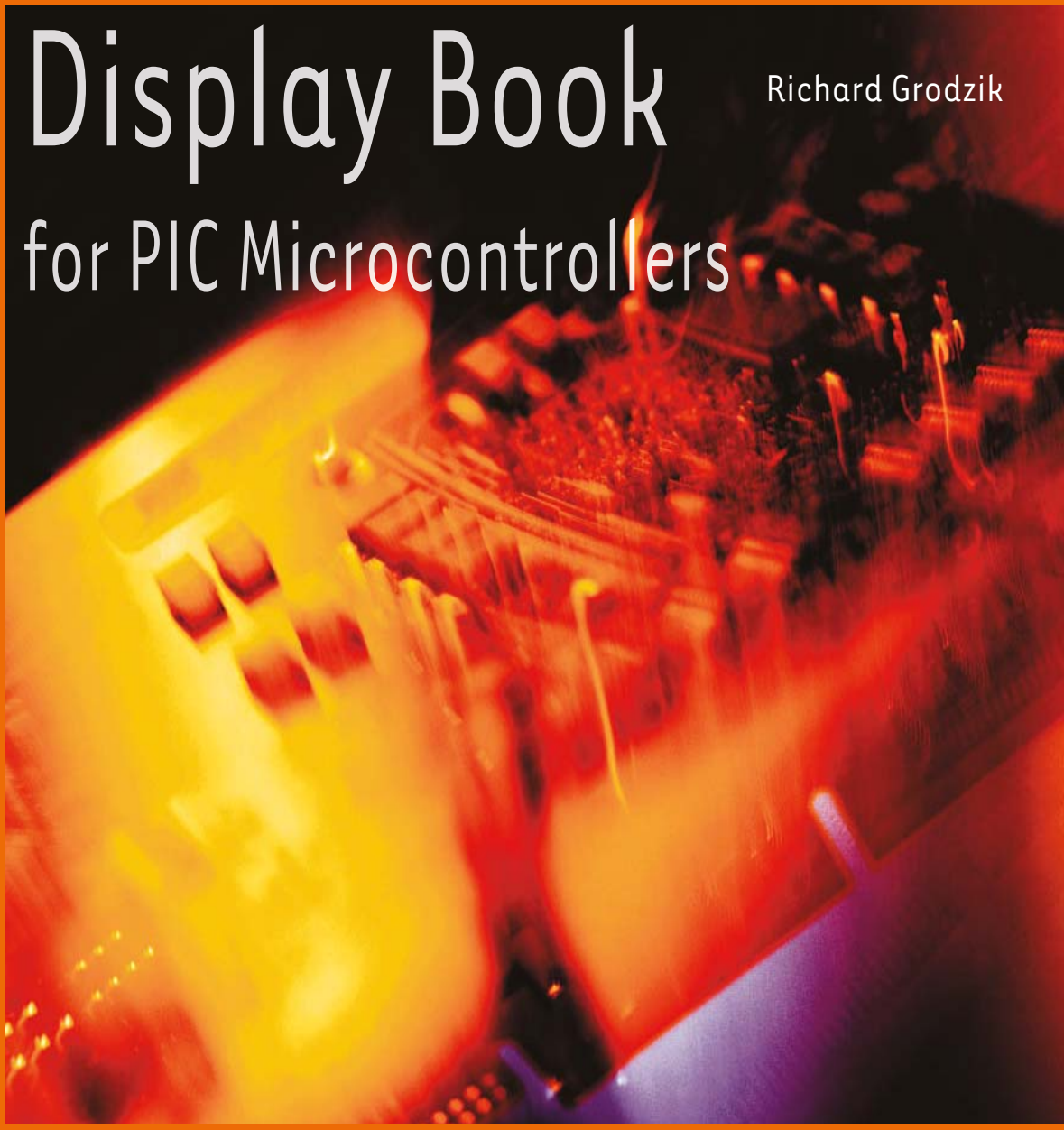
Universal Display Book for PIC Microcontrollers

Richard Grodzik



elektor

R. Grodzik

Universal Display Book

for PIC Microcontrollers

R. Grodzik

Universal Display Book

for PIC Microcontrollers

Elektor International Media BV
Postbus 11
6114 ZG Susteren
The Netherlands

Acknowledgement

I would like to dedicate this book to my mother – Walentyna, without whose support, kindness and objectivity, this book would never have been possible.

Richard Grodzik, september 2007.

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN 978-0-905705-73-6
NUR 980

Prepress production: Autronic, Blaricum
Design cover: Helfrich Ontwerpbureau, Deventer
First published in the United Kingdom 2008
Printed in the Netherlands by Wilco, Amersfoort

089007/UK

Preface

This book is a practical introduction to using and interfacing many types of electronic displays to Arizona Microchip's range of embedded microcontrollers, commonly know as 'PIC chips'. From the simple LED to colour graphic displays, the reader is shown the hardware interface requirements and the software programming both in Assembler and/or MPLAB C18 C compiler to achieve a functioning display. In addition, a small introductory tutorial for using the freely available 'EAGLE' PCB/Schematic CAD tool is included.

The PIC microcontrollers covered in this book include the PIC12C508, PIC12F629/675, PIC16F84, PIC16F876, PIC18F252, PIC18F452 and the PIC18F4550.

To utilise the various displays, many complete case studies, from a simple egg timer using a single 7-segment LED display to an electronic compass with colour graphic LCD display are included in this book.

I hope that the reader enjoys constructing some of these projects, since complete schematic drawings are included, including the source code and Hex dump for the various PIC microcontrollers.

In addition, all the source code examples in the book may be downloaded from the elector.com website and the PDF data-sheet files from the relevant manufacturer's web sites for all the case studies.

Table of Contents

1 Light emitting diodes

The LED is a simple indicator available in a variety of different shapes, colours and levels of light intensity. It can be made to stay permanently on, flash on and off at different frequencies, and vary its light output. To achieve this, an embedded microcontroller – the PIC chip – is used, whereby a program can easily change the functionality of the LED: for use as status (ON/OFF) and alarm conditions and, because it is available in a large range of colours, it can differentiate between the status of many signal channels. Also it can be used to indicate an analogue quantity either by varying its brightness or by altering the rate of flashing. LEDs are used in many portable applications because of their low current consumption and so the examples in this chapter concentrate on low power battery usage.

In this chapter, an overview of the LED is given, together with its history and characteristics. To enable the reader to design, construct and program the circuit, a simple 'walk-through' using a schematic and design package is included. In addition simple steps in using the 'MPLAB' programming environment to program the PIC are included. Finally several projects are included to demonstrate the use of LEDs.

1.1 History of the light emitting diode

Red, Green and Blue LEDs

A light-emitting diode (LED) is a semiconductor device that emits incoherent narrow-spectrum light when electrically biased in the forward direction of the p-n junction. This effect is a form of *electroluminescence*.

The phenomenon of *electroluminescence* was first observed in a piece of Silicon Carbide (SiC) in 1907 by Henry Joseph Round. The yellow light emitted by it was too dim to be of practical use, and difficulties in working with Silicon Carbide meant that research was abandoned. Further experiments were carried out in Germany in the late 1920s by Bernhard Gudden and Robert Wichard Pohl, using phosphor materials made from Zinc Sulphide doped with Copper (ZnS:Cu), although once again, the low level of light produced meant that no in-depth research was carried out. In 1936 George Destriau published a report on the emission of light by Zinc Sulphide (ZnS) powders following the application of an electric current and is widely credited with having invented the term 'electroluminescence'.

The first visible (red) light LEDs were produced in the late 1960s, using Gallium Arsenide Phosphide (GaAsP) on a GaAs substrate. Changing to a Gallium Phosphide (GaP) substrate led to an increase in efficiency, making for brighter red LEDs and allowing the colour orange to be produced.

By the mid 1970s Gallium Phosphide (GaP) was itself being used as the light emitter and was soon producing a pale green light. LEDs using dual GaP chips (one in red and one in green) were able to emit yellow light. Yellow LEDs were also made in Russia using Silicon Carbide at around this time, although they were very inefficient compared to their Western counterparts, which were producing purer green light by the end of the decade. The use of Gallium Aluminium Arsenide Phosphide (GaAlAsP) LEDs in the early to mid 1980s brought the first generation of super-bright LEDs, first in red, then yellow and finally green. By the early 1990s ultra-bright LEDs using Indium Gallium Aluminium Phosphide (InGaAlP) to produce orange-red, orange, yellow and green light had become available.

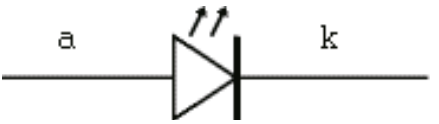
The first significant blue LEDs also appeared at the start of the 1990s, once again using Silicon Carbide – a throwback to the earliest semiconductor light sources, although like their yellow russian ancestors the light output was very dim by today's standards. Ultra-bright blue Gallium Nitride (GaN) LEDs arrived in the mid-1990s, with Indium Gallium Nitride (InGaN) LEDs producing high-intensity green and blue shortly thereafter.

The ultra-bright blue chips became the basis of white LEDs, in which the light emitting chip is coated with fluorescent phosphors. These phosphors absorb the blue light from the chip and then re-emit it as white light. This same technique has been used to produce virtually any colour of visible light and today there are LEDs on the market that can produce previously 'exotic' colours, such as aqua and pink.

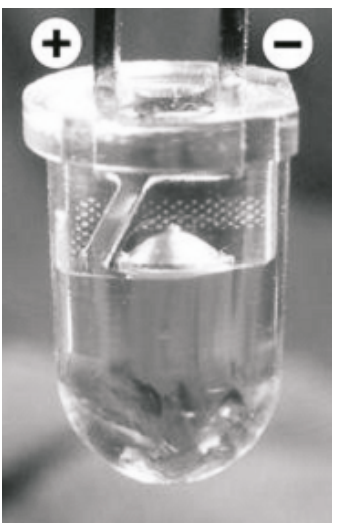
1.2 LED characteristics and parameters

LEDs are available in standard sizes of 3 mm and 5 mm as well as various shaped packages. They all have two terminals (cathode and anode) and are also available in surface

mount packages. The parameters of an LED include its colour, forward current (If) forward voltage (Vf), viewing angle and luminous intensity (mcd).

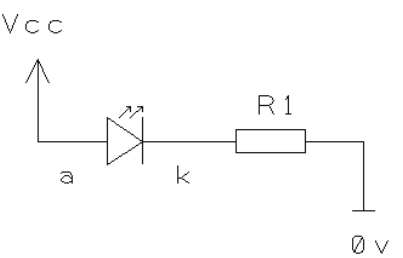


Circuit symbol



Close-up of a standard 5-mm LED

LEDs must be powered by a D.C. voltage source and connected the correct way round: the cathode terminal to –ve and the anode terminal to +ve of the supply. The diagram may be labelled **a** or + for anode and k or – for cathode. The cathode **k** is the short lead and there may be a slight flat on the body of round LEDs to indicate this lead.



Limiting the LED current.

Using a limiting resistor for an LED

A standard LED either 3 mm or 5 mm in diameter requires a potential difference of between 1.6 and 3.5 volts across it, dependent on its colour, and typically consumes 20 milliamps. For standard TTL and microprocessor-based circuits with a nominal 5-volt power supply, a limiting resistor is usually required to drop any excess voltage. Exceeding the rated current will cause the LED to burn out.

For example: a Vcc (supply voltage) of 5 volt with a Vf (forward voltage) of 2 volts for the LED and current of 20 mA:

Excess voltage to drop across the resistor is (5 V – 2 V) = 3 volts.

Value of dropping resistor using ohms law (R=V/I) = $3/(20 \times 10^{-3}$ ohms) = 150 ohms.

This calculation can be used as a general guide and the actual resistor value for required brightness could be found by experimentation.

Generally, for common standard LEDs in 3 mm or 5 mm packages, the following forward DC potential differences are typically measured. The forward potential difference depending on the LEDs chemistry, temperature, and on the current (values here are for approx. 20 milliamperes, a commonly found maximum value).

The following table gives typical PD values for different LED colours:

Colour	**Potential Difference**
Infrared	1.6 V
Red	1.8 V to 2.1 V
Orange	2.2 V
Yellow	2.4 V
Green	2.6 V
Blue	3.0 V to 3.5 V
White	3.0 V to 3.5 V
Ultraviolet	3.5 V

Different types of LEDs.

Reading a table of technical data for LEDs

Manufacturers' data sheets usually include tables of technical data for components such as LEDs. These tables contain a good deal of useful information in a compact form. The table below shows typical technical data for some 5 mm diameter round LEDs with diffused packages (plastic bodies).

Type	**Colour**	**I_F max.**	**V_F typ.**	**V_F max.**	**V_R max.**	**Luminous intensity**	**Viewing angle**	**Wavelength**
Standard	Red	30mA	1.7V	2.1V	5V	5mcd @ 10mA	60°	660nm
Standard	Bright red	30mA	2.0V	2.5V	5V	80mcd @ 10mA	60°	625nm
Standard	Yellow	30mA	2.1V	2.5V	5V	32mcd @ 10mA	60°	590nm
Standard	Green	25mA	2.2V	2.5V	5V	32mcd @ 10mA	60°	565nm
High intensity	Blue	30mA	4.5V	5.5V	5V	60mcd @ 20mA	50°	430nm
Super bright	Red	30mA	1.85V	2.5V	5V	500mcd @ 20mA	60°	660nm
Low current	Red	30mA	1.7V	2.0V	5V	5mcd @ 2mA	60°	625nm

Where:

I_F max.	Maximum forward current.
V_F typ.	Typical forward voltage.
V_F max.	Maximum forward voltage.
Luminous intensity	Brightness of the LED at the given current, mcd = millicandela.
Viewing angle	Standard LEDs have a viewing angle of 60°, others emit a narrower beam of about 30°.
Wavelength	The peak wavelength of the light emitted, this determines the colour of the LED.
Colour	Standard colours are red,blue and green

Although the standard operating voltage for most microprocessors is 5 volts, the PIC can operate at any voltage from 2V0 to 5V5 or 6V0 depending on the type of PIC. Since the PIC chip is ideal for portable battery applications, the power supply is usually in the form of batteries. Today a lithium battery has a standard e.m.f. of 3V6 and with its high ampere-hour capacity is ideal for powering the PIC – alternately 2 standard AA or AAA size alkaline batteries in series will provide a 3V0 or 4V5 supply. Three rechargeable metal hydride batteries will also provide sufficient power for a PIC (3 × 1.2V). A good alternative is the small 'button' type primary (non-rechargeable) CR2032 lithium coin cell in a suitable holder. Do not attempt to solder this type of battery since an explosion and fire is likely to occur.

In most cases, connecting the LED to a PIC powered from these low voltages does not require a limiting resistor, as the PIC has a maximum supply current of approximately 20/25 milliamps per output port pin. However in most portable applications a limiting resistor, typically 1 kilo-ohm, is used to dramatically limit the current consumption. Depending on the type of LED, this resistor value can be decreased/increased to provide the correct level of illumination. Alternately, to decrease power consumption, the LED is simply switched on for a brief period of time every second or so.

1.3 PIC interface for LED circuits – design and programming

The PIC embedded microcontroller

To-date, well in excess of 3 billion PICs have been sold world-wide, so it will come as not too great a surprise that it is the default choice of many when designing an embedded microcontroller-based product. The author was first introduced to the PIC when it was still in its infancy and was mainly an OTP (one time programmable) device with a supporting UV erasable JW type for development purposes. Typical earlier devices were the PIC12C508/9 and the PIC16C54/55.
And so the transition began from conventional microprocessor + external EPROM designs to a single PIC chip containing a microprocessor, memory and I/O peripherals, thus enabling single chip board solutions. The PIC chip has evolved into a miniature 'single chip computer' with Flash program memory of up to 256 Kbytes, 16 Kbytes of RAM for data variables, speeds of up to 40 Mips and current consumption of just 40 nA in 'sleep-mode' as well as a supply voltage down to 2.0 volts.

The sustainability of the PIC lies in the fact that, like the original INTEL 8088 microprocessor, its original hardware core and its instruction set has been preserved so that programs written many years ago still run on today's more powerful chips which have additional instructions and additional on-board hardware. Today's PICs have on-chip communications, control/timing and analogue peripherals including:

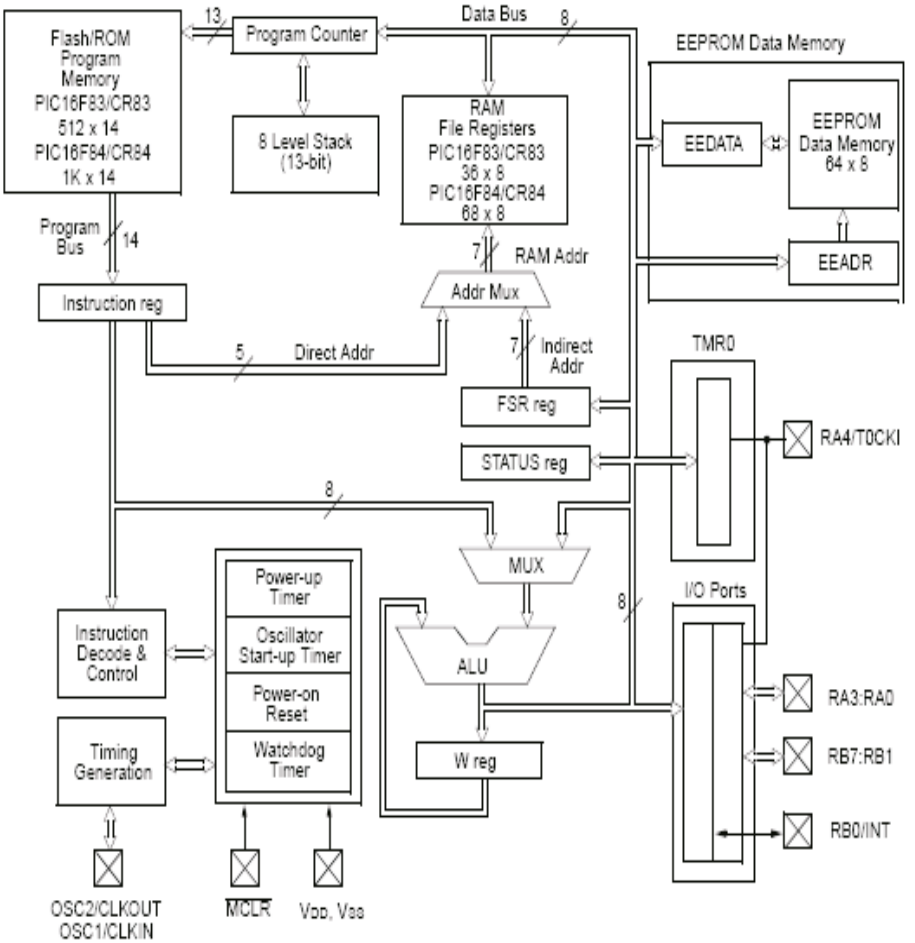
- RS232/RS485, SPI, IIC, USB, TCPIP, CAN, LIN, Radiofrequency, Capture/Compare, PWM, Counter/Timers, Watchdog timer, ADC converters, comparators/Opamps, and temperature sensors.

There are now many hundreds of different PICs available arranged in 5 families: The PIC10, PIC12, PIC16, PIC18 and PIC24 – varying in pin count from 6 (PIC10) to 100 (PIC24). The choice of a suitable PIC may be daunting, but each PIC microcontroller family is compatible within a given pin count, so that each pin on the PIC offers the same function. For example, on a 40 pin PIC, pin 1 is always the /MCRL-Vpp pin. Many of the pins on the larger PICs have 2, 3 or 4 functions available that are selectable by software. In addition, the data sheet of each PIC in a family is identical, with fewer or additional hardware peripherals dependent on the PIC used.

PIC Architecture

The typical PIC contains all the essential ingredients of a microprocessor, including a working register (accumulator 'W'), status (flag) register, instruction decoder, an ALU (arithmetic/Logic Unit), RAM registers, clock circuit, program counter (Instruction pointer), stack register and in addition programmable I/O ports (Ports A and B), an area of EEPROM and Flash program memory.

In this book, PICs from three different families have been chosen for the detailed case studies: These are the 12C508, 12F629/675, 16F84, 16F876, 18F252, 18F452 and the 18F4550. In the appendix you will find comparison charts between the various PICs (reproduced with the kind permission of Microchip).



PIC Architecture.

TABLE 3-1 PIC16F8X PINOUT DESCRIPTION

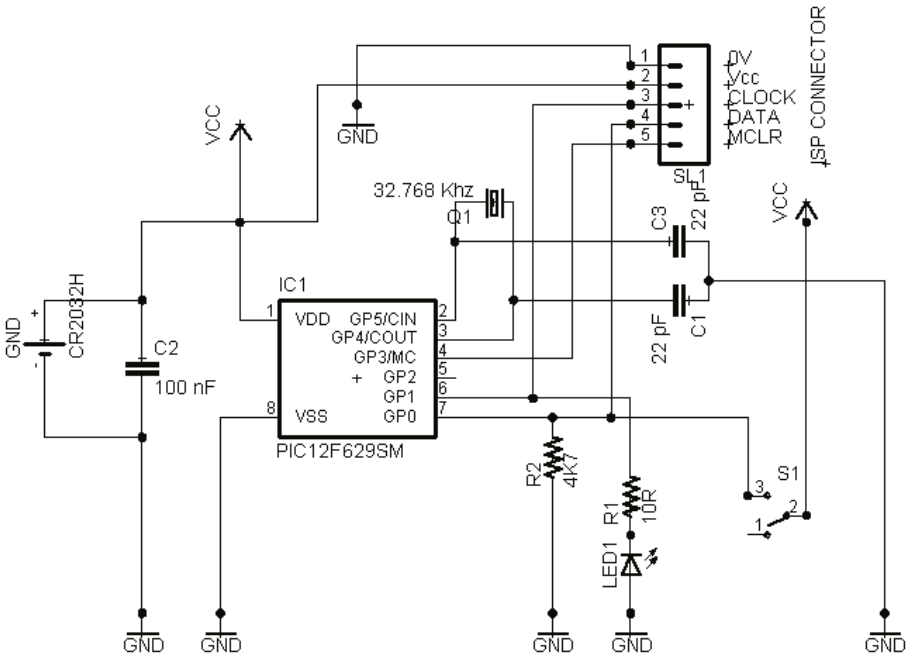
Pin Name	DIP No.	SOIC No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	I	ST/CMOS (3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
					PORTA is a bi-directional I/O port.
RA0	17	17	I/O	TTL	
RA1	18	18	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RA4/T0CKI	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/ counter. Output is open drain type.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	I/O	TTL/ST (1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	I/O	TTL	
RB2	8	8	I/O	TTL	
RB3	9	9	I/O	TTL	
RB4	10	10	I/O	TTL	Interrupt on change pin.
RB5	11	11	I/O	TTL	Interrupt on change pin.
RB6	12	12	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming clock.
RB7	13	13	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming data.
VSS	5	5	P	—	Ground reference for logic and I/O pins.
VDD	14	14	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = output I/O = Input/Output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

PIC interface for LED circuits

In this section an example of connecting a PIC to an LED is shown and the various stages in designing the circuit and board using the 'EAGLE' CAD package are demonstrated. A freeware version of 'EAGLE' can be downloaded from: http://www.cadsoft.de/. Also, the use of the 'MPLAB' programming environment is shown in constructing the firmware required for the PIC

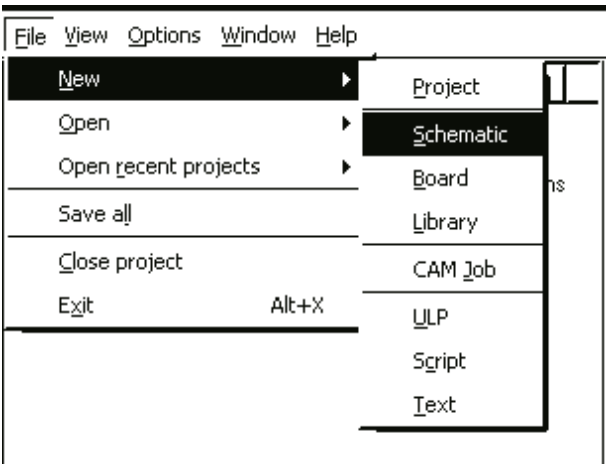


PROGRAMS LED_1.ASM, LED_2.ASM

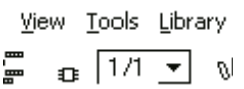
The circuit shown above was created using the 'EAGLE' PCB and schematic design package. It uses a PIC12F629 surface mount PIC although the 12F675 with a 4 channel ADC also may be used because the software examples for the circuit configure the PIC for digital I/O. A conventional LED may be substituted for the small surface mount LED, although the limiting resistor R1 may have to be changed for an appropriate value. A 32.768 kHz crystal provides the circuit with microampere current consumption, the power being provided by a single 3-volt lithium button cell. As the PIC is surface mounted, it cannot be removed easily from the PCB and therefore an ISP (in circuit programming) connector is used to connect to the PIC programmer.

What does this circuit do? Well, it will turn an LED on and off. But with the emphasis on very low power consumption and timing accuracy, various software techniques are used as demonstrated in the three following software examples. Before we discuss the software, an outline of how to actually construct the circuit will be given. A logic switch is included so that the reader may include this in the software to control the LED.

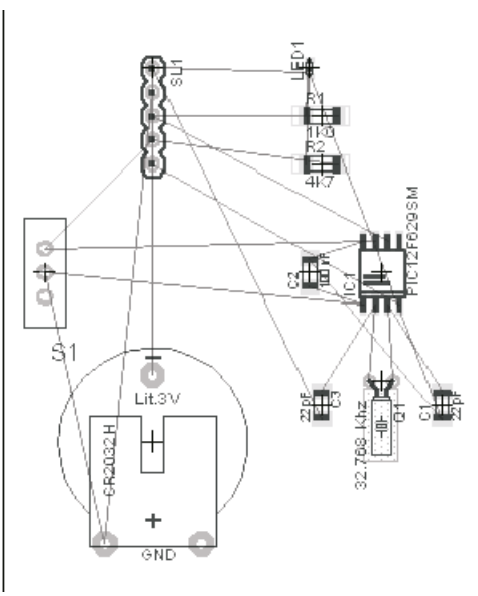
In order to design the schematic diagram, the 'EAGLE' schematic design is used. A rudimentary outline of using this software is provided, since this design package is very easy to use. Firstly, the schematic design environment is invoked by selecting in the menu bar File – New – Schematic – as below:



Next, the circuit is designed using components located in the schematic library. Once the circuit design is completed, the 'board' icon in the menu bar (the 2 circuit symbols found to the left of 1/1) is selected. A new screen appears showing the actual physical components and the yellow connecting wires ('rats nest').

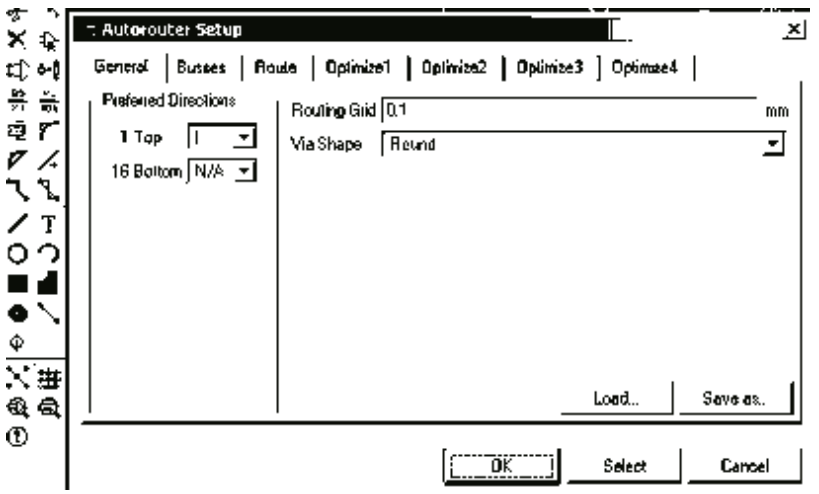


The various components are now positioned strategically inside the blank PCB to ensure that the auto-router has an easy job in routing the connections.

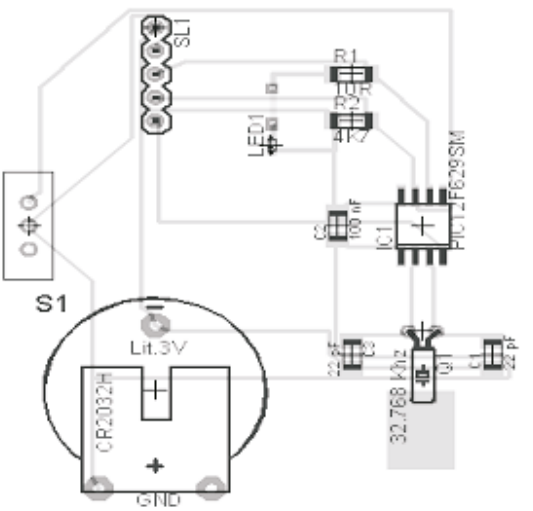


The 'rats-nest'

The board needs to have all the connections to the components automatically routed. Select the AutoRoute icon on the LH menu bar to do this.
Since this is a surface mount design, only 1 layer (the top-most) layer is used.

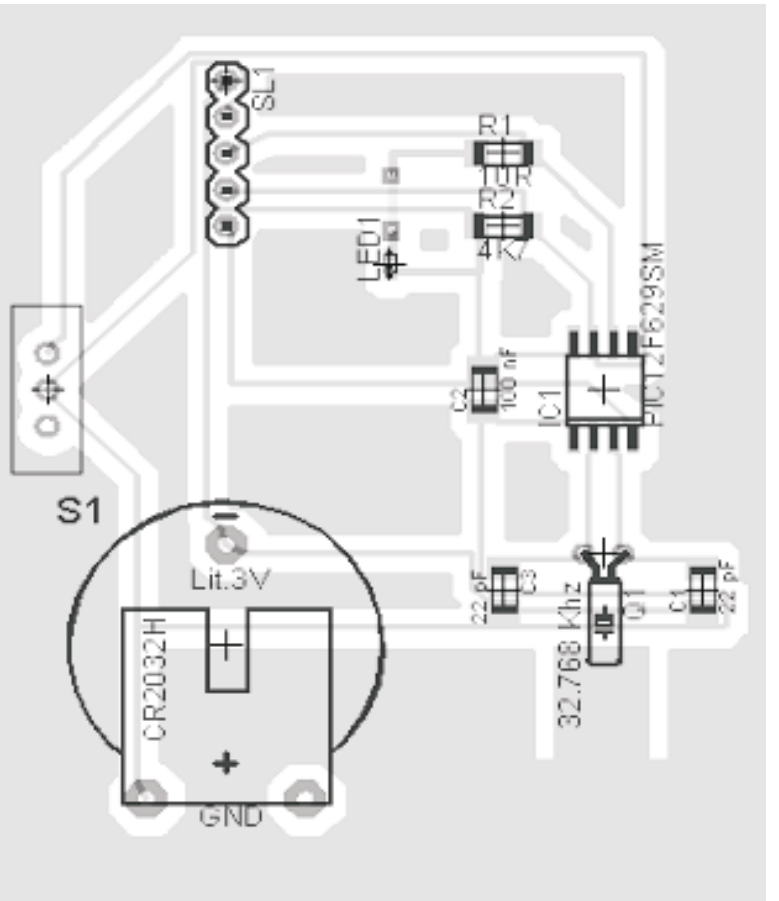


The routed board is shown below:



Now, to include a copper plane, the polygon symbol is selected and the isolation and spacing variables, typically 1.27, are selected on the top menu bar. These parameters need to be experimented with to optimise the final board design.

The PCB can then be manufactured in-house or by a **thirdparty**. The reader also has the option of using discreet components, and since this is a relatively simple circuit, it can also be constructed on copper strip board.



The completed PCB with ground plane.

The next stage is to program the PIC using the 'MPLAB' programming environment. Our first program lights the LED for 1 ms every 1.000 second. This accuracy is obtained by using the standard crystal frequency found in many digital watches, of 32768 Hz.

This program uses timer 1 to generate an interrupt every 1 second. When this occurs, the program vectors to address 0004H and the LED is lit. The timer 1 high and low registers are re-loaded with the correct value to achieve a delay of exactly 1 second. Returning from the interrupt service routine, the LED is extinguished, and the program loops, waiting for the next interrupt issued by the timer when it overflows. i.e. increments from FFFFH to 0000H. ad infinitum.

Program LED_1 .ASM

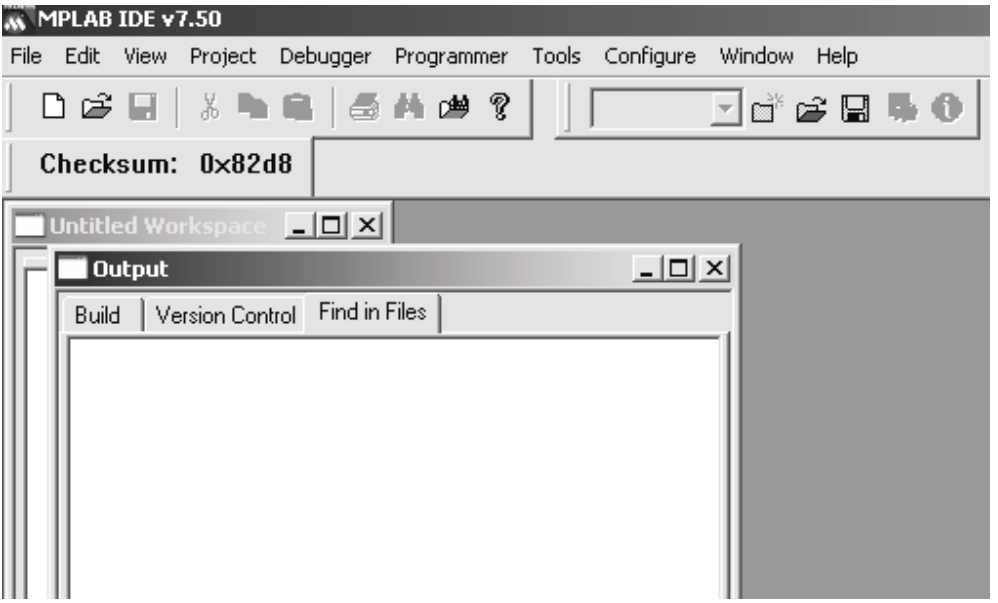
```
    List p=12F629; list directive to define processor
    #Include <p12f629.inc; processor specific variable definitions

    __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_LP_OSC &  _MCLRE_ON & _CPD_OFF

; Variables

RTCC EQU 1
PC EQU 2
STATUS EQU 3

GPIO EQU 5
CMCON EQU 019H
ANSEL EQU 09FH

    ORG 0x00; reset vector
    GOTO START

    ORG 0x04; interrupt vector
    BSF GPIO, 1                 ; switch on LED FOR 1 mS
    BCF PIR1, 0                 ;CLEAR INTERRUPT FLAG

    MOVLW 0xE0                  ; reload timer1
    MOVWF TMR1H                     ;HIGH BYTE

    MOVLW 0x08
    MOVWF TMR1L                 ;LOW BYTE
    RETFIE

START
    BCF STATUS, RP0; BANK 0

    MOVLW 7
    MOVWF CMCON                     ; DISABLE COMPARATOR
    CLRF GPIO


    BSF STATUS, RP0             ; BANK1
    CLRF ANSEL
    BSF INTCON, 7
```

```
    BSF INTCON, 6

    MOVLW B‘00001’
    MOVWF TRISIO

    BCF PIR1, 0
    BSF PIE1, 0

    BCF STATUS, RP0                ;change back to PORT memory bank

    MOVLW B‘00001101’
    MOVWF T1CON

    MOVLW 0xff; clear timer1
    MOVWF TMR1H

    MOVLW 0xfe
    MOVWF TMR1L
; Main

WAIT;     simple loop waiting for timer1 to time-out and generate an interrupt
    BCF GPIO, 1                    ; switch off LED
    GOTO WAIT

    END
```

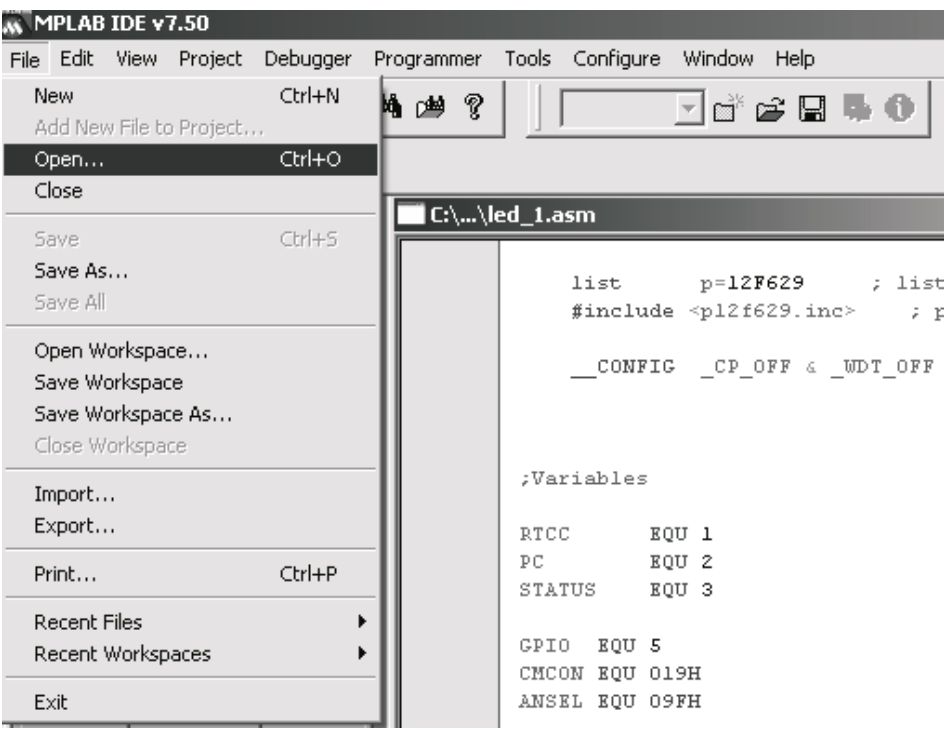
The MPLAB Programming environment

MPLAB IDE, free to download from www.microchip.com, is used to assemble the source code (LED_1.ASM) and to generate the machine (HEX) code for the PIC. MPLAB consists of a full software development environment including a text editor, simulator and debugger with viewable stopwatch and file registers. In this example, the program file LED_1.ASM is imported into MPLAB and assembled. But be warned, using MPLAB requires a short learning curve. Hopefully the following sequence will give the reader a start.

The program LED_1ASM is imported into the MPLAB environment:
Select from the menu: **File-Open-Led_1.asm**. This is the editing window and the file can also be typed in by hand and edited.

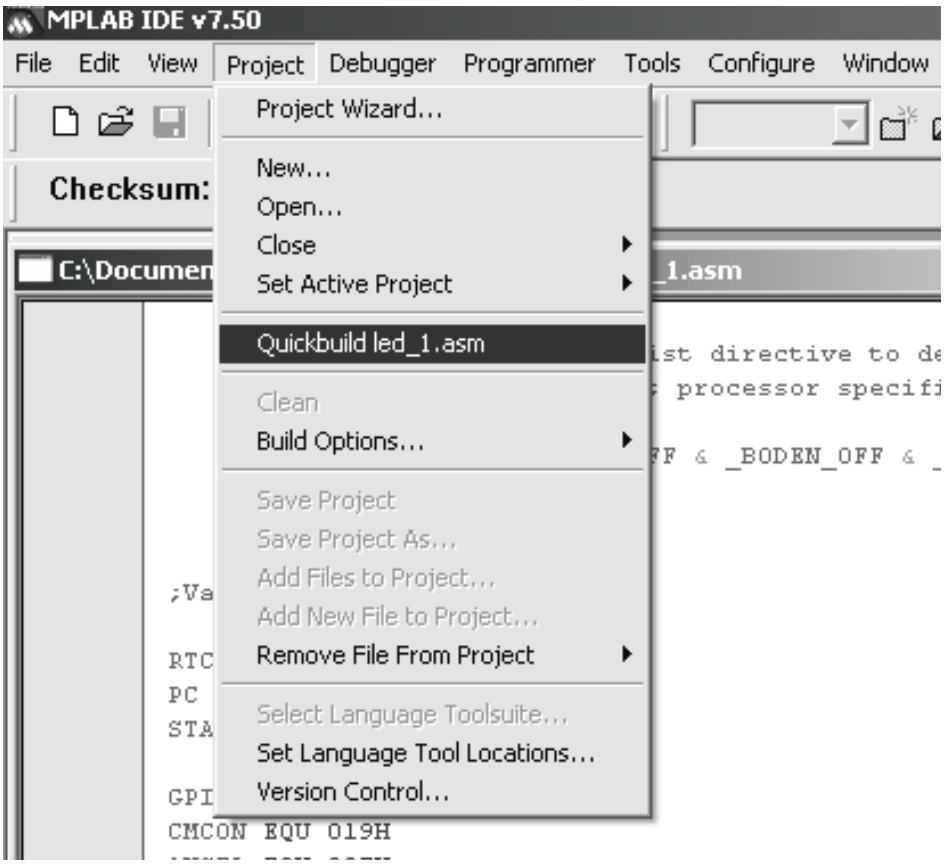


Opening screen of MPLAB.



Importing file LED_1.ASM into the editor.

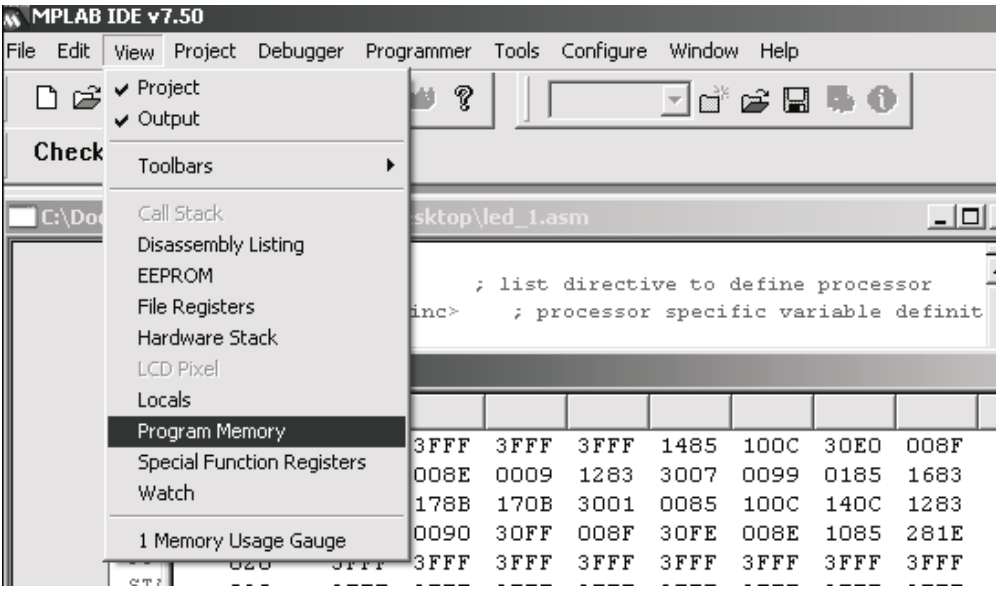
The resulting source code appears in the editing window. Ensure that file P12F629.inc is resident in the same directory as the source file. If an error message 'Filename too long' is generated, it means that the target source code file is located in a deep sub-directory and has to be re-located, preferably in a sub-directory of C:\.



Assembling the source code.

Now select Project-**Quickbuild led_1.asm**. If there is no syntax error, the programme will be assembled and 'ASSEMBLY SUCCEDED' message will be seen.

The program will be assembled and a hex file generated. This file is imported into the PIC programmer, connected to the target board and then the PIC is programmed. By selecting View, the Program Memory, and the Disassembly Listing can be seen:



Program LED_2 .ASM

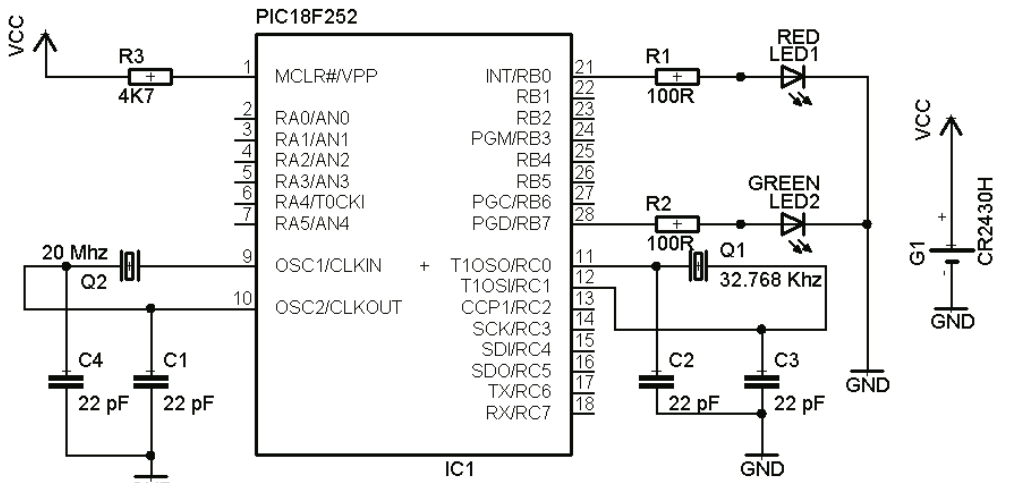
This program demonstrates how the LED can be switched with very low power consumption. The software makes use of the watchdog timer that wakes the PIC every half second and lights the LED for 0.75 ms. During 'sleep', the PIC current consumption is as little as 2 microamps.

```
    List p=12F629; list directive to define processor
    #Include <p12f629.inc>; processor specific variable definitions

    __CONFIG _CP_OFF & _WDT_ON & _BODEN_OFF & _PWRTE_ON &
_LP_OSC &  _MCLRE_ON & _CPD_OFF

RTCC EQU 1                  ; RTCC
PC EQU 2
STATUS EQU 3                ; STATUS REGISTER

GPIO EQU 5
CMCON EQU 019H              ;PORTS DIGITAL I/O
ANSEL EQU 09FH

ORG 0x00

    BCF STATUS, RP0         ; BANK 0
    MOVLW 7
```

```
    MOVWF CMCON             ; DISABLE COMPARATOR

    CLRF GPIO
    CLRF ANSEL
    BSF STATUS, RP0         ;BANK1
    BCF INTCON, 7           ;DISABLE GLOBAL INTERRUPT

    MOVLW B‘00001’
    MOVWF TRISIO

    MOVLW B‘00001101’
    MOVWF OPTION_REG

    BCF STATUS, RP0         ; change back to PORT memory bank

CYCLE
    BCF GPIO, 1             ; LED OFF
    CLRWDT
    SLEEP
    BSF GPIO, 1             ; LED ON
    NOP
    NOP
    NOP
    GOTO CYCLE

    END
```

Program LED_3 .C

This program is written in C for the PIC18F252 that is utilised to provide a real-time clock that switches an LED exactly every second. It makes use of an additional internal clock oscillator controlled by a 32.768 kHz crystal that is connected to pins 11 and 12. The normal system clock, e.g. 20 MHz crystal, is connected to pins 9 and 10. The main program execution is controlled by the 20 MHz clock and runs independently, while in the background the 32.768 oscillator also runs independently and is controlled by the TMR0 clock. It produces an interrupt every second and toggles the LEDs connected to port lines RB0 and RB7.



Circuit for Program led_3.C

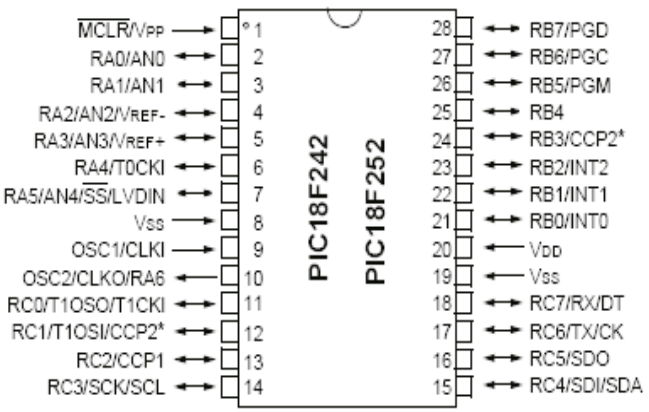
Program LED_3 .C

```
#include <p18F252.h>

void main (void);
void InterruptHandlerHigh (void);
static unsigned long SecondsCount = 0;          // real-time clock seconds counter
union
{
    struct
    {
    unsigned Timeout:1;
                                                //flag to indicate a TMR0 timeout
    unsigned None:7;
    } Bit;
    unsigned char Byte;
}   Flags;
int count;

void
main ()

{
```

```
    Flags.Byte = 0;
    INTCON = 0x20;              //disable global and enable TMR0 interrupt
    INTCON2 = 0x84;             //TMR0 high priority
    RCONbits.IPEN = 1;          //enable priority levels
    TMR0H = 0;                  //clear timer
    TMR0L = 0;                  //clear timer
    T0CON = 0xC4;               //set up timer0 – prescaler 1:2 0x80 16 bit 0xc0 8 bit
    INTCONbits.GIEH = 1;        //enable interrupts
    TRISB = 0;
    TRISA = 0;
    count=0;
    while (1)
    {
                                // do nothing and wait for interrupt
    }

}

// High priority interrupt vector

#pragma code InterruptVectorHigh = 0x08
void
InterruptVectorHigh (void)
{
    _asm
    goto InterruptHandlerHigh   //jump to interrupt routine
    _endasm
}

// High priority interrupt routine

#pragma code
#pragma interrupt InterruptHandlerHigh

void InterruptHandlerHigh ()

{
    if (INTCONbits.TMR0IF)
    // 32768 Hz timer, TIMER1 has overflowed
    {
    INTCONbits.TMR0IF=0;
    Flags.Bit.Timeout=1;
```

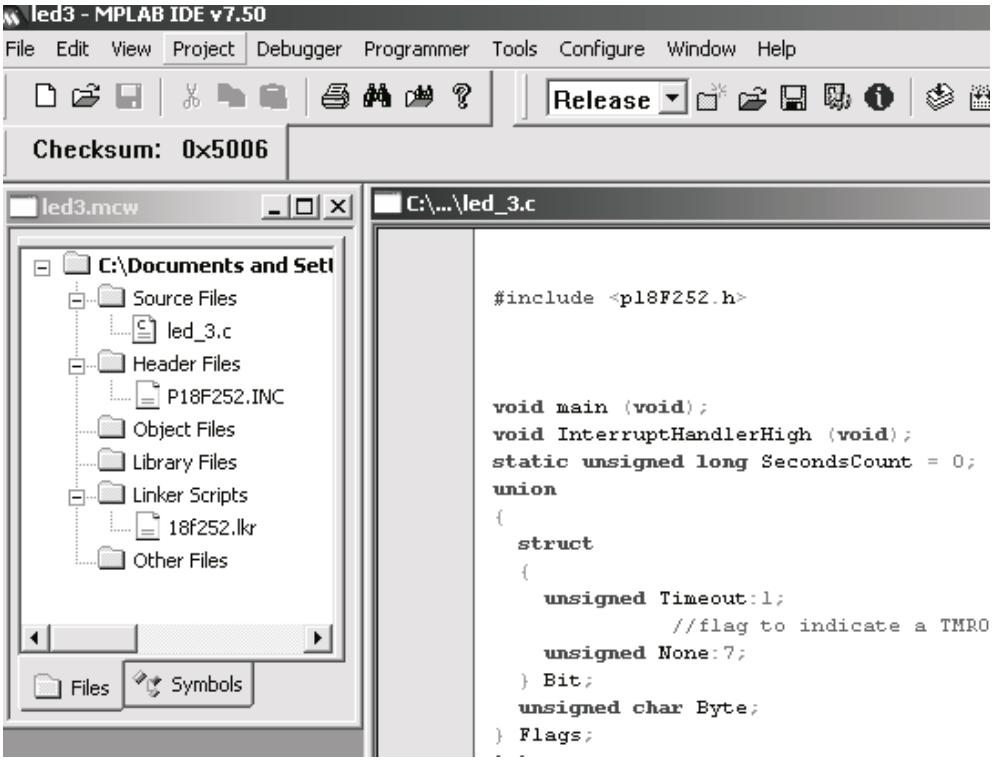
```
        SecondsCount++;
            // one second has elapsed

        if ((SecondsCount/2)*2 == SecondsCount)
        {    PORTBbits.RB0 = 1;

            PORTBbits.RB7 = 0;}
        else
        {PORTBbits.RB0 = 0;

            PORTBbits.RB7 = 1;}
    }
}
```

DIP, SOIC



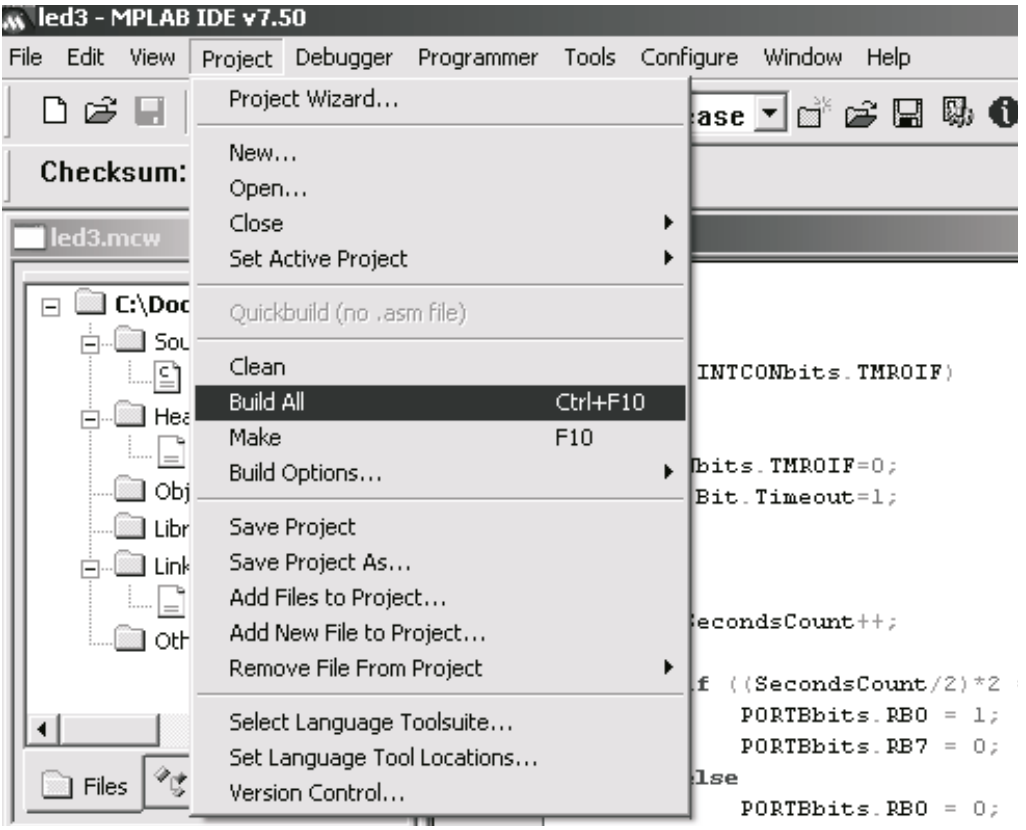
Pin out for the PIC18F252

Unlike an assembler file, (Filename.ASM), The C code is compiled inside a project File 'LED3.MCP'. Select **Project-Open-LED3.MCP** from the menu bar.



Select **View-Project**. It can be seen that the C file led_3.c has been selected as the C source file. In addition, a header file PIC18F2525.inc and a linker script 18f252.lkr is included. These additional files can be found in the MPLAB directory.

Now select Project-**Build All** in the menu bar. The C file is compiled and the hex code is generated for programming the PIC.

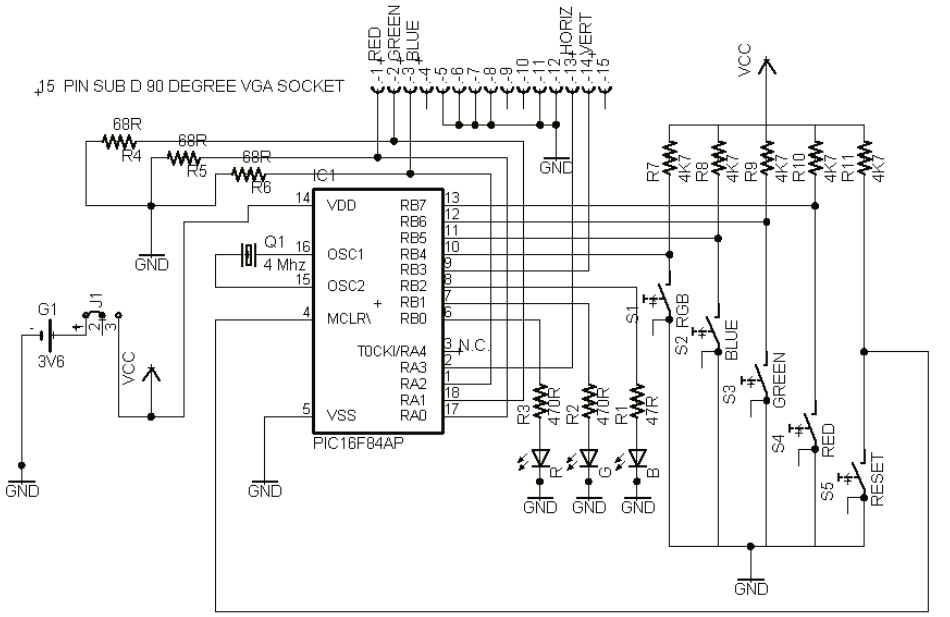


This concludes the short tutorial on using MPLAB to assemble or compile the source code.

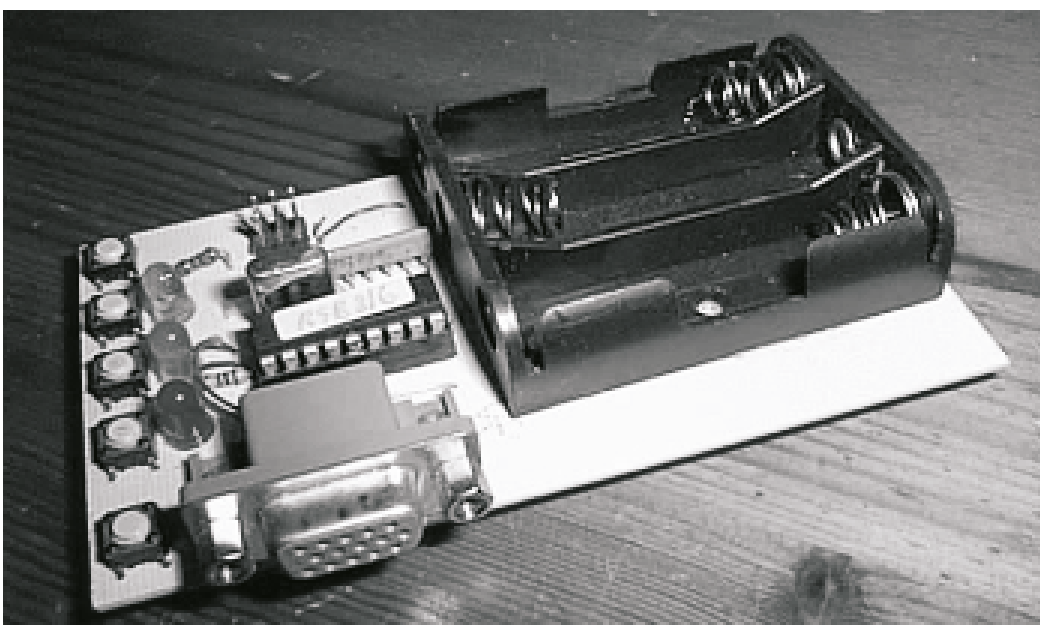
1.4 Case Study RGB VGA monitor tester

This project utilises a red, green, and blue LED to indicate which switch has been pressed, and therefore which software routine has been selected. This unit outputs a VGA signal to the 15-pin connector – a red, green, blue or white screen is produced on the monitor. Pressing the reset button blanks the display and the PIC goes to sleep. Note that the blue LED has a much smaller limiting resistor since it requires a larger forward voltage for sufficient brightness.

The video signal generated by the PIC is a standard VGA signal with a 61 Hz frame rate and 31 kHz line rate. But beware when testing CRT monitors, since, in the event of erroneous software, a much higher line frequency could be created with a corresponding increase in CRT EHT, and subsequent malfunction of the monitor.



Circuit diagram of the Portable RGB VGA monitor tester.



RGB Monitor tester.

Description of Firmware

PIC port lines RA0, RA1 and RA2 control the switching of the RGB signals to the PC monitor. On power-up, the routine at START configures port A and B lines for inputs (switches SW1 SW2 SW3 and SW4) and for outputs (RGB, line sync and frame sync).

The interrupt mechanism of the PIC is enabled by instruction: BSF INTCON, 7 and BSF INTCON, 3. All LEDs are extinguished and the PIC is sent to sleep. If any logic level on port B pins RB4 through RB7 changes logic state by pressing any switch, an interrupt is issued and the program vectors to address 04H, at which location INTSVC – the interrupt service routine is executed.

This routine polls each switch in turn and sets a flag in variable LIGHTS – the bit is set dependant on which switch has been pressed. Subroutines A, BB, C and D are identical in structure and generate red, green, blue or RGB (white) signals.

In summary:

- A frame sync pulse is issued.
- The RTCC is re-loaded with 60H (This can be fine-tuned for the correct frame rate).
- A line sync pulse is issued (3 microseconds).
- Instructions at label FIELDS* switch on or off the R G B outputs along one line scan period.
- Successive video lines are then repeated until bit 7 of the RTCC register is set, at which point a frame sync pulse is issued and the entire frame is repeated.
- Pressing the RESET button switch at any time will disable all outputs and place the PIC back into the SLEEP mode.

Program RGB.ASM

These program snippets taken from file RGB.ASM demonstrate how a complete VGA frame is produced. In this case only the green (pin 3) of the VGA connector is fired.

CYCLE_G

```
                                    ; FRAME
    BCF PORT_B, FRAME
    CALL DELAY
    BSF PORT_B, FRAME

    MOVLW B‘00000111’
    OPTION
    MOVLW .60                       ; DEFINES FRAME RATE
    MOVWF RTCC
```

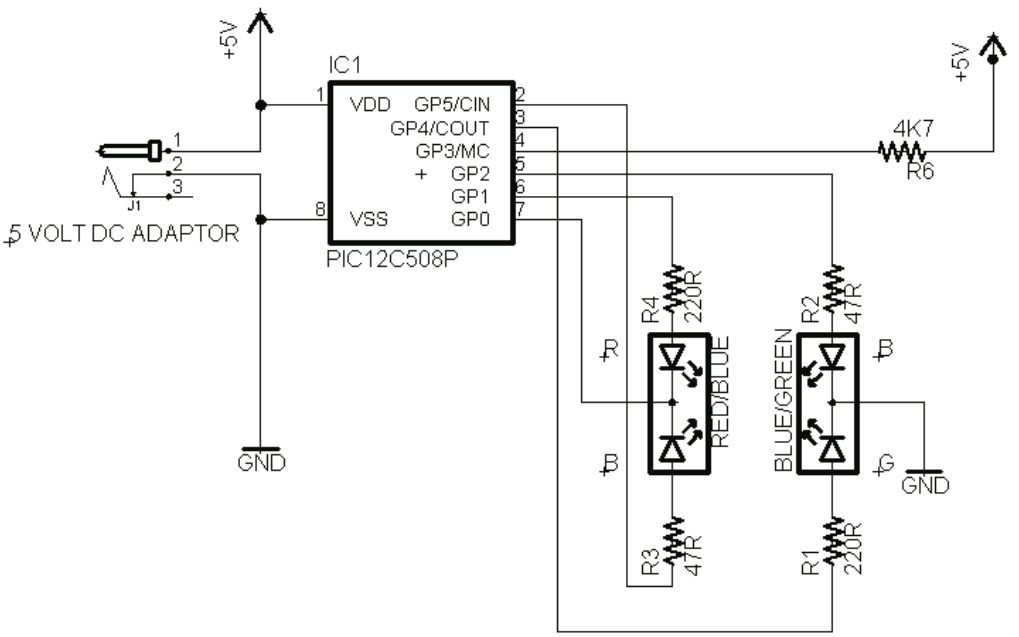
```
LINES_G

    BCF PORT_A, LINE; LINE SYNC
    NOP
    BCF PORT_A, GREEN; NO COLOUR SIGNAL TRANSMITTED
    BSF PORT_A, LINE; DURING LINE SYNC PERIOD

FIELDS_B
    BSF PORT_A, GREEN; GREEN SCREEN
    BCF PORT_A, RED
    BCF PORT_A, BLUE
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BTFSC LIGHTS, 3; CHECK IF RGB SWITCHES PRESSED
    GOTO D
    BTFSC LIGHTS, 2
    GOTO C
    BTFSC LIGHTS, 0
    GOTO A
    NOP
    BTFSS RTCC, 7; CHECK IF END OF FRAME
    GOTO LINES_B
    NOP
    GOTO CYCLE_G
```

1.5 Case Study Christmas light

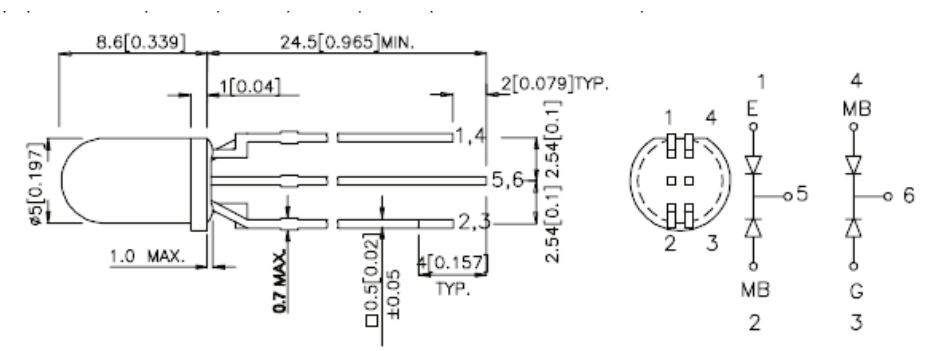
This simple PIC/LED combination will provide an interesting light for your Christmas tree. A Red, green and blue LED is available in a single package. The T – 1 ¾ full colour RGB lamp contains 2 blue LEDs, 1 green LED and 1 red LED which together can produce any colour in the visible spectrum including white light. This simple project uses a single PIC12C508 that is an OTP (one time programmable device). It is also available in a win-

dowed EPROM version (suffix JW) which can be erased under UV light and reprogrammed.

Since this light is intended to operate continuously, and because the high efficiency red LED can have a peak forward current of 200 mA, battery operation is not an option and therefore the unit is powered by a 5 volt mains adapter. The lamp consists of 6 electrodes, two of which are the common cathode connections for each pair of diodes. The circuit has been designed so that the blue-green LED pair's cathode is strapped to ground. The other LED pair has a switchable cathode via the GPIO pin. Note that the blue LEDs have a much smaller limiting resistor since they required a greater voltage drop to operate at peak brightness.
This circuit could also provide the basis of a status indicator, showing red for OFF, green for ON, orange (red+green) for STANDBY, and a flashing blue light for an ALARM condition.
In this case, by disconnecting the GPIO, 0 pin and configuring it as an input e.g. RS232 bit-bang method, whilst grounding the R-B cathode pair.



Circuit diagram of the Christmas light.



Tricolour led connections: Note. E signifies high Efficiency red.

Program XMAS_1.ASM

The PIC firmware.

A varied combination of LEDs are switched in by the PIC ‘s GPIO lines to produce varying colours. The reader can experiment with the software to produce a pleasing effect. The fuse configuration for the clock is the internal oscillator with a nominal frequency of 4 MHz.

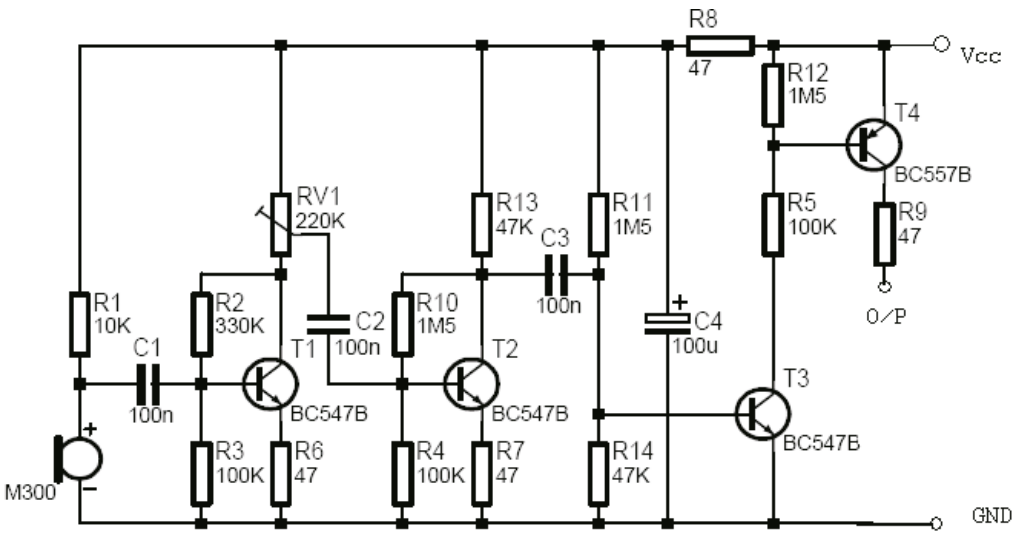
```
    LIST P=12C508

RTCC EQU 1
PC EQU 2
STATUS EQU 3                ;STATUS REGISTER
GPIO EQU 6; PORT
OSCAL EQU 5

SECS EQU 0CH

#DEFINE RED GPIO, 0
#DEFINE REDB GPIO, 4

#DEFINE GREEN GPIO, 1
#DEFINE GREENB GPIO, 5

#DEFINE BLUE GPIO, 2
```

```
    ORG 0

    MOVWF OSCAL         ;12x  PIC FAMILY GPIO PORT
    CLRF GPIO
    MOVLW 0
    TRIS GPIO

    MOVLW B‘00000’
    OPTION

CYCLE
    BCF RED
    BCF REDB            ;BLUE LED
    BCF GREEN           ;GREEN LED
    BCF GREENB          ;RED LED
    BCF BLUE            ;2ND BLUE LED

    BSF BLUE
    CALL DELAY5SEC
    BSF GREEN
    CALL DELAY5SEC
    BSF RED
    CALL DELAY5SEC

    BCF GREEN
    BCF BLUE
    BCF RED

    BSF REDB
    CALL DELAY5SEC
    BSF GREENB
    CALL DELAY5SEC
    BSF GREEN
    BSF RED
    CALL DELAY5SEC
    CALL DELAY5SEC
    BSF BLUE

    GOTO CYCLE

DELAY5SEC
    MOVLW B‘00000111’
    OPTION
```

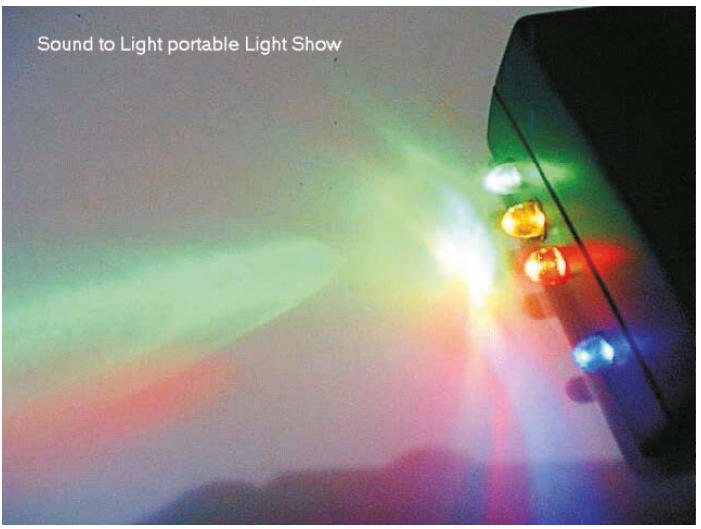
```
    CLRF RTCC

    MOVLW 10
    MOVWF SECS

WAIT CLRF RTCC
WAIT1 CLRWDT              ;TIMING LOOP
    BTFSS RTCC, 7
    GOTO WAIT1
    CLRF RTCC
    DECFSZ SECS, 1
    GOTO WAIT
    RETLW 0

    END
```

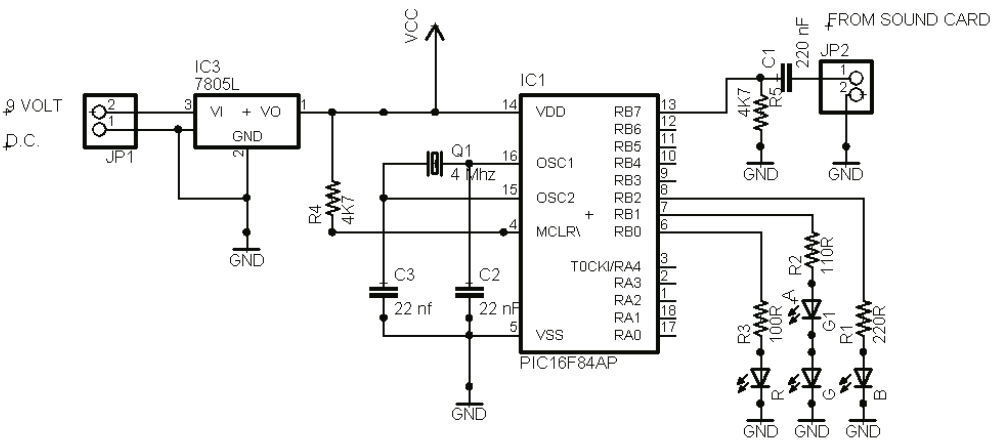
1.6 Case Study 3 channel sound to light

This LED project is based around the Velleman MK103 analogue sound to light kit. It consists of a microphone amplifier circuit using discrete transistors as shown below, to which is added a PIC-based circuit to discriminate between ranges of sound, i.e. bass, middle and high audio frequencies, and then to light the appropriate colour – in effect creating a digital sound to light unit.



Sound card circuit courtesy of Velleman NV.





The PIC circuit to convert audio to light.
Program sound_to_light. ASM.

Potentiometer RV1 on the sound card is the 'sensitivity' control and is adjusted according to the ambient sound intensity. The output from this audio amplifier is taken from transistor T4 via R9, a 47-ohm resistor to the input of PIC port RB7 via a dc-isolating capacitor C1.A simple 78L05 regulator (IC3) provides the necessary 5-volt supply for the PIC. Although only 3 channels are shown, the project can easily be extended to accommodate 8 channels. It can be seen that LEDs G and G1 are connected in series, and subsequently a lower limiting resistor is required to provide sufficient volts drop across the 2 LEDs. Different colour LEDs can also be connected.

Program sound_to_light. ASM

This program makes use of the RTCC timer to determine the frequency range of the incoming audio, and to light the corresponding LED. During silence the program loops at label ABCDE. In the event of a sound signal reaching PORTB, 7, the RTCC bits 1, 3 and 5 are checked. For lower frequencies, the lower bits will still be cleared, at higher frequencies either bits 3 or 5 will be set. Routines MID, TREBLE or BASS will then be selected according to the frequency detected.

A more sophisticated unit could be built utilising the PIC12F675 with on-board 4 channel 10-bit AD convertors. Here, one ADC channel could be used to measure the actual time period of the incoming audio at regular intervals. The ADC value can then be used to determine which LED is to be lit.

```
    LIST P=16F84

RTCC EQU 1
PC EQU 2
STATUS EQU 3                ; STATUS REGISTER
PORTB EQU 6
PORTA EQU 5

COUNT EQU 0CH
THIRTYTWO EQU 0DH

    ORG 0

    MOVLW B‘10000000’
    TRIS PORTB

CYCLE

    MOVLW B‘00000101’    ; 64 MICROSECONDS 100 US = 10kHz
    OPTION

AGAIN
    BCF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    CLRF RTCC

ABCDE BTFSS PORTB, 7
    GOTO ABCDE
```

```
    ; LOW START TIMING

GGH BTFSC PORTB, 7      ;WAIT FOR AUDIO SIGNAL LOW
    GOTO GGH
    BTFSC RTCC, 1       ;CHECK RTCC VALUE
    GOTO MIDD
    BTFSC RTCC, 3
    GOTO TREBLE
    BTFSC RTCC, 5
    GOTO BASS
    CALL SECOND

    GOTO CYCLE

MIDD BSF PORTB, 1
    CALL ZING
    GOTO AGAIN

TREBLE
    BSF PORTB, 0
    CALL ZING
    GOTO AGAIN

BASS
    BSF PORTB, 2
    CALL ZING
    GOTO AGAIN

ZING MOVLW 0FFH
    MOVWF COUNT

LONG
    DECFSZ COUNT, 1
    GOTO LONG
    RETLW 0

SECOND
    MOVLW B‘00000111’   ; PRESCALER 00000111 (/256)
    OPTION
    CLRF RTCC

    MOVLW .32
    MOVWF THIRTYTWO  ; COUNTER
```

```
SHORT BTFSC RTCC, 7       ; TEST FOR 128 DECIMAL
                          ; I.E. BIT 7 = HIGH
    GOTO JMP2

    GOTO SHORT

    JMP2 CLRF RTCC
                          ; 32 LOOP COUNTER
    DECFSZ THIRTYTWO, 1
    GOTO SHORT
    RETLW 0

    END
```

2 White light emitting diodes

A single LED's brightness can easily be controlled by inserting a suitable resistor in series with the LED. For example, a 220-ohm resistor will produce more brightness than a 1K0 resistor. However, high intensity white LEDs are typically used in backlighting LCD displays and typically require a forward voltage of between 3.2 and 3.6V. Usually two LEDs in series are used for backlighting and therefore a supply voltage of at least 6.4 is required. Since many of the case studies in this book use a single low-voltage supply such as a 3 volt CR2032 or 3.7 volt lithium cell, A higher voltage needs to be generated to supply the white LEDs.

Since the PIC16F876 and the PIC18F252 are both equipped with an internal PWM module, a simple switched mode DC to DC converter circuit shown below can be driven from the CCP1 (RC2 pin 13) of the PIC. A pulse train of the required frequency and mark space ratio from this pin can then be used to drive the switched mode circuit to produce the required voltage to drive the LED to a required brightness. This chapter shows three different ways to produce the pulse train.

In addition various charge pump voltage inverters, step-up PWM converter and switched-capacitor voltage converter ICs are available. This chapter shows some of the available voltage converter ICs- and how to connect them.

2.1 PWM LED brightness and voltage control

Both the PIC16F876 and the PIC18F252 contain PWM modules, which are software programmable to produce a PWM o/p of adjustable duty cycle and frequency. If we then connect the PWM output of the PIC to the PWM circuit, the output voltage and therefore the brightness of the LEDs can be adjusted by programming the PWM registers to produce a suitable frequency and duty cycle. Note that this circuit can easily produce a DC 9 volt output, much higher than the 6V2 required by the two high-brightness white LEDs, so precautions have to be taken such as connecting a 6.1 zener diode across the LEDs. By reducing the PWM output frequency, the voltage across the LEDs can also be reduced.

Program 876-pwm.ASM

The following program example for the PIC16F876 configures the PWM module and produces a pulse train for the PWM circuit. Note that once the PWM module is programmed, it will run in the background, outputting a pulse train on the RC2 pin of the PIC16F876 whilst the main system program is allowed to run.

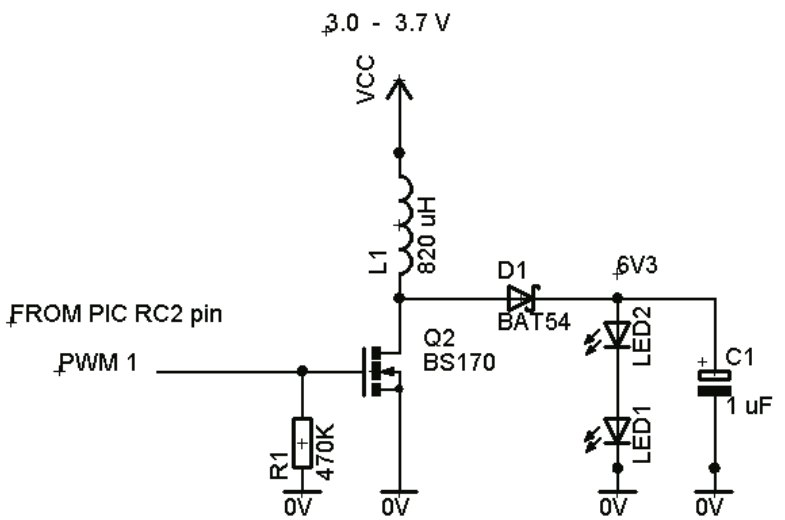
(a)

```
LIST p=16f876
INCLUDE P16F876.inc


PORTC EQU 7

ORG 0
MOVLW 0
TRIS PORTC


MOVLW 03CH
MOVWF T2CON

MOVLW .50
MOVWF PR2
```

(b)

```
MOVLW 01FH
MOVWF TMR2

    MOVLW 0CH
MOVWF CCP1CON

    MOVLW 029H
MOVWF CCPR1L

MOVLW 01CH
MOVWF T2CON

MOVLW 4; 01FH
MOVWF TMR2


BUG GOTO BUG
```

The data sheet shown on the next page shows how the PWM is configured.



CIRCUIT for Program 876-pwm.ASM and 876_2-pwm.C

8.3 PWM Mode (PWM)

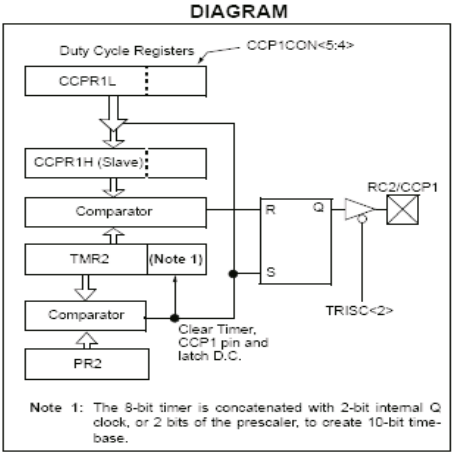
In Pulse Width Modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note:	Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 8-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 8.3.3.

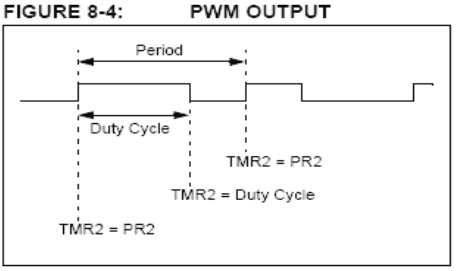
FIGURE 8-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: The 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time-base.

A PWM output (Figure 8-4) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 8-4: PWM OUTPUT



8.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

PWM frequency is defined as 1 / [PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see Section 7.1) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

8.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitch-free PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock, or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the formula:

$$\text{Resolution} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

Program 876_2-pwm.C

This program is written in C but unfortunately a PIC16xx C compiler is not currently freely available from the Microchip Website. However a compiler (CC5X) is available from B. Knudsen Data and can be integrated seamlessly into the MPLAB programming environment. This program uses the #asm and #endasm directives to allow inline assembler to be incorporated into the main C program. A pulse train of software programmable frequency and mark space ratio determines the o/p voltage of the SM power supply shown above and thus provides sufficient voltage to light the 2 series LEDs.

```
Void main (void);
Void delay10 (char);

Void main (void)

    {

    TRISC = 0X00;
    PORTC = 0XFF;
                                ;INLINE ASSEMBLER
    #asm

    MOVLW 0x3C
    MOVWF T2CON

    MOVLW 0x23
    MOVWF PR2

    MOVLW 0x1F
    MOVWF TMR2

    MOVLW 0x2C                  ;CONFIGURE PWM REGISTERS
    MOVWF CCP1CON               ;FOR FREQUENCY AND MARK/SPACE RATIO
    MOVLW 0x29
    MOVWF CCPR1L

    MOVLW 0x3C
    MOVWF T2CON

    MOVLW 0x1F
    MOVWF TMR2

    #endasm
```

```
delay10(1) ;

/* USER CODE GOES HERE   – C CODE      * /

}

void delay10( char n)
/*
  Delays a multiple of 10 milliseconds using the TMR0 timer
  Clock: 4 MHz   => period T = 0.25 microseconds
 */
{
     char i;

     OPTION = 7;
     do  {
          clrwdt();  // only if watchdog enabled
          i = TMR0 + 39; /* 256 microsec * 39 = 10 ms */
          while ( i != TMR0)
            ;
     } while ( --n > 0);
}
```

The PIC18F252 PWM module is identical to that of the PIC16F876. The following program configures the PWM module, starts with the LED at minimum intensity, after a delay increases to medium intensity, and then to full brightness. A loop routine then causes the LED to start from minimum intensity and gradually increases to full intensity.

Program 252-pwm.C

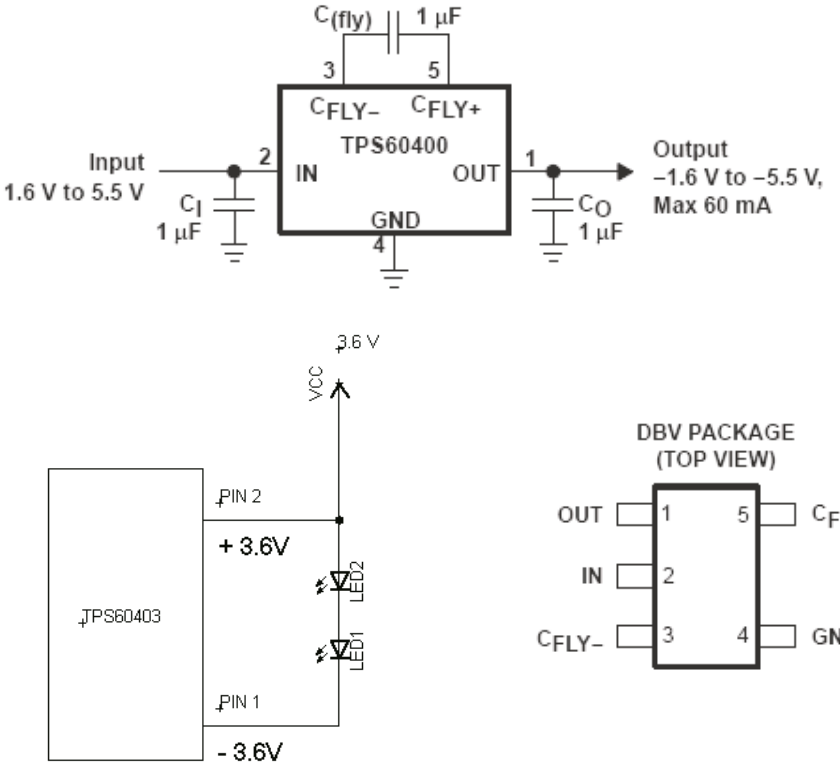
```
// 20 MHz crystal

#include <delays.h>
#include <pwm.h>
#include <stdlib.h>
#include <timers.h>

#Include <p18f252.h>

Void main (void)
     {
char romst;
```

```
TRISC = 0x00;
PORTC = 0xff;

OpenTimer2 (TIMER_INT_OFF & T2_PS_1_1 & T2_POST_1_8);

    OpenPWM1 (50);

    SetDCPWM1 (15);                    // low LED intensity

    SleepMS (8000);
    SetDCPWM1 (50);                    // medium LED intensity

    SleepMS (8000);
    SetDCPWM1 (100);                  // high LED intensity

    SleepMS (8000);
    for (romst=10; romst<100; romst++)   // DIMMER CONTROL

{
SetDCPWM1 (romst);
SleepMS (1000);

}

}
```

2.2 TPS60403 charge pump voltage inverter

A **charge pump** is an electronic circuit that uses capacitors as energy power source. Charge pump circuits are capable of high efficiencies, sometimes as high as 90-95%, while being electrically simple circuits.

Charge pumps use some form of switching device(s) to control the connection of voltages to the capacitor. For instance, in order to generate a higher voltage, the first stage involves the capacitor being connected across a voltage and charged up. In the second stage, the capacitor is disconnected from the original charging voltage and reconnected with its negative terminal to the original positive charging voltage. Because the capacitor retains the voltage across it (ignoring leakage effects), the positive terminal voltage is added to the original, effectively doubling the voltage. The pulsing nature of the higher voltage output is typically smoothed by the use of an output capacitor. This is the charge pumping action, which typically operates at tens of kilohertz up to several megahertz to minimise the amount of capacitance required. The capacitor used as the charge pump is typically known as the ‘flying capacitor’.



AVAILABLE OPTIONS

PART NUMBER†	MARKING DBV PACKAGE	TYPICAL FLYING CAPACITOR [µF]	FEATURE
TPS60400DBV	PFKI	1	Variable switching frequency 50 kHz–250 kHz
TPS60401DBV	PFLI	10	Fixed frequency 20 kHz
TPS60402DBV	PFMI	3.3	Fixed frequency 50 kHz
TPS60403DBV	PFNI	1	Fixed frequency 250 kHz

Typical application circuit

For an input voltage of 3.6 volts on pin 2, a –ve voltage of 3.6 volts will be produced on pin 1. Two white LEDs in series connected across pins 1 and 2 will then have a potential difference of 7.2 volts across them.

2.3 TPS61040 low power DC/DC boost converter

The TPS61040 is a high-frequency boost converter dedicated to small to medium LCD bias supplies and white LED backlight supplies. The circuit shown below can generate up to 120 mA of current and is used for ultra bright white LEDs. For portable applications, where cur-

rent consumption needs to be limited, a simple potentiometer between supply and ground can be inserted – the slider connecting to pin 5 of the TPS61040 i.c., and the Vcc supply to the potentiometer can be provided by 2 or 4 PIC port pins strapped together. The enable pin (4) can also be connected to one of the PIC port pins to switch the LEDs on or off.

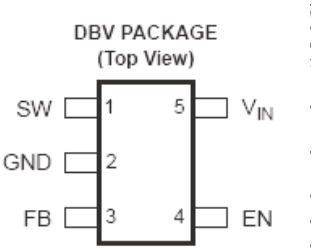
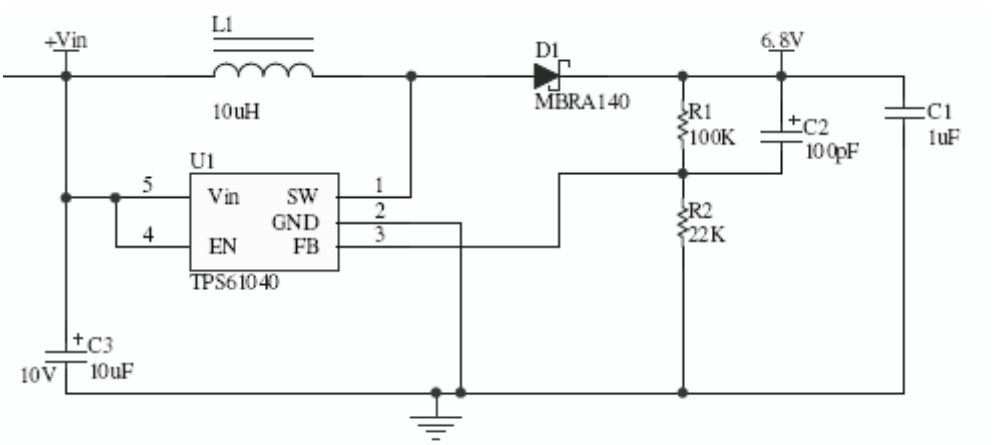
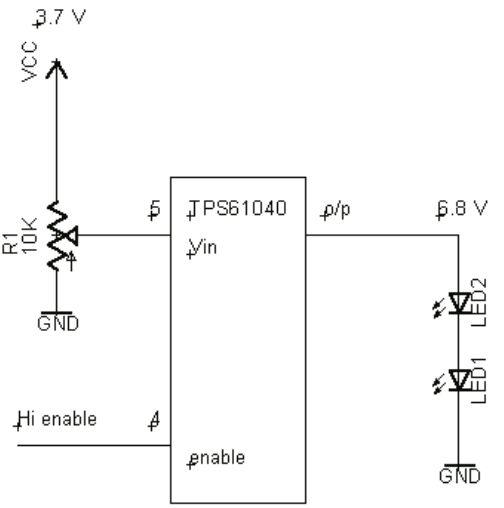


Table 1. Terminal Functions

TERMINAL NAME	TERMINAL DBV NO.	TERMINAL DRV NO.	I/O	DESCRIPTION
EN	4	3	I	This is the enable pin of the device. Pulling this pin to ground forces the device into shutdown mode reducing the supply current to less than 1 µA. This pin should not be left floating and needs to be terminated.
FB	3	4	I	This is the feedback pin of the device. Connect this pin to the external voltage divider to program the desired output voltage.
GND	2	1	–	Ground
NC	–	5	–	No connection
SW	1	6	I	Connect the inductor and the Schottky diode to this pin. This is the switch pin and is connected to the drain of the internal power MOSFET.
V_{IN}	5	2	I	Supply voltage pin



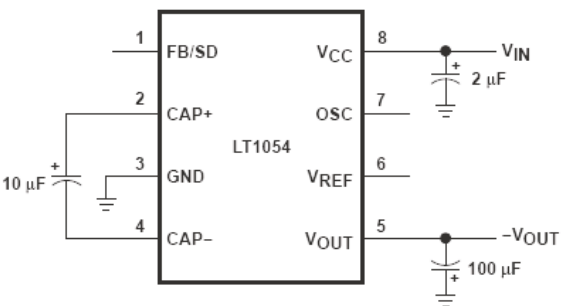
Typical circuit.



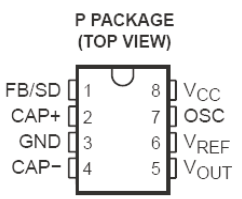
Controlling the brightness of the LEDs

2.4 LT1054 switched-capacitor voltage converter

The LT1054 is a bipolar switched-capacitor voltage converter with regulator. It requires just 3 external capacitors to form a functioning circuit. A control input (FB/SD) can be connected to a port pin to disable the i.c. A logic low will effect this. The two white LEDs are connected across pins 8 and 5 that produce a potential difference between them of twice the supply voltage.



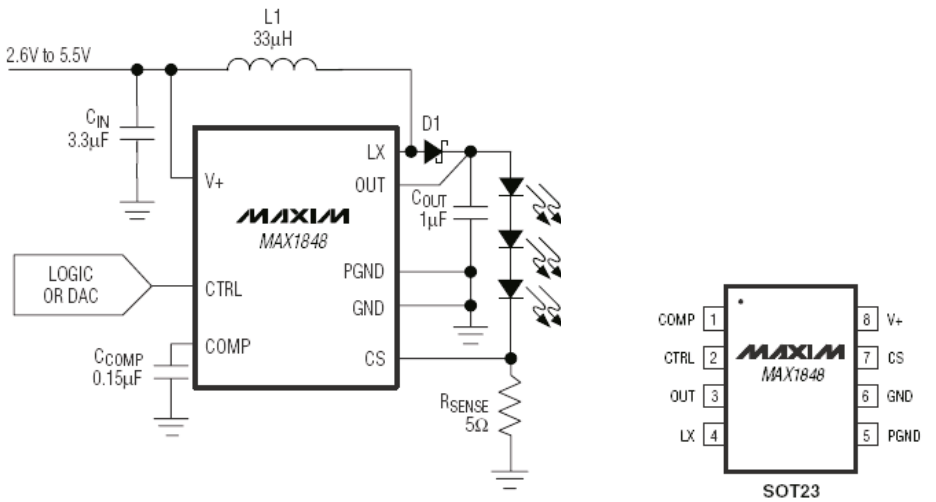
Typical operating circuit.



Pin out of the LT1054.

2.5 MAX1848 white LED step-up converter

The MAX1848 is a PWM step up converter that allows series connection of white LEDs so that the LED currents are identical for uniform brightness. A schottky diode with low voltage drop and fast recovery time such as the 1N5817 is required. Control of LED brightness is achieved by connecting a potentiometer or the output of a DAC to the CTRL pin (2). Alternately this pin can be used to shut down the converter by applying a logic 0 level.



Typical operating circuit and pin-out.

PIN	NAME	FUNCTION
1	COMP	Compensation Pin for Error Amplifier. Connect capacitor from COMP to GND. Startup time is set by the capacitance connected to this pin (0.833ms for each 0.01µF). V_{COMP} passively discharges to GND when in shutdown.
2	CTRL	Brightness/Shutdown Dual Mode Control Input. LED brightness and IC shutdown are controlled by the voltage on CTRL. Voltages between 250mV and 5.5V or (V+ + 2V), whichever is less, adjust the brightness from dim to bright, respectively. To put the IC into shutdown, drive below 100mV or connect to GND.
3	OUT	Overvoltage Sense. When V_{OUT} is greater than 13.25V, the internal N-channel MOSFET is turned off and V_{COMP} decays to GND. When V_{OUT} drops below 12.25V, the IC will re-enter soft-start. Connect a 1µF capacitor from OUT to GND.
4	LX	Inductor Connection. Drain of the internal high-voltage N-channel MOSFET.
5	PGND	Power Ground. Source of the internal high-voltage N-channel MOSFET.
6	GND	Ground
7	CS	Current-Sense Feedback Input. Connect a resistor from this pin to GND to set the LED bias current. This pin regulates to 7.5% of V_{CTRL}.
8	V+	Supply Voltage Input. The IC is powered from this pin. Input range is 2.6V to 5.5V. Bypass with a ceramic capacitor to GND.

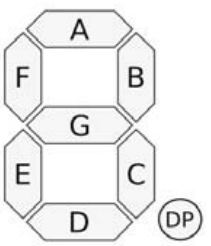
The table below shows the component selection (Ccomp and L) for different current requirements and number of white LEDs.

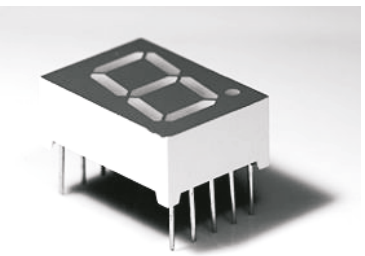
I_{LED} (mA)	NO. OF LEDs	C_{COMP} (µF)	INDUCTOR	
			L (µH)	I_{PEAK}(mA)
12	3	0.220	56	80
	2	0.100		
20	3	0.150	33	130
	2	0.068		
40	3	0.100	15	260
	2	0.047		
60	3	0.068	10	375
	2	0.01		

3 7-segment Displays

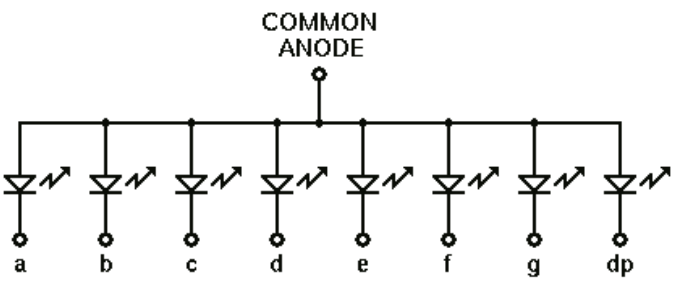
These displays use LED technology to produce a digital display and are well suited to portable applications. They are ideal for use in digital clocks and are found in a myriad of applications. They are easily interfaced to PIC chips requiring a minimum of extra components and are available in a variety of colours and sizes. This chapter describes the structure of 7-segment displays and includes several case studies that use multiplexed displays.

3.1 Fundamentals of 7-segment LED displays



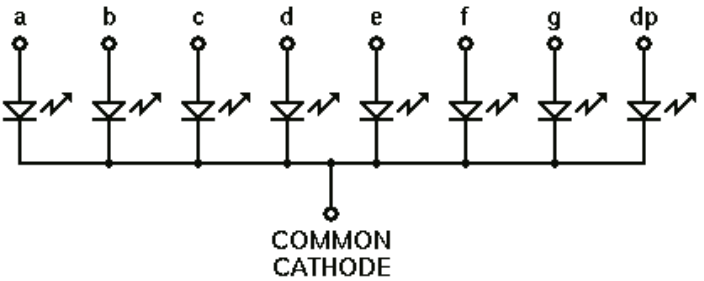


A seven-segment display, as its name indicates, is composed of seven individual LED elements. Individually on or off, they can be combined to produce simplified representations of numerals 0 to 9 and letters A to F. Often the seven segments are arranged in an *oblique*, or italic, arrangement, which aids readability.



As shown in the two schematics, the LEDs in a seven-segment display are not isolated from each other. Rather, either all of the cathodes, or all of the anodes, are connected together into a common lead, while the other end of each LED is individually available. This means fewer electrical connections to the package, and also allows us to easily enable or disable a particular digit by controlling the common lead. (In some cases, the common

connections are made to groups of LEDs, and the external wiring must make the final connections between them. In other cases, the common connection is made available at more than one location for convenience when laying out printed circuit boards. When laying out circuits using such devices, you simply need to take the specific connection details into account.)



There is no automatic advantage of the common-cathode seven-segment unit over the common-anode version, or vice-versa. Each type lends itself to certain applications, configurations, and logic families.

The seven segments are arranged as a rectangle of two vertical segments on each side, with one horizontal segment on the top and bottom. Additionally, the seventh segment bisects the rectangle horizontally. There are also fourteen-segment displays and sixteen-segment displays (for the full alphanumeric range) e.g. the 1414 display; however, these have mostly been replaced by dot-matrix displays.

The letters A to G refer to the segments of a 7-segment display, as shown where the optional DP decimal point (an eighth segment) is used for the display of non-integer numbers.

In a simple LED package, each LED is typically connected with one terminal to its own pin on the outside of the package and the other LED terminal connected in common with all other LEDs in the device and brought out to a shared pin. This shared pin will then make up all of the cathodes (negative terminals) OR all of the anodes (positive terminals) of the LEDs in the device; and so will be either a ‘Common Cathode’ or ‘Common Anode’ device depending how it is constructed. Hence a 7-segment plus DP package will only require nine pins to be present and connected.

Integrated displays also exist, with single or multiple digits. Some of these integrated displays incorporate their own internal decoder, though most do not – each individual LED is brought out to a connecting pin. Multiple-digit LED display devices used multiplexed displays to reduce the number of i.c. pins required to control the display. For example, all the anodes of the segments of each digit position would be connected together and to a driver

pin, while the cathodes of all segments for each digit would be connected. To operate any particular segment of any digit, the controlling integrated circuit would turn on the cathode driver for the selected digit, and the anode drivers for the desired segments; then, after a short blanking interval, the next digit would be selected and new segments lit in a sequential fashion (time division multiplexing). In this manner an eight-digit display with seven segments and a decimal point would require only 8 cathode drivers and 8 anode drivers, instead of sixty-four drivers and i.c. pins.

The common anode display requires that the required segments to be lit to have a logic 0 level applied to their cathodes. For example to produce the numeral '0', segments a, b, c, d, e and f are at logic 0, whereas segment g is at a logic 1 and is therefore unlit.

	A	B	C	D	E	F	G
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	1	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

Table showing how each character is constructed.

3.2 Case Study RS232 Data monitor

This project utilises two common anode LED displays to indicate incoming ASCII RS232 serial data as a hexadecimal byte. For example the letter 'A' will be transmitted from a PC in RS232 format and shown on the 7-segment display as '41'. The table below shows that any letter, upper or lower case, number or simple punctuation marks, can easily be accommodated as an 8-bit word. To find the HEX equivalent, of '8' for example, look at the row number, in this case 3, and the column number, 8 which together give 38H.

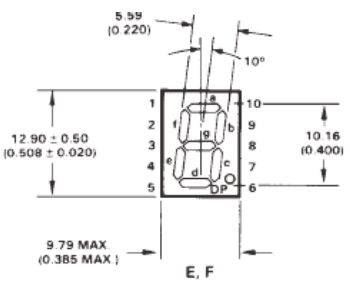
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

If we connect two 7-segment displays together, we need only to parallel all of the segment connections of each segment and to have 2 separate connections to the common anode lines. We can now send data to either display, depending on which anode is active (logic 1). Now if we send data in turn to the two displays in turn at a high rate, the display appears to display the two sets of data (e.g. 0 and 8) simultaneously, all thanks to the human persistence of vision.
This project uses 7-segment displays in this configuration. The display chosen was a 0.3" (7.6 mm) HE blue display as shown below.

	FUNCTION			
PIN	A	B	C	D
1	ANODE [4]	CATHODE [5]	ANODE [4]	CATHODE [5]
2	CATHODE f	ANODE f	CATHODE PLUS	ANODE PLUS
3	CATHODE g	ANODE g	CATHODE MINUS	ANODE MINUS
4	CATHODE e	ANODE e	NC	NC
5	CATHODE d	ANODE d	NC	NC
6	ANODE [4]	CATHODE [5]	ANODE [4]	CATHODE [5]
7	CATHODE DP	ANODE DP	CATHODE DP	ANODE DP
8	CATHODE c	ANODE c	CATHODE c	ANODE c
9	CATHODE b	ANODE b	CATHODE b	ANODE b
10	CATHODE a	ANODE a	NC	NC

Font design

Product not shown actual size



Pin connections (Package A) in function Table above.

Electro / Optical Characteristics - I_F = 20 mA (* HE Blue - I_F = 10 mA) T_a = 25° C

Part Number - Common Cathode		Part Number - Common Anode		Emitting Colour	Wavelength Peak λ_P	Forward Voltage V_F		Luminous Intensity I_V	
Farnell	Forge Europa	Farnell	Forge Europa			typical	max	min	typical
366-4648	FN1-0311B0500GW	366-4650	FN1-0312B0500GW	* HE Blue	465	3.30	3.70	-	8
366-4661	FN1-0311B0100GW	366-4673	FN1-0312B0100GW	Blue	428	3.80	4.50	-	6
Units					nm	V		mcd per segment	

Maximum Ratings T_a = 25° C - Derate above 25° C

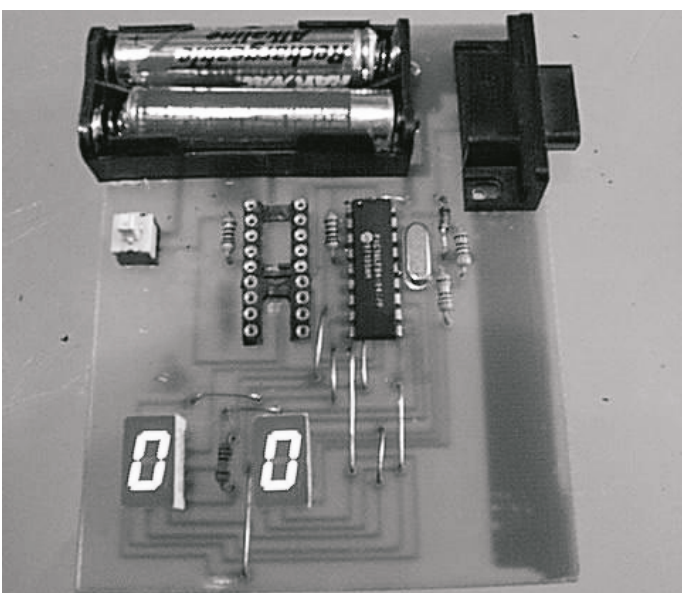
Characteristic	Condition	Symbol	Rating	Units
Pulse Forward Current	0.1 duty cycle @ 1KHz (HE Blue)	I_{FP}	100 (35)	mA
DC Forward Current	(HE Blue)	I_F	25 (15)	mA
Reverse Voltage	I_R = 100 µA	V_R	5	V
Power Dissipation		P_D	85	mW
Operating Temperature		T_{opr}	- 25 to + 80	° C
Storage Temperature		T_{stg}	- 30 to + 85	° C
Lead soldering temperature	1.6 mm from body - max 3 seconds		260	° C

Note

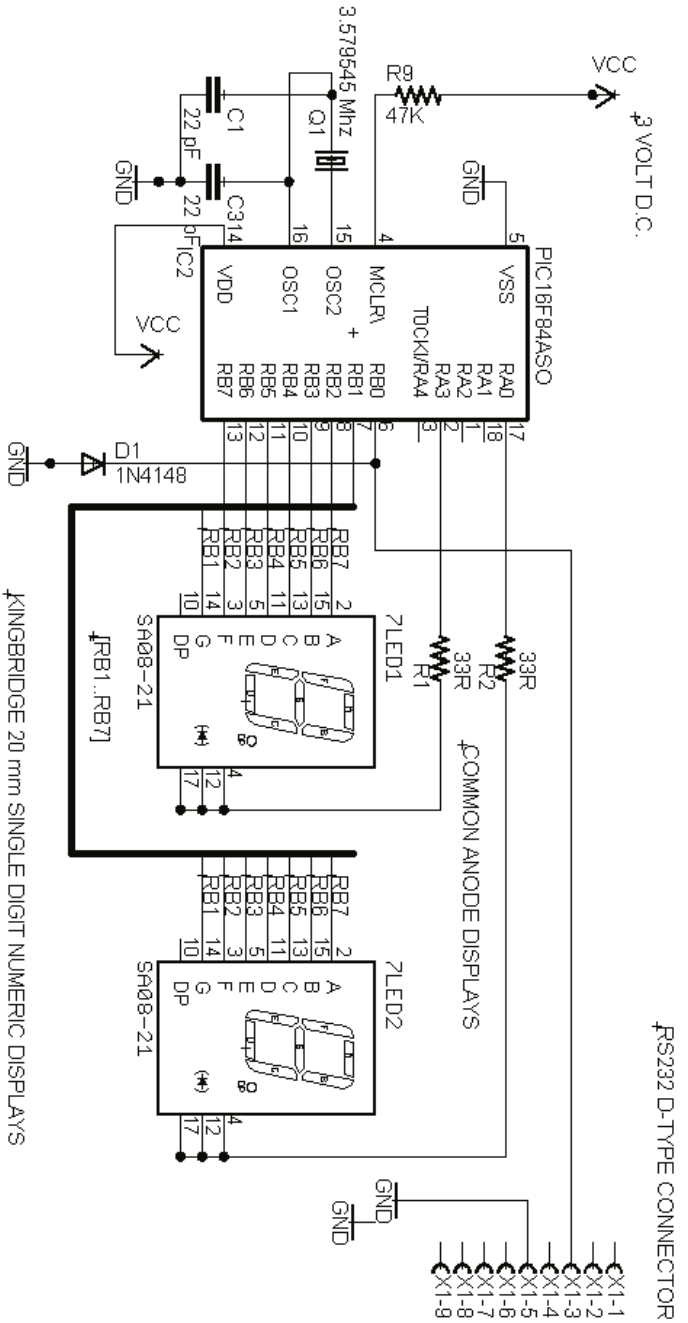
Industry standard procedures regarding static must be observed when handling product produced with blue die material.

Electrical characteristics of the display.

A PIC16F84 is used with a 3.579545 Mhz crystal and communicates with a RS232 source at 9600 baud. Port RB0 is the interrupt pin on the port and when serial data is received, the interrupt routine of the firmware reads in the data. Since this takes approx 1 millisecond, the display doesn't appear to be interrupted or to blink. The new incoming data is then displayed on the two 7-segment displays. Since the Blue displays require at least 3 volts forward drop for each LED segment, there is no need for any series limiting resistors. In this case a common 33R resistor limits the current to each display. The only disadvantage of using this method is that the display will vary slightly in brightness, depending on whether a '1' or '8' is displayed, but with the advantage of a fewer component count.



The RS232 Data Monitor



The circuit diagram of the RS232 Data Monitor.

Program DATAMONX.ASM

This program makes use of the interrupt pin RB0. Whenever serial RS232 data appears on this pin, an interrupt is issued and the program vectors to address 0004H, where the incoming byte is collected using the 'bit bang' method and stored as two bytes – HBYTE and LBYTE. The lower nibble and higher nibble of the captured data, e.g. A5H, is masked out and converted via a look-up table to two bytes (010H and 048H), which is the binary pattern required by the two 7-segment displays to display the data as A and 5. When there is no incoming data, the program loops at LABEL ALL whereby the two bytes held in HBYTE and LBYTE are continously multiplexed out to the display at a rate determined by software delay routine MUX – port lines RA3 and RA0 enabling each display in turn. This monitor is useful in the fact that it can display control codes such as carriage return (ODH) and line feed (0AH), which are transparent when transmitted and do not show on a conventional display.

```
    LIST    p=16F84

BAUD_1 EQU .28              ; 30  9600 BAUD 1 BIT PERIOD
BAUD_1ST EQU .42            ; 45  BITS

PC EQU 2
PTION EQU 081H
INTCON EQU 0BH

RTCC     EQU 1              ;RTCC
STATUS    EQU 3             ;STATUS REGISTER

DLYCNT EQU 0FH
THIRTYTWO EQU 010H      ;MODULUS 32 COUNTER

MSEC_20 EQU 013H
DB1 EQU 014H
GP  EQU 015H
DB2 EQU 016H
TONE2 EQU 017H
COUNTR   EQU 018H

LBYTE EQU 019H
HBYTE EQU 01AH
LONG EQU 01BH

SAVE EQU 01CH
SAVE2 EQU 01DH
```

```
DATABYTE EQU 01EH
BUFFER EQU 01FH
DATABYTE2 EQU 020H

LED EQU 0

PORTA EQU 5
PORTB EQU 6

LSN EQU 0
MSN EQU 1

        ORG 0
        GOTO COMMS
        ORG 4                   ;RECEIVE RS232 DATA
    BCF INTCON,1
    BCF INTCON,7
    MOVLW 8                     ;8 DATA BITS
    MOVWF COUNTR
    CLRF BUFFER
    CALL DELAY1ST

BITS
    BCF STATUS,0
    RRF BUFFER
    BTFSC PORTB,0
    BSF BUFFER,7
    CALL DELAY
    DECFSZ COUNTR,1
    GOTO BITS

    MOVLW 0FFH              ;INVERT DATA COLLECTED IN BUFFER
    XORWF BUFFER,0          ;I.E. CHANGE  FROM RS232 VOLTAGE LEVELS
                            TO TTL
    MOVWF DATABYTE
    MOVWF DATABYTE2         ;SAVE BYTE AS 2 NIBBLES (ONE FOR
                            EACH DISPLAY)
    MOVLW 0FH
    ANDWF DATABYTE,0
    MOVWF LBYTE
    SWAPF DATABYTE2,1
    MOVLW 0FH
    ANDWF DATABYTE2,0
```

```
    MOVWF HBYTE
    BCF INTCON,1
    BSF INTCON,7
    RETFIE

COMMS
    MOVLW B‘00000001’       ;INITIALISE PORTS AND ENABLE
                            RB0 INTERRUPT
    TRIS PORTB
    MOVLW 0
    TRIS PORTA
    BSF INTCON,4
    BSF PTION,6
    BSF PORTA,3             ;MSN
    BSF PORTA,0             ;LSN
    MOVLW 0FFH              ;ALL SEGMENTS OFF
    MOVWF PORTB
    MOVLW B’01000111’
    OPTION
    BSF PORTA,3             ;MSN  ON
    BSF PORTA,0             ;LSN  ON
    MOVLW 0                 ;ALL SEGMENTS ON
    MOVWF PORTB
    CALL SECOND
    MOVLW 0FFH              ;ALL SEGMENTS OFF
    MOVWF PORTB
    CALL SECOND
    BSF INTCON,7

    MOVLW 0
    MOVWF DATABYTE
XXX
    CALL ALL
    GOTO XXX

ALL
    BCF PORTA,3             ;MSN  ON      ;MULTPLEX DISPLAYS
    BSF PORTA,0             ;LSN  ON

    MOVF LBYTE,0
    CALL SEGMENTS
    MOVWF PORTB
```

```
    CALL MUX

    BSF PORTA,3     ;MSN  ON
    BCF PORTA,0     ;LSN  ON

    MOVF HBYTE,0
    CALL SEGMENTS
    MOVWF PORTB
    CALL MUX
    RETLW 0

SEGMENTS                ; LOOK UPTABLE FOR  '0 TO F'
    ADDWF PC

    RETLW 02H
    RETLW 09EH
    RETLW 024H
    RETLW 0CH
    RETLW 098H
    RETLW 048H
    RETLW 040H
    RETLW 01EH
    RETLW 0
    RETLW 018H
    RETLW 010H
    RETLW 0C0H
    RETLW 062H
    RETLW 084H
    RETLW 060H
    RETLW 070H

MUX
    MOVLW .4            ;DETERMINS RATE OF SWITCHING
                        BETWEEN DISPLAYS
    MOVWF THIRTYTWO     ;COUNTER
    CLRF RTCC           ;LOAD RTCC

SHORTM BTFSC RTCC,3     ;TEST FOR 128 DECIMAL
                        ;I.E. BIT 7 = HIGH
    GOTO JMP2M
    CLRWDT
    GOTO SHORTM
```

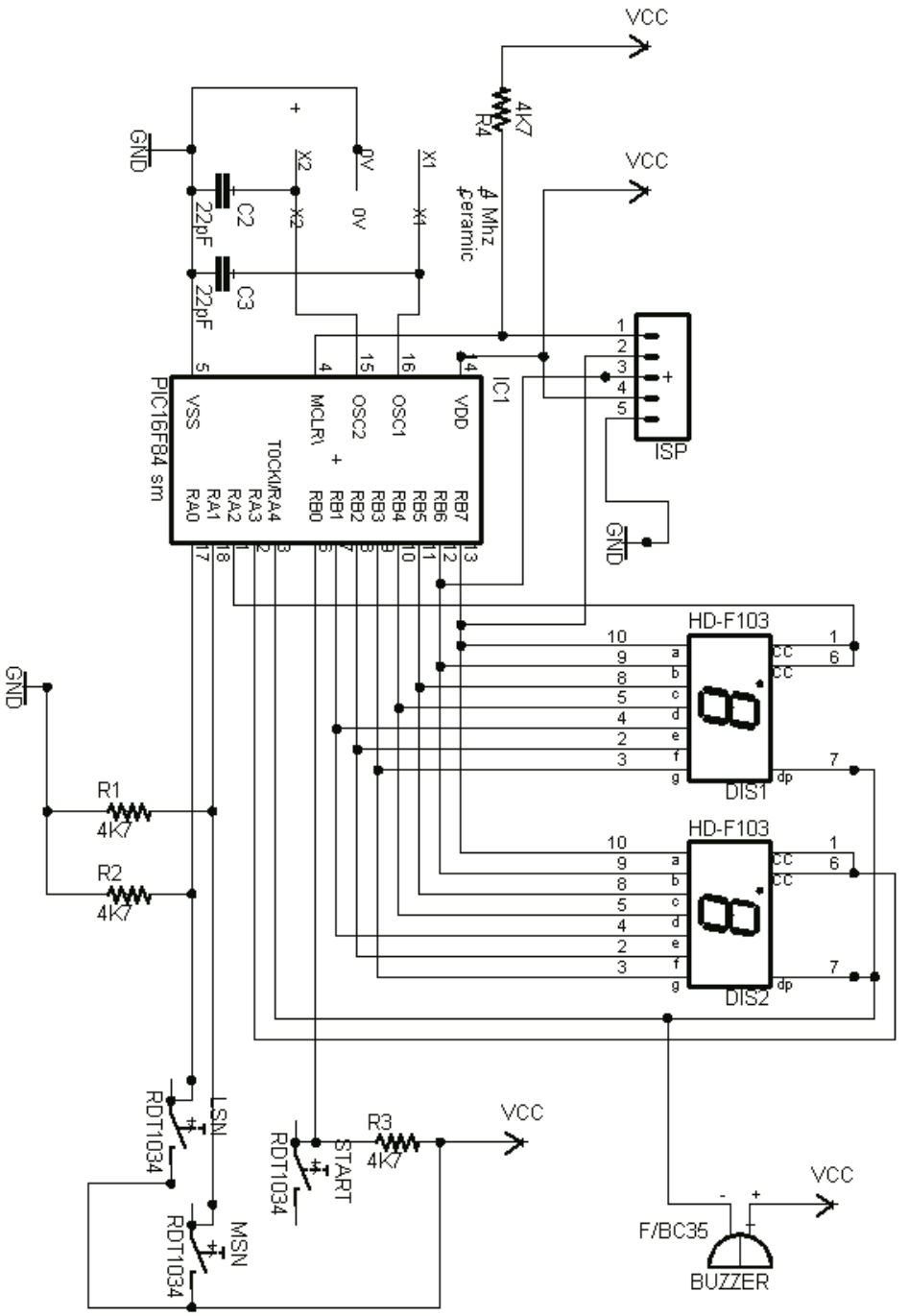
```
JMP2M CLRF RTCC
                        ;32 LOOP COUNTER
    DECFSZ THIRTYTWO,1
    GOTO SHORTM
    RETLW 0

DELAY
    MOVLW BAUD_1        ;104 MICROSECOND BIT PERIOD FOR 9600 BAUD
    MOVWF DLYCNT
REDO DECFSZ DLYCNT,1
    GOTO REDO
    RETLW 0

DELAY1ST
    MOVLW BAUD_1        ;1.5 BIT PERIODS TO CAPTURE MIDDLE
                        OF START BIT
    MOVWF DLYCNT
REDX DECFSZ DLYCNT,1
    GOTO REDX
    RETLW 0

    END
```

3.3 Case Study 00 to 99 minute programmable timer

Description of circuit

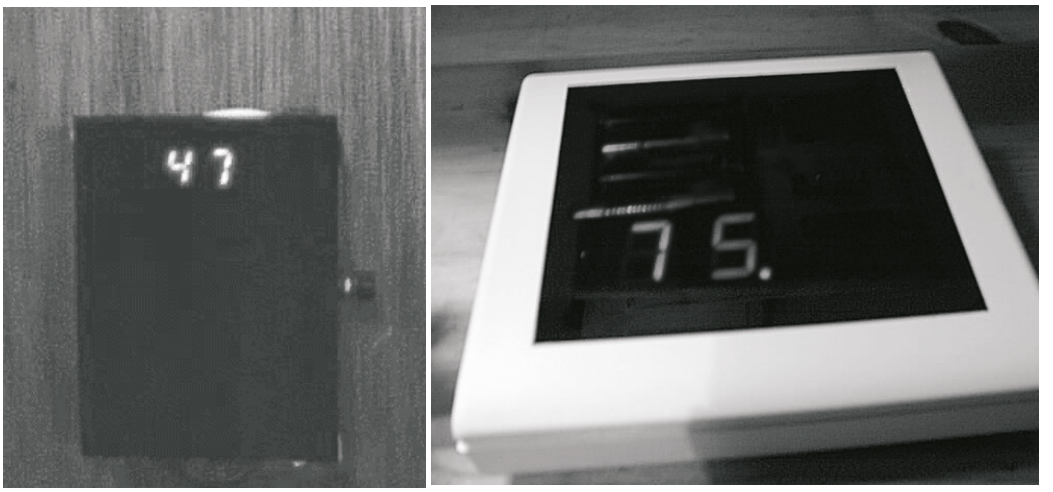
The circuit is operated by a single CR2032 3 volt lithium button cell, and because the software uses the PIC's 'Sleep Mode' when timing, battery life will be extended considerably.

To operate the timer, the START button is pressed, waking up the PIC and 2 flashing '0's are displayed. Pressing the LSN button will increment the units number from 0 to 9 at a slow rate and equaly, pressing the MSN button will also increment the tens number, so that any number between 00 and 99 minutes can be 'dialed up'. Pressing both the LSN and the MSN will cause both numbers to increment. Pressing the Start switch will blank the display, only causing the decimal points on both 7-segment displays to flash briefly every second. Concurrently a small 'tick' is issued from the buzzer. When the timer has timed out, an alarm will sound – the duration of the alarm progressively getting larger with time until the LSN button is depressed for a few seconds.



Circuit diagram of the programmable timer.

For an elegant cosmetic solution to the project, a small sheet of red plastic filter is supperglued to the two button of the control switches which are at the bottom of the circuit. Pressing the filter cover on the bottom left and right locations will increment the timer. The START switch is located at the side of the module. Red 7-segment high-efficiencey displays are used to conserve battery power. Finally, a 4.1943 MHz ceramic crystal is used for the timing and should provide great accuracy. Since surface mount components are used, an in-circuit programming header connector is provided.



The completed programmable timer.

Program Timer7.ASM

The software overhead is considerable and thus not listed here, but is available for complete download from the ELEKTOR Website.

It can be seen that a different form of display multiplexing is used in that each display is switched on briefly, the binary segment data is sent and then the segments are turned off: for example, to display the numeral THREE, first 0C0H is sent and then, after a short interval, FFH is sent.

THREE

```
BCF PORTB,1
BCF PORTB,6
BCF PORTB,4
BCF PORTB,7
BCF PORTB,5
CALL MS15
```

```
    BSF PORTB,1
    BSF PORTB,6
    BSF PORTB,4
    BSF PORTB,7
    BSF PORTB,5

    RETLW 0
```

FOUR

```
    BCF PORTB,7
    BCF PORTB,5
    BCF PORTB,6
    BCF PORTB,2
    CALL MS15

    BSF PORTB,7
    BSF PORTB,5
    BSF PORTB,6
    BSF PORTB,2

    RETLW 0
```

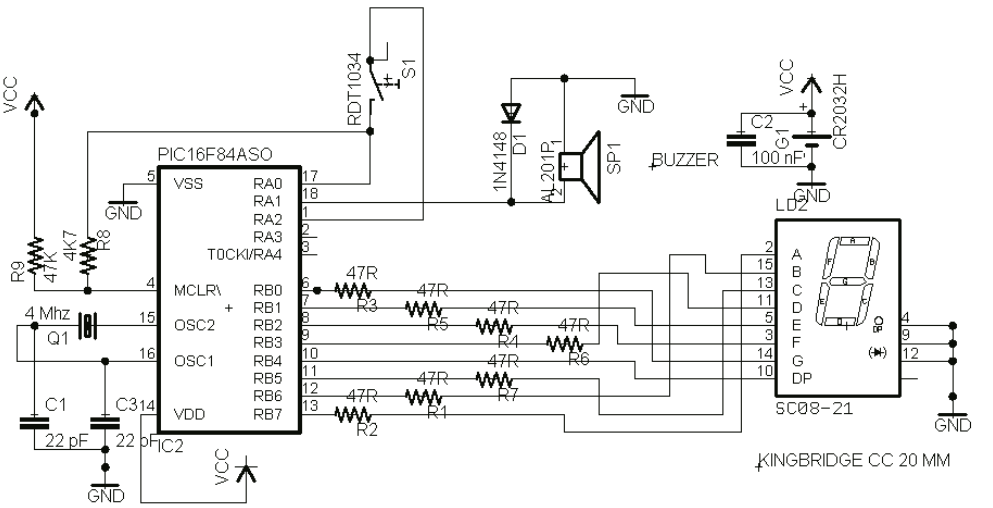
Every time the MSN button is pressed, variable HBYTE is incremented, and compared to the table shown below. If for example, the HBYTE value is now 2, CALL TWO instruction is executed and the numeral '2' appears on the L.H display. Similarly, pressing the LSN button will increment the LBYTE counter.

```
    BCF PORTA,3
    BSF PORTA,2  ; TENS

    MOVF HBYTE,0
    XORLW 0
    BTFSC STATUS,2
    CALL ZERO
    MOVF HBYTE,0
    XORLW 1
    BTFSC STATUS,2
    CALL ONE
    MOVF HBYTE,0
    XORLW 2
    BTFSC STATUS,2
    CALL TWO
```

```
MOVF HBYTE,0
XORLW 3
BTFSC STATUS,2
CALL THREE
```

After incrementing the two displays to the required time duration, pressing the START button will start the time out, the LBYTE and HBYTE containing the timeout value. Once the required time has elapsed, i.e. LBYTE and HBYTE are zero, an alarm is issued.

3.4 Case Study 4 minute egg timer

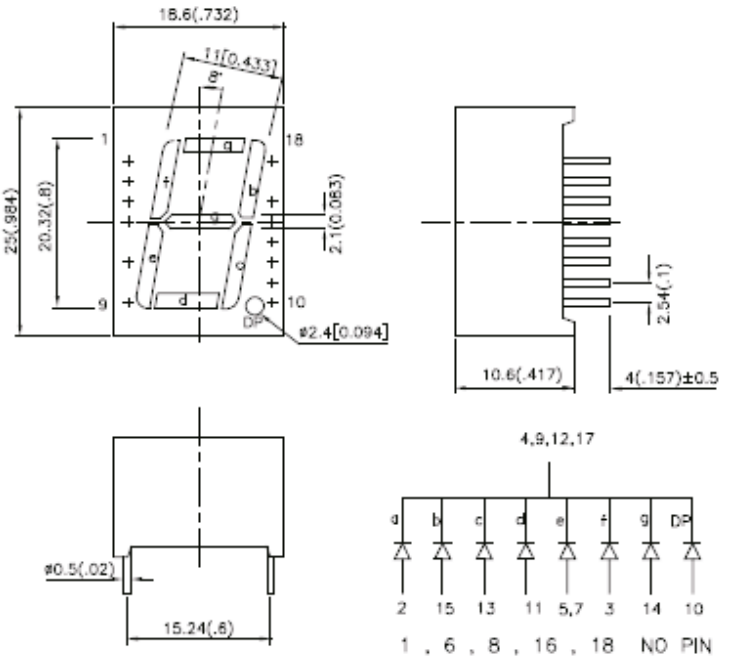


Program egg5.asm

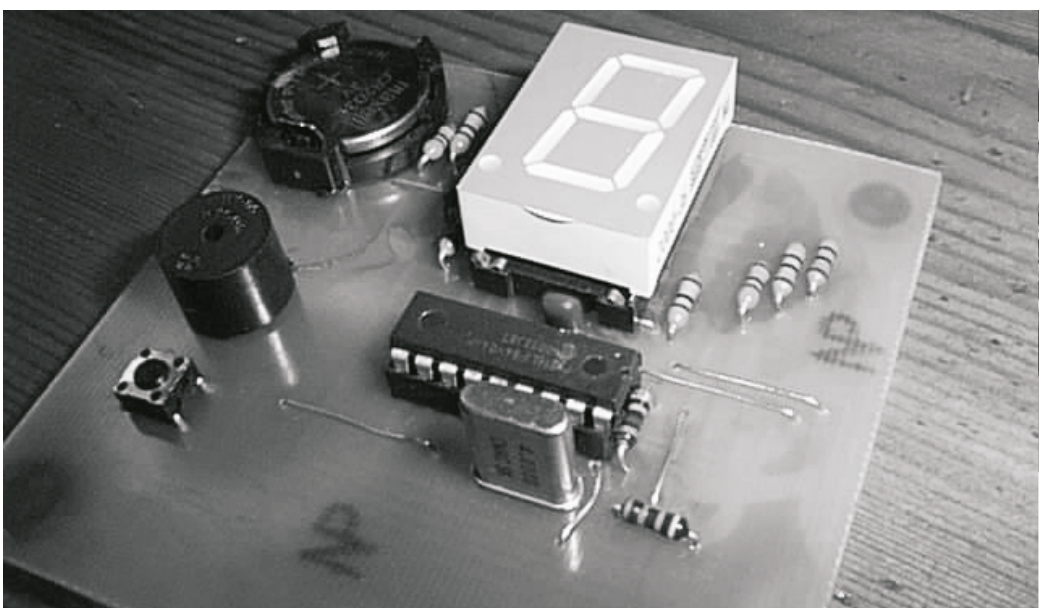
Description of circuit

This circuit illustrates the use of a large (20-mm) standard 7-segment display. It is a common cathode display and so, to light each segment, logic 1 has to be applied. All port B port pins are used so that the decimal point can flash at 1-second intervals to show that the egg timer is active. The 'wake-up from sleep' function is controlled by switch S1 (a momentary press to make switch) and associated port A line RA0 and RA2. This circuit is based loosely on Microchip's application note AN528 whereby multiple switches can be made to wake the PIC. In this case, only 1 switch is used. During the initialisation of the PIC, the SCAN1 port (RA2) is cleared low before the PIC goes to sleep. When the switch is pressed, the MCRL reset pin is connected to ground, waking up the PIC – when the switch is released, port RA0, which is an input, goes high and the main timer routine SWITCH is executed before the PIC goes back to sleep.

The diagrams below show details of the display and pin-out. A full data sheet is also included in the PDF folder of chapter 3.



Package Dimensions & Internal Circuit Diagram



4 minute egg timer.

Program egg5.asm

```
      LIST    p=16f84

RTCC       EQU 1                ;RTCC
STATUS     EQU 3                ;STATUS REGISTER
THIRTYTWO EQU 010H              ;MODULUS 32 COUNTER

BUZZ   EQU 011H
TONE   EQU 012H
MSEC_20 EQU 013H
DB1 EQU 014H
GP  EQU 015H
DB2 EQU 016H
TONE2 EQU 017H

PORTA EQU 5
PORTB EQU 6

SCAN1 EQU 2
SW1   EQU 0

        ORG 0
    MOVLW 0                     ;CONFIGURE PORTS
    TRIS  PORTB
    MOVLW 0
    MOVWF  PORTB

    MOVLW 1
    TRIS PORTA
    MOVLW 0FFH
    MOVWF PORTA
    BCF PORTA,1

    BSF PORTA,SCAN1
KEYON

    CALL DELAY
    BCF PORTA,SCAN1

    CLRWDT
    BTFSC PORTA,0
    GOTO CONT                   ;POWER ON RESET, AND THEN GOTO SLEEP
```

```
    BSF PORTA,SCAN1      ;DISABLE RESET I.E. SCAN1 LINE IS HIGH

    CALL SWITCH          ;SWITCH HAS BEEN PRESSED

DEBOUNCE                 ;DEBOUNCE SWITCH
    BSF PORTA,SCAN1
    CALL DELAY
    BCF PORTA,SCAN1
    CLRWDT
    BTFSS PORTA,0
    GOTO DEBOUNCE

CONT

    BCF PORTA,SCAN1      ;SCAN LINE LOW

    MOVLW 0
    MOVWF PORTB
    BCF PORTA,1
    BCF PORTA,3
    SLEEP

DELAY                    ;TO DEBOUNCE SWITCH
    MOVLW MSEC_20
    MOVWF DB1

DLY1
    CLRWDT
    CLRF DB2
    DECFSZ DB1
    GOTO DLY2
    RETLW 0

DLY2
    CLRWDT

    DECFSZ DB2
    GOTO DLY2
    GOTO DLY1

SWITCH                   ;WAIT 3,4 OR 5 MINUTES AND THEN ISSUE ALARM

    MOVLW B‘00000111’    ; PRESCALER 00000111 (/256)
```

```
OPTION
CLRF RTCC
CLRWDT
CALL ALARM          ; ISSUE 1 BEEP  TO SIGNIFY COUNTDOWN
                    HAS STARTED
CALL QUARTER

BSF PORTB,1         ;LIGHT/EXTINGUISH EACH LED SEGMENT IN TURN
CALL QUARTER
BCF PORTB,1
BSF PORTB,7
CALL QUARTER
BCF PORTB,7
BSF PORTB,5
CALL QUARTER
BCF PORTB,5
BSF PORTB,3
CALL QUARTER
BCF PORTB,3
BSF PORTB,0
CALL QUARTER
BCF PORTB,0
BSF PORTB,2
CALL QUARTER
BCF PORTB,2
BSF PORTB,4
CALL QUARTER
BCF PORTB,4
BSF PORTA,6
CALL QUARTER

BCF PORTA,6
BSF PORTB,6
BSF PORTB,7
BSF PORTB,5

CALL SECOND
CALL SECOND

BCF PORTB,2
BCF PORTB,6
BCF PORTB,7
BCF PORTB,5
```

```
    CALL MIN            ;5 MINUTES TO BOIL EGG
    CALL MIN            ;DELETE FOR SHORTER TIME
    CALL MIN            ;DITTO
    CALL MIN
    CALL MIN
    MOVLW 5             ;DURATION OF ALARM
    MOVWF TONE
NN
    BSF PORTB,1         ;ISSUE ALARM
    CALL ALARM
    CALL QUARTER
    BSF PORTB,7
    CALL ALARM
    CALL QUARTER
    BSF PORTB,5
    CALL ALARM
    CALL QUARTER
    BSF PORTB,3
    CALL ALARM
    CALL QUARTER
    BSF PORTB,0
    CALL ALARM
    CALL QUARTER
    BSF PORTB,2
    CALL ALARM
    CALL QUARTER
    CALL QUARTER

    BCF PORTB,1
    BCF PORTB,7
    BCF PORTB,5
    BCF PORTB,3
    BCF PORTB,0
    BCF PORTB,2

    CALL QUARTER

    DECFSZ TONE,1
    GOTO NN
    RETLW 0             ; RETURN TO SLEEP
```

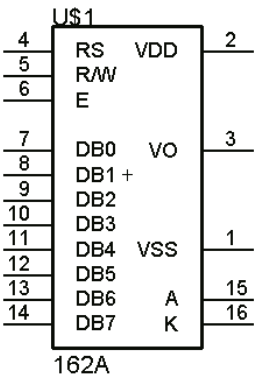
```
ALARM
    MOVLW 0FFH              ; 2 TONE ALARM
    MOVWF TONE2
JJ
    MOVLW 042H              ; TONE 1
    MOVWF BUZZ
    BSF PORTA,1
GG  DECFSZ BUZZ,1
    GOTO GG
    BCF PORTA,1
    MOVLW 022H              ; TONE 2
    MOVWF BUZZ
HH  DECFSZ BUZZ,1
    GOTO HH
    DECFSZ TONE2,1
    GOTO JJ

    RETLW 0

MIN MOVLW .60               ;1 MINUTE DELAY
    MOVWF TONE

COUNT  BSF  PORTB,4         ;DECIMAL POINT
    MOVLW .30               ;32
    MOVWF THIRTYTWO         ;COUNTER
    CLRF RTCC               ;CLEAR  RTCC

AHORT BTFSC RTCC,7          ;TEST FOR 128 DECIMAL
                            ;I.E. BIT 7 = HIGH
    GOTO JMP3
    CLRWDT
    BTFSC RTCC,6
    BCF PORTB,4

    GOTO AHORT
JMP3 CLRF RTCC
                            ;32 LOOP COUNTER
    DECFSZ THIRTYTWO,1
    GOTO AHORT
    DECFSZ TONE,1
    GOTO COUNT
```

```
    RETLW 0

SECOND                      ;1 SECOND DELAY
BCF  PORTB,4                ;DECIMAL POINT ON
    MOVLW .30               ;32
    MOVWF THIRTYTWO         ;COUNTER
    CLRF RTCC               ;LOAD RTCC

SHORT BTFSC RTCC,7          ;TEST FOR 128 DECIMAL
                            ;I.E. BIT 7 = HIGH
    GOTO JMP2
    CLRWDT
    GOTO SHORT

JMP2 CLRF RTCC
                            ;32 LOOP COUNTER
    DECFSZ THIRTYTWO,1
    GOTO SHORT
    RETLW 0

QUARTER                     ;DETERMINS FREQUENCY OF ALARM
    MOVLW .2
    MOVWF THIRTYTWO         ;COUNTER
    CLRF RTCC               ;LOAD RTCC

BHORT BTFSC RTCC,7          ;TEST FOR 128 DECIMAL
                            ;I.E. BIT 7 = HIGH
    GOTO JMP5
    CLRWDT
    GOTO BHORT

JMP5 CLRF RTCC

    DECFSZ THIRTYTWO,1
    GOTO BHORT
    BSF  PORTB,4            ;DECIMAL POINT OFF
    RETLW 0
    END
```

4 B/W Liquid Crystal Displays

These displays have been with us for at least 25 years and are still in use today. They include intelligent alphanumeric dot matrix modules with an integral CMOS microprocessor and LCD display drivers. They utilise a 5 × 8 dot matrix font format with cursor and are capable of displaying 189 different alphanumeric characters and symbols as well as user defined 5 × 8 graphics. They are available in standard sizes of 16, 20 and 40 characters / line with a choice of 1, 2 or 4 lines, giving a maximum of 160 characters on the display and are all controlled by the LCD controller chip such as the HD44780. Power consumption is very low, typically a few milliamps, and 5 volt and 3 volt versions are available leading them to be an ideal choice for portable applications. In addition, backlit versions are available to improve readability.

The displays are universally used in many applications including vending machines, EPOS e.g. chip and pin machines, PDAs e.g. the original Psion data organiser, portable instrumentation including monitoring and control and a vast myriad of consumer products.



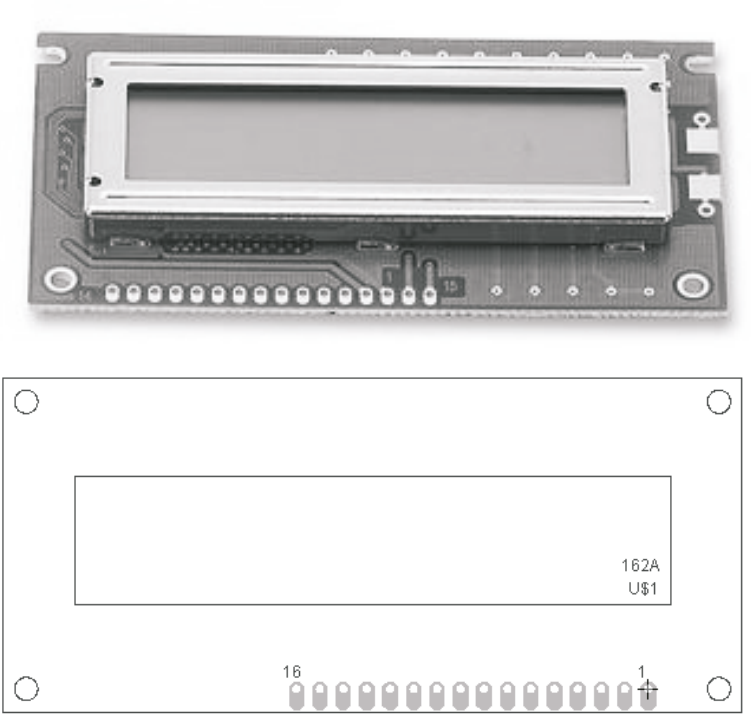
LCD circuit Symbol

Pin no.	Symbol	External connection	Function
1	V_{SS}	Power supply	Signal ground for LCM (GND)
2	V_{DD}		Power supply for logic (+5V) for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL "A" (+4.2V)
16	LED-		Power supply for BKL "K" (GND)

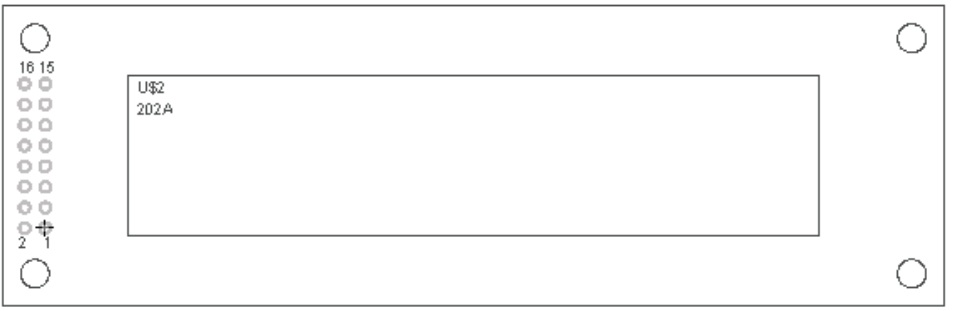
Interface pin description.

4.1 Industry standard alphanumerical LCD displays

The table above shows the interface requirements for a standard display. Only two control pins are used (RS and E) since the R/W pin can be permanently grounded. It can be seen that pins 15 and 16 supply the power to the integral backlight if it is provided. There are basically two types of display – one with the connections in 1 row and the other where two vertical rows of connections are used:



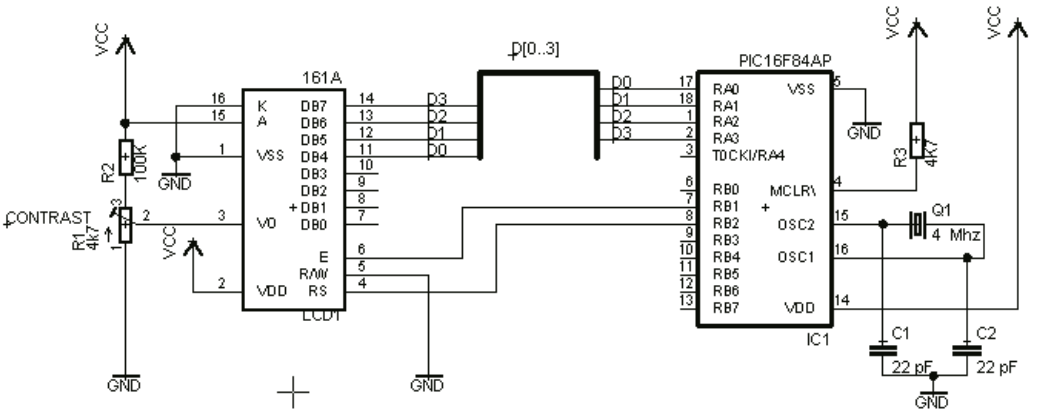
Standard LCD display (2 × 16)



2-row connections.

A 4 line x 40 character display.

We can now build a simple test circuit utilising a 2 line by 16-character display and a PIC16F84 to show how the LCD works:



LCD test circuit for Program LCDX.ASM.

LCDs have an 8-bit parallel data bus, (DB0 to DB7) but in order to reduce pin count, only the high order nibble (DB4 to DB7) is used in the following case studies.

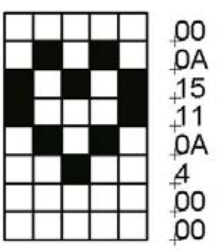
ASCII data is sent to the display as two nibbles over the 4-bit data bus (D0-D3). Port A pins RA0 to RA3 carry the ASCII data to the display. For example, for the letter 'A' (41H) to be displayed, it would be sent as 04H and 01H. Control lines E and RS on the display are used to send data or commands. When RS is low, commands are sent. With RS high, data can be sent to the display. The E pin enables the LCD.

Program LCDX.ASM

This program displays the following message:

HULLO ♥ ♥

That consists of standard ASCII characters and two graphics images (a heart and inverted heart shape). A total of 8 user defined 5 × 8 programmable shapes are available starting at character graphics memory address 00H, and are contiguous in memory. The graphics figure below shows how the heart is constructed by sending 8 bytes in the program.



Sub-routine **INIT_DISPLAY** initialises the LCD, sub-routine **GRAPHICS** writes the bit pattern for the two hearts and routine MESSAGE displays the message on the 2nd line of the display.

Routine '**COMMAND**' clears the RS pin, allowing various commands to be sent to the display including position cursor and clear display. '**WRITE**' sets the RS pin and ASCII data can now be sent.

The internal data memory consists of 80 bytes held in a circular-rotating buffer, with only a 'window' of 16, 20 or 40 bytes being displayed dependant on the display size. Sending a command to shift (rotate) the display, either left or right, will cause succeeding characters to be 'lost' as the memory is rotated. Sending command '80H' will select the 1st line of the display containing 40 bytes (half the available display memory). Selecting command 'A0H' will allow data to be written to the 2nd line. In the case of a 1-line display, all of the available 80 bytes of character are available in one continuous buffer.
The table below summarises the available command set for a standard LCD :

Instruction	Instruction code										Description	Execution Time (fosc= 270 KHZ
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRA and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" From AC and return cursor to Its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display	39us
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.	
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.	39us
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.	39us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39us
Read busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal Operation or not can be known By reading BF. The contents of Address counter can also be read.	0us
Write data to Address	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data From RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

Program LCDX.ASM

```
    LIST p=16F84

E EQU 1; RB1 LCD
RS EQU 2; RB2 LCD
RTCC EQU 1; RTCC
PC EQU 2
STATUS EQU 3; STATUS REGISTER
PORT_A EQU 5; PORT A LCD DATA BUS
PORT_B EQU 6; PORT B CONTROL/MONITOR
A_N EQU 0DH; ALPHANUMERIC COUNTER
DLYCNT EQU 012H
LCDDATA EQU 014H

    ORG 0

    MOVLW 0;                ;  INITIALISE PORTS AS OUTPUTS
    TRIS PORT_A
    MOVLW 0
    TRIS PORT_B

    BCF PORT_B, RS
    BCF PORT_B, E
    MOVLW 0FH
    MOVWF PORT_A

    CALL INIT_DISPLAY       ; MAIN PROGRAM
    CALL GRAPHICS
    CALL MESSAGE

    SLEEP

INIT_DISPLAY
    CALL LDELAY

    MOVLW 033H              ;SEND COMMANDS FOR 4 BIT LCD DATA BUS
    CALL COMMAND
    MOVLW 032H
    CALL COMMAND
    MOVLW 028H
    CALL COMMAND
    MOVLW 6
```

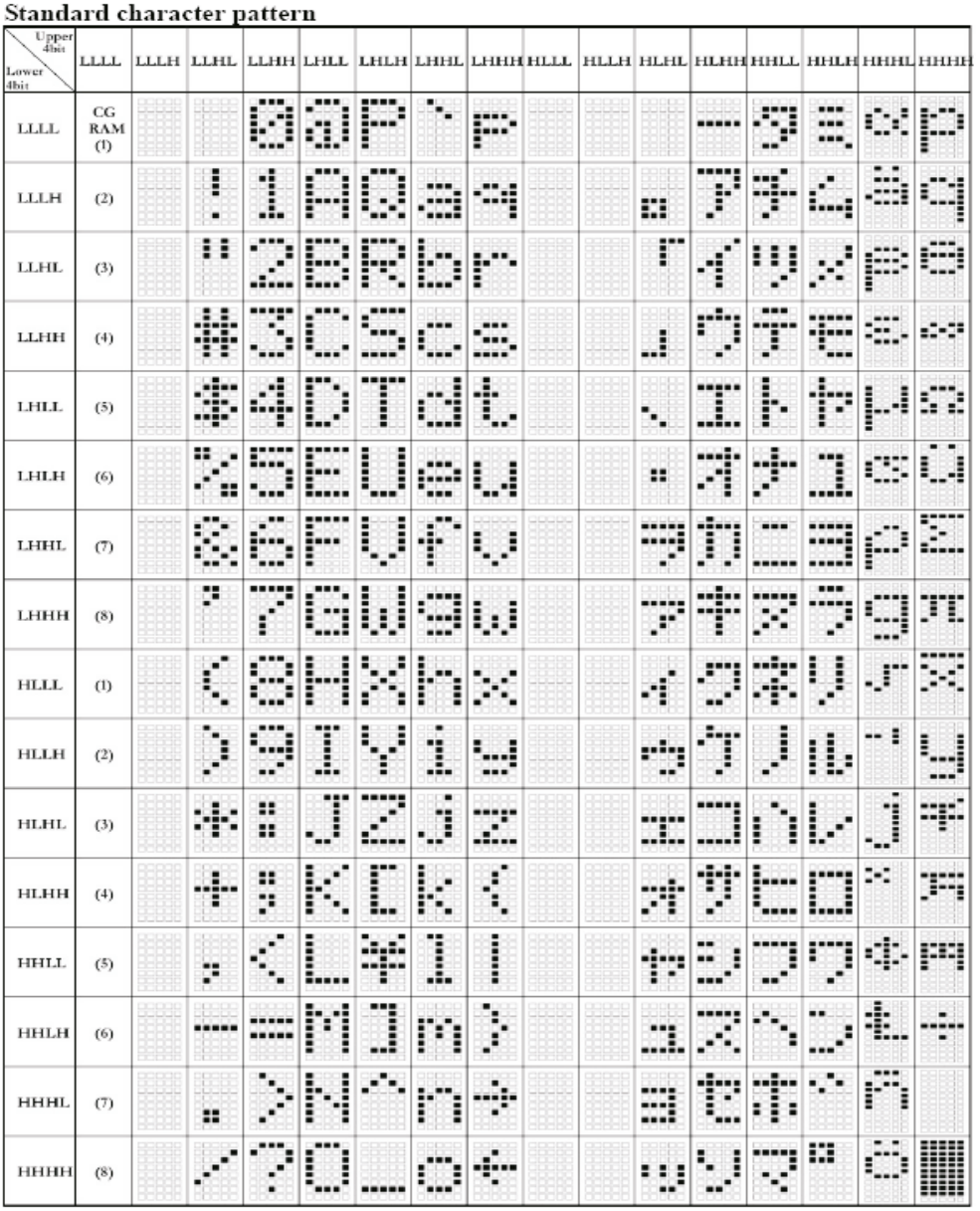
```
    CALL COMMAND
    MOVLW 0EH
    CALL COMMAND
    MOVLW 1
    CALL COMMAND
    CALL LDELAY
    RETLW 0

COMMAND
    MOVWF LCDDATA       ; SEPARATE COMMAND BYTE TO HIGH AND LOW
                        ; NIBBLES AND SEND TO DISPLAY
    SWAPF LCDDATA, 0    ; MSNIBBLE
    MOVWF  PORT_A       ;SEND DATA TO PORT A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E

    MOVF LCDDATA, 0     ; LSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E
    RETLW 0

WRITE
    MOVWF A_N           ;WRITE  DATA TO DISPLAY
    SWAPF A_N, 0        ;DIVIDE BYTE INTO 2 NIBBLES AND SEND
                        TO DISPLAY
    MOVWF  PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E

    MOVF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E
    RETLW 0
```

```
GRAPHICS
    MOVLW 040H
    CALL COMMAND

    MOVLW 0                 ;  HEART SHAPE -ADDRESS 0
    CALL WRITE              ; 8 CODE BYTES FOR GRAPHIC IMAGE
    MOVLW 0AH
    CALL WRITE
    MOVLW 015H
    CALL WRITE
    MOVLW 011H
    CALL WRITE
    MOVLW 0AH
    CALL WRITE
    MOVLW 4
    CALL WRITE
    MOVLW 0
    CALL WRITE
    MOVLW 0
    CALL WRITE
    MOVLW 0                 ; INVERTED HEART INVERTED- ADDRESS 1
    CALL WRITE
    MOVLW 0AH
    CALL WRITE
    MOVLW 01FH
    CALL WRITE
    MOVLW 01FH
    CALL WRITE
    MOVLW 0EH
    CALL WRITE
    MOVLW 4
    CALL WRITE
    MOVLW 0
    CALL WRITE
    MOVLW 0
    CALL WRITE

    RETLW 0

MESSAGE

    MOVLW 0A9H; 2ND LINE
    CALL COMMAND
```

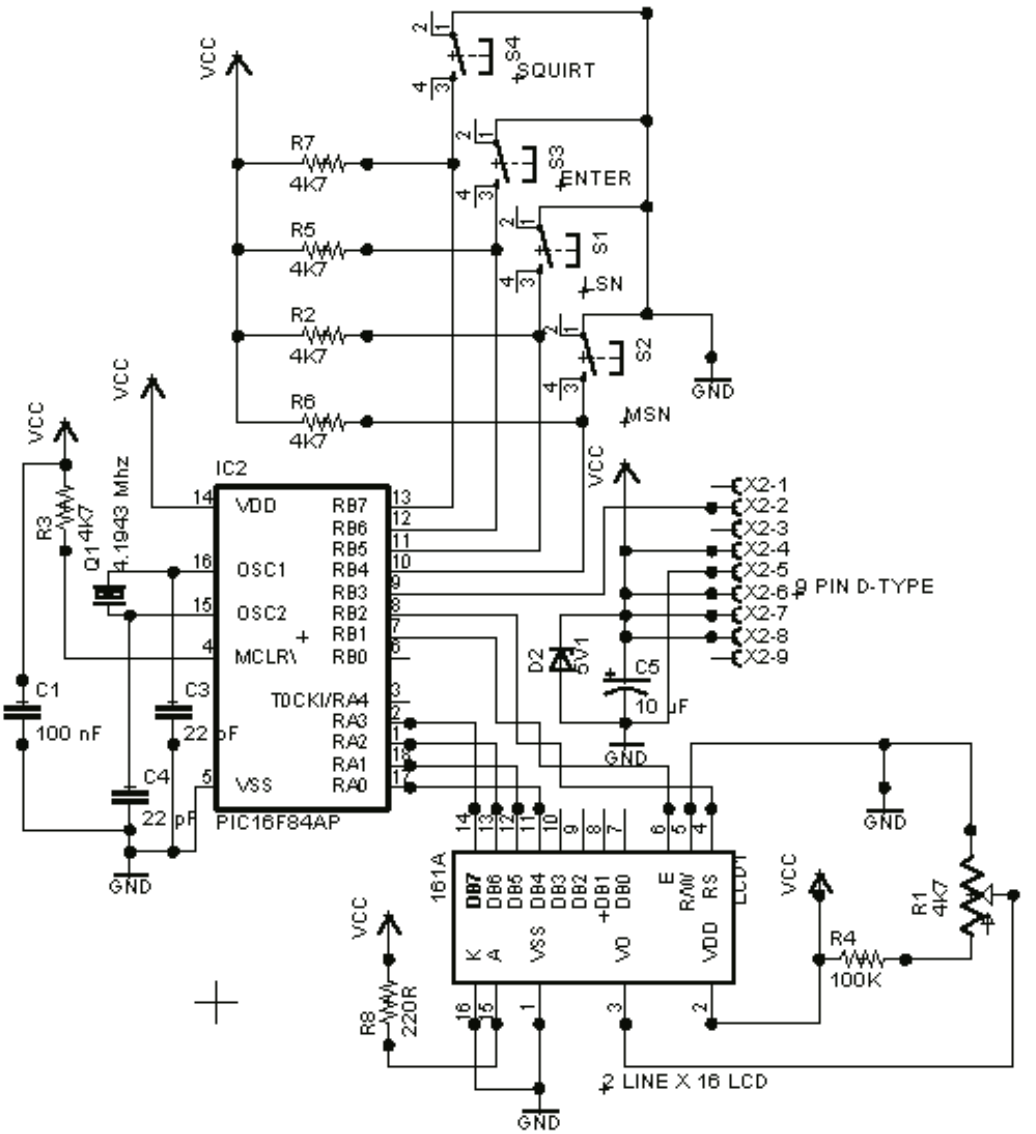
```
    MOVLW ‘H’           ;MESSAGE ‘HULLO’
    CALL WRITE
    MOVLW ‘U’
    CALL WRITE
    MOVLW ‘L’
    CALL WRITE
    MOVLW ‘L’
    CALL WRITE
    MOVLW ‘O’
    CALL WRITE

    MOVLW ‘ ‘; SPACE
    CALL WRITE
    MOVLW 0             ; SEND HEART TO DISPLAY ADDRESS 0
    CALL WRITE
    MOVLW 1             ;SEND  HEART INVERTED TO DISPLAY ADDRESS 2
    CALL WRITE

DELAY_DATA              ; APPROX. 500 MICROSECONDS DELAY FOR WRITE
                                MOVLW 0FFH          ; COMMAND
    MOVWF DLYCNT
REDB DECFSZ DLYCNT, 1
    GOTO REDB
    RETLW 0
LDELAY                  ;LONG DELAY FOR LCD INITIALISATION FOR
                        ; COMMAND- CLEAR DISPLAY AND HOME CURSOR
    MOVLW B‘00000011’
    OPTION
    CLRF RTCC
L_DELAY
    BTFSS RTCC, 7
    GOTO L_DELAY
    RETLW 0

    END
```

The standard character pattern for LCD displays follows the standard ASCII character set, both upper and low case, but with additional Greek and Japanese characters.

Standard character pattern



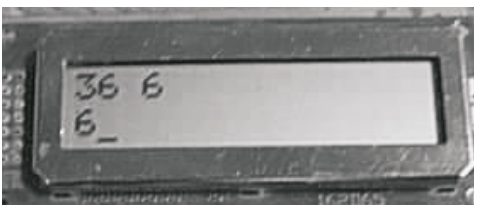
4.2 Case Study ASCII string generator



Circuit diagram of the ASCII string generator.

Sometimes an additional source of RS232 data is required, so instead of purchasing another PC, this project enables the reader to generate an ASCII string source. Based on a 16 × 2 line LCD, it connects to the main PC via the COM1 or COM2 serial port.

Two press-to-make switches, LSN and MSN, are used to program the required ASCII character. These switches select the least significant nibble and most significant nibble by incrementing the data on the display from 0 to F when the switch is pressed. For example, selecting '3' with the MSN switch and '6' with the LSN switch will produce the ASCII code for the number '6'. The numeral '6' is then displayed on the second line of the display. Successive characters are selected by pressing the 'ENTER' button. In this way, up to 32 characters are 'dialled-up'. Pressing the 'SQUIRT' button will send the ASCII string at 9600 baud from the 9 pin serial port on PIN 2 at 1-second intervals.



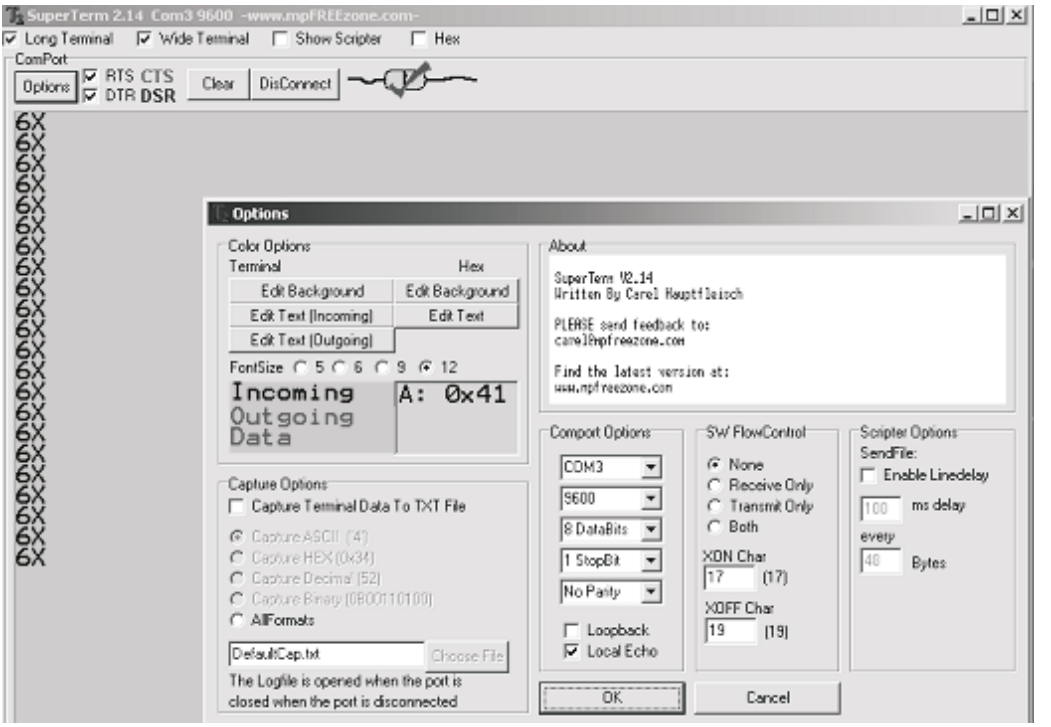
'6' dialed (36H)

'X' dialed (58H)

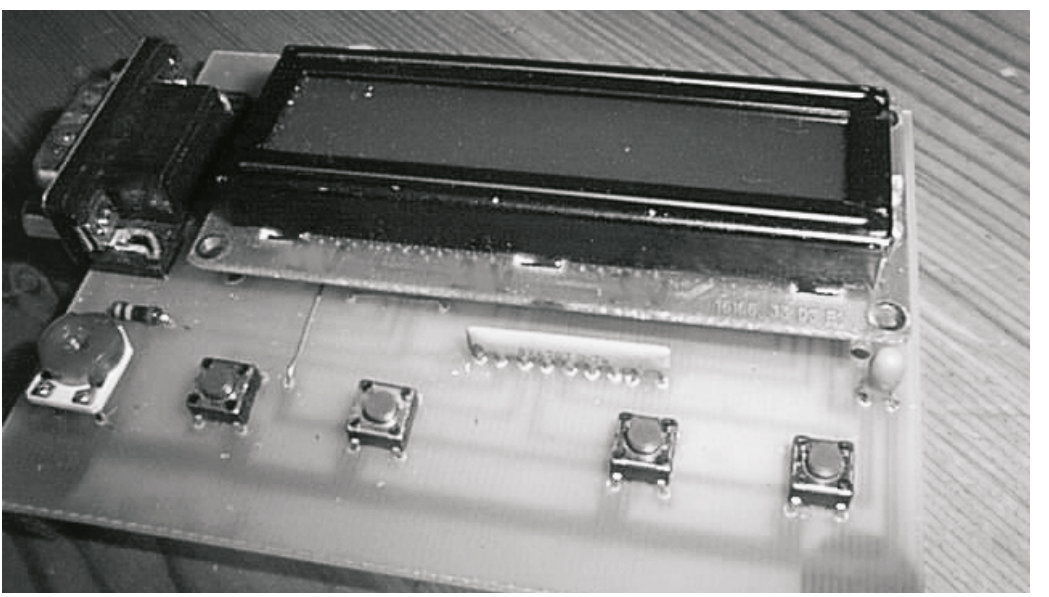
Carriage return (0DH)

Line feed (0AH) – signified by a black square.

The ASCII string '6X, carriage return, line feed' is transmitted, with a flashing cursor on the second line indicating at one second intervals. If we now run a terminal emulation program such as 'SuperTerm' on the PC, we shall see the following:



Ensure that the RTS and DSR hardware handshake lines are enabled, since the power from these lines is taken for the generator.



The completed project.

Program ASCII_STRING_GEN

Many of the routines in this program are identical to the ones in Program LCDX.ASM: INIT_DISPLAY,COMMAND,WRITE

(a)

```
            LIST p=16F84
E EQU 1             ;  RB1 LCD
RS EQU 2            ; RB2 LCD
TXD EQU 3           ; RB3 TXD
MSN EQU 4           ; RB4 SW
LSN EQU 5           ; RB5 SW
ENTER EQU 6         ; RB6 SW
SQUIRT EQU 7        ; RB7  SW
PORT_A EQU 5        ; PORT A LCD DATA
                    ;BUS
PORT_B EQU 6        ; PORT B
;CONTROL/MONITOR
DEBOUNCE EQU 0FFH
INDF EQU 0
RTCC EQU 1          ; RTCC
PC EQU 2
STATUS EQU 3        ; STATUS REGISTER
FSR EQU 4
BAUD9600 EQU .33
COUNTER EQU 0CH     ; ADDRESS
                    ;COUNTER
A_N EQU 0DH         ; ALPHANUMERIC
                    ;COUNTER
ADDRESS EQU 0EH     ; LCD
COUNTR EQU 0FH
BUFFER EQU 010H
DATABYTE EQU 011H

STRING EQU 012H
ASCII EQU 013H
LCDDATA EQU 014H
DLYCNT EQU 015H
LBYTE EQU 016H
HBYTE EQU 017H
SLOW EQU 018H
NUMBER EQU 019H
CHARS EQU 01AH
```

(b)

```
        ORG 0
    MOVLW 0
    TRIS PORT_A
    MOVLW B'11110000'
    TRIS PORT_B
    BCF PORT_B, TXD
    BCF PORT_B, RS
    BCF PORT_B, E
    MOVLW 0FH
    MOVWF PORT_A

CYCLEALL
    CALL INIT_DISPLAY
    CALL GRAPHICS       ;CR SYMBOL
    CALL SELECTINIT
    CALL SECOND
    CALL SECOND

    CALL INIT_DISPLAY

NEXTDATA
    MOVLW 0C0H; 080H
    MOVWF STRING
    MOVLW 0
    MOVWF LBYTE
    MOVLW 0
    MOVWF HBYTE

    MOVLW 020H      ;INDEX FOR
                    ;32 BYTES
    MOVWF FSR
    MOVLW 0
    MOVWF NUMBER
```

(c)

```
INC_DATA          ;POLL ALL SWITCHES

    BTFSS PORT_B, LSN
    CALL GETLOW
    BTFSS PORT_B, MSN
    CALL GETHIGH
    BTFSS PORT_B, ENTER
    CALL PRINTBYTE
    BTFSS PORT_B, SQUIRT
    GOTO SEND
    GOTO INC_DATA

PRINTBYTE
    MOVF STRING, 0
    CALL COMMAND
    CALL LONGTIME
    MOVF BUFFER, 0
    XORLW  0DH   ;ASCII FOR CR
    BTFSC STATUS, 2
    GOTO CARIAGERETURN
    GOTO VV
CARIAGERETURN

    MOVLW 0  ;ADDRESS 0 GRAPHICS
    CALL WRITE
    GOTO WW
VV MOVF BUFFER, 0
    CALL WRITE
WW

    INCF STRING, 1
    MOVF BUFFER, 0
    MOVWF INDF
    INCF FSR, 1
    INCF NUMBER, 1
    RETLW 0

GETLOW            ; DIAL UP NUMBER
    INCF LBYTE, 1;INCREMENT BYTE
    MOVF LBYTE, 0    ;TO SELECT
    XORLW 020H
    BTFSS STATUS, 2
    GOTO AA
    MOVLW 0
    MOVWF LBYTE    ;SAVE
AA
    CALL LDELAY
```

(d)

```
    CALL GETBYTE
    NOP
    RETLW 0

GETHIGH       ;INCREMENT NUMBER
    INCF HBYTE, 1      ;TO SELECT
    MOVF HBYTE, 0
    XORLW 020H
    BTFSS STATUS, 2
    GOTO BB
    MOVLW 0
    MOVWF HBYTE   ;SAVE
BB
    CALL LDELAY
    CALL GETBYTE
    NOP
    RETLW 0

GETBYTE ;CONVERT SELECTED BYTE
    MOVLW 080H  ;TO 2 ASCII NIBBLES
    CALL COMMAND

    MOVLW 0FH
    ANDWF HBYTE, 0
    CALL BINARY ;CONVERT TO ASCII
    CALL WRITE

    MOVF LBYTE, 0
    CALL BINARY  ;CONVER TO ASCII
    CALL WRITE

    MOVLW ' '       WRITE  SPACE
    CALL WRITE
    SWAPF HBYTE, 0
    ADDWF LBYTE, 0
    MOVWF BUFFER
    CALL WRITE

    CALL LONGTIME
    RETLW 0

SEND
    MOVF NUMBER, 0      ; SAVE
                        ; NUMBER OF
                        ;CHARS
    MOVWF CHARS
```

(e)

```
SEND2
    MOVF CHARS, 0
    MOVWF NUMBER
    MOVLW 020H ;NUMBER OF BYTES
    MOVWF FSR

ALLSTRING   ;COLLECT EACH BYTE
            ;USING INDEX POINTER
    CALL DELAY_DATA
    MOVF INDF, 0      ; GET DATA
    CALL CONVERT
    CALL TXD_DATA ;AND TRANSMIT
    INCF FSR, 1
    DECFSZ NUMBER, 1

    GOTO ALLSTRING
    MOVLW 0C0H
    CALL COMMAND
    MOVLW 0DH
    CALL COMMAND
    CALL SECOND

    BTFSC PORT_B, LSN ;POLL LSN SW.
    GOTO SEND2  ;IF PRESSED,RESET
    GOTO CYCLEALL

INIT_DISPLAY
    CALL LDELAY

    MOVLW 033H
    CALL COMMAND
    MOVLW 032H
    CALL COMMAND

    MOVLW 028H
    CALL COMMAND
    MOVLW 6
    CALL COMMAND
    MOVLW 0EH
    CALL COMMAND
    MOVLW 1
    CALL COMMAND
    CALL LDELAY
    RETLW 0
```

(f)

```
SELECTINIT
    MOVLW 080H      ; 1ST LINE 8TH;
    CALL COMMAND

    MOVLW 07EH
    CALL WRITE
    MOVLW 'R'     ; READY MESSAGE
    CALL WRITE
    MOVLW 'e'
    CALL WRITE
    MOVLW 'a'
    CALL WRITE
    MOVLW 'd'
    CALL WRITE
    MOVLW 'y'
    CALL WRITE
    RETLW 0

COMMAND         ;WRITE COMMAND
                ;TO LCD
    MOVWF LCDDATA
    SWAPF LCDDATA, 0
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E
    MOVF LCDDATA, 0     ; LSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E
    RETLW 0

WRITE     ;WRITE DATA TO DISPLAY
    MOVWF A_N
    SWAPF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E
    MOVF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
```

(g)

```
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E
    RETLW 0

TXD_DATA        ;TRANSMIT DA TA @
                ;9600 BAUD
    BSF PORT_B, TXD
NEXT CALL DELAY
    RRF BUFFER, 1
    BTFSC STATUS, 0
    BSF PORT_B, TXD

    BTFSS STATUS, 0
    BCF PORT_B, TXD
    DECFSZ COUNTR, 1
    GOTO NEXT
    CALL DELAY
    BCF PORT_B, TXD
    CALL DELAY
    CALL DELAY
    RETLW 0

CONVERT     ; TTL TO RS232 LEVEL
    XORLW 0FFH
    MOVWF BUFFER
    MOVLW 8
    MOVWF COUNTR
    RETLW 0

DELAY
    MOVLW BAUD9600
    MOVWF DLYCNT
REDX DECFSZ DLYCNT, 1
    GOTO REDX
    NOP
    RETLW 0
```

(h)

```
LDELAY
    MOVLW B'00000011'
    OPTION
    CLRF RTCC
L_DELAY
    CLRWDT
    BTFSS RTCC, 7
    GOTO L_DELAY
    RETLW 0

DELAY_DATA
    MOVLW DEBOUNCE
    MOVWF DLYCNT
REDB DECFSZ DLYCNT, 1
    Goto REDB
    CLRWDT
    RETLW 0

BINARY      ; ASCII LOOK-UP TABLE
    ADDWF PC
    RETLW '0'
    RETLW '1'
    RETLW '2'
    RETLW '3'
    RETLW '4'
    RETLW '5'
    RETLW '6'
    RETLW '7'
    RETLW '8'
    RETLW '9'
    RETLW 'A'
    RETLW 'B'
    RETLW 'C'
    RETLW 'D'
    RETLW 'E'
    RETLW 'F'
```

(i)

```
LONGTIME
                ; 1000.23 MS = 32
    MOVLW 8
    MOVWF SLOW
    MOVLW B'00000111'
    OPTION
LONGX CLRF RTCC
LONG CLRWDT
    NOP
    BTFSS RTCC, 7
    GOTO LONG
    DECFSZ SLOW, 1
    GOTO LONGX
    RETLW 0

SECOND
                ; 1000.23 MS = 32
    MOVLW .32
    MOVWF SLOW
    MOVLW B'00000111'
    OPTION
LONGA CLRF RTCC
LON CLRWDT
    NOP
    BTFSS RTCC, 7
    GOTO LON
    DECFSZ SLOW, 1
```

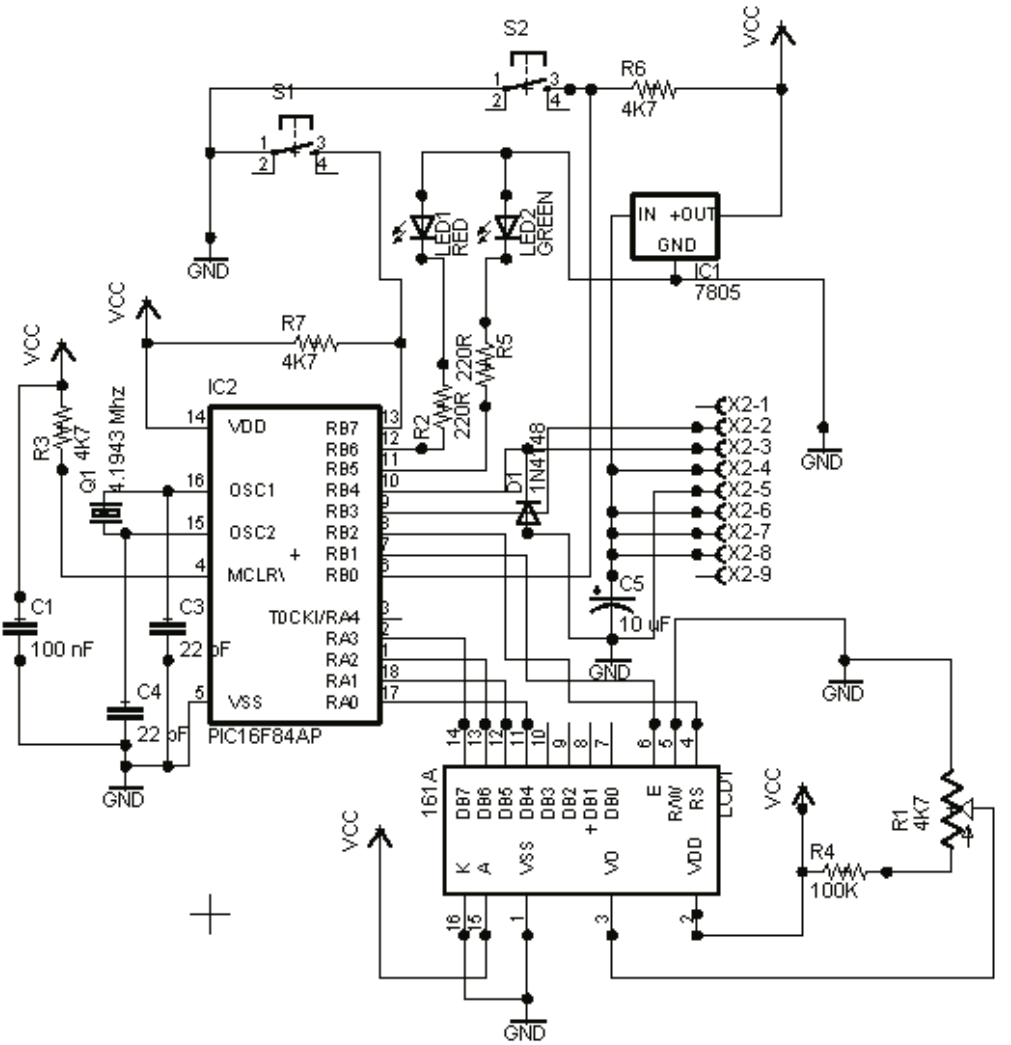
(j)

```
    GOTO LONGA
    RETLW 0

GRAPHICS
    MOVLW 040H
    CALL COMMAND

    MOVLW 0EH  ;CR SYMBOL
    CALL WRITE
    MOVLW 010H
    CALL WRITE
    MOVLW 0EH
    CALL WRITE
    MOVLW 0
    CALL WRITE
    MOVLW 01CH
    CALL WRITE
    MOVLW 012H
    CALL WRITE
    MOVLW 01CH
    CALL WRITE
    MOVLW 012H
    CALL WRITE
    RETLW 0

    END
```

4.3 Case Study RS232 data monitor

This projects captures 8 bytes of incoming RS232 data and displays it on an LCD in both HEX and ASCII. In addition, software handshake control character XON (17H) and XOFF (19H) will light a red or green LED. The data monitor is menu-driven using two simple switches S1 (ARM) and S2 (SELECT). For example:

The first line of the display shows that the eighth character captured is the numeral ‘4’ ,the second line displays ‘34H’ – the ASCII byte.

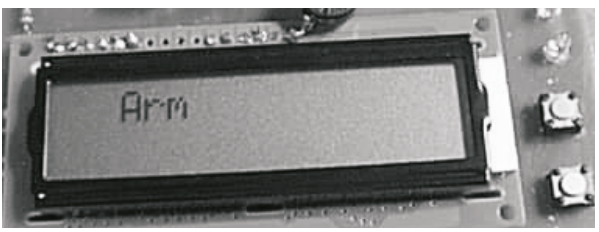


Circuit diagram of the RS232 Data Monitor

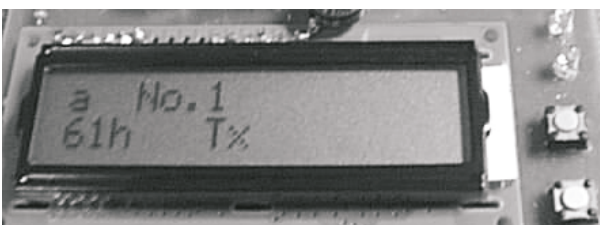
Operating Procedure

Power-up. 'Select' displayed.

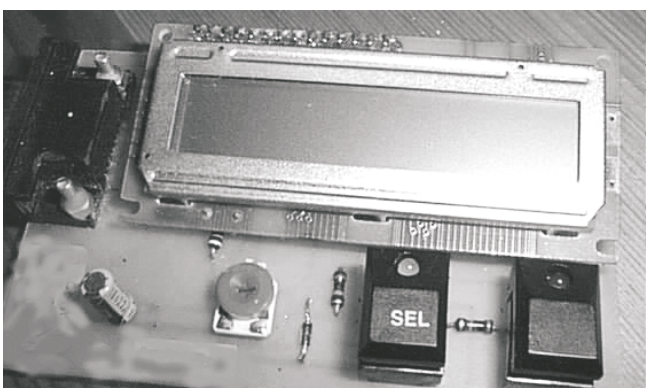
Press '**SELECT**' button – to cycle through the required baudrate – 2400, 4800, 9600, 19200, 38400 .



Press '**ARM**' button. Display now waits for incoming data.



Once 8 bytes are recieved, The first character is displayed. Pressing the ARM button in succession, displays the successive bytes , i,e, numbers 1 to 8. When the SELECT button is pressed, the display resets, allowing the user to select the baud rate.



RS232 Data capture unit.

Program 2LINELCD.ASM

(a)

```
        LIST p=16F84

    ;
    ORG 0
    CALL GO  ;INITIALISE DISPLAY
             ;AND PORTS
CYCLE
    BCF ABORT, 7
    BCF PORT_B, REDLED
    BCF PORT_B, GREENLED
    CALL SELECTINIT     ;WRITE
             ; 'SELECT' ON DISPLAY

WAITHERE CLRWDT ;WAIT FOR
             ; SELECT BUTTON TO
             ; BE PRESSED
    BTFSC PORT_B, SEL
    GOTO WAITHERE

    MOVLW 0FFH
    MOVWF XODE      ; CODE=0
    CALL SELECT_MENU

    MOVLW 0A9H      ; 2ND LINE
    CALL COMMAND

    MOVLW '2';  SELECT BAUD RATE
    CALL WRITE   ; AND DISPLAY
    MOVLW '4'; 2400 BAUD
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW '_'
    CALL WRITE
    CALL SELECT_MENU
    MOVLW 0A9H      ; 2ND LINE
    CALL COMMAND
```

(b)

```
    MOVLW '4'       ;4800 BAUD
    CALL WRITE
    MOVLW '8'
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    CALL SELECT_MENU

    MOVLW 0A9H; 2ND LINE
    CALL COMMAND

    MOVLW '9'       ;9600 BAUD
    CALL WRITE
    MOVLW '6'
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    MOVLW '0'
    CALL WRITE

    CALL SELECT_MENU

    MOVLW 0A9H      ; 2ND LINE
    CALL COMMAND

    MOVLW '1'       ;19200 BAUD
    CALL WRITE
    MOVLW '9'
    CALL WRITE
    MOVLW '2'
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    MOVLW '0'
    CALL WRITE

    CALL SELECT_MENU
    MOVLW 0A9H; 2ND LINE
    CALL COMMAND
    MOVLW '3'       ;38400 BAUD
    CALL WRITE
    MOVLW '8'
    CALL WRITE
```

(c)

```
    MOVLW '4'
    CALL WRITE
    MOVLW '0'
    CALL WRITE
    MOVLW '0'
    CALL WRITE

    CALL SELECT_MENU
    GOTO CYCLE      ;GOTO START

SELECT_MENU
MENU CLRWDT
    CALL DELAY_DATA
    BTFSS PORT_B, ENTER
    GOTO RELOAD     ; GOTO SETBAUD
                    ; RATE
    BTFSC PORT_B, SEL
    GOTO MENU
    INCF XODE
SEL_DEBOUNCE CALL DELAY_DATA
    CLRWDT
    BTFSS PORT_B, SEL
    GOTO SEL_DEBOUNCE
    RETLW 0

RELOAD
    MOVF XODE, 0
    CALL BAUD
    MOVWF CELL1

    MOVF XODE, 0
    CALL BAUD15
    MOVWF CELL15
    GOTO RELOAD_A

BAUD
    ADDWF PC    ;1 BIT PERIOD
    RETLW .138  ; 2400 BAUD
    RETLW .68   ; 4800 BAUD
    RETLW .34   ; 9600 BAUD
    RETLW .14   ; 19200 BAUD
    RETLW .5    ; 38400 BAUD; 5
```

(d)

```
BAUD15
  ADDWF PC      ;1.5 BIT PERIODS
  RETLW .180    ; 2400 BAUD
  RETLW .90     ; 4800 BAUD
  RETLW .45; 9600 BAUD
  RETLW .22; 19200 BAUD
  RETLW .7; 38400 BAUD; 8

RELOAD_A
    CALL DELAY_DATA
    BTFSC PORT_B, ENTER
    GOTO RELOAD_A
    CALL CLEAR
    MOVLW 8     ; 8 CHARACTERS
    MOVWF COUNTER

    MOVLW 01BH ; INITIALISE INDEX
    MOVWF FSR

NEXT        ;RECEIVE BYTE AND SAVE
    CALL RECEIVE_DATA
    BTFSC ABORT, 7
    GOTO Z
    BCF DATABYTE, 7;
    MOVF DATABYTE, 0
    MOVWF INDEX
    INCF FSR

    DECFSZ COUNTER, 1
    GOTO NEXT

        ; READ MEMORY AND DISPLAY
Z
CYCLE_A
    MOVLW '1'
    MOVWF XODE
    MOVLW 8; 8 CHARACTERS
    MOVWF COUNTER
    MOVLW 01BH; INITIALISE INDEX
    MOVWF FSR

NEXT_BYTE
    CALL POSITION

    MOVLW 080H      ; 1ST DISPLAY
                    ; ADDRESS
    CALL COMMAND
```

(e)

```
    MOVLW 8       ; DISPLAY OFF
    CALL COMMAND

    MOVF INDEX, 0
    ; TEST FOR XONXOFF HANDSHAKE
    ; GREEN ;LED; = XON RED
    ; LED = RED LED

    MOVWF BUFFER;
    XORLW 011H; XON
    BTFSS STATUS, 2
    GOTO A4
    BCF PORT_B, REDLED
    BSF PORT_B, GREENLED
A4
    MOVF BUFFER, 0
    XORLW 013H; XOFF
    BTFSS STATUS, 2
    GOTO A3
    BSF PORT_B, REDLED
    BCF PORT_B, GREENLED

A3
    MOVF BUFFER,0
    MOVWF A_N
    CALL WRITE

    CALL ASCII_HEX
    MOVLW 0CH; DISPLAY ON
    CALL COMMAND

    INCF FSR
A1  CLRWDT
    CALL DELAY_DATA
    BTFSS PORT_B, SEL    ;RESET
    GOTO CYCLE
    BTFSC PORT_B, ENTER
    GOTO A1
A2  CLRWDT
    BTFSS PORT_B, ENTER
    GOTO A2
    DECFSZ COUNTER, 1
    GOTO NEXT_BYTE
    GOTO CYCLE_A
```

(f)

```
COMMAND         ;WRITE COMMAND
                ; TO LCD
    MOVWF LCDDATA
    SWAPF LCDDATA, 0; MSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E

    MOVF LCDDATA, 0; LSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E

    RETLW 0

WRITE     ;WRITE DATA TO DISPLAY
    MOVWF A_N
    SWAPF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E

    MOVF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E

    RETLW 0

ASCII_HEX
    MOVLW 0A9H      ; 2ND DISPLAY
                    ; ADDRESS
    CALL COMMAND
    BCF A_N, 7
    MOVF A_N, 0
    MOVWF ASCII; SAVE ASCII
    SWAPF A_N, 1
    MOVLW 0FH
    ANDWF A_N, 0
```

(g)

```
    CALL BINARY
    MOVWF A_N
    CALL WRITE

    MOVLW 0FH
    ANDWF ASCII, 0
    CALL BINARY
    MOVWF A_N
    CALL WRITE

    MOVLW 'h'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    BTFSS BUFFER_2,7
    GOTO RXDTXD
    MOVLW 't'
    CALL WRITE
    MOVLW 'x'
    CALL WRITE
    RETLW 0

RXDTXD
    MOVLW 'R'
    CALL WRITE
    MOVLW 'x'
    CALL WRITE
    RETLW 0

BINARY
    ADDWF PC
    RETLW '0'  ;ASCII LOOK-UP TABLE
    RETLW '1'
    RETLW '2'
    RETLW '3'
    RETLW '4'
    RETLW '5'
    RETLW '6'
    RETLW '7'
    RETLW '8'
    RETLW '9'
    RETLW 'A'
    RETLW 'B'
```

(h)

```
    RETLW 'C'
    RETLW 'D'
    RETLW 'E'
    RETLW 'F'

DELAY1ST
    MOVF CELL15, 0
    MOVWF DLYCNT
REDA DECFSZ DLYCNT, 1
    GOTO REDA
    CLRWDT
    RETLW 0

DELAY
    MOVF CELL1, 0
    MOVWF DLYCNT
REDO DECFSZ DLYCNT, 1
    GOTO REDO
    CLRWDT
    RETLW 0

RECEIVE_DATA
START_BIT
    CLRWDT
    BTFSC PORT_B, RXDLINE
    GOTO TXD_DATA
    BTFSC PORT_B, TXDLINE
    GOTO RXD_DATA
    BTFSC PORT_B, SEL
    GOTO START_BIT
    BSF ABORT, 7

A90 CLRWDT
    CALL DELAY_DATA
    BTFSC PORT_B, SEL
    RETLW 0
    GOTO A90

TXD_DATA
    MOVLW 8; 8 DATA BITS
    MOVWF COUNTR
    CALL DELAY1ST; 1.5 BITS DELAY
    CLRF BUFFER; CLEAR BUFFER
```

(i)

```
NEXT_BIT
    BCF STATUS, 0
    RRF BUFFER
    BTFSC PORT_B, RXDLINE
    BSF BUFFER, 7
    CALL DELAY
    DECFSZ COUNTR, 1
    GOTO NEXT_BIT

    MOVLW 0FFH
    XORWF BUFFER, 0
    MOVWF DATABYTE   ; SAVE
                     ; RECEIVED DATA
    BCF BUFFER_2,7
    RETLW 0

RXD_DATA
    MOVLW 8; 8 DATA BITS
    MOVWF COUNTR
    CALL DELAY1ST; 1.5 BITS DELAY

CLRF BUFFER       ; CLEAR BUFFER

NEXT_BITR
    BCF STATUS, 0
    RRF BUFFER
    BTFSC PORT_B, TXDLINE; PC
    BSF BUFFER, 7
    CALL DELAY
    DECFSZ COUNTR, 1
    GOTO NEXT_BITR
    MOVLW 0FFH
    XORWF BUFFER, 0
    MOVWF DATABYTE   ; SAVE
                     ; RECEIVED DATA
    BSF BUFFER_2,7
    RETLW 0

INIT_DISPLAY
    CALL LDELAY

    MOVLW 033H
    CALL COMMAND
    MOVLW 032H
    CALL COMMAND
    MOVLW 028H
    CALL COMMAND
```

(j)

```
    MOVLW 6
    CALL COMMAND
    MOVLW 0EH
    CALL COMMAND
    MOVLW 1
    CALL COMMAND
    CALL LDELAY
    RETLW 0

LDELAY
    MOVLW B'00000011'
    OPTION
    CLRF RTCC
L_DELAY
    CLRWDT
    BTFSS RTCC, 7
    GOTO L_DELAY
    RETLW 0

DELAY_DATA
    MOVLW DEBOUNCE
    MOVWF DLYCNT
REDB DECFSZ DLYCNT,1

    Goto REDB
    CLRWDT
    RETLW 0
CLEAR
    MOVLW 080H
    CALL COMMAND
    MOVLW ' '
    MOVWF A_N
    CALL WRITE

    MOVLW 082H; 1ST LINE
    CALL COMMAND

    MOVLW 'A'          ;WRITE 'ARM'
    CALL WRITE
    MOVLW 'R'
    CALL WRITE
    MOVLW 'M'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
```

(k)

```
    MOVLW 0A9H; 2ND LINE
    CALL COMMAND

    MOVLW ' '      ;BLANK 2ND LINE
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    RETLW 0

SELECTINIT
    MOVLW 0A9H       ; 2ND LINE
    CALL COMMAND
    MOVLW ' '            ;WRITE 'SELECT'
    CALL WRITE
    MOVLW 'S'
    CALL WRITE
    MOVLW 'e'
    CALL WRITE
    MOVLW 'l'
    CALL WRITE
    MOVLW 'e'
    CALL WRITE
    MOVLW 'c'
    CALL WRITE
    MOVLW 't'
    CALL WRITE
    MOVLW 07EH
    CALL WRITE

    MOVLW 080H; 1ST LINE
    CALL COMMAND
    MOVLW 07FH       ;BLANK 1ST LINE
    CALL WRITE
    MOVLW ' '
    CALL WRITE
```

(l)

```
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    RETLW 0

GO
    MOVLW 0
    TRIS PORT_A
    MOVLW B'10011001'
    TRIS PORT_B
        ; RB0 SELECT
        ; RB3 RXD LINE
        ; RB4 TXD LINE
        ; RB7 ENTER

    BSF PORT_B, REDLED
    BSF PORT_B, GREENLED
                    ; AMBER SYSTEM ON
    BCF PORT_B, RS
    BCF PORT_B, E
    MOVLW 0FH
    MOVWF PORT_A
    CALL INIT_DISPLAY
    RETLW 0

POSITION
    MOVLW 082H
    CALL COMMAND

    MOVLW ' '
    CALL WRITE

    MOVLW 'N'          ;WRITE 'No.'
    CALL WRITE
    MOVLW 'o'
    CALL WRITE
    MOVLW '.'
    CALL WRITE
```

```
(m)
    MOVF XODE, 0
    CALL WRITE
    INCF XODE, 1
    RETLW 0

; BAUD EQU  .136; 2400 .68(4800)
;.14(19200) .34(9600) 1 BIT
; BAUD1ST EQU .180; 2400 .90(4800)
;.22(19200) .45(9600) 1.5 BITS
; Baud equ 5   (38k) equ 2 (57.6k)
; Baud1st equ .10 (38k) equ 4 (57.6k)

DEBOUNCE EQU 0FFH
SEL EQU 0 ; RB0 INPUT/INTERRUPT
ENTER EQU 7  ; RB7 ENTER
E EQU 1      ; RB1 LCD
RS EQU 2   ; RB2 LCD
REDLED EQU 5      ; RB5 XOFF
GREENLED EQU 6 ; RB6 XON

RXDLINE EQU 3    ; RB3
TXDLINE EQU 4    ; RB4

INDEX EQU 0
RTCC EQU 1    ; RTCC
PC EQU 2
```

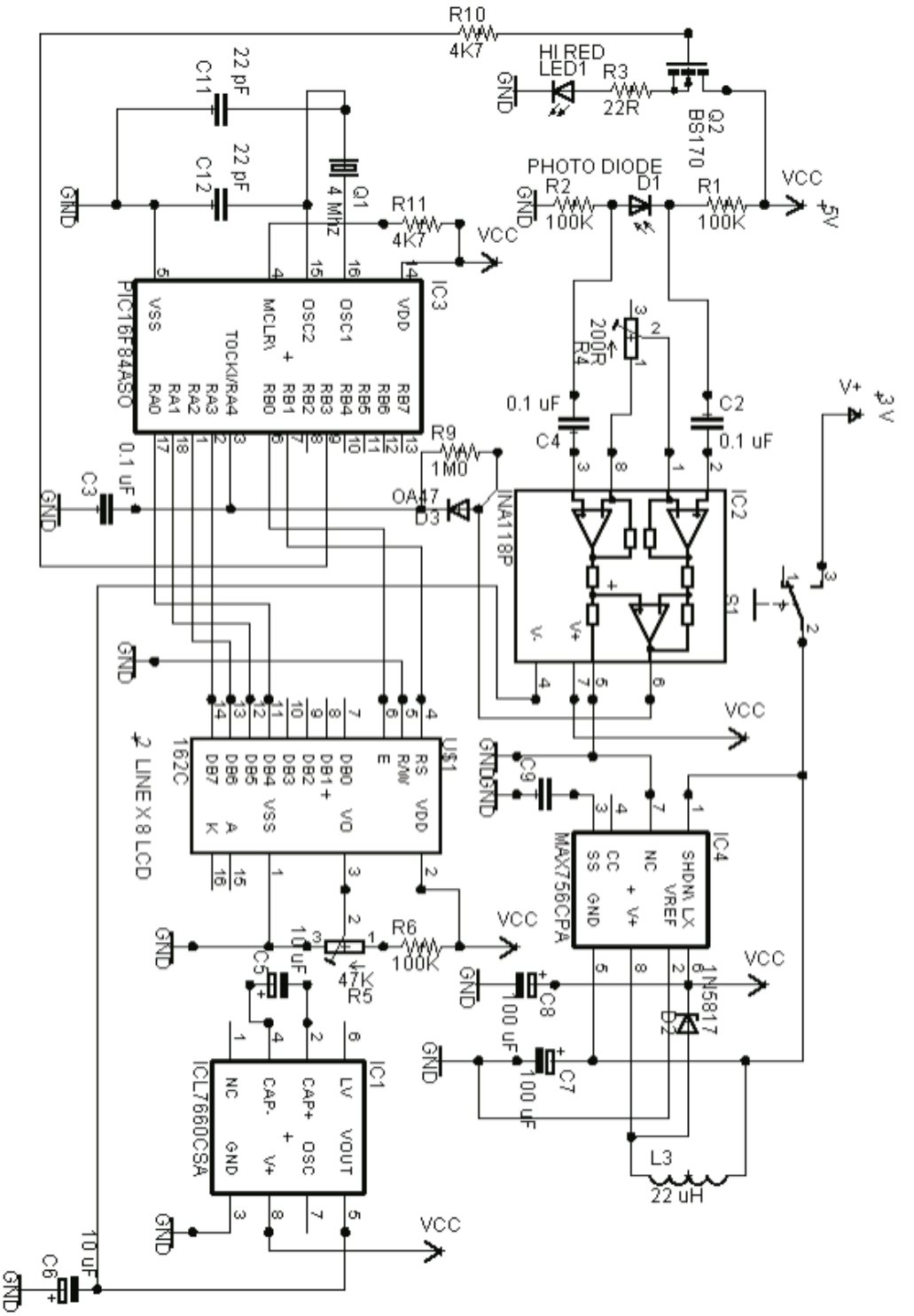
```
(n)
STATUS EQU 3      ; STATUS REGISTER
FSR EQU 4
PORT_A EQU 5 ; PORT A LCD DATA BUS
PORT_B EQU 6      ; PORT B
CONTROL/MONITOR

THIRTYTWO EQU 0BH      ; MODULUS 32
COUNTER
COUNTER EQU 0CH    ; ADDRESS
COUNTER
A_N EQU 0DH        ; ALPHANUMERIC
COUNTER
ADDRESS EQU 0EH     ; LCD ADDRESS
COUNTR EQU 0FH
BUFFER EQU 010H
DATABYTE EQU 011H
DLYCNT EQU 012H
ASCII EQU 013H
LCDDATA EQU 014H
CELL1 EQU 015H
CELL15 EQU 016H
XODE EQU 017H
ABORT EQU 018H
BUFFER_2 EQU 019H

    END
```

4.4 Case Study heart rate monitor -Program OXY.ASM

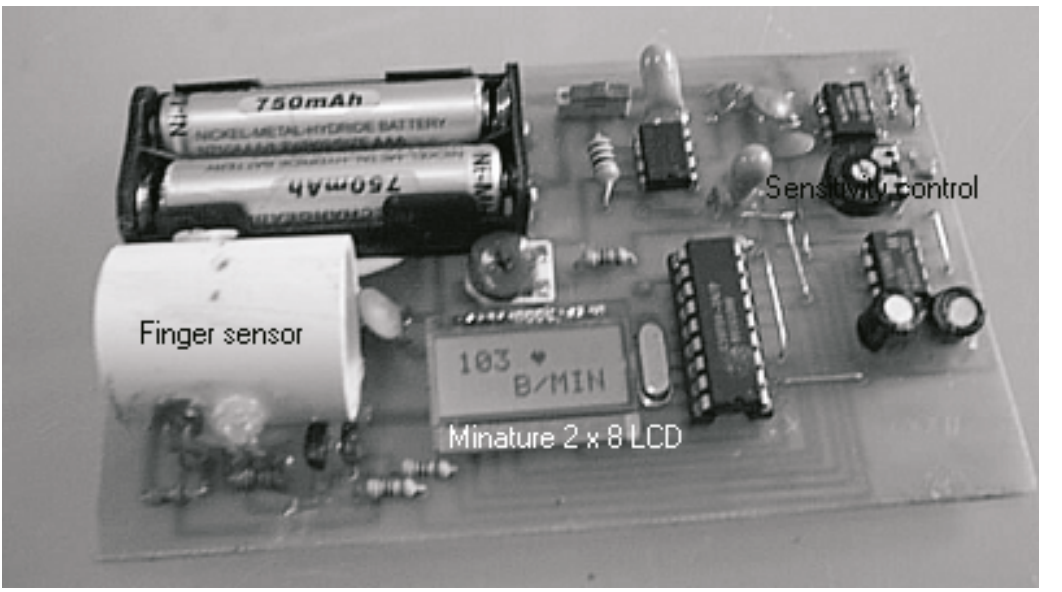
Introduction

This portable heart rate monitor relies on the fact that, when light shines through a finger, it is modulated by the blood flow in sympathy with the heart rate. A high efficiency red LED or an infra-red LED may be used, the light transmission being turned into an analogue signal by a conventional photo-diode. Since the o/p variations of the diode are miniscule, they need to be amplified by a high gain operational amplifier, in this case the 1NA118P. The gain is controlled by a single pre-set (R4) between terminals 1 and 8. The table above show that a gain of ×10000 is possible – the preset therefore acting as a sensitivity control. The author elected to

DESIRED GAIN	R_G (Ω)	NEAREST 1% R_G (Ω)
1	NC	NC
2	50.00k	49.9k
5	12.50k	12.4k
10	5.556k	5.62k
20	2.632k	2.61k
50	1.02k	1.02k
100	505.1	511
200	251.3	249
500	100.2	100
1000	50.05	49.9
2000	25.01	24.9
5000	10.00	10
10000	5.001	4.99

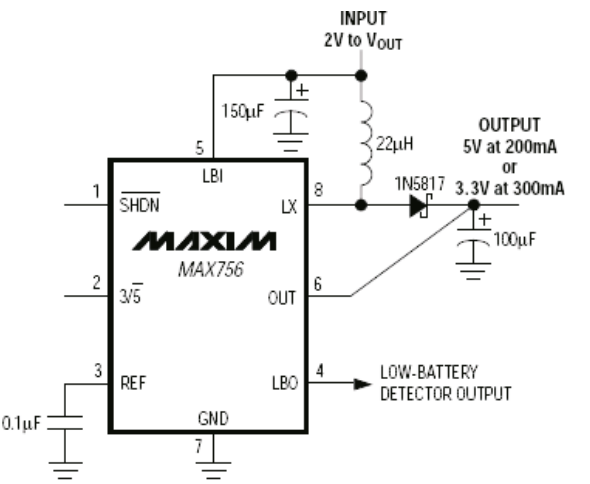

R10
4K7
HI RED
LED1
R3
22R
Q2
BS170
C11
22 pF
C12
22 pF
Q1
4 Mhz
R11
4K7
PHOTO DIODE
R2
100K
D1
R1
100K
VCC
+5V
GND
IC3
PIC16F84ASO
R4
200R
0.1 uF
C4
C2
0.1 uF
V+
+3 V
R9
1M0
OA47
D3
C3
0.1 uF
INA118P
IC2
S1
LINE X 8 LCD
162C
LS1
C9
IC4
MAX756CPA
R6
100K
10 uF
C5
47K
R5
C8
100 uF
1N5817
D2
C7
100 uF
L3
22 uH
IC1
ICL7660CSA
10 uF
C6

use a small piece of plastic plumbing tube to construct the finger sensor since the NHS issue ones are prohibitively expensive. A LED is inserted in one side of the tube in a 5 mm hole, whereas the surface mount photo-diode is attached to the opposite wall inside the tube. An alternate method is to dispense with the tube and mount both the emitter and receiver diodes alongside each other, whereby the finger is placed on both, and refracted light from the finger then modulates the reciever sensor. The software measures the duration of the heart 'pulse', coverts this to ASCII hundreds, tens and units and displays it on the LCD. E.G. For a heart rate of 120 beats/minute the rate of the pulse is 500 milliseconds. (60,000 / 500 =120 BPM).



The LCD display is a minature version approximately 1" wide and consists of one line of eight characters. It is still an industry standard LCD and its programming is identical to the larger displays. **Farnell** electronics supply these displays as part number 1137380.

The LCD requires a 5 volt supply and, since the project operates off a 3 volt or 3.6 volt lithium battery supply, a MAX756 step-up DC-DC switching regulator is used. It accepts a positive voltage down to 0.7 volts and converts it to 5 volts. The table below shows the



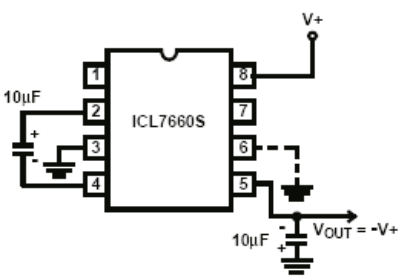
The MAX756 voltage convertor.

function of each pin. It should be noted that taking pin 1 low will shut down the i.c. but a few milliamps of current will still be consumed via the inductor and diode.

PIN		NAME	FUNCTION
MAX756	MAX757		
1	1	$\overline{SHDN}$	Shutdown Input disables SMPS when low, but the voltage reference and low-battery comparator remain active.
2	–	3/$\overline{5}$	Selects the main output voltage setting: 5V when low, 3.3V when high.
–	2	FB	Feedback Input for adjustable output operation. Connect to an external voltage divider between OUT and GND.
3	3	REF	1.25V Reference Voltage Output. Bypass with 0.22µF to GND (0.1µF if there is no external reference load). Maximum load capability is 250µA source, 20µA sink.
4	4	LBO	Low-Battery Output. An open-drain N-channel MOSFET sinks current when the voltage at LBI drops below +1.25V.
5	5	LBI	Low-Battery Input. When the voltage on LBI drops below +1.25V, LBO sinks current. Connect to V_{IN} if not used.
6	6	OUT	Connect OUT to the regulator output. It provides bootstrapped power to both devices, and also senses the output voltage for the MAX756.
7	7	GND	Power Ground. Must be low impedance; solder directly to ground plane.
8	8	LX	1A, 0.5Ω N-Channel Power MOSFET Drain

Pin description

Since the 1NA118P requires a balanced power supply, the ICL7660 performs a voltage conversion from a +ve to a -ve supply voltage.



Program OXY.ASM

(a)

```
LIST p=16C84

E EQU 0; LCD
RS EQU 1; LCD

RED EQU 3          ; RED LED
IR EQU 2; IR       ;NOT USED
```

(b)

```
Status EQU 3     ; STATUS REGISTER
c equ 0
w equ 0
f equ 1
z equ 2
```

(c)

```
PORTA EQU 5
PORT_A EQU 5; PORT A LCD DATA BUS
PORTB EQU 6
PORT_B EQU 6; PORT B
CONTROL/MONITOR
DEBOUNCE EQU 0FFH

INDF EQU 0
RTCC EQU 1          ; RTCC
PC EQU 2
STATUS EQU 3        ; STATUS REGISTER
FSR EQU 4

BAUD9600 EQU .30

COUNTER EQU 0CH     ; ADDRESS
                    ;COUNTER
A_N EQU 0DH         ; ALPHANUMERIC
                    ;COUNTER
SLOW EQU 0EH
DATABYTE EQU 0FH
ASCII EQU 010H
LCDDATA EQU 011H
DB1 EQU 012H
GP EQU 013H
DB2 EQU 014H
COUNTR EQU 015H
DLYCNT EQU 016H
BUFFER EQU 017H
TONE2 EQU 018H
BEAT EQU 019H
AOUNT EQU 01AH
SAVE EQU 01BH
MIN EQU 01CH

display1 equ 01dh
display2 equ 01eh
display3 equ 01fh

THIRTYTWO EQU 020H  ; MODULUS
                    ; 32 COUNTER
```

(d)

```
FLAG EQU 02AH
beatlsb     equ 029H    ;heartbeat counter,lsb
beatmsb     equ 028H    ;heartbeat counter,msb
aargb0      equ 027H    ;dividend,lsb
aargb1      equ 026H    ;dividend,msb
bargb0      equ 025H    ;divisor,lsb
bargb1      equ 024H    ;divisor,msb
remb0       equ 023H    ;remainder,lsb
remb1       equ 022H    ;remainder,msb

MSEC_20 EQU 02CH
NUMBER EQU 02DH
NUMBER2 EQU 02EH
EVERY100 EQU 02FH

    ORG 0

    CALL SECOND
    MOVLW B'10000'
    TRIS PORT_A

    TRIS PORT_B

    BCF PORT_B, RS
    BCF PORT_B, E
    MOVLW 0FH
    MOVWF PORT_A

    MOVLW B'00000111'
    OPTION
;FLASH LED TO SIGNAL BOOT
        BCF PORT_B, IR; IR OFF
        BSF PORT_B, RED
        CALL SECOND
        BCF PORT_B, RED

    CALL INIT_DISPLAY
    CALL SECOND
    CALL SELECTINIT
    CALL GRAPHICS

    BSF FLAG, 0; HEART
```

(e)

```
;MAIN PROGRAM
CYCLEALL BSF PORT_B, RED

    call Heartbeat
    call convert

    call display
    GOTO CYCLEALL

Heartbeat

    BTFSS PORTA,4    ;wait until signal
                     ;goes high

    MOVLW 0

    goto Heartbeat

    MOVLW .250; 250
    MOVWF MIN

AAA CLRWDT
    CALL MS1
    DECFSZ MIN, 1
    GOTO AAA

    MOVLW .100; 50
    MOVWF MIN

AAB CLRWDT
    CALL MS1
    DECFSZ MIN, 1
    GOTO AAB

    clrf  beatmsb    ;preset beat registers
                     ;with 150msec
;ELIMINATE SLOPE OF HEART
;TRACE BEFORE MEASUREMENT
;BEGINS

    movlw  05EH      ;FE
    movwf  beatlsb
    movlw  1
    movwf  beatmsb
```

(f)

```
pulse  CALL MS1
    INCF beatlsb,1
    btfsc PORTA,4
    goto pulse

vvv

    call MS1                ;wait 1msec
;MEASURE WIDTH OF HEART
;PULSE IN MILLISECONDS

    incf  beatlsb,1         ;increment
                            ;beatlsb by 1msec
    btfsc  status,z         ;lsb rollover?
    incf  beatmsb,1         ;yes -increment msb
    BTFSS PORTA,4           ;keep looping till
                            ;signal goes back
                            ;high
    goto  vvv
    NOP
    retlw 0

MSI

    MOVLW B'00000110'   ; PRESCALER
                        ;00000100 (/256)
    OPTION
    CLRWDT
    MOVLW 4
    MOVWF RTCC          ;CLRF RTCC
                        ;LOAD RTCC

YHORT BTFSC RTCC, 3     ; TEST FOR
                        ; 128 DECIMAL
                        ;I.E. BIT 7 = HIGH

    GOTO JMPXY
    CLRWDT
    GOTO YHORT
JMPXY CLRF RTCC
    MOVLW .121; 200
    MOVWF AOUNT
BBB
    DECFSZ AOUNT, 1
    GOTO BBB

        RETLW 0
```

(g)

```
;BEAT = MSB,LSB MILLISECONDS
;convert
;calculate bpm = 60000/beat (lsb and msb)
;MICROCHIP MATHS ROUTINE

    ;Make dividend 60000 == 0xEA60
    movlw  0EAH   ;0EAH        ;msb
    movwf  aargb1
    movlw  060H   ;060H        ;lsb
    movwf  aargb0

    ;Get measured heartbeat values, as
    ;divisor
    movf beatmsb,0 ; beatmsb,w ;msb
    movwf bargb1

    movf beatlsb,0
    movwf bargb0

    clrf   remb0      ;initialise
                      ;remainder registers
    clrf   remb1

        ; do an unsigned 16-bit by 16-bit
        ; division
    movlw       0x10
    movwf       COUNTR

Lu16  rlf   aargb1,w
    rlf     remb0,f
    rlf     remb1,f
    movf    bargb0,w
    subwf   remb0,f
    movf    bargb1,w
    btfss   status,c
    incfsz  bargb1,w
    subwf   remb1,f
    btfsc   status,c
    goto    uok16
    movf    bargb0,w
    addwf   remb0,f
    movf    bargb1,w
    btfsc   status,c
    incfsz  bargb1,w
    addwf   remb1,f
    bcf     status,c
```

(h)

```
uok16  rlf  aargb0,f
    rlf     aargb1,f
    decfsz  COUNTR,f
    goto    Lu16

    ; convert the lsb result of the
    ;previous division to unpacked bcd
    movlw   0x08        ;an 8-bit
                        ;division
    movwf   COUNTR

    movlw   0x0a        ;divide the previous
                        ;lsb result by 10
    movwf   bargb0
    clrf    remb0       ;initialise remainder

Lu8     rlf  aargb0,w
    rlf     remb0,f
    movf    bargb0,w
    subwf   remb0,f
    btfsc   status,c
    goto    uok8
    addwf   remb0,f
    bcf     status,c
uok8    rlf  aargb0,f
    decfsz  COUNTR,f
    goto    Lu8

    movlw 0x0a      ;3rd bcd
                    ;digit - 100's is
                    ;blank(if not c
    movwf display3

    movf remb0,w
    movwf display1      ;1st bcd digit - 1's

    movf aargb0,w
    movwf display2 ;2nd bcd digit - 10's
                        ;(if not changed fur

    sublw   0x09        ;is the quotient >=10?
                        ;bnc     $+2
                        ;yes -quotient is
                        ;>=10, so divide by
                        ;10
    btfsC status,c
```

(i)

```
    RETLW 0   ; go back

    movlw  0x08        ;an 8-bit ;division
    movwf  COUNTR

    movlw  0x0a        ;divide the previous
                       ;lsb result by 10
    movwf  bargb0
    clrf   remb0       ;initialise remainder

Lu81  rlf  aargb0,w
    rlf    remb0,f
    movf   bargb0,w
    subwf  remb0,f
    btfsc  status,c
    goto   uok81
    addwf  remb0,f
    bcf    status,c
uok81 rlf  aargb0,f
    decfsz COUNTR,f
    goto   Lu81

    movf   remb0,w
    movwf  display2    ;2nd bcd digit - 10's

    movf   aargb0,w
    movwf  display3    ;3rd bcd digit - 100's

    RETLW 0            ; go back

display
    MOVLW 3
    CALL COMMAND
    MOVF display3, 0   ;HUNDREDS
    CALL BINARY
    CALL WRITE

    MOVF display2, 0   ;TENS
    CALL BINARY
    CALL WRITE

    MOVF display1, 0   ;UNITS
    CALL BINARY
    CALL WRITE
    MOVLW ' '; HEART
```

(j)

```
    CALL WRITE
    BTFSC FLAG, 0
    GOTO HEARTON

;PULSE HEART SYMBOL ON LCD IN
;SYMPATHY WITH HEARTBEAT

HEARTOFF
    BSF FLAG, 0
    MOVLW 0       ; HEART
    CALL WRITE
    GOTO XXXX
HEARTON
    BCF FLAG, 0
    MOVLW 1       ; HEART
    CALL WRITE

XXXX
    MOVLW '       '; HEART
    CALL WRITE

    MOVLW 0C0H ; 2ND LINE
    CALL COMMAND
    MOVLW ' '     ;WRITE 'B  /  M'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW 'B'
    CALL WRITE
    MOVLW '/'
    CALL WRITE
    MOVLW 'M'
    CALL WRITE
    MOVLW 'I'
    CALL WRITE
    MOVLW 'N'
    CALL WRITE
    retlw 0
```

(k)

```
BINARY
    ADDWF PC
    RETLW '0'
    RETLW '1'
    RETLW '2'
    RETLW '3'
    RETLW '4'
    RETLW '5'
    RETLW '6'
    RETLW '7'
    RETLW '8'
    RETLW '9'
    RETLW '0'

QUARTER
    MOVLW B'00000111'   ;PRESCALER
                        ;00000111 (/256)
    OPTION
    CLRF RTCC
    CLRWDT

    MOVLW 1
    MOVWF THIRTYTWO ; COUNTER
    CLRF RTCC           ; LOAD RTCC

BHORT BTFSC RTCC, 7     ; TEST FOR 128
                        ; DECIMAL
                    ; I.E. BIT 7 = HIGH
    GOTO JMP5
    CLRWDT
    GOTO BHORT

JMP5 CLRF RTCC
                ; 32 LOOP COUNTER
  DECFSZ THIRTYTWO, 1
  GOTO BHORT
  RETLW 0

SELECTINIT
    MOVLW 0;
    CALL COMMAND

    MOVLW 'S'           ;WRITE 'SYSTEM'
    CALL WRITE
    MOVLW 'y'
    CALL WRITE
```

(l)

```
    MOVLW 's'
    CALL WRITE
    MOVLW 't'
    CALL WRITE
    MOVLW 'e'
    CALL WRITE
    MOVLW 'm'
    CALL WRITE

    CALL SECOND

    MOVLW 0C0H      ; 2ND LINE
    CALL COMMAND

    MOVLW 'R'           ;WRITE 'READY'
    CALL WRITE
    MOVLW 'E'
    CALL WRITE
    MOVLW 'A'
    CALL WRITE
    MOVLW 'D'
    CALL WRITE
    MOVLW 'Y'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE

    CALL SECOND

CALL INIT_DISPLAY

    MOVLW 0;
    CALL COMMAND

    MOVLW 'I'       ;WRITE 'INSERT'
    CALL WRITE
    MOVLW 'N'
    CALL WRITE
    MOVLW 'S'
    CALL WRITE
    MOVLW 'E'
    CALL WRITE
```

(m)

```
    MOVLW 'R'
    CALL WRITE
    MOVLW 'T'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE

  CALL SECOND

    MOVLW 0C0H      ; 2ND LINE
    CALL COMMAND

    MOVLW 'F'       ;WRITE 'FINGER'
    CALL WRITE
    MOVLW 'I'
    CALL WRITE
    MOVLW 'N'
    CALL WRITE
    MOVLW 'G'
    CALL WRITE
    MOVLW 'E'
    CALL WRITE
    MOVLW 'R'
    CALL WRITE
    MOVLW ' '
    CALL WRITE
    MOVLW ' '
    CALL WRITE

    RETLW 0

INIT_DISPLAY

    MOVLW 1
    CALL COMMAND
    CALL LDELAY

    MOVLW 2
    CALL COMMAND
    CALL LDELAY

    MOVLW 0FH
    CALL COMMAND
    CALL LDELAY
```

(n)

```
    MOVLW 028H      ; 028H
    CALL COMMAND

    CALL LDELAY
    RETLW 0

CARIAGERETURN
    MOVLW 0
    CALL WRITE
    RETLW 0

COMMAND
    MOVWF LCDDATA
    SWAPF LCDDATA, 0
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E
    MOVF LCDDATA, 0    ;LSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E
    RETLW 0

WRITE
    MOVWF A_N
    SWAPF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E
    MOVF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E
    RETLW 0
```

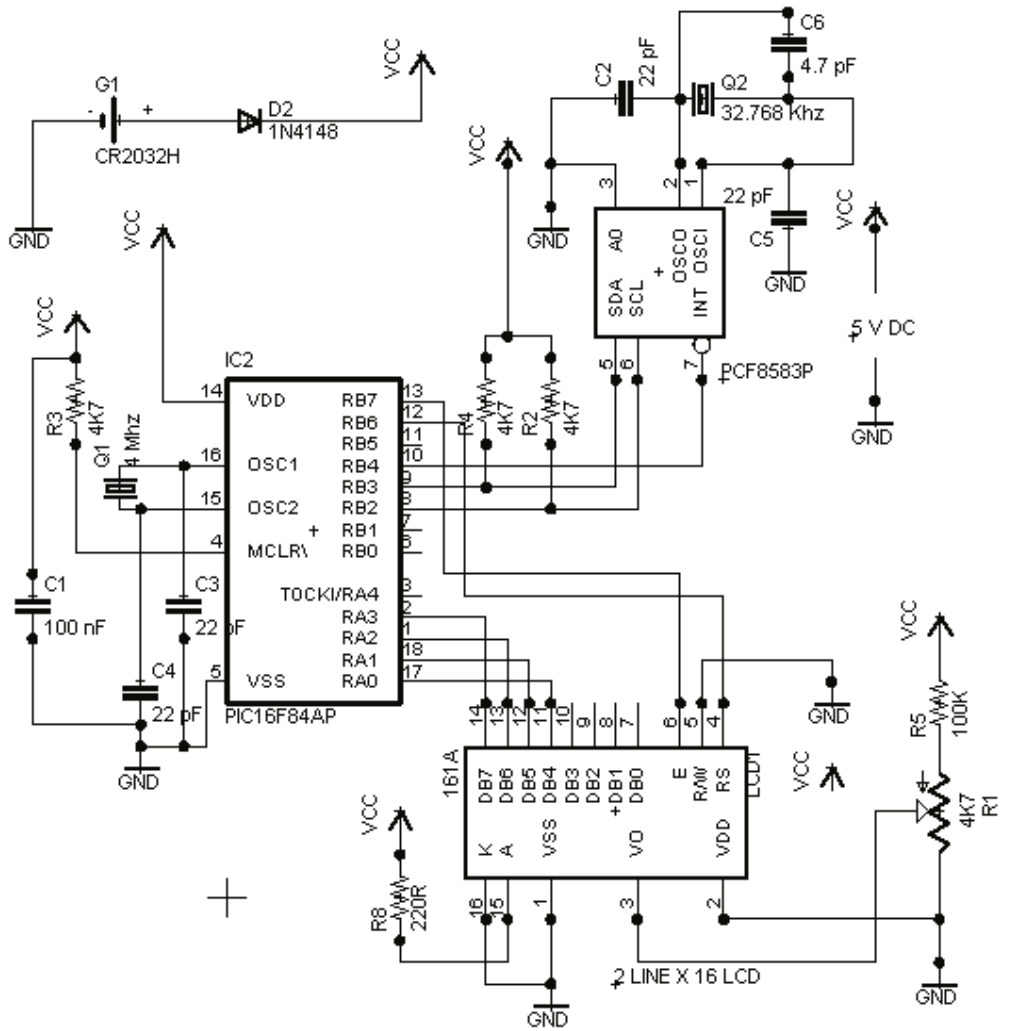
(o)

```
DELAY
    MOVLW BAUD9600
    MOVWF DLYCNT
REDX DECFSZ DLYCNT, 1
    GOTO REDX
    NOP
    RETLW 0

LDELAY
    MOVLW B'00000111'
    OPTION
    CLRF RTCC
L_DELAY
    CLRWDT
    BTFSS RTCC, 3; 7
    GOTO L_DELAY
    RETLW 0

DELAY_DATA
    MOVLW DEBOUNCE
    MOVWF DLYCNT
REDB DECFSZ DLYCNT, 1
    goto REDB
    CLRWDT
    RETLW 0

SECOND
                ; 1000.23 MS = 32
    MOVLW .32
    MOVWF SLOW
    MOVLW B'00000111'
    OPTION
LONGA CLRF RTCC
LON CLRWDT

    NOP
    BTFSS RTCC, 7
    GOTO LON
    DECFSZ SLOW, 1
    GOTO LONGA
    RETLW 0
```

(p)

```
GRAPHICS
    MOVLW 040H
    CALL COMMAND

    MOVLW 0       ;HEART SYMBOL
    CALL WRITE
    MOVLW 0AH
    CALL WRITE
    MOVLW 015H
    CALL WRITE
    MOVLW 011H
    CALL WRITE
    MOVLW 0AH
    CALL WRITE
    MOVLW 4
    CALL WRITE
    MOVLW 0
    CALL WRITE
    MOVLW 0
    CALL WRITE

    MOVLW 0       ;INVERTED HEART
    CALL WRITE
    MOVLW 0AH
    CALL WRITE
    MOVLW 01FH
    CALL WRITE
    MOVLW 01FH
    CALL WRITE
    MOVLW 0EH
    CALL WRITE
    MOVLW 4
    CALL WRITE
    MOVLW 0
    CALL WRITE
    MOVLW 0
    CALL WRITE

    RETLW 0

    END
```

4.5 Case Study IIC real time digital clock



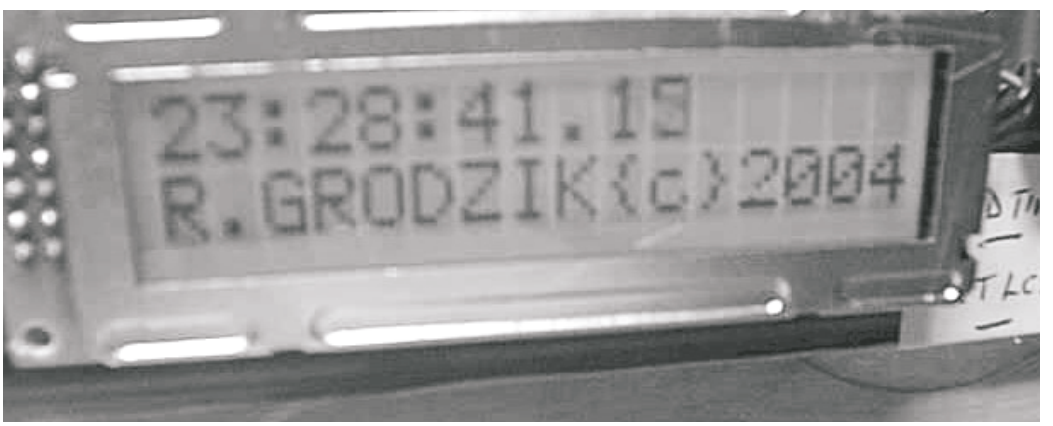
PROGRAM LCDTIME.ASM

Introduction

This LCD digital clock has the same accuracy of many clocks you can find in the high street. The design of this clock dispenses with the need for buttons to set the time. Wait until midnight, i.e. 00:00 hours and power up the circuit. The time will start @ 00:00:00.00, – that's right, this clock also displays 1/100's of seconds!

A standard 2 × 16 LCD display is used, preferably backlit, since the unit is powered by a 5 volt mains adaptor, so that current consumption is a lesser design issue. The circuit diagram is identical to previous LCD diagrams with the addition of the (PCF8583p).

The heart of the digital clock – the PCF8583p clock calendar chip – is controlled by a standard 32.768 kHz watch crystal. A CR2032 cell is used as a battery back-up in case of power failure and keeps the time held in the PCF8583's volatile memory alive. The PIC communicates with this chip via an IIC (2 wire) data bus consisting of a clock and data pin (SCL and SDA).



The completed digital clock.

Description of Software

On power up, the PCF8583p starts up at time 00.00.00.00. All the PIC needs to do is to interrogate the internal registers of this clock calendar chip via the IIC bus, convert the digital information to ASCII and send it to the LCD.

- All the LCD standard functions – COMMAND, WRITE/WRITEDATA, INIT_DISPLAY, LDELAY, DELAT_DATA are identical to previous LCD examples.

- The PCF8583p register address holds the following information:
  – address 00 1/100 seconds
  – address 01 seconds

- address 02 minutes
- address 03 hours

- These registers are accessed via the IIC bus via associated routines BSTART, BSTOP, BITIN AND BITOUT.

- PIC register DATAI contains the time variables which are converted to ASCII and sent to the display.

- Functions TX and RX clock in/out the 8 data bits from the PCF8583p registers.

- Routines TRASMIT1, 2, 3 and 4 write the time information to the LCD.

SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

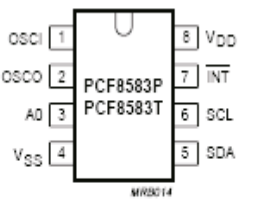


Fig.2 Pinning diagram.

The table shows the functionality of each pin of the PF8583.

Program LCDTIME.ASM

(a)

```
LIST p=16F84

DEBOUNCE EQU 01FH     ; 0FFH
E EQU 7               ; RB7 LCD
RS EQU 6              ; RB6 LCD

DO EQU 6
DI EQU 7

S_CLK EQU 2           ; PORT B
S_DATA EQU 3          ; PORT B

INTCON EQU 0BH
```

(b)

```
INDEX EQU 0
RTCC EQU 1        ; RTCC
PC EQU 2
STATUS EQU 3      ; STATUS REGISTER
FSR EQU 4
PORT_A EQU 5      ; PORT A LCD
                  ; DATA BUS
PORT_B EQU 6      ; PORT B
CONTROL/MONITOR
```

(c)

```
COUNTER EQU 0CH    ; ADDRESS
                   ; COUNTER
A_N EQU 0DH        ; ALPHANUMERIC
                   ; COUNTER
ADDRESS EQU 0EH    ; LCD ADDRESS
COUNTR EQU 0FH
BUFFER EQU 010H
DATABYTE EQU 011H
DLYCNT EQU 012H
ASCII EQU 013H
LCDDATA EQU 014H
CELL1 EQU 015H
CELL15 EQU 016H
CODES EQU 017H

DATAI EQU 018H
DATAO EQU 019H
SLAVE EQU 01AH

EEPROM EQU 01BH
TXBUF EQU 01CH
THIRTYTWO EQU 01DH
ADDR EQU 01EH
SAVEB EQU 01FH
SAVE EQU 020H
COUNT EQU 021H

    ORG 0
    MOVLW 0
    TRIS PORT_A

    MOVLW B'00010000'
    TRIS PORT_B
    BCF PORT_B, RS
    BCF PORT_B, E
    MOVLW 0FH
    MOVWF PORT_A

    CALL INIT_DISPLAY

    MOVLW 0A9H      ; 0A9H
    CALL COMMAND
```

(d)

```
    MOVLW 'R'
    CALL WRITEDATA
    MOVLW '.'
    CALL WRITEDATA
    MOVLW 'G'
    CALL WRITEDATA
    MOVLW 'R'
    CALL WRITEDATA
    MOVLW 'O'
    CALL WRITEDATA
    MOVLW 'D'
    CALL WRITEDATA
    MOVLW 'Z'
    CALL WRITEDATA
    MOVLW 'I'
    CALL WRITEDATA
    MOVLW 'K'
    CALL WRITEDATA
    MOVLW '{'
    CALL WRITEDATA
    MOVLW 'c'
    CALL WRITEDATA
    MOVLW '}'
    CALL WRITEDATA
    MOVLW '2'
    CALL WRITEDATA
    MOVLW '0'
    CALL WRITEDATA
    MOVLW '0'
    CALL WRITEDATA
    MOVLW '8'
    CALL WRITEDATA

    MOVLW 082H
    CALL COMMAND
    MOVLW ':'
    CALL WRITEDATA

    MOVLW 085H
    CALL COMMAND
    MOVLW ':'
    CALL WRITEDATA
    MOVLW 088H
    CALL COMMAND
    MOVLW '.'
    CALL WRITEDATA
```

(e)

```
HERE
    MOVLW B'00000111'
    OPTION
                        ; BSF INTCON, 3

    MOVLW B'00010000'
    TRIS PORT_B
                        ; SLEEP

    NOP
    BCF INTCON, 0

    MOVLW B'10100000'

    MOVWF SLAVE

    MOVLW 4             ; MINUTES =3
                        ; DATE=5
    MOVWF ADDR

    CALL BSTART
    MOVF SLAVE, W
    MOVWF TXBUF
    CALL TX
    MOVF ADDR, W
    MOVWF TXBUF
    CALL TX
    CALL BSTART
    MOVLW B'10100001'   ; READ
    MOVWF TXBUF
    CALL TX

    CALL RX
    MOVF DATAI, W
    CALL TRANSMIT2
    BSF EEPROM, DO
    CALL BITOUT
    CALL BSTOP
                        ; CALL SECOND

    MOVLW B'10100000'
    MOVWF SLAVE

    MOVLW 3             ; HOURS=4
                        ;5=MONTH
    MOVWF ADDR
                        ; CLRF ADDR
```

(f)

```
    CALL BSTART
    MOVF SLAVE, W
    MOVWF TXBUF
    CALL TX
    MOVF ADDR, W
    MOVWF TXBUF
    CALL TX
    CALL BSTART
    MOVLW B'10100001'   ; READ
    MOVWF TXBUF
    CALL TX

    CALL RX
    MOVF DATAI, W
    CALL TRANSMIT
    BSF EEPROM, DO
    CALL BITOUT
    CALL BSTOP
                        ; CALL SECOND

    MOVLW B'10100000'
    MOVWF SLAVE

    MOVLW 2             ; HOURS=4
                        ; 5=MONTH
    MOVWF ADDR
                        ; CLRF ADDR

    CALL BSTART
    MOVF SLAVE, W
    MOVWF TXBUF
    CALL TX
    MOVF ADDR, W
    MOVWF TXBUF
    CALL TX
    CALL BSTART

    MOVLW B'10100001'   ; READ
    MOVWF TXBUF
    CALL TX

    CALL RX
    MOVF DATAI, W
    CALL TRANSMIT3
    BSF EEPROM, DO
    CALL BITOUT
    CALL BSTOP
```

(g)

```
    MOVLW B'10100000'
    MOVWF SLAVE
    MOVLW 1             ; HOURS=4
                        ; 5=MONTH
    MOVWF ADDR
                        ; CLRF ADDR

    CALL BSTART
    MOVF SLAVE, W
    MOVWF TXBUF
    CALL TX
    MOVF ADDR, W
    MOVWF TXBUF
    CALL TX
    CALL BSTART
    MOVLW B'10100001'   ; READ
    MOVWF TXBUF
    CALL TX

    CALL RX
    MOVLW 01FH
    ANDWF DATAI, 1

    MOVF DATAI, W
    CALL TRANSMIT4
    BSF EEPROM, DO
    CALL BITOUT
    CALL BSTOP

    GOTO HERE

TRANSMIT3
     MOVLW 087H
     CALL COMMAND
     MOVF DATAI, 0
     MOVWF SAVEB
    MOVLW 0FH
    ANDWF DATAI, 0

    CALL BINARY
    CALL WRITEDATA

    MOVLW 086H
    CALL COMMAND

    SWAPF SAVEB, 1
    MOVLW 0FH;
```

(h)

```
    ANDWF SAVEB, 0

    CALL BINARY
    CALL WRITEDATA

    RETLW 0

TRANSMIT4
     MOVLW 08AH
     CALL COMMAND
     MOVF DATAI,0
     MOVWF SAVEB
     MOVLW 0FH
    ANDWF DATAI,0

    CALL BINARY
    CALL WRITEDATA

     MOVLW 089H
    CALL COMMAND

    SWAPF SAVEB, 1
    MOVLW 0FH;
    ANDWF SAVEB, 0

    CALL BINARY
    CALL WRITEDATA

    RETLW 0

TRANSMIT2
    MOVLW 081H
    CALL COMMAND
MOVF DATAI, 0
    MOVWF SAVEB
        MOVLW 0FH
        ANDWF DATAI, 0

     CALL BINARY
     CALL WRITEDATA

     MOVLW 080H
    CALL COMMAND

    SWAPF SAVEB, 1
    MOVLW 0FH
```

(i)

```
    ANDWF SAVEB, 0

    CALL BINARY
    CALL WRITEDATA

    RETLW 0

TRANSMIT

     MOVLW 084H
     CALL COMMAND
     MOVF DATAI, 0
     MOVWF SAVEB
     MOVLW 0FH
    ANDWF DATAI, 0

    CALL BINARY
    CALL WRITEDATA

     MOVLW 083H
     CALL COMMAND

     SWAPF SAVEB, 1
     MOVLW 0FH;
    ANDWF SAVEB, 0

     CALL BINARY

     CALL WRITEDATA

     RETLW 0

COMMAND
    MOVWF LCDDATA
    SWAPF LCDDATA, 0   ; MSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
CALL LDELAY
    BCF PORT_B, E

    MOVF LCDDATA, 0    ; LSNIBBLE
    MOVWF PORT_A
    BCF PORT_B, RS
    BSF PORT_B, E
    CALL LDELAY
    BCF PORT_B, E
```

(j)

```
    RETLW 0

    WRITEDATA
    MOVWF A_N
    SWAPF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF  PORT_B,E
    CALL DELAY_DATA
    BCF PORT_B, E

    MOVF A_N, 0
    MOVWF PORT_A
    BSF PORT_B, RS
    BSF PORT_B, E
    CALL DELAY_DATA
    BCF PORT_B, E

    RETLW 0

BINARY
    ADDWF PC
    RETLW '0'
    RETLW '1'
    RETLW '2'
    RETLW '3'
    RETLW '4'
    RETLW '5'
    RETLW '6'
    RETLW '7'
    RETLW '8'
    RETLW '9'

DELAY1ST
    MOVF CELL15, 0
    MOVWF DLYCNT

REDA DECFSZ DLYCNT, 1
    GOTO REDA
    CLRWDT
    RETLW 0

DELAY
    MOVF CELL1, 0
    MOVWF DLYCNT
REDO DECFSZ DLYCNT, 1
    GOTO REDO
```

(k)

```
    CLRWDT
    RETLW 0
INIT_DISPLAY
    CALL LDELAY

    MOVLW 033H
    CALL COMMAND
    MOVLW 032H
    CALL COMMAND
    MOVLW 028H
    CALL COMMAND
    MOVLW 6
    CALL COMMAND
    MOVLW 0CH; OEH
    CALL COMMAND
    MOVLW 1
    CALL COMMAND
    CALL LDELAY
    RETLW 0

LDELAY
    MOVLW B'00000011'
    OPTION
    CLRF RTCC
L_DELAY
    CLRWDT
    BTFSS RTCC, 7
    GOTO L_DELAY
    RETLW 0

DELAY_DATA
    MOVLW DEBOUNCE
    MOVWF DLYCNT
REDB DECFSZ DLYCNT, 1
    goto REDB
    CLRWDT
RETLW 0

SECOND
    MOVLW B'00000111'   ; PRESCALER
                        ; 00000111 (/256)
    OPTION
    CLRF RTCC

    MOVLW .32
    MOVWF THIRTYTWO     ; COUNTER
    CLRF RTCC; LOAD RTCC
```

(l)

```
SHORT BTFSC RTCC, 7     ; TEST FOR 128
                        ; DECIMAL
                  ; I.E. BIT 7 = HIGH
    GOTO JMP2

    GOTO SHORT

JMP2 CLRF RTCC  ; 32 LOOP COUNTER
    DECFSZ THIRTYTWO, 1
    GOTO SHORT
    RETLW 0

BSTART
  BSF PORT_B, S_DATA
  MOVLW 0
  TRIS PORT_B
  BCF PORT_B, S_CLK
  BSF PORT_B, S_CLK
  BCF PORT_B, S_DATA
  BCF PORT_B, S_CLK
  RETLW 0

BSTOP
  MOVLW 0
  TRIS PORT_B
  BCF PORT_B, S_DATA
  BSF PORT_B, S_CLK
  BSF PORT_B, S_DATA
  BCF PORT_B, S_CLK
  RETLW 0

BITIN
    BSF EEPROM, DI
    MOVLW B'00001000'
    TRIS PORT_B
    BSF PORT_B, S_DATA
    NOP

    CLRWDT

    NOP
    NOP
    NOP
    BSF PORT_B, S_CLK
    CLRWDT
    BTFSS PORT_B, S_DATA
    BCF EEPROM, DI
```

(m)

```
    NOP
    NOP
    BCF PORT_B, S_CLK

    RETLW 0

BITOUT
    MOVLW 0
    TRIS PORT_B
    BTFSS EEPROM, DO
    GOTO BITLOW
    BSF PORT_B, S_DATA
    GOTO CLKOUT

BITLOW BCF PORT_B, S_DATA
CLKOUT BSF PORT_B, S_CLK
     BCF PORT_B, S_CLK
     RETLW 0

TX MOVLW .8
    MOVWF COUNT
TXLP BCF EEPROM, DO
    BTFSC TXBUF, 7
    BSF EEPROM, DO
    CALL BITOUT
    RLF TXBUF, 1
```

(N)

```
    DECFSZ COUNT 1
    GOTO TXLP
    CALL BITIN
    BTFSC EEPROM, DI
    NOP
    RETLW 0

RX MOVLW B'00001000'
    TRIS PORT_B

    CLRF DATAI
    MOVLW .8
    MOVWF COUNT
CON BCF STATUS, 0
    RLF DATAI
    BSF PORT_B, S_DATA
    BSF PORT_B, S_CLK
    BTFSC PORT_B, S_DATA
    BSF DATAI, 0
    BCF PORT_B, S_CLK
    DECFSZ COUNT 1
    GOTO CON

    RETLW 0

    END
```

5 Graphic Liquid Crystal Displays

Introduction

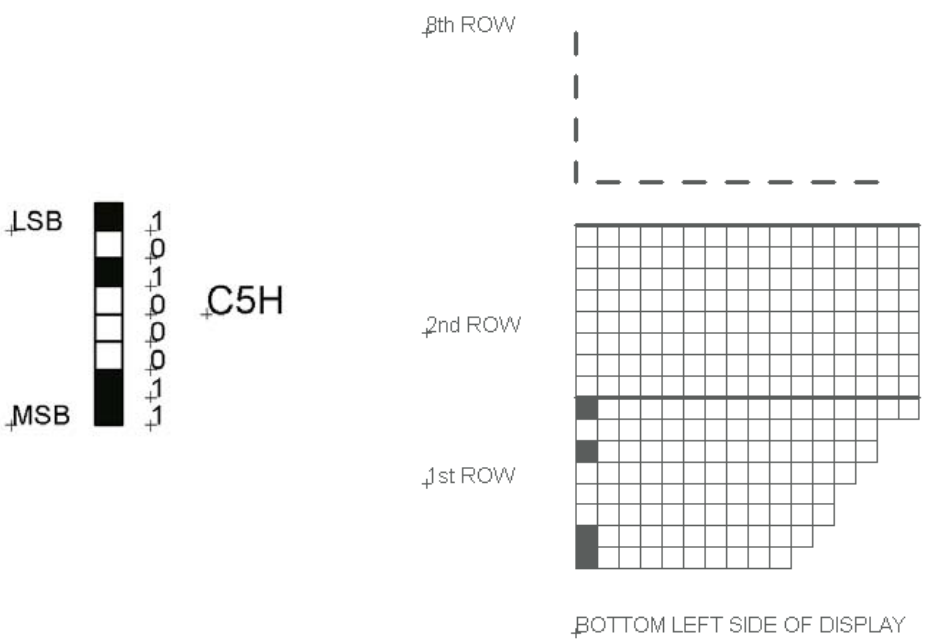
Graphic LCD modules give users more flexibility in displaying both text and graphical data such as large font characters and pictures, both in black and white and colour. The resolutions of these displays range from 80 × 32 to 640 × 480 dots or larger, and can display in monochrome and colour of up to 64,000 colours. This chapter starts with a case study using a 100 × 64 pixel and 84 × 48 B/W display and progresses to the colour LCD displays with a resolution of 131 × 131 pixels.

5.1 Case Study Densitron LM4068 B/W 100 x 64 pixel display

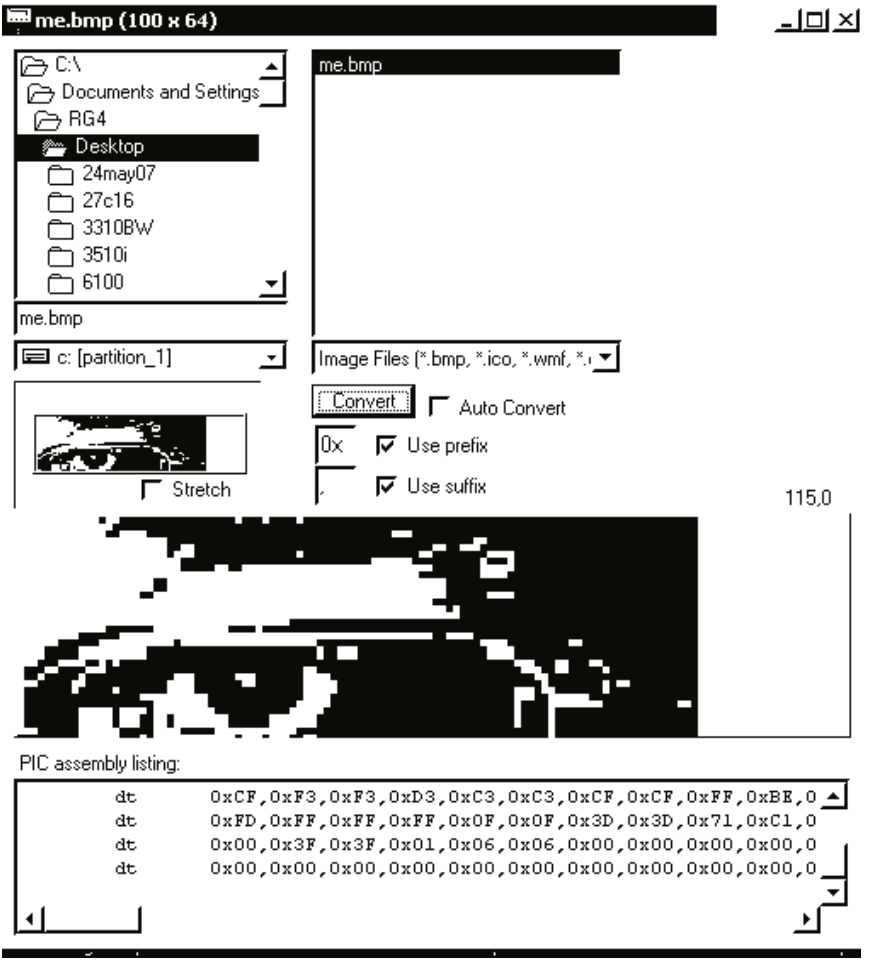
This display is backit and has an electronic contrast control whereby the contrast is software-controlled. Although it has a parallel data bus, it was decided to use its IIC 2 wire interface to program the display, in this case, to display an image of the author!

How is it done?

We start with a .BMP picure of exactly 100 rows × 64 horizontal pixels and digitise this using Bmp2ASM. 8 rows of 100 bytes are created since the vertical pixel pattern is organised in bytes i.e. 8 bytes × 8 bits = 64 pixels.



Each byte sets/clears eight pixels, as above, and sending a total of 800 bytes to the display in succession will form a complete picture.



Digitising the picture using Bmptoasm.

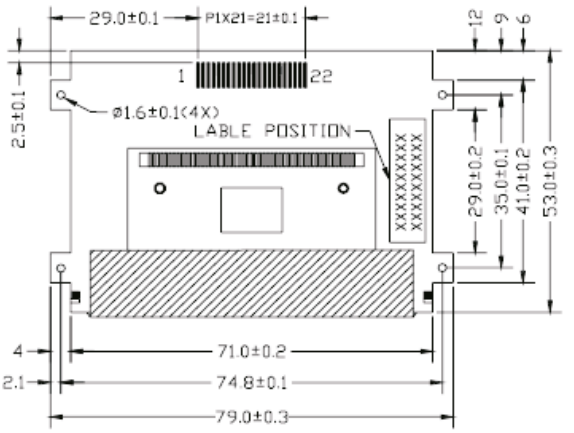
The generated byte data is then arranged as eight look-up tables in the software occupying 100 bytes each:

```
ORG 0200H

ONE

    ADDWF PC

    DT 0xC5,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
    DT 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xBF,0xBF
    DT 0x61,0xC1,0x87,0xFF,0x3F,0xFF,0xFF,0xFF,0xFF,0xFF
    DT 0xFF,0xFE,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x7F
```

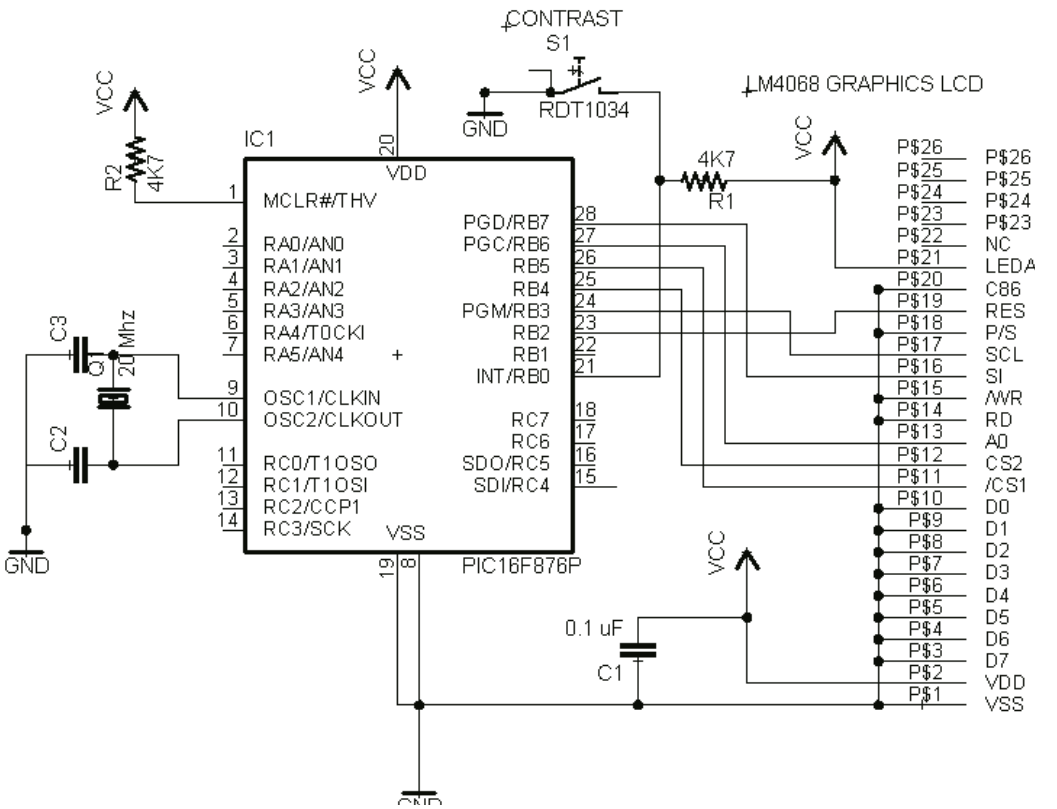
```
    DT 0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x80,0x80
    DT 0x40,0xC0,0x80,0x80,0x80,0xC0,0x40,0x40,0x40,0x40
    DT 0x40,0x40,0x40,0x00,0x00,0x00,0x00,0x40,0x40,0x40
    DT 0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x80,0x80
    DT 0x80,0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    DT 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

TWO
    ADDWF PC

    DT 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
    DT 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x7F,0x00,0xE3,0xE3
    DT 0xFF,0xFF,0xF7,0x00,0x00,0xFC,0xFF,0xFF,0xFF,0xFD
    DT 0xFF,0xFF,0xFF,0xFF,0xFF,0xFB,0xFB,0xFF,0xF3,0xF8
    DT 0xF8,0xFC,0xFC,0x7E,0x1F,0x1F,0x8F,0x07,0x03,0x83
    DT 0x89,0xF4,0xF4,0xF4,0xFC,0x9E,0xFA,0xFB,0xF9,0x01
    DT 0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x02,0x02
    DT 0x02,0x02,0x02,0x04,0x04,0x00,0x00,0x00,0x00,0x00
    DT 0x00,0x00,0x01,0x03,0x03,0x06,0x04,0x04,0x0C,0x78
    DT 0xF0,0x80,0x00,0x00,0x00,0x60,0x60,0x80,0xFC,0x00

ORG 0300H   etc.
```

Conecting the display requires a 26 WAY 1-mm flat flexible cable and two surface mount ZIF R/A bottom connector 1-mm 26 way connectors (one for the LCD and the other for the circuit board) although only 22 tracks are on the display :

To solder the ZIF connectors to the LCD, proceed as follows. Firstly, apply flux to the LCD connecting tracks and position the connector accurately, securing one end with a blob of solder. Run solder down the connector pins and finally remove all excess solder with solder wick. The second ZIF connector is similarly attached to the PCB – the PCB and the display are then connected using a 1-mm flat flex cable.



Position of the connecting tracks 1-22 (1-mm pitch) for the display.

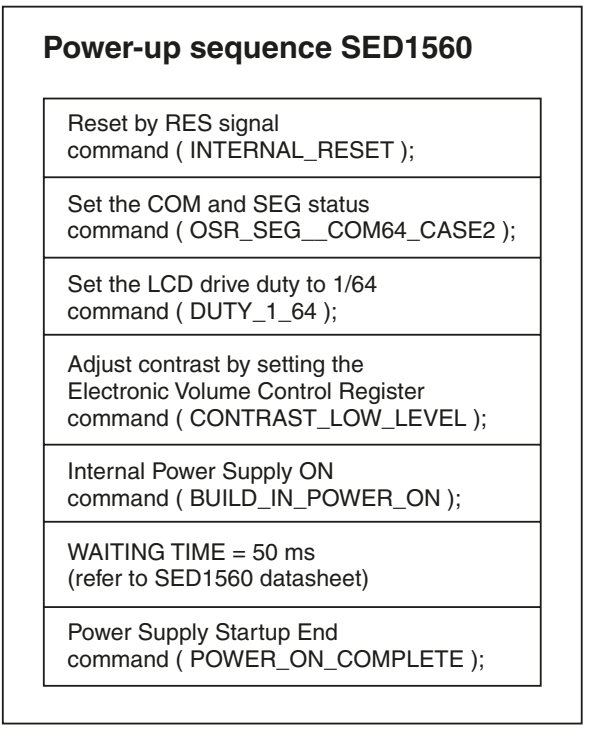


The circuit diagram of the LCD connected to a PIC16F876.

<table>
<tr><td>1</td><td>Vss</td><td>-</td><td>Ground (0V)</td></tr>
<tr><td>2</td><td>VDD</td><td>-</td><td>Logic Supply Voltage (+5V)</td></tr>
<tr><td>3</td><td>D7</td><td>I/O</td><td>Bi-directional data bus line 7</td></tr>
<tr><td>4</td><td>D6</td><td>I/O</td><td>Bi-directional data bus line 6</td></tr>
<tr><td>5</td><td>D5</td><td>I/O</td><td>Bi-directional data bus line 5</td></tr>
<tr><td>6</td><td>D4</td><td>I/O</td><td>Bi-directional data bus line 4</td></tr>
<tr><td>7</td><td>D3</td><td>I/O</td><td>Bi-directional data bus line 3</td></tr>
<tr><td>8</td><td>D2</td><td>I/O</td><td>Bi-directional data bus line 2</td></tr>
<tr><td>9</td><td>D1</td><td>I/O</td><td>Bi-directional data bus line 1</td></tr>
<tr><td>10</td><td>D0</td><td>I/O</td><td>Bi-directional data bus line 0</td></tr>
<tr><td>11</td><td>CS1</td><td>I/O</td><td>Chip select inputs. Data input/output is enabled when CS1 is LOW and CS2 is HIGH.</td></tr>
<tr><td>12</td><td>CS2</td><td>I/O</td><td>Chip select inputs. Data input/output is enabled when CS1 is LOW and CS2 is HIGH.</td></tr>
<tr><td>13</td><td>A0</td><td>I/O</td><td>Control/display data flag input. This is connected to the LSB of the microprocessor address bus.
• When LOW, the data on D0 to D7 is command data
• When HIGH, the data on D0 to D7 is display data</td></tr>
<tr><td>14</td><td>RD</td><td>I/O</td><td>Read</td></tr>
<tr><td>15</td><td>WR</td><td>I/O</td><td>Write</td></tr>
<tr><td>16</td><td>SI</td><td>I</td><td>Serial data input</td></tr>
<tr><td>17</td><td>SCI</td><td>I</td><td>Serial clock input. Data is read on the rising edge of SCL and converted to 8-bit parallel data.</td></tr>
<tr><td>18</td><td>P/S</td><td>I</td><td>Parallel/serial data input select
<table>
<tr><th>P/S</th><th>OPERATING MODE</th><th>CHIP SELECT</th><th>DATA/ COMMAND</th><th>DATA I/O</th><th>READ/ WRITE</th><th>SERIAL CLOCK</th></tr>
<tr><td>HIGH</td><td>Parallel</td><td>$\overline{CS1}$,CS2</td><td>A0</td><td>D0 to D7</td><td>RD,WR</td><td>-</td></tr>
<tr><td>LOW</td><td>Serial</td><td>$\overline{CS1}$,CS2</td><td>A0</td><td>SI</td><td>Write only</td><td>SCL</td></tr>
</table>
In serial mode, data cannot be read from the RAM, and D0 to D7, HZ, RD and WR must be HIGH or LOW. In parallel mode, SI and SCL must be HIGH or LOW.</td></tr>
<tr><td>19</td><td>RES</td><td>I</td><td>Reset input. Setting this pin low initializes the SED156X.</td></tr>
<tr><td>20</td><td>C86</td><td>I</td><td>Microprocessor interface select input.
• LOW when interfacing to 8080-series
• HIGH when interfacing to 6800-series</td></tr>
<tr><td>21</td><td>N/C</td><td>-</td><td>Not connected</td></tr>
<tr><td>22</td><td>N/C</td><td>-</td><td>Not connected</td></tr>
</table>

Interface description

PIC port line RB6 controls the A0 line of the display, determining if a command byte or data is being sent. The following sequence is used to initialise the display:



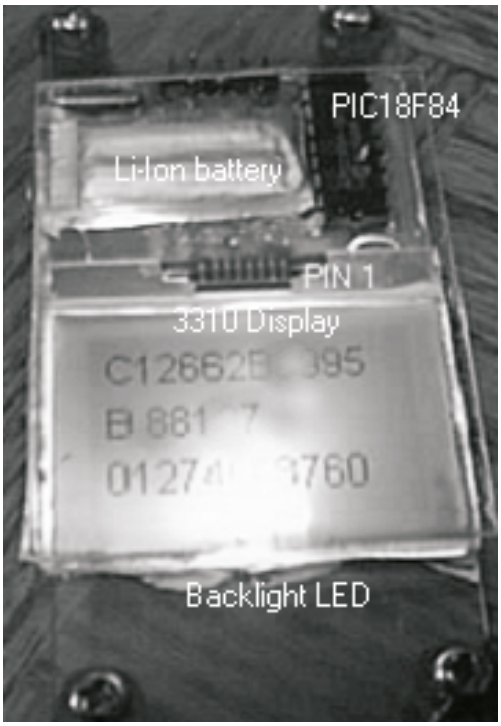
Program LM4068A.ASM

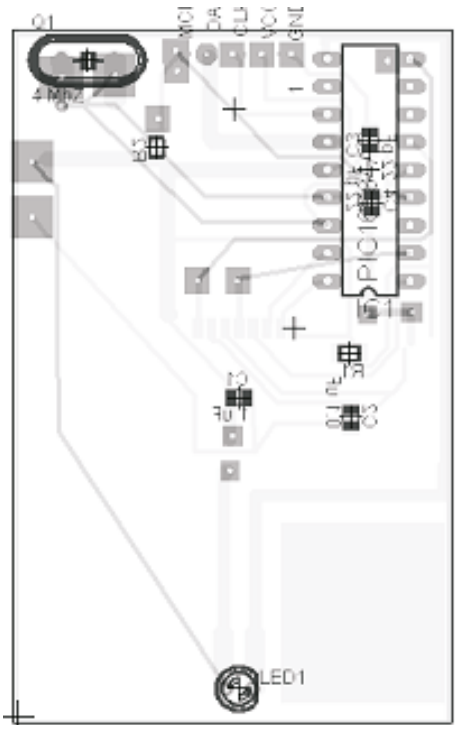
The complete source code listing can be found on the ELEKTOR website CODE folder of chapter 5. Changing the picture is just a simple matter of generating the picture data using 'Bmptoasm' and then cutting and pasting this data in place of the original look-up tables.

5.2 Case Study simple PDA using the Nokia 3310

Introduction

The Nokia 3310 LCD is a small graphical B/W LCD, suitable for a lot of various projects. The display is 38 × 35 mm, with an active display surface of 30 × 22 mm,and a 84 × 48 pixel resolution. The display is easy to interface, using standard SPI communication. A 1–10 μF electrolytic capacitor from VOUT to GND, is the only external component needed. It has a Logic supply voltage range VDD to VSS : 2.7 to 3.3 with VLow power consumption, and is suitable for battery operated systems. This project is powered by a minature 3.6 volt lithium battery, is compact and lightweight, with a single white LED for backlighting.





Pressing S1 'wakes up' the display, lights the LED and, after a pre-programmed time, e.g. one minute, goes back to sleep, thereby conserving battery energy. In this example only one screen of data is available, since the PIC84's memory is limited. A simple program – Nokia16f84.c – is used to test the display. It initialises the display and then writes 55H continously, which should show horizontal lines.
As the complexity increases, to program LCD displays, the software overhead when written in Assembler (ASM) is considerable, as is demonstrated in preceeding chapters. Here the program is written in C and the CC5X compiler is used in the MPLAB environment to generate the program code.

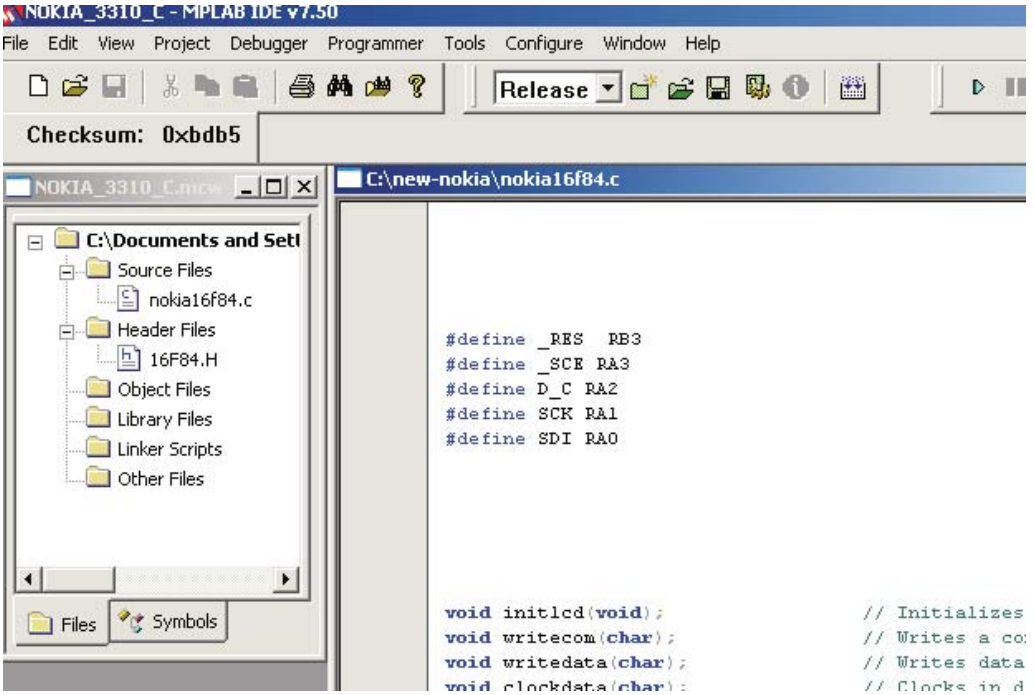
Program nokia16f84.c

```
#define _RES  RB3
#define _SCE RA3
#define D_C RA2
#define SCK RA1
#define SDI RA0
```

```
void initlcd(void);                 // Initializes the LCD.
void writecom(char);                // Writes a command.
void writedata(char);               // Writes data to DDRAM to illuminate the pixels.
void clockdata(char);               // Clocks in data into the PCD8544 controller.
                                    // Erase the DDRAM contents.
void cursorxy(char,char);           // Position cursor to x,y.
void mainrom(void);
void clearram(void);
void delay10(char);

/* These are global variables accessible by all functions within the firmware. */

void main(void)
{
    int roms;
        int i;

        TRISA = 0b00000;            // All SCK, SDI, D_C, _SCE and _RES are output pins

    TRISB = 0B00000000;

    initlcd();
 /*  U   */

    for (roms=0; roms<504; roms++)
{       writedata (0x55);
}

    #asm
        nop
:label
        nop
        goto label
    #endasm
}

void initlcd(void)
{
        _RES = 1;           // Set _RES HIGH.
        _SCE = 1;           // Disable Chip.
        _RES = 0;           // Reset the LCD.
```

```
    delay10(10);                // Wait 100ms.

        _RES = 1;               // Awake LCD from RESET state.

        writecom(0x21);         // Activate Chip and H=1.
        writecom(0xc8);         // c2 Set LCD Voltage to    about 7V.   vop
        writecom(0x13);         // Adjust voltage bias.
        writecom(0x20);         // Horizontal addressing and H=0.
        writecom(0x09);         // Activate all segments.
        clearram();             // Erase all pixel on the DDRAM.
        Writecom(0x08);         // Blank the Display.

    Delay10(10);                // Wait 100ms.
        Writecom(0x0C);         // Display Normal.
    Writecom(0x40);
        writecom(0x80);

    cursorxy(0,0);      // Cursor Home.
   Delay10(10);    // Wait 100ms.
   Delay10(10);    // Wait 100ms.
     Clearram();
   delay10(10);     // Wait 100ms.
}

void clearram(void)
{
        int ddram;
        cursorxy(0,0);                  // Cursor Home.
        For (ddram=504;ddram>0;ddram--)    {writedata(0x00);}
        // 6*84 = 504 DDRAM addresses.
}

void cursorxy(char x, char y)
{
        writecom(0x40|(y&0x07));     // Y axis
        writecom(0x80|(x&0x7f));     // X axis
}

void writecom(char command_in)
{
        D_C = 0;                        // Select Command register.
        _SCE = 0;                       // Select Chip.
        Clockdata(command_in);      // Clock in command bits.
```

```
        _SCE = 1;                       // Deselect Chip.
}

void writedata(char data_in)
{
        D_C = 1;                        // Select Data register.
        _SCE = 0;                       // Select Chip.
        Clockdata(data_in);             // Clock in data bits.
        _SCE = 1;                       // Deselect Chip.
}

void clockdata(char bits_in)
{
        int bitcnt;
        for (bitcnt=8; bitcnt>0; bitcnt--)
            {
                SCK = 0;                        // Set Clock Idle level LOW.
                If ((bits_in&0x80)==0x80) {SDI=1;}
                                                // PCD8544 clocks in the MSb first.
                Else {SDI=0;}
                SCK = 1;                        // Data is clocked on the rising edge of SCK.
                Bits_in=bits_in<<1;             // Logical shift data by 1 bit left.
            }
}

void delay10( char n)
/*
    Delays a multiple of 10 milliseconds using the TMR0 timer
    Clock : 4 MHz   => period T = 0.25 microseconds
    1 IS = 1 Instruction Cycle = 1 microsecond
    error: 0.16 percent
*/
{
    char i;

    OPTION = 7;
    do  {
        clrwdt();                   // only if watchdog enabled
        i = TMR0 + 39;              /* 256 microsec * 39 = 10 ms */
        while ( i != TMR0)
            ;
    }   while ( --n > 0);
}
```



The MPLAB screen for Program nokia16f84.c

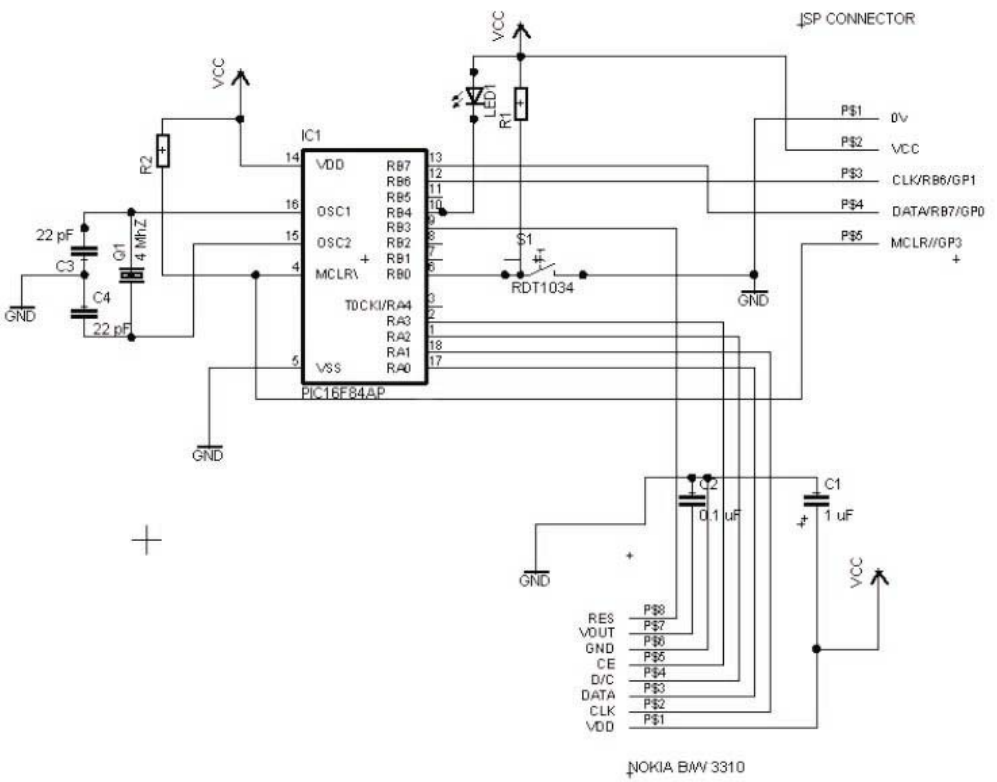
Details of initialising the 3310 display can be found in the PCD8544 PDF file. The instruction set below shows the various commands available including setting the X and Y address of RAM, and inverting and blanking the display. When commands are sent, the D/C pin of the display is cleared, and if data is to be written, this line is set.

INSTRUCTION	D/$\overline{C}$	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1)										
NOP	0	0	0	0	0	0	0	0	0	no operation
Function set	0	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)
Write data	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	writes data to display RAM
(H = 0)										
Reserved	0	0	0	0	0	0	0	1	X	X	do not use
Display control	0	0	0	0	0	1	D	0	E	sets display configuration
Reserved	0	0	0	0	1	X	X	X	X	do not use
Set Y address of RAM	0	0	1	0	0	0	Y_2	Y_1	Y_0	sets Y-address of RAM; $0 \le Y \le 5$
Set X address of RAM	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	sets X-address part of RAM; $0 \le X \le 83$
(H = 1)										
Reserved	0	0	0	0	0	0	0	0	1	do not use
	0	0	0	0	0	0	0	1	X	do not use

Temperature control	0	0	0	0	0	0	1	TC_1	TC_0	set Temperature Coefficient (TC_x)
Reserved	0	0	0	0	0	1	X	X	X	do not use
Bias system	0	0	0	0	1	0	BS_2	BS_1	BS_0	set Bias System (BS_x)
Reserved	0	0	1	X	X	X	X	X	X	do not use
Set V_{OP}	0	1	V_{OP6}	V_{OP5}	V_{OP4}	V_{OP3}	V_{OP2}	V_{OP1}	V_{OP0}	write V_{OP} to register

Table 2 Explanations of symbols in Table 1

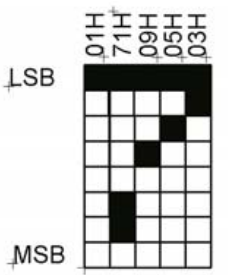
BIT	0	1
PD	chip is active	chip is in Power-down mode
V	horizontal addressing	vertical addressing
H	use basic instruction set	use extended instruction set
D and E		
00	display blank	
10	normal mode	
01	all display segments on	
11	inverse video mode	
TC_1 and TC_0		
00	V_{LCD} temperature coefficient 0	
01	V_{LCD} temperature coefficient 1	
10	V_{LCD} temperature coefficient 2	
11	V_{LCD} temperature coefficient 3	



A Simple PDA for the PIC16F84. Program nokia16f84.c

Generating the LCD characters/graphics

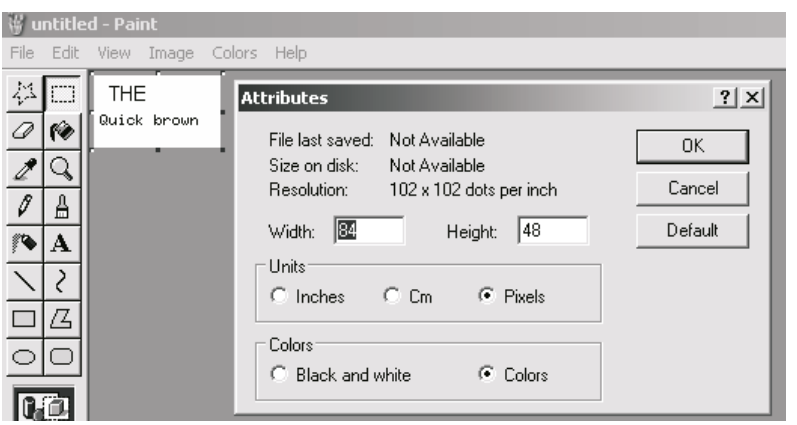
If we look at Program Nokia-9 in the CODE folder of chapter 5 it can be seen that each letter/number on the screen is composed of 5 bytes: For example the number 7:



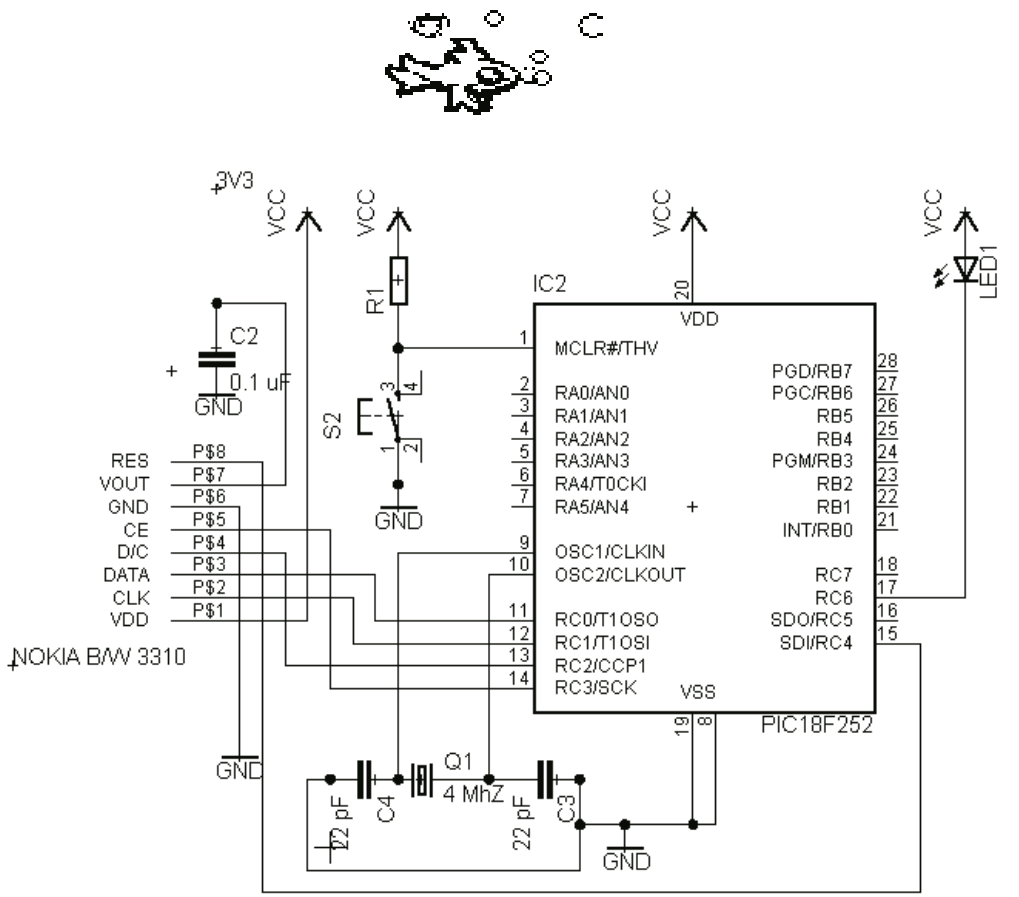

```
{ 0x3E, 0x51, 0x49, 0x45, 0x3E },       // 0
{ 0x00, 0x42, 0x7F, 0x40, 0x00 },       // 1
{ 0x42, 0x61, 0x51, 0x49, 0x46 },       // 2
{ 0x21, 0x41, 0x45, 0x4B, 0x31 },       // 3
{ 0x18, 0x14, 0x12, 0x7F, 0x10 },       // 4
{ 0x27, 0x45, 0x45, 0x45, 0x39 },       // 5
{ 0x3C, 0x4A, 0x49, 0x49, 0x30 },       // 6
{ 0x01, 0x71, 0x09, 0x05, 0x03 },       // 7
{ 0x36, 0x49, 0x49, 0x49, 0x36 },       // 8
{ 0x06, 0x49, 0x49, 0x29, 0x1E },       // 9
```

5.3 Icon image editing software

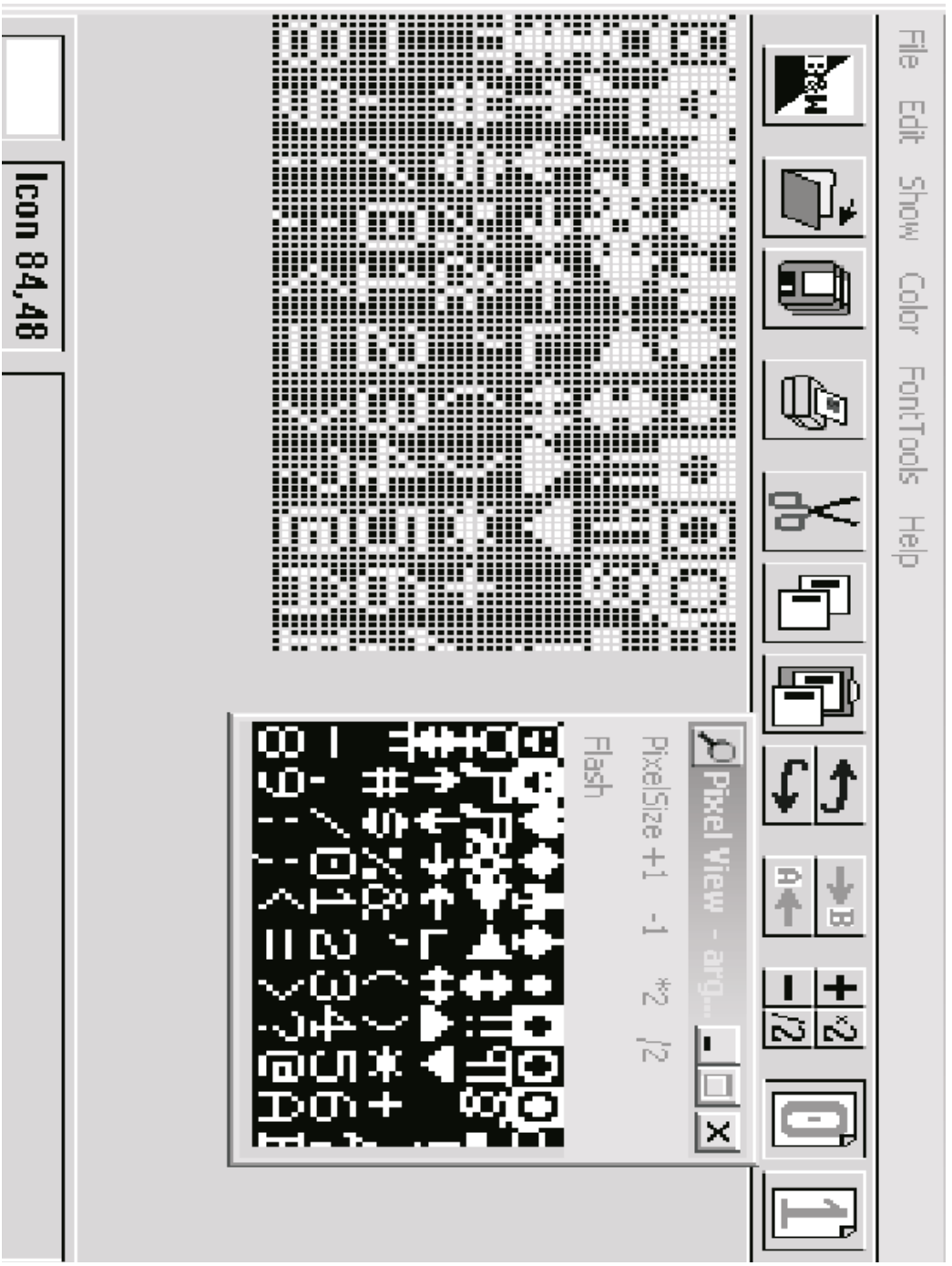
Constructing each character in turn is a cumbersome process, and so we shall use an icon edit program (LCDCOLOR) to construct the required image for the display: Here we use a virtual keyboard on the computer's screen to select the characters of any size and font.



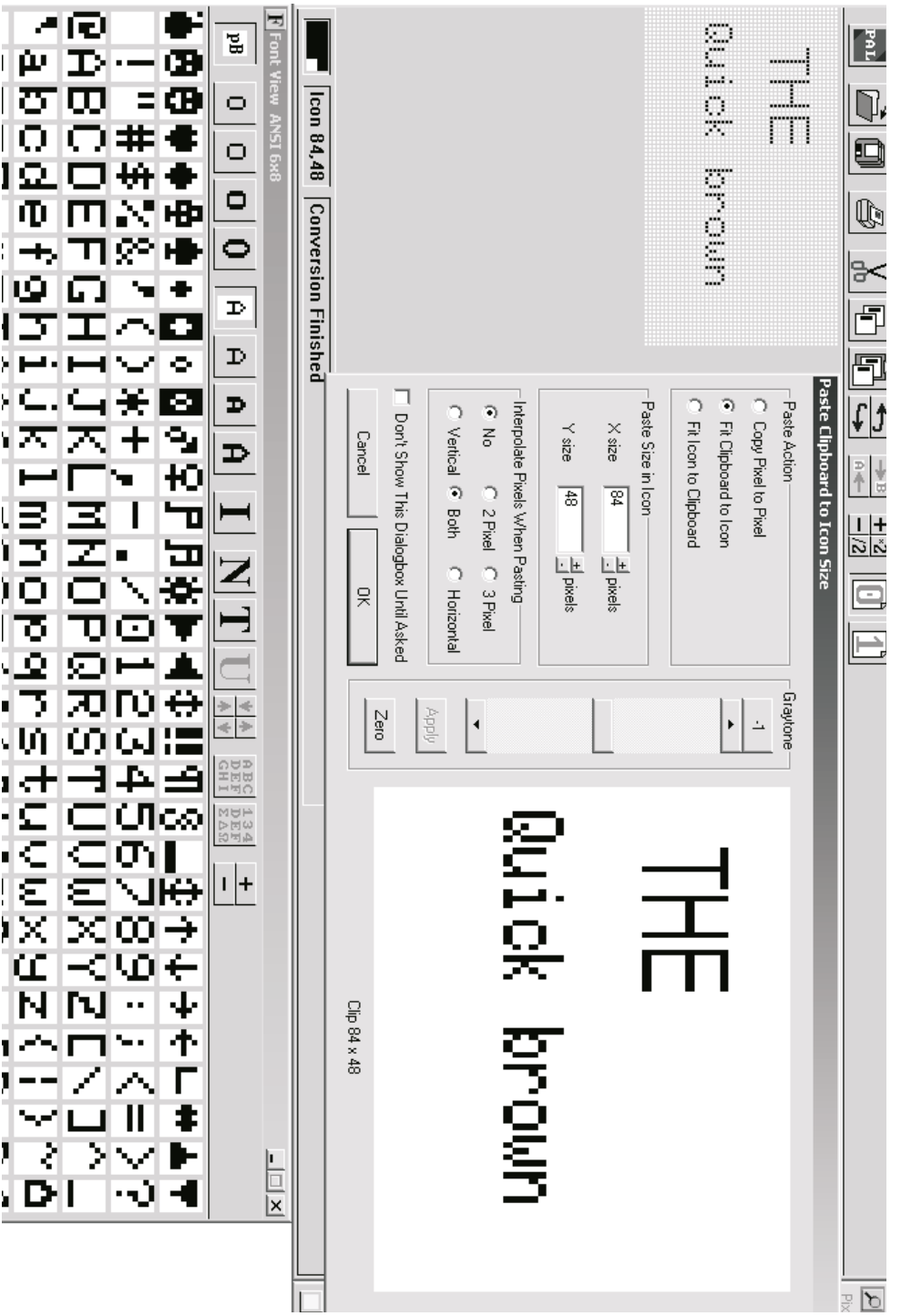
By clicking on the 'CUT' icon (the scissors image), we can paste the image into 'PAINT'. Ensuring that the image attributes of 84 × 48 pixels is correct, this is then saved to a BMP.file. This file is then converted to a byte table using Bmp2ASM. Unforunately the PIC16F84 has page boundaries for its memory and cannot accomodate the entire table in one block. However, by using the PIC18F252 which has a much large contiquous memory size of 32K bytes, this is achievable. The program FISH is written in 'C' for the PIC18F252 and demonstrates how a simple GIF animation of a swimming fish can be achieved on the Nokia 3310 display.



The Nokia 3310 connected to the PIC18F252.
*(Circuit for the FISH)-**Program FISH.C***

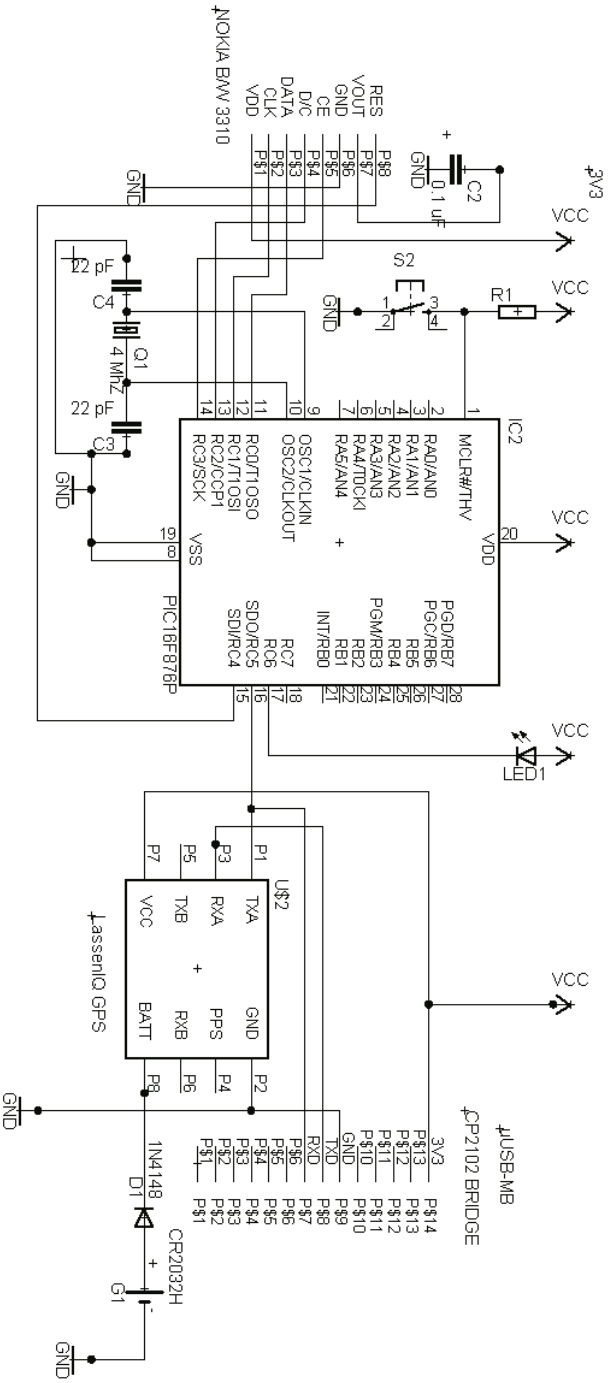


Constructing the display using ICONEDIT.

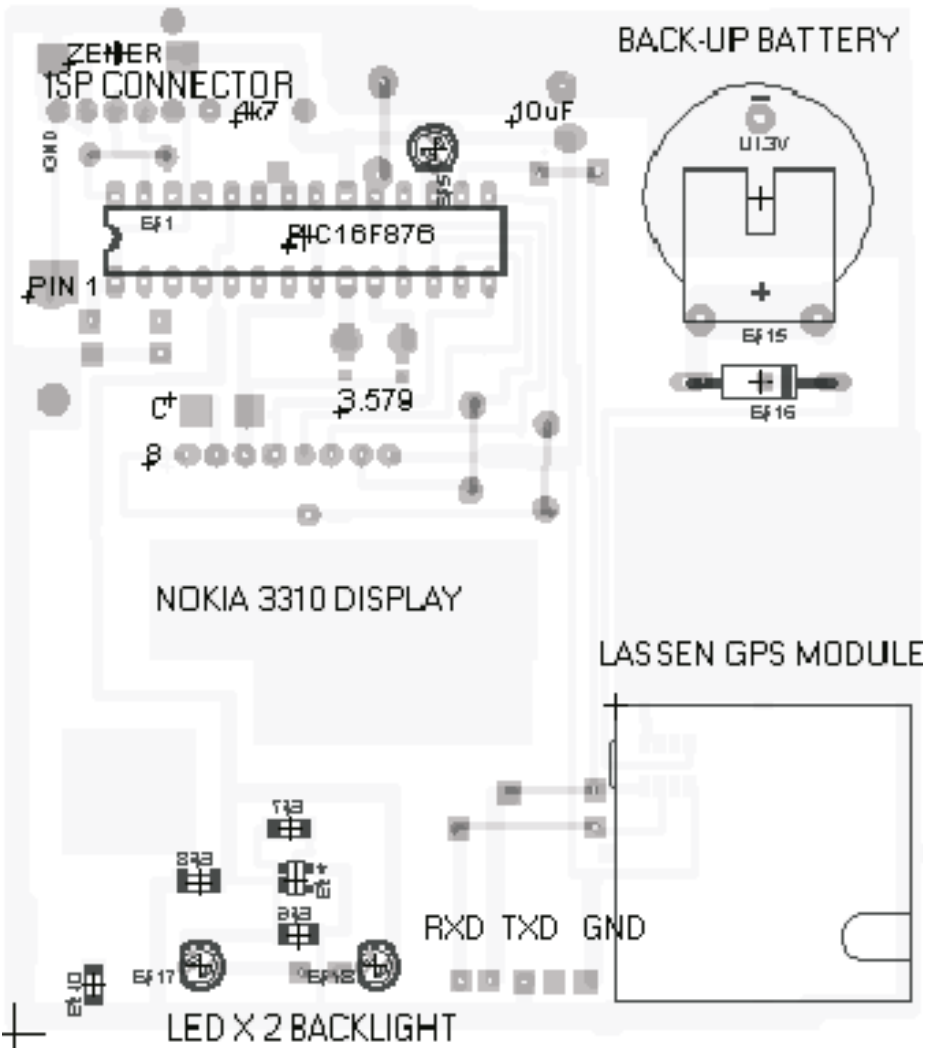


The 'ICONEDIT' PROGRAM

5.4 Case Study Nokia 3310 GPS digital clock



Display showing date and time.



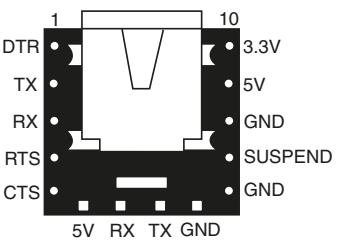
Introduction

The circuit diagram for this project is identical to the previous one, except for the fact that the PIC18F252 is replaced by the PIC16F876. This clock relies on the fact that the time/date of a GPS module can be received without the antenna being in sight of the sky, and also because it also contains an internal RTCC. A 3.3 volt DC switched mode mains adaptor is used to power the unit (available from Maplin Electronics).

The Lassen GPS module is a standard NMEA ASCII module that tracks 8 or 12 satellites and outputs ASCII data @ 4800 baud. The PDF file 'Lassen-GPS' (courtesy of ELEKTOR Electronics) describes how to connect and initialise the module. This clock takes the NMEA 'ZDA' data from the GPS receiver – containing date and time information – and displays it on a Nokia 3310 display.

The Lassen module needs to be initialised by connecting a RS232-bridge – in this case, 4D's microUSB bridge and running Trimble|Mon. The outgoing ASCII 4800 baud data stream can then be viewed on a terminal emulator such as T2.

Pin	Name	Description
1	DTR	Data Terminal Ready output (active low)
2	TX	Serial Data output (µUSB Transmit)
3	RX	Serial Data input (µUSB Recieve)
4	RTS	Ready To Send output (active low)
5	CTS	Clear To Send input (active low)
6	GND	Ground
7	SUSPEND	USB Suspend State (active high)
8	GND	Ground
9	5V	5V Power from USB (upto 500 mA)
10	3.3V	3.3V Power regulated (upto 100 mA)



µUSB-MB pin-out diagram.

Program 3310GPSX.asm for the GPS clock may be found in the CODE folder of chapter 5 on the ELEKTOR website.

Realtime Nokia 3310-based PDA

This project uses a PIC18F4550 with onboard USB transceiver to achieve realtime streaming of ASCII data from a PC terminal emulator direct to the 3310 LCD display. In addition, the received data is saved in non-volatile EEPROM for subsequent display after reset. Simply connect the unit to a PC USB port and send 84 text characters to fill the screen. Sending additional text will overwrite the display contents but will be saved in the

PIC's EEPROM up to a total of three screens of text (i.e. 84 × 3 = 252 bytes). On pressing the reset button, each screen (1 of 3) will be displayed at 5 second intervals. The USB connection can be reinstated at any time to change the PDA's text. Pressing '*' on the keyboard will zero the EEPROM address and start writing to the first line of the display. Since the PDA is powered by a 1.4 AmpH Lithium polymer battery, no provision is made for the PIC to sleep. The unit only consumes 600 uA of current in standby (with Backlight LEDs off). My friends and collegues always ask the proverbial question – 'what is it for?'. Well, in this case, I use the PDA as an 'electronic shopping list' instead of writing on scraps of paper. I just plug the USB cable into the unit, type in my shopping list for the day, unplug it, go shopping and press the reset button to view the PDA's contents.

Communications Device Class (CDC): RS-232 Emulation Firmware

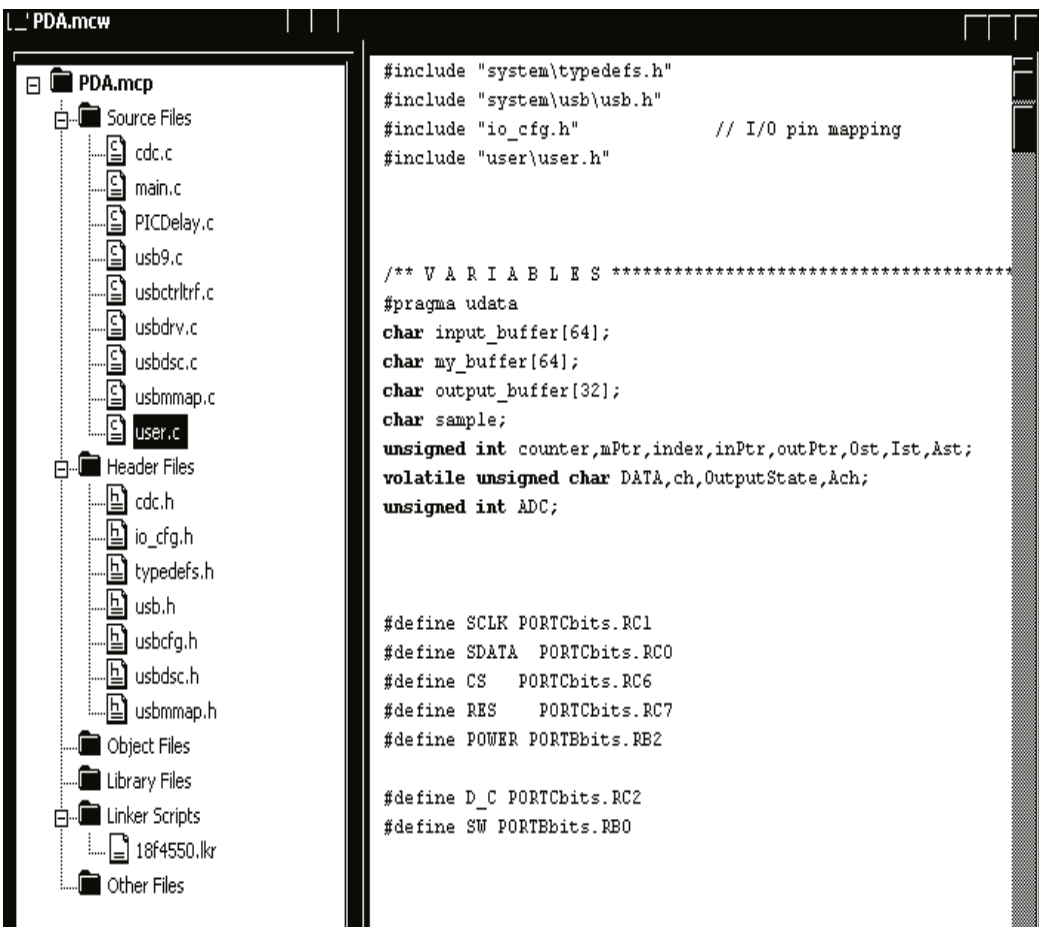
The software for the PDA is based on Microchips CDC example. Download the MCHPFUSB V1.3 USB FRAMEWORK file and install it on the PC. The CDC files then can be found in directory C:\MCHPFSUSB\fw\cdc. The firmware is intended to be used to provide RS-232 emulation over the USB port. For example, if a USB microcontroller is programmed with this firmware, upon plugging into the host PC, it will enumerate and appear in the device manager as a new communications port, with a new COM number. Windows-based applications can then interface with the embedded device by using standard function calls as would be used to interface with a standard RS-232-like communications port. Upon initially plugging in the USB cable to a device programmed with this CDC firmware, Windows will prompt the user for a driver. Upon receiving this prompt, point Windows to the set-up file located in the 'C:\MCHPFSUSB\driver\win2k_winxp' folder.

If we examine the pda.mcp file for the project shown below, it can be seen that there are many source and header files. Fortunately, all these files have been written by Microchip and form a 'Framework' around which a user can customise the program for their own use. The only file that needs editing is the user.c file. The following lists of routines are additionally included in the user.c file for the PDA:

```
void writecom(char command_in);                 /* USER ROUTINE */
void clearram(void);                            /* USER ROUTINE */
void initlcd(void);                             /* USER ROUTINE */
void delay10( char n);                          /* USER ROUTINE */
void clockdata(char bits_in);                   /* USER ROUTINE */
void cursorxy(char x, char y);                  /* USER ROUTINE */
void writedata(char data_in);                   /* USER ROUTINE */
void lcdchar (char c );                         /* USER ROUTINE */
void EEPROM_Write(unsigned int address, unsigned char data);
                                                /* USER ROUTINE */
void EEPROM_Read(unsigned int address);         /* USER ROUTINE */
void readeeprom();                              /* USER ROUTINE */
```

Lithium Polymer batteries

Unlike other battery technologies, these batteries need special handling. Even a momentary short across the battery will cause instantaneously destruction with a possibility of fire and explosion and the escape of noxious gasses. It is good practice to include a 250-mA fuse in series with the power supply, since a copper power supply track may be insufficient to act as a fuse. Also, if pierced by a sharp object the same can happen. The author protects the lithium battery by sandwiching it between a thin aluminium sheet and the copper side of the PCB with a layer of 'no more nails' adhesive to isolate the battery from any component solder 'spikes'. These disadvantages are more than offset by the advantages of using these types of batteries. A very large ampere-hour capacity/unit size and a very low power-to-weight ratio makes them ideal for use in embedded controller portable applications.



Program user.c

```
/**I N C L U D E S **************************************************/
#include <p18cxxx.h>
#include "system\typedefs.h"
#include "system\usb\usb.h"
#include "io_cfg.h"                 // I/O pin mapping
#include "user\user.h"

/**V A R I A B L E S*************************************************/
#pragma udata
char input_buffer[64];
char my_buffer[64];
char output_buffer[32];
char sample;
unsigned int counter,mPtr,index,inPtr,outPtr,Ost,Ist,Ast;
volatile unsigned char DATA,ch,OutputState,Ach;

#define SCLK PORTCbits.RC1          /*  port connections for the Nokia display */
#define SDATA          PORTCbits.RC0
#define CS       PORTCbits.RC6
#define RES          PORTCbits.RC7
#define POWER    PORTBbits.RB2

#define D_C PORTCbits.RC2
#define SW PORTBbits.RB0        /* unused */

/** P R I V A T E  P R O T O T Y P E S*********************************/
void User_Process(void);
 void Check_Update_Input(void);
void update_bit(unsigned char Chan,unsigned char Data);
void GetInputStatus(void);
 void writecom(char command_in);                    /* USER ROUTINE */
void clearram(void);                                /* USER ROUTINE */
void initlcd(void);                                 /* USER ROUTINE */
void delay10( char n);                              /* USER ROUTINE */
void clockdata(char bits_in);                       /* USER ROUTINE */
void cursorxy(char x, char y);                      /* USER ROUTINE */
void writedata(char data_in);                       /* USER ROUTINE */
void lcdchar (char c );                             /* USER ROUTINE */
void EEPROM_Write(unsigned int address, unsigned char data);
                                                    /* USER ROUTINE */
void EEPROM_Read(unsigned int address);         /* USER ROUTINE */
```

```
void readeeprom();                              /* USER ROUTINE */
#pragma romdata ROOT_TABLE=0x1000

rom const char table []  [5]= {0x00,0x00,0x00,0x00,0x00,      // 20 space
            ASCII table for Nokia LCD: 96 rows * 5 bytes= 480 bytes
0x00,0x00,0x5f,0x00,0x00,          // 21 ! Note that this is the same set of codes for
                                   character you would find on a HD44780 based
                                   character LCD.
        0x00,0x07,0x00,0x07,0x00,          // 22 "
        0x14,0x7f,0x14,0x7f,0x14,          // 23 #
                                   Also, given the size of the LCD (84 pixels by 48
                                   pixels), the maximum number of characters per
                                   row is only 14. :)

    0x24,0x2a,0x7f,0x2a,0x12,     // 24 $
    0x23,0x13,0x08,0x64,0x62,     // 25 %
    0x36,0x49,0x55,0x22,0x50,     // 26 &
    0x00,0x05,0x03,0x00,0x00,     // 27 ‘
    0x00,0x1c,0x22,0x41,0x00,     // 28 (
    0x00,0x41,0x22,0x1c,0x00,     // 29 )
    0x14,0x08,0x3e,0x08,0x14,     // 2a *
    0x08,0x08,0x3e,0x08,0x08,     // 2b +
    0x00,0x50,0x30,0x00,0x00,     // 2c ,
    0x08,0x08,0x08,0x08,0x08,     // 2d -
    0x00,0x60,0x60,0x00,0x00,     // 2e .
    0x20,0x10,0x08,0x04,0x02,     // 2f /
    0x3e,0x51,0x49,0x45,0x3e,     // 30 0
    0x00,0x42,0x7f,0x40,0x00,     // 31 1
    0x42,0x61,0x51,0x49,0x46,     // 32 2
    0x21,0x41,0x45,0x4b,0x31,     // 33 3
    0x18,0x14,0x12,0x7f,0x10,     // 34 4
    0x27,0x45,0x45,0x45,0x39,     // 35 5
    0x3c,0x4a,0x49,0x49,0x30,     // 36 6
    0x01,0x71,0x09,0x05,0x03,     // 37 7
    0x36,0x49,0x49,0x49,0x36,     // 38 8
    0x06,0x49,0x49,0x29,0x1e,     // 39 9
    0x00,0x36,0x36,0x00,0x00,     // 3a :
    0x00,0x56,0x36,0x00,0x00,     // 3b ;
    0x08,0x14,0x22,0x41,0x00,     // 3c <
    0x14,0x14,0x14,0x14,0x14,     // 3d =
    0x00,0x41,0x22,0x14,0x08,     // 3e >
    0x02,0x01,0x51,0x09,0x06,     // 3f ?
    0x32,0x49,0x79,0x41,0x3e,     // 40 @
```

```
0x7e,0x11,0x11,0x11,0x7e,   // 41 A
0x7f,0x49,0x49,0x49,0x36,   // 42 B
0x3e,0x41,0x41,0x41,0x22,   // 43 C
0x7f,0x41,0x41,0x22,0x1c,   // 44 D
0x7f,0x49,0x49,0x49,0x41,   // 45 E
0x7f,0x09,0x09,0x09,0x01,   // 46 F
0x3e,0x41,0x49,0x49,0x7a,   // 47 G
0x7f,0x08,0x08,0x08,0x7f,   // 48 H
0x00,0x41,0x7f,0x41,0x00,   // 49 I
0x20,0x40,0x41,0x3f,0x01,   // 4a J
0x7f,0x08,0x14,0x22,0x41,   // 4b K
0x7f,0x40,0x40,0x40,0x40,   // 4c L
0x7f,0x02,0x0c,0x02,0x7f,   // 4d M
0x7f,0x04,0x08,0x10,0x7f,   // 4e N
0x3e,0x41,0x41,0x41,0x3e,   // 4f O
0x7f,0x09,0x09,0x09,0x06,   // 50 P
0x3e,0x41,0x51,0x21,0x5e,   // 51 Q
0x7f,0x09,0x19,0x29,0x46,   // 52 R
0x46,0x49,0x49,0x49,0x31,   // 53 S
0x01,0x01,0x7f,0x01,0x01,   // 54 T
0x3f, 0x40,0x40,0x40,0x3f,  // 55 U
0x1f,0x20,0x40,0x20,0x1f,   // 56 V
0x3f,0x40,0x38,0x40,0x3f,   // 57 W
0x63,0x14,0x08,0x14,0x63,   // 58 X
0x07,0x08,0x70,0x08,0x07,   // 59 Y
0x61,0x51,0x49,0x45,0x43,   // 5a Z
0x00,0x7f,0x41,0x41,0x00,   // 5b [
0x02,0x04,0x08,0x10,0x20,   // 5c Yen Currency
0x00,0x41,0x41,0x7f,0x00,   // 5d ]
0x04,0x02,0x01,0x02,0x04,   // 5e ^
0x40,0x40,0x40,0x40,0x40,   // 5f _
0x00,0x01,0x02,0x04,0x00,   // 60 `
0x20,0x54,0x54,0x54,0x78,   // 61 a
0x7f,0x48,0x44,0x44,0x38,   // 62 b
0x38,0x44,0x44,0x44,0x20,   // 63 c
0x38,0x44,0x44,0x48,0x7f,   // 64 d
0x38,0x54,0x54,0x54,0x18,   // 65 e
0x08,0x7e,0x09,0x01,0x02,   // 66 f
0x0c,0x52,0x52,0x52,0x3e,   // 67 g
0x7f,0x08,0x04,0x04,0x78,   // 68 h
0x00,0x44,0x7d,0x40,0x00,   // 69 i
0x20,0x40,0x44,0x3d,0x00,   // 6a j
0x7f,0x10,0x28,0x44,0x00,   // 6b k
```

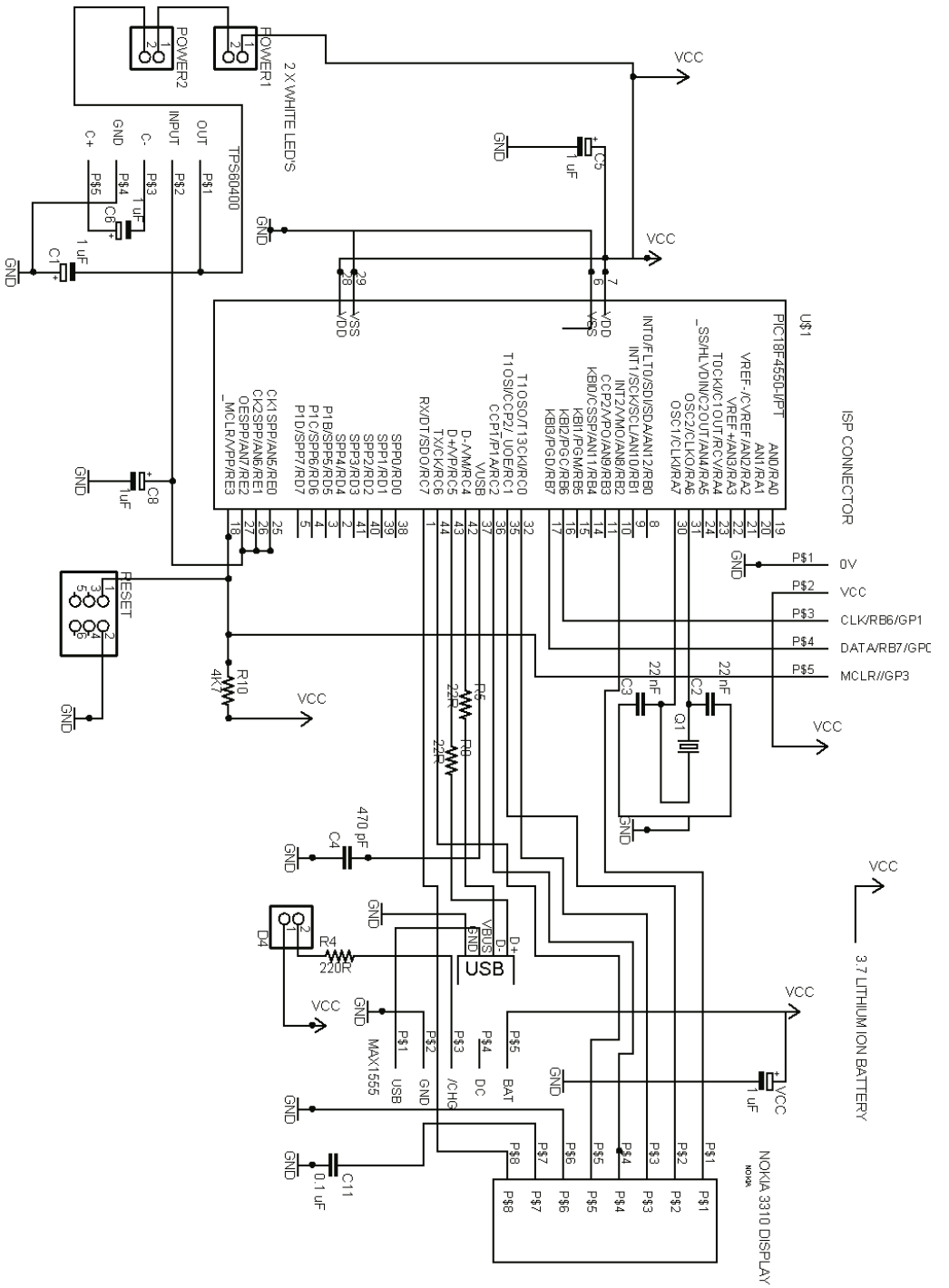
```
    0x00,0x41,0x7f,0x40,0x00,   // 6c l
    0x7c,0x04,0x18,0x04,0x78,   // 6d m
    0x7c,0x08,0x04,0x04,0x78,   // 6e n
    0x38,0x44,0x44,0x44,0x38,   // 6f o
    0x7c,0x14,0x14,0x14,0x08,   // 70 p
    0x08,0x14,0x14,0x18,0x7c,   // 71 q
    0x7c,0x08,0x04,0x04,0x08,   // 72 r
    0x48,0x54,0x54,0x54,0x20,   // 73 s
    0x04,0x3f,0x44,0x40,0x20,   // 74 t
    0x3c,0x40,0x40,0x20,0x7c,   // 75 u
    0x1c,0x20,0x40,0x20,0x1c,   // 76 v
    0x3c,0x40,0x30,0x40,0x3c,   // 77 w
    0x44,0x28,0x10,0x28,0x44,   // 78 x
    0x0c,0x50,0x50,0x50,0x3c,   // 79 y
    0x44,0x64,0x54,0x4c,0x44,   // 7a z
    0x00,0x08,0x36,0x41,0x00,   // 7b <
    0x00,0x00,0x7f,0x00,0x00,   // 7c |
    0x00,0x41,0x36,0x08,0x00,   // 7d >
    0x10,0x08,0x08,0x10,0x08,   // 7e Right Arrow  ->
    0x78,0x46,0x41,0x46,0x78};  // 7f Left Arrow    <-

#pragma code
void UserInit(void)
{

    mInitInput();
    mInitOutput();
    inPtr=outPtr=Ost=Ist=Ast=0;

    initlcd();              /* initialise the display */
    readeeprom();           /* read EEPROM memory and output to display on reset*/
}//end UserInit

void ProcessIO(void)
{
    // User Application USB tasks
    if((usb_device_state < CONFIGURED_STATE)||(UCONbits.SUSPND==1)) return;

    User_Process();

}//end ProcessIO
```

```
}
//----------------------------
// Update Output                    MAIN PROGRAM FOR PDA
//----------------------------
void Check_Update_Output(void)
{
    EEPROM_Write(sample,DATA);      /*save ASCII data in EEPROM */
    lcdchar(DATA);                  /* write ASCII received from
                                    USB 'usart' to display */

    writedata(0);

    if(DATA=='*')

{
    clearram();
    }

}

//**************************************
// User change here
//**************************************
void User_Process(void)
{

    if(getsUSBUSART(input_buffer,32))
    {
        mPtr = mCDCGetRxLength();
        for(index = 0; index < mPtr; index++)
        {
            my_buffer[inPtr]= input_buffer[index];
            inPtr++;
            if (inPtr==64) inPtr=0;
        }
    }

    if (inPtr!=outPtr)
    {
        DATA=my_buffer[outPtr];
        outPtr++;
        if (outPtr==64) outPtr=0;
```

```
        Check_Update_Output();

    }

}//end

void initlcd(void)              / *INITIALISE THE NOKIA 3310 DISPLAY  */
{   sample=0;
    POWER =1;                   /*connect power to display */

    SleepMS(10);                // Wait 100ms.

        RES = 1;                // Set _RES HIGH.
        SCLK = 1;               // Disable Chip.
        RES = 0;                // Reset the LCD.

    SleepMS(10);                // Wait 100ms.

    RES = 1;                    // Awake LCD from RESET state.

        writecom(0x21);         // Activate Chip and H=1.
        writecom(0xc8);         // c2 Set LCD voltage to about 7V.   vop
        writecom(0x13);         // Adjust voltage bias.
        writecom(0x20);         // Horizontal addressing and H=0.
        writecom(0x09);         // Activate all segments.
        clearram();             // Erase all pixels on the DDRAM.
        writecom(0x08);         // Blank the display.

    SleepMS(10);  // Wait 100ms.
        writecom(0x0C);         // Display Normal.
        writecom(0x40);
        writecom(0x80);

        cursorxy(0,0);          // Cursor Home.
    SleepMS(10);                // Wait 100ms.

         clearram();
    SleepMS(10);                // Wait 100ms.
}

void clearram(void)             /*  clear display */
{
        int ddram;
```

```
    sample=0;
        cursorxy(0,0);                  // Cursor Home.
        for (ddram=504;ddram>0;ddram--)     {writedata(0x00);}
                                        // 6*84 = 504 DDRAM addresses.
}

void cursorxy(char x, char y)
{
    writecom(0x40|(y&0x07));            // Y axis
    writecom(0x80|(x&0x7f));            // X axis
}

void writecom(char command_in)
{
    D_C = 0;                            // Select Command register.
    CS = 0;                             // Select Chip.
    clockdata(command_in);              // Clock in command bits.
    CS = 1;                             // Deselect Chip.
}

void writedata(char data_in)
{
    D_C = 1;                            // Select Data register.
    CS = 0;                             // Select Chip.
    clockdata(data_in);                 // Clock in data bits.
    CS = 1;                             // Deselect Chip.
}

void clockdata(char bits_in)
{
    int bitcnt;
    for (bitcnt=8; bitcnt>0; bitcnt--)
        {
            SCLK = 0;
            if ((bits_in&0x80)==0x80) {SDATA=1;}
            else {SDATA=0;}
            SCLK = 1;
            bits_in=bits_in<<1;
        }
}
```

```
void lcdchar (char c )
{

    int i;
    char s;
    for (i=0; i<5; i++)

{

    writedata(table[c-32]  [i]  );          /*  GET ASCII CHARACTER AND
                                            CONVERT TO 5 BYTES (pixel information)
                                            using lookup table and send to display */
    }
}


void EEPROM_Write(unsigned int address, unsigned char data)
{

    unsigned char return_value;
    unsigned char temp_GIEH;
    unsigned char temp_GIEL;

                        // save the state of the interrupt able bits
                        temp_GIEH = INTCONbits.GIEH;
                        temp_GIEL = INTCONbits.GIEL;

                // EEPROM operation (as opposed to a flash memory operation)
    EECON1bits.EEPGD = 0;

                // Only do a write
    EECON1bits.FREE = 0;

                // make sure the EEPROM write done flag is reset
    PIR2bits.EEIF = 0;

                // set EEPROM address
    EEADR =sample ;
    EEDATA = data;
    EECON1bits.WREN = 1;

        INTCONbits.GIEH = 0;
        INTCONbits.GIEL = 0;
```



Circuit diagram of the PDA.

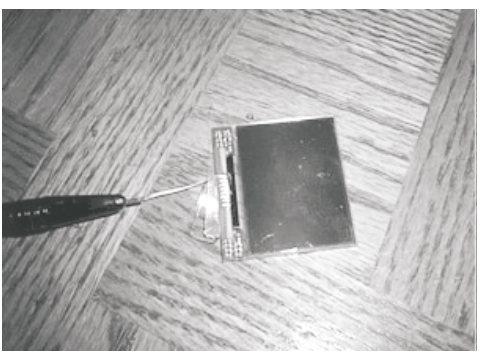
```
    EECON2 = 0x55;
    EECON2 = 0xAA;

    EECON1bits.WR = 1;

        INTCONbits.GIEH = temp_GIEH;
        INTCONbits.GIEL = temp_GIEL;

            while(PIR2bits.EEIF == 0);

        PIR2bits.EEIF = 0;

        sample=sample+1;
    }

 void EEPROM_Read(unsigned int address)
{

    EECON1bits.EEPGD = 0;
    EEADR = sample;
    EECON1bits.RD = 1;

}

void readeeprom()           /*read eeprom and send screenful of data to display */

{   char number;
    char next1;
    char next2;

    for (next2=0; next2<3;next2++)          /*continue reading eeprom for next two
                                            screens at 5 second intervals */

{

    for (next1=0; next1<6;next1++)          */ 6 rows per display */

{
    for (number=0; number<14; number++)     /* 14 characters per line */

{
    EEPROM_Read(sample);
```

```
    lcdchar(EEDATA);      /* convert each eeprom character to 5 pixel data bytes */
    writedata(0);         */ a space between each character on the display */
    sample=sample+1;      /* eeprom address  */
}

    }

    SleepMS(9000);        /* adjust to vary lcd viewing time per frame  */
    SleepMS(9000);
    SleepMS(9000);
    SleepMS(9000);

    }

}
```

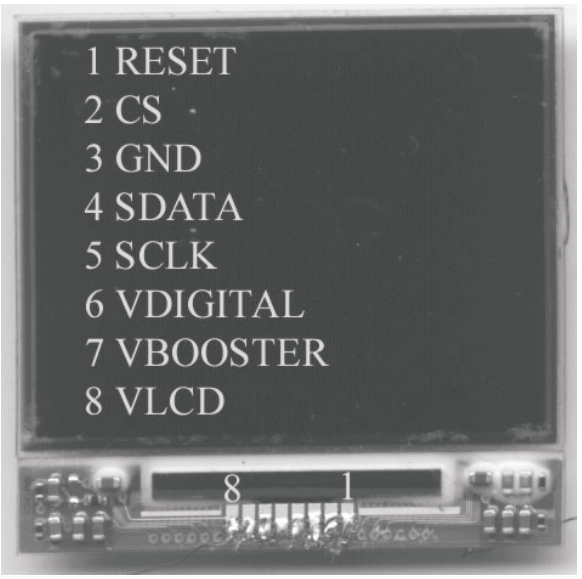
5.5 Case Study Nokia 3510i Electronic compass

Connecting the 3510i display

When the display is purchased, be aware that it also contains the plastic membrane for the phone which has to be carefully seperated. First, gently remove the metal display casing. The display can then be removed by bending the plastic at the top of the display by about 10 mm and carefully pulling it out. Some resistance will be encountered which is due to a small amount of impact adhesive. Connections are then made to the eight contacts of the display by first tinning – use solder wick to remove any excess – and using a 'Roadrunner' wiring pencil and a steady hand. The same pro-

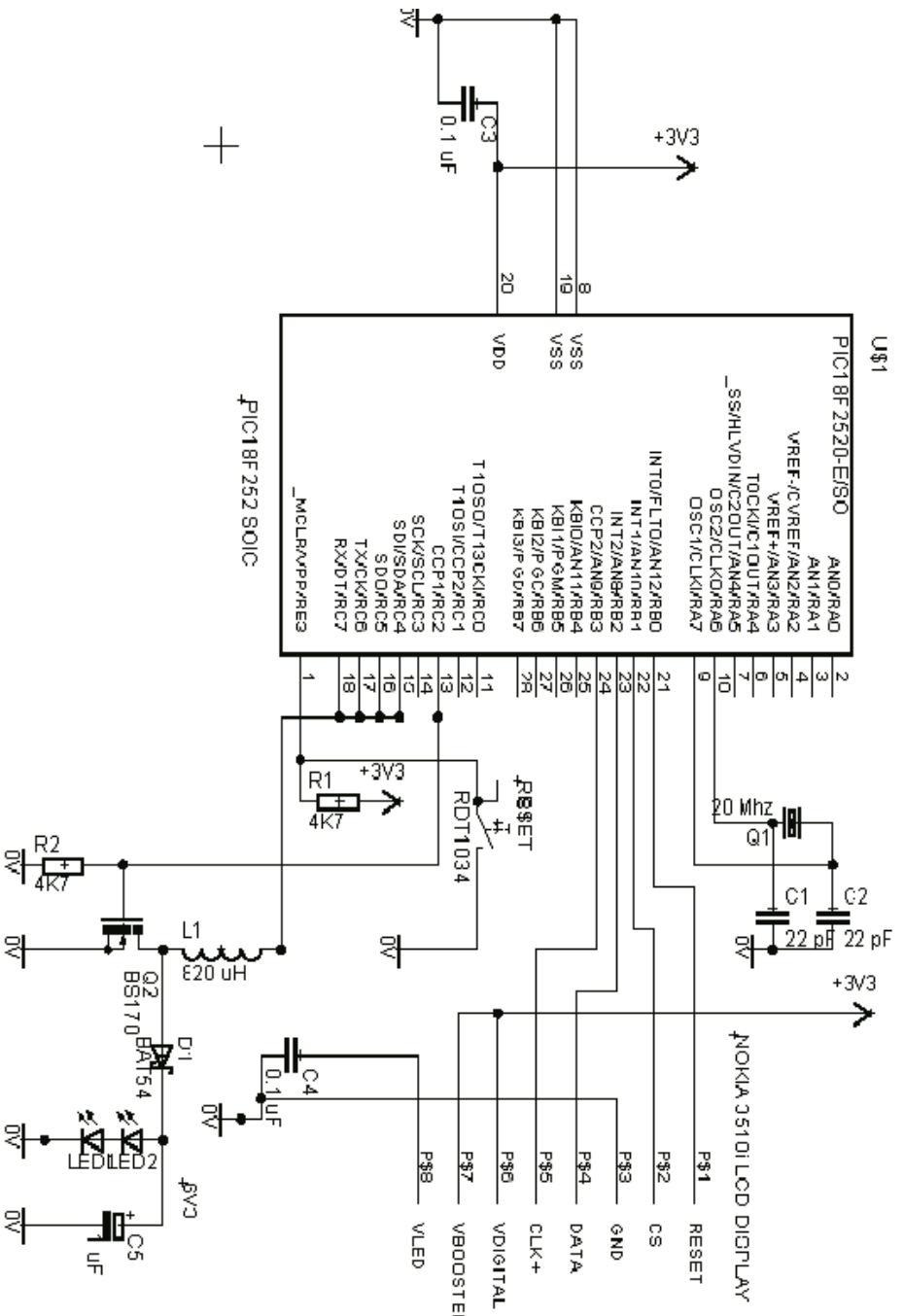


cedure is used for the Nokia 3310 LCD – however, the metal cover is secured by small plastic rivets which must be carefully cut to extricate the display.

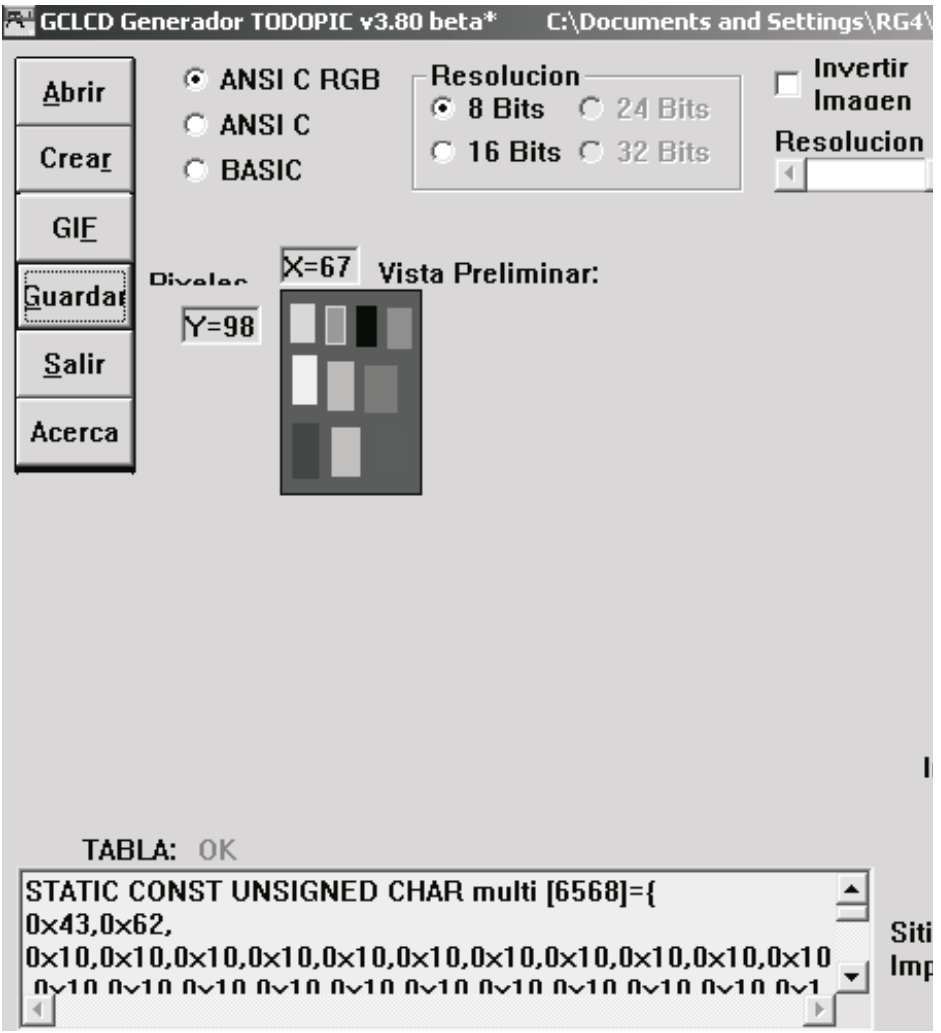


The 3510i colour LCD display has a 98 × 67 pixel resolution and displays 256 colours. Unlike B/W displays where each pixel is defined by one bit, i.e. on or off, Each pixel in this colour display requires 8 bits of data to determine the colour of the pixel. The RGB data consists of a RRRGGGBB pattern (one byte) so for example to print a red dot on the display, 11100000 (E0H) is sent to the display. A yellow dot would have a binary pattern of 11111100. So it can be seen that eight times more memory capacity is required for a colour display than a B/W display. Fortunately, the PIC18F252 has 32 Kbytes of non-volatile memory sufficient for 4 screens of colour graphics for the 3510i, each being displayed one after the other.

In the circuit diagram below, a 20 MHz clock source is used, since generating colour graphics is time intensive. In order to speed up the generation of the screen graphics, the PLL fuse in the PIC18F252 is selected for the clock source, providing an operating frequency of 4 × 20 MHz (80 MHz clock). Full-screen refreshing of data can then take place in a fraction of a second. A colour BMP image is digitised using the GCLCD program as below.



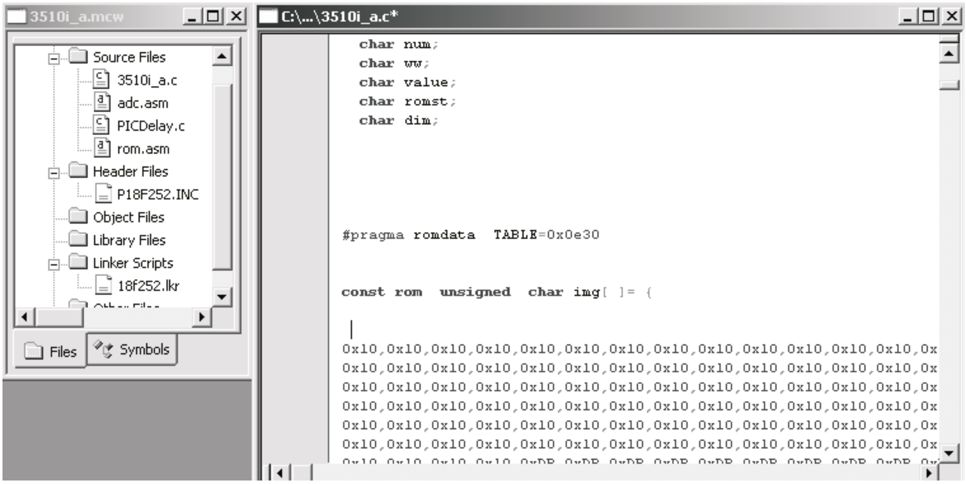
PROGRAM 3510I_a.C



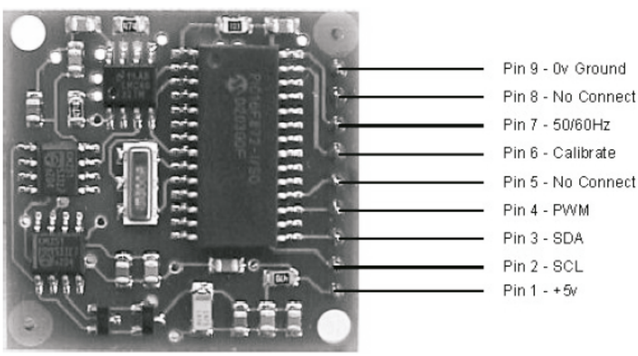
GCLCD digitiser.

Command ‘Abrir’ imports a 98 × 67 colour image, and selecting ‘Crear’ creates a byte table. Save this as a ‘C’ file by pressing ‘Guardar’.

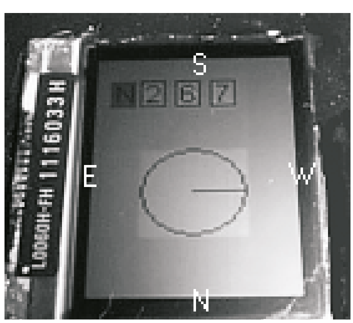
A total of 6568 bytes are generated for the complete picture and stored in file ‘Image.C’. This is then imported into the main program software as a ‘romdata’ table. **Program 3510i_a** contains two image tables which are displayed at two-second intervals.



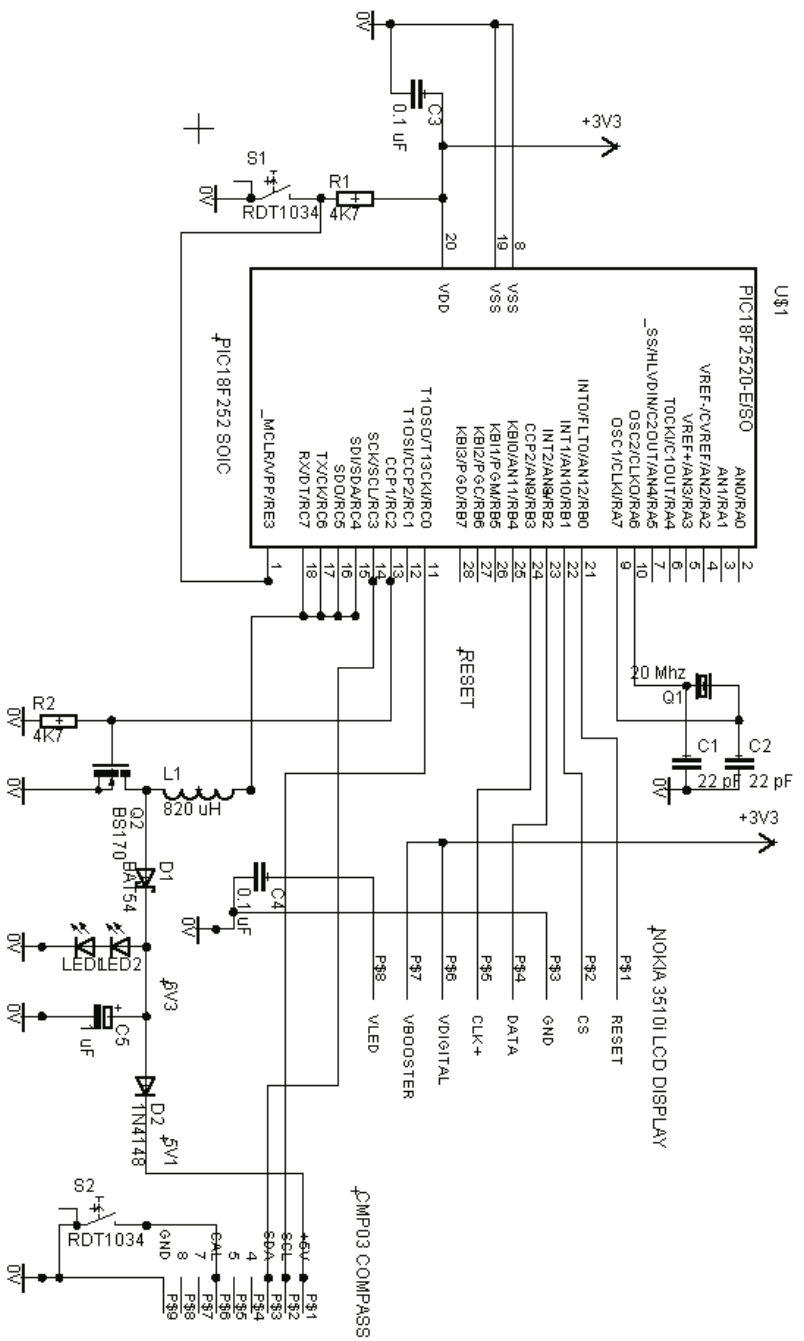
The electronic compass uses an identical circuit, as above, with the addition of a CMP03 electronic compass. It operates from a 3.6 volt lithium-ion battery. A switched mode supply consisting of L1, T1 and D1 provides a +6.2 voltage for the two back-lighting white LEDs, and also the 5 volt supply for the compass via a 1N4148 dropping diode. An LED is included which lights when the compass is pointing to magnetic north.



The CMP03 Compass module.

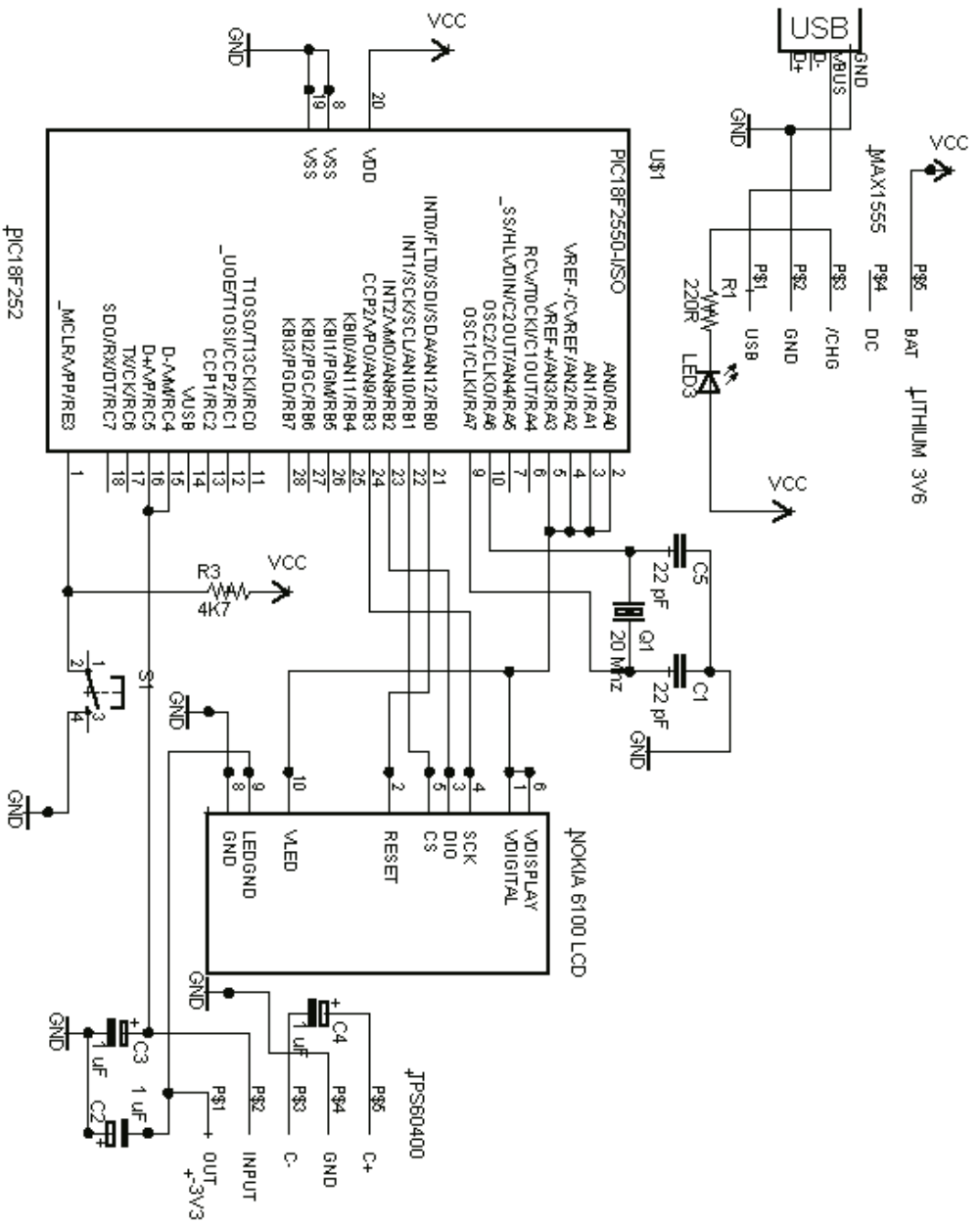


Digital and analogue readout on the display.



The electronic compass schematic.
PROGRAM NOKIA-COMPASS_LED.C

5.6 Case Study Nokia 6100 Epson display 8 bit colour



The Nokia -EPSON/PHILLIPS 6100 CIRCUIT

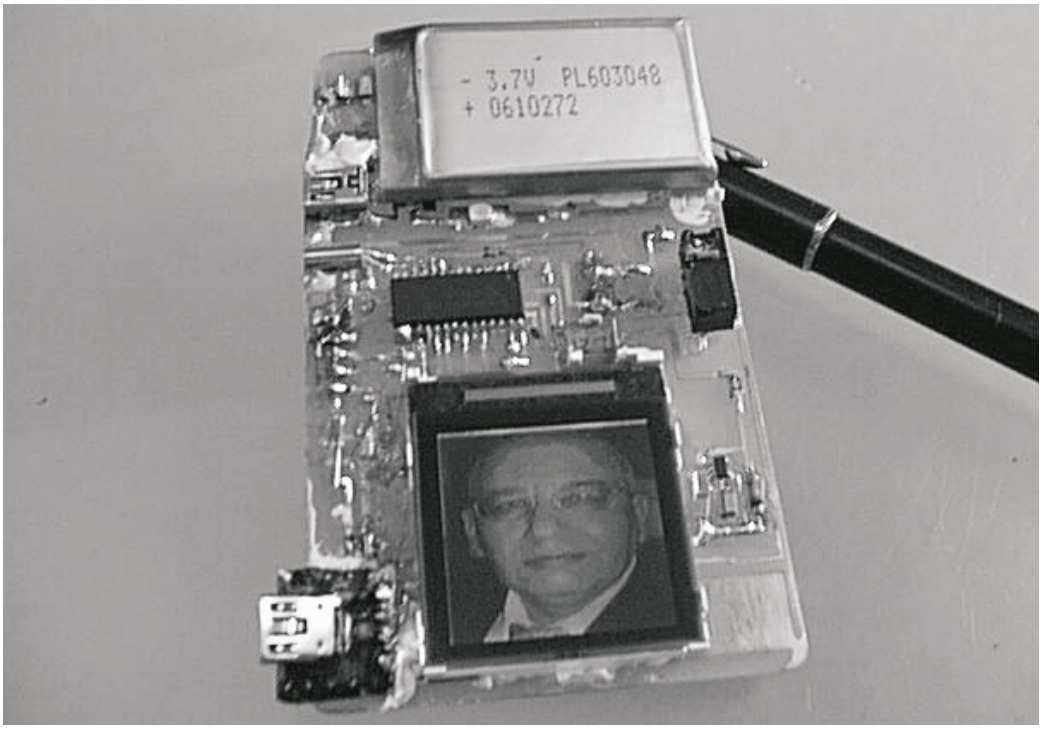


Photo of the Author on the 6100 colour display.

Introduction

The Nokia 6100 LCD has a 131 × 131 pixel resolution and is available with two types of onboard controller chip – the EPSON with 8 bit colour from 'Sparkfun Electronics', and the Philips chip with a 16 bit colour pallette available from 'Jelu'. Note that sourcing this LCD from other sources can produce problems, as there are also at least three other types of 6100s with unknown chip drivers and are therefore impossible to program.

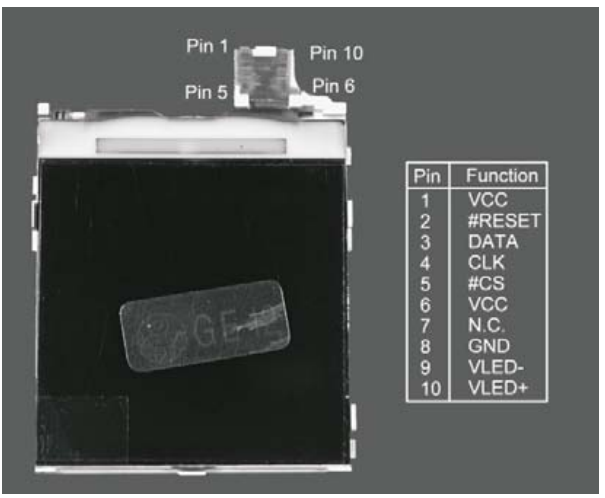
A detailed explaination of the command set for this display can be found in the PDF file 'SID15G00' . An example program '**Sparkfunx.C**' is provided in the part 5 code folder of the relevant elektor download

	Command	A0	$\overline{RD}$	$\overline{WR}$	D7	D6	D5	D4	D3	D2	D1	D0	Function	Hex	Parameter
1	DISON	0	1	0	1	0	1	0	1	1	1	1	Display on	AF	None
2	DISOFF	0	1	0	1	0	1	0	1	1	1	0	Display off	AE	None
3	DISNOR	0	1	0	1	0	1	0	0	1	1	0	Normal display	A6	None
4	DISINV	0	1	0	1	0	1	0	0	1	1	1	Inverse display	A7	None
5	COMSCN	0	1	0	1	0	1	1	1	0	1	1	Common scan direction	BB	1byte
6	DISCTL	0	1	0	1	1	0	0	1	0	1	0	Display control	CA	3byte
7	SLPIN	0	1	0	1	0	0	1	0	1	0	1	Sleep in	95	None
8	SLPOUT	0	1	0	1	0	0	1	0	1	0	0	Sleep out	94	None
9	PASET	0	1	0	0	1	1	1	0	1	0	1	Page address set	75	2byte
10	CASET	0	1	0	0	0	0	1	0	1	0	1	Column address set	15	2byte
11	DATCTL	0	1	0	1	0	1	1	1	1	0	0	Data scan direction, etc.	BC	3byte
12	RGBSET8	0	1	0	1	1	0	0	1	1	1	0	256-color position set	CE	20byte
13	RAMWR	0	1	0	0	1	0	1	1	1	0	0	Writing to memory	5C	Data
14	RAMRD	0	1	0	0	1	0	1	1	1	0	1	Reading from memory	5D	Data
15	PTLIN	0	1	0	1	0	1	0	1	0	0	0	Partial display in	A8	2byte
16	PTLOUT	0	1	0	1	0	1	0	1	0	0	1	Partial display out	A9	None
17	RMWIN	0	1	0	1	1	1	0	0	0	0	0	Read and modify write	E0	None
18	RMWOUT	0	1	0	1	1	1	0	1	1	1	0	End	EE	None
19	ASCSET	0	1	0	1	0	1	0	1	0	1	0	Area scroll set	AA	4byte
20	SCSTART	0	1	0	1	0	1	0	1	0	1	1	Scroll start set	AB	1byte
21	OSCON	0	1	0	1	1	0	1	0	0	0	1	Internal oscillation on	D1	None
22	OSCOFF	0	1	0	1	1	0	1	0	0	1	0	Internal oscillation off	D2	None
23	PWRCTR	0	1	0	0	0	1	0	0	0	0	0	Power control	20	1byte
24	VOLCTR	0	1	0	1	0	0	0	0	0	0	1	Electronic volume control	81	2byte
25	VOLUP	0	1	0	1	1	0	1	0	1	1	0	Increment electronic control by 1	D6	None
26	VOLDOWN	0	1	0	1	1	0	1	0	1	1	1	Decrement electronic control by 1	D7	None
27	TMPGRD	0	1	0	1	0	0	0	0	0	1	0	Temperature gradient set	82	1 byte
28	EPCTIN	0	1	0	1	1	0	0	1	1	0	1	Control EEPROM	CD	1 byte
29	EPCOUT	0	1	0	1	1	0	0	1	1	0	0	Cancel EEPROM control	CC	None
30	EPMWR	0	1	0	1	1	1	1	1	1	0	0	Write into EEPROM	FC	None
31	EPMRD	0	1	0	1	1	1	1	1	1	0	1	Read from EEPROM	FD	None
32	EPSRRD1	0	1	0	0	1	1	1	1	1	0	0	Read register 1	7C	None
33	EPSRRD2	0	1	0	0	1	1	1	1	1	0	1	Read register 2	7D	None
34	NOP	0	1	0	0	0	1	0	0	1	0	1	NOP instruction	25	None
35	STREAD	0	0	1	Status								Status read		

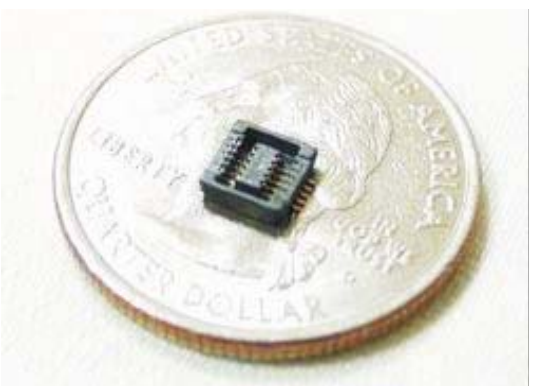
The command set for the 'Sparkfun' 6100 display.

Interfacing the 6100 display

This display is available as a 'stand-alone' unit with a sub-minature connector which mates with a surface mount socket. Soldering this 4 × 4 mm mini connector to a PCB requires a high degree of skill, although 'breakout' boards with 2.54 mm pin spacing are available. Again, the 8 bit colour format of each colour pixel is arranged as RRRGGGBB, and the image is digitised using 'GLCD'.



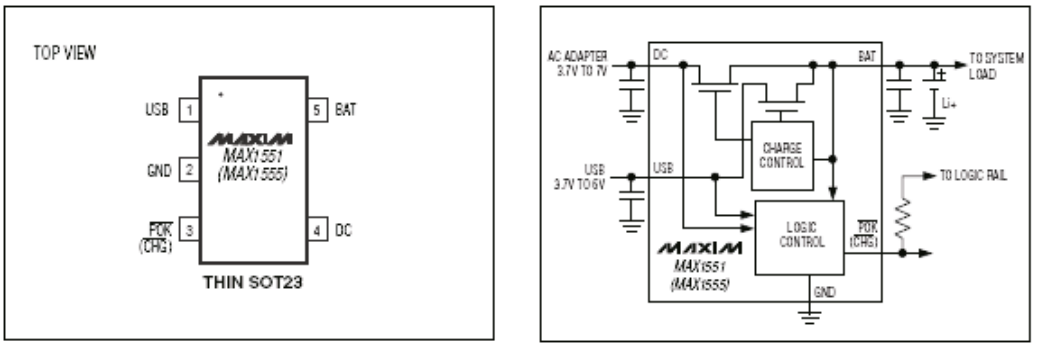
Pin-out of the 6100 display.



Surface mount connector.

Description of circuit

The power supply is derived from a 3.6 volt lithium battery. An onboard MAX1555 charges this battery from a usb source. Just plug in a USB cable from the PC to the mini surface mount USB connector. A single LED indicates that charging is in effect. When the LED goes out, the battery is fully charged.

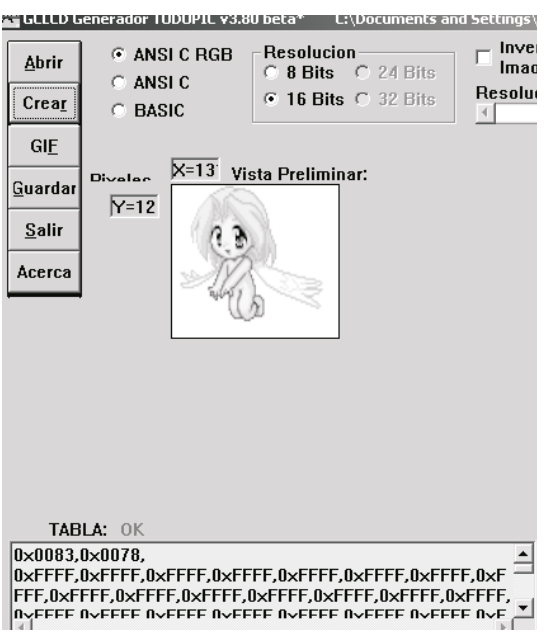


MAX1555 battery charging details.

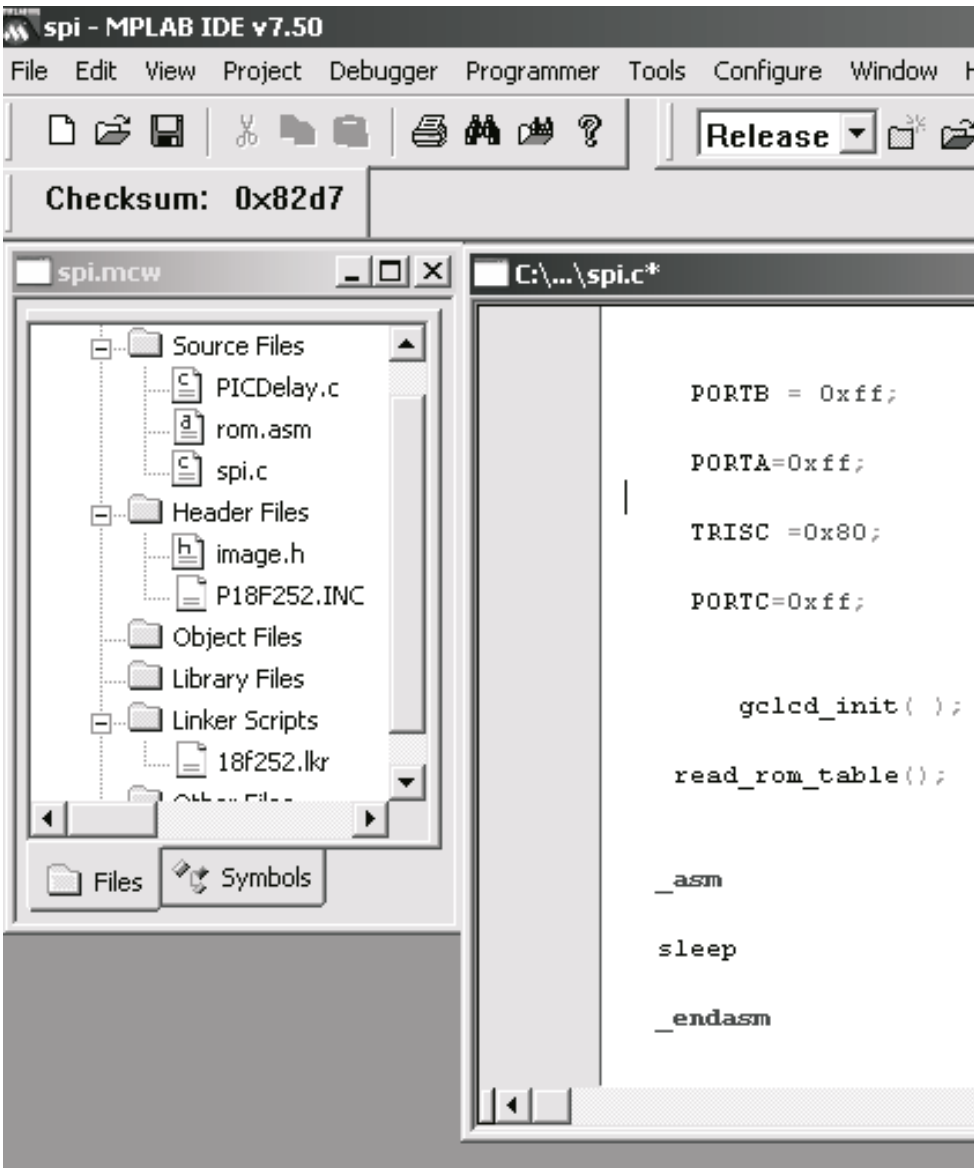
A TPS60400 DC voltage convertor provides the 6.2 volt potential difference across the two internal back-light LEDs integral to the display. The PIC18F252 has an internal memory size of 32 Kbytes, and so with a full-screen byte table of 17161 bytes (131 × 131), 1.75 screens of graphics can be displayed – the 3510i display with a smaller pixel resolution can display up to four screens.

5.7 Case Study Nokia 6100 Philips display 16 bit colour

The 6610 (6100 compatible) display available for 'JELU' is capable of displaying graphics with a pixel colour resolution of 16 bits. To accommodate this resolution, the size of the picture has to be restricted to approximately 131 × 120, since each pixel will generate a word (two bytes). The example shown below has a display size of 83H × 78H.



Program SPI.C demonstrates how to program the display for high-resolution colour. It can be seen that, in the MPLAB environment, the image file 'image.h' is now located as an external file, so this dispenses with the need to cut and paste the file into the main program.



The 'Jelu' display has a Philips driver chip and a completely different instruction set. Full details can be found in the relevant PDF file 'PCF8833'. Part of the instruction set is shown below:

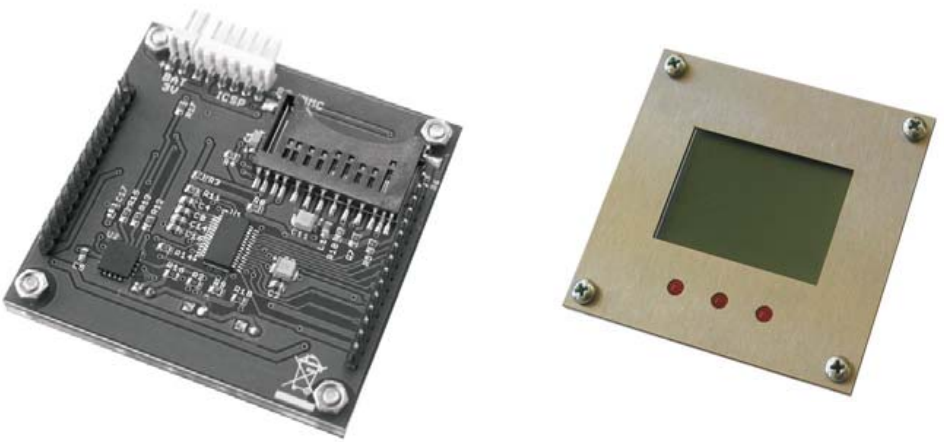
D/$\overline{C}$	7	6	5	4	3	2	1	0	DEFAULT	OTP	DESCRIPTION
0	0	0	0	0	0	0	0	0	00H	–	no operation (NOP)
0	0	0	0	0	0	0	0	1	01H	–	software reset (SWRESET)
0	0	0	0	0	0	0	1	0	02H	–	booster voltage off (BSTROFF)
0	0	0	0	0	0	0	1	1	03H	–	booster voltage on (BSTRON)
0	0	0	0	0	0	1	0	0	04H	–	read display identification (RDDIDIF)
0	0	0	0	0	1	0	0	1	09H	–	read display status (RDDST)
0	0	0	0	1	0	0	0	0	10H	–	Sleep_IN
0	0	0	0	1	0	0	0	1	11H	–	Sleep_OUT
0	0	0	0	1	0	0	1	0	12H	–	Partial mode on (PTLON)
0	0	0	0	1	0	0	1	1	13H	–	normal Display mode on (NORON)
0	0	0	1	0	0	0	0	0	20H	–	display inversion off (INVOFF)
0	0	0	1	0	0	0	0	1	21H	–	display inversion on (INVON)
0	0	0	1	0	0	0	1	0	22H	–	all pixel off (DALO)
0	0	0	1	0	0	0	1	1	23H	–	all pixel on (DAL)
0	0	0	1	0	0	1	0	1	25H	–	set contrast (SETCON)
1	X	$VCON_6$	$VCON_5$	$VCON_4$	$VCON_3$	$VCON_2$	$VCON_1$	$VCON_0$	00H	–	set contrast
0	0	0	1	0	1	0	0	0	28H	–	display off (DISPOFF)
0	0	0	1	0	1	0	0	1	29H	–	display on (DISPON)
0	0	0	1	0	1	0	1	0	2AH	–	column address set (CASET)
1	xs[7]	xs[6]	xs[5]	xs[4]	xs[3]	xs[2]	xs[1]	xs[0]	02H	–	X address start; 0 ≤ xs ≤ 83H
1	xe[7]	xe[6]	xe[5]	xe[4]	xe[3]	xe[2]	xe[1]	xe[0]	81H	–	X address end; xs ≤ xe ≤ 83H
0	0	0	1	0	1	0	1	1	2BH	–	page address set (PASET)
1	ys[7]	ys[6]	ys[5]	ys[4]	ys[3]	ys[2]	ys[1]	ys[0]	02H	–	Y address start; 0 ≤ ys ≤ 83H
1	ye[7]	ye[6]	ye[5]	ye[4]	ye[3]	ye[2]	ye[1]	ye[0]	81H	–	Y address end; ys ≤ ye ≤ 83H
0	0	0	1	0	1	1	0	0	2CH	–	memory write (RAMWR)
1	D7	D6	D5	D4	D3	D2	D1	D0	XXH	–	write data
0	0	0	1	0	1	1	0	1	2DH	–	colour set (RGBSET)
1	X	X	X	X	R3	R2	R1	R0	00H	–	red tone 000
1	6 bytes for 6 red tones									–	6 red tones

6 OEM colour Graphic Displays

Many LCD displays with embedded microcontrollers are available from OEM manufacturers. Some have USB interfaces whereby an image can be downloaded to the display with ease (no programming), some have "true colour" resolutions of up to 262K colours and others have onboard 2-gigabyte memory cards! In this chapter we will look at some of the OEM's and their LCD products.

6.1 OLIMEX

Olimex has a plethora of PIC development boards including LCD modules for the Nokia 3310 and 6100 LCDs.



3310 LCD with PIC16F866 from Olimex.

The PIC-LCD-3310 shown above is a development board with PIC16F886, Nokia 3310 BW 84 × 48 pixels LCD and three LEDs which could be used as touch buttons too(!). The board has an SD-MMC connector and all PIC ports are available on two extension connectors. Optionally, the board could mount a 3-axis accelerometer MMA7260 (as shown on the picture). It is perfect for motion data-logging/analysis.

Features

- MCU: PIC16F886 14KB Flash memory, 368 B RAM memory, 256 B EEPROM, x5 ADC, I2C, UART, nanowatt technology
- ICSP connector for PIC-ICD2 debugger, or PIC-PGx programmers
- LCD Nokia 3310 black/white 84 × 48 pixels
- Three LEDs which could be used both as status LEDs and as touch buttons
- SD/MMC card connector

- OPTIONAL MMA7260 accelerometer
- Extension connector for all PIC ports
- +3V battery connector
- PCB: FR-4, 1.5 mm (0,062"), solder mask, and silkscreen component print
- Dimensions: 76.2 × 64mm (3 × 2.5")

6.2 The MPS430-4619LCD (6100)

Features

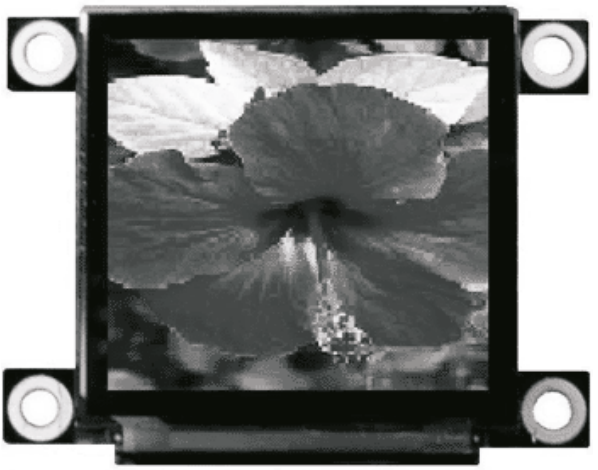
- MCU: MSP430FG4619 with 120K Bytes Program Flash, 256 Bytes data Flash, 4K Bytes RAM
- Nokia 6610 LCD 128 × 128 pixels 12 bit color LCD with backlight
- Joystick with 4 directions and push button function
- two buttons
- SD/MMC card connector
- MMA7620 3 axis accelerometer
- IrDA transciever
- UEXT connector which allow other Olimex modules to be connected like: MOD-MP3, MOD-NRF24L01, etc.
- JTAG connector
- 32 768 Hz oscillator crystal
- 8 Mhz crystall oscillator
- power supply voltage regulators and filtering capacitor
- extension headers for all uC pins
- PCB: FR-4, 1.5 mm (0,062"), soldermask, white silkscreen component print
- Dimensions: 81 × 62 mm (3.19 × 2.44")

6.3 4D SYSTEMS

This OEM probably provides the de-facto standard in colour LCD modules.They can be programmed via the PC USB port or by stand-alone embedded microcontrrolers.

Product Overview

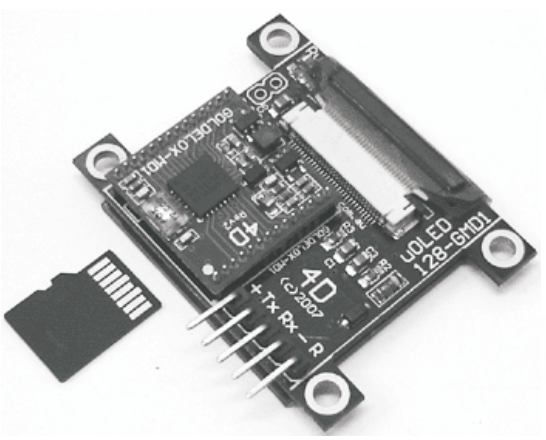
The micro-OLED (uOLED-128-GMD1) is a compact and cost effective all in one ‘SMART" OLED Display with an embedded graphics controller that will deliver ‘stand-alone’ functionality to your project. The ‘simple to use’ embedded commands not only control background colour but can produce text in a variety of sizes as well as draw shapes (which can include user-definable bitmapped characters such as logos) in 262,000 colours, whilst freeing up the host processor from the ‘processor hungry’ screen control functions.

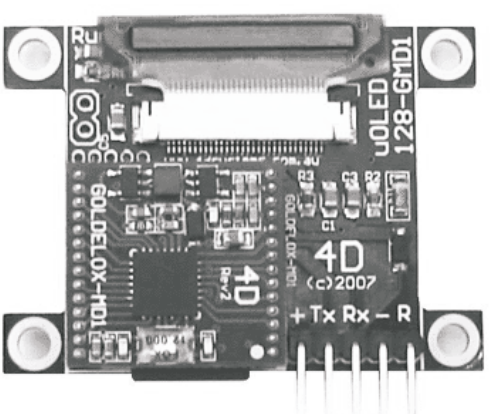


The 4D Systems uOLED-128-GMD1.

Main Features

- 128 × 128 pixel resolution, 65K or 262K true to life colours.
- 1.5" diagonal display. Module size 45.5 × 33.5 × 8.8mm.
- No backlighting with near 180° viewing angle.
- Easy 5 pin interface to any host device: VCC, TX, RX, GND, RESET
- Voltage supply from 3.6V to 6.0V, current @ 40mA nominal when using a 5.0V supply source.
- Serial RS-232 (0V to 3.3V) with auto-baud feature (300 to 256K baud). If interfacing to a system greater than 3.6V supply, a series resistor (1K) is required on the RX line.
- Powered by the fully integrated GOLDELOX-MD1 module (also available as separate OEM modules for volume users)
- Optional USB to Serial interface via the 4D micro-USB (uUSB-MB5) module.
- Onboard micro-SD (µSD) memory card adaptor for storing of icons, images, animations, etc. 64Mb to 1Gig µSD memory cards can be purchased separately.





- Three selectable font sizes (5 × 7, 8 × 8 and 8 × 12) for ASCII characters as well as user-defined bitmapped characters (64 @ 8 × 8)
- Built-in graphics commands such as: LINE, CIRCLE, RECTANGLE, TEXT, USER BITMAP, BACKGROUND COLOUR, PUT PIXEL, IMAGE, etc. just to name a few.

Typical Applications
- Full colour graphics OLED display for any microcontroller project.
- Interface to any PIC, AVR, Basic Stamp, ARM or any other microcontroller as well as a PC.
- Ideal to implement as an electronic panel meter, or gauge of any sort, for instrumentation or automotive applications.
- Can be used as a mini billboard for advertising.
- Ideal User Interface for any medical or industrial handheld equipment.
- Cost-effective, ready to go, intelligent display for the hobbyist, student or professional engineer.
- Download images, icons, animations and pages of text from any PC with the aid of freely available utility software tools.

6.4 The 4D-MICRO-LCD-320-PMD2 DISPLAY

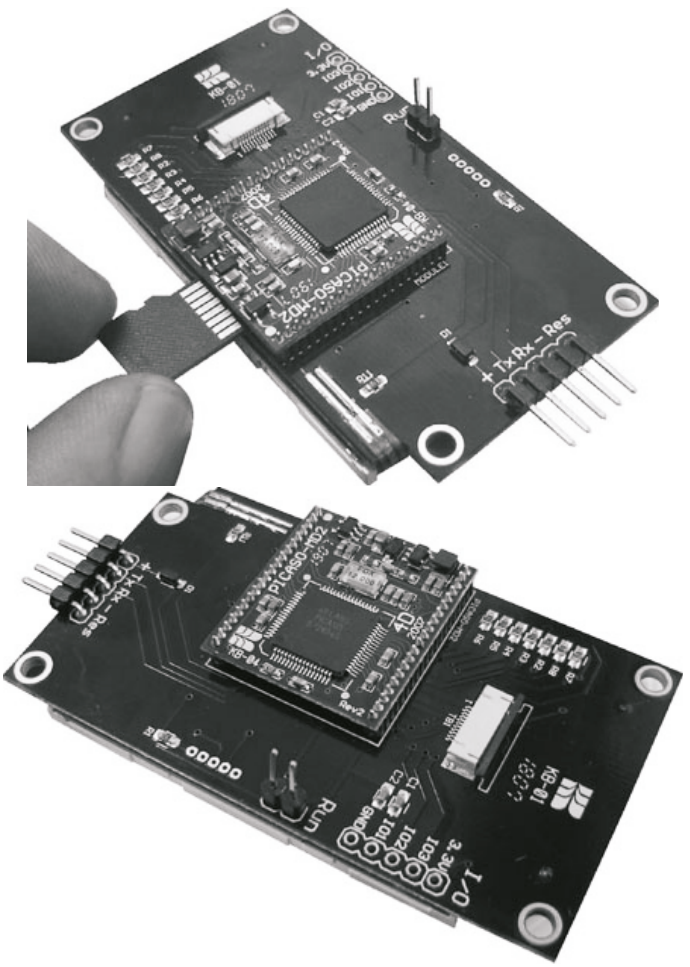
Available Options
uUSB-MB5 (USB to Serial)
uSD-64MB (memory card)
uSD-256MB (memory card)

Product Overview

The µLCD is a compact & cost-effective, all in one 'SMART' LCD module with an embedded graphics controller that will deliver 'stand-alone' functionality to your project. The 'simple to use' embedded commands not only control background colour, but can produce text in a variety of sizes, as well as draw shapes (which can include user definable bitmapped characters, such as logos) in 256 or 65,536 colours, whilst freeing up the host processor from the 'processor hungry' screen control functions. This means that a simple microcontroller with a standard serial or USB interface can drive the µLCD module with total ease.
The figures below show some of the graphics capability of the µLCD.

Main Features
The µLCD-320-PMD2 is aimed at being integrated into a variety of different applications via a wealth of features designed to facilitate any given functionality quickly and cost effectively and thus reduce 'time to market'. These features are as follows:
- 240 x 320 pixel resolution, 256 or 65K true to life colours, Enhanced LCD screen.
- 2.2" diagonal. Module Size: 74.5 x 40.5 x 10.3mm. Active Area: 33.75 x 45.25mm.

- LED backlighting with excellent viewing angle.
- Easy, 5 pin interface to any host device: VCC, TX, RX, GND, RESET
- Voltage supply from 3.6V to 6.0V, current at 60mA nominal when using a 5.0V supply source.
- Serial RS-232 (0V to 3.3V) with auto-baud feature (2400 to 500K baud). If interfacing to a system greater than 3.6V supply, a series resistor (1K) is required on the RX line.
- Powered by the fully integrated PICASO-MD2 module (also available as separate OEM modules for volume users).
- Optional USB to Serial interface via the 4D micro-USB (uUSB-MB5) module.
- Onboard micro-SD (µSD) memory card adaptor for storing of icons, images, animations, etc. 64Mb to 1Gig µSD memory cards can be purchased separately.
- Four selectable font sizes (5x7, 8x8 8x12 and 12x16) for ASCII characters as well as user-defined bitmapped characters.
- Built-in graphics commands such as: LINE, CIRCLE, RECTANGLE, TEXT, USER BITMAP, BACKGROUND COLOUR, PUT PIXEL, IMAGE, etc. just to name a few.



The 4D Systems 2.2" colour display.

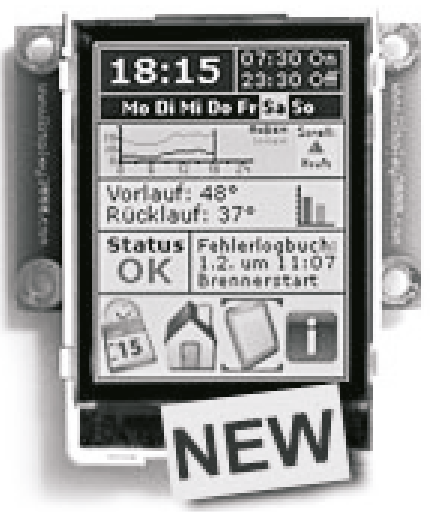
6.5 Display3000

Suppplies 2.1" and 1.5" colur LCD modules with embedded microcontroller.

D072 – 2.1" Color TFT mini module with microcontroller ATMega 128, RS232, I2C, 6 switches ..., complete solution for your development and/or solutions. Includes software and documentation.

D072

ED012-2,1" color TFT Display, with complete driving board – for ANY microcontroller. It is a full solution for your existing microcontroller (e.g. PIC, ARM, AVR ...) – and includes a complete module with programming documentation and complete software for C and Basic. Can be used in portrait and landscape (just switch by software)



ED012

6.6 ezLCD

If it's a touch LCD screen you're after, this is probably one of the best.

EzLCD-00.2

Features

- Integrated Touch Screen (resistive)
- Programmable Display Module
- Serial+USB+Parallel Interface
- I2C/AVR/BASIC Stamp/VB Compatible
- 64Kb Flash ROM (available for fonts & bitmaps)
- 1 Megabyte Flash ROM (use to be determined)
- Graphic/Text Commands
- Downloadable TTF Fonts
- Integrated Mini-USB Port (built-in to LCD module – no Eval Board required)
- Battery Operation (optional battery pack)
- 2.7" 240x160 Sony TFT Color LCD Display

** Note ** ezLCD-002 does not ship with cabling, power supply, or battery! ezLCD-002-EDK ships with all accessories for $229

Specifications

- Sony ACX705AKM-7 Specification Sheet
- RS232 and USB interfaces
- Multi-Font Text Layer
- LCD controller supports 256 colors (LCD panel alone supports 512 colors).
- Default serial data rate of 115200 bauds (user-changeable to any standard speeds between 2400 and 115200 bauds)
- Includes SPI and I2C serial interfaces
- Internal DC-DC converter accounts for a wide supply voltage range of 3-7 VDC (can be powered by USB port or external power supply)
- Driven by an extensive set of graphic, text and system instructions (firmware can be updated through embedded RS232)
- 1MB Flash (use to be determined, probably for custom fonts, bitmaps, icons, etc., uploadable via RS232 or USB)
- 96k bytes of Flash and 4k bytes of EEPROM available for custom fonts, icons, etc. (uploadable via RS232 or USB)

6.7 REACHtech

Founded in 1988, Reach Technology Inc. is an advanced design and manufacturing company. Reach makes electronic sub-assemblies for companies that build end-user equipment, commonly known as Original Equipment Manufacturers or OEMs. Custom electronics are required when the functionality required cannot be purchased

off-the-shelf, or when readily available items are a poor fit for the application due to cabling, feature set, integration, and so forth.. Reach can design and manufacture electronic assemblies such as circuit boards for less than the cost of internal development.

The effectiveness of the Reach business model has been proven in such areas as custom test equipment and flat panel interfaces and in specialty applications such as automated ticket machines and custom PC control boards.

7 Appendix

MICROCHIP

PIC12C5XX

8-Pin, 8-Bit CMOS Microcontrollers

Devices included in this Data Sheet:

- PIC12C508 • PIC12C508A • PIC12CE518
- PIC12C509 • PIC12C509A • PIC12CE519
- PIC12CR509A

Note: Throughout this data sheet PIC12C5XX refers to the PIC12C508, PIC12C509, PIC12C508A, PIC12C509A, PIC12CR509A, PIC12CE518 and PIC12CE519. PIC12CE5XX refers to PIC12CE518 and PIC12CE519.

High-Performance RISC CPU:

- Only 33 single word instructions to learn
- All instructions are single cycle (1 µs) except for program branches which are two-cycle
- Operating speed: DC - 4 MHz clock input
  DC - 1 µs instruction cycle

Device	Memory			
	EPROM Program	ROM Program	RAM Data	EEPROM Data
PIC12C508	512 x 12		25	
PIC12C508A	512 x 12		25	
PIC12C509	1024 x 12		41	
PIC12C509A	1024 x 12		41	
PIC12CE518	512 x 12		25	16
PIC12CE519	1024 x 12		41	16
PIC12CR509A		1024 x 12	41	

- 12-bit wide instructions
- 8-bit wide data path
- Seven special function hardware registers
- Two-level deep hardware stack
- Direct, indirect and relative addressing modes for data and instructions
- Internal 4 MHz RC oscillator with programmable calibration
- In-circuit serial programming

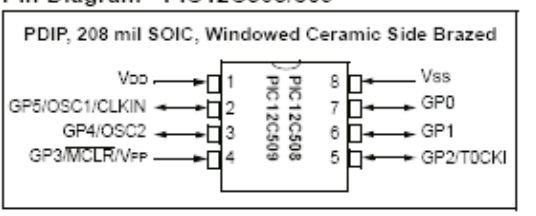
Peripheral Features:

- 8-bit real time clock/counter (TMR0) with 8-bit programmable prescaler
- Power-On Reset (POR)
- Device Reset Timer (DRT)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- 1,000,000 erase/write cycle EEPROM data memory
- EEPROM data retention > 40 years
- Power saving SLEEP mode
- Wake-up from SLEEP on pin change
- Internal weak pull-ups on I/O pins
- Internal pull-up on $\overline{\text{MCLR}}$ pin
- Selectable oscillator options:
  - INTRC: Internal 4 MHz RC oscillator
  - EXTRC: External low-cost RC oscillator
  - XT: Standard crystal/resonator
  - LP: Power saving, low frequency crystal

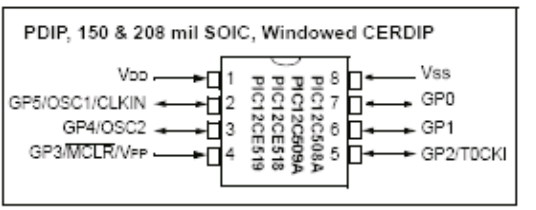
CMOS Technology:

- Low power, high speed CMOS EPROM/ROM technology
- Fully static design
- Wide operating voltage range
- Wide temperature range:
  - Commercial: 0°C to +70°C
  - Industrial: -40°C to +85°C
  - Extended: -40°C to +125°C
- Low power consumption
  - < 2 mA @ 5V, 4 MHz
  - 15 µA typical @ 3V, 32 KHz
  - < 1 µA typical standby current

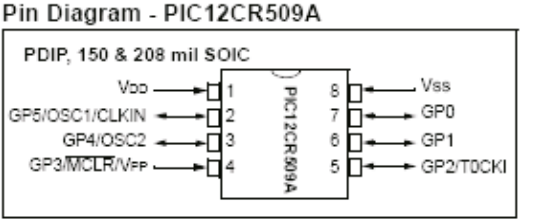
Pin Diagram - PIC12C508/509



Pin Diagram - PIC12C508A/509A, PIC12CE518/519



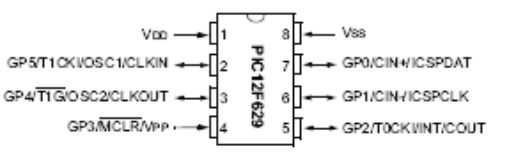
Pin Diagram - PIC12CR509A

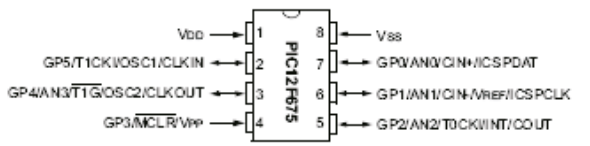


PIC12F629/675

Pin Diagrams

8-pin PDIP, SOIC, DFN-S





MICROCHIP

PIC12F629/675

8-Pin FLASH-Based 8-Bit CMOS Microcontroller

High Performance RISC CPU:

- Only 35 instructions to learn
  - All single cycle instructions except branches
- Operating speed:
  - DC - 20 MHz oscillator/clock input
  - DC - 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect, and Relative Addressing modes

Special Microcontroller Features:

- Internal and external oscillator options
  - Precision Internal 4 MHz oscillator factory calibrated to ±1%
  - External Oscillator support for crystals and resonators
  - 5 µs wake-up from SLEEP, 3.0V, typical
- Power saving SLEEP mode
- Wide operating voltage range - 2.0V to 5.5V
- Industrial and Extended temperature range
- Low power Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) with independent oscillator for reliable operation
- Multiplexed $\overline{\text{MCLR}}$/Input-pin
- Interrupt-on-pin change
- Individual programmable weak pull-ups
- Programmable code protection
- High Endurance FLASH/EEPROM Cell
  - 100,000 write FLASH endurance
  - 1,000,000 write EEPROM endurance
  - FLASH/Data EEPROM Retention: > 40 years

Low Power Features:

- Standby Current:
  - 1 nA @ 2.0V, typical
- Operating Current:
  - 8.5 µA @ 32 kHz, 2.0V, typical
  - 100 µA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current
  - 300 nA @ 2.0V, typical
- Timer1 oscillator current:
  - 4 µA @ 32 kHz, 2.0V, typical

Peripheral Features:

- 6 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - One analog comparator
  - Programmable on-chip comparator voltage reference (CV_{REF}) module
  - Programmable input multiplexing from device inputs
  - Comparator output is externally accessible
- Analog-to-Digital Converter module (PIC12F675):
  - 10-bit resolution
  - Programmable 4-channel input
  - Voltage reference input
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected
- In-Circuit Serial Programming™ (ICSP™) via two pins

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)				
PIC12F629	1024	64	128	6	–	1	1/1
PIC12F675	1024	64	128	6	4	1	1/1

* 8-bit, 8-pin devices protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

MICROCHIP

PIC16F8X

18-pin Flash/EEPROM 8-Bit Microcontrollers

Devices Included in this Data Sheet:

- PIC16F83
- PIC16F84
- PIC16CR83
- PIC16CR84
- Extended voltage range devices available (PIC16LF8X, PIC16LCR8X)

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle except for program branches which are two-cycle
- Operating speed: DC - 10 MHz clock input
  DC - 400 ns instruction cycle

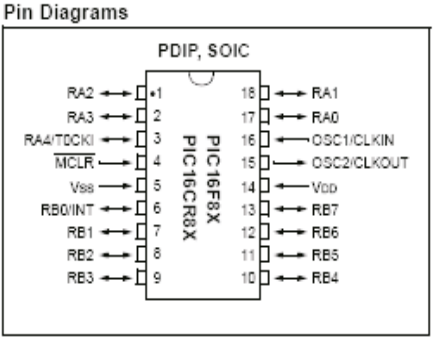
Device	Program Memory (words)	Data RAM (bytes)	Data EEPROM (bytes)	Max. Freq (MHz)
PIC16F83	512 Flash	36	64	10
PIC16F84	1 K Flash	68	64	10
PIC16CR83	512 ROM	36	64	10
PIC16CR84	1 K ROM	68	64	10

- 14-bit wide instructions
- 8-bit wide data path
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt on change
  - Data EEPROM write complete
- 1000 erase/write cycles Flash program memory
- 10,000,000 erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years

Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 20 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Pin Diagrams



Special Microcontroller Features:

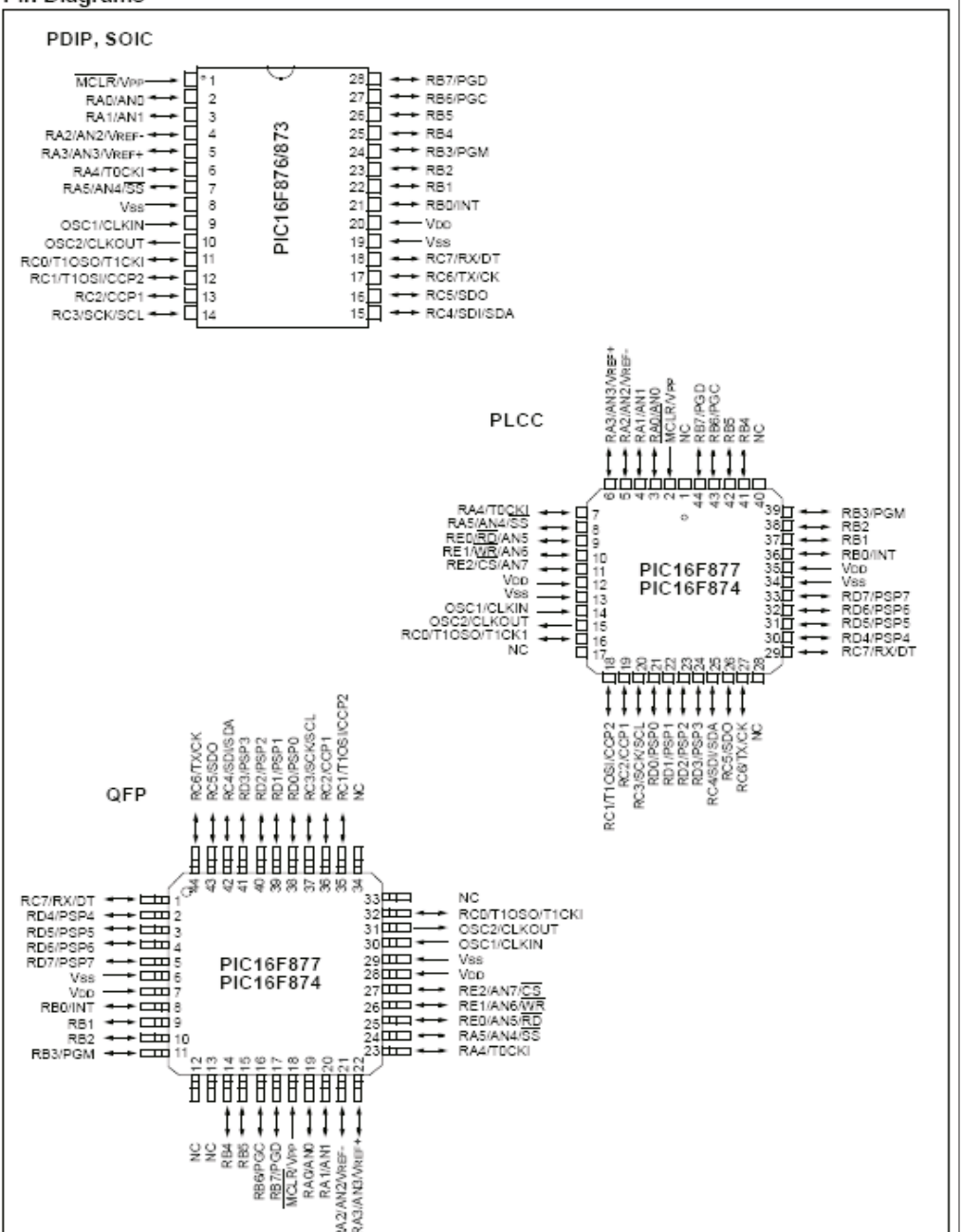
- In-Circuit Serial Programming (ICSP™) - via two pins (ROM devices support only Data EEPROM programming)
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

CMOS Flash/EEPROM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 6.0V
  - Industrial: 2.0V to 6.0V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 1 µA typical standby current @ 2V

PIC16F87X

Pin Diagrams



MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

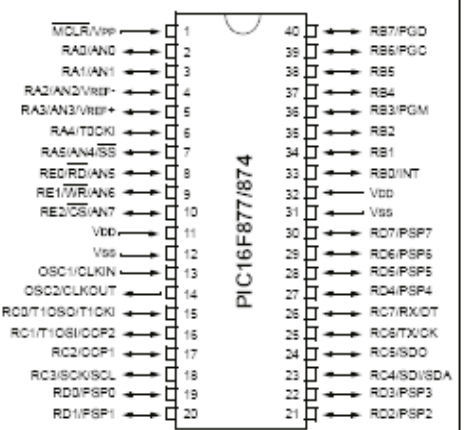
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP

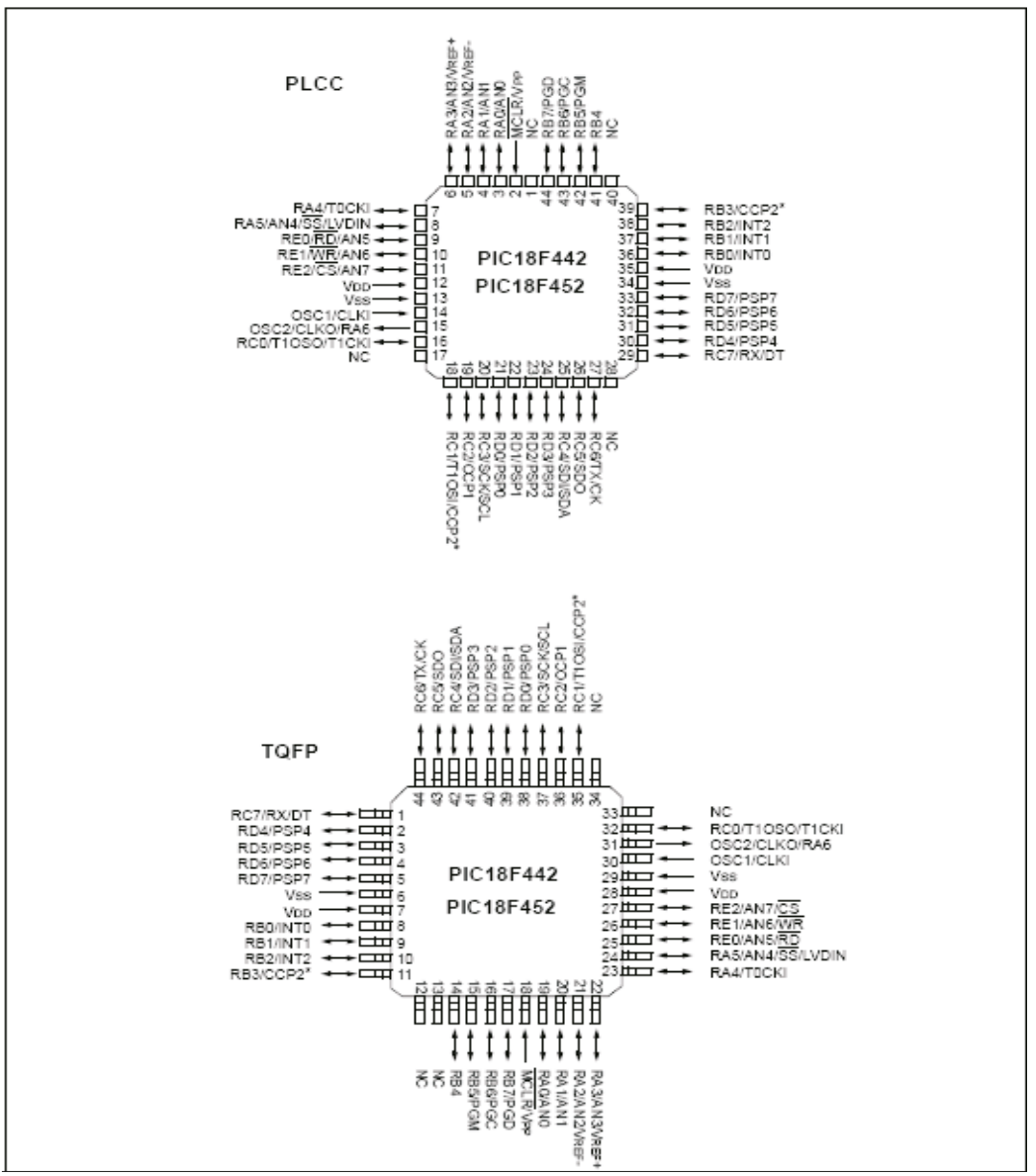


Peripheral Features:

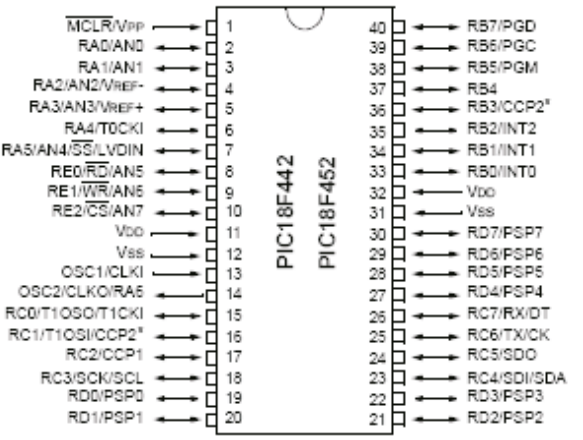
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

PIC18FXX2

Pin Diagrams

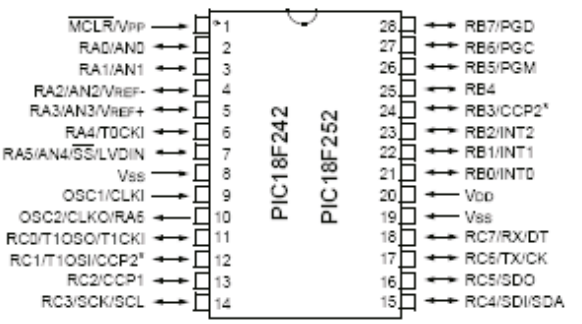


DIP



Note: Pin compatible with 40-pin PIC16C7X devices.

DIP, SOIC



* RB3 is the alternate pin for the CCP2 pin multiplexing.

MICROCHIP

PIC18FXX2

28/40-pin High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

High Performance RISC CPU:

- C compiler optimized architecture/instruction set
  - Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 32 Kbytes
- Linear data memory addressing to 1.5 Kbytes

Device	On-Chip Program Memory		On-Chip RAM (bytes)	Data EEPROM (bytes)
	FLASH (bytes)	# Single Word Instructions		
PIC18F242	16K	8192	768	256
PIC18F252	32K	16384	1536	256
PIC18F442	16K	8192	768	256
PIC18F452	32K	16384	1536	256

- Up to 10 MIPs operation:
  - DC - 40 MHz osc./clock input
  - 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules. CCP pins that can be configured as:
  - Capture input: capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1- to 10-bit, max. PWM freq. @: 8-bit resolution = 156 kHz, 10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I^2C™ Master and Slave mode

Peripheral Features (Continued):

- Addressable USART module:
  - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

Analog Features:

- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Conversion available during SLEEP
  - Linearity ≤ 1 LSb
- Programmable Low Voltage Detection (PLVD)
  - Supports interrupt on-Low Voltage Detection
- Programmable Brown-out Reset (BOR)

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- FLASH/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption:
  - < 1.6 mA typical @ 5V, 4 MHz
  - 25 µA typical @ 3V, 32 kHz
  - < 0.2 µA typical standby current

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

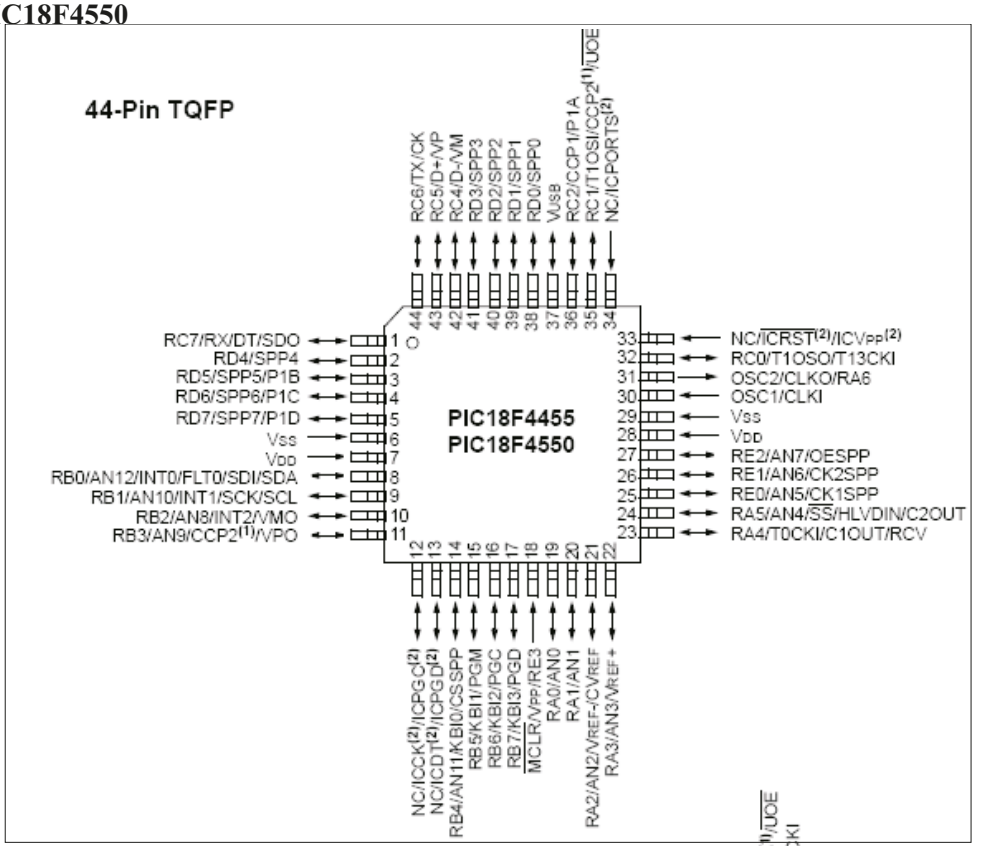
- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

PIC18F4550



7.1 References

- http://www.farnell.com/
  World wide electronic component supplier including PIC chips
- http://www.cadsoft.de/
  EAGLE CAD PCB and schematic design
- http://www.microchip.com/
  PIC datasheets and C18 PIC C compiler
- http://www.burntecshop.co.uk/
  Presto ISP PIC programmer
- http://www.quasarelectronics.com/
  3149 PIC programmer with ZIF socket
- http://www.melabs.com/
  U2 PIC programmer

- http://www.velleman.be/
  Sound to light kit
- http://www.bknd.com/cc5x/ PIC16Fxx C compiler
  PIC16XX C compiler
- http://www.ramtex.dk/iconedit/iconedit.htm
  Icon editor
- http://www.p-m-services.co.uk/index.htm
  PCB manufacturer
- http://www.jelu.se/shop/index.php
  Nokia 6100 16 bit colour display
- http://www.rfsolutions.co.uk/acatalog/index.html
  PIC Programmers
- http://www.burntecshop.co.uk
  Presto PIC Programmer
- http://www.4dsystems.com.au/
  OEM colour LCD displays
- http://www.quasarelectronics.co.uk/
  PIC Programmers
- http://www.densitron.com/displays/Displays.aspx?nCategoryID=3
  Densitron LCD displays
- http://www.formymobile.co.uk/products.php?cat=36
  Nokia 3310, 3510i and 6100 8 bit colour displays
- http://www.bluebird-electronics.co.uk/index.html
  B/W LCD displays
- http://www.sparkfun.com/commerce/categories.php
  The Holy Grail of LCD displays and more
- http://www.pcbtrain.co.uk/onestepquote.php
  PCB manufacturer
- http://www.active-robots.com/products/sensors/sensors-devantech.shtml
  Electronic compass module
- http://www.elektor-electronics.co.uk/
  International Electronics Magazine
- http://www.shop-en.display3000.com/
  OEM colour LCD displays
- http://www.ezlcd.com/
  OEM colour LCD displays
- http://www.reachtech.com/display/slcd-kits.html
  OEM colour LCD displays
- http://www.dontronics-shop.com/4DSystems-mid-1.html
  OEM colour LCD displays
- www.olimex.com/
  PIC Development Boards and LCD displays

Index

Universal Display Book

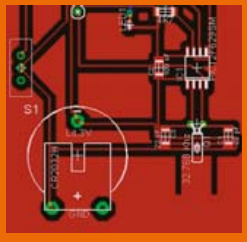
for PIC Microcontrollers

Richard Grodzik

The newcomer to Microchip's PIC microcontrollers invariably gets an LED to flash as their first attempt to master this technology. You can use just a simple LED indicator in order to show that your initial attempt is working, which will give you confidence to move forward.
This is how the book begins — simple programs to flash LEDs, and eventually by stages to use other display indicators such as the 7-segment display, alphanumeric liquid crystal displays and eventually a colour graphic LCD.

As the reader progresses through the book, bigger and upgraded PIC chips are introduced, with full circuit diagrams and source code, both in assembler and C.
In addition, a small tutorial is included using the MPLAB programming environment, together with the EAGLE schematic and PCB design package to enable readers to create their own designs using the book's many case studies as working examples to work from.





ISBN 978-0-905705-73-6
9 780905 705736
Elektor International Media B.V.
www.elektor.com