RoboNova PS2 Wireless Controller

Richard Ibbotson. December 2007

The currently available remote controllers from Hitec for the RoboNova are either the Infra-red Remote, Bluetooth, or a connected radio control receiver. The promised RoboNova wireless PS2 Controller seemed to make a short appearance in Japan, but has disappeared from the rest of the world.

The IR controller some has limitations due to directionality, but also due to poor integration to the RoboBasic code making it slow and erratic in behaviour. The radio receiver suffers from the same problems. The most common alternative is to use a Bluetooth receiver connected to the ETX/ERX pins. A remote Bluetooth device such as a PC, Pocket PC or phone, can be used to send and receive commands and data with the RoboNova. This allows a RoboBasic program to use more buttons than the standard remote, gives more reliable behaviour, and has the ability to have text labelled buttons.

I have used the Bluetooth control with a BlueSmirf and the excellent software for the Pocket PC made available by Pev. The major limitations I have found with the Bluetooth are poor response for real time control of the RoboNova, and the requirement of a PC to relay joystick controllers to the Bluetooth. I wanted to have better real time control of the RoboNova, not just select move routines. I therefore decided to embark on my own PS2 Wireless Game Controller interface for the RoboNova.

The PS2 Controller with 16 buttons and 2 joysticks is an ideal controller for the RoboNova. Wireless PS2 controllers are available at low cost and the interface is well described. More recent USB interface controllers are more difficult to interface and Bluetooth controllers use a HID interface which does not appear to be commonly supported on Bluetooth radios.

The following describes my design and build of the PS2 wireless controller. I have not given complete build instructions, but there should be enough information if anyone else wants to try this. I can’t promise this description is without error, so please take time and care in construction, based on your chosen controller and physical build.

1) For the design of the PS2 controller interface there is quite a lot of information and code already published on the Web. There are 2.4Ghz wireless controllers from Madcatz, Logitech, Lynxmotion, and some unbranded ones too. I have found the Madcatz to be best value in terms of quality to cost. I didn’t try the Logitech yet. Controllers are available in normal size and small size.

The receiver is a small box, which either plugs direct to the PS2 or via a very short cable.

2) **Interface to the RoboNova** The RoboNova has limited I/O ports available. I choose to use the ETX/ERX serial port because:
    a. That is what Hitec use for theirs
    b. It is buffered for one character
    c. Easy to access from RoboBasic
    d. No firmware modification to RoboNova required

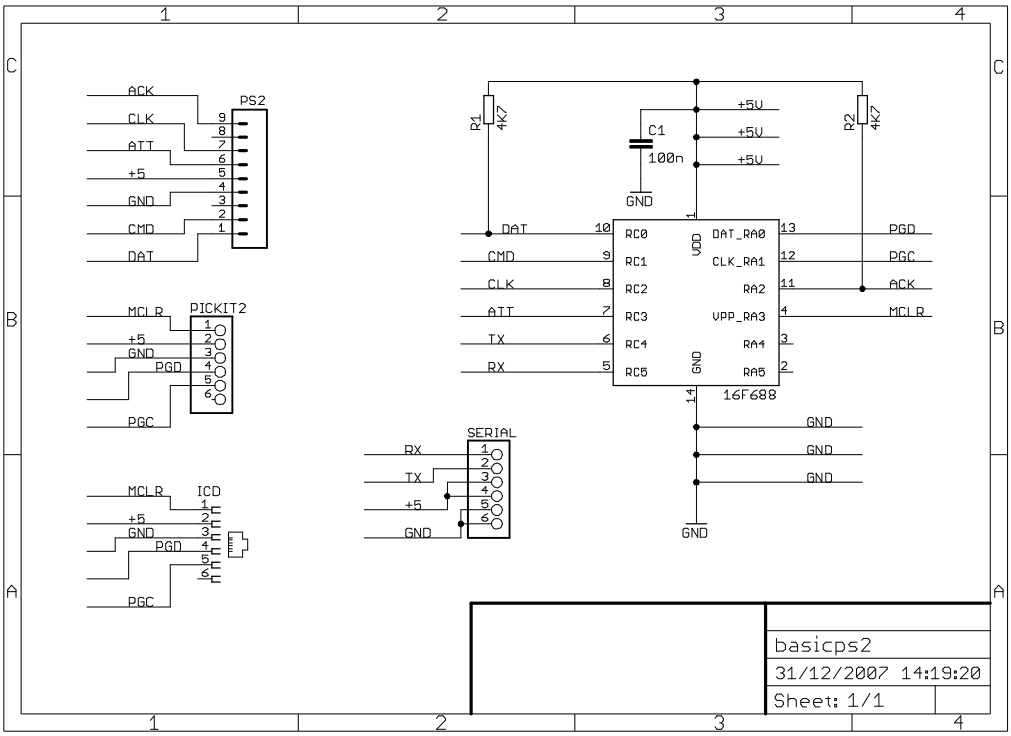
    It did mean that I lost the existing Bluetooth interface

3) **Formatting of data** RoboBasic code is interpreted at run time in the Robonova, and is fairly slow at about one instruction per millisecond. By formatting the data to be in a form easily used by RoboBasic, significant increase in speed and RoboBasic code simplicity is possible.

I chose to format the buttons to make a single byte which can be read and used in an 'ON' statement for move routine selection, and also to convert the 0 to 255 range of the joysticks and button pressures to more usable values. Changes can be made to the code for any format of output data.

4) **Serial / PS2 converter** I chose to use a Microchip 16F688 processor for the converter because:
   a. Small 14 pin package
   b. Internal Oscillator
   c. UART

   This enabled the interface to be built with a minimum number of components



5) **Building the converter** I choose to build the converter with a DIP PIC 16F688 on a scrap of prototype board. The board was only slightly larger than the DIP chip to include the capacitor and resistors, no IC socket was used. The wires for the cables went direct to the IC pins. I used a 3 pin servo connector to give access to the programming pins in the PIC. Then put it in heat shrink sleeving.

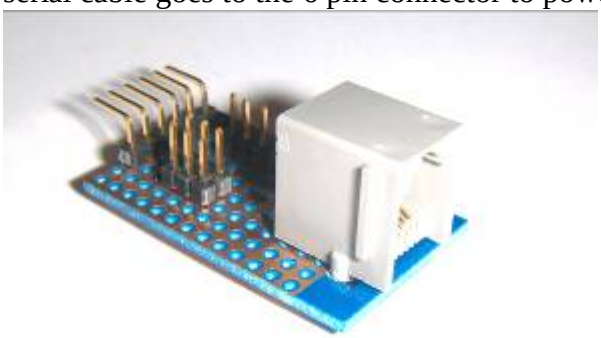
   It would be better to use an SMD PIC, and a small PCB, but I did not have time to do this.

6) Loading the converter software The software for the converter is written in assembler using the Microchip MPLAB software.

Here is the Code for the PIC
http://robosavvy.com/Builders/i-Bot/PS2.zip

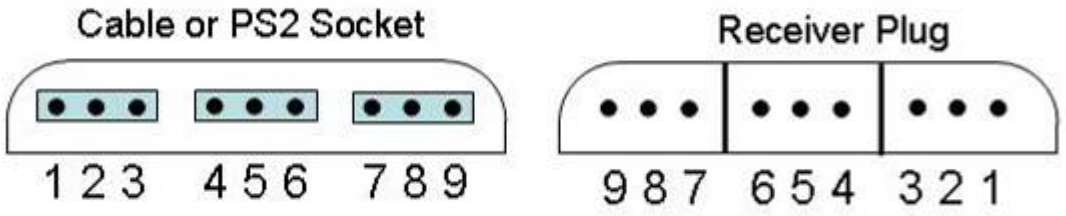
Ps2.asm

Code is down loaded to the converter using an ICD2 debugger in program mode or a PICKit 2.
I made an adapter to program the converter. The additional 3 pins on the pink connector in the photo are used with an adapter cable to the programmer. The serial cable goes to the 6 pin connector to power the converter.



7) Connecting to the PS2 Receiver. I removed the connector from the receiver and wired direct to the board to save space. It is also possible to get a cable from Lynxmotion or to cut down an extension cable. Alternatively a socket may be used, these are hard to get but could be cut from and old Multitap. The Logitech receiver is smaller, so this might fit.

The photo above shows the connector on the receiver on the right, and the socket it plugs into on the left. There are nine pins to the connector, but only seven are used.



Pins are:
Pin 1 = DAT
Pin 2 = CMD
Pin 3 = unused
Pin 4 = GND
Pin 5 = +5V
Pin 6 = ATT
Pin 7 = CLK
Pin 8 = unused
Pin 9 = ACK (not actually used in current software)

8) **Connecting to RoboNova**. The converter connects to the C3024 ERX/ETX pins to also pick up 5V power for the converter and receiver, using a 6 pin connector or two 3 pin connectors. The RX pin of the converter connects to the C3024 ERX pin, and the TX pin of the converter connects to the C3024 ETX pin.

9) **Mounting** The receiver is a bit big, so it needed to go either on the back or front. I chose the front, and put the PS2/serial interface in the chest.

10) **Interface to RoboBasic**. The serial ETX/ERX has buffering for only a single character. I chose to use a command and reply serial interface, where RoboBasic sends a single character command and the converter performs the PS2 interface, and always responds with a character back to RoboBasic. The interface runs at 19200, which seems to fit fine with the general timing required for RoboBasic, and for fast enough real time control.

Command Table:

Character Sent	Function	Character returned
A	Read Buttons 1	0xFF no buttons pressed, bits low for buttons pressed. Bit 7 = left button Bit 6 = down button Bit 5 = right button Bit 4 = up button Bit 3 = start button Bit 2 = right joystick button Bit 1 = left joystick button Bit 0 = select button
B	Read Buttons 2	0xFF no buttons pressed, bits low for buttons pressed. Bit 7 = square button Bit 6 = cross button Bit 5 = circle button Bit 4 = triangle button Bit 3 = R1 button Bit 2 = L1 button Bit 1 = R2 button Bit 0 = L2 button
C	Read Stick Right X	Value between 10 and 190 in analog or DS2 mode. (190 in digital)
D	Read Stick Right Y	Value between 10 and 190 in analog or DS2 mode. (190 in digital)
E	Read Stick Left X	Value between 10 and 190 in analog or DS2 mode. (190 in digital)

F	Read Stick Left Y	Value between 10 and 190 in analog or DS2 mode. (190 in digital)
G	Read Pressure right	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
H	Read Pressure left	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
I	Read Pressure up	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
J	Read Pressure down	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
K	Read Pressure triangle	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
L	Read Pressure circle	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
M	Read Pressure cross	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
N	Read Pressure square	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
O	Read Pressure L1	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
P	Read Pressure R1	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
Q	Read Pressure L2	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
R	Read Pressure R2	Pressure value between 0 and 15 in DS2 mode (0 in analog or digital)
W	Set and Lock DS2	Mode should be 0x79
X	Read encoded buttons	In priority order 0 = no button presses 1 = left button 2 = down button 3 = right button 4 = up button 5 = start button 6 = right joystick button 7 = left joystick button 8 = select button 9 = square button 10 = cross button 11 = circle button 12 = triangle button 13 = R1 button 14 = L1 button 15 = R2 button 16 = L2 button
Y	Set and Lock Analogue	Mode should be 0x73
Z	Set and Lock Digital	Mode should be 0x41
c	Read Stick Right X	Value between 33 and 55 in analog or DS2 mode. (44 in digital)

d	Read Stick Right Y	Value between 33 and 55 in analog or DS2 mode. (44 in digital)
e	Read Stick Left X	Value between 33 and 55 in analog or DS2 mode. (44 in digital)
f	Read Stick Left Y	Value between 33 and 55 in analog or DS2 mode. (44 in digital)

Sample Code

```
' Template program for PS2 wireless controller
' Richard Ibbotson November 2007
' PS2 wireless controller in DS2 mode connected to ETX/ERX

DIM rr AS BYTE                ' first variable is reserved
DIM txchar AS BYTE            ' store for character to send to
controller
DIM rxchar AS BYTE

'Empty the ERX buffer
begin:
ERX 19200, rxchar, bemt1
bemt1:
ERX 19200, rxchar, bemt2
bemt2:

' Set the PS2 controller to DS2 mode
txchar = "W"
ETX 19200, txchar
GETW:
ERX 19200, rxchar, GETW

'-------------------------------------------------------------------
---
' Motion Startup Stuff, note this is special for double knee joints
PTP SETON
PTP ALLON


DIR G6A,1,0,0,0,1,0
DIR G6B,1,1,1,1,1,1
DIR G6C,0,0,0,0,0,0
DIR G6D,0,1,1,1,0,1

'GETMOTORSET G6A,1,1,1,1,1,1
'GETMOTORSET G6B,1,1,1,0,0,0
'GETMOTORSET G6C,1,1,1,0,0,1
'GETMOTORSET G6D,1,1,1,1,1,1

SPEED 5

MOTOR G6A
MOTOR G6B
MOTOR G6C
MOTOR G6D

'Example Gyro Setup Code
GYROSET G6A,0,0,0,0,0,0
GYROSET G6D,0,0,0,0,0,0
```

```
GYRODIR G6A,0,1,1,0,1,1
GYRODIR G6D,0,1,1,0,1,1

GYROSENSE G6A,0,200,200,0,0,0
GYROSENSE G6D,0,200,200,0,0,0

GOSUB Standard_pose
'-------------------------------------------------------------------
---


MUSIC "CDE"

MAIN:
'Read the PS2 buttons
txchar ="X"
ETX 19200, txchar
GETX:
ERX 19200, rxchar, GETX

IF rxchar > 0 AND rxchar < 17 THEN
ON rxchar GOTO MAIN,K1,K2,K3,K4,K5,K6, K7, K8, K9, K10, K11, K12,
K13, K14, K15, K16
ENDIF
GOTO MAIN

K1: ' Left Button

      GOTO MAIN

K2:   ' Down Button

      GOTO MAIN

K3:   ' Right Button

      GOTO MAIN

K4:   ' Up Button

      GOTO MAIN

K5:   ' Start Button    Use to force restart

      GOTO begin

K6:   ' Right Joystick Button
 'read the Right joystick position
txchar ="C"
ETX 19200, txchar
GETC:
ERX 19200, rxchar, GETC
SERVO 17,rxchar


      GOTO MAIN

K7: ' Left Joystick Button

      GOTO MAIN
```

```
K8:   ' Select Button

      GOTO MAIN

K9:   ' Square Button

      GOTO MAIN

K10:' Cross Button

      GOTO MAIN

K11:' Circle Button

      GOTO MAIN

K12:' Triangle Button

      GOTO MAIN

K13:' R1 Button
' read the R1 button pressure
txchar ="P"
ETX 19200, txchar
GETP:
ERX 19200, rxchar, GETP
SPEED rxchar
      MOVE G6B,100,70,80,100,100,
      WAIT
SPEED 5
GOSUB Standard_Pose

      GOTO MAIN

K14:' L1 Button

      GOTO MAIN

K15:' R2 Button

      GOTO MAIN

K16:' L2 Button

      GOTO MAIN

'================================================
Standard_Pose:
      MOVE G6A, 100,85,90,90,90,100
      MOVE G6D, 100,85,90,90,90,100
      MOVE G6B, 100,30,80,100,100,100
      MOVE G6C, 100,30,80,100,100,100
      WAIT
      RETURN
'================================================
```