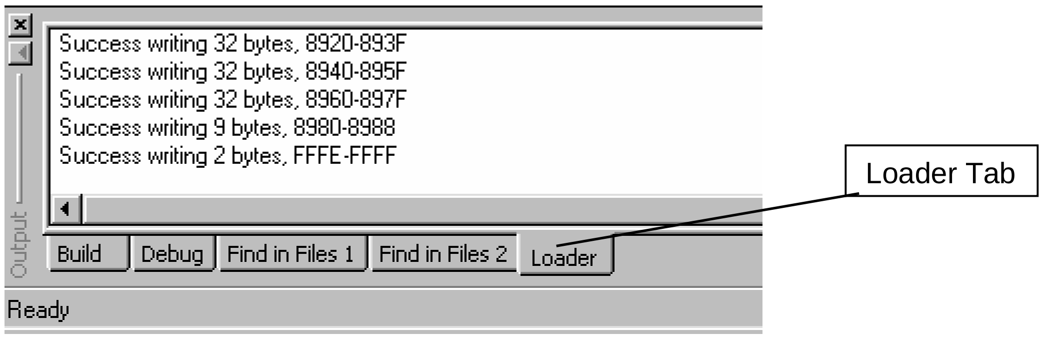
You can download the program to the board by clicking the Download button on the Standard toolbar:



If the board is powered and connected properly to the PC, a progress dialog will appear:



The progress dialog will disappear following a successful download. You can see the details of the download by selecting the Loader Tab on the Output Window

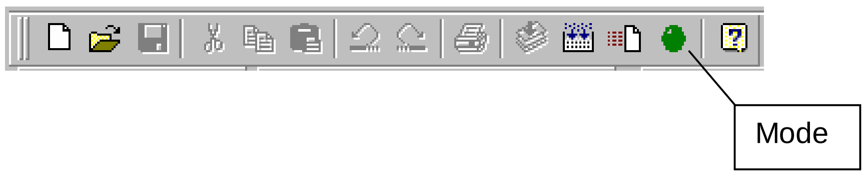


```
Success writing 32 bytes, 8920-893F
Success writing 32 bytes, 8940-895F
Success writing 32 bytes, 8960-897F
Success writing 9 bytes, 8980-8988
Success writing 2 bytes, FFFE-FFFF
```

If there is an error during download, check the following:

1. Is the power adapter plugged to the 6808 Training Kit?
2. Is the serial cable securely attached to the 6808 Training Kit?
3. Is the serial cable securely attached to the PC?

After the program has been successful downloaded, it can be started using the Mode button on the Standard toolbar:



Mode button puts the board into Run or Program modes. In Run mode, the micro-controller is executing the program in its memory. In Program mode, the micro-controller is in Reset state so no programs are running. In Program mode, micro-controller's flash memory can be changed and a new program can be downloaded.

The Mode button is Red in Program mode and Green in Run mode. Following a download, the Mode button will be Red. Click the Mode button to change the mode to Run mode. The program (test.c) that you just downloaded starts executing and prints the message " Hello World!" on the terminal window.

Click the Mode button once again so it turns Red. The board is now in Program mode.

LAB2 – Input/Output

Overview

The purpose of this lab is to control the outputs of the micro-controller in a given sequence. Green, yellow and red Light Emitting Diodes (LED’s) on the TB-1 board are connected to micro-controller outputs.

Initially you will write a program to turn on only one LED and then turn off the same LED, Then, you will improve the program to blink the LED.

The next part of the lab is to be able to read input switches. As switches are mechanical objects, some de-bounce time (dead time) will also be placed in the program. You will control an LED with one switch; as long as switch is active the respective LED is On and when switch is inactive, the corresponding LED is Off. Then, the LED will be made to blink as long as switch is On. Finally, you will write a little traffic light controller where the green, red, yellow LED's on the TB-1 board simulate the traffic lights and the switches simulate the car presence sensors at a crossroad.

Information

Most of the micro-controller pins and the power supply are available on the 20-pin connector (J1) for interfacing to external circuitry, prototyping boards and peripheral boards. Table 1 explains the J1 pinouts to the micro-controller pins.

MINI-MAX/908-C Expansion (J1)

Signal	Pin	Pin	Signal
PTD7	20	19	PTD6
PTD5 or IRQ	18	17	PTD4
PTD3	16	15	PTD2
PTD1	14	13	PTD0
PTB0	12	11	PTB1
PTB2	10	9	PTB3
PTB4	8	7	PTB5
PTB6	6	5	PTB7
VCC	4	3	GND
VCC	2	1	GND

Table 1.

Two ports of the micro-controller are accessible via the X3 terminal block. TB-1 board, which is connected to the micro-controller board as explained in Lab1, will be used as it already has input switches and output LED's connected. Table 2 describes the pinouts on the X3 terminal block and their designation as respective to input (switches) or output (LED's).

Micro-controller	**X3 terminal**	**Input or Output**
PTD5	18	Input, active Low, Sw1
PTD4	17	Input, active Low, Sw2
PTB3	9	Output, active High, Red Led
PTB1	11	Output, active High, Yellow Led
PTB0	12	Output, active High, Green Led

Table 2.

The term active low indicates that the micro-controller must be programmed in a manner such that the switch is made or activated when PTD5 or PTD4 read low logic. Correspondingly the term active high means that to activate an LED the corresponding micro-controller pin must generate high logic. The port directional register defines the state (input or output) of the micro-controller I/O lines.

Initially turn the individual LED's On and Off without interaction with the switches.

- Turn Red Led On only for a brief time interval and then Off
- Turn Yellow Led On only for a brief time interval and then Off
- Turn Green Led On only for a brief time interval and then Off
- Blink Red Led only
- Blink all three LED's

You now have the capability of controlling more than one output line. Next step is to read an input line. When the Switch is not activated the respective line is high and a Led should be put On. When the Switch is activated then Led should blink.

- Turn Red Led On if Switch 1 is inactive.
- Turn Yellow Led On if Switch 2 is inactive.
- Blink Red Led as long as Switch 1 is active and Red Led is On if Switch 1 is inactive.
- Blink Yellow Led as long as Switch 2 is active and Yellow Led is On if Switch 2 is inactive.

Write a program to generate the results shown in Table 3 for a traffic light controller:

Switch 1	**Switch 2**	**Red LED**	**Yellow LED**	**Green LED**
Open	Open	OFF	OFF	ON
Open	Closed	ON	ON	OFF
Closed	Open	OFF	OFF	ON
Closed	Closed	OFF	OFF	OFF

Table 3.

Build the program and download to the Training Board. Open and close Switch 1 and Switch 2 to change the LED’s to make sure that your program is handling every case correctly.

LAB3 – Analog/Digital Conversion

Overview

This lab is used to familiarize the student with the operation of an A/D converter. An analog voltage is applied to the input of the A/D converter by using a potentiometer and its respective output is read into the micro-controller and displayed on the terminal screen. The A/D and the potentiometer are connected on the TB-1. LM35 type temperature sensor should be connected to the A/D and temperature values will be displayed on the terminal window. C program will take care of all the software.

Information

Voltages that continuously vary as a function of time are defined as analog voltages and can have any value within certain range, e.g. 0V to 5V, or –5V to + 5V, or 0V to +12V etc. Digital voltages only have two values i.e. a 1 or a 0. An A/D converter (Analog to Digital converter) writes the value of the analog voltage into digital format or a code, which is then processed by the micro-controller.

An 8-bit A/D converter ADC0834 from National Semiconductor as shown in Figure 3 is connected on the TB-1 board.

The pinouts for this 14 pin IC (ADC0834) is shown in Table 4.

Pin#	Mnemonic	Description
1	V+	
2	CS	Active low
3	CH0	Analog Input Channel 0, 0V to 2.5V DC
4	CH1	Analog Input Channel 1, 0V to 2.5V DC
5	CH2	Analog Input Channel 2, 0V to 2.5V DC
6	CH3	Analog Input Channel 3, 0V to 2.5V DC
7	DGND	Digital Ground.
8	AGND	Analog Ground
9	V ref	Reference voltage for the A/D, 0V to 5V DC
10	DO	Output Signal generated by the A/D
11	SARS	Output, when high A/D in progress, when low data output
12	CLK	Input, Clock signal to the A/D
13	DI	Input, Data to the A/D
14	VCC	Input, Power to the A/D

Table 4.

Table 5 shows the connections of the 14 lines of the A/D to the J1 connector.

J1	Micro-controller	A/D Mnemonic	A/D
14	PTD1	CS	2
5	PTB7	DO	10
N/C	N/C	SARS	11
6	PTB6	CLK	12
13	PTD0	DI	13

Table 5.

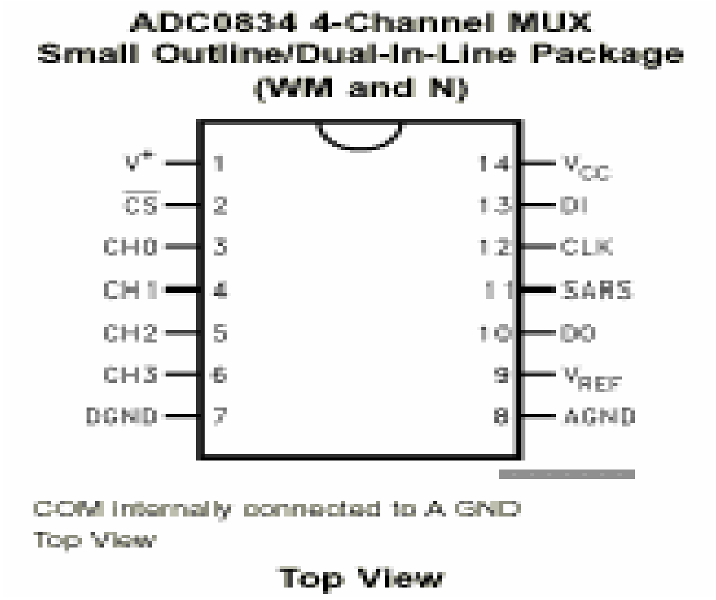
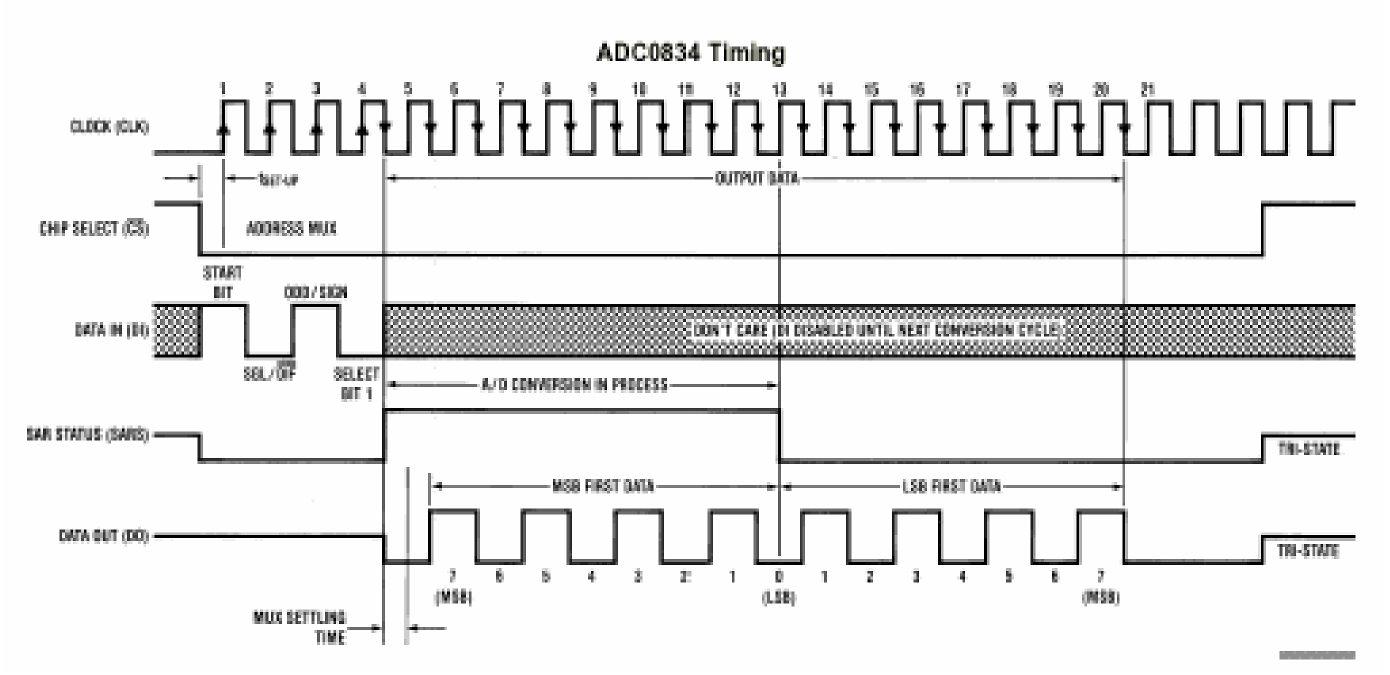
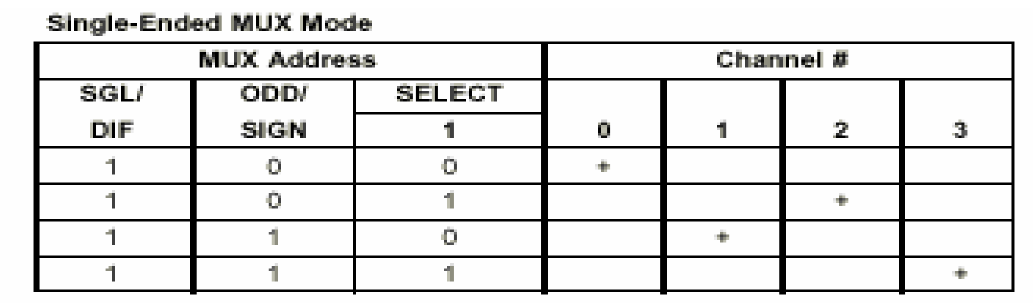


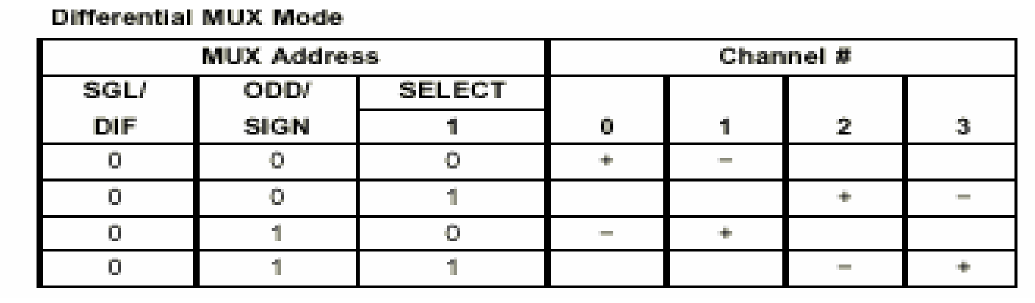
Figure 4.





Single-Ended MUX Mode

MUX Address			Channel #			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
1	0	0	+			
1	0	1			+	
1	1	0		+		
1	1	1				+



Differential MUX Mode

MUX Address			Channel #			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
0	0	0	+	–		
0	0	1			+	–
0	1	0	–	+		
0	1	1			–	+

Figure 5.

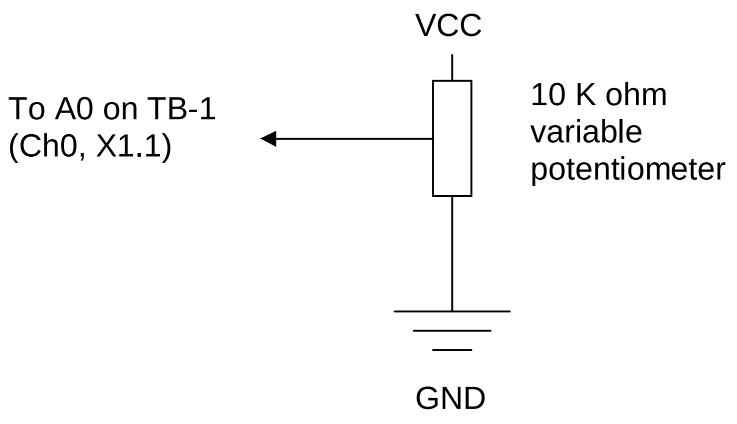


Figure 6.

Pin3 of the A/D receives the Analog input voltage signal. A DC Voltmeter, which is connected to A0 on TB-1 board, should read between VCC and GND as the potentiometer is varied from one extreme to the other as shown in Figure 6. The temperature sensor LM35 is later on also connected here.

The A/D converter consists of 4 input multiplexed analog channels, which may be software configured as 4 single-ended channels or 2 differential channels, or a new pseudo differential option. The input format is assigned during MUX addressing sequence prior to start of conversion and this selects the analog input and their mode (single or differential).

Steps for A/D Conversion

- Initially the DI and CS inputs must be high.
- Pull the CS (chip select) line low. This line must be held low for the entire duration of the conversion. The converter is waiting for the Start bit and the MUX assignment.
- A clock is generated by the processor (if not provided continuously) and output to the A/D clock input.
- The start bit is a logic "1" on the DI input line, after which, the MUX assignment word on the DI line is presented. The status of the DI line is clocked on each rising edge of the clock line.
- The SARS status output line from the A/D goes high. This indicates that a conversion is now in progress and the DI line is disabled.
- The data out (DO) line comes out of Tri-state and provides a leading zero for one clock period.
- After 8 clock periods the conversion is complete. The SARS line returns low ½ clock cycle later.
- The DO line now generates the binary equivalent of the analog value as 8 output bits with LSB first. The DO line than goes low and remains low until chip select (CS) is made to go high. This clears all the internal registers and another conversion may be started by making the CS go low again and repeating the process.

Exercise

Write a C program that uses the A/D in single ended mode with the potentiometer connected to Channel 0 or pin3 on the A/D. Vary the analog input voltage and display on the screen. After this connect input potentiometer to Pin4 also and display its value. Connect the temperature sensor to pin3 and potentiometer to pin4 and display both the values simultaneously on the terminal screen.

LAB4 - Keypad

Overview
Student connects a keypad to the keypad connector on the 6808 Training Kit and programs the keypad.

Information
The 6808 Training Kit board has a 10-pin connector (J3), which is connected to PTA of the micro-controller and is used as the keypad connector. The pinouts of the connector are listed below in Table 5.

Micro-controller Port Pin	Keypad Connector Pin#
PTA0	1
PTA1	2
PTA2	3
PTA3	4
PTA4	5
PTA5	6
PTA6	7
PTA7	8
Ground	9
VCC	10

Table 5.

Many keypads are wired as a matrix of rows and columns. The internal connections of a 4 row by 3 column keypad is shown in Figure 7.

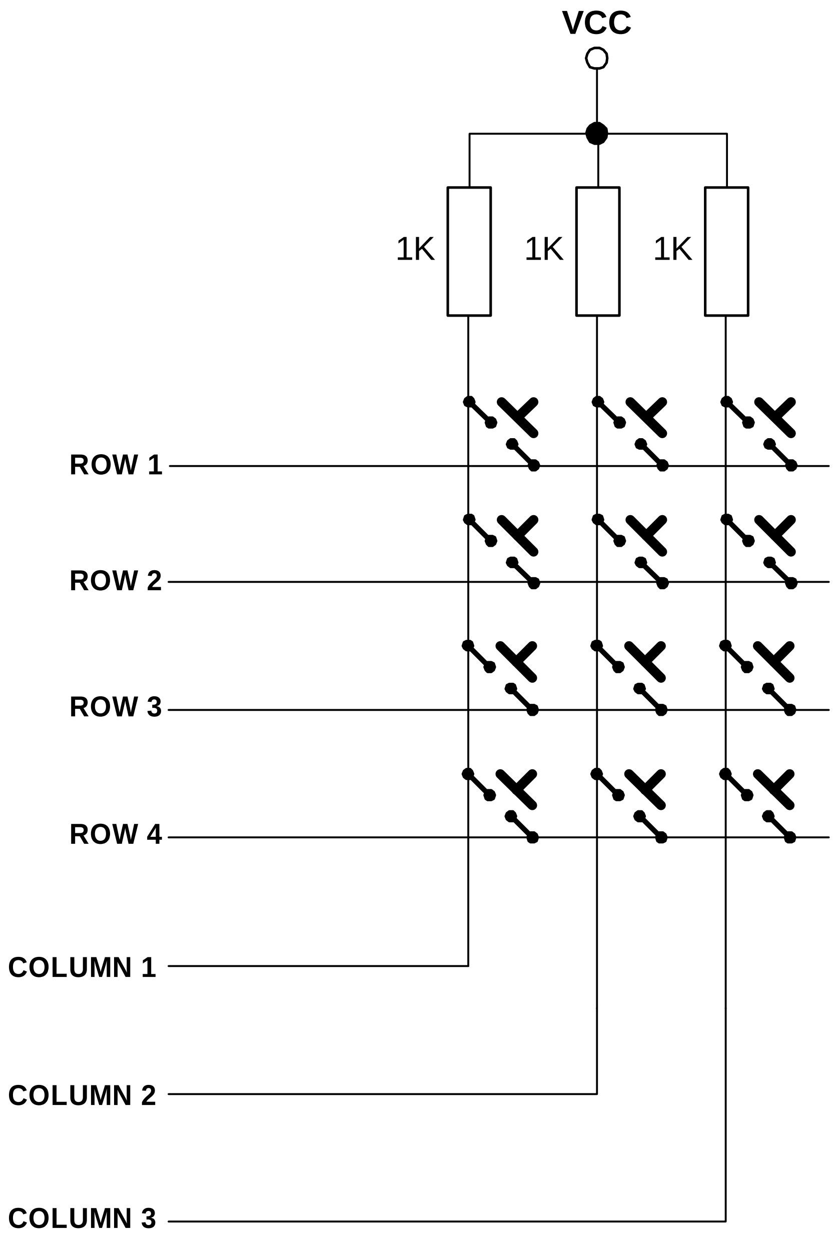


Figure 7.

Matrix connection saves on the number of connections and micro-controller port lines. For example, a 4 row by 3 column keypad would require 13 wires (12 + ground) if each key was individually connected to micro-controller ports. Using the matrix approach and scanning the keypad under software control reduces the number of wires and port pins to 7 (4 rows + 3 columns).

When a key is pressed, the row for that key will be physically connected to the column for that key. Therefore, the port input for the column will be at the same logic level as the port output for the row.

Since the columns (inputs) are normally at the HIGH logic level due to pull-up resistors, the only way to make a column LOW will be to press a key and make the row for that key LOW. By periodically strobing each row LOW one row at a time, and reading the column input levels during each strobe, one can determine which key is pressed.

This is illustrated Table 6 for the 4 by 3 keypad. In the Row Mask, Row 1 is assigned to the Most Significant Bit and Row 4 is assigned to the Least Significant Bit. Similarly in the Column Mask, Column 1 is assigned to the Most Significant Bit and Column 3 is assigned to the Least Significant Bit.

Action	Row Mask	Column Mask
No keys were pressed	XXXX	111
Row 1 Column 1 key pressed	0111	011
Row 1 Column 2 key pressed	0111	101
Row 1 Column 3 key pressed	0111	110
Row 2 Column 1 key pressed	1011	011
Row 2 Column 2 key pressed	1011	101
Row 2 Column 3 key pressed	1011	110
Row 3 Column 1 key pressed	1101	011
Row 3 Column 1 key pressed	1101	101
Row 3 Column 1 key pressed	1101	110
Row 4 Column 1 key pressed	1110	011
Row 4 Column 1 key pressed	1110	101
Row 4 Column 1 key pressed	1110	110

Table 6.

The Keypad algorithm can be based on the following rules.

<u>Algorithm</u>

- A Column is generally high (output).
- One row at a time is made to go low (input) and than the columns are read.
- If one or more columns are low than the switches of the corresponding columns are active and their respective values should be displayed on the terminal.

LAB5 – Liquid Crystal Display

Overview
This Lab familiarizes the student in connecting a dot matrix LCD to the LCD connector of the 6808 training kit. The student writes a program in C to display various characters using 4-bit mode.

Information
Dot matrix LCD displays are readily available from many companies like Sharp, Hitachi, Optrex, Noritake etc. They generally come in display formats of 16*1, 16*2, 24*2, 40*4 (column * row) etc. They have 8 data lines DB0 – DB7, Power VCC, Ground GND, Reset RS, R/W Read/Write and Enable E. Additional power or ground pins may be present. This exercise will use the Hitachi LM032L 20 * 2 display, to be connected to the LCD connector. The three control lines are explained below.

RS Register Select Control
1 = LCD in data mode
0 = LCD in command mode
E Data / Control state
Rising Edge = Latches control state
(RS and R_W)
Falling Edge = Latches data
R_W Read / Write control
1 = LCD to write data
0 = LCD to read data
In the 4-bit mode, data is transferred either on the lower or upper nibble of the port, this saves in I/O lines but the program occupies more space as two commands are required to display a character. Table 7 shows the connection between the display and the LCD connector in 4-bit mode with low nibble.

LCD Connector	Micro-controller	LCD Display (LM032L)
1	GND	1
2	VCC	2
3		
4	PTC3	DB4, 11
5	PTC4	DB5, 12
6	PTC5	DB6, 13
7	PTC6	DB7, 14
8	GND	
9	GND	
10	GND	
11	PTC0	RS, 4
12	PTC1	R / W, 5
13	PTC2	E, 6
14	VEE	

Table 7.

LAB6 - Buzzer

Overview
Student programs the buzzer on the TB-1 to generate different notes using software. Student then generates a little musical piece using the notes that he/she programmed.

Information
A buzzer or simple speaker will generate music when a series of square waves is applied to its positive input with the negative grounded. The frequency of the square wave should be up to 12KHz to 15KHz. Changing the duty cycle of the square wave may vary the loudness of the note. Different musical tones will be generated by varying the frequency of the square wave.

The buzzer on the TB-1 is connected to pin 10 of J1, which is PTB.2 on the micro controller. This pin needs to be programmed as an output pin and square waves of varying frequencies and duty cycle need to be generated.

Programming Steps
- Program PTB.2 on the micro-controller as an output pin
- Decide a time period or frequency of square wave between 1KHz to 15KHz.
- Generate a signal of 50% Duty cycle, where
  Duty Cycle = On Time / (On Time + Off Time) of square wave.
- Activate buzzer for approx 5 to 10 seconds
- Change frequency of buzzer and note the different sound
- Change only duty cycle and note the different intensity.