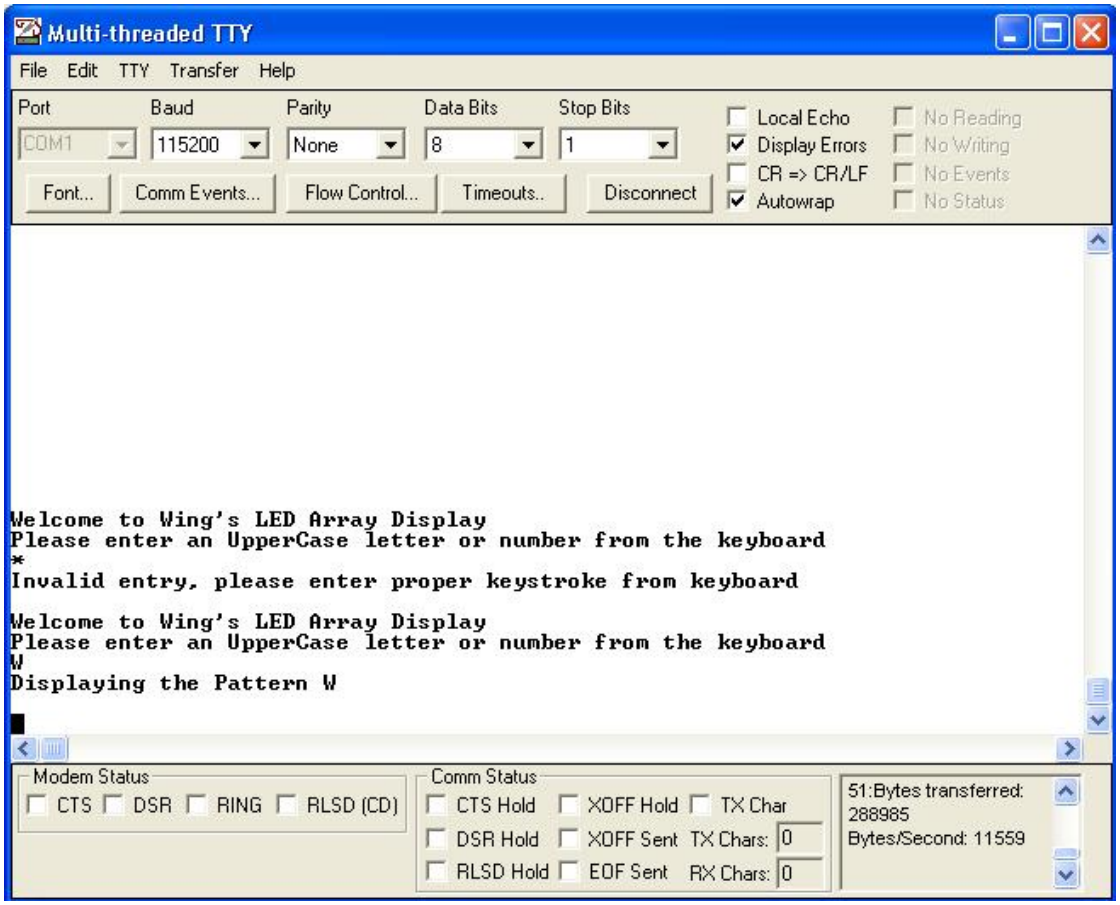
Lab 4: Introduction to Hardware Interfacing



Lab Dates

Refer to the schedule on the ECE212 Laboratories page for the latest schedule

Introduction

In this lab the students will be using the Netburner board along with a buffer board to interface with the GMC2288C 8X8 LED array. In addition, two 3X8 74LS138 decoders, two hex-input 74LS04 inverters and eight 1KΩ resistors will be used. A schematic for the hookup is provided. A hardware verification program is also provided so that the student may verify the hardware hookup before progressing to the lab requirements.

The bufferboard isolates the output pins from the Netburner board to prevent any shorting of the pins. The output pins used from the Netburner are mapped to the bufferboard 10 Pin Ribbon Cable as follows:

P37(Netburner) = P1(Ribbon Cable) ⟶ least significant bit of decoder that controls column selection
P38(Netburner) = P2(Ribbon Cable)⟶ second bit of decoder that controls column selection
P39(Netburner) = P3(Ribbon Cable) ⟶ most significant bit of decoder that controls column selection
P40(Netburner) = P4(Ribbon Cable) ⟶ least significant bit of decoder that controls row selection
P41(Netburner) = P5(Ribbon Cable) ⟶ second significant bit of decoder that controls row selection
P42(Netburner) = P6(Ribbon Cable) ⟶ most significant bit of decoder that controls row selection
P43(Netburner) = P7(Ribbon Cable) ⟶ Enable pin for both 74LS138 decoders. Logic 1 = ON Logic 0 = OFF
P8(Ribbon Cable) ⟶ Not Used
P9(Ribbon Cable) ⟶ Ground(0V)
P10(Ribbon Cable) ⟶ Power(5V)

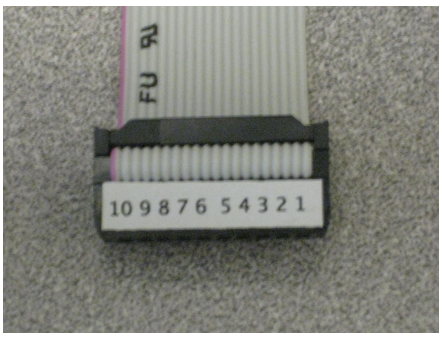


Figure 1 - Ribbon Cable

Note: From the LED array, think of every light as being on a 8 by 8 grid system. The orientation of the LED array in the schematic gives the orientation of the rows and columns. The upper left-hand corner of the array is row 0, column 0 as denoted in figure 2. Look for the part number on the side of the array and

orient it the same way as in the schematic. Wiring the components in any other way will modify the grid accordingly.

Example: To turn on the the LED at coordinates (0,0), we need to set the logic level on pins P1-P3(column) and P4-P6(row) low and P7 high. To turn on the the LED at coordinates (7,7), we need to set the logic level on pins P1-P3(column), P4-P6(row) and P7 high.

For convenience I have provided four subroutines Row,Col,LEDON,LEDOFF that will simplify this procedure of turning on a individual LED. The steps are as follows.

1. Push the LED row number onto the stack. Jump to Row subroutine. Clean up stack.
2. Push the LED col number onto the stack. Jump to Col subroutine. Clean up stack
3. Jump to LEDON subroutine to turn on the LED
4. Jump to LEDOFF subroutine to turn off the LED

Objectives:

1. To gain experience in interfacing external circuitry to the Netburner Board
2. To gain experience in using the Netburner I/O ports along with serial communcation from the keyboard.
3. To gain more experience in writing subroutines

Prelab and Preparation:

- Read the lab prior to coming to your lab section.
- Do the online prelab Quiz
- provide flowchart for each subroutine(3 in total)

View the datasheets of the various components:

- 74LS138
- 74LS04
- GMC2288C
- Pin Layout Diagrams of the Hardware Components

Lab Work and Specifications

1. Sign out the breadboard and components from the TA
2. View the schematic and hook up the circuit based on the orientation in the diagram. Refer to the picture in the Appendix for an example of the layout. The lab instructor's model is also available for viewing

3. Download the hardware verification program for this lab. After wiring up your array, decoders, and inverters based on the schematic, flash the board with this program to verify that your circuit is hooked up properly.
4. Download the template files

    - main.cpp
    - Lab4.s
    - Lab4a.s
    - Lab4b.s
    - Lab4c.s
    - pattern.s
    - getpatternaddress.s

Specifications

If you recall from the Lab1, the 'main.cpp' is the standard project template that is used to initialize and call standard functions/subroutines including our 'AssemblyProgram'. Do not modify any of the parameters in this file. 'Lab4.s' has been provided to call your subroutines. Do not modify any of the parameters in this file. Lab4a.s, Lab4b.s and Lab4c.s are provided for you to write your subroutines. Each subroutines has certain passed parameters and certain conditions. Carefull attention should be paid. A more detail description is provided below.

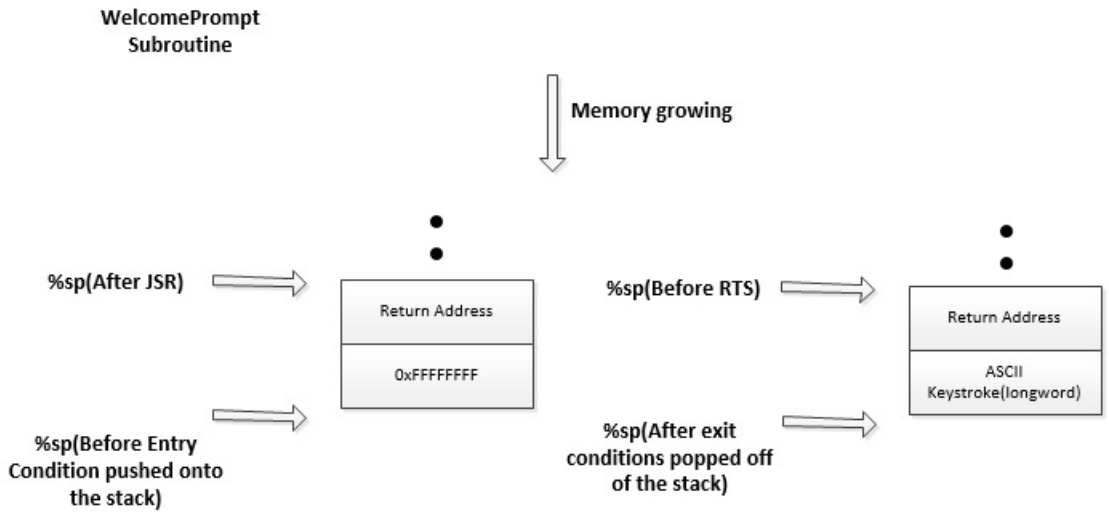
The requirements on each subroutine are indicated below.

1. WelcomePrompt

    In the Lab4a.s template file, you are to code a subroutine called *WelcomePrompt* that prompts the user to enter a keystroke from the keyboard. The keystroke entered by the user must be an upper case letter or number. If the condition is not met, your program will reprompt the user to enter a proper keystroke. The valid keystroke(longword) which is stored as its Ascii code will be passed out through the stack.

    Entry Condition = space allocated for the keystroke on stack(longword)

    Exit Condtion = Keystroke(longword)



Example - Your program should look something like this:

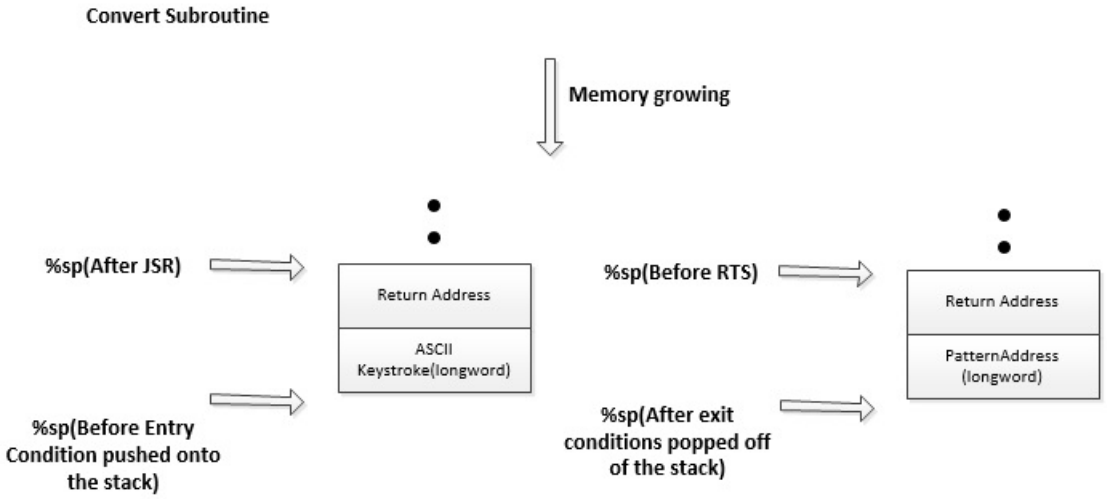
- Display a welcome string on startup.
  Welcome to Wing's LED Display
- Display a string prompting the user to enter the number of entries.
  Please enter an UpperCase letter or Number from the keyboard
- Display a string indicating invalid entry if an improper keystroke was entered.
  Invalid entry, please enter proper keystroke from keyboard

2. Convert

   In the Lab4b.s template file, you are to code a subroutine called *convert* which takes the keystroke from the stack and passes out the pattern address through the stack. The pattern is stored as 2 longwords(8bytes). For example, the keystroke 'A' might have its pattern address stored starting at memory location 0x2300000. A subroutine called convert1 has been provided to shorten the workload. Make use of it. The pattern address(longword) will be passed out through the stack. This is a short subroutine and should take no more than 10 lines of code.

   Entry Condition = Ascii character(longword) on stack

   Exit Condition = Pattern address(longword) on stack

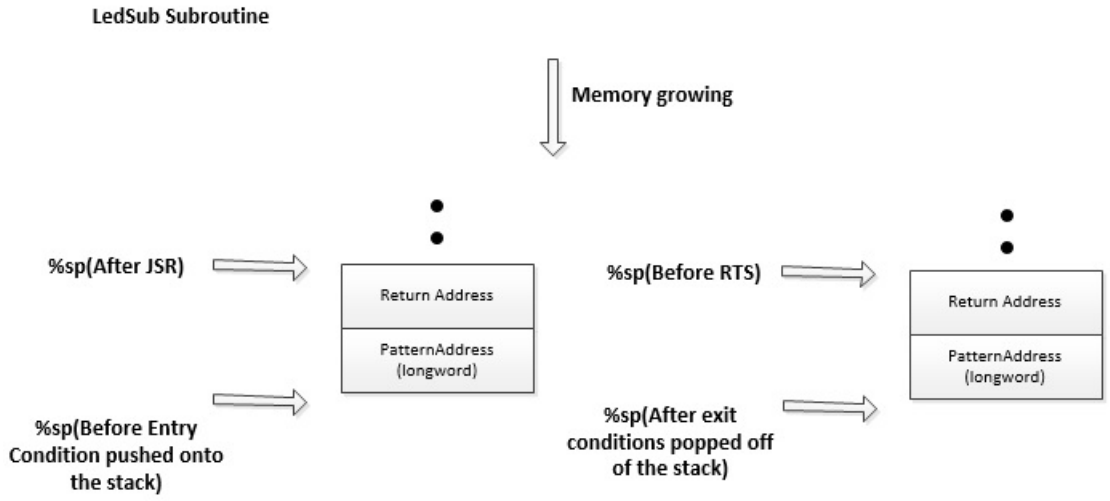


3. LedSub

   In the Lab4c.s template file, you are to code a subroutine called *LedSub* that takes the address of the pattern from the stack and displays the pattern on the LED array by lighting up the proper LEDs. To turn on the LEDS on the 8X8 LED segement display, use the 5 subroutines provided *Row*, *Column*, *TurnOnLed*, *TurnOffLed* and *Delay*. Only one LED can be turned on at a time. However, by changing the timing in delay, one can simulate having them all on by cycling through the LEDS at a fast speed.

   Entry Condition = Pattern address(long) on stack

   Exit Condition = Pattern address(long) on stack



To get started with this subroutine, you should initialize your pointer to the coordinates(0,0) on the LED array and have it turned off. From there you can start displaying your pattern one LED at time. Cycle through your pattern slowly by delaying it and once it works, increase the delay.

Note: For each subroutine, only the information provided can be used. No assumptions can be made. In addition, each subroutine must preserve the register values and the stack must be cleaned up if necessary.

Provided Subroutines
An example of how to turn on a LED will be provide at the end of this section.

1. convert1

   This subroutine is written in the template file getpatternaddress.s. It takes the Ascii character from keyboard and returns the address of where the character pattern is stored.

   Entry Condition = ascii data on the stack(long)

   Exit Condition = address of pattern of character on the stack(long)

   Example of pseduocode usage:
   Push Ascii character onto the stack
   Jump to convert1 subroutine
   Pop off pattern address off of the stack
   Clean up stack if necessary

2. TurnOnLed

   This subroutine outputs a logic one on P7(Ribbon). P7 is wired to the enable pins on the two 3by8 logic decoder chips. Essentially, turning both decoder chips on.

Entry Condition = None

Exit Condition = None

Example of pseduocode usage:
Jump to TurnOnLed subroutine
Clean up stack if necessary

3. TurnOffLed

   This subroutine outputs a logic zero on P7(Ribbon). P7 is wired to the enable pins on the two 3by8 logic decoder chips. Essentially, turning both decoder chips off.

   Entry Condition = None

   Exit Condition = None

   Example of pseduocode usage:
   Jump to TurnOffLed subroutine
   Clean up stack if necessary

4. Row

   This subroutine selects which row the LED will be turned on. For example, if you want LED in row 3 to be turned on, the value 3 would be pushed onto the stack and this subroutine would be called.

   Entry Condition = row number(longword) Valid entries are 0 to 7. (0=row0,7=row7)

   Exit Condition = row number(longword)

   Example of pseduocode usage:
   Push Decimal value onto the stack(0-7)
   Jump to Row subroutine
   Clean up stack if necessary

5. Column

   This subroutine selects which column the LED will be turned on. For example, if you want LED in column 3 to be turned on, the value 3 would be pushed onto the stack and this subroutine would be called.

   Entry Condition = column number(longword)
   Valid entries are 0 to 7. (0=column0,7=column7)

   Exit Condition = column number(longword)

   Example of pseduocode usage:
   Push Decimal value onto the stack(0-7)
   Jump to Column subroutine
   Clean up stack if necessary

6. Delay

   This subroutine delays the program a certain amount of time. This subroutine is used to give the illusion that all the LEDs for a particular pattern is on. By choosing a proper integer value, you can try to minimize the amount of flicker in the LEDS.

   Entry Condition = delay time(Longword)

   Exit Condition = delay time(Longword)

   Example of pseduocode usage:
   Push Decimal value onto the stack
   Jump to Delay subroutine
   Clean up stack if necessary

7. iprintf

   Prints a string to the monitor. No carriage return or linefeed is invoked after calling this function.

   Entry Condition = address of string on the stack(longword)

   Exit Condition = address of string on the stack(longword)

   Example of pseduocode usage:
   Push StringLabel onto the stack
   Jump to iprint subroutine
   Clean up stack if necessary

8. getchr

   A function that prompts the user to enter a stroke from the keyboard. The data entered is stored as an Ascii code character in the data register D0. For example, if the keystroke 'A' was entered, D0 would contain the data 0x41. No carriage return or linefeed is invoked after calling this function.

   Entry Condition = None

   Exit Condition = Ascii keystroke stored in register D0

   Example of pseduocode usage:

   Jump to getchr subroutine
   Clean up stack if necessary

9. putchr

   Prints an ascii character to the monitor. No carriage return or linefeed is invoked after calling this function.

   Entry Condition = ascii code on the stack(longword)

   Exit Condition = ascii code on the stack(longword)

Example of pseduocode usage:
Push ascii code onto the stack
Jump to putchr subroutine
Clean up stack if necessary

10. cr

    A function that generates a carriage return and linefeed

    Entry Condition = None

    Exit Condition = None

    Example of pseduocode usage:
    Jump to cr subroutine
    Clean up stack if necessary

Example of code snipet that turns on and off the LED at column 3, row 4 on the 8by8 LED array

move.l #3,-(%a7) /*push #3 onto the stack */
jsr Column /*Jump to Column subroutine */
move.l (%a7)+, %d2 /*clean up the stack */

move.l #4, -(%a7) /*push #4 onto the stack */
jsr Row /*Jump to Row subroutine */
move.l (%a7)+, %d2 /*clean up the stack */

jsr TurnOnLed /*jump to TurnOnLed subroutine */

move.l #300, -(%a7) /* push delay number onto the stack*/
jsr Delay /*jump to delay subroutine*/
move.l (%a7)+, %d2 /*clean up stack*/

jsr TurnOffLed /*turn off LED*/

Questions

1. Why were the two 74LS138 decoders used in the circuit instead of directly connecting the pins to the array? Discuss how removing the decoders would affect the function of the LED array in the context of this lab.

Marking Scheme

Lab 4 is worth 25 % of the final lab mark. Please view the Marking Sheet for this lab to ensure that you have completed all of the requirements of the lab. The Marking Sheet also provides a limited test suite in the demo section for you to think about. Make use of it!

Demo and Report

There is no report for this lab. The demo due dates are given on the ECE212 Laboratories page. Note that you have one week from the dating of your prelab to complete the demo. Late demos will not be accepted

Appendix

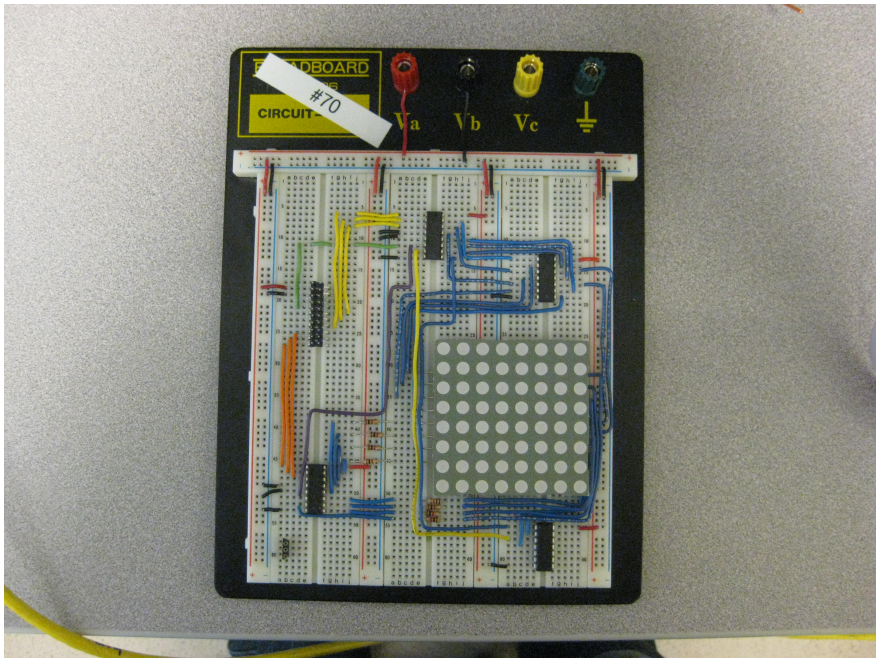


figure 2 - Breadboard Layout