INDUSTRIAL AUTOMATION

R2 SERIES OPERATORS MANUAL

Appendix

Model R2-X2 eXperimental Model

Cover artwork by John Jongsma

Appendix A – Motor Specifications

Pittman GM9413L275 Dome Drive Motor:

Assembly Data	Symbol	Units	Value	
Reference Voltage	E	V	12	
No-Load Speed	S_{NL}	rpm (rad/s)	142	(14.9)
Continuous Torque (Max.)[1]	T_C	oz-in (N-m)	45	(3.2E-01)
Peak Torque (Stall)[2]	T_{PK}	oz-in (N-m)	109	(7.7E-01)
Weight	W_M	oz (g)	15.2	(432)
Motor Data				
Torque Constant	K_T	oz-in/A (N-m/A)	5.60	(3.95E-02)
Back-EMF Constant	K_E	V/krpm (V/rad/s)	4.14	(3.95E-02)
Resistance	R_T	Ω	8.33	
Inductance	L	mH	6.17	
No-Load Current	I_{NL}	A	0.10	
Peak Current (Stall)[2]	I_P	A	1.44	
Motor Constant	K_M	oz-in/√W (N-m/√W)	1.94	(1.37E-02)
Friction Torque	T_F	oz-in (N-m)	0.50	(3.5E-03)
Rotor Inertia	J_M	oz-in-s^2 (kg-m^2)	3.9E-04	(2.8E-06)
Electrical Time Constant	τ_E	ms	0.74	
Mechanical Time Constant	τ_M	ms	14.7	
Viscous Damping	D	oz-in/krpm (N-m-s)	0.011	(7.6E-07)
Damping Constant	K_D	oz-in/krpm (N-m-s)	2.8	(1.9E-04)
Maximum Winding Temperature	θ_{MAX}	°F (°C)	311	(155)
Thermal Impedance	R_{TH}	°F/watt (°C/watt)	66.4	(19.1)
Thermal Time Constant	τ_{TH}	min	11.1	
Gearbox Data				
Reduction Ratio			19.7	
Efficiency			0.73	
Maximum Allowable Torque		oz-in (N-m)	175	(1.24)

Pittman GM9236S020 Center Foot Lift/Lower Motor:

Assembly Data	Symbol	Units	Value	
Reference Voltage	E	V	24	
No-Load Speed	S_{NL}	rpm (rad/s)	236	(24.7)
Continuous Torque (Max.)[1]	T_C	oz-in (N-m)	153	(1.1E+00)
Peak Torque (Stall)[2]	T_{PK}	oz-in (N-m)	860	(6.1E+00)
Weight	W_M	oz (g)	20.3	(576)
Motor Data				
Torque Constant	K_T	oz-in/A (N-m/A)	6.49	(4.58E-02)
Back-EMF Constant	K_E	V/krpm (V/rad/s)	4.80	(4.58E-02)
Resistance	R_T	Ω	2.49	
Inductance	L	mH	2.63	
No-Load Current	I_{NL}	A	0.16	
Peak Current (Stall)[2]	I_P	A	9.64	
Motor Constant	K_M	oz-in/√W (N-m/√W)	4.11	(2.90E-02)
Friction Torque	T_F	oz-in (N-m)	0.80	(5.6E-03)
Rotor Inertia	J_M	oz-in-s^2 (kg-m^2)	1.0E-03	(7.1E-06)
Electrical Time Constant	τ_E	ms	1.06	
Mechanical Time Constant	τ_M	ms	8.5	
Viscous Damping	D	oz-in/krpm (N-m-s)	0.053	(3.5E-06)
Damping Constant	K_D	oz-in/krpm (N-m-s)	12.5	(8.5E-04)
Maximum Winding Temperature	θ_{MAX}	°F (°C)	311	(155)
Thermal Impedance	R_{TH}	°F/watt (°C/watt)	56.3	(13.5)
Thermal Time Constant	τ_{TH}	min	13.5	
Gearbox Data				
Reduction Ratio			19.7	
Efficiency[3]			0.84	
Maximum Allowable Torque		oz-in (N-m)	500	(3.53)

EV Warrior Motor:

This motor was originally used in the EV Warrior electric bicycle. The company went bankrupt and thousands of the motors were sold as surplus. It is probably a custom version of the Bosch GPB automotive fan motor. There are two versions available, one with the timing advanced 10° in the clockwise (CW) direction and one with timing advanced 10° in the counter-clockwise (CCW) direction. One of the motor's terminals is grounded to the case and this grounding tab must be removed or you may short out your speed controller through the frame.

12 Volt Specifications

Specifications for clockwise rotation:	Motor only performance.	Typical system performance.
Voltage:	12 V	
Angular-velocity constant:	229 rpm/V	
Torque constant:	5.91 ozf·in/A	
Terminal resistance:	0.121 ohm	
No-load current:	3.9 A	
Peak efficiency:	64.3 %	
Power source resistance:	0 ohm	0.121 ohm
Peak power:	0.368 hp	0.168 hp
No-load angular velocity:	2640 rpm	2530 rpm
Stall current:	99.2 A	49.6 A
Stall torque:	563 ozf·in	270 ozf·in

Specifications for counter-clockwise rotation:	Motor only performance.	Typical system performance.
Voltage:	12 V	
Angular-velocity constant:	189 rpm/V	
Torque constant:	7.16 ozf·in/A	
Terminal resistance:	0.121 ohm	
No-load current:	2.6 A	
Peak efficiency:	70.2 %	
Power source resistance:	0 ohm	0.121 ohm
Peak power:	0.378 hp	0.18 hp
No-load angular velocity:	2210 rpm	2150 rpm
Stall current:	99.2 A	49.6 A
Stall torque:	691 ozf·in	337 ozf·in

24 Volt Specifications

Specifications for clockwise rotation:	Motor only performance.	Typical system performance.
Voltage:	24 V	
Angular-velocity constant:	229 rpm/V	
Torque constant:	5.91 ozf·in/A	
Terminal resistance:	0.121 ohm	
No-load current:	3.9 A	
Peak efficiency:	73.9 %	
Power source resistance:	0 ohm	0.121 ohm
Peak power:	1.53 hp	0.736 hp
No-load angular velocity:	5380 rpm	5270 rpm
Stall current:	198 A	99.2 A
Stall torque:	1150 ozf·in	563 ozf·in

Specifications for counter-clockwise rotation:	Motor only performance.	Typical system performance.
Voltage:	24 V	
Angular-velocity constant:	189 rpm/V	
Torque constant:	7.16 ozf·in/A	
Terminal resistance:	0.121 ohm	
No-load current:	2.6 A	
Peak efficiency:	78.4 %	
Power source resistance:	0 ohm	0.121 ohm
Peak power:	1.55 hp	0.757 hp
No-load angular velocity:	4480 rpm	4420 rpm
Stall current:	198 A	99.2 A
Stall torque:	1400 ozf·in	691 ozf·in

Scooter Motors:

MODEL:	MY6812 by JX Motor Co., Ltd.
TYPE:	Brush
VOLTAGE:	24 Volt DC
RATED SPEED:	2300 RPM
RATED CURRENT:	6A
SPROCKET:	3M Belt
OUTPUT:	100 Watts

Appendix B – Motor Speed Controller Specifications

Dimension Engineering SyRen 25A:

25A continuous, 45A peak.
6-24V nominal, 30V absolute maximum
Synchronous regenerative drive
Ultra-sonic switching frequency
Thermal and overcurrent protection
Lithium protection mode
Input modes: Analog, R/C, simplified serial, packetized serial
Size: 2.4” x 2.3” x .8”
61 x 58 x 21 mm

Dimension Engineering Dual Sabertooth:

25A continuous, 50A peak per channel.
6-24V nominal, 30V absolute maximum
Synchronous regenerative drive
Ultra-sonic switching frequency
Thermal and overcurrent protection
Lithium protection mode
Input modes: Analog, R/C, simplified serial, packetized serial
Size: 2.6” x 3.2” x .8”
65 x 80 x 20 mm

Roboteq AX-3500B:

Operating Voltage	12V to 40V DC
Number of Channels	2
Max Current	
30s	60A
1min	60A
3 min	60A
1h	50A
Surge Current	>250A
MOSFETs per Channel	8
ON Resistance	5 mOhm
Synchronous Rectification	Yes - Allows regenerative braking
Current Limiting	By automatic power output reduction according to load and elapsed time
Temperature protection	Automated current limit reduction starting at 80o C (175o F) heat sink temperature
Voltage protection	Output shut off below 12V and above 43V
Power Wiring	0.25" Faston Tabs

Appendix C – Source Code

Micro MP3 & Picaxe Sound system:

NOTE: Code is downloadable from the Astromech.net site files section
NOTE: This code is for the Picaxe 18X
NOTE: You need to solder 10K resistors in the slots marked R7 & R8 to make inputs 0 & 1 digital.

```
setfreq m8
b0 = 1   ' b0 = the mode number
b1 = 1   ' b1 = the non random file number to play
b2 = 1   ' b2 = the random file number to play
b3 = 50 ' b3 = the volume setting (fairly low to start out) higher number is lower vol
                    '           254 = vol off,
                    '           0   = highest vol
b4 = 1   ' do while loop value for random

serout 4,n9600_8,("ST V 60", CR) ' set the initial sound volume

pause 1000

' Main loop (waiting -> goto waiting)
waiting:

        b8 = 0

        ' Pins 0, 1, 2, 6, 7 are the Picaxe input pins available for use.
        if pin7 = 1 then button_1' play a single sound
        if pin6 = 1 then button_2' play random sounds
        if pin2 = 1 then button_3' increase the volume
        if pin1 = 1 then button_4' decrese the volume
        if pin0 = 1 then button_5' change the mode

goto waiting

' button_1 plays a sound file in a series each time it is pressed.
button_1:

if b0 = 1 then play_mode_1
if b0 = 2 then play_mode_2
if b0 = 3 then play_mode_3
if b0 = 4 then play_mode_4
if b0 = 5 then play_mode_5
pause 250
goto waiting

' button_2 plays random sound files at random times from a series of sounds.
button_2:
b4 = 1
w5 = 0
w6 = 0
do

        random w4
```

```
        w3 = w4 // 11                           ' w3 is a random number between 0 -> 11


        ' b0 = the mode number
        if b0 = 1 then play_rnd_mode_1
        if b0 = 2 then play_rnd_mode_2
        if b0 = 3 then play_rnd_mode_3
        if b0 = 4 then play_rnd_mode_4
        if b0 = 5 then play_rnd_mode_5

        ContinueRandom:

        pause 250

        ' Wait a random amount of time before continuing.
        if w3 = 1 then
                w5 = 5000
        elseif w3 = 2 then
                w5 = 10000
        elseif w3 = 3 then
                w5 = 15000
        elseif w3 = 4 then
                w5 = 20000
        elseif w3 = 5 then
                w5 = 25000
        elseif w3 = 6 then
                w5 = 30000
        elseif w3 = 7 then
                w5 = 35000
        elseif w3 = 8 then
                w5 = 40000
        elseif w3 = 9 then
                w5 = 45000
        elseif w3 = 10 then
                w5 = 50000
        else
                w5 = 25000
        endif

        w6 = 0
        do
                ' exit the loop if any other button is pressed
                if pin7 = 1 or pin2 = 1 or pin1 = 1 or pin0 = 1 then
                        b4 = 0
                        exit
                endif

                w6 = w6 + 1

        loop while w6 < w5

loop while b4 = 1
pause 250
goto waiting
```

```
' button_3 increases the volume.
button_3:
if b3 > 2 then
        b3 = b3 - 2
endif

serout 4,n9600_8,("ST V ",#b3, CR) 'PC V --> stands for "P"lay "C"ommand "V"olume
pause 250
goto waiting

' button_4 decreases the volume.
button_4:
if b3 < 252 then
        b3 = b3 + 2
endif

serout 4,n9600_8,("ST V ",#b3, CR)
pause 250
goto waiting

' button_5 increments the mode number of the system by one up to 5 then starts over at 1.
button_5:
if b0 < 5 then
        b0 = b0 + 1
else
        b0 = 1
endif

' Repeat a sound the same number as the mode number so the use knows what mode the system is in.
serout 4,n9600_8,("PC F /RepeatSounds/RepeatSnd", #b0, "X.mp3", CR) 'PC F --> stands for "P"lay
"C"ommand "F"ile
pause 250
goto waiting

'=========================================

play_mode_1:
serout 4,n9600_8,("PC F /SingleSounds/Mod1Snd",#b1,".mp3", CR) ' play the sound indicated by the
variable b1
pause 250
if b1 > 0 then increment_b1                    ' increment b1 by 1 or reset it to 1
goto waiting

play_mode_2:
serout 4,n9600_8,("PC F /SingleSounds/Mod2Snd",#b1,".mp3", CR)
pause 250
if b1 > 0 then increment_b1
goto waiting

play_mode_3:
serout 4,n9600_8,("PC F /SingleSounds/Mod3Snd",#b1,".mp3", CR)
pause 250
if b1 > 0 then increment_b1
goto waiting
```

```
play_mode_4:
serout 4,n9600_8,("PC F /SingleSounds/Mod4Snd",#b1,".mp3", CR)
pause 250
if b1 > 0 then increment_b1
goto waiting

play_mode_5:
serout 4,n9600_8,("PC F /SingleSounds/Mod5Snd",#b1,".mp3", CR)
pause 250
if b1 > 0 then increment_b1
goto waiting

'=========================================

play_rnd_mode_1:
serout 4,n9600_8,("PC F /RandomSounds/Mod1RndSnd",#w3,".mp3", CR)          ' play the sound
indicated by the variable w3 (random file number to play)
pause 250
goto ContinueRandom

play_rnd_mode_2:
serout 4,n9600_8,("PC F /RandomSounds/Mod2RndSnd",#w3,".mp3", CR)
pause 250
goto ContinueRandom

play_rnd_mode_3:
serout 4,n9600_8,("PC F /RandomSounds/Mod3RndSnd",#w3,".mp3", CR)
pause 250
goto ContinueRandom

play_rnd_mode_4:
serout 4,n9600_8,("PC F /RandomSounds/Mod4RndSnd",#w3,".mp3", CR)
pause 250
goto ContinueRandom

play_rnd_mode_5:
serout 4,n9600_8,("PC F /RandomSounds/Mod5RndSnd",#w3,".mp3", CR)
pause 250
goto ContinueRandom

'=========================================
' increments the non random file number to play.
increment_b1:
if b1 < 10 then
        b1 = b1 + 1
else
        b1 = 1
endif
pause 200
goto waiting
```

CFIII Sound system:

```
1  REM ********************************************************************************************************************************************
2  REM *** R2-D2 Builders Group 2009 (http://www.astromech.net/)                                                                                  ***
3  REM ***  This is an ACS BASIC program written to automate sounds played on an ACS CFIII sound board.                                          ***
4  REM ***  CFIII Sound board info. can be found at http://www.acscontrol.com/index_ACS.asp                                                      ***
5  REM ***  Rename the sound files in this program to the files you want to be played for the different                                           ***
6  REM ***  buttons.                                                                                                                              ***
6  REM ***  Copy the complete text of this program to a file named CFSOUND.BAS and copy it to the CF card.                                       ***
7  REM ***                                                                                                                                        ***
8  REM *** CFIII closures Functions                                                                                                               ***
9  REM *** CYCLE THROUGH SOUNDS: Cycle through 10 different sounds with each consecutive press of a button that ***
10 REM ***  is tied to CFIII colsures 9, 10, 11, & 12.                                                                                            ***
11 REM *** PLAY A SINGLE SOUND:  Play one sound file for each button press that is tied to CFIII colsures                                         ***
12 REM ***                                                    1, 2, 3, 13 and 14 .                                                              ***
13 REM *** SOUND VOLUME:                  CFIII colsures 8 & 9 increase and decrease sound volume.                                            ***
14 REM *** RANDOM SOUNDS:                 CFIII colsure 11 will play random sounds at random time intervals until ***
15 REM ***                                                    another button is pressed.                                                        ***
16 REM ***                                                                                                                                      ***
17 REM ***                                                                Programming Note                                                      ***
18 REM *** You can not have any blank lines in the program even at the end or beginning.                                                        ***
19 REM *** All lines must have a line number even if it is a comment.                                                                            ***
20 REM *** After pasting the program into hyper terminal you must hit enter then type "run" to execute the                                      ***
21 REM *** program.                                                                                                                              ***
22 REM *** To enter a new program hit enter then type "new" then hit enter again.                                                               ***
23 REM *** To stop a running program hit the Esc key twice.                                                                                     ***
24 REM *** The CF card should be in the CFIII Sound board so it can access the sounds it needs.                                                 ***
25 REM *** When working in BASIC mode the CFSOUND.BAS on the CF card should be a blank file.                                                     ***
26 REM *** Hyper terminal settings: Baud rate: 2400, Data bits 8, Parity none, Stop bits 1, Flow Control none                                   ***
27 REM ***                                                                                                                                      ***
28 REM ***                                  NOTE: Your Buttons may vary                                                                         ***
26 REM *** The buttons on the remote that perform these functions will vary depending on how you decide to wire                                  ***
30 REM *** the 12 ch relay to your CFIII sound board. The way I chose to wire my board was to wire relays 1-6                                    ***
31 REM *** to closures 9-14 and relays 7-12 to closures 1-6.                                                                                     ***
32 REM *** NOTE: Relays 1-6  are located from the bottom right to the top right (bottom is the side where the                                   ***
33 REM *** power port is located) and relays 7-12 are located from the top left to the bottom left                                               ***
34 REM *** I did this because I used a cable that has 6 individual conductors inside it and this made the                                       ***
35 REM *** wiring more organized. However, wiring it this way means that closures 7 & 8 of the first bank on                                    ***
36 REM *** the CFIII are not being used. If you choose to wire it the same way then the buttons on your remote                                  ***
37 REM *** will perform the following functions:                                                                                                 ***
38 REM *** Buttons 1, 2, 3, 4 will all cycle through the 10 different sounds.                                                                   ***
39 REM *** Buttons 5, 6, 7, 8 will each play a single sound when pressed (always the same sound).                                               ***
40 REM *** Button 9 will mute the speakers.                                                                                                     ***
41 REM *** Button 10 will kick off the random sounds at random times.                                                                           ***
42REM *** Buttons 11, 12 will act as the volume control where 11 decreases volume and 12 increases volume.                                      ***
43 REM ********************************************************************************************************************************************
90 ONERROR GOTO 10000
91 REM ==== Dim 4 variables to store the current value of each button (0 -> 9) ========
92 b1=0
93 b2=0
94 b3=0
95 b4=0
96 REM  ==== Dim 2 variables r = random number, t = sound toggle on/off (1=on 0=off)
97 r=5
98 t=0
99 REM == Contact closure logic for CFIII closures (10, 11, 12, 13) cycles through 10 sounds == (Maps to buttons 4, 3, 2, 1)
100 ONEVENT @CLOSURE(10), GOSUB 500
101 ONEVENT @CLOSURE(11), GOSUB 501
102 ONEVENT @CLOSURE(12), GOSUB 502
103 ONEVENT @CLOSURE(13), GOSUB 503
104 REM == Contact closure logic for CFIII closures (0, 1, 8, 9) play 1 sound per btn press == (Maps to buttons 7, 8, 6, 5)
201 ONEVENT @CLOSURE(0), GOSUB 600
202 ONEVENT @CLOSURE(1), GOSUB 604
203 ONEVENT @CLOSURE(8), GOSUB 623
204 ONEVENT @CLOSURE(9), GOSUB 627
205 REM ==== mute the speakers closure (2) ====== (Map to button 9)
206 ONEVENT @CLOSURE(2), GOSUB 609
207 REM ==== vol increase and decrease for CFIII closures (4 vol (-), 5 vol(+)) ======
```

```
208 ONEVENT @CLOSURE(5), GOSUB 700
209 ONEVENT @CLOSURE(4), GOSUB 704
210 REM ==== initialize the @timer for CFIII closure (3) ====== (Maps to button 10)
211 ONEVENT @CLOSURE(3), GOSUB 801
212 REM ==== when timer reaches 0 execute GOSUB 901 ======
213 ONEVENT @TIMER(1), GOSUB 901
214 GOTO 100
500 ON b1, GOTO 2001,2001,2001,2001,2001,2001,2001,2001,2001,2006
501 ON b2, GOTO 2101,2101,2101,2101,2101,2101,2101,2101,2101,2106
502 ON b3, GOTO 2201,2201,2201,2201,2201,2201,2201,2201,2201,2206
503 ON b4, GOTO 2301,2301,2301,2301,2301,2301,2301,2301,2301,2306
504 REM ==== 1 sound for each btn press (closure 1, 2, 3, 13, 14) =====
600 @TIMER(1)=0
601 @SOUND$="" : @SOUND$="opening.wav"
602 PRINT "closure 0"
603 RETURN
604 @TIMER(1)=0
605 @SOUND$="" : @SOUND$="cantina.wav"
606 PRINT "closure 1"
607 RETURN
608 REM ==== mute the speaker(s) =====
609 REM @TIMER(1)=0
610 IF @MUTE > 0 THEN 613
611 IF @MUTE = 0 THEN 616
612 RETURN
613 @MUTE=0
614 PRINT "closure 2 - muting speakers 1 = mute: " : PRINT "@MUTE Value = ", @MUTE
615 RETURN
616 @MUTE=1
617 PRINT "closure 2 - muting speakers 1 = mute: " : PRINT "@MUTE Value = ", @MUTE
618 RETURN
619 IF @SOUND$ <> "" THEN 620
620 @SOUND$=""
621 PRINT "closure 2 - muting speakers: " : PRINT "@SOUND$ Value = ", @SOUND$
622 RETURN
623 @TIMER(1)=0
624 @SOUND$="" : @SOUND$="yavin.wav"
625 PRINT "closure 8"
626 RETURN
627 @TIMER(1)=0
628 @SOUND$="" : @SOUND$="march.wav"
629 PRINT "closure 9"
630 RETURN
631 REM ==== vol increase and decrease =====
700 REM @TIMER(1)=0
701 @NSVOL = @NSVOL + 1
702 PRINT "vol increased by 1"
703 RETURN
704 REM @TIMER(1)=0
705 @NSVOL = @NSVOL - 1
706 PRINT "vol decreased by 1"
707 RETURN
800 REM ==== initialize the timer =====
801 @TIMER(1)=25
802 RETURN
900 REM ==== timer code for random sound play =====
901 PRINT "timer initiated"
902 @TIMER(1)=RND(2000) + 500
903 REM ==== get a random number (0 - 35) and play the associated sound =====
904 r=RND(35)
905 PRINT "random sound file number: ", r
906 IF r < 35 THEN S$=FMT$("rs%d.wav",r)
907 IF r < 35 THEN @SOUND$=S$
908 IF r < 35 THEN PRINT "playing random sound file: ", S$
909 RETURN
2000 REM ========== closure 10 ===============
2001 @TIMER(1)=0
2002 b1=b1+1
2003 PRINT "closure 10 = ", b1
```

```
2004 GOTO 5001
2005 RETURN
2006 @TIMER(1)=0
2007 b1=0
2008 PRINT "closure 10 = ", b1
2009 GOTO 5001
2010 RETURN
2100 REM ========== closure 11 ==============
2101 @TIMER(1)=0
2102 b2=b2+1
2103 PRINT "closure 11 = ", b2
2104 GOTO 5100
2105 RETURN
2106 @TIMER(1)=0
2107 b2=0
2108 PRINT "closure 11 = ", b2
2109 GOTO 5100
2110 RETURN
2200 REM ========== closure 12 ==============
2201 @TIMER(1)=0
2202 b3=b3+1
2203 PRINT "closure 12 = ", b3
2204 GOTO 5200
2205 RETURN
2206 @TIMER(1)=0
2207 b3=0
2208 PRINT "closure 12 = ", b3
2209 GOTO 5200
2210 RETURN
2300 REM ========== closure 13 ==============
2301 @TIMER(1)=0
2302 b4=b4+1
2303 PRINT "closure 13 = ", b4
2304 GOTO 5300
2305 RETURN
2306 @TIMER(1)=0
2307 b4=0
2308 PRINT "closure 13 = ", b4
2309 GOTO 5300
2310 RETURN
5000 REM ========== Closure 10 sounds ==============
5001 IF b1 = 1 THEN PLAY dske.wav
5002 IF b1 = 1 THEN PRINT "playing sound file: dske.wav"
5003 IF b1 = 2 THEN PLAY alarmrev.wav
5004 IF b1 = 2 THEN PRINT "playing sound file: alarmrev.wav"
5005 IF b1 = 3 THEN PLAY tuneful.wav
5006 IF b1 = 3 THEN PRINT "playing sound file: tuneful.wav"
5007 IF b1 = 4 THEN PLAY behind.wav
5008 IF b1 = 4 THEN PRINT "playing sound file: behind.wav"
5009 IF b1 = 5 THEN PLAY bobble.wav
5010 IF b1 = 5 THEN PRINT "playing sound file: bobble.wav"
5011 IF b1 = 6 THEN PLAY busybusy.wav
5012 IF b1 = 6 THEN PRINT "playing sound file: busybusy.wav"
5013 IF b1 = 7 THEN PLAY caution.wav
5014 IF b1 = 7 THEN PRINT "playing sound file: caution.wav"
5015 IF b1 = 8 THEN PLAY chatrev.wav
5016 IF b1 = 8 THEN PRINT "playing sound file: chatrev.wav"
5017 IF b1 = 9 THEN PLAY chatraz.wav
5018 IF b1 = 9 THEN PRINT "playing sound file: chatraz.wav"
5019 IF b1 = 0 THEN PLAY chuckraz.wav
5020 IF b1 = 0 THEN PRINT "playing sound file: chuckraz.wav"
5021 RETURN
5022 REM ========== Closure 11 sounds ==============
5100 IF b2 = 1 THEN PLAY fastchat.wav
5101 IF b2 = 1 THEN PRINT "playing sound file: fastchat.wav"
5102 IF b2 = 2 THEN PLAY zaap.wav
5103 IF b2 = 2 THEN PRINT "playing sound file: zaap.wav"
5104 IF b2 = 3 THEN PLAY wow.wav
5105 IF b2 = 3 THEN PRINT "playing sound file: wow.wav"
```

```
5106 IF b2 = 4 THEN PLAY worry.wav
5107 IF b2 = 4 THEN PRINT "playing sound file: worry.wav"
5108 IF b2 = 5 THEN PLAY worry2.wav
5109 IF b2 = 5 THEN PRINT "playing sound file: worry2.wav"
5110 IF b2 = 6 THEN PLAY wolfwhis.wav
5111 IF b2 = 6 THEN PRINT "playing sound file: wolfwhis.wav"
5112 IF b2 = 7 THEN PLAY whistlin.wav
5113 IF b2 = 7 THEN PRINT "playing sound file: whistlin.wav"
5114 IF b2 = 8 THEN PLAY whistle.wav
5115 IF b2 = 8 THEN PRINT "playing sound file: whistle.wav"
5116 IF b2 = 9 THEN PLAY tuneful.wav
5117 IF b2 = 9 THEN PRINT "playing sound file: tuneful.wav"
5118 IF b2 = 0 THEN PLAY chuckraz.wav
5119 IF b2 = 0 THEN PRINT "playing sound file: bobble.wav"
5120 RETURN
5121 REM ========== Closure 12 sounds ===============
5200 IF b3 = 1 THEN PLAY ditty.wav
5201 IF b3 = 1 THEN PRINT "playing sound file: ditty.wav"
5202 IF b3 = 2 THEN PLAY cuckoo.wav
5203 IF b3 = 2 THEN PRINT "playing sound file: cuckoo.wav"
5204 IF b3 = 3 THEN PLAY chunter.wav
5205 IF b3 = 3 THEN PRINT "playing sound file: chunter.wav"
5206 IF b3 = 4 THEN PLAY dski.wav
5207 IF b3 = 4 THEN PRINT "playing sound file: dski.wav"
5208 IF b3 = 5 THEN PLAY fastchat.wav
5209 IF b3 = 5 THEN PRINT "playing sound file: fastchat.wav"
5210 IF b3 = 6 THEN PLAY fasttalk.wav
5211 IF b3 = 6 THEN PRINT "playing sound file: fasttalk.wav"
5212 IF b3 = 7 THEN PLAY fnew.wav
5213 IF b3 = 7 THEN PRINT "playing sound file: fnew.wav"
5214 IF b3 = 8 THEN PLAY leiahelp.wav
5215 IF b3 = 8 THEN PRINT "playing sound file: leiahelp.wav"
5216 IF b3 = 9 THEN PLAY lookout.wav
5217 IF b3 = 9 THEN PRINT "playing sound file: lookout.wav"
5218 IF b3 = 0 THEN PLAY plea.wav
5219 IF b3 = 0 THEN PRINT "playing sound file: plea.wav"
5220 RETURN
5221 REM ========== Closure 13 sounds ===============
5300 IF b4 = 1 THEN PLAY clanger2.wav
5301 IF b4 = 1 THEN PRINT "playing sound file: clanger2.wav"
5302 IF b4 = 2 THEN PLAY surprise.wav
5303 IF b4 = 2 THEN PRINT "playing sound file: surprise.wav"
5304 IF b4 = 3 THEN PLAY skipraz.wav
5305 IF b4 = 3 THEN PRINT "playing sound file: skipraz.wav"
5306 IF b4 = 4 THEN PLAY shortcir.wav
5307 IF b4 = 4 THEN PRINT "playing sound file: shortcir.wav"
5308 IF b4 = 5 THEN PLAY r2itisu.wav
5309 IF b4 = 5 THEN PRINT "playing sound file: r2itisu.wav"
5310 IF b4 = 6 THEN PLAY procholo.wav
5311 IF b4 = 6 THEN PRINT "playing sound file: procholo.wav"
5312 IF b4 = 7 THEN PLAY process.wav
5313 IF b4 = 7 THEN PRINT "playing sound file: process.wav"
5314 IF b4 = 8 THEN PLAY process5.wav
5315 IF b4 = 8 THEN PRINT "playing sound file: process5.wav"
5316 IF b4 = 9 THEN PLAY process4.wav
5317 IF b4 = 9 THEN PRINT "playing sound file: process4.wav"
5318 IF b4 = 0 THEN PLAY process2.wav
5319 IF b4 = 0 THEN PRINT "playing sound file: process2.wav"
5320 RETURN
5321 REM ========== Error Handling ===============
10000 PLAY scream.wav
10001 PRINT "Error #", ERR()
10002 REM == PRINT "Error message:", ERR$()
10003 IF ERR() > 0 THEN 100
10004 REM == If the error is an invalid file (23) then continue ==
10005 REM == If the error is timer out of range (18) then continue ==
```

Picaxe 2-3-2 Control Code (Experimental):

```
pause 1000
b0 = 0
b1 = 0

main:

if pin6 = 1 then gosub TwoToThree
b0 = 0
if pin7 = 1 then gosub ThreeToTwo
b1 = 0
pause 1000
goto main

TwoToThree:

 do
  if pin0 = 1 and pin1 = 0 and pin2 = 0  then ' Only top sensor is triggered
   goto TwoToThree1stPhase
  elseif pin0 = 0 and pin1 = 1 and pin2 = 0 then ' Only middle sensor is triggered
   goto TwoToThree2ndPhase
  elseif pin2 = 1      then ' Only bottom sensor is triggered
   goto TwoToThreeComplete
  else
   exit
  endif

  TwoToThree1stPhase:
  do
   serout 1,T2400_4,(0)
  loop until pin1 = 1

  TwoToThree2ndPhase:
  do
   serout 2,T2400_4,(0) ' Start shoulder linear actuators
   serout 3,T2400_4,(0) ' Start leg linear actuators
  loop until pin2 = 1

  TwoToThreeComplete:
  serout 1,T2400_4,(127) ' Stop center foot motor
  serout 2,T2400_4,(127) ' Stop shoulder linear actuators
  serout 3,T2400_4,(127) ' Stop shoulder linear actuators
  b0 = 1

 loop while b0 <> 1
```

```
return


ThreeToTwo:

 do
  if pin0 = 0 and pin1 = 0 and pin2 = 1 then ' Only bottom sensor is triggered
   goto ThreeToTwoStart
  elseif pin0 = 1       then ' Only top sensor is triggered
   goto ThreeToTwoComplete
  else
   exit
  endif

  ThreeToTwoStart:
  do
   serout 1,T2400_4,(#254)
   serout 2,T2400_4,(#254)
   serout 3,T2400_4,(#254)
  loop until pin0 = 1

  ThreeToTwoComplete:
  serout 1,T2400_4,(127) ' Stop center foot motor
  serout 2,T2400_4,(127) ' Stop shoulder linear actuators
  serout 3,T2400_4,(127) ' Stop shoulder linear actuators
  b1 = 1

 loop while b1 <> 1
```

Picaxe front charging port light pattern software:

```
main:

 random w0   ' Generate a random number.

 let w1 = w0 % 5  ' Get the modulus (remainder) of the random number divided by 5
 let w2 = w0 % 8
 let w3 = w0 % 9
 let w4 = w0 % 10
 let w5 = w0 % 11
 let w6 = w0 % 12

 let outpins = %00000010

 gosub RandLn ' call sub-procedure(s)

 pause 150
```

```
 let outpins = %00000100
 gosub RandLn
 pause 150
 let outpins = %00001000
 gosub RandLn
 pause 150
 let outpins = %00010000
 gosub RandLn
 pause 150
 let outpins = %00100000
 gosub RandLn
 pause 150
 let outpins = %00100000
 gosub RandLn
 pause 150
 let outpins = %01000000
 gosub RandLn
 pause 150

 goto main

RandLn:

 ' Set one additional line of LEDs to high (i.e. turned on)
 if w1 > 0 AND w1 < 5 then ' w1 will be 1,2,3, or 4
  high w1
 end if

 if w2 > 0 AND w2 < 5 then ' w2 will be 1,2,3, or 4
  high w2
 end if

 if w3 > 0 AND w3 < 5 then ' w3 will be 1,2,3, or 4
  high w3
 end if

 if w4 > 0 AND w4 < 5 then ' w4 will be 1,2,3, or 4
  high w4
 end if

 if w5 > 0 AND w5 < 5 then ' w5 will be 1,2,3, or 4
  high w5
 end if

 if w6 > 0 AND w6 < 5 then ' w6 will be 1,2,3, or 4
  high w6
 end if

 return   ' return from sub-procedure
```