

# Using Visual Basic with E-blocks

## Introduction

I was recently asked by a friend of mine how he could develop a small window display which gave instructions to couriers on where to leave parcels when he was not in the office. The brief here was that he wanted to enter a text script (a neighbour's address) into a PC based application and have the message displayed in the Window. Now we could have another monitor which was visible from the outside of his house but that seems like overkill. Using a small LCD based circuit and a microcontroller would be much more appropriate. The essence of the task here is how to get a PC to communicate to a microcontroller based system. This happens to be a question that has been raised many times so I thought it would make a good project.

## The aiming point

Figure 1 shows you the basic diagram of the final system we need. In terms of hardware the display system we want needs a central microcontroller (in this case PICmicro microcontroller), a USB interface (FTDI) and an LCD display. For good measure I have also included some LEDs and switches. The switches will allow us to make a standard display from one of several messages, and the LEDs might come in handy later. I have a spare 'rats tail' power supply which gives 18V output and a standard 7805 will provide the 5V power rail the system needs.

So my first action was to develop a prototype system from E-blocks as you can see in Figure 2. Most readers will by now be familiar with the E-blocks boards we have used here: a PICmicro microcontroller Multiprogrammer, LED board, Switchboard and LCD board. I am using a PIC16F877A as the core microcontroller: this is a bit of overkill but I reasoned that I can always use a smaller PICmicro device later on. One E-blocks board that may not be familiar to you is the USB232 E-blocks board shown in figure 3. This is based on the highly successful FTDI USB to RS 232 interface. How this works internally I could not tell you: as far as using the device is concerned the FTDI device converts a USB signal to an RS232 signal and the board is supplied with a set of virtual COM drivers which allow you to

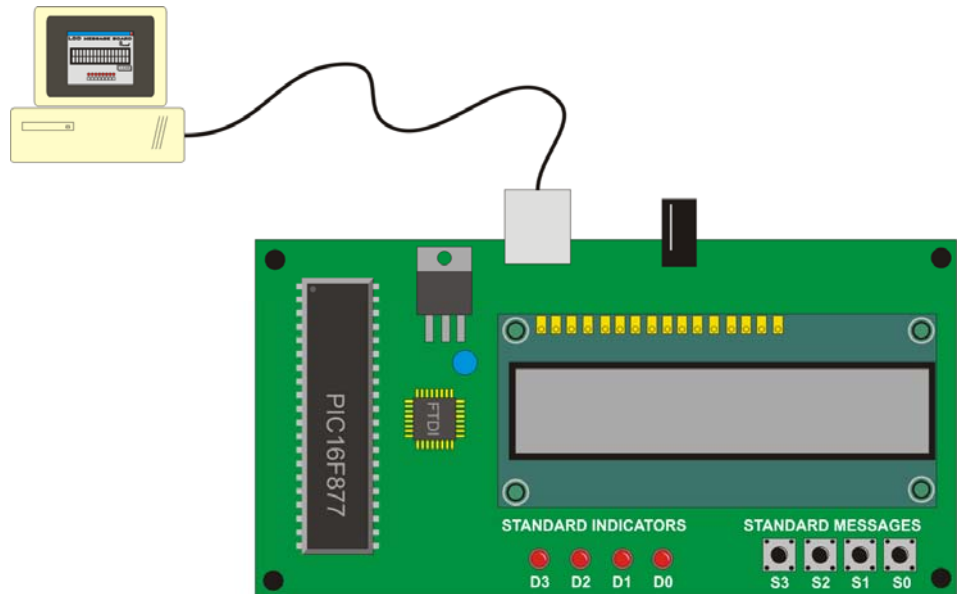


Figure 1 – what we want to end up with

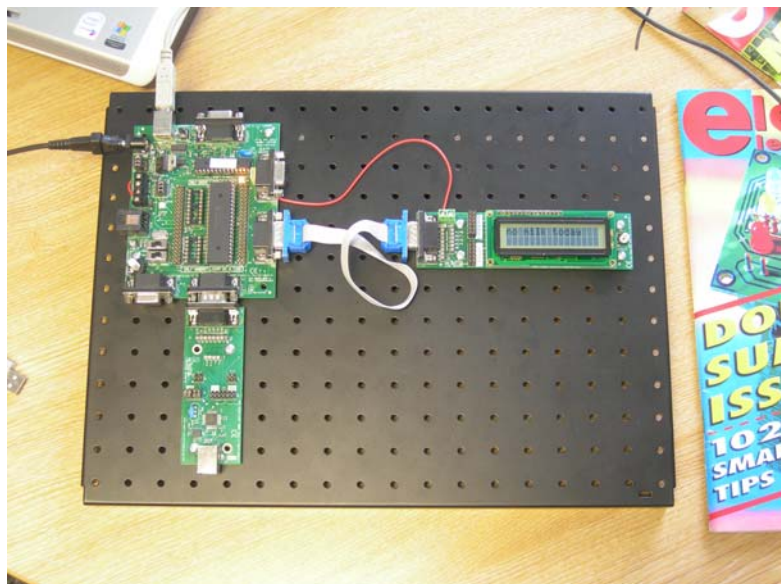


Figure 2 – E-blocks prototyping system – note the short 9 way ribbon cable on the LCD

communicate with it using standard Windows software – in this case Visual Basic. For a test jig I set up a Visual Basic program that did more than the overall brief.

I designed a screen with a display area, 8 switches, and 8 LEDs. The status of the 8 switches on port D of the Multiprogrammer is monitored by the 8 LEDs inside the Visual Basic application. Correspondingly the LEDs on port A of the Multiprogrammer reflect the status of the switches in my Visual Basic application. These were really useful during the initial design phase of my program

just to allow me to check that I could transfer data to and from the PICmicro device, and I left them in the software application as they might come in handy for further projects.

In the final program to display a message on the LCD I simply type the message into the PC application's text box which updates the LCD display on port B of the Multiprogrammer immediately. The CLEAR button allows me to wipe the display. The host application is written in Visual Basic which you can see in Figure 5. I think it is a little beyond the scope of this short article to delve

# Using Visual Basic with E-blocks

deeper into how the code in the Visual Basic program works – but those of you who are interested should email us and we can send them to you (you will need VB to view them).

For the PICmicro microcontroller I developed a program in Flowcode version 3 which includes a handy RS232 icon that allows communication to the FTDI device. Almost the entire program is shown in the Flowcode printout in figure 4. Those of you who do not have a copy of Flowcode can also download a 30 day demonstration version (which allows you to see how this program works) from our website.



Figure 3 – USB232 E-blocks board

## 16F877A Flowcode program explanation:

If you examine this printout from Flowcode you can see how the PICmicro program works. First we start up the LCD and show a welcome message. In the main program we get the inputs from the Port A switches – if any switches are pressed we send the data back to the PC using the input switch subroutine (not shown in detail). Then we check for an incoming message on the RS232 pins – if there is not (255 returned from the subroutine) then round the loop we go again.

The VB program sends characters '<' and '>' as instructions for Port B outputs ('<' = pin output on and '>' = pin output off). So if there is a message then the next decision box checks the message is not a port B output message – otherwise the incoming character is displayed on the LCD. The second decision box detects a line feed character – generated by the CLEAR button in the VB application: if one of these is received the program clears the display and sends the program back to point A. The next decision box detects a '<' character and if one is received gets the subsequent RS232 character, containing pin information, and puts the appropriate pin on port B high. The last decision box detects a '>' character and if one is received gets the subsequent RS232 character and puts the appropriate pin on port B low.

Those of you that know about LCDs will be aware of the fact that a 16 character display has 32 internal characters so this program might appear not to work – in practice the VB program takes care of the extra spaces.

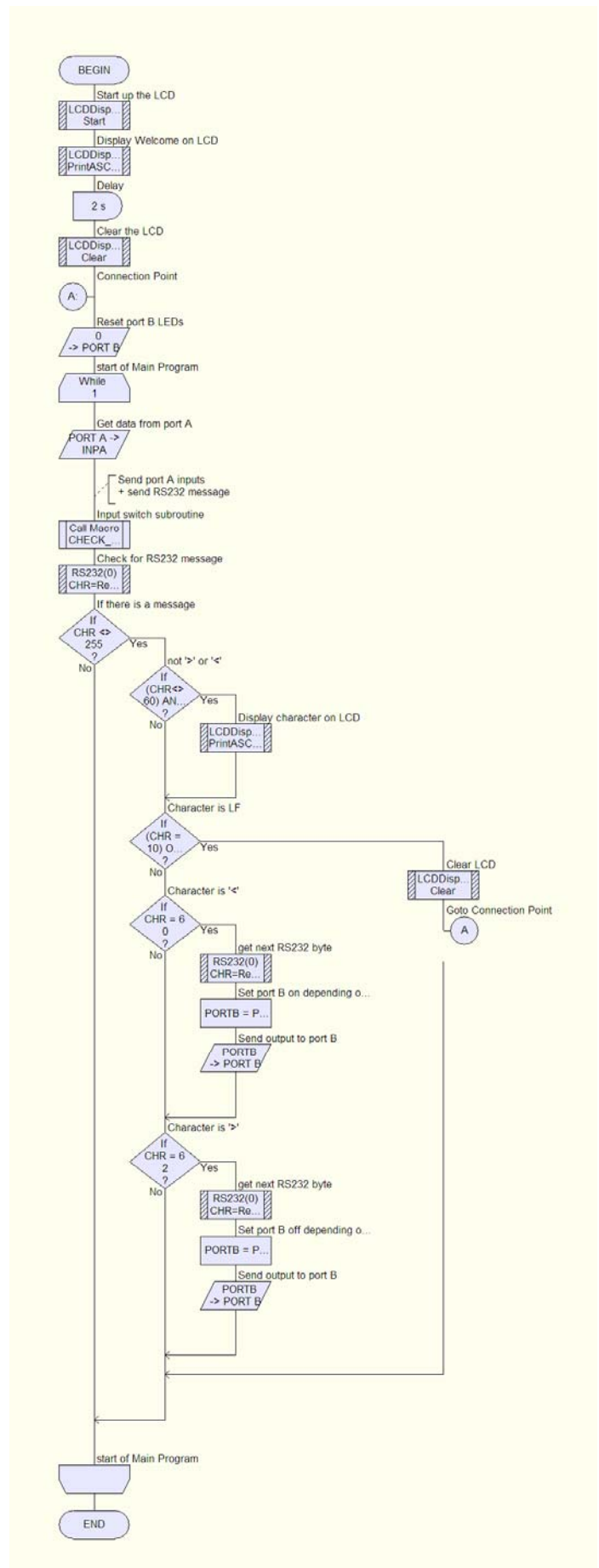


Figure 4 – Flowcode program

# Using Visual Basic with E-blocks

## Conclusion

The FTDI device is great for making the task of connecting to USB easier – we managed to create the whole application in less than a day and the results are great for a first prototype. The cost of the E-blocks involved was under Euro 150 (£100) ex tax which meant that it was not even worth making a PCB: in the end we just used the prototype system shown in figure 2 with a 0.6m IDC cable to the LCD display. It worked fine.

Note that this system can be used to make E-blocks link to any PC based application: Visual Basic, Visual C++, or National Instrument's LabVIEW.



Figure 5 – Visual Basic application