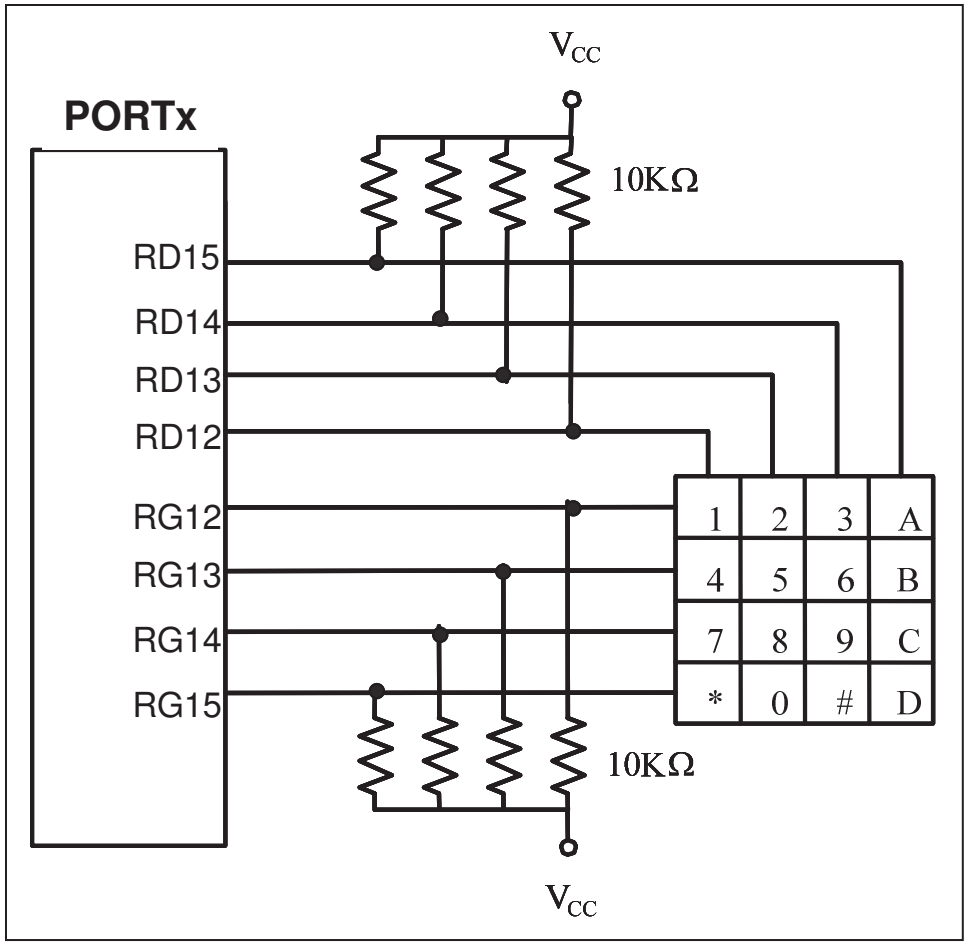
Laboratory #4
Keypad Interfacing

Objective: To write a device driver for interfacing a keypad and to make use of terminal I/O functions for message output on the HyperTerminal window.

Task:

The I/O expansion board attached to the Explorer 16 Development Board has a keypad module on it. In this lab assignment you will be implementing a device driver program to interface the keypad module to your PIC32 MCU. The hardware connection of the keypad module to the MCU's I/O ports is as shown in the schematic diagram below. The upper nibbles (pins 15..12) of PORTD and PORTG are used for the interfacing. External pull-up resistors are attached to all the I/O pins. To find out if and which key is pressed, in the implementation of your device driver, you can sequentially set one row (RG12:15) or column (RD12:15) to low and then check if any of the columns or rows, respectively, returns a zero value. The intersection of a column that gives a value of zero and the currently selected row corresponds to the key pressed.



In your program, you will implement the following two functions with the given prototypes:

1. *unsigned char* ***keyin****(void)*

   Anytime a user is interested to read a character from the keypad, the user will make a call to your *keyin()* function. Your subroutine will then perform a complete scan of the keypad to see if there is any user input at that moment and will return a one byte number. If a key was found pressed during the keypad scan period, the *keyin()* function will first have to *wait* for release of the key and then return the ASCII code of the key that was pressed. However, if there was no key pressed during the scan process, the function simply returns a NULL character which is designated by a byte value of zero.

2. *void* ***main****(void)*

   A test main function that first displays a prompt message (such as "Enter keys on the keypad …") on the user PC monitor using the I/O utility routine *_mon_puts().* Your test program then enters in an infinite loop calling the keypad scanning function *keyin()* to accept user inputs and displaying them on the monitor (using *_mon_putc()*). For information about *_mon_putc()* and *_mon_puts()* refer to the "C32 C Compiler Libraries Guide" that you can find at the Start menu -> Microchip -> MPLAB C32.

   The I/O functions *_mon_puts()* and *_mon_putc()* that we use for displaying messages on the monitor rely on the UART2 module on the PIC32 MCU for communication with the terminal via the RS-232 interface. Therefore, we need to properly initialize the UART2 module before using any of these I/O functions. For this lab assignment you are given the code below for the purpose. Include this code as part of your program source code:

```
#include <p32xxxx.h>
// configuration bit settings, Fcy=80MHz, Fpb=40MHz
#pragma config POSCMOD=XT, FNOSC=PRIPLL
#pragma config FPLLIDIV=DIV_2, FPLLMUL=MUL_20, FPLLODIV=DIV_1
#pragma config FPBDIV=DIV_2, FWDTEN=OFF, CP=OFF, BWP=OFF

#define BRATE 86        //U2BREG setting for ~115,200 baud
#define U_ENABLE 0x8008 //enable UART2, BREGH=1, 1 stop, no parity
#define U_TXRX 0x1400   //enable RX and TX, clear all flags

void initUART2(void)
{
    U2BRG = BRATE;      //intialize the baud rate generator
    U2MODE = U_ENABLE;  //intialize the UART module
    U2STA = U_TXRX;     //enable TX & RX
}

main()
{
       initUART2();
       …

}
…
```

When you are ready to test your program make sure that your Explorer 16 board is connected to your PC using a serial cable (directly or via a Serial-to-USB converter) and then open a HyperTerminal on your PC with the same communication parameters as in your program, i.e. 115200 baud rate, 8-bit data, no parity, 1 stop bit, and no flow control.

Note that as in your previous program you need to take care of the contact bounce problem of the switches in the keypad by using the same strategy. You can refer to the keypad's datasheet (available on Blackboard) for information about its internal configuration and the actual length of time that the contact bounce lasts.

Checkout and report:

Demonstrate your working program to the lab instructor and hand in a lab report that includes the following:

- Title page: course #, course title, Lab# & title, date, professor's name, names of group members
- This lab handout.
- Printout of the C source code.