

```

1 ' -----[ Title ]-----
2 ' PS2Control - Boe-BotVirtualWorldControl.bs2
3 ' Modification of Aaron Klapheck's program: RoboticHandAndMovementsUsingPs2WithPing.bs2
4 ' Modified on: 27-Aug-08
5 '
6 ' File..... PSX_Demo.BS2
7 ' Purpose... PlayStation Controller Interface
8 ' Author.... Jon Williams
9 ' E-mail.... jwilliams@parallax.com
10 ' Started...
11 ' Updated... 17 JUL 2003
12 '
13 '

```

```

14 ' {$STAMP BS2}
15 ' {$PBASIC 2.5}
16 '

```

```

17 ' -----[ Program Description ]-----
18 '

```

```

19 ' This program demonstrates the essential interface between the BASIC
20 ' Stamp and a Sony PlayStation (or compatible) game controller. This
21 ' code assumes that the clock signal is inverted between the Stamp and
22 ' the controller to allow simpler [less sophisticated] interface with
23 ' SHIFTOUT and SHIFTIN.
24 '

```

```

25 ' Note: The interface and portions of code are based on previous work by
26 ' Aaron Dahlen.
27 '

```

```

28 ' -----[ Aaron's Add-On's ]-----
29 '

```

```

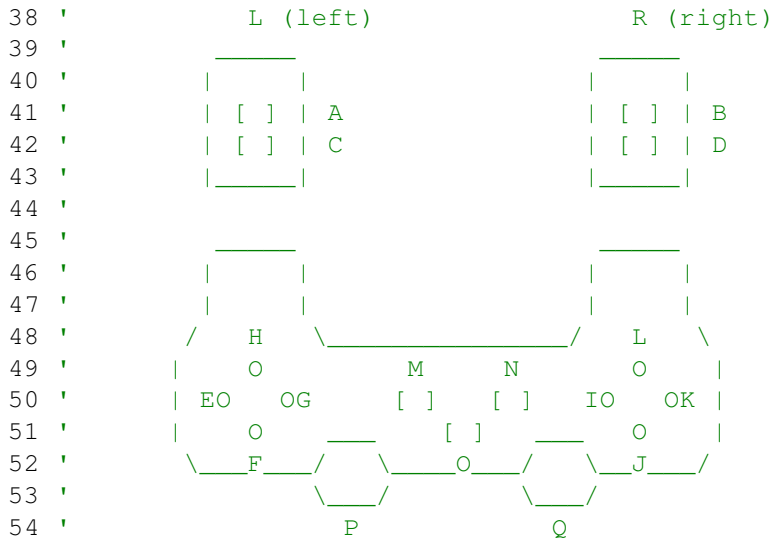
30 ' This section of code is written by Aaron Klapheck.
31 ' 6 AUG 2008
32 ' Below is the diagram of a Sony PlayStation 2 remote control.
33 ' The numbers below are specific to Sony brand controlers.
34 '
35 '

```

```

36 ' Diagram 1.
37 '

```



```

55 '
56 '
57 '
58 '
59 ' Button Numbers Chart: (see explanation below)
60 '
61 ' **Replace the BIN8 and DEC numbers with the values you found
62 '   when you ran DisplayValuesOfButtonsAndJoysticks.bs2**
63 '
64 ' psxThumbR:
65 ' A: (L 2),           BIN8 = 11111110, DEC = 254
66 ' B: (R 2),           BIN8 = 11111101, DEC = 253

```

```

67 ' C: (L 1),          BIN8 = 11111011, DEC = 251
68 ' D: (R 1),          BIN8 = 11110111, DEC = 247
69 '
70 ' psxThumbL
71 ' E: (L arrow),      BIN8 = 01111111, DEC = 127
72 ' F: (down arrow),  BIN8 = 10111111, DEC = 191
73 ' G: (R arrow),      BIN8 = 11011111, DEC = 223
74 ' H: (up arrow),     BIN8 = 11101111, DEC = 239
75 '
76 ' psxThumbR
77 ' I: (Square),       BIN8 = 01111111, DEC = 127
78 ' J: (X),            BIN8 = 10111111, DEC = 191
79 ' K: (Circle),       BIN8 = 11011111, DEC = 223
80 ' L: (Triangle),     BIN8 = 11101111, DEC = 239
81 '
82 ' psxThumbL
83 ' M: (Select),       BIN8 = 11111110, DEC = 254
84 ' N: (Start),        BIN8 = 11110111, DEC = 247
85 ' O: (Analog),       Press to activate joystic controles
86 '
87 ' No buttons pressed:
88 ' psxThumbL,         BIN8 = 11111111, DEC = 255
89 ' psxThumbR,         BIN8 = 11111111, DEC = 255
90 '
91 '

```

```

92 ' Below is the numbering schemae for the two joysticks (thy both use the same numbering scheme).
93 ' All numbers shown below are in decimal format.

```

```

94 '
95 '
96 '
97 '
98 ' Q: (psxJoyR),
99 ' P: (psxJoyL),      (000, 127) <-- (127, 127) --> (255, 127)
100 '
101 '
102 '
103 '
104 '
105 '
106 '

```

107 ' Explanation of Button numbers:

```

108 '   For each button pressed there is a specific number stored in either
109 '   the psxThumbR or psxThumbL variable depending on which side of the controller
110 '   the button is on.
111 '   The button values stored in the psxThumbR or psxThumbL variables are displayed
112 '   to the right of the button label in both binomial and decimal format.
113 '   When only one button is being pressed at a time the decimal format will work
114 '   fine. But if you want to be able to press multiple buttons at a time then the
115 '   binomial format may be easier to use.
116 '   Although the numbering scheme for all playstation remotes should be the same double check!

```

120 ' How to use the Button Numbers Chart:

```

122 '   What is shown above in "Button Numbers Chart":

```

```

123 '   psxThumbR:
124 '   A: (L 2),          BIN8 = 11111110, DEC = 254
125 '   .
126 '   .
127 '   .

```

129 ' What this means:

```

130 '   psxThumbR. Means: The variable used to store information
131 '   A. Means: the arbitrary designation of the button in question (See Diagram 1)
132 '   (L 2). Means: the more discriptive name that I refer to when typing the code

```

```

133 '     BIN8 = 11111110. Means: the binomial number that is stored in the variable listed
134 '     DEC = 254. Means: the decimal number that is stored in the variable listed
135 '
136 '
137 ' -----[ MATLAB transfer ]-----
138 '
139 ' In digital mode
140 '     All button values will be sent to MATLAB. Only one button value at a
141 '     time will be sent to MATLAB (only press one button at a time). However,
142 '     two buttons can be pressed if one is stored in psxThumbR and the other
143 '     is stored in psxThumbL.
144 '
145 ' In analog mode
146 '     Don't use this.
147 '
148 '
149 ' -----[ Variables/Constants/Pins ]-----
150 '
151 '
152 FreqDetectable    CON    3000
153
154 Piezospeaker      PIN     4           ' Speaker
155
156
157 ' Stuff for MATLAB communication
158 sPin              CON    16           'Serial Pin - P16, Programming port
159 Baud              CON    84           'Baud mode for a rate of 9600, 8-N-1
160                                     'BS2P, BS2SX use 240 for 9600, 8-N-1
161
162 WheelRight        PIN    12           ' Wheels left/right.
163 WheelLeft         PIN    13
164
165
166 ' For PS2 controller
167 ' -----[ I/O Definitions ]-----
168 '
169 PsxClk            PIN     9           ' PSX joystick interface
170 PsxAtt            PIN     8
171 PsxCmd            PIN     7
172 PsxDat            PIN     6
173
174 ' -----[ Constants ]-----
175 '
176 Inverted          CON     1           ' inverted clock signal
177 Direct            CON     0           ' no inverter in clock line
178 ClockMode         CON    Inverted
179
180 ' -----[ Variables ]-----
181 '
182 idx              VAR    Nib          ' loop counter
183 psxOut           VAR    Byte         ' byte to controller
184 psxIn            VAR    Byte         ' byte from controller
185
186 counter          VAR    Word
187
188 ' joystick packet
189 psxID            VAR    Byte         ' controller ID
190 psxThumbL        VAR    Byte         ' left thumb buttons
191 psxThumbR        VAR    Byte         ' right thumb buttons
192 psxStatus        VAR    Byte         ' status ($5A)
193 psxJoyRX         VAR    Byte         ' r joystick - X axis
194 psxJoyRY         VAR    Byte         ' r joystick - Y axis
195 psxJoyLX         VAR    Byte         ' l joystick - X axis
196 psxJoyLY         VAR    Byte         ' l joystick - Y axis
197
198

```

```

199 ' -----[ Initialization ]-----
200
201 FREQOUT Piezospeaker, 2000, FreqDetectable      'Signal program start/reset.
202
203 SEROUT sPin, Baud, [LF]                        'Send a lone LF to signal MATLAB start
204
205 SEROUT sPin, Baud, ["This is your Boe-Bot", LF]
206
207 Setup:
208 HIGH PsxAtt          ' deselect PSX controller
209 OUTPUT PsxCmd
210 PsxClk = ~ClockMode  ' release clock
211 OUTPUT PsxClk        ' make clock an output
212
213 ' -----[ Program Code ]-----
214
215 Main:
216 DO
217
218   GOSUB Get_PSX_Packet_Fast ' type and packet
219
220   IF (psxId <> $41) THEN ' If in analog mode (Don't use)
221
222     ' Only send information from the right joystic controler
223     ' SEROUT sPin, Baud, [DEC psxJoyRX, ",", DEC psxJoyRY, LF]
224
225   ELSE ' If in digital mode
226
227     ' All button values will be sent to MATLAB
228     SEROUT sPin, Baud, [BIN8 psxThumbL, ",", BIN8 psxThumbR, LF]
229
230
231     ' Left 4 button pad
232     ' *** You may need to change the values to match your controler ***
233     IF psxThumbL = 127 THEN ' (L arrow)
234       GOSUB TurnLeft
235     ELSEIF psxThumbL = 191 THEN ' (down arrow)
236       GOSUB Backward
237     ELSEIF psxThumbL = 223 THEN ' (R arrow)
238       GOSUB TurnRight
239     ELSEIF psxThumbL = 239 THEN ' (up arrow)
240       GOSUB Forward
241     ENDIF
242
243   ENDIF ' (psxId <> $41)
244 LOOP ' Main loop
245
246 END
247
248
249 ' -----[ PS2 Subroutines ]-----
250
251
252 ' This routine combines manual and built-in shifting
253 ' routines to get the best speed and all valid data.
254 '
255 ' Execution time on BS2 is ~40 ms.
256
257 Get_PSX_Packet_Fast:
258   LOW PsxAtt          ' select controller
259   SHIFTOUT PsxCmd, PsxClk, LSBFIRST, [$01] ' send "start"
260   psxOut = $42 : GOSUB PSX_TxRx          ' send "get data"
261   psxId = psxIn          ' save controller type
262   SHIFTTIN PsxDat, PsxClk, LSBPOST, [psxStatus] ' should be $5A ("ready")
263   SHIFTTIN PsxDat, PsxClk, LSBPOST, [psxThumbL]
264   SHIFTTIN PsxDat, PsxClk, LSBPOST, [psxThumbR]

```

```

265 SHIFTIN PsxDat, PsxClk, LSBPOST, [psxJoyRX]
266 SHIFTIN PsxDat, PsxClk, LSBPOST, [psxJoyRY]
267 SHIFTIN PsxDat, PsxClk, LSBPOST, [psxJoyLX]
268
269 SHIFTIN PsxDat, PsxClk, LSBPOST, [psxJoyLY] ' If you want to use the left joystick
270 ' then delete this part and uncomment
271 ' the gosub PSX_TxRx below.
272 'GOSUB PSX_TxRx : psxJoyLY = psxIn
273 HIGH PsxAtt ' deselect controller
274 RETURN
275
276
277
278
279 ' Transmit psxOut to, and receive psxIn from the PSX controller.
280 ' This allows a simultaneous data exchange between the
281 ' PS2 controller and the Basic Stamp.
282 ' See PSX_TxRx.bs2 to see how this works
283
284 PSX_TxRx:
285   FOR idx = 0 TO 7
286     PsxCmd = psxOut.LOWBIT(idx) ' setup command bit
287     PsxClk = ClockMode ' clock the bit
288     psxIn.LOWBIT(idx) = PsxDat ' get data bit
289     PsxClk = ~ClockMode ' release clock
290   NEXT
291 RETURN
292
293
294 ' -----[ Movement Subroutines ]-----
295
296 ' ----- Boe-Bot movements
297
298 Backward:
299   PULSOUT WheelRight, 780
300   PULSOUT WheelLeft, 720
301
302 RETURN
303
304
305 Forward:
306   PULSOUT WheelRight, 720
307   PULSOUT WheelLeft, 780
308
309 RETURN
310
311
312 TurnRight:
313   PULSOUT WheelRight, 780
314   PULSOUT WheelLeft, 780
315
316 RETURN
317
318
319 TurnLeft:
320   PULSOUT WheelRight, 720
321   PULSOUT WheelLeft, 720
322
323 RETURN
324

```