APPLICATION NOTE

LCD6402B and PIC16F876A Interface in C

1 INTRODUCTION

This application note presents an example interface between an LCD6402B-P intelligent graphic LCD and a Microchip PIC16F876A. microcontroller.

1.1 HARDWARE INTERFACE

The application circuit shown in APPENDIX A consists of an LCD6402B-P intelligent graphic LCD and 5V powered PIC16F876A microcontroller. The LCD6402B has a logic supply voltage of 3.3V so care is required when interfacing with 5V powered devices. This is achieved using the zener diode and resistor network formed by R1-4 and D1-2. Graphics can be programmed to the LCD6402B via the 'RS232 PROG PORT' using LCDLAB. Graphic objects (0-127) are recalled by writing the appropriate value to the LCD6402B command register.

1.2 SOFTWARE INTERFACE

The application code in this example is written in 'C' and compiled using a CCS PCM C-Compiler [1]. See APPENDIX B for source code listing.

The LCD6402B is communicated to via the I2C bus and the functions for this are show in the code headed 'LCD64 INTERFACE FUNCTIONS '

In general if a graphic function is used more than once then it's usually worth writing a wrapper function (see code headed 'LCD WRAPPER FUNCTIONS') to both save code and provide clarity. Examples of this are the functions LCD_Font() & LCD_Cursor(). More wrapper functions can be creates as required. For example the rectangle graphic function can be wrapped thus:

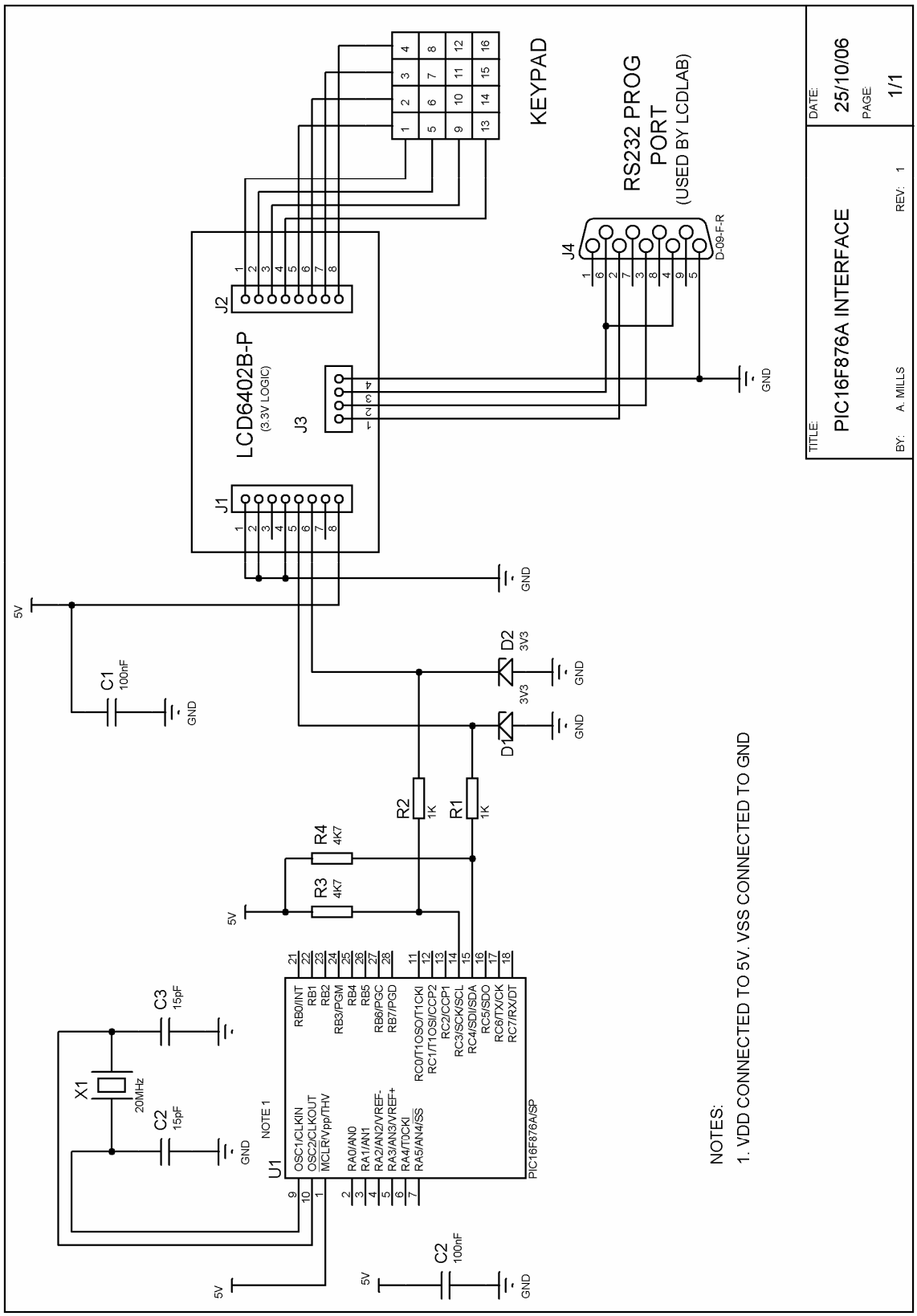
```
void LCD_Rect(int x0,y0,x1,y1)
{
    LCD64_BusyWait();
    LCD64_Write(rINS, gRECT);
    LCD64_Write(rINS, x0);
    LCD64_Write(rINS, y0);
    LCD64_Write(rINS, x1);
    LCD64_Write(rINS, y1);
}
```

When power is applied, the backlight is switched on and the displayed cleared. The display is then rendered with a rectangle and some introductory text and example graphics.

In the main program loop key presses are detected by reading the NEWKEY register. When the user presses a key its scan code is printed on the LCD.

This code could, with minimal modification, be made to work with other microcontrollers and compilers.

APPENDIX A. CIRCUIT SCHEMATIC



APPENDIX B. APPLICATION CODE

```
////////////////////////////////////////////////////////////////////////
////                          AN1001.C                              ////
////                                                                ////
////  See application note AN1001                                   ////
////                                                                ////
////  Compiled using CCS PCM 'C-compiler'                           ////
////////////////////////////////////////////////////////////////////////
#include <16f876a.H>

#fuses hs,nowdt,noprotect,put,nowrt,nolvp,nodebug,nobrownout
#case
#use delay(clock=20000000)


//-------------------------------------------------------//
//                                                       //
//          LCD64 INTERFACE FUNCTIONS                    //
//                                                       //
//-------------------------------------------------------//

#use i2c(sda=PIN_C4, scl=PIN_C3, FORCE_HW)

// LCD64 register defines
#define rCMD                    0x20
#define rSTATUS                 0x21
#define rNEWKEY                 0x22
#define rKEY                    0x23
#define rCUX                    0x24
#define rCUY                    0x25
#define rORX                    0x26
#define rORY                    0x27
#define rNVML                   0x28
#define rNVMH                   0x29
#define rOPTION                 0x2A
#define rINS                    0x2B
#define rCONTRAST               0x2C
#define rFGCLR                  0x2D
#define rBGCLR                  0x2E
#define rMODE                   0x2F
#define rFONT                   0x30

// LCD64 rCMD register defines
#define cmdRESET_INTERPRETER    0x91
#define cmdBACKLIGHT_ON         0x92
#define cmdBACKLIGHT_OFF        0x93
#define cmdCLEAR_SCREEN         0x94
#define cmdSAVE_CONTEXT         0xa8
#define cmdRESTOR_CONTEXT       0xa9
#define cmdRESET                0xff

// LCD64 rINS register defines          // Expected parameters
#define gRESET                  0x08    //
#define gCLS                    0x21    //
#define gORIGIN                 0x2A    // x, y
#define gCURSOR                 0x3A    // x, y
#define gFGCLR                  0x41    // clr
```



```
#define gBGCLR                  0x42    // clr
#define gMODE                   0x43    // mode
#define gPIX                    0x4A    // x, y, n
#define gLINETO                 0x52    // x, y
#define gRECT                   0x5C    // x0, y0, x1, y1
#define gRECTF                  0x64    // x0, y0, x1, y1
#define gFONT                   0x69    // f
#define gPUTC                   0x71    // c
#define gHRAST                  0x79    // rast
#define gPUTS                   0x80    // c0..cn, 0
#define gHBMP                   0x8A    // w/8, H, rdata,
#define gCIRCLE                 0x99    // r
#define gCIRCLEF                0x9A    // r
#define gVRAST                  0xA1    // r
#define gVBMP                   0xAA    // w, h/8, rdata,
#define gCALLI                  0xF1    // n

//
// Read and return register r from LCD64.
//
int LCD64_Read(int r) {
    int result;

    i2c_start();
    i2c_write(0xa0);
    i2c_write(r);
    i2c_start();
    i2c_write(0xa0 | 1);
    result=i2c_read(0);
    i2c_stop();
    return(result);
}

//
// Write data to LCD64 register r.
//
void LCD64_Write(int r,int data) {
    i2c_start();
    i2c_write(0xa0);
    i2c_write(r);
    i2c_write(data);
    i2c_stop();
}

//
// Wait for LCD64 busy flag to clear.
//
void LCD64_BusyWait( void) {
    int status;
    do {
        delay_ms(1);
        status = LCD64_Read(0x21);
        status &= 0x80;
    } while ( status != 0);
}

//
// Write command cmd to LCD64 command register.
//
```



```
void LCD64_CMD( int cmd) {
    LCD64_BusyWait();
    LCD64_Write( rCMD, cmd);
}


//-------------------------------------------------------//
//                                                       //
//          LCD WRAPPER FUNCTIONS                        //
//                                                       //
//-------------------------------------------------------//

//
// Print character c on LCD.
//
void LCD_Putc(int c ) {
    LCD64_BusyWait();
    LCD64_Write(rINS, gPUTC);
    LCD64_Write(rINS, c);
}

//
// Move LCD cursor to coordinate x, y.
//
void LCD_Cursor(int x, y ) {
    LCD64_BusyWait();
    LCD64_Write(rINS, gCURSOR);
    LCD64_Write(rINS, x);
    LCD64_Write(rINS, y);
}

//
// Set LCD font to n.
//
void LCD_Font(int n)
{
    LCD64_BusyWait();
    LCD64_Write(rINS, gFONT);
    LCD64_Write(rINS, n);
}

//
// Draw rectangle with top left corner at coordinate x0, y0 and
// bottom right at coordinate x1, y1. If fill TRUE draw a filled
// rectangle otherwise draw it unfilled.
//
void LCD_Rect(int x0, y0,x1, y1, fill)
{
    LCD64_BusyWait();
    if (fill)
        LCD64_Write(rINS, gRECTF);
    else
        LCD64_Write(rINS, gRECT);
    LCD64_Write(rINS, x0);
    LCD64_Write(rINS, y0);
    LCD64_Write(rINS, x1);
    LCD64_Write(rINS, y1);
}
```



```
//
// Draw circle with radius r. If fill is TRUE draw a filled circle
// otherwise draw a unfilled circle.
//
void LCD_Circle(int r, int fill)
{
    LCD64_BusyWait();
    if (fill)
        LCD64_Write(rINS, gCIRCLEF);
    else
        LCD64_Write(rINS, gCIRCLE);

    LCD64_Write(rINS, r);
}

//
// Set the drawing mode to mask.
//
void LCD_Mode(int mask)
{
    LCD64_BusyWait();
    LCD64_Write(rMODE, mask);
}

//-------------------------------------------------------//
//                                                       //
//          APPLICATION FUNCTIONS                        //
//                                                       //
//-------------------------------------------------------//

//
// Program entry.
//
void main() {
    int key, i, x, y;


    // Demonstrate some graphic functions.
    // -----------------------------------

    // Swtich backlight on.
    LCD64_CMD(cmdBACKLIGHT_ON);

    // Clear screen.
    LCD64_CMD(cmdCLEAR_SCREEN);

    // Draw filled rectangle.
    LCD_Rect(32, 16, 110, 32, TRUE);

    // Set the drawing mode to exclusive or (XOR) foreground color,
    // turn off back ground color. (This mode will be used to
    // print negative text below).
    LCD_Mode(0x21);

    // Print "Hello World" using negative text at coordinate 36, 20
    // using font type 1.
    LCD_Cursor(36, 20);
    LCD_Font(1);
    printf(LCD_Putc, "Hello World");
```



```
// Revert to the default drawing mode.
LCD_Mode(0);

// Pause for 3 seconds and then clear screen.
delay_ms(3000);
LCD64_CMD(cmdCLEAR_SCREEN);

// Draw some circles from the top left to bottom right of
// display.
y = 8;
x = 8;
while (x < 120) {
    LCD_Cursor(x, y);
    LCD_Circle(8, 1);
    x += 16;
    y += 8;
}

// Pause for 3 seconds and then clear screen.
delay_ms(3000);
LCD64_CMD(cmdCLEAR_SCREEN);

// Draw unfilled rectangle with top left corner at coordinate
// 0, 0 and bottom right corner at coordinate 127, 63.
LCD_Rect(0, 0, 127, 63, FALSE);

// Print "Welcome to the" using font type 1 at coordinate 16, 4.
LCD_Cursor(16, 4);
LCD_Font(1);
printf(LCD_Putc, "Welcome to the");

// Print "LCD6402B"" using font type 2 at coordinate 30, 16.
LCD_Font(2);
LCD_Cursor(30, 16);
printf(LCD_Putc, "LCD6402B");

// Pause for 3 seconds and then clear screen.
delay_ms(3000);
LCD64_CMD(cmdCLEAR_SCREEN);

// Recall graphic object #0 from the LCD64's on board
// non-volatile memory. If no graphics are stored the
// display will appear blank. See LCDLAB on how to
// programm the LCD64 with graphics.
LCD64_CMD(0);


// Pause for 3 seconds and then clear screen.
delay_ms(3000);
LCD64_CMD(cmdCLEAR_SCREEN);


// Demonstrate the keypad interface indefinitely.
// -----------------------------------------------

// Clear screen and print message at coordinate 4, 4 using font
// type 1.
LCD64_CMD(cmdCLEAR_SCREEN);
```



```
    LCD_Cursor(4, 4);
    LCD_Font(1);
    printf(LCD_Putc, "Press a key...");

    // Loop forever.
    while(1) {

        // Get key press from LCD64
        key = LCD64_Read(rNEWKEY);

        // If key pressed print scan number on display.
        if (key) {
            // Print message at coordinate 9, 42.
            LCD_Cursor(8, 32);
            printf(LCD_Putc, "You pressed key #%2d", key);
        }
    }
}
```

