FMS Flight Simulator Encoder

Practice without crashing your models

By Dean Sarelius d.sarelius@bigpond.com

We all know that even the best pilots spend many hours practising on flight simulators before taking to the real thing, so why should it be any different for radio control planes? With the FMS R/C Flight Encoder described here you can polish your flying skills and gain the valuable experience you need without having to invest in a radio control or even to build a model airplane. In fact the FMS encoder will even allow you to use your own radio control to further enhance your flying ability.

Last year the author purchased a JR 4-channel radio control (R/C) and built his first electric powered plane. Unfortunately the maiden flight was a disaster so a more serious look at flight simulators was in order. Unfortunately, the cost of the so called professional simulators was too much to justify. Searching the Internet, a website called FMS was found.

FMS (Flying Model Simulator) is a freeware 3D flight simulator program developed and written by Roman & Michael Möller. FMS has been compiled to run in both DOS and Windows and with thanks to Roman and Michael it is available as a free download from their website http://n.ethz.ch/student/mmoeller/fms/index_e.html.

The FMS simulator provides control of planes and helicopters either by keyboard, joystick, parallel port or the serial port. The keyboard and joystick options are fine but if you wish to practice flying planes with your own radio control it is necessary to use an adapter which converts the 'buddy' signal from the radio into data to interface to the computer. More about this further on.

A brief history

In the 1960's two engineers, Doug Spreng and Don Mathers of NASA, developed the original digital proportional control system and a standard was set for the type of pulses required to control servos in radio control applications. With the Mathers and Spreng pulse position modulation system (or PPM) pulses are transmitted — five pulses for a 4-channel, six for a 5-channel and so on. It is the space between these pulses which determines the position for the servos. These pulses are modulated such that the position between them is allowed to vary between 1 ms and 2 ms. For a 1.5-ms delay between pulses the servo would remain at centre, while anything less than 1.5 ms would force the servo to turn anti-clockwise and anything above would make it turn clockwise. The greater the time delay positive or negative from 1.5 ms the further the rotation of the servo. When the encoder is used to interface to your radio control, it is the time delay between these pulses which is converted into numerical form to interface to the computer.

There are a number of different ways to obtain the PPM signals from your R/C radio and this will depend on which radio you have. More about this later.

So how does the encoder work?

The encoder can work 'stand alone', without any need for a radio control. In this mode you simply plug it into the serial port of your computer and use both joysticks as you would a normal R/C.

In the second mode of operation you will need the PPM signal from your radio. Once you have connected this signal you only need to press down on the joystick (actuating the internal switch) to toggle the mode so that the PPM signal from your radio control is used to interface to the simulator. In this mode you use the encoder to decode the PPM signals

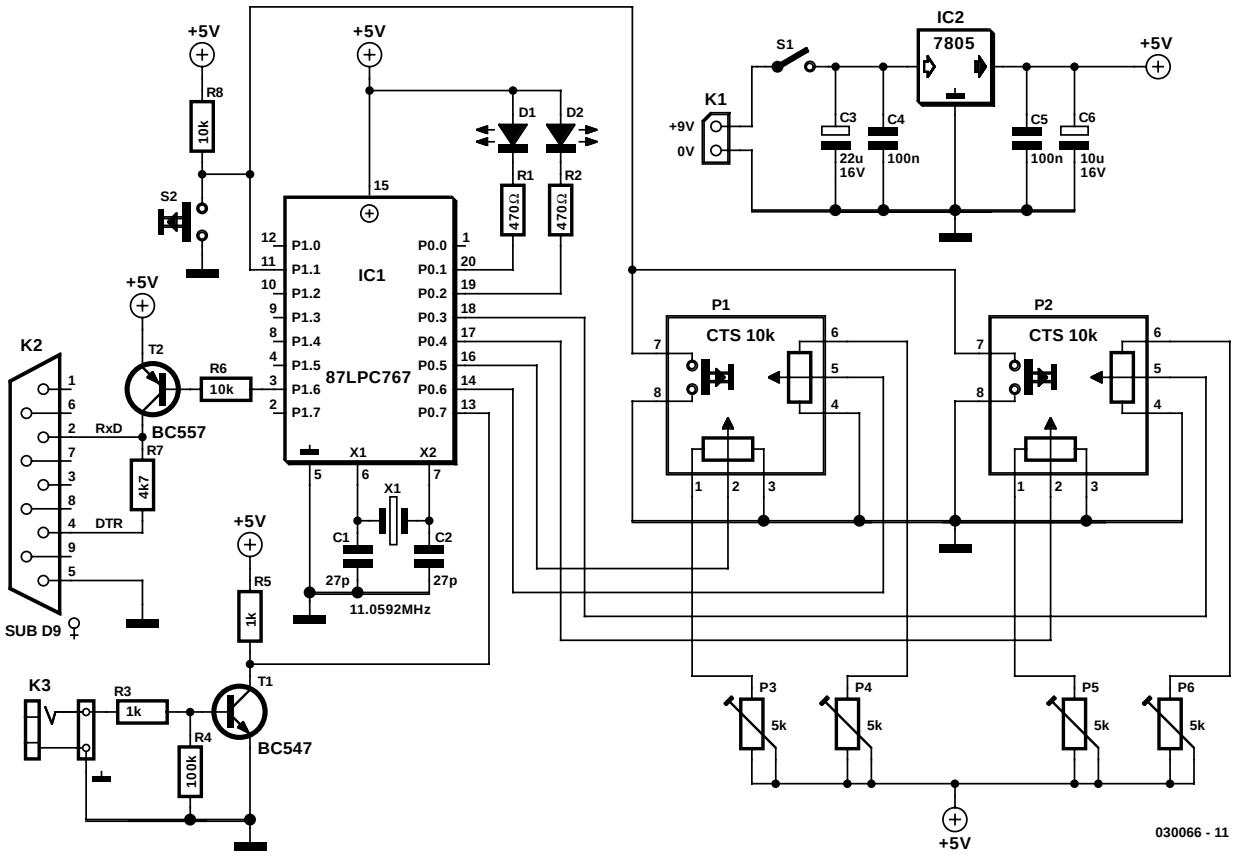


Figure 1. Circuit diagram of the FMS Encoder. All intelligence is vested in microcontroller IC1. Connector K3 allows you to connect your own R/C transmitter unit via its 'buddy/trainer' outlet.

from the radio and convert them into serial data for the computer. You are now using your own radio control to fly the virtual models, so with practice you gain confidence flying with your own R/C.

Within FMS options for 'Control' you will find the 'Serial PIC-interface'. You can choose from two options, but it is the 19200 Baud / 0xFF Sync which this design is set to run with. Make sure to select the correct COM port and set it up in the following way:

– 19200 baud
– 8 bit data, 1 stop bit, no parity
– RTS set to HI
– DTR set to LOW

With the FMS encoder plugged in and the correct COM port set up and calibrated you are ready to start flying.

The encoder sends one packet for all channels. Every packet starts with 0xFF and a byte is added with a value between 0 and 254 (0x00 to 0xFE hex) which provides the information for the servo position. Since 0 to 254 values are available for a range of 1 ms to 2 ms, the value which equates to 1.5 ms or centre position is 127 (0x7F hex).

So for a 4-channel R/C with all servos in the neutral position the data being sent to the computer would look like this:

Pause *0xFF 0x7F 0x7F 0x7F 0x7F* **Pause** *0xFF 0x7F 0x7F 0x7F 0x7F* **Pause** etc.

The pause is actually the pause between PPM signals from the radio. This value would vary depending on how many channels your R/C generates — the more channels, the shorter the pause.

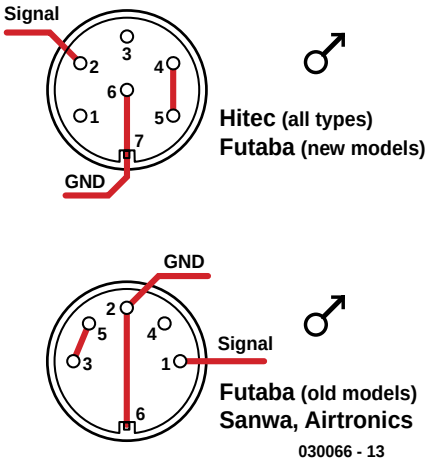


Figure 3. Signal functions on the buddy/trainer socket of some commonly used R/C transmitters. If yours is not shown, consult the owners manual or ask help from fellow FMS users via the FMS Forum.

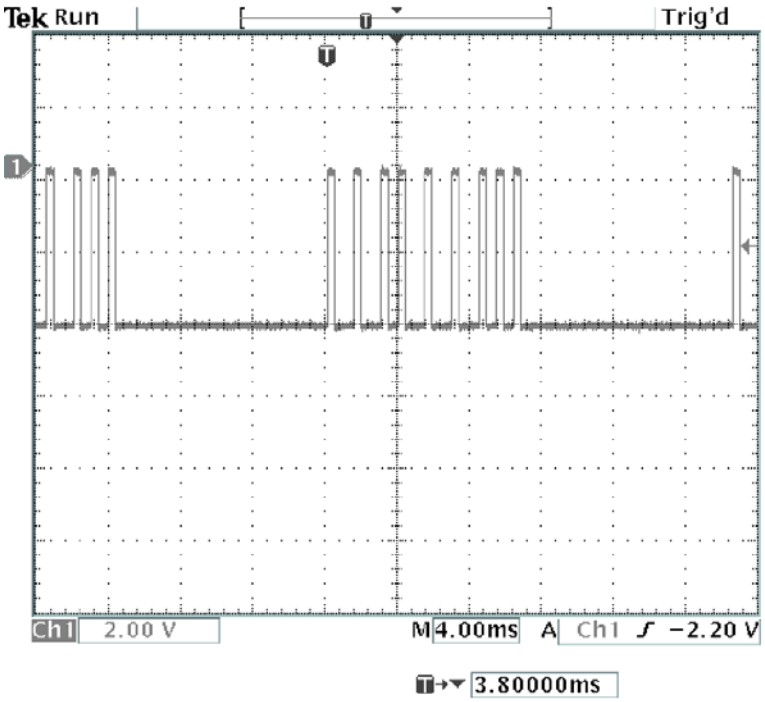


Figure 2. Pulses captured from an 8-channel R/C.

The circuit

The circuit diagram of the FMS Flight Simulator Encoder appears in **Figure 1**. You can see that it is basically the Philips low pin count P87LPC767 MCU, that's doing all the work. The LPC767 is an 8-bit 8051 based MCU with one-time programmable (OTP) PROM and four (!) channels of 8-bit analogue to digital converters. A programmer for the LPC76x family was described in *Elektor Electronics* December 2003.

At power-up the encoder is in **SIM mode**, reading the joystick positions from the four 10-kΩ linear potentiometers contained in joysticks P1, P2, then converting this analogue information into packets of data to send to the serial port on the PC. To provide a means of trimming the controls, 5-kΩ presets (P3-P6) have been added in series with the joysticks pots.

To engage the **R/C mode** (which allows you to use your own R/C transmitter) you simply press down on either of the two joysticks to activate an internal tactile switch. If the joysticks are not fitted, the same is achieved by pressing S2. Pulling the P1.1 port line to ground causes the MCU firmware to change to the R/C encoder mode.

In this mode pulses continuously sent to the MCU are converted and conveyed to the computer. The MCU waits for a pause before it samples the time delay between consecutive pulses. After the MCU detects the first pulse, it counts up in steps of 10 microseconds until it detects the next pulse. Thus for a delay of 1.5 milliseconds between pulses, the value it sends to the serial port is approximately 127 decimal or 0x7F. Minor variations are not so critical since the built-in driver for FMS takes care of any calibration differences.

If the MCU fails to detect any more pulses it simply bails out and sends only those channels which it could count, this may be anything from a 2-channel to an 8-channel R/C pulse train.

Figure 2 shows the pulses from an 8-channel R/C. You can see that there are actually nine pulses, and that there is a delay of around 10 ms between pulse trains.

It is the inverted version of these pulses which are sent to buffer transistor T1. These pulses are inverted

COMPONENTS LIST

Resistors:
R1,R2 = 470Ω
R3,R5 = 1kΩ
R4 = 100kΩ
R6,R8 = 10kΩ
R7 = 4kΩ7
P1,P2 = mini joystick, CTS model 25A104A60TB (order code CTS25A from www.dil.nl)*
P3-P6 = 5kΩ preset

Capacitors:
C1,C2 = 27pF
C3 = 22µF 16V axial
C4,C5 = 100nF
C6 = 10µF 16V axial

Semiconductors:
D1 = LED, green, 3mm
D2 = LED, red, 3mm
IC1 = 87LPC767BN, programmed, order code **030066-41**
IC2 = 7805CP
T1 = BC547B
T2 = BC557B

Miscellaneous:
K1 = 9-V battery with clip-on leads
K2 = 9-way sub-D socket (female), PCB mount
K3 = 3.5mm mono jack socket, chassis mount
S1 = on/off switch
S2 = pushbutton, 1 make contact
X1 = 11.0952MHz quartz crystal
PCB, order code **030066-1**
3 wire links
RS-232 cable (non-crossed)
Case, e.g., Pactec WM-46 (Conrad Electronics # 54 13 03)

* Worldwide distributor information from www.ctscorp.com

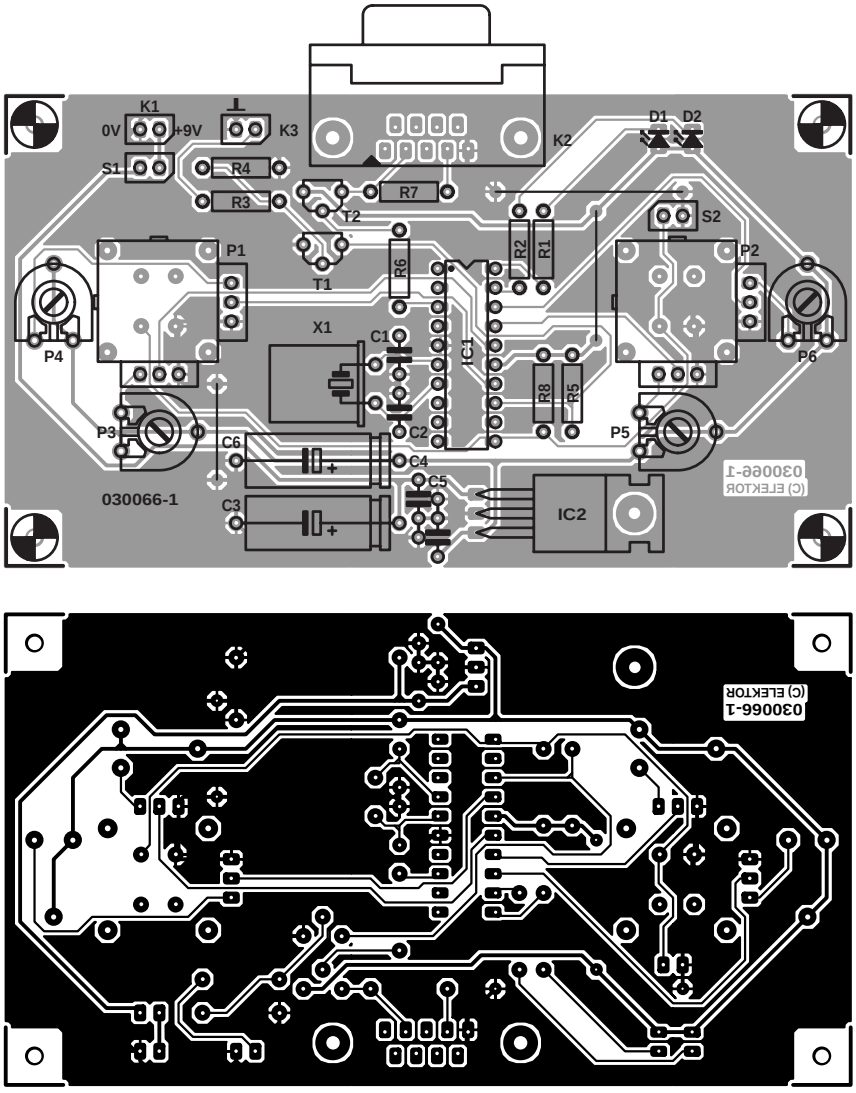


Figure 4. Copper track layout and component mounting plan of the PCB designed for the FMS Encoder (board available ready-made).

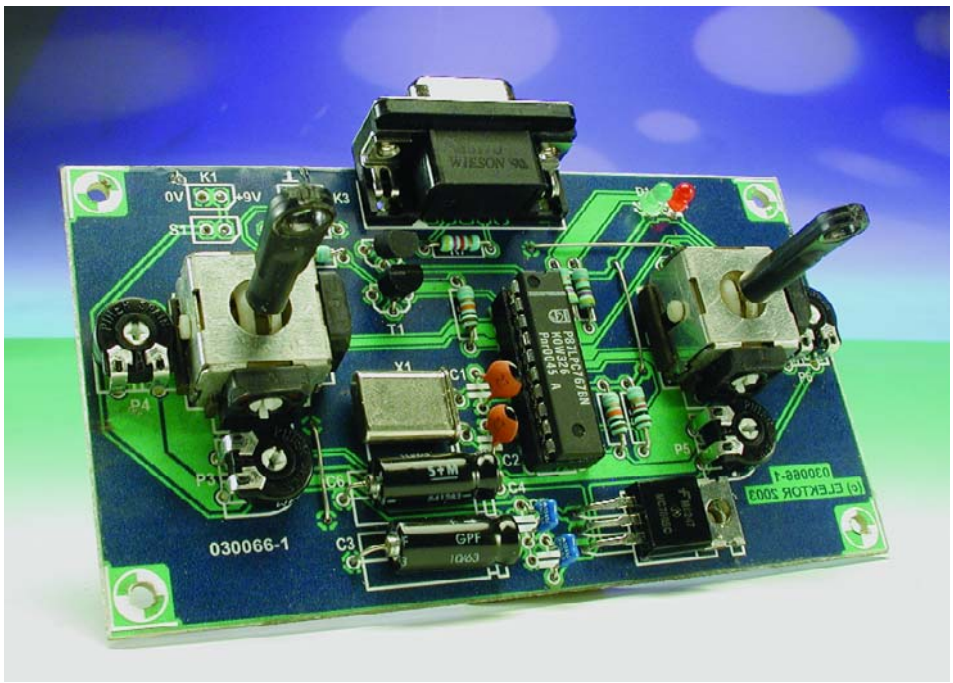


Figure 5. Our finished and fully working prototype.

by T1 thus bringing them back to normal before they are sent to MCU pin13.

Output data from MCU port line P1.6 (pin 3) is buffered by T2, with R7 providing the required level shifting to drive the RS232 port.

Only two wires are required from your R/C radio to interface to the

FMS encoder. Where these connections are will depend on what type of R/C you have. A few are listed in **Figure 3**. You may have more information available from your radio control manual. If you are unsure which connections to use then don't be afraid to experiment since the buffered input is quite safe. If you fail to place them the right way around the encoder will behave erratically — just swap them over and everything should be okay.

The whole circuit runs off a single 9-V (6F22) battery and draws only 21 mA, most of which is consumed by the LEDs D1 and D2 which indicate SIM mode and R/C mode respectively. A 7805 regulates the battery supply down to 5 V to power the MCU. At least five hours play time should be available from a standard 9-V battery.

Construction

Construction of the FMS Flight Simulator Encoder board should be easy if you use a ready-made printed circuit board and a programmed MCU supplied through the *Elektor Electronics* Readers Services. The respective order codes are **030066-11** and **030066-41**. The MCU for this project is only available ready-programmed. The author receives royalty payments for every copy sold hence no hex code or source code files can be supplied.

The PCB (**Figure 4**) is spacious and single-sided. As usual, polarized components like LEDs, electrolytic capacitors, transistors and integrated circuits should mounted the right way around on penalty of creating a hard to find circuit fault (best case) or destroying them when the power is applied (worst case). Don't swap T1 and T2, the devices looking identical but being complementary (npn and pnp respectively).

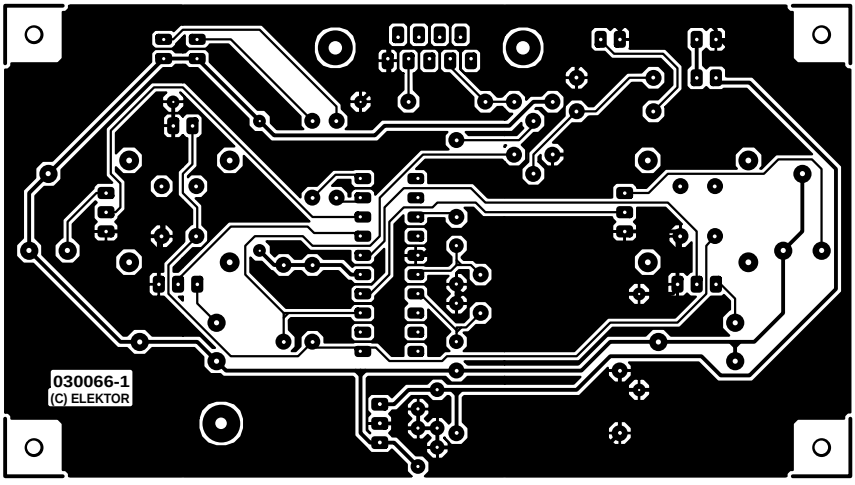
The MCU being the most expensive component, it really deserves to be fitted in a good quality IC socket. However, it should not be inserted in the socket before you have verified and approved the presence of the +5 volt supply voltage at a number of relevant points in the circuit, for example, IC1 socket pin 15, the emitter of T2 and the anode wires of LEDs D1 and D2. Okay so far? Then switch off and insert the MCU, observing the position of the notch on the PCB overlay.

Pinheader strips beside joysticks P1 and P2 are provided to connect joysticks that do not fit the board, or an old R/C of which the RF sections are faulty or even missing.

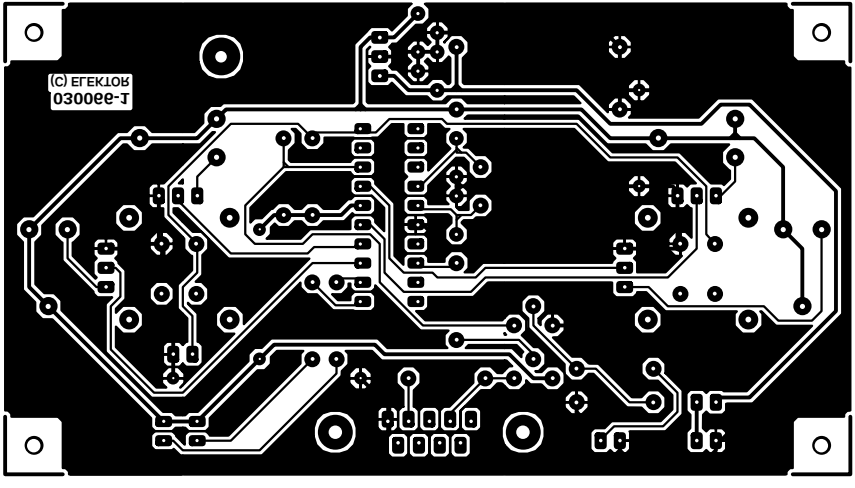
Finally

Users of the Encoder described in this article are encouraged to communicate with other FMS users via the Forum which can be accessed via the FMS homepage. Happy flying!

(030066—1)



non reflected



reflected