Miscellaneous information about the Sensormatic RS422 Translator

Minor software Listings

22 March 2002

TABLE OF CONTENTS

LIST OF FIGURES

[1] $Header: d:/sears/RCS/txbslist.tex,v 1.4 2002-01-25 13:54:52-08 Hamilton Exp Hamilton $

Note

A change bar will be used where any changes have been made since the last version of this set of miscellaneous information was printed.

As is common, those pages with changed page numbers and references to those numbers, (this includes the Table of Contents and the Index) do not qualify for a change bar. Neither will there be a change bar on this page.

A summary of the changes will also be listed here.

Date	Notes
None	

1 GW-Basic support programs

This is a collection of minor GW-Basic routines. There were additional .BAS programs used on this project. These additional programs were used to generate tables of addresses and were quite boring, so I didn't list them here. Programs in the "terminaly boring" class include all of the following: ADDRESS0.BAS, ADDRESS1.BAS, ADDRESS2.BAS and ADDRESS3.BAS.

1.1 CKSUM.BAS

```
 1 10 CLS: REV$ = "$Header: d:/sears/RCS/cksum.bas,v 1.2 2001-03-26 11:30:09-08 Hamilton Exp Hamilton $"
 2 20 PRINT REV$
 3 30 '
 4 40 TOTAL = 0
 5 50 PRINT "Do a running calculation of Sensormatic's checksum"
 6 60 INPUT "Input a hexidecimal number (cr = stop)";HEXIN$
 7 70 GOSUB 160: REM Convert input from hex to decimal
 8 80 INPUT "Next number (cr = stop)";HEXIN$
 9 90 GOSUB 160: REM Convert input from hex to decimal
10 100 GOTO 80
11 110 '
12 120 REM All done
13 130 PRINT "All done, quitting"
14 140 END
15 150 '
16 160 REM Convert a hex string into decimal
17 170 HEXVALUE = 0
18 180 IF LEN(HEXIN$) = 0 THEN 120
19 190 '
20 200 FOR I = 1 TO LEN(HEXIN$)
21 210 X$ = MID$(HEXIN$,I,1)
22 220 IF X$ = " " THEN 380
23 230 '
24 240 REM Now convert the input digit from character to numeric
25 250 IF X$ < "0" OR X$ > "9" THEN 290
26 260 X = ASC(X$) - 48
27 270 GOTO 360
28 280 '
29 290 IF X$ > "F" THEN 330
30 300 X = ASC(X$) - 55
31 310 GOTO 360
32 320 '
33 330 IF X$ > "f" THEN 380
34 340 X = ASC(X$) - 87
35 350 '
36 360 HEXVALUE = (HEXVALUE*16) + X
37 370 REM PRINT " hexvalue = '";HEXVALUE;"' "
38 380 NEXT I
39 390 TOTAL = TOTAL + HEXVALUE
40 400 CHECKSUM = &H100 - TOTAL
41 410 PRINT "Checksum is '";RIGHT$(HEX$(CHECKSUM),2);"' ";
42 420 RETURN
```

[2] $Header: d:/sears/RCS/basic.inc,v 1.8 2002-01-25 13:54:42-08 Hamilton Exp Hamilton $

1.2 DIAGONAL.BAS

```
10 CLS: REV$ = "$Header: d:/sears/RCS/diagonal.bas,v 1.11 2001-12-28 15:03:06-08 Hamilton Exp Hamilton $"
20 PRINT REV$
30 PRINT "Simulate a Sensormatic, SensorVison"
40 PRINT "RC58 controller sending diagonal movement commands."
50 PRINT "From the keypad, when NOT in 'num lock' mode."
60 PRINT
70 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
80 PRINT "or this program will time out when the serial communications is opened."
90 PRINT "(With a 9 pin connector these are 7,8 and 6,1,4.)"
100 PRINT "(With a 25 pin connector these are 4,5 and 6,8,20.)"
110 PRINT
120 YES = 1: NO = 0
130 DIRECTIONMODE% = 0: REM 0 = tilt then pan, 1 = pan then tilt
140 '
150 OPEN "status.log" FOR OUTPUT AS #3
160 PRINT #3,: PRINT #3,REV$
170 PRINT #3,DATE$,TIME$
180 '
190 INPUT "For this run should we have real outputs (cr = yes)";X$
200 IF X$ = "" THEN REAL = YES ELSE REAL = NO
210 IF REAL = NO THEN 310
220 PRINT "COM1 = DB-25, COM2 = DB-9"
230 INPUT "Use which COM port for IO (cr = COM1, anything else is COM2)";X$
240 IF X$ = "" THEN PORT$ = "COM1" ELSE PORT$ = "COM2"
250 OPENSTRING$ = PORT$ + ":4800,N,8,1"
260 OPEN OPENSTRING$ AS #2
270 PRINT #3,"The IO setup string was '";OPENSTRING$;"'"
280 PRINT #3,"This run writes data out to a dome"
290 GOTO 330
300 '
310 PRINT #3,"This is a simulation run that does not write data out"
320 '
330 INPUT "Should polling be used (cr = no)";X$
340 IF X$ = "" THEN POLLIT = NO ELSE POLLIT = YES
350 '
360 INPUT "Use what for a camera address (cr = 44)";ADDRESS
370 IF ADDRESS = 0 THEN ADDRESS = 44
380 PRINT "Using a camera address of ";ADDRESS
390 PRINT #3,"Writing to camera ";ADDRESS
400 '
410 INPUT "Send each command how many times (cr = 1)";RESENDCOUNT
420 IF RESENDCOUNT = 0 THEN RESENDCOUNT = 1
430 PRINT #3,"Resending each command ";RESENDCOUNT;" times"
440 '
450 FAST       = &H8D: FASTEST    = &H8E: FASTSTOP   = &H8F
460 FASTER     = &H9A: FASTERSTOP = &H9B
470 FOCUSNEAR  = &H87: FOCUSFAR   = &H88: FOCUSSTOP  = &H89
480 IRISOPEN   = &H90: IRISCLOSE  = &H91: IRISSTOP   = &H92
490 LEFT       = &H81: RIGHT      = &H82: PANSTOP    = &H83
500 UP         = &H84: DOWN       = &H85: TILTSTOP   = &H86
510 ZOOMIN     = &H8A: ZOOMOUT    = &H8B: ZOOMSTOP   = &H8C
520 ONAIR      = &H9E: OFFAIR     = &H9F: RESETT     = &HC6
530 POLL       = &H94: STOPALL    = &H93
540 '
550 CLS: GOSUB 2460
560 TIMESTEPSPERMS = 0: GOSUB 2730
570 COMMANDTYPE$ = "On": COMMAND = ONAIR: GOSUB 2130: REM Turn it on
580 STARTPOLL = 1: LASTPOLL = 58: SOMECOMMAND = YES
590 BYTE1 = 99: BYTE2 = 99: BYTE3 = 99
```

```
600 PRINT #3,"Direction mode is Tilt then Pan"
610 PRINT #3,
620 '
630 REM All commands return here and look for more to do
640 REM Send a poll command about once a second
650 IF TIME$ <> LASTSECOND$ THEN GOSUB 2250
660 LASTSECOND$ = TIME$
670 IF TIME$ <> LASTSECOND$ THEN GOSUB 2250
680 KPAD$ = INKEY$
690 IF LEN(KPAD$) = 0 THEN 670
700 PRINT " ";
710 IF ASC(KPAD$) >= ASC("A") AND ASC(KPAD$) <= ASC("Z") THEN 720 ELSE 740
720 KPAD$ = STR$(ASC(KPAD$) - 32)
730 '
740 REM Decode what just came in
750 IF KPAD$ = " " THEN 1500: REM Stop the current movement
760 IF KPAD$ = "?" THEN 2420: REM Show help
770 IF KPAD$ = "/" THEN 2420: REM Show help
780 IF KPAD$ = "\" THEN ID1$ = "Reset":      GOTO 1810
790 IF KPAD$ = ";" THEN ID1$ = "StopZoom":   GOTO 1810
800 IF KPAD$ = "a" THEN ID1$ = "Fast":       GOTO 1580
810 IF KPAD$ = "d" THEN ID1$ = "Fastest":    GOTO 1580
820 IF KPAD$ = "e" THEN ID1$ = "FocusFar":   GOTO 1580
830 IF KPAD$ = "f" THEN ID1$ = "StopFast":   GOTO 1810
840 IF KPAD$ = "g" THEN ID1$ = "StopFaster": GOTO 1810
850 IF KPAD$ = "h" THEN ID1$ = "StopFocus":  GOTO 1810
860 IF KPAD$ = "j" THEN ID1$ = "StopIris":   GOTO 1810
870 IF KPAD$ = "k" THEN ID1$ = "StopTilt":   GOTO 1810
880 IF KPAD$ = "l" THEN ID1$ = "StopPan":    GOTO 1810
890 IF KPAD$ = "m" THEN DIRECTIONMODE% = DIRECTIONMODE% XOR 1: GOTO 1250
900 IF KPAD$ = "r" THEN ID1$ = "IrisClose":  GOTO 1580
910 IF KPAD$ = "s" THEN ID1$ = "Faster":     GOTO 1580
920 IF KPAD$ = "t" THEN ID1$ = "IrisOpen":   GOTO 1580
930 IF KPAD$ = "q" THEN ID1$ = "StopAll":    GOTO 1810
940 IF KPAD$ = "u" THEN ID1$ = "ZoomOut":    GOTO 1580
950 IF KPAD$ = "w" THEN ID1$ = "FocusNear":  GOTO 1580
960 IF KPAD$ = "x" THEN 1430: REM Exit
970 IF KPAD$ = "y" THEN ID1$ = "ZoomIn":     GOTO 1580
980 REM Processing for numeric inputs
990 X = ASC(KPAD$) - 48
1000 IF X < 0 OR X > 9 THEN 1050
1010 IF BYTE1 = 99 THEN BYTE1 = X: GOTO 630
1020 IF BYTE2 = 99 THEN BYTE2 = X: GOTO 630
1030 IF BYTE3 = 99 THEN BYTE3 = X: GOTO 1340
1040 REM Processing for keypad, multibyte, inputs
1050 IF LEN(KPAD$) = 1 THEN INPUTKEY = ASC(KPAD$): GOTO 1210
1060 IF LEN(KPAD$) <> 2 THEN 630
1070 INPUTKEY = ASC(MID$(KPAD$,2,1))
1080 '
1090 IF INPUTKEY = &H47 THEN ID1$ = "UL"     : GOTO 1930: REM Up-Left
1100 IF INPUTKEY = &H48 THEN ID1$ = "U"      : GOTO 1580: REM Up
1110 IF INPUTKEY = &H49 THEN ID1$ = "UR"     : GOTO 1930: REM Up-Right
1120 IF INPUTKEY = &H4B THEN ID1$ = "L"      : GOTO 1580: REM Left
1130 IF INPUTKEY = &H4D THEN ID1$ = "R"      : GOTO 1580: REM Right
1140 IF INPUTKEY = &H4F THEN ID1$ = "DL"     : GOTO 1930: REM Down-Left
1150 IF INPUTKEY = &H50 THEN ID1$ = "D"      : GOTO 1580: REM Down
1160 IF INPUTKEY = &H51 THEN ID1$ = "DR"     : GOTO 1930: REM Down-Right
1170 IF INPUTKEY = &H52 THEN ID1$ = "Fast"   : GOTO 1580: REM Fast
1180 IF INPUTKEY = &H53 THEN ID1$ = "Fastest": GOTO 1580: REM Fastest
1190 '
```

```
120 1200 REM Processing for undefined inputs
121 1210 PRINT "'0x";HEX$(INPUTKEY);"', ";
122 1220 PRINT #3,TIME$," '0x";HEX$(INPUTKEY);"', "
123 1230 GOTO 630
124 1240 '
125 1250 REM Indicate that the direction mode has changed
126 1260 IF DIRECTIONMODE% = 1 THEN 1300
127 1270 PRINT: PRINT "New direction mode = pan then tilt";
128 1280 PRINT #3,"New direction mode = pan then tilt"
129 1290 GOTO 1320
130 1300 PRINT: PRINT "New direction mode = tilt then pan";
131 1310 PRINT #3,"New direction mode = tilt then pan"
132 1320 GOTO 630
133 1330 '
134 1340 REM Send an arbitrary command
135 1350 COMMAND = (BYTE1*100) + (BYTE2*10) + BYTE3
136 1360 BYTE1 = 99: BYTE2 = 99: BYTE3 = 99
137 1370 PRINT "'0x";HEX$(COMMAND);"'";
138 1380 REM PRINT #3,"Arbitrary command ";HEX$(COMMAND)
139 1390 COMMANDTYPE$ = "Ar"
140 1400 GOSUB 2130
141 1410 GOTO 630
142 1420 '
143 1430 REM All done
144 1440 PRINT #3,"All done"
145 1450 PRINT #3,DATE$,TIME$
146 1460 PRINT
147 1470 PRINT "Quitting"
148 1480 END
149 1490 '
150 1500 REM Stop the current movement
151 1510 IF STOP1 = NO THEN 1530
152 1520 COMMANDTYPE$ = STOPID1$: COMMAND = STOP1: GOSUB 2130
153 1530 IF STOP2 = NO THEN 1550
154 1540 COMMANDTYPE$ = STOPID2$: COMMAND = STOP2: GOSUB 2130
155 1550 STOP1 = NO: STOP2 = NO
156 1560 GOTO 630
157 1570 '
158 1580 REM Send a single starting command
159 1590 IF ID1$ = "U" THEN COMMAND1 = UP:    STOP1 = TILTSTOP: STOPID1$ = "Stilt"
160 1600 IF ID1$ = "D" THEN COMMAND1 = DOWN:  STOP1 = TILTSTOP: STOPID1$ = "Stilt"
161 1610 IF ID1$ = "R" THEN COMMAND1 = RIGHT: STOP1 = PANSTOP:  STOPID1$ = "Span"
162 1620 IF ID1$ = "L" THEN COMMAND1 = LEFT:  STOP1 = PANSTOP:  STOPID1$ = "Span"
163 1630 '
164 1640 REM These commands don't change the type of "stop" currently specified
165 1650 IF ID1$ = "Fast"     THEN COMMAND1 = FAST:
166 1660 IF ID1$ = "Faster"   THEN COMMAND1 = FASTER:
167 1670 IF ID1$ = "Fastest"  THEN COMMAND1 = FASTEST:
168 1680 '
169 1690 IF ID1$ = "ZoomIn"    THEN COMMAND1 = ZOOMIN
170 1700 IF ID1$ = "ZoomOut"   THEN COMMAND1 = ZOOMOUT
171 1710 IF ID1$ = "FocusNear" THEN COMMAND1 = FOCUSNEAR
172 1720 IF ID1$ = "FocusFar"  THEN COMMAND1 = FOCUSFAR
173 1730 IF ID1$ = "IrisClose" THEN COMMAND1 = IRISCLOSE
174 1740 IF ID1$ = "IrisOpen"  THEN COMMAND1 = IRISOPEN
175 1750 '
176 1760 REM Send out stops and starts from here
177 1770 STOP2 = NO
178 1780 COMMANDTYPE$ = ID1$: COMMAND = COMMAND1: GOSUB 2130
179 1790 GOTO 630
```

```
180 1800 '
181 1810 REM Processing for the seven types of stop commands
182 1820 IF ID1$ = "Reset"      THEN COMMAND1 = RESETT
183 1830 IF ID1$ = "StopFast"   THEN COMMAND1 = FASTSTOP
184 1840 IF ID1$ = "StopFaster" THEN COMMAND1 = FASTERSTOP
185 1850 IF ID1$ = "StopFocus"  THEN COMMAND1 = FOCUSSTOP
186 1860 IF ID1$ = "StopIris"   THEN COMMAND1 = IRISSTOP
187 1870 IF ID1$ = "StopTilt"   THEN COMMAND1 = TILTSTOP
188 1880 IF ID1$ = "StopPan"    THEN COMMAND1 = PANSTOP
189 1890 IF ID1$ = "StopZoom"   THEN COMMAND1 = ZOOMSTOP
190 1900 IF ID1$ = "StopAll"    THEN COMMAND1 = STOPALL
191 1910 GOTO 1760
192 1920 '
193 1930 REM Send a diagonal command
194 1940 IF DIRECTIONMODE% = 1 THEN 2020
195 1950 IF ID1$ = "UL" THEN COMMAND1 = UP:   COMMAND2 = LEFT
196 1960 IF ID1$ = "UR" THEN COMMAND1 = UP:   COMMAND2 = RIGHT
197 1970 IF ID1$ = "DL" THEN COMMAND1 = DOWN: COMMAND2 = LEFT
198 1980 IF ID1$ = "DR" THEN COMMAND1 = DOWN: COMMAND2 = RIGHT
199 1990 STOP1 = PANSTOP: STOP2 = TILTSTOP
200 2000 STOPID1$ = "Span": STOPID2$ = "Stilt"
201 2010 GOTO 2080
202 2020 IF ID1$ = "UL" THEN COMMAND2 = UP:   COMMAND1 = LEFT:  ID1$ = "LU"
203 2030 IF ID1$ = "UR" THEN COMMAND2 = UP:   COMMAND1 = RIGHT: ID1$ = "RU"
204 2040 IF ID1$ = "DL" THEN COMMAND2 = DOWN: COMMAND1 = LEFT:  ID1$ = "LD"
205 2050 IF ID1$ = "DR" THEN COMMAND2 = DOWN: COMMAND1 = RIGHT: ID1$ = "RD"
206 2060 STOP2 = PANSTOP: STOP1 = TILTSTOP
207 2070 STOPID2$ = "Span": STOPID1$ = "Stilt"
208 2080 REM now send the command
209 2090 COMMANDTYPE$ = ID1$: COMMAND = COMMAND1: GOSUB 2130
210 2100 COMMANDTYPE$ = "":   COMMAND = COMMAND2: GOSUB 2130
211 2110 GOTO 630
212 2120 '
213 2130 REM Send a command
214 2140 PRINT COMMANDTYPE$;: SOMECOMMAND = YES
215 2150 CHECKSUM = &H100 - (ADDRESS + COMMAND)
216 2160 PRINT #3,USING"& 0x\\,\         \";TIME$,HEX$(COMMAND),COMMANDTYPE$;
217 2170 PRINT #3,USING"### ### ###";ADDRESS,COMMAND,CHECKSUM
218 2180 IF REAL = NO THEN 2230
219 2190 FOR I = 1 TO RESENDCOUNT
220 2200 PRINT #2,CHR$(ADDRESS);CHR$(COMMAND);CHR$(CHECKSUM);
221 2210 GOSUB 2730
222 2220 NEXT I
223 2230 RETURN
224 2240 '
225 2250 REM Send a poll command about once a second
226 2260 LASTSECOND$ = TIME$
227 2270 THISPOLL = THISPOLL + 1
228 2280 IF THISPOLL > LASTPOLL THEN THISPOLL = STARTPOLL
229 2290 IF THISPOLL <> 1 THEN 2320
230 2300 PRINT ".";: IF SOMECOMMAND = YES THEN PRINT
231 2310 SOMECOMMAND = NO
232 2320 CHECKSUM = &H100 - (THISPOLL + POLL)
233 2330 IF POLLIT = NO OR REAL = NO THEN 2400
234 2340 PRINT #3,USING"& 0x\\,\         \";TIME$,HEX$(POLL),"Poll";
235 2350 PRINT #3,USING"### ### ###";THISPOLL,POLL,CHECKSUM
236 2360 FOR I = 1 TO RESENDCOUNT
237 2370 GOSUB 2730
238 2380 PRINT #2,CHR$(THISPOLL);CHR$(POLL);CHR$(CHECKSUM);
239 2390 NEXT I
```

```
240 2400 RETURN
241 2410 '
242 2420 REM Help Caller
243 2430 GOSUB 2460
244 2440 GOTO 630
245 2450 '
246 2460 REM Help data
247 2470 PRINT
248 2480 PRINT "Stop most operations with a 'blank', exit with an 'x', ? = this"
249 2490 '
250 2500 PRINT "Dome addres is:";ADDRESS;" Active keys are:"
251 2510 PRINT "+----------+------+-----------+"
252 2520 PRINT "| UpLeft   | Up   | UpRight   |"
253 2530 REM      47         48     49
254 2540 PRINT "+----------+------+-----------+"
255 2550 PRINT "| Left     | ---  | Right     |"
256 2560 REM      4B                4D
257 2570 PRINT "+----------+------+-----------+"
258 2580 PRINT "| DownLeft | Down | DownRight |"
259 2590 REM      4F         50     51
260 2600 PRINT "+----------+------+-----------+"
261 2610 PRINT "| Fast            | Faster    |"
262 2620 REM      52                53
263 2630 PRINT "+-----------------+-----------+"
264 2640 '
265 2650 PRINT "a = fast, s = faster, d = fastest"
266 2660 PRINT "o = zoom out, i = zoom in, m = toggle directionmode"
267 2670 PRINT "Stops  q = all,  f = fast, g = faster, h = focus"
268 2680 PRINT "       j = iris, k = tilt, l = pan,    ; = zoom"
269 2690 PRINT "Starts w = focus near, e = far, r = iris close, t = open"
270 2700 PRINT "       y = zoom in,    u = out"
271 2710 RETURN
272 2720 '
273 2730 REM Delay long enough for the command to go out
274 2740 FOR K = 1 TO 30
275 2750 GOSUB 2790
276 2760 NEXT K
277 2770 RETURN
278 2780 '
279 2790 REM Delay a short while
280 2800 IF TIMESTEPSPERMS = 0 THEN GOSUB 2920
281 2810 FOR J = 1 TO TIMESTEPSPERMS
282 2820 ' GOSUB 2830
283 2830 NEXT J
284 2840 RETURN
285 2850 '
286 2860 REM Don't change anything in the "Timer thing", including comments as
287 2870 REM this may change the timing values it generates.
288 2880 REM Timer thing
289 2890 REM X = x + 1: ' 37 / 17
290 2900 RETURN
291 2910 '
292 2920 REM Calibrate the timer so as to get steps/millieseconds
293 2930 PRINT "Calibrating the loop timer, takes 1 to 3 seconds"
294 2940 X$ = TIME$
295 2950 IF X$ = TIME$ GOTO 2940
296 2960 X$ = TIME$
297 2970 GOSUB 2880
298 2980 TIMESTEPSPERMS = TIMESTEPSPERMS + 1
299 2990 IF X$ = TIME$ GOTO 2970
```

```
300 3000 TIMESTEPSPERMS = TIMESTEPSPERMS / 100
301 3010 PRINT USING "Using ### steps to give a one ms delay";TIMESTEPSPERMS
302 3020 PRINT #3,USING "Using ### steps to give a one ms delay";TIMESTEPSPERMS
303 3030 RETURN
```

1.3 MONADKBD.BAS

```
 1 10 CLS
 2 20 REV$ = "$Header: d:/sears/RCS/monadkbd.bas,v 1.3 2001-08-09 12:40:28-07 Hamilton Exp Hamilton $"
 3 30 PRINT REV$
 4 40 PRINT "Monitor AD2078 keyboard communications": PRINT
 5 50 PRINT "Active key is: X to exit"
 6 60 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
 7 70 PRINT "or 7,8 and 1,4,6 for 9 pin connectors"
 8 80 PRINT "or this program will time out when COM1 is opened."
 9 90 PRINT
10 100 OPEN "COM1:1200,N,8,1" AS #1
11 110 FILES
12 120 INPUT "Write into which file (cr = adkeybrd.out)";OUTFILE$
13 130 IF OUTFILE$ = "" THEN OUTFILE$ = "adkeybrd.out"
14 140 OPEN OUTFILE$ FOR APPEND AS #2
15 150 PRINT #2,
16 160 PRINT #2,REV$
17 170 PRINT #2,DATE$,TIME$
18 180 PRINT #2,
19 190 '
20 200 CMNDCOUNT% = 1
21 210 PRINT USING"& ### ";TIME$;CMNDCOUNT%;
22 220 PRINT #2,USING"& ### ";TIME$;CMNDCOUNT%;
23 230 '
24 240 REM Keep looking for an exit command or an input byte
25 250 KPAD$ = INKEY$
26 260 IF KPAD$ = "x" OR KPAD$ = "X" THEN 400
27 270 A$ = INPUT$(1,#1)
28 280 IF LEN(A$) = 0 THEN 240
29 290 X% = ASC(A$)
30 300 GOSUB 470
31 310 PRINT X$;" ";: PRINT #2,X$;" ";
32 320 IF A$ <> "a" THEN 380
33 330 PRINT: PRINT #2,
34 340 PRINT USING"& ### ";TIME$;CMNDCOUNT%;
35 350 PRINT #2,USING"& ### ";TIME$;CMNDCOUNT%;
36 360 CMNDCOUNT% = CMNDCOUNT% + 1
37 370 IF CMNDCOUNT% > 999 THEN CMNDCOUNT% = 0
38 380 GOTO 240
39 390 '
40 400 REM All done
41 410 PRINT #2,
42 420 PRINT #2,DATE$,TIME$
43 430 PRINT #2,
44 440 PRINT "Exiting"
45 450 END
46 460 '
47 470 REM Convert an integer into ascii-hex
48 480 X$ = HEX$(X%)
49 490 IF LEN(X$) <>2 THEN X$ = "0" + X$
50 500 RETURN
```

1.4 PFS.BAS

```
 1 10 CLS: REV$ = "$Header: d:/sears/RCS/pfs.bas,v 1.4 2002-01-22 14:17:13-08 Hamilton Exp Hamilton $"
 2 20 PRINT REV$
 3 30 PRINT "Send Pan Fast, Faster and Fastest commands(with stops)  to a Spectra"
 4 40 PRINT
 5 50 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
 6 60 PRINT "or this program will time out when the serial communications is opened."
 7 70 PRINT "(With a 9 pin connector these are 7,8 and 6,1,4.)"
 8 80 PRINT "(With a 25 pin connector these are 4,5 and 6,8,20.)"
 9 90 PRINT
10 100 YES = 1: NO = 0
11 110 '
12 120 OPEN "status.log" FOR OUTPUT AS #3
13 130 PRINT #3,: PRINT #3,REV$
14 140 PRINT #3,DATE$,TIME$
15 150 '
16 160 INPUT "For this run should we have real outputs (cr = yes)";X$
17 170 IF X$ = "" THEN REAL = YES ELSE REAL = NO
18 180 IF REAL = NO THEN 280
19 190 PRINT "COM1 = DB-25, COM2 = DB-9"
20 200 INPUT "Use which COM port for IO (cr = COM1, anything else is COM2)";X$
21 210 IF X$ = "" THEN PORT$ = "COM1" ELSE PORT$ = "COM2"
22 220 OPENSTRING$ = PORT$ + ":4800,N,8,1"
23 230 OPEN OPENSTRING$ AS #2
24 240 PRINT #3,"The IO setup string was '";OPENSTRING$;"'"
25 250 PRINT #3,"This run writes data out to a dome"
26 260 GOTO 300
27 270 '
28 280 PRINT #3,"This is a simulation run that does not write data out"
29 290 '
30 300 INPUT "Should polling be used (cr = no)";X$
31 310 IF X$ = "" THEN POLLIT = NO ELSE POLLIT = YES
32 320 '
33 330 INPUT "Use what for a camera address (cr = 44)";ADDRESS
34 340 IF ADDRESS = 0 THEN ADDRESS = 44
35 350 PRINT "Using a camera address of ";ADDRESS
36 360 PRINT #3,"Writing to camera ";ADDRESS
37 370 '
38 380 INPUT "Send each command how many times (cr = 1)";RESENDCOUNT
39 390 IF RESENDCOUNT = 0 THEN RESENDCOUNT = 1
40 400 PRINT #3,"Resending each command ";RESENDCOUNT;" times"
41 410 '
42 420 FAST       = &H8D: FASTEST    = &H8E: FASTSTOP   = &H8F
43 430 FASTER     = &H9A: FASTERSTOP = &H9B
44 440 FOCUSNEAR  = &H87: FOCUSFAR   = &H88: FOCUSSTOP  = &H89
45 450 IRISOPEN   = &H90: IRISCLOSE  = &H91: IRISSTOP   = &H92
46 460 PANLEFT    = &H81: PANRIGHT   = &H82: PANSTOP    = &H83
47 470 TILTUP     = &H84: TILTDOWN   = &H85: TILTSTOP   = &H86
48 480 ZOOMIN     = &H8A: ZOOMOUT    = &H8B: ZOOMSTOP   = &H8C
49 490 ONAIR      = &H9E: OFFAIR     = &H9F: RESETT     = &HC6
50 500 POLL       = &H94: STOPALL    = &H93: VARSPEED   = &HC0
51 510 '
52 520 CLS: GOSUB 1790
53 530 TIMESTEPSPERMS = 0: GOSUB 2000
54 540 COMMANDTYPE$ = "On ":       COMMAND = ONAIR:    GOSUB 1440: REM Turn it on
55 550 SPEED = 1:
56 560 STARTPOLL = 1: LASTPOLL = 58: SOMECOMMAND = YES
57 570 BYTE1 = 99: BYTE2 = 99
58 580 PRINT #3,
59 590 '
```

```
600 REM All commands return here and look for more to do
610 REM Send a poll command about once a second
620 IF TIME$ <> LASTSECOND$ THEN GOSUB 1580
630 LASTSECOND$ = TIME$
640 IF TIME$ <> LASTSECOND$ THEN GOSUB 1580
650 KPAD$ = INKEY$
660 IF LEN(KPAD$) = 0 THEN 640
670 PRINT " ";
680 IF ASC(KPAD$) >= ASC("A") AND ASC(KPAD$) <= ASC("Z") THEN 690 ELSE 710
690 KPAD$ = STR$(ASC(KPAD$) - 32)
700 '
710 REM Decode what just came in
720 IF KPAD$ = " " THEN ID1$ = "StopAll":    GOTO 1250
730 IF KPAD$ = "?" THEN 1750: REM Show help
740 IF KPAD$ = "/" THEN 1750: REM Show help
750 IF KPAD$ = "\" THEN ID1$ = "Reset":      GOTO 1250
760 IF KPAD$ = "q" THEN ID1$ = "StopAll":    GOTO 1250
770 IF KPAD$ = "x" THEN 1180: REM Exit
780 '
790 IF KPAD$ = "a" THEN ID1$ = "Rail Left":  GOTO 1250
800 IF KPAD$ = "f" THEN ID1$ = "Fast":       GOTO 1250
810 IF KPAD$ = "g" THEN ID1$ = "Fast Up":    GOTO 1250
820 IF KPAD$ = "h" THEN ID1$ = "Rail Right": GOTO 1250
830 IF KPAD$ = "j" THEN ID1$ = "Rail Up":    GOTO 1250
840 IF KPAD$ = "l" THEN ID1$ = "Left":       GOTO 1250
850 IF KPAD$ = "d" THEN ID1$ = "Down":       GOTO 1250
860 IF KPAD$ = "p" THEN ID1$ = "Pan Stop":   GOTO 1250
870 IF KPAD$ = "r" THEN ID1$ = "Right":      GOTO 1250
880 IF KPAD$ = "s" THEN ID1$ = "Rail Up":    GOTO 1250
890 IF KPAD$ = "t" THEN ID1$ = "Tilt Stop":  GOTO 1250
900 IF KPAD$ = "u" THEN ID1$ = "Up":         GOTO 1250
910 '
920 REM Processing for numeric inputs
930 X = ASC(KPAD$) - 48
940 IF X < 0 OR X > 9 THEN 980
950 IF BYTE1 = 99 THEN BYTE1 = X: GOTO 600
960 IF BYTE2 = 99 THEN BYTE2 = X: GOTO 1120
970 REM Processing for keypad, multibyte, inputs
980 IF LEN(KPAD$) = 1 THEN INPUTKEY = ASC(KPAD$): GOTO 1080
990 IF LEN(KPAD$) <> 2 THEN 600
1000 INPUTKEY = ASC(MID$(KPAD$,2,1))
1010 '
1020 IF INPUTKEY = &H48 THEN ID1$ = "Up":    GOTO 1250: REM Up
1030 IF INPUTKEY = &H4B THEN ID1$ = "Left":  GOTO 1250: REM Left
1040 IF INPUTKEY = &H4D THEN ID1$ = "Right": GOTO 1250: REM Right
1050 IF INPUTKEY = &H50 THEN ID1$ = "Down":  GOTO 1250: REM Down
1060 '
1070 REM Processing for undefined inputs
1080 PRINT "'0x";HEX$(INPUTKEY);"', ";
1090 PRINT #3,TIME$," '0x";HEX$(INPUTKEY);"', "
1100 GOTO 600
1110 '
1120 REM Send an arbitrary speed
1130 SPEED = (BYTE1*10) + BYTE2
1140 BYTE1 = 99: BYTE2 = 99
1150 PRINT "New speed =";SPEED;" ";
1160 GOTO 600
1170 '
1180 REM All done
1190 PRINT #3,"All done"
```

```
120 1200 PRINT #3,DATE$,TIME$
121 1210 PRINT
122 1220 PRINT "Quitting"
123 1230 END
124 1240 '
125 1250 REM Send a single motion starting command
126 1260 IF ID1$ = "Up"         THEN COMMAND = TILTUP
127 1270 IF ID1$ = "Down"       THEN COMMAND = TILTDOWN
128 1280 IF ID1$ = "Right"      THEN COMMAND = PANRIGHT
129 1290 IF ID1$ = "Left"       THEN COMMAND = PANLEFT
130 1300 '
131 1310 IF ID1$ = "Fast"       THEN COMMAND = FASTER
132 1320 IF ID1$ = "Fast Up"    THEN COMMAND = FASTERSTOP
133 1330 IF ID1$ = "Rail Left"  THEN COMMAND = FAST
134 1340 IF ID1$ = "Rail Right" THEN COMMAND = FASTEST
135 1350 IF ID1$ = "Rail Up"    THEN COMMAND = FASTSTOP
136 1360 '
137 1370 IF ID1$ = "Tilt Stop"  THEN COMMAND = TILTSTOP
138 1380 IF ID1$ = "Pan Stop"   THEN COMMAND = PANSTOP
139 1390 IF ID1$ = "StopAll"    THEN COMMAND = STOPALL
140 1400 '
141 1410 COMMANDTYPE$ = ID1$: GOSUB 1440
142 1420 GOTO 600
143 1430 '
144 1440 REM Send a command
145 1450 PRINT COMMANDTYPE$;",";: SOMECOMMAND = YES
146 1460 '
147 1470 CHECKSUM = &H100 - (ADDRESS + COMMAND)
148 1480 PRINT #3,USING"& \         \: ";TIME$,COMMANDTYPE$;
149 1490 PRINT #3,USING"0x\\ 0x\\ ";HEX$(ADDRESS),HEX$(COMMAND);
150 1500 PRINT #3,USING"0x\\";HEX$(CHECKSUM)
151 1510 IF REAL = NO THEN 1560
152 1520 FOR I = 1 TO RESENDCOUNT
153 1530 PRINT #2,CHR$(ADDRESS);CHR$(COMMAND);CHR$(CHECKSUM);
154 1540 GOSUB 2000
155 1550 NEXT I
156 1560 RETURN
157 1570 '
158 1580 REM Send a poll command about once a second
159 1590 LASTSECOND$ = TIME$
160 1600 THISPOLL = THISPOLL + 1
161 1610 IF THISPOLL > LASTPOLL THEN THISPOLL = STARTPOLL
162 1620 IF THISPOLL <> 1 THEN 1650
163 1630 PRINT ".";: IF SOMECOMMAND = YES THEN PRINT
164 1640 SOMECOMMAND = NO
165 1650 CHECKSUM = &H100 - (THISPOLL + POLL)
166 1660 IF POLLIT = NO OR REAL = NO THEN 1730
167 1670 PRINT #3,USING"& 0x\\,\         \: ";TIME$,HEX$(POLL),"Poll";
168 1680 PRINT #3,USING"### ### ###";THISPOLL,POLL,CHECKSUM
169 1690 FOR I = 1 TO RESENDCOUNT
170 1700 GOSUB 2000
171 1710 PRINT #2,CHR$(THISPOLL);CHR$(POLL);CHR$(CHECKSUM);
172 1720 NEXT I
173 1730 RETURN
174 1740 '
175 1750 REM Help Caller
176 1760 GOSUB 1790
177 1770 GOTO 600
178 1780 '
179 1790 REM Help data
```

```
180 1800 PRINT: PRINT
181 1810 PRINT "Stop most operations with a 'blank', exit with an 'x', ? = this"
182 1820 '
183 1830 PRINT "Dome addres is:";ADDRESS;" Active keys are:"
184 1840 PRINT "+------+------+------ +  A   S   D     F     G   H   J"
185 1850 PRINT "| ---  | Up   | ---   |  RL  Rs  Down  Fast  Fs  RR  Rs"
186 1860 REM      47     48     49
187 1870 PRINT "+------+------+-------+"
188 1880 PRINT "| Left | ---  | Right |"
189 1890 REM      4B            4D
190 1900 PRINT "+------+------+------ +  L     U   D     R      P         T"
191 1910 PRINT "| ---  | Down | ---   |  Left  Up  Down  Right  Pan Stop  Tilt Stop"
192 1920 REM      4F     50     51
193 1930 PRINT "+------+------+------ +"
194 1940 PRINT "| ---         | ---   |"
195 1950 REM      52            53
196 1960 PRINT "+-------------+------ +  Stop q = all, ' ' = all"
197 1970 PRINT
198 1980 RETURN
199 1990 '
200 2000 REM Delay long enough for the command to go out
201 2010 FOR K = 1 TO 30
202 2020 GOSUB 2060
203 2030 NEXT K
204 2040 RETURN
205 2050 '
206 2060 REM Delay a short while
207 2070 IF TIMESTEPSPERMS = 0 THEN GOSUB 2190
208 2080 FOR J = 1 TO TIMESTEPSPERMS
209 2090 ' GOSUB 2830
210 2100 NEXT J
211 2110 RETURN
212 2120 '
213 2130 REM Don't change anything in the "Timer thing", including comments as
214 2140 REM this may change the timing values it generates to an unusable value.
215 2150 REM Timer thing
216 2160 REM X = x + 1: ' 37 / 17
217 2170 RETURN
218 2180 '
219 2190 REM Calibrate the timer so as to get steps/millieseconds
220 2200 PRINT "Calibrating the loop timer, takes 1 to 3 seconds"
221 2210 X$ = TIME$
222 2220 IF X$ = TIME$ GOTO 2210
223 2230 X$ = TIME$
224 2240 GOSUB 2150
225 2250 TIMESTEPSPERMS = TIMESTEPSPERMS + 1
226 2260 IF X$ = TIME$ GOTO 2240
227 2270 TIMESTEPSPERMS = TIMESTEPSPERMS / 100
228 2280 PRINT USING "Using ### steps to give a one ms delay";TIMESTEPSPERMS
229 2290 PRINT #3,USING "Using ### steps to give a one ms delay";TIMESTEPSPERMS
230 2300 RETURN
```

1.5 SEQTEST.BAS

```
 1 10 CLS: REV$ = "$Header: d:/sears/RCS/seqtest.bas,v 1.10 2001-12-14 10:42:39-08 Hamilton Exp Hamilton $"
 2 20 PRINT REV$
 3 30 PRINT "Sequentially send all possiable commands to a dome. There are "
 4 40 PRINT "three modes of operation: "
 5 50 PRINT "1. Send one, incremented, command each time the 'space bar' is "
 6 60 PRINT "hit. "
 7 70 PRINT "2. Send a random command each time the  'space bar' is hit. "
 8 80 PRINT "3. Send one, decremented, command each time the 'space bar' is "
 9 90 PRINT "hit. "
10 100 PRINT
11 110 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
12 120 PRINT "or this program will time out when the serial communications is opened."
13 130 PRINT "(With a 9 pin connector these are 7,8 and 6,1,4.)"
14 140 PRINT "(With a 25 pin connector these are 4,5 and 6,8,20.)"
15 150 PRINT
16 160 YES = 1: NO = 0
17 170 '
18 180 OPEN "status.log" FOR APPEND AS #3
19 190 PRINT #3,: PRINT #3,REV$
20 200 PRINT #3,DATE$,TIME$
21 210 '
22 220 INPUT "For this run should we have real outputs (cr = yes)";X$
23 230 IF X$ = "" THEN REAL = YES ELSE REAL = NO
24 240 IF REAL = NO THEN 340
25 250 PRINT "COM1 = DB-25, COM2 = DB-9"
26 260 INPUT "Use which COM port for IO (cr = COM1, anything else is COM2)";X$
27 270 IF X$ = "" THEN PORT$ = "COM1" ELSE PORT$ = "COM2"
28 280 OPENSTRING$ = PORT$ + ":4800,N,8,1"
29 290 OPEN OPENSTRING$ AS #2
30 300 PRINT #3,"The IO setup string was '";OPENSTRING$;"'"
31 310 PRINT #3,"This run writes data out to a dome"
32 320 GOTO 360
33 330 '
34 340 PRINT #3,"This is a simulation run that does not write data out"
35 350 '
36 360 INPUT "Should polling be used (cr = no)";X$
37 370 IF X$ = "" THEN POLLIT = NO ELSE POLLIT = YES
38 380 '
39 390 INPUT "Use what for a camera address (cr = 24)";ADDRESS
40 400 IF ADDRESS = 0 THEN ADDRESS = 24
41 410 PRINT "Using a camera address of ";ADDRESS
42 420 PRINT #3,"Writing to camera ";ADDRESS
43 430 '
44 440 PRINT "Operating modes are:"
45 450 PRINT "1 = Incrementing sequential"
46 460 PRINT "2 = Random"
47 470 PRINT "3 = Decrementing sequential"
48 480 INPUT "What mode should this be run in (cr = 1)";OPMODE
49 490 IF OPMODE = 0 THEN OPMODE = 1
50 500 IF OPMODE = 1 THEN PRINT #3,"Incrementing sequential command mode": GOTO 550
51 510 IF OPMODE = 2 THEN PRINT #3,"Random command mode": GOTO 550
52 520 IF OPMODE = 3 THEN PRINT #3,"Decrementing sequential command mode": GOTO 550
53 530 BEEP: PRINT "An invalid mode was selected, try again": GOTO 440
54 540 '
55 550 INPUT "Time in between commands (cr = 1 second)";TIMEBETWEEN
56 560 IF TIMEBETWEEN = 0 THEN TIMEBETWEEN = 1
57 570 IF TIMEBETWEEN > 9 THEN PRINT "Time is too long, try again": GOTO 550
58 580 '
59 590 FASTSTOP   = &H8F: FASTERSTOP = &H9B: FOCUSSTOP  = &H89
```

```
600 IRISSTOP   = &H92: PANSTOP    = &H83: TILTSTOP   = &H86
610 ZOOMSTOP   = &H8C: STOPALL    = &H93
620 '
630 IF OPMODE = 1 THEN THISCOMMAND = -1
640 IF OPMODE = 2 THEN GOSUB 1320
650 IF OPMODE = 3 THEN THISCOMMAND = 0
660 CLS
670 CONTINOUSMODE = NO
680 PRINT #3,
690 '
700 GOSUB 1380
710 PRINT "'X' will exit the program"
720 PRINT "'C' causes continous operation"
730 PRINT "'S' stops continous operation"
740 PRINT "' ' for sending a single command"
750 PRINT "'digit' for changing the delay between commands"
760 PRINT "'.' or max speed in continous mode"
770 '
780 REM Each cycle start here
790 KPAD$ = INKEY$
800 IF LEN(KPAD$) = 0 AND CONTINOUSMODE = NO THEN 790
810 IF LEN(KPAD$) <> 0 THEN 830
820 IF CONTINOUSMODE = YES THEN 990
830 IF ASC(KPAD$) >= ASC("A") AND ASC(KPAD$) <= ASC("Z") THEN 840 ELSE 850
840 KPAD$ = STR$(ASC(KPAD$) - 32)
850 REM Now decode that input
860 IF KPAD$ = "." THEN TIMEBETWEEN = .1: GOTO 1020
870 IF KPAD$ = "1" THEN TIMEBETWEEN = 1: GOTO 1020
880 IF KPAD$ = "2" THEN TIMEBETWEEN = 2: GOTO 1020
890 IF KPAD$ = "3" THEN TIMEBETWEEN = 3: GOTO 1020
900 IF KPAD$ = "4" THEN TIMEBETWEEN = 4: GOTO 1020
910 IF KPAD$ = "5" THEN TIMEBETWEEN = 5: GOTO 1020
920 IF KPAD$ = "6" THEN TIMEBETWEEN = 6: GOTO 1020
930 IF KPAD$ = "7" THEN TIMEBETWEEN = 7: GOTO 1020
940 IF KPAD$ = "8" THEN TIMEBETWEEN = 8: GOTO 1020
950 IF KPAD$ = "9" THEN TIMEBETWEEN = 9: GOTO 1020
960 IF KPAD$ = "x" OR KPAD$ = "q" THEN 1050:        REM exit
970 IF KPAD$ = "c" THEN CONTINOUSMODE = YES: PRINT: REM Enable continous sending
980 IF KPAD$ = "s" THEN CONTINOUSMODE = NO:  PRINT: REM Stop continous sending
990 IF KPAD$ = " " OR CONTINOUSMODE = YES THEN GOSUB 1120
1000 GOTO 780
1010 '
1020 PRINT: PRINT "New time between commands is: ";TIMEBETWEEN;" seconds"
1030 GOTO 1000
1040 '
1050 REM All done
1060 PRINT #3,"All done"
1070 PRINT #3,DATE$,TIME$
1080 PRINT
1090 PRINT "Quitting"
1100 END
1110 '
1120 REM Send a command
1130 IF OPMODE = 1 THEN THISCOMMAND = THISCOMMAND + 1
1140 IF OPMODE = 3 THEN THISCOMMAND = THISCOMMAND - 1
1150 IF ((THISCOMMAND AND &HF) = 0) AND (OPMODE = 1) THEN PRINT
1160 IF ((THISCOMMAND AND &HF) = F) AND (OPMODE = 3) THEN PRINT
1170 IF OPMODE = 2 THEN GOSUB 1320
1180 COMMAND = THISCOMMAND AND &HFF
1190 IF COMMAND = &HA4 THEN 1120
```

```
120 1200 CHECKSUM = &HFF AND (&H100 - (ADDRESS + COMMAND))
121 1210 PRINT #3,USING"& ";TIME$;
122 1220 PRINT #3,USING"0x\\ 0x\\ 0x\\";HEX$(ADDRESS),HEX$(COMMAND),HEX$(CHECKSUM)
123 1230 PRINT USING "\\  ";HEX$(COMMAND);
124 1240 IF REAL = NO THEN 1260
125 1250 PRINT #2,CHR$(ADDRESS);CHR$(COMMAND);CHR$(CHECKSUM);
126 1260 GOSUB 1380
127 1270 ENDSECOND = TOTALSECONDS + TIMEBETWEEN
128 1280 IF TIMEBETWEEN > .5 THEN GOSUB 1380 ELSE GOTO 1300
129 1290 IF ENDSECOND > TOTALSECONDS THEN 1280
130 1300 RETURN
131 1310 '
132 1320 REM Generate a random command
133 1330 THISCOMMAND = INT(RND * 257)
134 1340 IF THISCOMMAND > 256 THEN THISCOMMAND = THISCOMMAND - 256
135 1350 IF THISCOMMAND = &HA4 THEN 1330
136 1360 RETURN
137 1370 '
138 1380 REM Get total seconds - 200, when real time is greater than 500.
139 1390 REM This is so that we never have end of day turn over problems.
140 1400 THISTIME$ = TIME$
141 1410 HH = VAL(MID$(THISTIME$,1,2)) * 3600
142 1420 MM = VAL(MID$(THISTIME$,4,2)) * 60
143 1430 SS = VAL(MID$(THISTIME$,7,2))
144 1440 TOTALSECONDS = HH + MM + SS
145 1450 IF TOTALSECONDS > 500 THEN TOTALSECONDS = TOTALSECONDS - 200
146 1460 RETURN
```

1.6 SPAT.BAS

```
10 CLS: REV$ = "$Header: d:/sears/RCS/spat.bas,v 1.6 2001-11-30 11:37:43-08 Hamilton Exp Hamilton $"
20 PRINT REV$
30 PRINT "Send a pattern command sequence to a dome"
40 PRINT
50 PRINT "Active keys are:"
60 PRINT "'S' = Start"
70 PRINT "'X' = Exit"
80 PRINT
90 PRINT "Generate a sequence of pattern commands to a Sensormatic Dome"
100 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
110 PRINT "or this program will time out when COM1 is opened."
120 PRINT
130 '
140 OPEN "SPAT.LOG" FOR APPEND AS #3
150 PRINT #3,
160 PRINT #3,REV$
170 PRINT #3,DATE$,TIME$
180 PRINT #3,
190 GOSUB 1340: REM Get dome address
200 STARTADDRESS = 1
210 ENDADDRESS = 99
220 SKIPADDRESS = ADDRESS%
230 THISADDRESS = STARTADDRESS
240 NOPOLL = 0
250 '
260 OPEN "COM1:4800,N,8,1" AS #2
270 CMNDCOUNT% = 0
280 '
290 REM Start sending commands out
300 '
310 PRINT "Hit 'S' to start the pattern programming going:"
320 KPAD$ = INKEY$
330 IF LEN(KPAD$) = 0 THEN 320
340 IF KPAD$ = "x" OR KPAD$ = "X" THEN 1290
350 IF KPAD$ = "s" OR KPAD$ = "S" THEN 390
360 REM PRINT ",";LEN(KPAD$);", ";
370 GOTO 320
380 '
390 REM Start sending a pattern to the dome
400 TIMEOUT = 1: BYTE2% = &HC9: GOSUB 1490: REM Get software version
410 TIMEOUT = 10: BYTE2% = &H8B: GOSUB 1490: REM Zoom Out
420 TIMEOUT = 10: BYTE2% = &H8C: GOSUB 1490: REM Stop Zoom
430 TIMEOUT = 1: GOSUB 1700
440 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1490: REM Poll
450 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1490: REM Poll
460 TIMEOUT = 1: BYTE2% = &H9E: GOSUB 1490: REM On Air
470 TIMEOUT = 1: BYTE2% = &H80: GOSUB 1490: REM Unknown 80
480 TIMEOUT = 0: BYTE2% = &HA0: GOSUB 1490: REM Define Pattern 1
490 TIMEOUT = 1: BYTE2% = &H97: GOSUB 1490: REM Ack dome response
500 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
510 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
520 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
530 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
540 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
550 TIMEOUT = 1: BYTE2% = &H81: GOSUB 1490: REM Pan Left
560 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
570 TIMEOUT = 1: BYTE2% = &HB8: GOSUB 1490: REM Pattern end
580 TIMEOUT = 1: BYTE2% = &HB0: GOSUB 1490: REM Run Pattern 1
590 TIMEOUT = 6: GOSUB 1700
```

```
 60 600 TIMEOUT = 1: BYTE2% = &HA3: GOSUB 1490: REM Accept New Pattern
 61 610 TIMEOUT = 1: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
 62 620 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
 63 630 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
 64 640 TIMEOUT = 0: BYTE2% = &HA4: GOSUB 1490: REM Unknown A4
 65 650 TIMEOUT = 5: NOPOLL = 1: GOSUB 1750: REM 5 second delay with no polling
 66 660 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
 67 670 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
 68 680 TIMEOUT = 0: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
 69 690 TIMEOUT = 1: BYTE2% = &H93: GOSUB 1490: REM Stop All
 70 700 '
 71 710 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1490: REM Poll
 72 720 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1490: REM Poll
 73 730 TIMEOUT = 1: BYTE2% = &H9E: GOSUB 1490: REM On Air
 74 740 TIMEOUT = 1: BYTE2% = &H80: GOSUB 1490: REM Unknown 80
 75 750 TIMEOUT = 0: BYTE2% = &HA1: GOSUB 1490: REM Define Pattern 2
 76 760 TIMEOUT = 1: BYTE2% = &H97: GOSUB 1490: REM Ack dome response
 77 770 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
 78 780 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
 79 790 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
 80 800 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
 81 810 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
 82 820 TIMEOUT = 1: BYTE2% = &H81: GOSUB 1490: REM Pan Left
 83 830 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
 84 840 TIMEOUT = 1: BYTE2% = &HB8: GOSUB 1490: REM Pattern end
 85 850 TIMEOUT = 1: BYTE2% = &HB1: GOSUB 1490: REM Run Pattern 2
 86 860 TIMEOUT = 6: GOSUB 1700
 87 870 TIMEOUT = 1: BYTE2% = &HA3: GOSUB 1490: REM Accept New Pattern
 88 880 TIMEOUT = 1: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
 89 890 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
 90 900 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
 91 910 TIMEOUT = 0: BYTE2% = &HA4: GOSUB 1490: REM Unknown A4
 92 920 TIMEOUT = 5: NOPOLL = 1: GOSUB 1750: REM 5 second delay with no polling
 93 930 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
 94 940 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
 95 950 TIMEOUT = 0: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
 96 960 TIMEOUT = 1: BYTE2% = &H93: GOSUB 1490: REM Stop All
 97 970 '
 98 980 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1490: REM Poll
 99 990 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1490: REM Poll
100 1000 TIMEOUT = 1: BYTE2% = &H9E: GOSUB 1490: REM On Air
101 1010 TIMEOUT = 1: BYTE2% = &H80: GOSUB 1490: REM Unknown 80
102 1020 TIMEOUT = 0: BYTE2% = &HA2: GOSUB 1490: REM Define Pattern 3
103 1030 TIMEOUT = 1: BYTE2% = &H97: GOSUB 1490: REM Ack dome response
104 1040 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
105 1050 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
106 1060 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
107 1070 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
108 1080 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1490: REM Pan Right
109 1090 TIMEOUT = 1: BYTE2% = &H81: GOSUB 1490: REM Pan Left
110 1100 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1490: REM Pan Stop
111 1110 TIMEOUT = 1: BYTE2% = &HB8: GOSUB 1490: REM Pattern end
112 1120 TIMEOUT = 1: BYTE2% = &HB2: GOSUB 1490: REM Run Pattern 3
113 1130 TIMEOUT = 6: GOSUB 1700
114 1140 TIMEOUT = 1: BYTE2% = &HA3: GOSUB 1490: REM Accept New Pattern
115 1150 TIMEOUT = 1: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
116 1160 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
117 1170 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1490: REM Unknown 98
118 1180 TIMEOUT = 0: BYTE2% = &HA4: GOSUB 1490: REM Unknown A4
119 1190 TIMEOUT = 5: NOPOLL = 1: GOSUB 1750: REM 5 second delay with no polling
```

```
120 1200 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
121 1210 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
122 1220 TIMEOUT = 0: BYTE2% = &H99: GOSUB 1490: REM Unknown 99
123 1230 TIMEOUT = 1: BYTE2% = &H93: GOSUB 1490: REM Stop All
124 1240 '
125 1250 PRINT "Pattern setting is done": BEEP
126 1260 PRINT #3,"Pattern setting is done"
127 1270 GOTO 310
128 1280 '
129 1290 REM All done
130 1300 PRINT "All done"
131 1310 PRINT DATE$,TIME$
132 1320 END
133 1330 '
134 1340 REM Ask for a camera address
135 1350 REM Note that camera addresses are one based
136 1360 INPUT "What camera address should be used (cr = camera 6)";ADDRESS%
137 1370 IF ADDRESS% = 0 THEN ADDRESS% = 6
138 1380 IF ADDRESS% > 99 OR ADDRESS% < 1 THEN 1360
139 1390 PRINT "Using a camera address of:";ADDRESS%
140 1400 PRINT #3,"Using a camera address of: ";ADDRESS%
141 1410 PRINT #3,
142 1420 RETURN
143 1430 '
144 1440 REM Calculate a checksum
145 1450 CHECKSUM% = &H100 - (ADDRESS% + BYTE2%)
146 1460 CHECKSUM% = CHECKSUM% AND &HFF
147 1470 RETURN
148 1480 '
149 1490 REM Send a command and then delay some
150 1500 GOSUB 1700: REM Delay
151 1510 GOSUB 1540: REM Send it
152 1520 RETURN
153 1530 '
154 1540 REM Send a command
155 1550 REM
156 1560 REM BYTE2% is what changes and gets sent out
157 1570 REM
158 1580 IF BYTE2% = &H98 OR BYTE2% = &H99 THEN SAVEDADDRESS% = ADDRESS%: ADDRESS% = &H40
159 1590 GOSUB 1440: REM Do checksum
160 1600 PRINT #3, USING "##,### &: ";COMMANDCOUNT+1;ID$;
161 1610 COMMANDCOUNT = COMMANDCOUNT + 1
162 1620 X% = ADDRESS%:  GOSUB 1930: ADDRESS$ = X$
163 1630 X% = BYTE2%:    GOSUB 1930: BYTE2$ = X$
164 1640 X% = CHECKSUM%: GOSUB 1930: CHECKSUM$ = X$
165 1650 PRINT #3, USING "0x& 0x& 0x& &";ADDRESS$;BYTE2$;CHECKSUM$;TIME$
166 1660 PRINT #2,CHR$(ADDRESS%);CHR$(BYTE2%);CHR$(CHECKSUM%);
167 1670 IF BYTE2% = &H98 OR BYTE2% = &H99 THEN ADDRESS% = SAVEDADDRESS%
168 1680 RETURN
169 1690 '
170 1700 REM Delay a variable number of seconds
171 1710 REM TIMEOUT = number of seconds to wait,
172 1720 REM           zero (or less) gives an immediate exit
173 1730 REM NOPOLL  = do not poll during this timeout
174 1740 REM
175 1750 IF TIMEOUT < 1 THEN 1910
176 1760 Y$ = MID$(TIME$,7,2)
177 1770 IF MID$(TIME$,7,2) = Y$ THEN 1770
178 1780 TIMEOUT = TIMEOUT - 1
179 1790 IF NOPOLL = 1 THEN 1890
```

```
180 1800 DELAYSAVEDADDRESS% = ADDRESS%
181 1810 THISADDRESS = THISADDRESS + 1
182 1820 IF THISADDRESS > ENDADDRESS THEN THISADDRESS = STARTADDRESS
183 1830 IF THISADDRESS = SKIPADDRESS THEN THISADDRESS = THISADDRESS + 1
184 1840 IF THISADDRESS > ENDADDRESS THEN THISADDRESS = STARTADDRESS
185 1850 ADDRESS% = THISADDRESS
186 1860 BYTE2% = &H94: REM Send a poll to get a timer going for ID1
187 1870 GOSUB 1540
188 1880 ADDRESS% = DELAYSAVEDADDRESS%
189 1890 IF TIMEOUT <> 0 THEN 1760
190 1900 NOPOLL = 0
191 1910 RETURN
192 1920 '
193 1930 REM Convert an integer into ascii-hex
194 1940 X$ = HEX$(X%)
195 1950 IF LEN(X$) <>2 THEN X$ = "0" + X$
196 1960 RETURN
```

1.7 SPATD.BAS

```
10 CLS: REV$ = "$Header: d:/sears/RCS/spatd.bas,v 1.2 2001-11-30 11:38:13-08 Hamilton Exp Hamilton $"
20 PRINT REV$
30 PRINT "Send a pattern command sequence to a dome"
40 PRINT "Request a memory dump before and after."
50 PRINT
60 PRINT "Active keys are:"
70 PRINT "'S' = Start"
80 PRINT "'X' = Exit"
90 PRINT
100 PRINT "Generate a sequence of pattern commands to a Sensormatic Dome"
110 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
120 PRINT "or this program will time out when COM1 is opened."
130 PRINT
140 '
150 OPEN "SPAT.LOG" FOR APPEND AS #3
160 PRINT #3,
170 PRINT #3,REV$
180 PRINT #3,DATE$,TIME$
190 PRINT #3,
200 GOSUB 1400: REM Get dome address
210 STARTADDRESS = 1
220 ENDADDRESS = 99
230 SKIPADDRESS = ADDRESS%
240 THISADDRESS = STARTADDRESS
250 NOPOLL = 0
260 '
270 OPEN "COM1:4800,N,8,1" AS #2
280 CMNDCOUNT% = 0
290 '
300 REM Start sending commands out
310 '
320 PRINT "Hit 'S' to start the pattern programming going:"
330 KPAD$ = INKEY$
340 IF LEN(KPAD$) = 0 THEN 330
350 IF KPAD$ = "x" OR KPAD$ = "X" THEN 1350
360 IF KPAD$ = "s" OR KPAD$ = "S" THEN 410
370 REM PRINT ",";LEN(KPAD$);", ";
380 PRINT KPAD$;
390 GOTO 330
400 '
410 REM Start sending a pattern to the dome
420 PRINT "Starting"
430 TIMEOUT = 2: BYTE2% = &HA4: GOSUB 1550; REM Dump memory
440 TIMEOUT = 1: BYTE2% = &HC9: GOSUB 1550: REM Get software version
450 TIMEOUT = 10: BYTE2% = &H8B: GOSUB 1550: REM Zoom Out
460 TIMEOUT = 10: BYTE2% = &H8C: GOSUB 1550: REM Stop Zoom
470 TIMEOUT = 1: GOSUB 1760
480 TIMEOUT = 2: BYTE2% = &HA4: GOSUB 1550; REM Dump memory
490 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1550: REM Poll
500 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1550: REM Poll
510 TIMEOUT = 1: BYTE2% = &H9E: GOSUB 1550: REM On Air
520 TIMEOUT = 1: BYTE2% = &H80: GOSUB 1550: REM Unknown 80
530 TIMEOUT = 0: BYTE2% = &HA0: GOSUB 1550: REM Define Pattern 1
540 TIMEOUT = 1: BYTE2% = &H97: GOSUB 1550: REM Ack dome response
550 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
560 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
570 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
580 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
590 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
```

```
 60 600 TIMEOUT = 1: BYTE2% = &H81: GOSUB 1550: REM Pan Left
 61 610 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
 62 620 TIMEOUT = 1: BYTE2% = &HB8: GOSUB 1550: REM Pattern end
 63 630 TIMEOUT = 1: BYTE2% = &HB0: GOSUB 1550: REM Run Pattern 1
 64 640 TIMEOUT = 6: GOSUB 1760
 65 650 TIMEOUT = 1: BYTE2% = &HA3: GOSUB 1550: REM Accept New Pattern
 66 660 TIMEOUT = 2: BYTE2% = &HA4: GOSUB 1550; REM Dump memory
 67 670 TIMEOUT = 1: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
 68 680 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
 69 690 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
 70 700 TIMEOUT = 0: BYTE2% = &HA4: GOSUB 1550: REM Unknown A4
 71 710 TIMEOUT = 5: NOPOLL = 1: GOSUB 1810: REM 5 second delay with no polling
 72 720 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
 73 730 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
 74 740 TIMEOUT = 0: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
 75 750 TIMEOUT = 1: BYTE2% = &H93: GOSUB 1550: REM Stop All
 76 760 '
 77 770 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1550: REM Poll
 78 780 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1550: REM Poll
 79 790 TIMEOUT = 1: BYTE2% = &H9E: GOSUB 1550: REM On Air
 80 800 TIMEOUT = 1: BYTE2% = &H80: GOSUB 1550: REM Unknown 80
 81 810 TIMEOUT = 0: BYTE2% = &HA1: GOSUB 1550: REM Define Pattern 2
 82 820 TIMEOUT = 1: BYTE2% = &H97: GOSUB 1550: REM Ack dome response
 83 830 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
 84 840 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
 85 850 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
 86 860 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
 87 870 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
 88 880 TIMEOUT = 1: BYTE2% = &H81: GOSUB 1550: REM Pan Left
 89 890 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
 90 900 TIMEOUT = 1: BYTE2% = &HB8: GOSUB 1550: REM Pattern end
 91 910 TIMEOUT = 1: BYTE2% = &HB1: GOSUB 1550: REM Run Pattern 2
 92 920 TIMEOUT = 6: GOSUB 1760
 93 930 TIMEOUT = 1: BYTE2% = &HA3: GOSUB 1550: REM Accept New Pattern
 94 940 TIMEOUT = 1: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
 95 950 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
 96 960 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
 97 970 TIMEOUT = 0: BYTE2% = &HA4: GOSUB 1550: REM Unknown A4
 98 980 TIMEOUT = 5: NOPOLL = 1: GOSUB 1810: REM 5 second delay with no polling
 99 990 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
100 1000 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
101 1010 TIMEOUT = 0: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
102 1020 TIMEOUT = 1: BYTE2% = &H93: GOSUB 1550: REM Stop All
103 1030 '
104 1040 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1550: REM Poll
105 1050 TIMEOUT = 1: BYTE2% = &H94: GOSUB 1550: REM Poll
106 1060 TIMEOUT = 1: BYTE2% = &H9E: GOSUB 1550: REM On Air
107 1070 TIMEOUT = 1: BYTE2% = &H80: GOSUB 1550: REM Unknown 80
108 1080 TIMEOUT = 0: BYTE2% = &HA2: GOSUB 1550: REM Define Pattern 3
109 1090 TIMEOUT = 1: BYTE2% = &H97: GOSUB 1550: REM Ack dome response
110 1100 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
111 1110 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
112 1120 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
113 1130 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
114 1140 TIMEOUT = 1: BYTE2% = &H82: GOSUB 1550: REM Pan Right
115 1150 TIMEOUT = 1: BYTE2% = &H81: GOSUB 1550: REM Pan Left
116 1160 TIMEOUT = 1: BYTE2% = &H83: GOSUB 1550: REM Pan Stop
117 1170 TIMEOUT = 1: BYTE2% = &HB8: GOSUB 1550: REM Pattern end
118 1180 TIMEOUT = 1: BYTE2% = &HB2: GOSUB 1550: REM Run Pattern 3
119 1190 TIMEOUT = 6: GOSUB 1760
```

```
120 1200 TIMEOUT = 1: BYTE2% = &HA3: GOSUB 1550: REM Accept New Pattern
121 1210 TIMEOUT = 1: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
122 1220 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
123 1230 TIMEOUT = 0: BYTE2% = &H98: GOSUB 1550: REM Unknown 98
124 1240 TIMEOUT = 0: BYTE2% = &HA4: GOSUB 1550: REM Unknown A4
125 1250 TIMEOUT = 5: NOPOLL = 1: GOSUB 1810: REM 5 second delay with no polling
126 1260 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
127 1270 TIMEOUT = 0: NOPOLL = 1: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
128 1280 TIMEOUT = 0: BYTE2% = &H99: GOSUB 1550: REM Unknown 99
129 1290 TIMEOUT = 1: BYTE2% = &H93: GOSUB 1550: REM Stop All
130 1300 '
131 1310 PRINT "Pattern setting is done": BEEP
132 1320 PRINT #3,"Pattern setting is done"
133 1330 GOTO 320
134 1340 '
135 1350 REM All done
136 1360 PRINT "All done"
137 1370 PRINT DATE$,TIME$
138 1380 END
139 1390 '
140 1400 REM Ask for a camera address
141 1410 REM Note that camera addresses are one based
142 1420 INPUT "What camera address should be used (cr = camera 6)";ADDRESS%
143 1430 IF ADDRESS% = 0 THEN ADDRESS% = 6
144 1440 IF ADDRESS% > 99 OR ADDRESS% < 1 THEN 1420
145 1450 PRINT "Using a camera address of:";ADDRESS%
146 1460 PRINT #3,"Using a camera address of: ";ADDRESS%
147 1470 PRINT #3,
148 1480 RETURN
149 1490 '
150 1500 REM Calculate a checksum
151 1510 CHECKSUM% = &H100 - (ADDRESS% + BYTE2%)
152 1520 CHECKSUM% = CHECKSUM% AND &HFF
153 1530 RETURN
154 1540 '
155 1550 REM Send a command and then delay some
156 1560 GOSUB 1760: REM Delay
157 1570 GOSUB 1600: REM Send it
158 1580 RETURN
159 1590 '
160 1600 REM Send a command
161 1610 REM
162 1620 REM BYTE2% is what changes and gets sent out
163 1630 REM
164 1640 IF BYTE2% = &H98 OR BYTE2% = &H99 THEN SAVEDADDRESS% = ADDRESS%: ADDRESS% = &H40
165 1650 GOSUB 1500: REM Do checksum
166 1660 PRINT #3, USING "##,### &: ";COMMANDCOUNT+1;ID$;
167 1670 COMMANDCOUNT = COMMANDCOUNT + 1
168 1680 X% = ADDRESS%:  GOSUB 1990: ADDRESS$ = X$
169 1690 X% = BYTE2%:    GOSUB 1990: BYTE2$ = X$
170 1700 X% = CHECKSUM%: GOSUB 1990: CHECKSUM$ = X$
171 1710 PRINT #3, USING "0x& 0x& 0x& &";ADDRESS$;BYTE2$;CHECKSUM$;TIME$
172 1720 PRINT #2,CHR$(ADDRESS%);CHR$(BYTE2%);CHR$(CHECKSUM%);
173 1730 IF BYTE2% = &H98 OR BYTE2% = &H99 THEN ADDRESS% = SAVEDADDRESS%
174 1740 RETURN
175 1750 '
176 1760 REM Delay a variable number of seconds
177 1770 REM TIMEOUT = number of seconds to wait,
178 1780 REM           zero (or less) gives an immediate exit
179 1790 REM NOPOLL  = do not poll during this timeout
```

```
180 1800 REM
181 1810 IF TIMEOUT < 1 THEN 1970
182 1820 Y$ = MID$(TIME$,7,2)
183 1830 IF MID$(TIME$,7,2) = Y$ THEN 1830
184 1840 TIMEOUT = TIMEOUT - 1
185 1850 IF NOPOLL = 1 THEN 1950
186 1860 DELAYSAVEDADDRESS% = ADDRESS%
187 1870 THISADDRESS = THISADDRESS + 1
188 1880 IF THISADDRESS > ENDADDRESS THEN THISADDRESS = STARTADDRESS
189 1890 IF THISADDRESS = SKIPADDRESS THEN THISADDRESS = THISADDRESS + 1
190 1900 IF THISADDRESS > ENDADDRESS THEN THISADDRESS = STARTADDRESS
191 1910 ADDRESS% = THISADDRESS
192 1920 BYTE2% = &H94: REM Send a poll to get a timer going for ID1
193 1930 GOSUB 1600
194 1940 ADDRESS% = DELAYSAVEDADDRESS%
195 1950 IF TIMEOUT <> 0 THEN 1820
196 1960 NOPOLL = 0
197 1970 RETURN
198 1980 '
199 1990 REM Convert an integer into ascii-hex
200 2000 X$ = HEX$(X%)
201 2010 IF LEN(X$) <>2 THEN X$ = "0" + X$
202 2020 RETURN
```

1.8 SPROTO.BAS

```
10 CLS: REV$ = "$Header: d:/sears/RCS/sproto.bas,v 1.6 2001-04-02 16:04:13-07 Hamilton Exp Hamilton $"
20 PRINT REV$
30 PRINT
40 PRINT "Active keys are:": PRINT
50 PRINT "Control a Sensormatic dome via serial IO with keyboard commands"
60 PRINT "from the keypad, when not in 'num lock' mode."
70 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
80 PRINT "or this program will time out when COM1 is opened."
90 PRINT
100 '
110 REM Key pad keys generate the following data
120 '
130 REM 0 = 0x00 0x52 DomeType
140 REM 1 = 0x00 0x4F Faster
150 REM 2 = 0x00 0x50 Down
160 REM 3 = 0x00 0x51
170 REM 4 = 0x00 0x4B Left
180 REM 5 = 0x00 ---
190 REM 6 = 0x00 0x4D Right
200 REM 7 = 0x00 0x47 Fast
210 REM 8 = 0x00 0x48 Up
220 REM 9 = 0x00 0x49 Fastest
230 '
240 PRINT "Active keys are:": PRINT
250 PRINT "Fast   Up   Fastest"
260 PRINT "Left   ---  Right"
270 PRINT "Faster Down ---"
280 PRINT "DomeType    ---": PRINT
290 '
300 OPEN "SPROTO.LOG" FOR APPEND AS #3
310 PRINT #3,
320 PRINT #3,REV$
330 PRINT #3,DATE$,TIME$
340 PRINT #3,
350 GOSUB 1270: REM Get dome address
360 '
370 OPEN "COM1:4800,N,8,1" AS #2
380 CMNDCOUNT% = 0
390 '
400 REM Start sending commands out
410 '
420 KPAD$ = INKEY$
430 IF LEN(KPAD$) = 0 THEN 420
440 IF KPAD$ = "x" OR KPAD$ = "X" THEN 1220
450 REM PRINT ",";LEN(KPAD$);", ";
460 IF LEN(KPAD$) = 1 THEN 490: REM Single byte commands
470 IF LEN(KPAD$) = 2 THEN 850: REM Double byte commands
480 GOTO  420: REM Incorrect length, don't process at all
490 REM Process single byte commands
500 X$ = KPAD$
510 '
520 IF X$ = "T" OR X$ = "t" THEN BYTE2% = &H8A: ID$ = "T": GOTO 1160: REM Tele
530 IF X$ = "W" OR X$ = "w" THEN BYTE2% = &H8B: ID$ = "W": GOTO 1160: REM Wide
540 IF BYTE2% = &H8A OR BYTE2% = &H8B THEN ZOOMSTOP = 1
550 IF X$ = "F" OR X$ = "f" THEN BYTE2% = &H88: ID$ = "F": GOTO 1160: REM Far focus
560 IF X$ = "N" OR X$ = "n" THEN BYTE2% = &H87: ID$ = "N": GOTO 1160: REM Near focus
570 IF BYTE2% = &H88 OR BYTE2% = &H87 THEN FOCUSSTOP = 1
580 IF X$ = "O" OR X$ = "o" THEN BYTE2% = &H90: ID$ = "O": GOTO 1160: REM Iris open
590 IF X$ = "C" OR X$ = "c" THEN BYTE2% = &H91: ID$ = "C": GOTO 1160: REM Iris close
```

```
600 IF BYTE2% = &H90 OR BYTE2% = &H91 THEN IRISSTOP = 1
610 IF X$ = "S" OR X$ = "s" THEN BYTE2% = &H93: ID$ = "s": GOTO 1160: REM Stop all motion
620 IF X$ = "R" OR X$ = "r" THEN BYTE2% = 0:    ID$ = "R": GOTO 1080: REM Reset dome
630 IF X$ = "L" OR X$ = "l" THEN BYTE2% = &H8E: ID$ = "L": GOTO 1160: REM Flip
640 IF X$ = "H" OR X$ = "h" THEN 680: REM Build up a hex command to send
650 PRINT "'";X$;"'";
660 GOTO 420
670 '
680 REM Build up a hex command to send
690 PRINT
700 INPUT "Hex command (cr = no command)";HEXCMND$
710 IF HEXCMND$ = "" THEN 420
720 IF LEN(HEXCMND$) <> 2 THEN PRINT "Incorrect digit count": GOTO 700
730 HEXCMND = ASC(LEFT$(HEXCMND$,1)) - 48
740 IF HEXCMND > 9 THEN HEXCMND = HEXCMND - 7
750 IF HEXCMND > 33 THEN HEXCMND = HEXCMND - 32
760 HEXCMND = HEXCMND
770 H2 = ASC(RIGHT$(HEXCMND$,1)) - 48
780 IF H2 > 9 THEN H2 = H2 - 7
790 IF H2 > 33 THEN H2 = H2 - 32
800 HEXCMND = (HEXCMND * 16) + H2
810 ID$ = "X"
820 BYTE2% = HEXCMND
830 GOTO 1160
840 '
850 REM Process two byte commands
860 X% = ASC(MID$(KPAD$,2,1))
870 REM FOR I = 1 TO LEN(KPAD$)
880 REM PRINT ".";
890 REM PRINT HEX$(ASC(MID$(KPAD$,I,1)));
900 REM PRINT ". ";
910 REM NEXT I
920 REM PRINT: PRINT
930 '
940 IF X% = &H48 OR X% = &H50 THEN TILTSTOP = 1
950 IF X% = &H4B OR X% = &H4D THEN PANSTOP = 1
960 IF X% = &H48 THEN BYTE2% = &H84: ID$ = "U": GOTO 1160: REM Up
970 IF X% = &H50 THEN BYTE2% = &H85: ID$ = "D": GOTO 1160: REM Down
980 IF X% = &H4B THEN BYTE2% = &H81: ID$ = "L": GOTO 1160: REM Left
990 IF X% = &H4D THEN BYTE2% = &H82: ID$ = "R": GOTO 1160: REM Right
1000 REM IF X% = &H52 THEN BYTE2% = &H94: ID$ = "T": GOTO 800: REM DomeType
1010 IF X% = &H52 THEN BYTE2% = &HC9: ID$ = "-": GOTO 1160: REM Software rev
1020 IF X% = &H4F THEN BYTE2% = &H9A: ID$ = "F": GOTO 1160: REM Faster
1030 IF X% = &H47 THEN BYTE2% = &H8D: ID$ = "f": GOTO 1160: REM Fast
1040 IF X% = &H49 THEN BYTE2% = &H8E: ID$ = "S": GOTO 1160: REM Fastest
1050 PRINT "'";HEX$(BYTE2%);"'";
1060 GOTO 420
1070 '
1080 REM Reset the dome
1090 PRINT ID$
1100 BYTE2% = &H9A: GOSUB 1420: REM Faster
1110 BYTE2% = &H8B: GOSUB 1420: REM Zoom out
1120 BYTE2% = &H88: GOSUB 1420: REM Focus far
1130 BYTE2% = &H90: GOSUB 1420: REM Iris open
1140 GOTO 1200
1150 '
1160 REM Send the command
1170 PRINT ID$;
1180 GOSUB 1420: REM Send first command
1190 GOSUB 1590: REM wait until another key is hit
```

```
120 1200 GOTO 420
121 1210 '
122 1220 REM All done
123 1230 PRINT "All done"
124 1240 PRINT DATE$,TIME$
125 1250 END
126 1260 '
127 1270 REM Ask for a camera address
128 1280 REM Note that camera addresses are one based
129 1290 INPUT "What camera address should be used (cr = camera 6)";ADDRESS%
130 1300 IF ADDRESS% = 0 THEN ADDRESS% = 6
131 1310 IF ADDRESS% > 99 OR ADDRESS% < 1 THEN 1290
132 1320 PRINT "Using a camera address of:";ADDRESS%
133 1330 PRINT #3,"Using a camera address of: ";ADDRESS%
134 1340 PRINT #3,
135 1350 RETURN
136 1360 '
137 1370 REM Calculate a checksum
138 1380 CHECKSUM% = &H100 - (ADDRESS% + BYTE2%)
139 1390 CHECKSUM% = CHECKSUM% AND &HFF
140 1400 RETURN
141 1410 '
142 1420 REM Send a command
143 1430 REM
144 1440 REM BYTE2% is what changes and gets sent out
145 1450 REM
146 1460 '
147 1470 GOSUB 1370: REM Do checksum
148 1480 PRINT #3, USING "##,### &: ";COMMANDCOUNT+1;ID$;
149 1490 COMMANDCOUNT = COMMANDCOUNT + 1
150 1500 X% = ADDRESS%:  GOSUB 1810: ADDRESS$ = X$
151 1510 X% = BYTE2%:    GOSUB 1810: BYTE2$ = X$
152 1520 X% = CHECKSUM%: GOSUB 1810: CHECKSUM$ = X$
153 1530 PRINT #3, USING "0x& 0x& 0x& &";ADDRESS$;BYTE2$;CHECKSUM$;TIME$
154 1540 '
155 1550 SENDSTOP = TILTSTOP + PANSTOP + ZOOMSTOP + FOCUSSTOP + IRISSTOP
156 1560 PRINT #2,CHR$(ADDRESS%);CHR$(BYTE2%);CHR$(CHECKSUM%);
157 1570 RETURN
158 1580 '
159 1590 REM Wait for either the start of a second or until a second key is hit.
160 1600 IF SENDSTOP = 0 THEN 1790
161 1610 KPAD$ = INKEY$
162 1620 IF LEN(KPAD$) = 0 THEN 1600
163 1630 IF SENDSTOP  > 1 THEN BYTE2% = &H93: GOTO 1690
164 1640 IF ZOOMSTOP  > 0 THEN BYTE2% = &H8C: GOTO 1690
165 1650 IF IRISSTOP  > 0 THEN BYTE2% = &H92: GOTO 1690
166 1660 IF FOCUSSTOP > 0 THEN BYTE2% = &H89: GOTO 1690
167 1670 IF TILTSTOP  > 0 THEN BYTE2% = &H86: GOTO 1690
168 1680 IF PANSTOP   > 0 THEN BYTE2% = &H83
169 1690 GOSUB 1420: REM Send stop command when needed
170 1700 ZOOMSTOP = 0
171 1710 IRISSTOP = 0
172 1720 FOCUSSTOP = 0
173 1730 TILTSTOP = 0
174 1740 PANSTOP = 0
175 1750 SENDSTOP = 0
176 1760 '
177 1770 REM Y$ = MID$(TIME$,7,2)
178 1780 REM IF MID$(TIME$,7,2) = Y$ THEN 1280
179 1790 RETURN
```

```
180 1800 '
181 1810 REM Convert an integer into ascii-hex
182 1820 X$ = HEX$(X%)
183 1830 IF LEN(X$) <>2 THEN X$ = "0" + X$
184 1840 RETURN
185 1850 H1 = ASC(RIGHT$(HEXCMND$,1)) - 48
```

2 FLEX routines written to analyze Communications line dumps

`.CAP` ("capture") files are processed by using a small set of custom written routines and some general purpose routines. They are normally invoked by using the Y.BAT file:

```
copy  a:%1.cap
db /r   %1.cap | crit | nbl | mark1 > %1.doc
id1     %1.doc                        %1.id1
dir     %1.*
l       %1.id1
```

This file copies a file from the floppy and then does a complete automatic processing of it. When processing is complete the results are displayed with a varient of LIST (here named L).

2.1 CRIT.L

CRIT.L is a FLEX program that is used to put carriage returns into files processed by DB.

```
%{
// $Header: d:/sears/RCS/crit.l,v 1.3 2001-06-18 15:48:53-07 Hamilton Exp Hamilton $
// CRIT is used to add CR/LF following many commands
#include <stdio.h>
#include <stdlib.h>
#undef yywrap

void sendcr(void);

%}
CAMERAIDOUT " 0-"[0-7][0-9A-F]
CAMERAIDIN  " 1-"[0-6][0-9A-F]
DATAOUT     " 0-"[0-9A-F][0-9A-F]
CHECKSUMOUT " 0-"[0-9A-F][0-9A-F]
CHECKSUMIN  " 1-"[0-9A-F][0-9A-F]
%%
{CAMERAIDOUT}" 0-"[A6]{DATAOUT}{11}             sendcr();
{CAMERAIDOUT}" 0-C0 0-8"[1245]{DATAOUT}{2}      sendcr();
{CAMERAIDOUT}" 0-C3"{DATAOUT}{2}                sendcr();
{CAMERAIDOUT}" 0-"[89A-F][0-9A-F]{CHECKSUMOUT}  sendcr();

{CAMERAIDIN}" 1-C9"{CHECKSUMIN}{8}          sendcr();
{CAMERAIDIN}" 1-"[B][0-9A-F]{CHECKSUMIN}    sendcr();
{CAMERAIDIN}" 1-C1"{CHECKSUMIN}             sendcr();
{CAMERAIDIN}" 1-D"[1234]{CHECKSUMIN}        sendcr();
{CAMERAIDIN}" 1-E5"{CHECKSUMIN}             sendcr();
{CAMERAIDIN}" 1-F5"{CHECKSUMIN}             sendcr();
{CAMERAIDIN}" 1-F8"{CHECKSUMIN}             sendcr();
{CAMERAIDIN}                                ECHO;

.|\n     ECHO;
%%
// Used to stick out a CR/LF sequence
void sendcr(void)
{
        fprintf(yyout,"\n");
```

[3] `$Header: d:/sears/RCS/flex.inc,v 1.1 2001-12-11 15:13:39-08 Hamilton Exp Hamilton $`

```
 37         ECHO;
 38         fprintf(yyout,"\n");
 39         return;
 40 }
 41
 42 // yywrap is for End Of File processing, 1 = done
 43 int yywrap(void)
 44 {
 45         return 1;
 46 }
 47
 48 main(int argc, char *argv[])
 49 {
 50 char infilename[100];
 51 char outfilename[100];
 52
 53         fprintf(stderr,"$Id: crit.l,v 1.3 2001-06-18 15:48:53-07 Hamilton Exp Hamilton $\n");
 54
 55         // Do we have to deal with stdin/stdout?
 56         // 3 = one for the program name and one each for input and output names
 57         if (argc == 3) {
 58             // Nope
 59             // Get over the program, etc., name
 60             argc--;
 61             argv++;
 62             yyin = fopen(argv[0],"r");
 63             strcpy(infilename,argv[0]);
 64             if (yyin == NULL) {
 65                 fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
 66                                 argv[0]);
 67                 exit(EXIT_FAILURE);
 68             }
 69
 70             // Now get over the input file name and get to the output file name
 71             argc--;
 72             argv++;
 73             yyout = fopen(argv[0],"w");
 74             strcpy(outfilename,argv[0]);
 75             if (yyout == NULL) {
 76                 fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
 77                                 argv[0]);
 78                 exit(EXIT_FAILURE);
 79             }
 80         }
 81         else {
 82             // Woops we are using std... type io
 83             yyin  = stdin;
 84             yyout = stdout;
 85             strcpy(infilename,"stdin");
 86             strcpy(outfilename,"stdout");
 87         }
 88
 89         fprintf(yyout,"%%%12s, %4d: $Id: crit.l,v 1.3 2001-06-18 15:48:53-07 Hamilton Exp Hamilton
$\n",__FILE__,__LINE__);
 90         fprintf(yyout,"%%%12s, %4d: Reading from \"%s\" and writing into \"%s\"\n",
 91                         __FILE__,__LINE__,infilename,outfilename);
 92         yylex();
 93         exit(EXIT_SUCCESS);
 94 }
 95
```

```
 96 // $Log: crit.l,v $
 97 // Revision 1.3  2001-06-18 15:48:53-07  Hamilton
 98 // Normal end of day data saving
 99 //
100 // Revision 1.2  2001-06-14 15:09:49-07  Hamilton
101 // Normal end of day data saving
102 //
103 // Revision 1.1  2001-06-13 14:03:21-07  Hamilton
104 // Normal end of day data saving
105 //
106 // Revision 1.6  1999-07-09 07:54:17-07  Hamilton
107 // Deleted references to NOGETOPT, as I am not using it with the new vesion
108 // of FLEX.
109 //
110 // Revision 1.5  1997-01-12 21:12:50-08  ham
111 // Added in define for NO_GETOPT.  When set this will inhibit the inclusion
112 // of GNU's getopt routines, kinda.
113 //
114 // Revision 1.4  1996-11-14 19:36:28-0800  ham
115 // Add in end of file processing by adding the yywrap() function.
116 //
117
```

2.2 ID1.L

ID1.L is a FLEX program that further processes BREAKOUT .CAP file that have been processed by DB, CRIT, NBL and MARK1. The processing consists of verifying checksums, annotating command types and computing statisticts about the data.

```
%{
// $Header: d:/sears/RCS/id1.l,v 1.50 2002-02-14 11:51:13-08 Hamilton Exp Hamilton $
// ID1 is used to identify dome 01's messages.
//
// Special data line options are:
//
// % = Everything following this is to be printed and considered a comment
//     may be in any column.
// #polltime nn = nn is how long between each poll in 10th's of a second
//     may be in any column also may be in a comment (%) line.
//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#undef yywrap

#define MAXCAMS      256
#define MAXMESSAGELENGTH 1000000 // Maximum message length for an ASCII command
#define MESSAGETYPES 256

void Commonin(char message[]);
void Commonout(char message[]);
void CommonoutC0(char message[], int type);
int  Makenumb(void);
int  Makenumb5(void);
void Maketime(void);
void Printokbad(long cksum);
void Showdetails1(char message[], char char1, char char2);
void Showdetails2(char message[], char char1, char char2, char char3, char char4);

enum nextmessage {NOTHING, ALARM, MEMORYDUMP};

int  Linenumber = 1;
int  Polltime   = 10;  // Polltime is in 10's of a second
int  Totalin    = 1;
int  Totalout   = 1;

int  Ackincount;
int  Ackinidcount[MAXCAMS];
int  Ackoutcount;
int  Ackoutidcount[MAXCAMS];
int  Badchecksum;
int  Bc1[MAXCAMS];
int  Bc2[MAXCAMS];
int  Bc3[MAXCAMS];
int  Bc4[MAXCAMS];
int  Camid;
int  Camidsin[MAXCAMS];
int  Camidsout[MAXCAMS];
long Checksum;
char Commandout[MAXMESSAGELENGTH];
int  Goodchecksum;
char Messagein[MAXMESSAGELENGTH];
int  Nextresponse = NOTHING;
int  Outcount[MESSAGETYPES];
char Idmessage[500];
```

```
int  Incount[MESSAGETYPES];
int  Lastcmnd;
int  Lastcamid;
int  Lastresp;
int  Length;
int  Thisspeed;
int  Totaltime;         // All time starts at 00:00:00
int  Type;
int  Undefin;
int  Undefinmsg[MESSAGETYPES];
int  Undefout;
int  Undefoutcmnd[MESSAGETYPES];

%}
%x IDITOUT IDITIN FINDCR GETOUTCMND GETINCMND
%x COMMENT COMMENT1 FINDINCR FINDOUTCR FINDCRC0
%x DUMPLINE POLLTIME INCRTIMER
%%
^"%%"                    BEGIN DUMPLINE;
<DUMPLINE>.              ;
<DUMPLINE>\n             Linenumber++; BEGIN 0;

^"%"                     ECHO; BEGIN COMMENT;
<COMMENT>"#polltime"     ECHO; BEGIN POLLTIME;
<COMMENT>.               ECHO;
<COMMENT>\n              ECHO; Linenumber++; BEGIN 0;

<POLLTIME>[0-9]+        {ECHO;
                         Polltime = atoi(yytext);
                          fprintf(yyout,"\n%%%12s, %4d: Using a polltime of %d",
                                      __FILE__,__LINE__,Polltime);
                          fprintf(yyout,"\n%%%12s, %4d: Or increment the timer every %2.1f second",
                                      __FILE__,__LINE__,Polltime*.1);
                        }
<POLLTIME>.              ECHO;
<POLLTIME>\n             Linenumber++; ECHO, BEGIN 0;

"%"                      ECHO; BEGIN COMMENT1;
<COMMENT1>"#polltime"    ECHO; BEGIN POLLTIME;
<COMMENT1>.              ECHO;
<COMMENT1>\n             ECHO; BEGIN 0;


^" 0-"                   {Maketime();
                          ECHO;
                          BEGIN IDITOUT;
                         }
<IDITOUT>.{2}            {ECHO; Totalout++;
                          Camid = Checksum = Makenumb();
                          Camidsout[Makenumb()]++;
                          BEGIN GETOUTCMND;
                         }

^" 1-"                   {Maketime();
                          ECHO;
                          BEGIN IDITIN;
                         }
<IDITIN>.{2}             {ECHO; Totalin++;
                          Camid = Checksum = Makenumb();
                          Camidsin[Makenumb()]++;
```

```
118                          BEGIN GETINCMND;
119                         }
120
121 <GETOUTCMND>\n            {fprintf(yyout,"                    <%2d: %s",
122                                      Camid,"ACK.");
123                          Ackoutidcount[Camid]++;
124                          Printokbad(Checksum);
125                          Ackoutcount++; ECHO;
126                          BEGIN 0;
127                         }
128 <GETOUTCMND>" 0-80"      Commonout("Unknown 80"); BEGIN FINDCR;
129 <GETOUTCMND>" 0-81"      Commonout("Pan Left"); BEGIN FINDCR;
130 <GETOUTCMND>" 0-82"      Commonout("Pan Right"); BEGIN FINDCR;
131 <GETOUTCMND>" 0-83"      Commonout("Pan Stop"); BEGIN FINDCR;
132 <GETOUTCMND>" 0-84"      Commonout("Tilt Up"); BEGIN FINDCR;
133 <GETOUTCMND>" 0-85"      Commonout("Tilt Down"); BEGIN FINDCR;
134 <GETOUTCMND>" 0-86"      Commonout("Tilt Stop"); BEGIN FINDCR;
135 <GETOUTCMND>" 0-87"      Commonout("Focus Near"); BEGIN FINDCR;
136
137 <GETOUTCMND>" 0-88"      Commonout("Focus Far"); BEGIN FINDCR;
138 <GETOUTCMND>" 0-89"      Commonout("Focus Stop"); BEGIN FINDCR;
139 <GETOUTCMND>" 0-8A"      Commonout("Zoom In"); BEGIN FINDCR;
140 <GETOUTCMND>" 0-8B"      Commonout("Zoom Out"); BEGIN FINDCR;
141 <GETOUTCMND>" 0-8C"      Commonout("Zoom Stop"); BEGIN FINDCR;
142 <GETOUTCMND>" 0-8D"      Commonout("Fast (Rail left)"); BEGIN FINDCR;
143 <GETOUTCMND>" 0-8E"      Commonout("Fastest (Rail right)"); BEGIN FINDCR;
144 <GETOUTCMND>" 0-8F"      Commonout("Fast Stop (Rail button up)"); BEGIN FINDCR;
145
146 <GETOUTCMND>" 0-90"      Commonout("Iris Open"); BEGIN FINDCR;
147 <GETOUTCMND>" 0-91"      Commonout("Iris Close"); BEGIN FINDCR;
148 <GETOUTCMND>" 0-92"      Commonout("Iris Stop"); BEGIN FINDCR;
149 <GETOUTCMND>" 0-93"      Commonout("Stop All"); BEGIN FINDCR;
150 <GETOUTCMND>" 0-94"      {// Only increment timer if its a different poll command
151                          // Assume that the checksum will be OK
152                          if ((Camid != Lastcamid) || (Lastcmnd != 0x94)) {
153                              Totaltime += Polltime;
154                          }
155                          Commonout("Poll, Dome type query");
156                          Lastcamid = Camid;
157                          BEGIN FINDCR;
158                         }
159 <GETOUTCMND>" 0-95"      Commonout("Request Alarm Status"); Nextresponse = ALARM; BEGIN FINDCR;
160 <GETOUTCMND>" 0-96"      Commonout("Unknown 96"); BEGIN FINDCR;
161 <GETOUTCMND>" 0-97"      Commonout("ACK to dome"); BEGIN FINDCR;
162
163 <GETOUTCMND>" 0-98"      Commonout("Start Temp No Xmit"); BEGIN FINDCR;
164 <GETOUTCMND>" 0-99"      Commonout("End Temp No Xmit"); BEGIN FINDCR;
165 <GETOUTCMND>" 0-9A"      Commonout("Faster (\"Fast\")"); BEGIN FINDCR;
166 <GETOUTCMND>" 0-9B"      Commonout("Faster Stop (\"Fast\" button up)"); BEGIN FINDCR;
167 <GETOUTCMND>" 0-9C"      Commonout("Define Boundary"); BEGIN FINDCR;
168 <GETOUTCMND>" 0-9D"      Commonout("End Boundary"); BEGIN FINDCR;
169 <GETOUTCMND>" 0-9E"      Commonout("On Air"); BEGIN FINDCR;
170 <GETOUTCMND>" 0-9F"      Commonout("Reset On Air"); BEGIN FINDCR;
171
172 <GETOUTCMND>" 0-A0"      Commonout("Define Pattern 1"); BEGIN FINDCR;
173 <GETOUTCMND>" 0-A1"      Commonout("Define Pattern 2"); BEGIN FINDCR;
174 <GETOUTCMND>" 0-A2"      Commonout("Define Pattern 3"); BEGIN FINDCR;
175 <GETOUTCMND>" 0-A3"      Commonout("Accept New Pattern"); BEGIN FINDCR;
176 <GETOUTCMND>" 0-A4"      Commonout("Dump Dome Memory"); Nextresponse = MEMORYDUMP; BEGIN FINDCR;
177 <GETOUTCMND>" 0-A5"      Commonout("Request Dome Position"); BEGIN FINDCR;
```

```
178 <GETOUTCMND>" 0-A6"     Commonout("Go To Absolute Position"); BEGIN FINDCR;
179 <GETOUTCMND>" 0-A7"     Commonout("Acknowledgment to Dome"); BEGIN FINDCR;
180
181 <GETOUTCMND>" 0-A8"     Commonout("Mark Target 1"); BEGIN FINDCR;
182 <GETOUTCMND>" 0-A9"     Commonout("Mark Target 2"); BEGIN FINDCR;
183 <GETOUTCMND>" 0-AA"     Commonout("Mark Target 3"); BEGIN FINDCR;
184 <GETOUTCMND>" 0-AB"     Commonout("Mark Target 4"); BEGIN FINDCR;
185 <GETOUTCMND>" 0-AC"     Commonout("Go To Pattern 1 Start"); BEGIN FINDCR;
186 <GETOUTCMND>" 0-AD"     Commonout("Go To Pattern 2 Start"); BEGIN FINDCR;
187 <GETOUTCMND>" 0-AE"     Commonout("Go To Pattern 3 Start"); BEGIN FINDCR;
188 <GETOUTCMND>" 0-AF"     Commonout("Go To Pattern 4 Start"); BEGIN FINDCR;
189
190 <GETOUTCMND>" 0-B0"     Commonout("Run Pattern 1"); BEGIN FINDCR;
191 <GETOUTCMND>" 0-B1"     Commonout("Run Pattern 2"); BEGIN FINDCR;
192 <GETOUTCMND>" 0-B2"     Commonout("Run Pattern 3"); BEGIN FINDCR;
193 <GETOUTCMND>" 0-B3"     Commonout("Run New Pattern"); BEGIN FINDCR;
194 <GETOUTCMND>" 0-B4"     Commonout("Go To Target 1"); BEGIN FINDCR;
195 <GETOUTCMND>" 0-B5"     Commonout("Go To Target 2"); BEGIN FINDCR;
196 <GETOUTCMND>" 0-B6"     Commonout("Go To Target 3"); BEGIN FINDCR;
197 <GETOUTCMND>" 0-B7"     Commonout("Go To Target 4"); BEGIN FINDCR;
198 <GETOUTCMND>" 0-B8"     Commonout("Pattern End"); BEGIN FINDCR;
199
200 <GETOUTCMND>" 0-B9"     Commonout("Mark Target 5"); BEGIN FINDCR;
201 <GETOUTCMND>" 0-BA"     Commonout("Mark Target 6"); BEGIN FINDCR;
202 <GETOUTCMND>" 0-BB"     Commonout("Mark Target 7"); BEGIN FINDCR;
203 <GETOUTCMND>" 0-BC"     Commonout("Go To Target 5"); BEGIN FINDCR;
204 <GETOUTCMND>" 0-BD"     Commonout("Go To Target 6"); BEGIN FINDCR;
205 <GETOUTCMND>" 0-BE"     Commonout("Go To Target 7"); BEGIN FINDCR;
206
207 <GETOUTCMND>" 0-C3"     Commonout("Proportional Speed"); BEGIN FINDCR;
208 <GETOUTCMND>" 0-C6"     Commonout("Reset"); BEGIN FINDCR;
209
210 <GETOUTCMND>" 0-C9"     Commonout("Get software ID"); BEGIN FINDCR;
211
212 <GETOUTCMND>" 0-E0"     Commonout("Aux 1,2,3,4 off"); BEGIN FINDCR;
213 <GETOUTCMND>" 0-E1"     Commonout("Aux 1 on, 2,3,4 off"); BEGIN FINDCR;
214 <GETOUTCMND>" 0-E2"     Commonout("Aux 2 on, 1,3,4 off"); BEGIN FINDCR;
215 <GETOUTCMND>" 0-E3"     Commonout("Aux 1,2 on, 3,4 off"); BEGIN FINDCR;
216 <GETOUTCMND>" 0-E4"     Commonout("Aux 3 on, 1,2,4 off"); BEGIN FINDCR;
217 <GETOUTCMND>" 0-E5"     Commonout("Aux 1,3, on, 2,4 off"); BEGIN FINDCR;
218 <GETOUTCMND>" 0-E6"     Commonout("Aux 2,3, on, 1,4 off"); BEGIN FINDCR;
219 <GETOUTCMND>" 0-E7"     Commonout("Aux 1,2,3 on, 4 off"); BEGIN FINDCR;
220
221 <GETOUTCMND>" 0-E8"     Commonout("Aux 4 on, 1,2,3, off"); BEGIN FINDCR;
222 <GETOUTCMND>" 0-E9"     Commonout("Aux 1,4 on, 2,3 off"); BEGIN FINDCR;
223 <GETOUTCMND>" 0-EA"     Commonout("Aux 2,4 on, 1,3 off"); BEGIN FINDCR;
224 <GETOUTCMND>" 0-EB"     Commonout("Aux 1,2,4 on, 3 off"); BEGIN FINDCR;
225 <GETOUTCMND>" 0-EC"     Commonout("Aux 3,4 on, 1,2 off"); BEGIN FINDCR;
226 <GETOUTCMND>" 0-ED"     Commonout("Aux 1,3,4 on, 2 off"); BEGIN FINDCR;
227 <GETOUTCMND>" 0-EE"     Commonout("Aux 2,3,4 on, 1 off"); BEGIN FINDCR;
228 <GETOUTCMND>" 0-EF"     Commonout("Aux 1,2,3,4 on"); BEGIN FINDCR;
229
230 <GETOUTCMND>" 0-F0"     Commonout("Terminate Pattern"); BEGIN FINDCR;
231
232 <GETOUTCMND>.{5}        {Commonout("Undefined command");
233                          Undefout++;
234                          Undefoutcmnd[Makenumb5()]++;
235                          BEGIN FINDOUTCR;
236                         }
237 <FINDOUTCR>.{5}         {ECHO;
```

```
238                          Checksum += Lastcmnd = Makenumb5();
239                          Outcount[Lastcmnd]++;
240                          Length++;
241                         }
242 <FINDOUTCR>\n            {if (Length != 3) {
243                              fprintf(yyout,"           <%2d: %s %d Len = %d",
244                                           Camid,Idmessage,Undefout,Length);
245                          }
246                          else {
247                              fprintf(yyout,"           <%2d: %s %d",
248                                           Camid,Idmessage,Undefout);
249                          }
250                          Printokbad(Checksum);
251                          ECHO;
252                          BEGIN 0;
253                         }
254
255 <FINDCR>.{5}             {ECHO;
256                          strcat(Commandout,yytext);
257                          Checksum += Makenumb5();
258                          Length++;}
259 <FINDCR>\n               {switch(Length) {
260                              case 13: // Goto location
261                                 fprintf(yyout," <%2d: Goto Position",
262                                               Camid);
263                                 break;
264                              default: // All the rest
265                                 fprintf(yyout,"           <%2d: %s",
266                                               Camid,Idmessage);
267                                 break;
268                          }
269                          Printokbad(Checksum);
270                          switch (Length) {
271                             case 13:
272                                 Showdetails2("Pan Position
",Commandout[8],Commandout[9],Commandout[13],Commandout[14]);
273                                 Showdetails2("Tilt Position
",Commandout[18],Commandout[19],Commandout[23],Commandout[24]);
274                                 Showdetails2("Zoom Position
",Commandout[28],Commandout[29],Commandout[33],Commandout[34]);
275                                 Showdetails2("Digital Zoom
",Commandout[38],Commandout[39],Commandout[43],Commandout[44]);
276                                 Showdetails1("Iris Offset          ",Commandout[48],Commandout[49]);
277                                 Showdetails1("Electronic Zoom Limit",Commandout[53],Commandout[54]);
278                                 break;
279                             default:
280                                 break;
281                          }
282                          ECHO;
283                          BEGIN 0;
284                         }
285
286
287 <GETOUTCMND>" 0-C0 0-81" {CommonoutC0("Var Speed Pan Left ",0x81);
288                           BEGIN FINDCRC0;
289                          }
290 <GETOUTCMND>" 0-C0 0-82" {CommonoutC0("Var Speed Pan Right",0x82);
291                           BEGIN FINDCRC0;
292                          }
293 <GETOUTCMND>" 0-C0 0-84" {CommonoutC0("Var Speed Tilt Up  ",0x84);
```

```
294                          BEGIN FINDCRC0;
295                         }
296 <GETOUTCMND>" 0-C0 0-85" {CommonoutC0("Var Speed Tilt Down",0x85);
297                          BEGIN FINDCRC0;
298                         }
299
300 <FINDCRC0>.{5}          {ECHO;
301                          Checksum += Lastcmnd = Makenumb5();
302                          if (Thisspeed == 0)
303                                 Thisspeed = Lastcmnd;
304                          Outcount[Lastcmnd]++;
305                          Length++;
306                         }          //Commonout("");
307 <FINDCRC0>\n            {fprintf(yyout," <%2d: %s",
308                                       Camid,Idmessage);
309                          Printokbad(Checksum);
310                          ECHO;
311                          BEGIN 0;
312                         }
313
314 <GETINCMND>\n           {fprintf(yyout,"                      <%2d: %s>",
315                                      Camid,"ACK");
316                          Ackinidcount[Camid]++;
317                          Ackincount++; ECHO;
318                          BEGIN 0;
319                         }
320 <GETINCMND>" 1-B0"      {Commonin("Boundary Crossing 1");
321                          Bc1[Camid]++;
322                          BEGIN FINDINCR;
323                         }
324 <GETINCMND>" 1-B1"      {Commonin("Boundary Crossing 2");
325                          Bc2[Camid]++;
326                          BEGIN FINDINCR;
327                         }
328 <GETINCMND>" 1-B2"      {Commonin("Boundary Crossing 3");
329                          Bc3[Camid]++;
330                          BEGIN FINDINCR;
331                         }
332 <GETINCMND>" 1-B3"      {Commonin("Boundary Crossing 4");
333                          Bc4[Camid]++;
334                          BEGIN FINDINCR;
335                         }
336 <GETINCMND>" 1-B4"      Commonin("Boundary Confusion"); BEGIN FINDINCR;
337 <GETINCMND>" 1-B5"      Commonin("Pattern Done"); BEGIN FINDINCR;
338 <GETINCMND>" 1-C1"      Commonin("Dome Power Up Message"); BEGIN FINDINCR;
339 <GETINCMND>" 1-C4"      Commonin("Unknown Response C4"); BEGIN FINDINCR;
340 <GETINCMND>" 1-C9"      Commonin("Software ID Response"); BEGIN FINDINCR;
341 <GETINCMND>" 1-D0"      Commonin("Dome Alarm 1"); BEGIN FINDINCR;
342 <GETINCMND>" 1-D1"      Commonin("Dome Alarm 2"); BEGIN FINDINCR;
343 <GETINCMND>" 1-D2"      Commonin("Dome Alarm 3"); BEGIN FINDINCR;
344 <GETINCMND>" 1-D4"      Commonin("Dome Alarm 4"); BEGIN FINDINCR;
345 <GETINCMND>" 1-E5"      Commonin("Dome Query Response"); BEGIN FINDINCR;
346 <GETINCMND>" 1-E8"      Commonin("Dome Query Response"); BEGIN FINDINCR;
347 <GETINCMND>" 1-F5"      Commonin("Dome Query Response"); BEGIN FINDINCR;
348 <GETINCMND>" 1-F8"      Commonin("Dome Query Response"); BEGIN FINDINCR;
349
350 <GETINCMND>.{5}         {ECHO;
351                          strcat(Messagein,yytext);
352                          Undefin++;
353                          Length++;
```

```
354                          Type = Makenumb5();
355                          Undefinmsg[Type]++;
356                          Checksum += Type;
357                          switch (Nextresponse) {
358                             case ALARM:
359                                 strcpy(Idmessage,"Dome Alarm status");
360                                 break;
361                             case MEMORYDUMP:
362                                 strcpy(Idmessage,"Dome memory dump");
363                                 break;
364                             default:
365                                 strcpy(Idmessage,"Undefined message");
366                                 break;
367                          }
368                          Nextresponse = NOTHING;
369                          BEGIN FINDINCR;
370                         }
371 <FINDINCR>.{5}           {ECHO;
372                          strcat(Messagein,yytext);
373                          Checksum += Makenumb5();
374                          Length++;
375                         }
376 <FINDINCR>\n             {switch(Length) {
377                               case 12: // Current location
378                                   fprintf(yyout," <%2d: "
379                                                 "Current Dome Position",
380                                                  Camid);
381                                   Undefinmsg[Type]--;
382                                   Undefin--;
383                                   break;
384                               case 10: // Software rev
385                                   fprintf(yyout," <%2d: %s",
386                                                  Camid,Idmessage);
387                                   Undefinmsg[Type]--;
388                                   Undefin--;
389                                   break;
390                               default: // All the rest
391                                   fprintf(yyout,"           <%2d: %s",
392                                                  Camid,Idmessage);
393                                   break;
394                          }
395                          Printokbad(Checksum);
396                          switch(Length) {
397                              case 12:
398                                   Showdetails1("Iris xunning   ",Messagein[3],Messagein[4]);
399                                   Showdetails1("Zoom limit     ",Messagein[8],Messagein[9]);
400                                   Showdetails2("Tilt Position
",Messagein[13],Messagein[14],Messagein[18],Messagein[19]);
401                                   Showdetails2("Zoom Position
",Messagein[23],Messagein[24],Messagein[28],Messagein[29]);
402                                   Showdetails2("Electronic
Zoom",Messagein[33],Messagein[34],Messagein[38],Messagein[39]);
403                                   Showdetails2("Pan Position
",Messagein[43],Messagein[44],Messagein[48],Messagein[49]);
404                                   break;
405                              case 10:
406                              default:
407                                   break;
408                          }
409                          ECHO;
```

```
410                          BEGIN 0;
411                         }
412
413
414 .         ECHO;
415 \n        Linenumber++; ECHO;
416 %%
417
418 // Commonin is used to do things required for each message
419 //
420 void Commonin(char message[])
421 {
422         strcpy(Idmessage,message);
423         strcat(Messagein,yytext);
424 // fprintf(yyout,"%%%12s, %4d: Messagein = \"%s\"",__FILE__,__LINE__,Messagein);
425         ECHO;
426         Checksum += Lastresp = Makenumb5();
427         Incount[Lastresp]++;
428         Length++;
429         return;
430 }
431
432 // Commonout is used to do things required for each command
433 //
434 void Commonout(char message[])
435 {
436         strcpy(Idmessage,message);
437         strcat(Commandout,yytext);
438 // fprintf(yyout,"%%%12s, %4d: Commandout = \"%s\"",__FILE__,__LINE__,Commandout);
439         ECHO;
440         Checksum += Lastcmnd = Makenumb5();
441         Outcount[Lastcmnd]++;
442         Length++;
443         return;
444 }
445
446
447 // CommonoutC0 is used to do things required for each 0xC0 command
448 //
449 void CommonoutC0(char message[], int type)
450 {
451         strcpy(Idmessage,message);
452         strcat(Commandout,yytext);
453 // fprintf(yyout,"%%%12s, %4d: Commandout = \"%s\"",__FILE__,__LINE__,Commandout);
454         ECHO;
455         Checksum += Lastcmnd = Makenumb5();
456         Checksum += type;
457         Outcount[Lastcmnd]++;
458         Length += 2;
459         return;
460 }
461
462 // Showdetails1 is used to printout the contents of a long message.
463 //
464 void Showdetails1(char message[], char char1, char char2)
465 {
466         fprintf(yyout,"\n");
467 //        fprintf(yyout,"                                                  ");
468         fprintf(yyout,"                                  ");
469         fprintf(yyout,"%s = 0x%c%c",message,char1,char2);
```

```
470         return;
471 }
472
473
474 // Showdetails2 is used to printout the contents of a long message.
475 //
476 void Showdetails2(char message[], char char1, char char2, char char3, char char4)
477 {
478         fprintf(yyout,"\n");
479 //        fprintf(yyout,"                                                  ");
480         fprintf(yyout,"                                  ");
481         fprintf(yyout,"%s = 0x%c%c 0x%c%c",message,char1,char2,char3,char4);
482         return;
483 }
484
485
486 // Makenumb is used to convert an input ASCII hex number to binary.
487 //
488 // yytext must contain the two digit ASCII hex value for conversation.
489 //
490 int Makenumb(void)
491 {
492 char *pointer,string[4];
493 int number;
494
495     strcpy(string,yytext);
496     string[2] = 0;
497     number = strtol(string,&pointer,16);
498     return number;
499 }
500
501
502 // Makenumb5 is used to convert an input 5 char ASCII field with a
503 // hex number right justified in it to binary.
504 //
505 // yytext must contain the five digit ASCII hex value for conversation.
506 //
507 int Makenumb5(void)
508 {
509 char *pointer,string[14];
510 int number;
511
512     string[0] = yytext[3];
513     string[1] = yytext[4];
514     string[2] = 0;
515     number = strtol(string,&pointer,16);
516     return number;
517 }
518
519
520 // Maketime is used to convert total seconds into HH:MM:SS and to print
521 // it out.
522 void Maketime(void)
523 {
524 int hh;
525 int ss;
526 int mm;
527
528         Length = 1;                     // Start this out at one each time
529         ss = (Totaltime/10) % 60;
```

```
530         mm = (Totaltime/600) % 60;
531         hh = (Totaltime/36000);
532 //        fprintf(yyout,"%02d:%02d:%02d,",hh,mm,ss);
533         fprintf(yyout,"%02d:%02d,",mm,ss);
534 //        fprintf(yyout,"%4d:",Linenumber%1000);
535         fprintf(yyout,"%6d:",Linenumber%100000);
536         Linenumber++;
537         return;
538 }
539
540
541 void Printokbad(long cksum)
542 {
543     if (cksum % 0x100 == 0) {
544 //        fprintf(yyout,", cksum OK, length %d>",Length);
545         if (Length > 15) {
546             fprintf(yyout," msg length = %d",Length);
547         }
548         if (Thisspeed == 0) {
549             fprintf(yyout,">");
550         }
551         else {
552             fprintf(yyout," %2d/sec>",Thisspeed);
553         }
554         Goodchecksum++;
555     }
556     else {
557         fprintf(yyout,", cksum bad = 0x%04lX, msg length = %d>",
558                       cksum,Length);
559         Badchecksum++;
560     }
561 //  fprintf(yyout," = 0x%x",cksum);
562 //    Checksum = 0;
563     Commandout[0] = '\0';
564     Messagein[0] = '\0';
565     Thisspeed = 0;
566     return;
567 }
568
569
570 // yywrap is for End Of File processing, 1 = done
571 int yywrap(void)
572 {
573 int i, j;
574 int foundzero;
575 int hadbc;
576 int hadanybc = 0;
577 int intotal  = Ackincount;
578 int outtotal = Ackoutcount;
579
580         fprintf(yyout,"\f\n");
581         fprintf(yyout,"%%%12s, %4d: Commands are sent out from the controller\n",
582                       __FILE__,__LINE__);
583         fprintf(yyout,"%%%12s, %4d: Messages are received by the controller\n",
584                       __FILE__,__LINE__);
585         fprintf(yyout,"%%%12s, %4d: IDs are the first byte of a message/command\n\n",
586                       __FILE__,__LINE__);
587
588         fprintf(yyout,"%%%12s, %4d: There were %d messages with bad checksums\n",
589                       __FILE__,__LINE__,Badchecksum);
```

```
590         fprintf(yyout,"%%%12s, %4d: There were %d messages with good checksums\n\n",
591                         __FILE__,__LINE__,Goodchecksum);
592
593         fprintf(yyout,"         The following command codes were sent\n");
594
595         for (i = 0; i < MAXCAMS; i++) {
596              if (Outcount[i] > 999) {
597                 fprintf(yyout,"%%%12s, %4d: There were %d 0x%0X commands recognized\n",
598                                 __FILE__,__LINE__,Outcount[i],i);
599                 Outcount[i] = -1;
600              }
601         }
602
603         // Print out the transmitted command counts
604         if (Undefout) {
605             fprintf(yyout,"\n%%%12s, %4d: There were %d unrecognized commands\n\n",
606                             __FILE__,__LINE__,Undefout);
607
608             fprintf(yyout,"       The unrecognized commands were\n");
609             fprintf(yyout,"       ");
610             for (i = 0; i < 16; i++) {
611                     fprintf(yyout,"%4X",i);
612             }
613
614             fprintf(yyout,"\n");
615             for (i = 0; i < MAXCAMS; i += 16) {
616                 fprintf(yyout,"  %4X ",i);
617                 for (j = 0; j < 16; j++) {
618                     if (Undefoutcmnd[i+j] != 0) {
619                         fprintf(yyout,"%4d",Undefoutcmnd[i+j]);
620                     }
621                     else {
622                         fprintf(yyout,"   .");
623                     }
624                 }
625                 fprintf(yyout,"\n");
626             }
627             fprintf(yyout,"\n");
628         }
629         else {
630             fprintf(yyout,"%%%12s, %4d: All %d commands were recognized\n",
631                             __FILE__,__LINE__,Totalout);
632             fprintf(yyout,"\n");
633         }
634
635         fprintf(yyout,"\n       ");
636         for (i = 0; i < 16; i++) {
637                 fprintf(yyout,"%4X",i);
638         }
639         fprintf(yyout,"\n");
640         for (i = MAXCAMS/2; i < MAXCAMS; i += 16) {
641                 fprintf(yyout,"  %4X ",i);
642                 for (j = 0; j < 16; j++) {
643                         if (Outcount[i+j] != 0) {
644                                 fprintf(yyout,"%4d",Outcount[i+j]);
645                         }
646                         else {
647                                 fprintf(yyout,"   .");
648                         }
649                 }
```

```
650             fprintf(yyout,"\n");
651         }
652         fprintf(yyout,"\n");
653
654
655         for (i = 0; i < MAXCAMS; i++) {
656              if (Incount[i] > 999) {
657                 fprintf(yyout,"%%%12s, %4d: There were %d 0x%02X responses received\n",
658                                 __FILE__,__LINE__,Incount[i],i);
659                 Incount[i] = -1;
660              }
661         }
662
663         fprintf(yyout,"\n");
664
665         // Print out the received message counts
666         fprintf(yyout,"         The following message types were received\n");
667         fprintf(yyout,"       ");
668         for (i = 0; i < 16; i++) {
669                 fprintf(yyout,"%4X",i);
670         }
671
672         fprintf(yyout,"\n");
673         for (i = MAXCAMS/2; i < MAXCAMS; i += 16) {
674                 fprintf(yyout,"  %4X ",i);
675                 for (j = 0; j < 16; j++) {
676                         if (Incount[i+j] != 0) {
677                                 fprintf(yyout,"%4d",Incount[i+j]);
678                         }
679                         else {
680                                 fprintf(yyout,"   .");
681                         }
682                 }
683                 fprintf(yyout,"\n");
684         }
685
686         fprintf(yyout,"\n\f");
687         fprintf(yyout,"%%%12s, %4d: There were %d ACKs received\n",
688                         __FILE__,__LINE__,Ackincount);
689
690         // Print out the received ACK counts
691         fprintf(yyout,"\n");
692         for (foundzero = MAXCAMS-1; foundzero > 0; foundzero--)
693                 if (Ackinidcount[foundzero] != 0) break;
694
695         if (foundzero != 0) {
696             fprintf(yyout,"            The following devices sent ACKs\n");
697             fprintf(yyout,"            ");
698             for (i = 0; i < 10; i++) {
699                     fprintf(yyout," %4d",i);
700             }
701
702             fprintf(yyout,"\n");
703             for (i = 0; i < MAXCAMS; i += 10) {
704                     if (i < foundzero+1) {
705                         fprintf(yyout,"        %3d ",i);
706                         for (j = 0; j < 10; j++) {
707                             if (j+i < foundzero+1) {
708                                 if (Ackinidcount[i+j] != 0) {
709                                         fprintf(yyout,"%5d",Ackinidcount[i+j]);
```

```
710                             }
711                             else {
712                                     fprintf(yyout,"    .");
713                             }
714                         }
715                     }
716                 fprintf(yyout,"\n");
717                 }
718         }
719     }
720     fprintf(yyout,"\n");
721
722     if (Undefin) {
723         fprintf(yyout,"\n%%%12s, %4d: There were %d unrecognized messages\n\n",
724                         __FILE__,__LINE__,Undefin);
725         fprintf(yyout,"       The unrecognized messages were\n");
726         fprintf(yyout,"       ");
727         for (i = 0; i < 16; i++) {
728                 fprintf(yyout,"%4X",i);
729         }
730
731         fprintf(yyout,"\n");
732         for (i = 0; i < MAXCAMS; i += 16) {
733             fprintf(yyout,"  %4X ",i);
734             for (j = 0; j < 16; j++) {
735                 if (Undefinmsg[i+j] != 0) {
736                     fprintf(yyout,"%4d",Undefinmsg[i+j]);
737                 }
738                 else {
739                     fprintf(yyout,"   .");
740                 }
741             }
742             fprintf(yyout,"\n");
743         }
744         fprintf(yyout,"\n");
745     }
746     else {
747         fprintf(yyout,"%%%12s, %4d: All %d messages were recognized\n\n",
748                         __FILE__,__LINE__,Totalin);
749     }
750
751
752     fprintf(yyout,"\n");
753     fprintf(yyout,"            The following message codes came in\n");
754     fprintf(yyout,"       ");
755     for (i = 0; i < 16; i++) {
756             fprintf(yyout,"%4X",i);
757     }
758
759     fprintf(yyout,"\n");
760     for (i = MAXCAMS/2; i < MAXCAMS; i += 16) {
761             fprintf(yyout,"  %4X ",i);
762             for (j = 0; j < 16; j++) {
763                     if (Incount[i+j] != 0) {
764                             fprintf(yyout,"%4d",Incount[i+j]);
765                     }
766                     else {
767                             fprintf(yyout,"   .");
768                     }
769             }
```

```
770             fprintf(yyout,"\n");
771         }
772
773         fprintf(yyout,"\n\f");
774         // Print out camera IDs for output and input cameras
775         fprintf(yyout,"            Active input camera IDs\n");
776
777         for (i = 0; i < MAXCAMS; i++) {
778              if (Camidsin[i] > 9999) {
779                 fprintf(yyout,"%%%12s, %4d: There were %d responses received from camera %d\n",
780                                 __FILE__,__LINE__,Camidsin[i],i);
781                 Camidsin[i] = -1;
782              }
783         }
784
785         fprintf(yyout,"            ");
786         for (i = 0; i < 10; i++) {
787                 fprintf(yyout," %4d",i);
788         }
789
790         fprintf(yyout,"\n");
791         for (foundzero = MAXCAMS-1; foundzero > 0; foundzero--)
792                 if (Camidsin[foundzero] != 0) break;
793
794         for (i = 0; i < MAXCAMS; i += 10) {
795                 if (i < foundzero+1) {
796                     fprintf(yyout,"        %3d ",i);
797                     for (j = 0; j < 10; j++) {
798                         if (j+i < foundzero+1) {
799                             if (Camidsin[i+j] != 0) {
800                                     fprintf(yyout,"%5d",Camidsin[i+j]);
801                             }
802                             else {
803                                    fprintf(yyout,"    .");
804                             }
805                         }
806                     }
807                 fprintf(yyout,"\n");
808                 }
809         }
810
811         fprintf(yyout,"\n\n");
812         fprintf(yyout,"            Messages were sent to IDs\n");
813         for (i = 0; i < MAXCAMS; i++) {
814              if (Camidsout[i] > 9999) {
815                 fprintf(yyout,"%%%12s, %4d: There were %d messages sent to camera %d\n",
816                                 __FILE__,__LINE__,Camidsout[i],i);
817                 Camidsout[i] = -1;
818              }
819         }
820
821         fprintf(yyout,"\n             ");
822         for (i = 0; i < 10; i++) {
823                 fprintf(yyout," %4d",i);
824         }
825
826         fprintf(yyout,"\n");
827         for (foundzero = MAXCAMS-1; foundzero > 0; foundzero--)
828                 if (Camidsout[foundzero] != 0) break;
829
```

```
830         for (i = 0; i < MAXCAMS; i += 10) {
831                 if (i < foundzero+1) {
832                     fprintf(yyout,"        %3d ",i);
833                     for (j = 0; j < 10; j++) {
834                         if (j+i < foundzero+1) {
835                             if (Camidsout[i+j] != 0) {
836                                     fprintf(yyout,"%5d",Camidsout[i+j]);
837                             }
838                             else {
839                                     fprintf(yyout,"    .");
840                             }
841                         }
842                     }
843                 fprintf(yyout,"\n");
844                 }
845         }
846 
847         fprintf(yyout,"\n\n");
848         for (i = 0; i < MAXCAMS; i++) {
849             hadanybc += Bc1[i] + Bc2[i] + Bc3[i] + Bc4[i];
850         }
851         if (hadanybc != 0) {
852             fprintf(yyout,"\n\f");
853             fprintf(yyout,"%%%12s, %4d: There were a total of %d boundary crossings\n\n",
854                             __FILE__,__LINE__,hadanybc);
855             fprintf(yyout,"           Camera    Boundary crossings\n");
856             fprintf(yyout,"                          1    2    3    4\n");
857             for (i = 0; i < MAXCAMS; i++) {
858                 hadbc = Bc1[i] + Bc2[i] + Bc3[i] + Bc4[i];
859                 if (hadbc != 0) {
860                     fprintf(yyout,"            %3d    ",i);
861                     if (Bc1[i] != 0) fprintf(yyout,"%5d",Bc1[i]);
862                     else             fprintf(yyout,"    .");
863                     if (Bc2[i] != 0) fprintf(yyout,"%5d",Bc2[i]);
864                     else             fprintf(yyout,"    .");
865                     if (Bc3[i] != 0) fprintf(yyout,"%5d",Bc3[i]);
866                     else             fprintf(yyout,"    .");
867                     if (Bc4[i] != 0) fprintf(yyout,"%5d",Bc4[i]);
868                     else             fprintf(yyout,"    .");
869                     fprintf(yyout,"\n");
870                 }
871             }
872         }
873         else {
874             fprintf(yyout,"\n\n");
875             fprintf(yyout,"%%%12s, %4d: There were no Boundary Crossing messages received\n",
876                             __FILE__,__LINE__);
877         }
878 
879         fprintf(yyout,"\n\n");
880 
881         return 1;
882 }
883 
884 int main(int argc, char *argv[])
885 {
886 char infilename[100];
887 char outfilename[100];
888 
889         fprintf(stderr,"$Id: id1.l,v 1.50 2002-02-14 11:51:13-08 Hamilton Exp Hamilton $\n");
```

```
890
891         // Do we have to deal with stdin/stdout?
892         // 3 = one for the program name and one each for input and output names
893         if (argc == 3) {
894             // Nope
895             // Get over the program, etc., name
896             argc--;
897             argv++;
898             yyin = fopen(argv[0],"r");
899             strcpy(infilename,argv[0]);
900             if (yyin == NULL) {
901                 fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
902                                 argv[0]);
903                 exit(EXIT_FAILURE);
904             }
905
906             // Now get over the input file name and get to the output file name
907             argc--;
908             argv++;
909             yyout = fopen(argv[0],"w");
910             strcpy(outfilename,argv[0]);
911             if (yyout == NULL) {
912                 fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
913                                 argv[0]);
914                 exit(EXIT_FAILURE);
915             }
916         }
917         else {
918             // Woops we are using std... type io
919             yyin  = stdin;
920             yyout = stdout;
921             strcpy(infilename,"stdin");
922             strcpy(outfilename,"stdout");
923         }
924
925         fprintf(yyout,"%%%12s, %4d: $Id: id1.l,v 1.50 2002-02-14 11:51:13-08 Hamilton Exp Hamilton
$\n",__FILE__,__LINE__);
926         fprintf(yyout,"%%%12s, %4d: Reading from \"%s\" and\n",
927                         __FILE__,__LINE__,infilename);
928         fprintf(yyout,"%%%12s, %4d: writing into \"%s\"\n",
929                         __FILE__,__LINE__,outfilename);
930         Commandout[0] = '\0';
931         Messagein[0] = '\0';
932         yylex();
933         exit(EXIT_SUCCESS);
934 }
935
936 // $Log: id1.l,v $
937 // Revision 1.50  2002-02-14 11:51:13-08  Hamilton
938 // Normal end of day data saving
939 //
940 // Revision 1.45  2001-12-17 16:25:48-08  Hamilton
941 // Normal end of day data saving
942 //
943 // Revision 1.44  2001-12-11 15:30:14-08  Hamilton
944 // Normal end of day data saving
945 //
946 // Revision 1.43  2001-12-11 11:44:41-08  Hamilton
947 // Normal end of day data saving
948 //
```

```
949 // Revision 1.42  2001-10-24 09:34:05-07  Hamilton
950 // Normal end of day data saving
951 //
952 // Revision 1.41  2001-07-31 08:29:09-07  Hamilton
953 // Normal end of day data saving
954 //
955 // Revision 1.40  2001-07-30 15:56:39-07  Hamilton
956 // Normal end of day data saving
957 //
958 // Revision 1.39  2001-07-27 11:17:21-07  Hamilton
959 // Normal end of day data saving
960 //
961 // Revision 1.38  2001-07-27 08:42:59-07  Hamilton
962 // Normal end of day data saving
963 //
964 // Revision 1.37  2001-07-26 16:06:25-07  Hamilton
965 // Normal end of day data saving
966 //
967 // Revision 1.36  2001-07-26 15:11:32-07  Hamilton
968 // Normal end of day data saving
969 //
970 // Revision 1.35  2001-07-26 14:15:33-07  Hamilton
971 // Normal end of day data saving
972 //
973 // Revision 1.34  2001-07-25 14:50:52-07  Hamilton
974 // Normal end of day data saving
975 //
976 // Revision 1.33  2001-07-25 14:02:55-07  Hamilton
977 // Normal end of day data saving
978 //
979 // Revision 1.32  2001-07-18 13:54:42-07  Hamilton
980 // Normal end of day data saving
981 //
982 // Revision 1.31  2001-07-18 13:02:57-07  Hamilton
983 // Normal end of day data saving
984 //
985 // Revision 1.30  2001-07-03 15:59:55-07  Hamilton
986 // Normal end of day data saving
987 //
988 // Revision 1.29  2001-07-03 09:28:05-07  Hamilton
989 // Normal end of day data saving
990 //
991 // Revision 1.28  2001-06-27 11:40:03-07  Hamilton
992 // Normal end of day data saving
993 //
994 // Revision 1.27  2001-06-19 15:20:06-07  Hamilton
995 // Normal end of day data saving
996 //
997 // Revision 1.26  2001-06-19 11:27:39-07  Hamilton
998 // Normal end of day data saving
999 //
1000 // Revision 1.25  2001-06-18 13:34:48-07  Hamilton
1001 // Normal end of day data saving
1002 //
1003 // Revision 1.24  2001-06-18 12:36:38-07  Hamilton
1004 // Normal end of day data saving
1005 //
1006 // Revision 1.23  2001-06-18 12:29:02-07  Hamilton
1007 // Normal end of day data saving
1008 //
```

```
1009 // Revision 1.22  2001-06-18 09:28:21-07  Hamilton
1010 // Normal end of day data saving
1011 //
1012 // Revision 1.21  2001-06-14 15:09:50-07  Hamilton
1013 // Normal end of day data saving
1014 //
1015 // Revision 1.19  2001-06-13 08:04:13-07  Hamilton
1016 // Normal end of day data saving
1017 //
1018 // Revision 1.18  2001-06-12 16:18:07-07  Hamilton
1019 // Normal end of day data saving
1020 //
1021 // Revision 1.17  2001-06-08 15:46:07-07  Hamilton
1022 // Normal end of day data saving
1023 //
1024 // Revision 1.16  2001-06-08 13:33:25-07  Hamilton
1025 // Normal end of day data saving
1026 //
1027 // Revision 1.15  2001-06-07 14:29:16-07  Hamilton
1028 // Normal end of day data saving
1029 //
1030 // Revision 1.14  2001-06-06 13:18:41-07  Hamilton
1031 // Normal end of day data saving
1032 //
1033 // Revision 1.13  2001-06-04 14:51:34-07  Hamilton
1034 // Normal end of day data saving
1035 //
1036 // Revision 1.12  2001-06-01 08:46:21-07  Hamilton
1037 // Normal end of day data saving
1038 //
1039 // Revision 1.11  2001-05-31 14:00:00-07  Hamilton
1040 // Normal end of day data saving
1041 //
1042 // Revision 1.10  2001-05-22 16:14:44-07  Hamilton
1043 // Normal end of day data saving
1044 //
1045 // Revision 1.9  2001-05-10 15:31:58-07  Hamilton
1046 // Normal end of day data saving
1047 //
1048 // Revision 1.8  2001-05-10 15:14:00-07  Hamilton
1049 // Normal end of day data saving
1050 //
1051 // Revision 1.7  2001-03-30 16:00:09-08  Hamilton
1052 // Normal end of day data saving
1053 //
1054 // Revision 1.6  2001-03-30 12:13:54-08  Hamilton
1055 // Normal end of day data saving
1056 //
1057 // Revision 1.5  2001-03-30 09:37:20-08  Hamilton
1058 // Normal end of day data saving
1059 //
1060 // Revision 1.4  2001-03-30 09:11:50-08  Hamilton
1061 // Normal end of day data saving
1062 //
1063 // Revision 1.3  2001-03-30 08:58:46-08  Hamilton
1064 // Normal end of day data saving
1065 //
1066 // Revision 1.2  2001-03-29 16:15:20-08  Hamilton
1067 // Normal end of day data saving
1068 //
```

```
1069 // Revision 1.1  2001-03-29 13:44:49-08  Hamilton
1070 // Normal end of day data saving
1071 //
1072 // Revision 1.2  2001-03-29 12:58:27-08  Hamilton
1073 // Normal end of day data saving
1074 //
1075 // Revision 1.1  2001-03-29 09:23:14-08  Hamilton
1076 // Normal end of day data saving
1077 //
1078 // Revision 1.1  2001-03-27 08:00:17-08  Hamilton
1079 // Normal end of day data saving
1080 //
1081
```

2.3 LATEXIT.L

LATEXIT.L is a FLEX program that reads a PIC assembly and extracts almost useful information from the comments to make a maintence document.

```
%{
// $Header: d:/sears/RCS/latexit.l,v 1.15 2002-02-26 11:42:00-08 Hamilton Exp Hamilton $
// LATEXIT is used to process include files in PIC assembly format and
//         make somewhat useful LaTeX documents from them.
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
#define FIRST   0
#define SECOND  1
int first = FIRST;
int firstpass = 0;
int linenumber = 0;
void Idproc(void);
%}
DIGIT    [0-9]
HEXDIGIT [0x][0-0A-Fa-f]*
ID1      [a-z][a-zA-Z0-9_]*
ID2      [A-Z][a-zA-Z0-9_]*

%x OTHER LATEXMODE TITLE IDPROC HEXNUMBER
%s VERBMODE INVERBMODE TABLEMODE PARAGRAPHMODE
%x SPECIALMODE SPECIALHEXMODE SPECIALUSEMODE NOMOREINDEX
%%
"   page"               fprintf(yyout,"\\clearpage\n"); ECHO; BEGIN 0;

" title "|" subtitle "  BEGIN TITLE;
<TITLE>.                ;
<TITLE>\n               {linenumber++;
                         ECHO; BEGIN 0;
                        }

^";;latex"              BEGIN LATEXMODE;
<LATEXMODE>"_"          fprintf(yyout,"\\_");
<LATEXMODE>"#"          fprintf(yyout,"\\#");
<LATEXMODE>.            ECHO;
<LATEXMODE>\n           {linenumber++;
                         ECHO; BEGIN 0;
                        }

";;ps"                  {BEGIN PARAGRAPHMODE;
                         if (firstpass == 0) {
                             fprintf(yyout,"\\begin{note}\nNote that small ");
                             fprintf(yyout,"``boxed'' numbers, ");
                             fprintf(yyout,"i.e.{\\tiny\\fbox{123}},\n ");
                             fprintf(yyout,"refer to approximate source code ");
                             fprintf(yyout,"line numbers.\n \\end{note}\n");
                             firstpass = 1;
                         }
                         fprintf(yyout,"\\rightline{\\fbox{\\tiny{%d}}}\n",linenumber);
                        }
<PARAGRAPHMODE>";"      ;
<PARAGRAPHMODE>^"; "    ;
<PARAGRAPHMODE>\x22     {if (first == FIRST) {        // quote processing
                             fprintf(yyout,"``"); first = SECOND;
                          }
                          else {
                             fprintf(yyout,"''"); first = FIRST;
```

```
58                          }
59                         }
60 <PARAGRAPHMODE>"->"     fprintf(yyout,"$\\longrightarrow$");
61 <PARAGRAPHMODE>"<-"     fprintf(yyout,"$\\longleftarrow$");
62 <PARAGRAPHMODE>"-->"    fprintf(yyout,"$\\longrightarrow$");
63 <PARAGRAPHMODE>"<--"    fprintf(yyout,"$\\longleftarrow$");
64 <PARAGRAPHMODE>"--->"   fprintf(yyout,"$\\longrightarrow$");
65 <PARAGRAPHMODE>"<---"   fprintf(yyout,"$\\longleftarrow$");
66 <PARAGRAPHMODE>\xF8     fprintf(yyout,"\\degree ");
67 <PARAGRAPHMODE>"_"      fprintf(yyout,"\\_");
68 <PARAGRAPHMODE>"#"      fprintf(yyout,"\\#");
69 <PARAGRAPHMODE>"0x"     fprintf(yyout,"{\\tt 0x"); BEGIN HEXNUMBER;
70 <PARAGRAPHMODE>.        ECHO;
71 <PARAGRAPHMODE>\n      {linenumber++;
72                          ECHO;
73                         }
74 <PARAGRAPHMODE>";;pe"   BEGIN 0;
75
76 <HEXNUMBER>[0-9A-Fa-f]{2} fprintf(yyout,"%s}",yytext); BEGIN PARAGRAPHMODE;
77 <HEXNUMBER>.             BEGIN PARAGRAPHMODE;
78
79
80 ";;ts"                 {BEGIN TABLEMODE;
81                          //fprintf(yyout,"\\centerline{\\fbox{\\tiny{%d}}}\n",linenumber);
82                          fprintf(yyout,"\\begin{verbatim}");
83                         }
84 <TABLEMODE>"#define"    fprintf(yyout,"       ");
85 <TABLEMODE>";"          fprintf(yyout," ");
86 <TABLEMODE>.            ECHO;
87 <TABLEMODE>\n          {linenumber++;
88                          ECHO;
89                         }
90 <TABLEMODE>";;te"      {fprintf(yyout,"\\end{verbatim}\\bigskip\n");
91                          BEGIN 0;
92                         }
93
94 ^";;v"                  fprintf(yyout,"\\verb|"); BEGIN VERBMODE;
95 ^";v"                   fprintf(yyout,"\\verb|"); BEGIN VERBMODE;
96 <VERBMODE>" "           ;
97 <VERBMODE>.             ECHO; BEGIN INVERBMODE;
98 <INVERBMODE>.           ECHO;
99 <VERBMODE>\n           {fprintf(yyout,"|\n");
100                         linenumber++;
101                         BEGIN 0;
102                        }
103 <INVERBMODE>\n         {fprintf(yyout,"|\n");
104                         linenumber++;
105                         BEGIN 0;
106                        }
107
108 ";;specialstart"        {fprintf(yyout,"\\bigskip\n");
109                          fprintf(yyout,"\\rightline{\\fbox{\\tiny{%d}}}\n",linenumber);
110                          fprintf(yyout,"\\tablefirsthead{\\hline\n");
111                          fprintf(yyout,"Command & Use \\\\ \\hline\\hline}\n");
112                          fprintf(yyout,"\\tablehead{\\hline\\multicolumn{2}{|l|}\n");
113                          fprintf(yyout,"{\\small\\sl Continued from the previous page.} \\\\ \\hline\n");
114                          fprintf(yyout,"Command & Use \\\\ \\hline\\hline}\n");
115                          fprintf(yyout,"\\tabletail{\\hline\\multicolumn{2}{|r|}\n");
116                          fprintf(yyout,"{\\small\\sl Continued on the next page.}\\\\ \\hline}\n");
117                          fprintf(yyout,"\\tablelasttail{\\hline}\n");
```

```
118                          fprintf(yyout,"\\begin{center}\\begin{supertabular}{|c|l|}\n");
119                          BEGIN SPECIALMODE;
120                         }
121 <SPECIALMODE>"; 0x"     fprintf(yyout,"{\\tt 0x"); BEGIN SPECIALHEXMODE;
122 <SPECIALHEXMODE>.{2}    fprintf(yyout,"%s} & ",yytext); BEGIN SPECIALUSEMODE;
123 <SPECIALUSEMODE>\xF8    fprintf(yyout,"\\degree ");
124 <SPECIALUSEMODE>.       ECHO;
125 <SPECIALUSEMODE>\n     {fprintf(yyout," \\\\ \n");
126                          linenumber++;
127                          BEGIN SPECIALMODE;
128                         }
129 <SPECIALMODE>";;specialend" {fprintf(yyout,"\\end{supertabular}\\end{center}");
130                              fprintf(yyout,"\\bigskip\n"); BEGIN 0;
131                             }
132 <SPECIALMODE>.          ;
133 <SPECIALMODE>\n         linenumber++; ECHO;
134
135 .                       fprintf(yyout,"%%"); ECHO; BEGIN OTHER;
136 <OTHER>" "              ECHO;
137 <OTHER>","              ;
138 <OTHER>{ID1}|{DIGIT}    ;
139 <OTHER>{HEXDIGIT}       ;
140 <OTHER>"$"|"#"|"_"      fprintf(yyout,"\\"); ECHO;
141 <OTHER>";"              fprintf(yyout,"%%"); ECHO; BEGIN NOMOREINDEX;
142 <OTHER>{ID2}            fprintf(yyout,"\\index{%s}",yytext);
143 <OTHER>\n              {linenumber++;
144                          ECHO; BEGIN 0;
145                         }
146 <NOMOREINDEX>.          ECHO;
147 <NOMOREINDEX>\n        {linenumber++;
148                          ECHO; BEGIN 0;
149                         }
150
151 \n                      linenumber++; ECHO;
152
153 %%
154
155 // yywrap is for End Of File processing, 1 = done
156 int yywrap(void)
157 {
158         return 1;
159 }
160
161 main(int argc, char *argv[])
162 {
163 char infilename[100];
164 char outfilename[100];
165
166         fprintf(stderr,"$Id: latexit.l,v 1.15 2002-02-26 11:42:00-08 Hamilton Exp Hamilton $\n");
167
168         // Do we have to deal with stdin/stdout?
169         // 3 = one for the program name and one each for input and output names
170         if (argc == 3) {
171             // Nope
172             // Get over the program, etc., name
173             argc--;
174             argv++;
175             yyin = fopen(argv[0],"r");
176             strcpy(infilename,argv[0]);
177             if (yyin == NULL) {
```

```
178                 fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
179                                 argv[0]);
180                 exit(EXIT_FAILURE);
181             }
182 
183             // Now get over the input file name and get to the output file name
184             argc--;
185             argv++;
186             yyout = fopen(argv[0],"w");
187             strcpy(outfilename,argv[0]);
188             if (yyout == NULL) {
189                 fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
190                                 argv[0]);
191                 exit(EXIT_FAILURE);
192             }
193         }
194         else {
195             // Woops we are using std... type io
196             yyin  = stdin;
197             yyout = stdout;
198             strcpy(infilename,"stdin");
199             strcpy(outfilename,"stdout");
200         }
201 
202         fprintf(yyout,"\\documentstyle[scopepic,changeba,twoside,");
203         fprintf(yyout,"supertab,11pt]{pelco}\n");
204         fprintf(yyout,"\\pagestyle{pelcodatetime}\n");
205         fprintf(yyout,"\\makeindex\n");
206         fprintf(yyout,"\\begin{document}\n");
207         fprintf(yyout,"\\rcsdocrev{\\}); fprintf(yyout,"$Header: d:/sears/RCS/latexit.l,v 1.15 2002-02-26
11:42:00-08 Hamilton Exp Hamilton $}\n");
208         fprintf(yyout,"\\newcounter{numbers}\n");
209         fprintf(yyout,"\n");
210         fprintf(yyout,"\\def\\itt#1{{\\tt #1}\\index{#1}}\n");
211         fprintf(yyout,"\\def\\isc#1{{\\sc #1}\\index{#1}}\n");
212         fprintf(yyout,"\n");
213         fprintf(yyout,"%% showlablel controls the listing of lables\n");
214         fprintf(yyout,"\\newif\\ifshowlabel\n");
215         fprintf(yyout,"\\showlabeltrue\n");
216         fprintf(yyout,"%% implement printing labels iff showlabel is true\n");
217         fprintf(yyout,"\\def\\llabel#1{\\sloppy\\ifshowlabel{");
218         fprintf(yyout,"\\footnotesize$<L=$ #1$>$}\\fi\\label{#1}\\fussy}\n");
219         fprintf(yyout,"\\def\\llisting#1{\\sloppy\\ifshowlabel{");
220         fprintf(yyout,"\\footnotesize$<F=$ #1$>$}\\fi{\\footnotesize\\listing{#1}}\\fussy}\n");
221         fprintf(yyout,"\\def\\iindex#1{\\sloppy\\ifshowlabel{");
222         fprintf(yyout,"\\footnotesize$<I=$ #1$>$}\\fi\\index{#1}\\fussy}\n");
223         fprintf(yyout,"\n");
224         fprintf(yyout,"%% Reading from \"%s\" and",infilename);
225         fprintf(yyout," writing into \"%s\"\n\n",outfilename);
226         fprintf(yyout,"\\addtocounter{footnote}{1}\n");
227         fprintf(yyout,"\\footnotetext{Input file = ``\\itt{%s}''.}\n",infilename);
228         yylex();
229         fprintf(yyout,"\n\\insertindex\n");
230         fprintf(yyout,"\\end{document}\n");
231         exit(EXIT_SUCCESS);
232 }
233 
234 // $Log: latexit.l,v $
235 // Revision 1.15  2002-02-26 11:42:00-08  Hamilton
236 // Normal end of day data saving
```

```
237 //
238 // Revision 1.14  2002-02-26 11:37:11-08  Hamilton
239 // Normal end of day data saving
240 //
241 // Revision 1.13  2002-02-08 16:05:47-08  Hamilton
242 // Normal end of day data saving
243 //
244 // Revision 1.12  2002-01-29 14:51:37-08  Hamilton
245 // Normal end of day data saving
246 //
247 // Revision 1.11  2002-01-29 08:13:38-08  Hamilton
248 // Normal end of day data saving
249 //
250 // Revision 1.10  2002-01-22 16:03:43-08  Hamilton
251 // Normal end of day data saving
252 //
253 // Revision 1.9  2002-01-03 15:35:14-08  Hamilton
254 // Normal end of day data saving
255 //
256 // Revision 1.8  2002-01-03 14:12:48-08  Hamilton
257 // Normal end of day data saving
258 //
259 // Revision 1.7  2001-12-27 12:37:58-08  Hamilton
260 // Normal end of day data saving
261 //
262 // Revision 1.6  2001-12-03 09:31:32-08  Hamilton
263 // Normal end of day data saving
264 //
265 // Revision 1.5  2001-11-29 14:18:29-08  Hamilton
266 // Normal end of day data saving
267 //
268 // Revision 1.4  2001-11-29 12:42:39-08  Hamilton
269 // Normal end of day data saving
270 //
271 // Revision 1.3  2001-10-16 15:35:33-07  Hamilton
272 // Normal end of day data saving
273 //
274 // Revision 1.2  2001-10-16 14:27:57-07  Hamilton
275 // Normal end of day data saving
276 //
277 // Revision 1.1  2001-08-03 09:39:54-07  Hamilton
278 // Normal end of day data saving
279 //
280
```

2.4 MARK1.L

MARK1.L is a FLEX program that inserts a blank line in front of the first poll message to camera ID #1. This is done so that scanning through a long dump listing is somewhat easier. (Finding a blank line is really useful when the listing has over 20,000 lines in it.

```
%{
// $Header: d:/sears/RCS/mark1.l,v 1.2 2001-07-27 09:08:59-07 Hamilton Exp Hamilton $
// CRIT is used to stick a CR/LF preceeding a poll to ID 1
#include <stdio.h>
#include <stdlib.h>
#undef yywrap

#define ID1    1
#define NOTID1 2

int Lastwas = NOTID1;

%}
%x ID1PROC COMMENT
%%
^"%"               ECHO; BEGIN COMMENT;
<COMMENT>.         ECHO;
<COMMENT>\n        ECHO, BEGIN 0;

" 0-01 0-94 0-6B" {if (Lastwas != ID1) {
                       Lastwas = ID1;
                       fprintf(yyout,"\n");
                   }
                   ECHO;
                   BEGIN ID1PROC;
                  }
<ID1PROC>.        ECHO;
<ID1PROC>\n       ECHO; BEGIN 0;

.|\n              Lastwas = NOTID1; ECHO;
%%

// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
char infilename[100];
char outfilename[100];

        fprintf(stderr,"$Id: mark1.l,v 1.2 2001-07-27 09:08:59-07 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            strcpy(infilename,argv[0]);
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
```

```
 57                                argv[0]);
 58                exit(EXIT_FAILURE);
 59            }
 60
 61            // Now get over the input file name and get to the output file name
 62            argc--;
 63            argv++;
 64            yyout = fopen(argv[0],"w");
 65            strcpy(outfilename,argv[0]);
 66            if (yyout == NULL) {
 67                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
 68                                argv[0]);
 69                exit(EXIT_FAILURE);
 70            }
 71        }
 72        else {
 73            // Woops we are using std... type io
 74            yyin  = stdin;
 75            yyout = stdout;
 76            strcpy(infilename,"stdin");
 77            strcpy(outfilename,"stdout");
 78        }
 79
 80        fprintf(yyout,"%%%12s, %4d: $Id: mark1.l,v 1.2 2001-07-27 09:08:59-07 Hamilton Exp Hamilton
$\n",__FILE__,__LINE__);
 81        fprintf(yyout,"%%%12s, %4d: Reading from \"%s\" and writing into \"%s\"\n",
 82                        __FILE__,__LINE__,infilename,outfilename);
 83        yylex();
 84        exit(EXIT_SUCCESS);
 85 }
 86
 87 // $Log: mark1.l,v $
 88 // Revision 1.2  2001-07-27 09:08:59-07  Hamilton
 89 // Normal end of day data saving
 90 //
 91 // Revision 1.1  2001-07-27 08:49:23-07  Hamilton
 92 // Normal end of day data saving
 93 //
 94
```

3 Sensormatic protocol definitions

```
; "$Header: d:/sears/RCS/sdefs.inc,v 1.17 2002-02-27 15:58:31-08 Hamilton Exp Hamilton $"
; Definitions to use with Sensormatic's RS422 protocol
; Copyright by Pelco, 2001, 2002
        nolist
        ifdef listincludes
         list
        endif ; listincludes

; Commands to the dome
;
#define S_UNKNOWN80        0x80 ; Unknown three byte command
#define S_PAN_LEFT         0x81 ; Pan left (24/sec) until Pan Right or Pan
                                ; Stop
#define S_PAN_RIGHT        0x82 ; Pan right (24/sec) until Pan Left or Pan
                                ; Stop
#define S_PAN_STOP         0x83 ; Stop panning
#define S_TILT_UP          0x84 ; Tilt up until Tilt Down or Tilt Stop
#define S_TILT_DOWN        0x85 ; Tilt Down until Tilt Up or Tilt Stop
#define S_TILT_STOP        0x86 ; Stop tilting
#define S_FOCUS_NEAR       0x87 ; Focus near until Focus Far or Focus Stop
#define S_FOCUS_FAR        0x88 ; Focus far until Focus Near or Focus Stop
#define S_FOCUS_STOP       0x89 ; Stop Focus
#define S_ZOOM_IN          0x8A ; Zoom in until Zoom Out or Zoom Stop
#define S_ZOOM_OUT         0x8B ; Zoom out until Zoom In or Zoom Stop
#define S_ZOOM_STOP        0x8C ; Stop zoom
#define S_FAST             0x8D ; Increase pan and tilt speeds to 48/sec
                                ; until Fast Stop
#define S_FASTEST          0x8E ; Increase pan and tilt speeds to 96/sec
                                ; until Fast Stop
#define S_FAST_STOP        0x8F ; Stop fast/fastest speeds (back to normal
                                ; 24/sec)
#define S_IRIS_OPEN        0x90 ; Opens iris (manual iris mode/lightens
                                ; Iris Preference offset (auto iris mode)
                                ; until Iris Close or Iris Stop
#define S_IRIS_CLOSE       0x91 ; Closes iris (manual iris mode/darkens
                                ; Iris Preference offset (auto iris mode)
                                ; until Iris Open or Iris Stop
#define S_IRIS_STOP        0x92 ; Stop iris offset adjustment (also stops
                                ; V-Phase Adjust)
#define S_ALL_STOP         0x93 ; Stop all movement
#define S_GET_DOME_TYPE    0x94 ; Request dome type or poll
#define S_GET_ALARMS       0x95 ; Request status of alarm inputs
#define S_ACK              0x97 ; ACKnowledge sent to dome responds to
                                ; asynchronous commands.
#define S_UNKNOWN98        0x98 ; Start temp no transmit
#define S_UNKNOWN99        0x99 ; End temp no transmit
#define S_FASTER           0x9A ; Increase pan and tilt speeds to 72/sec
                                ; until Fast Stop
#define S_FASTER_STOP      0x9B ; Stop faster speeds (back to normal 24/sec)
#define S_DEFINE_BOUNDARY  0x9C ; Start boundary definition. This command
                                ; is followed by dome movement commands and
                                ; tour Mark Boundary commands.
#define S_MARK_BOUNDARY    0x9D ; Marks the current position as a boundary
#define S_ON_AIR           0x9E ; Set On Air status to tell the dome to
                                ; send the asynchronous boundary crossing
                                ; command
#define S_ON_AIR_RESET     0x9F ; Reset On Air status
#define S_DEFINE1          0xA0 ; Start defining Pattern 1
```

```
#define S_DEFINE2          0xA1 ; Start defining Pattern 2
#define S_DEFINE3          0xA2 ; Start defining Pattern 3
#define S_NEW_PATTERN      0xA3 ; Accept the new pattern as the current
                                ; pattern and delete the old pattern.
#define S_DUMP_DOME_MEMORY 0xA4 ; Dump dome memory
#define S_GET_POSITION     0xA5 ; Request Dome position Coordinates
#define S_GOTO_POSITION    0xA6 ; Goto absolute position (Multiple-byte format)
#define S_MARK1            0xA8 ; Mark the current position as Target 1
#define S_MARK2            0xA9 ; Mark the current position as Target 2
#define S_MARK3            0xAA ; Mark the current position as Target 3
#define S_MARK4            0xAB ; Mark the current position as Target 4
#define S_GOTO_PAT1        0xAC ; Go to the start of pattern 1
#define S_GOTO_PAT2        0xAD ; Go to the start of pattern 2
#define S_GOTO_PAT3        0xAE ; Go to the start of pattern 3
#define S_GOTO_PAT4        0xAF ; Go to the start of pattern 4
#define S_RUN1             0xB0 ; Run Pattern 1
#define S_RUN2             0xB1 ; Run Pattern 2
#define S_RUN3             0xB2 ; Run Pattern 3
#define S_RUN_NEW          0xB3 ; Run a newly defined pattern to review it
                                ; before accepting it to replace the
                                ; previous pattern.
#define S_GOTO1            0xB4 ; Go to preset position called Target 1
#define S_GOTO2            0xB5 ; Go to preset position called Target 2
#define S_GOTO3            0xB6 ; Go to preset position called Target 3
#define S_GOTO4            0xB7 ; Go to preset position called Target 4
#define S_PATTERN_END      0xB8 ; Tells the dome to stop recording
                                ; (defining) a pattern
#define S_MARK5            0xB9 ; Mark the current position as Target 5
#define S_MARK6            0xBA ; Mark the current position as Target 6
#define S_MARK7            0xBB ; Mark the current position as Target 7
#define S_GOTO5            0xBC ; Go to preset position called Target 5
#define S_GOTO6            0xBD ; Go to preset position called Target 6
#define S_GOTO7            0xBE ; Go to preset position called Target 7
#define S_VARIABLE_SPEED   0xC0 ; Variable speed control (New command.
                                ; Multiple-byte format)
#define S_UNKNOWN_C1       0xC1 ; Unknown
#define S_PROP_SPEED       0xC3 ; Proportional speed pan or tilt movement
                                ; commands (Multiple-byte format)
#define S_PROP_LEFT        0x81 ; Subcommand of 0xC3, pan left
#define S_PROP_RIGHT       0x82 ; Subcommand of 0xC3, pan right
#define S_PROP_UP          0x84 ; Subcommand of 0xC3, tilt up
#define S_PROP_DOWN        0x85 ; Subcommand of 0xC3, tilt down
#define S_UNKNOWN_C4       0xC4 ; Unknown three byte command
#define S_RESET            0xC6 ; Run default "Apple Peel" pattern for a
                                ; spiral view of everything (only supported
                                ; by SpeedDome Ultra IV and DeltaDome, or
                                ; later products)
#define S_SOFTWARE_VERSION 0xC9 ; Get software version number from dome
#define S_OUTPUT0          0xE0 ; Clears all active drivers
#define S_OUTPUT1          0xE1 ; Set output Driver #1
#define S_OUTPUT2          0xE2 ; Set output Driver #2
#define S_OUTPUT3          0xE4 ; Set output Driver #3
#define S_OUTPUT4          0xE8 ; Set output Driver #4
#define S_TERM_PAT         0xF0 ; Stop/terminate current pattern

        space 2
; Messages from the dome

#define S_DOME_TYPE_IS     0x94 ; Response to Request Dome Type
#define S_BOUNDARY0CROSSED  0xB0 ; Boundary crossing #1
```

```
119 #define S_BOUNDARY1CROSSED  0xB1 ; Boundary crossing #2
120 #define S_BOUNDARY2CROSSED  0xB2 ; Boundary crossing #3
121 #define S_BOUNDARY3CROSSED  0xB3 ; Boundary crossing #4
122 #define S_BOUNDARY_CONFUSION 0xB4; Boundary Confusion (sent by dome if
123                                   ; problem defining boundaries)
124 #define S_PATTERN_DONE     0xB5 ; Pattern Done (sent by dome when it
125                                   ; completes a pattern)
126 #define S_POWERED_UP       0xC1 ; Dome Powered Up (sent by dome to indicate
127                                   ; it has powered up and is on line)
128 #define S_DOME_ALARM0      0xD0 ; Bit 0, Switch 1 (0 = on, 1 = off)
129 #define S_DOME_ALARM1      0xD1 ; Bit 1, Switch 2 (0 = on, 1 = off)
130 #define S_DOME_ALARM2      0xD2 ; Bit 2, Switch 3 (0 = on, 1 = off)
131 #define S_DOME_ALARM3      0xD3 ; Bit 3, Switch 4 (0 = on, 1 = off)
132
133         list
134 ; End of SDEFS.INC
135
```

4 Using SIM58.BAS to control a camera/dome

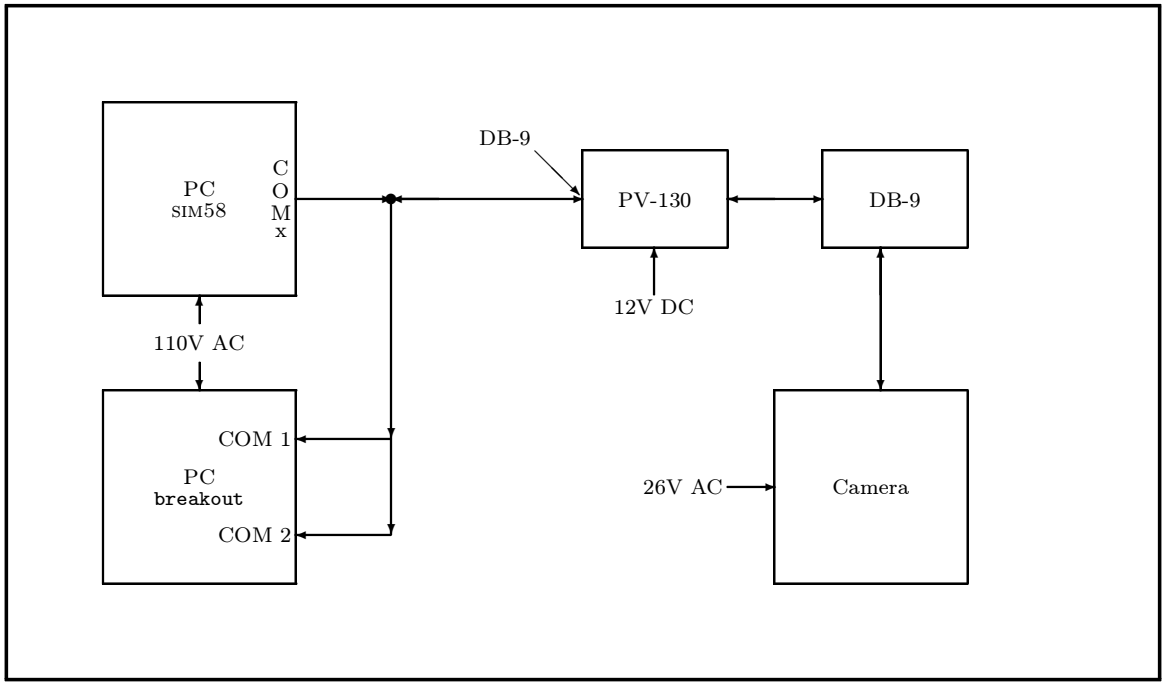
The SIM58 program has been developed to enable sending arbitrary sequences of commands to a dome. Some of the uses were:

1. Determine how a SpeedDome, or DeltaDome/UltraDome, responds to specific commands.
2. Verify that a Spectra/Esprit equipped with a TXB-S422 responds the same way.

SIM58 is written in GW-BASIC and runs under DOS on almost any PC type computer. The calling sequence is:

```
basic sim58
```

When SIM58 is running, several questions must be answered. Usually the most likely questions will only require a carriage return as the defaults have been carefully chosen so at to make my life easier.



$RCSfile: sim58.inc,v $

Figure 1. SIM58 simulation connections

[4] $Header: d:/sears/RCS/sim58.inc,v 1.3 2001-12-12 14:08:10-08 Hamilton Exp Hamilton $

Note

Following this note is a listing of SIM58. It is unlikely that it is accurate or up-to-date. This is intentional, since SIM58 is written in an interpreted language that may be easily changed at any time to meet the requirements "of the day". It is only included here as a guide as to what may be done with/to a dome.

Likewise since it is *expected* that the program will be constantly changing, no explanation of the various options or runtime commands are given.

```
 1 10 CLS: REV$ = "$Header: d:/sears/RCS/sim58.bas,v 1.9 2001-12-05 16:18:37-08 Hamilton Exp Hamilton $"
 2 20 PRINT REV$
 3 30 PRINT "Simulate parts of the operation of a Sensormatic, SensorVison"
 4 40 PRINT "RC58 matrix switch for testing a TXB-S422."
 5 50 PRINT "From the keypad, when NOT in 'num lock' mode."
 6 60 PRINT
 7 70 PRINT "Be sure that there are 4,5 and 6,8,20 jumpers in place"
 8 80 PRINT "or this program will time out when the serial communications is opened."
 9 90 PRINT "(With a 9 pin connector these are 7,8 and 6,1,4.)"
10 100 PRINT "(With a 25 pin connector these are 4,5 and 6,8,20.)"
11 110 PRINT
12 120 YES = 1: NO = 0
13 130 '
14 140 OPEN "status.log" FOR APPEND AS #3
15 150 PRINT #3,: PRINT #3,REV$
16 160 PRINT #3,DATE$,TIME$
17 170 '
18 180 INPUT "For this run should we have real outputs (cr = yes)";X$
19 190 IF X$ = "" THEN REAL = YES ELSE REAL = NO
20 200 IF REAL = NO THEN 300
21 210 PRINT "COM1 = DB-25, COM2 = DB-9"
22 220 INPUT "Use which COM port for IO (cr = COM1, anything else is COM2)";X$
23 230 IF X$ = "" THEN PORT$ = "COM1" ELSE PORT$ = "COM2"
24 240 OPENSTRING$ = PORT$ + ":4800,N,8,1"
25 250 OPEN OPENSTRING$ AS #2
26 260 PRINT #3,"The IO setup string was '";OPENSTRING$;"'"
27 270 PRINT #3,"This run writes data out to a dome"
28 280 GOTO 320
29 290 '
30 300 PRINT #3,"This is a simulation run that does not write data out"
31 310 '
32 320 INPUT "Should polling be used (cr = no)";X$
33 330 IF X$ = "" THEN POLLIT = NO ELSE POLLIT = YES
34 340 '
35 350 INPUT "Use what for a camera address (cr = 24)";ADDRESS
36 360 IF ADDRESS = 0 THEN ADDRESS = 24
37 370 PRINT "Using a camera address of ";ADDRESS
38 380 PRINT #3,"Writing to camera ";ADDRESS
39 390 '
40 400 INPUT "Send each command how many times (cr = 1)";RESENDCOUNT
41 410 IF RESENDCOUNT = 0 THEN RESENDCOUNT = 1
42 420 PRINT #3,"Resending each command ";RESENDCOUNT;" times"
43 430 '
44 440 FAST       = &H8D: FASTEST    = &H8E: FASTSTOP   = &H8F
45 450 FASTER     = &H9A: FASTERSTOP = &H9B
46 460 FOCUSNEAR  = &H87: FOCUSFAR   = &H88: FOCUSSTOP  = &H89
47 470 IRISOPEN   = &H90: IRISCLOSE  = &H91: IRISSTOP   = &H92
48 480 LEFT       = &H81: RIGHT      = &H82: PANSTOP    = &H83
49 490 UP         = &H84: DOWN       = &H85: TILTSTOP   = &H86
50 500 ZOOMIN     = &H8A: ZOOMOUT    = &H8B: ZOOMSTOP   = &H8C
51 510 ONAIR      = &H9E: OFFAIR     = &H9F
52 520 POLL       = &H94: STOPALL    = &H93
```

```
530 '
540 CLS: GOSUB 2810
550 TIMESTEPSPERMS = 0: GOSUB 3050
560 COMMANDTYPE$ = "On": COMMAND = ONAIR: GOSUB 1950: REM Turn it on
570 STARTPOLL = 1: LASTPOLL = 58: SOMECOMMAND = YES
580 BYTE1 = 99: BYTE2 = 99: BYTE3 = 99
590 PRINT #3,
600 '
610 REM All commands return here and look for more to do
620 REM Send a poll command about once a second
630 IF TIME$ <> LASTSECOND$ THEN GOSUB 2070
640 LASTSECOND$ = TIME$
650 IF TIME$ <> LASTSECOND$ THEN GOSUB 2070
660 KPAD$ = INKEY$
670 IF LEN(KPAD$) = 0 THEN 650
680 PRINT #3,: PRINT " ";
690 IF ASC(KPAD$) >= ASC("A") AND ASC(KPAD$) <= ASC("Z") THEN 700 ELSE 710
700 KPAD$ = STR$(ASC(KPAD$) - 32)
710 IF KPAD$ = " " THEN 1320: REM Stop the current movement
720 IF KPAD$ = "-" THEN ID1$ = "_": GOTO 2590: REM Focus Far
730 IF KPAD$ = "_" THEN ID1$ = "_": GOTO 2590: REM Focus Far
740 IF KPAD$ = "a" THEN 1710:                  REM fAster
750 IF KPAD$ = "c" THEN ID1$ = "C": GOTO 2500: REM Iris Close
760 IF KPAD$ = "d" THEN ID1$ = "D": GOTO 1400: REM Down
770 IF KPAD$ = "f" THEN 1760:                  REM Fast
780 IF KPAD$ = "h" OR KPAD$ = "?" THEN 2770:   REM Help
790 IF KPAD$ = "i" THEN 2240:                  REM Send tilt up + fast
800 IF KPAD$ = "j" THEN 2290:                  REM Send tilt down + fast
810 IF KPAD$ = "l" THEN ID1$ = "L": GOTO 1400: REM Left
820 IF KPAD$ = "n" THEN ID1$ = "N": GOTO 2590: REM Focus Near
830 IF KPAD$ = "o" THEN ID1$ = "O": GOTO 2500: REM Iris Open
840 IF KPAD$ = "p" THEN ID1$ = "P": GOTO 2340: REM Forced Poll
850 IF KPAD$ = "q" THEN ID1$ = "Q": GOTO 2460: REM Stop all
860 IF KPAD$ = "r" THEN ID1$ = "R": GOTO 1400: REM Right
870 IF KPAD$ = "s" THEN 1810:                  REM faStest
880 IF KPAD$ = "t" THEN ID1$ = "T": GOTO 2680: REM Zoom Tele
890 IF KPAD$ = "u" THEN ID1$ = "U": GOTO 1400: REM Up
900 IF KPAD$ = "w" THEN ID1$ = "W": GOTO 2680: REM Zoom Wide
910 IF KPAD$ = "x" THEN 1250:                  REM Exit
920 X = ASC(KPAD$) - 48
930 IF X < 0 OR X > 9 THEN 970
940 IF BYTE1 = 99 THEN BYTE1 = X: GOTO 610
950 IF BYTE2 = 99 THEN BYTE2 = X: GOTO 610
960 IF BYTE3 = 99 THEN BYTE3 = X: GOTO 1160
970 IF LEN(KPAD$) = 1 THEN INPUTKEY = ASC(KPAD$): GOTO 1120
980 IF LEN(KPAD$) <> 2 THEN 610
990 INPUTKEY = ASC(MID$(KPAD$,2,1))
1000 '
1010 IF INPUTKEY = &H47 THEN ID1$ = "L"   : GOTO 1400: REM Left
1020 IF INPUTKEY = &H4B THEN ID1$ = "Lf"  : GOTO 1500: REM Left F
1030 IF INPUTKEY = &H4F THEN ID1$ = "Lff" : GOTO 1620: REM Left FF
1040 IF INPUTKEY = &H52 THEN ID1$ = "Lfff": GOTO 1500: REM Left FFF
1050 IF INPUTKEY = &H49 THEN ID1$ = "R"   : GOTO 1400: REM Right
1060 IF INPUTKEY = &H4D THEN ID1$ = "Rf"  : GOTO 1500: REM Right F
1070 IF INPUTKEY = &H51 THEN ID1$ = "Rff" : GOTO 1620: REM Right FF
1080 IF INPUTKEY = &H53 THEN ID1$ = "Rfff": GOTO 1500: REM Right FFF
1090 IF INPUTKEY = &H48 THEN ID1$ = "U"   : GOTO 1400: REM Up
1100 IF INPUTKEY = &H50 THEN ID1$ = "D"   : GOTO 1400: REM Down
1110 '
1120 PRINT " '0x";HEX$(INPUTKEY);"', ";
```

```
113 1130 PRINT #3,TIME$," '0x";HEX$(INPUTKEY);"', "
114 1140 GOTO 610
115 1150 '
116 1160 REM Send an arbitrary command
117 1170 COMMAND = (BYTE1*100) + (BYTE2*10) + BYTE3
118 1180 BYTE1 = 99: BYTE2 = 99: BYTE3 = 99
119 1190 PRINT "'0x";HEX$(COMMAND);"'";
120 1200 REM PRINT #3,"Arbitrary command ";HEX$(COMMAND)
121 1210 COMMANDTYPE$ = "Ar"
122 1220 GOSUB 1950
123 1230 GOTO 610
124 1240 '
125 1250 REM All done
126 1260 PRINT #3,"All done"
127 1270 PRINT #3,DATE$,TIME$
128 1280 PRINT
129 1290 PRINT "Quitting"
130 1300 END
131 1310 '
132 1320 REM Stop the current movement
133 1330 IF STOP1 = NO THEN 1350
134 1340 COMMANDTYPE$ = STOPID1$: COMMAND = STOP1: GOSUB 1950
135 1350 IF STOP2 = NO THEN 1370
136 1360 COMMANDTYPE$ = STOPID2$: COMMAND = STOP2: GOSUB 1950
137 1370 STOP1 = NO: STOP2 = NO
138 1380 GOTO 610
139 1390 '
140 1400 REM Send a normal command
141 1410 PRINT " ";
142 1420 IF ID1$ = "U" THEN COMMAND1 = UP:    STOP1 = TILTSTOP: STOPID1$ = "St"
143 1430 IF ID1$ = "D" THEN COMMAND1 = DOWN:  STOP1 = TILTSTOP: STOPID1$ = "St"
144 1440 IF ID1$ = "R" THEN COMMAND1 = RIGHT: STOP1 = PANSTOP:  STOPID1$ = "Sp"
145 1450 IF ID1$ = "L" THEN COMMAND1 = LEFT:  STOP1 = PANSTOP:  STOPID1$ = "Sp"
146 1460 STOP2 = NO
147 1470 GOSUB 1860: REM Send one command
148 1480 GOTO 610
149 1490 '
150 1500 REM Send a pan and a fast or fastest command
151 1510 PRINT " ";
152 1520 IF ID1$ = "Rf"   THEN COMMAND1 = RIGHT: COMMAND2 = FAST:    ID2$ = "F"
153 1530 IF ID1$ = "Lf"   THEN COMMAND1 = LEFT:  COMMAND2 = FAST:    ID2$ = "F"
154 1540 IF ID1$ = "Rfff" THEN COMMAND1 = RIGHT: COMMAND2 = FASTEST: ID2$ = "FFF"
155 1550 IF ID1$ = "Lfff" THEN COMMAND1 = LEFT:  COMMAND2 = FASTEST: ID2$ = "FFF"
156 1560 STOP1 = FASTSTOP: STOPID1$ = "Sf"
157 1570 IF LEN(ID1$) = 4 THEN STOPID1$ = "Sfff"
158 1580 STOP2 = PANSTOP: STOPID2$ = "Sp"
159 1590 GOSUB 1900: REM Send two commands
160 1600 GOTO 610
161 1610 '
162 1620 REM Send a pan and a faster command
163 1630 PRINT " ";
164 1640 IF ID1$ = "Rff"  THEN COMMAND1 = RIGHT: COMMAND2 = FASTER:  ID2$ = "FF"
165 1650 IF ID1$ = "Lff"  THEN COMMAND1 = LEFT:  COMMAND2 = FASTER:  ID2$ = "FF"
166 1660 STOP1 = FASTERSTOP: STOPID1$ = "Sff"
167 1670 STOP2 = PANSTOP: STOPID2$ = "Sp"
168 1680 GOSUB 1900: REM Send two commands
169 1690 GOTO 610
170 1700 '
171 1710 REM fAster processing
172 1720 COMMANDTYPE$ = "FF": COMMAND = FASTER: GOSUB 1950
```

```
1730 STOP2 = FASTERSTOP: STOPID2$ = "Sff"
1740 GOTO 610
1750 '
1760 REM Fast processing
1770 COMMANDTYPE$ = "F": COMMAND = FAST: GOSUB 1950
1780 STOP2 = FASTSTOP: STOPID2$ = "Sf"
1790 GOTO 610
1800 '
1810 REM faStest processing
1820 COMMANDTYPE$ = "FFF": COMMAND = FASTEST: GOSUB 1950
1830 STOP2 = FASTSTOP: STOPID2$ = "Sfff"
1840 GOTO 610
1850 '
1860 REM Send a command
1870 COMMANDTYPE$ = ID1$: COMMAND = COMMAND1: GOSUB 1950
1880 RETURN
1890 '
1900 REM Send two commands
1910 COMMANDTYPE$ = ID1$: COMMAND = COMMAND1: GOSUB 1950
1920 COMMANDTYPE$ = ID2$: COMMAND = COMMAND2: GOSUB 1950
1930 RETURN
1940 '
1950 REM Send a command
1960 PRINT COMMANDTYPE$;: SOMECOMMAND = YES
1970 CHECKSUM = &H100 - (ADDRESS + COMMAND)
1980 PRINT #3,USING"& 0x\\,\  \";TIME$,HEX$(COMMAND),COMMANDTYPE$;
1990 PRINT #3,USING"### ### ###";ADDRESS,COMMAND,CHECKSUM
2000 IF REAL = NO THEN 2050
2010 FOR I = 1 TO RESENDCOUNT
2020 PRINT #2,CHR$(ADDRESS);CHR$(COMMAND);CHR$(CHECKSUM);
2030 GOSUB 3050
2040 NEXT I
2050 RETURN
2060 '
2070 REM Send a poll command about once a second
2080 LASTSECOND$ = TIME$
2090 THISPOLL = THISPOLL + 1
2100 IF THISPOLL > LASTPOLL THEN THISPOLL = STARTPOLL
2110 IF THISPOLL <> 1 THEN 2140
2120 PRINT ".";: IF SOMECOMMAND = YES THEN PRINT
2130 SOMECOMMAND = NO
2140 CHECKSUM = &H100 - (THISPOLL + POLL)
2150 IF POLLIT = NO OR REAL = NO THEN 2220
2160 PRINT #3,USING"& 0x\\,\  \";TIME$,HEX$(POLL),"Poll";
2170 PRINT #3,USING"### ### ###";THISPOLL,POLL,CHECKSUM
2180 FOR I = 1 TO RESENDCOUNT
2190 GOSUB 3050
2200 PRINT #2,CHR$(THISPOLL);CHR$(POLL);CHR$(CHECKSUM);
2210 NEXT I
2220 RETURN
2230 '
2240 REM Send an up + fast set of commands
2250 COMMANDTYPE$ = "U": COMMAND = UP: GOSUB 1950
2260 COMMANDTYPE$ = "F": COMMAND = FAST: GOSUB 1950
2270 GOTO 610
2280 '
2290 REM Send a down + fast set of commands
2300 COMMANDTYPE$ = "D": COMMAND = DOWN: GOSUB 1950
2310 COMMANDTYPE$ = "F": COMMAND = FAST: GOSUB 1950
2320 GOTO 610
```

```
2330 '
2340 REM Forced poll logic
2350 PRINT ",";
2360 CHECKSUM = &H100 - (ADDRESS + POLL)
2370 PRINT #3,USING"& 0x\\,\  \";TIME$,HEX$(POLL),"POLL";
2380 PRINT #3,USING"### ### ###";ADDRESS,POLL,CHECKSUM
2390 IF REAL = NO THEN 2440
2400 FOR I = 1 TO RESENDCOUNT
2410 PRINT #2,CHR$(ADDRESS);CHR$(POLL);CHR$(CHECKSUM);
2420 GOSUB 3050
2430 NEXT I
2440 GOTO 610
2450 '
2460 REM Stop all
2470 COMMANDTYPE$ = "Q": COMMAND = STOPALL: GOSUB 1950
2480 GOTO 610
2490 '
2500 REM Iris commands
2510 IF ID1$ = "O" THEN 2550
2520 COMMANDTYPE$ = "C": COMMAND = IRISCLOSE: GOSUB 1950
2530 GOTO 2560
2540 '
2550 COMMANDTYPE$ = "O": COMMAND = IRISOPEN: GOSUB 1950
2560 STOP2 = IRISSTOP: STOPID2$ = "Is"
2570 GOTO 610
2580 '
2590 REM Focus commands
2600 IF ID1$ = "_" THEN 2640
2610 COMMANDTYPE$ = "N": COMMAND = FOCUSNEAR: GOSUB 1950
2620 GOTO 2650
2630 '
2640 COMMANDTYPE$ = "_": COMMAND = FOCUSFAR: GOSUB 1950
2650 STOP2 = FOCUSSTOP: STOPID2$ = "fs"
2660 GOTO 610
2670 '
2680 REM Zoom commands
2690 IF ID1$ = "W" THEN 2730
2700 COMMANDTYPE$ = "T": COMMAND = ZOOMIN: GOSUB 1950
2710 GOTO 2740
2720 '
2730 COMMANDTYPE$ = "W": COMMAND = ZOOMOUT: GOSUB 1950
2740 STOP2 = ZOOMSTOP: STOPID2$ = "Zs"
2750 GOTO 610
2760 '
2770 REM Help Caller
2780 GOSUB 2810
2790 GOTO 610
2800 '
2810 REM Help data
2820 PRINT
2830 PRINT "Terminate all operations with a 'blank', exit with an 'x'"
2840 '
2850 PRINT "Active keypad keys are:"
2860 PRINT "+------------+------+---------------+"
2870 PRINT "|Left        | Up   | Right         |"
2880 REM     47            48     49
2890 PRINT "+------------+------+---------------+"
2900 PRINT "|Left Fast   |      | Right Fast    |"
2910 REM     4B           nothing 4D
2920 PRINT "+------------+------+---------------+"
```

```
293 2930 PRINT "|Left fAster | Down | Right fAster  |"
294 2940 REM     4F            50     51
295 2950 PRINT "+-------------+------+---------------+"
296 2960 PRINT "|Left faStest       | Right fasTest |"
297 2970 REM     52                  53
298 2980 PRINT "+-------------------+---------------+"
299 2990 '
300 3000 PRINT "Focus -,far; n,near -- Iris c,close; o,open -- Zoom t,tele; w,wide"
301 3010 PRINT "d,down; -- u,up; -- l,left; -- r,right -- p,poll"
302 3020 PRINT "Fast --- f,fast; -- a,faster; -- s,fastest; q = stop all motion"
303 3030 RETURN
304 3040 '
305 3050 REM Delay long enough for the command to go out
306 3060 FOR K = 1 TO 30
307 3070 GOSUB 3110
308 3080 NEXT K
309 3090 RETURN
310 3100 '
311 3110 REM Delay a short while
312 3120 IF TIMESTEPSPERMS = 0 THEN GOSUB 3220
313 3130 FOR J = 1 TO TIMESTEPSPERMS
314 3140 ' GOSUB 2830
315 3150 NEXT J
316 3160 RETURN
317 3170 '
318 3180 REM Timer thing
319 3190 REM X = x + 1: ' 37 / 17
320 3200 RETURN
321 3210 '
322 3220 REM Calibrate the timer so as to get steps/millieseconds
323 3230 PRINT "Calibrating the loop timer, takes 1 to 3 seconds"
324 3240 X$ = TIME$
325 3250 IF X$ = TIME$ GOTO 3240
326 3260 X$ = TIME$
327 3270 GOSUB 3180
328 3280 TIMESTEPSPERMS = TIMESTEPSPERMS + 1
329 3290 IF X$ = TIME$ GOTO 3270
330 3300 TIMESTEPSPERMS = TIMESTEPSPERMS / 100
331 3310 PRINT USING "Using ### steps to give a one ms delay";TIMESTEPSPERMS
332 3320 PRINT #3,USING "Using ### steps to give a one ms delay";TIMESTEPSPERMS
333 3330 RETURN
```

Index