EPE ICEBREAKER by MARK STUART

A real-time PIC In-Circuit Emulator (ICE), programmer, debugger, and development system

This project combines Microchip's MPLAB development software with the advanced self-debugging features of the latest PIC chips. The result is a user-friendly advanced development system for a very low cost.

DEVELOPMENT SYSTEMS

Developing projects with microcontrollers is extremely interesting, especially the ability to alter the way hardware responds just by making simple changes to program code.

The problem is that each change of code has to be written, compiled (converted into suitable form for loading) and programmed into a chip before it can be tested. Several low cost systems – such as the *EPE PICtutor* are available (see our web page at **www.epemag.com** for more details) and are adequate for the development of simple programs, but for larger changes and programs it can be tedious, and errors are almost as easy to introduce as to eliminate.

Better methods of testing programs rely on more advanced software that can "run" in a virtual chip on a PC screen and advanced hardware which communicates with the program in the PC, reads input pins, and switches output pins to match the levels of the virtual chip. Such a system is called an In-Circuit Emulator or "ICE" and professional systems are available for practically every type of microcontroller.

The problem with this is cost. A professional ICE for the PIC series of chips costs a reasonable 2000 UK Pounds or so – not a lot if you are a professional programmer being paid twice that each month – but more certainly enough to make your eyes water if you are an amateur!

Just lately a new type of development system has appeared called In-Circuit Debugging (ICD). This requires a chip with special built in hardware (known as a "Background Debugger") and software which can communicate its status via a serial link.

The chip is fitted to its working printed circuit board (PCB) and all external hardware is connected and active. Code is then programmed, run, and debugged under PC control, until it is running correctly. For Microchip PIC users, the good news is that the PIC16F877 and its close relatives the '876, '874, and '873 have built in ICD facilities and can be used to develop

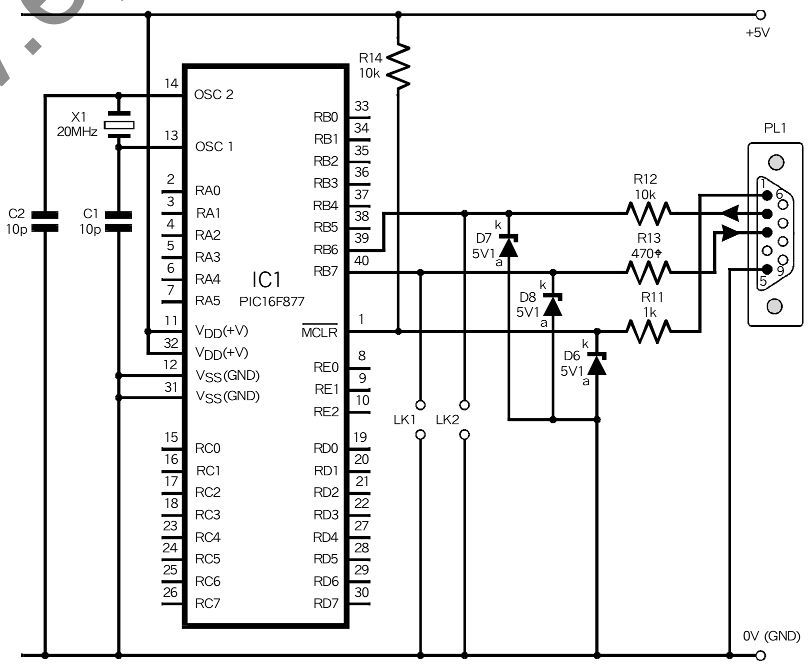


Fig.1. Minimum circuit for using the PIC16F877 as a "background" debugger.

code which can be run in these and smaller chips in the range – such as the most popular PIC16F84.

PIC IN-CIRCUIT DEBUGGING

This article is intended as an easy introduction to ICD with very simple demonstration programs, users can then progress to using the more complicated features of the chips. It is *not* intended as a programming tutorial, but the operation of some programs is described in the course of demonstrating the ICD hardware.

Simple programs can be loaded, run and debugged without knowing much about the entire ICD system which is extremely complicated and occupies many pages of the PIC data sheets. The Microchip web site (**www.microchip.com**) provides an enormous amount of information for those wishing to know more.

MINIMUM ICD SET UP

The minimum hardware required for ICD, using the PIC16F877, is shown in Fig.1. Communication to a computer serial port is achieved via the Port RB6 and RB7 pins of the chip, which cannot be used for other functions. As there are plenty of other port pins available this is not a significant limitation.

Port RB6 receives data from the computer, and RB7 transmits data to the computer. Both of these pins operate at simple 0V to 5V logic levels. Some computer serial port output pins swing 10V positive and negative, and so limiting resistors and 5·1V Zener diodes are used for protection.

The serial data sent back to the computer should also be capable of swinging 10V, but it has been found that practically all computers read serial data correctly when 0V to 5V swings are used. A third connection links the serial port RTS output to the VPP or MCLR pin of the chip. This allows control of the programming voltage (programming at 5V, as opposed to the 12V normally required) and resetting of the chip by the computer.

After a Reset, the chip checks if pins RB6 and RB7 are shorted to 0V. If they are, it ignores the ICD functions and just runs the code directly starting from location 0. Links fitted in the positions marked LK enforce this option.

As well as the hardware connections, the computer needs to run a program to communicate with the chip, and the chip must have a program to communicate with the computer. The program in the chip is loaded and copy-protected into the upper half of the chip's 8K program memory.

It may seem wasteful to use half of the chip's memory for protected code, but in practice, the 4K remaining is a vast amount of space for the PIC program and it is very doubtful that it will ever be filled. Once working code has been developed and debugged it can be loaded and will run alone in any chip in the 16x range – the protected communication code is required only in the chip used for debugging.

Connections for suitable serial leads are shown in Table 1. These are standard connections, but can be made up with 4-way cable (flat telephone cable is ideal) if required. Take care when making leads to get the pin numbering correct – it is very confusing, the only safe way is to read the molded numbers on the connectors.

The port connections for the PIC 16F877/874 and the alternative connections for the 28-pin PIC16F876/873 versions of the chip are shown in Fig.2.

HARDWARE

Whilst the minimum system could be used, it is unlikely that any PIC application would operate without external hardware. Most systems have power supplies, input switches, output devices and so on. It is irritating and time consuming to have to set up these simple hardware requirements when the object is to get a

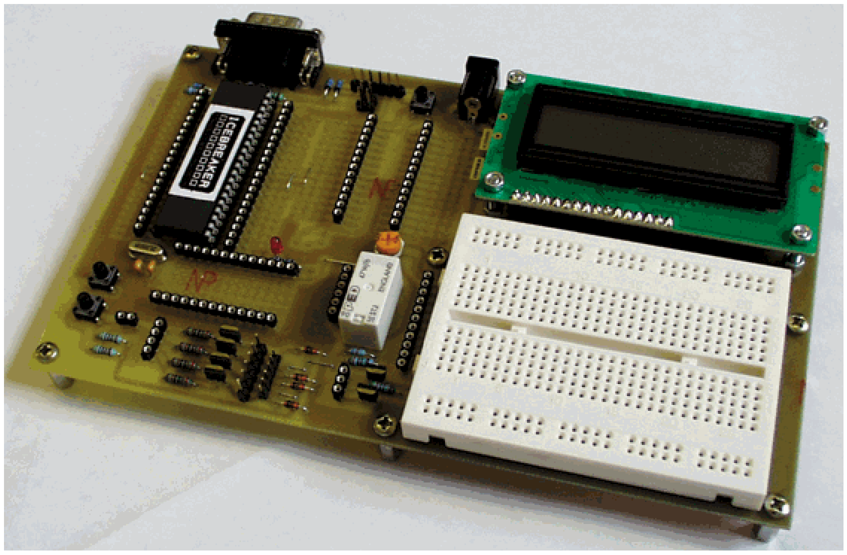


Table 1: Serial Lead Connections.

9-way to 9-way ICE	Computer
1	7
2	3 TxD
3	2 RxD
5	5 GND
9-way to 25-way ICE	**Computer**
1	4
2	2 TxD
3	3 RxD
5	7

program written and tested. *EPE ICEbreaker* was designed to include a number of input and output devices along with a solderless connection system so that many applications could be tested from a notebook PC without the use of a soldering iron.

The full *ICEbreaker* circuit diagram is shown in Fig.3, whilst Fig.4 gives the PCB details.

Resistors R11 to R13, and R14, Zener diodes D6 to D8 and links LK1/2 are the same as the minimum system. PL2 allows the power and computer connections to be extended so that the PIC could be fitted into another board with other hardware and still debugged via the same computer lead.

Voltage regulator IC2 allows a range of power adapters to be used connected to 2·1mm power socket SK5. Links 3 and 4 allow positive inner or outer connections to be set. If accidental power reversal is possible, the positive link connection can be made using a 1A diode (e.g. 1N4001) instead of a piece of wire. Power is indicated by light-emitting diode (LED) D5 via resistor R8.

Switch S1 provides an alternative hardware reset which can be useful for stopping programs quickly and for restarting from location 0.

For ICD operation the PIC needs accurate timing. A 20MHz crystal X1 together with capacitors C1 and C2 provide the standard oscillator components. Alternative positions (X2) and (C3, C4) are to be used with 28-pin chips.

Resistors R15 and R16 allow *RC* oscillator options to be used if required for testing or running fully debugged code, but as *RC* oscillator stability is poor, this option is not recommended for ICD use. Other crystal frequencies can be used and the computer serial port speed altered accordingly. 20MHz gives the fastest communication (38,400 BAUD) and is best if there are no other special frequency requirements.

Stepping motor driving is a very popular PIC application. Transistors TR1 to TR4 and associated resistors R2 to R5 and protection diodes D1 to D4 provide four open collector drivers for four-phase unipolar motors. Connectors PL3 and PL4 allow for 2·54mm and 2mm pitch motor connectors. Input to the drivers is via SK4. The transistors can also be used individually as simple open-collector *npn* switches for driving relays, lamps and similar loads up to 24V and 400mA.

Two other output transistors are fitted. TR5 is a simple open collector *npn* device and TR6 drives a double-pole changeover relay RLA. These two devices are useful for bidirectional control of a DC motor. RLA can be wired as a reversing switch and the motor can be

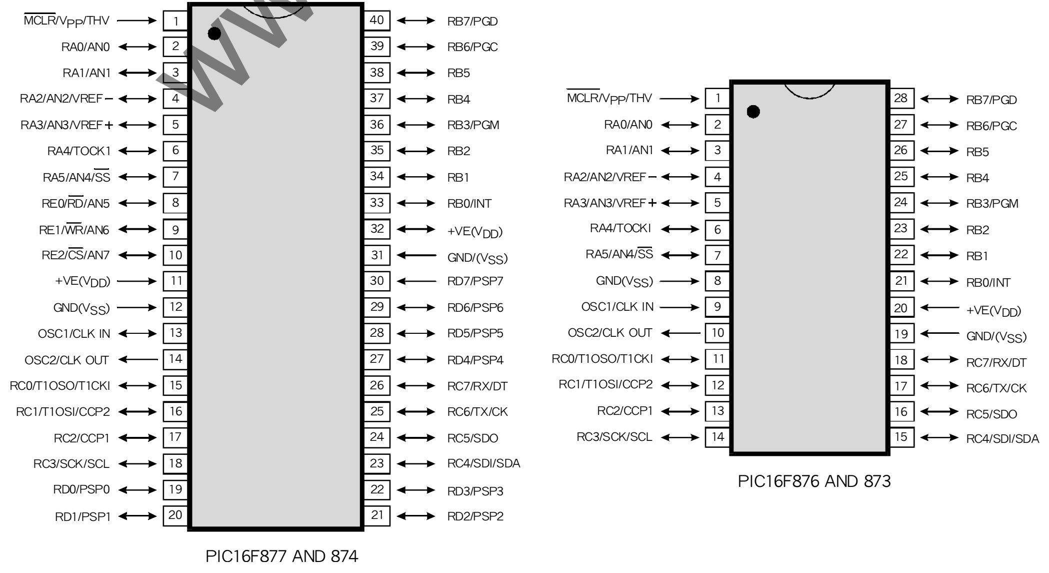


Fig.2. Port connections for the PIC16F877/874 and the alternative 28-pin PIC16F876/873.

turned on and off by TR5.

Many applications require display of information, and an intelligent LCD module is an ideal display device. X3 has standard 4- or 8-bit drive capability, and requires a minimum of six output lines for driving. All pins are available at connector SK1. Preset VR1 allows the contrast of the LCD to be altered to suit the lighting conditions and viewing angle.

Just two input devices are fitted. S2 and S3 are simple single-pole push-to-make switches with pull-up resistors R6 and R7.

Whilst the circuit diagram seems simple, the PCB layout shows that there are far more connection points, and that a prototyping area with a solderless breadboard is provided. Each side of the main integrated circuit (IC) sockets there are spaces for rows of turned-pin sockets. The inner two rows connect to the adjacent IC pins, whilst the outer two rows are power and ground connections.

Turned-pin socket strips can be fitted in all rows, but it is more practical to have a single row of sockets each side of the chip and leave the other spaces blank so that pull up or pull down resistors can be soldered in position if required. An additional "patch" area is provided below IC1 and is ideal for adding "permanent" hardware such as LEDs or presets.

ICEBREAKER SOFTWARE

ICEbreaker must be run from a PC with at least *Windows 95*. This helps keep the software simple, and is not a serious restriction as PCs that can run Win95 are available at very low prices. A standard Pentium 133 without special sound, graphics or multimedia is more than adequate provided it has a spare serial port (COM 1 – 4).

The software is designed to be run in conjunction with Microchip's MPLAB software. This is available from many sources – the Microchip web site is the ideal one as it allows the very latest version to be loaded, alternatively the Microchip CD-ROM is widely available and good for those without internet access.

MPLAB is used in "editor only" mode to allow assembly language source code to be written and then assembled to produce the necessary **.HEX** code for programming into the chip. MPLAB also produces a **.COD** file which is used by the *ICEbreaker* software to keep track of the program execution when debugging, single stepping and running the program. Like many other PC programs, MPLAB has a lot of features that are not regularly used, however the advanced features don't get in the way when using it at the simple level required by *ICEbreaker*.

The *ICEbreaker* software is a simple stand-alone application that can be run directly from a floppy disk if necessary. This article assumes that the contents of the *ICEbreaker* disk are copied into a new folder (directory) on the C-drive, which has been labeled "**icebreak**". The only files required are **icebreak.exe** and **icebreak.ini**, but it is also convenient to store program files in the same directory.

ICEbreaker and MPLAB should be run together, and the "Alt" and "Tab" keys or the taskbar buttons used to switch from one to the other.

CONSTRUCTION

The *EPE* ICEbreaker printed circuit board component layout and (approximately) full size copper foil master are shown in Fig.4. This board is available from the *EPE Online Store* (code 7000257) from **www.epemag.com**

Assembly of the board is straightforward. Begin by fitting seven 12mm pillars with short M3 screws before adding any components. Refer to the component layout drawing and then fit plain uninsulated wire links in all of the positions shown. Fit two-way pin headers in the position for LK1 and LK2 so that two shorting links can be connected if required.

Links LK3 and LK4 provide the facility to set the input power socket for positive or negative inner connection. For positive inner fit the links in position B, for negative fit them in position A. As mentioned previously it is possible to add a diode in place of one of the links to protect against polarity reversal. To do this, fit the cathode of the diode to the point marked with a + sign, and the anode of the diode to the appropriate A or B position.

Fit the diodes and resistors next, taking care to identify the type and polarity. Usually the cathodes are marked with a black or dark blue band, which should be positioned to match the line on the component layout diagram. The transistors TR1 to TR6 are all the same type and are fitted with their curved sides as shown in the diagram. They should be fitted close to the board surface so that they cannot get bent and

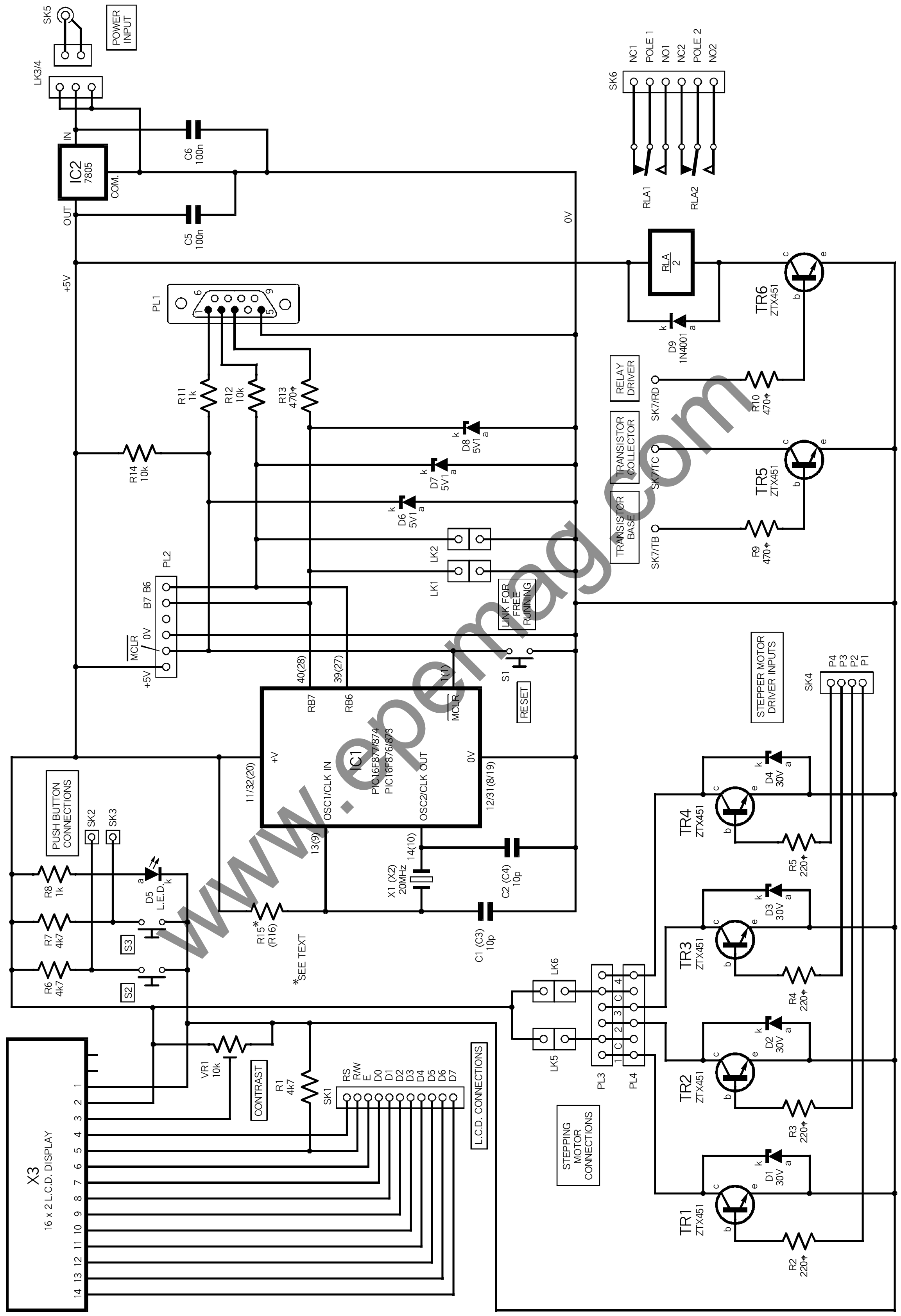


Fig.3. Complete circuit diagram for the EPE ICEbreaker.

moved around when the board is handled.

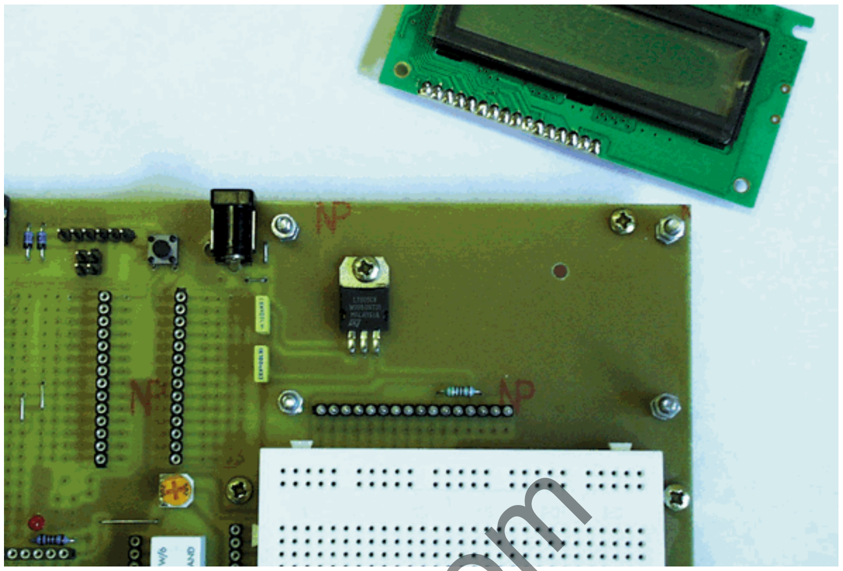
Use turned-pin socket strips for the 40 and 28-way IC positions, and position a second row of sockets alongside. Note that there is also the option of a narrow-bodied version of the 28-pin device, and holes have been drilled to allow for this type to be used. If required, socket strips can be fitted for both types without causing any difficulty. Also fit turned-pin socket strips for SK1, SK2, SK3, SK4, SK6, and the three connections TB, TC, and RD. Also fit two 13-way strips to the upper and lower rows of the patch area – these are the positive and negative "rails" and make very convenient connection points for taking power to the breadboard.

Fit pin headers for PL2, PL3 and PL4 – the holes for these are made tight to give extra support and so the pins may need pressing home against a hard surface. Fit push-switches S1, S2 and S3, preset VR1, relay RLA and the voltage regulator IC2; an M3 screw and nut should be used to secure the tab. A heatsink is not required for most applications, but there is space to fit a low profile type, or even a small piece of aluminum if higher current loads are to be used.

The 20MHz crystal and its associated capacitors C1 and C2 should be fitted if the (usual) 40-pin device is being used for IC1. If a 28-pin version is used then fit these components to the alternative locations X2, C3 and C4. If both types of device may be used, it is possible to fit two crystals and two pairs of capacitors.

DISPLAY MODULE

The LCD module fits above the board on 16mm long 6BA or M2.5 screws. Fit the four screws from the track side of the board and secure them with nuts. Fit four more nuts and position them equally so that the LCD lies level and approximately 10mm from the board. The connections to the LCD are made to allow it to be unplugged for access to IC2 and for use in other applications.


Display module removed from the PCB to reveal the regulator IC mounted underneath.

Fit a 16-way pin-to-pin connector to the board, with the slightly thinner pins upwards. Fit (but do not solder) a 16-way wirewrap turned-pin socket strip to the LCD so that the sockets face downwards and plug onto the pin-to-pin connector. Make sure the LCD is level, solder the wire wrap pins to the LCD and cut off the excess. The LCD can now be secured by fitting another four nuts.

The serial port connector PL2 and power connector SK5 fit directly onto the board. Make sure that they are pressed fully home before soldering.

The solderless breadboard is secured to the board simply by its self-adhesive backing. Make sure that it is accurately positioned (and the right way up) before pressing it firmly into place.

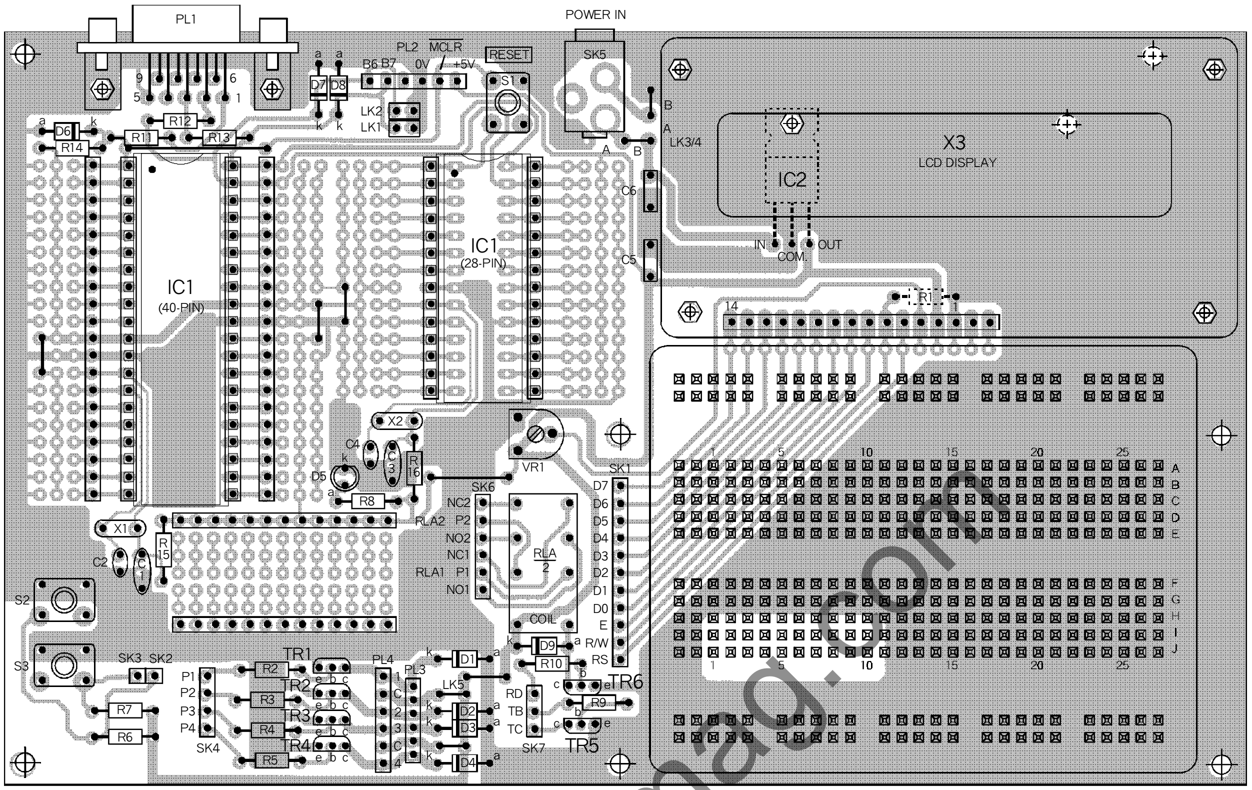
TESTING

Once the hardware has been assembled and before fitting IC1, check for dry joints, solder bridges and component polarities.

Once everything looks correct, connect the power supply. Check that D5 lights and, if a meter is available, check the 5V regulated supply. The LCD contrast control VR1 should alter the density of a single top row of block characters as the LCD initializes itself for one-line mode.

Switch off, insert IC1 and connect a suitable lead between SK1 and the serial port of the PC, which has MPLAB and the *ICEbreaker* software (see the **Shoptalk** page) installed.

SOFTWARE INITIALISATION



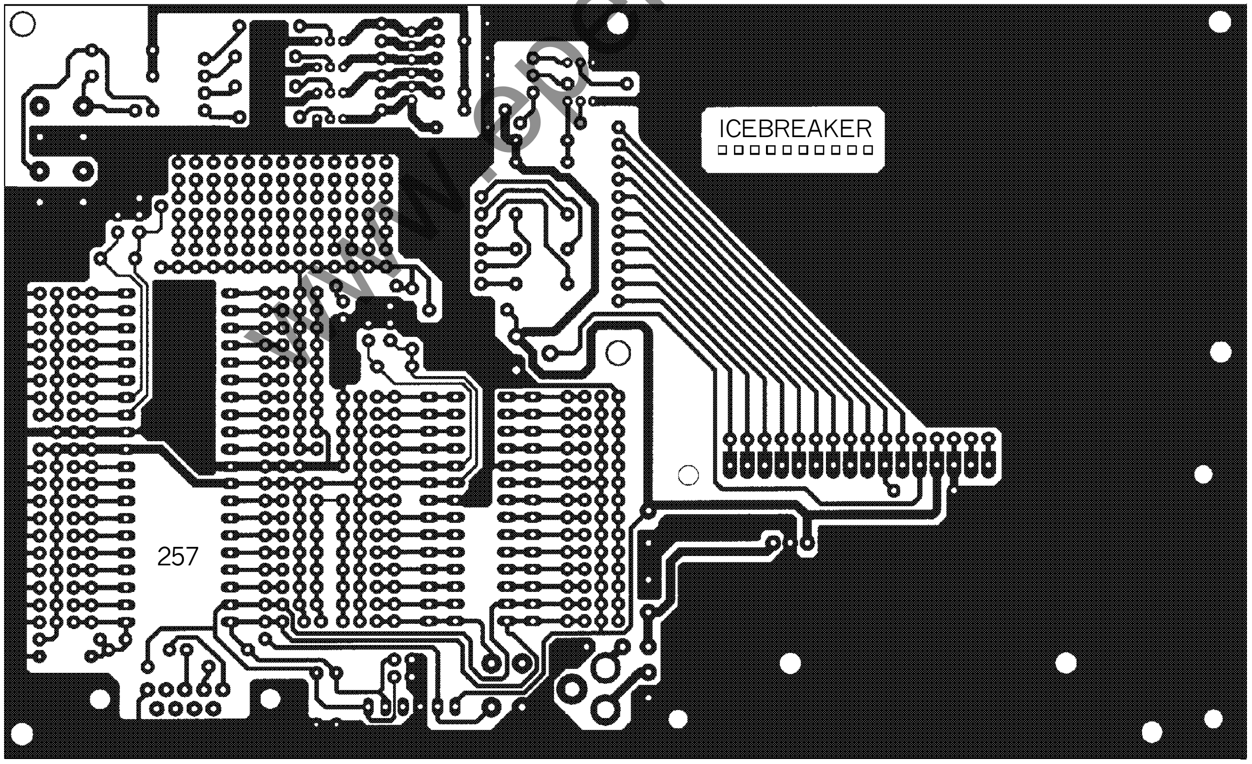


Fig.4. Printed circuit board component layout and (approximately) full-size copper foil master pattern.

COMPONENTS

Resistors
R1, R6, R7 4k7 (3 off)
R2 to R5 220 ohms (4 off)
R8, R11 1k (2 off)
R8, R10, R13 470 ohms (3 off)
R12, R14 10k (2 off)
R15, R16 See text
All 0.25W 5% carbon film

Potentiometer
VR1 10k carbon preset

Capacitors
C1 to C4 10p ceramic, 2.5mm pitch (4 off), see text
C5, C6 100n multilayer polyester (2 off)

Semiconductors
D1 to D4 30V 400mW Zener diodes (4 off)
D5 3mm low-current red LED
D6 to D8 5.1V 400mW Zener diodes (3 off)
D9 1N4001 diode
TR1 to TR6 ZTX451 *npn* transistors (6 off)
IC1 PIC16F877P20 microcontroller, pre-programmed
IC2 7805 voltage regulator
X1 (X2) 20MHz low-profile crystal (see text)
X3 16x2 alphanumeric LCD module

Miscellaneous
Socket strips to make up the following: 11-way (SK1); 1-way (SK2, SK3 -- 2 off); 4-way (SK4); 6-way (SK6)
SK5 2.1mm PCB power connector
S1 to S3 s.p.s.t. push-to-make switches (3 off)
RLA d.p.c.o. 5V coil relay (BT47)

PCB available from the *EPE Online Store* code 7000257 (**www.epemag.com**); breadboard; 9-way 90º male D-type connector (PL1); 6-way 0.1in. pin header (PL2, PL4 -- 2 off); 6-way pin strip, 2mm pitch (PL3); 2-way 0.1in. pin header with DP link plug (2 off -- LK1, LK2); socket strips, 20-way (4 off), 14-way (2 off), 13-way (2 off); 16-way pin-to-pin strip for LCD; 16-way long-pin socket strip for LCD.

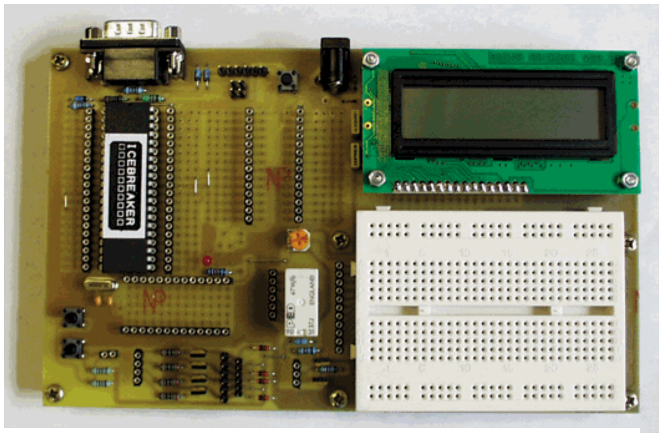
Hardware: 12mm M3 HEX pillars (7 off); M3 screw x 6mm (7 off); screws CSK (4 off) and 12 nuts (6BA or M2.5) for LCD mounting.

See also the SHOP TALK Page!

Approx. Cost Guidance Only **$60**

Run MPLAB, select the “project” tab and then “new project”. In the directory box select **c:\ icebreak** and set up a project named **ib.prj** and edit the project so that it contains the simple test program **ib1.asm** which is included on the *ICEbreaker* disk. Close the “Project Edit” window then select “File” and **ib1.asm**. This will open the **ib1.asm** file on the MPLAB screen.

Next select “Project” and “Make project”. This will then run the MPASM program and produce


Layout of components on the completed EPE ICEbreaker PCB.

files called **ib1.lst**, **ib1.hex**, **ib1.cod**, and **ib1.err** in the **icebreak** directory. The **ib1.err** file will contain a few warning messages, which can be ignored.

Leave MPLAB running, but minimize it by clicking on the appropriate box. Open the **icebreak** file and double click on **icebreak.exe** to start the program. The screen will display the main *ICEbreaker* window as shown in Fig.5, and possibly the Watch and Source windows (Figs. 7 and 8). In the main *ICEbreaker* window click on “Options” and then select “Programmer” this will produce the communications set up box shown in Fig 6. In this box set up the serial COM port that you are using. If a 20MHz crystal is fitted the Baud box must be set to 38400. Other crystal frequencies can be used and the Baud rate adjusted proportionally – e.g. a 5MHz crystal would operate at 38400/4 or 9600 Baud. Once set up close the box by pressing OK.

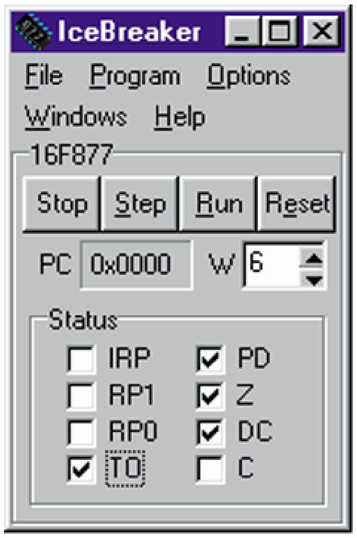
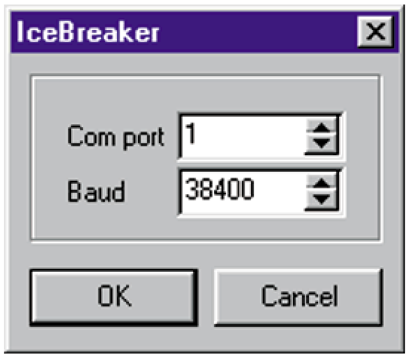


Fig.5. Main ICEbreaker window.

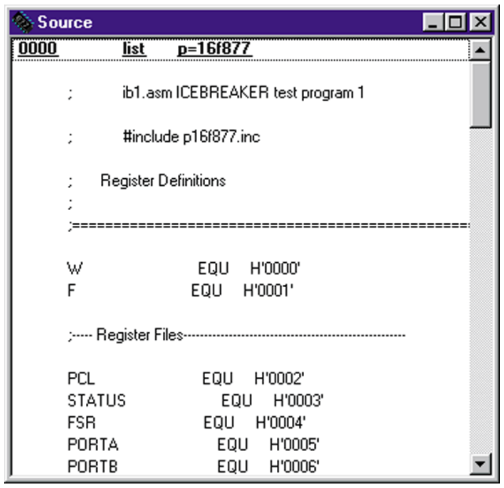
Back in the main box select “File” and “Open”, which will reveal a standard file select dialog window listing the files in the **icebreak** directory. Select and load **ib1.asm** and then open the source code window by selecting ‘Window’ and then ‘Source’. The window should contain the **ib1.asm** source file with numbered lines as shown in Fig.7.

Before the program can be run it must be sent to IC1 by selecting



Com port 1
Baud 38400

Fig.6. Setup box.



```
0000         list      p=16f877

;         ib1.asm ICEBREAKER test program 1

;         #include p16f877.inc

;    Register Definitions
;
;==========================================================

W                 EQU    H'0000'
F                 EQU    H'0001'

;---- Register Files----------------------------------------

PCL               EQU    H'0002'
STATUS            EQU    H'0003'
FSR               EQU    H'0004'
PORTA             EQU    H'0005'
PORTB             EQU    H'0006'
```

Fig.7. Source file window.

"Program" and then "Program" in the main *ICEbreaker* window. A progress bar appears and *ICEbreaker* sends the program to the first 4K of the program memory in IC1. Once this is completed, it should be possible to step, run, and reset the code one line at a time using the "Step" button in the main window. In single step mode, at each step, a highlight line progresses through the source code window, and the main *ICEbreaker* window shows the Program Counter, the contents of the W register and the Status register bits.

Sometimes the highlight does not track the source code exactly, and is one line above or below the current line. This is due to the communications between the computer and IC1 and depends upon the way some of the source code is written; it is only a minor inconvenience as the actual line of code is easily worked out from the program counter in the *ICEbreaker* main window.

Other registers may be set up in the "Watch" window – select "Windows", "Watch" in the main window. Fig.8 shows the main Watch window and Fig.9 the "Add" window. Registers may also be "Watched" by setting the first location and entering the number of registers in the "Array" box.

The selector box in the "Watch Add" window allows a choice of locations, labels and registers to be selected. It is important to understand that some of these options are not what they seem, for example the W option returns not the value of the W register, but the number 0 which has been assigned to the label W in the fifth line of the source code.

TESTING A PROGRAM

Once some familiarity has been achieved with the *ICEbreaker* windows, it is time to connect some hardware and see how it operates. As with all good microcontroller hardware systems, the first thing to do is flash a LED! The **ib1.asm** program counts up through PORTA, which is set to output mode, and so all that is necessary is to connect a LED from pin 2 of IC1 via a current limiting resistor (anything from 100Ω to 2k2) to 0V.

Using solid core 1/0·6 connecting wire links it is an easy matter to put the LED and resistor on the breadboard and make the two connections to the turned-pin socket strips. The row of 13 sockets at the bottom of the patch area is a good place to find 0V. Provided the program has been set up correctly and loaded into IC1, the LED should flash when the program is set to "Run".

In order to flash the LED slowly, the program has three nested counting loops. To single step through them would take years, and so it is impractical to go right through all of the states of PORTA. The alternative to single stepping is to insert a breakpoint and run the program to there.

Select "Options", "Breakpoint" from the main *ICEbreaker* window and set the value to 24. Fig.10 shows the "Breakpoint" setting window. Entering a breakpoint highlights the line in the "Source" window. "Reset" and then "Run" the program and it will now stop at the breakpoint. Press run again and it will loop again to the same breakpoint – each time incrementing the value at PORTA so that the LED on PORTA 0 turns on and off alternately. Try connecting the LED to IC1 pin 3 PORTA 1 and see that it switches every other loop.

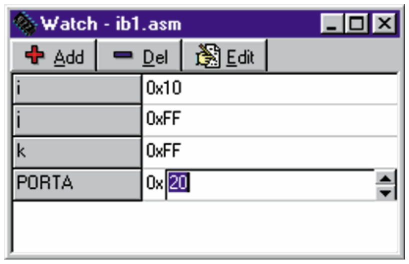
Now that a LED can be flashed, it is just a few more steps to controlling all sorts of peripheral devices, and whilst the PIC16F877 cannot run the proverbial "Power Station" it is capable of an amazing number of very complicated feats. The development of longer programs controlling more hardware is so much easier when it is possible to test the programs quickly in this way. Single stepping and watching the program and data registers allows even complicated routines to be tested and debugged, and simple changes can be made and checked immediately.

To make a simple change to the **ib1** program, select "Program", "Code" from the icebreaker main menu and then select location 14. The contents can be read and should be 30FF, which means MOVLW FF. Modify the value to 3010 which will load the value 10 instead of FF and press the "Write" button. Clear the breakpoint, "Run" the program, and see the change in speed.

The program in IC1 has been modified, but remember that the Source Code has not, and so will need changing to the new value once the required speed has been set. To modify the source code run MPLAB, modify **ib1.asm**, recompile the code by selecting "Project", "Make project" (or by pressing the appropriate shortcut button) and then switch back to *ICE-breaker* .

Select "File", "Reopen" and the modified source code will appear in the *ICEbreaker* "Source" window. To complete the operation select "Program", "Program" and the new code will be loaded into IC1. Although the program in IC1 had already been modified, it is always good practice to reprogram with the newly compiled code to prevent simple errors creeping in – especially when a number of modifications might have been made.

As well as changes to the program memory, the same procedure can be used to modify the EEPROM, and register



i	0x10
j	0xFF
k	0xFF
PORTA	0x20

Fig.8. ICEbreaker "Watch" window.

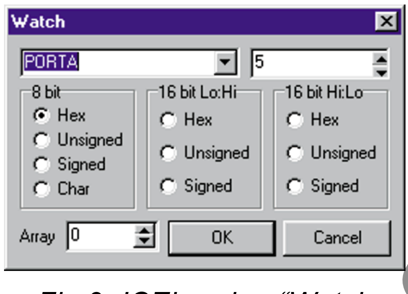


Fig.9. ICEbreaker "Watch Add" window.

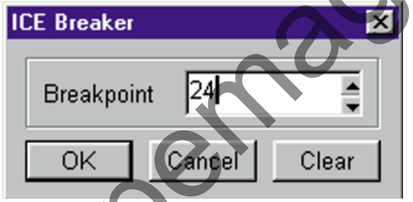


Fig.10. ICEbreaker "Breakpoint" window.

files. Changing register file contents is particularly useful when combined with single stepping, as it allows routines to be tested with a range of values, for example a timing loop can be set up with 00 in the loop counting register and single stepped to see what happens at the end of the loop.

Once experience is gained, the range of tools available will be understood, and it will become easy to set up and check simple routines and combine them into full programs.

OTHER PROGRAMS

Program **ib2.asm** is a simple driver for the stepping motor. Connect PORTA 0 to 3 to the four stepping motor drive sockets P1 to P4 and then follow the procedures used for **ib1.asm** to compile, load and run the program. Notes are included in the code suggesting modifications that can be made to the code for altering speed, direction, and duration of travel.

Program **ib3.asm** runs the LCD. It uses six connections from PORTC 2 to 8 to connect to RS, E, and D4, 5, 6, and 7 of the display in that order. The code initializes the display, and then can be set up as a subroutine to write any character to any display location. The source code has notes to explain the operation and to suggest possible changes for more advanced applications.

COMPLETED PROGRAMS

Once a program has been debugged and is working correctly, it can be programmed into another PIC16F877 or any other suitable PIC chip using an appropriate programmer (*PIC Toolkit Mk2* from the May and June issues of *EPE Online* is ideal – **www.epemag.com/0599p1.htm**). The *ICEbreaker* code does not have to be in the chip and so any blank chip can be used with this method. The *ICEbreaker* board can only program chips that already contain the special **icebreak** code – this is necessary because the chip has to communicate with the PC via the standard serial port interface. Chips with **icebreak** code are readily available – see the **Shoptalk** page.

Once programmed, *ICE-breaker* chips will run normally in other circuits if required to do so but it is important to make sure that the two pins RB6 and RB7 are connected to 0V. This is because the *ICEbreaker* software automatically starts to run, and immediately checks for ground connections on these pins. If it finds that they are grounded, the program jumps to location 0000 and starts running the program from there, as a normal chip would.

THE NEXT STEPS

It is tempting to continue and describe the many features of the PIC16F877, but it really is an impossible task because the chip is so powerful (see also our *PIC16F87x Mini Tutorial* in the Oct '99 issue of *EPE Online*). The beauty of the PIC range of devices is that it is possible to run the same code on many different chips.

EPE ICEbreaker allows programs that are intended to be run on much simpler chips to be checked and debugged. All that is necessary is to ensure that the ports and register addresses are compatible with the smaller chips. Applications for the PIC16F84 are particularly suitable for development using *ICEbreaker*, and so the programs previously published by *EPE* can be used. Note though that the MPLAB environment uses MPASM code, and so the *PIC Toolkit Mk2* software will be necessary to convert the original TASM source code to MPASM assembly language.

ICEbreaker provides an advanced way to learn programming. Along with the PIC programming and data sheets and back issues of *EPE,* it will become an indispensable tool for learning, development, and testing of PIC projects.

Go to next section