```
//-----------------------------------------------------------------------------
// Keypad.c
//-----------------------------------------------------------------------------
// Author: Baylor Electromechanical Systems
//
// Operates on an external 18.432MHz oscillator.
//
// Target: Cygnal Educational Development Board / C8051F020
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51
//
// This program interfaces Cygnal's C8051F02x with a 4x4, 16 pad keypad.  The
// program was designed for the Grayhill 96BB2-056-F.  With keypad DIP switch
// (SW5) toggled to the on positions, pins 1-8 on the keypad are
// connected to P2.0 - P2.7.
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include <c8051f020.h>                 // SFR declarations
#include <stdio.h>

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------

#define BAUDRATE     9600              // Baud rate of UART in bps
#define SYSCLK       18432000          // SYSCLK frequency in Hz


// Lookup table for converting keycode to ASCII
unsigned int keytab[4][4] ={{'1','2','3','A'},
                            {'4','5','6','B'},
                            {'7','8','9','C'},
                            {'*','0','#','D'}};


//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------

void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
int button_dn(void);
unsigned int scankey (void);
void delay_ms(int ms);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------

void main (void)
{
   unsigned int rd1;

   WDTCN = 0xde;                            // disable watchdog timer
   WDTCN = 0xad;

   SYSCLK_Init ();                          // initialize oscillator
   PORT_Init ();                            // initialize crossbar and GPIO
   UART0_Init ();                           // initialize UART0

   EA = 1;                                  // Enable global interrupts

    while (1)
    {
        if(button_dn())                     // check for key press
        {
            delay_ms(5);                    // delay for debouncing
            rd1 = scankey();                // read keypad
            if(rd1 != 0)
            {
                putchar (254);              // LCD command
                putchar (0x01);             // clear LCD
                printf (" You pressed:\r   " );
                putchar (rd1);
            }
            while(button_dn());             // check for key release
        }
        delay_ms(5);
    }
}

//-----------------------------------------------------------------------------
// Initialization Subroutines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// This routine initializes the system clock to use an 18.432MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                              // delay counter

    OSCXCN = 0x67;                      // start external oscillator with
                                        // 18.432MHz crystal

    for (i=0; i < 256; i++) ;           // XTLVLD blanking interval (>1ms)

    while (!(OSCXCN & 0x80)) ;          // Wait for crystal osc. to settle

    OSCICN = 0x88;                      // select external oscillator as SYSCLK
                                        // source and enable missing clock
                                        // detector
}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0    = 0x04;                 // Enable UART0
    XBR1    = 0x00;
    XBR2    = 0x40;                 // Enable crossbar and weak pull-ups
    P0MDOUT |= 0x01;                // enable TX0 as a push-pull output

   // PORT   3 CONFIGURATION
    P2MDOUT = 0xF0;                 // P2 u.n. push pull, lower-nibble input
    P2 = 0x0F;                      // upper nibble hi-imp, allowing input read

}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
```

```
//
void UART0_Init (void)
{
   SCON0   = 0x50;                     // SCON0: mode 1, 8-bit UART, enable RX
   TMOD    = 0x20;                     // TMOD: timer 1, mode 2, 8-bit reload
   TH1    = -(SYSCLK/BAUDRATE/16);     // set Timer1 reload value for baudrate
   TR1    = 1;                         // start Timer1
   CKCON |= 0x10;                      // Timer1 uses SYSCLK as time base
   PCON  |= 0x80;                      // SMOD00 = 1
   TI0    = 1;                         // Indicate TX0 ready
}

//-----------------------------------------------------------------------------
// Local Functions
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// button_dn
//-----------------------------------------------------------------------------
//
// Function: test keypad for the presence of a key press.
// Return:   1 if keypress; 0 otherwise.

int button_dn()
{
    int tmp;
    tmp = (P2 & 0x0F)^0x0F;             // read P2.3->P2.0 and XOR output

    if(tmp)                            // if button is depressed, tmp != 0
        return 1;
    else
        return 0;
}

//-----------------------------------------------------------------------------
// scankey
//-----------------------------------------------------------------------------
//
// Function: read keypad and convert keypress into equiv. ASCII code.
// Return: ASCII equivalent of pressed key's label.

unsigned int scankey(void)
{
    int row = 0;
    int col = 0;
    int k,j;

    P2 = 0x0F;                         // set data register
    P2MDOUT = 0xF0;                    // drive P2.3->P2.0 as output
    delay_ms(10);                      // let drive signals settle

    row = (P2 & 0x0F)^0x0F;            // read P2.3->P2.0 and XOR output

    delay_ms(2);

    if(row == 0)
        return 0;                      // no closure detected

    P2 = 0xF0;                         // set data register
    P2MDOUT = 0x0F;                    // drive P2.7->P2.4 as output
    delay_ms(2);                       // let drive signals settle

    col = (P2 & 0xF0)^0xF0;            // P2.7->P2.4 and XOR output
    col = col >> 4;                    // move hi nibble to lo nibble

    if(col == 0)
        return 0;                      // no closure detected
    P2 = 0x0F;                          // set data register
    P2MDOUT = 0xF0;                     // drive P2.3->P2.0 as output
    delay_ms(2);                        // let drive signals settle

    switch(row)                         // convert 1-of-4 to binary
        {
          case 1:   j = 0; break;
          case 2:   j = 1; break;
          case 4:   j = 2; break;
          case 8:   j = 3; break;
          default:  return 0;
        }

    switch(col)                         // convert 1-of-4 to binary
        {
          case 1:   k = 0; break;
          case 2:   k = 1; break;
          case 4:   k = 2; break;
          case 8:   k = 3; break;
          default:  return 0;
        }

    return keytab[j][k];    // return the ASCII value at that row and column
}

//-----------------------------------------------------------------------------
// delay_ms
//-----------------------------------------------------------------------------
//
// an approximate x ms delay.

void delay_ms(int ms)
{
    int y;
    int z;
    for (y=1; y<=250; y++) for (z=1; z<= ms; z++);
}
```