

PC to RC Interface

By Ken Hewitt

Introduction

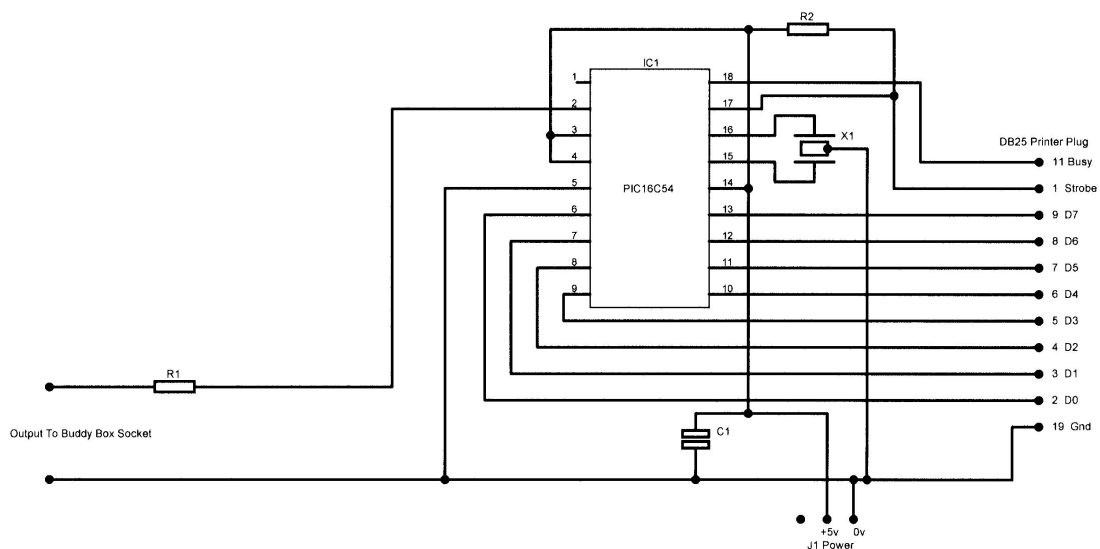
This again was a project inspired by watching the RC news groups on the Internet and also by emails I have received over the last year or so. There have been a lot of people asking how to connect a PC to an RC transmitter via the buddy box socket so that they could have their PC control some sort of RC vehicle. The result is the design you see here, I have made it a little more versatile than just a simple 4 channel unit, you can use it to send anything from 1 to 8 channels and you can also set the frame rate from 15mS to 30mS.

I have a simple windows program that works with this interface and allows you to adjust the frame rate and select how many channels you want to generate and then set the servo positions for each channel with a slider on screen. This is just a demo program and the user will have to write their own PC program to control an RC vehicle themselves. You could use this demon program as an 8 channel servo controller by connecting the output of the interface into the input of the direct servo controller I did a couple of years ago.

Since starting this project another use for it has arisen, I was asked about using a PC joystick connected to a transmitter for someone suffering from motor neuron disease who could not use a standard RC transmitter anymore well this interface is of use there, all someone needs to do is write a PC program to read the PC joystick and send the data to the interface, this can be done on a laptop and used at the flying field.

Technical Bit

The interface is built around a PIC micro controller and is plugged into the PC's printer port, there is no printer pass through function so the printer must be disconnected or the interface used on a spare printer port.



The interface receives information from the PC about how many channels to generate and their values and also what frame rate to use. The unit will then generate this signal in a standard buddy box format even if disconnected from the PC.

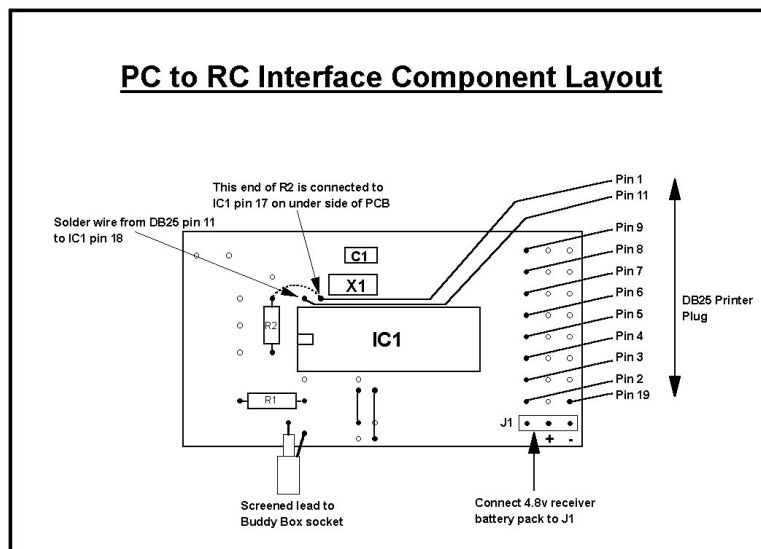
The PIC16C54

The microcomputer used in the differential is a PIC16C54, which has a RISC like CPU, and supports 33 instructions. The chip contains everything that is required to form a fully working microcomputer, it has 12 input or output pins, 512 program memory locations and 25 bytes of RAM. This may not sound like much but because of the RISC type architecture the resulting code can be very compact. It also has a wide range of power supply limits, 2.5 volts to 6.25 volts at less than 2mA, making it ideal for use in model avionics systems.

Construction

The unit is built on a PCB that was originally design for a past project (the direct servo controller) and so there are a few parts of the construction process that need careful attention, namely fitting R2 and connecting the

wires that go to pins 1 and 11 of the 25 way D plug.



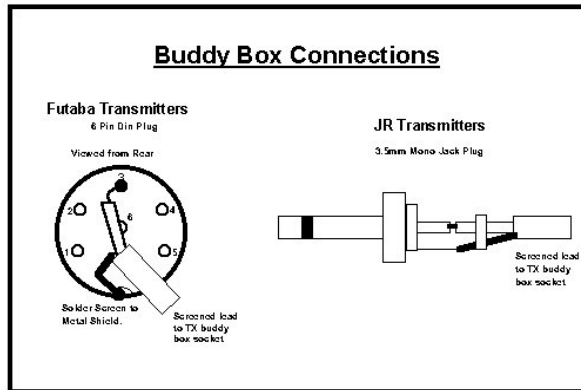
R2 is fitted to the board and the end which goes through PCB where there is no pad on the trackside for it to be soldered to is routed so as to miss any pads or tracks and is then soldered to pin 17 of IC1.

The wires that go to pins 1 and 11 of the 25 way D plug for the printer should be soldered to pins 17 and 18 of IC1 through the holes in the board next to these pins.

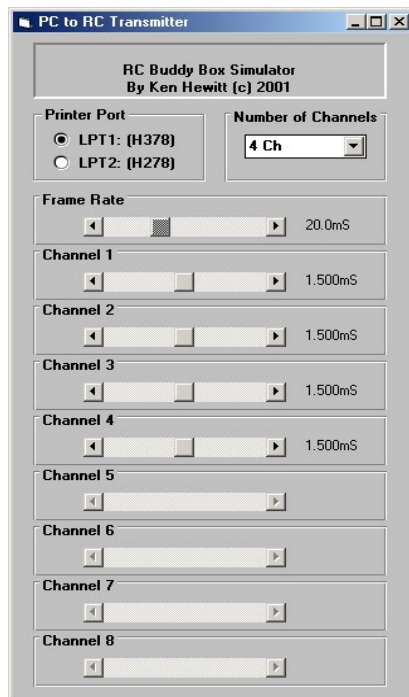
The unit only consists of 5 components so there is very little that can go wrong. The only component that needs to be fitted the correct way round is IC1. The correct orientation of this component can be clearly seen in the component layout drawing.

The main component IC1 used in this design is a CMOS device and can be damaged by static electricity. When handling this item it is advisable to take some basic precautions, do not wear clothing which builds up a static charge, or handle the item until needed and before you touch it, try to touch a water

pipe which should earth any static charge you have built up. DO NOT connect yourself directly to the mains earth.



Windows Demo Program



A windows demo program written in Visual Basic 3.0 is available for download from my web site <http://www.welwyn.demon.co.uk> just for the links to the section on the PC to RC interface. The demon program is downloaded as a zip file and you need to unzip it into a temporary directory and then run the set up file setup.exe this should install all of the parts in the right places.

I am not a windows programmer so it is only a fairly basic program to show what can be done and how to do it.

PC to Interface Protocol

This section is the bit for those people that wish to write their own software and will describe the format of the data transfer between the PC and the interface unit.

The PC will always send 9 bytes to the interface no matter how many channels are to be generated. The first byte is the frame rate and must be a value of between 59 and 117 this represents 15.1 to 30mS frame rates.

The next 8 bytes are the channel data bytes and represent the value of the pulse width produced for each channel; they must be in the range 0 to 255. The value 0 means do not generate a pulse so if you want to only generate a 3-channel signal the all of the remaining bytes after channel 3 should be 0. The values between 1 and 255 will produce a pulse width from 0.865 mS to 2.035mS with the value 128 giving a centre value of 1.5mS

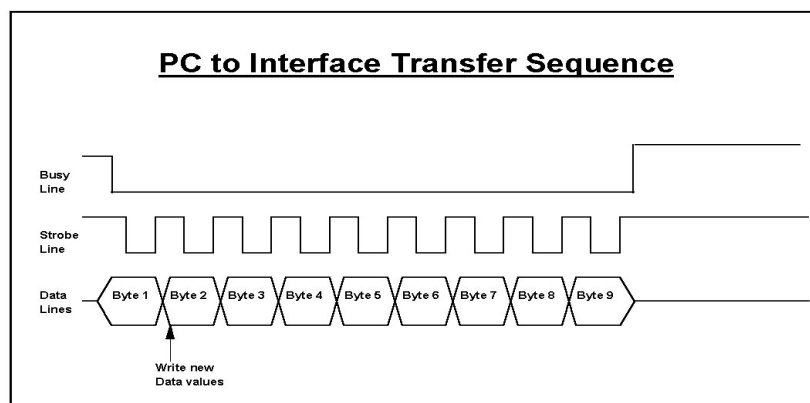
You only need to send data when you want to change the values, the interface will keep generating the signal to the last set of values that you sent to it, this frees up the PC from having to keep sending data every 16 to 30mS.

Data Transfer sequence

The interface unit will only accept data from the PC when it is not generating pulses during the frame synchronisation period, it signals this by setting the busy line (pin 11) low.

When the PC has data to send to the interface it should wait for the busy line to go low and then write the first byte of data to the port and then set the strobe line low (pin 1), when the interface sees the strobe line go low it will read the data on the data lines of the port and then wait for the strobe line to go high again.

The PC should set the strobe line high after a short delay (50uS) then write the next data byte to the port and again wait for about 50uS before setting the strobe line low. The interface will read the data on the port every time it sees the strobe line got low. The PC should only change the data on the data lines of the port while the strobe line is high.



This process is repeated until all 9 bytes have been sent at which point the interface will set the busy line high and wait for the start of the next RC frame and generate pulses with the new values.

Finally

I hope that you have found this to be of interest and I hope that it will be just one of a range of projects for brand new items that have never been presented before.