Thank you for

Brendan Hughes

Over the years, there have been a fair number of designs published enabling a radio-control (RC) transmitter to interface with a personal computer. Having this interface enables prospective model aircraft pilots to hone their skills using a simulation program rather than aviating their pride and joy nose-down into the nearby landscape.

Arguably, many RC modelling enthusiasts would rather see a PC 'crash' than the latest model built with blood sweat and tears, not forgetting lots of time and money. In this respect, the follow a buzzword in modern electronics: *simulation*. Simulating flights, landings and takeoffs for a given model is a great way of familiarising yourself with its response to your actions (and errors) on the RC transmitter. Excellent flight simulators are available these days that give very realistic results — to the extent of users actually starting to sweat and exhaust themselves trying to keep the model where it should be — up in the air!

The circuit described in this article is the 'glue' between the 'buddy' (or 'trainer') connection on your RC model transmitter and the virtual model aircraft, car, boat or even helicopter gracefully finding its way across the PC screen in response to your operating the joystick(s) and pressing buttons. No model will be lost to unforgiving rocks, trees, church steeples or Farmer Jim's cowherd. If you crash, simply start the simulator again and *do better*.

flying USB-FliteSim

An RC transmitter-to-USB interface

Goodbye gameport, welcome USB

Most designs interface to the PC via the gameport which is now becoming less common on newer PCs and has disappeared completely from laptops and notebooks. The design discussed in this article utilises the USB port which offers greater accuracy. Some commercial designs offer similar capabilities but most only have 6-bit precision on the linear axis, so small trim changes may not be effective.

Capabilities and limitations

As published here, four linear controls and four switched controls are catered for as would be used in a typical 8-channel transmitter, i.e., two 2-axis joysticks and four switched inputs. The linear inputs are measured with 12-bit accuracy although in reality just over 11-bit accuracy is achieved with this software on a typical RC set. With this level of resolution, poor joystick centring is easily measured using the joystick calibration program in Windows (Select 'Display raw data'). More channels could be easily added but it was felt that eight would be adequate for most users.

Super simple hardware

The hardware is simplicity itself, see **Figure 1**. At the heart of the circuit is a PIC18F2550 clocked at 8 MHz, with a simple transistor buffer/inverter on the input. Eight jumpers have been included although only four are presently used to select between different options. The remainder are to enable possible future enhancements.
When plugged into an USB connection on the PC, the HID firmware in the F2550 enables the circuit to be enumerated as a 4-axis with 4-button joystick, so no additional drivers are required.
Note that due to the PIC software used, the circuit is a low-speed USB device and Chapter 6.4.4 of the USB1.1 specification states that USB cables should be hardwired to the peripheral and not use the USB 'B' connector. However, considering that the circuit will typically be for personal use only, the 'B' connector was elected.

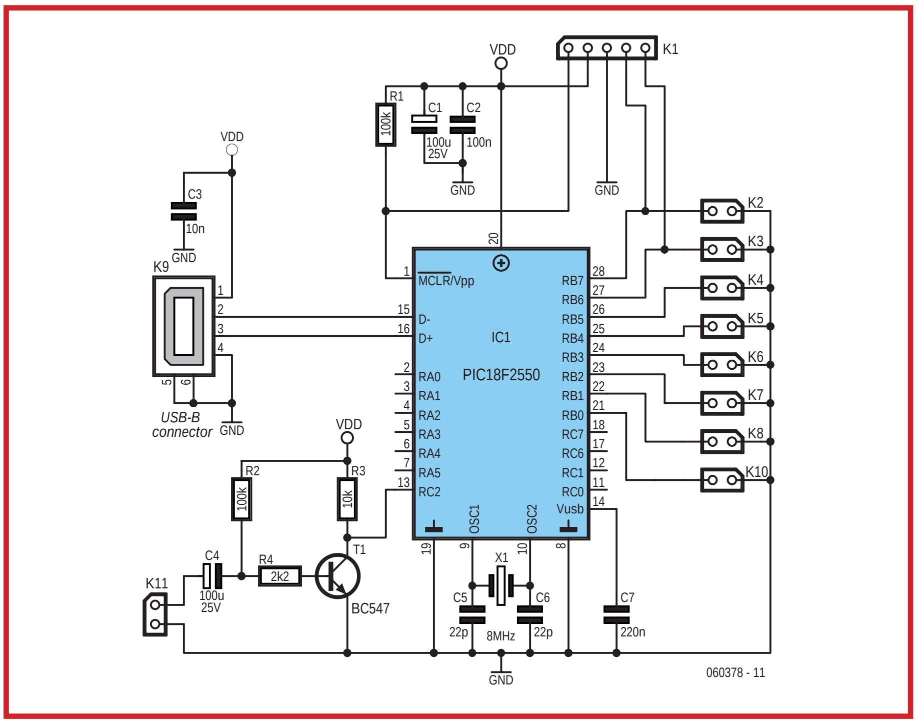

Figure 1. Circuit diagram of the RC TX to USB interface. Hardware, what hardware?

Software

The following description of the software is pertinent to the PIC 16C745. See the heading 'Project History' for a brief overview of the differences to the current 18F2550 software.
All the USB dedicated software is available from the Microchip website and is included with the source files supplied free of charge through the Elektor website as file no. **060378-11.zip** (see month of publication). A snippet of the extremely well-commented source code listing is shown in **Listing 1** — very useful for the jumper descriptions!
Of the Microchip supplied files, both *DESCRIPT.ASM* and *USB_CH9.ASM* need to be modified. *USB_CH9.ASM* needs the following compiler directive commenting out (or removing) so that port B is available for our use:

```
#define SHOW_ENUM_STATUS
```

DESCRIPT.ASM needs some more serious editing of the various descriptors to allow for proper enumeration and operation of the USB functions. Seven bytes are sent to the PC every 10 ms. The arrangement of the data within these seven bytes is laid out in the report descriptor. Essentially, four blocks of 12 bits representing the four joystick axes followed by four bits representing the four switches are sent. That makes a total of 52 bits, which falls short of the 56 bits available in seven bytes,

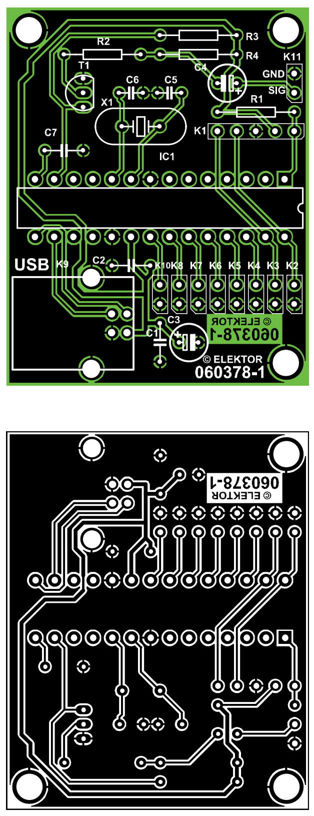

Figure 2. Copper track layout and component mounting plan of the miniature PCB designed for the interface.

COMPONENTS LIST

Resistors
R1,R2 = 100kΩ
R3 = 10kΩ
R4 = 2kΩ2

Capacitors
C1,C4 = 100µF 25V radial
C2 = 100nF
C3 = 10nF
C5,C6 = 22pF

Semiconductors
IC1 = PIC18F2550-I/S, programmed, order code **060378-41**
T1 = BC547

Miscellaneous
K1 = 5-way SIL pinheader
K2-K10 = 2-way SIL pinheader with jumper
K11 = 2-way SIL pinheader
K9 = USB-B connector, PCB mount
X1 = 8MHz quartz crystal
PCB no. **060378-1** from The PCBShop
PIC source code files, free download no. **060378-11** from www.elektor-electronics.co.uk

therefore a further four bits of padding are sent.
The *RC_USB.ASM* source file has a good number of comments so should be fairly easy to follow. Because the USB functions make unpredictable use of the interrupts, these are not used for pulsewidth measurements. Therefore, the only user of the interrupt facility is the USB routine.
Pulsewidth measurements are made using the Capture/Compare/PWM module. Capture register CCPR1 is a 16-bit register configured to capture the contents of Timer1 on either the High-to-Low or the Low-to-High transitions on the input (as selected by jumper K10 on RB0). Timer1 runs continuously with a ÷2 prescaler at 3 MHz and therefore increments every 333 ns. Pulsewidth can therefore be detected to an accuracy within 666 ns.
Due to the way servos are controlled, pulsewidths vary from 1-2 ms for each channel, therefore we have a range of approximately 0 to 3000.
When the program starts, *InitRC_USB* is called that configures the ports, sets up the CCPR to capture on a rising edge and starts Timer1. Next, InitUSB is called and the device is enumerated. The firmware waits until enumeration is complete.

LOOP is the main body of the program. If a pulse is detected (CCP1IF bit set), we check if it is a synchronisation pulse (>2.7 ms) or one of the channel pulses, which vary between 1 and 2 ms pulsewidth. The last value of CCPR1 (Tmr1Lo and Tmr1Hi) is subtracted from CCPR1 to give pulsewidth in units of 333 ns. If it is a sync pulse, we send the data in the *BUFFER* to the USB routines for transmission to the PC. Else, if a normal channel pulse is detected, we subtract 4500 (4500 counts of 333 ns = 1.5 ms) to centralise the pulse on 1.5 ms so that positive numbers indicate a positive swing from neutral on the joystick and negative numbers indicate a negative swing. Next, the pulse width information is stored at the appropriate place in *BUFFER* as pointed to by the *Pulse_Count* variable. *Temp_Count* is a working copy of *Pulse_Count* that can be manipulated without losing track of the channel number.

Jumpers for unusual cases

Left-handed modellers may wish to have the aileron/elevator joystick on the left. To this end, jumper K8 on port line RB1 adjusts the value of *Temp_Count* so that the data is stored in the correct part of *BUFFER*.
Certain RC transmitters use a non-standard sync pulse. This may affect the operation of the device. Installing the jumper on RB0 causes *CCPR1* to capture on the falling edge of the pulse train. Unfortunately we did not have access to any of these non-standard RC radios so we cannot guarantee that this will help.

Construction

The interface is built on a small printed circuit board of which the true-size artwork is reproduced in Figure 2. This board is available from Elektor's business partner The PCBShop who reside at www.thepcbshop.com.

With so few parts on the circuit board, — and all of the 'leaded' variety as opposed to SMDs — there should be no problems building the interface if you exercise normal care in fitting the parts to match the component overlay, and of course the soldering. We recommend fitting the PIC micro (IC1) is a 28-way narrow DIL socket.
We reckon there's much to be learned, enjoyed and economised upon if the project is undertaken as a joint undertaking by RC modelling club members. Subtasks can be assigned like component/PCB purchasing, soldering, programming and software tinkering to those with the relevant skills or their arm twisted.

Calibration

When the interface is plugged into a USB port on a PC, it should enumerate with a message stating that a 'RC/USB Interface' has been found. Open up the Control Panel and select 'Game Controllers'. Listed in the dialogue box should be 'RC/U' or 'RC/USB Interface'. Select the controller and click on Properties. Movement of the joysticks should produce the required movement on the screen. If no movement is observed, then toggle jumper K10. Huh, "toggle"? If the jumper is Fitted then Remove it and vice versa. Similarly, toggling jumper K8 will cause the two joysticks to be swapped. When it is working as required, the system will need to be calibrated. Select 'Settings' and in the new dialogue box select 'Calibrate'. Follow the instructions onscreen. This completes the installation.

Wrong enumeration

For some reason the device may be referred to as ‘RC/U’ even though Windows retrieves the full name of ‘RC/USB Interface’ during enumeration. If this bothers you, simply edit the registry setting at

```
HKEY_LOCAL_MACHINE\SYSTEM\
  ControlSet\Control\MediaProper-
  ties\PrivateProperties\
  Joystick\OEM\VID_04D8&PID_FE70
```

Each USB device manufacturer is allocated a unique Vendor ID code (VID) and each device model that the manufacturer produces is allocated a Product ID code (PID). We have obtained a sub-licence from Microchip to use the Microchip VID (04D8) with a PID code of FE70. This should ensure that this device will not conflict with any other commercial USB device.

Interlude — odds & ends

Note that the interface will only decode Pulse Position Modulation (PPM) pulses and not Pulse Code Modulation (PCM), and therefore the transmitter will need to be in PPM mode.

A list of the buddy-lead pinouts for various RC transmitter manufacturers can be found at [1] and [2].
A good tutorial on the on the principles of PPM can be found at [3] and [4].

Project History

Originally the software was written for the PIC16C745, and later modified to work on a 18F2550. Microchip did not (yet) release USB framework code for the 18F2550 in assembler format. Fortunately, Brad Minch of Olin College has generated an assembler framework that is freely available [5]. This code was adapted and mated to the file *rc_usb.asm* that was tweaked for 18F2550 code to produce the file *RC_USB_18F2550.asm* which needs to be compiled with the included *ENGR2210.inc* and *usb_defs.inc* files. The code should also run on the 18F2455 without any further adjustments.

The advantage of the 18F devices is that they are flash-programmable and faster to erase. K1 is a 5-pin header that allows in-circuit programming of the device with an appropriate programmer, like the Microchip PICkit2 (pin 1 of PiCkit podule not used).

Listing 1. Source code snippet

```
;*******************************************************************
; filename: RC_USB_18F2550.ASM   Ver 1.0 - 01 Dec 2006
;
; This file implements the conversion of a PPM mo-
   dulated output from a radio
; control transmitter to a 3 axis plus throt-
   tle and 4 button USB joystick.
; PORTB,0 pin header selects inverted in-
   put i.e. pulses are active low
; PORTB,1 pin header selects joystick swapping
; PORTB,2 pin header selects the Airtronics option
; PORTB,3 pin header selects the JR option
; PORTB,4..7 not used
; The code is written for a Futaba transmit-
   ter but by installing EITHER PortB,2 or 3
;   pin header, then it can be configu-
   red for an Airtronics or JR radio
; The USB port is configured to interrupt eve-
   ry 10mS and sends 7 bytes of data
;  (maximum is 8).  The 4 joystick chan-
   nels are sent as 12 bit values and the 4
;  switches as boolean values.  Therefore, 52
   bits are required to be sent and the
;  7th byte is filled with 4 bits of ‘padding’
; The following shows how the bits are saved
   in the Buffer prior to being sent
;  to the USB port
;  Throttle=T  Rudder=R  Aileron=A  Elevator=E
   Switches=S  Padding=P
;              MSB                          LSB
; Buffer0      A7   A6   A5   A4   A3   A2   A1   A0
; Buffer1      E3   E2   E1   E0   A11  A10  A9   A8
; Buffer2      E11  E10  E9   E8   E7   E6   E5   E4
; Buffer3      T7   T6   T5   T4   T3   T2   T1   T0
; Buffer4      R3   R2   R1   R0   T11  T10  T9   T8
; Buffer5      R11  R10  R9   R8   R7   R6   R5   R4
; Buffer6      P    P    P    P    S4   S3   S2   S1
;
;********************************************
   **********************************
; All USB routines kindly provided by Brad-
   ley A. Minch of the Franklin W. Olin
;  College of Engineering and the origi-
   nal source can be obtained from
;      http://pe.ece.olin.edu/ece/projects.html.
; The source was the Lab2 project that was
   then modified by myself with
;  permission from the author to distribu-
   te as required.  The main areas of
;  change are the descriptors up to line 265
   and all code after line 1178 is
;  new.  There are a few small changes in between.
;
; Revision History:
;      2006-12-01             Versi-
   on 1.0                     Brendan Hughes
;*******************************************************************

#include <p18F2550.inc>
#include <usb_defs.inc>
#include <ENGR2210.inc>
```

Those interested in learning more about USB are advised to have a look at websites [6] through [9].

A full set of source files for both the 16C745 and 18F2550 processors is supplied through Elektor’s website. It should be noted though that hardware changes are required in the circuit if the C745 is used: change the quartz crsystal to 6 MHz and fit a 1kΩ5 resistor between Vusb and the USB D-line.

(060378-I)

Web links

[1] http://users.belgacom.net/TX2TX/tx2tx/english/tx2txgb3.htm
[2] www.rc-circuits.com/Transmitter%20Connector%20Pinout.htm
[3] www.mh.ttu.ee/risto/rc/electronics/radio/signal.htm
[4[http://rc-circuits.com/PPM%20signal.htm
5] http://pe.ece.olin.edu/ece/projects.html
[6] www.usb.org
[7] www.lvr.com/
[8] www.beyondlogic.org/usbnutshell/usb1.htm
[9] http://pe.ece.olin.edu/ece/projects.html